Universidad de La Laguna

Escuela Superior de Ingeniería y Tecnología

Grado en Ingeniería Electrónica Industrial y Automática

— Trabajo de Fin de Grado —

Integración en PCB de la electrónica para el Scorbot-ER V+ y el Scorbot IX, y diseño del gestor de alarmas

Autor: Óscar Jesús Díaz de la Fe

Tutorizado por: Santiago Torres Álvarez

Julio, 2020

Índice

1. Agradecimientos **2**

2. Abstract **3**

3. Introducción **5**
3.1. Precedentes 5
3.2. Descripción general del proyecto 5
3.3. Objetivos 7
3.3.1. Objetivo general 7
3.3.2. Objetivos específicos 7
3.4. Planteamiento inicial 8

4. Cronograma **17**

5. Implementación en PCB de la electrónica de control **20**
5.1. Consideraciones iniciales 20
5.2. Consideraciones generales para el diseño y fabricación de la placa PCB 22
5.3. Evolución del diseño de la placa PCB 28
5.4. Diseño final de la placa PCB 35

6. Implementación del sistema a la electrónica el Scorbot-ER IX. **38**

7. Sistema de gestión de alarmas **47**
7.1. Parada de emergencia Z 47
7.2. Parada por sobretensión 50

8. Pruebas **54**

9. Dificultades encontradas **55**

10. Mejoras futuras **56**

11. Conclusiones **57**

12. Conclusions **58**

13. Presupuesto **59**

14. Bibliografía **60**

15. Anexos **62**

16. Datasheets **74**
16.1. Datasheet Scorbot-ER IX 74
16.2. Datasheet Scorbot-ER V 129
16.3. Darasheet LS7166 274

1. Agradecimientos

Para empezar me gustaría dedicar este apartado a agradecer a las personas que nos han ayudado tanto a mí y como a mi compañero a sacar adelante este Trabajo de Fin de Grado durante estos meses de trabajo.

A nuestro tutor D. Santiago Torres Álvarez por habernos guiado y haber resuelto nuestras dudas a lo largo del desarrollo del trabajo a través de las tutorías.

Al Servicio de Electrónica de la Universidad de La Laguna por prestarnos apoyo y ayudarnos en la elaboración de la placa PCB en todo momento.

A mi familia por apoyarme siempre en el desempeño de mi etapa académica.

A mi compañero de Trabajo de Fin de Grado, Cristian Francisco Fariña Melián, por apoyarme y ayudarme durante la realización del mismo.

2. Abstract

This project consists in the development of a Printed Circuit Board (PCB) that can perform the functions of the control electronics for the robot manipulator arms placed at the Robotics Laboratory of the Applied Sciences and Systems Department of the University of La Laguna.

Nowadays, in the laboratory of robotic, there is a design of the of the electronic of control of the robotic arm called 'Scorbot-ER V+'. This design is an alternative design of the electronic control circuit provided by the producer of the robotic arm. The design has been implemented and tested using some 'protoboards' and it is the result of a final degree project of the course of 2018-2019 called '*Implementation of the electronic for the dynamic control of a Scorbot-ER manipulator*'. The project was developed by two mates called Ana Estévez Pérez and Carlos Javier Siverio Suárez. This new design has the advantage of allowing a very fast communication with the robot because it does not use the serial communication.

Our final degree project (developed by Óscar Jesús Díaz de la Fe and Cristian Francisco Fariña Melián) consists of integrating this alternative design of the electronic of control for the robotic arm 'Scorbot-ER V+' in a printed circuit board to improve its features and gives the design a more compact, professional and aesthetic appearance. In fact, this implementation has the final objective of substitute the actual electronic of control of the 'Scorbot-ER V+' in a future. Also, the project consists of the testing of the design of the electronic and the software that allows the control of the movements of the manipulator and improve it in case it was possible.

Besides, another part of the project is the developed of an interface to join the new electronic of control to the robotic arm 'Scorbot-ER IX'. In the Robotics Laboratory of the Applied Sciences and Systems Department there are two types of manipulators. It would be interesting that the new electronic design could work with both kinds of robotic arms and not only with the 'Scorbot-ER V+'. Due to this, one part of our project is related to find a method to connect the new electronic of control to the 'Scorbot-ER IX', that has a similar structure and way of functioning than the other one.

On the other hand, the manipulator needs a new alarm system to work in the correct way without problems of developing elevated voltages and currents that could damage the manipulator itself. This is the reason why another part of our project is orientated to create an alarm system for the robotic arm. This alarm system is studied in two ways. First way is from a hardware point of view and is related to develop some electronic circuits to protect the manipulator in case of working in an incorrect way. Second way is from a software point of view and it has the objective of changing the current functional program of the manipulator, so this can take into account the possible failures that can occur during the work of the robotic arm.

Also, another part of the project is related to study and implement ation of some new control systems in the manipulator that improve the current ones. In

this moment, the only module of control that is implemented in the robotic arm is the PID (Proportional Integral Derivative). This is the most extended and common control system that can be implemented in a manipulator, but this is not the most efficient for this type of devices. In this project, it is proposed to implement a PID with gravity compensation control scheme. It is a better option to manage the control of a robotic because it is a specific control scheme for manipulators that takes into account the gravity effects on the robot dynamics, avoiding the PID to do this task of control

Finally, the last part of the work is about improving the current instruction set for the manipulator controller. This program is divided in two parts and is programmed in 'C' and 'Python' respectively. Some of the typical functions of a manipulator have been left out in the current program and other instructions are not programmed in the correct way, so one part of the project is oriented to complete and improve this programs.

3. Introducción

3.1. Precedentes

El presente Trabajo de Fin de Grado tiene como precedente los Trabajos de Fin de Grado '*Implementación de la electrónica para el control dinámico de un manipulador Scorbot-ER Vplus. Interfaz electrónica, lectura de codificadores digitales de posición y cálculos cinemáticos*' e '*Implementación de la electrónica para el control dinámico de un manipulador Scorbot-ER. Desarrollo de la comunicación*' realizados por Ana Estévez Pérez y Carlos Javier Siverio Suárez respectivamente.

Se pretende llevar a cabo la implementación del circuito de control para el manipulador 'Scorbot-ER V+' detallado en estos Trabajos de Fin de Grado en forma de Placa de Circuito Impreso (Printed Circuit Board o PCB), ya que actualmente se encuentra implementada dicha electrónica de control en 'protoboards' en el laboratorio de Robotica del Departamento de Ciencias y Sistemas Aplicados de la Universidad de La Laguna.

Además, se pretende mejorar esta electrónica de control en muchos aspectos. Por un lado, se pretende realizar una interfaz que permite conectar esta electrónica de control al manipulador 'Scorbot-ER IX' que se encuentra en el laboratorio de Robotica del Departamento de Ciencias y Sistemas Aplicados de la Universidad de La Laguna, para que ambos tipos de manipuladores funcionen con la misma electrónica de control, ya que tienen una estructura similar. Por otro lado, se pretende mejorar la gestión de alarmas del manipulador tanto vía software como implementando una nueva electrónica de gestión de alarmas. También se pretende implementar nuevas estructuras de control específicas para manipuladores como el PID con compensación de gravedad, debido a que es una mejor opción para administrar el control de un robot ya que es un esquema de control específico para manipuladores que tiene en cuenta los efectos de la gravedad en la dinámica del robot, evitando que el PID realice esta tarea de control.
Por último, se pretende ampliar y mejorar las instrucciones de movimiento del actual programa de control del manipulador. Este programa se divide en dos partes y está programado en 'C' y 'Python' respectivamente.

3.2. Descripción general del proyecto

En primer lugar, se debe destacar que este Trabajo de Fin de Grado se divide en dos partes diferenciadas, aunque ambas parte disponen de elementos comunes. Una de estas partes es el presente trabajo y la otra es el Trabajo de Fin de Grado de Óscar Jesús Díaz de la Fe con título: 'Integración en PCB de la electrónica del Scorbot-ER V+ y diseño del gestor de alarmas'.

Este proyecto consiste en el desarrollo de una placa de circuito impreso (Printed Circuit Board o PCB) que pueda realizar las funciones de electrónica de control para el brazo robótico 'Scorbot-ER V' que se encuentra situado en el laboratorio de Robotica del Departamento de Ciencias y Sistemas Aplicados de la Universidad de La Laguna.

Actualmente, en el laboratorio de robótica del Departamento de Ciencias y Sistemas Aplicados, existe un diseño de la electrónica de control del brazo robótico llamado 'Scorbot-ER V +'. Este es un diseño alternativo del circuito electrónico de control proporcionado por el fabricante del brazo robótico. El diseño se ha implementado y probado utilizando 'protoboards', ya que se trata simplemente de un prototipo, y es el resultado de un Trabajo de Fin de Grado del curso de 2018-2019 llamado '*Implementación de la electrónica para el control dinámico de un manipulador Scorbot-ER Vplus*'. Este proyecto fue desarrollado por dos compañeros llamados Ana Estévez Pérez y Carlos Javier Siverio Suárez, quienes tienen todo el mérito de haber realizado ese nuevo diseño, el cual tiene la ventaja de permitir una comunicación muy rápida con el robot ya que no hace uso de la comunicación serial.

Nuestro Trabajo de Fin de Grado (desarrollado por Óscar Jesús Díaz de la Fe y Cristian Francisco Fariña Melián) consiste en integrar este diseño alternativo de la electrónica de control para el brazo robótico 'Scorbot-ER V+' en una placa de circuito impreso para mejorar sus características y darle al diseño una apariencia más compacta, profesional y estética; ya que como actualmente se encuentra implementado mediante el uso de 'protoboards', no es práctico su utilización de forma eficaz. De hecho, esta implementación tiene el objetivo final de sustituir el control electrónico real del 'Scorbot-ER V+' en un futuro. Además, el proyecto también considera el hecho de probar el diseño de la electrónica y del software que permite el control de los movimientos del manipulador y de mejorarlo en caso de que fuera posible.

También, otra parte del proyecto consiste en el desarrollo de una interfaz para unir el nuevo sistema electrónico de control al brazo robótico 'Scorbot-ER IX'. En el laboratorio de robótica del Departamento de Ciencias y Sistemas Aplicados de la Universidad de La Laguna existen dos tipos de manipuladores, el 'Scorbot-ER V+' y el 'Scorbot-ER IX', y sería interesante que el nuevo diseño electrónico pudiera funcionar con ambos tipos de brazos robóticos y no solo con el 'Scorbot-ER V+'. Debido a esto, una parte de nuestro proyecto está relacionada con encontrar un método para conectar el nuevo sistema de control electrónico al 'Scorbot-ER IX', que tiene una estructura y una forma de funcionamiento similares a la del otro manipulador. Esta parte del proyecto será estudiada en el presente trabajo.

Por otro lado, el manipulador necesita un nuevo sistema de alarmas para funcionar de forma correcta sin tener problemas funcionamiento como por ejemplo desarrollar voltajes e intensidades elevados en los motores del manipulador que puedan dañar al mismo. Esta es la razón por la cual otra parte de nuestro proyecto está orientada a crear un sistema de alarmas para dicho manipulador. Este sistema de alarma se va a estudiar de dos formas. La primera forma es hardware y está relacionada con el desarrollo de circuitos electrónicos específicos para proteger al manipulador en caso de que funcione de manera incorrecta. La segunda forma es vía software y tiene como objetivo cambiar el programa actual que controla el funcionamiento del manipulador, de forma que se puedan tener en cuenta las posibles fallas que pueden ocurrir durante el trabajo del brazo robótico y evitar el mal funcionamiento del manipulador directamente desde el

software. Esta parte del proyecto también será estudiada en el presente trabajo.

Además, otra de las partes del proyecto está relacionada con el estudio y la implementación de nuevos sistemas de control en el manipulador que sean mejores que el actual, lo cual implica el estudio de la dinámica del propio brazo robótico. En este momento, el único módulo de control que se implementa en el manipulador es el PID (Derivativo Integral Proporcional). Este es el sistema de control más extendido y común que se puede implementar en un manipulador y en la mayoría de sistemas que requieren regulación, pero no es el más eficiente para este tipo de dispositivos. En este proyecto, se propone implementar un PID con compensación de gravedad para el sistema de control del manipulador, ya que es una mejor opción para administrar el control de un brazo robótico, debido a que este es un sistema de control específico para manipuladores. En esencia, un sistema PID con compensación de gravedad tiene en cuenta que el manipulador está bajo el efecto de la gravedad y libera el sistema PID de esta parte del trabajo, de forma que se puede alcanzar la consigna de una manera más eficiente incluso sin necesidad de añadir al controlador una acción integral. Esta parte del proyecto será estudiada por Cristian Francisco Fariña Melián.

Finalmente, la última parte del trabajo trata de mejorar el repertorio actual de instrucciones de movimiento del programa del controlador que se encuentra implementado para manipulador, ya que este no está completo. Este programa está dividido en dos partes y se encuentra programado en los lenguajes 'C' y 'Python' respectivamente.

Algunas de las funciones típicas de un manipulador se han omitido en el programa actual y otras instrucciones no están programadas de la manera más adecuada, por lo que una parte de nuestro proyecto se encuentra orientada a completar y mejorar este programa. Esta parte del proyecto también será estudiada por el alumno Cristian Francisco Fariña Melián.

3.3. Objetivos

3.3.1. Objetivo general

Como se ha dicho anteriormente, el objetivo principal de nuestro proyecto consiste en implementar la electrónica de control alternativa para el manipulador 'Scorbot-ER V+' en una placa de circuito impreso PCB, de forma que resulte más compacta y eficiente para en un futuro poder implementar este diseño de control en los manipuladores del laboratorio de Robótica del Departamento de Ingeniería Informática y de Sistemas.

Además, se intentará mejorar el diseño actual de la electrónica en la medida de lo posible, de forma que se aumente la eficacia del circuito electrónico.

3.3.2. Objetivos específicos

Los objetivos específicos de nuestro proyecto son los que han sido explicados anteriormente. De forma resumida son los siguientes:

- Establecer una interfaz que permita conectar la nueva electrónica de control con el brazo manipulador 'Scorbot-ER IX', de forma que ambos tipos

de manipuladores puedan funcionar con el mismo diseño de control.

- Integrar en el diseño del circuito de control un sistema de alarmas que permita proteger el diseño en caso de mal funcionamiento. Esta gestión de alarmas se intenta abordar de forma tanto hardware como software.

3.4. Planteamiento inicial

Para la elaboración de nuestro proyecto, era necesario el estudio en profundidad del circuito de electrónica de control en el que nos íbamos a basar para implementarlo en la placa PCB, así como comprender correctamente el funcionamiento del mismo mediante la lectura de la bibliografía correspondiente y en concreto el estudio de los Trabajos de Fin de Grado en los que se basa el nuestro.

Como se ha dicho anteriormente, el circuito el cual se pretende implementar en una placa PCB ya se encuentra construido en el laboratorio de robótica a modo de prototipo mediante el uso de 'protoboards', así que en primer lugar se decidió visitar este laboratorio para observar y comprender este circuito.

Este circuito prototipo había sido probado anteriormente y, por tanto, sabíamos que funcionaba correctamente por lo que se nos ocurrió usarlo como modelo para simplemente traducir su esquema a la placa PCB.

En las figuras 1 y 2 se observa el circuito que se encuentra en el laboratorio.

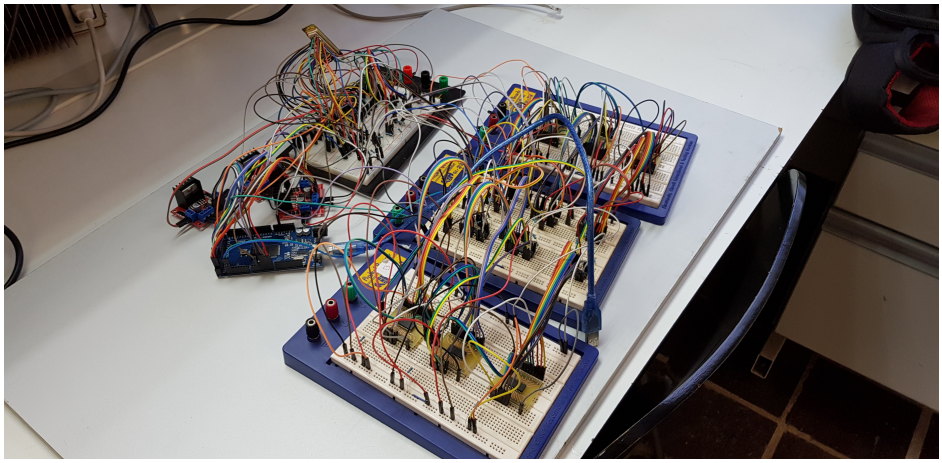


Figura 1: Foto del circuito prototipo. Vista 1

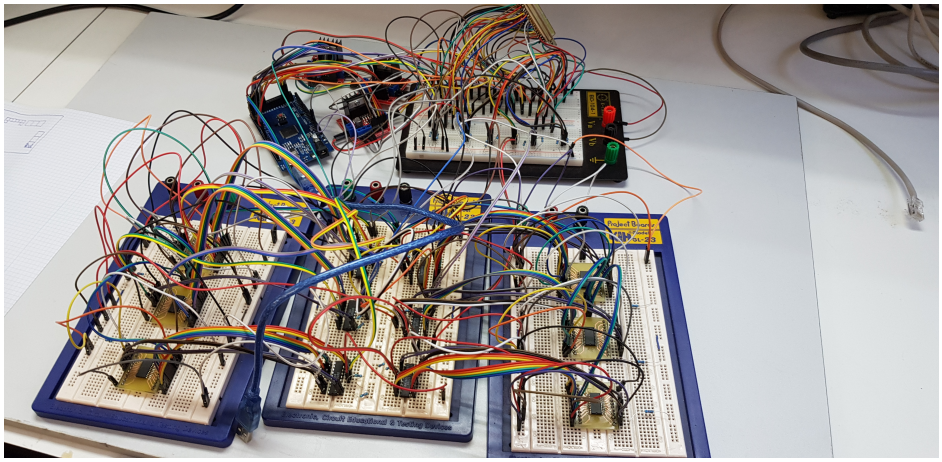


Figura 2: Foto del circuito. Vista 2

Como podemos ver, este circuito electrónico prototipo se encuentra implementado mediante cuatro 'protoboars', un arduino, tres módulos L298N y varios componentes eléctricos como circuitos integrados, resistencias y cables. Este modelo prototipo no resulta nada eficiente debido a su gran tamaño y a que es demasiado engorroso para trabajar con él. Por este motivo,el objetivo de nuestro trabajo es implementarlo en una placa de circuito impreso. A continuación, se irá explicando grosso modo la estructura del circuito prototipo para que sirva de contexto para su futura implementación en placa PCB.

En la figura 3, extraída del Trabajo de Fin de Grado de Ana Estévez Pérez: '*Implementación de la electrónica para el control dinámico de un manipulador Scorbot-ER Vplus. Interfaz electrónica, lectura de codificadores digitales de posición y cálculos cinemáticos*' se puede observar mejor la estructura esquemática del circuito: [1]

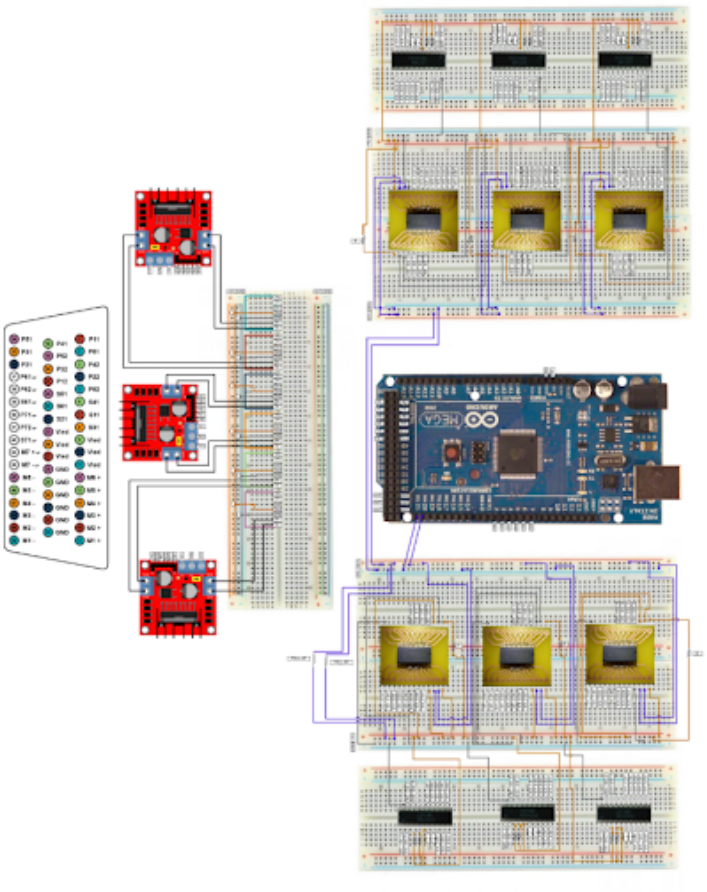


Figura 3: Esquema del prototipo del circuito de control

Este fue el esquema en el que nos basamos principalmente a la hora de realizar nuestra implantación del circuito en PCB, ya que en esta imagen se pueden ver todas las conexiónes entre los distintos componentes del circuito. No obstante, también se recurrió a inspeccionar el circuito físico directamente, ya que existen algunas discrepancias entre el circuito real que está fabricado y las instrucciones dadas en el esquema. Estas discrepancias serán explicadas más adelante en este documento.

En primer lugar, se debe explicar el funcionamiento del bloque Arduino en el circuito. Como se puede ver el circuito de control consta de un Arduino Mega 2560, el cual es un microcontrolador con las características de memoria y de número de pines necesarias para el correcto funcionamiento del circuito. En este caso, se plantea para el circuito el uso de este Arduino para realizar el control dinámico del brazo robótico mediante su programación. Este Arduino envía las señales de potencia a unos amplificadores que mueven los motores del manipulador y, además, mide las señales que provienen de los sensores de los que se dispone para realizar el control correctamente. Además, se hace uso de

unos contadores para que el Arduino pueda tomar de ellos la posición de las articulaciones del brazo cuando lo necesite y no cargar de esta forma demasiado al microcontrolador. De esta forma, el Arduino debe ir conectado mediante un cable USB a un ordenador para poder volcar desde él el código que permite su funcionamiento y nos permite interactuar con el mismo para que el manipulador obedezca las instrucciones que se le den.

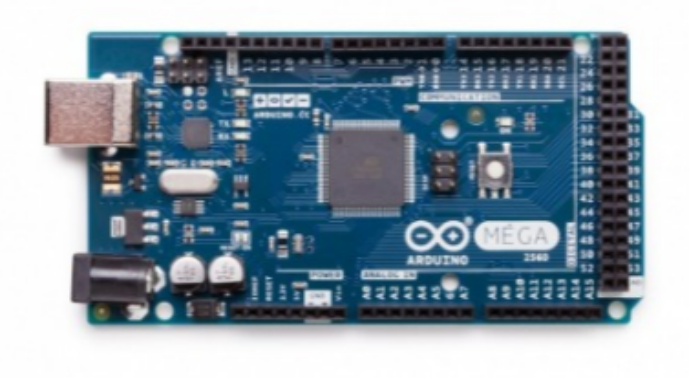


Figura 4: Arduino Mega 2560

Por otro lado, el circuito necesita dos tipos de alimentación para su funcionamiento, una alimentación de 5 voltios para el circuito de control y una de 24 voltios para la parte del circuito de potencia, ya que los motores del manipulador necesitan ese voltaje para funcionar. En el circuito prototipo se plantea el uso de dos fuentes de alimentación diferentes para solucionar este problema, pero en nuestro caso se planteó usar uno de los pines del Arduino para suministrar la alimentación de 5 voltios, ya que este dispositivo viene preparado para suministrar este voltaje a través de uno de sus pines mientras se encuentre conectado el Arduino mediante el cable USB. De esta manera, solamente es necesario el uso de una fuente de alimentación para alimentar la parte del circuito de potencia, al contrario del circuito prototipo. [1]

Especificaciones	Mega 2560 Rev3
Microcontrolador	ATmega2560
Voltaje Funcionamiento	5V
Voltaje Entrada (recomendado)	7-12V
Voltaje Entrada (límite)	6-20V
Pines I/O Digitales	54
Pines I/O PWM Digitales	15
Pines de Entrada Analógicos	16
Corriente DC por Pin I/O	20 mA
Corriente DC por Pin de 3,3V	50 mA
Memoria Flash	256 KB
SRAM	8 KB
EEPROM	4 KB
Velocidad Reloj	16 MHz
LEDs Incorporados	13
Largo	101,52 mm
Ancho	53,3 mm
Peso	37 g

Figura 5: Especificaciones del Arduino Mega 2560

En segundo lugar, podemos ver en el esquema que se hace uso de tres módulos L298N. Estos módulos son drivers que sirven para controlar y mover los motores del manipulador y funcionan a partir de una señal PWM (modulación por ancho de pulso) que genera el Arduino.

De forma general, este módulo consta, lógicamente, de un circuito integrado L298N formado por un doble puente en H. Además de esto, el módulo cuenta con un regulador de tensión de 5 voltios de salida, varios diodos de protección, un disipador de calor, 'jumpers' para gestionar las salidas del puente en H y el regulador, una entrada de alimentación y salidas para los motores. [1]

Las partes que componen el módulo L298N se puede ver en la figura 6. [3]

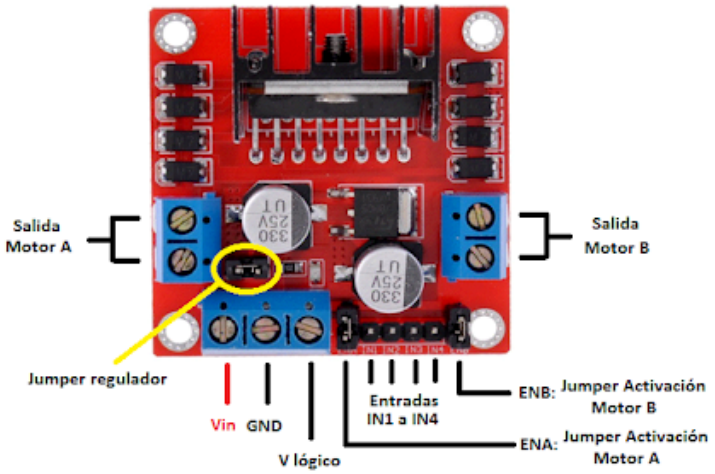


Figura 6: Esquema de partes del módulo L298N

Como se ha dicho anteriormente, los puentes en H de los que consta este módulo se utilizan para controlar los motores del manipulador, así que podría decirse que esta es la salida final de nuestra electrónica de control del brazo robótico; siendo esta parte la del circuito de potencia que precisa de la alimentación de 24 voltios. Además, en el circuito se hace uso de tres módulos, ya que cada uno puede controlar dos motores y son estos los que controlan tanto la dirección de giro com la velocidad de estos motores. Finalmente, este módulo ofrece en sí mismo una protección que hace la función de aislamiento entre esta parte de potencia y el circuito de control en sí, por lo que no se requiere de un aislamiento externo en el circuito.

Tanto el Arduino como estos módulos L298N son elementos físicos que no pueden ser soldados a una placa PCB, es decir, son dispositivos electrónicos independientes en sí mismos y no pueden ir unidos al diseño de nuestra placa,

salvo mediante el uso de cables de conexión entre la placa y estos dispositivos.

En tercer lugar, el circuito consta de varios circuitos integrados que se detallarán a continuación y que deben estar, lógicamente, incluidos en el diseño final de nuestra placa PCB para estar soldados a ella.

Por un lado, nos encontramos con los circuitos integrados LS7166. Estos circuitos son los contadores que reciben y cuentan los impulsos generados por los encoders del brazo manipulador. Existen en el diseño seis contadores, ya que son seis los motores del brazo robótico que se deben mover y deben funcionar en el modo de conteo de cuadratura porque los encoders tienen una salida de dos señales en cuadratura que permiten determinar el sentido de giro y la posición de los motores. El diseño de la electrónica de este circuito de control del brazo manipulador está preparado para trabajar tanto con los encoders del manipulador 'Scorbot-ER V+' como del 'Scorbot-ER IX' y, en concreto, se ha tenido en cuenta que este segundo manipulador tiene una resolución de cuentas inferior al del primer caso. El objetivo principal de este contador LS7166 es realizar la cuenta de los pulsos que emiten estos encoders para que de esta forma el controlador conozca la posición y el sentido de giro de los motores del manipulador. [1]

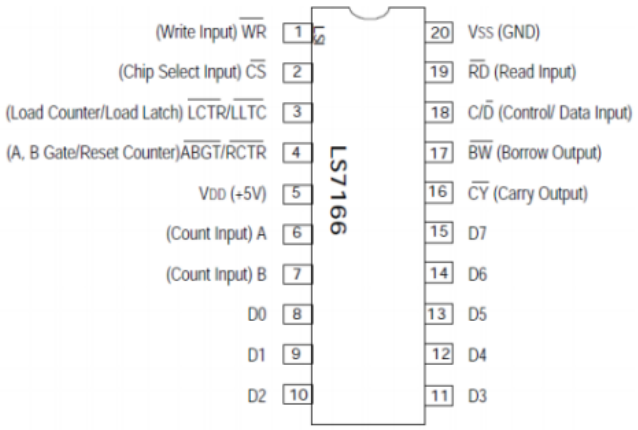


Figura 7: Circuito integrado LS7166

Por otro lado, podemos encontrar en el diseño varios circuitos integrados PCA9535. Estos circuitos son los que componen el bus I2C del que precisa el diseño de la electrónica de control para funcionar correctamente y el cual está compuesto por la líneas de comunicación SDA (System Data) que se utiliza para el envío de datos y la líneas de comunicación SCL (System Clock) que funciona como la señal de reloj que produce el arduino. Este bus establece el enlace entre el maestro y los esclavos de nuestro sistema a través de la sincronización producida por los pulsos de reloj enviados por SCL, siendo el maestro el Arduino y los esclavos los controladores I2C. De esta forma, el Arduino es el encargado de establecer y detener la comunicación con el dispositivo que se requiera en ese momento, siendo esta comunicación bidireccional entre dispositivos.

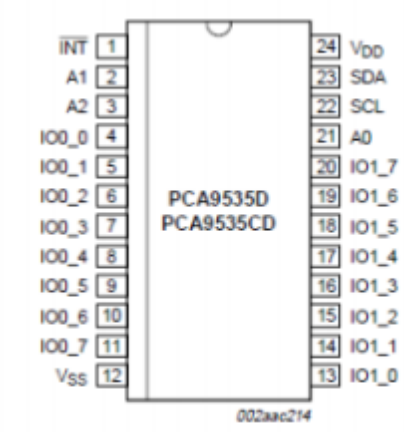


Figura 8: Circuito integrado PCA9535

Por otra parte, debemos considerar como un elemento del circuito de control también al conector D-50 del 'Scorbot-ER V+'; ya que, en definitiva, es esta la salida final de nuestro circuito de interés y, por tanto, debe ser tenido en cuenta como una parte fundamental del mismo. Este conector es el que establece la interfaz entre el diseño de la electrónica de control y el propio brazo manipulador. Este conector tiene en total 50 pines y se encarga de permitir el paso de varios tipos de señales: Las dos señales de los sensores de cada uno de los encoders del manipulador (en nuestro caso sólo se usan 6 motores, por lo que hay varios pines que no tienen ninguna función), las señales de fin de carrera de cada uno de los motores del manipulador, las alimentaciones de 5 voltios de cada uno de los encoders del manipulador, las tierras de cada uno de los encoders del manipulador y los polos positivos y negativos de las alimentaciones de 24 voltios de cada uno de los motores del manipulador

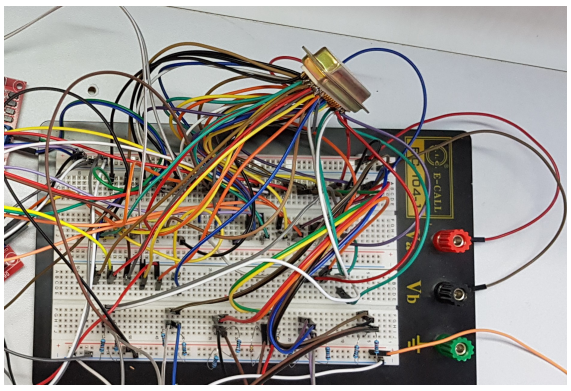


Figura 9: Conector D-50 en el circuito prototipo

Por último, el circuito de control, además de todos los elementos anteriormente descritos, también hace uso de varias resistencias pull-up. Estas resistencias son necesarias ya que los circuitos integrados digitales funcionan usando la lógica transistor a transistor (TTL) y se debe asegurar que los valores de tensiones (0 voltios para el 0 lógico y 5 voltios para el 1 lógico) son los adecuados para que estos circuitos digitales funcionen correctamente. El circuito consta de cuatro tipos de resistencias pull-up. En las líneas SDA y SCL son utilizadas resistencias de 2,2 KΩ conectadas a la alimentación. También se usan resistencias de 47 KΩ en varias entradas de los contadores para fijarlas al valor lógico de alta. Por otro lado, se usan resistencias de 10 KΩ en los integrados PCA9535, que están conectadas a la alimentación. Finalmente, se usan resistencias de 100 KΩ en las entradas de los contadores en las que se introducen las salidas de los encoders del manipulador. Todas estas resistencias serían lógicamente tenidas en cuenta para la elaboración de la placa PCB e irían soldadas a la misma.

En la siguiente figura, se pueden observar varios componentes del circuito prototipo que se intenta implementar en PCB. Concretamente, podemos ver el Arduino, los módulos L298N y algunas resistencias pull-up. Todo esto está unido mediante cables al conector D-50 del 'Scorbot-ER V+'. Por otro lado, se encontrarían las tres "protoboards" que contendrían los circuitos integrados que contienen los contadores y los buses I2C, pero estos elementos no se conectan (al menos directamente) al conector D-50; aunque sí que lo hacen a través de unas resistencias pull-up que se pueden ver en la figura 10, en la parte inferior de la protoboard.

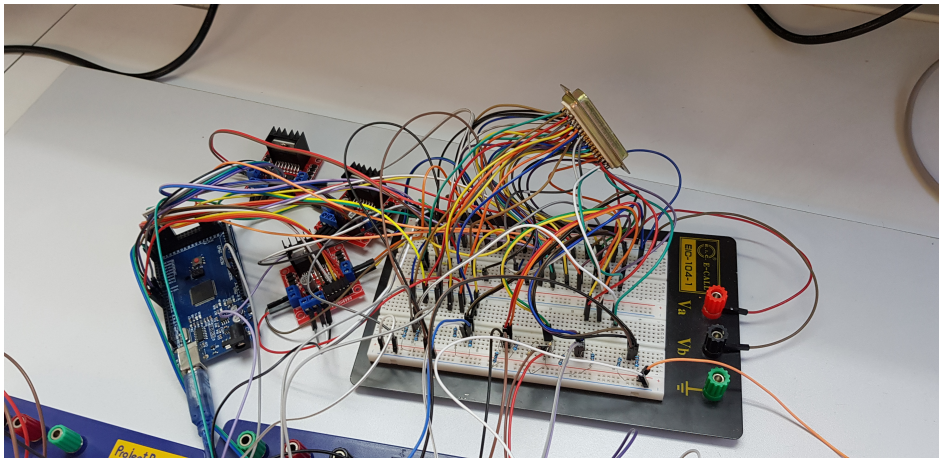


Figura 10: Foto de parte del circuito prototipo

Para más información acerca de la interfaz electrónica del circuito y sus componentes se puede visitar el Trabajo de Fin de Grado de Ana Estévez Pérez 'Implementación de la electrónica para el control dinámico de un manipulador Scorbot-ER Vplus. Interfaz electrónica, lectura de codificadores digitales de posición y cálculos cinemáticos' [1], ya que en ese documento se encuentra esta información de manera mucho más detallada en la sección dedicada precisamente a la explicación de la interfaz electrónica de este circuito de control alternativo para el 'Scorbot-ER V+'. En el presente documento sólo se ha hecho a modo de resumen una explicación del circuito de control que se va a implementar en PCB para dar un contexto a nuestro trabajo y no incluir en el mismo únicamente la explicación del diseño de la placa de circuito impreso. Es decir, antes de comenzar a detallar nuestra aportación a la electrónica de control a través de este proyecto, se ha considerado necesario realizar una explicación simple del circuito a implementar.

4. Cronograma

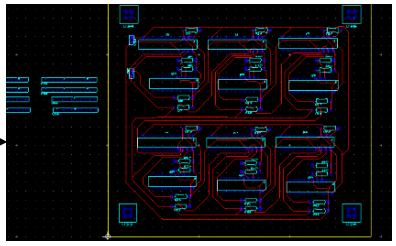
A continuación, en la siguiente página, se muestra el cronograma el proyecto. En este cronograma se realiza una descripción general de las diferentes tareas y actividades que se han llevado a cabo durante el desarrollo del trabajo, para así dar una idea de cómo han ido evolucionando las etapas del mismo a lo largo de los meses que ha durado el curso académico (y tener en cuenta también las circunstancias especiales que han habido durante éste, ya que muchas de las cosas que se plantearon hacer durante el desarrollo del trabajo no han podido hacerse).

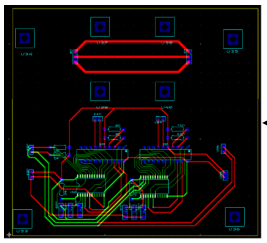
Noviembre 2019

PRIMEROS DISEÑO PCB

1 Diseños conceptuales de la electrónica del Scorbot V+ en el programa Multisim 11.0

Estos diseños se basan en el proyecto "Implementación de la electrónica para el control dinámico de un manipulador Scorbot-ER"





Diciembre 2019

PRIMERA VERSIÓN DE LA PLACA PCB

2 Finalización del primer diseño de la placa PCB.

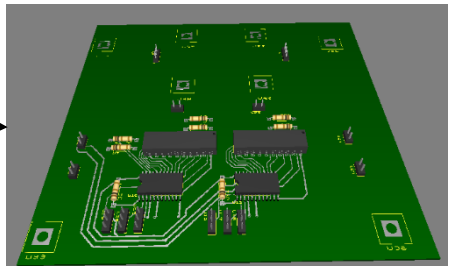
Modificado posteriormente para obtener un mejor diseño y solventar fallos

Enero 2020

FINALIZACIÓN DEL DISEÑO DEFINITIVO

3 Se terminó el diseño definitivo de la placa PCB, arreglando todos los fallos de la versión anterior.

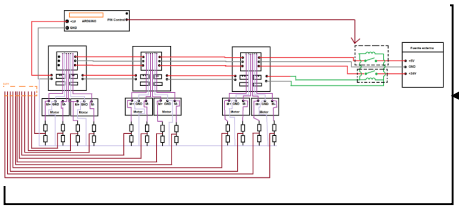
Planificación del diseño de sistemas de alarma



Febrero 2020

PRIMEROS DISEÑOS DE SISTEMAS DE ALARMA

4 Se preparan dos tipos de alarmas, una que se genere por acción del usuario y provoque una parada total del programa, y un sistema que detecte sobtetensión en los motores y los detenga



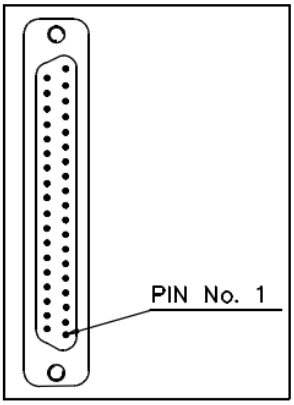
Marzo 2020

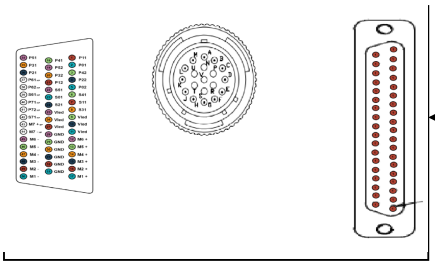
MODIFICACIONES DE LOS SISTEMAS DE ALARMA Y ANALIZIS DE LAS CONEXIONES DEL SCORBOT IX

5 Se van realizando cambios en las alarmas en base a las dificultades encontradas, buscando la solución más eficiente

Se busca información acerca del Scorbot IX para poder hacer la conexión entre este y el Scorbot V

Se decide replantear el proyecto inicial de ejecución práctica a uno de diseño teórico, debido a las limitaciones impuestas por la alerta sanitaria del COVID-19.





Abril 2020

MODIFICACIONES DEL DISEÑO Y PROGRAMACIÓN DE LOS SISTEMAS DE ALARMAS y SCORBOT IX

6

Realizamos el código necesario para que se generen las interrupciones de los sistemas de alarma

Se realizan los primeros bocetos de la conexiones para el Scorbot IX

Mayo

PREPARACIÓN DE LA REDACCIÓN DEL TFG Y MODIFICACIONES

7

Se diseña el código y diseño final de los sistemas de alarma a falta de poder realizar pruebas

Se empieza a hacer diseños en Multisim para las conexiones entre scorbot V y IX

```
/*-----------------------------------------
 * Carlos Javier Siverio Suares. Modificados por Oscar Jesus Diaz de la Fe
 * 9/06/2020
 * Codigo principal control SCORBOT-ER V
 -----------------------------------------*/

#include "counter.h"
#include "homeV.h"
#include "motor.h"
#include "speed.h"
#include "controllerPID.h"
#include "artDirection.h"
#include "stopMotor.h"

#include "TimerOne.h" // Libreria que nos permite hacer "interraciones" temporalizadas

#include <Wire.h>

Timer1.initialize(1000000);         // Dispara cada segundo
Timer1.attachInterrupt(Sobretension); // Activa la interrupcion por sobretension de los motores, asociado a Sobretension
pinMode(18, INPUT);
attachInterrupt(digitalPinToInterrupt(18), Parada2, RISING);
```

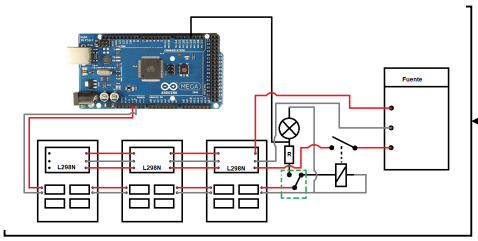
Junio

FINALIZACIÓN DEL TFG Y ÚLTIMOS DETALLES

8

Se terminan los detalles de la redacción del TFG

Se terminan los diseños de las alarmas y las conexiones de los Scorbots



5. Implementación en PCB de la electrónica de control

5.1. Consideraciones iniciales

Esta parte del Trabajo de Fin de Grado es la que tiene como función estudiar el objetivo general del proyecto, el cual consiste en implementar la electrónica de control alternativa para el manipulador 'Scorbot-ER V+' en una placa de circuito impreso PCB.

Esta es la parte común del proyecto, que involucra tanto al alumno Óscar Jesús Díaz de la Fe como al alumno Cristian Francisco Fariña Melián, y es por este motivo que este apartado del presente informe aparece en ambos trabajos.

En primer lugar, lo primero que se decidió en esta parte del Trabajo de Fin de Grado, fue qué programa utilizar para realizar los diseños de las placas PCB que compondrían nuestro circuito de control. Se decidió utilizar el programa 'Multisim 11.0', ya que ofrecía una gran facilidad para adaptar nuestras preferencias a la hora de diseñar las placas y por ser un programa bastante eficiente y práctico a la hora de realizar su cometido. Además, este programa es el que se usa en el Servicio de Electrónica de la Universidad de La Laguna y, por ello, se le podría preguntar a este servicio en caso de que hubiera alguna duda sobre el funcionamiento del programa. Por otra parte, el alumno Oscar Jesús Díaz de la Fe ya tenía experiencia utilizando este programa debido precisamente a que había realizado sus prácticas externas en el Servicio de Electrónica y, por este motivo, ya tenía familiaridad con el programa y con el propio servicio al que podía consultar en caso de duda.

En segundo lugar, se discutió acerca de cómo iba a ser grosso modo el diseño de la placa y se optó por hacer un diseño distribuido en cuatro placas. El motivo de esta decisión es que el diseño de la electrónica de control alternativa está dividido en cuatro partes diferenciadas, Este esquema de diseño modular facilita su reparación en caso de avería. Así, si una de las cuatro placas se rompiera, sólo sería necesario la sustitución de esa placa y no del conjunto del circuito. Además, se puede hacer un testeo individual de las distintas partes de la electrónica antes de su montaje definitivo. Por razones como esta, el diseño de una electrónica modular es mucho más eficiente y adecuado a nuestro circuito.

De esta forma, el diseño de las cuatro placas se haría teniendo en cuenta una cierta simetría o repetibilidad del circuito. Como se puede ver en el circuito prototipo, la electrónica está diseñada para controlar los seis motores del brazo manipulador, de forma que estos motores se controlan de dos en dos. Es decir, que necesitamos tres módulos para controlar los seis motores, por lo que sería conveniente que se hicieran tres placas PCB.

Debido a esto, y teniendo en cuenta la simetría del circuito, se harían tres placas PCB para la electrónica de control de los motores y una cuarta placa para simplemente establecer la interfaz con los elementos externos y con el propio conector D-50. Las tres placas principales inicialmente tienen un diseño idéntico

y, por tanto, sólo debemos diseñar una de las tres placas para luego replicarla dos veces más y así tener las tres placas principales. Es decir, la simetría del circuito nos permite diseñar un tercio del mismo para luego replicarlo hasta tener el diseño completo. En cada uno de estos tercios del diseño final se encontrarían dos contadores LS7166, dos integrados PCA9535 que se encargarían de conformar el bus I2C y varias resistencias pull-up necesarias para el funcionamiento de la electrónica. Además de esto, en las placas también habría una gestión de las alimentaciones y tierras del circuito, así como una distribución de las distintas señales de entrada y de salida relevantes para el diseño. Estas tres placas estarían interconectadas entre sí ya que deben trabajar juntas para el diseño completo, y además, al ser idénticas e independientes, se pueden intercambiar en principio; es decir, que la parte del diseño que se encarga de controlar, por ejemplo, la muñeca del manipulador, podría ser intercambiada por la que controla la base y el hombro del manipulador sin que haya ningún conflicto (este es otro de los beneficios del diseño modular de la electrónica de control).

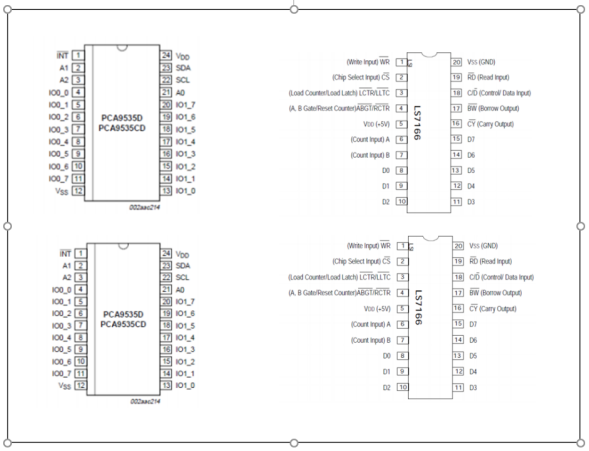


Figura 11: Diseño esquemático de un tercio de la placa PCB

En la figura 11, se puede ver un esquema de cómo sería una de las placas que se tendría que replicar tres veces. En ella figuran los pares de circuitos integrados que se habían mencionado anteriormente, pero además, habría que añadirle las resistencias pull-up correspondientes.

Cada uno de estos tercios de placa, estarían conectados a uno de los tres módulos L298N de forma que, con sus dos puentes en H, todo este conjunto serviría para controlar dos de los seis motores del brazo manipulador. Además, cada una de las placas debe ir conectada al Arduino a través de múltiples conexiónes para transmitir todas las señales necesarias para el funcionamiento del circuito.

Por otro lado, la cuarta placa a realizar trata simplemente de una placa de interfaz diseñada para contener las resistencias pull-up necesarias para el circuito, que no es posible incluir en el diseño de las otras tres placas, además de para establecer la conexión con el conector D-50. Es decir, esta es la placa

de salida del circuito. A priori, la necesidad de esta cuarta placa no se puede deducir del esquema del circuito prototipo que se había visto en la introducción, sin embargo, este esquema no es del todo preciso ni fiel al diseño final del circuito prototipo (probablemente debido a dificultades encontradas por parte de los diseñadores a la hora de llevarlo a cabo de forma práctica). Debido a que nos consta que el circuito prototipo funciona correctamente, se decidió basarse completamente en él en lugar de en el esquema a la hora de realizar nuestro diseño de placa PCB. Si bien las diferencias entre ambos elementos son mínimas, sí existen algunas discrepancias entre los dos, y por ello se optó por tener más en cuenta el circuito real que sabemos que funciona.

En este caso en concreto, en el esquema del circuito no constaban varias resistencias pull-up necesarias para el funcionamiento de la electrónica, pero debido a que en el circuito real que está implementado en el laboratorio sí constan estas resistencias, se decidió ponerlas en esta cuarta placa aparte. Por otro lado, esta cuarta placa también podría utilizarse para ejercer como interfaz con el 'Scorbot ER IX' en el caso de que quisiéramos utilizar esta electrónica con ese manipulador también.

5.2. Consideraciones generales para el diseño y fabricación de la placa PCB

Así pues, una vez comprendidos los aspectos específicos de nuestra placa PCB, se expondrán algunos conocimientos generales sobre el diseño y la fabricación de placas de circuito impreso para que luego nos sirvan de ayuda para comprender mejor el diseño final de la placa. [4]

Como sabemos, una placa de circuito impreso es una plancha de cobre en la que han sido perfiladas una serie de pistas que funcionan a modo 'cables' para que ejerzan la función de conexión entre los elementos de un circuito electrónico, ya que se trata de un elemento conductor. Además, la propia placa PCB ejerce como base para los elementos electrónicos que se encuentran ancladas a la misma y por ello los mantiene unidos a ella.

En nuestro caso, como ya se ha dicho, nuestras placas PCB tienen la función de establecer las conexiónes entre los circuitos integrados LS7166, los PCA9535 y algunas resistencias. Sin embargo, también tienen que aparecer en el diseño de nuestras placas algunos otros elementos que sirven para ejercer las conexiónes con los elementos del exterior de la placa (como puede ser otra de las placas del circuito).

La producción de placas de circuito impreso suele ser automatizada, sin embargo, en nuestro caso se ha planteado la posibilidad de fabricar nuestras placas de manera más manual a través de los recursos de los que dispone el Servicio de Electrónica de nuestra universidad de forma similar a como se realizan las prácticas de la asignatura de 'Diseño y tecnología de circuitos impresos' en el laboratorio de electrónica de la universidad.

Por otra parte, otra peculiaridad de las placas de circuito impreso son la capas de las que están compuestas. En la parte más inferior, se encuentra un

sustrato que conforma la base de la placa, a continuación se encuentra la capa de cobre que conforma el elemento conductor de la placa que permite el paso de corriente por él para establecer la conexión eléctrica. Por último, se aplica una capa de aislante como tercera capa para proteger al cobre de la misma y evitar que se deteriore. [5]

Todo esto conformaría una placa PCB de tipo simple, sin embargo, en nuestro diseño se decidió utilizar una placa doble (con una capa de cobre por cada lado), ya que nos permitía simplificar muchísimo el diseño y resultaba ser mucho más conveniente en algunas situaciones que se estudiarán posteriormente. Además de esto, también existen placas PCB de más de dos capas, pero se consideró innecesaria la utilización de este tipo de tecnologías para nuestro diseño.

Por otra parte, se hace necesario explicar una serie de términos importantes sobre la fabricación de placas de circuito impreso para que sean tenidos en cuenta en las consecuentes explicaciones sobre nuestra PCB. En primer lugar, con cara de componentes nos referimos a la cara sobre la que se colocan los componentes de tecnología de agujeros pasantes o de montaje superficial (los tipos de encapsulados de componentes se estudiarán en el apartado siguiente), cuyos terminales pasan a través de orificios practicados en la placa (esta sería la cara top). En segundo lugar, la cara de soldadura sería la cara en la que se sueldan los terminales de los dispositivos de tecnología de agujeros pasantes, pero también se pueden colocar en esta cara los componentes de montaje superficial (esta sería la cara bottom). En tercer lugar, con huella de soldadura o 'pad' nos referimos a los elementos que permiten la soldadura del componente a la placa, el cual consta siempre de una zona metalizada dentro de la placa (y de un taladro dependiendo del tipo de componente que se vaya a soldar). En cuarto lugar, con serigrafía nos referimos a la capa de etiquetas que se imprime con tinta sobre una placa PCB para identificar los nombres de los componentes. En último lugar, las pistas de la placa son los conductores que permiten conectar unos elementos con otros. Estos elementos serían como los cables de nuestro circuito y pueden encontrarse en ambas caras de nuestra placa, es decir, tanto en la cara top como en la cara bottom, aunque normalmente se encuentran en la cara bottom. [9]

Encapsulados

Otro aspecto importante a la hora de diseñar una placa PCB es el relacionado con los tipos de encapsulados posibles para los circuitos impresos que van a ser usados en el diseño. Existen muchos tipos de encapsulados, pero a groso modo pueden ser agrupados en tres grandes familias. [6]

La primera familia de encapsulados es la llamada 'Thru-Hole' (THD), encapsulado de tecnología de agujero pasante o a través de orificio. Estos componentes tienen pines preparados para ser instalados en perforaciones metalizadas y ser soldados por la capa opuesta a la que se instalan. Normalmente, este tipo de componentes se instala sólo por la capa top de la placa. [7]

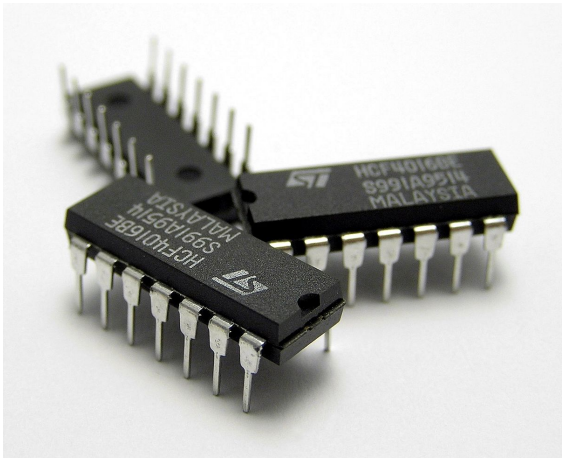


Figura 12: Ejemplo de encapsulado de tecnología de agujero pasante

La segunda gran familia de encapsulados son los llamados 'SMD' o tecnología de montaje superficial. Este tipo de componentes se montan y se sueldan como su nombre indica de manera superficial en la misma capa en la que se quieren instalar. Estos encapsulados tienen la ventaja de que pueden ser montados tanto en la capa top como en la capa bottom de la placa y, además, suelen ser más pequeños que los encapsulados de agujero pasante, por lo que permiten hacer diseños más compactos y más eficientes. [8]

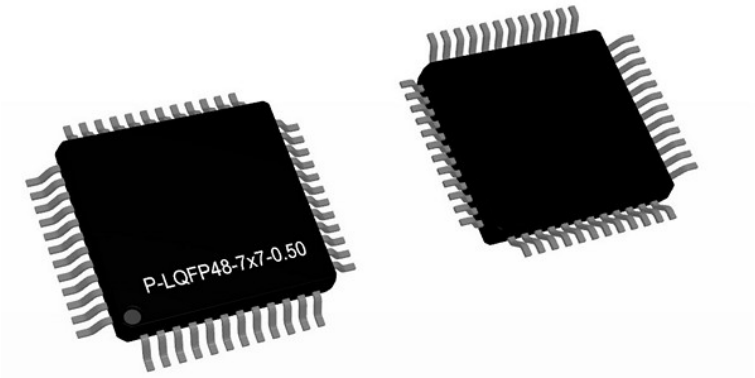


Figura 13: Ejemplo de encapsulado de tecnología de montaje superficial

Estos dos tipos de encapsulados que hemos visto son los que van a ser utilizados en nuestro diseño de la electrónica de control. Existe un tercer tipo de encapsulado llamado 'BGA' o matriz de rejilla de bolas que se caracteriza por tener una gran cantidad de pines en forma de bolas que se sueldan a la superficie de la placa a la que se le quiere instalar. No obstante, no se explicará este tipo de encapsulado en profundidad en este documento debido a que no será utilizado en nuestro diseño. [6]

Método típico para la producción manual de circuitos impresos

Como se ha dicho anteriormente, la producción de circuitos impresos suele estar automatizada. No obstante, en nuestro caso se requiere fabricar la placa PCB de manera manual con los recursos con los que cuenta el Servicio de Electrónica de la universidad. A continuación, en este apartado, se expondrá el procedimiento que se seguiría a la hora de elaborar nuestra placa PCB una vez acabado el diseño. Todo este procedimiento sigue las pautas que se estudiaron en la asignatura de 'Diseño y tecnología de circuitos impresos'. [4]

En primer lugar, habría que tener acabado el diseño de la electrónica a través de un programa de diseño de placas de circuito impreso como el 'Multisim 11.0'. Esta parte del proceso se estudiará más adelante en este documento. Una vez tuviéramos el diseño terminado, se deberían generar una serie de documentos finales utilizando el programa de edición de circuitos en cuestión. Estos documentos sirven principalmente para ser enviados a una impresora especial que se encuentra, por ejemplo, en el laboratorio de electrónica de la universidad y con ella imprimir los fotolitos correspondientes. Estos fotolitos son unas láminas transparentes que tienen dibujado sobre ellas las pistas de nuestro circuito a fabricar. [9]

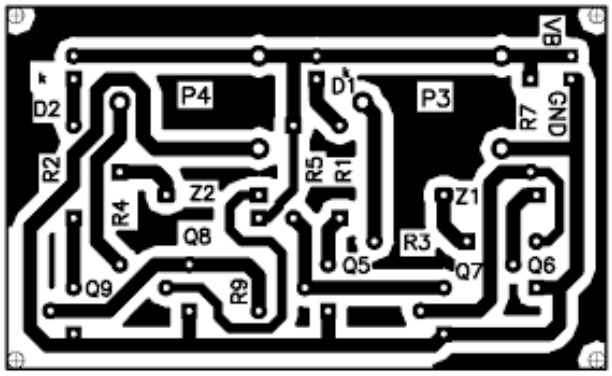


Figura 14: Ejemplo de fotolito

El siguiente paso consiste en utilizar los fotolitos generados en la etapa anterior para realizar un ataque con una insoladora de rayos ultravioleta. Esta insoladora emitiría una luz muy potente sobre la plancha de cobre con la que se quiere elaborar el circuito impreso, debilitando el cobre de las zonas donde no aparezca impresa la imagen del fotolito, ya que ambas se encuentran pegadas. Es decir, ya que en el fotolito aparecen dibujadas las pistas de nuestro circuito y el resto de la imagen es transparente, la luz de la insoladora sólo debilitaría estas zonas transparentes y dejaría intactas las pistas del circuito. Este insolado suele durar un tiempo alrededor de dos minutos. [10]

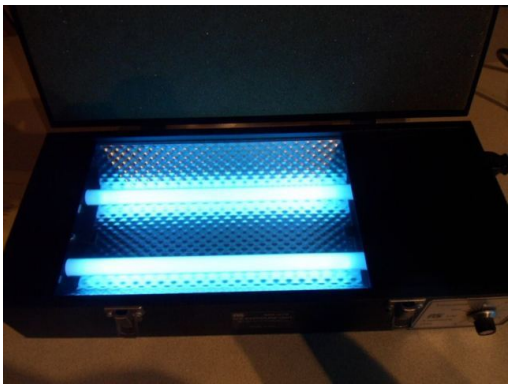


Figura 15: Insoladora

A continuación, el siguiente paso sería el ataque químico. Una vez debilitadas las partes de cobre sobrantes de nuestra placa a través de la insoladora, se procede a introducir la plancha de cobre (sin el fotolito) en un ácido especial que elimina el cobre sobrante de las partes debilitadas. Este ataque químico suele durar varios minutos y no elimina la parte del cobre que conformaría las pistas de nuestro circuito. Posteriormente, tras el ataque con ácido, se introduciría la placa en acetona. [11]

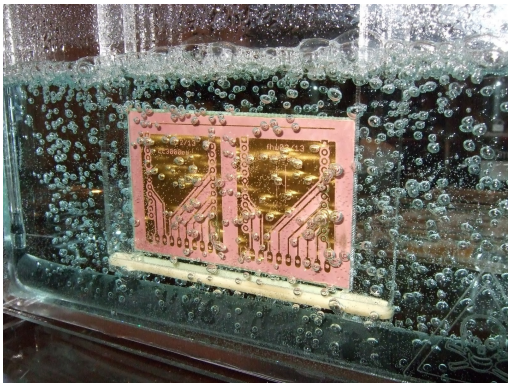


Figura 16: Ataque químico

Posteriormente, la última parte del proceso de elaboración de la placa sería el taladrado de la placa y la soldadura de componentes. Durante la etapa de taladrado, se procede a realizar los taladros correspondientes a los pines de los componentes de tecnología de agujeros pasantes y los taladros de las vías. En el caso de que se vayan a utilizar circuitos integrados de tecnología de montaje superficial, no sería necesario realizar los taladros de sus pines. Los grosores de las brocas que se suelen usar para realizar los taladros dependen del componente y suelen variar desde los 0.8 mm hasta los 2 mm. Por otro lado, para nuestro circuito en concreto, también es necesario realizar los taladros de las vías del circuito, ya que en nuestro diseño se usan tanto la cara top de la placa como la cara bottom y es necesario que ambas caras se comuniquen por medio de

estas vías. Las vías son precisamente conexiónes entre pistas para placas de dos caras a través de pequeños agujeros metalizados. Por último, con respecto a la soldadura, una vez colocados los componentes en sus respectivos sitios, se deben ir soldando uno a uno con estaño. Se comienza soldando por las vías y se continúa por aquellos que tengan menos altura hasta finalmente los de mayor altura. En el proceso de soldadura hay que tener en cuenta lógicamente el tipo de componente con el que se está trabajando, ya que no se sueldan igual los componentes de tecnología de agujeros pasantes que los componentes de montaje superficial. [12] [13]

Figura 17: Taladrado de la PCB

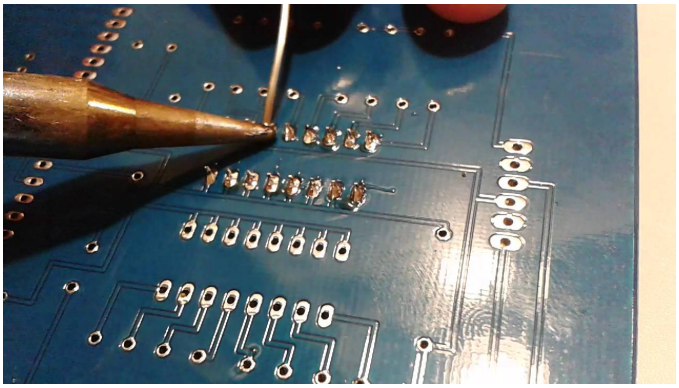


Figura 18: Soldado de la PCB

Finalmente, una vez terminada la placa, sería necesario aplicar una capa de serigrafía a la misma y comprobar su correcto funcionamiento para posteriormente poder ser puesta en servicio.

5.3. Evolución del diseño de la placa PCB

A continuación, en este apartado del Trabajo de Fin de Grado, se expondrán las diferentes etapas por las que ha ido evolucionando el diseño de nuestra placa de circuito impreso. Estos diseños, han sido elaborados con el programa 'Multisim 11.0', como se ha dicho anteriormente. Además, como veremos, el diseño ha ido pasando por diferentes modelos a raíz de los múltiples problemas que se han ido encontrando durante su desarrollo, pero siempre se ha pretendido confirmar que el modelo realizado sigue cumpliendo los objetivos deseados para el sistema.

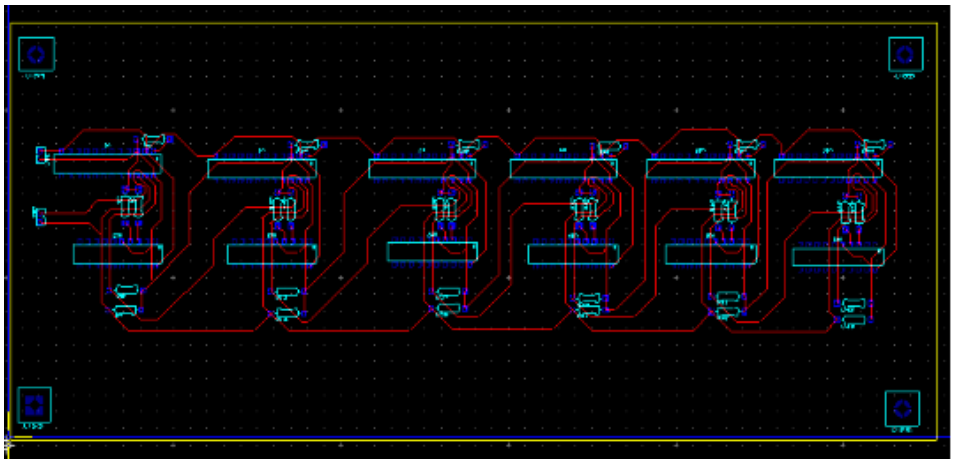


Figura 19: Primer diseño de la placa PCB

En la figura 19 se muestra el primer diseño que se realizó de la placa PCB con el programa 'Multisim 11.0'. Este diseño serviría como plantilla para realizar todos los modelos siguientes, de forma que se podría decir que este es el diseño prototipo de nuestra placa de circuito impreso para la electrónica de control.

En este diseño sólo se hace uso de la capa bottom de la placa para realizar las correcciones mediante las pistas del circuito y en él se puede ver que ya aparecen los elementos básicos como los circuitos integrados de los contadores LS7166 y los buses I2C PCA9535. Además, aparecen las resistencias pull-up necesarias para el circuito y ya están presentes las conexiónes para los otros elementos externos al circuito como las alimentaciones, el Arduino y los módulos L298N.

El problema principal de este esquema de diseño para el circuito era su gran tamaño, y que los elementos estaban distribuidos bastante mal en la placa y quedaba un diseño bastante alargado. Además, como veremos posteriormente, se habían interpretado mal los esquemas que se nos habían entregado para realizar el diseño, y no estábamos añadiendo todas las conexiónes necesarias al circuito. Por último, nótese que las esquinas de las pistas del circuito están realizadas a modo de no haber ningún giro de 90 grados en las mismas, ya que esto produciría un gran aumento de la resistencia eléctrica que no es conveniente y se pretende evitar.

