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

Autor: Pol Ocaña Hernández
Tutorizado por: Santiago Torres
Julio, 2023

Índice

Agradecimientos

He dedicado un apartado especial para expresar mi más sincero agradecimiento a todas aquellas personas que, ya sea de manera directa o indirecta, han brindado su valioso apoyo para el desarrollo y éxito de mi Trabajo de Final de Grado.

Me gustaría agradecer primeramente al tutor D. Santiago Torres Alvarez por depositar su confianza de esta tarea en mí y por haber servido de guía a lo largo del desarrollo del proyecto. Su disponibilidad para resolver mis dudas, tanto durante las tutorías como fuera de ellas, ha sido fundamental para alcanzar los resultados obtenidos.

A D. Manuel Fernández Vera por el apoyo y ayuda con los materiales que necesitábamos para cumplimentar el trabajo.

No puedo dejar de mencionar mi agradecimiento al Servicio de Electrónica de la Universidad de La Laguna, cuya invaluable ayuda en la elaboración de las placas PCB y su puesta a punto ha sido fundamental para el desarrollo exitoso de este proyecto.

A mi familia, quiero expresarles mi más sincero agradecimiento por su apoyo incondicional a lo largo de esta etapa académica. Su aliento y respaldo han sido un pilar fundamental en mi proceso de formación.

Y, por último, a mis compañeros de etapas anteriores quienes comenzaron dicho proyecto que, sin su establecimiento de bases tanto en electrónica como en informática, no hubiera sido posible la finalización de dicho proyecto.

A todas estas personas, mi más profundo agradecimiento por su contribución, su confianza y su apoyo incondicional. Su presencia ha sido fundamental en este camino académico y su ayuda ha dejado una huella imborrable en la realización de este Trabajo de Final de Grado.

1. Resumen

El objetivo de este proyecto es desarrollar una Placa de Circuito Impreso (PCB) que servirá como la electrónica de control para los brazos manipuladores de robots en el Laboratorio de Robótica de Ciencias Aplicadas y el Departamento de Ingeniería Informática y Sistemas de la Universidad de La Laguna.

El proyecto se basa en un diseño alternativo para el circuito de control electrónico del brazo robótico 'Scorbot-ER V+', que fue implementado y probado en un proyecto anterior. El diseño original fue desarrollado por Ana Estévez Pérez y Carlos Javier Siverio Suárez en sus respectivos Trabajos de Fin de Grado. Este nuevo diseño tiene como objetivo mejorar la velocidad de comunicación con el robot utilizando un método de comunicación diferente en lugar de la comunicación serie utilizada por el fabricante.

Posteriormente, se continuó el desarrollo de este diseño buscando una implementación en Placa de Circuito Impreso (PCB) por Oscar Jesús Díaz de la Fe y por Cristian Francisco Fariña Melián. Sin embargo, debido a la situación extraordinaria generada por el Covid-19, no se pudo realizar la implementación física de sus diseños.

El objetivo principal de este proyecto es continuar el desarrollo de los diseños aportados en los citados Trabajos de Fin de Grado, e integrar el diseño alternativo de la electrónica de control en una placa de circuito impreso, mejorando sus características y dándole un aspecto más compacto y profesional. En última instancia, el proyecto tiene como objetivo reemplazar la electrónica de control actual del 'Scorbot-ER V+' con este nuevo diseño. El proyecto también implica probar el diseño electrónico y el software para controlar los movimientos del manipulador y su posible mejora.

Además, se ha realizado un estudio exhaustivo de la adaptación de la electrónica para su uso con el brazo robótico 'Scorbot-ER IX'. El Laboratorio de Robótica del Departamento de Ingeniería Informática y Sistemas alberga actualmente los dos tipos de brazos manipuladores (ER-V+ y ER-IX), y sería ventajoso si el nuevo diseño electrónico pudiera ser compatible con ambos brazos robóticos. Por lo tanto, se desarrollará una interfaz para establecer una conexión entre la nueva electrónica de control y los dos modelos de robot, que comparten estructuras y modos de funcionamiento similares.

Una vez completado el diseño y la fabricación de las placas de circuito impreso, el proyecto entrará en una fase dedicada a examinar el software desarrollado en los trabajos originales. El objetivo es asegurarse de que cumpla con las funciones requeridas para el funcionamiento del brazo robótico. Esta fase implicará pruebas de diversas características en varios escenarios, prestando especial atención a los casos límite. Cualquier error o problema identificado se abordará y resolverá adecuadamente.

Por último, la parte final del proyecto tiene como objetivo mejorar, en la medida de lo posible, el repertorio de las instrucciones existentes para el controlador del manipulador. El programa se desarrolla utilizando la placa de desarrollo Arduino y utiliza como lenguaje de programación 'C'. Se elegirá el lenguaje Python para facilitar la comunicación entre Arduino y la computadora. El programa actual carece de ciertas funciones clave y contiene instrucciones incorrectamente programadas. En consecuencia, una parte importante del proyecto se dedicará a completar y mejorar estas instrucciones.

2. Abstract

The objective of this project is to develop a Printed Circuit Board (PCB) that will serve as the control electronics for the robotic manipulator arms in the Robotics Laboratory of the Department of Applied Sciences and the Department of Computer Engineering and Systems at the University of La Laguna.

The project is based on an alternative design for the electronic control circuitry of the 'Scorbot-ER V+' robotic arm, which was implemented and tested in a previous project. The original design was developed by Ana Estévez Pérez and Carlos Javier Siverio Suárez in their respective Bachelor's Theses. This new design aims to improve the communication speed with the robot by using a different communication method than the serial communication used by the manufacturer.

Subsequently, the development of this design was continued by Oscar Jesús Díaz de la Fe and Cristian Francisco Fariña Melián, with the intention of implementing it on a Printed Circuit Board (PCB). However, due to the extraordinary situation caused by Covid-19, the physical implementation of their designs could not be carried out.

The main objective of this project is to continue with the development of the designs provided in the aforementioned Final Degree Projects mentioned above and to integrate the alternative design of electronic control in a printed circuit board, improving its performance and giving it a more compact and compact appearance. professional. Ultimately, the project aims to replace the current control electronics of the 'Scorbot-ER V+' with this new design. The project also involves testing the electronic design and software to control the movements of the manipulator and exploring possible improvements.

In addition, an exhaustive study has been carried out on the adaptation of the electronics for use with the 'Scorbot-ER IX' robotic arm. The Robotics Laboratory of the Department of Engineering and Information Systems currently houses both types of manipulator arms (ER-V+ and ER-IX), and it would be advantageous if the new electronic design could be compatible with both robotic arms. Therefore, an interface will be developed to establish a connection between the new control electronics and the two robot models, which share similar structures and operating modes.

Once the design and manufacturing of the printed circuit boards is complete, the project will enter a phase dedicated to examining the software developed in the original works. The objective is to ensure that it complies with the functions required for the operation of the robotic arm. This phase will involve testing various features in different scenarios, paying special attention to edge cases. Any bugs or issues identified will be addressed and resolved appropriately.

Finally, the final part of the project aims to improve, as much as possible, the repertoire of existing instructions for the manipulator's controller. The program is developed using the Arduino development board and utilizes the 'C' programming language. The Python language will be chosen to facilitate communication between Arduino and the computer. The current program lacks certain key functions and contains incorrectly programmed instructions. As a result, a significant part of the project will be dedicated to completing and enhancing these instructions.

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 V+. Interfaz electrónica, lectura de codificadores digitales de posición y cálculos cinemáticos', 'Implementación de la electrónica para el control dinámico de un manipulador Scorbot-ER. Desarrollo de la comunicación'*, '*Integración en PCB de la electrónica para el Scorbot-ER V+ y el Scorbot IX, y diseño del gestor de alarma*s' y '*Integración en PCB de la electrónica para el Scorbot-ER V+ Diseño de algoritmos de control'* realizados por Ana Estévez Pérez, Carlos Javier Siverio Suarez, Oscar Jesús Díaz de la Fe y Cristian Francisco Fariña Melián respectivamente.

Cabe destacar que el proyecto que nos ocupa tiene una estrecha relación con el 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', ya que comparten algunos de sus objetivos. Debido a las condiciones extraordinarias que se produjo en su realización debido a la pandemia ocasionada por el Covid-19, no fue posible implementar de manera física sus diseños, quedando únicamente las bases teóricas del mismo. En este proyecto se pretende continuar el desarrollo de los diseños ideados con el objetivo de que puedan implementarse físicamente. [1] [2]

Por lo tanto, el objetivo de este proyecto es llevar a cabo la implementación del circuito de control detallado en los Trabajos de Fin de Grado en forma de Placa de Circuito Impreso (*Printed Circuit Board* o PCB) para el manipulador Scorbot-ER V+. Actualmente, esta electrónica de control se encuentra implementada en protoboard en el laboratorio de Robótica del Departamento de Informática y Sistemas de la Universidad de La Laguna.

Además, se busca mejorar esta electrónica de control en varios aspectos. Por un lado, se pretende desarrollar una interfaz que permita asimismo conectar esta electrónica de control al manipulador 'Scorbot-ER IX', que se encuentra en el laboratorio de Robótica del Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna. De esta manera, ambos tipos de manipuladores podrán funcionar con la misma electrónica de control, aprovechando su estructura similar. Esto es posible ya que en el trabajo original de Carlos Siverio y Ana Estévez, el diseño en protoboard de la electrónica fue dimensionado para dar cobertura a ambos robots, eligiendo los componentes sensores y actuadores necesarios para acometer el control de los dos modelos de Scorbot presentes en el laboratorio.

Por otro lado, se llevará a cabo un estudio exhaustivo del software controlador desarrollado por los estudiantes anteriores, con el objetivo de ampliarlo, calibrarlo si es necesario y asegurarse de que cumple con todos los requerimientos de movimiento del programa actual de control del manipulador. Esto permitirá su reemplazo en caso de avería, garantizando un funcionamiento óptimo del manipulador.

3.2 Descripción general del proyecto

Este proyecto de Trabajo de Fin de Grado se divide en tres partes diferenciadas pero relacionadas entre sí. Son las siguientes:

1- Desarrollo de la placa de circuito impreso (PCB) para el control de los brazos robóticos 'Scorbot-ER V+' y 'Scorbot-ER IX': El objetivo principal de esta parte del proyecto es diseñar y desarrollar una PCB que pueda desempeñar las funciones de electrónica de control para ambos brazos robóticos. La PCB se encargará de gestionar y controlar las operaciones del brazo robótico, reemplazando o mejorando el sistema original. Se debe garantizar un óptimo funcionamiento y la capacidad de reemplazo del sistema original.

2- Interfaz de comunicación entre el nuevo sistema electrónico y los brazos robóticos ‘Scorbot-ER V+’ y ‘Scorbot-ER IX’: En esta parte del proyecto, se desarrolla una interfaz que permita la comunicación entre el nuevo sistema electrónico, implementado en la PCB, y el brazo robótico. Para lograr esta comunicación, se utiliza Python junto con la librería “PySerial” que facilita la interacción con el Arduino. Se diseña un programa que permite enviar comandos y órdenes al brazo robótico utilizando una codificación específica. Cabe destacar que esta parte del proyecto ya ha sido desarrollado en el Trabajo de fin de Grado *’Implementación de la electrónica para el control dinámico de un manipulador Scorbot-ER. Desarrollo de la comunicación’.* En este caso, se pretende testear su funcionamiento y, si es posible, aumentar la funcionalidad del código diseñado.[3]

3- Adaptación del diseño electrónico y mejora del repertorio de instrucciones de movimiento para el brazo robótico ‘Scorbot-ER V+’: El objetivo en esta etapa es encontrar un método para conectar el nuevo sistema de control electrónico desarrollado en la PCB al brazo robótico Scorbot-ER IX. Dado que este brazo robótico tiene una estructura y un funcionamiento similares al Scorbot-ER V+, se busca adaptar el diseño electrónico para que pueda funcionar con ambos brazos robóticos. Además, se pretende mejorar el repertorio de instrucciones de movimiento del programa del controlador actualmente implementado en el manipulador, el cual se considera incompleto.

3.3. Objetivos

3.3.1. Objetivo general

En base a lo mencionado, el objetivo principal de este proyecto es implementar una electrónica de control alternativa para el manipulador "Scorbot-ER V+" en una placa de circuito impreso (PCB) más compacta y eficiente. Se busca mejorar el diseño actual de la electrónica con el fin de aumentar la eficacia del circuito electrónico y en un futuro, poder implementar este diseño de control en los manipuladores del laboratorio de Robótica del Departamento de Ingeniería Informática y de Sistemas.

3.3.2. Objetivos específicos

Los objetivos específicos del proyecto son los que han sido explicados con anterioridad. De forma resumida son los siguientes:

- Establecer una interfaz que permita la conexión del nuevo sistema de control electrónico con el brazo manipulador "Scorbot-ER IX", de manera que ambos tipos de manipuladores puedan funcionar con el mismo diseño de control.

- Realizar pruebas exhaustivas de funcionamiento y calibración para asegurar el pleno funcionamiento del controlador y su capacidad de reemplazar el sistema original.

- Diseñar una PCB más compacta que el sistema original para albergar la electrónica de control del manipulador "Scorbot-ER V+", optimizando el espacio y mejorando su rendimiento.

- Implementación del nuevo diseño: Una vez completado el diseño de la PCB, se procederá a la implementación de la electrónica de control en la placa. Esto implica la soldadura de los componentes electrónicos en la PCB y asegurar su correcta conexión.

En resumen, el proyecto se centra en implementar una electrónica de control alternativa y más eficiente para el manipulador "Scorbot-ER V+" y 'Scorbot-ER IX' en una PCB. Se buscará mejorar el diseño actual y evaluar las mejoras alcanzadas.

3.4 Planteamiento inicial

Para llevar a cabo este proyecto, fue fundamental realizar un estudio exhaustivo del circuito de electrónica de control diseñado previamente. Esto implicó la revisión de la bibliografía relevante y la consulta de los Trabajos de Fin de Grado relacionados que sentaron las bases para el proyecto.

Inicialmente, visitamos el laboratorio de robótica para examinar y comprender en detalle el circuito de control existente, el cual estaba implementado como un prototipo en "protoboards". Sin embargo, encontramos que, debido al paso del tiempo, el conexionado del circuito prototipo se había deteriorado, lo que resultó en su mal funcionamiento durante nuestras pruebas.

A pesar de esto, pudimos utilizar el circuito prototipo como referencia para el diseño de la placa PCB. Utilizamos el esquema y la estructura del circuito existente como punto de partida para la traducción a un diseño más robusto y eficiente en la placa PCB.

Es importante destacar que, a pesar de las dificultades encontradas con el circuito prototipo, la revisión de la bibliografía y los Trabajos de Fin de Grado relacionados nos proporcionaron el conocimiento necesario para comprender el funcionamiento del circuito y adaptarlo de manera adecuada en el proyecto.

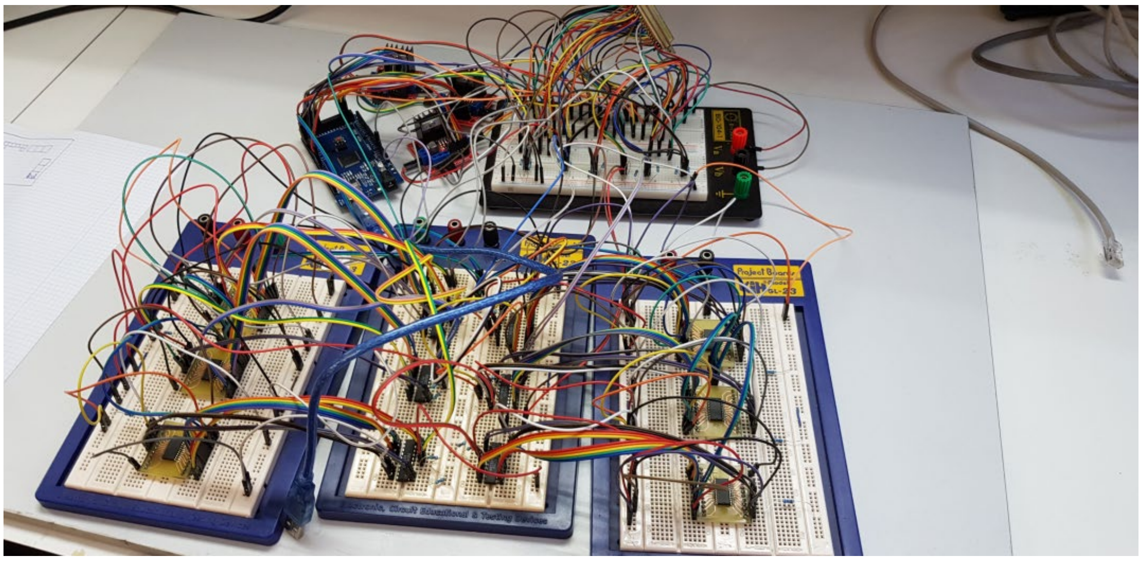


Figura 1. Foto del circuito extraída del TFG de Oscar Jesús Díaz de la Fe.[1]

Como se muestra en la figura 1, el circuito electrónico está implementado en cuatro "protoboards" junto con un Arduino, tres módulos L298N y otros circuitos integrados y elementos pasivos. Sin embargo, este prototipo presenta limitaciones en términos de eficiencia, tamaño y dificultad para trabajar con él, además de ser propenso a problemas de conexión.

Por este motivo, el objetivo del trabajo será mejorar estas limitaciones e 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 implementación en placa PCB.

En la figura 2, extraída del Trabajo de Fin de Grado de Ana Estévez Pérez: 'Implementación de la electrónica para el control dinámico de un manipulador Scorbot-ER V+. Interfaz electrónica, lectura de codificadores digitales de posición y cálculos cinemáticos' se puede observar mejor la estructura esquemática del circuito: [4]

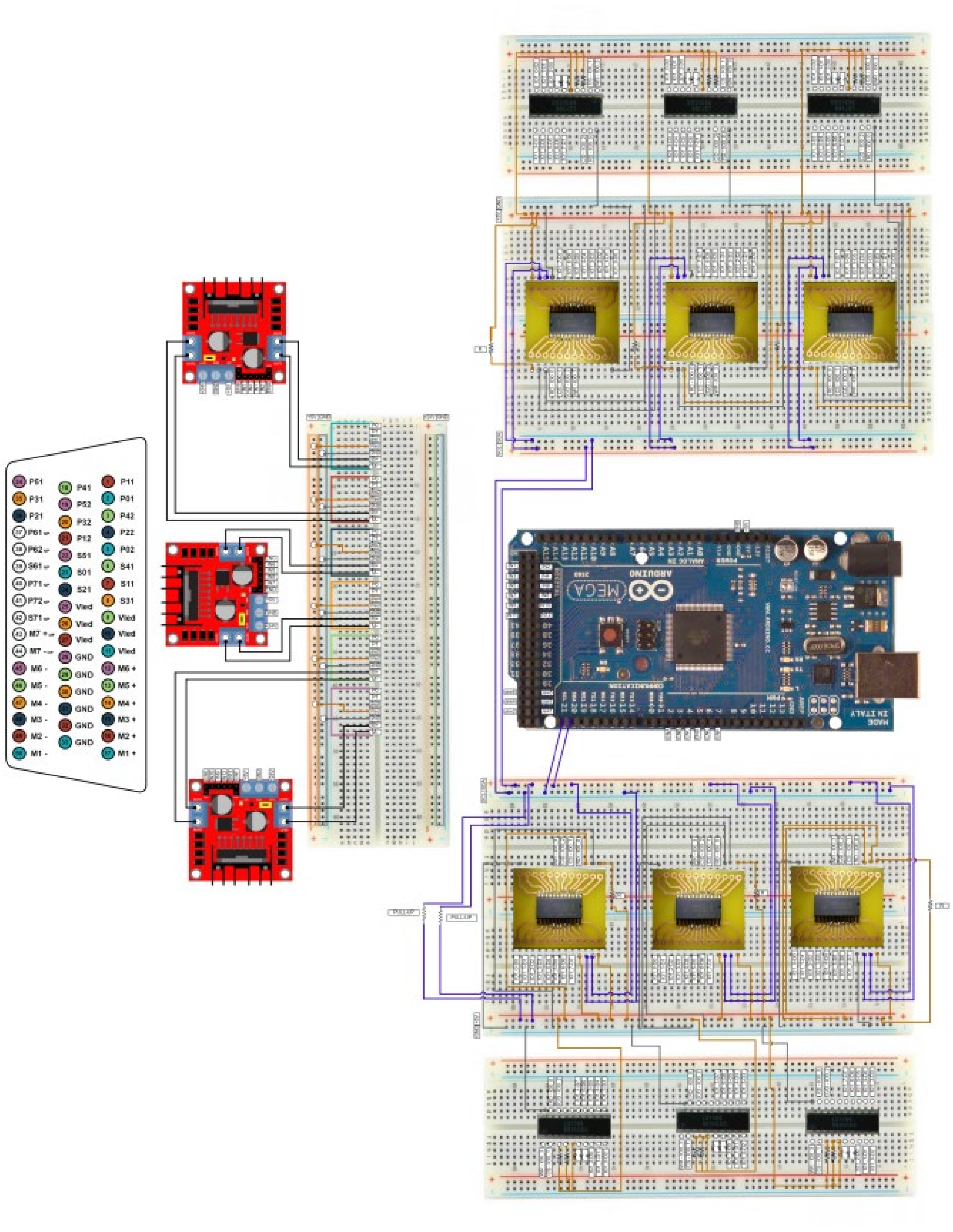


Figura 2. Esquema del prototipo de circuito de control.

Debido al estado en el que se encontraba el conexionado del prototipo ubicado en el laboratorio robótico, se decidió tomar como referencia el esquema anterior a la hora de realizar nuestra implantación del circuito en PCB. En la Figura 2, se pueden ver todas las conexiones entre los distintos componentes del circuito. No obstante, también fue necesario inspeccionar el circuito físico, ya que se encontraron algunas discrepancias entre el circuito real implementado en el prototipo y las instrucciones dadas en el esquema. Estas discrepancias serán descritas de forma detallada más adelante en este documento.

3.4.1 Arduino

Entrando en el desarrollo del circuito de control, se ha seleccionado primeramente un Arduino Mega 2560, un microcontrolador que cumple con los requerimientos solicitados en términos de memoria y número de pines para el manejo del manipulador. [5]

El Arduino se encargará de envíar señales de potencia PWM a los amplificadores que mueven los motores del manipulador, así como su sentido de giro. También mide las señales provenientes de los sensores de final de carrera ubicados en los diferentes motores del brazo robótico, que son utilizados para realizar el control de manera precisa. Por último, se emplean contadores para que el Arduino pueda obtener la posición de las articulaciones del brazo cuando sea necesario, evitando así sobrecargar el microcontrolador con demasiada información.

Para que el Arduino funcione correctamente, se conecta a un ordenador a través de un cable USB. Esto permite tanto cargar el código necesario para su funcionamiento enviando las instrucciones al manipulador para que las ejecute. El Arduino actúa como el cerebro del sistema de control, coordinando las acciones y garantizando un funcionamiento adecuado del brazo robótico.

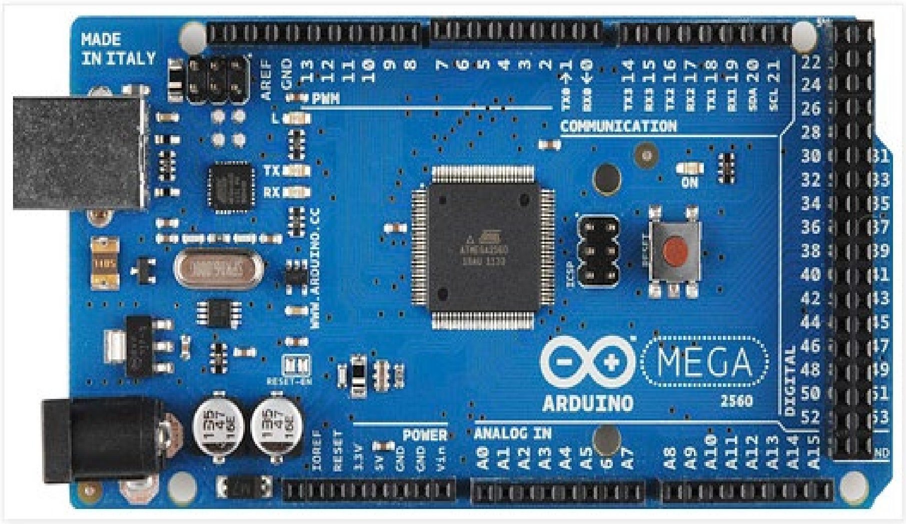
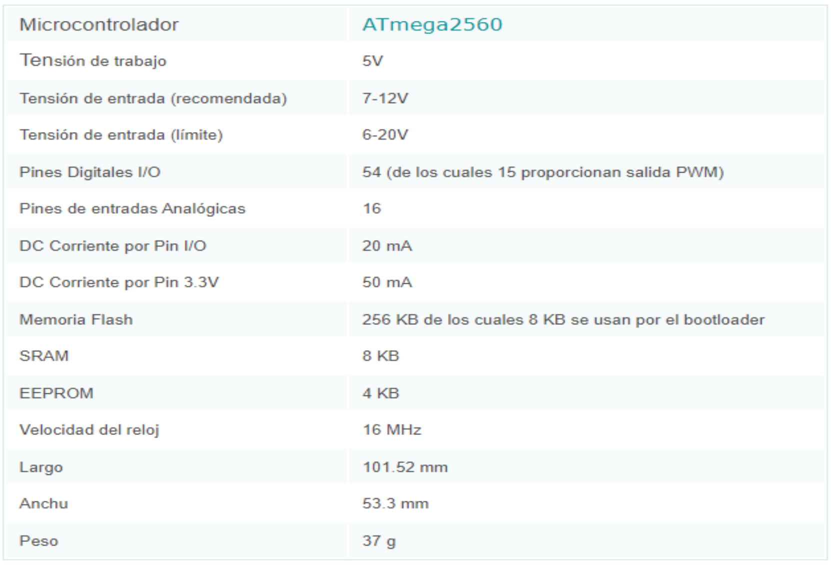


Figura 3: Arduino Mega 2560

Por otro lado, el circuito requiere de dos fuentes 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 analógico, necesaria para alimentar los motores del manipulador.

En el circuito prototipo se plantea el uso de dos fuentes de alimentación diferentes para abordar este problema. Sin embargo, en este 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, solo se requiere una fuente de alimentación para alimentar la parte del circuito de potencia, en contraste con el circuito prototipo, que utilizaba dos fuentes de alimentación separadas. Esta simplificación permite reducir la complejidad del circuito y facilitar su implementación tanto estética como funcional.



Microcontrolador	ATmega2560
Tensión de trabajo	5V
Tensión de entrada (recomendada)	7-12V
Tensión de entrada (límite)	6-20V
Pines Digitales I/O	54 (de los cuales 15 proporcionan salida PWM)
Pines de entradas Analógicas	16
DC Corriente por Pin I/O	20 mA
DC Corriente por Pin 3.3V	50 mA
Memoria Flash	256 KB de los cuales 8 KB se usan por el bootloader
SRAM	8 KB
EEPROM	4 KB
Velocidad del reloj	16 MHz
Largo	101.52 mm
Anchu	53.3 mm
Peso	37 g

Figura 4: Especificaciones del Arduino Mega 2560

3.4.2 Módulo L298N.

En segundo lugar, podemos ver en el esquema que se hace uso de tres módulos L298N. Estos módulos son *drivers* que permiten controlar la velocidad y la dirección de dos motores de corriente continua o un motor paso a paso de una forma muy sencilla a partir de una señal PWM (modulación por ancho de pulso) que genera el Arduino.[6]

De forma general, este módulo consta del circuito integrado L298N que integra un doble puente en H. El rango de tensiones en el que trabaja este módulo va desde 3V hasta 35V y una intensidad de hasta 2A, cumpliendo con los requerimientos del circuito. Adicionalmente incluye 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.

En el circuito, utilizaremos los pines IN1, IN2, IN3 e IN4 para controlar el sentido de giro. Asimismo, utilizaremos ENA y ENB para controlar la velocidad de giro de los motores. Los conectaremos a dos salidas PWM de Arduino de forma que le enviemos un valor entre 0 y 255 que controlará la velocidad de giro.

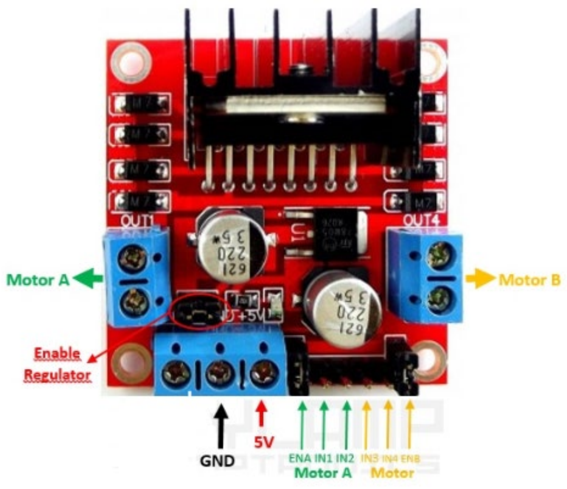


Figura 5: Esquema del módulo L298N

En el circuito a implementar, se utilizarán tres módulos L298N, ya que cada brazo robótico dispone de un total de 6 motores. Cada módulo será responsable de controlar tanto la dirección de giro como la velocidad de dos de los motores del manipulador.

Es importante destacar que los módulos L298N proporcionan una protección incorporada gracias a sus puentes en H. Estos puentes en H actúan como un aislamiento entre la parte de potencia del circuito y el circuito de control. Por lo tanto, no es necesario agregar aislamiento externo en el circuito.

Por último, tanto el Arduino como los módulos L298N son dispositivos físicos separados que no se pueden soldar directamente en la placa de circuito impreso (PCB). Son componentes electrónicos independientes y deberán estar conectados al diseño de la PCB mediante cables de conexión adecuados. Los cables permitirán establecer la conexión entre la placa PCB y estos dispositivos externos para garantizar su correcto funcionamiento en conjunto.

3.4.3 Circuitos Integrados.

En tercer lugar, el circuito consta de varios circuitos integrados que deberán quedar incluidos en el diseño final de nuestra placa PCB para estar soldados en ella.

Primeramente, se utilizarán seis circuitos integrados LS7166 que desempeñan la función de contadores. Estos contadores reciben y cuentan los impulsos generados por los encoders del brazo manipulador. En este proyecto, se utilizan 6 contadores correspondientes a los motores del brazo robótico que deben ser controlados. Los contadores deben operar en modo de conteo de cuadratura, ya que los encoders tienen una salida de dos señales en cuadratura que permiten determinar el sentido de giro y la posición de los motores. [7]

Es importante destacar que el diseño de la electrónica de control del brazo manipulador está preparado para trabajar tanto con los encoders del manipulador Scorbot-ER V+ como del Scorbot-ER IX. Para ello se ha tenido en cuenta que el manipulador Scorbot-ER V+ tiene una resolución de cuentas inferior en comparación con el Scorbot-ER IX.

La función principal de los contadores LS7166 es contar los pulsos emitidos por los encoders, de modo que el controlador pueda conocer la posición y el sentido de giro de los motores del manipulador. Estos contadores son esenciales para obtener información precisa sobre la posición de los motores y permitir un control adecuado del brazo robótico. En el diseño final de la placa PCB, estos circuitos integrados estarán soldados en la placa para formar parte integral del circuito de control.

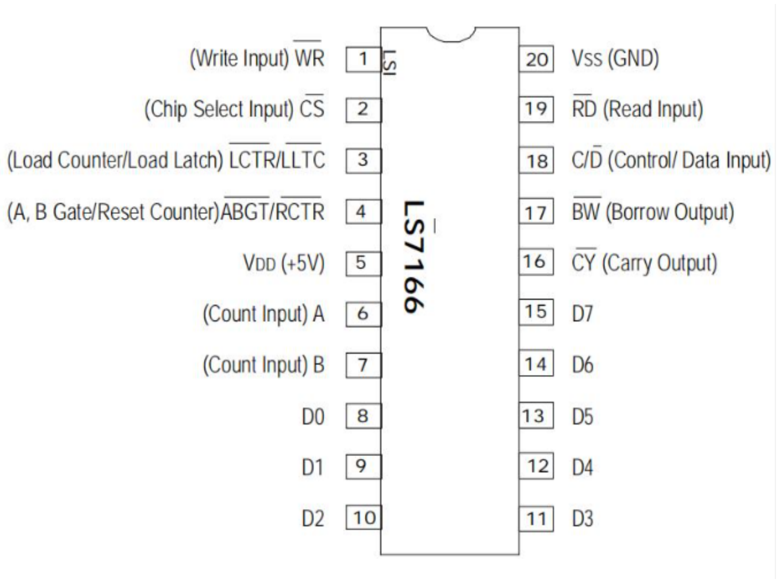


Figura 6: Esquema del Circuito Integrado LS7166

Por otro lado, en el diseño se emplearán el mismo número de circuitos integrados PCA9535. Estos circuitos son responsables de establecer el bus I2C necesario para el correcto funcionamiento de la electrónica de control. El bus I2C está compuesto por las líneas de comunicación SDA (System Data), que se utilizan para el envío de datos, y las líneas de comunicación SCL (System Clock), que funcionan como la señal de reloj generada por el Arduino.

El bus I2C establece una conexión entre el maestro (Arduino) y los esclavos (controladores I2C) en el sistema. La comunicación en el bus I2C es bidireccional y se realiza a través de la sincronización de pulsos de reloj enviados por la línea SCL. El Arduino tiene la capacidad de iniciar y detener la comunicación con los dispositivos esclavos según sea necesario.

Este bus I2C permite una comunicación eficiente y fiable entre los componentes del sistema, facilitando el intercambio de información y comandos necesarios para el control del brazo manipulador. El Arduino desempeña un papel fundamental como maestro en el bus I2C, coordinando la comunicación con los dispositivos esclavos para un funcionamiento sincronizado y eficaz del sistema de control electrónico.

El PCA9535 y el PCA9535C son dispositivos CMOS de 24 pines utilizados como GPIO (General Purpose Input/Output) para aplicaciones I2C-bus/SMBus. Constan de dos registros de configuración de 8 bits (selección de entrada o salida), entrada, salida e inversión de polaridad (operación activa ALTA o BAJA). El controlador del sistema puede habilitar las E/S como entradas o salidas escribiendo en los bits de configuración de E/S. Los datos de cada entrada o salida se guardan en el registro de Entrada o Salida correspondiente. La polaridad del registro de lectura se puede invertir con el registro de inversión de polaridad. Todos los registros pueden ser leídos por el controlador del sistema. [8]

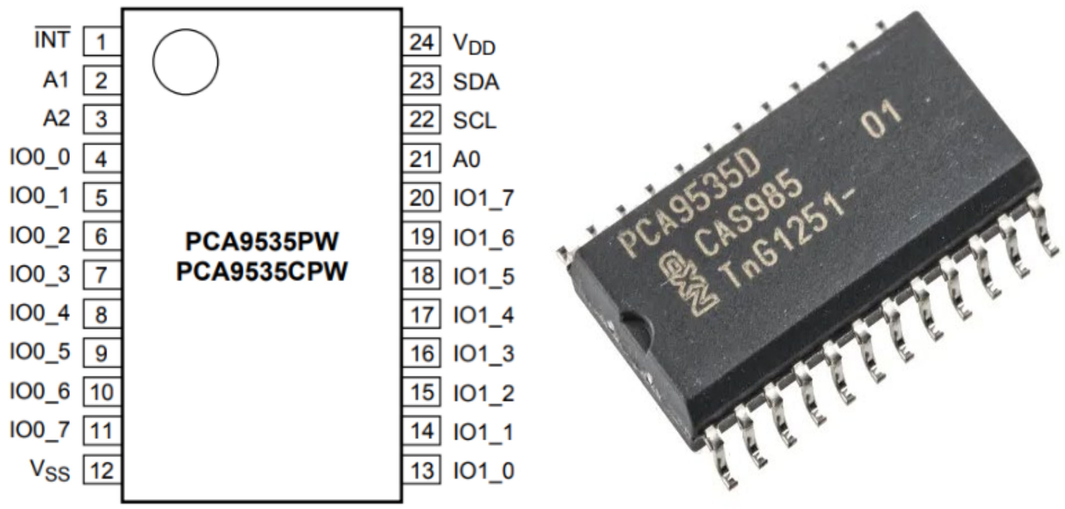


Figura 7: Esquema del Circuito Integrado PCA9535

3.4.3. Conectores

Por otro lado, debemos considerar los conectores D-50 del Scorbot-ER V+ y conector D-37 junto al conector tubular del Scorbot-ER IX como elementos fundamentales en el circuito de control. Estos conectores representan las salidas finales del circuito y establecen la interfaz entre el diseño de la electrónica de control y el brazo manipulador. [9]

El conector D-50 del Scorbot-ER V+ consta de 50 pines y permite el paso de todas las señales del manipulador. Entre estas señales se incluyen las dos señales de los sensores de cada uno de los encoders del manipulador (en este caso solo se utilizan 6 motores, por lo que, algunos pines pueden no tener una función específica), las señales de fin de carrera de cada motor del manipulador, las alimentaciones de 5 voltios para cada uno de los encoders del manipulador, las conexiones a tierra de cada uno de los encoders del manipulador, y los polos positivos y negativos de las alimentaciones de 24 voltios para cada uno de los motores del manipulador.

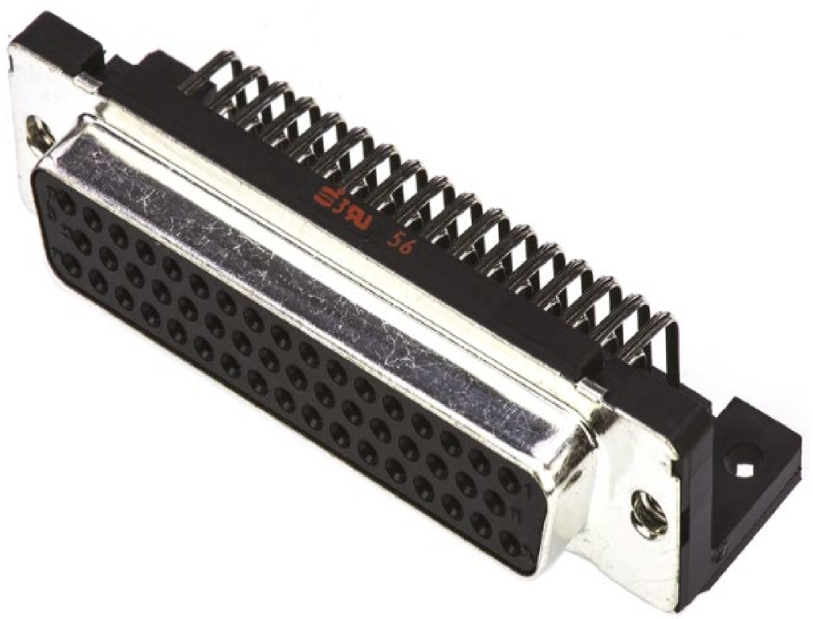


Figura 8: Imagen conector D-50 del Scorbot-ER V+

En el 'Scorbot-ER IX', aunque requiere del mismo número de entradas y salidas, se han dividido en dos conectores diferentes, uno encargado del sistema de control y otro encargado de las señales de potencia. El conector D-37, encargado del control del brazo robótico, tiene en total 37 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, las señales de fin de carrera de cada uno de los motores del manipulador, la alimentación de 5 voltios de los encoders del manipulador y las tierras de cada uno de los encoders del manipulador.

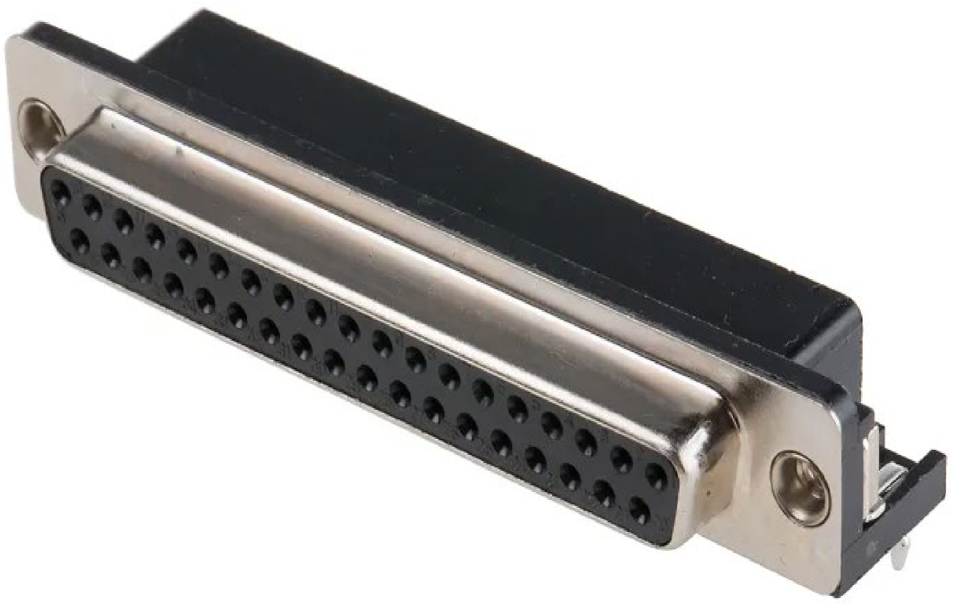


Figura 9: Imagen conector D-37 del Scorbot-ER IX.

El conector tubular, encargado del circuito de potencia del brazo robótico, tiene en total 19 pines y se encarga de alimentar los polos positivos y negativos de las alimentaciones de 24 voltios de cada uno de los motores del manipulador.

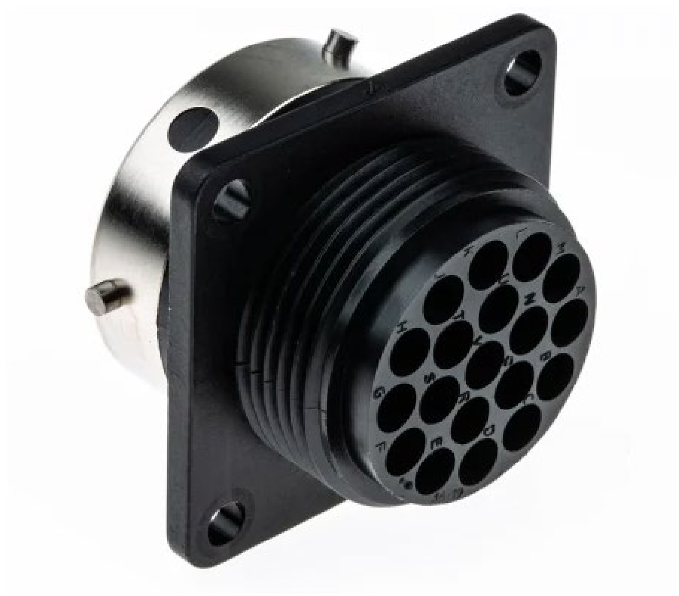


Figura 10: Imagen conector tubular del Scorbot-ER IX.

Estos conectores son esenciales para establecer la comunicación y la alimentación adecuadas entre el diseño de la electrónica de control y el brazo manipulador. Proporcionan los enlaces necesarios para la transmisión de señales de control, información de posición y alimentación eléctrica, permitiendo así el control preciso y eficiente del manipulador en el sistema de control electrónico.

Además de los elementos previamente descritos, el circuito de control propuesto utiliza varias resistencias pull-up. Estas resistencias son necesarias para garantizar que los circuitos integrados digitales funcionen correctamente según la lógica transistor a transistor (TTL), en la cual los valores de voltaje deben ser adecuados para representar el 0 lógico (0 voltios) y el 1 lógico (5 voltios).

El circuito incluye cuatro tipos de resistencias pull-up. Se utilizan resistencias de 2,2 KΩ en las líneas SDA y SCL, 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, finalizando con resistencias de 100 KΩ en las entradas de los contadores en las que se introducen las salidas de los encoders del manipulador.

Estas resistencias son componentes importantes para asegurar el funcionamiento adecuado del circuito de control, y deben ser consideradas y soldadas en la placa de circuito impreso (PCB) durante su elaboración.

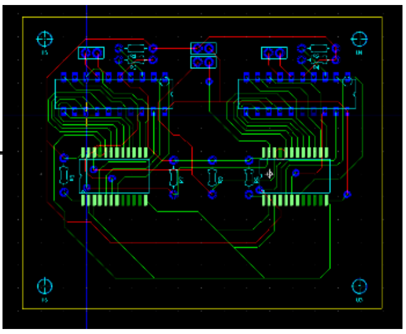
Para obtener información más detallada sobre la interfaz electrónica del circuito y sus componentes, se puede consultar el Trabajo de Fin de Grado de Ana Estévez Pérez ‘Implementación de la electrónica para el control dinámico de un manipulador Scorbot-ER V+. Interfaz electrónica, lectura de codificadores digitales de posición y cálculos cinemáticos’ [1]. En este documento se proporciona una explicación más detallada de la interfaz electrónica del circuito de control alternativo para el ’Scorbot-ER V+’. En el presente documento solo se ha proporcionado a modo de resumen una explicación del circuito de control que se va a implementar en PCB para dar un contexto al trabajo.

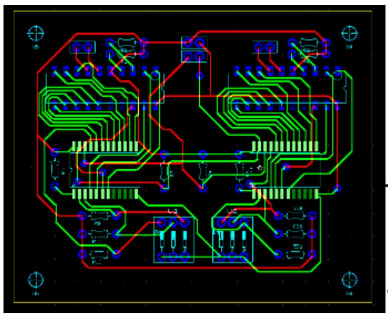
4. Cronograma

A continuación, se presenta una descripción general en forma de cronograma de las diferentes tareas y actividades llevadas a cabo durante el desarrollo del proyecto, con el objetivo de mostrar una evolución visual de las etapas de desarrollo del proyecto

Enero de 2023
Primeros diseños PCB I2C
Estudio de prototipo y comienzo de diseño de placa PCB I2C

Estos diseños se basan en el proyecto "Integración en PCB de la electrónica para el Scorbot-ER V+ y el Scorbot IX, y diseño del gestor de alarmas"



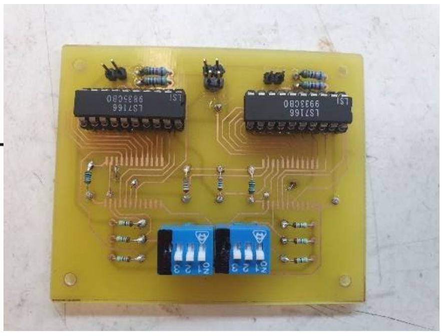


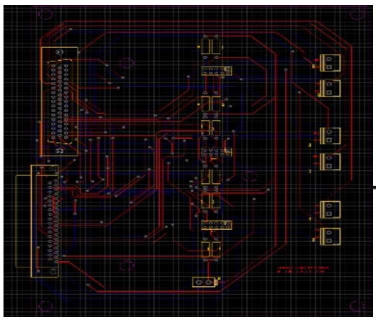
Febrero de 2023
Finalización de diseño PCB I2C

Se termina el diseño definitivo de las primeras placas PCB.

Marzo de 2023
Implementación del Diseño PCB I2C

Se sueldan los componentes a la placa fabricada por el Servicio de Electrónica y se testean los resultados.





Abril de 2023
Finalización de diseño PCB *Main*

Se termina el diseño definitivo placas PCB *Main*.

Mayo de 2023
Implementación del Diseño PCB *Main*

Se sueldan los componentes a la placa fabricada por el Servicio de Electrónica y se testean los resultados.

Junio de 2023
REDACCIÓN DEL TFG Y MODIFICACIONES

Se supervisa el funcionamiento del código y diseño final de las placas PCB.
Se empieza a desarrollar la memoria escrita del TFG

Julio de 2023
FINALIZACIÓN DEL TFG Y ÚLTIMOS DETALLES

Se terminan los detalles de la redacción del TFG. Se terminan las últimas pruebas del prototipo terminado

5. Implementación en PCB de la electrónica de control

5.1. Consideraciones iniciales

En esta etapa del Trabajo de Fin de Grado, nos centramos en el diseño de la placa de circuito impreso (PCB) para la implementación de la electrónica de control alternativa de los manipuladores 'Scorbot-ER V+' y 'Scorbot-ER IX'. Para ello, se tomaron varias decisiones importantes.

En primer lugar, se seleccionó el software "NI Multisim 14.0" como la herramienta para diseñar las placas PCB. Este software ofrecía facilidad de uso y se ajustaba a nuestras preferencias, además de ser utilizado en el Servicio de Electrónica de la Universidad de La Laguna, lo que nos permitiría buscar asesoramiento si surgían dudas sobre su funcionamiento.

En segundo lugar, se analizó todas las variaciones posibles del diseño de la placa y se optó por hacer un diseño distribuido en cuatro placas, tal y como se había propuesto en el 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' realizado por Oscar Jesús Díaz de la Fe y Cristian Francisco Fariña Melián. Este enfoque consistía en dividir el diseño de la electrónica de control en cuatro placas separadas, lo que facilitaría la reparación en caso de avería y permitiría realizar pruebas individuales de las diferentes partes del circuito antes del montaje final. Por razones como esta, el diseño de una electrónica modular es mucho más eficiente y adecuado al circuito.[1][2]

De esta forma, se propone un diseño simétrico y similar en el diseño de las tres primeras placas para facilitar su réplica en el futuro. 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 idénticas y posteriormente una placa principal (o placa *Main)* que serviría como interfaz con los elementos externos y los conectores de los brazos robóticos.

Las tres placas principales tendrían un diseño idéntico y simétrico, lo que nos permitiría diseñar un tercio del circuito y replicarlo para obtener el diseño completo. Cada tercio del diseño incluiría dos contadores LS7166, dos circuitos integrados PCA9535 para formar el bus I2C, y varias resistencias pull-up necesarias para el funcionamiento de la electrónica. Además, las placas gestionarían las alimentaciones, las conexiones a tierra y la distribución de las señales relevantes de entrada y salida.

Estas tres placas principales serían independientes entre sí y no necesitarían estar interconectadas directamente, ya que solo requerirían la conexión con la placa principal. Además, al ser idénticas e independientes, podrían intercambiarse según las necesidades del diseño. Por ejemplo, la placa que controla la muñeca del manipulador podría intercambiarse con la que controla la base y el hombro, sin causar conflictos. Esto es otro beneficio del diseño modular de la electrónica de control.

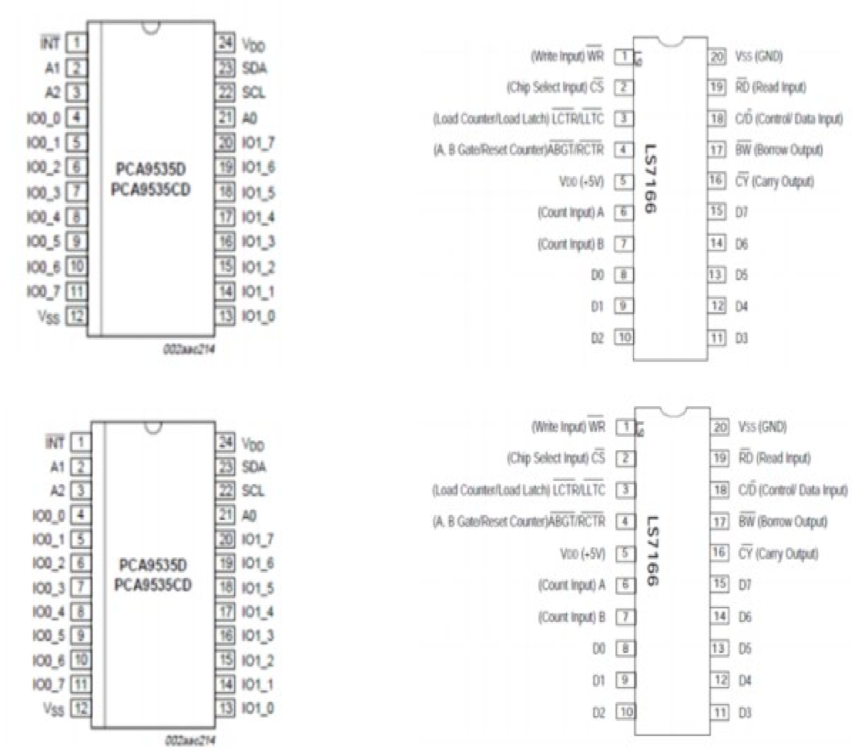


Figura 11: Esquema planificación componentes PCB I2C

En la Figura 11, se muestra un esquema de una de las placas que se replicarían tres veces. En este esquema se incluyen los pares de circuitos integrados mencionados anteriormente, faltando las resistencias pull-up y los conectores correspondientes.

Cada una de estas placas estarían conectada a uno de los tres módulos L298N que, con sus puentes en H, se encargarían de controlar dos de los seis motores del brazo manipulador. Además, cada placa debe estar conectada al Arduino a través del conexionado necesario para transmitir las señales que proveen el correcto funcionamiento del circuito.

Por otro lado, para desarrollar la placa *Main* que actúa como enlace entre los brazos robóticos y los diferentes dispositivos de control, se estudió el esquema del prototipo mencionado con anterioridad en el apartado 3.4. En dicho esquema, se visualizan las entradas y salidas del conector D-50, siendo estas salidas, dirigidas al resto del hardware. Por ello, para poder redirigir dicho conexionado, es necesario un diseño con una amplia cantidad de conectores que sirvan como transmisor desde el conector D-50, a su respectivo dispositivo. Con objetivo de conseguir un diseño con mayor rendimiento y facilidad a la hora de su conexión, dichos conectores se dividirán en grupos de 2 coincidiendo de manera descendente con las placas modulares. En la figura 12 se mostrará un esquema inicial de la disposición de estos conectores.

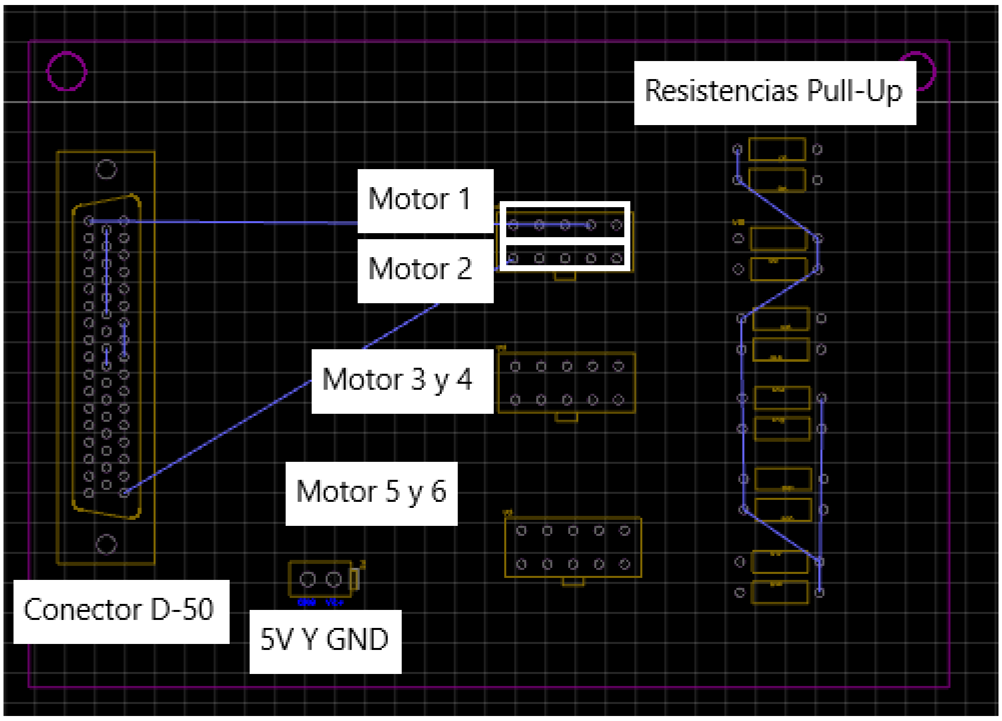


Figura 12: Esquema planificación inicial PCB *Main*.

Sin embargo, este diseño presenta dos problemas principales. En primer lugar, es necesario ajustar el diseño para poder abarcar las conexiones del brazo robótico 'Scorbot-ER IX', ya que el espacio disponible en la placa puede ser insuficiente. En segundo lugar, el esquema del circuito no incluye las resistencias pull-up necesarias para el funcionamiento de la electrónica, las cuales fueron implementadas posteriormente en el circuito real montado en el laboratorio. Para solucionar estos inconvenientes, se ha ido desarrollando progresivamente el diseño de la placa, asegurándose de cumplir con todos los requisitos necesarios.

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]

Una placa de circuito impreso (PCB) es una lámina de un material aislante, como la fibra de vidrio, sobre la que se imprimen o graban circuitos o pistas metálicas para proporcionar conductividad eléctrica. Además, es una placa base para soportar y conectar físicamente los componentes de montaje superficial y de enchufe de la mayoría de los productos electrónicos. Además de proporcionar la base física para estos componentes, las placas de circuito impreso permiten la conductividad eléctrica mediante vías que permiten su comunicación. Las placas de circuito impreso se construyen con materiales de núcleo dieléctrico que tienen escasas propiedades de conducción eléctrica para mantener las transferencias de los circuitos lo más limpias posible, y luego se espacian con más capas de metal y materiales dieléctricos según sea necesario. El material dieléctrico estándar utilizado para las placas de circuitos es un compuesto resistente al fuego de tela de fibra de vidrio trenzada y una resina epoxi conocida como FR-4, y las pistas y planos metálicos para los circuitos suelen ser de cobre.

En este caso, como ya se ha dicho, nuestras placas PCB tienen la función de establecer las conexiones 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 conexiones 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, se ha planteado la posibilidad de fabricar nuestras placas de forma manual a través de los recursos de los que dispone el Servicio de Electrónica de nuestra universidad.

En el diseño se decidió utilizar una placa doble (con una capa de cobre por cada lado), ya que nos permitía simplificar de forma significativa el diseño y resultaba ser mucho más conveniente en algunas situaciones que se estudiarán posteriormente.

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.

a. En primer lugar, con cara superior o 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*).

b. En segundo lugar, la cara inferior o de soldadura sería la cara en la que se sueldan la mayor parte de 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*).

c. 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).

d. En cuarto lugar, el término 'vía' (u orificio pasante) se refiere a una conexión eléctrica entre diferentes capas de una placa de circuito impreso. La vía es básicamente un pequeño orificio realizado a través de los laminados del PCB que cruza dos o más capas adyacentes. El orificio se cubre internamente con cobre (mediante proceso galvánico, remachado o insertando un pequeño tubo de material conductor), formando una conexión eléctrica en el material aislante que separa las capas del PCB.

e. En quinto 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.

f. En sexto lugar, con el término huella o 'footprint' nos referimos al conjunto de pads que conforman la silueta de un componente físico. Los softwares de diseño de placas PCB suelen incluir ciertas librerías donde se guardan diferentes huellas de componentes para que el usuario pueda elegir un componente específico. Esto facilita al usuario el poder realizar un diseño complejo sin necesidad de crear todos los componentes a usar, puesto que ha quedado registrado en la huella todas sus dimensiones.

g. En último lugar, las pistas de la placa son los conductores que permiten conectar unos elementos con otros. Estos elementos serían como los cables del 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. [10]

5.2.1. 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, por ello, nos centraremos en los que utilizaremos en ese proyecto:

La primera familia de encapsulados es la llamada *Through-Hole* (THT) o Agujeros Pasantes en castellano, son encapsulados de tecnología que se aprovechan de puentes electrónicos entre una cara y otra de la PCB por medio de un tubo conductor que puede ser de diferentes materiales como son zinc, cobre o plata, estos materiales permiten que los componentes se puedan soldar correctamente y evitar que se oxiden. 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 solo por la capa top de la placa.

Alguna de las características que tienen los componentes THT es su fragilidad y sensibilidad al calor, por lo cual, si estos componentes son afectados por altas temperaturas durante un periodo de tiempo, pueden llegar a fallar.

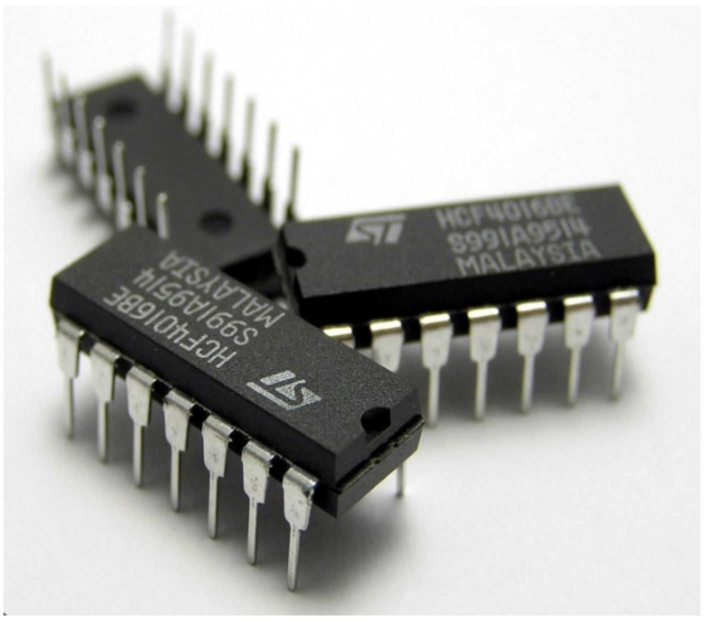


Figura 13: Imagen de encapsulados *Through-Hole* (THT).

La segunda gran familia de encapsulados son los llamados *Surface-Mount Device* (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 eficientes.

Esta tecnología se ha vuelto muy útil y está reemplazando la tecnología de agujeros pasantes ya que es ideal para la producción masiva de PCB. Gracias al bajo consumo de energía que tienen estos componentes aguanta mayores temperaturas y son muy útiles para aplicaciones donde el tamaño y espacio es muy reducido. [8]

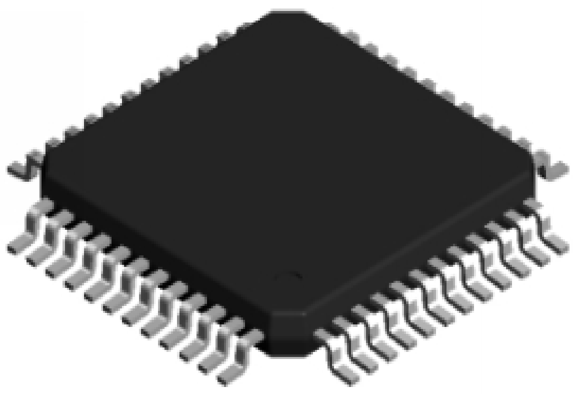


Figura 14: Imagen de encapsulados *Surface-Mount Device* (SMD).

5.2.2. Producción manual de circuitos impresos

Como se ha dicho anteriormente, la producción de circuitos impresos suele estar automatizada. No obstante, en este caso, para la elaboración de las tres primeras placas 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.

Debido a las condiciones complejas de diseño de la placa *Main*, será necesario una fabricación automatizada diferente mediante maquinaria profesional ubicada en el Servicio de Electrónica de la universidad. Esto es debido a la gran cantidad de 'Vías' utilizadas, siendo necesario la metalización de las mismas para una mejora significativa en las garantías de funcionalidad y profesionalidad. Dicho diseño será explicado con una mayor profundidad en el apartado 5.5 Evolución del diseño de la placa PCB *Main*.

Inicialmente, tras completar el diseño de la electrónica utilizando un software especializado como "Multisim 14.0", se procede a generar una serie de documentos finales mediante el programa de edición de circuitos. Estos documentos contienen la información necesaria para la posterior impresión de los fotolitos correspondientes. Los fotolitos son láminas transparentes en las que se dibujan las pistas del circuito a fabricar. Para este propósito, se utiliza una impresora especial ubicada en el laboratorio de electrónica de la universidad. [11]

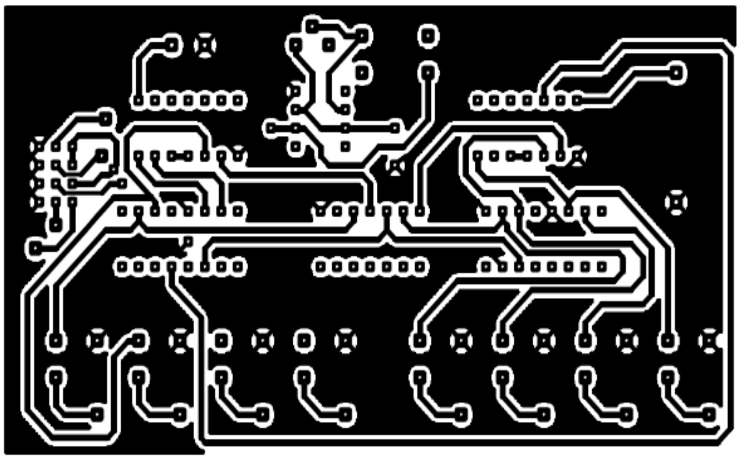


Figura 15: Imagen de ejemplo de Fotolito.