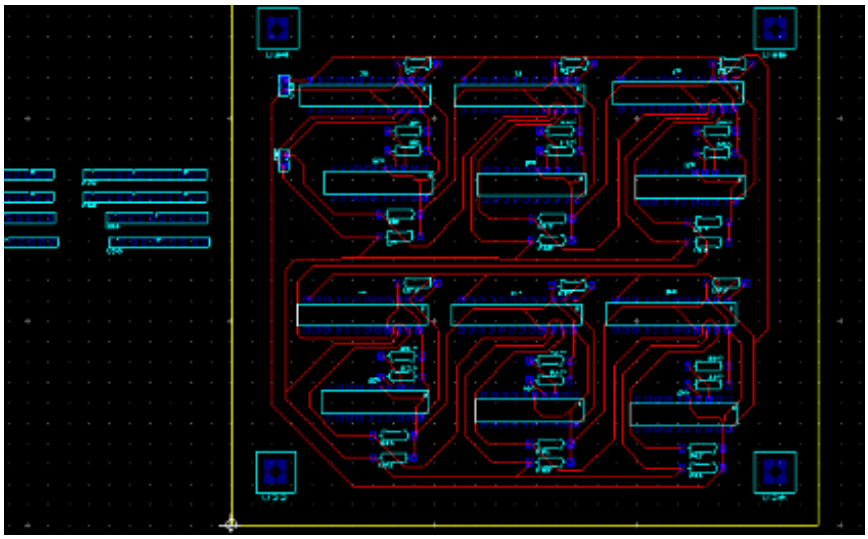


Figura 20: Segundo diseño de la placa PCB

En la figura 20 se muestra el segundo diseño de nuestra placa PCB. Este diseño es similar al anterior, pero los elementos de la placa aparecen distribuidos en forma de cuadrado para reducir el tamaño de la placa. Además, el diseño se había reorganizado de forma que era lo más compacto posible, de forma que este diseño sería posteriormente aprobado por el Servicio de Electrónica como un buen diseño para la electrónica de control, es decir, que este servicio nos daba el visto bueno para realizar nuestra placa de forma física si queríamos. Lamentablemente, como veremos posteriormente, este diseño estaba mal elaborado, ya que se habían malinterpretado los esquemas del circuito prototipo a la hora de hacerlo, así que se tuvo que reimaginar este diseño completamente tras darnos cuenta de ese error.

Otro dato de interés de este diseño (y de todos los demás) es que parecía ser que los esquemas de los circuitos integrados PCA9535 y LS7166 no se encontraban en la base de datos del sistema del programa 'Multisim 11.0', por lo que se tuvo que diseñar sus modelos en el programa a través de la información extraída de sus respectivos datasheets. Por este motivo, habría que comprobar que las resistencias y encapsulados encajen bien en el diseño una vez impresos los fotolitos antes de hacer la placa final, ya que esta comprobación no se ha hecho previamente (aún así los datos extraídos de los datasheets tienen una gran validez, así que se confía en que no habría ningún problema a la hora de hacer la producción final de la placa).

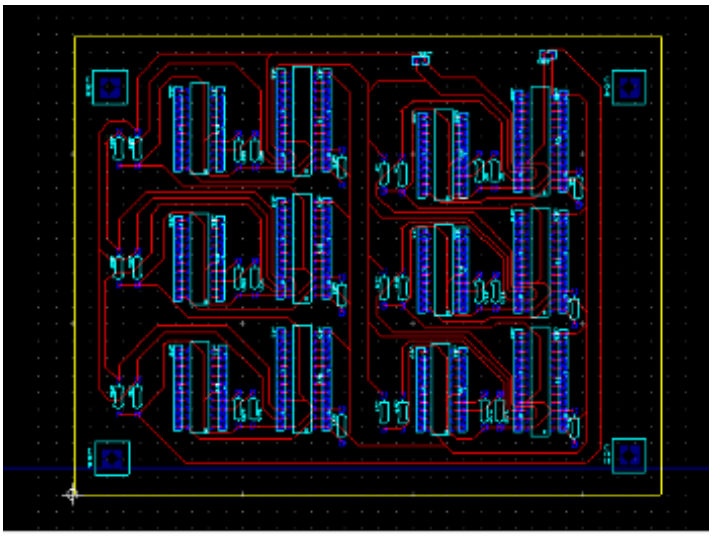


Figura 21: Tercer diseño de la placa PCB

En la figura 21 se muestra el tercer diseño de nuestra placa de circuito impreso. Este fue quizás el más importante en la elaboración de nuestro diseño de la electrónica de control en PCB, ya que es el que presenta un mayor cambio de enfoque con respecto a los diseños anteriores. Como se ha dicho anteriormente, el segundo diseño de la placa PCB parecía estar en perfectas condiciones y no tenía ningún problema grave, por lo que se pensaba que ya se había acabado el trabajo de realizar este diseño en el programa 'Multisim 11.0'. No obstante, los diseños anteriores tenían un grave problema, ya que les faltaban una gran cantidad de conexiónes a los pines de los circuitos integrados del sistema.

Para entender mejor el error que se cometió a la hora de realizar el diseño, es necesario volver a observar el esquema del circuito prototipo en el que nos basamos para realizar los diseños: [1]

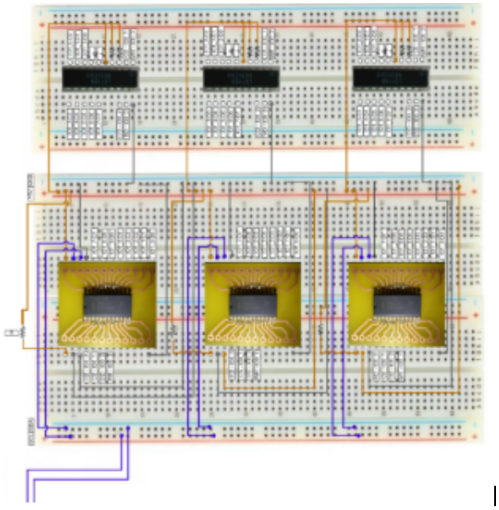


Figura 22: Parte del esquema del prototipo del circuito de control

Si nos fijamos bien en la figura 22, podemos ver varios circuitos integrados que forman parte de nuestro esquema con sus distintas conexiónes entre sus pines. A la hora de realizar los dos primeros diseños para la placa, se habían tenido en cuenta las conexiónes entre pines que se pueden ver en color naranja y azul en la imagen, no obstante, esto es un grave error, ya que estas conexiónes no son las únicas que tienen los circuitos integrados. Si prestamos atención detenidamente a la imagen, observamos que algunas de las conexiónes entre los pines de los integrados PCA9535 y LS7166 se encuentran en el esquema representados mediante cuadros de texto colocados cerca de los pines a los que hacen referencia, lo que quiere decir que nuestro diseño necesitaba de muchas más conexiónes de las que tenía en ese momento y, por ese motivo hubo que rediseñar el esquema del circuito casi completamente. Al darnos cuenta del error que habíamos cometido, comprendimos el motivo por el que realizar los otros dos diseños nos había resultado tan fácil.

Ahora que debíamos incluir nuevas conexiónes al circuito, su diseño se volvía mucho más complejo. Además, como en ese momento aún no se había planteado la posibilidad de utilizar la capa top de la placa para trazar pistas, nos daba la sensación de que era imposible realizar el diseño de esta electrónica de control en PCB. Por este motivo, se planteó la idea provisional en este diseño de usar conexiónes externas mediante cables para realizar las uniones entre pines que no era posible hacer en ese momento.

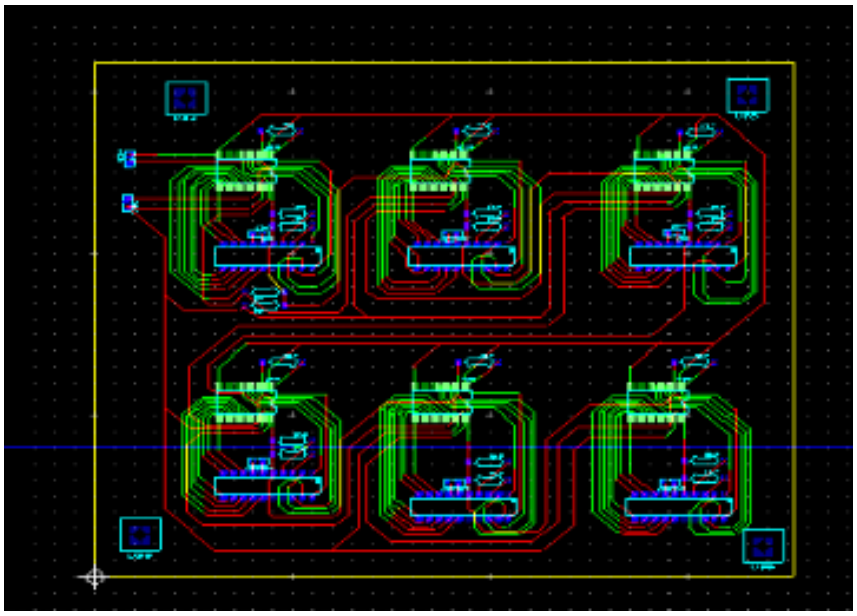


Figura 23: Cuarto diseño de la placa PCB

En la figura 23, podemos ver la cuarta etapa del diseño de la placa PCB. Este fue el diseño clave de nuestro proceso de elaboración de la placa de circuito impreso, ya que es el primero de los diseños en hacer uso de las capas top y bottom para trazar las pistas. Realmente, este es el primer diseño funcional de nuestro circuito, ya que aunque no es del todo eficiente, se piensa que sí cumplía con las características necesarias para funcionar correctamente. Un aspecto importante de este diseño de placa es que hacía uso de muchas vías para conectar las capas top y bottom, pues no se había planteado inicialmente usar los propios pads de los contadores para comunicar ambas capas.

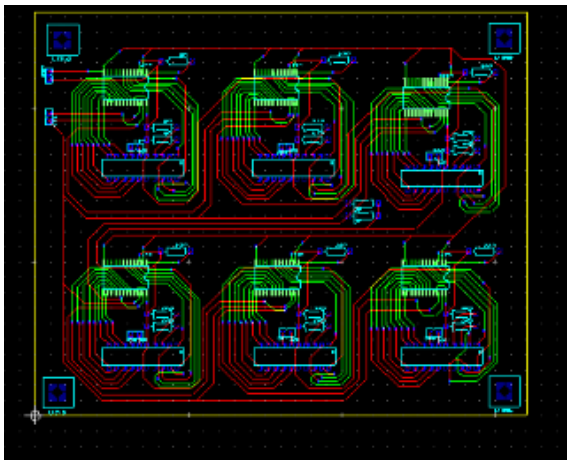


Figura 24: Quinto diseño de la placa PCB

La figura 24, nos muestra el quinto diseño de nuestra placa de circuito impreso. Este planteamiento tenía un aspecto muy similar al anterior, sin embargo, había sido reajustado para ser mucho más compacto y pulido que sus predecesores. Este diseño solucionaba todos los problemas que tenía el anterior, de forma que volvíamos a estar aprobados por el Servicio de Electrónica de la universidad para realizar nuestra placa si queríamos. Sin embargo, aún nos quedaba un aspecto de importancia a considerar en nuestra placa PCB, el diseño modular.

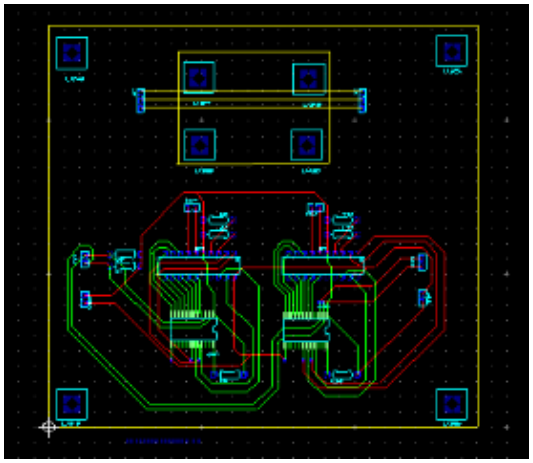


Figura 25: Sexto diseño de la placa PCB

La figura 25, se muestra el sexto diseño de nuestra placa PCB. Este fue el primer diseño en considerar la idea de hacer nuestro circuito de forma modular, es decir, realizar varias placas en lugar de sólo una, como se ha explicado en apartados anteriores de este documento, para que nuestro sistema fuera más robusto en caso de que se rompiera alguno de los componentes (no haría falta

cambiar el conjunto, sino la placa en la que se ha roto el componente en cuestión).

En este diseño de placa, se ha prescindido de usar la gran cantidad de vías que se usaban en diseños anteriores, ya que se utilizan los mismos pads de los contadores para relacionar las capas top y bottom de la placa PCB. Además, debido a su estructura modular, en la imagen podemos ver que el diseño cuenta sólo con dos integrados PCA9535 y dos integrados LS7166, junto con sus respectivas resistencias. Por otro lado, podemos ver en la parte superior que se ha incluido un espacio para anclar el módulo L298N, de forma que todo el circuito estaría dividido en tres partes como se ha explicado anteriormente. El diseño también plantea las conexiónes de algunos de los pines comunes de este módulo L298N.

Por otra parte, en este momento de la evolución del diseño, se descubrió un inconveniente a la hora de establecer nuestro circuito de forma modular, y es que los tres tercios de circuito que se habían planteado en un principio no eran exactamente iguales, por lo que no se podía realizar su diseño una sola vez para luego replicarlo dos veces más. La solución a este problema se irá planteando en los diseños siguientes.

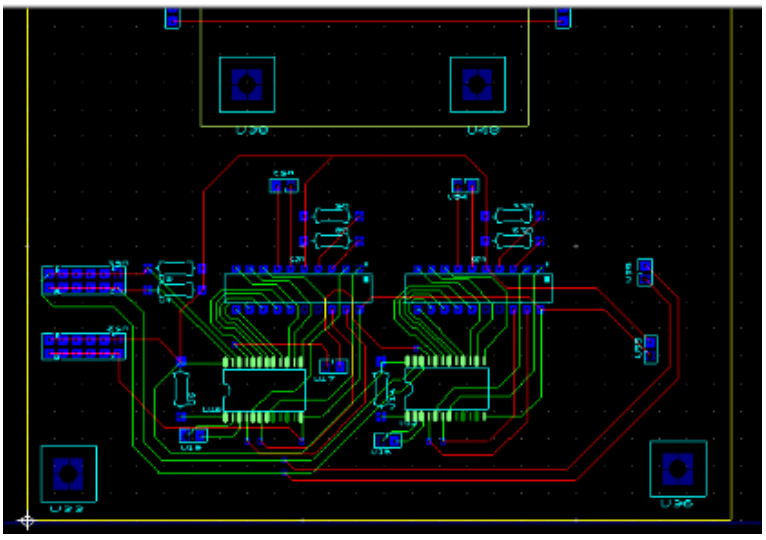


Figura 26: Séptimo diseño de la placa PCB

La figura 26 nos muestra el último diseño de nuestra placa de circuito impreso antes de llegar al diseño final que se adoptaría como el definitivo. Esta imagen está cortada, ya que la parte superior del circuito no ha sufrido cambios con respecto a la versión anterior. De hecho, este diseño del circuito es prácticamente igual al anterior, sólo que atendiendo al problema de que nuestro diseño modular tiene el inconveniente de que algunas de las conexiónes entre pines no son exactamente iguales entre los tercios de la placa completa. En concreto, el problema es que algunos de los pines de los circuitos integrados deben ser pues-

tos a tierra o a la alimentación de 5 voltios alternativamente de forma distinta para cada uno de ellos. Por este motivo, este diseño tiene varios conectores con la idea de usar cables para realizar las conexiónes concretas individuales de cada pin del integrado en cuestión.

5.4. Diseño final de la placa PCB

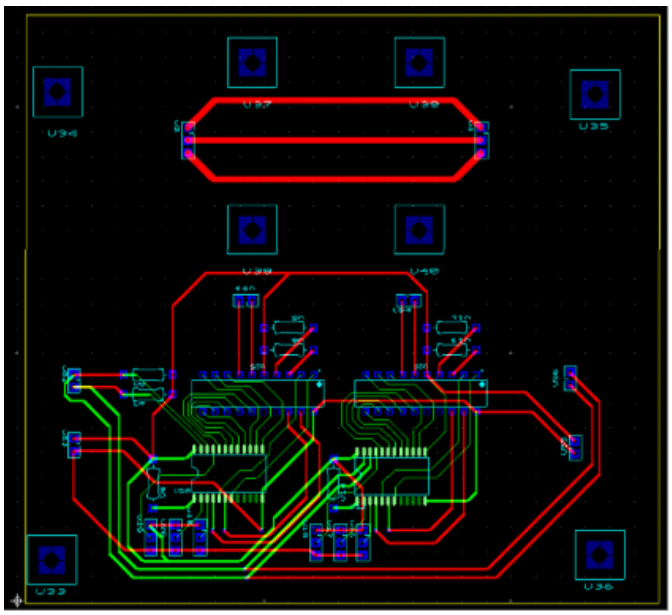


Figura 27: Diseño final de la placa PCB

En la figura 27, se muestra el diseño final que se adoptó para la implementación de la electrónica de control alternativa del brazo manipulador en placa PCB. Este circuito tiene todas las características necesarias para ser implementado de forma funcional. A de tenerse en cuenta que este circuito debe ser fabricado de forma física tres veces para formar el circuito completo.

La principal diferencia entre este circuito y su versión anterior es que en este se decidió hacer uso de 'jumpers' para realizar las conexiónes necesarias que en el circuito anterior se planteaba hacer mediante cables. Esto se hace con el fin de simplificar el diseño y mejorar la forma de realizar la conexión concreta de cada una de las tres placas.

Las principales características a destacar de este diseño final de la placa son las siguientes:

- La placa tiene unas dimensiones de 131 mm por 121 mm.
- Las pistas de cara top tienen anchos de entre 0.7 mm las más gruesas y de 0.4 mm las más finas.

- Las pistas de la cara bottom son las de los circuitos integrados de los contadores LS7166 y tienen un ancho de 0.7 mm. También en esta cara se encuentran varias pistas para el módulo L298N, que tienen un ancho de 1.5 mm ya que se trata de la parte de la electrónica de potencia de nuestro circuito.
- Se usan seis Headers HDR1X2 y ocho headers HDR1x3 para realizar las conexiónes con los dispositivos externos.
- Los dos conectores HDR1x3 de la parte superior son para el módulo L298N (son las pistas para conectar las alimentaciones de 5 voltios, 24 voltios y tierra) y los otros seis conectores HDR1X3 son tres para un contador y los otros tres para el otro. La función de estos últimos conectores es la de realizar la conexiónes concretas de cada uno de los contadores a través de un jumper dependiendo de si el pin debe ir conectado a tierra o a 5 voltios (como se ha dicho anteriormente). Las conexiónes superiores son para 5 voltios, las inferiores son para tierra y las del centro van conectadas a su pin concreto (de izquierda a derecha: pines 24, 2 y 3 del contador).
- Cuatro de los conectores HDR1X2 son para las señales de salida que van al Arduino (señales SDA y SCL) y señales de alimentación de 5 voltios y tierra. Los otros dos conectores HDR1X2 son para las señales PA y P0 de cada contador.

Por otro lado, con el programa Multisim 11.0 se puede hacer una simulación de cómo sería la placa en 3D una vez fabricada. El resultado sería el de la figura 28 y 29:

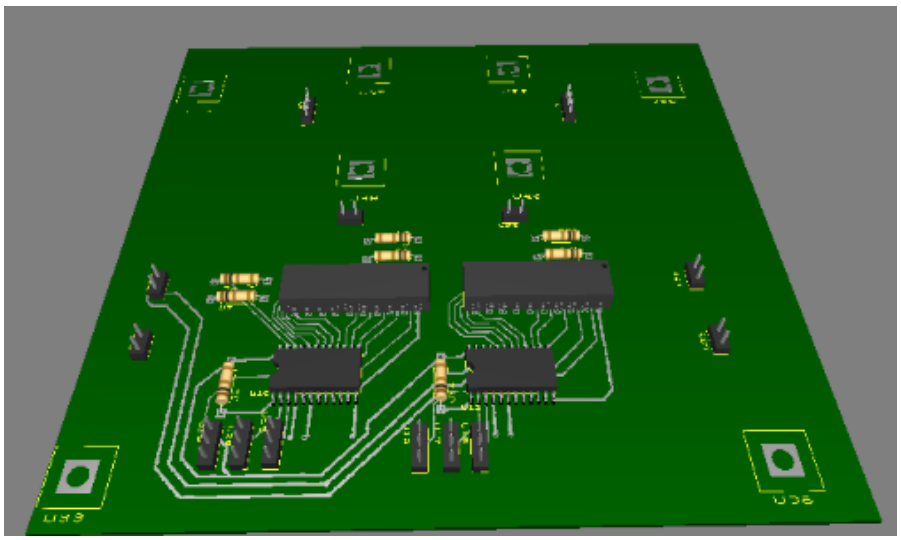


Figura 28: Placa final en 3D (cara top)

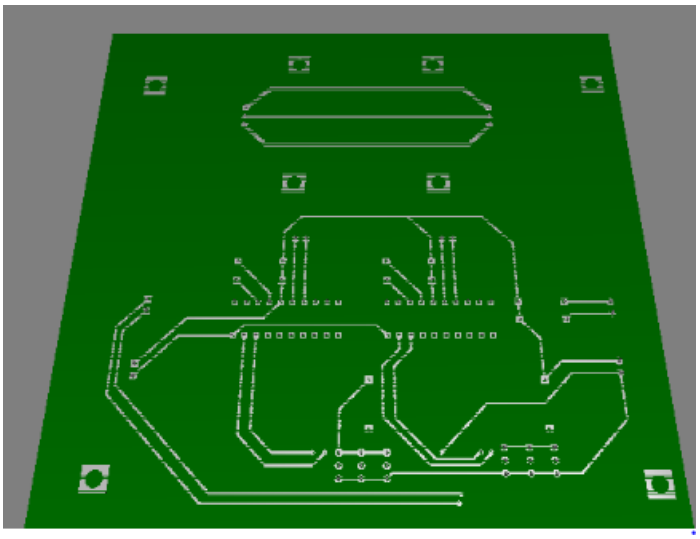


Figura 29: Placa final en 3D (cara bottom)

Finalmente, el esquema de cómo sería el conjunto de los tres tercios de placas necesarios para formar el circuito completo sería el de la figura 30:

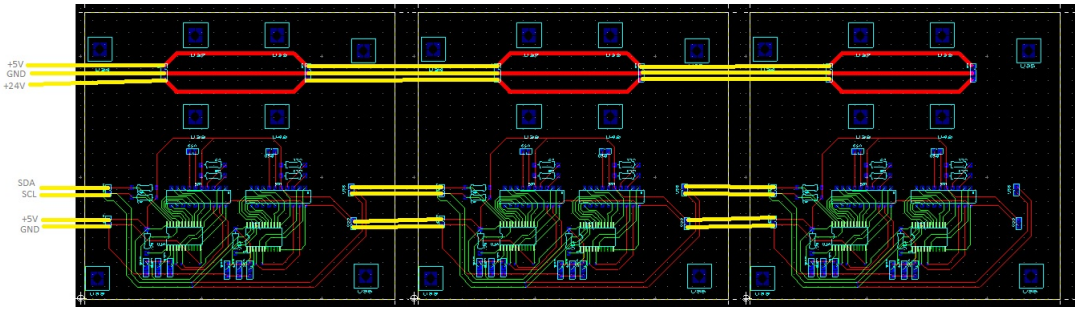


Figura 30: Unión entre las placas

6. Implementación del sistema a la electrónica el Scorbot-ER IX.

El diseño para el Scorbot-ER V realizado por Ana Estévez Pérez y Carlos Javier Siverio Suárez [1] está planificado para que en principio se contemplara la opción de disponer de una electrónica que fuese aplicable con el Scorbot-ER IX. Por esto el disño base de la electrónica se mantendra igual para ambos caso y se diseñara la conexión con los distinteos Scorbot-Er.

Lo primero que se hara sera buscar el catálogo del Scorbot-ER IX de ïntelitek"[14] para poder ver como son las conexiónes y el funcionamiento de cada uno de sus pines. A continuacion se mostrarán los conectores y una tabla mostrando las conexiónes requeridas

El conector macho Burndy de 19 pines que une el cable de alimentación a la parte posterior del controlador. El cable del robot contiene 12 cables.

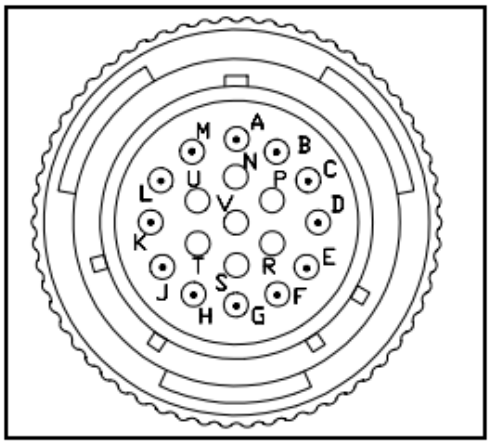


Figura 31: Conector Burndy de 19 pines

Robot (Power) Cable Wiring and Connector			
Pin ID	Pin Description Robot Side (J1)	Beldan Color	Pin Description Controller Side (P1)
A	Motor 1 –	black	M0_A
M	Motor 1 +	red	M0_B
C	Motor 2 –	brown	M1_A
L	Motor 2 +	orange	M1_B
E	Motor 3 –	yellow	M2_A
H	Motor 3 +	purple	M2_B
B	Motor 4 –	light blue	M3_A
K	Motor 4 +	blue	M3_B
D	Motor 5 –	grey	M4_A
J	Motor 5 +	pink	M4_B
F	Motor 6 –	white	M5_A
G	Motor 6 +	green	M5_B

Figura 32: Tabla del conector Burndy

El cable del encoder, que conecta el controlador a los codificadores del motor y los interruptores, Contiene 36 derivaciones. La imagen muestra el conector hembra D37.

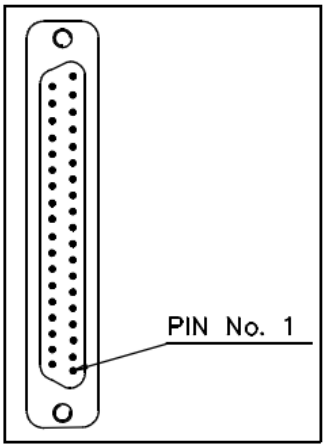


Figura 33: Coble del encoder D37

Encoder Cable and D37 Connector				
Pin ID	Pin Description Robot Side (J1)	Axis	Telephone Cable Color	Pin Description Controller Side (J2)
1	+5V	1	red	+5V
8	COMMON		yellow	COMMON 0
5	CHA1 (Encoder Pulse A)		green	CHA 0
6	CHB1 (Encoder Pulse B)		white	CHB 0
7	CHC1(Encoder Index Pulse)		black	CHC 0
31	MSWITCH (Home Switch)		blue	MSWITCH
1	+5V	2	red	+5V
12	COMMON		yellow	COMMON 1
9	CHA2 (Encoder Pulse A)		green	CHA 1
10	CHB2 (Encoder Pulse B)		white	CHB 1
11	CHC2 (Encoder Index Pulse)		black	CHC 1
32	MSWITCH (Home Switch)		blue	MSWITCH
1	+5V	3	red	+5V
16	COMMON		yellow	COMMON 2
13	CHA3 (Encoder Pulse A)		green	CHA 2
14	CHB3 (Encoder Pulse B)		white	CHB 2
15	CHC3 (Encoder Index Pulse)		black	CHC 2
33	MSWITCH (Home Switch)		blue	MSWITCH

Figura 34: Tabla del cable del encoder 1

Encoder Cable and D37 Connector				
Pin ID	Pin Description Robot Side (J1)	Axis	Telephone Cable Color	Pin Description Controller Side (J2)
2	+5V	4	red	+5V
20	COMMON		yellow	COMMON 3
17	CHA4 (Encoder Pulse A)		green	CHA 3
18	CHB4 (Encoder Pulse B)		white	CHB 3
19	CHC4 (Encoder Index Pulse)		black	CHC 3
34	MSWITCH (Home Switch)		blue	MSWITCH
2	+5V	5	red	+5V
24	COMMON		yellow	COMMON 4
21	CHA5 (Encoder Pulse A)		green	CHA 4
22	CHB5 (Encoder Pulse B)		white	CHB 4
23	CHC5 (Encoder Index Pulse)		black	CHC 4
35	MSWITCH (Home Switch)		blue	MSWITCH
2	+5V	6	red	+5V
28	COMMON		yellow	COMMON 5
25	CHA6 (Encoder Pulse A)		green	CHA 5
26	CHB6 (Encoder Pulse B)		white	CHB 5
27	CHC6 (Encoder Index Pulse)		black	CHC 5
36	MSWITCH (Home Switch)		blue	MSWITCH

Figura 35: Tabla del cable del encoder 2

Ahora que conocemos la descripcion de cada pin, procedemos compabarlo con el diseño del Scorbot-ER V

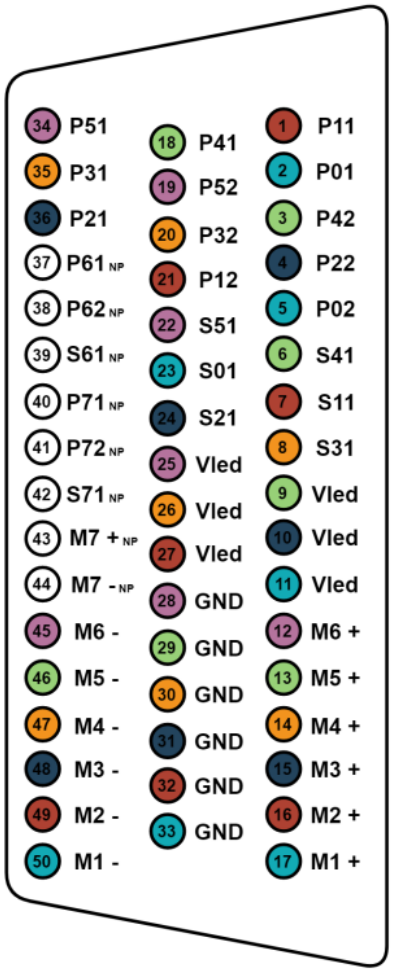


Figura 36: conexiónes Scorbot-ER V extraido del TFG de Ana Estévez Pérez

Como podemos comprobar, la tension Vled de 5V y la GND es común. Las señales de M+, M-, switch, pulse A y pulse B son propios de cada motor.
A continuación debemos ver como sacar el .enˆcoder index pulse.º pulso Z, esta es una señal digital que se genera en el encoder una vez por revolución. Que permite dar mayor precisión en los codificadores rotativos.

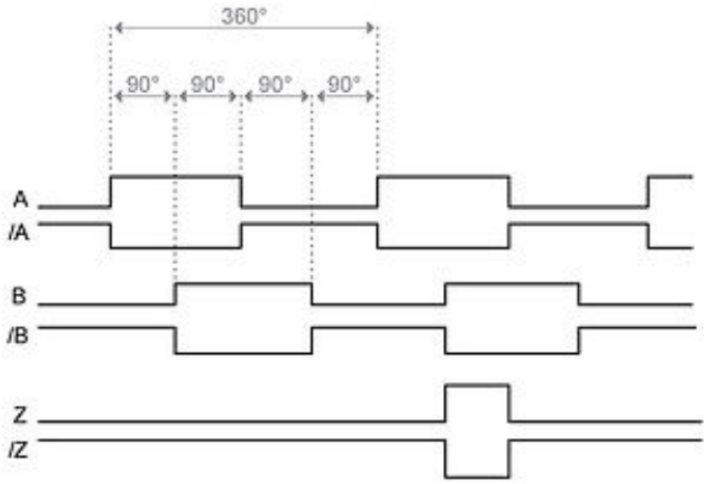


Figura 37: pulso A B y Z

Por lo que revisaremos el datasheet del LS7166 [19] en busca de la señal que podamos utilizar para nuestro sistema

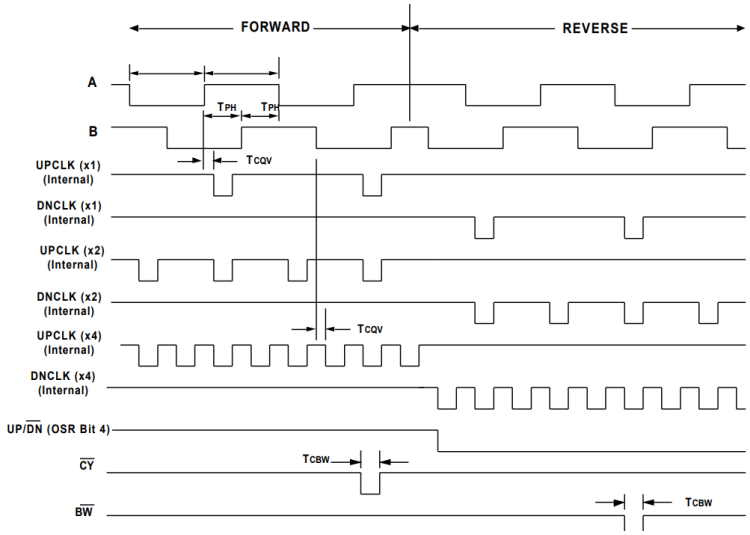


Figura 38: Modos de configuración de conteo en cuadratura

Consideramos que la señal que vamos a utilizar para el incoder es el CY no negado. Procedemos a hacer las modificaciones en el código de Çounter.cpp"

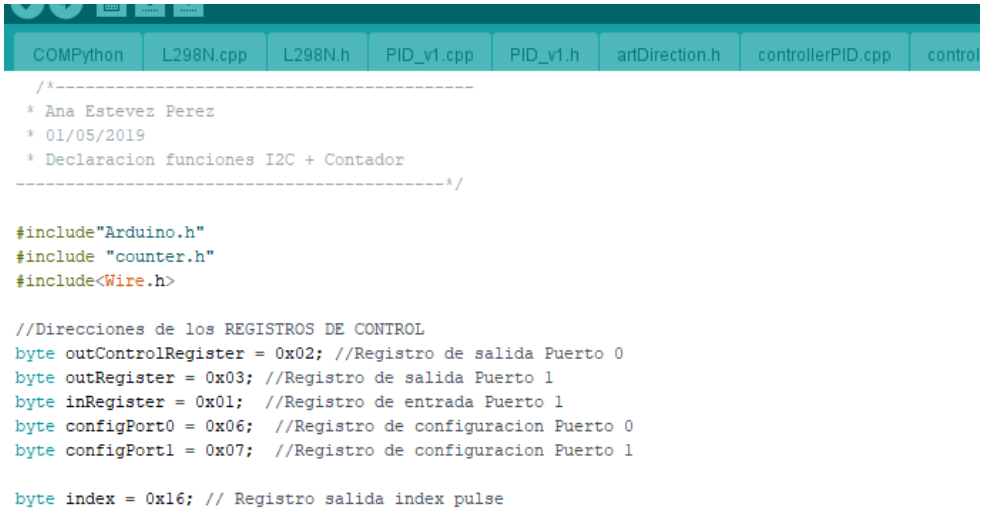


Figura 39: Codigo Çounter.cpp"modificado 1

Con estos cambios, ahora solo faltaría el diseñar una placa PCB con la que incorporar los 3 tipos de conectores y conectar cada uno de sus pines. Además se podria añadir un interruptor para que se permita la alimentación a uno o a otro Scorbot y dejar el uso del brazo concreto a elección del usuario.

```
void counterSetup(int art){
  //Configuramos los puertos como salida inicialmente
  outPortConfig(art, configPort0);
  outPortConfig(art, configPort1);ç

  outPortConfig(art, !index);

  //Inicializamos variables de control del contador
  controlCounter(art, WR, CD, CS, RD);

  //Inicializamos el contador
  inicializeCounter(art);
}
```

Figura 40: Codigo Çounter.cpp"modificado 2

7. Sistema de gestión de alarmas

Se pretenden diseñar sistemas para evitar los posibles fallos que puedan ocurrir en el proyecto, para ello tenemos encuenta las siguientes posibilidades.

7.1. Parada de emergencia Z

Esta parte del Trabajo de Fin de Grado tiene el objetivo de diseñar la parada de emergencia para el controlador del Scorbot-ER, pues se necesita que exista un método para desactivar el sistema en caso de que se produzca algun problema.

Este apartado resulta de gran importancia, debido a que sirve para prevenir diversas situaciones de riesgo, tanto para el usuario como para la maquinaria. Por tal motivo es por el que la parada de emergencia se debe llegar a realizar con una sola maniobra, sin la dependencia de ningún programa electrónico, para que se ejecute lo más rápido posible.

Tras analizar el sistema y diseñar varios conceptos, se optó por diseñar la parada de emergencia como se ve en la figura 41.

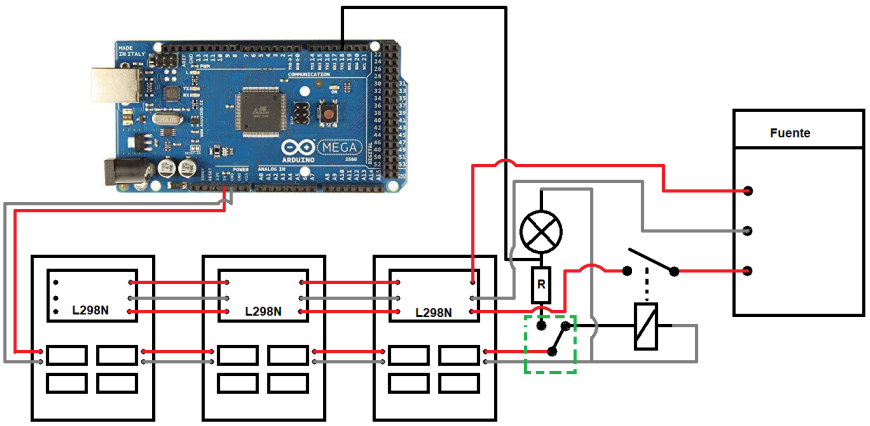


Figura 41: Sistema analógico de la parada Z

Utilizando como alimentación la tensión de 5V del arduino, conectado a un interruptor SPDT (Simple Polo Doble Corte) y a un relay, que permite el cierre del circuito de la fuente de alimentación de 24V, encargada de alimentar el módulo L298N, para dar la tensión necesaria a los motores

El concepto de funcionamiento de este sistema es el siguiente. Si ocurre cualquier problema, el usuario debe accionar manualmente el interruptor. Esto impedirá el paso de corriente a traves de la bobina del relay, generando su desconexión y, por tanto, abriendo el circuito de la fuente de 24V, que impedirá la alimentación de los motores del Scorbot-ER que se está usando. Además, al accionar el interruptor, se cerrará un circuito que se encarga de alimentar un led rojo, destinado a avisar de la activación de la parada y de una señal que informará a uno de los pines del arduino, para que se finalice el código en ejecución y mande por pantalla una señal de error

Con respecto al código necesario para la desactivación del programa que se está ejecutando en el arduino puede ser bastante simple, pues básicamente solo se debe generar un mensaje de error por pantalla y mandando un exit(1) para que se deje de ejecutar.

Sin embargo, la mayor complicación es el uso de interrupciones externas, pues al no poder determinar cuando se activa la parada, se debe encontrar la manera de que el arduino este detectando constantemente esta señal. Finalmente se optó por el uso del comando 'attachInterrupt ()' [15] con el cual podemos hacer que el código detecte la variación en un pin digital concreto, en nuestro caso el pin digital 18, y ejecutar una sección de código concreta.

En las figuras 42 y 43 se puede ver el código principal del controlador, el cual a sido modificado para que actúen las interrupciones mencionadas anteriormente. En concreto, en la figura 42 se puede ver la línea de código en la que se usa la función 'attachInterruption' a la que va referida para la función 'ParadaZ', y en la figura 43 podemos ver dicha función

```
/*-----------------------------------------
 * Carlos Javier Siverio Suarez. Modificados por Oscar Jesus Diaz de la Fe
 * 9/06/2020
 * Codigo principal control SCORBOT-ER V
-----------------------------------------*/

#include "counter.h"
#include "homeV.h"
#include "motor.h"
#include "speed.h"
#include "controllerPID.h"
#include "artDirection.h"
#include "stopMotor.h"

#include <TimerOne.h> // Libreria que nos permite hacer "interraciones" temporalizadas

#include <Wire.h>


//Definimos variables de entrada de la COM
String string;
int command = 0;
long int option1, option2, option3, option4, option5;
double r;

void setup(){

  Timer1.initialize(1000000);         // Dispara cada segundo
  Timer1.attachInterrupt(Sobretension); // Activa la interrupcion por sobretension de los motores, asociado a Sobretension
  pinMode(18, INPUT);
  attachInterrupt(digitalPinToInterrupt(18), ParadaZ, RISING);

  Serial.begin(9600);
  Wire.begin();
  //Inicializamos las funcionalidades de "counter", "homeV" y "ControllerPID"
  completeCounterSetup(art1, art2, art3, art4, art5, art6);
  homeVSetup(microSwitch1, microSwitch2, microSwitch3, microSwitch4, microSwitch5, microSwitch6);
  SetupPID();
}
```

Figura 42: Código modificado para incorporar la parada Z. Diseño de interrupción

```
void ParadaZ(){
  Serial.println("ERROR Parada Z");
    exit(1);
  }
```

Figura 43: Código modificado para incorporar la parada Z. Función parada Z

Como se ha indicado anteriormente, este sistema necesitaría únicamente de un interruptor SPDT y un relay de 24V. Además se le puede añadir un led rojo de 5mm y una resistencia de un kilo ohmios para que el dicho led no se queme, para realizar la indicación visual de que la parada de emergencia está activa

7.2. Parada por sobretensión

En este apartado, se tratará la parada por sobretensión para el controlador. Consiste en que el programa del arduino detecte si la tensión que llega a los motores es superior al límite previamente establecido. Si este fuera el caso, a causa de un mal funcionamiento de los motores o por encontrarse estos atascados, el arduino deberá detener inmediatamente el código en ejecución.

Tras plantear este punto, se procedió a realizar las ideas conceptuales del diseño, pues originalmente se iba a realizar una desconexión mediante hardware de la alimentación de los motores, de manera similar a la explicara anteriormente para la parada de emergencia Z. Sin embargo, se optó al final con que se realizará por medida software.

El primer paso consistió en determinar un sistema para medir la tensión que hay en los motores. Teniendo en cuenta que cada motor puede recibir tensiones positivas o negativas, determinados en los pines del conector M+ y M-, por lo que en total tenemos unas 12 tensiones posibles (6 para M+ de los motores y 6 para M-). Además, tenemos que tener en cuenta que las tensiones que tienen los motores es demasiado alta para al arduino, para ello deberemos usar un divisor de tensión diseñado para reducir la tensión a 5V DC [16]

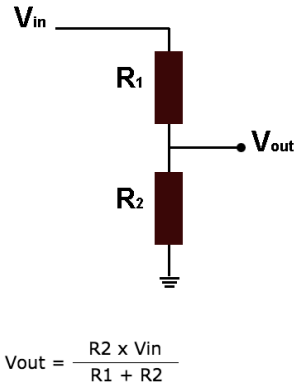


Figura 44: Divisor de tension

Para esto, también tendremos que determinar la tensión máxima que pueden soportar los motores del Scorbot-er, en este caso están diseñados para funcionar a 24V DC, por lo que lo tomaremos como valor máximo

Como Vout será 5V, pues es la tensión que soporta el arduino, y consideraremos Vin como 24V obtendremos que 19*R2 = 5*R1. Si ponemos que R2 es 1 kilo ohmios, entonces R1 es 19/5 = 3.8 Kilo ohmios. Cogeremos un R1 inferior, de 3 kilo ohmios, dándonos una Vin de 20 V DC

De esta forma el montaje físico para los motores M0 y M1 quedarían como se muestra en la figura 45.

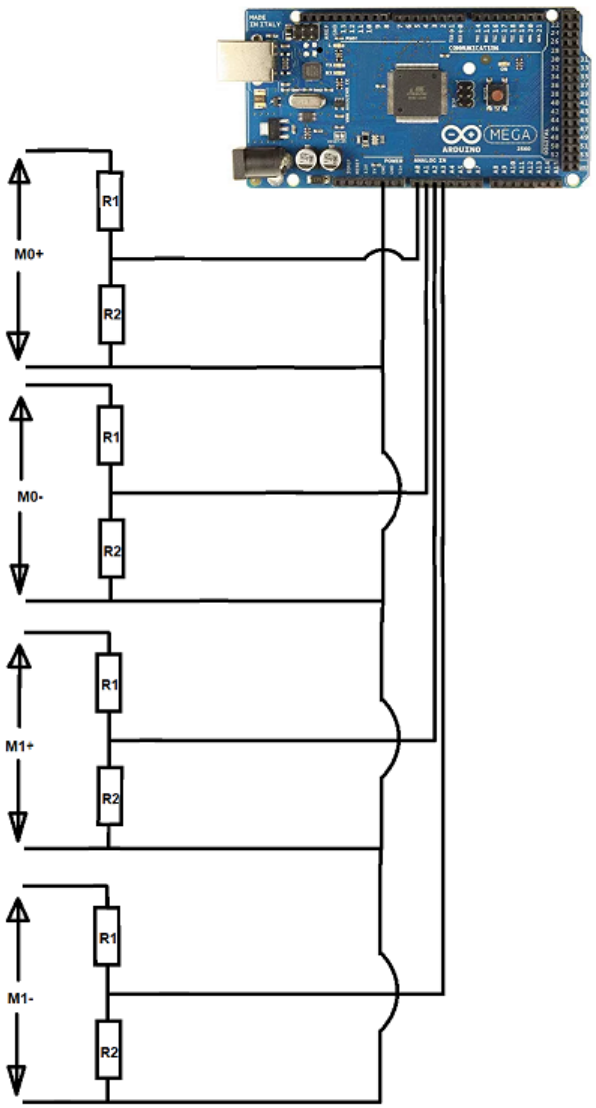


Figura 45: Divisor de tension para M0 y M1

De esta forma ocuparemos los pines desde A0 a A11, para los motores desde M0 a M5.

En cuanto al software, como hay demasiadas tensiones analógicas, el comando attachinterruption() no nos será útil. Para poder controlar este tipo de fallo la solución que hemos tomado es la siguiente, programaremos una interrupción que se ejecutará cada segundo, cuya funcionalidad será medir las tensiones analógicas y si detecta una tensión superior a nuestro umbral, mandara un mensaje de error y generara un exit(1) para cerrar el sistema.
Esto se hará usando la librería "TimerOne.h"[17] que es la que nos permitirá generar las interrupciones temporales, creando el objeto Timer1, el cual lo indicaremos con un tiempo de un segundo, y usando el comando 'Timer1.attachInterrupt()' cuya finalidad resulta igual a la del 'attachInterrupt()', como podemos ver en la imagen 46.
Estas líneas del código harán que se active la función "sobretensión"que se encargara de medir la tensión de cada uno de los motores, y si detecta un valor superior a la tensión máxima permitida procederá a detener la ejecución del código entero.

```
/*-----------------------------------------
 * Carlos Javier Siverio Suarez. Modificados por Oscar Jesus Diaz de la Fe
 * 9/06/2020
 * Codigo principal control SCORBOT-ER V
-----------------------------------------*/

#include "counter.h"
#include "homeV.h"
#include "motor.h"
#include "speed.h"
#include "controllerPID.h"
#include "artDirection.h"
#include "stopMotor.h"

#include <TimerOne.h> // Libreria que nos permite hacer "interraciones" temporalizadas

#include <Wire.h>


//Definimos variables de entrada de la COM
String string;
int command = 0;
long int option1, option2, option3, option4, option5;
double r;

void setup(){

  Timer1.initialize(1000000);         // Dispara cada segundo
  Timer1.attachInterrupt(Sobretension); // Activa la interrupcion por sobretension de los motores, asociado a Sobretension
  pinMode(18, INPUT);
  attachInterrupt(digitalPinToInterrupt(18), ParadaZ, RISING);

  Serial.begin(9600);
  Wire.begin();
  //Inicializamos las funcionalidades de "counter", "homeV" y "ControllerPID"
  completeCounterSetup(art1, art2, art3, art4, art5, art6);
  homeVSetup(microSwitch1, microSwitch2, microSwitch3, microSwitch4, microSwitch5, microSwitch6);
  SetupPID();
}
```

Figura 46: Código modificado para por sobretención. Diseño de interrupción

```
void Sobretension(){
  for(int i=0; i<12; i++){                    //Todo lo que ocurre en este for es una lecura de la tension de los distintos puntos de los motores
    float v= (analogRead(i) * 4.98) / 1024.0;
    float v2 = v / (1 / (3 + 1));             //Se genera tambien una transformacion matematica por si desea ver segun la tension en los motores
    if(v2 > 26){
      Serial.println("ERROR SOBRETENSION");
      exit(1);
    }
  }
}
```

Figura 47: Código modificado para por sobretención. Función

8. Pruebas

Por desgracia debido al COVID-19 y a la cuarentena causada se nos ha hecho imposible realizar pruebas de laboratorio. Sin embargo al poder aprovechar una placa UNO en nuestra casa hemos podido probar en una menor medida los códigos que hemos estado realizando y experimentar con las múltiples ideas y problemas que se han realizado a lo largo de este proyecto.

9. Dificultades encontradas

La dificultad más grande encontrada sin duda ha sido la causada por el COVID-19 y la cuarentena causada, pues nos ha impedido realizar el proyecto en los laboratorios de la universidad.

También uno de los mayores desafios ha sido el aprender a realizar interrupciones en arduino y como aplicarlas en el código diseñado.

El diseñar de la formas más eficiente el proyecto en una placa de circuito impreso, teniendo completada una placa basada en lo diseñado en los trabajos de final de grado '*Implementación de la electrónica para el control dinámico de un manipulador Scorbot-ER Vplus. Interfaz electrónica, lectura de codificadores digitales de posición y cálculos cinemáticos*' e '*Implementación de la electrónica para el control dinámico de un manipulador Scorbot-ER. Desarrollo de la comunicación*'

Al estar usando una versión de prueba hemos tenido algún problema con el programa, como puede ser el hecho de no haber podido imprimir el fotolito del circuito. Esto es porque se tenia previsto dar el codigo al servicio electronico, los cuales ya poseen este programa, y que pudieran hacer la placa en físico

10. Mejoras futuras

El alcance de este proyecto fue muy ambicioso en un principio. Sin embargo, debido a las circunstancias adversas que han acontecido a lo largo del mismo y debido también a la falta de tiempo, el proyecto ha tenido algunas limitaciones que no se han podido solventar. Por este motivo, en este apartado del documento se incluirán algunas mejoras que se le deberían añadir al proyecto en el caso de que alguien quisiera continuarlo desde el punto en el que se ha dejado.

- En primer lugar, la mejora más evidente para este proyecto sería la realización de la placa de circuito impreso con la electrónica de control de manera física para ser implementada en el manipulador. Esta era la finalidad de este proyecto, pero debido a los motivos que se han ido explicando, no se ha podido llevar a cabo.
- Otra mejora importante sería fabricar una carcasa donde meter la placa PCB con los demás componentes importantes para su funcionamiento como el Arduino. Esta carcasa se podría fabricar con una impresora 3D y debería tener unas dimensiones adecuadas para su finalidad.
- Además, se debería implementar la interfaz entre la electrónica de control y el brazo robótico 'Scorbot IX' y probar su funcionamiento.

11. Conclusiones

Las conclusiones a las que se han llegado a través de la ejecución de este Trabajo de Fin de Grado son muy diversas. A lo largo de la realización de este trabajo se han desarrollado una gran cantidad de conocimientos técnicos de gran importancia para el desempeño de un futuro ingeniero técnico industrial.

A través de este trabajo, se ha aprendido a usar un programa profesional de diseño de circuitos impresos como es Multisim 11.0 y a solucionar todos los problemas que el diseño de una placa PCB conlleva. De esta forma, se ha aprendido no sólo a manufacturar una placa de circuito impreso que haga las funciones de la electrónica de control de un brazo manipulador, sino también se ha aprendido a diseñar placas PCB complejas en general.

Además, otros conocimientos tangenciales que se han adquirido han sido los relacionados con la planificación de proyectos y la solución de inconvenientes que van surgiendo durante el desarrollo de un proyecto en sí. Así pues, también se ha profundizado en el estudio y comprensión de documentación técnica, ya que se ha tenido que investigar mucho acerca de los componentes de la electrónica de control y sobre los propios brazos manipuladores en sí.

Si bien, debido a inconvenientes externos, la realización del objetivo general propuesto para este trabajo (la realización física de la placa de circuito impreso) no ha sido posible, el estudio realizado sobre esta cuestión ha dado sus frutos. De esta forma, a través del presente informe se han explicado todas las instrucciones necesarias para que se elabore esta placa PCB en el futuro, por lo que de cierta forma el objetivo principal sí ha sido conseguido y estamos satisfechos con el trabajo realizado.

Por otro lado, también estamos satisfechos con los desarrollos de las partes específicas de nuestro trabajo, ya que nos han permitido también adquirir muchos conocimientos y destrezas que nos serán útiles en el futuro. Estas destrezas abarcan tanto el ámbito de la programación, como el ámbito del control de sistemas dinámicos y el desarrollo y fabricación de hardware para el campo de la robótica.

En definitiva, el desarrollo del presente trabajo nos ha permitido acercarnos y aprender un poco más acerca de la realidad del trabajo de un ingeniero técnico industrial.

12. Conclusions

The conclusions that have been reached through the development of this End of Grade Project are very diverse. Through the course of this work, we have developed a big amount of technical knowledge that has a great importance for the performance of a future industrial engineer.

Through this work, they have learned how to use a professional printed circuit design program such as Multisim 11.0 and how to solve all the problems that the design of a PCB entails. In this way, we have learned how to manufacture a printed circuit board that performs the functions of the control electronics of a manipulator arm, but we have also learned how to design complex printed circuit boards in general.

In addition, other tangential knowledge that has been learn has been related to how to plan a project and how to solve problems that appear during the development of a project. Besides, we have learnt about the study and understanding of technical documentation, because we have done a lot of researches about the components of the electronic of control and about the manipulator arms.

Although, due to external inconveniences, the realization of the general objective proposed for this work (the physical realization of the printed circuit board) has not been possible, the study that we have carried out about this topic has been correct. Because of it, , all the necessary instructions for making this PCB board in the future have been explained, so we are satisfied with the work we have done.

On the other hand, we are also satisfied with the developments of the specific parts of our work. It has also allowed us to acquire many knowledge and skills that will be useful to us in the future. This knowledge is about programming, the dynamic control and the field of the robotic.

Finally, the development of this work has allowed us to learn a little more about the reality of the work of an industrial engineer.

13. Presupuesto

En esta sección del proyecto se elaborará un presupuesto ficticio suponiendo que se quiera elaborar la placa PCB y soldar y unir todos los componentes para fabricar el diseño de la electrónica de control que se detalla en el proyecto.

Los componentes tienen los precios aproximados que tienen al ser comprados en el mercado (algunos de ellos se compran por lotes y no se puede conseguir una sóla unidad del producto). El número de horas de la mano de obra se ha estimado a partir del número de horas que figura en la guía docente que se debe invertir en la realización del Trabajo de Fin de Grado (270 horas no presenciales).

Material/Componente	Cantidad	Precio unidad	Precio total/material
Placa de cobre para PCB	4	1,79 €	7,16 €
Arduino MEGA 2560	1	15,98 €	15,98 €
Módulos L298N	3	4,95 €	14,85 €
LS7166	6	7,15 €	42,90 €
PCA9535	6	1,55 €	9,30 €
HDR1x2	6	0,24 €	1,44 €
HDR1x3	8	0,26 €	2,08 €
Conector D50	1	3,41 €	3,41 €
Resistencias (paquete de resistencias)	1	10,99 €	10,99 €
Pulsador SPDT	1	1,02 €	1,02€
Relay	1	8,24 €	8,24€
Precio total			117,37 €

	Horas	Precio/hora	Precio total
Mano de Obra	270	20€/h	5400€

Concepto		Importe
Proyecto		5508,11 €
Beneficio Industrial	(+6 %)	330,49 €
Total proyecto		5838,60 €
IGIC	(+6,5 %)	379,51 €
Total, IGIC incluido		6218,11 €

14. Bibliografía

Referencias

Las referencias están colocadas en el orden en que van apareciendo por primera vez.

[1] Ana Estévez Pérez, *Implementación de la electrónica para el control dinámico de un manipulador Scorbot-ER Vplus. Interfaz electrónica, lectura de codificadores digitales de posición y cálculos cinemáticos*, trabajo de fin de grado, Julio 2019.

[2] Carlos Javier Siverio Suárez, *Implementación de la electrónica para el control dinámico de un manipulador Scorbot-ER. Desarrollo de la comunicación*, trabajo de fin de grado, Julio 2019.

[3] https://www.prometec.net/l298n/

[4] Apuntes de la asignatura: Diseño y tecnología de circuitos impresos.

[5] https://www.luisllamas.es/que-son-las-capas-de-una-pcb/

[6] http://www.pcb.electrosoft.cl/04-articulos-circuitos-impresos-desarrollo-sistemas/01 -conceptos-circuitos-impresos/conceptos-circuitos-impresos-pcb.html

[7] https://es.wikipedia.org/wiki/Tecnolog %C3 %ADa_de_agujeros_pasantes

[8] http://microensamble.com/glosario/qfp-component/

[9] http://www.dinel.us.es/ASIGN/CE_2T/pracs/2010-2011/p2/practica8.pdf

[10] https://www.gmtextil.es/INSOLADORA-Ultravioleta-Pcb-unidad-de-Exposicio

[11] http://www.film-uv.com/quimicos/

[12] https://www.forosdeelectronica.com/resources/fabricaci %C3 %B3n-de-circuitos-impresos -pcb.4/

[13] https://www.youtube.com/watch?v=mgrmZq48dEU

[14] https://www.manualslib.com/manual/1346253/Intelitek-Scorbot-Er-9.html

[15] https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/

[16] https://www.hispavila.com/voltimetro-con-arduino/

[17] https://www.prometec.net/timers/

[18] https://www.quantumdev.com/understanding-incremental-encoder-signals/

[19] https://pdf1.alldatasheet.es/datasheet-pdf/view/195024/LSI/LS7166.html

15. Anexos

Código de Carlos Javier Siverio Suárez modificado por Oscar Diaz del codigo ÇOMPhyton"

COMPython § | L298N.cpp | L298N.h | PID_v1.cpp | PID_v1.h | artDirection.h | controllerPID.cpp | controllerPID.h | counter.cpp | coun

```
/*-------------------------------------------
 * Carlos Javier Siverio Suarez. Modificados por Oscar Jesus Diaz de la Fe
 * 9/06/2020
 * Codigo principal control SCORBOT-ER V
-------------------------------------------*/

#include "counter.h"
#include "homeV.h"
#include "motor.h"
#include "speed.h"
#include "controllerPID.h"
#include "artDirection.h"
#include "stopMotor.h"

#include "TimerOne.h" // Libreria que nos permite hacer "interraciones" temporalizadas

#include <Wire.h>

Timer1.initialize(1000000);         // Dispara cada segundo
Timer1.attachInterrupt(Sobretension); // Activa la interrupcion por sobretension de los motores, asociado a Sobretension
pinMode(18, INPUT);
attachInterrupt(digitalPinToInterrupt(18), ParadaZ, RISING);


//Definimos variables de entrada de la COM
String string;
int command = 0;
long int option1, option2, option3, option4, option5;
double r;

void setup(){
  Serial.begin(9600);
  Wire.begin();
  //Inicializamos las funcionalidades de "counter", "homeV" y "ControllerPID"
  completeCounterSetup(art1, art2, art3, art4, art5, art6);
  homeVSetup(microSwitch1, microSwitch2, microSwitch3, microSwitch4, microSwitch5, microSwitch6);
  SetupPID();
}
```

COMPython § | L298N.cpp | L298N.h | PID_v1.cpp | PID_v1.h | artDirectio

```
void loop(){
  while(Serial.available()){
    string = Serial.readStringUntil(':');
    command = atoi(string.c_str());
    if(string != ""){
      string = Serial.readStringUntil(',');
      option1 = atol(string.c_str());
    }
    if(string != ""){
      string = Serial.readStringUntil('!');
      option2 = atol(string.c_str());
    }
    if(string != ""){
      string = Serial.readStringUntil('?');
      option3 = atol(string.c_str());
    }
    if(string != ""){
      string = Serial.readStringUntil(';');
      option4 = atol(string.c_str());
    }
    if(string != ""){
      string = Serial.readStringUntil('&');
      option5 = atol(string.c_str());
    }
  }
```

```
//Comando HOME
if(command == 1){
  //Serial.println("Preparando articulaciones para setup...");
  Serial.println("Homing AXIS 1...");
  homeV(motor1, 85, art1, microSwitch1);
  delay(500);
  Serial.println("Homing AXIS 3...");
  homeV(motor3, 130, art3, microSwitch3);
  delay(1000);
  elbowMotorControlPID(motor3, art3, 8388608);
  delay(500);
  Serial.println("Homing AXIS 2...");
  homeV(motor2, 130, art2, microSwitch2);
  delay(1000);
  shoulderMotorControlPID(motor2, art2, 8388608);
  delay(500);
  Serial.println("Homing ROLL...");
  homeVWristRoll(motor4, motor5, 140, art4, art5, microSwitch5);
  delay(500);
  Serial.println("Homing PITCH...");
  homeVWristPitch(motor4, motor5, 135, art4, art5, microSwitch4);
  delay(500);
  Serial.println("Homing GRIPPER-CLAMP...");
  homeVGripperClamp(motor6, art6);
  delay(500);
  Serial.println("Homing COMPLETE.");
  delay(500);
}

//Comando HERE
else if(command == 2){
  //Mandamos valor cuentas encoder a Python para que lo guarde
  Serial.println(readCounter(art1));
  Serial.println(readCounter(art2));
  Serial.println(readCounter(art3));
  Serial.println(readCounter(art4));
  Serial.println(readCounter(art5));
  Serial.println(readCounter(art6));
}
```

```
//Comando MOVE pos
else if(command == 3){
  //Implementado para el movimiento de 1 solo motor
  //baseMotorControlPID(motor1, art1, option1);
  shoulderMotorControlPID(motor2, art2, option2);
  //elbowMotorControlPID(motor3, art3, option3);
  Serial.println("Se ha alcanzado el punto con exito");
}

//Comando SPEED
else if(command == 4){
  setNewSpeed(motor1, motor2, motor3, motor4, motor5, option1);
}

//Comando EXIT, libera el puerto serie y detiene todos los movimientos
else if(command == 5){
  stopMotors(motor1, motor2, motor3, motor4, motor5, motor6);
}

else {
  //No hacemos nada
}
//Reiniciamos el valor de la variable comando
command = 0;
```

```
void Sobretension(){
  for(int i=0; i<12; i++){                    //Todo lo que ocurre en este for es una lecura de la tension de los distintos puntos de los motore
    float v= (analogRead(i) * 4.98) / 1024.0;
    float v2 = v / (1 / (4 + 1));             //Se genera tambien una transformacion matematica por si desea ver segun la tension en los motores
    if(v2 > 26){
      Serial.println("ERROR SOBRETENSION");
      exit(1);
    }
  }
```

```
void ParadaZ(){
  Serial.println("ERROR Parada Z");
    exit(1);
  }
```

Código de Carlos Javier Siverio Suárez modificado por Oscar Diaz del codigo çounter.cpp”

```
  /*-----------------------------------------
 * Carlos Javier Siverio Suarez. Modificados por Oscar Jesus Diaz de la Fe
 * 10/06/2020
 * Declaracion funciones I2C + Contador
-----------------------------------------*/

#include"Arduino.h"
#include "counter.h"
#include<Wire.h>

//Direcciones de los REGISTROS DE CONTROL
byte outControlRegister = 0x02; //Registro de salida Puerto 0
byte outRegister = 0x03; //Registro de salida Puerto 1
byte inRegister = 0x01;  //Registro de entrada Puerto 1
byte configPort0 = 0x06;  //Registro de configuracion Puerto 0
byte configPort1 = 0x07;  //Registro de configuracion Puerto 1

byte index = 0x16; // Registro salida index pulse

//Variables globales
int long OL = 0;
int long _buff[2];

//Variables de configuracion de puertos
byte output = 0x00;
byte input = 0xff;

//Datos enviados al I2C
byte dataSend = 0x00;

//Declaracion de variables auxiliares de CONTROL
bool WR = LOW;
bool CD = HIGH;
bool CS = LOW;
bool RD = LOW;

//Funcion de escritura para el I2C
void writeFunction(int art, byte outRegister, byte dataSend){
  Wire.beginTransmission(art);
  Wire.write(outRegister);
  Wire.write(dataSend);
  Wire.endTransmission();
}
```

```
//Funcion de lectura para el I2C
long int readFunction(int art, byte inRegister, int i){
  Wire.beginTransmission(art);
  Wire.write(inRegister);
  Wire.endTransmission(false);

  Wire.beginTransmission(art);
  Wire.requestFrom(art, 1);
  while(Wire.available()){
    _buff[i] = Wire.read();
  }
  Wire.endTransmission();

  return _buff[i];
}

//Funcion de configuración del puerto como entrada
void inPortConfig(int art, byte configPort){
  writeFunction(art, configPort, input);
  delayMicroseconds(20);
}

//Funcion de configuración del puerto como salida
void outPortConfig(int art, byte configPort){
  writeFunction(art, configPort, output);
  delayMicroseconds(20);
}

//Funcion CONTROL del contador
void controlCounter(int art, bool WR, bool CD, bool CS, bool RD){
  dataSend = 0;
  dataSend = WR;
  dataSend = (dataSend << 1) + CD;
  dataSend = (dataSend << 1) + CS;
  dataSend = (dataSend << 1) + RD;
  writeFunction(art, outControlRegister, dataSend);
  delayMicroseconds(20);
}
```

```
//Funcion para envíar los registros de control
void flankWR(int art){
  RD = HIGH;
  CD = HIGH;

  delayMicroseconds(20);
  WR = LOW;
  controlCounter(art, WR, CD, CS, RD);
  delayMicroseconds(20);
  WR = HIGH;
  controlCounter(art, WR, CD, CS, RD);
  delayMicroseconds(20);
}

//Funcion para acceder a los registros de datos
void flankPR(int art){
  RD = HIGH;
  CD = LOW;

  delayMicroseconds(20);
  WR = LOW;
  controlCounter(art, WR, CD, CS, RD);
  delayMicroseconds(20);
  WR = HIGH;
  controlCounter(art, WR, CD, CS, RD);
  delayMicroseconds(20);
}
```

```
//Lectura de los contadores
long int readCounter(int art){
  //Re-configuramos el puerto como salida
  outPortConfig(art, configPort1);

  //Reseteamos puntero OL y transferimos el CNTR a OL (00000011)
  dataSend = 0x03;
  writeFunction(art, outRegister, dataSend);
  flankWR(art);

  //Deshabilitamos la salida
  dataSend = 0;
  writeFunction(art, outRegister, dataSend);

  CD = LOW;
  RD = LOW;
  controlCounter(art, WR, CD, CS, RD);

  //Re-configuramos el puerto como entrada
  inPortConfig(art, configPort1);

  //Primera lectura
  _buff[0] = readFunction(art, inRegister, 0);

  RD = HIGH;
  controlCounter(art, WR, CD, CS, RD);
  delayMicroseconds(20);
  RD = LOW;
  controlCounter(art, WR, CD, CS, RD);

  //Segunda lectura
  _buff[1] = readFunction(art, inRegister, 1);

  RD = HIGH;
  controlCounter(art, WR, CD, CS, RD);
  delayMicroseconds(20);
  RD = LOW;
  controlCounter(art, WR, CD, CS, RD);

  //Tercera lectura
  _buff[2] = readFunction(art, inRegister, 2);
```

```
  RD = HIGH;
  controlCounter(art, WR, CD, CS, RD);

  OL = _buff[0];
  OL |= (_buff[1] << 8);
  OL |= (_buff[2] << 16);

  //Mensaje de depuracion - VALOR CONTADORES -
  //Serial.println(OL);

  //Re-configuramos el puerto como salida
  outPortConfig(art, configPort1);

  return OL;
}

//Reseteo de los contadores
void resetCounter(int art){
  byte PR0 = 0x00;
  byte PR1 = 0x00;
  byte PR2 = 0x80;

  //Reseteamos el contador (00000100)
  dataSend = 0x04;
  writeFunction(art, outRegister, dataSend);
  flankWR(art);
  delayMicroseconds(20);

  //--------------------------------------
  //Reseteamos puntero direccion PR (00000001)
  dataSend = 0x01;
  writeFunction(art, outRegister, dataSend);
  flankWR(art);
  delayMicroseconds(20);

  //Envio de los valores de inicio del PR
  writeFunction(art, outRegister, PR0);
  flankPR(art);
  delayMicroseconds(20);

  writeFunction(art, outRegister, PR1);
  flankPR(art);
  delayMicroseconds(20);
```

```
  writeFunction(art, outRegister, PR2);;
  flankPR(art);
  delayMicroseconds(20);
  //--------------------------------------

  //Transferencia de datos del PR al CNTR
  dataSend = B0001000;
  writeFunction(art, outRegister, dataSend);
  flankWR(art);
  delayMicroseconds(20);
}

//Inicializacion de los contadores
void inicializeCounter(int art){
  byte PR0 = 0x00;
  byte PR1 = 0x00;
  byte PR2 = 0x80;

  WR = HIGH;
  RD = HIGH;
  controlCounter(art, WR, CD, CS, RD);

  //Reseteo GENERAL usando MCR (00100000)
  dataSend = 0x20;
  writeFunction(art, outRegister, dataSend);
  flankWR(art);

  //Habilitar entradas A y B usando ICR (01101001)
  dataSend = 0x69;
  writeFunction(art, outRegister, dataSend);
  flankWR(art);
  delayMicroseconds(60);

  //Modo binario usando OCCR (10000000)
  dataSend = 0x80;
  writeFunction(art, outRegister, dataSend);
  flankWR(art);
  delayMicroseconds(60);
```

```
  //Modo cuadratura mediante QR (11000011)
  dataSend = 0xC3;
  writeFunction(art, outRegister, dataSend);
  flankWR(art);
  delayMicroseconds(60);

  //Reseteo CNTR - MCR(00000100)
  dataSend = 0x04;
  writeFunction(art, outRegister, dataSend);
  flankWR(art);
  delayMicroseconds(60);

  //Reseteamos puntero direccion PR (00000001)
  //---------------------------------------
  dataSend = 0x01;
  writeFunction(art, outRegister, dataSend);
  flankWR(art);
  delayMicroseconds(60);

  //Envio de los valores de inicio del PR
  writeFunction(art, outRegister, PR0);
  flankPR(art);
  delayMicroseconds(60);

  writeFunction(art, outRegister, PR1);
  flankPR(art);
  delayMicroseconds(60);

  writeFunction(art, outRegister, PR2);;
  flankPR(art);
  delayMicroseconds(60);
  //---------------------------------------

  //Transferencia de datos del PR al CNTR - MCR (0001000)
  dataSend = 0x08;
  writeFunction(art, outRegister, dataSend);
  flankWR(art);
  delayMicroseconds(60);
}
```

```
void counterSetup(int art){
  //Configuramos los puertos como salida inicialmente
  outPortConfig(art, configPort0);
  outPortConfig(art, configPort1);ç

  outPortConfig(art, !index);

  //Inicializamos variables de control del contador
  controlCounter(art, WR, CD, CS, RD);

  //Inicializamos el contador
  inicializeCounter(art);
}

void completeCounterSetup(int art1, int art2, int art3, int art4, int art5, int art6){
  counterSetup(art1);
  counterSetup(art2);
  counterSetup(art3);
  counterSetup(art4);
  counterSetup(art5);
  counterSetup(art6);
}
```

16. Datasheets

16.1. Datasheet Scorbot-ER IX

SCORBOT-ER 9

User Manual

Catalog #100066 Rev. B

intelitek

SCORBOT-ER 9

Catalog # 100066 Rev. B

March 1996

Safety Warning!
Use the SCORBOT ER-9 with extreme caution.
The SCORBOT ER-9 can be dangerous and can cause severe injury.
Setup up a protective screen or guard rail around the robot to keep people away from its working range.