Una vez impresos los fotolitos, se continúa con el proceso manual de fabricación de las placas PCB. Este proceso implica la preparación de una placa base de material aislante, como fibra de vidrio o resina epoxi, sobre la cual se colocará el fotolito impreso. La placa y el fotolito se exponen a una fuente de luz ultravioleta con una insoladora durante un tiempo determinado, lo que permite la transferencia del diseño del fotolito a la placa. 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, es decir, la luz de la insoladora solo debilitaría estas zonas transparentes dejando intactas las pistas del circuito.[12]

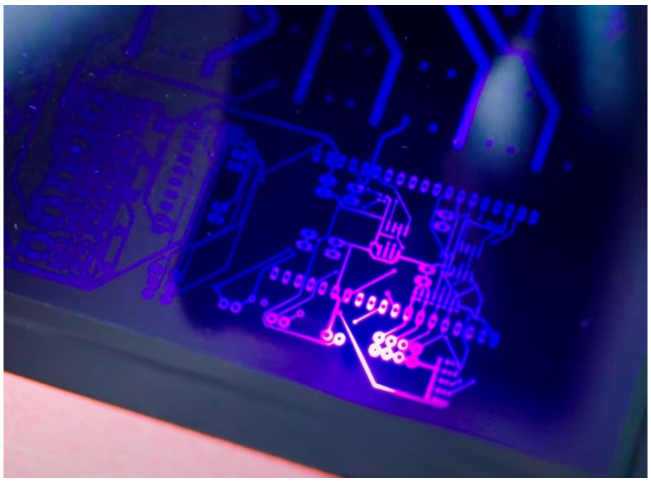


Figura 16: Proceso de exposición ratos ultravioleta.

A continuación, se realiza un proceso químico conocido como revelado, en el cual se eliminan las áreas no expuestas a la luz ultravioleta en el proceso de insolado, dejando únicamente las pistas del circuito impresas en la placa. Posteriormente, se lleva a cabo el proceso de corrosión, en el cual se sumerge la placa en una solución química que disuelve el cobre no protegido por el diseño impreso, dejando únicamente las pistas del circuito.

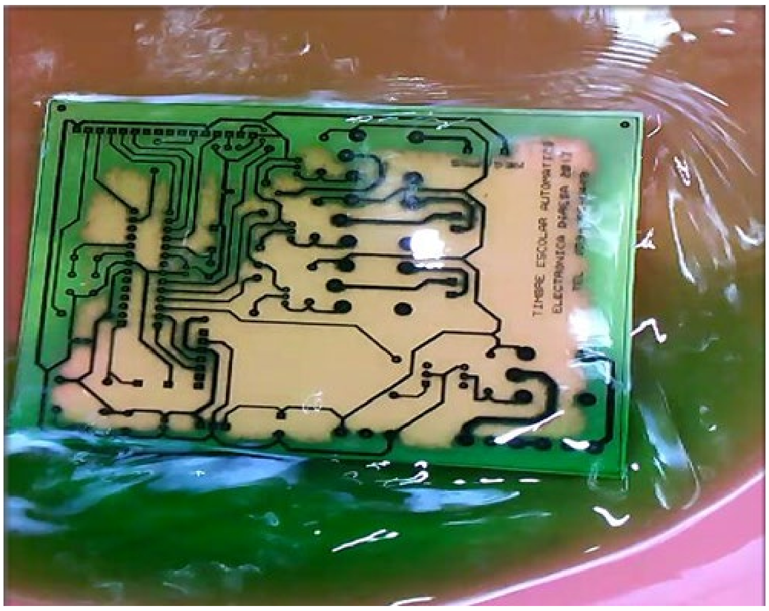


Figura 17: Proceso de revelado.

Una vez completados estos pasos, se procede a limpiar y secar la placa PCB resultante. Luego, se realiza una inspección minuciosa para verificar la calidad y la integridad del circuito impreso. En este proceso se comprobará la continuidad de las pistas para en caso de ser necesario, se pueden realizar ajustes o reparaciones adicionales.

Después de completar el proceso de fabricación de las pistas del circuito, se procede a la etapa de taladrado de la placa y la soldadura de los componentes. Durante el taladrado, se realizan los agujeros necesarios para los pines de los componentes de tecnología de agujeros pasantes y para las vías del circuito. En el caso de los circuitos integrados de montaje superficial, no se requiere el taladrado de los pines.

En este diseño específico, es necesario taladrar las vías del circuito, ya que se utiliza tanto la cara superior como la inferior de la placa, y se requiere que ambas caras estén conectadas a través de estas vías. Los tamaños de las brocas utilizadas según las instrucciones del Servicio de Electrónica han sido de 1,1 mm para el taladro de los pads de los diferentes componentes y de 0,6 mm para el taladro de las vías. Por otra parte, se ha dejado en ambos casos 1 mm adicional para la conectividad con la pista correspondiente.

Por último, con respecto a la soldadura, una vez colocados los componentes en sus respectivos sitios, se deben ir soldando individualmente 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.

Durante el proceso de soldadura, se deben seguir las prácticas adecuadas para garantizar una conexión sólida y confiable. Se requiere habilidad y precisión para evitar cortocircuitos, daños en los componentes y asegurar una soldadura de calidad.

Una vez finalizada la soldadura, se realiza una inspección visual para verificar que todos los componentes estén correctamente soldados y que no existan problemas de conexión. En caso necesario, se realizan correcciones y retoques adicionales.

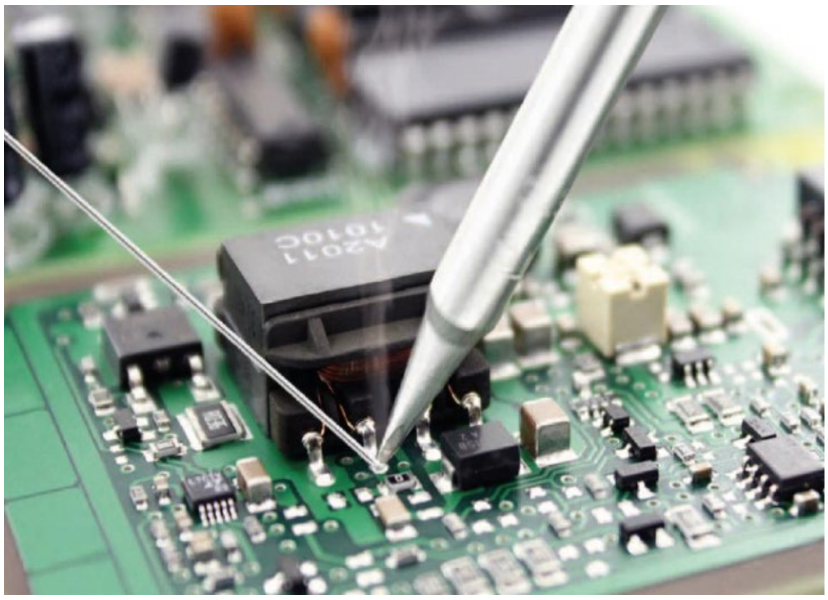


Figura 18: Proceso de soldadura.

5.3 Evolución del diseño de las placas I2C PCB

A continuación, se expondrán las diferentes etapas por las que ha ido evolucionando el diseño de nuestras placas de circuito impreso. Primeramente, realizaremos el diseño de las tres primeras placas modulares denominadas "Placas I2C PCB", que compartirán un diseño idéntico, reducido y eficiente. Estos diseños, han sido elaborados con el programa 'Multisim 14.0', 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.

Como se ha mencionado anteriormente, el proyecto que nos ocupa actualmente, parte de diferentes Trabajos de Fin de Grado finalizados en promociones anteriores. Por ello, a modo de realizar un trabajo eficiente, se ha tomado como referencia el diseño final propuesto en *'Integración en PCB de la electrónica para el Scorbot-ER V+ y el Scorbot IX, y diseño del gestor de alarmas'*. En la figura 19 se muestra el 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.

Tras estudiar detenidamente el diseño propuesto y contrastarlo con el prototipo diseñado mediante placas 'protoboard', se han identificado varios problemas de eficiencia y limitaciones de espacio en dicho diseño.

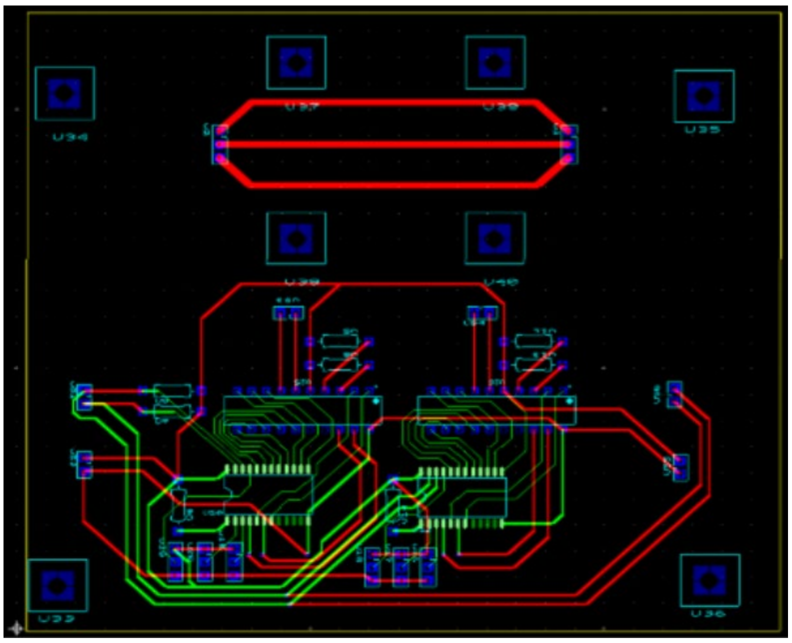


Figura 19: Diseño Base de PCB

Analizando el diseño propuesto, detectamos en la parte superior 2 pares de 3 conectores conectados entre sí que no guardan relación con el diseño inferior. Dichos conectores debido al grosor empleado los podemos relacionar con las entradas de corriente de los motores, Sin embargo, al no tener relación alguna con el circuito de la parte inferior, el diseño teórico presenta deficiencias en términos de rendimiento y consumo de energía, lo cual comprometía su viabilidad práctica.

Además, encontramos limitaciones de espacio en el diseño propuesto. El tamaño y la disposición de los componentes requeridos resultaron ser incompatibles con las restricciones físicas y las dimensiones del sistema en el que se iba a implementar. Esta falta de adaptabilidad dificultaba su integración en un entorno real.

Dada esta evaluación, consideramos necesario llevar a cabo una revisión exhaustiva del diseño con el objetivo de mejorar su eficiencia y abordar las limitaciones de espacio identificadas. Es fundamental realizar ajustes y optimizaciones que permitan una implementación práctica y exitosa.

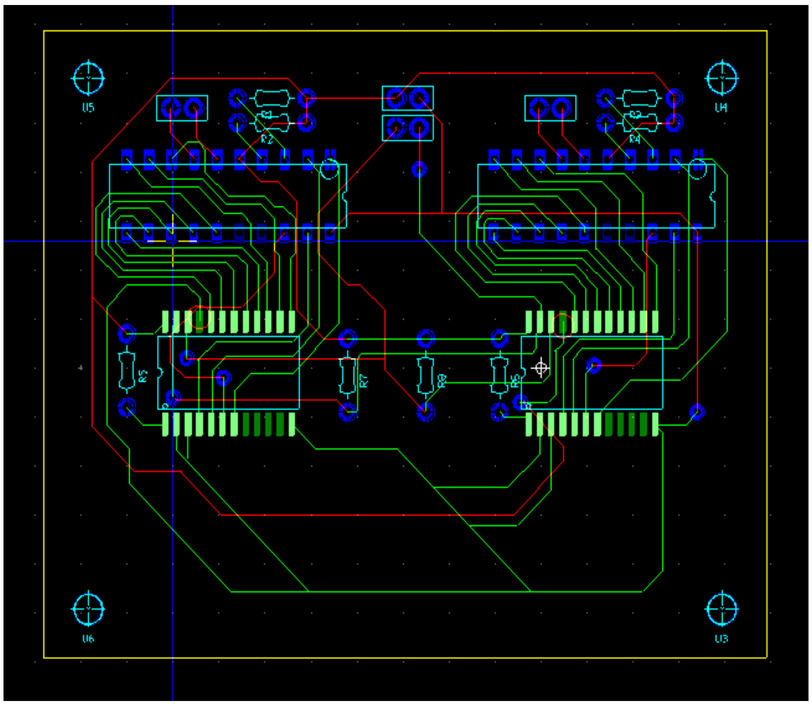


Figura 20: Primera propuesta de Diseño de la placa PCB.

En la figura 20 se presenta el primer diseño generado después de realizar una revisión exhaustiva del modelo propuesto anteriormente. Este nuevo diseño se basa en las partes del diseño anterior que se consideraron adecuadas, y se han eliminado las partes innecesarias, resultando en un prototipo mucho más compacto y eficiente.

Las modificaciones realizadas se centran en optimizar el rendimiento y la eficiencia del diseño, así como en abordar las limitaciones de espacio identificadas previamente. Se ha llevado a cabo un proceso de selección cuidadoso para determinar qué partes son esenciales y cuáles pueden ser eliminadas sin comprometer la funcionalidad del sistema.

Además de reducir el tamaño y mejorar la eficiencia, se ha prestado especial atención a la integración de componentes y a la disposición de los mismos. Se ha buscado una configuración que permita una mejor organización y un uso más efectivo del espacio disponible.

Sin embargo, aunque el diseño presentado cumplía con los requerimientos de optimización de rendimiento y la eficiencia del diseño, así como el cumplimiento de las limitaciones de espacio, se presentaba un problema de No modulación, siendo necesaria la aplicación de diferentes cambios a la hora de replicarla.

Para conocer con qué dispositivo se establece la comunicación, el maestro (Arduino) ha de enviar un byte con los bits que componen la dirección del dispositivo esclavo (controlador I2C) con el que se quiera establecer la comunicación. La dirección de cada dispositivo esclavo se establece con los pines A0, A1 y A2 del controlador I2C, de tal forma que, para definir la dirección del primer esclavo, por ejemplo, conectaremos A0, A1 y A2 a GND (masa) mientras que para la dirección del segundo esclavo A0 y A1 irán conectados a GND y A2 a VDD. Conectar estos pines a GND implica asignarles un ‘0 lógico’ y conectarlos a VDD implica asignarles un ‘1 lógico’. [3]

Número de esclavo.	Dirección de Bits (A0, A1 y A2).			Número de Motor.
	A0	A1	A2	
0	0	0	0	Motor 1
1	0	0	1	Motor 2
2	0	1	0	Motor 3
3	0	1	1	Motor 4
4	1	0	0	Motor 5
5	1	0	1	Motor 6
6	1	1	0	-
7	1	1	1	-

Tabla 1. Tabla de direcciones de esclavo.

Con objetivo de poder definir dichas direcciones de esclavo sin necesidad de variar el diseño del circuito, se ha propuesto la implementación de un dispositivo Switch pulsador en cada dispositivo esclavo que permita asignar mediante el pulsador un ‘0 lógico’ o un ‘1 lógico’ dependiendo de los requerimientos del usuario.

Por último, fue necesaria la incorporación de nuevas resistencias PULL-UP para asegurar el 1 lógico (5v) y el 0 lógico (0v) en los dispositivos Switch pulsadores. Utilizando para ello resistencias de 4,7 KΩ

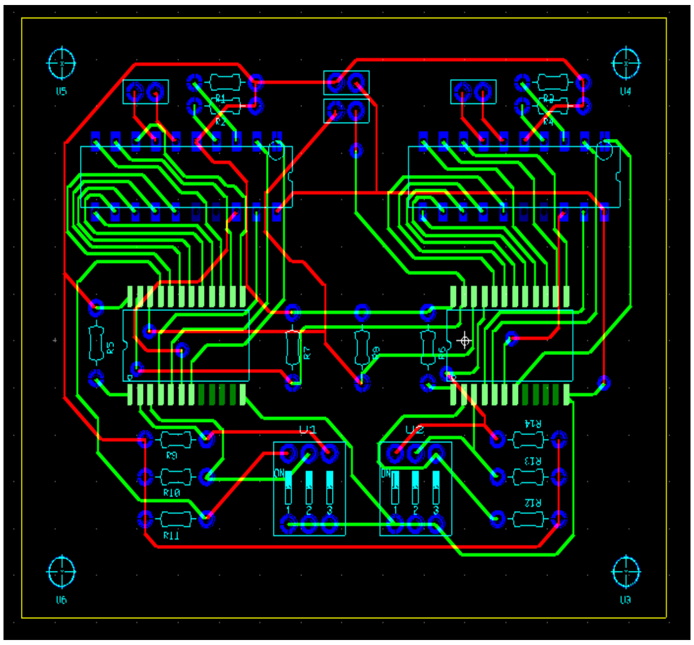


Figura 21: Propuesta final Diseño de la placa PCB I2C.

En la figura 21, 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. Ha de tenerse en cuenta que este circuito debe ser replicado tres veces para formar el circuito completo.

Las principales características a destacar en este diseño final de la placa I2C son las siguientes:

- La placa tiene unas dimensiones de 65 mm por 80 mm.
- Las pistas tienen anchos de 0,4 mm.
- Se usan cuatro Headers HDR1X2 para realizar las conexiones con los dispositivos externos.
- Se usan 2 Switch pulsadores que asignan la dirección del esclavo.
- Dos 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 P1 y P0 de cada contador.
- Un total de 14 resistencias Pull-Up que aseguran el 1 lógico (5V) y el 0 lógico en los diferentes elementos del circuito.

Por otro lado, con el programa Multisim 11.0 se puede hacer una simulación de cómo sería la placa en 3D una vez fabricada. A continuación, en la figura 22 se puede observar en detalle la cara *Top* del diseño propuesto y en la figura 22 se puede observar la cara *Bottom.*

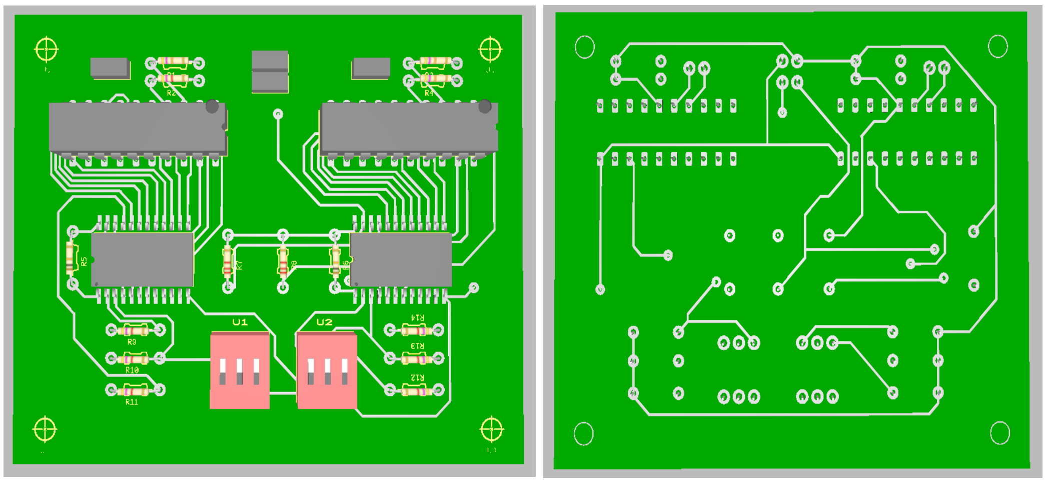


Figura 22: Diseño 3D placa PCB I2C

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

En este apartado se presenta la investigación realizada en el Trabajo de Fin de Grado titulado "Integración en PCB de la electrónica para el Scorbot-ER V+ y el Scorbot-ER IX, y diseño del gestor de alarmas"[1], llevado a cabo por Oscar Jesús Díaz de la Fe. La contribución de este trabajo es fundamental para el proyecto, ya que proporciona información valiosa sobre el método de implementación de los conectores del 'Scorbot-ER V+' en el 'Scorbot-ER IX'. [1]

La información y los resultados obtenidos de este trabajo son de gran relevancia para el proyecto, ya que nos brindan un enfoque y una base sólida para abordar la integración de los conectores del 'Scorbot-ER IX' en el diseño de nuestra PCB.

Lo primero que se hará será buscar el catálogo del Scorbot-ER IX de Intelitek para poder ver cómo son las conexiones y el funcionamiento de cada uno de sus pines. A continuación, se mostrarán los conectores y una tabla mostrando las conexiones requeridas.

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

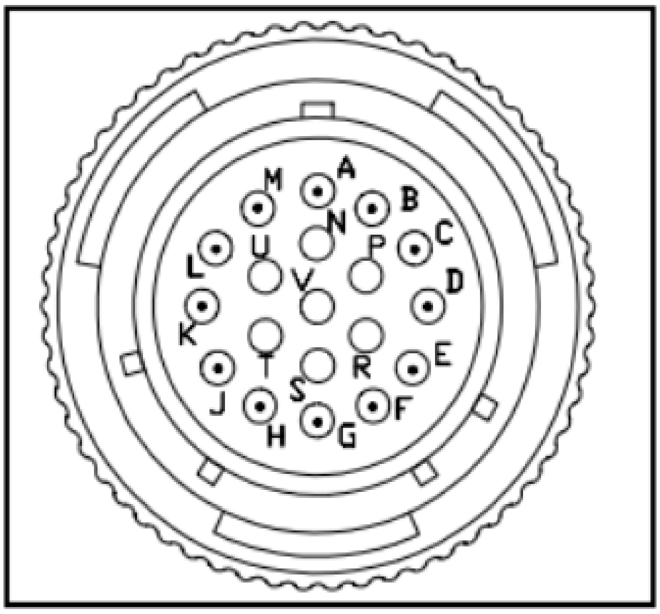


Figura 23: Conector tipo *burndy* de 19 pines

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 24: Tabla del conector Burndy

El cable del encoder, que conecta el controlador a los codificadores del motor y los interruptores, contiene 36 derivaciones. La figura 26 muestra el conector hembra D37.

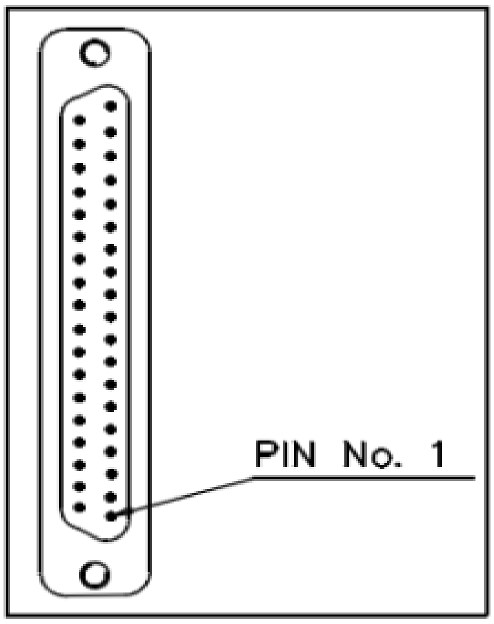
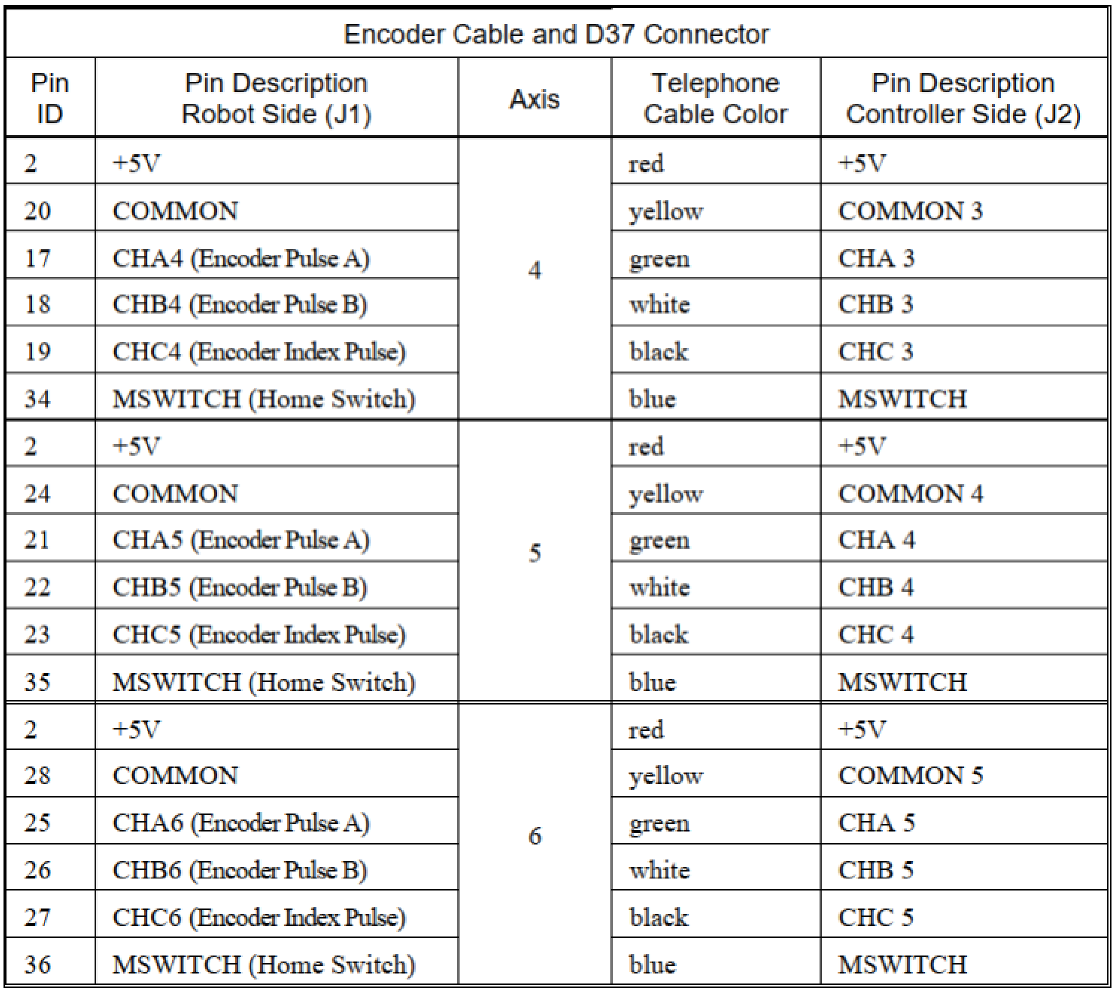


Figura 25: Conector D-37

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 26: 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 27: Tabla del cable del encoder 2

Ahora que conocemos la descripción de cada pin, procedemos compararlo con el diseño del 'Scorbot-ER V+'.

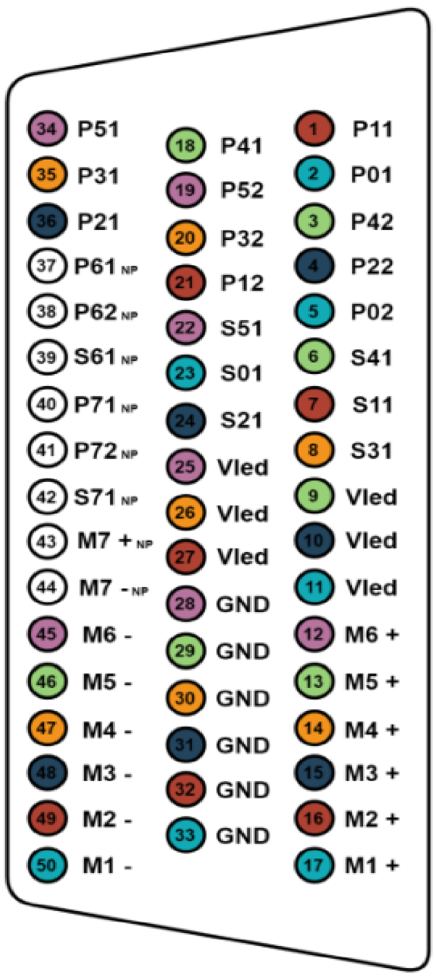


Figura 28: conexiones Scorbot-ER V extraído del TFG de Ana Estévez Pérez

Como podemos comprobar, la tensión Vled de 5V y la GND es común. Las señales de M+, M-, switch, pulse A y pulse B son propios de cada motor. El ‘encoder index pulse’, es una señal digital que se genera en el encoder una vez por revolución, permitiendo dar mayor precisión en los codificadores rotativos. Sin embargo, siguiendo los objetivos del proyecto, se busca un diseño que permita ser utilizado para ambos brazos robóticos. Teniendo en cuenta que el dispositivo ‘Scorbot-ER V+’ no dispone de dicha señal, no queda incluida esta nueva señal digital en el software diseñado por lo que se ha decidido excluir su estudio en este proyecto. Se ha considerado que los beneficios y objetivos principales del proyecto se pueden lograr sin la inclusión de esta señal adicional.

5.5. Evolución del diseño de la placa *Main* PCB

A continuación, se expondrán las diferentes etapas por las que ha ido evolucionando el diseño de la placa principal o placa *Main* de circuito impreso. La realización de este diseño presentó varios problemas iniciales.

Primeramente, se necesitaba una huella del conector D-50 Para ello, se estudió el datasheet del modelo de conector comprado y se buscó una huella que coincidiera con las dimensiones del conector en la biblioteca disponible del ’Multisim 14.0’. Tras no poder localizar una huella que coincidiera con las dimensiones solicitadas, se procedió a crear una huella de forma manual. Sin embargo, debido a las prestaciones que nos ofrece dicho software, no se podía generar una huella de tal grado de complejidad. Como medida resolutiva, se probó la instalación de diferentes softwares libres de diseño tales como 'Altium', KiCad y 'EasyEda'. El objetivo principal era encontrar un software que nos permitiera acceder a la huella de los conectores solicitados para poder realizar el diseño. Tras haber realizado una inspección de los programas mencionados nos decidimos por 'EasyEda'.

'EasyEda' se trata de un software de libre acceso universitario cuyas prestaciones y funciones son similares al 'Multisim 14.0', teniendo una interfaz intuitiva y fácil de aprender. Además, el software incluye una amplia gama de componentes reales en diferentes tiendas internacionales pudiendo acceder a las huellas indispensables.

Una vez encontrado un software donde tuviéramos acceso a las huellas necesarias, el siguiente paso fue comenzar el diseño. La idea del diseño inicial era sencilla, separar las casi 50 conexiones del brazo robótico 'Scorbot-ER V+' en 6 grupos, un grupo para cada motor. Una vez separadas dichas conexiones, se tendrían que ubicar de forma ordenada en un conector externo para llevarlas mediante un cable externo al resto de los componentes hardware. Como componentes adicionales, necesitábamos la inclusión de 12 resistencias Pull-Up para asegurar el '1 Lógico' de los encoders y un conector de 2 pines para las conexiones de alimentación de 5 voltios y tierra.

Posteriormente, ca fin de cumplir todos los objetivos del diseño, se estudiará la posibilidad de la inclusión de una segunda placa *Main* que sirviera de soporte para el 'Scorbot-ER IX'. La idea presentada fue hacer una réplica de la placa *Main* en la que se variara únicamente los conectores de entrada, es decir, realizar un cambio del conector D-50 por los conectores D-37 y el conector 'Burndy'.

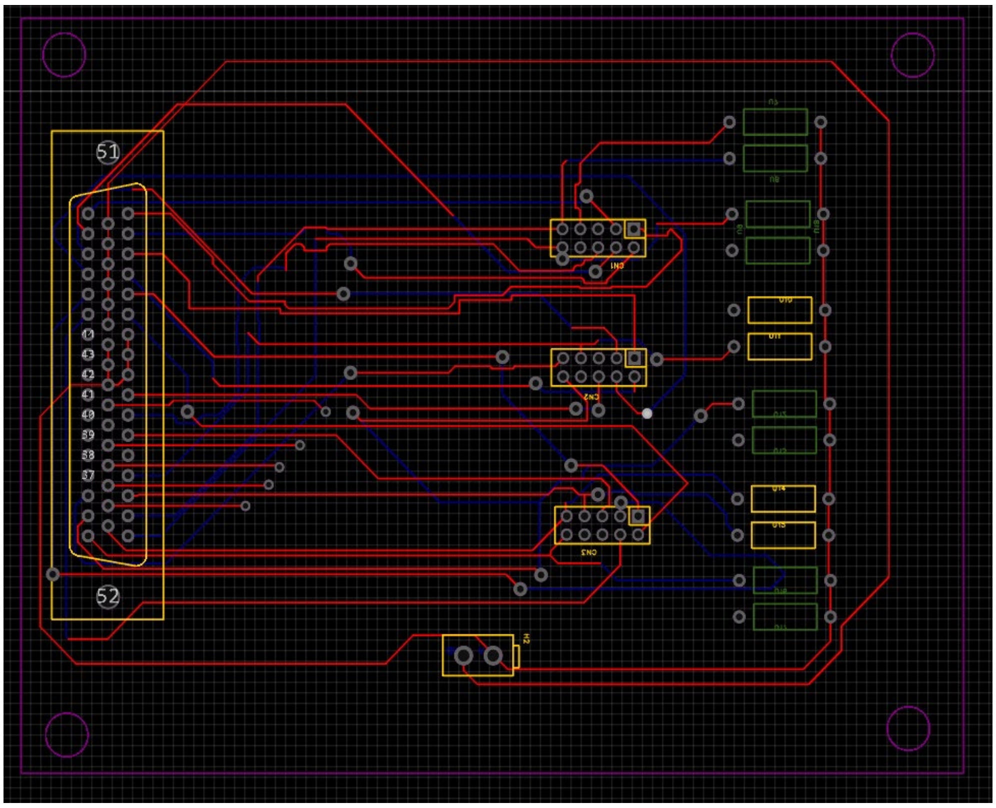


Figura 29. Propuesta inicial incompleta del diseño de placa PCB *Main*.