INTELITEK INC.
444 East Industrial Park Drive
Manchester NH 03109-537
Tel: (603) 625-8600
Fax: (603) 625-2137
Web site www.intelitek.com

Table of Contents

CHAPTER 1 Unpacking and Handling

Unpacking the Robot 1-1
Handling Instructions 1-2
Acceptance Inspection 1-2
Repacking for Shipment 1-3

CHAPTER 2 Specifications

Structure 2-2
Work Envelope 2-3

CHAPTER 3 Safety

Precautions 3-1
Warnings 3-2

CHAPTER 4 Installation

Preparations 4-1
Controller and Computer/Terminal Setup 4-1
Robot Setup 4-1
SCORBOT-ER IX Installation 4-3
Controller Installation 4-3
Robot Installation 4-3
Homing the Robot 4-4
Gripper Installation 4-5
Pneumatic Gripper 4-5
DC Servo Gripper 4-7
Activating the Gripper 4-8

CHAPTER 5 Operating Methods

Software 5-1
ACL 5-1
ATS 5-1
ACLoff-line 5-2
SCORBASE Software 5-2
Teach Pendant 5-2

CHAPTER 6 Drive System

Motors 6-2
DC Motor Structure 6-3
SCORBOT-ER IX Motors 6-4
Harmonic Drive Gears 6-5
Harmonic Drive Gear Ratios 6-6
Axis Gear Ratios 6-7

CHAPTER 7 **Position and Limit Devices**

Encoders 7-1
Encoder Resolution 7-3
End of Travel (Limit) Switches 7-4
Hard Stops 7-5
Home Switches 7-6

CHAPTER 8 **Wiring**

Robot (Power) Cable and Connector 8-2
Encoder Cable and Connector 8-3

CHAPTER 9 **Maintenance**

Daily Operation 9-1
Periodic Inspection 9-2
Troubleshooting 9-2
Messages 9-6

CHAPTER 1

Unpacking and Handling

This chapter contains important instructions for unpacking and inspecting the **SCORBOT-ER IX** robot arm.

☞ *Read this chapter carefully before you unpack the **SCORBOT-ER IX** robot and controller.*

Unpacking the Robot

The robot is packed in expanded foam, as shown in Figure 1-1.

To protect the robot during shipment, a metal plate holds the gripper- mounting flange to the robot base. The plate is fixed to the flange with three bolts and to the base with two bolts. Use a 3mm hex socket wrench to detach these bolts.

Save these bolts and the plate. You will need them should you repack the robot for shipment.

Save the original packing materials and shipping carton. You may need them later for shipment or for storage of the robot.

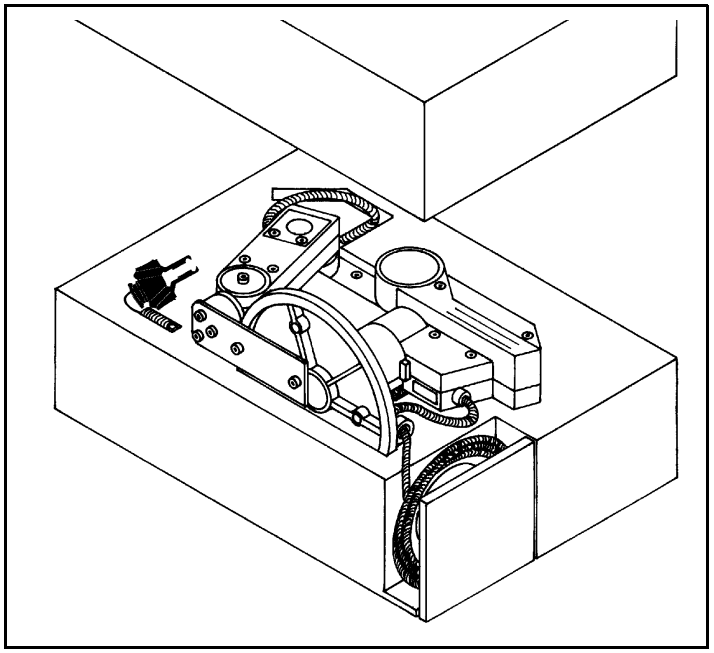


Figure 1-1: SCORBOT-ER IX in Packing

Handling Instructions

The robot arm weighs 38 kilos (83 pounds). Two people are needed to lift or move it.

Lift and carry the robot arm by grasping its body and/or base. Do not lift or carry the robot arm by its upper arm or forearm.

Acceptance Inspection

After removing the robot arm from the shipping carton, examine it for signs of shipping damage. If any damage is evident, do not install or operate the robot. Notify your freight carrier and begin appropriate claims procedures.

The following items are standard components in the **SCORBOT-ER IX** package. Make sure you have received all the items listed on the shipment's packing list. If anything is missing, contact your supplier.

Item	Description
SCORBOT-ER IX Robot Arm	Includes: Cabling with air hoses; Hardware for mounting robot: 3 M8x60 bolts; 3 M8 washers; 3 M8 nuts.
Gripper: 2 options	**Pneumatic Gripper** includes: pneumatic solenoid valve; Hardware for mounting gripper: 6 4Mx8 screws.
	Electric DC Servo Gripper with encoder includes: Hardware for mounting gripper: 4 M4x10 screws.
ACL Controller-B	Includes: Power Cable 100/110/220/240VAC; RS232 Cable; 3 driver cards for 6 axes.
	Optional: Emergency By-Pass Plug (required when TP not connected) Additional driver cards for control of up to 12 axes; Auxiliary multiport RS232 board, cable and connectors.
Teach Pendant: optional	Includes: mounting fixture; connector adapter plug; *Teach Pendant for Controller-B User's Manual*
Software	ATS (Advanced Terminal Software) diskette; includes **ACLoff-line** software
	SCORBASE Level 5 Software diskette
Documentation	***SCORBOT-ER IX*** *User's Manual*
	ACL Controller-B *User's Manual*
	ACL *for Controller-B Reference Guide*
	ATS *for Controller-B Reference Guide*
	ACLoff-line *User's Manual*
	SCORBASE *Level 5 for Controller-B Reference Guide*

Repacking for Shipment

Be sure all parts are back in place before packing the robot.

When repacking the robot for shipping, **bolt the flange and base to the metal plate**. Failure to do so may result in irreversible damage to the arm, particularly to the Harmonic Drive transmissions. Also be sure to secure the cables around the foam spool.

The robot should be repacked in its original packaging for transport.

If the original carton is not available, wrap the robot in plastic or heavy paper. Put the wrapped robot in a strong cardboard box at least 15 cm (about 6 inches) longer in all three dimensions than the robot. Fill the box equally around the unit with resilient packing material (shredded paper, bubble pack, expanded foam chunks).

Seal the carton with sealing or strapping tape. Do not use cellophane or masking tape.

This page intentionally left blank.

CHAPTER 2

Specifications

The following table gives the specifications of the **SCORBOT-ER IX** robot arm.

Robot Arm Specifications		
Mechanical Structure	Vertical articulated, enclosed casting	
Number of Axes	5 plus gripper	
Axis Movement Axis 1: Base rotation Axis 2: Shoulder rotation Axis 3: Elbow rotation Axis 4: Wrist pitch Axis 5: Wrist roll	Axis Range 270° 145° 210° 196° 737°	Effective Speed 79°/sec 112°/sec 68°/sec 99°/sec 76°/sec 112°/sec 87°/sec 133°/sec 166°/sec
Maximum Operating Radius	691mm (27.2") without gripper	
End Effector: options:	Pneumatic Gripper	
	Electric DC Servo Gripper	
Hard Home	Fixed position on all axes	
Feedback	Incremental optical encoders with index pulse	
Actuators	DC servo motors	
Transmission	Harmonic Drive gears and timing belts	
Maximum Payload	2 kg (4.4 lb.), including gripper	
Position Repeatability	±0.09mm (0.0035")	
Weight	38 kg (83 lb.)	
Ambient Operating Temperature	2°–40°C (36°–104°F)	

Structure

The **SCORBOT-ER IX** is a vertical articulated robot, with five revolute joints. With gripper attached, the robot has six degrees of freedom. This design permits the end effector to be positioned and oriented arbitrarily within a large work space.

Figures 2-1 and 2-2 identify the joints and links of the mechanical arm.

Each joint is driven by a permanent magnet DC motor via a Harmonic Drive gear transmission and timing belt.

The movements of the joints are described in the following table:

Axis No.	Joint Name	Motion	Motor No.
1	Base	Rotates the body.	1
2	Shoulder	Raises and lowers the upper arm.	2
3	Elbow	Raises and lowers the forearm.	3
4	Wrist Pitch	Raises and lowers the end effector.	4
5	Wrist Roll	Rotates the end effector.	5

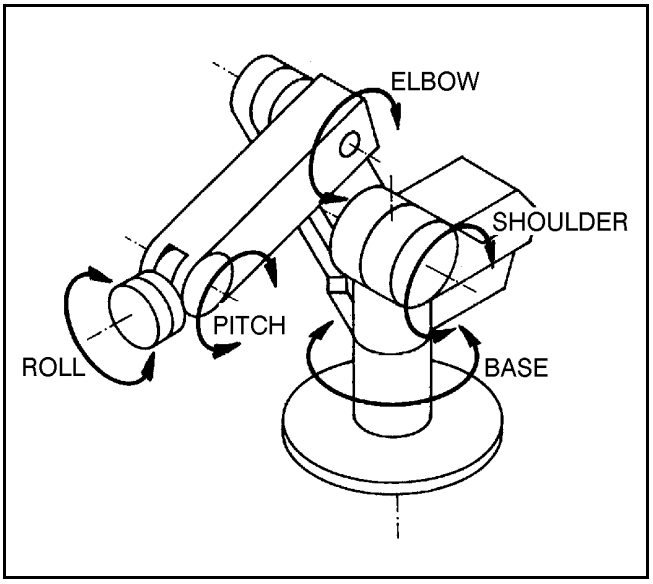


Figure 2-1: SCORBOT-ER IX Joints

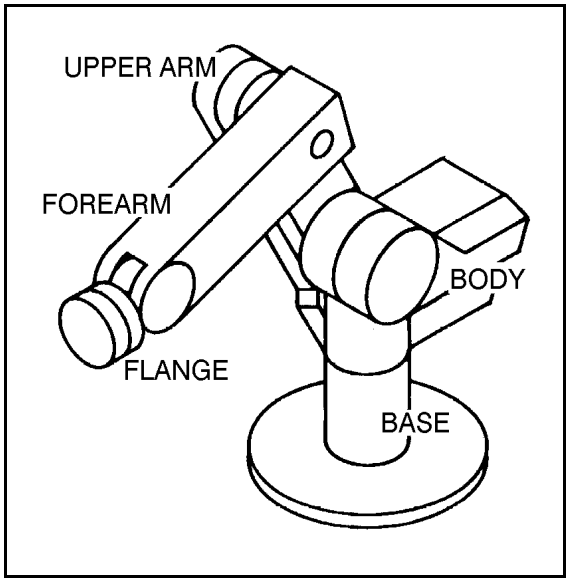


Figure 2-2: SCORBOT-ER IX Links

Work Envelope

The length of the links and the degree of rotation of the joints determine the robot's work envelope. Figure 2-3 shows the dimensions and reach of the **SCORBOT-ER IX**, while Figure 2-4 gives a top view of the robot's work envelope.

The base of the robot is normally fixed to a stationary work surface. It may, however, be attached to a slidebase, resulting in an extended working range.

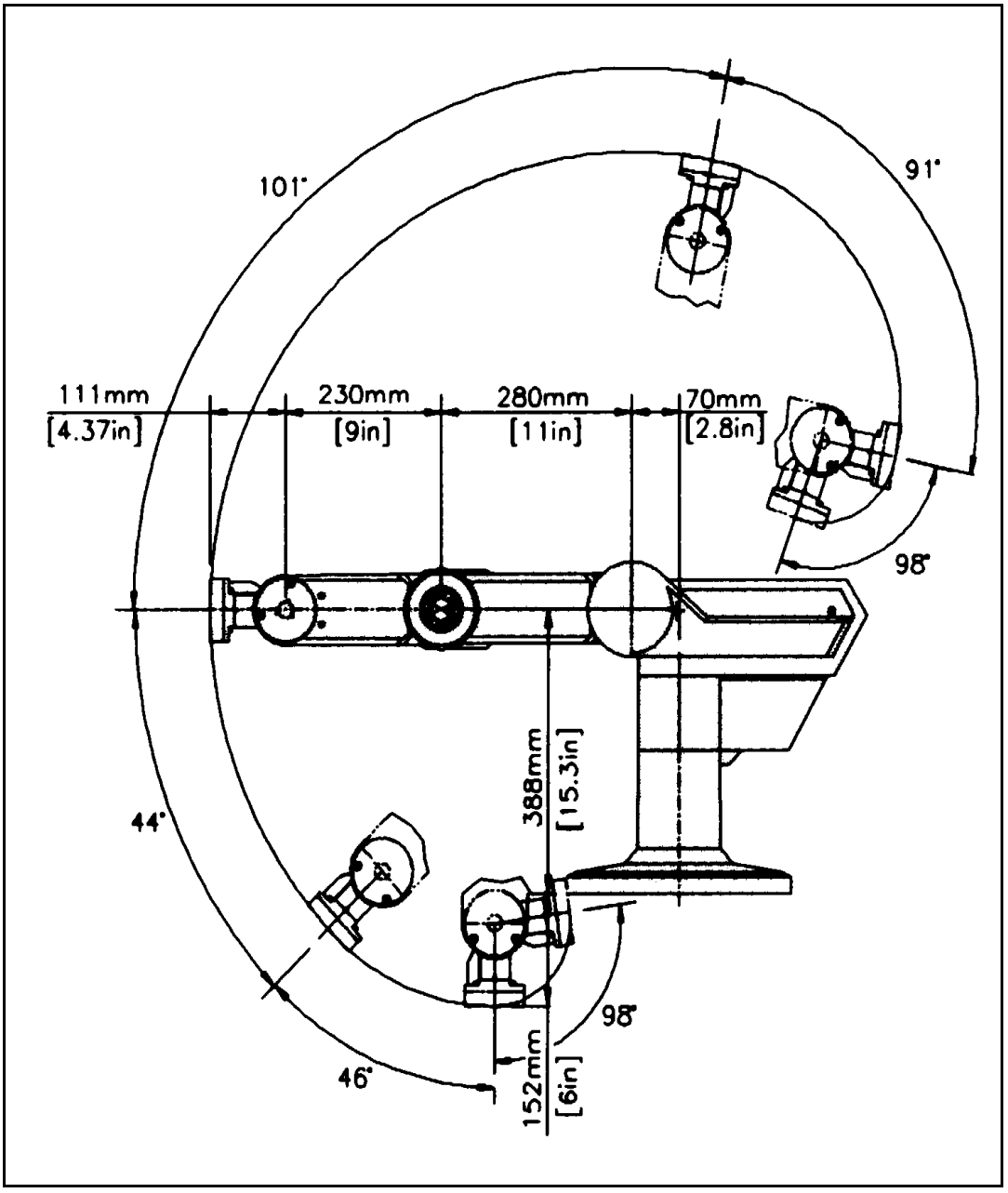


Figure 2-3: Operating Range (Side View)

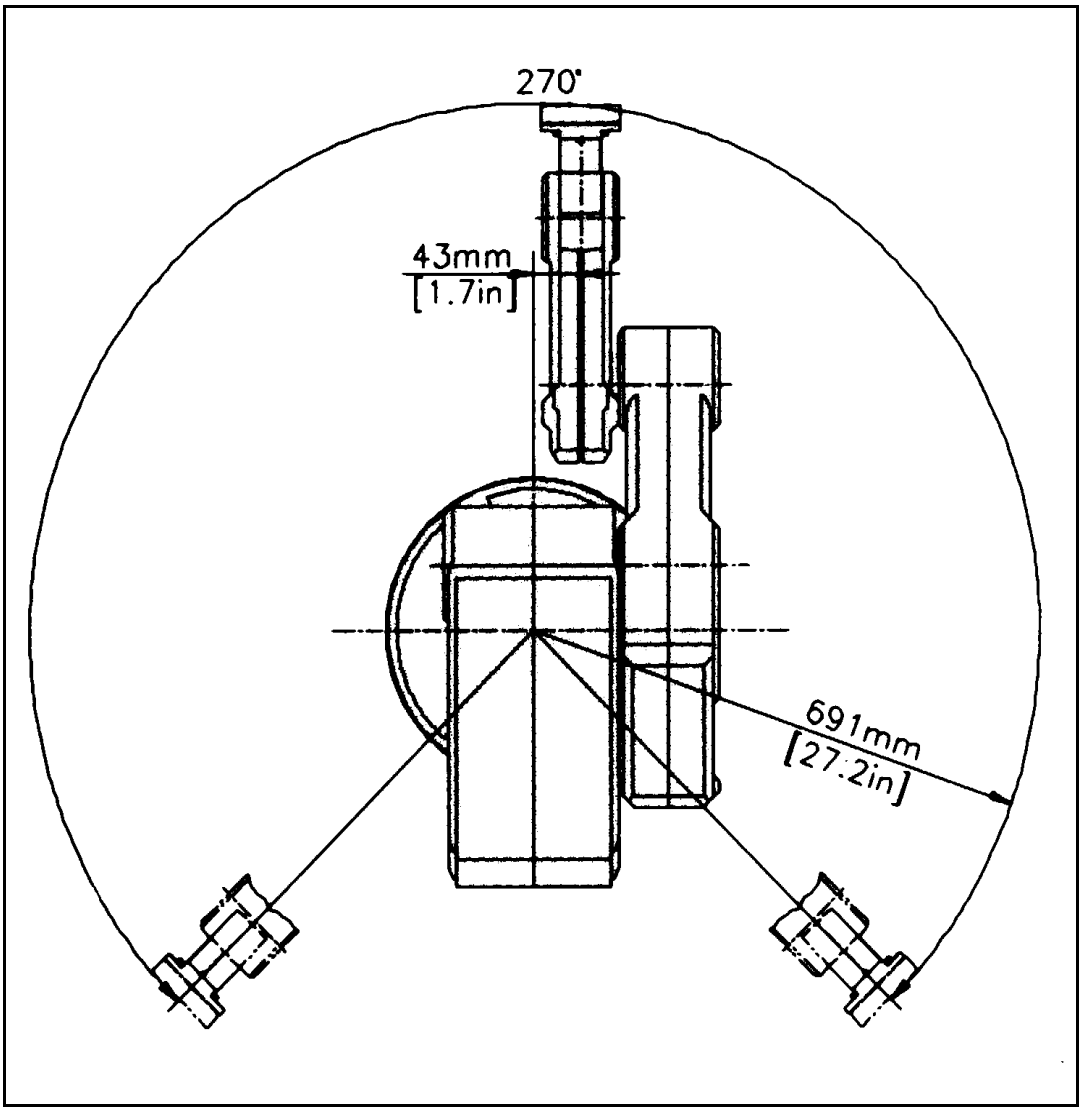


Figure 2-4: Operating Range (Top View)

CHAPTER 3

Safety

The **SCORBOT-ER IX** is a potentially dangerous machine. Safety during operation is of the utmost importance. Use extreme caution when working with the robot.

Precautions

The following chapters of this manual provide complete details for proper installation and operation of the **SCORBOT-ER IX**. The list below summarizes the most important safety measures.

1. Make sure the robot base is properly and securely bolted in place.
2. Make sure the cable from the body to the base can move freely during all movements of the robot's base axis.
3. Make sure both the encoder cable and the robot power cable are properly connected to the controller before it is turned on.
4. Make sure the robot arm has ample space in which to operate freely.
5. Make sure a guardrail or rope has been set up around the **SCORBOT-ER IX** operating area to protect both the operator and bystanders.
6. Do not enter the robot's safety range or touch the robot when the system is in operation.
7. Press the controller's EMERGENCY switch before you enter the robot's operating area.
8. Turn off the controller's POWER switch before you connect any inputs or outputs to the controller.

☞ *To immediately abort all running programs and stop all axes of motion, do any of the following:*

- *press the teach pendant's EMERGENCY button;*
- *use the ACL command **A** <**Enter**>;*
- *press the controller's red EMERGENCY button.*

Warnings

1. Do not operate the **SCORBOT-ER IX** until you have thoroughly studied both this *User's Manual* and the ***ACL Controller-B*** *User's Manual*. Be sure you follow the safety guidelines outlined for both the robot and the controller.
2. Do not install or operate the **SCORBOT-ER IX** under any of the following conditions:
   - Where the ambient temperature drops below or exceeds the specified limits.
   - Where exposed to large amounts of dust, dirt, salt, iron powder, or similar substances.
   - Where subject to vibrations or shocks.
   - Where exposed to direct sunlight.
   - Where subject to chemical, oil or water splashes.
   - Where corrosive or flammable gas is present.
   - Where the power line contains voltage spikes, or near any equipment which generates large electrical noises.
3. Do not abuse the robot arm:
   - Do not operate the robot arm if the encoder cable is not connected to the controller.
   - Do not overload the robot arm. The combined weight of the workload and gripper may not exceed 2kg (4.4 lb.). It is recommended that the workload be grasped at its center of gravity.
   - Do not use physical force to move or stop any part of the robot arm.
   - Do not drive the robot arm into any object or physical obstacle.
   - Do not leave a loaded arm extended for more than a few minutes.
   - Do not leave any of the axes under mechanical strain for any length of time. Especially, do not leave the gripper grasping an object indefinitely.

CHAPTER 4

Installation

Preparations

Before you make any cable connections, set up the system components according to the following "Preparation" instructions.

Controller and Computer/Terminal Setup

Place the controller and computer at a safe distance from the robot—well outside the robot's safety range.

Make sure the setup complies with the guidelines defined in the chapter, "Safety," in the ***ACL Controller-B*** *User's Manual*.

Robot Setup

Refer to Figures 4-1, 4-2 and 4-3.

1. Set up the **SCORBOT-ER IX** on a sturdy surface with at least one meter of free space all around the robot.
2. Note that the robot cable clamp is located at the midpoint of the robot's horizontal range. Using this midpoint as a reference, set up the robot so that it faces in the proper direction—towards the application or machine it will serve.
3. Fasten the base of the robot to the work surface with three sets of M8 bolt, washer and nut.

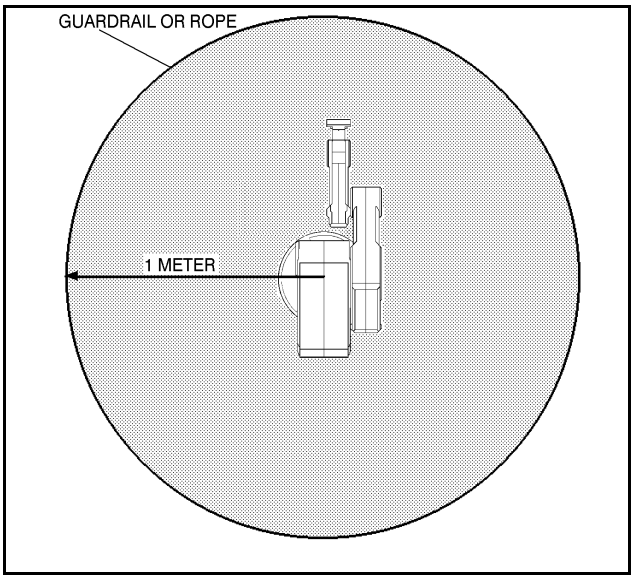


Figure 4-1: Robot Safety Range

Make sure the robot is securely bolted in place. Otherwise the robot could become unbalanced and topple over while in motion.

4. Grasp the robot body and turn the robot to each extreme of its base axis.

☞ *Make sure the segment of cable from the body to the base is not obstructed, and/or cannot become caught under a corner of the robot's platform or work surface during all movements of the base axis.*

Make sure the robot is mounted on a surface large enough to provide support for this segment of the robot cable during all movements of the base axis.

5. Set up a guardrail or rope around the **SCORBOT-ER IX** operating area to protect both the operator and bystanders.

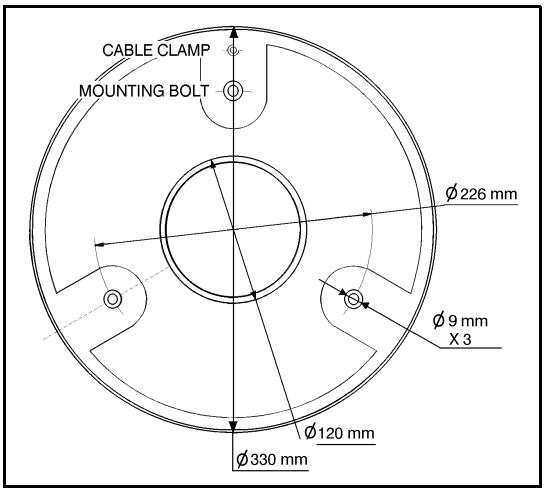


Figure 4-2: Robot Base Layout

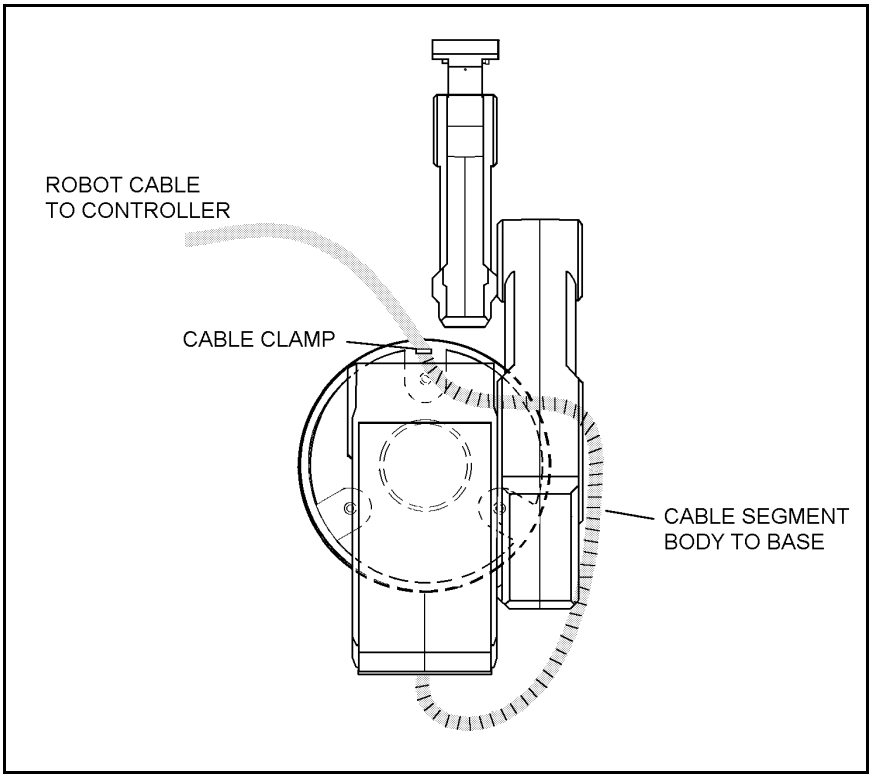


Figure 4-3: Robot Setup

SCORBOT-ER IX Installation

Controller Installation

Perform the installation procedures detailed in the following sections of Chapter 2, "Installation," in the *Controller-B User's Manual*:

- **Computer/Terminal–Controller Installation**
- **Power On**
- **Controller Configuration**

☞ When the Peripheral Setup screen appears at the end of the controller configuration, select **Gripper Connection: None**. (You will change this setting after the gripper is installed.) Refer to the section, "Peripheral Devices and Equipment--Robot Gripper," in the *Controller-B User's Manual*.

Robot Installation

☞ *Before you begin, make sure the controller POWER switch is turned off.*

The robot cable has a number of connectors. Connect them to the controller according to following three steps. Refer to Figure 4-4.

1. Connect the green/yellow wire to the Safety Ground:
   Unscrew and remove the ground nut and washer from the Safety Ground stud. Place the ground wire terminal onto the stud, then replace and tighten the washer and nut.
2. Plug the the D37 connector into the Robot Encoders port.
   Tighten the retaining screws on the connector.
3. Plug the 19-pin round connector into the Robot Power port.

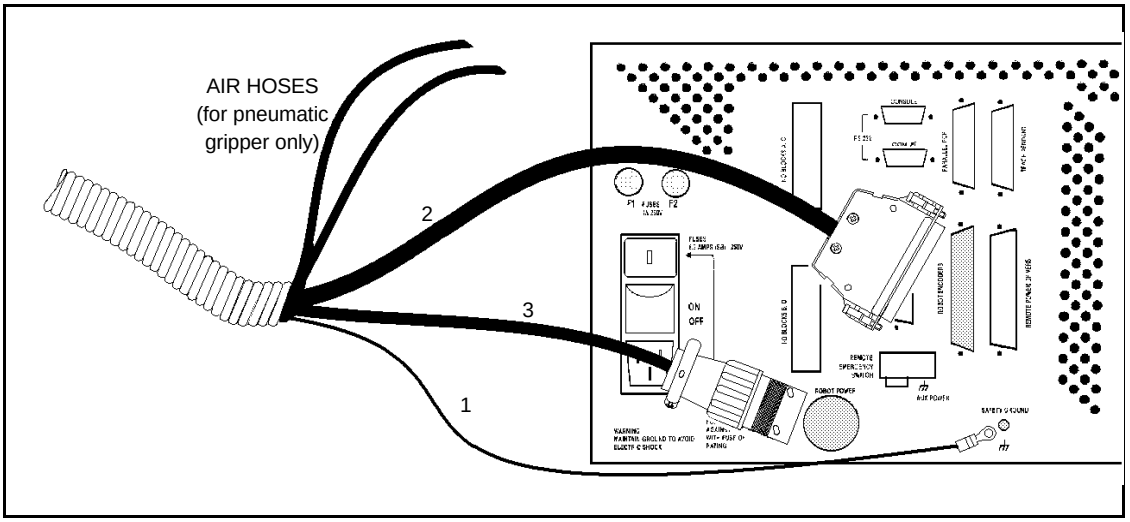


Figure 4-4: Robot—Controller Cable Connections

Note: When disconnecting the robot from the controller, do it in the reverse order; that is:

- Disconnect the 19-pin round Robot Power connector.
- Disconnect the 37-pin Encoders connector.
- Disconnect the ground wires.

Homing the Robot

After you have completed the robot installation, execute the robot's Home routine, as described below.

☞ *The robot must be homed before you mount the gripper.*

☞ *Before you begin the homing procedure, make sure the robot has ample space in which to move freely and extend its arm.*

1. Turn on the controller. Turn on the computer.
2. From the ATS diskette or directory, activate the ATS software. Type:

   `ats  <Enter>`

   If the controller is connected to computer port COM2, type:

   `ats /c2`
3. When the ATS screen and > prompt appear, you may proceed.
4. Give the ACL command to home the robot. Type:

   `home  <Enter>`

   The monitor will display:

   ```
   WAIT!! HOMING...
   ```

   During the Home procedure, the robot joints move and search for their home positions in the following sequence: shoulder, elbow, pitch, roll, base.

   If home is found, a message is displayed:

   ```
   HOMING COMPLETE (ROBOT)
   ```

   If the HOME process is not completed, an error message identifying the failure is displayed. For example:

   ```
   *** HOME FAILURE AXIS 3
   ```

   If the home switch is found, but not the encoder's index pulse, the following message is displayed:

   ```
   * * * INDEX PULSE NOT FOUND AXIS 2
   ```

Gripper Installation

The gripper is attached to the flange at the end of the robot arm whose layout is shown in Figure 4-5.

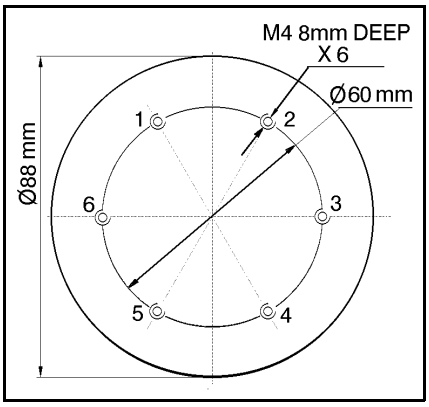


Figure 4-5: Gripper Mounting Flange Layout

Pneumatic Gripper

The pneumatic gripper, shown in Figure 4-6, is controlled by a 5/2 solenoid pneumatic valve which is activated by one of the controller's relay outputs. The valve may be 12VDC or 24VDC and can draw its power from the controller's User Power Supply.

☞ *The robot must be homed before you mount the gripper.*

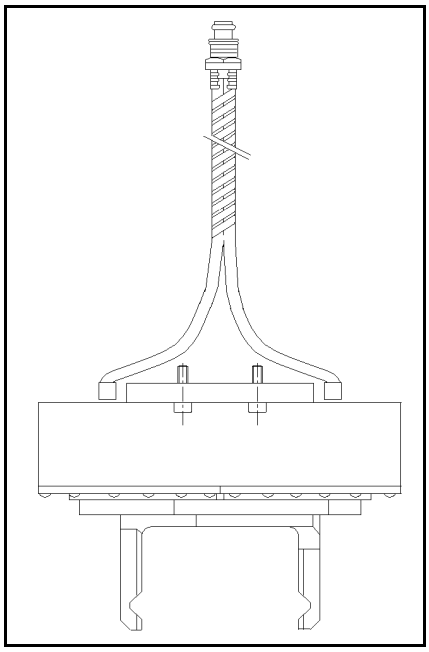


Figure 4-6: Pneumatic Gripper

1. Using a hex wrench and six M4x8 socket screws, attach the gripper to the robot arm flange.
2. Connect the coiled double hose from the gripper to the quick coupling on the robot's forearm, as indicated in Figure 4-7.
3. Refer to Figure 4-8.
   - Connect the two transparent 1/4" O.D. hoses from the robot cable to the CYL ports on the pneumatic valve.
   - Connect a 5 bar/90 PSI air supply to the IN port on the valve.
4. Refer to Figure 4-9.

   Connect the valve to the controller's User Power Supply as follows:

   - Connect the black wire to a common terminal.
   - Connect the red wire to the normally open (NO) terminal of any unused relay output.
5. Connect 12VDC or 24VDC (in accordance with your valve's specification) to the common (C) terminal of the **same** relay output, as shown in Figure 4-9.

6. Attach the valve to the controller or any other metalic surface by means of the valve's magnetic base.

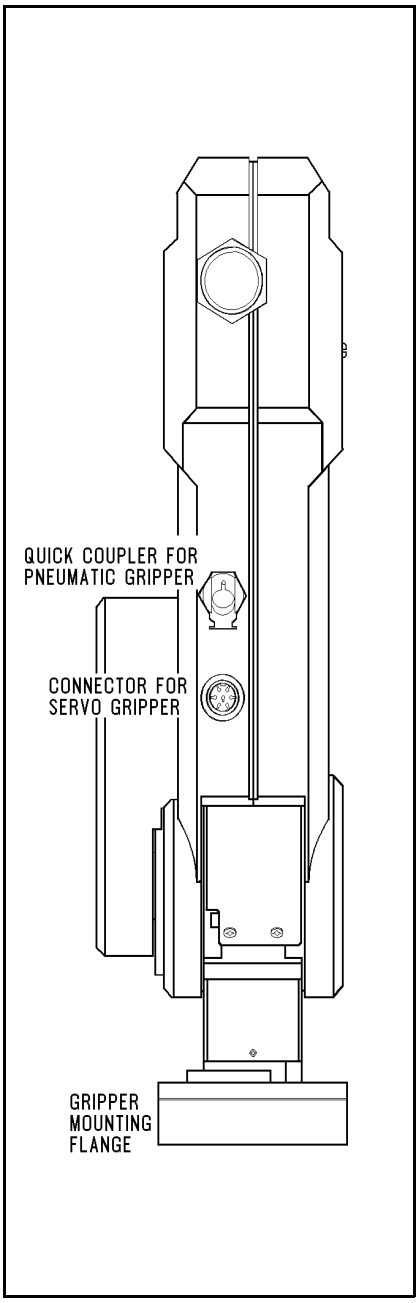


Figure 4-7:
Gripper Connectors

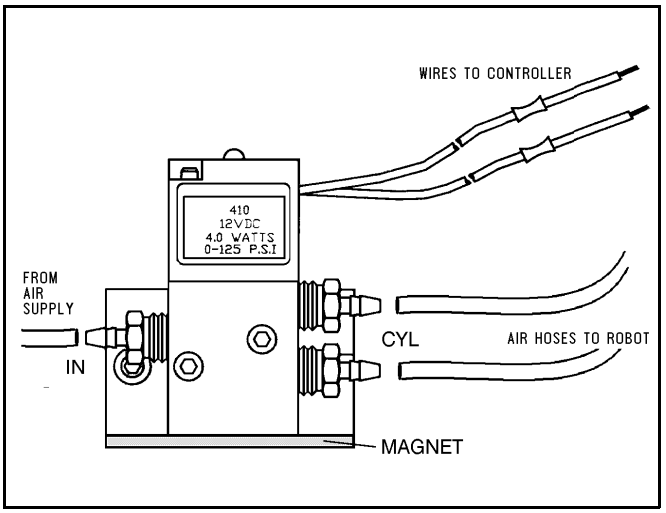


Figure 4-8: Pneumatic Solenoid Valve

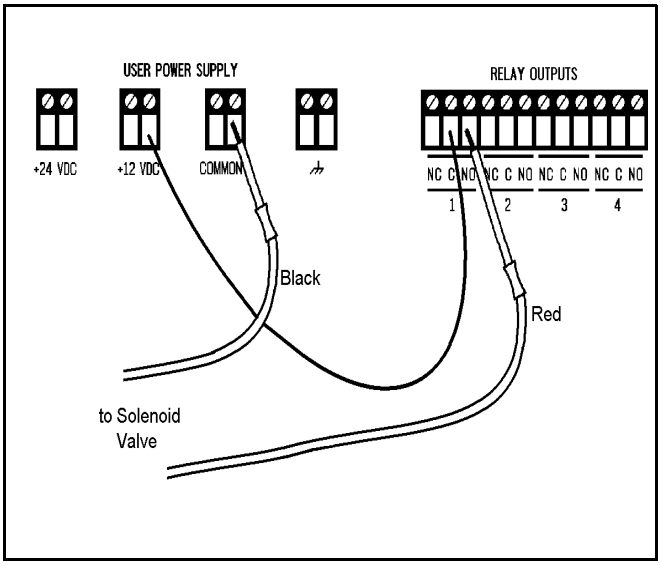


Figure 4-9: Valve—Controller Connections

DC Servo Gripper

The electric DC servo gripper is shown in the inset in Figure 4-10.

☞ *The robot must be homed before you mount the gripper.*

Refer to Figures 4-10 and 4-11.

1. Using a 3 mm hex wrench and four M4x10 socket screws, attach the gripper to the gripper mounting flange at the end of the robot arm.
2. Connect the gripper cable to the electrical connector on the robot arm. Make sure the connector is oriented as shown in Figure 4-10.
3. Make sure the gripper cable is positioned as shown in Figure 4-11.
4. Carefully execute the robot HOME command. Stay close to the teach pendant or controller. If the gripper cable becomes entangled or excessively stretched during the homing, abort the procedure immediately.
5. The gripper has a rotation of ±270°. Do not attempt to move the gripper beyond this limit.
6. At the end of each work session (before turning off the controller), or before homing the robot, make sure the gripper's position is as shown in Figure 4-11.

☞ *Axis 6 is reserved by default controller configuration for a servo gripper. To connect a different device as axis 6, you must change the system configuration by means of the ACL command CONFIG.*

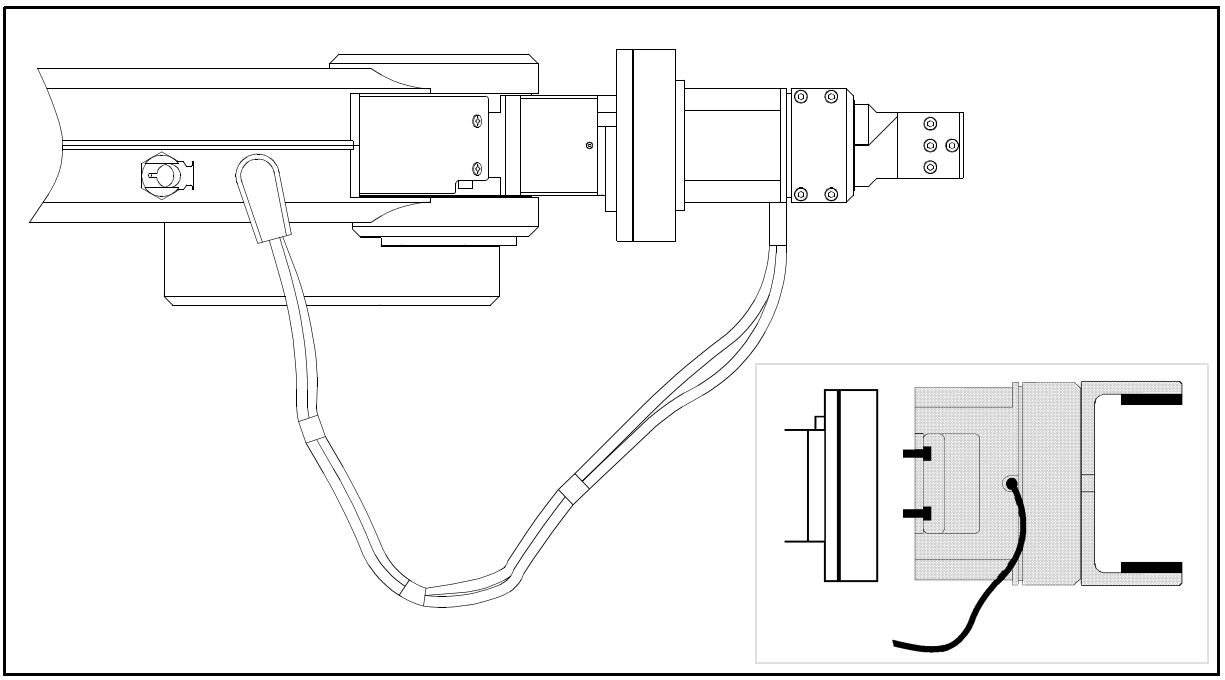


Figure 4-10: Connecting Gripper to SCORBOT-ER IX

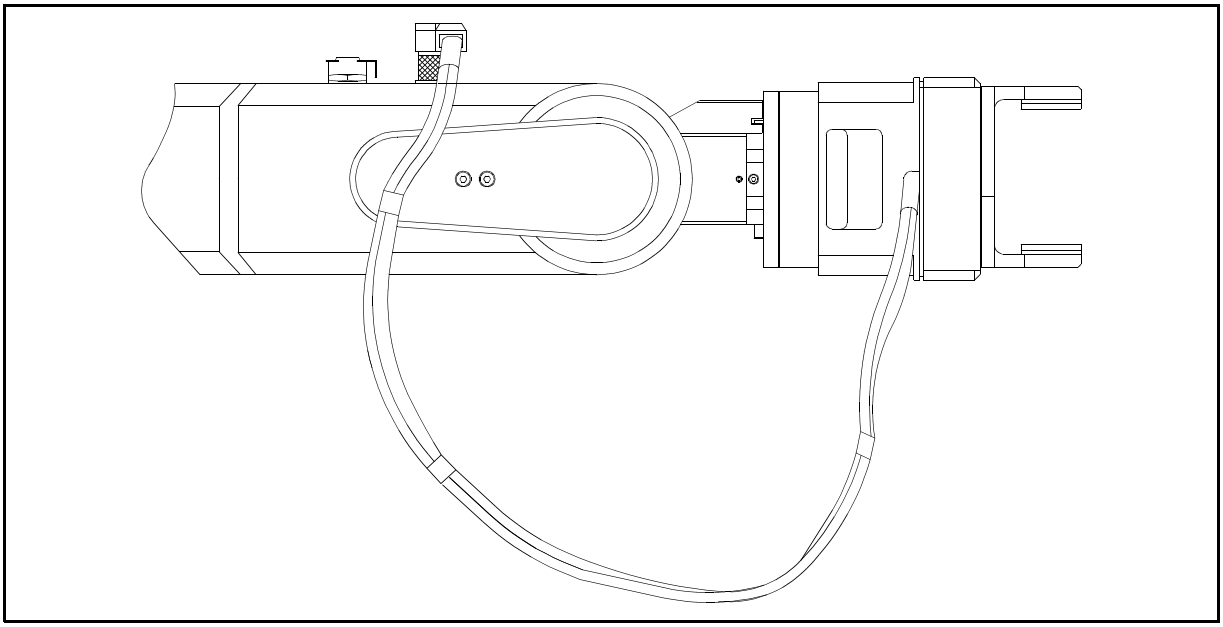


Figure 4-11: Connecting Gripper to SCORBOT-ER IX

Activating the Gripper

1. Activate **ATS**. Press <Ctrl>+F3 to activate the Peripheral Setup screen.
2. Change the robot gripper definition according to the gripper you have installed. Refer to the section, "Peripheral Devices and Equipment--Robot Gripper," in Chapter 2 of the ***ACL Controller-B*** *User's Manual*.
3. Open and close it in order to verify that it is functioning. The following commands work for both the electric and the pneumatic gripper.

PC

Type:

```
open <Enter>
```

The gripper opens.

Type:

```
close <Enter>
```

The gripper closes.

TP

Key in:

Open/Close

The Open/Close key toggles the gripper between its open and closed states. programs you have just written.

CHAPTER 5

Operating Methods

The **SCORBOT-ER IX** robot can be programmed and operated in a number of ways.

The ***ACL Controller-B User's Manual*** includes two chapters which guide you through the basic commands for operating and programming the robot.

Software

ACL

ACL, Advanced Control Language, is an advanced, multi-tasking robotic programming language developed by Eshed Robotec. **ACL** is programmed onto a set of EPROMs within **Controller-B**, and can be accessed from any standard terminal or PC by means of an RS232 communication channel.

ACL features include the following:

- Direct user control of robotic axes.
- User programming of robotic system.
- Input/output data control.
- Simultaneous and synchronized program execution (full multi-tasking support).
- Simple file management.

The ***ACL Reference Guide for Controller-B*** provides detailed descriptions and examples of the **ACL** commands and functions.

ATS

ATS, Advanced Terminal Software, is the user interface to the **ACL** controller. **ATS** is supplied on diskette and operates on any PC. The software is a terminal emulator which enables access to the **ACL** environment from a PC host computer.

ATS features include the following:

- Short-form controller configuration.
- Definition of peripheral devices.
- Short-cut keys for command entry.
- Program editor.
- Backup manager.
- Print manager.

The ***ATS Reference Guide for Controller-B*** is a complete guide to **ATS**.

ACLoff-line

ACLoff-line is a preprocessor software utility, which lets you access and use your own text editor to create and edit **ACL** programs even when the controller is not connected or not communicating with your computer.

After communication is established, the **Downloader** utility lets you transfer your program to the controller. The Downloader detects the preprocessor directives, and replaces them with a string or block of ACL program code.

ACLoff-line also enables activation of **ATS**, Advanced Terminal Software, for on-line programming and system operation.

ACLoff-line is described fully in the ***ACLoff-line*** *User's Manual*.

SCORBASE Software

SCORBASE Level 5 is a robot control software package which is supplied on diskette with the controller. Its menu-driven structure and off-line capabilities facilitate robotic programming and operation.

SCORBASE runs on any PC system and communicates with **ACL**, the controller's internal language, by means of an RS232 channel.

The ***SCORBASE Level 5 for Controller-B Reference Guide*** provides detailed descriptions and examples of the **SCORBASE** commands.

Teach Pendant

The teach pendant is a hand-held terminal which is used for controlling the **SCORBOT-ER IX** robot and peripheral equipment. The teach pendant is most practical for moving the axes, recording positions, sending the axes to recorded positions and activating programs. Other functions can also be executed from the teach pendant.

The ***Teach Pendant for Controller-B User's Manual*** fully describes the various elements and functions of the teach pendant.

CHAPTER 6

Drive System

The three main elements of the **SCORBOT-ER IX** drive system are shown in Figure 6-1:

- DC electrical motor
- Harmonic Drive gear
- Timing belt and pulleys

Figure 6-1 shows the drive system for axes 1 through 4 of the **SCORBOT ER-IX**. The roll axis (axis 5) transmission does not contain the pulleys and timing belt; only a Harmonic Drive is used.

☞ *Note that the illustrations of components shown in this chapter are for descriptive purposes, and may not be the actual components used in the **SCORBOT-ER IX**.*

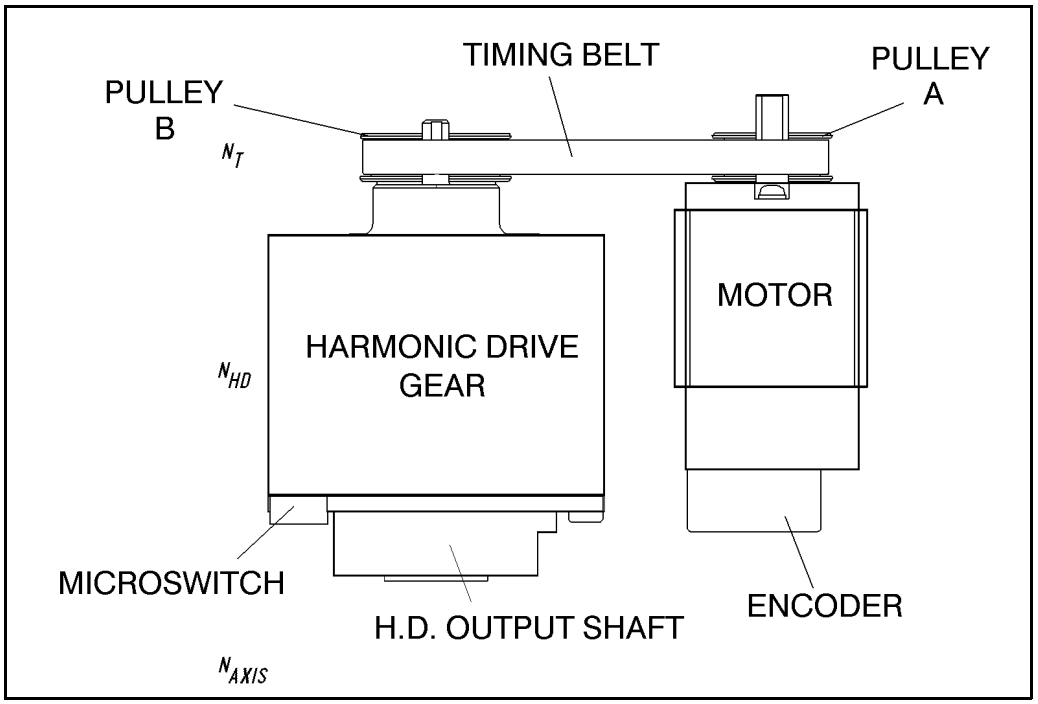


Figure 6-1: The SCORBOT-ER IX Drive System

Motors

The **SCORBOT-ER IX** robot arm is driven by DC electric motors. These actuators converts signals from the controller (electric power) into rotations of the motor shaft (mechanical power).

A robot arm such as the **SCORBOT-ER IX** imposes severe requirements on the actuators, such as the following:

- The robot motor must rotate at different speeds, and with a high degree of accuracy. For example, if the robot is to be used for a spray painting application, it must be able to accurately follow the defined path at the specified speed.
- The robot motor must allow fine speed regulation so that the robot will accelerate and decelerate as required by the application.
- The robot motor must supply large torques throughout its speed range and also when the joint is stationary.
- The robot motor must be able to stop extremely quickly without overshooting the target position, and perform rapid changes in direction.
- Since mounting motors on the robot arm adds to the robot's weight and inertia, the robot motors must be light and compact, yet powerful. As shown in Figure 6-2, the motors of the **SCORBOT-ER IX** are located on the axes they drive, with a two-stage (axes 1–4) or one-stage (axis 5) transmission.

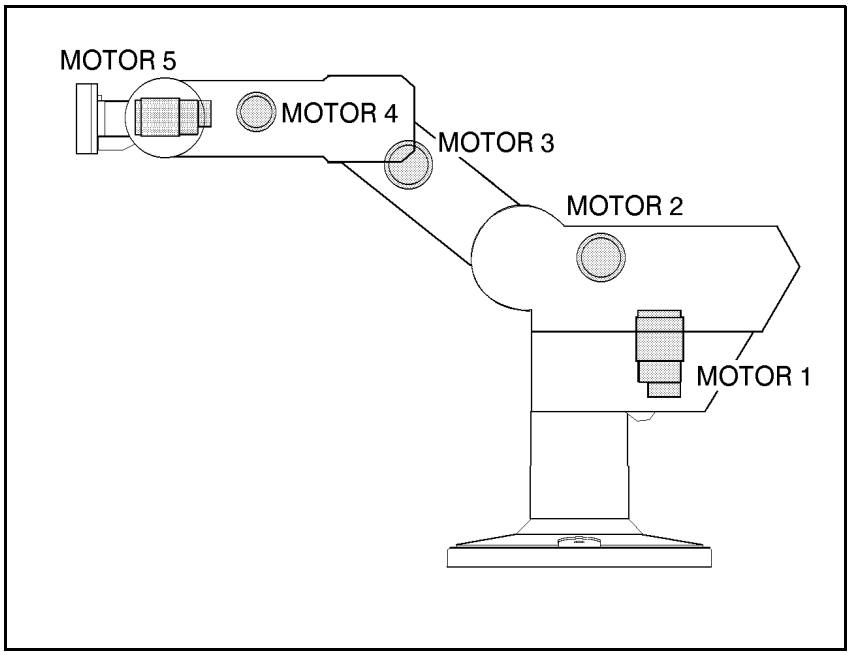


Figure 6-2: Motor Locations in SCORBOT-ER IX

DC Motor Structure

The principles of operation of electrical motors in general, and DC motors in particular, are based on an electrical current flowing through a conductor situated within a magnetic field. This situation creates a force which acts on the conductor.

Figure 6-3 shows the basic structure and components of a DC motor comparable to the structure of the motors used in the **SCORBOT-ER IX**. This motors has three main components:

- ***Stator***: This is a static component which creates the magnetic field. The stator may be a permanent magnet, or an electromagnet consisting of a coil wound around thin iron plates.
- ***Rotor***: This is the component which rotates within the magnetic field. The external load is connected to the rotor shaft. The rotor is generally composed of perforated iron plates, and a conducting wire is wound several times around the plates and through the perforations. The two ends of the conductor are connected to the two halves of the commutator, which are connected to the electric current via the brushes.
- ***Brushes***: These connect the rotating commutator to the electric current source.

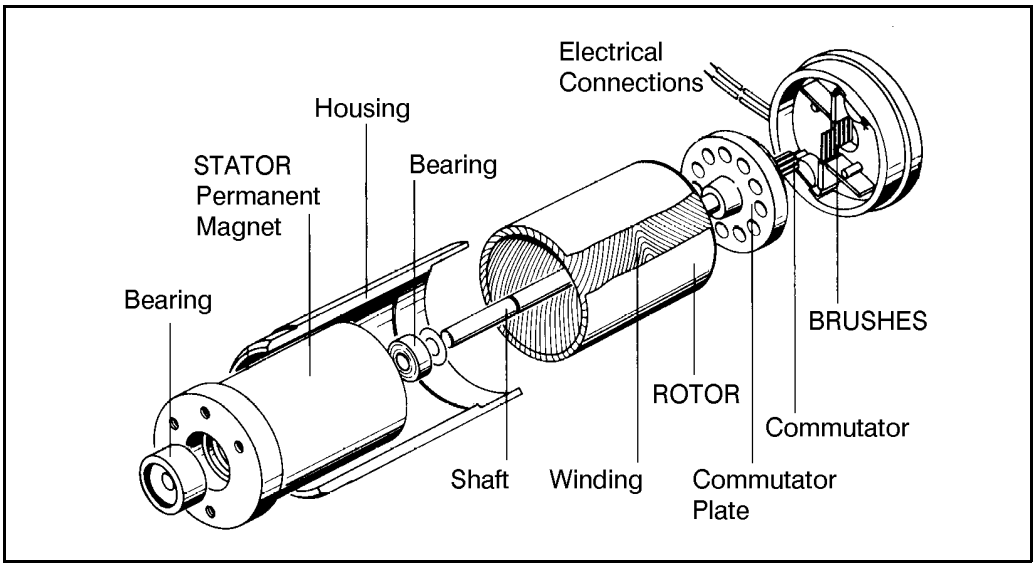


Figure 6-3: Basic Structure of a DC Motor

SCORBOT-ER IX Motors

The **SCORBOT ER-IX** uses permanent magnet DC motors to drive the axes.

Axes 1, 2 and 3 of the **SCORBOT ER-IX** are powered by the motor shown in Figure 6-4. Axes 4 and 5 are powered by the motor shown in Figure 6-5.

These motors are able to move at extremely high rates of revolution, to move loads with high torques, and (with encoder attached) to achieve a very high resolution.

Motor Specifications		
	Motor Axes 1, 2, 3	Motor Axes 4, 5
Peak Rated Torque	143 oz·in	27.8 oz·in
Rated Torque	32 oz·in	12.5 oz·in
Maximum Operating Speed	4000 rpm	4500 rpm
Weight	1.29 k / 2.84 lb	0.28 k / 0.62 lb

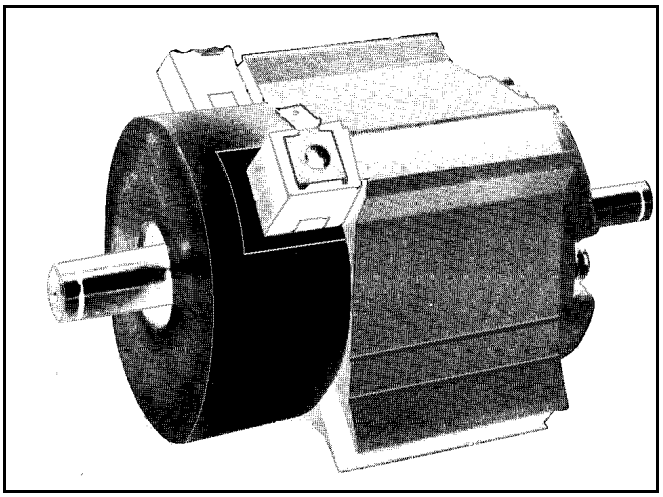


Figure 6-4: Motor on Axes 1, 2 and 3

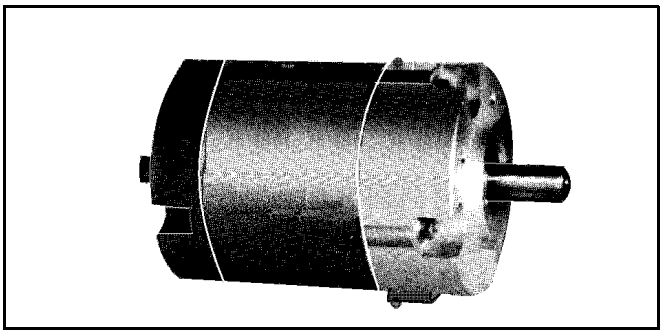


Figure 6-5: Motor on Axes 4 and 5

Harmonic Drive Gears

The Harmonic Drive transmission used in the **SCORBOT-ER IX**, shown in Figure 6-6, offers a very high gear ratio.

The Harmonic Drive gears used in the **SCORBOT-ER IX** have four main components:

- ***Circular spline***:
  a solid steel ring, with internal gear teeth, usually fixed to the robot link.
- ***Wave generator***:
  a slightly elliptical rigid disk, which is connected to the input shaft, with a ball bearing mounted on the outer side of the disk.
- ***Flexspline***:
  a flexible, thin-walled cylinder, with external gear teeth, usually connected to the output shaft.
- ***Dynamic spline***: a solid steel cylinder, with internal gear teeth.

The external gear teeth on the flexspline are almost the same size as the internal gear teeth on the circular spline except there are two more teeth on the circular spline, and the teeth only mesh when the wave generator pushes the flexspline outwards.

Because the wave generator is elliptical, the flexspline is pushed out in two places. As the motor rotates the input shaft, the wave generator rotates and the location of meshing teeth rotates with it. However, because there are two less teeth on the flexspline, it has to rotate backwards slightly as the wave generator rotates forwards. For each complete rotation of the input shaft, the flexspline moves backwards by two teeth. Figures 6-7 and 6-8 show the different steps in this process.

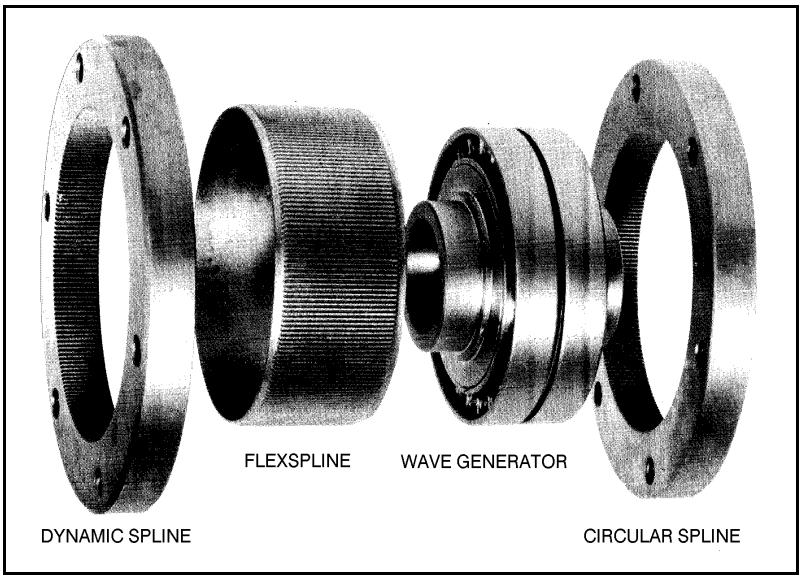


Figure 6-6: Harmonic Drive Structure

Harmonic Drive Gear Ratios

As in all gears, the gear ratio of the Harmonic Drive is the ratio of the input speed to the output speed. If the number of teeth on the flexspline is N_f, then for every revolution of the input shaft, the output shaft rotates by $2/N_f$ of a revolution (that is, two teeth out of N_f teeth). Hence:

$$HD\ gear\ ratio\ =\ \frac{1}{\left(\frac{2}{N_f}\right)}=\frac{N_f}{2}$$

The Harmonic Drive gear ratios for each of the **SCORBOT-ER IX** axes are as follows:

Axis 1 — 161:1
Axis 2 — 160:1
Axis 3 — 160:1
Axis 4 — 100:1
Axis 5 — 100:1

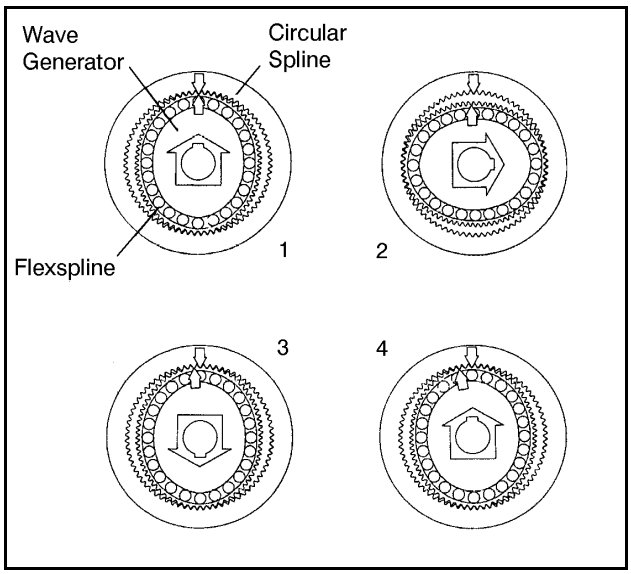


Figure 6-7: Operation of the Harmonic Drive

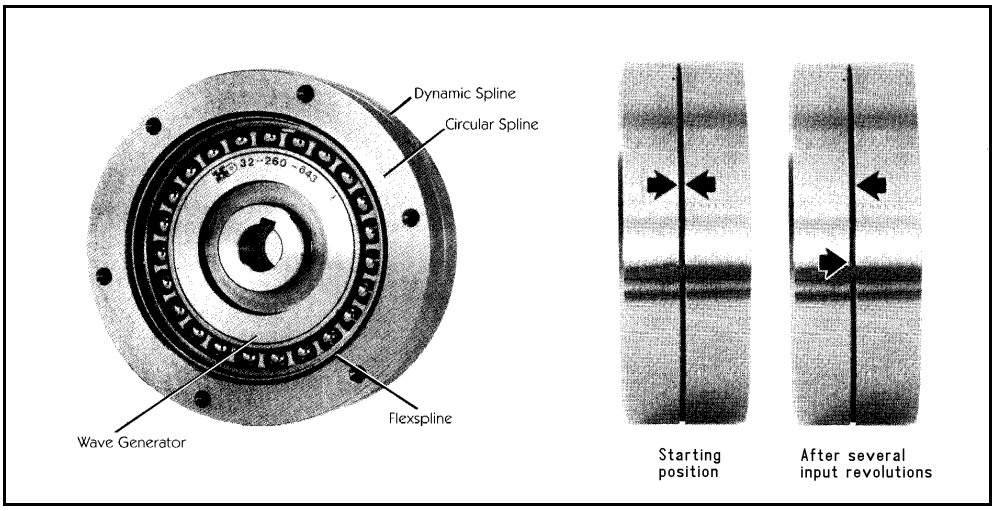


Figure 6-8: Operation of the Harmonic Drive

Axis Gear Ratios

Referring again to Figure 6-1, the transmission of axes 1 through 4 consists of two stages: the timing belt drive, and the Harmonic Drive.