En la figura 29, se muestra el primer diseño propuesto para la fabricación de la placa *Main*, el cual resultó en un modelo incompleto y, por lo tanto, inservible.

Mientras que se iban terminando de realizar las últimas conexiones y revisiones de la placa, se estudió la viabilidad del plan inicial sobre realizar dos placas *Main*, una para cada brazo robótico. Tras un estudio minucioso de dicho planteamiento, se llegaron a las conclusiones de que era un plan poco viable debido al gran derroche tanto económico como material que supondría dicha fabricación.

En primer lugar, aunque el diseño propuesto era un diseño compacto y eficiente en sí, al deber realizarse dos placas diferentes, siendo necesarias múltiples conexiones externas, estas podrían ocasionar nuevamente un problema de limitaciones de espacio, las cuales podrían generar un problema de funcionalidad del sistema.

Asimismo, otro de los grandes desafíos encontrados fue en las conexiones a las 'Placas I2C PCB' y el resto del hardware. Si se llegara a aprobar la fabricación de dos placas *Main*, al realizarse las conexiones de la primera placa, la segunda placa no tendría conectores disponibles en el hardware para el conexionado, ya que tanto las conexiones del 'Scorbot-ER V+' como el 'Scorbot-ER IX' desembocan en el mismo dispositivo.

Estos obstáculos llevaron a reconsiderar el enfoque inicial del proyecto. En lugar de tener dos placas *Main* separadas, se buscó una solución que permita la compatibilidad y el funcionamiento conjunto de ambos brazos robóticos con una sola placa *Main*. Esto ayudará a superar los desafíos relacionados con el espacio y las conexiones.

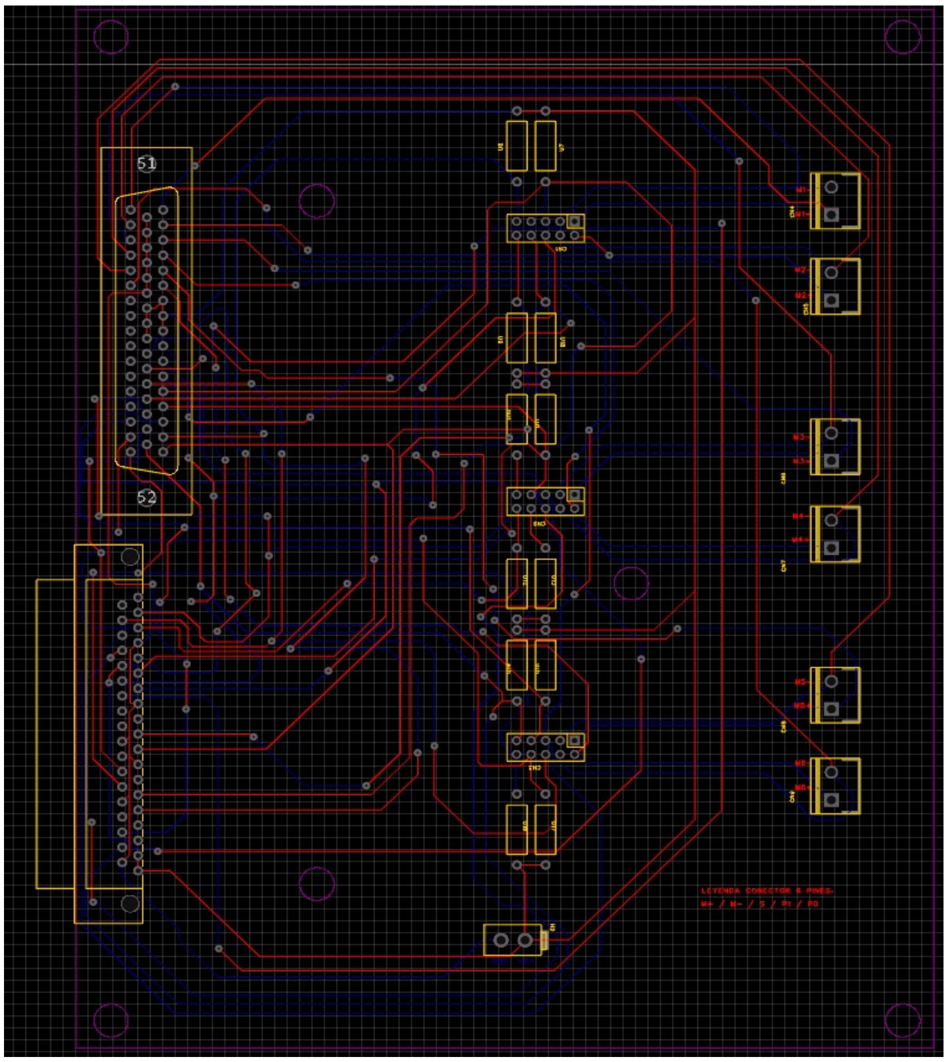


Figura 30. Propuesta final del diseño de placa PCB *Main*.

En la figura 30, 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.

Siguiendo lo comentado anteriormente se buscó realizar un diseño compacto y eficiente que implementaran las conexiones de ambos brazos robóticos. En este diseño propuesto, el usuario conecta el brazo robótico que desea utilizar y dichas conexiones se transmiten de manera directa a su correspondiente salida.

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

- La placa tiene unas dimensiones de 150 mm por 190 mm.
- Las pistas tienen anchos de 0,2 mm.
- Se usan seis Headers HDR1X10 para realizar las conexiones con los dispositivos externos. El conexionado del mismo queda incorporado en la serigrafía de la placa para una mayor facilidad del usuario.
- Se usan seis Headers HDR1X2 para realizar las conexiones con el conector tubular ya que no se encontró un conector físico adaptado a PCB. El conexionado del mismo queda incorporado en la serigrafía de la placa para una mayor facilidad del usuario.
- Un conector D-50 adaptado a PCB.
- Un conector D-37 adaptado a PCB.
- Dos conectores HDR1X2 para las señales de alimentación de 5 voltios y tierra.
- Un total de 12 resistencias Pull-Up de 100KΩ que aseguran el 1 lógico (5V) y el 0 lógico en las salidas de los encoders.

5.5.1. Integración del conector 'Burndy' en Placa PCB *Main*

Durante el desarrollo de este proyecto, se logró integrar todos los requisitos iniciales en el diseño de la placa PCB, a excepción del conector tubular de 19 pines 'Burndy'. Desde el inicio, una de nuestras primeras tareas fue buscar y adquirir el material necesario para la elaboración del proyecto. Realizamos búsquedas en diversas tiendas locales y páginas web internacionales. Sin embargo, nos encontramos con la dificultad de no encontrar el conector adecuado que se pudiera soldar directamente en la placa PCB.

Ante esta situación, se tuvo que buscar una solución alternativa y se optó por utilizar el conector tubular 'Burndy' que se ajustara a la caja y llevar las conexiones necesarias mediante cables hasta la placa PCB *Main*. Aunque esta solución implicaba un enfoque distinto al originalmente planeado, nos permitó avanzar con el proyecto y cumplir con los demás requisitos.

Para garantizar la seguridad y el rendimiento del circuito, era crucial seleccionar una sección de cable adecuada que evitara caídas de voltaje significativas durante la transmisión de corriente y previniera cualquier riesgo de incendio. Dado que el conector 'Burndy' es responsable de transmitir la corriente de alimentación de los motores, es necesario manejar altas intensidades de corriente de manera segura.

Para calcular la sección del cable necesaria, se realizaron los siguientes cálculos:

$$S = \frac{2 * L * I * COS(\phi) * \rho}{\Delta V} = \frac{2 * 0{,}5 * 2 * COS(0.9) * 0.019}{0.03} = 1{,}27\ mm^2$$

Sabiendo que:

- S: Sección del cable.
- L: Longitud cable.
- I: Intensidad de corriente.
- Φ: Factor de potencia.
- ρ: Conductividad del cobre.
- ΔV: Caída de tensión máxima.

De acuerdo con los requerimientos del diseño y para mantener una funcionalidad óptima del circuito, se determinó que era necesario utilizar un cable con una sección mínima de 1,27 mm^2.

Es importante destacar que la adaptación de la solución del conector 'Burndy' no compromete la funcionalidad ni el rendimiento del circuito. Aunque se requirió realizar ajustes en la implementación, se logró mantener la integridad de las conexiones y asegurar una comunicación efectiva entre los brazos manipuladores y la placa PCB.

La selección de una sección de cable adecuada y la adopción de medidas de seguridad adicionales fueron pasos cruciales para garantizar la operatividad y la seguridad del circuito en su conjunto. Estos ajustes nos permitieron superar los desafíos encontrados y lograr un diseño final confiable y eficiente.

5.6. Prueba de funcionamiento del prototipo y ajustes realizados

Una vez finalizada la fabricación de los 4 diseños de placas PCB y tras realizar una exhaustiva revisión del funcionamiento de estas, el siguiente paso es el montaje del hardware.

Para alojar el hardware, se ha optado por utilizar una caja de ordenador reciclada, la cual ha sido desmontada por completo para crear espacio para la instalación del prototipo. Aunque lo ideal sería diseñar un contenedor específico y adaptado a las necesidades del hardware, la elección de esta caja de ordenador nos permite ahorrar tanto en costos como en tiempo de desarrollo. Además, cabe destacar que el diseño de una caja de ordenador se ajusta adecuadamente a los requisitos de espacio del prototipo.

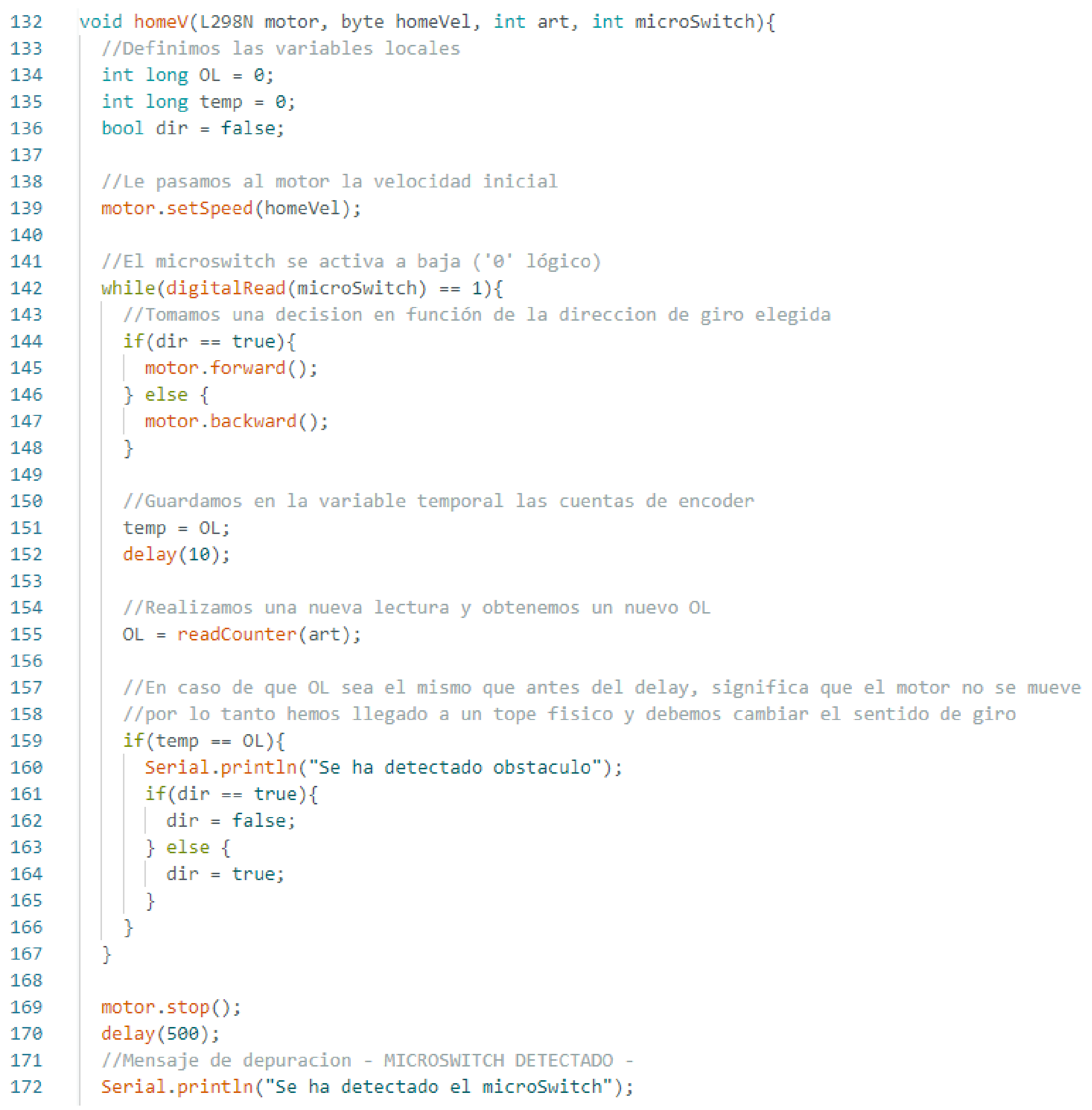
Una vez completado el ensamblaje del hardware, el paso final para concluir el proyecto será realizar un estudio del rendimiento del software desarrollado en el Trabajo de Fin de Grado titulado "*Implementación de la electrónica para el control dinámico de un manipulador Scorbot-ER. Desarrollo de la comunicación*", realizado por Carlos Javier Siverio Suarez, utilizando el prototipo de control que se ha fabricado.

Este estudio permitirá evaluar y analizar el funcionamiento del software en conjunto con el hardware implementado, asegurando que cumple con los objetivos establecidos y que se logra un control dinámico efectivo del manipulador Scorbot-ER V+. Para ello, se realizarán pruebas y mediciones, se compararán los resultados obtenidos con los esperados y se identificarán posibles mejoras o ajustes necesarios. El objetivo final es garantizar un rendimiento óptimo y una comunicación adecuada entre el software y el hardware del sistema de control del brazo manipulador.

El objetivo de estas pruebas será ajustar el prototipo para que pueda operar de manera óptima y sin ningún inconveniente, asegurando un funcionamiento similar al de su predecesor. Durante las pruebas, se evaluará el rendimiento del prototipo en diferentes escenarios y se verificará su capacidad para realizarlas funciones desarrolladas en el software. Se prestará especial atención a aspectos como la precisión de los movimientos, la respuesta a comandos y la estabilidad en diferentes configuraciones y condiciones de funcionamiento llevando a cabo una comparación detallada con su predecesor. Se buscará identificar posibles áreas de mejora y realizar los ajustes necesarios para garantizar un funcionamiento óptimo del prototipo.

Una vez comenzadas las pruebas, observamos primeramente en la función principal 'HOME' que el dispositivo falla en su funcionamiento bajo ciertos escenarios. Para solucionar los inconvenientes encontrados, realizamos una comprobación y estudio exhaustivo del código implementado.

Durante este estudio, se identificó un inconveniente en el código original. El código había sido diseñado y preparado para funcionar en condiciones ideales, lo que ocasionaba que, cuando el brazo del manipulador se encontraba en diferentes posiciones, no se generaran los movimientos adecuados.[12]



```
void homeV(L298N motor, byte homeVel, int art, int microSwitch){
  //Definimos las variables locales
  int long OL = 0;
  int long temp = 0;
  bool dir = false;

  //Le pasamos al motor la velocidad inicial
  motor.setSpeed(homeVel);

  //El microswitch se activa a baja ('0' lógico)
  while(digitalRead(microSwitch) == 1){
    //Tomamos una decision en función de la direccion de giro elegida
    if(dir == true){
      motor.forward();
    } else {
      motor.backward();
    }

    //Guardamos en la variable temporal las cuentas de encoder
    temp = OL;
    delay(10);

    //Realizamos una nueva lectura y obtenemos un nuevo OL
    OL = readCounter(art);

    //En caso de que OL sea el mismo que antes del delay, significa que el motor no se mueve
    //por lo tanto hemos llegado a un tope fisico y debemos cambiar el sentido de giro
    if(temp == OL){
      Serial.println("Se ha detectado obstaculo");
      if(dir == true){
        dir = false;
      } else {
        dir = true;
      }
    }
  }

  motor.stop();
  delay(500);
  //Mensaje de depuracion - MICROSWITCH DETECTADO -
  Serial.println("Se ha detectado el microSwitch");
```

Figura 31. Fragmento de código extraído.

En la figura 31 se muestra un fragmento del código desarrollado para la función 'HOME', la cual tiene como objetivo posicionar los motores en una posición de reposo determinada por los finales de carrera y establecer los contadores en cero.

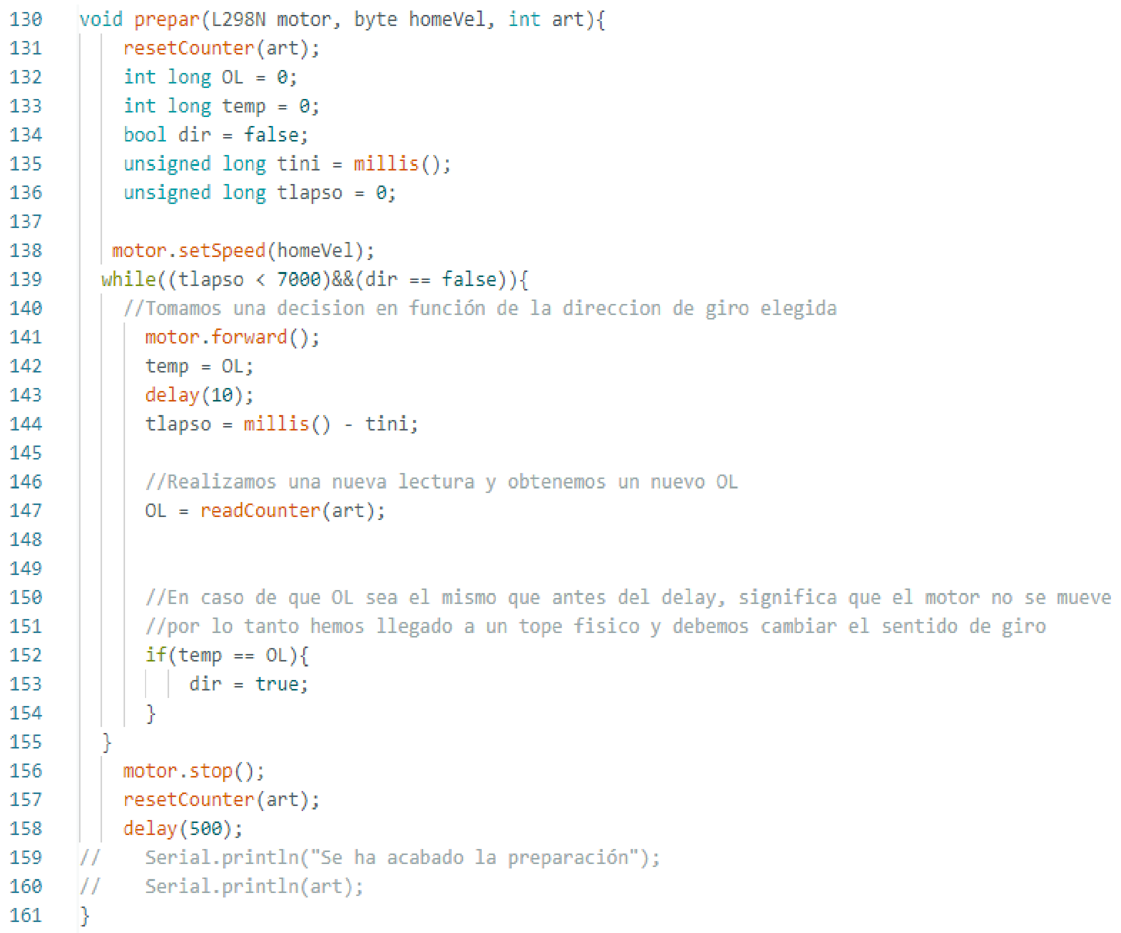
El propósito de esta función es preparar el manipulador para un funcionamiento adecuado posteriormente, especialmente cuando el manipulador se apaga o se manipula externamente y se pierde la información sobre su posición.

En el fragmento de código presentado, se puede observar que se activa el motor correspondiente y se realiza un barrido hasta que se encuentre un obstáculo o se alcance el final de carrera. Una vez se encuentra un obstáculo, el motor cambia de dirección y comienza el ciclo nuevamente hasta que se detecta el final de carrera. El orden de movimiento de los motores propuesto comenzaría por el motor 1, la base. Continuaría con el movimiento del motor 3, posteriormente el motor 2 y por último los motores que accionan la pinza del manipulador.

Este enfoque funciona correctamente en la mayoría de los motores. Sin embargo, se ha identificado que en ciertas condiciones iniciales de la articulación del motor 2, puede ocurrir que el motor 3 no logre alcanzar su final de carrera siguiendo estos movimientos. Esto se debe a que el movimiento de la articulación del motor 3 se ve afectado por el movimiento de la articulación del motor 2, de manera que cuando la articulación 2 se encuentra retraída, la articulación del motor 3 tiene restringido su movimiento.

Después de realizar diversas modificaciones en el código original sin lograr una solución óptima al problema, se decidió implementar una nueva función llamada "Prepar" para abordar el inconveniente registrado. El propósito de esta función era elevar la articulación del motor 3 lo suficiente para permitir que la función "Home" se ejecutara de manera fluida.

Se desarrolló una función "Prepar" cuyo propósito era levantar la articulación del motor 3 lo suficiente para poder ejecutar posteriormente la función "Home" con naturalidad. Este sistema solucionó el inconveniente que teníamos en el motor 3 ya que, al comenzar la función "HOME" levantado, al ejecutar el movimiento del motor 2, la articulación 3 tendría posteriormente suficiente rango de movimiento para conseguir accionar el sensor de final de carrera.



```
void prepar(L298N motor, byte homeVel, int art){
    resetCounter(art);
    int long OL = 0;
    int long temp = 0;
    bool dir = false;
    unsigned long tini = millis();
    unsigned long tlapso = 0;

   motor.setSpeed(homeVel);
  while((tlapso < 7000)&&(dir == false)){
    //Tomamos una decision en función de la direccion de giro elegida
      motor.forward();
      temp = OL;
      delay(10);
      tlapso = millis() - tini;

      //Realizamos una nueva lectura y obtenemos un nuevo OL
      OL = readCounter(art);


      //En caso de que OL sea el mismo que antes del delay, significa que el motor no se mueve
      //por lo tanto hemos llegado a un tope fisico y debemos cambiar el sentido de giro
      if(temp == OL){
          dir = true;
      }
  }
    motor.stop();
    resetCounter(art);
    delay(500);
//    Serial.println("Se ha acabado la preparación");
//    Serial.println(art);
}
```

Figura 32. Propuesta de función "Prepar".

En resumen, al activar el modo "Home" en el software, el dispositivo primero ejecuta la función "Preparatoria" para elevar la articulación del motor 3 y evitar restricciones de movimiento. Una vez que la articulación se ha elevado, se ejecuta la función original "Home", que funciona correctamente en condiciones ideales.
Por motivos de operar el manipulador de la forma más similar al controlador original, se ha decidido cambiar el orden de funcionamiento de los motores quedando el controlador de la siguiente manera:

1- Se ejecuta función "Prepar"; la articulación del motor 3 se levantará.
2- Se ejecuta la función "HOME"; comienza a moverse el motor 2 hasta el final de carrera.
3- Comienza a moverse el motor 3 hasta el final de carrera.
4- Comienzan a moverse los motores 3 y 4 en el mismo sentido (*Pitch* o Cabeceo) hasta el final de carrera.
5- Comienzan a moverse los motores 3 y 4 en sentido opuesto (*Roll* o Rotar) hasta el final de carrera.
6- Comienza a moverse el motor 6 hasta el final de carrera.
7- Comienza a moverse el motor 1 hasta el final de carrera.
8- Finaliza el modo Home y el dispositivo queda listo para utilizarse.

De esta manera, se logra solucionar el problema identificado y se establece un orden de funcionamiento coherente para el controlador del manipulador original.

Tras solucionar aparentemente todos los inconvenientes presentados en la función 'home', se detectó un error en la articulación 3 donde el manipulador tras finalizar dicha función depende de su acción previa se quedaba en diferentes posiciones.

La función 'home' tiene como objetivo inicial llevar a todos los motores desde cualquier posición inicial a un punto de reposo. Para ello, hace uso de los sensores *microswitch* ubicados en cada motor como final de carrera. Sin embargo, estos sensores en el motor 3 abarcan activados un amplio rango de movimiento. Esto ocasiona que si el brazo viene desde la posición superior su posición de detección de flanco sea diferente a si viene desde la posición inferior. En la figura 33 y 34 se pueden observar las diferentes posiciones en las que quedaría el manipulador mediante este error.

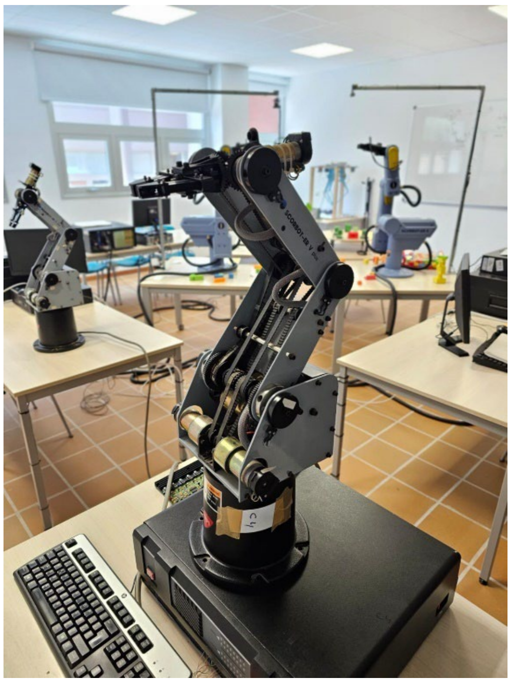
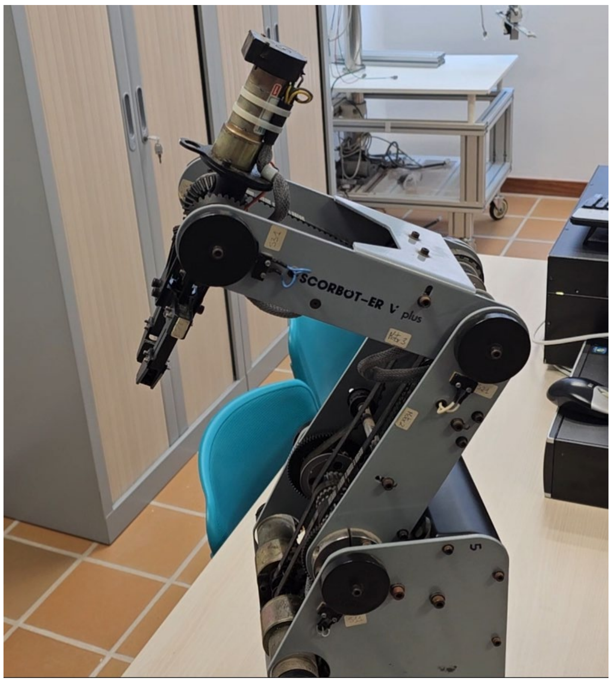



Figura 33 y 34. Error en función 'home'.

El movimiento designado por la articulación 3 constaba de tres pasos:

- Primeramente, el motor se dirigía dirección ascendente hasta que encontrara un obstáculo o se produjera la activación del sensor *microswitch*
- En segundo lugar, si el motor no hubiera accionado el sensor y hubiera encontrado un obstáculo, el motor se dirigía dirección contraria hasta que detectara el sensor *microswitch.*
- Una vez activado el sensor microswitch, entraba en la función 'descendFlanck' donde bajaba la velocidad del motor y volvía a cambiar su dirección en la que permanecía hasta encontrar el flanco descendente del mismo sensor.

Esta orden de movimiento ocasionaba que, si la posición original de dicha articulación era inferior al final de carrera, se encontraría el sensor *microswitch* en el flanco de subida del mismo (figura 34), y en caso contrario, se encontraría el sensor *microswitch* en el flanco de bajada (figura 33). En la figura 35 se puede observar en detalle la ubicación de ambos flancos.

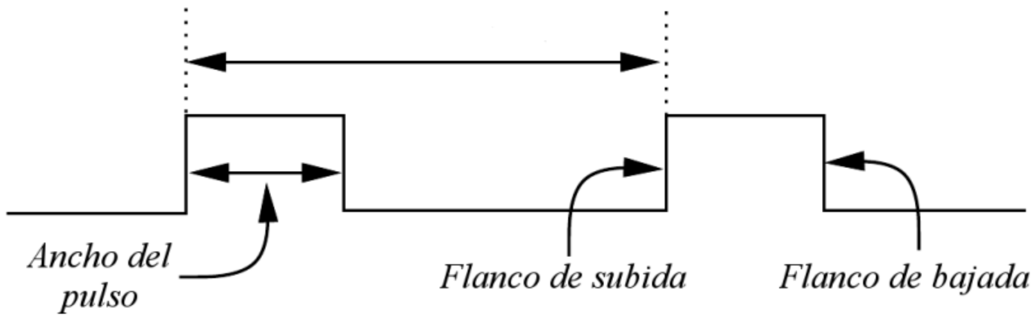
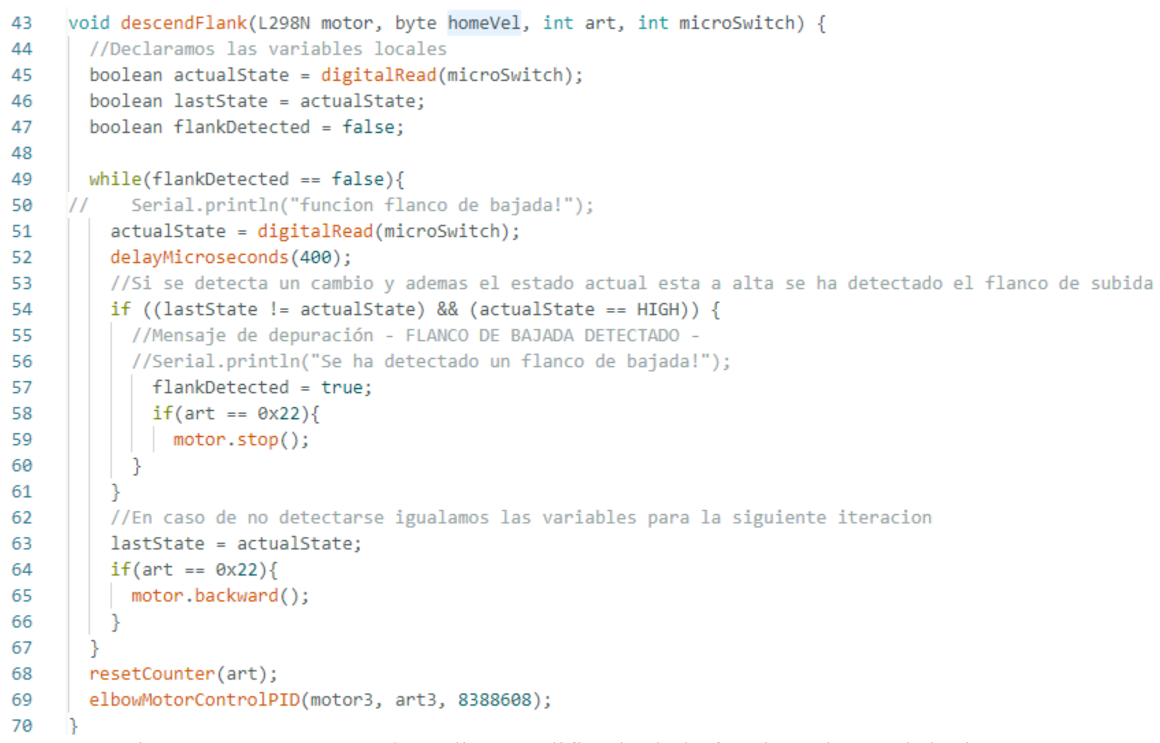


Figura 35. Esquema de los flancos del sensor *microswitch.*

Como medida resolutiva, se produzco una modificación en la función de detección de flancos 'descendFlanck' únicamente para la articulación del motor 3. De manera que, cuando se detectará el sensor *microswitch* el motor bajaría considerablemente su velocidad, y continuará dirección descendente hasta encontrar el flanco de subida. Una vez encontrado el flanco de subida, el motor se parará guardando la posición y se activará una función PID que mantenga el brazo robótico en esa misma posición. En la figura 36 se muestra el fragmento del código modificado.



```
void descendFlank(L298N motor, byte homeVel, int art, int microSwitch) {
  //Declaramos las variables locales
  boolean actualState = digitalRead(microSwitch);
  boolean lastState = actualState;
  boolean flankDetected = false;

  while(flankDetected == false){
//    Serial.println("funcion flanco de bajada!");
    actualState = digitalRead(microSwitch);
    delayMicroseconds(400);
    //Si se detecta un cambio y ademas el estado actual esta a alta se ha detectado el flanco de subida
    if ((lastState != actualState) && (actualState == HIGH)) {
      //Mensaje de depuración - FLANCO DE BAJADA DETECTADO -
      //Serial.println("Se ha detectado un flanco de bajada!");
        flankDetected = true;
        if(art == 0x22){
          motor.stop();
      }
    }
    //En caso de no detectarse igualamos las variables para la siguiente iteracion
    lastState = actualState;
    if(art == 0x22){
      motor.backward();
    }
  }
  resetCounter(art);
  elbowMotorControlPID(motor3, art3, 8388608);
}
```

Figura 36. Fragmento de código modificado de la función ‘descendFlank’.