The overall gear ratio of the output shaft which moves the axis is therefore expressed as:

$$N_T \times N_{HD} = N_{AXIS}$$

Where:

N_T is the belt drive ratio (that is, the radii ratio): $\frac{Pulley\ B}{Pulley\ A}$

N_{HD} is the Harmonic drive ratio, as described above.

N_{AXIS} is the overall gear ratio of the axis.

SCORBOT-ER IX Gear Ratios			
	N_T	N_{HD}	N_{AXIS}
Axis 1	1.33 : 1	161 : 1	214.13 : 1
Axis 2	1.52 : 1	160 : 1	243.8 : 1
Axis 3	1.33 : 1	160 : 1	213.33 : 1
Axis 4	1.8 : 1	100 : 1	180 : 1
Axis 5		100 : 1	100 : 1

Thus, one rotation (360°) of axis 3, for example, requires 213.33 rotations of the motor shaft. The actual movement of the axis, however, is limited by the arm's mechanical structure.

This page intentionally left blank.

CHAPTER 7

Position and Limit Devices

This chapter describes the various elements in the **SCORBOT-ER IX** which play a part in the positioning of the robot arm and the limiting of its motion.

- Encoders
- End of Travel Switches
- Hard Stops
- Home Switches

Encoders

The location and movement of each **SCORBOT-ER IX** axis is measured by an electro-optical encoder attached to the motor which drives the axis. The encoder translates the rotary motion of the motor shaft into a digital signal understood by the controller.

Figure 7-1 shows the encoder mounted on a **SCORBOT-ER IX** motor.

The encoder used on the **SCORBOT-ER IX** contains a single light emitting diode (LED) as its light source. Opposite the LED is a light detector integrated circuit. This IC contains several sets of photodetectors and the circuitry for producing a digital signal. A perforated, rotating disk is located between the emitter and detector IC.

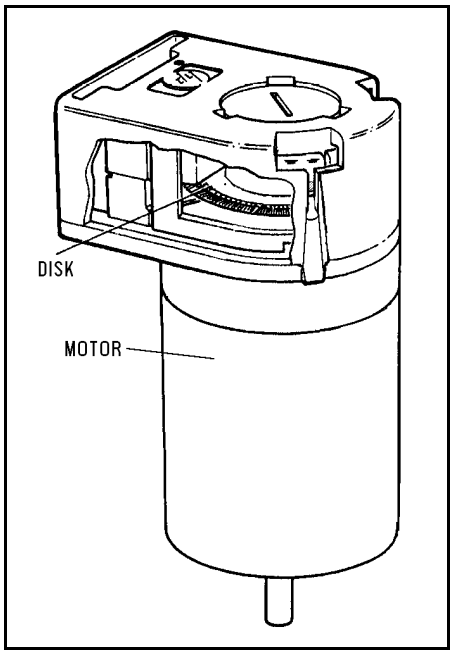


Figure 7-1:
SCORBOT-ER IX Encoder

As the encoder disk rotates between the emitter and detectors, the light beam is interrupted by the pattern of "bars" and "windows" on the disk, resulting in a series of pulses received by the detectors.

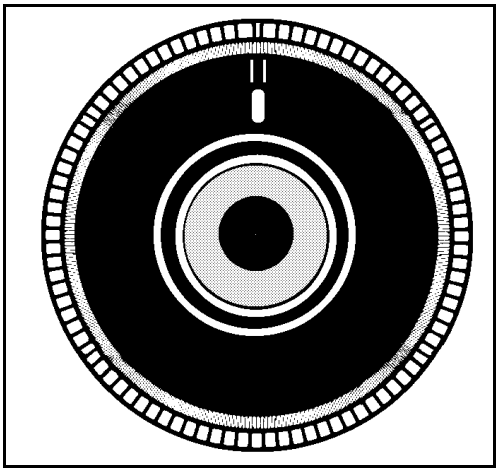


Figure 7-2:
SCORBOT-ER IX Encoder Disk

The **SCORBOT-ER IX** encoders have 512 slots, as shown in Figure 7-2. An additional slot on the encoder disk is used to generate an index pulse (C-pulse) once for each full rotation of the disk. This index pulse serves to determine the home position of the axis.

The photodetectors are arranged so that, alternately, some detect light while others do not. The photodiode outputs are then fed through the signal processing circuitry, resulting in the signals A, $\overline{A}$, B, $\overline{B}$, I and $\overline{I}$, as shown in Figure 7-3.

Comparators receive these signals and produce the final digital outputs for channels A, B and I. The output of channel A is in quadrature with that of channel B (90° out of phase), as shown in Figure 7-4. The final output of channel I is an index pulse.

When the disk rotation is counterclockwise (as viewed from the encoder end of the motor), channel A will lead channel B. When the disk rotation is clockwise, channel B will lead channel A.

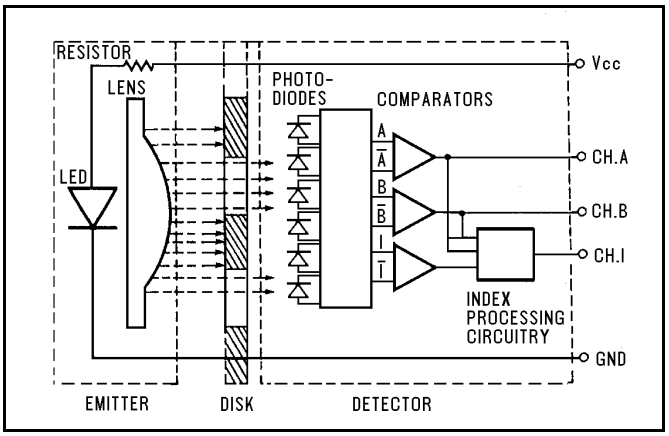


Figure 7-3: Encoder Circuitry

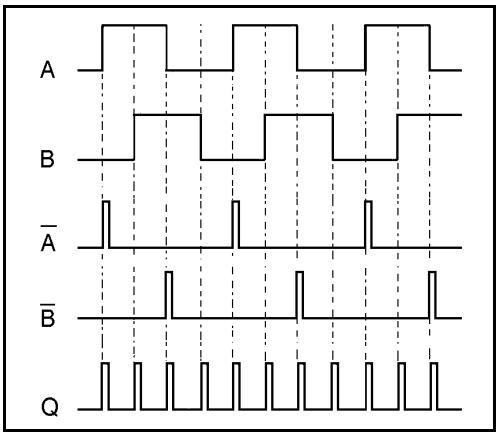


Figure 7-4: Encoder Output Signals

Encoder Resolution

From the quadrature signal the **SCORBOT-ER IX** controller measures four counts for each encoder slot, thus quadrupling the effective resolution of the encoder.

The resolution of the encoder is expressed as:

$$S_E = \frac{360°}{n}$$

Where:

S_E is the resolution of the encoder.

n is the number of counts per encoder revolution.

The encoders used in the **SCORBOT-ER IX** have 512 slots, generating 2048 counts per motor revolution. The encoder resolution is therefore:

$$S_E = \frac{360°}{2048} = .176°$$

When the encoder resolution is divided by the overall gear ratio of the axis, the resolution of the joint is obtained.

Since the encoder is mounted on the motor shaft, and turns along with it, the resolution of the joint is expressed as:

$$S_{JOINT} = \frac{S_E}{N_{AXIS}}$$

Thus, for example, the resolution of joint 3 of the **SCORBOT-ER IX** is therefore as follows:

$$S_{J3} = \frac{0.176°}{213.33} = 0.000825°$$

The resolution is the smallest possible increment which the control system can identify and theoretically control. The accuracy of the axis—that is, the precision with which it is positioned—is affected by such factors as backlash, mechanical flexibility, and control variations.

End of Travel (Limit) Switches

The **SCORBOT-ER IX** uses limit switches to prevent the joints from moving beyond their functional limits. When a control error fails to stop the axis at the end of its working range, the limit switch serves to halt its movement. The switch is part of an electric circuit within the robot arm, independent of the robot controller.

The limit switches used in the **SCORBOT-ER IX** are shown in Figure 7-5.

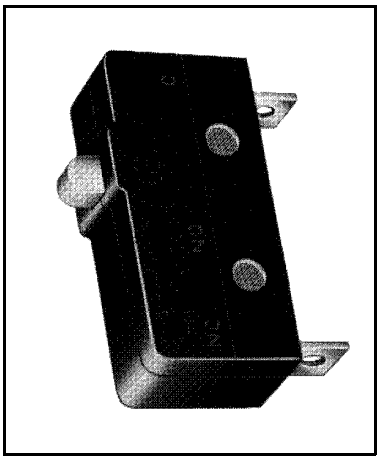


Figure 7-5: SCORBOT-ER IX Limit Switch

Each of axes 1 through 4 has two limit switches: one at each end of the axis' working range.

Axis 5 (roll) has no travel limit switches; it can rotate endlessly. When a gripper is attached to axis 5, its movements are controlled and limited by means of software only (encoder).

The limit switches are mounted on a disk which is attached to the robot's frame. The disk for axis 3 is shown in Figure 7-6.

The output shaft of the Harmonic Drive moves relative to the microswitch disk.

As the joint moves, a cam on the Harmonic Drive output shaft reaches a point at which it forces the actuating button of the limit switch into a position which activates the switch.

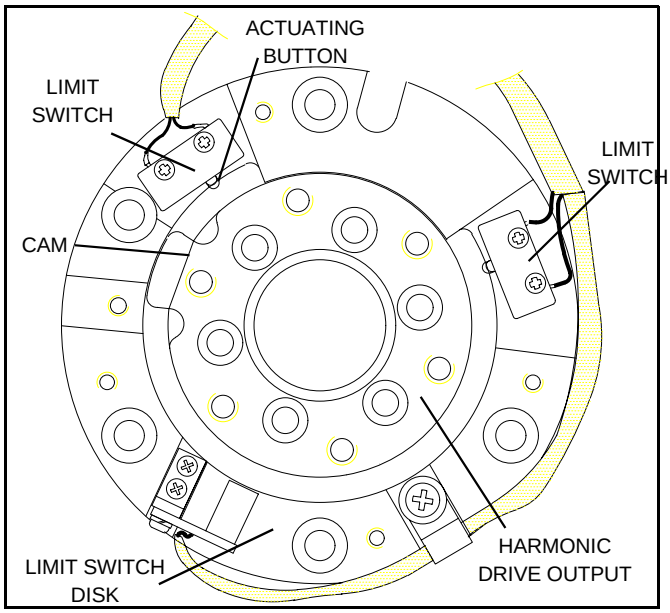


Figure 7-6: Limit Switch Activation

As shown in Figure 7-7A, when limit switch 1 is activated (that is, when the button is depressed), the relay contact opens and the relay is deenergized. The motor cannot move the joint beyond this point. The diode allows the motor to reverse direction, thus permitting the joint to move away from the limit switch.

When the limit switch is activated, it causes a control error, resulting in the activation of COFF (control off mode), and an impact protection message.

CON (control on mode) must be activated and the robot arm must be manually moved (using keyboard or teach pendant) away from the impact condition.

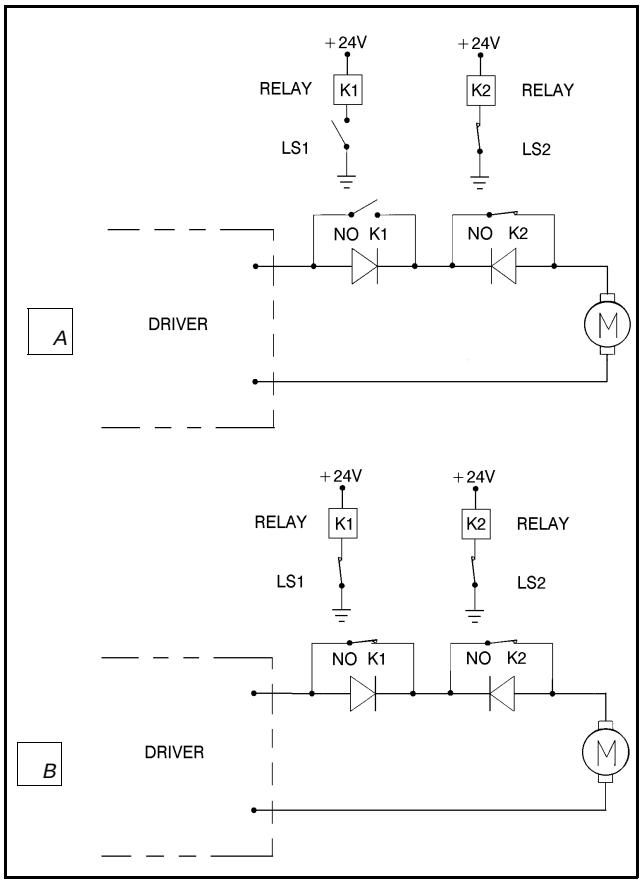


Figure 7-7: Axis Limit Circuit

As long as the axis has not reached one of its limits, the relay contact remains closed, and the diode has no effect on the circuit, as shown in Figure 7-7B. Current can flow in either direction; the motor is thus able to rotate in either direction.

Hard Stops

When the software limits and/or the end of travel switches fail to halt the movement of the robot arm, it is possible that the momentum of the robot arm will drive it until it reaches its mechanical limit.

When the joint reaches this hard stop, the impact protection and thermic protection processes detect an error, thus activating COFF.

CON must be activated and the robot arm must be manually moved away from the impact condition.

Home Switches

The **SCORBOT-ER IX** uses an optical home switch on each axis to identify the fixed reference, or home, position.

The home switch is mounted on the same disk as the end of travel switches, and a "flag" is attached to the Harmonic Drive output shaft, as shown in Figure 7-8.

During the homing procedure, the robot joints are moved, one at a time. Each axis is moved until the flag cuts the beam of light. When that occurs, the optical detector on each joint sends a specific signal to the controller.

Once the home switch location has been detected, the axis motor continues to rotate until its encoder produces an index pulse. The point at which that occurs is the axis home position.

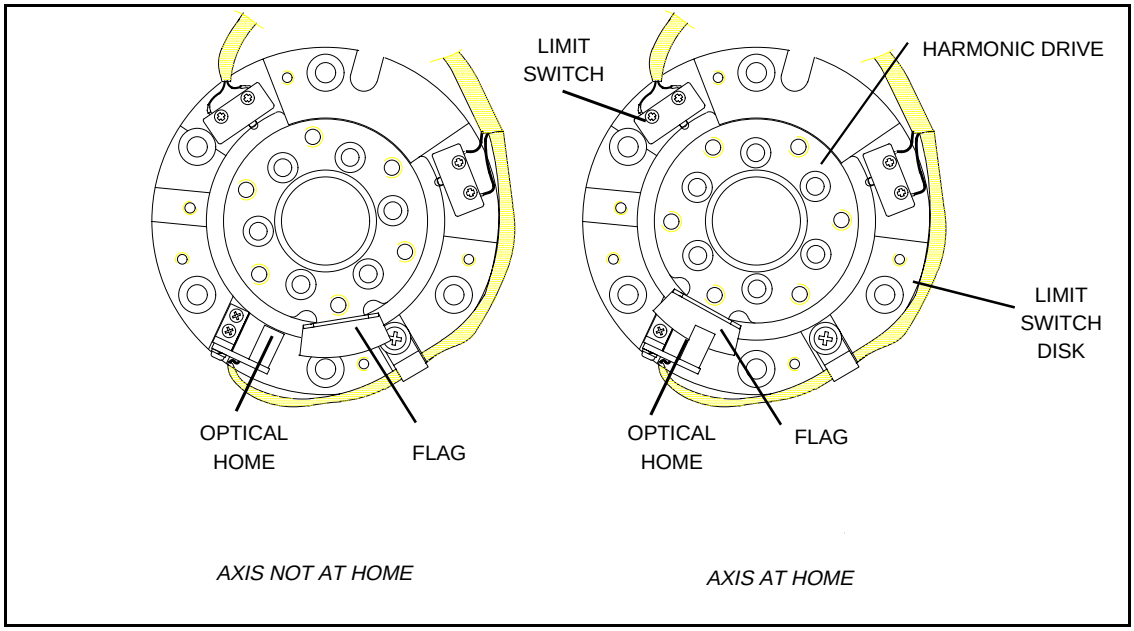


Figure 7-8: Home Switch Activation

CHAPTER 8

Wiring

Figure 9-1 is a schematic diagram of the **SCORBOT-ER IX** cable connections.

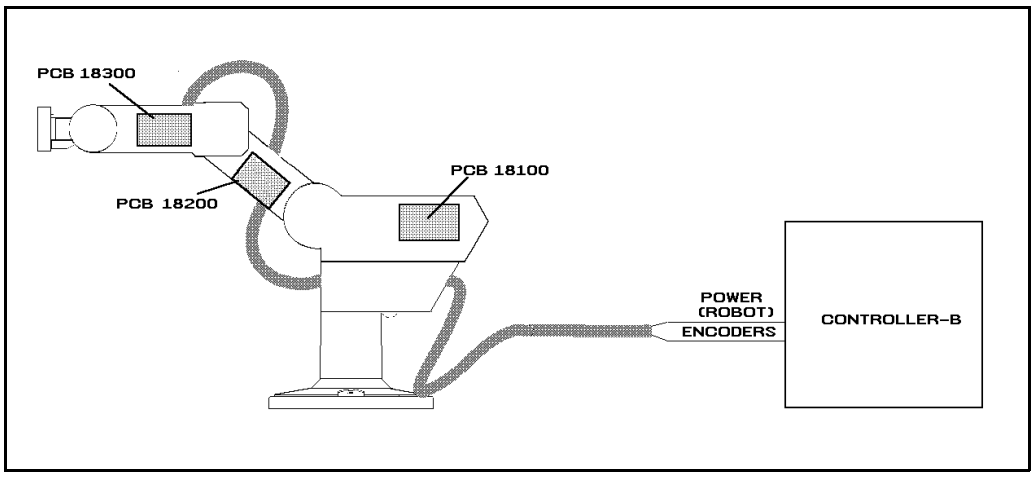


Figure 8-1: SCORBOT-ER IX Cabling

The wire braid which connects the robot to the controller contains a power (robot) cable and an encoder cable.

The body, upper arm and forearm links each contain a printed circuit board (PCB). The motors, encoders, limit switches and home switches for each axis are directly connected to one of these three internal PCBs. Two wire braids connect the PCBs. Each PCB transfers power to the motors to which it is directly connected, and receives signals from the corresponding limit and home switches. When a limit switch is triggered, the PCB automatically cuts off power to the motor that drives the axis. In addition, each PCB transfers power to the next PCB and sends encoder and home switch signals to the previous PCB.

The robot and encoder cable are directly connected to PCB 18100. The robot cable supplies power to the PCB and the encoder cable carries information from the encoders and the home switches for all six axes to the controller.

Robot (Power) Cable and Connector

Figure 8-2 shows the Burndy 19 pin male connector that joins the power cable to the controller's back panel.

The robot cable contains 12 leads. The following table details the connector pin functions and cable wiring.

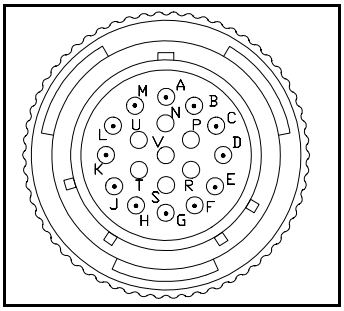


Figure 8-2: Burndy 19 Pin Connector

Robot (Power) Cable Wiring and Connector			
Pin ID	Pin Description Robot Side (J1)	Beldan Color	Pin Description Controller Side (P1)
A	Motor 1 –	black	M0_A
M	Motor 1 +	red	M0_B
C	Motor 2 –	brown	M1_A
L	Motor 2 +	orange	M1_B
E	Motor 3 –	yellow	M2_A
H	Motor 3 +	purple	M2_B
B	Motor 4 –	light blue	M3_A
K	Motor 4 +	blue	M3_B
D	Motor 5 –	grey	M4_A
J	Motor 5 +	pink	M4_B
F	Motor 6 –	white	M5_A
G	Motor 6 +	green	M5_B

Encoder Cable and Connector

The encoder cable, which connects the controller to the motor encoders and optical home switches, contains 36 leads.

Figure 8-3 shows the D37 female connector that joins the encoder cable to the controller's back panel.

The following table details the connector pin functions and describes the cable wiring.

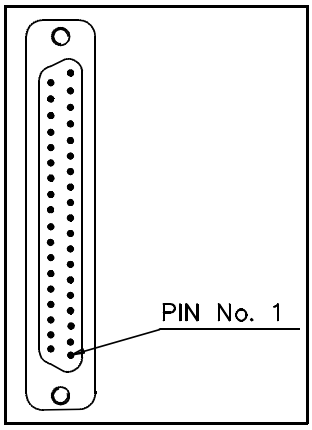


Figure 8-3:
D37 Connector

Encoder Cable and D37 Connector				
Pin ID	Pin Description Robot Side (J1)	Axis	Telephone Cable Color	Pin Description Controller Side (J2)
1	+5V	1	red	+5V
8	COMMON		yellow	COMMON 0
5	CHA1 (Encoder Pulse A)		green	CHA 0
6	CHB1 (Encoder Pulse B)		white	CHB 0
7	CHC1(Encoder Index Pulse)		black	CHC 0
31	MSWITCH (Home Switch)		blue	MSWITCH
1	+5V	2	red	+5V
12	COMMON		yellow	COMMON 1
9	CHA2 (Encoder Pulse A)		green	CHA 1
10	CHB2 (Encoder Pulse B)		white	CHB 1
11	CHC2 (Encoder Index Pulse)		black	CHC 1
32	MSWITCH (Home Switch)		blue	MSWITCH
1	+5V	3	red	+5V
16	COMMON		yellow	COMMON 2
13	CHA3 (Encoder Pulse A)		green	CHA 2
14	CHB3 (Encoder Pulse B)		white	CHB 2
15	CHC3 (Encoder Index Pulse)		black	CHC 2
33	MSWITCH (Home Switch)		blue	MSWITCH

Encoder Cable and D37 Connector				
Pin ID	Pin Description Robot Side (J1)	Axis	Telephone Cable Color	Pin Description Controller Side (J2)
2	+5V	4	red	+5V
20	COMMON		yellow	COMMON 3
17	CHA4 (Encoder Pulse A)		green	CHA 3
18	CHB4 (Encoder Pulse B)		white	CHB 3
19	CHC4 (Encoder Index Pulse)		black	CHC 3
34	MSWITCH (Home Switch)		blue	MSWITCH
2	+5V	5	red	+5V
24	COMMON		yellow	COMMON 4
21	CHA5 (Encoder Pulse A)		green	CHA 4
22	CHB5 (Encoder Pulse B)		white	CHB 4
23	CHC5 (Encoder Index Pulse)		black	CHC 4
35	MSWITCH (Home Switch)		blue	MSWITCH
2	+5V	6	red	+5V
28	COMMON		yellow	COMMON 5
25	CHA6 (Encoder Pulse A)		green	CHA 5
26	CHB6 (Encoder Pulse B)		white	CHB 5
27	CHC6 (Encoder Index Pulse)		black	CHC 5
36	MSWITCH (Home Switch)		blue	MSWITCH

CHAPTER 9

Maintenance

The maintenance and inspection procedures recommended below will ensure the best possible performance of the robot over an extended period.

Daily Operation

At the start of each working session, check the robot and controller, in the following order:

1. Before you power on the system, check the following items:
   - The installation meets all safety standards.
   - All cables are properly and securely connected.
     Cable connector screws are fastened.
   - The gripper is properly connected.
     The air supply (for a pneumatic gripper) is functioning properly.
   - Any peripheral devices or accesssories which will be used, such as the teach pendant or a remote emergency button, are properly connected to the controller.
2. After you have powered on the system, check the following items:
   - No unusual noises are heard.
   - No unusual vibrations are observed in any of the robot axes.
   - There are no obstacles in the robot's working range.
3. Bring the robot to a position near home, and activate the Home procedure. Check the following items:
   - Robot movement is normal.
   - No unusual noise is heard when robot arm moves.
   - Robot reaches home position in every axis.

Periodic Inspection

The following inspections should be performed regularly:

- Check robot mounting bolts for looseness using a wrench. Retighten as needed.
- Check all visible bolts and screws for looseness using a wrench and screwdriver. Retighten as needed.
- Check cables. Replace if any damage is evident.

The following robot components may require replacing after prolonged use of the robotic arm causes them to wear or fail:

- DC Servo Motors
- Motor Brushes
- Timing Belts
- V-Rings
- Harmonic Drives
- Cross-Roller Bearings

Troubleshooting

Whenever you encounter a problem with your system, try to pinpoint its source by exchanging the suspected faulty component—for example, robot, controller, teach pendant, cable—with one from a functioning system.

In general, when trying to determine the source of a malfunction, first check the power source and external hardware, such as controller switches, LEDs and cable connections. Then check fuses; you may also open the controller to check components, according to the procedures and instructions detailed in the ***Controller-B*** *User's Manual*.

In addition, make sure the controller is properly configured for the robot and gripper, the software commands have been correctly issued, and system parameters are properly set.

All troubleshooting procedures described in the section can be performed by the user.

☞ ***Do not attempt to open the robot arm****. There are no user-serviceable parts inside.*

If you are unable to determine and/or correct the problem, contact your service representative. Only qualified technicians may remove and/or replace robot components.

1. *Controller's MOTORS switch does not turn on; the green LED does not light.*

   - Make sure the Emergency button is released.
   - Turn off the controller, disconnect it from the power source, and open the cover.

     Check the 0.5A (SB) fuse (marked FAN/POWER/RELAYS)

2. *Controller functioning, but the robot cannot be activated.*

   - Make sure an obstacle is not blocking the robot.
   - Make sure the controller's MOTORS switch is on and the green LED is lit.
   - Make sure the controller is in the control off (COFF) state. Then activate the control on (CON) state from PC or TP.
   - Make sure all robot and encoder cables are properly connected.
   - Check driver card fuses. Each driver card has a pair of LEDs and a pair of fuses (accessible from controller back panel). The upper LED and fuse correspond to the axis number at the top of the card; the lower LED and fuse correspond to the axis number at the bottom of the card.
     Both LEDs on each card in use should be lit, indicating that power is being supplied to the axis driver. If one of the LEDs is not lit, remove the fuse for the corresponding axis and examine it. (To remove the fuse, press it in and rotate counter-clockwise.)

3. *Robot does not find Home position in one or all of the axes.*

   - Make sure the homing command was properly issued.
   - Make sure all robot and encoder cables are properly connected.
   - If the robot has just undergone maintenance or repair, use the command ZSET. Then issue the home command.
   - Make sure system homing parameters have not been erased.
     Make sure system homing parameters are properly set.
     Refer to the *ACL Reference Guide*.
   - Check whether the optical home switch for this axis is functioning.

     Manually move the faulty axis (from teach pendant or keyboard) and check the value of system variable HS[*n*] (where *n* is the index of the axis). The value of HS will change to either 1 or 0 (defined by parameter 560+*axis*) when the home switch is detected.

     To help you perform this test, prepare and continuously run a simple ACL program, as follows:

```
LABEL 1
PRINTLN HS[n]
DELAY 20
GOTO 1
```

If the value of HS does not change, possible causes:

- Faulty arm circuitry.
- Faulty optical switch; optical switch not properly mounted.
- Faulty driver circuitry
- Problem in controller power supply unit +5V1.

---

4. *One of the axes does not function.*

   - Check the driver card LED for this axis at the back of the controller. If the LED is not lit, check the corresponding fuse.
   - Check the motor drive circuitry.
   - Check the encoder:
     Enter the command SHOW ENCO to display the encoder readings.
     Enter the command COFF (to disable servo control) and then *physically* move the axis in question in both directions.

     The encoder reading should rise for rotation in one direction and fall for rotation in the opposite direction. If this does not occur, there is a problem in the encoder or its circuitry.

     If the encoder readings do not change, check whether the encoder connector is properly connected to the rear controller panel.

     The problem may be caused by faulty encoder connectors on the robot's internal PCB's.

---

5. *Motors suddenly stop. No message on screen. No response to keyboard entries.*

   - Check the power source.
   - Make sure the MOTORS power switch is on; make sure the Emergency button is not depressed.
   - Turn off the controller and open up the cover. Turn on the controller.
     Check the yellow "watchdog" LED on the main board. If it is lit, it indicates that that one of the following fuses on the power supply unit has blown out: +12VA, –12VA, +12VDR, –12VDR.
     Turn off the controller and disconnect it from the power source. Check each of these four fuses. Replace the blown fuse.

6. *Errors in the repeatability of the robot.*

   - Try to identify the faulty axis. If many or all axes are faulty, look for an electrical noise source in your environment.
   - Check the controller's ground and the robot's ground connection to the safety ground terminal at the back of the controller.
   - Check the encoder.

     Bring the robot to a starting position. Using a pencil, draw a fine, continuous line on the robot which crosses from the cover of one link to the cover of the adjacent link at the joint in question.

     Enter the command SHOW ENCO to display the encoder readings.
     Enter the command COFF (to disable servo control) and then *physically* move the axis to another position. Then return to the starting position marked by the line you drew. Check the encoder reading for the axis again. It should be within 5 counts of the previous reading; if not, the encoder needs to be replaced.

7. *Unusual noise.*

   - Loose screws.
   - Poor lubrication.
   - Ratcheting.
   - Worn motor brushes.
   - Worn timing belt.
   - Damaged harmonic drive.

8. *Unusual smell.*

   - A motor has burnt out and needs to be replaced.

9. *Axis/axes vibrating, too weak to carry load, motion not smooth, or jerks during or at end of motion.*

   - System parameters are not properly adjusted.
     Refer to the ***ACL** Reference Guide*.
   - Problem in axis driver card(s) in the controller.
     Refer to the ***Controller-B** User's Manual.*

10. *Pneumatic gripper does not respond.*

- Check that all air hoses are connected properly.
- Make sure the gripper is connected to the proper controller output.
- Check the relay output to which the gripper is connected.

  Check whether the relays have been switched (LED is lit):
  - In output OFF, NC is shorted to COM, NO is disconnected from COM.
  - In output ON, NO is shorted to COM, NC is disconnected from COM.

  If outputs have not been switched, check the flat cable in the controller connecting the main board (J17) and the I/O card.

Messages

Following is a alphabetical listing of system messages which indicate a problem or error in the operation of the robot arm. Refer to the ***ACL** Reference Guide* for additional error messages.

`Axis disabled.`

(1) A movement command could not be executed because servo control of the arm has been disabled (COFF).
(2) A previous movement of the arm resulted in an Impact or Trajectory error, thereby activating COFF and disabling the arm.

- Check the movements of the robot, and correct the command(s).

`CONTROL DISABLED.`

Motors have been disconnected from servo control. Possible causes:

(1) COFF (control off) command was issued.

(2) CON (control on) has not been issued; the motors have not been activated.

(3) A previous error (such as Impact Protection, Thermic Overload or Trajectory Error) activated COFF, thereby disabling the arm.

`*** HOME FAILURE AXIS n.`

The homing procedure failed for the specified axis. Possible causes:

(1) The home microswitch was not found.

(2) The motor power supply is switched off.

(3) Hardware fault on this axis.

`Home on group/axis not done.`

You attempted to move the arm to a recorded positions, or to record a position, before homing was performed on the group or axis.

***** IMPACT PROTECTION axis *n***

The controller has detected a position error which is too large. The system aborted all movements of that axis group, and disabled all axes of that group. The user routine CRASH, if it exists, has been executed. Possible causes:

(1) An obstacle prevented the movement of the arm.
(2) An axis driver fuse has blown.
(3) The motor power switch is turned off.
(4) An encoder fault.
(5) A mechanical fault.
(6) The axis is not connected.

- Determine and correct the cause of the position error. Then reenable servo control of the motors (CON), and restart the program.

INDEX pulse not found ***axis n***

The index pulse of the encoder was not found during the homing of the specified axis. Possible causes:

(1) The distance between the index pulse and the home switch transition position has changed, due to a mechanical fault on the axis or a maintenance procedure (such as replacement of the motor, motor belt, encoder, or gear).

- Enter the command ZSET. Then retry homing.

(2) Index pulse faulty.

- Check the encoder and wiring.

***** LOWER LIMIT AXIS *n.***

During keyboard or TP manual movement of the specified axis, its encoder attained its minimum allowed value.

- Move the axis in the opposite direction.

Motor power switch is OFF.

Be sure the controller's MOTORS switch is on. Activate CON. Then repeat the motor or movement command.

No hard homing axis n.

The specified axis has not been configured for hard homing.

- Use the HOME command (instead of HHOME). OR
- Check the type of homing suitable for that axis. If necessary, change the system parameters to allow hard homing of the axis.

No homing.

The homing parameters for the axis (PAR 460+*axis* and PAR 600+*axis*) are set to 0; as a result, the homing procedure will not be performed on the axis.

***** OUT OF RANGE axis *n***

An attempt was made to record a position (HERE, HEREC, etc.) while the robot arm was out of its working envelope.

- Manually move the arm to a location within its working envelope. Then repeat the command.

***** THERMIC OVERLOAD axis *n***

Through a software simulation of motor temperature, the system has detected a dangerous condition for that motor. The system aborted all movements of that axis group, and disabled all axes of that group. The user routine CRASH, if it exists, has been executed. Possible causes:

(1) The arm attempted to reach a position, which could not be reached due to an obstacle (for example, a position defined as being above a table, but actually slightly below the table's surface). The impact protection is not activated because the obstacle is close to the target position. However, integral feedback will increase the motor current and the motor will overheat, subsequently causing the Thermic Protection to be activated.

(2) An axis driver is faulty or its fuse has blown.

(3) The robot arm is near to the target position, but does not succeed in reaching it, due to a driver fault. The software will then detect an abnormal situation.

(4) The Thermic Protection parameters are improperly set, or have been corrupted by improper loading of parameters.

- Check the positions, the axis driver card and parameters. Reenable servo control of the motors (CON).

***** TOO LARGE SPEED axis *n*.**

Possible causes:

(1) The controller has detected a movement which is too fast; that is, the required displacement of the encoder, as calculated from the speed limit parameter, PAR 180+*axis*, is too great.

(2) Since the trajectory is not calculated prior to a linear or circular movement, the linear or circular movement may cause one of the joints to move too fast.

- Lower the value of speed for that movement.

***** TRAJECTORY ERROR !**

During movement, the robot arm reached its envelope limits, and the system aborted the movement. This may occur when executing the following types of movements: linear (MOVEL), circular (MOVEC) , MOVES, and SPLINE. Since the trajectory is not computed prior to motion, the movement may exceed the limits of the working envelope.

- Modify the coordinate values of the positions which define the trajectory.

***** UPPER LIMIT AXIS *n***

During keyboard or TP manual movement of the specified axis, its encoder attained its maximum allowed value.

- Move the axis in the opposite direction.

16.2. Datasheet Scorbot-ER V

SCORBOT-ER 5Plus

User Manual

Catalog #100016 Rev. C

intelitek

SCORBOT-ER 5Plus

Catalog # 100016 Rev. C

February 1996

Safety Warning!
Use the SCORBOT ER-5Plus with extreme caution.
The SCORBOT ER-5Plus can be dangerous and can cause severe injury.
Setup up a protective screen or guard rail around the robot to keep people away from its working range.

INTELITEK INC.
444 East Industrial Park Drive
Manchester NH 03109-537
Tel: (603) 625-8600
Fax: (603) 625-2137
Web site www.intelitek.com

Table of Contents

CHAPTER 1 General Information 1-1

Handling Instructions 1-1
Acceptance Inspection 1-2
Repacking for Shipment 1-4
Safety Precautions 1-4
Robot 1-4
Controller 1-5
Warnings 1-5
Robot 1-6
Controller 1-6

CHAPTER 2 The Robot Arm 2-1

Specifications 2-2
Structure 2-3
Work Envelope 2-4
Motors 2-5
Encoders 2-5
Microswitches 2-6
Transmissions 2-6
Gripper 2-7

CHAPTER 3 The Controller 3-1

Specifications 3-2
Controller Functions 3-5
Power On/Off Switch and LED 3-5
Motors and User Power Supply Switch and LED 3-5
Emergency Switch and Lamp 3-6
User Power Supply Terminals 3-7
Input and Output Terminals and LEDs 3-7
Inputs 3-7
Outputs 3-8
Relay Outputs 1-4 3-8
Open Collector Outputs 5-16 3-8
Input and Output LEDs 3-10

CHAPTER 4 Installation 4-1

Preparations 4-1
Cable Connections 4-3
Peripheral Axes 4-4
Power On 4-5
Controller Configuration 4-7

CHAPTER 5 Operating Methods 5-1

Software 5-1
ACL 5-1
ATS 5-1
SCORBASE 5-2
Teach Pendant 5-3
Keypad Functions 5-3
The Display Panel 5-7

CHAPTER 6 Operating the Robot 6-1

DIRECT Mode 6-1
Manual Mode 6-1
Using this Manual 6-2
Activating the Sytem 6-2
Homing the Robot and Peripheral Axes 6-3
Coordinate Systems 6-5
Cartesian (XYZ) Coordinates 6-5
Joint Coordinates 6-5
Servo Control 6-7
Axis Control Groups 6-8
Moving the Axes 6-9
XYZ and Joint Movements 6-9
Activating the Gripper 6-11
Setting the Speed 6-12
Defining and Recording Positions 6-13
Relative Positions 6-15
Listing Positions 6-16
Deleting Positions 6-16
Moving to Recorded Position 6-17
Linear Movement 6-18
Circular Movement 6-18

CHAPTER 7 Programming with ACL 7-1

EDIT Mode 7-1
Help 7-1
Creating a Program 7-2
Writing a Program 7-2
Running a Program 7-3
Program Loop 7-3
Displaying Program Lines 7-4
Halting Program Execution 7-4
Suspend the Program 7-4
Abort the Program 7-5
Stop the Program 7-5
Delaying Program Execution 7-6
Variable Programming 7-6
Mathematical and Logical Functions 7-7
Iteration Functions 7-8
Conditional Functions 7-9

Input and Output Programming 7-10
Displaying Input/Output Status 7-10
Inputs 7-10
Outputs 7-11
Activating Output-Driven Devices 7-11
Pneumatic End Effectors or Devices 7-11
Warning Light 7-11
Sample Program: INOUT 7-12
Program Directory 7-14
Multi-Tasking 7-14
Displaying Program Status 7-15
Activating a Program from Another Program 7-15
Simultaneous Execution 7-15
Program Interrupt 7-16
Downloading a Program (Restore) to Controller 7-16
Calculating and Moving Along a Path 7-17
Parabola 7-17
CALC 7-17
PARAB 7-17
Sine 7-19
Saving a Program (Backup) to Disk 7-21

CHAPTER 8 Maintenance 8-1

Maintenance 8-1
Daily Operation 8-1
Periodic Inspection 8-2
Troubleshooting 8-3
General System Check 8-3
Diagnostic Procedures 8-4
Error Messages 8-14
Adjustments and Repairs 8-16
Adjusting the Timing Belts 8-16
Adjusting Base Anti-Backlash 8-17
Tightening the Oldham Coupling in Gripper 8-18
Gripper Disassembly 8-18
Gripper Reassembly 8-18
Opening the Controller Cover 8-19
Changing the Voltage Setting 8-19
Replacing Fuses 8-20
Logic Power Supply Fuse 8-20
Power Transformer Fuse 8-20
User Power Supply Fuse 8-20
Driver Card Fuses 8-21
Changing the I/O Logic Mode 8-22
Replacing or Adding a Driver Card 8-23
Adjusting Driver Card Current Limit 8-24
Driver Card Jumper Configuration 8-24
Installing the Auxiliary RS232 Communication Card . . 8-25

CHAPTER 9 **Parts Lists** **9-1**

Robot 9-6
Controller 9-11

CHAPTER 10 **Wiring** **10-1**

Robot Wiring 10-1
Single Axis Wiring 10-3
Controller-Computer RS232 Cable 10-4

APPENDIX A **Theory of Control** **A-1**

Servo Control A-1
Open Loop Control A-1
Closed-Loop Control A-1
Digital Control A-2
Transient and Steady State Responses A-4
Controller-A Control Process A-6
Trajectory Control A-7
Paraboloid A-7
Trapezoid A-7
Path Control A-7
Point-to-Point Control A-7
Continuous Path Control A-8
The Control Parameters A-9
Proportional Control A-9
Differential Control A-10
Integral Control A-10
Proportional–Integral–Differential Control A-11
Offset A-11
Changing Parameter Values A-12

List of Figures

Figure 1-1: Robot Arm Parts 1-1
Figure 2-1: SCORBOT-ER Vplus Robot Arm 2-1
Figure 2-2: Robot Arm Links 2-3
Figure 2-3: Robot Arm Joints 2-3
Figure 2-4: Operating Range (Top View) 2-4
Figure 2-5: Operating Range (Side View) 2-4
Figure 2-6: Motor 2-5
Figure 2-7: Encoder 2-5
Figure 2-8: Microswitch 2-6
Figure 2-9: Transmissions 2-6
Figure 2-10: SCORBOT-ER Vplus Gripper 2-7
Figure 3-1: Controller-A 3-1
Figure 3-2: Controller Front Panel 3-5
Figure 3-3: Input Terminals 3-7
Figure 3-4: Relay Output Terminals 3-8
Figure 3-5: Relay Output States 3-8
Figure 3-6: Open Collector Output Terminals 3-8
Figure 3-7: Open Collector Output: PNP Mode 3-9
Figure 3-8: Open Collector Output: NPN Mode 3-9
Figure 4-1: SCORBOT-ER Vplus Installation 4-1
Figure 4-2: Robot Base Plate Layout 4-2
Figure 4-3: Controller Rear Panel 4-3
Figure 5-1: Teach Pendant 5-3
Figure 6-1: SCORBOT-ER Vplus Home Position 6-3
Figure 6-2: Cartesian Coordinates 6-5
Figure 7-1: Parabola 7-17
Figure 8-1: Belt Tension 8-2
Figure 8-2: Main Board - Memory 8-5
Figure 8-3: Driver Card LEDs 8-6
Figure 8-4: Driver Card Fuse 8-7
Figure 8-5: Encoder Signals 8-9
Figure 8-6: Main Board - Connectors 8-12
Figure 8-7: Tightening Belts in Forearm 8-16
Figure 8-8: Tightening Belts in Upper Arm 8-16
Figure 8-9: Tightening Belts in Robot Base 8-16
Figure 8-10: Shoulder Cover Screws 8-17
Figure 8-11: Controller Voltage Setting 8-19
Figure 8-12: I/O Card - Logic Jumpers 8-22
Figure 8-13: Axis Driver Card 8-23
Figure 8-14: Auxiliary RS232 Communication Card 8-25
Figure 9-1: Gripper Assembly 9-2
Figure 9-2: Robot Arm Assembly 9-3

Figure 9-3: Anti-Backlash Assembly 9-4
Figure 9-4: Base and Motors Assembly 9-5
Figure 9-5: Controller-A 9-10
Figure 10-1: Robot D50 Connector 10-1
Figure 10-2: Motor Wiring 10-3
Figure 10-3: Motor with D9 Connector 10-3
Figure A-1: Open-Loop and Closed-Loop Control A-2
Figure A-2: Analog and Digital Signals A-3
Figure A-3: Transient and Steady States A-4
Figure A-4: Transient State Responses A-5
Figure A-5: Controller-A Control Loop A-5
Figure A-6: Controller-A Control Signals A-6
Figure A-7: Trajectory Control Profiles A-7
Figure A-8: Linearity and Non-Linearity A-11
Figure A-9: Control System Offset A-11

CHAPTER 1

General Information

☞ *Read this chapter carefully before you unpack the robot and controller.*

This chapter contains instructions for handling the **SCORBOT-ER Vplus** and **Controller-A**.

This chapter also includes important safety guidelines and warnings.

Handling Instructions

Lift and carry the robot arm only by grasping the body or the base. See Figure 1-1.

Do not lift and/or carry the robot arm by its gripper, upper arm or forearm.

Do not touch the microswitches, cams or encoders.

Lift and carry the controller by grasping it on and under the left and right side panels.

Do not grasp the controller on either its front or back panel, and avoid handling near the power switch.

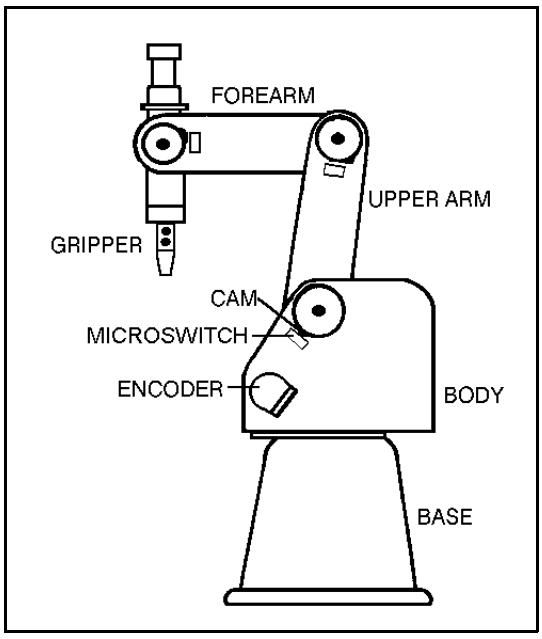


Figure 1-1: Robot Arm Parts

Acceptance Inspection

The robot arm and the controller are packed in two separate cartons. *Save the original packing materials and shipping carton*. You may need them later for shipment or storage.

After removing the robot arm and controller from their shipping cartons, examine them for signs of shipping damage. If any damage is evident, do not install or operate the system. Notify your freight carrier and begin appropriate claims procedures.

The following table lists standard components in the **SCORBOT-ER Vplus** package.

Make sure you have received all the items listed on the shipment's packing list. If anything is missing, contact your supplier.

SCORBOT-ER Vplus Standard Package (Catalog # 403)	
Item	Description
SCORBOT-ER Vplus Robot Arm and Gripper	Includes: power cable 100/110/220VAC; RS232 cable; gripper path cable; 4 driver cards (for 8 axes); 3 bolts for mounting robot; set of hex wrenches.
ACL Controller-A	
Software	ATS (Advanced Terminal Software): 2 diskettes; one is write-protected
	SCORBASE Level 5 Software: 1 diskette
Documentation	***SCORBOT-ER Vplus*** *User's Manual*
	ACL *Reference Guide*
	ATS *Reference Guide*
	SCORBASE *Software Reference Guide*

The following table is a sampling of the optional accessories which are compatible for use with the **SCORBOT-ER Vplus** system.

For a complete list of the accessories, devices, software and documentation for integration and use with the **SCORBOT-ER Vplus** sytem, contact your agent.

Optional Components for SCORBOT-ER Vplus System		
Item	Cat. #	Notes
Teach Pendant for **Controller-A**	1703	
Driver Card for Peripheral Axes: Card with two 2A fuses Card with one 2A fuse and one 4A fuse Card with two 4A fuses	 45018 45019 45020	 Each card drives 2 axes; If ordered with controller, card is factory-installed.
Auxiliary RS232 Communication Card Cable with 8 connectors for aux. card	45012 40024	If ordered with controller, card is factory-installed.
SCORBASE Levels 1-3 software	9004	
DC Motor Kits: Motor with 5.9:1 gear ratio Motor with 19.5:1 gear ratio Motor with 65.5:1 gear ratio Motor with 127:1 gear ratio	 1210 1212 1211 1206	12VDC, Includes encoder and connector cable.
Rotary Table (black)	1004	12VDC, ∅350mm plate
Proximity Sensor for Rotary Table	1209	
Conveyor Belt (gray)	1006	12VDC, 20-slot encoder
Proximity Sensor for Conveyor	1203	
Experiment Table	1201	
48" Linear Slidebase 72" Linear Slidebase	1001 1002	12VDC
1.0M Linear Slidebase 1.5M Linear Slidebase	1008 1007	24VDC
Robot Adapter Plate for Linear Slidebase	10001	
Gripper Adapter for Round/Square Pieces	609	
Vacuum Gripper (1 suction cup)	601	Requires #1204 and #1208
Vacuum Gripper (3 suction cups)	602	
Air Brush Paint Gun	603	
Syringe Dispenser	604	
Utilities Control Box	1204	Includes solenoid value, air regulator, fittings, power supply.
Air Supply Adapter Kit	1208	Includes 2 quick connectors and air hose.
I/O Interface Box for **Controller-A**	1215	

Repacking for Shipment

Be sure all parts are back in place before packing the robot/controller.

☞ *The robot and controller should be repacked in their original packaging for transport.*

If the original carton is not available, wrap the robot/controller in plastic or heavy paper. Put the wrapped robot/controller in a strong cardboard box at least 15 cm (about 6 inches) longer in all three dimensions than the robot. Fill the box equally around the unit with resilient packing material (shredded paper, bubble pack, expanded foam chunks).

Seal the carton with sealing or strapping tape. Do not use cellophane or masking tape.

Safety Precautions

This manual provides complete details for proper installation and operation of the **SCORBOT-ER Vplus** and **Controller-A**. Do not install or operate the robot or controller until you have thoroughly studied this *User's Manual*. Be sure you fheed the safety guidelines for both the robot and the controller.

Robot

1. Make sure the robot base is properly and securely bolted in place.
2. Make sure the robot arm has ample space in which to operate freely.
3. Make sure a guardrail, rope or safety screen has been set up around the **SCORBOT-ER Vplus** operating area to protect both the operator and bystanders.
4. Do not enter the robot's safety range or touch the robot when the system is in operation. Before approaching the robot, make sure the motor switch on the controller front panel has been shut off.
5. Make sure loose hair and clothing is tied back when you work with the robot.

Controller

1. The power cable must have a ground connection. If your power outlet does not have a safety ground, do not connect the controller. *Failure to connect the power cable to a grounded outlet could result in electrical shock.*

1. Turn off the controller's motor switch before you enter the robot's operating area.

2. Turn off the controller's power switch before you connect any inputs or outputs to the controller.

3. Turn off the controller's power switch *and* disconnect the controller power cable from the AC power outlet before you open the controller cover or remove any fuses. The power cable must be disconnected to remove possible shock hazard.

4. Never open the controller cover during robot operation.

☞ *Be sure you know how to immediately abort all running programs and stop all axes of motion:*

- *press the Abort key on the teach pendant, or*
- *use the ACL command A <Enter>, or*
- *press the controller's red EMERGENCY button.*

Warnings

Do not install or operate the **SCORBOT-ER Vplus** or **Controller-A** under any of the following conditions:

- Where the ambient temperature or humidity drops below or exceeds the specified limits.
- Where exposed to large amounts of dust, dirt, salt, iron powder, or similar substances.
- Where subject to vibrations or shocks.
- Where exposed to direct sunlight.
- Where subject to chemical, oil or water splashes.
- Where corrosive or flammable gas is present.
- Where the power line contains voltage spikes, or near any equipment which generates large electrical noises.

Robot

- Do not overload the robot arm. The combined weight of the workload and gripper may not exceed 1kg (2.2 lb). It is recommended that the workload be grasped at its center of gravity.
- Do not use physical force to move or stop any part of the robot arm.
- Do not drive the robot arm into any object or physical obstacle.
- Do not leave a loaded arm extended for more than a few minutes.
- Do not leave any of the axes under mechanical strain for any length of time. Especially, do not leave the gripper grasping an object indefinitely.
- Since the **SCORBOT-ER Vplus** motors are rated 12VDC nominal, while the controller motor drivers supply 24VDC, do not drive axes continuously in one direction at maximum speeds. Specifically, when using the **ACL** command: SET ANOUT[*n*]=*DAC*, make sure the *DAC* value is in the range ±2500.

Controller

- Before you plug the controller into the AC outlet, make sure its voltage requirement (as seen on the tag at the back of the controller) matches your voltage supply.

  If the voltage setting does not match your supply, do not connect the controller; contact your agent.
- Do not connect any voltage in excess of 24 VDC to the input terminals.
- Do not connect any voltage in excess of 24VDC to the output terminals.
- Never connect voltage from a power supply directly to any open collector outputs (terminals 5–16). The open collector ouputs must always be connected to a load. Never connect a load to voltage exceeding 24VDC.
- Never drive a current of more than 4A through the relay outputs (terminals 1-4).
  Never drive a current of more than 0.5A through the open collector outputs (terminals 5–16).

CHAPTER 2

The Robot Arm

This chapter details the specifications and components of the **SCORBOT-ER Vplus** robot arm.

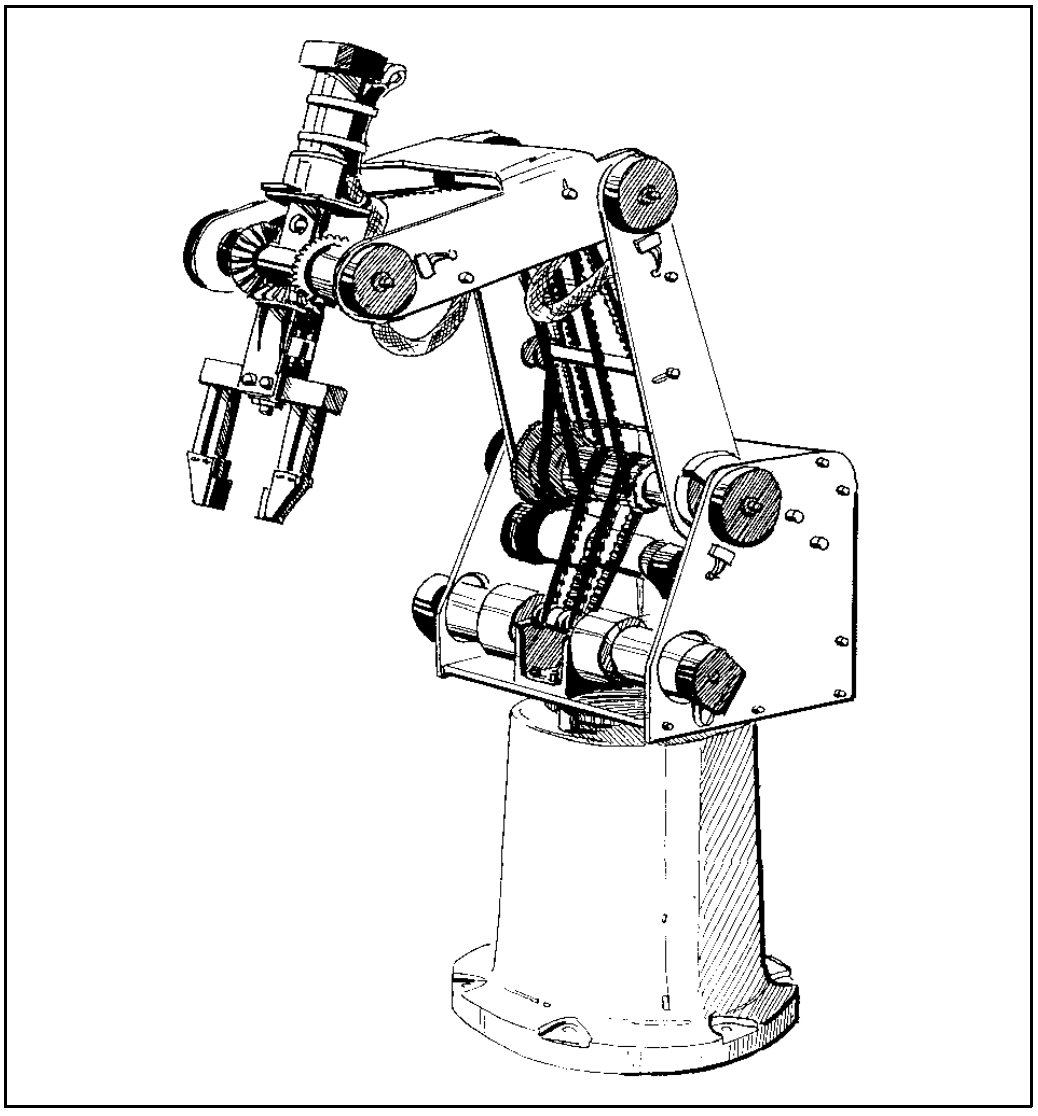


Figure 2-1: SCORBOT-ER Vplus Robot Arm

Specifications

The following table details the robot arm specifications.

SCORBOT-ER Vplus Specifications	
Mechanical Structure	Vertical articulated
Number of Axes	5 axes plus servo gripper
Axis Movement Axis 1: Base rotation Axis 2: Shoulder rotation Axis 3: Elbow rotation Axis 4: Wrist pitch Axis 5: Wrist roll	 310° +130° / –35° ±130° ±130° Unlimited (mechanically); ±570° (electrically)
Maximum Operating Radius	610mm (24.4")
End Effector	DC servo gripper, with optical encoder, parallel finger motion; Measurement of object's size/gripping force by means of gripper sensor and software.
Maximum Gripper Opening	75 mm (3") without rubber pads 65 mm (2.6") with rubber pads
Hard Home	Fixed position on each axis, found by means of microswitches
Feedback	Optical encoder on each axis
Actuators	12VDC servo motors
Motor Capacity (axes 1–6)	15 oz. in Peak Torque (stall) 70 W Power for Peak Torque
Gear Ratios	Motors 1, 2, 3: 127.1:1 Motors 4, 5: 65.5:1 Motor 6 (gripper) 19.5:1
Transmission	Gears, timing belts, lead screw
Maximum Payload	1 kg (2.2 lb.), including gripper
Position Repeatability	±0.5 mm (0.02") at TCP (tip of gripper)
Weight	11.5 kg (25 lb)
Maximum Path Velocity	600 mm/sec (23.6"/sec)
Ambient Operating Temperature	2°–40°C (36°–104°F)

Structure

The **SCORBOT-ER Vplus** is a vertical articulated robot, with five revolute joints. With gripper attached, the robot has six degrees of freedom. This design permits the end effector to be positioned and oriented arbitrarily within a large work space.

Figures 2-2 and 2-3 identify the joints and links of the mechanical arm.

The movements of the joints are described in the following table:

Axis No.	Joint Name	Motion	Motor No.
1	Base	Rotates the body.	1
2	Shoulder	Raises and lowers the upper arm.	2
3	Elbow	Raises and lowers the forearm.	3
4	Wrist Pitch	Raises and lowers the end effector (gripper).	4+5
5	Wrist Roll	Rotates the end effector (gripper).	4+5

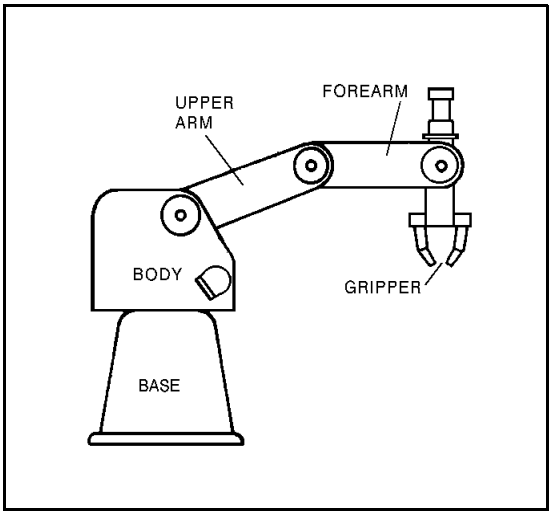


Figure 2-2: Robot Arm Links

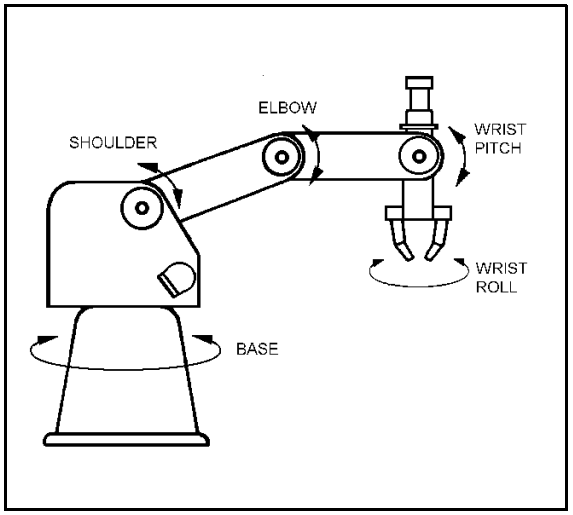


Figure 2-3: Robot Arm Joints

Work Envelope

The length of the links and the degree of rotation of the joints determine the robot's work envelope. Figures 2-4 and 2-5 show the dimensions and reach of the **SCORBOT-ER Vplus**.

The base of the robot is normally fixed to a stationary work surface. It may, however, be attached to a slidebase, resulting in an extended working range.

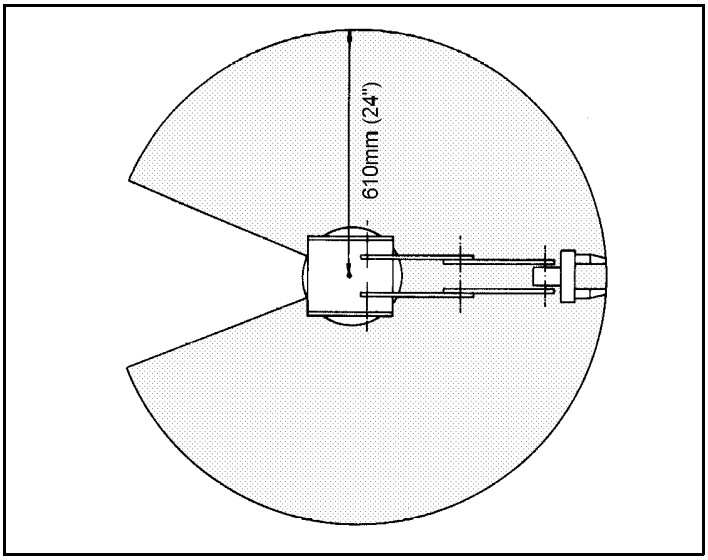


Figure 2-4: Operating Range (Top View)

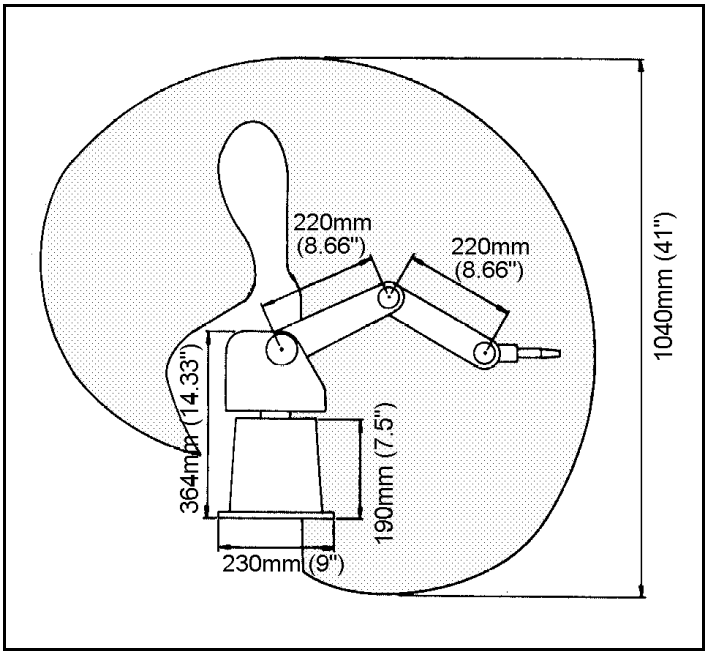


Figure 2-5: Operating Range (Side View)

Motors

The robot's five axes and gripper are operated by DC servo motors. The direction of motor revolution is determined by the polarity of the operating voltage: positive DC voltage turns the motor in one direction, while negative DC voltage turns it in the opposite direction.

Each motor is fitted with an encoder for closed-loop control.

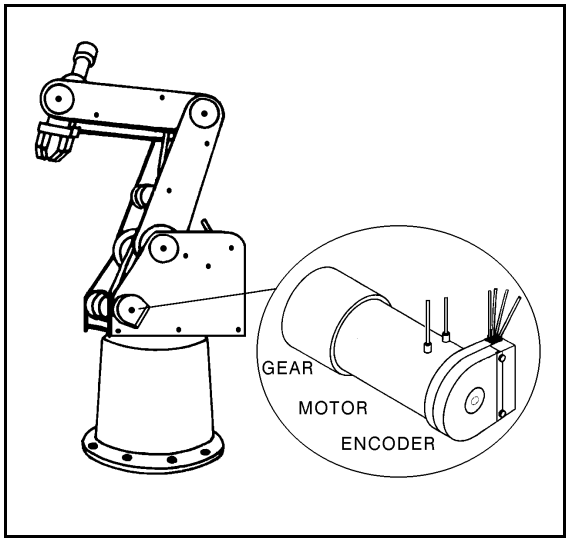


Figure 2-6: Motor

Encoders

The location and movement of each axis is measured by an electro-optical encoder attached to the shaft of the motor which drives the axis.

When the robot axis moves, the encoder generates a series of alternating high and low electrical signals. The number of signals is proportional to the amount of axis motion. The sequence of the signals indicates the direction of movement.

The controller reads these signals and determines the extent and direction of axis movement.

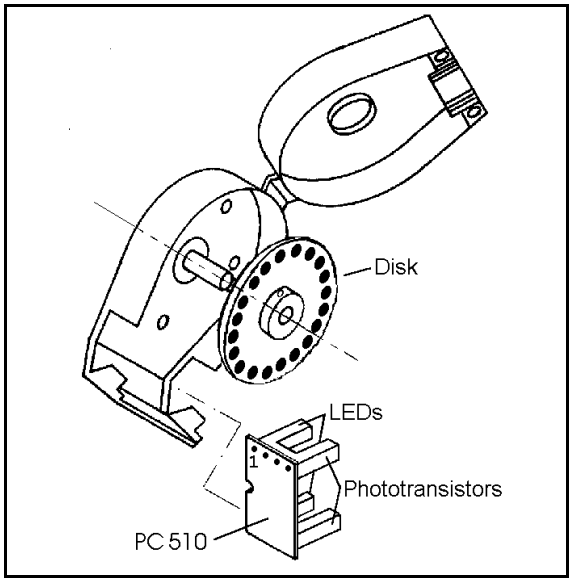


Figure 2-7: Encoder

Microswitches

Five microswitches are fitted onto the frame of the robot arm. When the robot assumes the position in which the microswitch for each joint is depressed (by means of a cam), this predetermined position is known as home. This is the point of reference for robot operation. Whenever the system is turned on, the robot should be sent to this position, by means of a software homing routine.

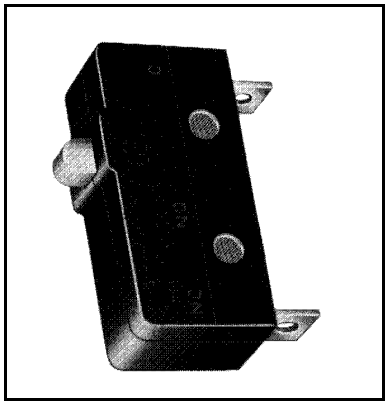


Figure 2-8: Microswitch

Transmissions

Several kinds of transmissions are used to move the links of the robot arm.

- Spur gears move the base and shoulder axes.
- Pulleys and timing belts move the elbow axis.
- Pulleys and timing belts, and a bevel gear differential unit at the end of the arm move the wrist pitch and roll axes.
- A lead screw transmission opens and closes the gripper.

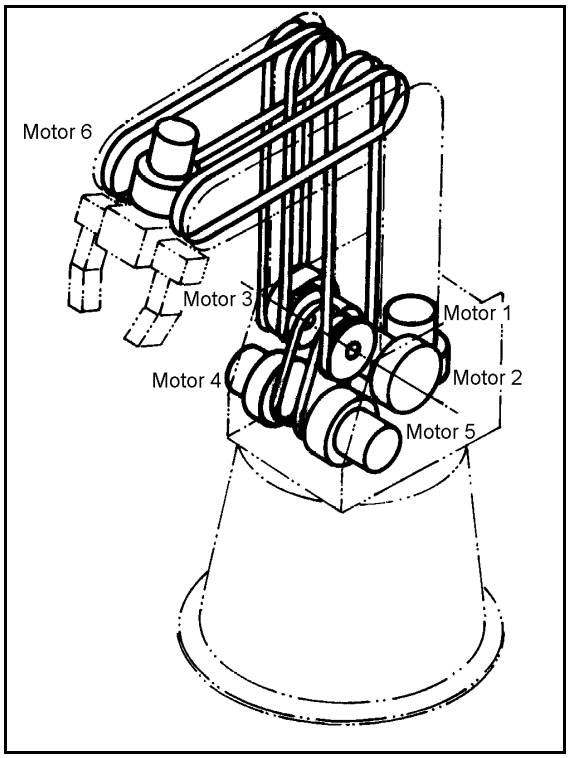


Figure 2-9: Transmissions

Gripper

The **SCORBOT-ER Vplus** has a jaw gripper fitted with rubber pads. These pads can be removed to allow the attachment of other end effector devices, such as suction pads.

Three bevel gears form a differential gear train which moves the wrist joint. When motors 4 and 5 are driven in opposite directions, the wrist pitch moves up and down. When motors 4 and 5 are driven in the same direction, the wrist rolls clockwise and counterclockwise. A leadscrew coupled directly to motor 6 causes the gripper to open and close.

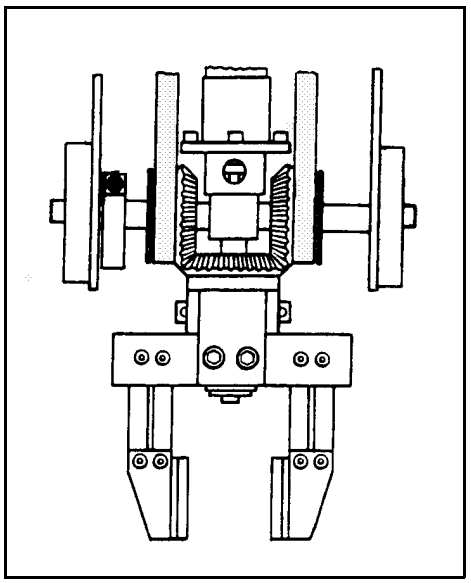


Figure 2-10: SCORBOT-ER Vplus Gripper

This page intentionally left blank.

CHAPTER 3

The Controller

This chapter details the specifications and functions of **Controller-A**, which controls the **SCORBOT-ER Vplus** robotic system.

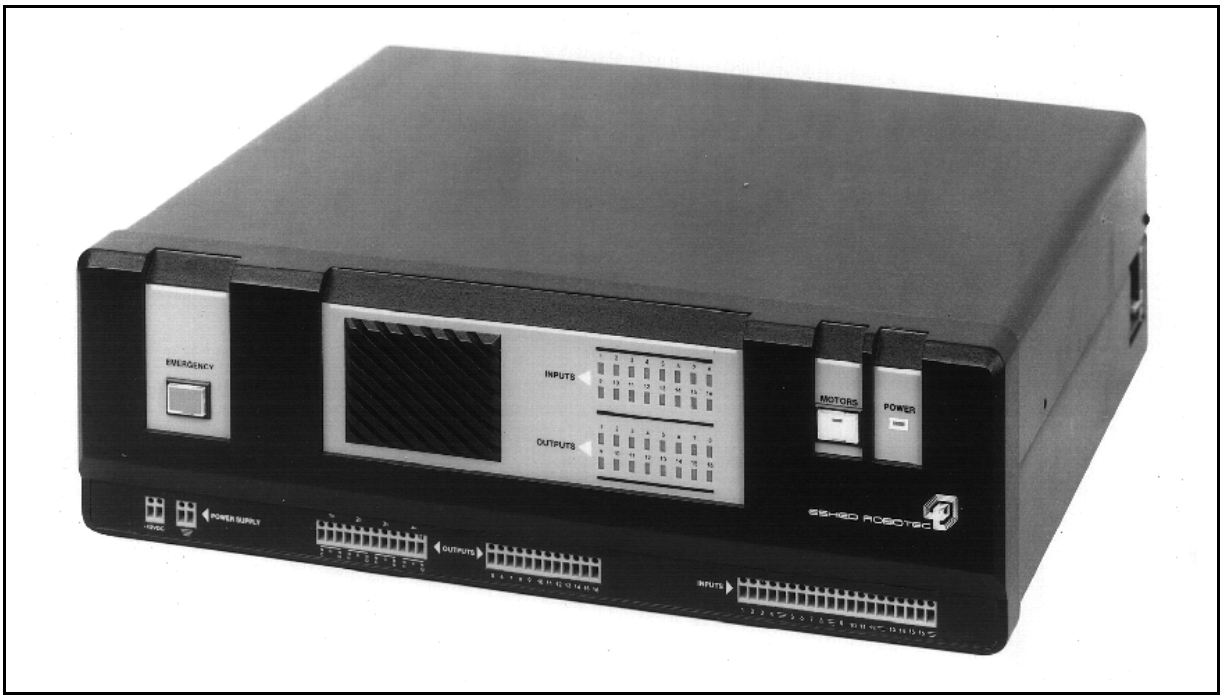


Figure 3-1: Controller-A

Specifications

Controller-A Specifications		
Item	Specification	Notes
Type of Control	Stand-alone Real-time Multi-tasking PID (proportional, integral, differential) PWM (pulse width modulation)	Terminal or PC required only for programming stage.
Number of Servo Axes	Standard: 8 Maximum: 11	
Groups of Control	11 axes can be divided into 3 groups: Group A Group B Group C (independent axes)	Each group has independent control. Axis interpolation in groups A and B.
Axis Drivers	PWM (pulse width modulation) 20 KHz	
Path Control	PTP (point to point), CP (continuous path) Joint Linear Circular User-defined path	10 ms control cycle. Software controlled acceleration/deceleration. PID parameters.
Trajectory Control	Paraboloid Trapezoid Open Loop (not for user)	
Speed Control	Speed Travel time	Speed programmed as a percentage of range.
Control Parameters	Servo control Speed, velocity profile, smoothing Axis position error Gripper operation Thermic, impact, limit protection Homing Encoder interface Cartesian calculations	
Power Requirements	100/110/220V AC, 50/60Hz, 500W max.	± 5%
Internal Power Supplies	Motors: +24VDC, 18A User: +12VDC, 2A	
Weight	19 kg (42 lbs)	
Dimensions	490mm (19.3") L 445mm (17.5") W 150mm (5.9") H	

Controller-A Specifications		
Item	Specification	Notes
Ambient Operating Temperature	2°–40°C (36°–104°F)	
CPU	Motorola 68010	
EPROM	384KB	
RAM	System: 64KB User: 128KB	
Communication	RS232 serial port	
Inputs	16 inputs (with indicator LEDs); NPN (default) and PNP logic modes.	
Outputs	12 open collector outputs (with indicator LEDs); NPN (default) and PNP logic modes.	24VDC maximum
	4 relay outputs (with indicator LEDs)	
Programming Languages	**ACL**: Advanced Control Language	Using any terminal Using PC with **ATS**
	SCORBASE Level 5 Software	Using PC
Position Recording	Absolute Relative Cartesian Joint	Using: ACL, **SCORBASE**, Teach Pendant
No. of program lines/positions	12800 lines or 6375 positions (or any combination)	
No. of programs in user RAM	Hundreds; depends on length of programs.	
Multi-tasking	Maximum simultaneous execution: 20 user tasks	
Positioning System	Incremental optical encoders	
Coordinate System	XYZ coordinates Joint coordinates	
LED Indicators	Main power Inputs/Outputs Servo Power Emergency	On front panel
	Axis power	On rear panel
Safety Features	Emergency switch Motor power switch	On front panel
	Adjustable current limit Automatic fuse	On all axes
	Thermic, impact and limit software protection	

Controller-A Specifications		
Item	Specification	Notes
Connectors	Inputs/Outputs User power supply	Terminals on front panel
	Axis drivers Gripper RS232 channel Teach Pendant Robot Auxiliary RS232 channels (optional)	On rear panel: D9 connectors D9 connector D25 connector D25 connector D50 connector D37 connector
Teach Pendant	30 multi-function keys 2 line LCD display; 16 characters per line Full control features	

Controller Functions

The front panel of the controller contains switches, LEDs and connection terminals for operator use. Refer to Figure 3-2.

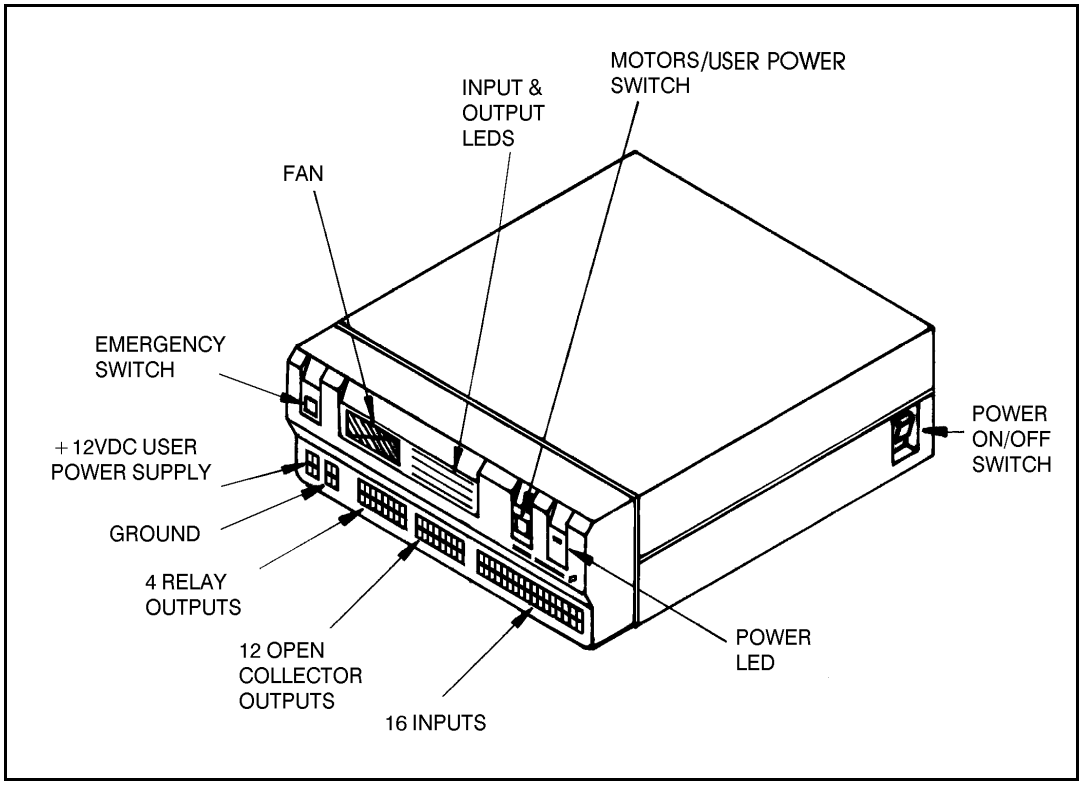


Figure 3-2: Controller Front Panel

Power On/Off Switch and LED

The controller's power switch, which is located on the side of the controller, connects and disconnects AC power to the controller.

The yellow power LED on the controller's front panel lights up when the power switch is turned on. It indicates that power is being supplied to the controller.

Motors and User Power Supply Switch and LED

This switch connects and and disconnects DC voltage to all the connected motors and to the user power supply. A green LED embedded in the switch lights up when the switch is on.

The motors switch is turned off in the following circumstances:

- To disconnect power to the motors, user power supply, and inputs without turning off the controller.
- To prevent possible motion of axes.

When the motors switch is turned off, the robot motors and all connected axes are unable to move. In addition, it disconnects the user power supply, making the controller inputs and open collector outputs inoperative.

Emergency Switch and Lamp

This switch halts all controller operations. A red lamp embedded in the switch lights up with the switch is on. When the switch is depressed, the following occurs:

- The red emergency lamp lights up.
- All running programs are aborted.
- Motor power is disconnected; all motor movement stops; the green motors LED shuts off. All the green LEDs on the rear panel shut off.
- The user power supply is shut off.
- The inputs and outputs are shut off.

When the switch is pressed again, the following occurs:

- The red emergency lamp shuts off.
- The green LED on the motors switch lights up.
- The green LEDs on the rear panel light up again.
- The controller's CPU is reset and the following appears on the screen:

```
---- RAM TEST COMPLETE.
---- ROM TEST COMPLETE.
SYSTEM READY!
>_
```

☞ *The robot must be homed before work can resume following an Emergency.*

User Power Supply Terminals

The user power supply allows external devices in the user's applications to receive power from the controller. The controller user power supply has four terminals:

- Two +12VDC, 2A regulated power supply
- Two safety ground

When the motors switch is turned off, it also disconnects the user power supply.

Input and Output Terminals and LEDs

Controller-A is equipped with an I/O board which allows you to individually configure the inputs and open collector outputs to operate in either negative (NPN) or positive (PNP) logic. The controller is factory-configured for operation in NPN mode.

Refer to the section, "Adjustments and Repairs," in Chapter 8 for instructions on altering the I/O logic mode.

Inputs

The controller's inputs allow the robotic system to receive signals from external devices in the robot's environment. The controller has 16 input terminals and four ground connection points, as shown in Figure 3-3.
All inputs are coupled to the controller system with opto-couplers.

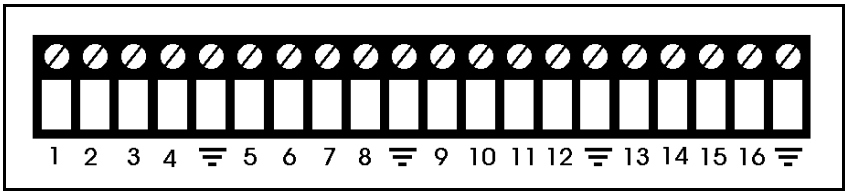


Figure 3-3: Input Terminals

Inputs can be operated in either of two modes:

- Negative logic mode (NPN) ; default mode:
  - ON is defined as low voltage (less than 1.5VDC or ground).
  - OFF is defined as high voltage (+5VDC to +24VDC).
- Positive logic mode (PNP):
  - ON is defined as high voltage (+5VDC to +24VDC),
  - OFF is defined as low voltage (less than 1.5VDC or ground).

When the motors switch is turned off, it disconnects the user power supply, making the controller inputs and open collector outputs inoperative.

To simulate the operation of an input when no device is connected, short the input manually; use a wire or an unraveled paper clip, for example.

- When the input is operating in NPN mode, short the input by connecting it to a ground connector.
- When the input is operating in PNP mode, short the input by connecting it to the user power supply.

Outputs

The controller's outputs allow the robotic system to transmit signals to external devices in the robot's environment. The controller has 4 relay outputs and 12 open collector outputs.

Relay Outputs 1-4

Outputs 1 to 4, shown in Figure 3-4, include relays in their final stage. Each relay includes three contact points:

- Common Tab (COM)
- Normally Closed Tab (NC)
- Normally Open Tab (NO)

Maximum voltage allowed: 24VDC
Maximum current allowed: 4A

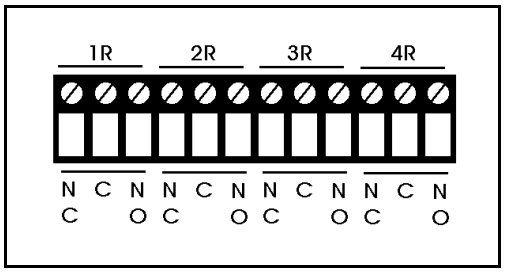


Figure 3-4: Relay Output Terminals

Figure 3-5 shows the ON and OFF states of a relay output.

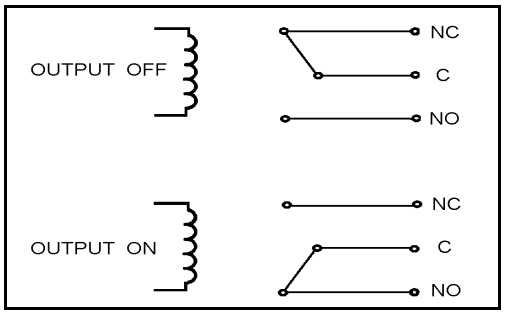


Figure 3-5: Relay Output States

Open Collector Outputs 5-16

Outputs 5 to 16, shown in Figure 3-6, include a transistor with an open collector in their final stage. These outputs must be connected to a load.

Never connect open collector outputs directly to a power supply or ground.

When using an inductive load, such as a solenoid or relay, connect a protection diode across the load. You may connect an open collector output directly to an input.

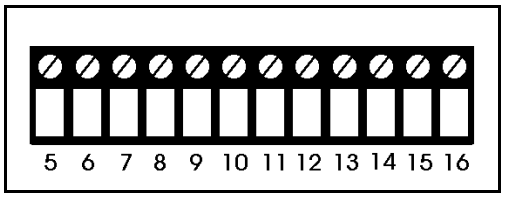


Figure 3-6: Open Collector Output Terminals

Open collector outputs can be operated in either of two modes:

- Negative logic mode (NPN); default mode:
  - ON is defined as low voltage (0.3VDC or less)
  - OFF is defined as V (refer to Figure 3-7)
- Positive logic mode (PNP):
  - ON is defined as +12VDC
  - OFF is defined as 0 volts (refer to Figure 3-8)

Maximum voltage allowed: 24VDC
Maximum current allowed: 0.5A

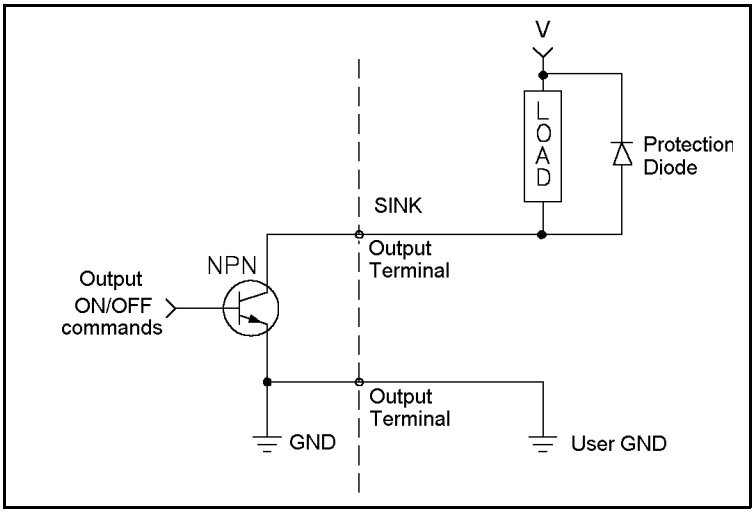


Figure 3-8: Open Collector Output: NPN Mode

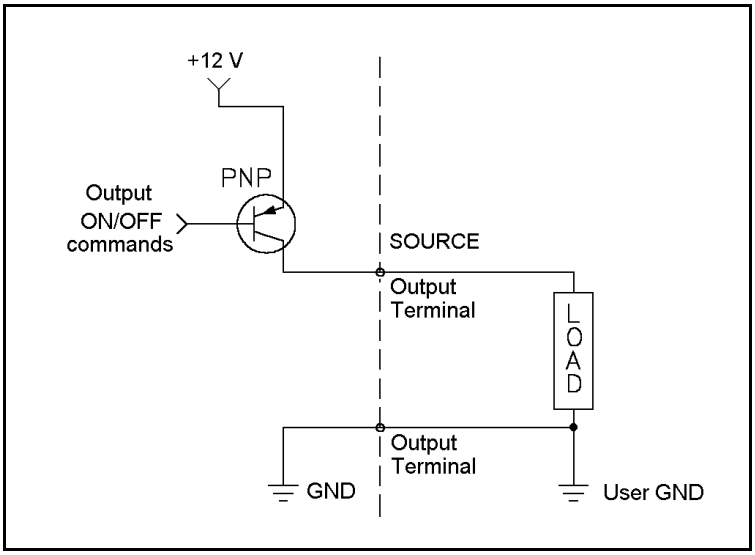


Figure 3-7: Open Collector Output: PNP Mode

Input and Output LEDs

16 yellow LEDs, corresponding to outputs 1–16, light up when the outputs are ON.

16 orange LEDs, corresponding to inputs 1–16, light up when the inputs are ON.

CHAPTER 4

Installation

Before installing the **SCORBOT-ER Vplus**, be sure you have read and understood the safety instructions and warnings detailed in Chapter 1.

Preparations

Be sure you have ample space to set up the robotic system, as shown in Figure 4-1.

1. Set up the **SCORBOT-ER Vplus** on a sturdy surface with a minimum 700mm of free space all around the robot.

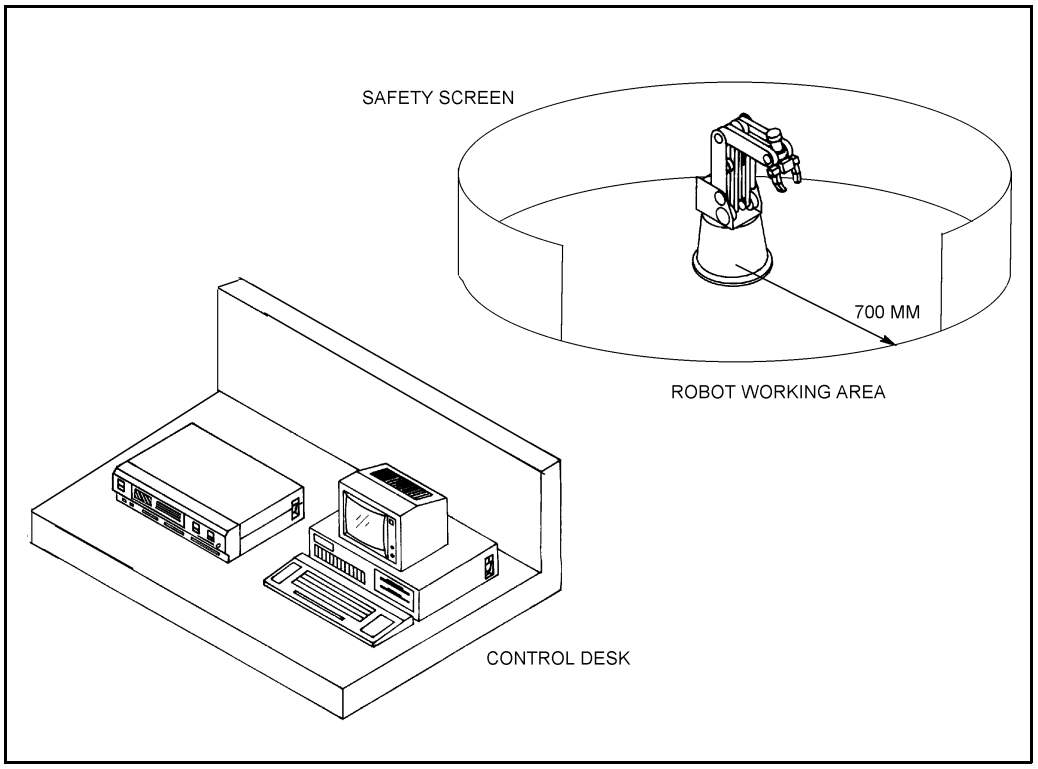


Figure 4-1: SCORBOT-ER Vplus Installation

2. Fasten the base of the robot arm to the work surface with at least 3 bolts 120° apart, as shown in Figure 4-2.

   Robot Base ∅ 240 mm (9.49")
   Pitch Circle ∅ 207 mm (8.15")
   Hole (6 off) ∅ 8.5 mm (0.33")

   Make sure the robot is securely bolted in place. Otherwise the robot could become unbalanced and topple over while in motion.

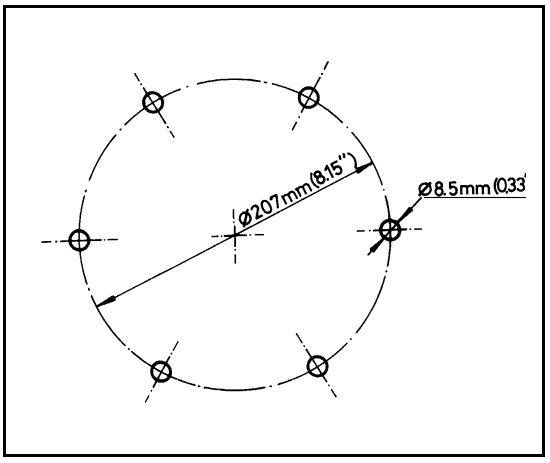


Figure 4-2: Robot Base Plate Layout

3. Set up a guardrail, rope or safety screen around the robot's operating area to protect both the operator and bystanders.

4. Place the controller and computer on a sturdy surface at a safe distance from the robot—well outside the robot's safety range.

Cable Connections

Be sure to verify that the controller's voltage setting matches your voltage supply before you connect the controller to the AC power outlet.

1. Install and configure your computer/terminal and monitor according to the manufacturer's instructions.
2. Connect the computer power cable to an AC power source.

   It is recommended, though not imperative, that you connect the computer to an AC power source other than the one used by the controller.

   Make sure the power switch on the computer and the controller is in the OFF position before you continue to the next step.

   For the following steps, refer to Figure 4-3.
3. Connect the gripper path cable (D9 connectors) to both the gripper port and the axis 6 driver port.
4. Connect the robot cable (D50 connector) to the controller.
5. Connect the RS232 cable (D25 connector) to the RS232 port on the controller and to the RS232 port on the computer. You may use either COM1 or COM2 on the computer.

   If your computer's COM port requires a D9 connector, use a standard D25-D9 adapter to connect the RS232 cable to your computer.

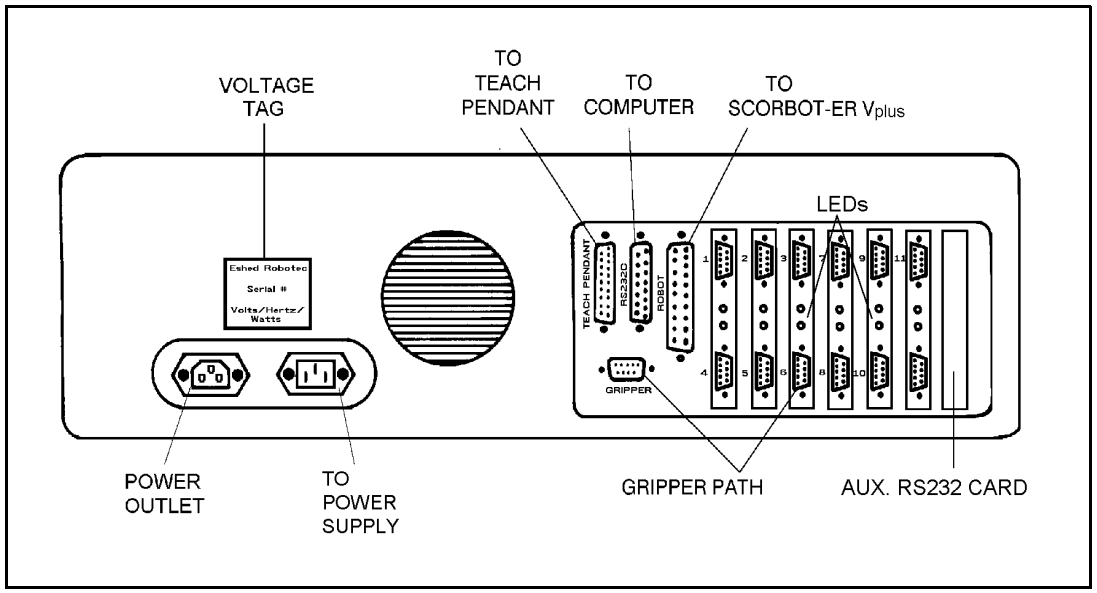


Figure 4-3: Controller Rear Panel

6. If an auxiliary RS232 communication card is installed in the controller, make the following cable connections:

   - Connect the cable's D37 connector to the auxiliary RS232 port on the controller.
   - The auxiliary card may have a cable with either two or eight D25 connectors. Connect the cable's D25 connectors to the corresponding COM ports on the other controllers or computers. (If any of these COM ports requires a D9 connector, use a standard D25-D9 adapter to connect the RS232 cable).

   To install an auxiliary RS232 communication card in your controller, follow the instructions described in the section, "Adjustments and Repairs," in Chapter 8.

7. If you will be using a teach pendant, connect it to the Teach Pendant port (D25 connector) on the controller.
8. If you will be operating additional axes by means of the controller, connect them at this time. Refer to the following section, "Peripheral Axes."
9. When you have completed all cable connections, tighten all retaining screws on all the connectors.
10. Make sure the controller's power switch is off. Then plug the controller's power cable into AC power supply outlet.
11. You may now proceed to the section, "Power On."

Peripheral Axes

When the controller is configured for operation with **SCORBOT-ER Vplus**, *do not connect peripheral axes to the axis driver connectors labeled 1, 2, 3, 4, 5 and 6*, which are reserved for the robot axes and gripper.

Although axis 6 is reserved by default for an electrical gripper, it can be used to drive a peripheral device.

- Make sure the gripper path cable is disconnected from both the Gripper and the Axis 6 connectors on the controller. You may then connect the peripheral device to the Axis 6 driver.
- Use the **ACL** command CONFIG to configure Axis 6 for a peripheral axis.

For information on installing additional driver cards, refer to the section, "Adjustments and Repairs," in Chapter 8.

For instructions on configuring **Controller-A** for use with peripheral devices, refer to the ***ATS*** *Reference Guide*.

Power On

1. Once you have made all the required hardware connections, you can power on the controller.

   - Turn on the controller's power switch.
   - Turn on the controller's motors power switch.

   The green (power and motor) LEDs light up.

2. Turn on your computer, and boot using your own DOS.

   If your computer does not "wake up", disconnect the RS232 cable, then power on the computer, and then reconnect the RS232 cable

3. If your computer has a hard drive, make a directory for **ATS**, and copy the files from the **ATS** distribution disk to that directory.

   If your computer does not have a hard drive, make a backup copy of the **ATS** disk. Keep the original disk in a safe place, and use the copy for operation.

4. Make the **ATS** directory or disk drive the default.

   At the DOS prompt, activate **ATS**.

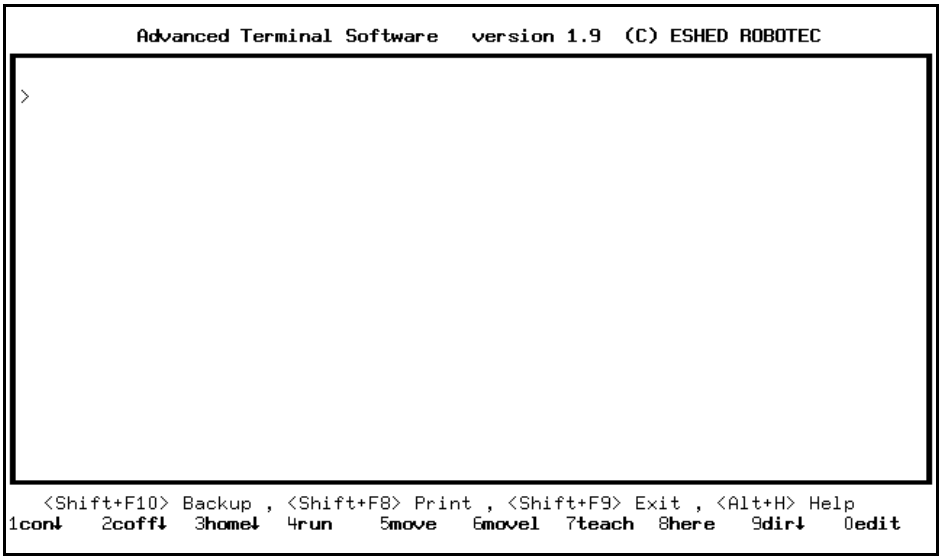
   If the controller is connected to computer port COM1 (default), type:

   ```
   ats  <Enter>
   ```

   If the controller is connected to computer port COM2, type:

   ```
   ats  /c2 <Enter>
   ```

5. Once the software has loaded, the **ATS** main screen will appear on your monitor:



Press <Enter> to receive the > prompt, if it is not already displayed.

If you have connected a teach pendant, the TP display will show:

6. You can now communicate directly with the controller.

   Perform the controller configuration, as described in the following section.

Controller Configuration

This section describes the short-form controller configuration which loads default parameter settings according to your responses to the prompts. The procedure described below should be sufficient for you to begin operating the system.

☞ *If the controller has already been in operation, be sure to back-up all data before initiating this configuration procedure.*

The configuration procedure is initiated from the **ATS** main screen by pressing the hot-key combination:

<Ctrl> + <F1>

You are prompted:

```
Controller Configuration
ARE YOU SURE (Y/N)? N
```

Press Y to proceed with the configuration, or
Press N or <Enter> to cancel the configuration.

You are prompted by a short series of Controller Configuration options.

Make sure you select the proper options for your installation.
Incorrect selections may result in damage to your equipment.

```
Robot type: ER V / ER-Vplus / ER-VII / OTHER
```

This defines the robot which is connected to the controller.

Use the left and right arrow keys to highlight the name of the robot which is connected to the controller. Then press <Enter> to accept.

When a robot (not "OTHER") is selected, the controller reserves axis 6 (the first available axis after the robot axes) for an electrical servo gripper.

```
How many axes are installed (8)? ..
```

This defines the number of axes which can be driven by the controller.

Press <Enter> to accept 8 axes (default), or
Type any other valid number and press <Enter>.

```
Is expanded memory installed (Y/N) Y
```

Press Y or <Enter>if controller has 128K RAM (default), or
Press N if controller has 32K RAM .

```
Does the controller have an auxiliary RS232 board?(Y/N)? N
```

Press Y if the auxiliary multiport RS232 board is installed in your controller, or
Press N or <Enter> if the board is not installed (default).

```
Working directory is:  c:\ATS
Is this correct (Y/N)?Y
```

The first time this prompt appears, it shows the DOS directory from which the **ATS** software was activated.

The Working directory must be the directory which contains the parameter files and the **SCORBASE** program file (.CBU files).

If you change the directory definition, it is written to a file named SETUP.DIR. Thereafter, whenever **ATS** is loaded, the Working directory is set according to the definition in the SETUP.DIR file. Similarly, the SETUP.DIR file determines the definition of the Backup directory shown in the Backup Manager screen. SETUP.DIR is updated when either the Working directory or Backup directory definition is changed.

Press N if you want to change the directory. The cursor moves to the directory line, prompting you to type and <Enter> a different directory.

Press Y if the directory is correct.

Press <Esc> if you are not sure whether the displayed directory is correct. This will cancel the configuration procedure. Press F10 to access the **ATS** Backup Manager menu to verify the proper directory definition. Or exit to DOS to verify the location of the .CBU files.

```
WARNING ! USER RAM WILL BE ERASED !!
ARE YOU SURE(Y/N)? N
```

Press Y to proceed with the configuration.
Press N or <Enter> to cancel the configuration.

After you confirm, **ATS** compares your selections with the controller's current configuration. You are warned of any differences, and again prompted to confirm the configuration.

After you again confirm, **ATS** performs the configuration and loads the proper parameter files in accordance with your selections.

For complete instructions on the short-form controller configuration, including configuration for use with the **SCORBASE** software, refer to the ***ATS** Reference Guide*.

For definitions not included in the short-form configuration procedure—such as axes in control group C, a robot of another make, and memory allocation— you will need to use the **ACL** command CONFIG. Refer to the ***ACL** Reference Guide*.

CHAPTER 5

Operating Methods

SCORBOT-ER Vplus can be programmed and operated in a number of ways. This chapter introduces the robotic software and the teach pendant functions. Software and teach pendant operation is described in other chapters of this manual, and in the other manuals supplied with the system.

Software

ACL

ACL, Advanced Control Language, is an advanced, multi-tasking robotic programming language developed by Eshed Robotec. **ACL** is programmed onto a set of EPROMs within **Controller-A**, and can be accessed from any standard terminal or PC computer by means of an RS232 communication channel.

ACL features include the following:

- Direct user control of robotic axes.
- User programming of robotic system.
- Input/output data control.
- Simultaneous, synchronized and interactive program execution; full multi-tasking support.
- Simple file management.

ACL is described fully in the ***ACL** Reference Guide*.

ATS

ATS, Advanced Terminal Software, is the user interface to the **ACL** controller. **ATS** is supplied on diskette and operates on any PC host computer. The software is a terminal emulator which enables access to **ACL** from a PC computer.

ATS features include the following:

- Short-form controller configuration.
- Definition of peripheral devices.
- Short-cut keys for command entry.
- Program editor.
- Backup manager.
- Print manager.

ATS is described fully in the ***ATS*** *Reference Guide*.

SCORBASE

SCORBASE is a robotic control software package which can be used with **Controller-A**. Its menu-driven structure and off-line capabilities facilitate robotic programming and operation.

SCORBASE is supplied on diskette and operates on any PC system. **SCORBASE** communicates with **ACL**, the controller's internal language, by means of an RS232 channel.

Levels 1, 2 and 3 of the **SCORBASE** software can be ordered separately, and are recommended for those who wish to learn robotic programming from the most basic stages.

SCORBASE is described fully in the ***SCORBASE*** *Level 5 Reference Guide*.

Teach Pendant

The teach pendant is an optional device.

The teach pendant (TP) is a hand-held terminal, used for controlling the robot and axis connected to **Controller-A**. The teach pendant is most practical for moving the axes, recording positions, sending the axes to recorded positions, and activating programs. Other functions can also be executed from the teach pendant.

The teach pendant's display panel is a 2-line, 32 character liquid crystal display (LCD). It shows the current status of the controller, the current user command, and system messages.

The teach pendant has 30 function keys. These functions are described in this chapter. Many of the command keys on the teach pendant are **ACL** commands; these commands are described fully in the ***ACL** Reference Guide*.

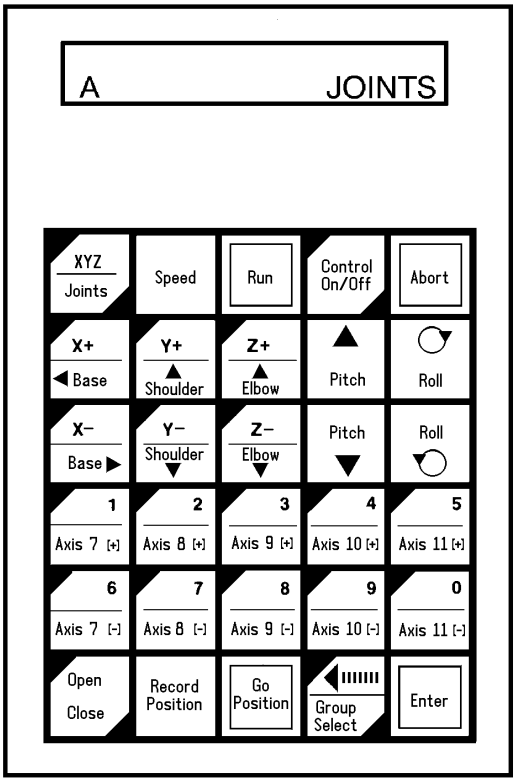
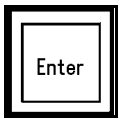


Figure 5-1: Teach Pendant

Keypad Functions

The teach pendant's keypad has 30 color-coded keys. Most of the keys are multi-functional; for example, some keys include both an axis drive command and a numeric function. The controller recognizes the keys from the order in which they are pressed. Thus, the numeric function will be active only if a function such as SPEED, RUN, or MOVE has been keyed in first; otherwise, the axis drive command will be active.

Following are descriptions of the teach pendant's keys and instructions for activating them. Bulleted items indicate the different functions of multi-functional keys.



Accepts and/or executes the command which has been entered.

Starts execution of a program following a Run command.

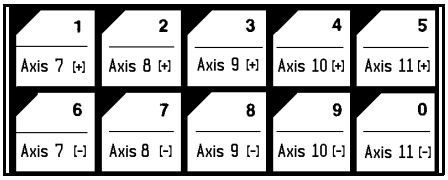
A toggle key. Switches the command mode between Joints and Cartesian (XYZ).

- When used following a numeric function, this key acts as a backspace function; it cancels the last numeric entry and moves the cursor one position to the left.
- Enables TP control of a specific axis group.

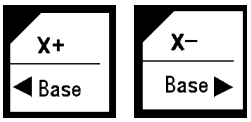
  Successively press for group A, group B, group C, and again for group A, and so on. When group C is displayed, enter the axis number on the numerical keys. Then press **Enter**.

  The Record Position and Speed functions apply only to the currently selected group.

A toggle key. Enables (CON) and disables (COFF) control of the selected group.

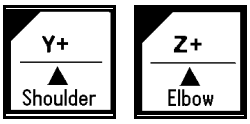


- The **Axis** keys move axes 7 through 11 in two directions.
- The numeric keys are operative if one of the following functions has been activated: **Speed**, **Run**, **Record Position**, **Go Position**, **Group Select**.



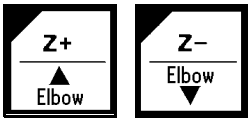
In Joint mode: the **Base/X** keys move the base axis in two directions.

In XYZ mode: the **Base/X** keys move the TCP (tip of gripper) along the X-axis; Y and Z coordinates do not change.



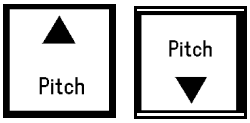
In Joint mode: the **Shoulder/Y** keys move the shoulder axis in two directions.

In XYZ mode: the **Shoulder/Y** keys move the the TCP (tip of gripper) along the Y-axis; X and Z coordinates do not change.



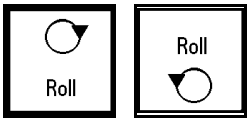
In Joint mode: the **Elbow/Z** keys move the elbow axis in two directions.

In XYZ mode, the **Elbow/Z** keys move the TCP (tip of gripper) along the Z-axis; X and Y coordinates do not change.

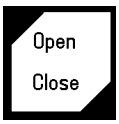


In Joint mode: the **Pitch** keys move the TCP (tip of gripper) up or down, without moving the other axes.

In XYZ mode: the **Pitch** keys move three axes (shoulder, elbow and pitch) in order to change the pitch angle without changing the position of the TCP (tip of gripper).



In both Joint and XYZ modes: the **Roll** keys move the roll axis in two directions.



A toggle key. Opens and closes the electrical gripper.

Sets the speed of manual axis movement of the current axis control group; that is, group A, B, or C. The speed is defined as a percentage (1-100) of maximum speed.

Press **Speed**. The current speed is displayed.

Press **Enter** to accept the displayed default speed. Or use the numerical keys to enter a different speed, and press **Enter**.

Defines and records a position.

Only numerical position names, of up to five digits, can be entered from the TP. The position is defined for the currently active group, and receives the current values of the axes in that group.

Press **Record Position**. Then press up to five digits for the position name. Then press **Enter** to record the position coordinates.

If you use a position name which has already been defined, the new coordinates will overwrite the existing ones.

This command is also used to record positions in a vector. The vector must first be attached to the teach pendant by means of the **ACL** command ATTACH.

Moves the axes to a target position.

Press **Go Position**. Then use the numeric keys to enter the position name. Then press **Enter** to execute the move.

In Joint mode: robot movement is by joints.

In XYZ mode: robot movement is linear.

To send the axes to their home position, enter the following commands:

Go Position 0 sends all the axes of group A to their HOME position.

Go Position 00 sends all the axes of group B to their HOME position.

Executes a program.

Press **Run**. Then press the program's identity number on the numerical keys. The program name will be displayed in brackets. Then press **Enter** to begin program execution.

The controller automatically assigns an ID number to each user program. The **ACL** command DIR lists the programs and their assigned (IDENTITY) number.

Aborts execution of all running programs. Stops movement of the robot and all peripheral axes.

The Display Panel

The LCD panel shows the current status of the controller, the current user command, and system messages.

A resident note shows the coordinates system currently active: **JOINTS** or **XYZ**.

Another resident note shows the currently active group: **A**, **B**, or the ***number*** of one of the independent axes in control group C.

This page intentionally left blank.

CHAPTER 6

Operating the Robot

This chapter introduces you to the basic commands for operating the **SCORBOT-ER Vplus** robot by means of both the **ACL** software and the teach pendant.

DIRECT Mode

This chapter describes the operation of the robotic system when it is functioning in the DIRECT mode. When the system is in DIRECT mode, the user has direct control of the axes, and the controller executes commands as soon as they are entered by the user.

When in DIRECT mode, the screen prompt appears like this: >_

When the system is operating in EDIT mode, commands are entered into a user program, which can be saved and executed at a later time. Program editing procedures are described in Chapter 7.

Manual Mode

Manual mode is available when the system is in DIRECT mode. The Manual mode enables direct control of the robot axes when a teach pendant is not connected.

When using the keyboard to perform some of the procedures described in this chapter, the system must be in Manual mode.

To activate Manual mode, hold the <Alt> key and press the character M:

```
<Alt> + m
```

The system will respond in one of the following ways:

```
MANUAL MODE!          MANUAL MODE!
>_                    >_
JOINT MODE            XYZ MODE
```

The system's response indicates the currently active coordinate system.

To exit Manual mode, the same command is used.

Press: **<Alt> + m**

```
EXIT MANUAL MODE...
>_
```

Using this Manual

To familiarize yourself with the system, you should read through this chapter (and the following ones) and practice entering the commands described in each section.

All operations described in this chapter can be performed from the keyboard. The steps for using the keyboard are indicated by the heading PC. The teach pendant is optional. The operations which can also be performed from the teach pendant are indicated by the heading TP.

This manual uses the following typographical conventions:

Descriptions of PC operation show user entries in bold, lowercase text. System responses are shown in capital letters. (The actual screen display may be different.) For example:

home <Enter>

```
WAIT!! HOMING...
```

The system is not case-sensitive. You may use either uppercase and lowercase characters to enter commands and data.

Descriptions of TP operation show the teach pendant keys which the user must press. System responses are shown in boxed capital letters. For example:


```
CONTROL  ENABLED
A        JOINTS
```

Activating the Sytem

Activate the system and load the **ATS** software, as described in the section, "Power On," in Chapter 4.

Homing the Robot and Peripheral Axes

The location of the robot axes is monitored by encoders which track the amount of movement relative to an initial—home—position. To obtain repeatable robot performance, this reference position must be identical each time the robot is used. Thus, whenever the system is activated, the homing program, which is internally programmed into the controller, must be executed.

During the homing procedure, the robot joints move and search for their home positions, one at a time, in the following sequence:
shoulder (axis 2), elbow (axis 3), pitch (axis 4), roll (axis 5), base (axis 1), gripper (axis 6).

To find its home position, the axis is moved until the microswitch which is mounted on the joint sends a specific signal to the controller, indicating the axis is at home.

When the homing is completed, the robot assumes the position shown in Figure 6-1.

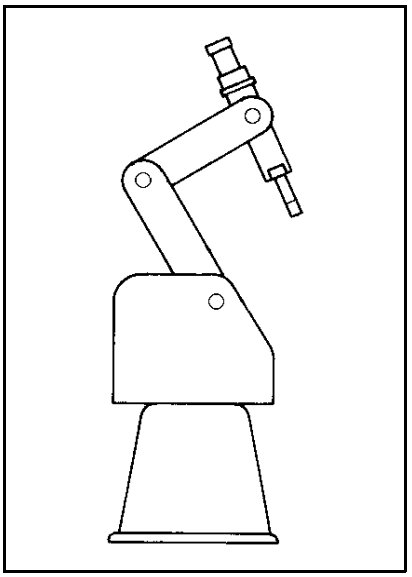
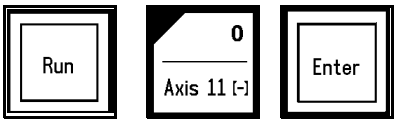


Figure 6-1: SCORBOT-ER Vplus Home Position

☞ *Before you begin the homing procedure, make sure the robot has ample space in which to move freely and extend its arm.*

TP

Press:



This instructs the controller to execute Program 0, the robot homing routine. The display panel on the teach pendant will show:

```
HOMING. . .
```

When the Home search is successfully completed, the display panel will show:

```
HOMING   COMPLETE
```

If the robot is unable to find a home position in one or more of the axes, you will see a message such as:

```
HOME FAIL [4]
```

To stop the homing while the operation is in progress, press the **Abort** key.

The peripheral axes are homed by means of the TP command **Run 00**.

PC

To home the robot axes (Group A), use the **ACL** command HOME.

Type: **home <Enter>**

```
WAIT!! HOMING...
```

If all axes reach their home postion, a message is displayed:

```
HOMING COMPLETE (ROBOT)
```

If the homing process is not completed, an error message identifying the failure is displayed:

```
*** HOME FAILURE AXIS 4
```

To stop the homing while the operation is in progress, use the abort commands:

Type: **A <Enter>**

or press: **<Ctrl>+A**

To home peripheral axes, each axis must be homed individually; for example:

Type:
```
home 7 <Enter>
home 8 <Enter>
home 9 <Enter>
```

To home an axis, such as a slidebase, which uses a hard stop rather than a microswitch, use the **ACL** command HHOME.

Type: **hhome 8 <Enter>**

Coordinate Systems

The **SCORBOT-ER Vplus** can be operated and programmed in two different coordinate systems: Joint and Cartesian (XYZ) coordinates.

Cartesian (XYZ) Coordinates

The Cartesian, or XYZ, coordinate system is a geometric system used to specify the position of the robot's TCP (tool center point=tip of gripper) by defining its distance, in linear units, from the point of origin (the center bottom of its base) along three linear axes, as shown in Figure 6-2.

To complete the position definition, the pitch and roll are specified in angular units.

When robot motion is executed in XYZ mode, all or some of the axes move in order to move the TCP along an X, Y or Z axis.

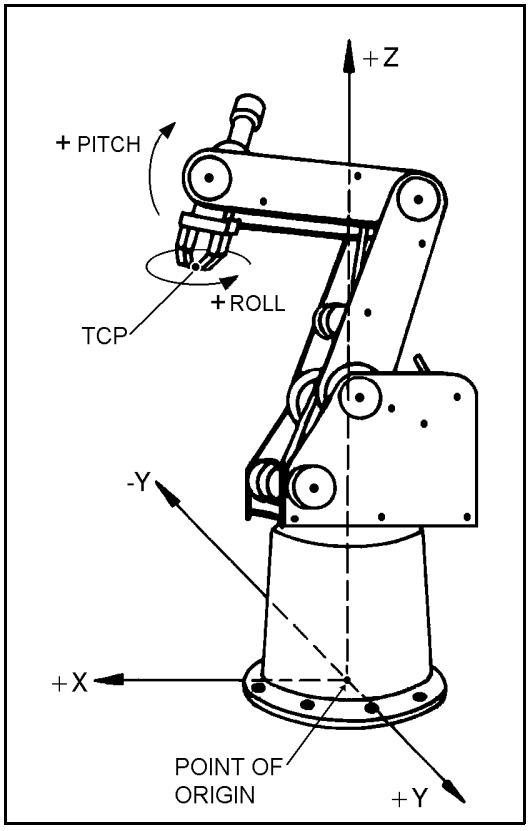


Figure 6-2: Cartesian Coordinates

Joint Coordinates

Joint coordinates specify the location of each axis in encoder counts. When the axes move, the optical encoders generate a series of alternating high and low electrical signals. The number of signals is proportional to the amount of axis motion; the controller counts the signals and determines how far an axis has moved. Similarly, a robot movement or position can be defined as a specific number of encoder counts for each axis, relative to the home position, or another coordinate.

When robot motion is executed in Joint mode, individual axes move according to the command.

If any peripheral devices are connected to the robotic system, the position of their axes is always stated in encoder counts.

TP

To toggle between the two coordinate systems:

Press:

A	JOINTS

Press again:

A	XYZ

The display reflects the currently active coordinate system. Manual movement of the axes will be executed according to the currently active coordinate system.

PC

To select a coordinate system from the keyboard, you must first activate Manual mode.

To activate the Joint coordinate system:

Press: **j**

```
JOINT MODE
```

To activate the XYZ coordinate system:

Press: **x**

```
XYZ MODE
```

Servo Control

The controller must be in the servo control (CON) state for the axes to execute movement commands.

Activating the Home routine will activate CON.

Certain events, such as impact, overheating (thermic error), or activation of the Emergency switch, will automatically switch off the servo control state (COFF). CON must be activated to resume motion and servo control.

While the controller is in the COFF state, you cannot operate the axes.

TP

To toggle servo control on and off:

Press: 


```
CONTROL   ENABLED
```

Press again: 


```
CONTROL  DISABLED
```

When Control On/Off is activated from the teach pendant, the CONTROL ENABLED/CONTROL DISABLED message also appears on the computer screen.

PC

If Manual mode is active you can enable and disable control from the keyboard.

Press: **c**

```
CONTROL ENABLED
```

Press: **f**

```
CONTROL DISABLED
```

The commands C and F enable and disable control of *all axes* which are connected to the controller.

If Manual mode is not active, you can use the **ACL** commands CON and COFF.

Type: **`con <Enter>`** Enables control of all axes.

Type: **`coff <Enter>`** Disables control of all axes.

The format can be altered to enable and disable control of specific groups of axes; for example:

`cona` Enables control of robot axes (Group A).

`coffb` Disables control of peripheral axes (Group B).

`con 9` Enables control of axis 9 (Group C).

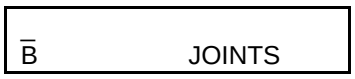
Axis Control Groups

By default, the controller assumes the five robot axes (Group A) are under servo control. The Group Select key allows you to switch control to peripheral axes (Group B), or to an independent axis (Group C).

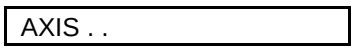
TP

To select the axis control group:

Press: 



Press again: 



When selecting an independent (Group C) axis, you must also key in the axis number followed by **Enter**.

Continue pressing this key until the desired axis group is displayed.

PC

ACL does not have a command for selecting the axis control group. The specific format of each command indicates the axis control group.

Moving the Axes

XYZ and Joint Movements

When the coordinate system is set to the XYZ mode, movement commands cause linear motion of the TCP (tip of gripper) along the X, Y and Z axes, while maintaining the angles of the pitch and roll relative to the robot's point of origin.

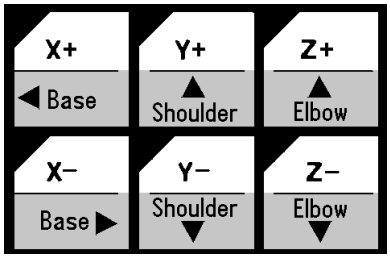
When the coordinate system is set to the Joint mode, the robot responds to movement commands by moving from one defined point to another.

Peripheral axes always move according to Joint coordinates.

TP

When in XYZ mode, the controller recognizes the Cartesian functions of the teach pendant keys.

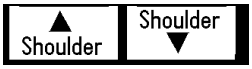
When in Joint mode, the controller recognizes the joint functions (shaded in diagram) of the teach pendant keys.



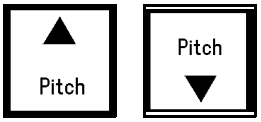
The teach pendant offers the easiest method for moving the robot arm. You simply press an axis movement key, and the robot moves. When you release the key, movement stops.

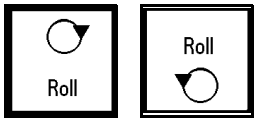
Before you press the keys shown below, make sure JOINTS, Group A, and Control On are active. Move the axes of the robot, in both directions.

Press: 

Press: 

Press: 

Press: 

Press: 

Before you press the keys shown below, make sure XYZ appears on the teach pendant display. Watch how the keys now affect the movement of the TCP.

Press: 

Press: 

Press: 

PC

To directly control movement of the robot axes from the keyboard, Manual mode and Control On must first be activated. The keys listed below are then used to move the robot.

The axes will move as long as the activating key is depressed, until a fixed stop is reached. The gripper will either open completely or close completely.

In Joint mode, the keys produce the following movements:

Press:		
	1, Q	Move axis 1 (base)
	2, W	Move axis 2 (shoulder)
	3, E	Move axis 3 (elbow)
	4, R	Move axis 4 (wrist pitch)
	5, T	Move axis 5 (wrist roll)
	6, Y	Closes/Opens **electrical gripper** (axis 6)

In XYZ mode the following changes in manual movement occur:

Press:		
	1, Q	TCP moves along X+ and X– axes.
	2, W	TCP moves along Y+ and Y– axes.
	3, E	TCP moves along Z+ and Z– axes.
	4, R	Pitch moves; TCP maintains position.

All other movements are the same as in Joint mode.

In XYZ mode, moving the robot to positions at the maximum range of reach may result in jerky movements. Use Joint mode to reach these positions.

While moving the arm, you may alternate between XYZ and Joint modes as often as required.

If peripheral axes are connected, the following keys are also used:

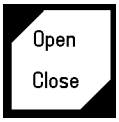
Press:		
	7, U	Move axis 7
	8, I	Move axis 8
	9, O	Move axis 9
	0, P	Move axis 10
	–, [	Move axis 11

Activating the Gripper

In response to the commands to open and close, the electrical gripper goes completely from one state to the other.

TP:

Press:



The **Open/Close** key toggles the gripper between its open and closed states. If the gripper was open it will now close, and vice versa. Repeat the command.

PC:

When Manual mode is active, the following keys activate the gripper.

Press: **Y** Opens the gripper.

Press: **6** Closes the gripper.

When Manual mode is not active, the **ACL** commands OPEN and CLOSE are used.

Type: **open <Enter>**

Type: **close <Enter>**

Pneumatic grippers or devices are controlled by means of **ACL** output commands.

To activate a pneumatic gripper or end effector from the teach pendant, **ACL** output commands must be written to two program (one for opening the gripper and one for closing it). Each program can then be called from the teach pendant, resulting in activation of the pneumatic gripper or device.

For more information, refer to the section on I/O programming in Chapter 7.

Setting the Speed

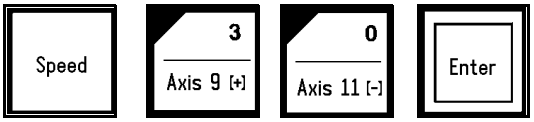
TP

The speed of the robot during **Go Position** movements controlled from the teach pendant is defined as a percentage of maximum speed. Speed defined as 100 gives the robot maximum speed, while a speed of 1 is the minimum. When the system is first turned on, the default speed is set at 50, approximately half the robot's maximum speed.

The speed of the robot during *manual* movements controlled from the teach pendant is relative to the speed setting, and much slower than Go Position movements.

Use the teach pendant to set the robot's speed to a speed of 30%, for example:

Press:



All Go Position movement commands will be executed at a speed of 30, until a different speed is entered.

PC

When Manual mode is active, use the key S to set the speed of manual movement.

Press: **s**

```
SPEED..
```

You are prompted for a speed value—a percentage of the maximum speed. Type a number between 1–100, and press <Enter>.

When Manual mode is not active, the **ACL** command SPEED is used to define the speed at which movements are executed. For example:

`speed 50`	Sets speed movements of Group A axes to 50% of maximum speed.
`speedb 20`	Sets speed of movements of peripheral axes (Group B) to 20% of maximum speed.

Defining and Recording Positions

Defining a position reserves space in controller memory, and assigns it a name.

Recording a position writes coordinate values to the allocated space in controller memory.

Two types of position names are possible:

- Numerical names (such as 3, 22, 101) of up to five digits.
  Positions with this type of name do not need to be defined before they are recorded by means of the teach pendant; the position recording command automatically defines and records positions with numerical names.
- Alphanumeric names (such as P, POS10, A2). The name may be a combination of up to five characters, and should begin with a letter. These positions cannot be accessed from the teach pendant.

Positions may belong to a vector; that is an array of positions identified by a specific name and an index; for example, PVEC[1] and PVEC[5] are positions in a vector named PVEC. When a vector is attached to the teach pendant (by means of the ACL command ATTACH), vector positions can be accessed from the teach pendant by means of their index number.

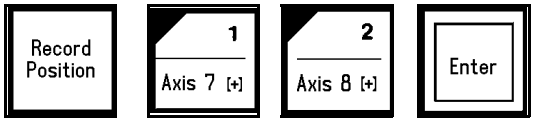
If you accidently record coordinates for position 0, execute the Home program. The homing routine records the proper coordinates for position 0.

TP

The teach pendant allows you to simultaneously define and record a position.

To record a robot position, first be sure the Group A is selected. Then use the axis movement keys to bring the robot to any location. Record this as position 12.

Press:



DONE

Move the robot to another location and record it as position 13.

The Record Position key records the position of the currently active axis control group (A, B or an independent axis) in *joint coordinates*.

If you want to define the location of both the robot and the peripheral axes, you must record two positions, one for each group.

PC

To define and record positions from the keyboard, you must first exit Manual mode.

Use the **ACL** command DEFP to define a robot position. For example:

Type: `defp A1 <Enter>` Defines position A1 for the robot.

When a position is defined, it is assigned to a specific axis control group. By default, it is assigned to the robot (Group A) axes. To define a position for Group B, or an independent axis, the command format determines the group to which the position is dedicated.

Type: **`defpb B24 <Enter>`** Defines position B24 for Group B.

`defpc C3 10 <Enter>` Defines position C3 for axis 10.

Define three robot positions:

Type:
```
defp A31
defp A32
defp A33
```

The **ACL** command HERE records a position—*in joint coordinates*—according to the current location of the axes.

Remember to activate Manual mode before starting motion, and to exit Manual mode when the motion is completed. Also be sure the position is defined before you attempt to record it.

Move the robot to any location, and record its coordinates for position A31.

Type: **`here A31 <Enter>`**

Move the robot two more times, and record coordinates for positions A32 and A33.

If you attempt to record a position which has not been defined (for example HERE A34), the system will display an error message.

If you specify a name of a position which has already been recorded (for example, HERE A31), the HERE command will will overwrite the existing coordinates with new coordinates.

The **ACL** command TEACH records a robot position—*in Cartesian coordinates*— according to user defined settings; it does not record the coordinates of the robot's current location.

Relative Positions

TP

Relative positions cannot be recorded by means of the teach pendant.

PC

The **ACL** commands HERER and TEACHR allow you to record a position as relative to another position, or as relative to the current position of the robot.

To record a position which is relative to another position by *joint coordinates*, move the robot to the relative location and record the position. For example:

Type: **herer A99 A33 <Enter>**

The coordinates of position A99 are actually offset values; that is, the difference in the encoder count at position A31 and at position A99. If the coordinates of position A31 change, position A99 will remain relative to position A31 by the same number of encoder counts.

To record a position relative to the current location of the robot by *joint coordinates*, you are prompted to enter values (encoder counts) for each of the axes. If offset values have already been recorded for this position they will appear in the brackets; otherwise the brackets are empty. For example:

Type: **here A99 <Enter>**

1--[.]>0 <Enter>	Base = no offset
2--[.]>500 <Enter>	Shoulder = 500 counts offset
3--[.]>250 <Enter>	Elbow = 250 counts offset
4--[.]>0 <Enter>	Pitch = no offset
5--[.]>0 <Enter>	Roll = no offset

The command TEACHR allows you to record a position which is relative to another position, or relative to the current position of the robot, in *Cartesian coordinates.* TEACHR can be easily used to maintain a vertical offset (along the Z-axis) between two positions; for example:

Type: **>teachr over**

```
X [.] > 0
Y [.] > 0
Z [.] > 500
P [.] > 0
R [.] > 0
```

Relative position OVER will always be 50mm above the current position of the robot.

ACL has a number of commands for recording position coordinates; they are detailed in the ***ACL** Reference Guide*, and will not be discussed in this manual.

Listing Positions

PC

To see a list of the defined positions, use the **ACL** command LISTP.

Type: **`listp <Enter>`**

The list of defined positions is displayed on the screen. Positions 12, 13, A31, A32, A33 and A99 should now appear in the list.

To view the coordinates of position A31, use the **ACL** command LISTPV.

Type: **`listpv A31 <Enter>`**

Position coordinates are displayed on the screen in the following manner.

```
1:0        2:1791     3:2746     4:0        5:-1
X:1690     Y:0        Z:6011     P:-636     R:-1
```

Two sets of values are displayed for robot positions:

- The first line shows the joint coordinates; defined in encoder counts.
- The second line shows the Cartesian (XYZ) coordinates. X, Y and Z are defined in tenths of millimeters; P (Pitch) and R (Roll) are defined in tenths of degrees. For example:

  `Z: 6011` Z = 601.1mm

  `P:-636` P =–63.6°

Deleting Positions

PC

To delete positions, use the **ACL** command DELP.

Type: **`delp A99 <Enter>`**

```
DO YOU REALLY WANT TO DELETE THAT POINT? (YES/NO)>_
```

Type: **`yes  <Enter>.`**

```
A99 DELETED.
```

To prevent accidental deletion of a position, you are required to respond by entering the entire word “yes”, followed by <Enter>. Entering any other other character, including Y, is regarded as “no.”