6. Pruebas.

Durante el desarrollo del proyecto, nos enfrentamos a diversos inconvenientes menores que surgieron en diferentes etapas del proceso. Esto ocasionó que se tuvieran que realizar innumerables pruebas con fin de solucionar la mayor parte de los inconvenientes encontrados A continuación, resumiremos algunos de estos inconvenientes encontrados durante dichas pruebas y las soluciones que se implementaron para superarlos:

1- Incompatibilidad de componentes: En algunas ocasiones, durante el desarrollo de los diseños PCB encontramos que ciertos componentes no eran compatibles o no se ajustaban adecuadamente al diseño. Por ello, tuvimos que imprimir numerosos diseños y comprobar manualmente el tamaño de estos componentes con el fin de verificar si eran del tamaño adecuado.

2- Errores de conexión: Durante el montaje del hardware, nos encontramos con errores de conexión, como cables invertidos o conexiones sueltas. Para solucionar esto, revisamos detalladamente todas las conexiones y corregimos cualquier error de cableado o sujeción de componentes. En este caso tuvimos que adelantar la fabricación de la placa PCB *Main* ya que las soldaduras de conexiones del prototipo se habían degradado.

3- Problemas de alimentación: En algunas ocasiones, experimentamos problemas de suministro de energía, como caídas de voltaje o sobrecalentamiento. Al comenzar los testeos del diseño se utilizó una fuente externa proporcionada por el laboratorio cuya corriente estaba limitado a 2 amperios, siendo insuficiente para mover los diferentes motores del manipulador.

4- Relacionado con el punto anterior, debido a un fallo en un módulo L298N se produjo un calentamiento extremo del mismo estando cerca de ocasionar un incendio.

5- Errores de programación: Durante la implementación del software, nos encontramos con errores de programación que afectaban el funcionamiento del sistema. Se tuvo que modificar algunas funciones para conseguir una mejora en el funcionamiento.

7. Mejoras Futuras.

El alcance general de este proyecto ha sido desde su origen bastante ambicioso, buscando como objetivo el conseguir reemplazar todos los controladores de los manipuladores Scorbot-ER originales que se encontraban en el laboratorio.

En este trabajo de Fin de Grado se ha logrado cumplir estos objetivos de manera superficial, pero sigue siendo insuficiente para poder realizar un completo reemplazo. Para ello, es necesario realizar en su mayoría tareas de ajuste de tiempos y añadir mayor cantidad de funciones al software diseñado.

Por este motivo, en este apartado del documento se incluirán algunas propuestas de mejoras que mejorarían tanto estos tiempos de espera como las funcionalidades del prototipo. El objetivo final es desarrollar un diseño definitivo que pueda competir plenamente con los controladores originales, logrando así su reemplazo total.

- Desarrollo de una Interfaz gráfica de usuario también conocida como GUI (*graphical user interface*); actualmente la interfaz de operación del brazo robótico se realiza a través de Python mediante líneas de comando, siendo una interfaz estéticamente muy rudimentaria. Además, el programa requiere que se compile primeramente el código de Arduino. Esto hace que sea un sistema de difícil acceso sin conocimientos previos. Con el desarrollo de una correcta Interfaz gráfica permitiría una interacción más sencilla e intuitiva.
- Optimización general del Software implementado; actualmente tanto los tiempos de ejecución de las interacciones como los tiempos de espera son más altos de lo esperado. La razón principal recae en que la implementación actual solo permite mover los motores individualmente y de forma secuencial. Actualmente, estos tiempos pueden ser reducidos mediante ajustes en los controladores PID de los diferentes motores. Sin embargo, para solucionar el principal inconveniente, será necesario un reajuste completo del código para la implementación de tareas paralelas, es decir, mover varios motores simultáneamente.
- Desarrollo de un control PID para los motores 4,5 y 6; sin la implementación de este control el brazo robótico del prototipo no es capaz de realizar movimientos precisos con la pinza.
- Implementación de una interfaz de control para el Scorbot-ER IX; actualmente el prototipo diseñado no contiene una interfaz informática que permita operar este manipulador. Al tener una estructura física y una lectura de codificadores de posición diferentes, será necesaria la elaboración con sus respectivos cálculos cinemáticos de una nueva interfaz apropiada para este manipulador.

8. Conclusiones

El Trabajo de Fin de Grado se ha centrado en el desarrollo e implementación de una placa de Circuito Impreso (PCB) para el control de los brazos robóticos Scorbot-ER V+ y Scorbot-ER IX en el laboratorio de Robótica de la Universidad de La Laguna. Durante la realización de dicho proceso se han desarrollado principalmente conocimientos de diferentes programas de diseño de placas tales como Multisim 14.0 o EasyEda, así como unas nociones de su fabricación. Además, se han desarrollado habilidades de soldadura precisa de componentes THT y SMD, familiarización con el uso de datasheets de los componentes utilizados y, por último, un dominio profundo en el conocimiento de los brazos manipuladores usados.

Aunque no se haya podido implementar de forma satisfactoria la electrónica de control a ambos controladores Scorbot-ER, se han logrado los objetivos iniciales propuestos, consiguiendo un óptimo funcionamiento del Scorbot-ER V+, así como una implementación exitosa de los conectores del Scorbot-ER IX.

Se han desarrollado conocimientos de planificación de proyectos y un gran dominio de la búsqueda y resolución de inconvenientes encontrados, generando un profundo estudio en el estudio y comprensión de documentación técnica.

Por ello, se ha logrado una satisfacción con el desarrollo de las partes específicas del trabajo, ya que han permitido adquirir conocimientos y destrezas que 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 ha permitido un acercamiento y aprendizaje acerca de la realidad del trabajo de un ingeniero técnico industrial.

9. Conclusions

The Bachelor's Thesis has focused on the development and implementation of a Printed Circuit Board (PCB) for controlling the Scorbot-ER V+ and Scorbot-ER IX robotic arms in the Robotics Laboratory of the University of La Laguna. Throughout the project, knowledge of various PCB design programs such as Multisim 14.0 and EasyEda has been developed, along with an understanding of the fabrication process. Precise soldering skills for through-hole technology (THT) and surface-mount devices (SMD), as well as familiarity with component datasheets and a deep understanding of the manipulator arms used, have also been acquired.

Although the electronic control could not be successfully implemented in both Scorbot-ER controllers, the initial objectives were achieved, resulting in optimal functioning of the Scorbot-ER V+ and successful implementation of connectors for the Scorbot-ER IX.

Skills in project planning, problem-solving, and thorough study and understanding of technical documentation have been developed. This has led to a sense of satisfaction with the specific parts of the work, as they have provided valuable knowledge and skills for the future. These skills encompass programming, dynamic system control, and hardware development and manufacturing in the field of robotics.

In conclusion, the development of this project has provided an insight into the realities of work as a technical industrial engineer.

10. Presupuesto.

CONCEPTO	PRECIO UNITARIO	UNIDADES	IMPORTE
COSTES MATERIALES			**208,77 €**
COMPONENTES ELECTRÓNICOS			**92,59 €**
Arduino Mega 2560	29,95 €	1	29,95 €
Módulo L298N	3,48 €	3	10,44 €
LS7166	7,15 €	6	42,90 €
PCA9535	1,55 €	6	9,30 €
Resistencias (Paquete de resistencias)	5,99 €	1	5,99 €
Cables y conectores.			**36,70 €**
Conector D-50	23,20 €	1	23,20 €
Conector D-37	2,00 €	1	2,00 €
Conector tubular tipo Burndy	11,50 €	1	11,50 €
Elementos mecánicos			**66,98 €**
HDR1X2	0,81 €	18	14,58 €
HDR1X10 (caja completa)	13,40 €	1	13,40 €
Pines Conector tubular	1,95 €	20	39,00 €
Materiales de impresión			**12,50 €**
Material Fabricación Placa	2,50 €	5	12,50 €
COSTES DE EJECUCIÓN			**6.878,0 €**
TRABAJO DE INGENIERÍA			**6.000,00 €**
Horas de trabajo ingeniero técnico industrial	20,00 €	300	6.000,00 €
TRABAJO TÉCNICO			**878,00 €**
Horas de fabricación de carcasa	15,00 €	2	30,00 €
Unidad por fabricación de PCB	132,00 €	4	528,00 €
Horas de soldadura y montaje de PCB, estimado por técnico y materiales.	16,00 €	20	320,00 €

CONCEPTO		TOTAL	
COSTES MATERIALES			208,77 €
COSTES DE EJECUCIÓN			6.878,00 €
	TOTAL, COSTES		**7.086,8 €**
	GASTOS GENERALES (G.G.)	13%	921,28 €
	BENEFICIO INDUSTRIAL (B.I.)	6%	425,21 €
	SUMA DE G.G. Y B.I.		1.346,49 €
	COSTE ESTIMADO		**8.433,3 €**
	I.G.I.C	7%	590,33 €
	COSTE TOTAL DEL PROYECTO		**9.023,6 €**
EL COSTE TOTAL DEL PROYECTO ES DE NUEVE MIL VEINTITRÉS EUROS CON SESENTA CÉNTIMOS			

11. Bibliografía.

Referencias

[1] Oscar Jesús Díaz de la Fe, "*Integración en PCB de la electrónica para el Scorbot-ER V+ y el Scorbot IX, y diseño del gestor de alarmas"*, trabajo de fin de grado, julio 2020.

[2] Cristian Francisco Fariña Melián, "*Integración en PCB de la electrónica para el Scorbot-ER V+ y el Scorbot IX. Diseño algoritmos de control"*, trabajo de fin de grado, julio 2020.

[3] 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.

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

[5] http://manueldelgadocrespo.blogspot.com/p/arduino-mega-2560.html

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

[7] https://www.proto-electronics.com/es/

[8] https://www.nxp.com/

[9] https://es.rs-online.com/web/

[10] https://www.fineline-global.com/es/

[11] https://www.mokotechnology.com/es/pcb-manufacturing/

[12]https://www.tecnosalva.com/como-hacer-tus-circuitos-impresos-con-una-insoladora-casera-de-diodos-led/

[13] https://www.arduino.cc/

12. Datasheets

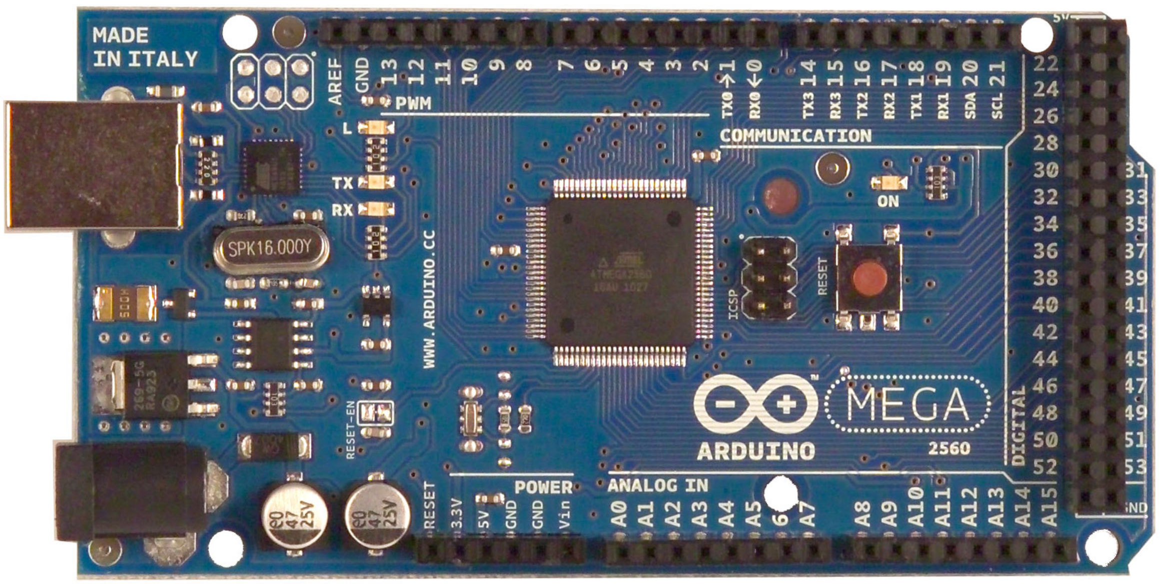
RobotShop

www.robotshop.com

La robotique à votre service! - Robotics at your service!

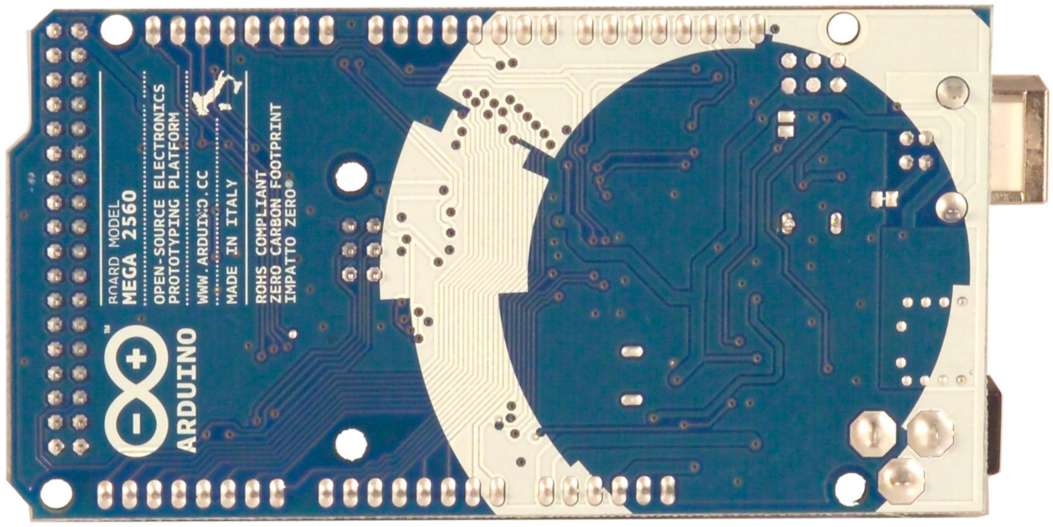
ARDUINO

Arduino Mega 2560 Datasheet



www.robotshop.com

La robotique à votre service! - Robotics at your service!



Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Schematic & Reference Design

EAGLE files: arduino-mega2560-reference-design.zip

Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Power

The Arduino Mega can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode() , digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Uno, Duemilanove and Diecimila.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH

value, the LED is on, when the pin is LOW, it's off.
- **I2C: 20 (SDA) and 21 (SCL).** Support I2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I2C pins on the Duemilanove or Diecimila.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).
A SoftwareSerial library allows for serial communication on any of the Mega2560's digital pins.
The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. For SPI communication, use the SPI library.

Programming

The Arduino Mega can be programmed with the Arduino software (download). For details, see the reference and tutorials.
The ATmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It

communicates using the original STK500 protocol (reference, C header files).
You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.
This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.
The Mega2560 contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Mega2560 has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics and Shield Compatibility

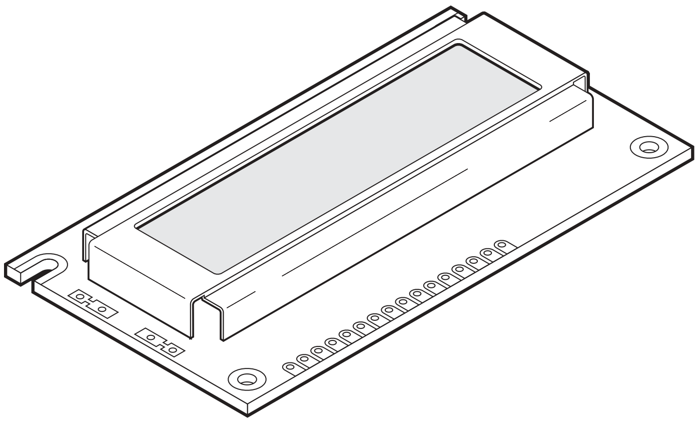
The maximum length and width of the Mega2560 PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega2560 is designed to be compatible with most shields designed for the Uno, Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega2560 and Duemilanove / Diecimila. *Please note that I_2C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).*

Alphanumeric LCD Display (16 x 2)

Order Code

LED008	16 x 2 Alphanumeric Display
FRM010	Serial LCD Firmware (optional)



Contents

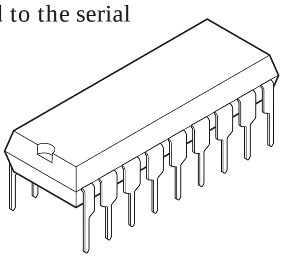
1 x 16x2 Alphanumeric Display
1 x data booklet

Introduction

Alphanumeric displays are used in a wide range of applications, including palmtop computers, word processors, photocopiers, point of sale terminals, medical instruments, cellular phones, etc. The 16 x 2 intelligent alphanumeric dot matrix display is capable of displaying 224 different characters and symbols. A full list of the characters and symbols is printed on pages 7/8 (note these symbols can vary between brand of LCD used). This booklet provides all the technical specifications for connecting the unit, which requires a single power supply (+5V).

Further Information

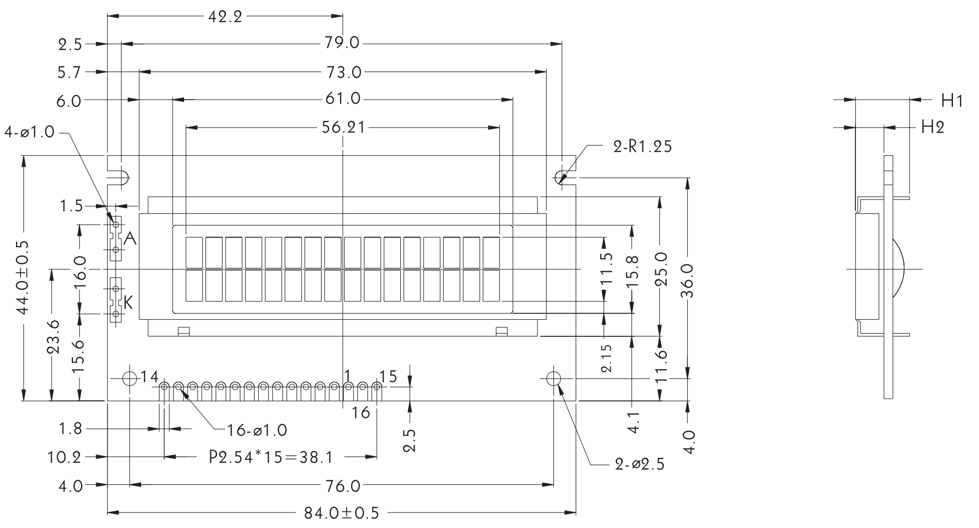
Available as an optional extra is the Serial LCD Firmware, which allows serial control of the display. This option provides much easier connection and use of the LCD module. The firmware enables microcontrollers (and microcontroller based systems such as the PICAXE) to visually output user instructions or readings onto an LCD module. All LCD commands are transmitted serially via a single microcontroller pin. The firmware can also be connected to the serial port of a computer.

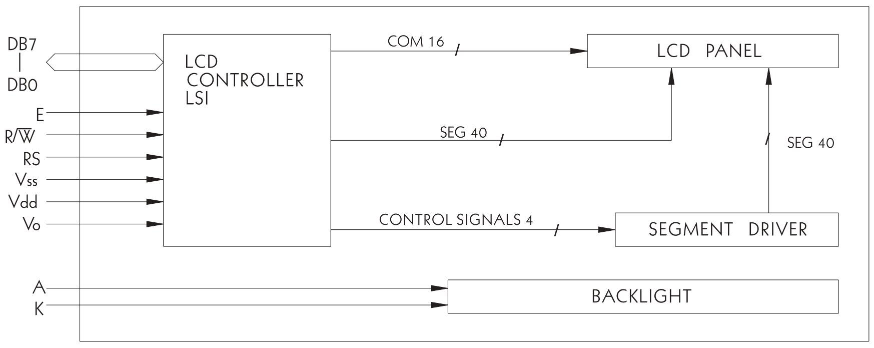


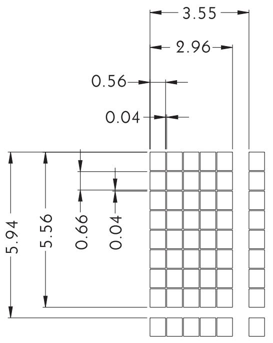
An example PICAXE instruction to print the text 'Hello' using the **serout** command is as follows:

```
serout 7,T2400,("Hello")
```

Outline Dimension and Block Diagram







The tolerance unless classified ±0.3mm

MECHANICAL SPECIFICATION			
Overall Size	84.0 * 44.0	Module	H2 / H1
View Area	61.0 * 15.8	W/O B/L	5.1 / 9.7
Dot Size	0.56 * 0.66	EL B/L	5.1 / 9.7
Dot Pitch	0.60 * 0.70	LED B/L	9.4 / 14.0

PIN ASSIGNMENT

Pin no.	Symbol	Function
1	Vss	Power supply (GND)
2	Vdd	Power supply (+5V)
3	Vo	Contrast Adjust
4	RS	Register select signal
5	R/W	Data read /write
6	E	Enable signal
7	DB0	Data bus line
8	DB1	Data bus line
9	DB2	Data bus line
10	DB3	Data bus line
11	DB4	Data bus line
12	DB5	Data bus line
13	DB6	Data bus line
14	DB7	Data bus line
15	A	Power supply for LED B/L (+)
16	K	Power supply for LED B/L (−)

ABSOLUTE MAXIMUM RATING

Item	Symbol	Conditions	Min.	Max.	Unit
Power Supply Voltage	Vdd−Vss	−	0	7	V
LCD Driving Supply Voltage	Vdd−Vee	−	0	13	V
Input Voltage	Vin	−	−0.3	Vdd+0.3	V
Operating Temperature	Topr	Nor.	0	50	℃
Storage Temperature	Tstg	Nor.	−20	+70	℃

ELECTRICAL CHARACTERISTICS (Vdd =+5V, Ta =25°C)

Item	Symbol	Conditions	Min.	Typ.	Max.	Unit
Logic Supply Voltage	Vdd	−	4.5	5	5.5	V
"H" Input Voltage	V_{IH}	−	2.2	−	−	V
"L" Input Voltage	V_{IL}	−	−	−	0.6	V
"H" Output Voltage	V_{OH}	−	2.4	−	−	V
"L" Output Voltage	V_{OL}	−	−	−	0.4	V
Supply Current	Idd	−	2	−	−	mA
LCD Driving Voltage	V_{LCD}	Vdd−Vo	4.3	−	4.8	V

Electrical Characteristics

Vdd = 5V±5%
Vss = 0V

Item	Symbol	Condition	Standard value			Unit	Applicable terminal
			Min.	Typ.	Max.		
Power voltage	Vdd		4.5	5.00	5.5	V	Vdd
Input H- level voltage	V_{IH}		2.2	—	Vdd	V	RS,R/$\overline{W}$,E DB0~DB7
Input L - level voltage	V_{IL}		−0.3	—	0.6	V	
Output H - level voltage	V_{OH}	$-I_{OH}=0.205mA$	2.4	—	—	V	DB0~DB7
Output L - level voltage	V_{OL}	$I_{OL}=1.2mA$	—	—	0.4	V	
I/O leakage current	I_{IL}	Vin = 0~Vdd	−1	—	1.0	μA	RS,R/$\overline{W}$,E DB0~DB7
Supply current	Idd	Vdd = 5V	2	—	—	mA	Vdd
LCD operating voltage	V_{LCD}	Vdd—V0	3.0	—	11.0	V	V0

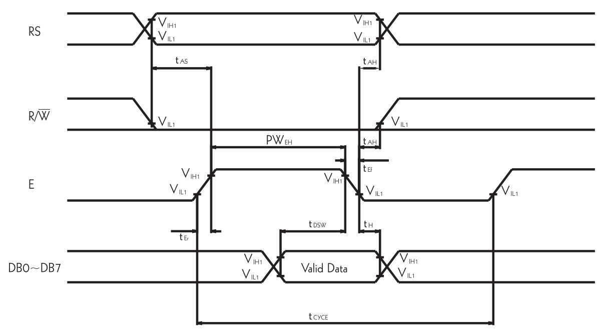
Timing Characteristics

Vdd = 5V±5%
Vss = 0V

Item		Symbol	Min.	Max.	Unit
Enable cycle time		T_{CYCE}	500	—	ns
Enable pulse width	"High" level	P_{WEH}	220	—	ns
Enable rise / fall time		T_{ER},T_{EF}	—	25	ns
Set-up time	RS,R/$\overline{W}$,E	T_{AS}	40	—	ns
Address hold time		T_{AH}	10	—	ns
Data set–up time		T_{DSH}	60	—	ns
Data delay time		T_{DDR}	60	120	ns
Data hold time (writing)		T_H	10	—	ns
Data hold time (reading)		T_{DHR}	20	—	ns
Clock oscillating frequency		T_{OSC}	270(TYP.)		KHz

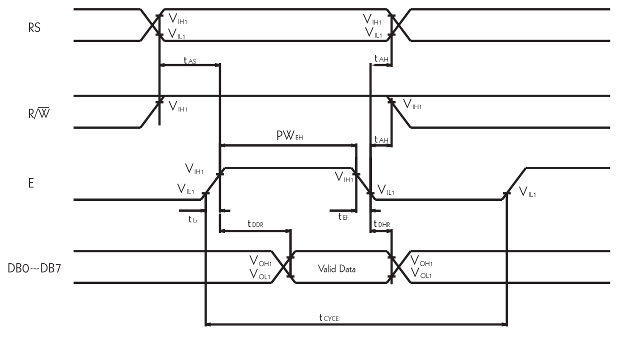
Timing Chart

◆ FIG.1 WRITE OPERATION



(Write Data from MPU to MODULE)

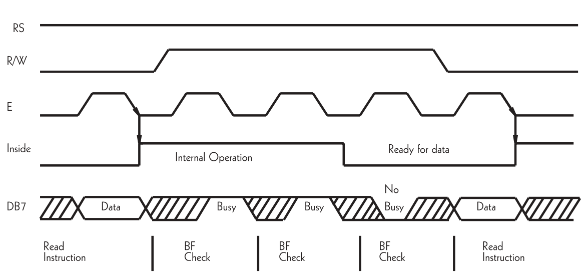
◆ FIG.2 READ OPERATION

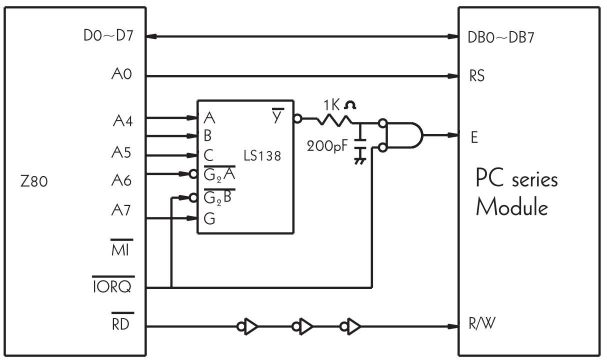


(Read Data from MODULE to MPU)

Interface with MPU

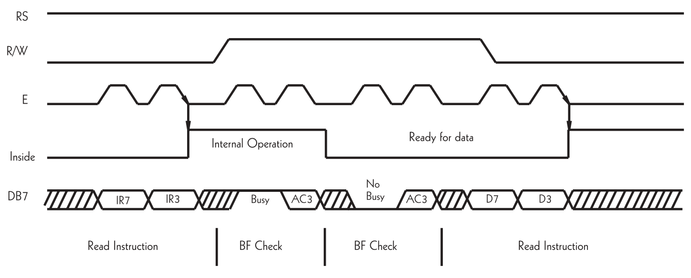
◆ Example of Interface with 8-bit MPU (Z80)



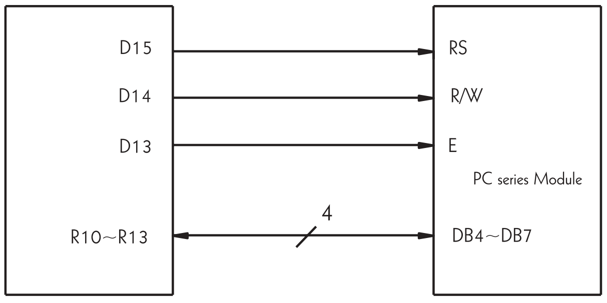


◆ Example of interface with 4-bit MPU

Interface with 4-bit MPU can be made through I / O port of 4-bit MPU. If there are enough I / O ports, data can be transfered by 8-bit, however, if there are not data transfer can be done by 4-bit in twice (select interface is 4-bit long), and timing sequence will be complicated in this case. Please take into account that 2 cycles of BF check is necessary, while 2 cycles of data transfer are also necessary.



Note:IR7,IR3:7th bit,3rd bit of instruction
AC3:3th bit of Address Counter



Features

(1) Interface with 8-bit or 4-bit MPU is available.

(2) 192 kind of alphabets, numerals, symbols and special characters can be displayed by built-in character generator (ROM).

(3) Other preferred characters can be displayed by character generator (RAM).

(4) Various functions of instruction are available by programming.
- Clear display • Cursor at home • On / off cursor
- Blink character • Shift display • Shift cursor
- Read / write display data.....etc.

(5) Compact and light weight design which can be easily assembled in devices.

(6) Single power supply +5V drive (except for extended temp. type).

(7) Low power consumption.

*Interface between data bus line and 4-bit or 8-bit MPU is available. Data transfer are made in twice in case of 4-bit MPU, and once in case of 8-bit MPU.

◆ If interface data is 4-bit long

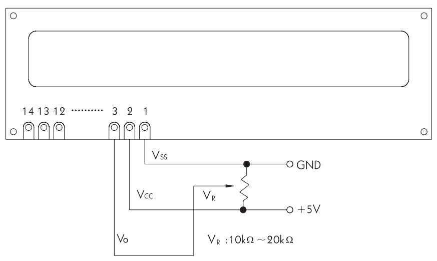
Data transfer are made through 4 bus lines from DB4 to DB7. (while the rest of 4 bus lines fromDB0 to DB3 are not used.) Data transfer with MPU are completed when 4-bit data are transfered in twice.
(first upper 4-bit data. then lower 4-bit data.)

◆ If interface data is 8-bit long

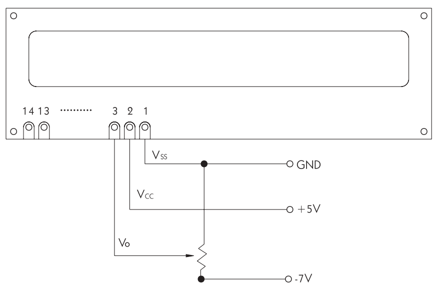
Data transfer are made through all of 8 bus lines from DB0 to DB7.