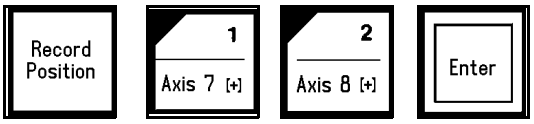
Moving to Recorded Position

Once a position has been recorded, you can easily send the robot (or other devices connected to the controller) to that position. Depending on the currently active coordinate system, the movement of the robot (Group A) will be either point to point (in Joint mode) or along a linear or curved path (in XYZ mode).

TP

Assuming the robot is at position 13, send the robot back to position 12.

Press:



DONE

Use the command **Go Position 0** to send all the axes of group A to the home position.

PC

Use the **ACL** command MOVE to send the robot to a position.

Assuming the robot is at home, send the robot to position A31.

Type: **`move A31 <Enter>`**

In this command the robot moves at the current speed setting.

The MOVE command may contain a duration parameter, which is defined in hundredths of a second. To send the robot to position A32 in 10 seconds:

Type: **`move A32 1000  <Enter>`**

You can use the PC to move to positions recorded by the TP.
Alternately, you can use the TP to move to positions recorded by means of the PC, providing the positions are defined by ***numerical*** names. For example:

Type **`move 13 <Enter>`**

Linear Movement

To move the TCP in a straight path, use the **ACL** command MOVEL.

For example, send the robot from the home position to position A33.

Type: **`move 0 <Enter>`**

Type: **`movel A33 <Enter>`**

Circular Movement

To move the TCP along a curved path, use the **ACL** command MOVEC, use the **ACL** command MOVEC.

You must specify two positions for MOVEC. Otherwise there are infinite possibilities for defining the curve. For example, send the robot from the home position to position A31, via position A32.

Be careful when using this command. For the first attempt, set the speed to a low setting, such as 20.

Type: **`move 0 <Enter>`**

Type: **`speed 20 <Enter>`**

Type: **`movec A31 A32 <Enter>`**

Reset the speed to 50 when you have completed the movement.

CHAPTER 7

Programming with ACL

This chapter serves as a tutorial to help you become familiar with program editing. To learn how to write and edit a program, you should follow, in sequence, the procedures described in this chapter.

☞ *This chapter introduces you to the basic commands for programming the **SCORBOT-ER Vplus**. Many more commands and formats are available in the **ACL** language. Refer to the **ACL** Reference Guide for complete lists and descriptions of editing functions and **ACL** commands.*

*For additional instruction in the procedures introduced in this chapter, the **ACL** Laboratory Manual (catalog #100039) is recommended.*

EDIT Mode

So far you have learned to operate the robot in the DIRECT mode, in which all commands are executed the moment you press <Enter>.

To write programs which will be executed by the robotic system, you will use the EDIT mode.

Whenever the EDIT mode is active, the screen shows the current program line number and a prompt, such as this: `143:?_`

The controller assigns the line numbers; they are not user definable.

Help

Quick, on-line help is available while you are working with **ACL**.
Simply enter the command HELP.

A list of DIRECT mode commands are displayed when in DIRECT mode;
a list of EDIT mode commands are displayed when in EDIT.

Enter the command DO HELP when in DIRECT mode in order to display the EDIT mode commands.

Creating a Program

To create a program, activate the EDIT mode by using the command EDIT, followed by the name you want to call the program. Program names are limited to five characters; for example:

Type: **edit aaa <Enter>**

```
AAA NEW PROGRAM
DO YOU WANT TO CREATE THAT PROGRAM (Y/N)>
```

Type: **y <Enter>**

```
   PROGRAM    AAA
   ********************
25:?_
```

At the ?_ prompt, you can begin entering program command lines.

Writing a Program

To write a program which will send the robot to each of the positions recorded earlier, enter the following command lines:

Type:

```
moved A31 <Enter>
moved A32 <Enter>
moved A33 <Enter>
exit
AAA IS VALID
```

Although the command MOVE may be used in EDIT mode, the command MOVED is preferable. MOVED ensures that the robot will accurately reach the target position before continuing to the next command.

The commands MOVEL and MOVEC are also available in EDIT mode. As with the MOVE command, it is preferable to use the command format with the D suffix; that is, MOVELD and MOVECD.

The EXIT command is used to end the current editing session and return the system to DIRECT mode.

Running a Program

When the > prompt is displayed, it indicates the system is in DIRECT mode. To check the program you have just created, do the following:

Make sure control is enabled (CON) and the robot is at its home position.

Type: **`run aaa <Enter>`**

```
DONE
```

The robot moves to positions A31, A32 and A33, and then stops.

Program Loop

You will now edit the program and add command lines which will cause the program to run in a loop.

Program loops are created by using the companion commands, LABEL and GOTO.

- LABEL *n* marks the beginning of a routine.
- GOTO *n* sends program execution to the line which follows the corresponding LABEL.

Type: **`edit aaa <Enter>`**

```
WELCOME TO ACL EDITOR, TYPE HELP WHEN IN TROUBLE.
   PROGRAM   AAA
   ******************
25:?_
```

The prompt shows the first line of the program. Entering a new command inserts a command line at this point.

Pressing <Enter> without entering a new command simply displays and accepts the line as is, and moves the editor to the next line.

Type: **`label 1 <Enter>`**
Press: **`<Enter>`**
Press: **`<Enter>`**
Press: **`<Enter>`**
Type: **`goto  1 <Enter>`**
Type: **`exit <Enter>`**

```
AAA IS VALID
```

Displaying Program Lines

To view the program you have edited, use the command LIST, followed by the name of the program.

Type: **`list aaa <Enter>`**

```
   PROGRAM     AAA
   ********************
25:  LABEL 1
26:  MOVED 31
27:  MOVED 32
28:  MOVED 33
29:  GOTO 1
30:  END
(END)
```

END marks the end of a program; (END) marks the end of a listing. They are written by the controller; they are not entered by the user.

Halting Program Execution

Bring the robot to its home position, and then run program AAA. The robot moves to positions A31, A32 and A33 in a continuous loop, without stopping.

Since you have now created and executed a program which will run in an endless loop, this section describes the **ACL** commands which are used to halt a program during its execution.

Suspend the Program

The companion commands SUSPEND and CONTINUE, respectively, suspend execution of a program, and then restart it from the point of interruption by executing the next program command line.

Type: **`suspend aaa <Enter>`**

The robot completes the current movement command and then stops. Program AAA is now suspended.

Type: **`continue aaa <Enter>`**

The CONTINUE command causes the robot to continue moving from the point where it was halted by a SUSPEND command.

Abort the Program

To immediately abort running programs and stop all axis movement, enter the abort command in either one of the following ways:

Type: **a <Enter>**

Press: **<Ctrl>+A**

```
PROGRAM AAA ABORTED
```

Program AAA can now be reactivated only by means of the RUN command, which will start the program from the beginning.

If several programs are running, and you want to abort only one of them, following the command by the name of the specific program; for example:

a aaa <Enter>

This format aborts the specified program only after the command currently being executed has been completed.

Stop the Program

To include an abort command in a program you are editing, use the command STOP.

The STOP command will abort a program only after all axis movement commands which have already been sent to the controller (movement buffer) have completed execution.

Use the STOP command in one of the following ways:

Type: **stop aaa <Enter>** Aborts only program AAA.

Type: **stop <Enter>** Aborts all running programs.

STOP is available in EDIT mode only.
STOP cannot be used to abort a running program when in DIRECT mode.

Delaying Program Execution

The DELAY command causes program execution to pause for a specified amount of time.

The DELAY command ensures that preceding commands have been properly executed before the next command is executed.

The command format includes a time parameter, *n*, which is expressed in hundreths of a second; for example, if *n* = 200, the delay is 2 seconds.

Edit program AAA. Insert delay commands following each MOVED command line.

Press: `<Enter>`
Press: `<Enter>`
Type: **`delay 200 <Enter>`**
Press: `<Enter>`
Type: **`delay 200 <Enter>`**
Press: `<Enter>`
Type: **`delay 200 <Enter>`**
Press: `<Enter>`
Type: **`exit <Enter>`**

Another **ACL** command, WAIT, command causes program execution to pause until a certain condition is met.

Variable Programming

Variables are locations in controller memory which are defined by name and hold values. Variables simplify programming by allowing instructions to be executed conditionally and repeatedly.

ACL has a number of system defined variables whose values indicate the status of inputs, outputs, encoders and other control system elements. Some of these variables can accept user defined values. None of these variables can be deleted from the system.

User variables are defined and manipulated by the user, and can be created or deleted as needed. User variables may be either private (local) or global.

- **Private variables** are defined and manipulated in the EDIT mode and recognized only by the specific program in which they are defined.
- **Global variables** can be defined and manipulated in both the EDIT and DIRECT modes, and can be used in any program.

The command DEFINE is used to define a private variable. Up to twelve variables can be defined in one command. For example:

Type: **`define pv <Enter>`** Defines PV as a private variable.

`define va vb vc <Enter>` Defines VA, VB and VC as private variables.

The command GLOBAL is used to define a global variable. Up to twelve variables can be defined in one command. For example:

`global gv <Enter>` Defines GV as a global variable.

`global gva gvb gvc <Enter>` Defines GVA, GVB and GVC as global variables.

Variable names must begin with an alphabetical character and may have up to 5 characters.

The commands DIM[*n*] and DIMG[*n*] are used to define arrays of private and global variable arrays, respectively; *n* defines the dimension (number of elements) in the array.

Mathematical and Logical Functions

The SET command performs mathematical and logical operations on variables. The command format may be one of the following:

```
set var1=var2
set var1=oper var2
set var1=var2 oper var3
```

Where: *var1* is a variable;
var2 and *var3* can be either a variable or a constant.
oper is: Arithmetic operator: + – * /
Algebraic operator: ABS, EXP, LOG, MOD
Trigonometrical operator: COS, SIN, TAN, ATAN
Logical (Boolean) operator: AND, OR, NOT

The simplest format assigns a variable the value of a constant or another variable. For example:

```
set var = 1
set var1 = var2
```

The value of a variable can be the result of an operation performed on another variable. For example:

`set var1 = abs var2` If the value of *var2* is -1, *var1* is set to 1.

The value of a variable can be the result of a mathematical operation performed on either two other variables or another variable and a constant. For example:

`set var1 = var2 + 1`	The value of *var1* is greater by 1 than the value of *var2*.
`set vara = varb * varc`	The value of *vara* is the result of *varb* multiplied by *varc*.
`set var = var + 100`	The result of an operation can equal the same variable, thereby changing its value. The value of *var* now equals the previous value of *var* plus 1000.

Iteration Functions

Many applications require task iteration, or repetition. Variables can be used to produce program loops which repeat a command or commands, thereby avoiding the need for redundant command lines within a program

The command format **FOR *var1* = *var2* TO *var3*** enables a program routine to be executed repeatedly. *Var1* must be a variable; *var2* and *var3* may be either variables or constants. For example, enter the following commands to create program LOOP:

```
edit loop
for var=1 to 10
println "LOOP"
endfor
exit
```

The variable is a counter, which is set initially to 1 and increased by one each time the loop is performed. When the counter value reaches the final value (10 in this example), the loop is performed for the last time.

The ENDFOR command is required to mark the end of the loop.

The PRINTLN command causes comments (text within quotation marks) to be displayed on the screen during program execution. Thus, when you run program LOOP, the word "LOOP" will be displayed 10 times.

By altering the PRINTLN command line you can cause the system to report which loop has been completed. Bring the cursor to the ENDFOR command line. Enter the command DEL; this will delete the preceding command line. Then enter a new command line:

```
println "LOOP " var
```

Make sure you have included a space following the text "LOOP."

The PRINTLN command causes the current value of a variable to be displayed on

the screen during program execution. Thus, when you run program LOOP, the following will now appear on the screen.

```
LOOP 1
LOOP 2
LOOP 3
```

. . . and so on, until LOOP 10 is displayed.

In the section on input/output programming later in this chapter, you will see additional examples of program loops which enable the system to check and respond to the state of the controller's 16 inputs.

Conditional Functions

Many applications require the program to flow according to certain conditions.

The command format **IF *var1 oper var2*** checks the relation between *var1* and *var2*. *Var1* must be a variable; *var2* may be either a variable or a constant. *Oper* is one of the following comparison operators: `>  <  +  >=  <=  <>`

When the IF statement is true, the program executes the next line(s), until it reaches an ENDIF command, which marks the end of the conditional routine.

```
if var1=var2
   goto    1
endif
```

The IF statement may, however, be followed by another conditional statement. The next line may be an alternative condition (ORIF) or an additional condition (ANDIF).

```
if var1=var2
   orif var3>10
   goto    2
endif
```

At least one of the two conditions must be true in order for the program to jump to label 2

```
if var1=var2
   andif var3>10
   goto    2
endif
```

Both conditions must be true in order for the program to jump to label 2.

The conditional routine may also contain a routine to be executed when the IF condition is false. The beginning of such a routine begins with the command ELSE.

```
if var1=var2
   goto    2
   else
   goto    1
endif
```

If the condition is not true, the program will jump to label 1.

Input and Output Programming

The state of the controller's 16 inputs and 16 outputs is determined by means of two system variables, IN[*n*] and OUT[*n*]; *n* specifies the I/O index; that is, 1–16.

The value of the variable indicates whether the input or output is on or off; when the value of the variable is 1, the input or output is ON; when the value is 0, the input or output is OFF.

Displaying Input/Output Status

The I/O LEDs on the front panel of the controller turn on and off to reflect the status of the inputs and outputs. If you are not close enough to see the controller panel, you may want another means to check the I/O status.

In DIRECT mode, use the following commands to display the status of all 16 inputs and outputs, respectively:

Type:	**`show din <Enter>`**	Shows status of the inputs.
Type:	**`show dout <Enter>`**	Shows status of the outputs.

The display will indicate the I/O status in the following manner:

```
1>16: 0 1 0 1 0 0 0 0 0 1 0 0 1 1 0 0
O.K.
```

When editing a program, use the command PRINTLN to display the status of a specific input or output during program execution. For example:

`println in[5]`	When this command is encountered during program execution, either 1 or 0 will be displayed (that is, the value of variable IN[5]), depending on the state of input 5;

Inputs

Conditional commands, such as IF and WAIT, are used to read and respond to the state of the inputs. For example, you can use the following routine in a program:

`if in[3]=1`	If input 3 is ON, then
`move A31`	Move to position A31.
`else`	If input 3 is NOT ON (off), then
`move A32`	Move to position A32.
`endif`	End of conditional routine.

Outputs

As with inputs, conditional commands can read and respond to the state of the outputs. Commands can also be used to alter the state of outputs.

To change the state of an output—in both DIRECT and EDIT modes—use the SET command. For example:

`set out[6]=1 <Enter>`	Turns ON input 6.
`set out[8]=0 <Enter>`	Turns OFF input 8.

Activating Output-Driven Devices

Pneumatic End Effectors or Devices

As mentioned in Chapter 6, pneumatic end effectors or devices are connected to controller outputs and controlled by means of **ACL** output commands.

Assuming a pneumatic gripper is connected to controller (relay) output 2, use the following command format

`set out[2]=1 <Enter>`	Turn on output 2 to open the gripper.
`set out[2]=0 <Enter>`	Turn off output 2 to close the gripper.

In order to activate the pneumatic gripper from the teach pendant, you need to create two programs (named OGRIP and CGRIP, for example) which can be called from the teach pendant by means of the **Run** key. Each program contains one of the commands shown above.

- Program OGRIP contains the command to turn on output 2.
- Program CGRIP contains the command to turn off output 2.

Using the **ACL** command DIR note the identity number of programs OGRIP and CGRIP. (The command DIR is explained more fully later in this chapter.)

Let's assume programs OGRIP and CGRIP are identified as program 8 and program 9, respectively. Now, whenever you want to open the pneumatic gripper by means of the teach pendant:

Press: **Run 9 Enter**

Warning Light

A flashing warning light can be integrated into the **SCORBOT-ER Vplus** system. A program named ONOFF is included in the ONOFF.CBU file on the **ATS** diskette supplied with the system. When the ONOFF program is activated, it will *automatically* turn on the warning light whenever the robot is in motion.

The light is normally connected to (relay) output 1. Therefore, the following commands are used in program ONOFF.

`set out[1]=1` When output 1 turns on, the light turns on.

`set out[1]=0` When output 1 turns off, the light turns off.

(In order to download this program file for use, refer to the downloading procedure described later in this chapter.)

Sample Program: INOUT

A program named INOUT can be found in the file DEMO.CBU which is factory-loaded into the controller, and included in the **ATS** diskette supplied with the system.

The program contains two loops; one loop has instructions for checking the status and responding to the state of all the inputs; the other loop has instructions for responding when input 16 is on.

This sample program demonstrates program loops and conditional routines. In addition, it shows how to include user comments within a program.

Use the LIST command to view the program shown below. Explanatory notes are provided below.

Type: **`list inout <Enter>`**

```
                PROGRAM   INOUT
                **********************
PRINTLN    "this program tests inputs & sets outputs"
PRINTLN
LABEL      1
FOR        I = 1 TO 16
   IF      IN[I] = 1
   * TEST IF INPUT I IS ON
   SET     OUT[I] = 1
   * SET OUTPUT I ON
   ELSE
   SET     OUT[I] = 0
   * SET OUTPUT I OFF
   ENDIF
   DELAY   3
ENDFOR
IF         IN[16] = 1
   * IF INPUT 16 IS ON EXIT FROM PROGRAM
   SET     OUT[16] = 0
   PRINTLN     " program inout stopped "
   PRINTLN
   GOTO    2
ENDIF
```

```
GOTO        1
LABEL       2
END
```

- PRINTLN comments will be displayed on the screen during program execution.
- PRINTLN without a comment or argument simply enters a carriage return, and brings the screen cursor to the beginning of the next line.
- The variable I is used as the counter for 16 loops.
- FOR starts a program loop which checks state of all 16 inputs.
- The first IF command starts a conditional routine with instructions for responding to the state of an input: if an input is turned on, the output of the same index is also turned on; if the input is turned off; the output is turned off.
- The asterisk * precedes a user comment within a program; the comment is not displayed during program execution.
- ENDIF ends the IF conditional routine.
- ENDFOR ends the FOR loop.
- The second IF command starts a routine which checks and responds to the the state of input 16. If input 16 is on, output 16 will not light; the program will go to label 2 and terminate.
- If input 16 is off, the program will go to label 1 and repeat.

When running this program you can simulate an external input by shorting the input terminals. *Be sure you do so according the instructions for shorting inputs detailed in Chapter 3*.

Run the program, and prepare to short the inputs.

When you short any of inputs 1 through 15, the output with the same index (1–15) will turn on. When you short input 16, the program will stop. Note the messages on the screen during program execution.

Program Directory

The **ATS** diskette supplied with the system contains a number of files with the extension CBU. These files contain programs, positions, variables and parameters. Some of these CBU files are factory-loaded into the controller and stored in battery backed-up RAM. These files are not erased when the controller is turned off, but their contents may be totally or partially erased during certain configuration and restore procedures.

To view the list of programs which are found in the controller's BBRAM, use the DIR command in DIRECT mode. For example:

Type: **`dir <Enter>`**

```
name       : validity : identity : priority
AA         :          : 1        : 5
LOOP       :          : 2        : 5
DEMO       :          : 3        : 5
IO         :          : 4        : 5
IOA        :          : 5        : 5
```

. . . and so on.

- Validity: If the program is valid no message appears. "Not valid" will appear if the program contains a logic error, such as a FOR command without an ENDFOR command.
- Identity: This is the controller-assigned program identity number, which is needed for executing a program from the teach pendant. (Since certain controller operations can cause program identity numbers to change, use the DIR command at the beginning of each working session to verify the identity of program which you may want to call from the teach pendant.)
- Priority: By default the controller assigns each program a run-time priority of 5, on a scale of 1–10. The user can define a program's priority by means of the PRIORITY or RUN command.

Multi-Tasking

Controller-A is a multi-tasking real-time controller; it can simultaneously execute and control 20 independent programs.

Use the DIR command, and note the programs: PICP, IO, IOA.

To run these three programs concurrently, use three RUN commands to start execution—in both DIRECT and EDIT mode.

Type:
```
run picp <Enter>
run io   <Enter>
run ioa  <Enter>
```

Program PICP takes the robot through a series of pick and place movements. Programs IO and IOA both turn controller outputs on and off; watch the LED display on the controller while these programs are being executed.

To abort all three programs, use the Abort command.

Displaying Program Status

While programs are running, use the command STAT to view their status.

Type:
```
stat <Enter>
```

```
JOB_NAME    PRIORITY    STATUS
PICP        000005      PEND
IO          000005      DELAY
IOA         000005      SUSPEND
```

- PEND: program is executing a movement command.
- DELAY: program execution is currently being delayed.
- SUSPEND: execution has been halted by SUSPEND command.

Activating a Program from Another Program

As indicated throughout this chapter, **Controller-A** enables interaction and synchronization of programs.

Simultaneous Execution

The RUN command can be included in a program in order to start execution of another program. When a running program encounters a RUN *prog* command, both program are executed concurrently.

When several programs are running, those with a higher priority have precedence; those with the same priority share controller CPU time by means of an equal distribution algorithm.

Program Interrupt

Since two programs may conflict with one other, it may be preferable to use the GOSUB command rather than RUN.

Like RUN, the GOSUB command is used to start execution of another program. Unlike RUN, however, when a program encounters a GOSUB *prog* command, the program is suspended until the called program has completed execution. At that point, the first program resumes execution from the line which follows the GOSUB command.

The TRIGGER command can be used to execute another program when a specified input or output is turned off or on. However, it will activate the program only once, regardless of subsequent changes in the I/O state.

Downloading a Program (Restore) to Controller

Since the controller's battery-backed RAM is limited (and can be accidently erased), program files should be saved to disk. They can be downloaded to the controller as needed. **ATS** has a Backup Manager which serves this purpose.

A program file named PARABOLA.CBU is included in the **ATS** diskette supplied with the system. The following steps will download the contents of this file to the controller BBRAM.

1. From the DIRECT mode, press <Alt>+10. The Backup Manager menu will appear on your screen.
2. **Backup directory**: Type and <Enter> the name of the drive where the **ATS** diskette files are located ; it may be a floppy disk drive, or a subdirectory on your hard disk. (You can press F9 (CATALOG) to make sure PARABOLU.CBU is in the directory.)

   Use the arrow keys to highlight "Restore PROGRAMS" and "ADD TO Controller Contents." Press <Enter> to accept these options.

   File name: Type PARABOLA and press <Enter>. The CBU extension is not needed.
3. Press F5 to load (RESTORE) the file from disk to the controller BBRAM.

   When "DONE" appears, press <Esc> to return to the main **ATS** screen.

Calculating and Moving Along a Path

The **SCORBOT-ER Vplus** system allows you to calculate the coordinates of positions along a path (vector) defined by a mathematical function, and to then move the robot through all these positions.

Parabola

The demonstration file PARABOLA which you have downloaded contains two programs: CALC and PARAB.

CALC

Program CALC calculates the Cartesian coordinates of 50 positions in a vector named V, according to the parabola equation: $Z=Y^2/5000$.

Where: $-250\text{mm} \le Y \le +250\text{mm}$
X=300mm (constant)
P=–90°(constant pitch)
R=0°(constant roll)

The program calculates the value of the Z coordinate at intervals of 10mm along the Y axis, that is: Y= –240mm, –230mm . . . 240mm, 250mm.

Three global variables have been defined:

YV	Y coordinate value
ZV	Z coordinate value
I	loop counter

A vector named V containing 50 positions has been defined.

PARAB

Program PARAB moves the robot smoothly through all the positions in the vector, from position V[1] to V[50].

To run the PARABOLA demonstration:

Type: **`run calc`**

After the vector has been created:

Type: **`run parab`**

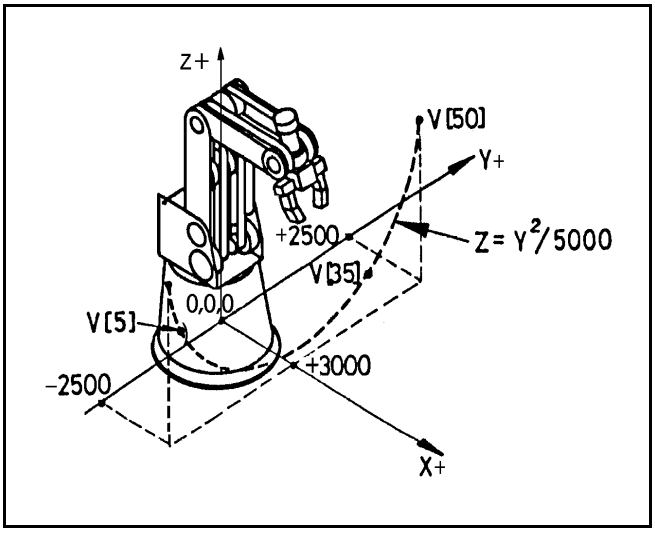


Figure 7-1: Parabola

The PARABOLA demonstration programs contain several commands which have not yet been introduced:

- SETPVC modifies the value of one Cartesian coordinate of a position.
- SETP copies the coordinates of one position to another position.
- MPROFILE defines the type of trajectory; TRAPEZE (trapezoid) profile has quick acceleration and deceleration, with constant speed along path.
- MOVES moves the robot smoothly through consecutive positions in a vector.

```
      PROGRAM   CALC
      ********************
DELAY     10
SET       YV = -2500          The initial Y value.
FOR       I = 1 TO 50         Starts a loop of 50 repetitions.
  SET       YV=YV + 100       Distance between positions: Y=10mm
  SET       ZV=YV * YV        Value of Z will be Y2
  SET       ZV=ZV / 5000      Value of Z will be Y2/5000
  SET       ZV=ZV + 1000      Keeps parabola 100mm above table.
  SETP      V[I] = 0          Initial coordinates of V are copied
                              from robot's home position.
  SETPVC    V[I] X  3000      Value of X is constant (300mm).
  SETPVC    V[I] Y  YV        Value of Y is taken from variable YV.
  SETPVC    V[I] Z  ZV        Value of Z will be ZV = the result of
                              the calculation Y2/5000.
  SETPVC    V[I] P  -900      Value of pitch is constant (–90°)
  SETPVC    V[I] R  0         Value of roll is constant (0°)
  DELAY     1                 Wait 10 milliseconds
  PRINT     I                 Announces each loop=position.
ENDFOR                        Coordinates have been calculated.
PRINTLN    "vector V created" Announces program completion.
PRINTLN    ""
END
```

```
           PROGRAM   PARAB
           ********************
SPEED     25
MOVE      V[1]                Sends robot to starting position.
MPROFILE  TRAPEZE
LABEL      1
MOVES     V 1 50              Moves robot from first to last
MOVES     V 50 1              position in vector, and back again.
GOTO       1
END
(END)
```

Sine

Controller-A uses integer arithmetic. The results of division operations are truncated to the next lower integer and therefore may not be accurate. In such instances, the operation should be preceded by a command which will perform an operation which produces a value which can be acceptably divided.

Normally only one mathematical operation can be included in a SET command. However, SET commands which perform a SIN operation also include a scaling factor, so that the result of the operation will be an acceptable value.

The following SINE program demonstrates scaling. The program RSINE moves the robot smoothly through all the positions in the vector calculated in SINE, from position S[1] to S[120].

These programs do not exist on the **ATS** disk. You may attempt to write and run them yourself.

Program SINE calculates 120 positions in a sine curve vector named S according to the equation: $Z = 1500 \sin Y + 2000$

Where: X = 200 mm (constant)
$-300 \text{ mm} \leq Y \leq +300 \text{ mm}$
P=-90°(constant pitch)
R=0°(constant roll)

The program calculates the value of the Z coordinate at intervals of 5mm along the Y axis, that is: Y= -300mm, -295mm. . . 295mm, 300mm.

These program require the same three global variables which were used in the PARABOLA demonstration:

YV	Y coordinate value
ZV	Z coordinate value
I	loop counter

A vector named S containing 120 positions must be defined: S[120]

```
          PROGRAM     SINE
          ********************
DELAY     10
SET     YV=-3050
FOR       I=1 TO 120
  SET       YV=YV + 50
  SET       ZV=YV * 360
  SET       ZV=ZV / 3000
  SET       ZV=1500 SIN ZV
  SET       ZV=ZV + 2000
  SETP      S[I] = 0
  SETPVC    S[I] X 2000
  SETPVC    S[I] Y YV
  SETPVC    S[I] Z ZV
  SETPVC    S[I] P -900
  SETPVC    S[I] R 0
  DELAY   1
  PRINT   I
  ENDFOR
PRINTLN   ">"
END
```

The initial *Y* coordinate.

Starts a loop of 120 repetitions.

Distance between positions: *Y*=5mm

These two operations scale the *Y*-axis displacement to a degree value and produce a wavelength of 300mm. The order of the commands is important: ZV=YV*360 must come first, then ZV=ZV/3000, since the result of the division is always less than 1.

Sin *Z* is multiplied by a scaling factor of 1500 to produce an acceptable result.

Offsets the *Z* value by 200mm to keep movement above table.

```
       PROGRAM     RSINE
       ********************
SPEED        25
MOVE S[1]
MPROFILE      TRAPEZE
LABEL        1
MOVES        S 2 120
MOVES        S 119 1
GOTO 1
END
```

Saving a Program (Backup) to Disk

The programs, positions and variables used in the SINE and RSINE programs will remain stored in the controller's BBRAM. In order to save them to disk, perform the following steps.

1. From the DIRECT mode, press <Alt>+10. The Backup Manager menu will appear on your screen.
2. Use the arrow keys to highlight "Backup PROGRAMS" and press <Enter>.

   Backup directory: type and <Enter> the name of the drive where you want the file to be saved (it may be a floppy drive disk, or a subdirectory on your hard disk).

   File name: type SINE (or any name of up to 8 characters) and press <Enter>. The CBU extension will automatically be written.
3. Press F2 to save the file to disk.

   When "DONE" appears, press <Esc> to return to the main **ATS** screen.

Note that this procedure saves *all* programs, positions and variables which are currently in the controller's BBRAM to the file SINE.CBU.

This page intentionally left blank.

CHAPTER 8

Maintenance

Maintenance

The maintenance and inspection procedures detailed below will ensure continued optimum performance of the **SCORBOT-ER Vplus** system.

Daily Operation

At the start of each working session, check the robot and controller, in the following order:

1. Before you power on the system, check the following items:
   - The installation meets all safety standards.
   - The robot is properly bolted to the work surface.
   - All cables are properly and securely connected.
     Cable connector screws are fastened.
   - The teach pendant, and any peripheral devices or accesssories which will be used, are properly connected to the controller.
   - None of the open collector outputs is connected directly to a power supply.
   - No people are within the robot's working range.
2. After you have powered on the system, check the following items:
   - The power and motors LEDs on the controller light up.
   - The fan in the front panel rotates and draws air into the controller.
   - The fan in the rear panel, within the supply unit, extracts air from the controller.
   - All green LEDs on the controller rear panel light up.
   - No unusual noises are heard.
   - No unusual vibrations are observed in any of the robot axes.
   - There are no obstacles in the robot's working range.

3. Bring the robot to a position near home, and activate the homing procedure. Check the following items:

   - Robot movement is normal.
   - No unusual noise is heard when robot arm moves.
   - Robot reaches home position in every axis.

Periodic Inspection

The following inspections should be performed regularly:

1. Visually check leads, cables and rubber components. Replace if any damage is evident.
2. Check all bolts and screws in the robot arm using a wrench and screwdriver. Retighten as needed.
3. Check all the tension of robot arm belts. When you press on a belt, the slack should be no greater than 2mm (0.08"). Refer to Figure 8-1.

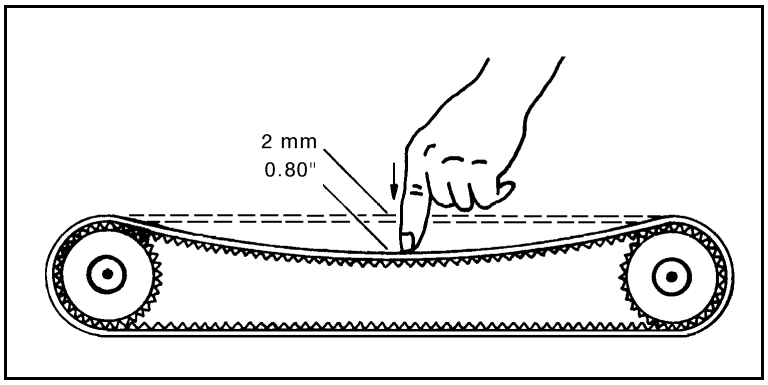


Figure 8-1: Belt Tension

Tighten the belts only if you are absolutely certain they are slipping or retarding the motors. For complete information, refer to the section, "Adjustments and Repairs," later in this chapter.

4. Check for execessive backlash in the base axis. For complete information, refer to the section, "Adjustments and Repairs," later in this chapter.

Troubleshooting

☞ *The procedures in the section are intended only for technicians who have received proper training and certification from the manufacturer.*

Do not attempt to perform procedures for which you are not qualified.

Whenever you encounter a malfunction, try to pinpoint its source by exchanging the suspected faulty component—for example, robot, controller, teach pendant, cable—with an identical component from a working system.

In general, when trying to determine the source of a malfunction, first check the power source and external hardware, such as controller switches, LEDs and cable connections. Fuses should also be checked.

In addition, make sure the controller is properly configured for the robot and gripper, the software commands have been correctly issued, and system parameters are properly set.

Make sure the controller's power switch is turned off before you open the controller cover. Make sure the power cable is disconnected from the AC power source before you remove fuses.

Complete instructions for removing and replacing controller components are given in the section, "Adjustments and Repairs," later in this chapter.

General System Check

When a problem occurs, use the **ACL** command TEST as a first step in diagnosing the problem. TEST activates an internal system procedure which checks the movement of the robot axes and the input/output functions of the controller. During the test the following occurs:

- In sequence, each configured axis is moved briefly in both directions; a message will display an axis failure.
- All outputs are turned on, and then off.
- All inputs are scanned. If an input is on, the corresponding output is also turned on.

  To simulate the activation of an input when no device is connected, short the input manually by means of a wire or an unraveled paper clip.
  - When the input is operating in NPN mode, short the input by connecting it to a ground connector.
  - When the input is operating in PNP mode, short the input by connecting it to the user power supply.

If you want to check the homing microswitches, use the command LSON before entering the TEST command. Use command SHOW DIN to see the results.

Diagnostic Procedures

1. *Controller does not turn on. The yellow power LED does not light up. Fans do not rotate.*

   - Make sure the AC power supply matches the controller's voltage requirement, as seen on the tag at the back of the controller. If the voltage supply and controller voltage setting do not match, change the voltage setting, as described later in this chapter.
   - Make sure AC power is being supplied to the power outlet.
   - Make sure the power cable is connected to both the proper power source and the controller.
   - If RS232 cable is connected, disconnect it and and retry power on. If successful, reconnect the RS232 cable.
   - Check for a blown logic power supply fuse. Using an ohmeter, measure the resistance of the fuses. If resistance is close to 0Ω, the fuse is functioning.

2. *Controller's motors switch does not turn on. The green motors LED does not light up.*

   - Check for a blown power transformer fuse. Using an ohmeter, measure the resistance of the fuses. If resistance is close to 0Ω, the fuse is functioning.

3. *No communication between the controller and the computer/terminal. Message appears on screen "Controller Not Responding".*

   - Make sure the controller's power switch is turned on.
   - Make sure the RS232 cable between the controller RS232 port and the computer COM port is properly connected.
   - Make sure you have loaded **ATS** with the proper /C switch.
   - If teach pendant also does not function, make sure the flat cable is properly connected between the communication card (PC700) and connector J8 on the main board. Refer to Figure 8-6.
   - Make sure there is no break in the wires.
   - If problem persists, continue to Item 4.

4. *Controller is totally inoperative although all power supplies are working.*

- Make sure the Emergency switch is not pressed.
- Turn the controller power switch off and on again.
- If switching off and on does not solve problem, turn off the controller again and open up the cover.

  Turn on the controller. Check the red LED on the main board, as shown in Figure 8-2. If it remains lit, the CPU is in the HALT state. Turn off the controller and remove all the driver cards. (For instructions on removing driver cards, refer to the section, "Adjustments and Repairs," later in this chapter.) Then turn the controller on again.

  If the red LED shuts off, the problem is one of the driver cards. Return the driver cards to the controller one at a time, until you determine which one is faulty. Replace the faulty driver card.
- Make sure the hardware and software configuration are compatible with the user RAM ICs:
  - Hardware: Refer to Figure 8-2. Make sure *all four jumpers* for configuring user RAM (W1, W1A, W2, W2A) are set for 128K RAM. The two lower pins must be shorted.
  - Software: Make sure the controller is configured for 128K memory by entering the command: CONFIG ?
- Remove the user RAM ICs (U4, U10, U16, U21) and reinsert them.
- If problem persists, replace the main board.

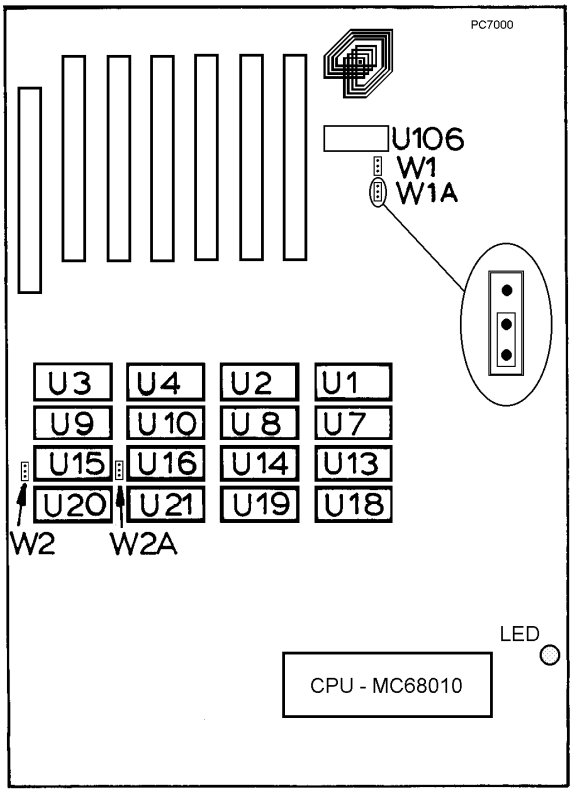


Figure 8-2: Main Board - Memory

5. *Controller is inoperative. Message on PC screen reads: "`bus error trap`", "`exception trap`", etc. and data on PC screen reads:*

```
Address error trap
D0->D7 00000000.....
A0->A7 00000C10,      000AO3FC....
PC=EB942404, SR==0008, SSP=0008796C, USP=0
```

- Turn the controller off and on.
- If problem persists, remove the driver cards and again turn the controller power switch off and on again.
- If these messages continually appear, or even occasionally reoccur, replace the main board.

6. *Controller functioning, but the robot cannot be activated.*

- Make sure an obstacle is not blocking the robot.
- Make sure the controller's motors switch is on and the green LED is lit.
- Make sure the controller is in the control off (COFF) state. Then activate the control on (CON) state from PC or TP.
- Make sure the robot cable is properly connected to the controller.
- Check whether all driver card LEDs on the controller rear panel are lit.

  Each driver card has a pair of LEDs: the upper LED corresponds to the axis number at the top of the card; the lower LED correspond to the axis number at the bottom of the card.

  Both LEDs on each card in use should be lit, indicating that power is being supplied to the axis driver. If one of the LEDs is not lit, proceed to Item 8.

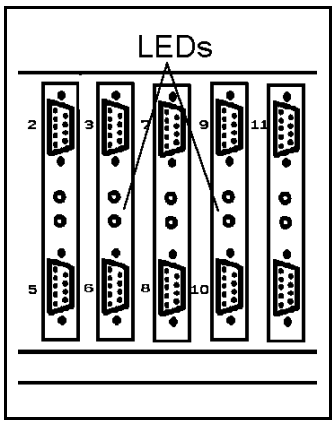


Figure 8-3: Driver Card LEDs

7. *Robot does not find Home position in one or all of the axes.*

- Make sure the homing command was properly issued.
- Make sure the robot cable is properly connected to the controller.
- Make sure system homing parameters are properly set.
  Make sure system homing parameters have not been erased.

- **Check the microswitch** for this axis.
  - Manually move the faulty axis (from teach pendant or keyboard) and use the LSON and SHOW DIN commands to check the microswitch. The value will change to either 1 or 0 when the microswitch is detected.
  - Use the commands LSON and TEST. Or prepare and continuously run a simple **ACL** program to test the microswitches, as follows:

```
LSON
LABEL 1
PRINTLN IN[n]
DELAY 200
GOTO 1
```

  - If values do not change, check the microswitch itself.

    Use a small screwdriver to press down on the microswitch. You should hear it click and see it pop back up. If this does not happen, the microswitch should be fixed or replaced.
  - If the microswitch has clicked, depress it again and, with an ohmeter, check whether the microswitch shorts its two poles.
  - If there is a short, depress the switch again and check the wires between the microswitch and D50 connector.
  - If there is a short, depress the switch and check the two microswitch pins in the D50 connector. (Refer to Chapter 10 for wiring and pin information).
  - If there is a short, replace the driver card for that specific axis.
- If the problem persists, replace the main board.
  (Alternately check ICs U88, U93, U98, U82 and U87.)

---

8. *One of the axes does not function.*
   - Make sure you have performed all steps in Item 5 and Item 6.
   - If the driver card LED for this axis is not lit, check the corresponding fuse on the axis driver card. (Refer to Figure 8-12 later in this chapter.)
     - Turn off the controller and open the cover.
     - Check the fuse on the top of the driver card for the faulty axis. (Refer to Figure 8-4). If the fuse has blown, replace it.

       (Earlier models of **Controller-A** have semi-automatic fuses on the driver cards; simply press the switch on those fuses to reset.)

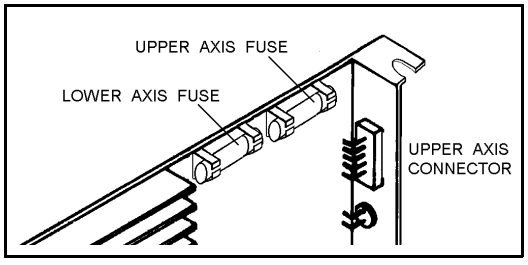


Figure 8-4: Driver Card Fuse

- **Check the motor drive circuitry**.
  - Drive the motor in open loop *for a few seconds only*.

    Use the command: SET ANOUT [*n*]=*DAC*
    n is the axis number
    DAC is the drive level: –5000 ≤ *DAC* ≤ 5000

    Note the following DAC values and their effect:

DAC value	Motor Speed
+5000	+ full speed
+2500	+ half speed
0	motor stops
–2500	– half speed
–5000	– full speed

    DAC values of 1500–2000 are recommended for this test.

☞ *Use extreme caution when applying the SET ANOUT command to robot axes or accessories whose movements are mechanically restricted. High DAC values may cause unwanted mechanical impact and can damage the robot or accessory.*

  - To cancel the SET ANOUT command, use an Abort command, or enter the command: SET ANOUT [*n*]=0.

    To help you perform the motor test, you can also prepare and run a simple **ACL** program which contains the following routine:

```
SET ANOUT [n]=1500
DELAY 200
SET ANOUT[n]=0
```

  - If the axis does not rotate, the problem can be either in the arm (motor, transmission, cabling) or in the controller (driver card, main board, communication card, or flat cable connections).
  - If the axis rotates as expected in both directions, proceed to check the encoder feedback readings.

- **Check the encoder**.
  - Enter the command SHOW ENCO to display the encoder readings. Enter the command COFF (to disable servo control) and then *physically* move the axis in question in both directions.

    The encoder reading should rise for rotation in one direction and fall for rotation in the opposite direction.
  - If the encoder readings do not change, the problem is caused by a faulty encoder, a break in the encoder wiring, or a faulty connection on a PCB within the robot. Follow the procedures in Item 9 and Item 10.

9. *Errors in the accuracy of the robot.*
Controller does not read the encoder (fails to respond to command SHOW ENCO).

- Using an oscilloscope, check the signals (P_0 and P_1) received from the encoder's two phototransistors. Figure 8-5 shows the wave diagrams which emanate from the two channels of the encoder (P_0 and P_1) with respect to the time axis. The top two signals should be clean square waves:

  V_L (low) value should be 0.4V or less.
  V_H (high) value should exceed 4 V.

  In addition, check the third wave, which shows the sum of the two waves. The diagram reflects a time shift of a quarter cylce between the two waves.

  If the waves are distorted with an incorrect shift between them, the encoder is faulty and should be adjusted or replaced.

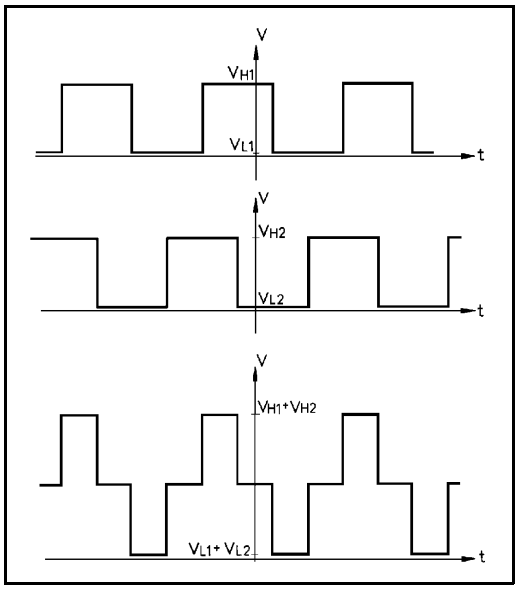


Figure 8-5: Encoder Signals

10. *Errors in the repeatability of the robot.*

- Try to identify the faulty axis. If many or all axes are faulty, look for an electrical noise source in your environment.
- Check the encoder. Follow the procedures in Item 8 and Item 9.
- If no problem found by means of Items 8 and 9, do the following:
  - Bring the robot to a starting position. Using a pencil, draw a fine, continuous line on the robot which crosses from one link to the adjacent link at the joint in question.
  - Enter the command SHOW ENCO to display the encoder readings.
  - Enter the command COFF to disable servo control.
  - *Physically* move the axis to another position. Then return to the starting position marked by the line you drew. Check the encoder reading for the axis again. It should be within several counts of the first reading. Repeat this step a number of times. If the error in the encoder reading accumulates, the encoder needs to be replaced.
- Check the transmission for loose points or damage. Check for continuity of movement in all the relevant transmission components (gears and belts moving together with the drive shaft of the motor).

11. *One axis turns constantly in one direction.*

- Reset the controller by pressing and releasing the Emergency button. Then give the command to home the robot.
- If problem persists, replace the driver card.

12. *Axis/axes vibrating, too weak to carry load, motion not smooth, or jerks during or at end of motion.*

- System parameters are not properly adjusted. Refer to the ***ACL** Reference Guide*.
- If problem persists, replace the driver card.

13. *Electric gripper does not respond at all.*

- Make sure the jumper cable is connected at the rear of controller from axis 6 port to the one marked GRIPPER.
- Check whether the gripper is defined as axis 6 by typing the ACL command: CONFIG ?
- Check the value of PAR 75. It should be within 3000-4000.
- If problem persists, proceed with corrective actions recommended for other axes.

14. *Gripper opens and closes but does not react properly to JAW command.*

- The problem is probably in the feedback. Check the encoder, the wiring, and the driver card. Follow the procedures in Item 8 and Item 9.

15. *Gripper opens and closes too freely; weak gripping force; or the gripper motor rotates endlessly.*

- The Oldham coupling in the gripper assembly is loose. Follow the instructions in the section, "Adjustments and Repairs," later in this chapter.
- Alternately, the gripper gear is broken. Replace it.

16. *Too much freedom (backlash) in the base axis.*

- Refer to the section, "Adjustments and Repairs," later in this chapter.

17. *Unusual noise.*

- Loose screws.
- Poor lubrication.
- Worn motor brushes.
- Worn timing belt.

18. *Controller does not receive an input signal.*

- Make sure motors switch is on and make sure user power supply is +12VDC. If not, none of the inputs will be operative.
- To determine whether the problem is in the controller or a user application, enter the command: SHOW DIN. Zeros and ones appear on the screen, corresponding to the status of the 16 inputs (0=OFF and 1=ON).

  Short the specific input:

  - If the input is configured as NPN (default): Short the specific input to a ground connection.
  - If the input is configured as PNP: Short the specific input to the 12V user power supply.

  Again enter the command SHOW DIN Look for a status change. If the status of the input changes, the problem is in the user application.
- If the status of the input does not change, check the flat cable connections between:
  - Display card and main board (J11).
  - I/O card and main board (J10).
- If the input LED also does not light up, refer to Item 20.
- If, when checking the homeswitches (LSON) , the input LED lights up and home is found (only the input is not functioning), replace the main board. (Alternately, check U99, U100, U101, U102, U88, U93, U98.)

19. *I/O display LEDs do not light up.*

- Refer to Figure 8-6. Check the flat cable connection between:
  - Display card and main board (J11).
  - I/O card and main board (J10).
- If the input or output functions, but not the LED, replace the LED or the display card PC7300.
- Replace the I/O card PC7400A.
  (Alternately, check U1, U2, U3, U4, U5, U6, U7.)
- Replace the main board.
  (Alternatively check: for inputs: U99, U100, U101, U102, U88, U93, U98; for outputs: U76, U77, U78, U79, U80, U81, U103, U108.)

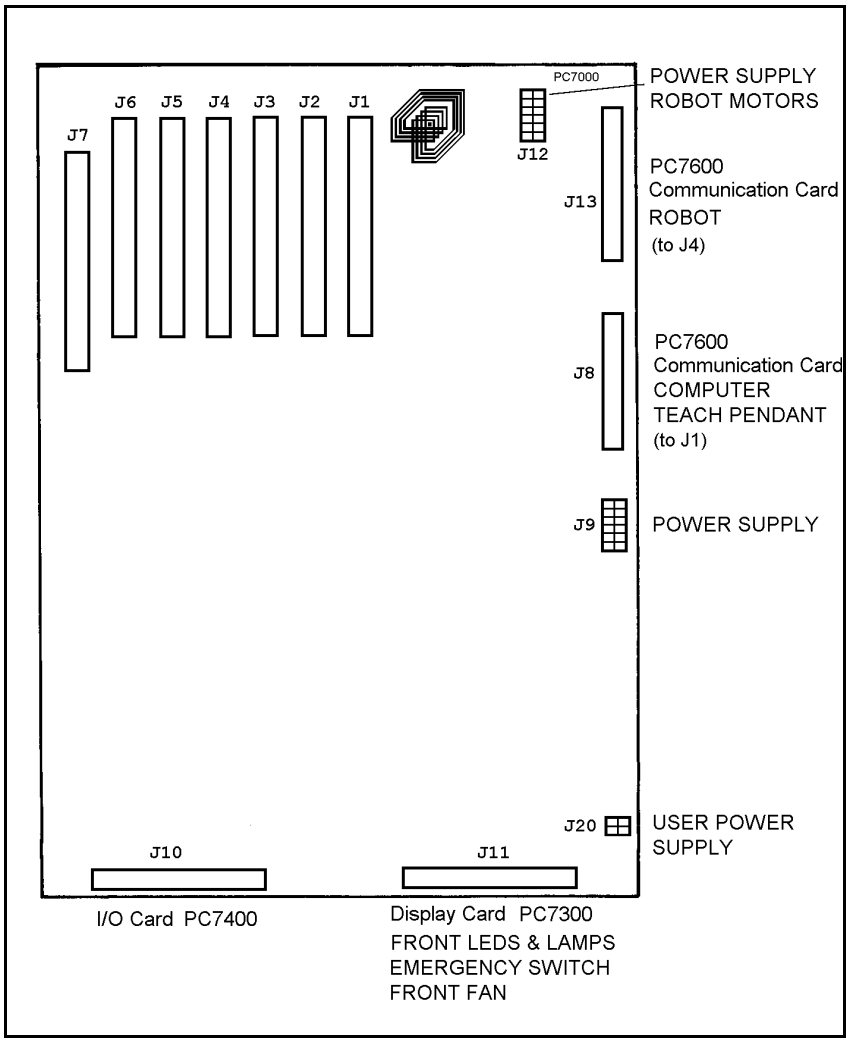


Figure 8-6: Main Board - Connectors

20. *Controller does not give output signal.*

Relay Outputs

- For Outputs 1-4, check whether the relays have been switched (LED is lit):
  - In output OFF, NC is shorted to COM, NO is disconnected from COM.
  - In output ON, NO is shorted to COM, NC is disconnected from COM.
- Refer to Figure 8-6. If outputs have not been switched, check the flat cable connection between the I/O card and the main board (J10).

Open Collector Outputs

- For Outputs 5-16, check whether the load and voltage source have been properly connected. (If the supply has been connected directly to the output terminal, the output transistor will blow out immediately).
- Refer to Figure 8-6. Check the flat cable connection between the I/O card and the main board (J10).

  (Alternately check the ICs which drive the open collector outputs signals: U76, U77, U78.)

21. *Pneumatic gripper or end effector does not respond.*

- Make sure all air hoses are connected properly.
- Make sure the gripper/device is connected to the proper controller output.
- Check the relay output to which the gripper is connected according to the instructions in Item 21.

Error Messages

Following is a alphabetical listing of system messages which indicate a problem or error in the operation of the robot arm. Refer to the ***ACL** Reference Guide* for additional error messages.

`Axis disabled.`

(1) A movement command could not be executed because servo control of the arm has been disabled (COFF).
(2) A previous movement of the arm resulted in an Impact or Trajectory error, thereby activating COFF and disabling the arm.

- Check the movements of the robot, and correct the command(s).

`CONTROL DISABLED.`

Motors have been disconnected from servo control. Possible causes:

(1) COFF (control off) command was issued.
(2) CON (control on) has not been issued; motors have not been activated.
(3) A previous error (such as Impact Protection, Thermic Overload or Trajectory Error) activated COFF, thereby disabling the arm.

`*** HOME FAILURE AXIS `*`n`*`.`

The homing procedure failed for the specified axis. Possible causes:

(1) The home microswitch was not found.
(2) The motor power supply is switched off.
(3) Hardware fault on this axis.

`*** IMPACT PROTECTION axis `*`n`*

The controller has detected a position error which is too large. The system aborted all movements of that axis group, and disabled all axes of that group. Possible causes:

(1) An obstacle prevented the movement of the arm.
(2) An axis driver fuse has shut off.
(3) An encoder fault.
(4) A mechanical fault.
(5) The axis is not connected.

- Determine and correct the cause of the position error. Then reenable servo control of the motors (CON), and restart the program.

`*** LOWER LIMIT AXIS `*`n.`*

During keyboard or TP manual movement of the specified axis, its encoder attained its minimum allowed value.

- Move the axis in the opposite direction.

***** THERMIC OVERLOAD axis *n***

Through a software simulation of motor temperature, the system has detected a dangerous condition for that motor. The system aborted all movements of that axis group, and disabled all axes of that group. Possible causes:

(1) The arm attempted to reach a position, which could not be reached due to an obstacle (for example, a position defined as being above a table, but actually slightly below the table's surface). The impact protection is not activated because the obstacle is close to the target position. However, integral feedback will increase the motor current and the motor will overheat, subsequently causing the Thermic Protection to be activated.

(2) An axis driver is faulty or its fuse has shut off.

(3) The robot arm is near to the target position, but does not succeed in reaching it, due to a driver fault. The software will then detect an abnormal situation.

(4) The Thermic Protection parameters are improperly set, or have been corrupted by improper loading of parameters.

- Check the positions, the axis driver card and parameters. Reenable servo control of the motors (CON).

***** TRAJECTORY ERROR !**

During movement, the robot arm reached its envelope limits, and the system aborted the movement. Since the trajectory is not computed prior to motion, the movement may exceed the limits of the working envelope.

- Modify the coordinate values of the positions which define the trajectory.

***** UPPER LIMIT AXIS *n***

During keyboard or TP manual movement of the specified axis, its encoder attained its maximum allowed value.

- Move the axis in the opposite direction.

Adjustments and Repairs

☞ *These procedures are to be performed only by a qualified technician who has received proper training and certification from the manufacturer.*

Adjusting the Timing Belts

When you check the tension of robot arm belts, as indicated in Figure 8-1 at the beginning of this chapter, the slack should be no greater than 2mm (0.08"). Tighten the belts only if you are absolutely certain they are slipping or retarding the motors.

- Figure 8-7 shows how to tighten the belts in the forearm which move the wrist axes (pitch and roll). Loosen the two screws (1) which hold the tension shaft. Press down on the shaft and retighten the screws.
- Figure 8-8 shows how to tighten the belts in the upper arm which move the wrist axes (2), and the belt which moves the elbow axis (3).
- Figure 8-9 shows how to tighten the two belts in the robot base which move the wrist axes. First, loosen the screw (5), and then loosen either one or both screws (4). Then, to tighten the belts, simultaneously pull the appropriate motor and retighten screw(s) (4). Finally, retighten screw 5.

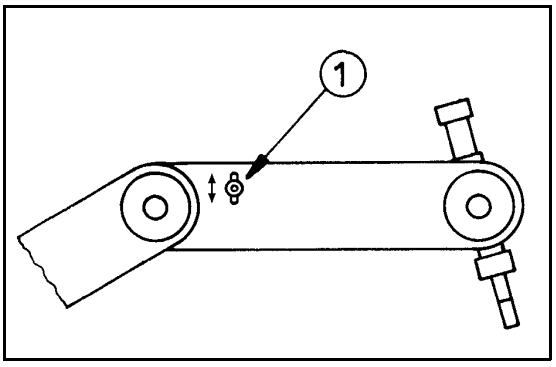


Figure 8-7: Tightening Belts in Forearm

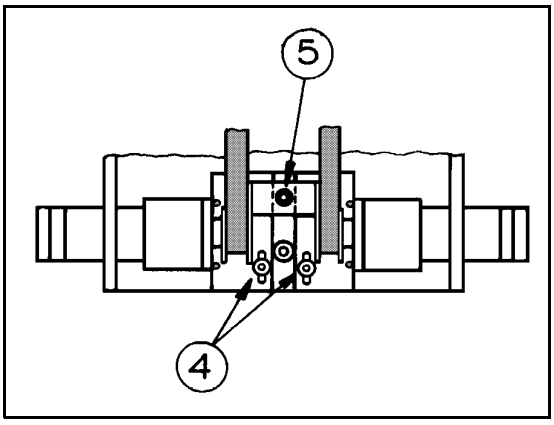


Figure 8-9: Tightening Belts in Robot Base

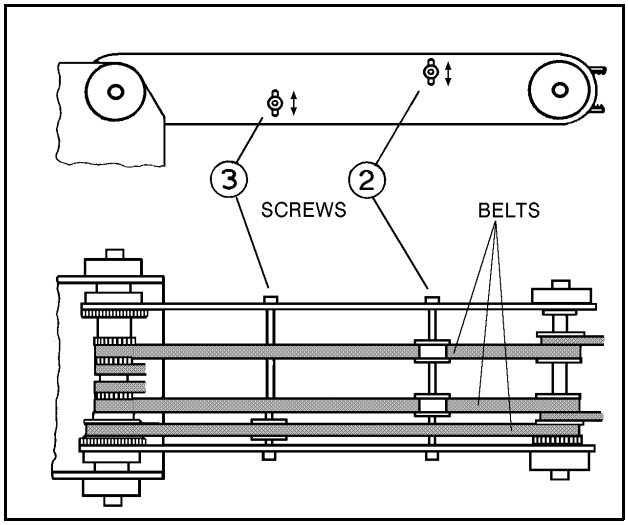


Figure 8-8: Tightening Belts in Upper Arm

Adjusting Base Anti-Backlash

Refer to the exploded views of the robot in Figures 9-3 and 9-4.

1. Refer to Figure 8-10. Remove the shoulder cover:

   - Remove the top three screws on each side of the shoulder cover.
   - Loosen (or remove) the bottom screw on each side.

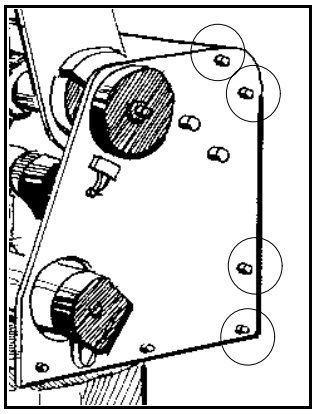


Figure 8-10: Shoulder Cover Screws

2. Refer to Figure 9-4. Remove the base lock nut (S286).
3. Refer to Figure 9-3.

   - Remove the two socket head cap screws (S19), and detach the base motor from the base plate (12).
   - Check the set screw (S151) that holds the spur gear (S25) to the base motor gear (S309). If it is loose, tighten it.
   - Reattach the base motor to the base plate.

4. Refer to Figure 9-3. The anti-backlash unit has four gears. Two gears (22 and 27) are on top of one other with a spring (23) fitted in between. Stretch the anti-backlash spring in the base transmission:

   - Make sure the robot is bolted in place.
   - Remove the outermost gear (20). The gear (22) is now free. Note the small unused hole on the base plate near the gears (22 and 27). It will enable you to lock the gear (22) in the next step.
   - To prevent the gear (22) from moving during the following steps, lock the gear by inserting a short pin through this hole and into a groove in this gear. Make sure the pin does not touch the gear (27) and that the gear (27) is free to rotate.
   - Mark the two teeth which are directly above one another on the gears (22 and 27), one on the upper gear and one on the lower gear.
   - Manually turn the robot counterclockwise a distance of six teeth between the marked teeth. The spring should now be correctly stretched.
   - Return the gear (20) to its position and fasten the screw.
   - Remove the locking pin.

5. Replace the base lock nut (S286).
6. Replace the shoulder cover.

Tightening the Oldham Coupling in Gripper

Refer to the exploded view of the gripper assembly in Figure 9-1 .

Gripper Disassembly

1. Remove the gripper motor (S312) from the plate (112) by unscrewing the three bolts (2 bolts S12 and one bolt S14). The Oldham coupling (S313) has three parts—two metal parts fitted with bolts and an intermediate plastic part. When you remove the motor, one metal piece of the coupling stays attached to the shaft. The second metal piece of the coupling stays attached to the lead screw (94). The plastic piece remains attached to either one of the two metal pieces.
2. Remove the lead screw (94) from within the shaft (105) by turning it counterclockwise.
3. Fasten both metal pieces to their respective shafts by firmly tightening the Allen screws (one piece to the motor output shaft; the other to the lead screw.)
   Note: When tightening the coupling piece to the motor output shaft, make sure the coupling is 1.5mm to 2mm away from the plate (112).

Gripper Reassembly

1. Make sure the coupling's plastic piece is attached to the metal piece attached to the lead screw (94). Keep the gripper fingers closed. Screw the lead screw (94) with the coupling piece attached, clockwise into the shaft (105), as tightly as possible. Now release the gripper fingers.
2. Refit the motor by aligning the coupling fitted to the motor output shaft together with the plastic coupling piece attached to the metal piece attached to the lead screw (94).
3. When all the coupling sections are aligned and attached, turn the motor body until the holes in the plate (112) align with those in the gear motor support (91). Reinsert and tighten the three bolts which you removed at the beginning of the procedure.

Opening the Controller Cover

1. Turn off the controller's power switch.
2. Unscrew the 4 Phillips screws which hold the cover.

   Unscrewing just the two screws at the front of the controller and lifting up the cover is possible, but not recommended, as it prevents access to the rear (connector) panel of the controller.
3. Carefully lift off the cover and set it aside.

Changing the Voltage Setting

To change the controller's voltage setting, you must change the controller's power transformer fuse and the voltage switches. Refer to Figure 8-11.

1. Open the controller cover and replace the power transformer fuse (1).

   100/110V requires 4A (SB) fuse.

   220V requires 2.5A (SB) fuse.

   This fuse is accessed from the side of the transformer housing. Using a screwdriver, push down on the fuse holder cover while turning it counterclockwise. Remove the fuse holder and replace the fuse. Reinsert the fuse holder, and retighten it clockwise, until it is securely in place.
2. Change the two switches inside the controller:
   - Using a small tool, push the red switch (2) on the transformer housing to the opposite side.
   - Manually push the switch (3) on the logic power supply to the proper setting.

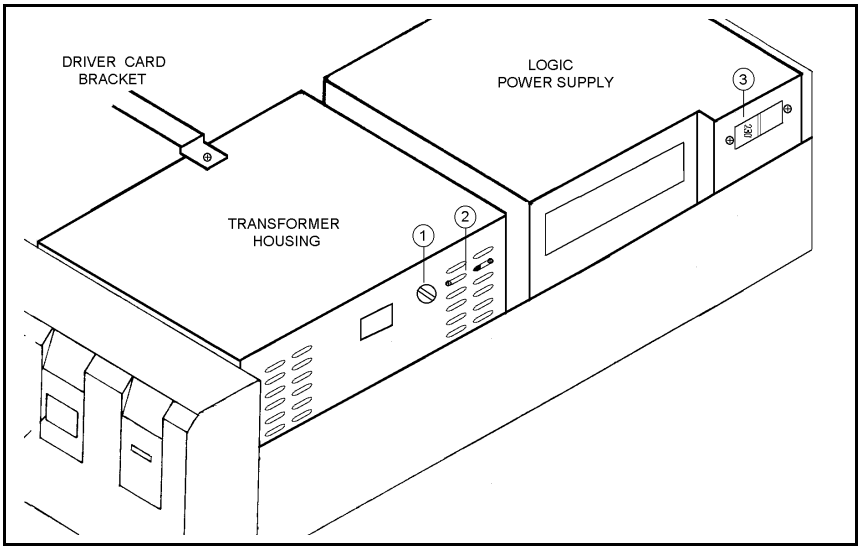


Figure 8-11: Controller Voltage Setting

Replacing Fuses

☞ *Warning! Before you begin to check or remove fuses, turn off the controller's power switch, and disconnect the power cable from the AC power source.*

Logic Power Supply Fuse

One 4A (220/110V) fuse inside the logic power supply.

The logic power supply is an IBM/PC type power supply. It will not "wake up" if it is not loaded, and it is protected against short load. Therefore, when searching for a blown fuse, be sure you are trying to operate the logic power supply under loaded conditions.

To replace this fuse:

- Disconnect the cable from the logic power supply to the main board (J9).
- Remove the power supply from the controller by unscrewing the four screws on the rear panel of the controller.
- Open the power supply and change the fuse, which is mounted in a standard fuse holder.

Power Transformer Fuse

One 2.5A (SB) fuse (220V) *or* 4A (SB) fuse (110V), on the side of the transformer housing. Feeds AC power to the transformer, from which the motors and user's power supplies are produced.

To replace this fuse:

- Using a screwdriver, push down on the fuse holder cover while turning it counterclockwise.
- Remove the fuse holder and replace the fuse.
- Reinsert the fuse holder, and retighten it clockwise, until it is securely in place.

User Power Supply Fuse

One 2A (12VDC) fuse on the power supply card PC7500 inside the transformer housing. Protects the user's power supply.

To replace this fuse you must open the transformer housing.

☞ *Warning! The large motors capacitor may be loaded with an electrical charge even after you have disconnected power. Be careful not to touch or short it.*

Driver Card Fuses

Each driver card has two slow blow (SB) fuses (one fuse per axis). See Figures 8-4 and 8-13.

- The first three driver cards, for axes 1 through 6, have a 2A (24VDC) fuse for each axis.
- The fourth driver card, for peripheral axes 7 and 8, has a 6A (24VDC) fuse for the top axis (axes 7) and a 2A fuse for the lower axis (axis 8).
- Additional driver cards, for axes 9 through 11, can have either 2A or 4A fuses.

A driver card fuse can be replaced without removing the driver card from the controller. To replace a driver card fuse, simply grasp it and extract it from its holder. You may need to use a tweezers. Insert the new fuse into the holder, and make sure it is firmly in place.

(Driver cards in earlier models of **Controller-A** have two automatic fuses (one fuse per axis). These fuses should not need to be replaced, only reset. However, if you do replace such a fuse, *solder it only when in the open state*.)

Changing the I/O Logic Mode

The I/O logic mode can be set individually for each input and open collector output terminal by means of jumpers on the I/O card PC7400A, as shown in Figure 8-12.

The jumpers are marked I1 to I16 and O5 to O16. Note that the relay outputs do not require jumpers.

Use tweezers or a fine-tip pliers to lift off the jumpers and reset them. You do not need to remove the I/O card from the controller.

- **NPN Logic**

  Shorting the two pins on the left sets the corresponding terminal to negative (NPN) logic. All inputs and outputs area factory configured for operation in negative (NPN) logic mode.

  - ON = low voltage or ground
  - OFF = high voltage

- **PNP Logic**

  Shorting the two pins on the right sets the corresponding terminal to positive (PNP) logic.

  - ON = high voltage
  - OFF = low voltage or ground

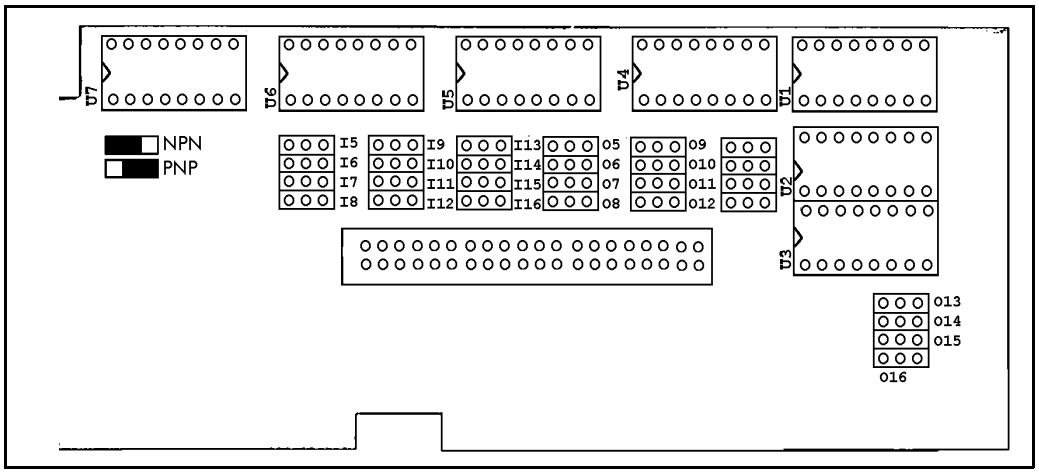


Figure 8-12: I/O Card - Logic Jumpers

Replacing or Adding a Driver Card

Refer to Figure 8-13.

(Skip Steps 4 and 5 when adding a driver card.)

1. Turn off the controller and disconnect the power cable from the power outlet.
2. Remove the cover of the controller.
3. Remove the long bracket which extends across the driver cards:
   - Remove the screw which holds the long bracket to the transformer cover (see Figure 8-10).
   - Using pliers to grip the self-locking washer from inside the controller frame, remove the screw that fastens the bracket to the side of the controller.
   - Remove the screws and washers which hold the long bracket to each driver card.
4. Note the location of each driver cards (you will replace them later in these same positions). Remove the screw at the top of each driver card bracket to detach the driver card from the connector slot.
5. Holding the card with two hands, lift it out very carefully.

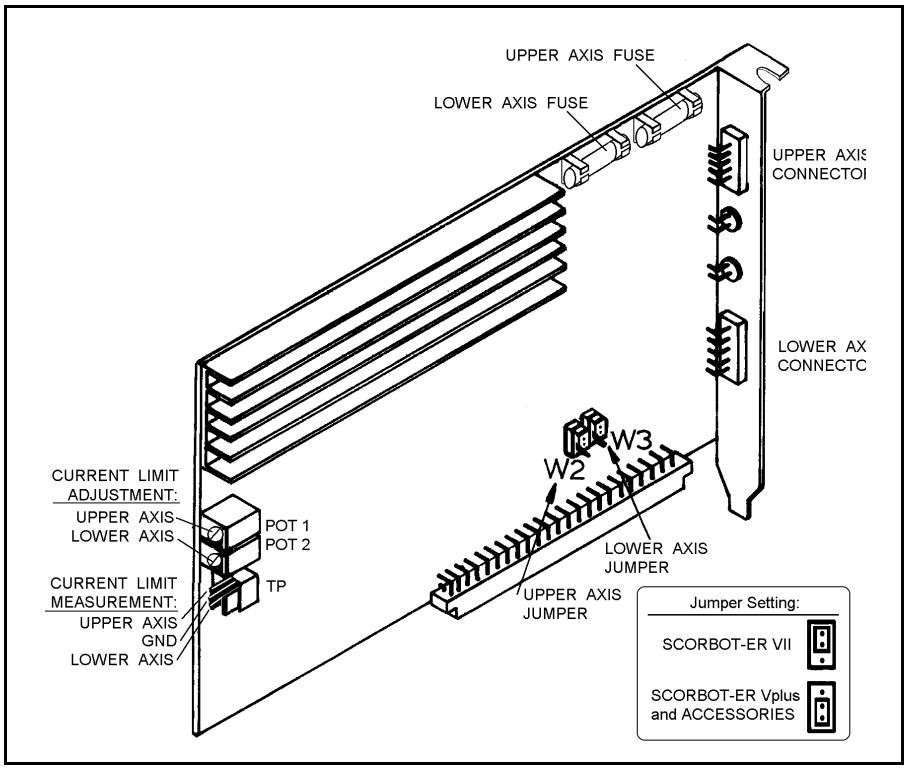


Figure 8-13: Axis Driver Card

6. Before inserting the new driver card, make sure none of the 64 pins in the male DIN connector is bent. Then, make sure the driver card is directly above the female DIN connector on the main board, and that the metal bracket fits the rear panel. Firmly but gently press the driver card into the driver card slot.
7. Reattach the long bracket to the transformer housing and the controller frame. Reattach each driver card to the long bracket and its connector slot. Retighten all the screws.
8. Check and adjust the current limit, according to the instructions in the following section.

Adjusting Driver Card Current Limit

Refer to Figure 8-13.

1. Turn off the controller.
2. Connect the common probe of the voltmeter to the middle point in TP (marked GND in Figure 8-13).
3. Turn on the controller.
4. Using a small screwdriver, rotate POT 1 (for upper axis) or POT 2 (for lower axis).

   Rotating clockwise reduces the level of the current limit;
   Rotating counterclockwise increases the level of the current limit.

   Watch the voltmeter reading; the voltmeter reading reflects the amperage of the current limit level. Adjust the current limit as follows:

Robot Axes Driver Cards	Current Limit
SCORBOT-ER Vplus Upper Axis (Axes 1, 2, 3) SCORBOT-ER Vplus Lower Axis (Axes 4, 5, 6)	-2.25 V -2.25 V
Peripheral Axes Driver Cards	
Peripheral Device Upper Axis (Axes 7, 9, 11) Peripheral Device Lower Axis (Axes 8, 10)	-4.0 V -2.25 V

Driver Card Jumper Configuration

Note the configuration of the two jumpers, W2 and W3, on the driver card, shown in Figure 8-13.

The jumpers must be mounted in the **lower position** when connecting the **SCORBOT-ER Vplus** and peripheral devices.

(Conversely, the jumpers must be in the **upper position** when connecting a **SCORBOT-ER VII** robot to the controller.)

Installing the Auxiliary RS232 Communication Card

An auxiliary (multiport) RS232 communication card may be installed in the controller to provide additional RS232 communication channels. The cable leading from the card may have either two or eight D25 connectors. Refer to Figures 4-3 and 8-14.

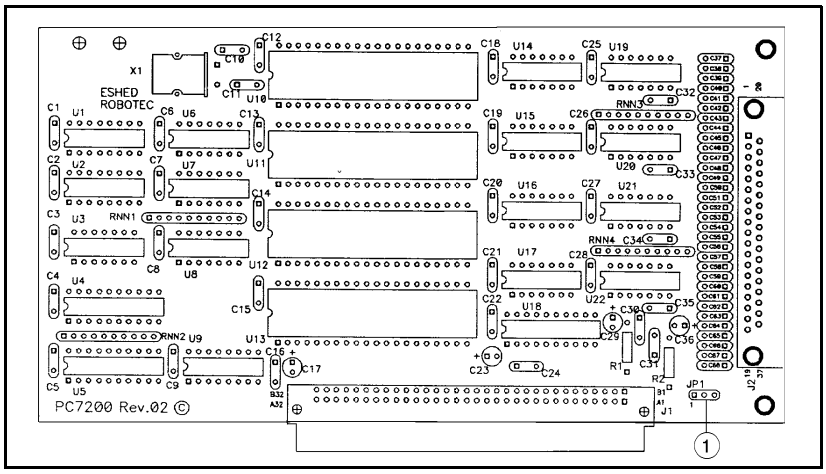


Figure 8-14: Auxiliary RS232 Communication Card

1. First, make sure Pins 1 and 2 are shorted on the card's Jumper JP1 (1).

   Jumper JP1 allows the software to determine whether or not the communication card has been installed in the controller.
   Default factory setting: Pin 1 and pin 2 shorted.

2. Turn off the controller.

3. Remove the cover of the controller.

4. Remove the blank bracket at the back of the controller on the slot (J7) for the auxiliary RS232 card.

5. Before inserting the auxiliary RS232 card, first check that none of the 64 pins in the male DIN connector is bent. Then, make sure the card is directly above the female DIN connector (J7) on the main board, and that the metal bracket fits the rear panel. Firmly but gently press the card into the slot.

6. Tighten the bracket screw.

7. Make the cable connections:

   - Connect the D37 connector from the multiport connector cable to the auxiliary RS232 port on the controller.
   - Connect the D25 connectors on the multiport connector cable to the corresponding COM ports on the other controllers or computers.

8. The controller must be reconfigured for the auxiliary RS232 card.

Before you perform the configuration, you must backup to disk the entire contents of the controller, including all parameters.

Power on the system. From the **ATS** Backup Manager menu, select the options "Backup ALL" and "BACKUP to disk (F3)."

9. Perform the configuration, using either of the following methods.
   - Use the command <Ctrl>+F1, as described in the section, "Controller Configuration," in Chapter 4; or
   - Use the **ACL** command CONFIG, as described in the ***ACL** Reference Manual*.
10. Reload the contents of the controller, including all parameters, which you backed up to disk. From the **AT**S Backup Manager menu, select the options "Restore ALL" and "RESTORE from disk (F5)."

CHAPTER 9

Parts Lists

This chapter contains isometric drawings of the robot arm and the controller.

Note that the exploded views of the robot arm show the **SCORBOT-ER V** robot. The **SCORBOT-ER Vplus** robot arm has several enhanced features which do not appear in these drawings. They are:

- Improved encoders on all motors provide greater accuracy. The encoder disk has 20 slots; the encoder housing and circuitry have also been upgraded.
- The motor supports (items 34 and 35) for the shoulder and elbow axes been improved; their dimensions have changed, and counter bearings have been added, to increase strength and stability.
- Plates have been added to the robot arm frame, across the forearm and upper arm, and around the shoulder, to increase strength and stability.

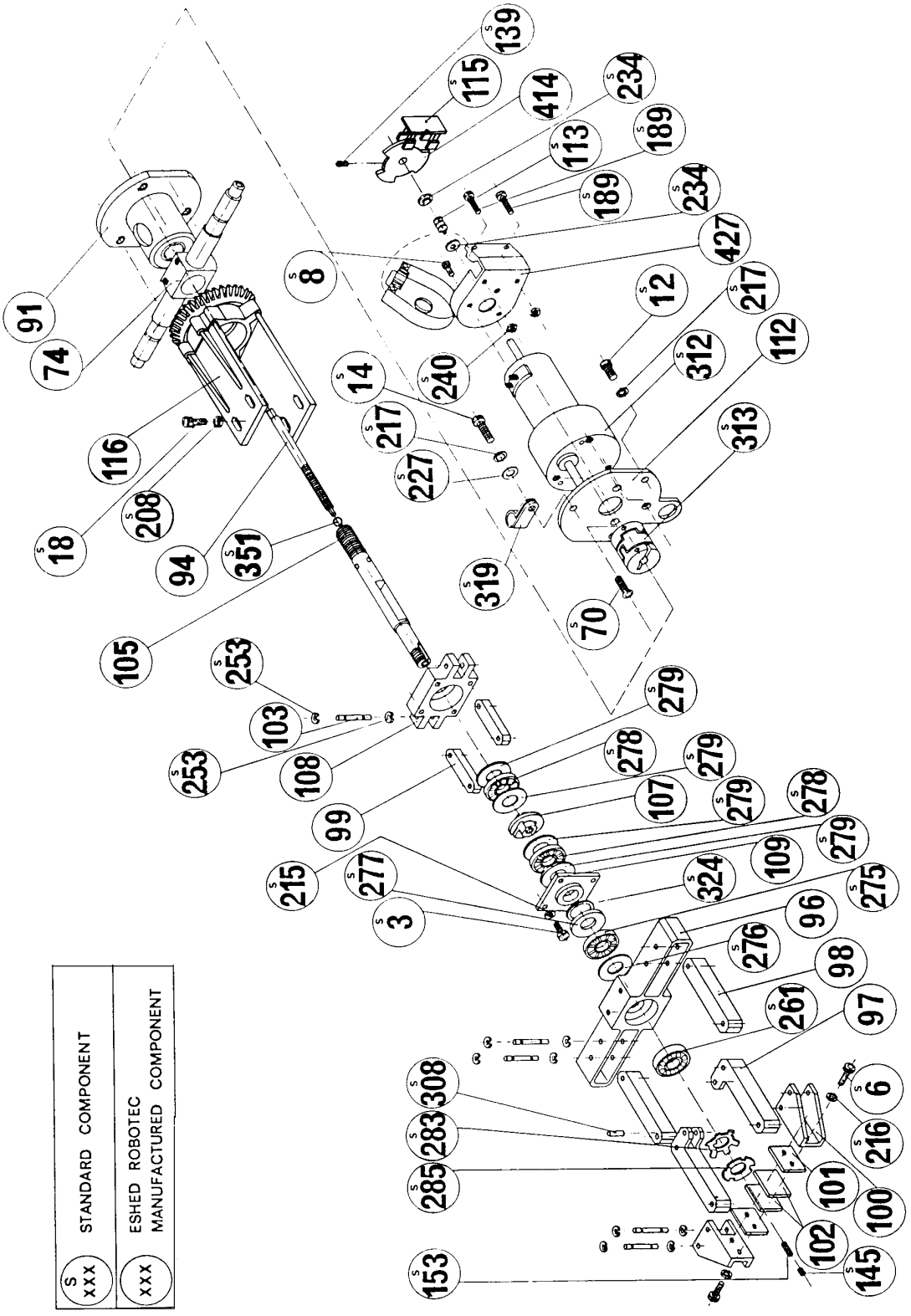


Figure 9-1: Gripper Assembly

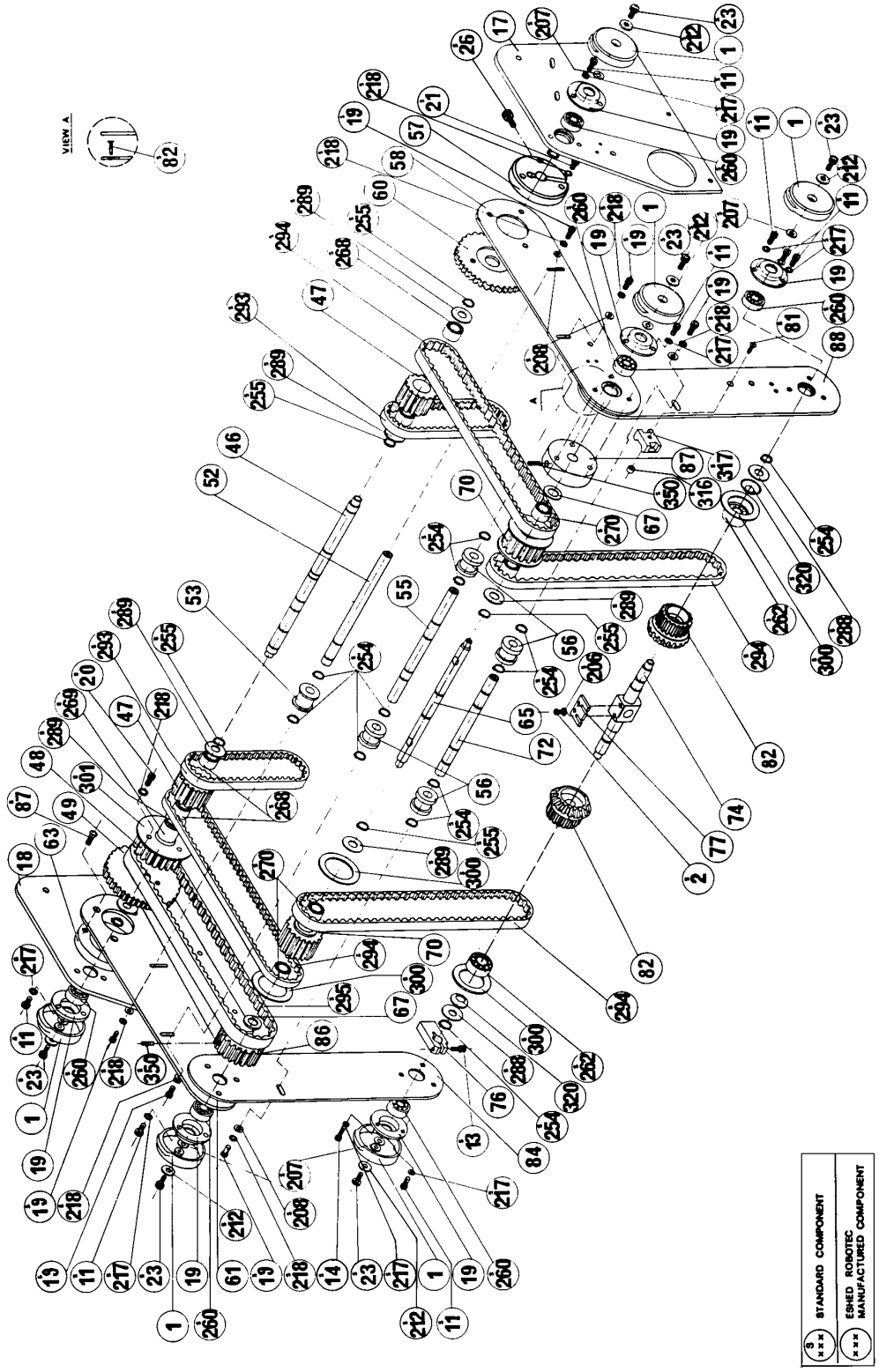


Figure 9-2: Robot Arm Assembly

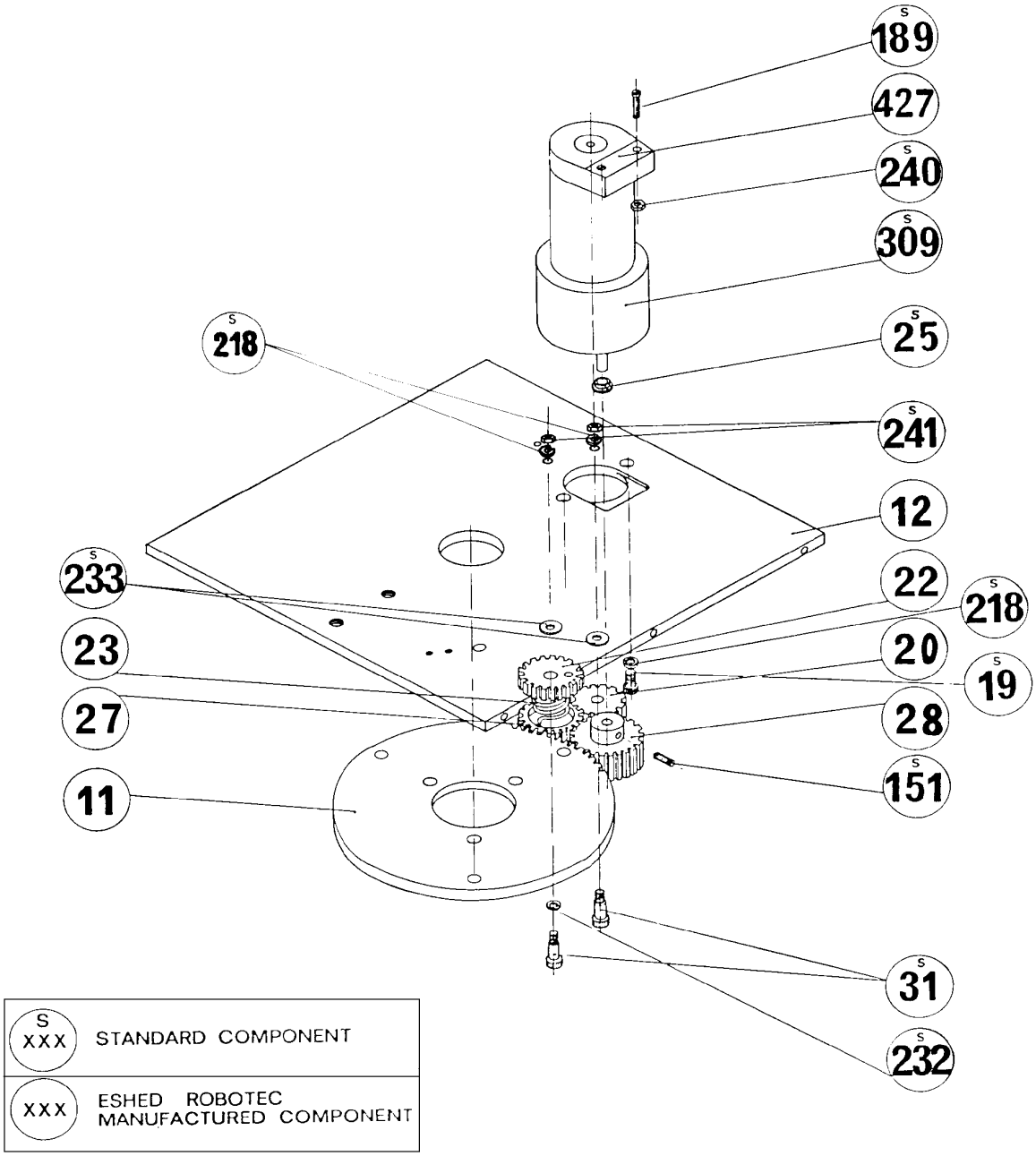


Figure 9-3: Anti-Backlash Assembly

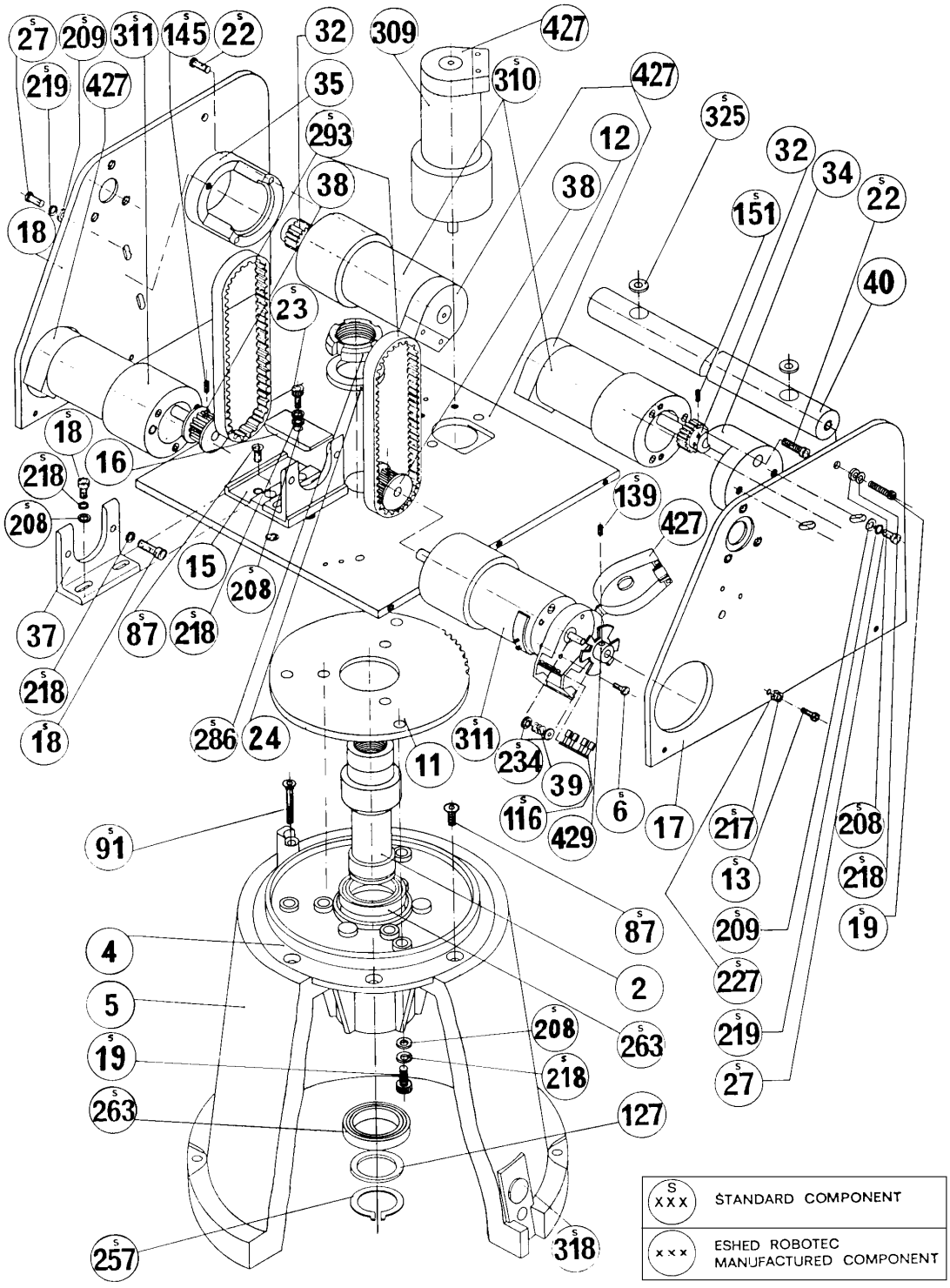


Figure 9-4: Base and Motors Assembly

Robot

Dwg #	Cat #	Description
1	113012	Bearing housing cover (plastic)
2	111401	Main shaft base
S 2	306003	Socket head cap screw #4-40 X 1/4
S 3	306004	Socket head cap screw #4-40 X 3/8
4	113004	Base plate
5	113001	Base
S 6	306201	Socket head cap screw #6-32 X 1/4
S 8	306002	Socket head cap screw #2-56 x 3/8
11	111906	Spur gear (120 teeth)
S 11	306204	Socket head cap screw #8-32 x 1/4
12	112103	Bottom Plate - shoulder
S 12	301205	Socket head cap screw #8-32 x 3/8
S 13	306206	Socket head cap screw #8-32 x 1/2
S 14	306207	Socket head cap screw #8-32 x 5/8
15	112401	Support base - motors 4+5
16	112403	Support clamp - motors 4+5
17	110205	Right side plate - shoulder
18	110210	Left side plate - shoulder
S 18	306401	Socket head cap screw #10-32 x 3/8
S 19	306402	Socket head cap screw #10-32 x 1/2
20	111901	Anti-backlash spur gear (transfer)
S 20	306404	Socket head cap screw #10-32 x 3/4
S 21	306405	Socket head cap screw #10-32 x 7/8
22	111902	Anti-backlash spur gear (upper)
S 22	306407	Socket head cap screw #10-32 x 1/4
23	113501	Anti-backlash spring
S 23	306403	Socket head cap screw #10-32 x 5/8
24	107003	Washer
S 24	306408	Socket head cap screw #10-32 x $1^1/2$
S 25	321001	Ball bearing (motor 1 gear)
S 26	306602	Socket head cap screw #1/4-20 x 1
27	111903	Anti-backlash spur gear (base)
S 27	306602	Socket head cap screw #1/4-20 x 5/8
28	111907	Spur gear (base motor)
S 31	306414	Socket head cap screw #10-32 x 3/4 x 1/4 shoulder
32	319404	Spur gear (motors 2+3)
34	112405	Motor support (motor 2) [*differs in ER Vplus*]
35	112404	Motor support (motor 3) [*differs in ER Vplus*]
37	112402	Motor support (motors 4+5)
38	319406	Timing belt pulley (motors 4+5)
40	111606	Rear cross bar [*not used in ER Vplus*]

Dwg #	Cat #	Description
46	111402	Main shoulder shaft
47	111909	Timing belt pulley
48	111911	Timing belt pulley
49	111905	Spur gear (72 teeth)
52	111405	First tension shaft
53	113013	Tension wheel
55	111406	Second tension shaft
56	113014	Tension pulley
57	112406	Clamp – lower arm – left side plate
58	110215	Upper arm – right side plate
60	111904	spur gear (right – 72 teeth)
61	110220	Upper arm – left side plate
63	112407	Clamp – lower arm – left side plate
64	111403	Middle shaft
67	107001	Aluminium spacer
70	111910	Timing belt pulley
S 70	306007	Flat head socket screw #4-40 x 1/4
72	111407	Third tension shaft
74	111404	Gripper axis
76	112439	Stopper (motors 4+5)
77	110705	Baseplate limit switch
S 81	306201	Flat head socket screw #8-32 x 3/8
82	113008	Timing belt pulley + miter gear
S 82	306211	Flat head socket screw #8-32 x 1/2
84	110228	Forearm left side plate
86	111912	Timing belt pulley
87	112114	Flange
S 87	306410	Flat head socket screw #10-32 x 1/2
88	110223	Forearm – right side plate
91	112408	Gripper gear motor support
S 91	306412	Flat head socket screw #10-32 x 1/4
94	113801	Lead screw
96	112117	Gripper bridge
97	112118	Gripper finger (inner)
98	112119	Gripper finger (outer)
99	112120	Gripper finger (short)
100	112113	Gripper clamp
101	110703	Mounting plate – gripper
102	113201	Rubber pad – gripper
103	111409	Pivot pin
105	111408	Main shaft – gripper
107	113802	Lead nut – gripper
108	112115	Bearing housing
109	112116	Bearing housing cover

Dwg #	Cat #	Description
112	110229	Gripper motor base plate
113	113505	Spring 120 g. (gripper motor) [*not used in ER Vplus*]
S 115	45007	Encoder circuitry (3 slots) [*differs in ER Vplus*]
116	113009	Miter gear (bottom)
S 116	45006	Encoder circuitry (6 slots) [*differs in ER Vplus*]
127	107009	Spacer washer (for base bearing)
S 139	306008	Socket head set screw #4-40 x 1/8
S 145	306213	Socket head set screw #8-32 x 3/16
S 151	306413	Socket head set screw #10-32 x 3/16
S 153	306214	Socket head set screw #8-32 x 1/4 (without head)
S 187	302002	Socket binding head screw M2 x 10 (limit switch)
S 188	302001	Slotted binding head screw M2 x 8 (limit switch)
S 189	302006	Slotted binding head screw M2x20 (encoder housing)
S 206	313001	Washer (for screw #4-40)
S 207	107012	Washer (black); internal; for plastic cover ∅ 12.5 x ∅ 5.5 x 0.6
S 208	313004	Washer for screw #10-32
S 209	313005	Washer for screw ∅1/4
S 212	314508	Washer lock; black; external ∅ 5
S 215	314002	Spring washer (for screw #4-40)
S 216	314003	Spring washer (for screw #6-32)
S 217	314004	Spring washer (for screw #8-32)
S 218	314005	Spring washer (for screw #10-32)
S 219	314006	Spring washer (for screw ∅ 1/4)
S 225	314503	Lock washer M2
S 227	313003	Washer (for screw #8-32)
S 232	107008	Teflon washer ∅ 1/4" x ∅ 3/8" x 0.6mm
S 233	107007	Teflon washer ∅ 1/4" x ∅ 1/2" x 0.6mm
S 234	113016	Nylon washer ∅ 11 x ª 4 [not used in ER Vplus]
S 240	310001	Hexagonal nut M2
S 253	316006	E-Ring ∅ 1/8 DIN 6799
S 254	316003	Retaining ring ∅ 10 DIN 471
S 255	316004	Retaining ring ∅ 12 DIN 471
S 257	316302	Retaining ring ∅ 25 DIN 471
S 260	320005	Ball bearing ∅ 8 x ∅ 22 x 7
S 261	320004	Ball bearing ∅ 10 x ∅ 19 x 5
S 262	320006	Ball bearing ∅ 10 x ∅ 26 x 8
S 263	320203	Ball bearing ∅ 25 x ∅ 47 x 8
S 268	320701	Needle bearing ∅ 12 x ∅ 16 x 10
S 269	320702	Needle bearing ∅ 12 x ∅ 19 x 16
S 270	320704	Needle bearing ∅ 15 x ∅ 21 x 12
S 270	320705	Bushing for #320704
S 275	320501	Thrust bearing ∅ 10 x ∅ 24 x 2
S 276	320502	Thrust washer ∅ 10 x ∅ 24 x 1
S 277	320503	Thrust washer ∅ 10 x ∅ 24 x 2.5

Dwg #	Cat #	Description
S 278	320504	Thrust bearing ∅ 12 x ∅ 26 x 2
S 279	320505	Thrust washer ∅ 12 x ∅ 26 x 1
S 283	314501	Lock washer
S 285	310401	Lock nut – gripper
S 286	310402	Lock nut – base KM 5
S 288	100706	Washer ∅ 10.5 x ∅ 20 x 0.5
S 289	100705	Washer ∅ 12.5 x ∅ 22 x 0.5
S 293	319201	Timing belt
S 294	319202	Timing belt
S 295	319203	Timing belt
S 300	315202	Flange – timing belt pulley
S 301	315201	Flange – timing belt pulley
S 308	317501	Pivot pin ∅ 1/8" x 3/8"
S 309	430901	Motor Gear - base; 127.7:1
S 310	430901	Motor Gear - shoulder/elbow; 127.7:1
S 311	430902	Motor Gear - pitch/wrist 65.5:1
S 312	430903	Motor Gear - gripper
S 313	319001	Coupling
S 315	410802	Limit switch
S 316	310802	Nut for harness
S 317	300006	Harness clamp
S 318	113006	Rubber plug (base)
S 319	300007	Harness clamp
S 320	314007	Conical washer
S 322	113203	Rubber grommet
S 324	113202	O-ring (rubber)
S 325	113204	Rubber stopper
S 350	317801	Roll pin ∅ 1/8 x 1 1/4
S 351	317502	Ball bearing ∅ - 3.5 mm
414	105001	Encoder disk (3 slots) - gripper [*differs in ER Vplus*]
427	113005	Encoder housing (plastic) [*differs in ER Vplus*]
429	105002	Encoder disk (6 slots) [*differs in ER Vplus*]

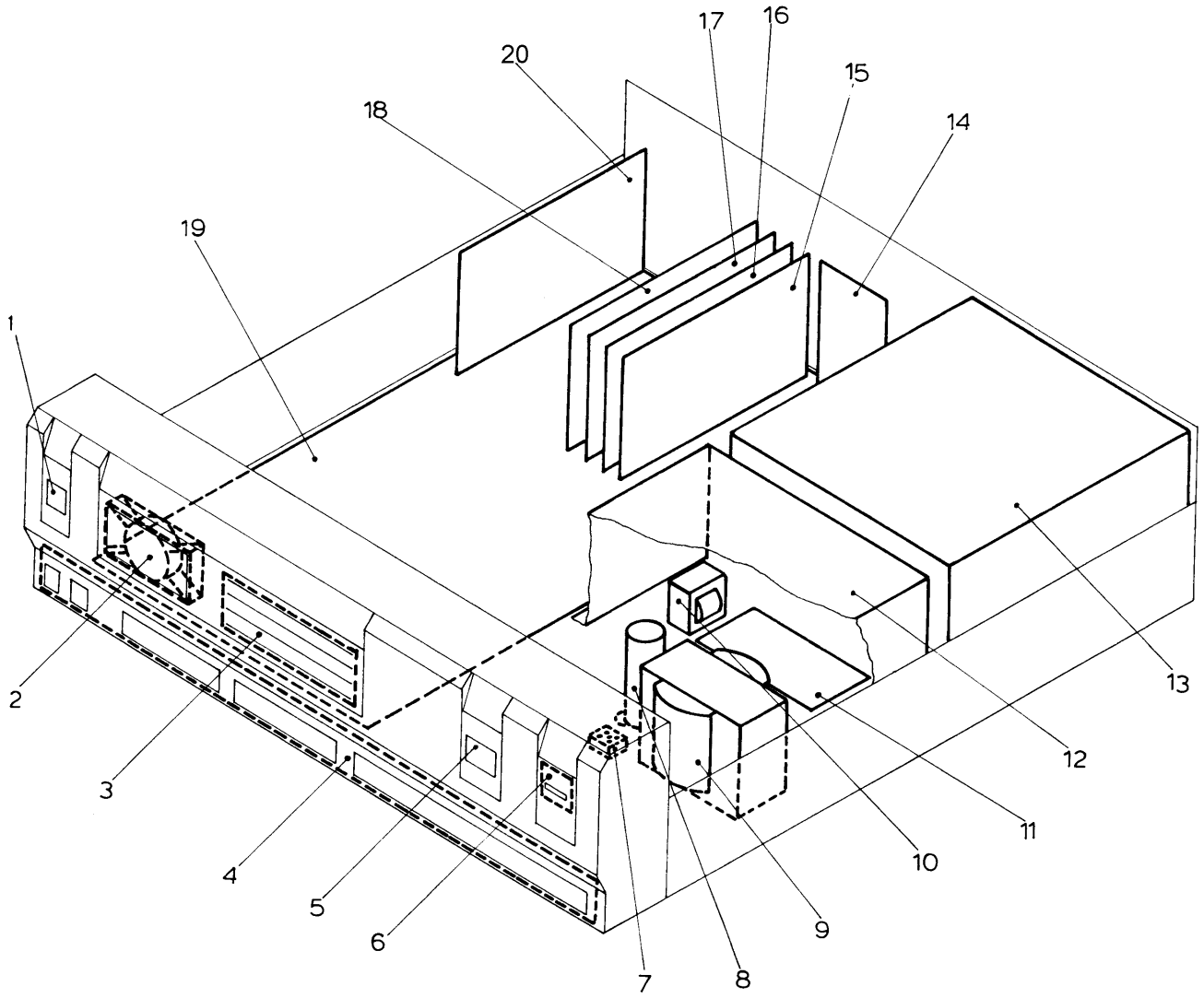


Figure 9-5: Controller-A

Controller

Dwg #	Cat #	Description
	110715	Metal case - lower part
	110717	Metal case - upper part
	113002	Controller front panel
	110719	Metal case - rear panel
	110723	Coil fastener
	102501	Lexan tags for front panel
	110725	Long bracket - driver card support
	107204	Blank brackets
12	110721	Transformer cover
13	35008	Logic power supply (220/110VAC)
9	35006	24V/12V Transformer (100VAC)
10	35003	Gripper coil assembly
2	35001	Fan plus cabling and connector
19	450541	Main board
15-17	45018	Driver cards for robot
18	45019	Driver cards for accessories
3	45011	Display card
4	45013	I/O card
11	45023	User power supply card
14	45009	Communication card
5-6	45003	Power LED card plus motors switch plus cabling
1	40004	Emergency switch plus cabling
	40018	+24VDC feed cable (from J12 to capacitor J12)
	40007	Diode bridge cabling
	40005	Switching cable (from J20 to user power supply.)
	411807	Flat cable (from J10 to I/O card)
	411806	Flat cable (from J13 to communicationcard)
	411808	Flat cable (from J11 to display card)
	411805	Flat cable (from J8 to communication card)
	40017	Gripper cable (jumper)
	40010	Grounding cable for capacitor
	40009	Grounding cable for transformer metal cover
	40006	Resistors cable for capacitor
7	408102	Diode bridge
8	404501	10,000 µF/63V capacitor
	45024	Teach pendant card [*inside teach pendant*]

This page intentionally left blank.

CHAPTER 10

Wiring

Robot Wiring

The robot is connected to the controller by means of a cable which runs from the robot base to the D50 connector marked ROBOT on the rear panel of the controller. See Figure 10-1.

The leads from the five motors on the robot body and their encoders are connected directly to the D50 connector on the robot cable. The leads from the gripper motor and the microswitches on the arm reach the D50 connector via a square 12-pin Molex connector in the base of the robot; these leads are particularly flexible and resistant to breakage, even after extensive movement of the robot arm.

The following table details the wiring for the various electrical components in the **SCORBOT-ER Vplus** robot.

(* indicates two wires on same pin.)

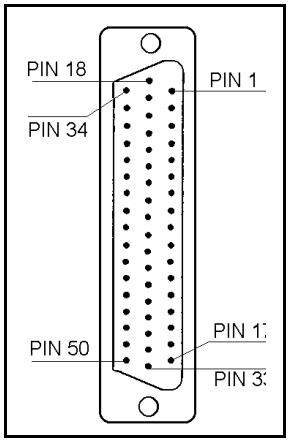


Figure 10-1: Robot D50 Connector

SCORBOT-ER Vplus Wiring								
	Robot Arm Signal				Lead to Molex 12-pin Connector		Lead to D50 Connector	
Axis	Motor	Encoder	Pad #	Microsw.	Color	Pin#	Color	Pin #
1	+						white	50
	–						gray/green	17
2	+						white	49
	–						white/green	16
3	+						white	48
	–						orange/brown	15
4	+						white	47
	–						orange/green	14
5	+						white	46
	–						orange/gray	13
Gripper	+				gray	8	white	45
	–				yellow	7	orange/blue	12
1		GND	1				white	33*
		P_1	3				white/gray	5
		V_{LED}	2				yellow	11
		P_0	4				brown	2
2		GND	1				white	32*
		P_1	3				white/orange	21
		V_{LED}	2				yellow	27
		P_0	4				gray	1
3		GND	1				white	31*
		P_1	3				brown/blue	4
		V_{LED}	2				yellow	10
		P_0	4				green	36
4		GND	1				white	30*
		P_1	3				green/brown	20
		V_{LED}	2				yellow	26
		P_0	4				orange	35
5		GND	1				white	29*
		P_1	3				green/blue	3
		V_{LED}	2				yellow	9
		P_0	4				blue	18
Gripper		GND	1		black	12	white	28*
		P_1	3		green	11	gray/blue	19
		V_{LED}	2		yellow	10	white	25
		P_0	4		brown	9	white/blue	34

SCORBOT-ER Vplus Wiring								
	Robot Arm Signal				Lead to Molex 12-pin Connector		Lead to D50 Connector	
Axis	Motor	Encoder	Pad #	Microsw.	Color	Pin#	Color	Pin #
1				GND MS			white brown	33* 23
2				GND MS			white gray	32* 7
3				GND MS	white white	1 2	white orange	31* 24
4				GND MS	blue blue	3 4	white green	30* 8
5				GND MS	orange orange	5 6	white blue	29* 6
Gripper				*no connection*			white brown/gray	28* 22

Single Axis Wiring

In addition to the robot's six motors, the controller can control five additional motors (axis drivers 7 through 11) which operate peripheral devices. Moreover, by disconnecting the gripper cable at the rear of the controller, axis driver 6 can be used for other applications. These additional motors are connected to the controller by means of the driver cards' D9 connector ports. (Refer to the installation instructions in Chapter 4.)

The following table details the wiring for a motor, encoder, and (optional) microswitch when connnected to an axis driver card by means of a D9 connector. Refer to Figures 10-2 and 10-3.

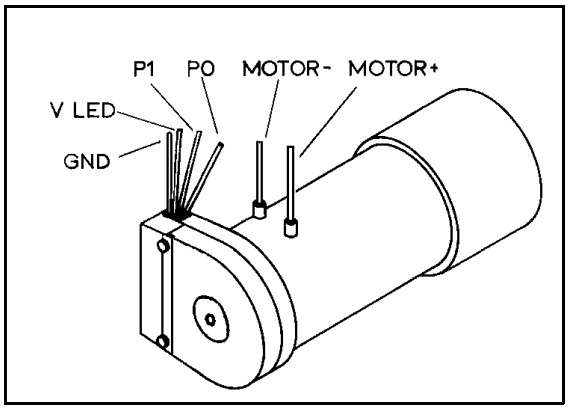


Figure 10-2: Motor Wiring

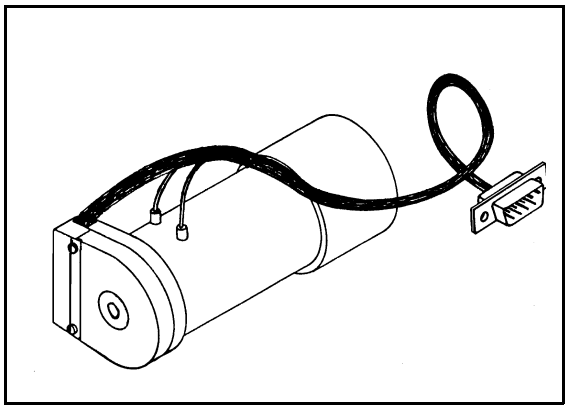


Figure 10-3: Motor with D9 Connector

The last column in the table shows the colors of the leads used in the accessory Motor Kit.

Function	Encoder (PC510) Pad #	D9 Connector Pin #	Motor Kit Lead Color
Motor Power (+)		1	red
Motor Power (–)		9	green
Encoder Phototransistor (P_0)	4	8	brown
Encoder Phototransistor (P_1)	3	6	white
Encoder LED voltage (V_{LED})	2	3	yellow
Encoder Ground (GND)	1	5 + Shield	black
Microswitch Signal (MS) *		4	orange
Microswitch (GND) *		5	orange

Controller-Computer RS232 Cable

The computer and controller communicate on the RS232 channel at 9600 baud, with 8 data bits, 1 stop bit, no parity and XON/XOFF protocol.

The RS232 cable connections between the computer and controller are as follows:

Computer D25 female connector	Controller D25 male connector
Pin 2 (Transmit)	Pin 3 (Receive)
Pin 3 (Receive)	Pin 2 (Transmit)
Pin 7 (Logic GND)	Pin 7 (Logic GND)
Pin 4 to Pin 5	
Pin 6 to Pin 8 and 20	

APPENDIX **A**

Theory of Control

The function of the controller is to instruct the movements of the robot arm or other devices in the robotic system, to monitor these movements, and to make adjustments automatically in order to correct any errors.

Servo Control

Open Loop Control

In open-loop (non-servo) control, the system does not check whether the actual output (position or velocity) equals the desired output.

In open-loop control systems the controller output signal (U_r) is determined only by the input signal (r). If the system response is incorrectly predicted, or if the output signal is affected by other factors, deviations from the desired state will occur. Since no feedback exists, the system is unable to correct output errors.

In open loop robotic control, power is applied to the motors according to a predefined program. The path and speed cannot be precisely predicted, since they are determined by the torque and load on the motors, and other environmental factors.

Closed-Loop Control

In closed-loop control, the control system measures the output signal (C), compares it with the input (desired) signal (r), and corrects any errors.

Figure A-1 compares schematic diagrams of open-loop and closed-loop control systems.

In servo control systems, a feedback device, commonly an optical encoder, measures the output (C) (the amount, speed and direction of motor rotation), converts it to an output signal (U_b), and transmits it to the comparator.

A comparator ($\otimes$) connects the input and feedback signals, produces an error signal equal to the algebraic difference of its two input signals. The comparator output—the error signal—is generally denoted as U_e .

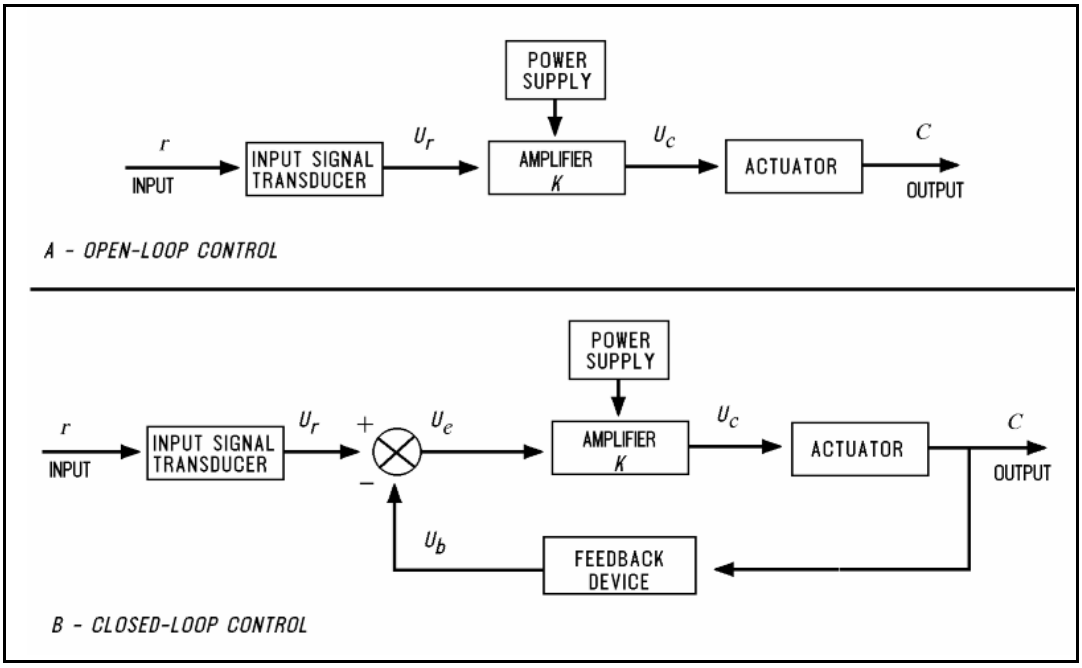


Figure A-1: Open-Loop and Closed-Loop Control

The error signal is the most important value in the closed-loop system. The system aims to reduce U_e to the smallest possible value. When $U_e = 0$, the output signal (the actual state) is equal to the input signal (the desired state).

Digital Control

Unlike analog control systems, in which all signals within the controller are continous analog signals, digital control systems are those in which some of the signals within the controller are discrete digital signals, due to the presence of microprocessors.

In digital control systems, the controller must be capable of converting between analog and digital signals. For the microprocessor to read an analog signal, the signal must first pass through an Analog to Digital Converter. The ADC samples—that is, reads—the signal at periodic intervals and stores the value for the processor to read. For the microprocessor to transmit an analog signal, it must send the discrete values of the signal to a Digital to Analog Converter. The DAC holds the output continuously until given a new value.

Controllers use microprocessors to calculate the state (position, velocity, etc.) error (e) for each motor and the control signal (U_c) which is sent to the motors to correct the error. The control signal is converted to an analog signal by a DAC and then amplified before driving the motor.

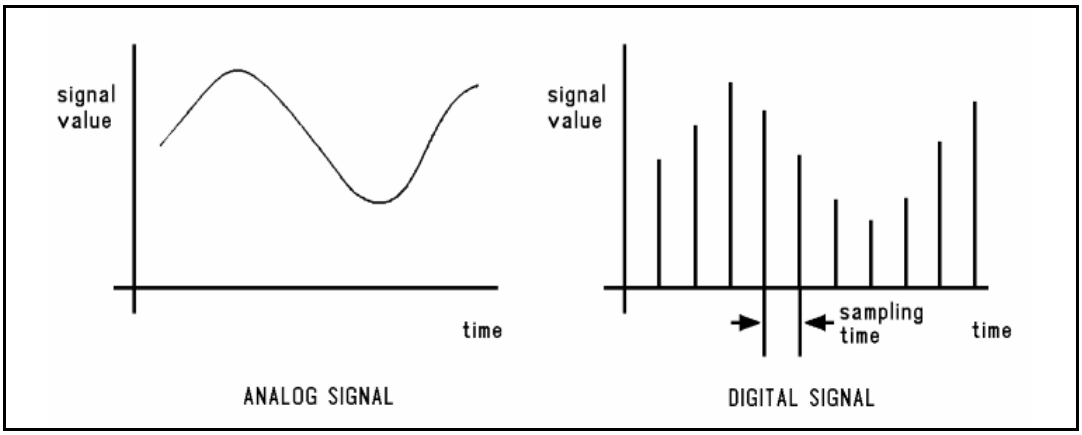


Figure A-2: Analog and Digital Signals

The feedback device measures the actual state and produces an analog signal. The feedback signal is converted by a ADC so that the processor can read it to compute *e*.

A digital control system can be programmed to compute any number of control equations. The processor's control program is a continuous loop whose basic steps are as follows:

1. Read desired state from memory.
2. Read actual state from feedback device.
3. Calculate the state error (*e*).
4. Calculate control signal from control equation.
5. Go back to step 1.

The main difference between digital and analog controllers is the time delay caused by the processor's computations. This time delay is, in effect, the sampling time of the DAC and of the output control signal it produces. If the processor can complete a loop within a few milliseconds, the sampling time will be rapid, and the digital controller will produce an output similar to the equivalent analog controller.

On the other hand, if the processor is slow to make the computations, the controller will be unaware of fast changes in the feedback signal and the control signal will be based on "old" measurements. The greater the delay, the more the response will oscillate, eventually becoming unstable.

Transient and Steady State Responses

When the desired input signal (*r*) changes suddenly, the system will react in two phases, as shown in Figure A-3. The initial reaction to a change in the input signal is called the transient response. The second part of the reaction is known as the steady state response. Once the input signal (*r*) has remained constant for some time, and the error between the input and output signals has stabilized, the system is said to be in steady state. The transition from transient to steady state is not a cleanly defined break.

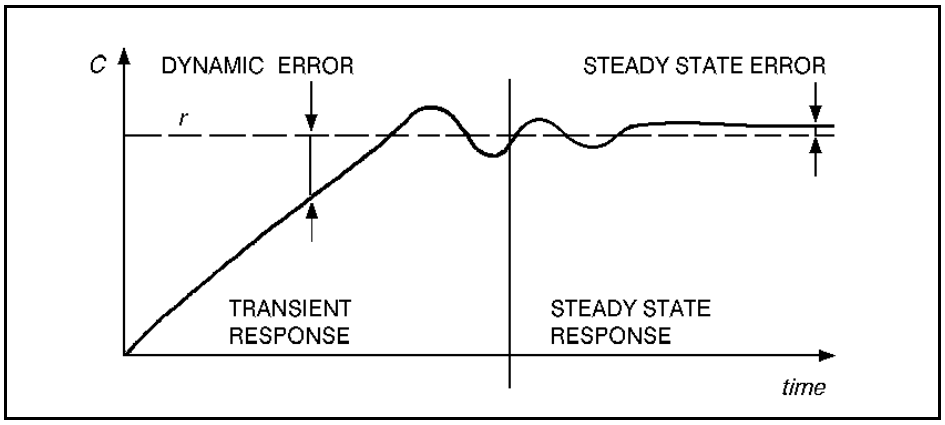


Figure A-3: Transient and Steady States

This constant error, known as the steady state error, should be reduced as much as possible by the control system.

Increasing the amplitude of the controller output signal (that is, increasing the controller gain) can reduce the steady state error and enable a more rapid approach to the steady state value. The greater the controller gain, the faster the system reacts.

However, excessive gain may lead to a phenomenon called overshoot—a rise in the controlled value to a point above the desired value, followed by a drop below the desired value, repeated several times before stabilization. This, in effect, causes the actual value to oscillate around the desired value. Further increase of the controller gain may lead to instability of the entire system— that is, uncontrollable oscillation.

A control system is damped when it reaches steady state without overshoot. A critically damped response is the fastest approach to steady state without overshooting; an overdamped response is a slow approach to steady state.

Figure A-4 shows different transient responses.

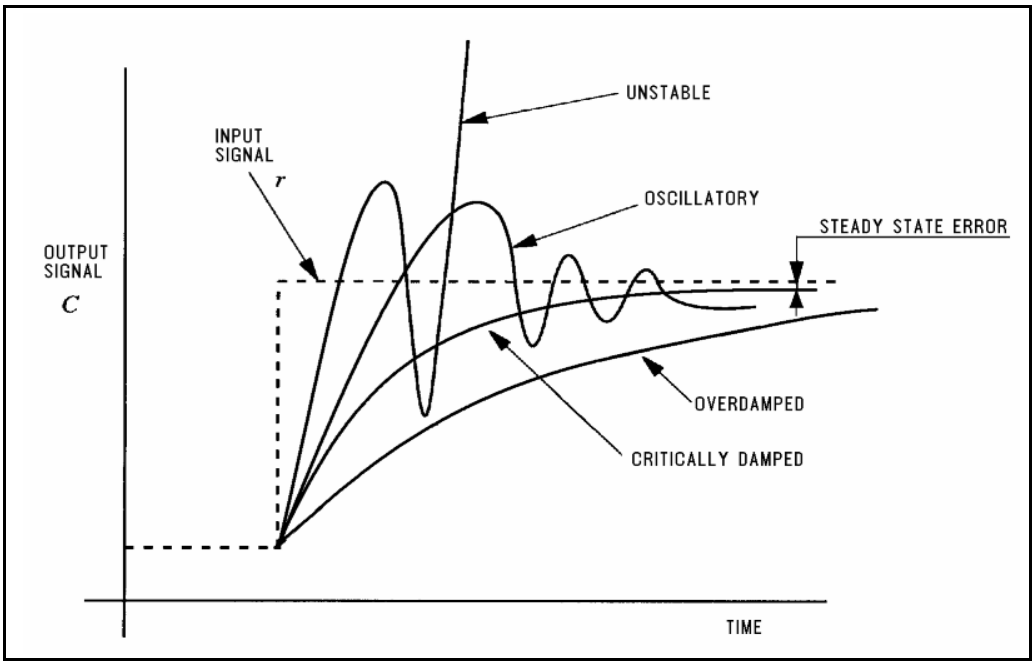


Figure A-4: Transient State Responses

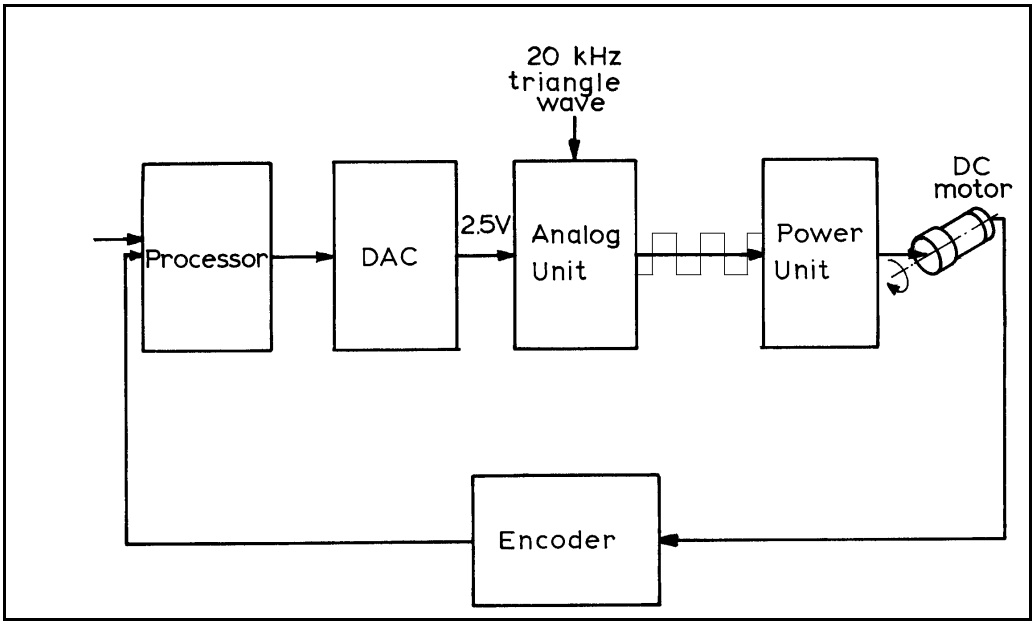


Figure A-5: Controller-A Control Loop

Controller-A Control Process

The basic steps of the **Controller-A** control loop are described below. Refer to Figure A-5. The entire control cycle takes 10ms.

The processor calculates the command position and speed once per cycle. It outputs a digital value to the DAC unit in the range of ±5000.

The analog unit creates a series of pulses, resulting in an average voltage value proportional to the DAC input.

The power unit drives the motor by switching ±24V to it at 20KHz , according to the input pulse. The motor cannot react to this high frequency of switching and is therefore affected by only the average value of the voltage.

This method of controlling the time during which current flows through the motor, rather than controlling the value of the current, is known as PWM (Pulse Width Modulation) control. Refer to Figure A-6.

Once per cycle the processor reads the encoder's count and calculates the motor's position and speed (rate of encoder counts). The processor then compares the actual (output) position and speed values with the desired (input) ones, determines the error values and takes the necessary action to cancel them.

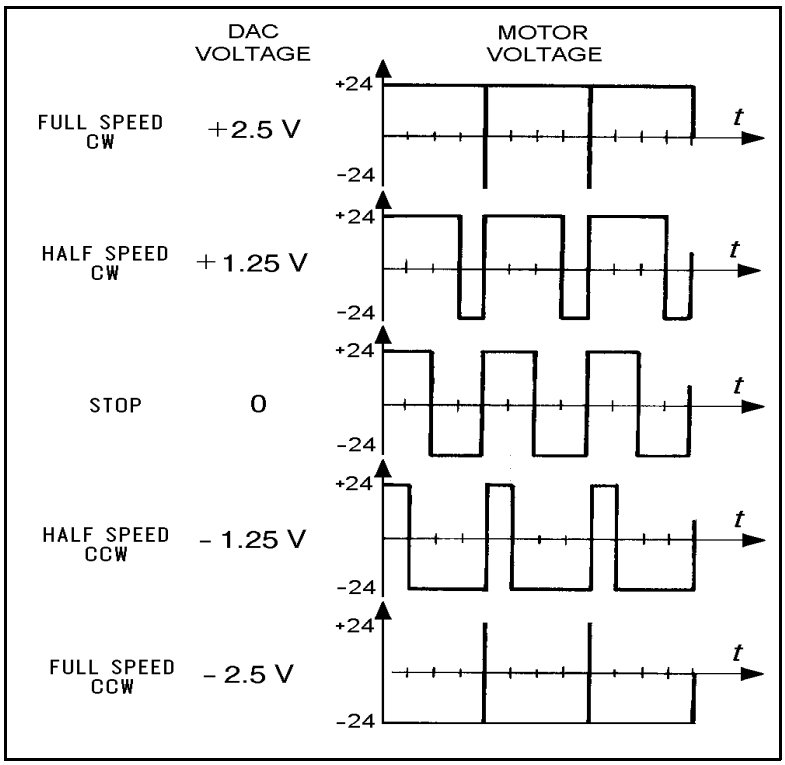


Figure A-6: Controller-A Control Signals

Trajectory Control

For better path performance (that is, to accurately reach the desired state and avoid overshoots), trajectory control profiles, may be programmed into the control system. **Controller-A** offers two profiles: paraboloid and trapezoid. Refer to Figure A-7.

Paraboloid

The paraboloid profile causes the motors to accelerate slowly until maximum speed is reached, then decelerate at the same rate.

Trapezoid

The trapezoid profile causes the motors to accelerate and decelerate quickly at the start and end of movement, with a constant speed along the path.

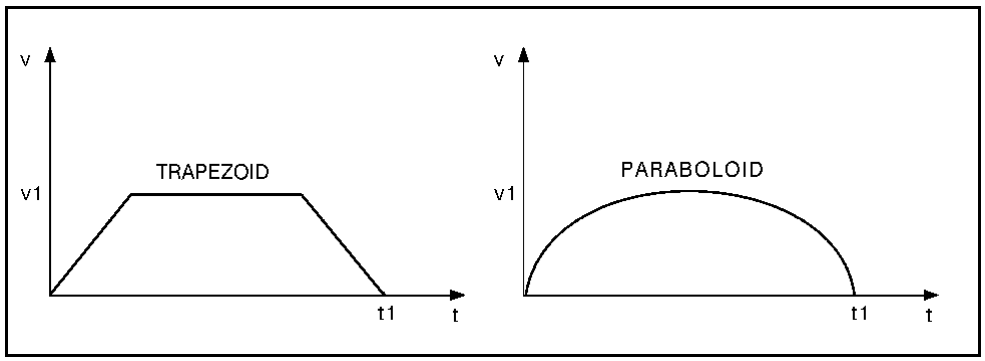


Figure A-7: Trajectory Control Profiles

Path Control

It is desirable that the path and speed of a robot between taught points be predictable. Ideally, the path between consecutive points is traversed at a constant velocity with defined acceleration and deceleration segments.

Along the path, motion of all joints should be proportional, so that all the joints start and finish moving at the same time. The method of coordinating the movement of the joints so that all joints reach the desired location simultaneously is termed joint interpolation.

Point-to-Point Control

Point-to-point control (PTP) involves the positioning of the robot's end effector at given points, without defining the exact path of the end effector between any two points.

Point-to-point control is suitable for applications which require an exact and static position of the end effector at the points where operations will be performed.

In principle, point-to-point control can be used to guide the robot through a large array of positions, thus resulting in a complex path. In order to obtain such a path, points must be defined and recorded in a very close sequence. The number of positions will be limited, however, by the capacity of the control system to maintain positions in memory.

Continuous Path Control

Continuous path control (CP) involves the movement of the end effector between two points along a path defined by a mathematical formula. This method of control is suitable for applications in which the end effector executes operations along a precise trajectory.

During program execution, the control system calculates and plans the path, and instructs the robot motors to move accordingly.

When continuous path control is required, the processor divides the path into short segments, and interpolates the motion of the joints as frequently as possible.

Three type of CP control are possible.

- **Joint Contro**l: Each axis moves according to the trajectory profile. The gripper path is not defined; only the start and end points are defined. All axes start and stop movement at same time.
- **Linear Path Control**: The axes are coordinated in order to move the TCP (tool center point; tip of the gripper) in a straight line according to the trajectory profile.
- **Circular Path Control**: The axes are coordinated in order to move the TCP along a circular path according to the trajectory profile.

The Control Parameters

In the robotic system controller by **Controller-A**, as in common in closed-loop systems, the controlled value (*C*) is measured by an optical encoder. The encoder signals serve as feedback to the controller, enabling it to correct any deviations from the desired value.

Since control systems cannot react immediately to the input signal, there will always be a lag between the generation of an error signal and the actual correction of the controlled value.

The PID (proportional, integral, differential) control parameters allow the controller to adapt to various conditions of operation, such as overcoming nonlinear functions in the system.

Proportional Control

The proportional parameter is the gain of the control system. Its value determines the reaction time to position errors.

When a position error exists (that is, the actual motor position is off by a certain amount of encoder counts), the processor multiplies the error by the proportional parameter and adds the product to the DAC value, thereby reducing the error.