Example of Power Supply

◆ Normal Temperature Type



◆ Extended Temperature Type



◆ Examples of Temperature Compensation Circuits for Extended Temp Type. (Only for reference)

(A) 1/8Duty—1/4Bias

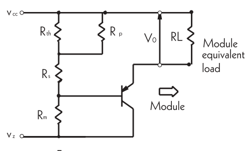


Fig.1

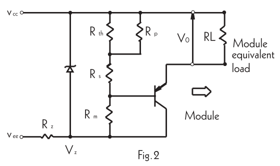


Fig.2

Thermistor: Rth(25˚C)=15[k-ohm], B=4200[K]
Resistors: Rp=30[k-ohm)], Rs=6.8[k-ohm], Rm=3.3[k-ohm]
Transistor: PNP Type
Vcc: +5V , Vss: 0V (Logic Supply)
Vz: -8[V] (-7.8 to -8.2[V])
Vee<Vz[V]. Rz=(Vz-Vee) / 5[k-ohm]

(B) 1/16Duty—1/5Bias

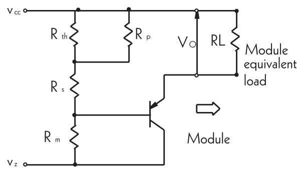


Fig.1

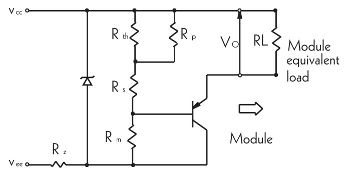


Fig.2

Thermistor: Rth(25˚C)=15[k-ohm], B=4200[K]
Resistors: Rp=510[k-ohm)], Rs=8.2[k-ohm], Rm=3.9[k-ohm]
Transistor: PNP Type
Vcc: +5V , Vss: 0V (Logic Supply)
Vz: -11[V] (-10.725 to -11.275[V])
Vee<Vz[V]. Rz=(Vz-Vee) / 5[k-ohm]

Instructions

Instruction	Code										Description	Executed Time(max.)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears all display and returns the cursor to the home position (Address 0)	1.64mS
Cursor At Home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position (Address 0). Also returns the display being shifted to the original position. DD RAM contents remain unchanged.	1.64mS
Entry Mode Set	0	0	0	0	0	0	0	1	1/D	S	Sets the cursor move direction and specifies or not to shift the display. These operations are performed during data write and read.	40μS
Display On / Off Control	0	0	0	0	0	0	1	D	C	B	Sets ON / OFF of all display (D), cursor NO / OFF (C), and blink of cursor position character (B).	40μS
Cursor / Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing DD RAM contents.	40μS
Function Set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL) number of display lines (L) and character font (F)	40μS
CG RAM Address Set	0	0	0	1	ACG						Sets the CG RAM address. CG RAM data is sent and received after this setting.	40μS
DD RAM Address Set	0	0	1	ADD							Sets the DD RAM address. DD RAM data is sent and received after this setting.	40μS
Busy Flag / Address Read	0	1	BF	AC							Reads Busy flag (FB) indicating internal operation is being performed and reads address counter counts.	0μS
CG RAM / DD RAM Data Write	1	0	WRITE DATA								Writes data into DD RAM or CG RAM.	40μS
CG RAM / DD RAM Data Read	1	1	READ DATA								Reads data from DD RAM or CG RAM.	40μS

Code		Descripion	Executed Time (max)

I/D = 1:Increment
I/D = 0:Decrement
S= 1:With display shift
S/C= 0:cursor movement
R/L= 1: Shift to the right
R/L= 0: Shift to the left
DL = 1: 8-bit
	DL = 0:4-bit
	N = 1:2lines
	N = 0:1line
	F = 1:5×10dots
	F = 0.5×7dots
	BF = 1:Internal operation is being performed
	BF = 0:Instruction acceptable
		DD RAM:Display Data RAM
		CG RAM:Character Generator RAM
		ACG:CG RAM Address
		ADD:DD RAM Address Corresponds to cursor address.
		AC:Address Counter, used for both DD RAM and CG RAM
		*:Invalid
			fcp or fosc = 250KHz
			However. when frequency changes. eecution time also changes
			Example
			if fcp or fosc is 270KHz,
			70μS x 250/270 = 37μS

Power Supply Reset

The internal reset circuit will be operated properly when the following power supply conditions are satisfied. If it is not operated properly, please perform initial setting along with the instruction.

Item	Symbol	Measuring Condition	Standard Value			Unit
			Min.	Typ.	Max.	
Power Supply RISE Time	tree	—	0.1	—	10	mS
Power Supply CFF Time	toff	—	1	—	—	mS

Reset function

◆ **Initialization Made by Internal Reset Circuit**

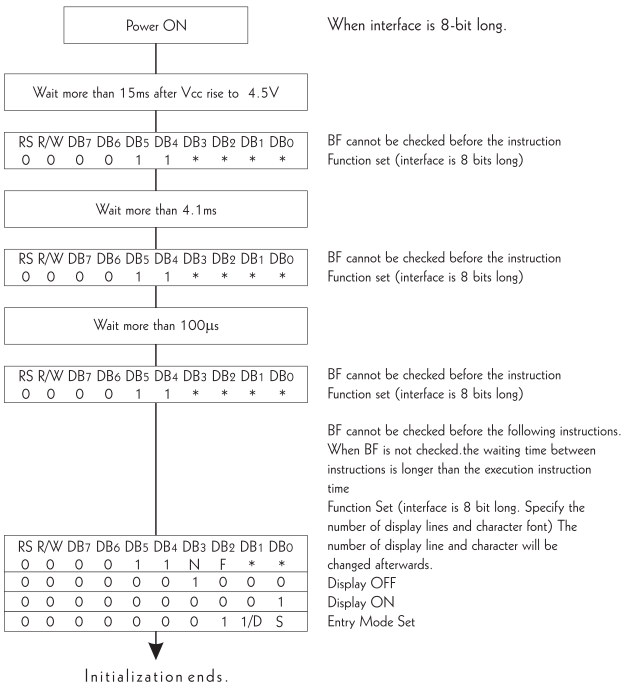
HD44780 automatically initializes (resets) when power is supplied (builtin internal reset circuit).
The following instructions are executed in initialization.
The busy flag (BF) is kept in busy state until initialization ends. (BF=1) The busy state is 10 ms after Vdd reachs to 4.5V.

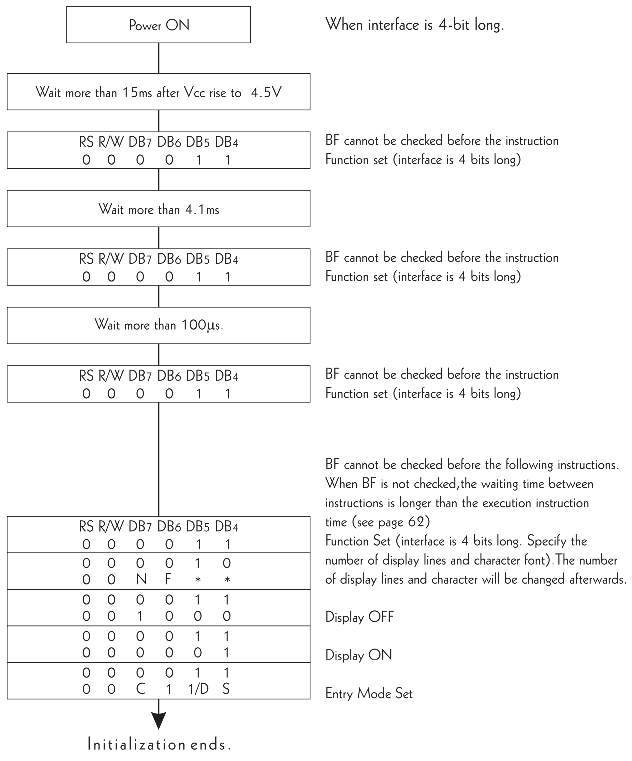
(1) Display clear
(2) Function set
DL= 1:8 bit long interface data
DL= 0:4 bit F= 0:5 x 7dots character font
N= 1:2 lines
N= 0:1 line
(3) Display ON / OFF control
D= 0:Display OFF C= 0:Cursor OFF
B= 0:Blink OFF
(4) Entry mode set
1 / D= 1:+1(increment) S= 0:No shift

Note:When conditions stated in power supply conditions using internal reset circuit are not satisfied.The internal reset circuit will not operate properly and initialization will not be performed. Please make initialization using MPU along with instruction.

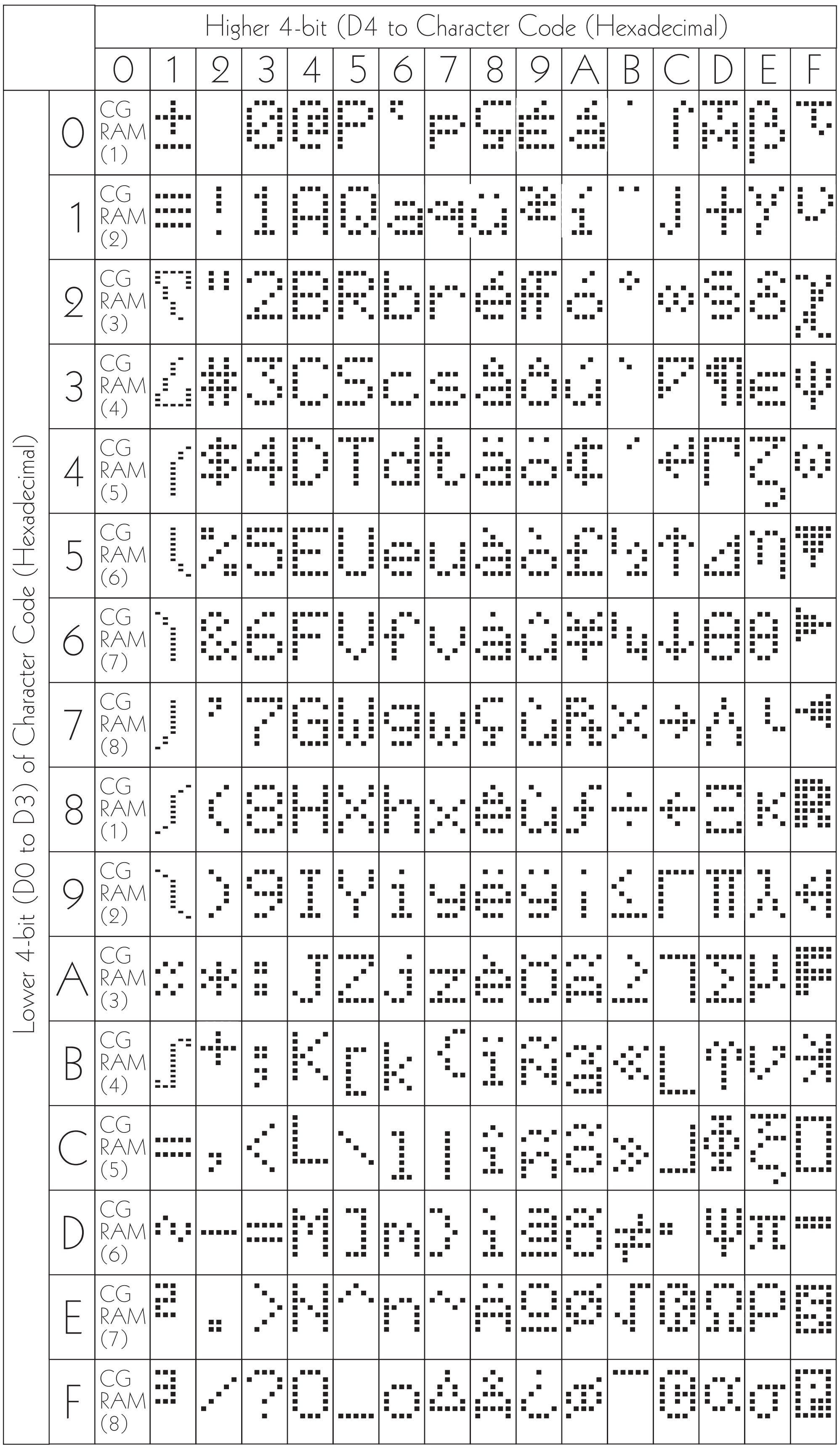
◆ **Initialization along with instruction**

If power supply conditions are not satisfied, which for proper operation of internal rest circuit, it is req-uired to make initialzation along with instruction. Please make following procedures.


Power ON
When interface is 8-bit long.
Wait more than 15ms after Vcc rise to 4.5V
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 1 1 * * * *
BF cannot be checked before the instruction
Function set (interface is 8 bits long)
Wait more than 4.1ms
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 1 1 * * * *
BF cannot be checked before the instruction
Function set (interface is 8 bits long)
Wait more than 100μs
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 1 1 * * * *
BF cannot be checked before the instruction
Function set (interface is 8 bits long)
BF cannot be checked before the following instructions. When BF is not checked.the waiting time between instructions is longer than the execution instruction time
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 1 1 N F * *
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1 1/D S
Function Set (interface is 8 bit long. Specify the number of display lines and character font) The number of display line and character will be changed afterwards.
Display OFF
Display ON
Entry Mode Set
Initialization ends.


Power ON
When interface is 4-bit long.
Wait more than 15ms after Vcc rise to 4.5V
RS R/W DB7 DB6 DB5 DB4
0 0 0 0 1 1
BF cannot be checked before the instruction
Function set (interface is 4 bits long)
Wait more than 4.1ms
RS R/W DB7 DB6 DB5 DB4
0 0 0 0 1 1
BF cannot be checked before the instruction
Function set (interface is 4 bits long)
Wait more than 100μs.
RS R/W DB7 DB6 DB5 DB4
0 0 0 0 1 1
BF cannot be checked before the instruction
Function set (interface is 4 bits long)
BF cannot be checked before the following instructions. When BF is not checked,the waiting time between instructions is longer than the execution instruction time (see page 62)
RS R/W DB7 DB6 DB5 DB4
0 0 0 0 1 1
0 0 0 0 1 0
0 0 N F * *
0 0 0 0 1 1
0 0 1 0 0 0
0 0 0 0 1 1
0 0 0 0 0 1
0 0 0 0 1 1
0 0 C 1 1/D S
Function Set (interface is 4 bits long. Specify the number of display lines and character font).The number of display lines and character will be changed afterwards.
Display OFF
Display ON
Entry Mode Set
Initialization ends.

Standard Character Pattern (Powertip Module)


Higher 4-bit (D4 to Character Code (Hexadecimal)
0 1 2 3 4 5 6 7 8 9 A B C D E F
Lower 4-bit (D0 to D3) of Character Code (Hexadecimal)
CG RAM (1)
CG RAM (2)
CG RAM (3)
CG RAM (4)
CG RAM (5)
CG RAM (6)
CG RAM (7)
CG RAM (8)

Standard Character Pattern (Elec & Eltek Module)

Upper(4bit) / Lower(4bit)	LLLL	LLHL	LLHH	LHLL	LHLH	LHHL	LHHH	HLLL	HLLH	HLHL	HLHH	HHLL	HHLH	HHHL	HHHH
LLLL	CG RAM (1)		0	@	P	`	p				ー	タ	ミ	α	p
LLLH	(2)	!	1	A	Q	a	q			。	ア	チ	ム	ä	q
LLHL	(3)	"	2	B	R	b	r			「	イ	ツ	メ	β	θ
LLHH	(4)	#	3	C	S	c	s			」	ウ	テ	モ	ε	∞
LHLL	(5)	$	4	D	T	d	t			、	エ	ト	ヤ	μ	Ω
LHLH	(6)	%	5	E	U	e	u			・	オ	ナ	ユ	σ	ü
LHHL	(7)	&	6	F	V	f	v			ヲ	カ	ニ	ヨ	ρ	Σ
LHHH	(8)	'	7	G	W	g	w			ァ	キ	ヌ	ラ	g	π
HLLL	(1)	(	8	H	X	h	x			ィ	ク	ネ	リ	√	x̄
HLLH	(2)	)	9	I	Y	i	y			ゥ	ケ	ノ	ル	⁻¹	y
HLHL	(3)	*	:	J	Z	j	z			ェ	コ	ハ	レ	j	千
HLHH	(4)	+	;	K	[	k	{			ォ	サ	ヒ	ロ	ˣ	万
HHLL	(5)	,	<	L	¥	l	|			ャ	シ	フ	ワ	¢	円
HHLH	(6)	-	=	M	]	m	}			ュ	ス	ヘ	ン	£	÷
HHHL	(7)	.	>	N	^	n	→			ョ	セ	ホ	゛	ñ	
HHHH	(8)	/	?	O	_	o	←			ッ	ソ	マ	゜	ö	█

T-1 (3mm) SUPER BRIGHT LED LAMPS

L-934SRC-x SUPER BRIGHT RED

L-934SRD-x SUPER BRIGHT RED

L-934SGx SUPER BRIGHT GREEN

Features

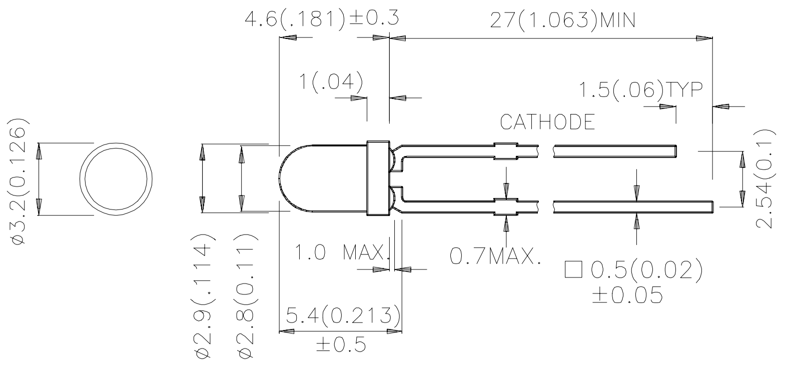
- ●ULTRA BRIGHTNESS.
- ●BOTH DIFFUSED AND WATER CLEAR LENS ARE AVAILABLE.
- ●OUTSTANDING MATERIAL EFFICIENCY.
- ●RELIABLE AND RUGGED.
- ●IC COMPATIBLE/LOW CURRENT CAPABILITY.

Description

The Super Bright Red source color devices are made with Gallium Aluminum Arsenide Red Light Emitting Diode.

The Super Bright Green source color devices are made with Gallium Phosphide Green Light Emitting Diode.

Package Dimensions



Notes:
1. All dimensions are in millimeters (inches).
2. Tolerance is ±0.25(0.01") unless otherwise noted.
3. Lead spacing is measured where the lead emerge package.
4. Specifications are subject to change without notice.

Kingbright

Selection Guide

Part No.	Dice	Lens Type	Iv (mcd) @ 20 mA		Viewing Angle
			Min.	Typ.	2θ1/2
L-934SGC	SUPER BRIGHT GREEN(GaP)	WATER CLEAR	80	150	50°
L-934SGD		GREEN DIFFUSED	20	40	60°
L-934SRC-D	SUPER BRIGHT RED (GaAlAs)	WATER CLEAR	500	600	50°
L-934SRC-E			700	900	
L-934SRC-F			1000	1100	
L-934SRC-G			1200	1300	
L-934SRC-H			1400	1700	
L-934SRC-J			1800	2300	
L-934SRD-D	SUPER BRIGHT RED (GaAlAs)	RED DIFFUSED	100	150	60°
L-934SRD-E			200	250	
L-934SRD-F			300	350	
L-934SRD-G			400	500	
L-934SRD-H			600	900	
L-934SRD-J			1000	1200	

Note:
1. θ1/2 is the angle from optical centerline where the luminous intensity is 1/2 the optical centerline value.

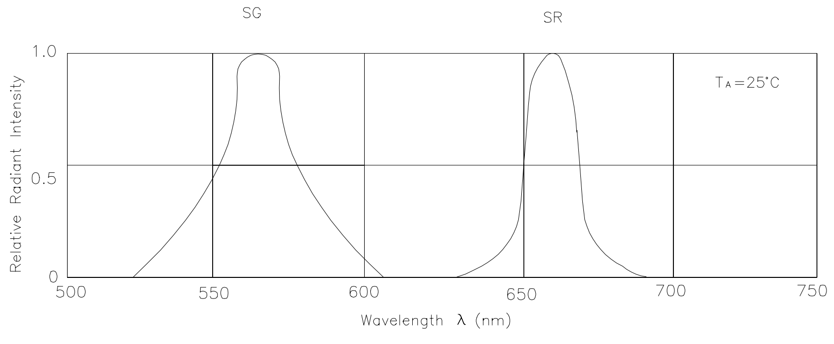
Electrical / Optical Characteristics at T_A=25°C

Symbol	Parameter	Device	Typ.	Max.	Units	Test Conditions
λpeak	Peak Wavelength	Super Bright Red Super Bright Green	660 565		nm	IF=20mA
λ D	Dominate Wavelength	Super Bright Red Super Bright Green	640 568		nm	IF=20mA
Δλ1/2	Spectral Line Halfwidth	Super Bright Red Super Bright Green	20 30		nm	IF=20mA
C	Capacitance	Super Bright Red Super Bright Green	45 15		pF	VF=0V;f=1MHz
V_F	Forward Voltage	Super Bright Red Super Bright Green	1.85 2.2	2.5 2.5	V	IF=20mA
I_R	Reverse Current	All		10	uA	VR = 5V

Absolute Maximum Ratings at T_A=25°C

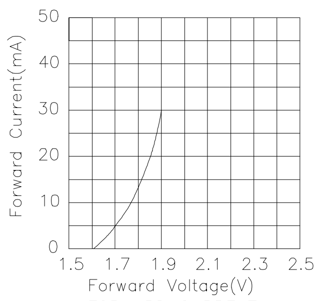
Parameter	Super Bright Red	Super Bright Green	Units
Power dissipation	100	105	mW
DC Forward Current	30	25	mA
Peak Forward Current [1]	155	140	mA
Reverse Voltage	5	5	V
Operating/Storage Temperature	-40°C To +85°C		
Lead Solder Temperature [2]	260°C For 5 Seconds		

Notes:
1. 1/10 Duty Cycle, 0.1ms Pulse Width.
2. 4mm below package base.

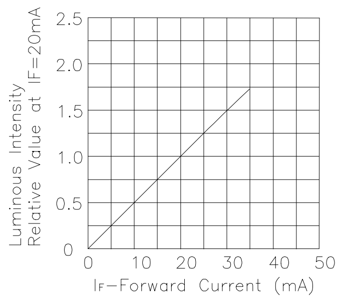


RELATIVE INTENSITY Vs. WAVELENGTH

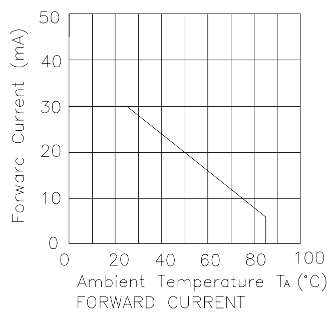
Super Bright Red L-934SRC-x,L-934SRD-x



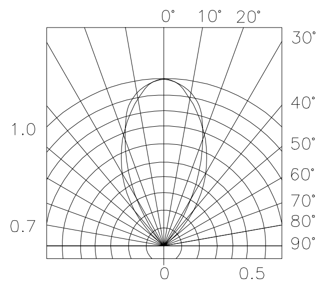
FORWARD CURRENT Vs FORWARD VOLTAGE



LUMINOUS INTENSITY Vs. FORWARD CURRENT

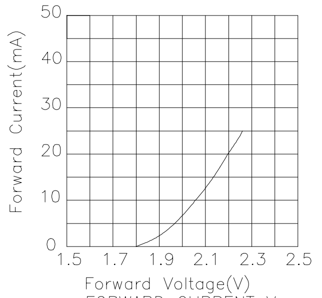


FORWARD CURRENT DERATING CURVE

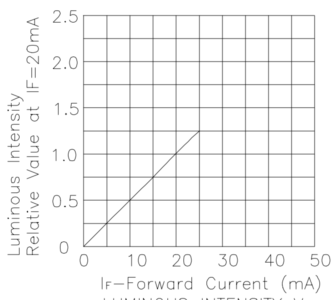


SPATIAL DISTRIBUTION

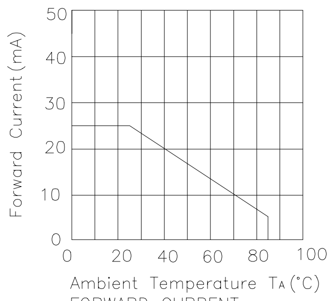
Super Bright Green L-934SGC,L-934SGD



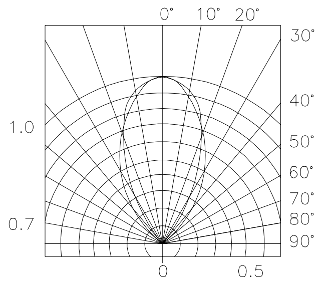
FORWARD CURRENT Vs
FORWARD VOLTAGE



LUMINOUS INTENSITY Vs.
FORWARD CURRENT



FORWARD CURRENT
DERATING CURVE



SPATIAL DISTRIBUTION

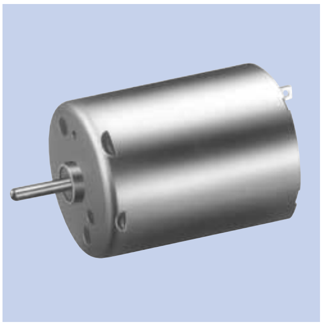
MABUCHI MOTOR

RF-370CA

Jameco Part Number 238473

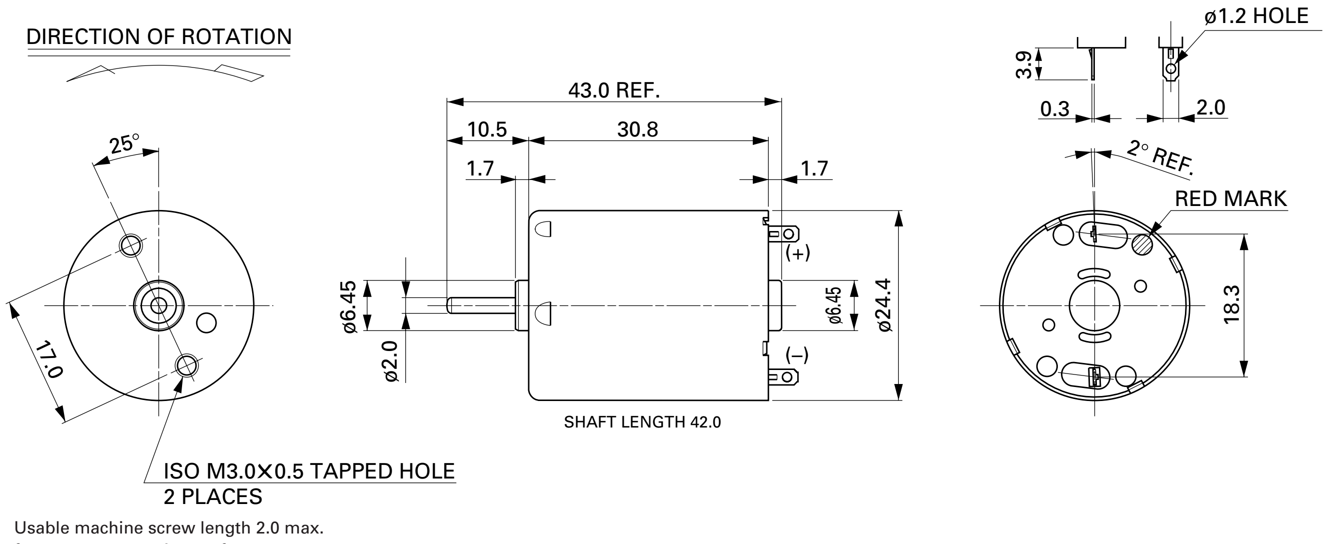
OUTPUT: APPROX 0.6W~2.0W *Precious metal-brush motors*

Typical Applications **Audio and Visual Equipment** : CD Player / DVD Player / VCR
Home Appliances : Kitchen Appliance



MODEL	VOLTAGE		NO LOAD		AT MAXIMUM EFFICIENCY					STALL		
	OPERATING RANGE	NOMINAL	SPEED	CURRENT	SPEED	CURRENT	TORQUE		OUTPUT	TORQUE		CURRENT
			r/min	A	r/min	A	mN·m	g·cm	W	mN·m	g·cm	A
RF-370CA-15370	3 ~ 12	12V CONSTANT	5600	0.026	4840	0.17	2.48	25.3	1.25	18.3	187	1.06
RF-370CA-12560	4 ~ 12	8V CONSTANT	2400	0.015	1970	0.069	1.57	16.0	0.32	8.82	90	0.32

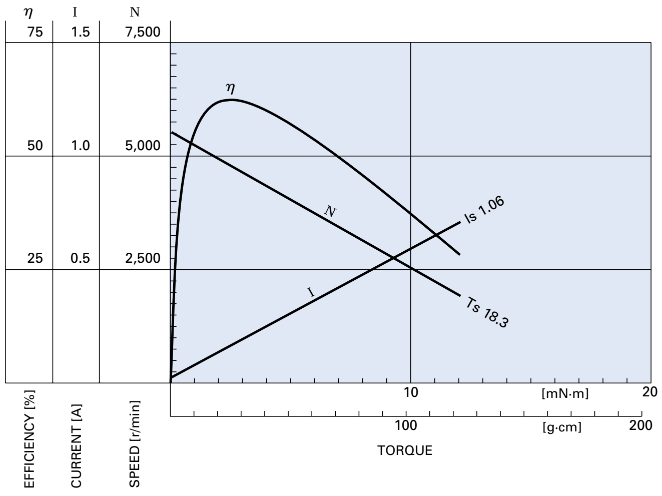
UNIT: MILLIMETERS



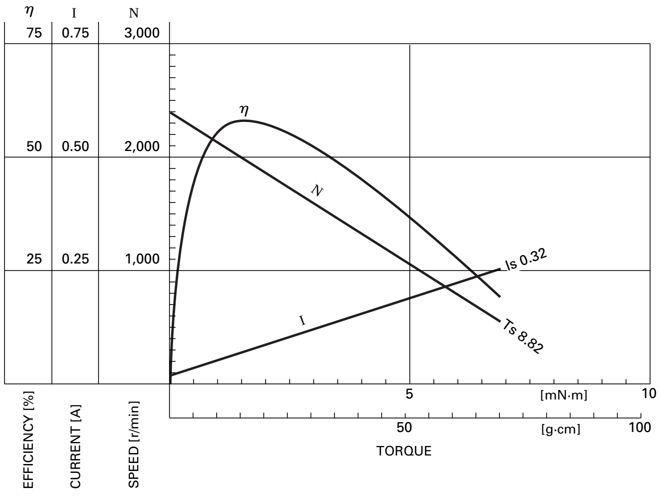
Usable machine screw length 2.0 max.
from motor mounting surface.

WEIGHT: 51g (APPROX)

RF-370CA-15370 12.0V



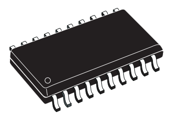
RF-370CA-12560 8.0V

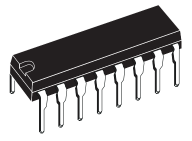


L293D
L293DD

PUSH-PULL FOUR CHANNEL DRIVER WITH DIODES

- 600mA OUTPUT CURRENT CAPABILITY PER CHANNEL
- 1.2A PEAK OUTPUT CURRENT (non repetitive) PER CHANNEL
- ENABLE FACILITY
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)
- INTERNAL CLAMP DIODES


SO(12+4+4)


Powerdip (12+2+2)

ORDERING NUMBERS:

L293DD L293D

DESCRIPTION

The Device is a monolithic integrated high voltage, high current four channel driver designed to accept standard DTL or TTL logic levels and drive inductive loads (such as relays solenoides, DC and stepping motors) and switching power transistors.

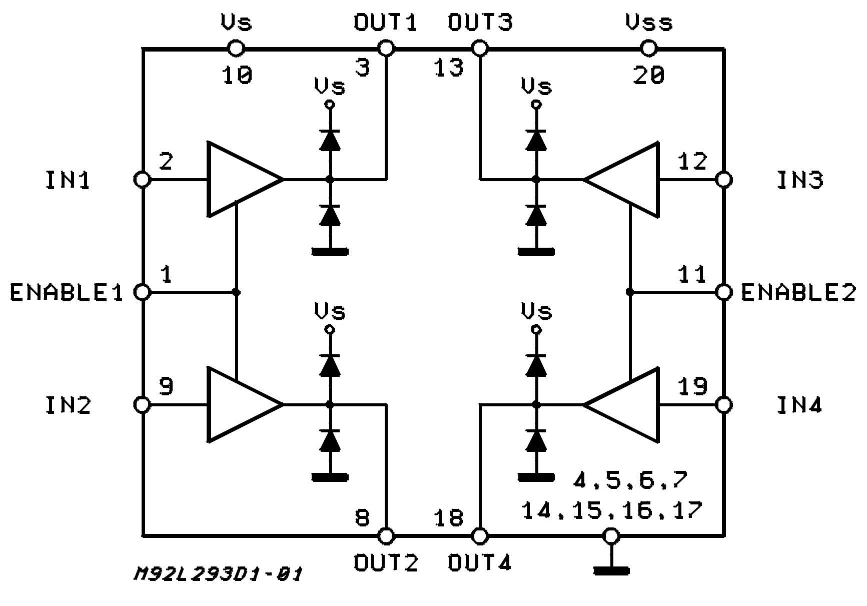
To simplify use as two bridges each pair of channels is equipped with an enable input. A separate supply input is provided for the logic, allowing operation at a lower voltage and internal clamp diodes are included.

This device is suitable for use in switching applications at frequencies up to 5 kHz.

The L293D is assembled in a 16 lead plastic packaage which has 4 center pins connected together and used for heatsinking

The L293DD is assembled in a 20 lead surface mount which has 8 center pins connected together and used for heatsinking.

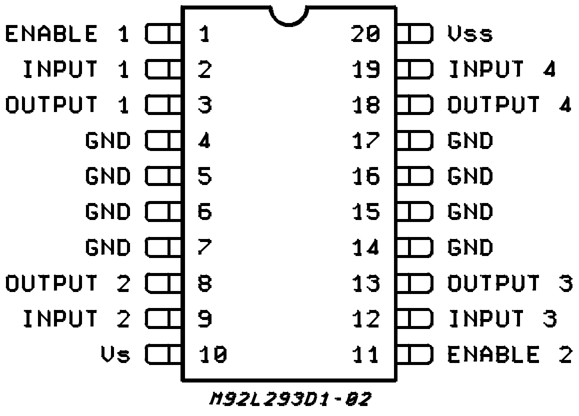
BLOCK DIAGRAM



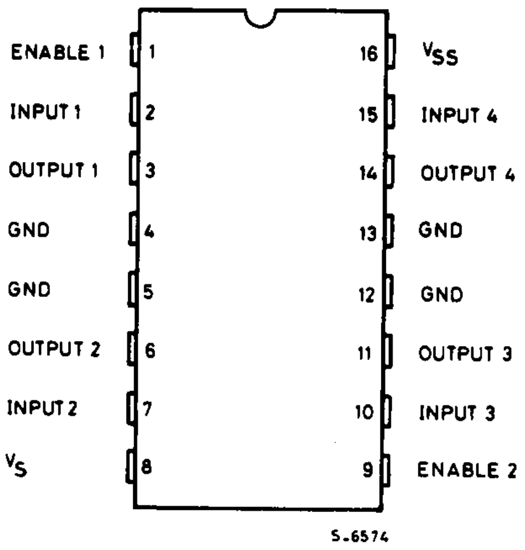
ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Supply Voltage	36	V
V_{SS}	Logic Supply Voltage	36	V
V_i	Input Voltage	7	V
V_{en}	Enable Voltage	7	V
I_o	Peak Output Current (100 µs non repetitive)	1.2	A
P_{tot}	Total Power Dissipation at T_{pins} = 90 °C	4	W
T_{stg}, T_j	Storage and Junction Temperature	– 40 to 150	°C

PIN CONNECTIONS (Top view)



SO(12+4+4)



Powerdip(12+2+2)

THERMAL DATA

Symbol	Decription		DIP	SO	Unit
$R_{th\ j\text{-}pins}$	Thermal Resistance Junction-pins	max.	–	14	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance junction-ambient	max.	80	50 (*)	°C/W
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	max.	14	–	

(*) With 6sq. cm on board heatsink.

ELECTRICAL CHARACTERISTICS (for each channel, V_S = 24 V, V_{SS} = 5 V, T_{amb} = 25 °C, unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 10)		V_{SS}		36	V
V_{SS}	Logic Supply Voltage (pin 20)		4.5		36	V
I_S	Total Quiescent Supply Current (pin 10)	V_i = L ; I_O = 0 ; V_{en} = H		2	6	mA
		V_i = H ; I_O = 0 ; V_{en} = H		16	24	mA
		V_{en} = L			4	mA
I_{SS}	Total Quiescent Logic Supply Current (pin 20)	V_i = L ; I_O = 0 ; V_{en} = H		44	60	mA
		V_i = H ; I_O = 0 ; V_{en} = H		16	22	mA
		V_{en} = L		16	24	mA
V_{IL}	Input Low Voltage (pin 2, 9, 12, 19)		– 0.3		1.5	V
V_{IH}	Input High Voltage (pin 2, 9, 12, 19)	$V_{SS} \leq 7$ V	2.3		V_{SS}	V
		$V_{SS} > 7$ V	2.3		7	V
I_{IL}	Low Voltage Input Current (pin 2, 9, 12, 19)	V_{IL} = 1.5 V			– 10	µA
I_{IH}	High Voltage Input Current (pin 2, 9, 12, 19)	2.3 V $\leq V_{IH} \leq V_{SS}$ – 0.6 V		30	100	µA
$V_{en\ L}$	Enable Low Voltage (pin 1, 11)		– 0.3		1.5	V
$V_{en\ H}$	Enable High Voltage (pin 1, 11)	$V_{SS} \leq 7$ V	2.3		V_{SS}	V
		$V_{SS} > 7$ V	2.3		7	V
$I_{en\ L}$	Low Voltage Enable Current (pin 1, 11)	$V_{en\ L}$ = 1.5 V		– 30	– 100	µA
$I_{en\ H}$	High Voltage Enable Current (pin 1, 11)	2.3 V $\leq V_{en\ H} \leq V_{SS}$ – 0.6 V			± 10	µA
$V_{CE(sat)H}$	Source Output Saturation Voltage (pins 3, 8, 13, 18)	I_O = – 0.6 A		1.4	1.8	V
$V_{CE(sat)L}$	Sink Output Saturation Voltage (pins 3, 8, 13, 18)	I_O = + 0.6 A		1.2	1.8	V
V_F	Clamp Diode Forward Voltage	I_O = 600nA		1.3		V
t_r	Rise Time (*)	0.1 to 0.9 V_O		250		ns
t_f	Fall Time (*)	0.9 to 0.1 V_O		250		ns
t_{on}	Turn-on Delay (*)	0.5 V_i to 0.5 V_O		750		ns
t_{off}	Turn-off Delay (*)	0.5 V_i to 0.5 V_O		200		ns

(*) See fig. 1.

TRUTH TABLE (one channel)

Input	Enable (*)	Output
H	H	H
L	H	L
H	L	Z
L	L	Z

Z = High output impedance
(*) Relative to the considered channel

Figure 1: Switching Times

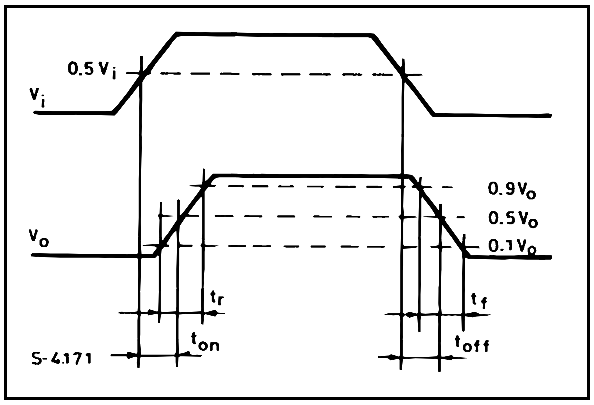
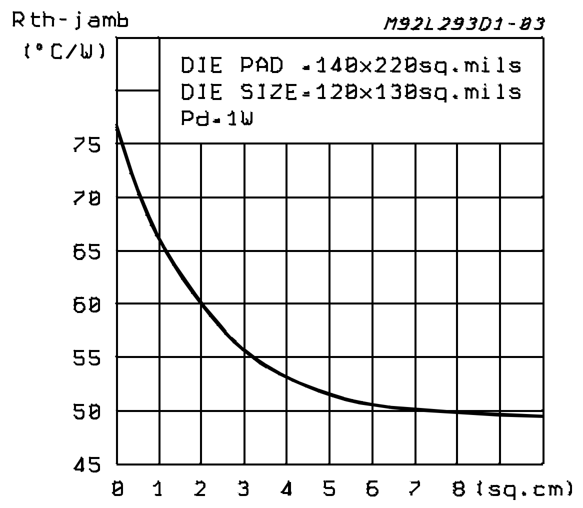
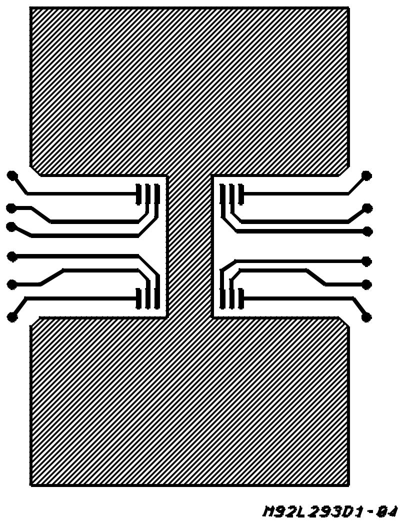


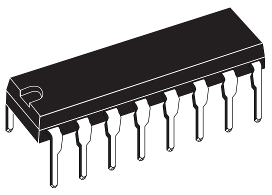
Figure 2: Junction to ambient thermal resistance vs. area on board heatsink (SO12+4+4 package)



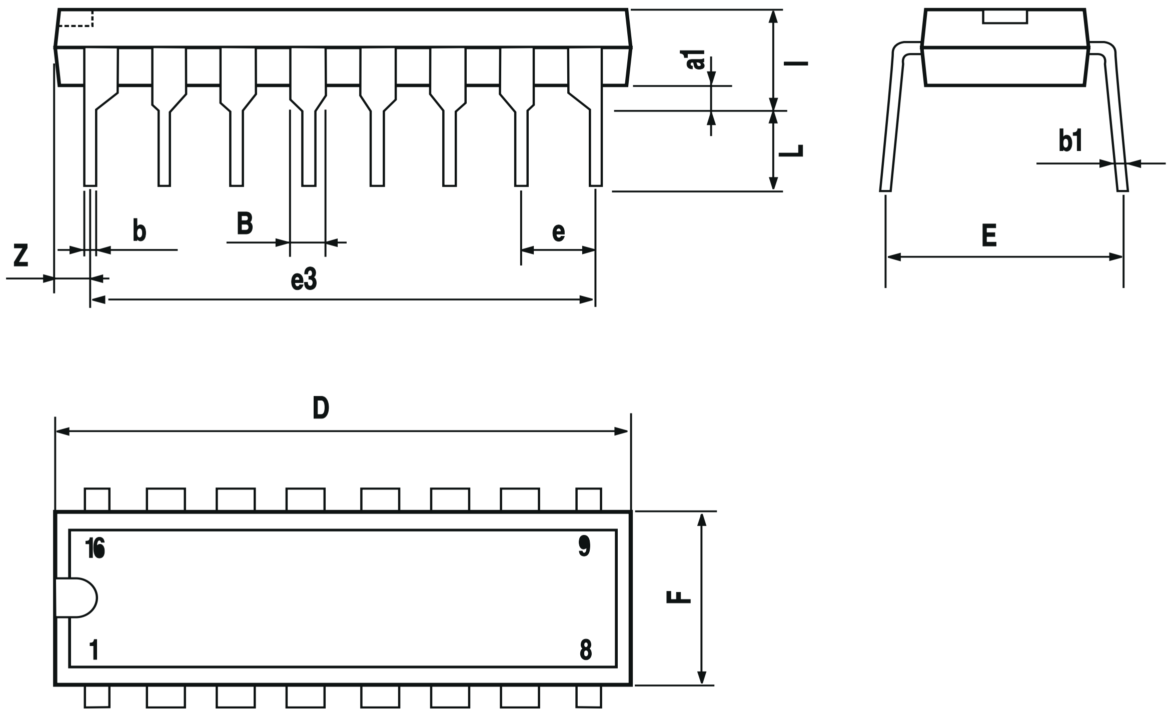


DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
a1	0.51			0.020		
B	0.85		1.40	0.033		0.055
b		0.50			0.020	
b1	0.38		0.50	0.015		0.020
D			20.0			0.787
E		8.80			0.346	
e		2.54			0.100	
e3		17.78			0.700	
F			7.10			0.280
I			5.10			0.201
L			3.30		0.130	
Z			1.27			0.050

OUTLINE AND MECHANICAL DATA

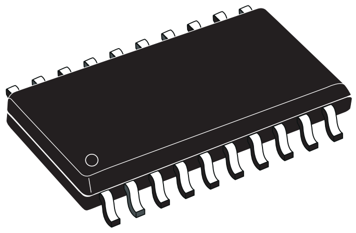


Powerdip 16

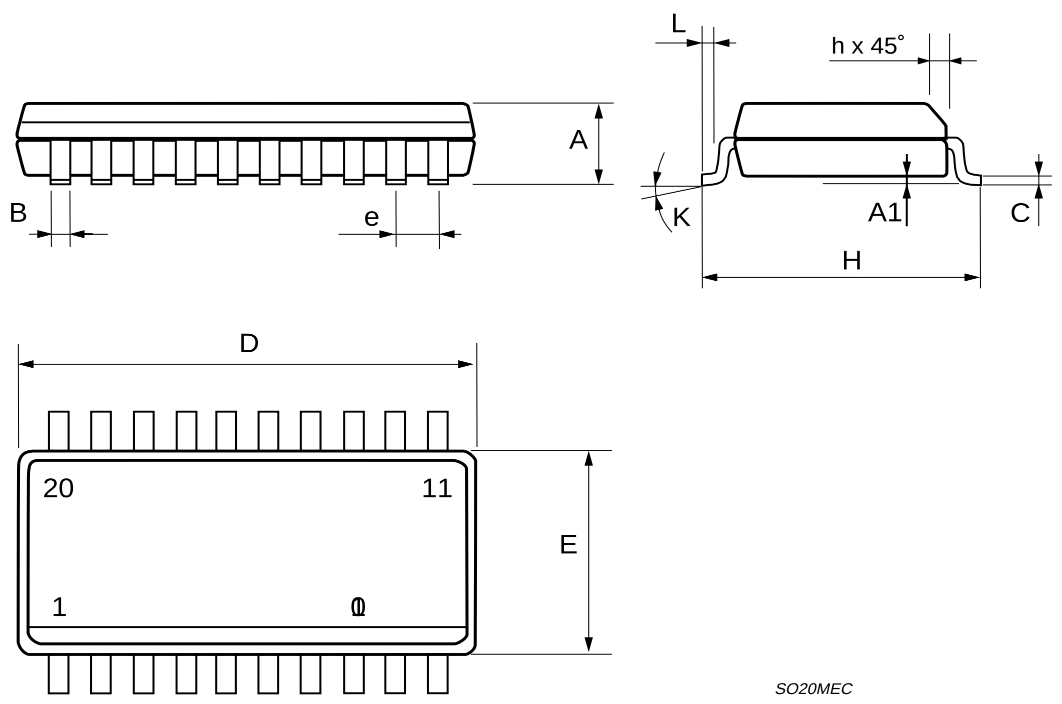


<table>
<tr><th rowspan="2">DIM.</th><th colspan="3">mm</th><th colspan="3">inch</th></tr>
<tr><th>MIN.</th><th>TYP.</th><th>MAX.</th><th>MIN.</th><th>TYP.</th><th>MAX.</th></tr>
<tr><td>A</td><td>2.35</td><td></td><td>2.65</td><td>0.093</td><td></td><td>0.104</td></tr>
<tr><td>A1</td><td>0.1</td><td></td><td>0.3</td><td>0.004</td><td></td><td>0.012</td></tr>
<tr><td>B</td><td>0.33</td><td></td><td>0.51</td><td>0.013</td><td></td><td>0.020</td></tr>
<tr><td>C</td><td>0.23</td><td></td><td>0.32</td><td>0.009</td><td></td><td>0.013</td></tr>
<tr><td>D</td><td>12.6</td><td></td><td>13</td><td>0.496</td><td></td><td>0.512</td></tr>
<tr><td>E</td><td>7.4</td><td></td><td>7.6</td><td>0.291</td><td></td><td>0.299</td></tr>
<tr><td>e</td><td></td><td>1.27</td><td></td><td></td><td>0.050</td><td></td></tr>
<tr><td>H</td><td>10</td><td></td><td>10.65</td><td>0.394</td><td></td><td>0.419</td></tr>
<tr><td>h</td><td>0.25</td><td></td><td>0.75</td><td>0.010</td><td></td><td>0.030</td></tr>
<tr><td>L</td><td>0.4</td><td></td><td>1.27</td><td>0.016</td><td></td><td>0.050</td></tr>
<tr><td>K</td><td colspan="6">0° (min.)8° (max.)</td></tr>
</table>

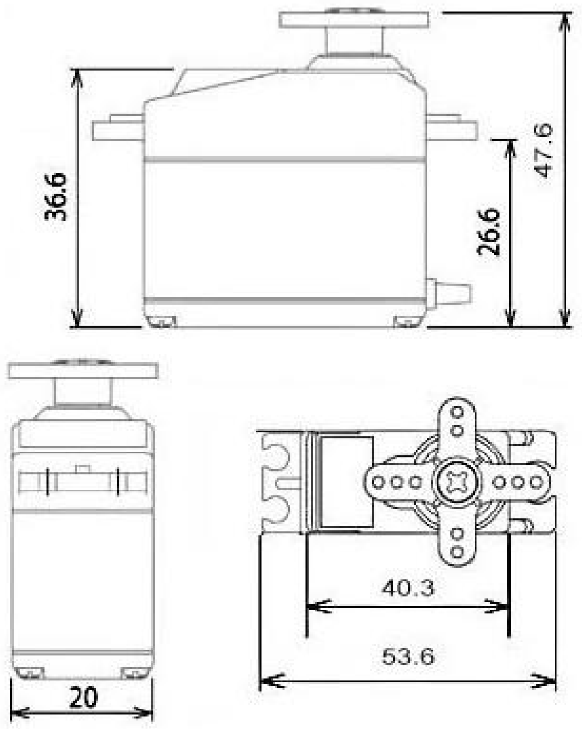
OUTLINE AND MECHANICAL DATA



SO20



MG996R High Torque Metal Gear Dual Ball Bearing Servo



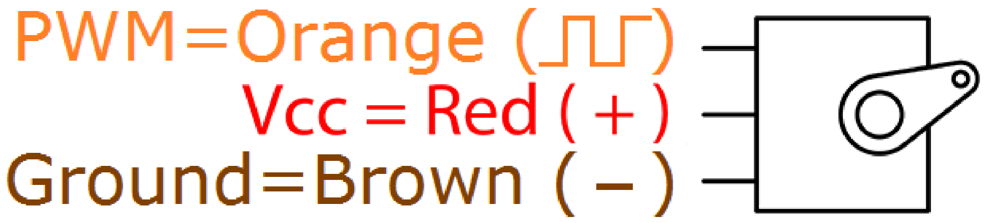
This High-Torque MG996R Digital Servo features metal gearing resulting in extra high 10kg stalling torque in a tiny package. The MG996R is essentially an upgraded version of the famous MG995 servo, and features upgraded shock-proofing and a redesigned PCB and IC control system that make it much more accurate than its predecessor. The gearing and motor have also been upgraded to improve dead bandwith and centering. The unit comes complete with 30cm wire and 3 pin 'S' type female header connector that fits most receivers, including Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum and Hitec.

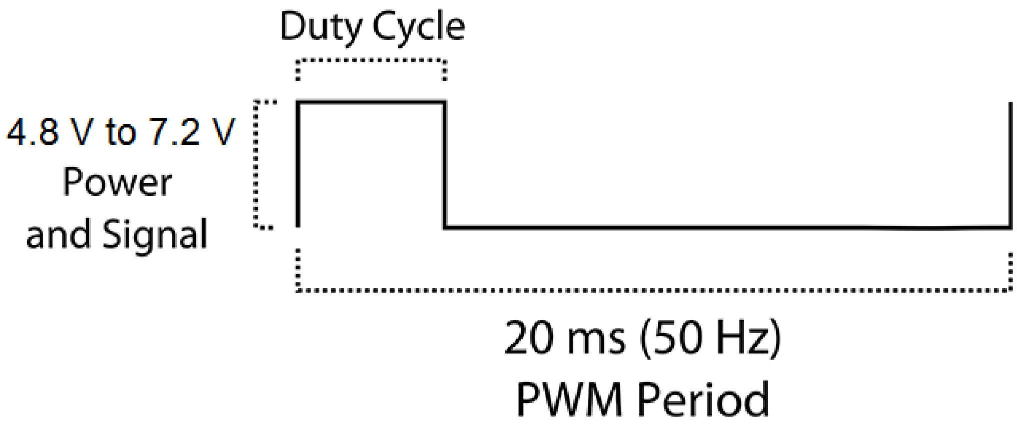
This high-torque standard servo can rotate approximately 120 degrees (60 in each direction). You can use any servo code, hardware or library to control these servos, so it's great for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. The MG996R Metal Gear Servo also comes with a selection of arms and hardware to get you set up nice and fast!

Specifications

- Weight: 55 g
- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 9.4 kgf·cm (4.8 V), 11 kgf·cm (6 V)
- Operating speed: 0.17 s/60º (4.8 V), 0.14 s/60º (6 V)

- Operating voltage: 4.8 V a 7.2 V
- Running Current 500 mA – 900 mA (6V)
- Stall Current 2.5 A (6V)
- Dead band width: 5 µs
- Stable and shock proof double ball bearing design
- Temperature range: 0 ºC – 55 ºC





Technical Data Sheet
3mm Silicon PIN Photodiode T-1

PD204-6C/L3

Features

- Fast response time
- High photo sensitivity
- Small junction capacitance
- Pb free

Descriptions

PD204-6C/L3 is a high speed and high sensitive PIN photodiode in a standard 3Φplastic package. Due to its water clear epoxy the device is sensitive to visible and infrared radiation.

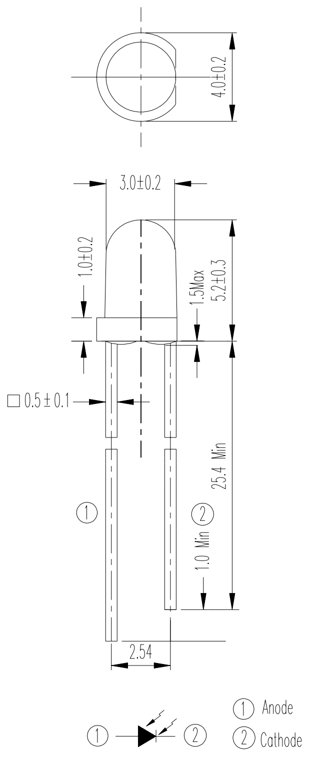
Applications

- Automatic door sensor
- Camera
- Game machine
- High speed photo detector

Device Selection Guide

LED Part No.	Chip Material	Lens Color
PD	Silicon	Water clear

Package Dimensions



Notes: 1.All dimensions are in millimeters
2.Tolerances unless dimensions ±0.25mm

Absolute Maximum Ratings (Ta=25℃)

Parameter	Symbol	Rating	Units
Reverse Voltage	V_R	32	V
Operating Temperature	T_{opr}	-25 ~ +85	℃
Storage Temperature	T_{stg}	-40 ~ +85	℃
Soldering Temperature	T_{sol}	260	℃
Power Dissipation at(or below) 25℃Free Air Temperature	P_c	150	mW

Notes: *1:Soldering time≦5 seconds.

Electro-Optical Characteristics (Ta=25℃)

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Rang Of Spectral Bandwidth	$\lambda_{0.5}$	---	400	---	1100	nm
Wavelength Of Peak Sensitivity	λ_P	---	---	940	---	nm
Open-Circuit Voltage	V_{OC}	Ee=5mW/cm^2 λp=940nm	---	0.44	---	V
Short- Circuit Current	I_{SC}	Ee=1mW/cm^2 λp=940nm	---	10	---	μA
Reverse Light Current	I_L	Ee=1mW/cm^2 λp=940nm V_R=5V	---	10	---	μA
Reverse Dark Current	I_D	Ee=0mW/cm^2 V_R=10V	---	---	10	nA
Reverse Breakdown Voltage	B_{VR}	Ee=0mW/cm^2 I_R=100μA	32	170	---	V
Total Capacitance	C_t	Ee=0mW/cm^2 V_R=5V f=1MHz	---	10	---	pF
Rise Time	t_r	V_R=10V R_L=100Ω	---	10	---	nS
Fall Time	t_f		---	10	---	

Typical Electro-Optical Characteristics Curves

Fig.1 Power Dissipation vs.
Ambient Temperature

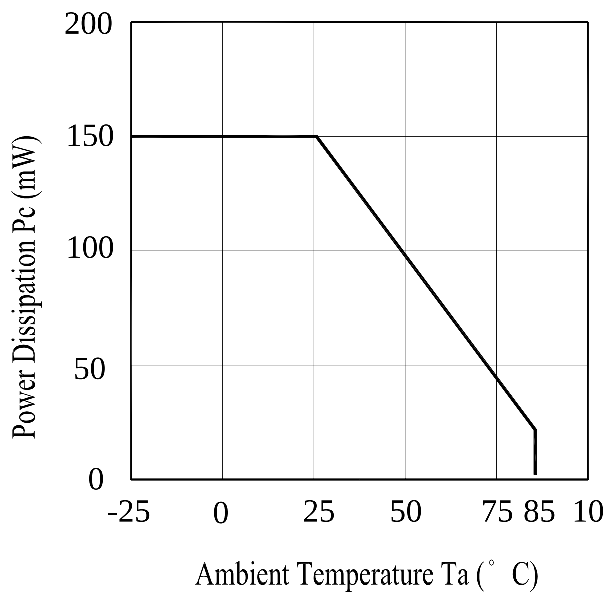


Fig.2 Spectral Sensitivity

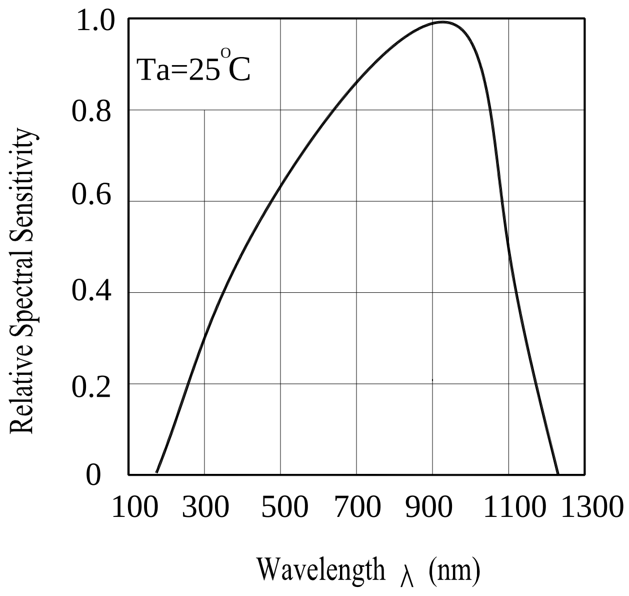


Fig.3 Dark Current vs.
Ambient Temperature

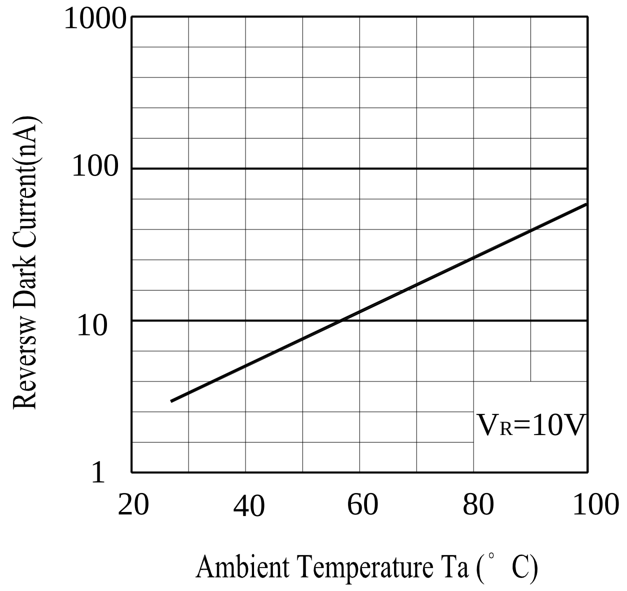
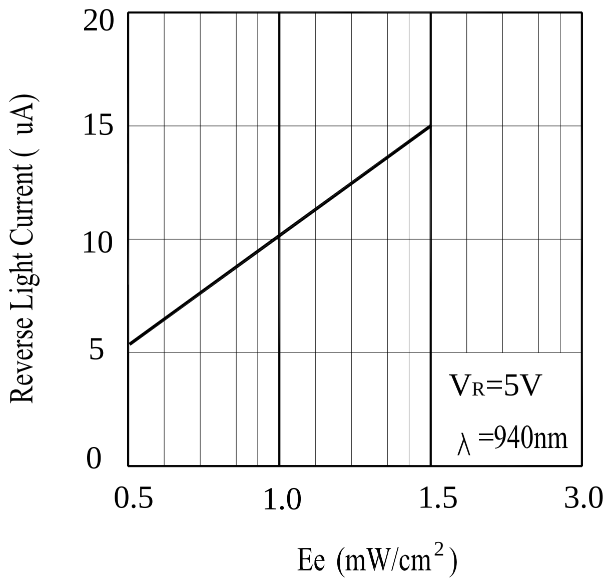


Fig. 4 Reverse Light Current vs.
Ee



Typical Electro-Optical Characteristics Curves

Fig.5 Terminal Capacitance vs.
Reverse Voltage

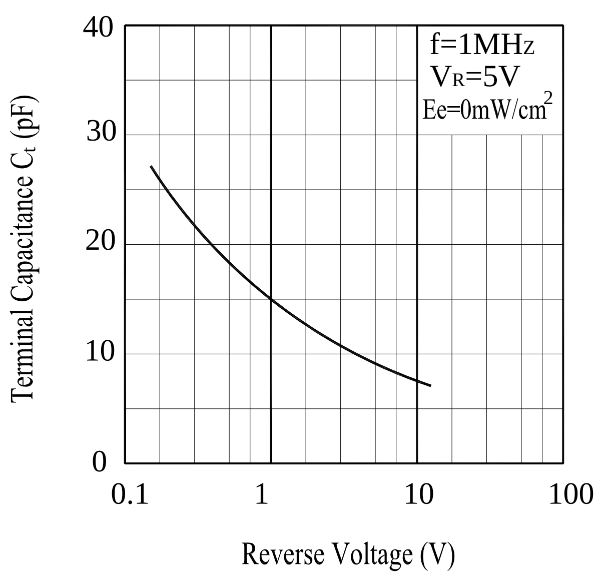
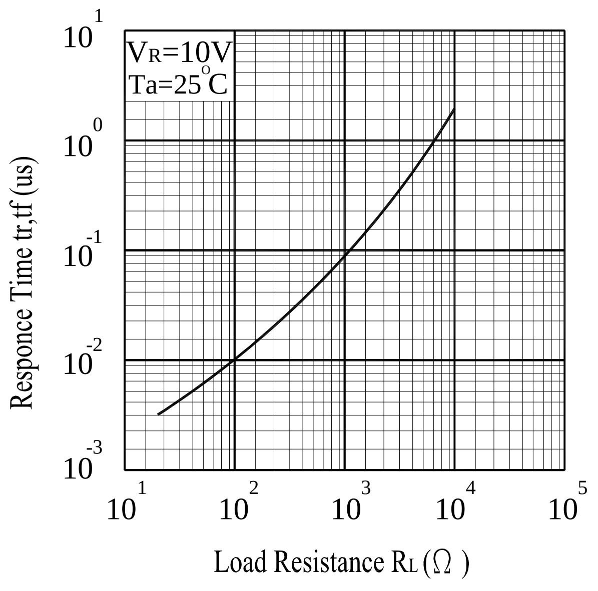


Fig.6 Response Time vs.
Load Resistance



Reliability Test Item And Condition

The reliability of products shall be satisfied with items listed below.