The proportional parameter is the parameter in the PID control system which acts most quickly in reducing the position error, especially during motion. It is also the first parameter to respond to position errors when the robot has stopped at a target position.

The greater the proportional parameter, the faster the sytem responds and reduces the error. But, using too great a value for the proportional parameter will cause the axis to oscillate.

The main disadvantage of proportional control is that it cannot completely cancel the error, because once it has reduced the error it cannot generate enough power to overcome friction in the system and propel the axis to its target position.

Even in steady state, under load, the controlled value (output signal) will always be different from the desired value (input signal). The steady state error can be reduced by increasing the gain, but this will increase the oscillation and reduce stability.

Differential Control

In differential control, the controller output (C) is a function of the rate at which the error (U_e) changes. The faster the rate of change of the error, the greater the controller output (C). In other words, the controller is sensitive to the slope of the error signal.

The differential parameter is responsible for reducing the speed error. The control system calculates the actual speed once per cycle and compares it to the desired value. While the robot is accelerating (during the first part of path) the differential acts as a driving factor.

While the robot is decelerating (during the second, and last, part of path), the differential acts as a braking factor. A good differential setting will result in a clean and smooth motion along the entire path. Lack of the differential will cause overshoot at the end of path. High differential values will cause small vibrations along the path.

In this control method, the controller predicts the value of the error in accordance with the error signal slope, and causes the correction to take place in advance. However, if the error is constant and unchanging, differential control will not be able to reduce the error to zero.

Integral Control

In integral control, all the state errors which have been recorded each cycle are totalled and their sum is multiplied by the integral parameter value.

In integral control, the controller output (C) reduces the error signal (U_e) to zero at a rate proportional to the size and duration of the error. In other words, the greater the error, the greater the controller output; and, the longer the duration of the error, the greater the controller output.

The main advantage of integral control is that the steady state error is always reduced to zero since its value increases each cycle, thus strengthening the control system's ability to react and reduce the error. However, using too great a value for the integral parameter may cause overshoots, while too small a value may prevent the cancellation of a steady state error.

Unlike the proportional parameter, the integral parameter takes effect more slowly and is less noticeable during motion. However, when the axis comes to a complete stop and the proportional parameter can no longer reduce the steady state error, the integral parameter takes over and can cancel the error completely.

Proportional–Integral–Differential Control

The PID control method enables optimal exploitation of all three types of control—proportion, integral and differential. In this manner, it creates an output response which follows the input signal closely, without gaps or lags, in both slow and rapid processes, including those in which the load is in a constant state of change. In summary, the PID control parameters serve the following functions:

- **Proportional Parameter**: Enables fast and powerful reactions of the arm to movement commands. Responsible for the repeatability of the motion.
- **Integral Parameter**: Assists the proportional parameter in eliminating steady state errors.
- **Differential Parameter**: Provides the required damping.

Offset

Control theories often assume complete linearity; that is, the speed is proportional to the power supplied to the motor.

However, at low levels of power, the motor will not move, mainly due to friction; that is, the static friction is higher than the dynamic friction. This is a non-linearity. Figure A-8 shows linearity and non-linearity.

The offset is a threshold level of the DAC. Above this DAC value the control system acts as a linear system. Below this value, the control system acts as an on/off system. Figure A-9 shows the offset.

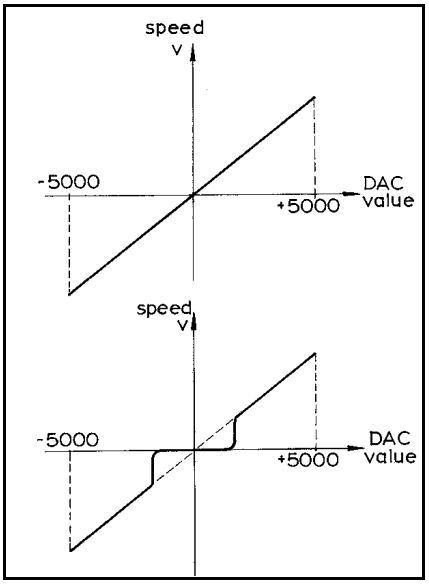


Figure A-8:
Linearity and Non-Linearity

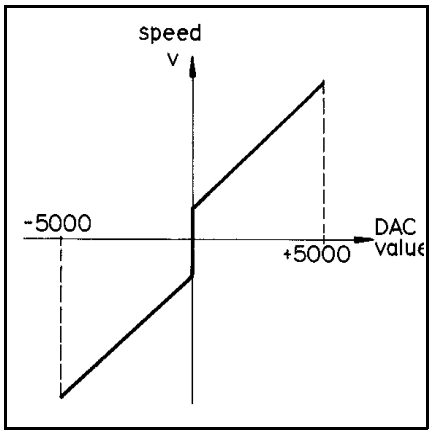


Figure A-9:
Control System Offset

Changing Parameter Values

The control system parameters of **Controller-A** are factory-set, and are suitable for most robotic applications.

Although parameter values can be manipulated by user commands, only experienced users should attempt to do so.

For more details refer to the ***ACL** Reference Guide*.

16.3. Darasheet LS7166

LSI Computer Systems, Inc. 1235 Walt Whitman Road, Melville, NY 11747 (631) 271-0400 FAX (631) 271-0405

LS7166

August 2006

24-BIT QUADRATURE COUNTER

FEATURES:

- Programmable modes are: Up/Down, Binary, BCD, 24 Hour Clock, Divide-by-N, x1 or x2 or x4 Quadrature and Single-Cycle.
- DC to 25MHz Count Frequency.
- 8-Bit I/O Bus for uP Communication and Control.
- 24-Bit comparator for pre-set count comparison.
- Readable status register.
- Input/Output TTL and CMOS compatible.
- 3V to 5.5V operation (V_{DD} - V_{SS}).
- LS7166 (DIP); LS7166-S (SOIC); LS7166-TS24 (24-Pin TSSOP) **- See Figure 1 -**

GENERAL DESCRIPTION:

The LS7166 is a CMOS, 24-bit counter that can be programmed to operate in several different modes. The operating mode is set up by writing control words into internal control registers (see Figure 8). There are three 6-bit and one 2-bit control registers for setting up the circuit functional characteristics. In addition to the control registers, there is a 5-bit output status register (OSR) that indicates the current counter status. The IC communicates with external circuits through an 8-bit three state I/O bus. Control and data words are written into the LS7166 through the bus. In addition to the I/O bus, there are a number of discrete inputs and outputs to facilitate instantaneous hardware based control functions and instantaneous status indication.

REGISTER DESCRIPTION:

Internal hardware registers are accessible through the I/O bus (D0 - D7) for READ or WRITE when $\overline{CS}$ = 0. The C/$\overline{D}$ input selects between the control registers (C/$\overline{D}$ = 1) and the data registers (C/$\overline{D}$ = 0) during a READ or WRITE operation. (See Table 1)

PIN ASSIGNMENTS - Top View

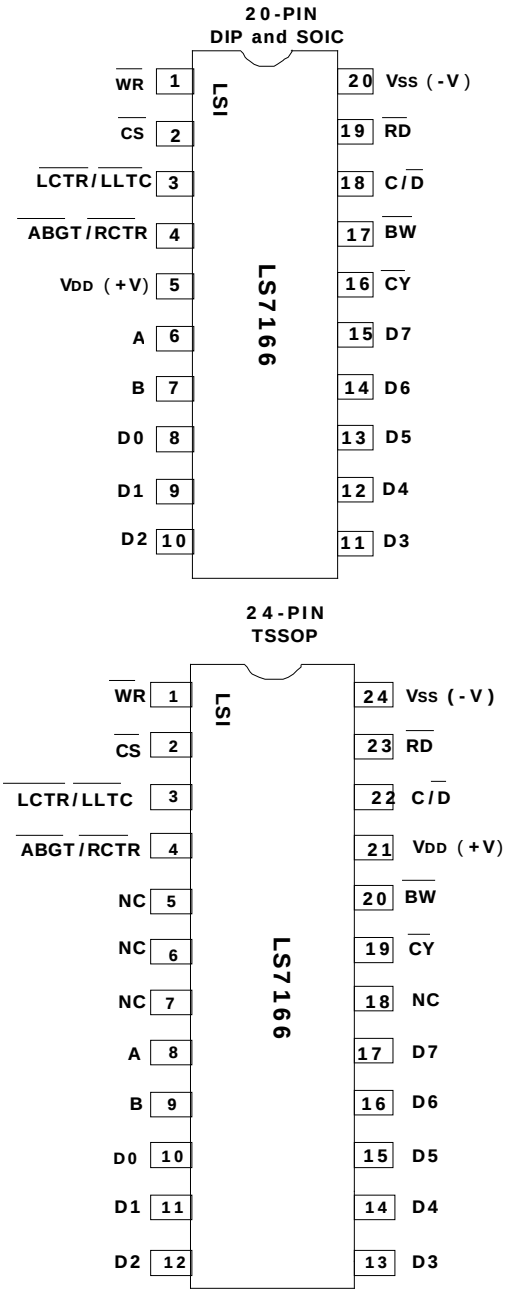
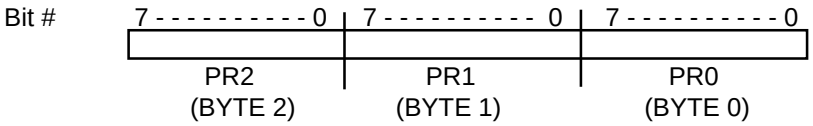


FIGURE 1

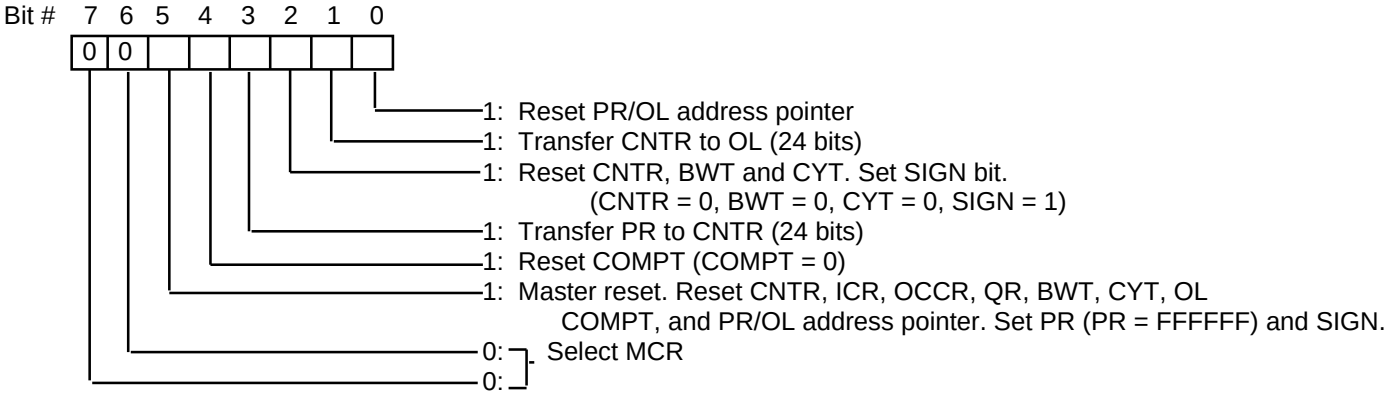
PR (Preset register). The PR is the input port for the CNTR. The CNTR is loaded with a 24 bit data via the PR. The data is first written into the PR in 3 WRITE cycle sequence of Byte 0 (PR0), Byte 1 (PR1) and Byte 2 (PR2).
The address pointer for PR0/PR1/PR2 is automatically incremented with each write cycle.
Accessed by**:** WRITE when C/$\overline{D}$ = 0, $\overline{CS}$ = 0.



Standard Sequence for Loading PR and Reading CNTR**:**

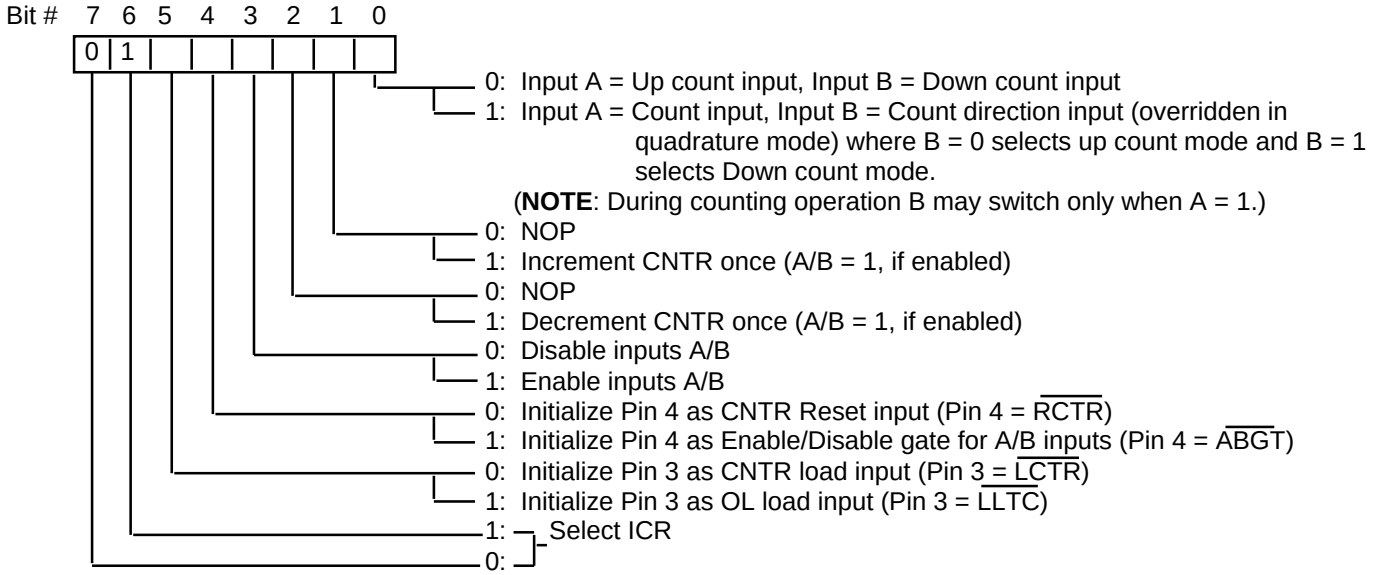
```
1 ⟶ MCR     ;  Reset PR address pointer
WRITE PR    ;  Load Byte 0 and into PR0 increment address
WRITE PR    ;  Load Byte 1 and into PR1 increment address
WRITE PR    ;  Load Byte 2 and into PR3 increment address
8 ⟶ MCR     ;  Transfer PR to CNTR
```

MCR (Master Control Register). Performs register reset and load operations. Writing a "non-zero" word to MCR does not require a follow-up write of an "all-zero" word to terminate a designated operation.
Accessed by**:** WRITE when C/$\overline{D}$ = 1, $\overline{CS}$ = 0.



NOTE: Control functions may be combined.

ICR (Input Control Register). Initializes counter input operating modes.
Accessed by**:** WRITE when C/$\overline{D}$ = 1, $\overline{CS}$ = 0.



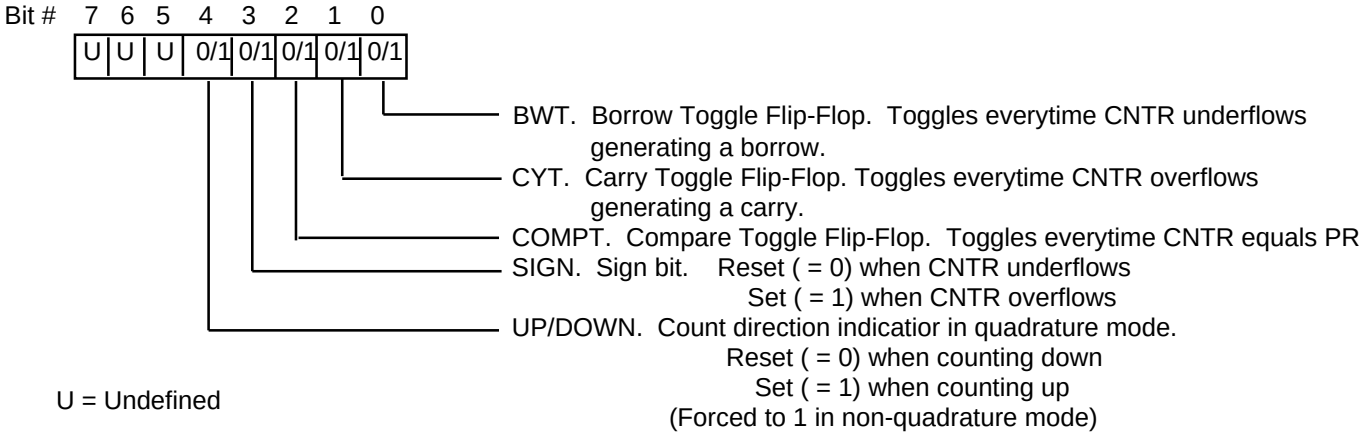
NOTE: Control functions may be combined.

TABLE 1 - Register Addressing Modes

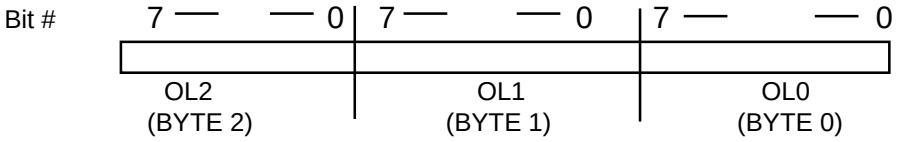
D7	D6	C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	COMMENT
X	X	X	X	X	1	Disable Chip for READ/WRITE
0	0	1	1	⊓⊔⊓	0	Write to Master Control Register (MCR)
0	1	1	1	⊓⊔⊓	0	Write to input control register (ICR)
1	0	1	1	⊓⊔	0	Write to output/counter control register (OCCR)
1	1	1	1	⊓⊔⊓	0	Write to quadrature register (QR)
X	X	0	1	⊓⊔	0	Write to preset register (PR) and increment register address counter.
X	X	0	⊓⊔	1	0	Read output latch (OL) and increment register address counter
X	X	1	⊓⊔⊓	1	0	Read output status register (OSR).

X = Don't Care

OSR (Output Status Register). Indicates CNTR status: Accessed by**:** READ when C/$\overline{D}$ = 1, $\overline{CS}$ = 0.



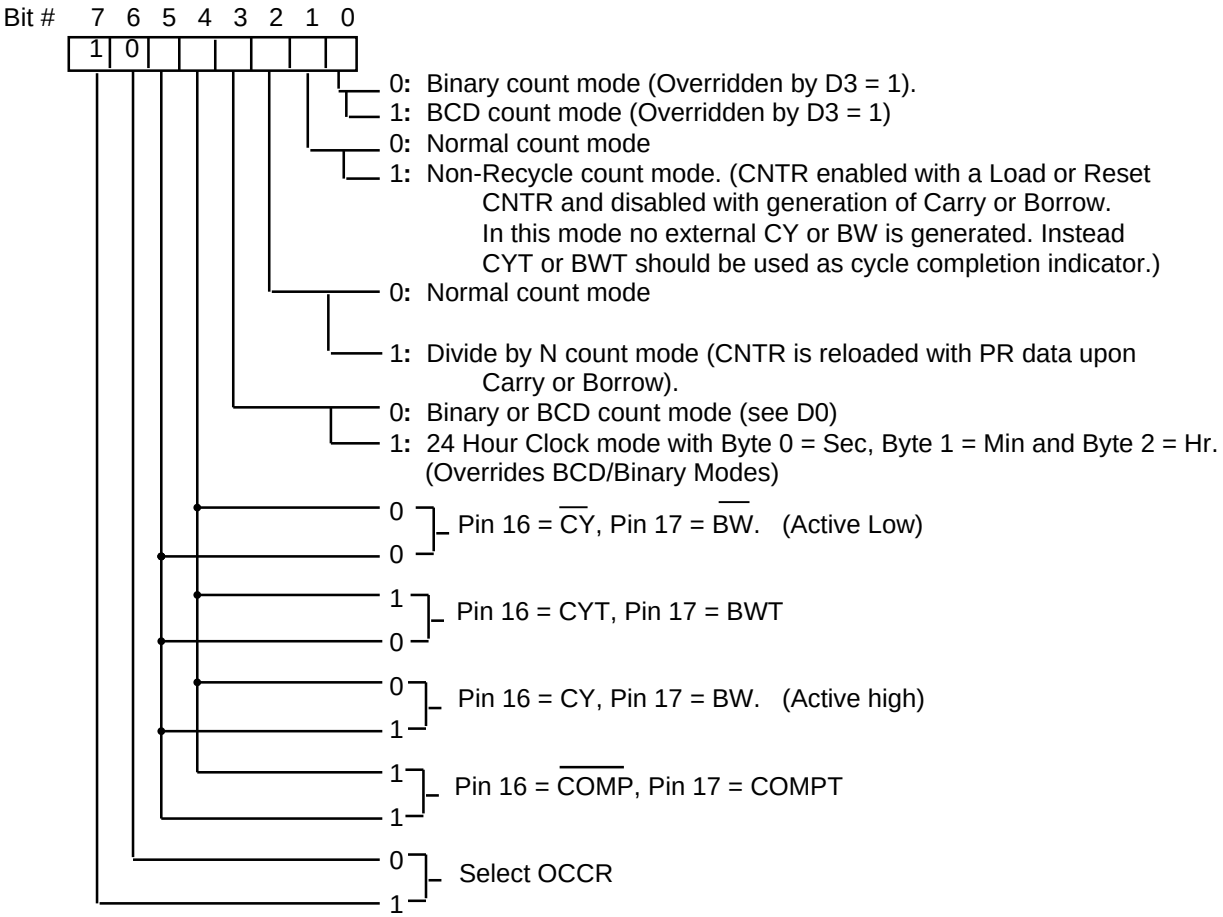
OL(Output latch). The OL is the output port for the CNTR. The 24 bit CNTR Value at any instant can be accessed by performing a CNTR to OL transfer and then reading the OL in 3 READ cycle sequence of Byte 0 (OL0), Byte 1 (OL1) and Byte 2 (OL2). The address pointer for OL0/OL1/OL2 is automatically incremented with each READ cycle. Accessed by**:** READ when C/$\overline{D}$ = 0, $\overline{CS}$ = 0.



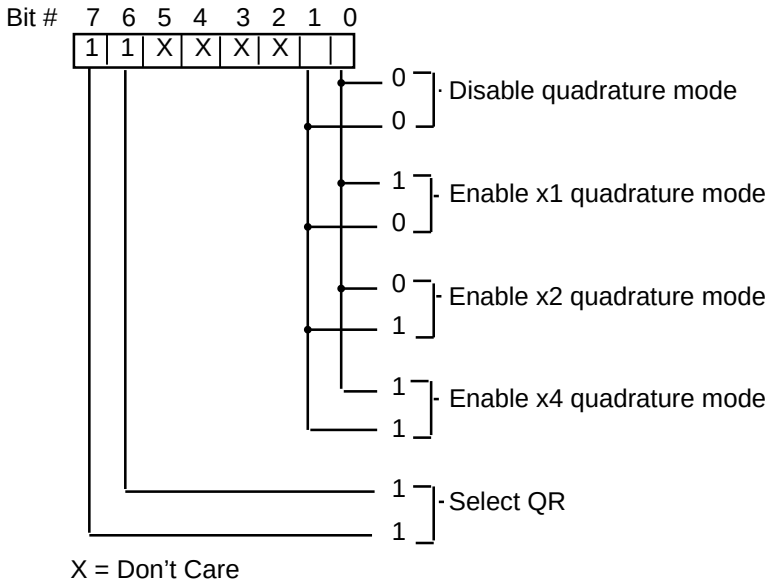
Standard Sequence for Loading and Reading OL**:**

```
3 ⟶ MCR     ; Reset OL address pointer and Transfer CNTR to OL
READ OL     ; Read Byte 0 and increment address
READ OL     ; Read Byte 1 and increment address
READ OL     ; Read Byte 2 and increment address
```

OCCR (Output Control Register) Initializes CNTR and output operating modes.
Accessed by **:** WRITE when C/$\overline{D}$ = 1, $\overline{CS}$ = 0.



QR (Quadrature Register). Selects quadrature count mode (See Fig. 7)
Accessed by**:** WRITE when C/$\overline{D}$ = 1, $\overline{CS}$ = 0.



X = Don't Care

I/O DESCRIPTION:
(See REGISTER DESCRIPTION for I/O Prgramming.)

Data-Bus (D0 - D7) (Pin 8 - Pin 15). The 8-line data bus is a three-state I/O bus for interfacing with the system bus.

$\overline{CS}$ (Chip Select Input) (Pin 2). A logical "0" at this input enables the chip for Read and Write.

$\overline{RD}$ (Read Input) (Pin 19). A logical "0" at this input enables the OSR and the OL to be read on the data bus.

$\overline{WR}$ (Write Input) (Pin 1). A logical "0" at this input enables the data bus to be written into the control and data registers.

C/$\overline{D}$ (Control/Data Input) (Pin 18). A logical "1" at this input enables a control word to be written into one of the four control registers or the OSR to be read on the I/O bus. A logical "0" enables a data word to be written into the PR, or the OL to be read on the I/O bus.

A (Pin 6). Input A is a programmable count input capable of functioning in three different modes, such as up count input, down count input and quadrature input. In non-quadrature mode, the counter advances on the rising edge of Input A.

B (Pin 7). Input B is also a programmable count input that can be programmed to function either as down count input, or count direction control gate for input A, or quadrature input. In non- quadrature mode, and when programmed as the Down Count input, the counter advances on the rising edge of Input B. When B is programmed as the count direction control gate, B = 0 enables A as the Up Count input and B = 1 enables A as the Down Count input. When programmed as the direction input, B can switch state only when A is high.

$\overline{ABGT}$/$\overline{RCTR}$ (Pin 4). This input can be programmed to function as either inputs A and B enable gate or as external counter reset input. A logical "0" is the active level on this input. In non-quadrature mode, if Pin 4 is programmed as A and B enable gate input, it may switch state only when A is high (if A is clock and B is direction) or when both A and B are high (if A and B are clocks). In quadrature mode, if Pin 4 is programmed as A and B enable gate, it may switch state only when either A or B switches.

$\overline{LCTR}$/$\overline{LLTC}$ (Pin 3**)** This input can be programmed to function as the external load command input for either the CNTR or the OL. When programmed as counter load input, the counter is loaded with the data contained in the PR. When programmed as the OL load input, the OL is loaded with data contained in the CNTR. A logical "0" is the active level on this input.

$\overline{CY}$ (Pin 16) This output can be programmed to serve as one of the following:
A. $\overline{CY}$. Complemented Carry out (active "0").
B. CY. True Carry out (active "1").
C. CYT. Carry Toggle flip-flop out.
D. $\overline{COMP}$. Comparator out (active "0")

$\overline{BW}$ (Pin 17) This output can be programmed to serve as one of the following:
A. $\overline{BW}$. Complemented Borrow out (active "0").
B. BW. True Borrow out (active "1").
C. BWT. Borrow Toggle flip-flop out.
D. COMPT. Comparator Toggle output.

V_{DD} (Pin 5**)** Supply voltage positive terminal.

V_{SS} (Pin 20**)** Supply voltage negative terminal.

Absolute Maximum Ratings:

Parameter	Symbol	Values	Unit
Voltage at any input	V_{IN}	V_{SS} - 0.3 to V_{DD} + 0.3	V
Operating Temperature	T_A	-40 to +125	^oC
Storage Temperature	T_{STG}	-65 to +150	^oC
Supply Voltage	V_{DD} - V_{SS}	+7.0	V

DC Electrical Characteristics. (All voltages referenced to V_{SS}.
T_A = 0˚ to 85˚C, V_{DD} = 3V to 5.5V, fc = 0, unless otherwise specified)

Parameter	Symbol	Min. Value	Max.Value	Unit	Remarks
Supply Voltage	V_{DD}	3.0	5.5	V	-
Supply Current	I_{DD}	-	350	µA	Outputs open
Input Low Voltage	V_{IL}	0	0.8	V	-
Input High Voltage	V_{IH}	2.0	V_{DD}	V	-
Output Low Voltage	V_{OL}	-	0.4	V	4mA Sink, V_{DD} = 5V
Output High Voltage	V_{OH}	2.5	-	V	200µA Source, V_{DD} = 5V
Input Current	-	-	15	nA	Leakage Current
Output Source Current	I_{SRC}	200	-	µA	V_{OH} = 2.5V, V_{DD} = 5V
Output Sink Current	I_{SINK}	4	-	mA	V_{OL} = 0.4V, V_{DD} = 5V
Data Bus Off-State Leakage Current	-	-	15	nA	-

TRANSIENT CHARACTERISTICS (See Timing Diagrams in Fig. 2 thru Fig. 7, V_{DD} = 3V to 5.5V, T_A = 0° to 85°C, unless otherwise specified)

Parameter	Symbol	Min.Value	Max.Value	Unit
Clock A/B "Low"	T_{CL}	18	No Limit	ns
Clock A/B "High"	T_{CH}	22	No Limit	ns
Clock A/B Frequency (See NOTE 1)	fc	0	25	MHz
Clock UP/DN Reversal Delay	T_{UDD}	100	-	ns
LCTR Positive edge to the next A/B positive or negative edge delay	T_{LC}	100	-	ns
Clock A/B to CY/BW/COMP "low" propagation delay	T_{CBL}	-	65	ns
Clock A/B to CY/BW/COMP "high" propagation delay	T_{CBH}	-	85	ns
LCTR and LLTC pulse width	T_{LCW}	60	-	ns
Clock A/B to CYT, BWT and COMPT "high" propagation delay	T_{TFH}	-	100	ns
Clock A/B to CYT, BWT and COMPT "low" progagation delay	T_{TFL}	-	100	ns
WR pulse width	T_{WR}	60	-	ns
RD to data out delay (C_L=20pF)	T_R	-	110	ns
CS, RD Terminate to Data-Bus Tri-State	T_{RT}	-	30	ns
Data-Bus set-up time for WR	T_{DS}	30	-	ns
Data-Bus hold time for WR	T_{DH}	30	-	ns
CS set-up time for RD	T_{SRS}	0	-	ns
CS hold time for RD	T_{SRH}	0	-	ns
Back to Back RD delay	T_{RR}	60	-	ns
RD to WR delay	-	60	-	ns
C/D set-up time for RD	T_{CRS}	0		
C/D hold time for RD	T_{CRH}	30		
C/D set-up time for WR	T_{CWS}	30	-	ns
C/D hold time for WR	T_{CWH}	30	-	ns
CS set-up time for WR	T_{SWS}	60	-	ns
CS hold time for WR	T_{SWH}	0	-	ns
Back to Back WR delay	T_{WW}	60	-	ns
WR to RD delay	-	60	-	ns
Quadrature Mode:				
Clock A/B Validation delay (See NOTE 1)	T_{CQV}	-	160	ns
A and B phase delay	T_{PH}	208	-	ns
Clock A/B frequency	f_{CQ}	-	1.2	MHz
CY, BW, COMP pulse width	T_{CBW}	85	200	ns

NOTE 1: In quadrature mode A/B inputs are filtered and required to be stable for at least T_{CQV} length to be valid.

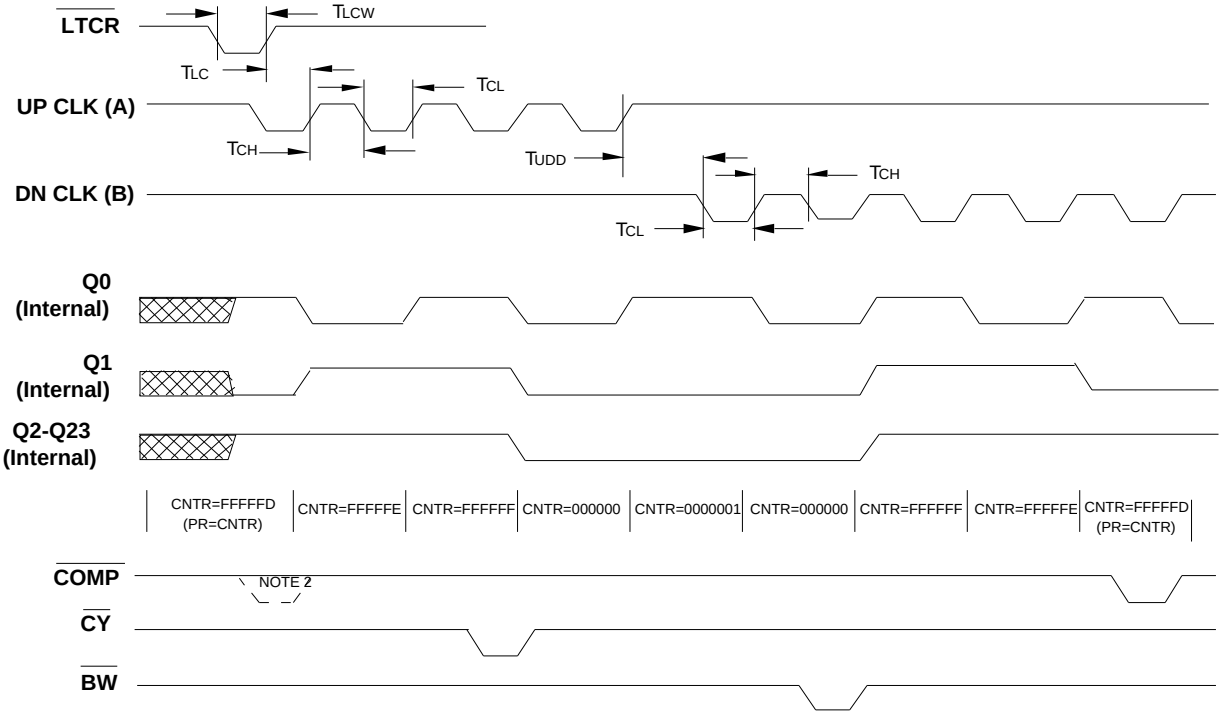


FIGURE 2 . LOAD COUNTER, UP CLOCK, DOWN CLOCK, COMPARE OUT, CARRY, BORROW

NOTE 1: The counter in this example is assumed to be operating in the binary mode.
NOTE 2: No COMP output is generated here, although PR = CNTR. COMP output is disabled with a counter load command and enabled with the rising edge of the next clock, thus eliminating invalid COMP outputs whenever the CNTR is loaded from the PR.
NOTE 3: When UP Clock is active, the DN Clock should be held "HIGH" and vice versa.

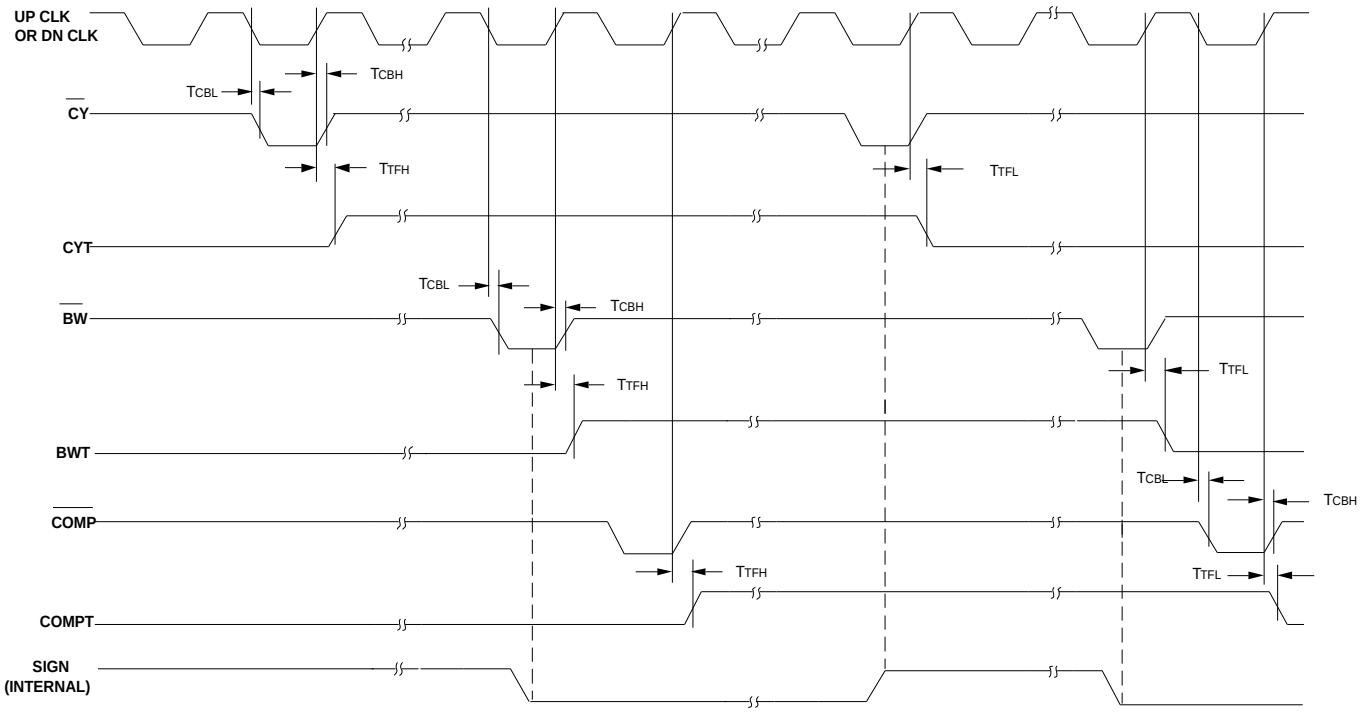


FIGURE 3. CLOCK TO $\overline{CY}$/$\overline{BW}$ OUTPUT PROPAGATION DELAYS

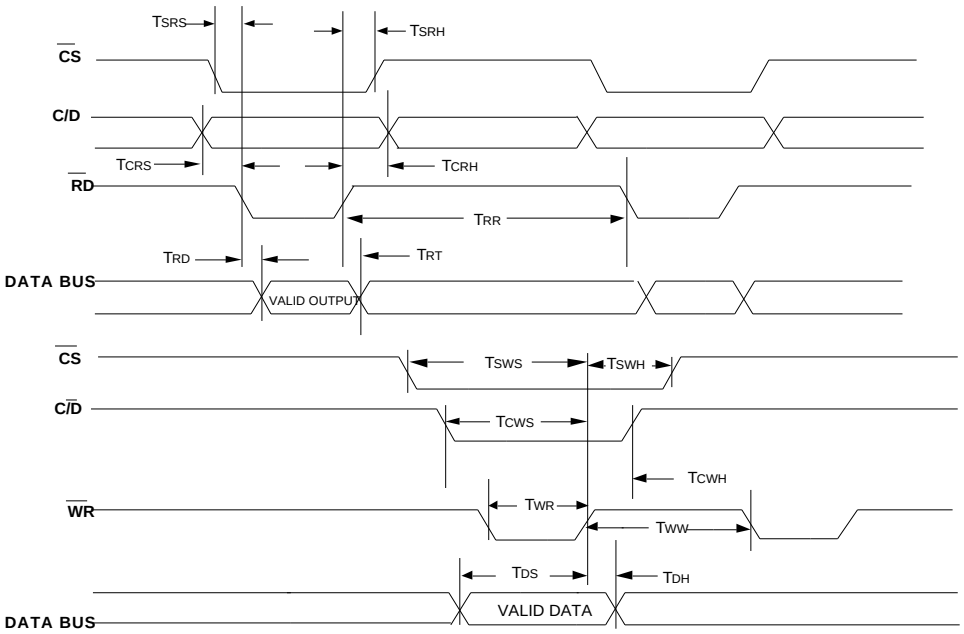
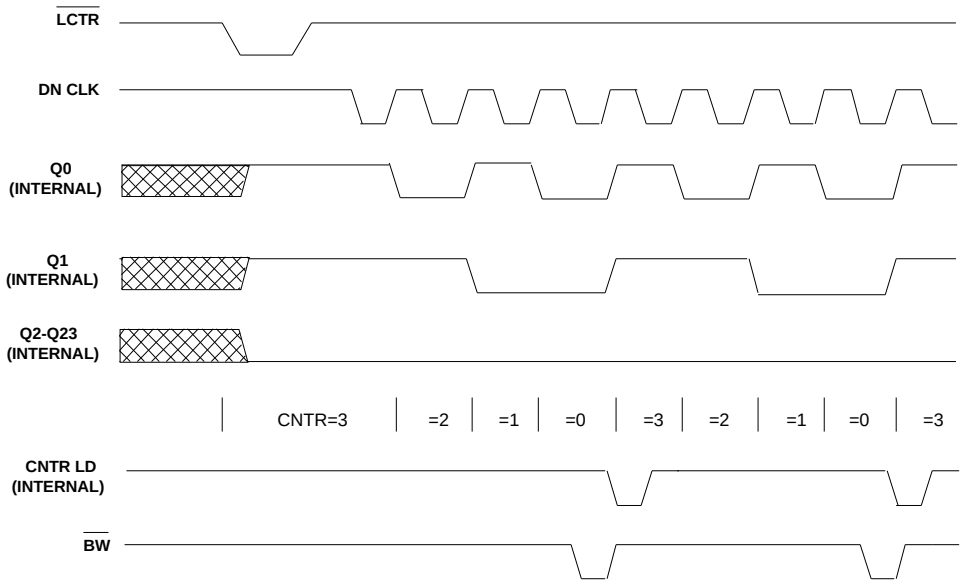


FIGURE 4. READ/WRITE CYCLES



NOTE: EXAMPLE OF DIVIDE BY 4 IN DOWN COUNT MODE

FIGURE 5. DIVIDE BY N MODE

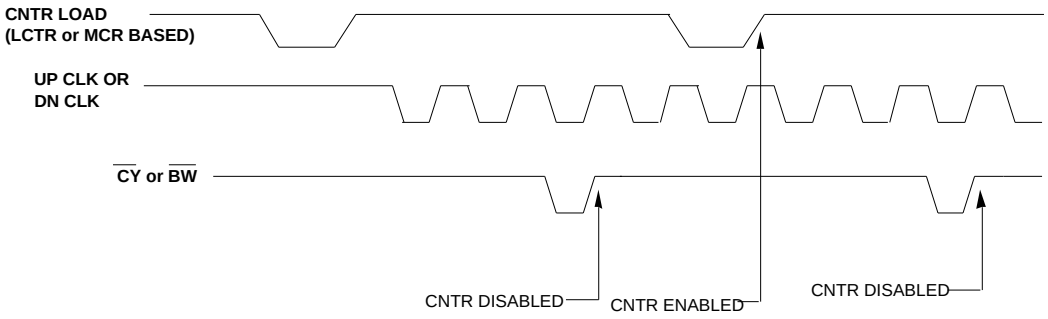


FIGURE 6 . CYCLE ONCE MODE

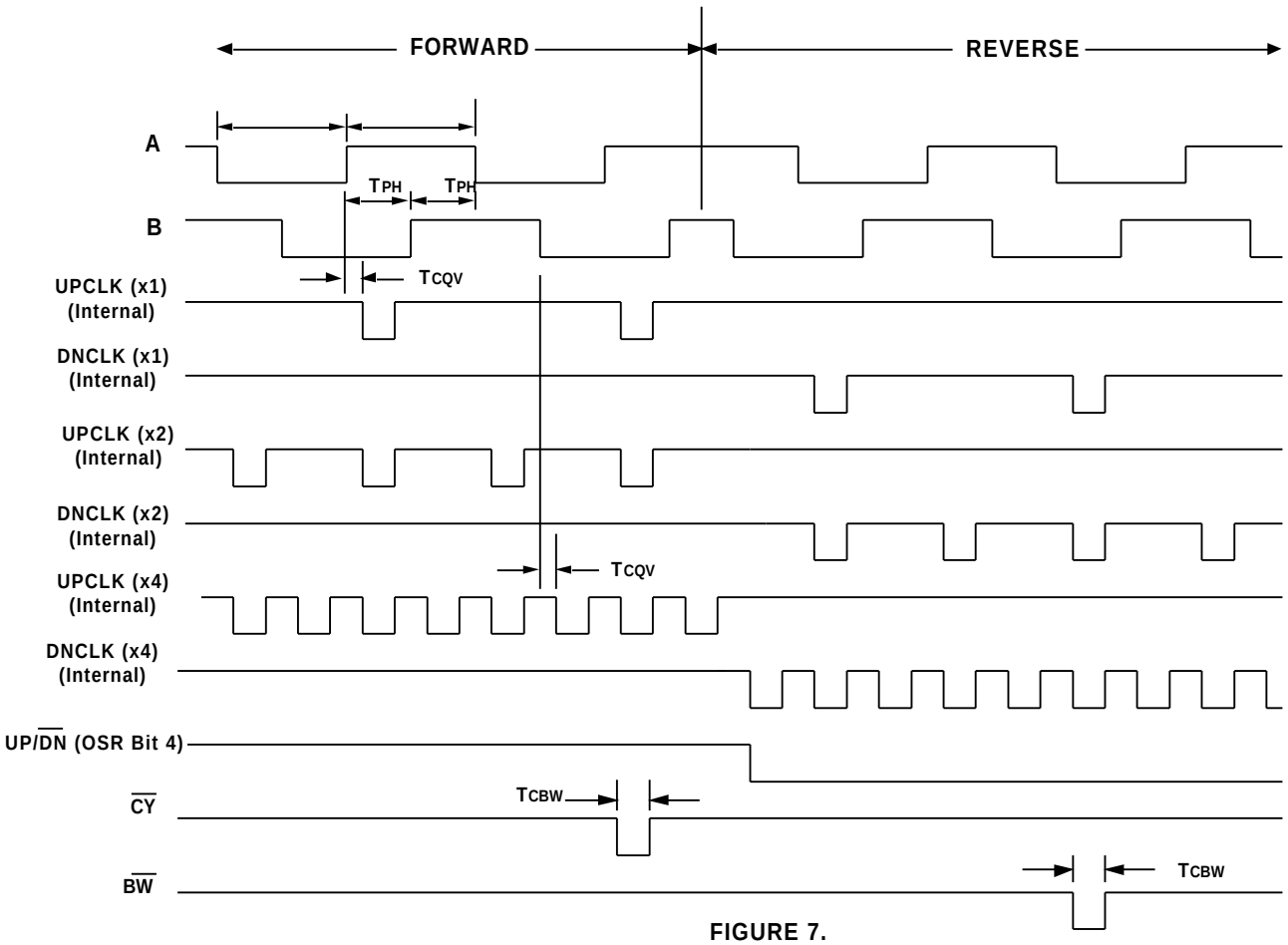


FIGURE 7.

QUADRATURE MODE INTERNAL CLOCKS

7166-110503-9

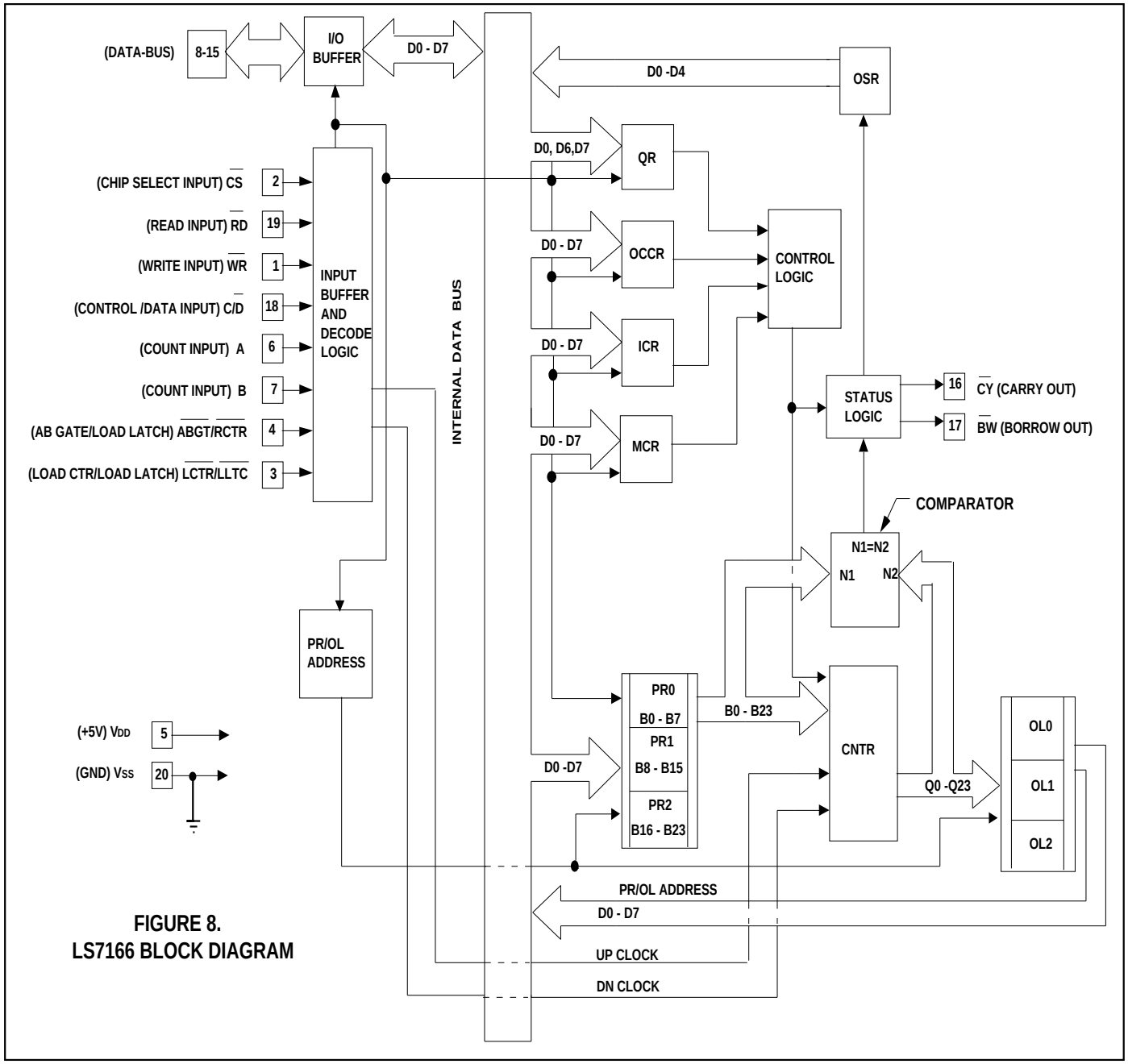
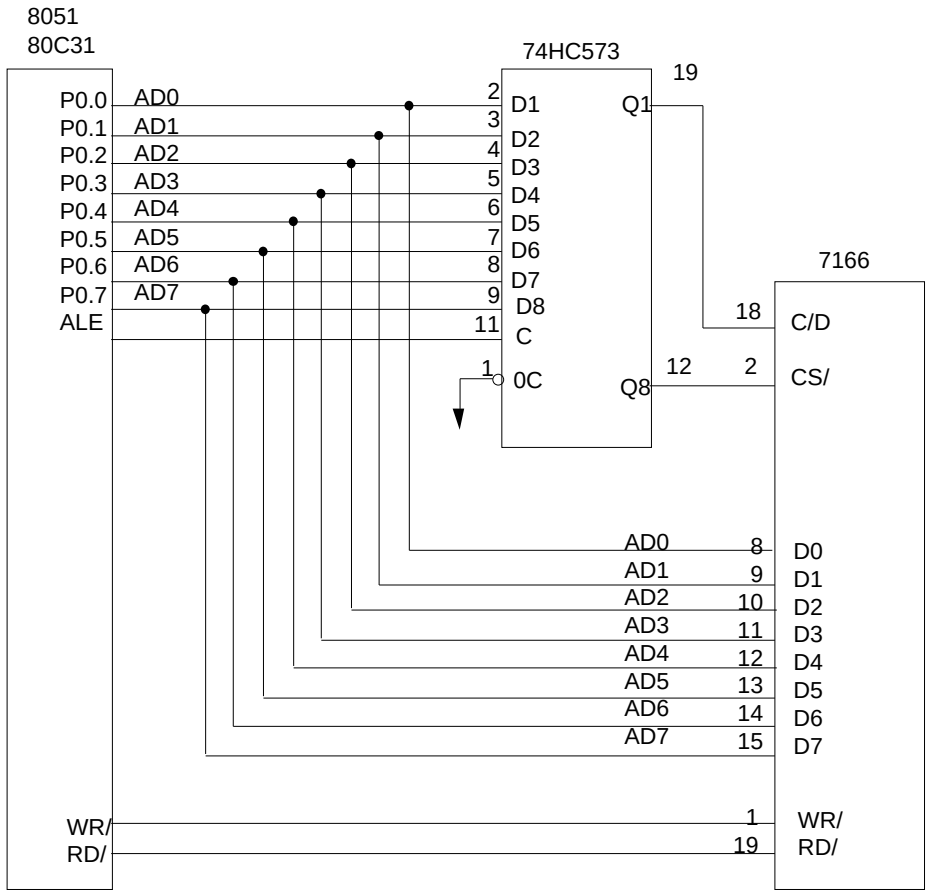


FIGURE 8.
LS7166 BLOCK DIAGRAM

7166-110103-10

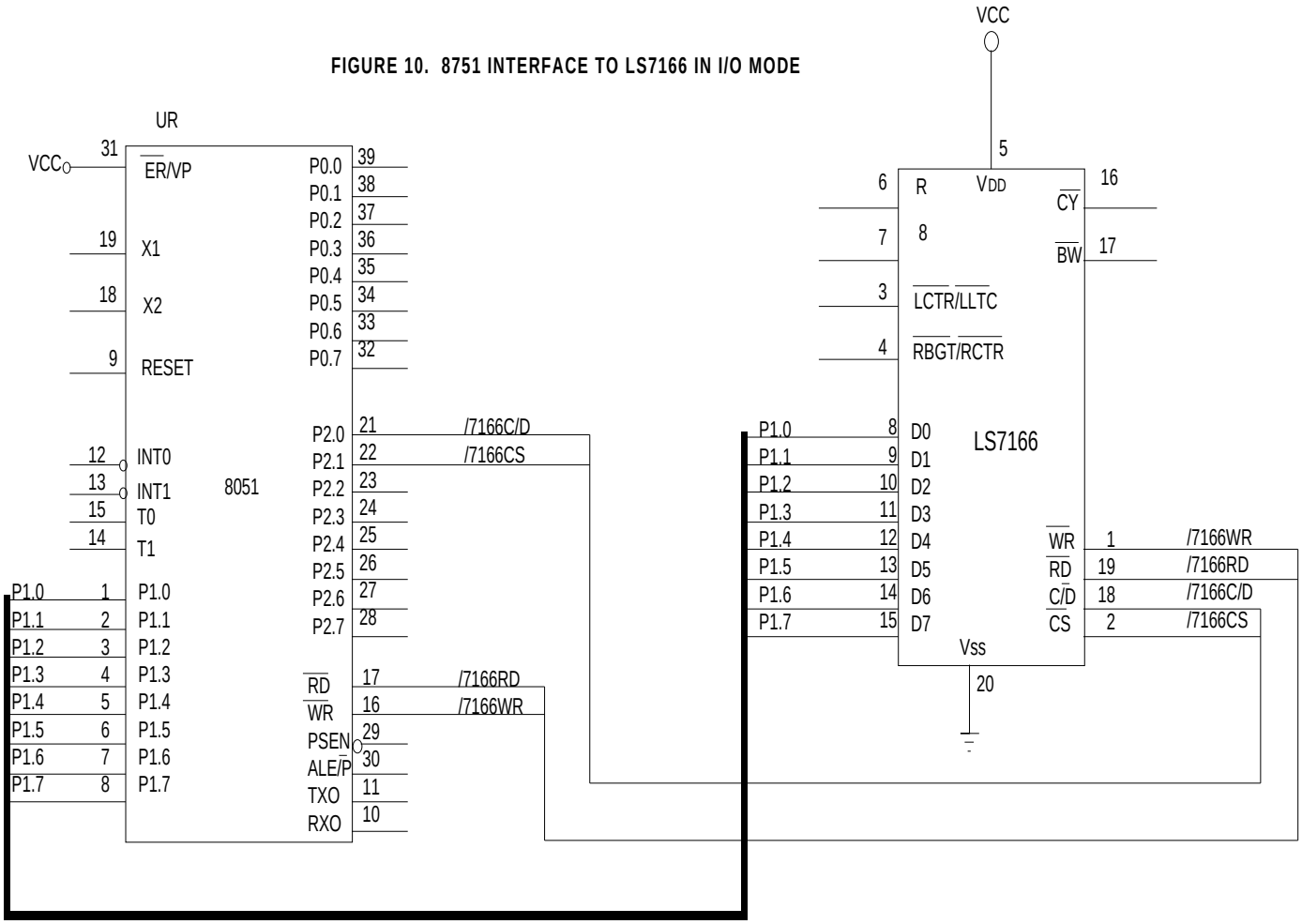
FIGURE 9. 80C31/8051 TO LS7166 INTERFACE IN EXTERNAL ADDRESS MODE



NOTE: Port_0 is open drain output. Add pull-up resistors to all Port_0 i/0 lines.

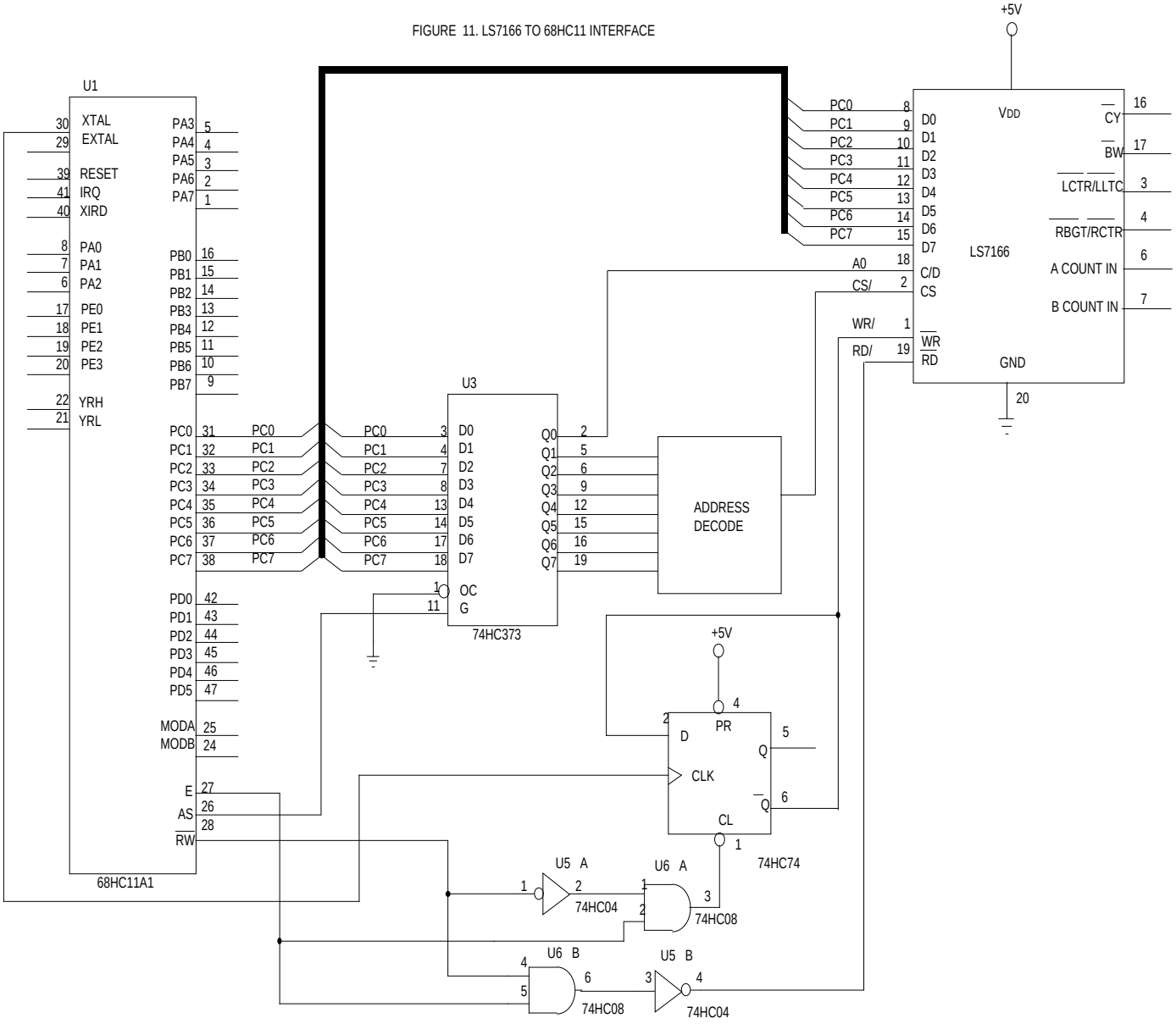
7166-110503-11

FIGURE 10. 8751 INTERFACE TO LS7166 IN I/O MODE



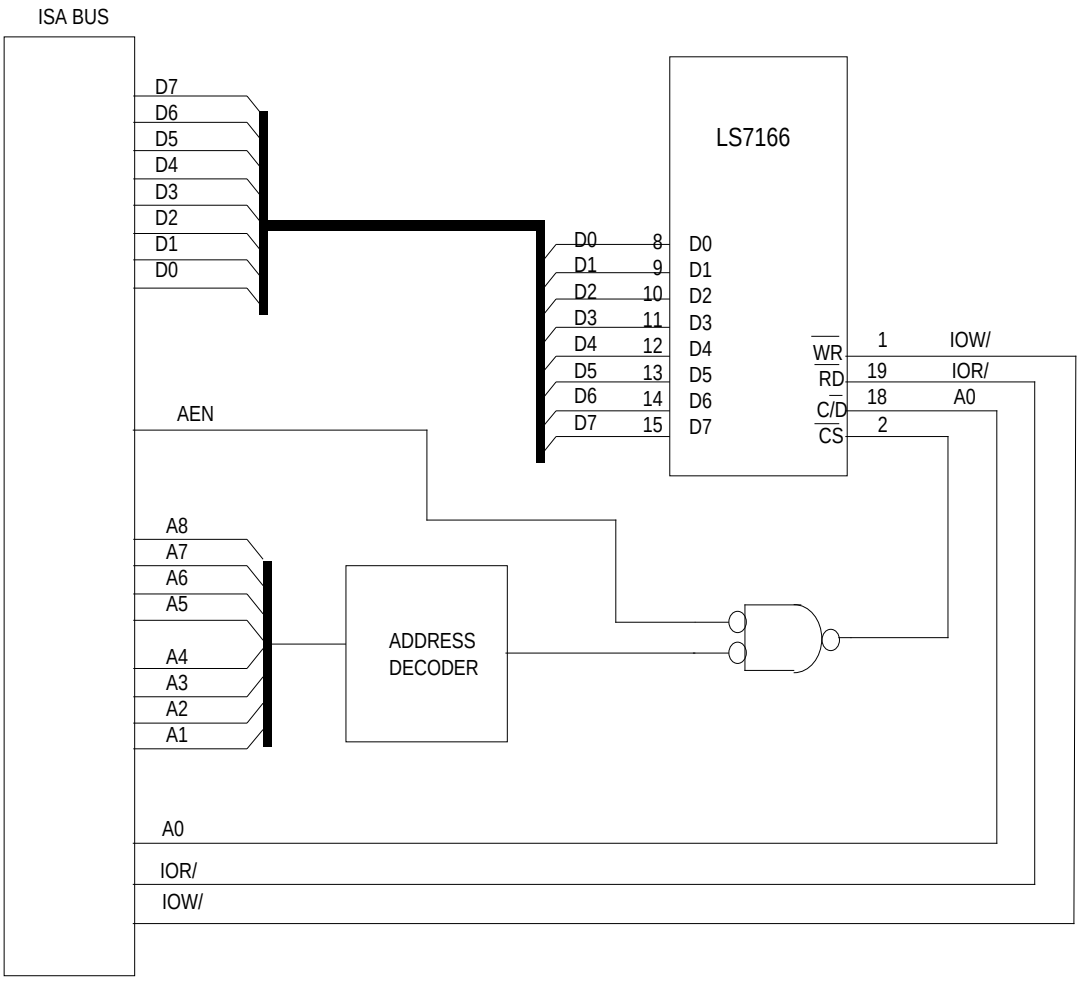
7166-092304-12

FIGURE 11. LS7166 TO 68HC11 INTERFACE



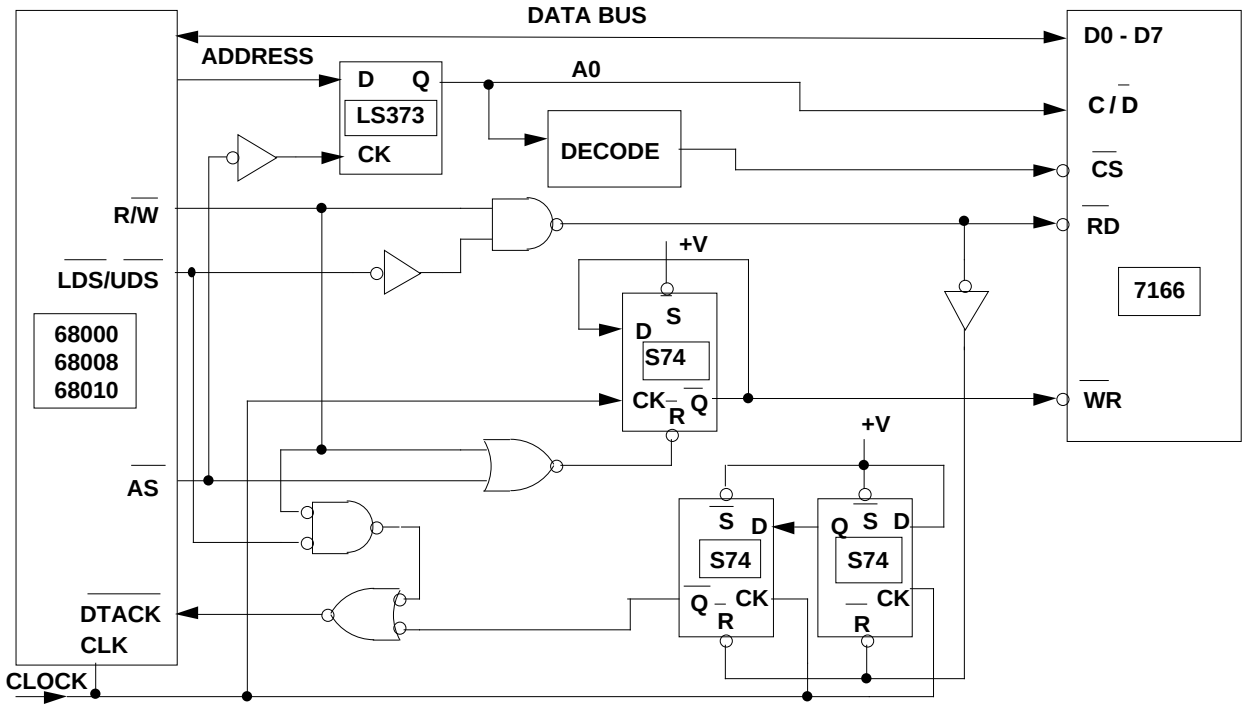
7166-110103-13

FIGURE 12. LS7166 INTERFACE EXAMPLE



7166-110503-14

FIGURE 13. 68000 INTERFACE TO LS7166



7166-062306-15