Confidence level：90%

LTPD：10%

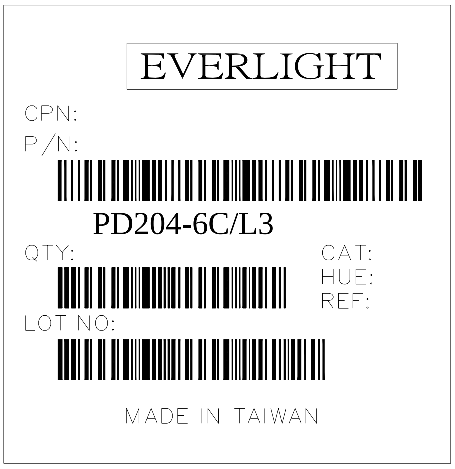
NO.	Item	Test Conditions	Test Hours/ Cycles	Sample Sizes	Failure Judgement Criteria	Ac/Re
1	Solder Heat	TEMP.：260℃±5℃	10secs	22pcs	$I_L \leqq L\times0.8$ L：Lower Specification Limit	0/1
2	Temperature Cycle	H : +100℃ 15mins ↕ 5mins L : -40℃ 15mins	50Cycles	22pcs		0/1
3	Thermal Shock	H :+100℃ 5mins ↕ 10secs L :-10℃ 5mins	50Cycles	22pcs		0/1
4	High Temperature Storage	TEMP.：+100℃	1000hrs	22pcs		0/1
5	Low Temperature Storage	TEMP.：-40℃	1000hrs	22pcs		0/1
6	DC Operating Life	V_R=5V	1000hrs	22pcs		0/1
7	High Temperature/ High Humidity	85℃ / 85% R.H	1000hrs	22pcs		0/1

Packing Quantity Specification

1.1000PCS/1Bag，4Bags/1Box

2.10Boxes/1Carton

Label Form Specification



CPN: Customer's Production Number

P/N : Production Number

QTY: Packing Quantity

CAT: Ranks

HUE: Peak Wavelength

REF: Reference

LOT No: Lot Number

MADE IN TAIWAN: Production Place

Notes

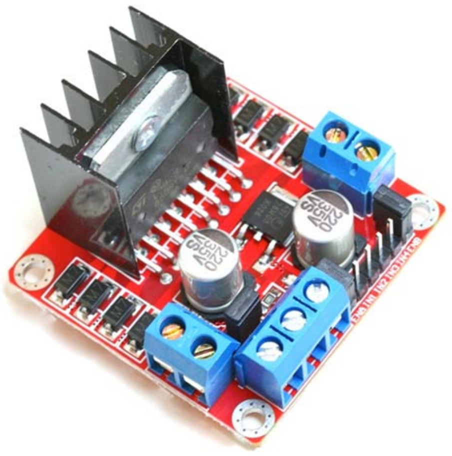
1. Above specification may be changed without notice. EVERLIGHT will reserve authority on material change for above specification.
2. When using this product, please observe the absolute maximum ratings and the instructions for using outlined in these specification sheets. EVERLIGHT assumes no responsibility for any damage resulting from use of the product which does not comply with the absolute maximum ratings and the instructions included in these specification sheets.
3. These specification sheets include materials protected under copyright of EVERLIGHT corporation. Please don't reproduce or cause anyone to reproduce them without EVERLIGHT's consent.

EVERLIGHT ELECTRONICS CO., LTD.
Office: No 25, Lane 76, Sec 3, Chung Yang Rd,
Tucheng, Taipei 236, Taiwan, R.O.C
Tel: 886-2-2267-2000, 2267-9936
Fax: 886-2267-6244, 2267-6189, 2267-6306
http:\\www.everlight.com

User Guide

L298N Dual H-Bridge Motor Driver

This dual bidirectional motor driver, is based on the very popular L298 Dual H-Bridge Motor Driver Integrated Circuit. The circuit will allow you to easily and independently control two motors of up to 2A each in both directions.It is ideal for robotic applications and well suited for connection to a microcontroller requiring just a couple of control lines per motor. It can also be interfaced with simple manual switches, TTL logic gates, relays, etc. This board equipped with power LED indicators, on-board +5V regulator and protection diodes.

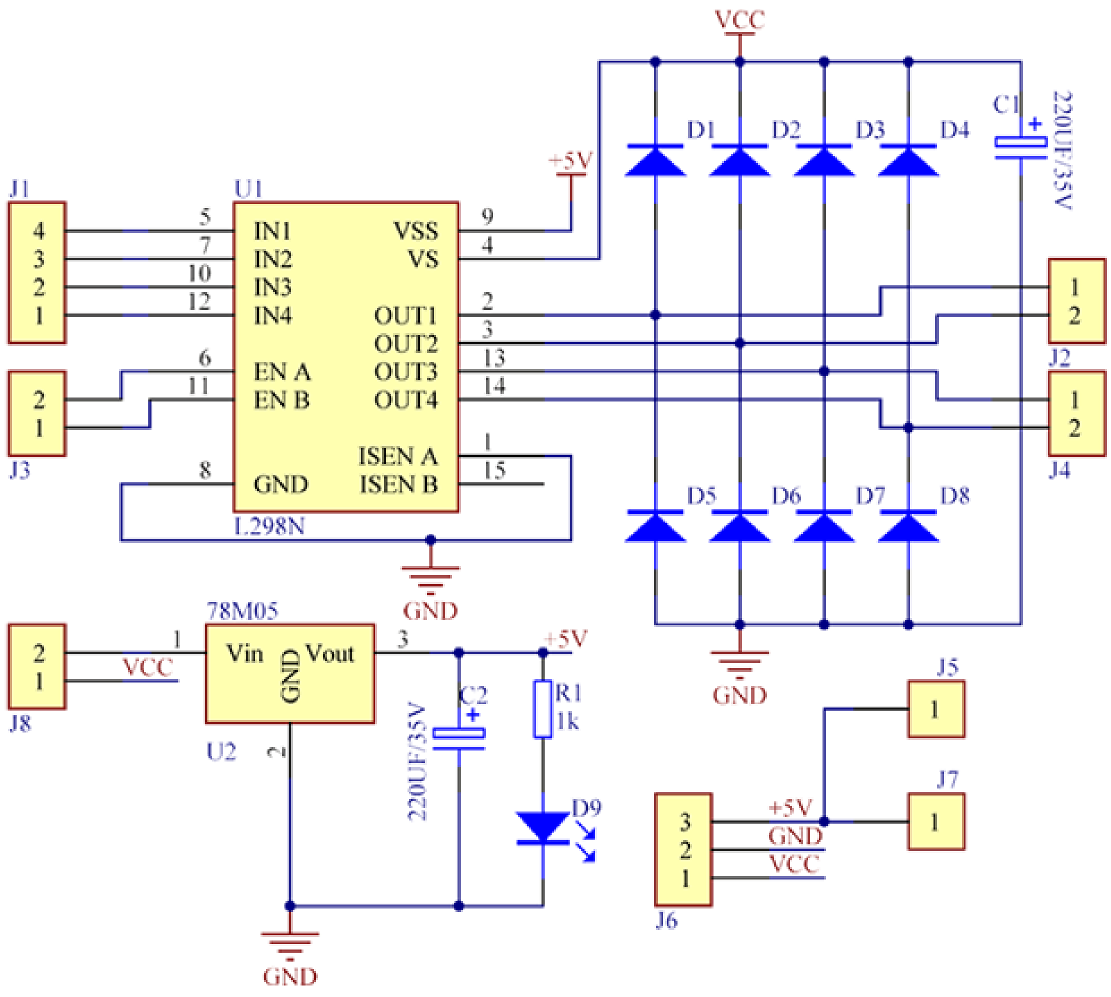


SKU: MDU-1049

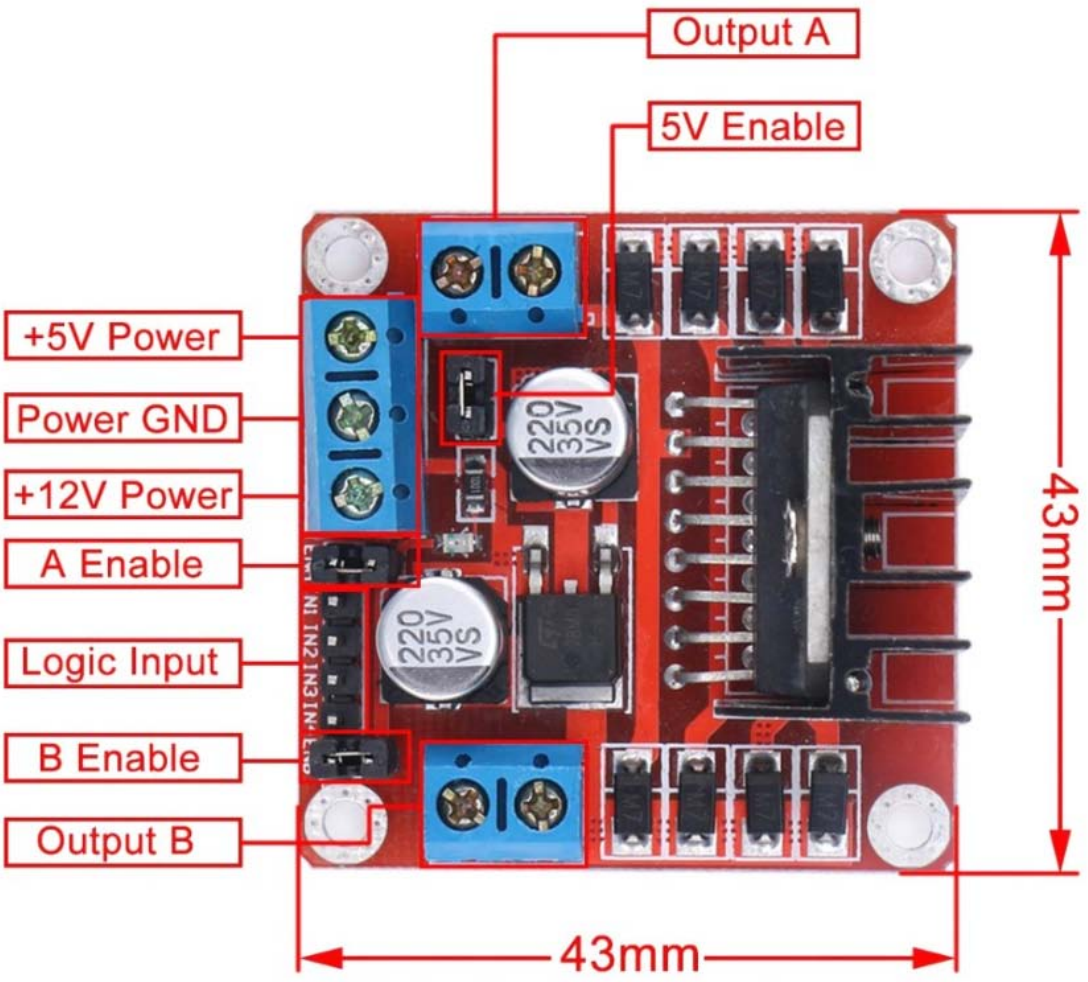
Brief Data:

- Input Voltage: 3.2V~40Vdc.
- Driver: L298N Dual H Bridge DC Motor Driver
- Power Supply: DC 5 V - 35 V
- Peak current: 2 Amp
- Operating current range: 0 ~ 36mA
- Control signal input voltage range :
- Low: -0.3V ≤ Vin ≤ 1.5V.
- High: 2.3V ≤ Vin ≤ Vss.
- Enable signal input voltage range :
  - Low: -0.3 ≤ Vin ≤ 1.5V (control signal is invalid).
  - High: 2.3V ≤ Vin ≤ Vss (control signal active).
- Maximum power consumption: 20W (when the temperature T = 75 ℃).
- Storage temperature: -25 ℃ ~ +130 ℃.
- On-board +5V regulated Output supply (supply to controller board i.e. Arduino).
- Size: 3.4cm x 4.3cm x 2.7cm

Schematic Diagram:



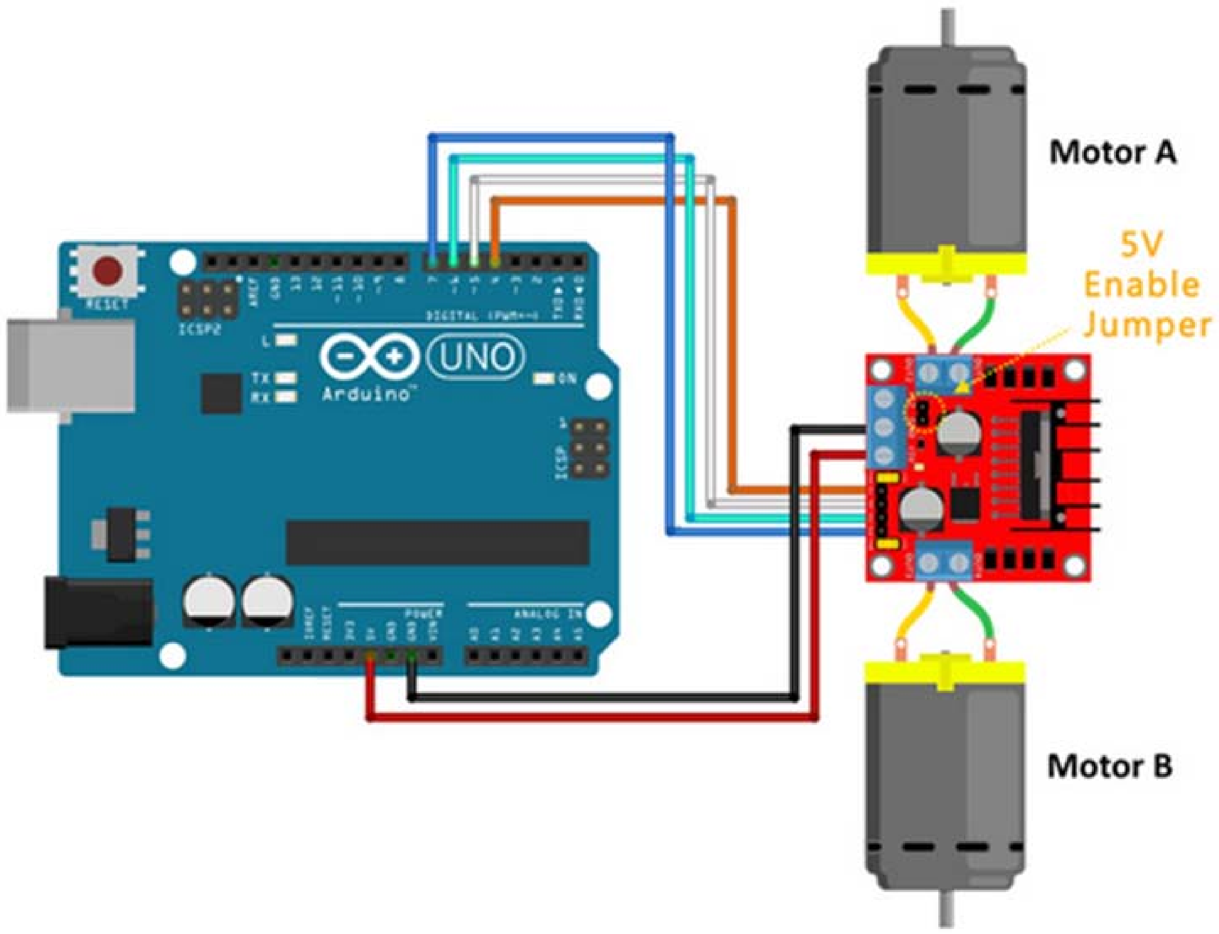
Board Dimension & Pins Function:



Connection Examples:

Controlling 2-DC Motor with +5V Arduino onboard Power Supply:

Below is the circuit connection use the on-board +5V power supply from Arduino board, and should be done without the 5V Enable Jumper on (Active 5V). This connection can drive two 5V DC motors simultaneously.



Sketch Listing:

Copy and paste the sketch below to Arduino IDE and upload to Arduino Uno/Mega board.

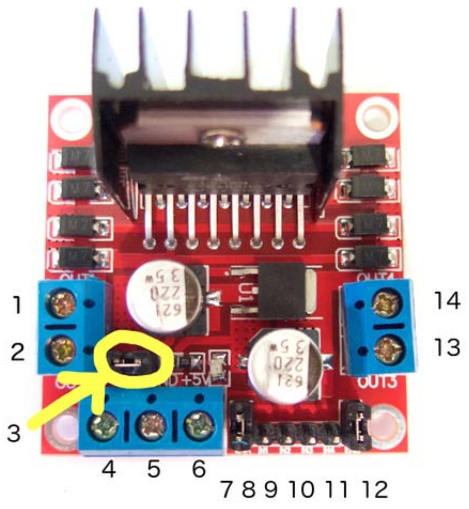
```
/*=====================================================================
//   Author      : Handson Technology
//   Project     : Arduino Uno
//   Description : L298N Motor Driver
//   Source-Code : L298N_Motor.ino
//   Program: Control 2 DC motors using L298N H Bridge Driver
//=====================================================================
*/

// Definitions Arduino pins connected to input H Bridge
int IN1 = 4;
int IN2 = 5;
int IN3 = 6;
int IN4 = 7;

void setup()
{
 // Set the output pins
```

```
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
 }

 void loop()
 {
  // Rotate the Motor A clockwise
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  delay(2000);
  // Motor A
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, HIGH);
  delay(500);

  // Rotate the Motor B clockwise
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  delay(2000);
  // Motor B
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, HIGH);
  delay(500);

  // Rotates the Motor A counter-clockwise
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  delay(2000);
  // Motor A
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, HIGH);
  delay(500);

  // Rotates the Motor B counter-clockwise
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  delay(2000);
  // Motor B
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, HIGH);
  delay(500);
 }
```

Controlling Stepper Motor

In this example we have a typical NEMA-17 stepper motor with four wires:

The key to successful stepper motor control is identifying the wires - that is which one is which. You will need to determine the A+, A-, B+ and B- wires. With our example motor these are red, green, yellow and blue. Now let's get the wiring done.



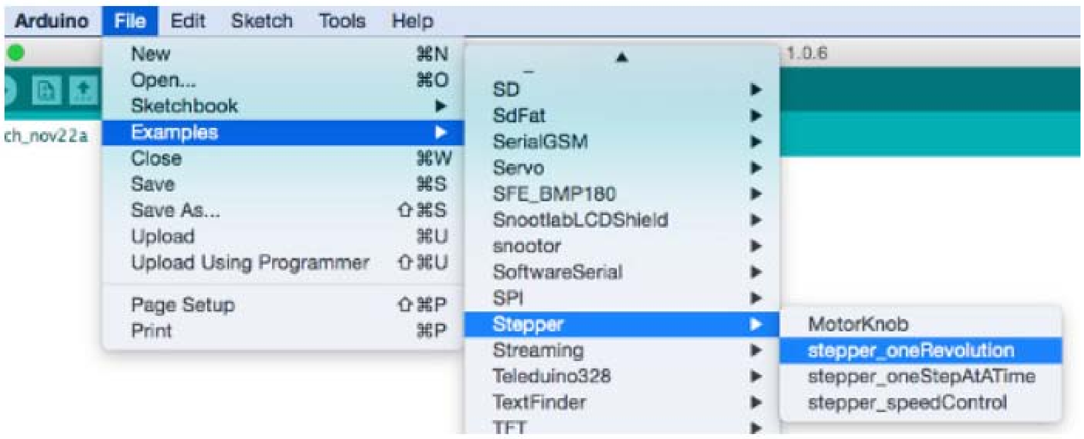
Connect the A+, A-, B+ and B- wires from the stepper motor to the module connections 1, 2, 13 and 14 respectively. Place the jumpers included with the L298N module over the pairs at module points 7 and 12. Then connect the power supply as required to points 4 (positive) and 5 (negative/GND).

Once again if your stepper motor's power supply is less than 12V, fit the jumper to the module at point 3 which gives you a neat 5V power supply for your Arduino.

Next, connect L298N module pins IN1, IN2, IN3 and IN4 to Arduino digital pins D8, D9, D10 and D11 respectively. Finally, connect Arduino GND to point 5 on the module, and Arduino 5V to point 6 if sourcing 5V from the module.

Controlling the stepper motor from your sketches is very simple, thanks to the *Stepper* Arduino library included with the Arduino IDE as standard.

To demonstrate your motor, simply load the *"stepper_oneRevolution"* sketch that is included with the *Stepper* library, for example:



Finally, check the value for

```
        const int stepsPerRevolution = 200;
```

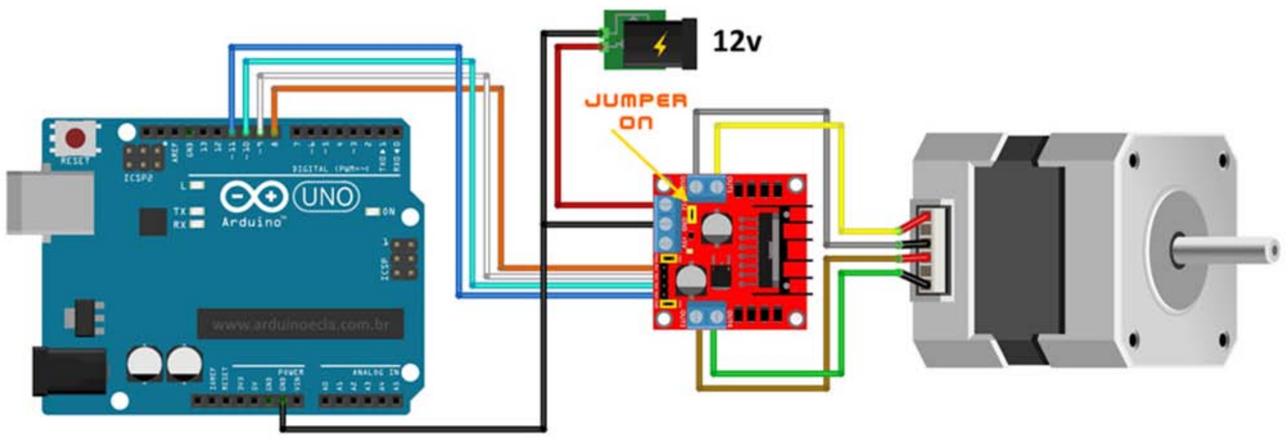
in the sketch and change the 200 to the number of steps per revolution for your stepper motor, and also the speed which is preset to 60 RPM in the following line:

```
        myStepper.setSpeed(60);
```

Now you can save and upload the sketch, which will send your stepper motor around one revolution, then back again. This is achieved with the function

```
        myStepper.step(stepsPerRevolution); // for clockwise
        myStepper.step(-stepsPerRevolution); // for anti-clockwise
```

Connection for the sketch "*stepper_oneRevolution*":



Web Resources:

LSI/CSI

LS7166

LSI Computer Systems, Inc. 1235 Walt Whitman Road, Melville, NY 11747 (631) 271-0400 FAX (631) 271-0405

A3800

December 2002

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 20 MHz Count Frequency.
- 8-Bit I/O Bus for Microprocessor Communication and Control.
- 24-Bit comparator for pre-set count comparison.
- Readable status register.
- Input/Output TTL and CMOS compatible.
- 5 Volt operation (Vdd - Vss).
- LS7166 (DIP); LS7166-S (SOIC); LS7166-TS24 (24-Pin TSSOP)* - See Fig. 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)

20-Pin Package PIN ASSIGNMENT - Top View

*(Contact factory for 24-Pin TSSOP Package Pinout)

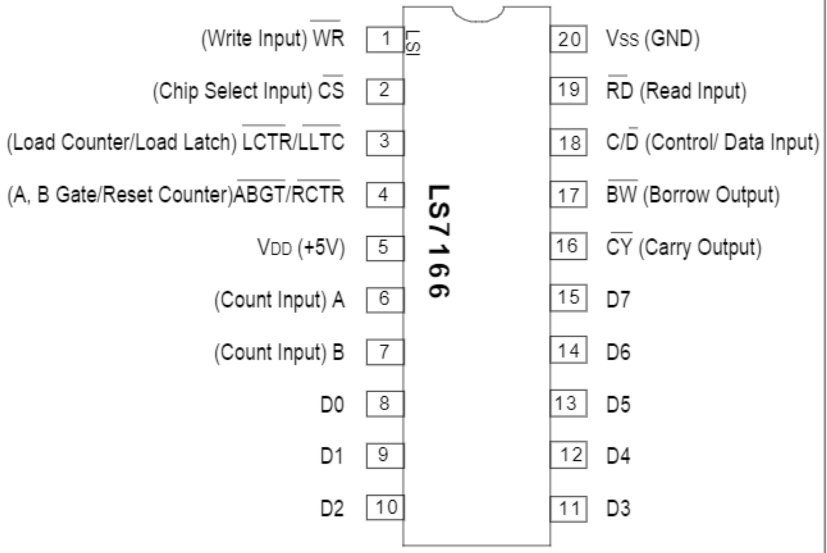

(Write Input) $\overline{WR}$ 1
(Chip Select Input) $\overline{CS}$ 2
(Load Counter/Load Latch) $\overline{LCTR}$/$\overline{LLTC}$ 3
(A, B Gate/Reset Counter) $\overline{ABGT}$/$\overline{RCTR}$ 4
VDD (+5V) 5
(Count Input) A 6
(Count Input) B 7
D0 8
D1 9
D2 10
20 Vss (GND)
19 $\overline{RD}$ (Read Input)
18 C/$\overline{D}$ (Control/ Data Input)
17 $\overline{BW}$ (Borrow Output)
16 $\overline{CY}$ (Carry Output)
15 D7
14 D6
13 D5
12 D4
11 D3
LSI
LS7166

FIGURE 1

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.

Bit #	7	6	5	4	3	2	1	0
	U	U	U	0/1	0/1	0/1	0/1	0/1

- BWT. Borrow Toggle Flip-Flop. Toggles everytime CNTR underflows generating a borrow.
- CYT. Carry Toggle Flip-Flop. Toggles everytime CNTR overflows generating a carry.
- COMPT. Compare Toggle Flip-Flop. Toggles everytime CNTR equals PR
- SIGN. Sign bit. Reset (= 0) when CNTR underflows
  Set (= 1) when CNTR overflows
- UP/DOWN. Count direction indicatior in quadrature mode.
  Reset (= 0) when counting down
  Set (= 1) when counting up
  (Forced to 1 in non-quadrature mode)

U = Undefined

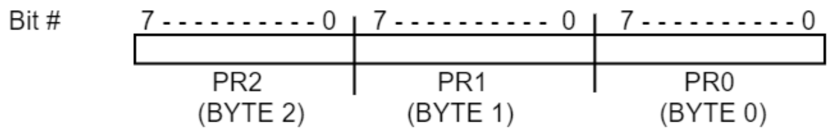
OL(Output latch). The OL is the output port for the CNTR. The 24 bit CNTR Value at any instant can be accessed by performing a CNTR to OL transfer and then reading the OL in 3 READ cycle sequence of Byte 0 (OL0), Byte 1 (OL1) and Byte 2 (OL2). The address pointer for OL0/OL1/OL2 is automatically incremented with each READ cycle.
Accessed by**:** READ when C/$\overline{D}$ = 0, $\overline{CS}$ = 0.

Bit #	7 — — 0	7 — — 0	7 — — 0
	OL2 (BYTE 2)	OL1 (BYTE 1)	OL0 (BYTE 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
```

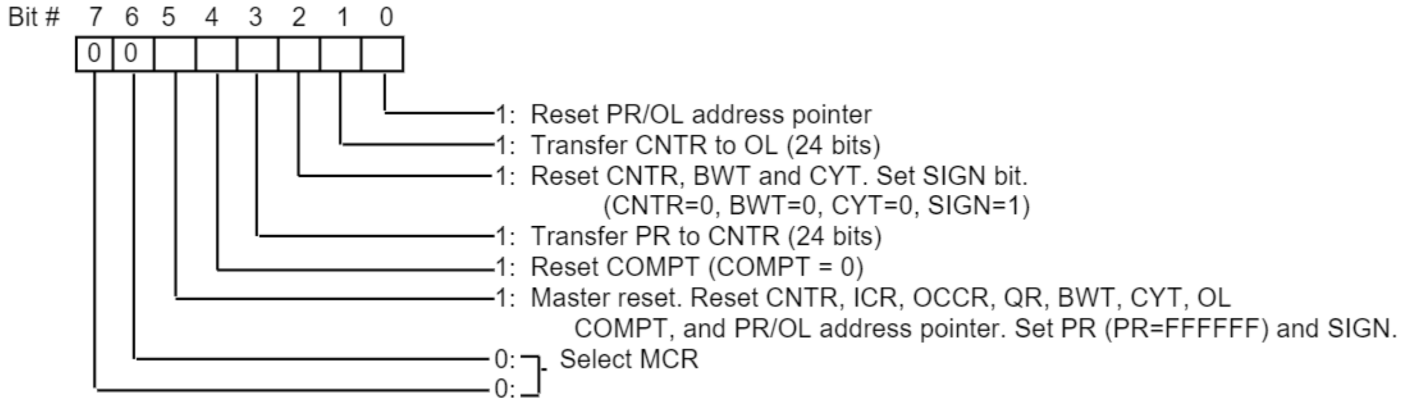
PR (Preset register). The PR is the input port for the CNTR. The CNTR is loaded with a 24 bit data via the PR. The data is first written into the PR in 3 WRITE cycle sequence of Byte 0 (PR0), Byte 1 (PR1) and Byte 2 (PR2).
The address pointer for PR0/PR1/PR2 is automatically incremented with each write cycle.
Accessed by**:** WRITE when $C/\overline{D} = 0$, $\overline{CS} = 0$.



Standard Sequence for Loading PR and Reading CNTR**:**

$1 \longrightarrow$ 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
$8 \longrightarrow$ MCR	;	Transfer PR to CNTR

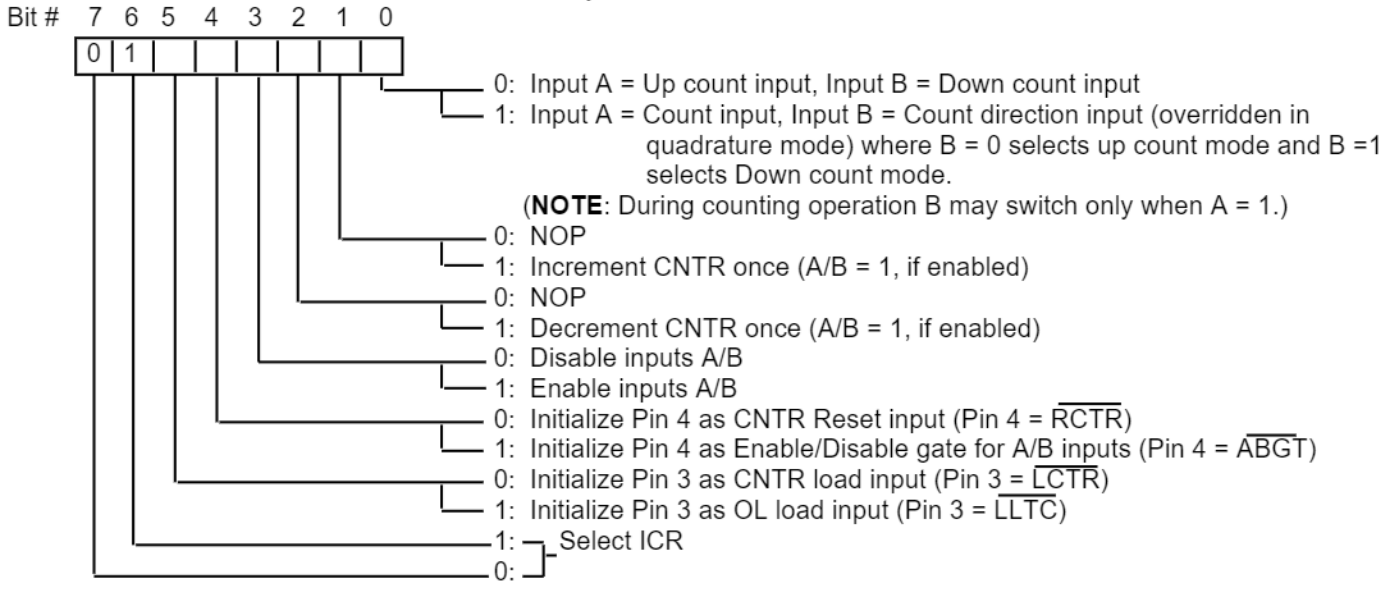
MCR (Master Control Register). Performs register reset and load operations. Writing a "non-zero" word to MCR does not require a follow-up write of an "all-zero" word to terminate a designated operation.
Accessed by**:** WRITE when $C/\overline{D} = 1$, $\overline{CS} = 0$.



1: Reset PR/OL address pointer
1: Transfer CNTR to OL (24 bits)
1: Reset CNTR, BWT and CYT. Set SIGN bit.
(CNTR=0, BWT=0, CYT=0, SIGN=1)
1: Transfer PR to CNTR (24 bits)
1: Reset COMPT (COMPT = 0)
1: Master reset. Reset CNTR, ICR, OCCR, QR, BWT, CYT, OL
COMPT, and PR/OL address pointer. Set PR (PR=FFFFFF) and SIGN.
0: } Select MCR
0:

NOTE: Control functions may be combined.

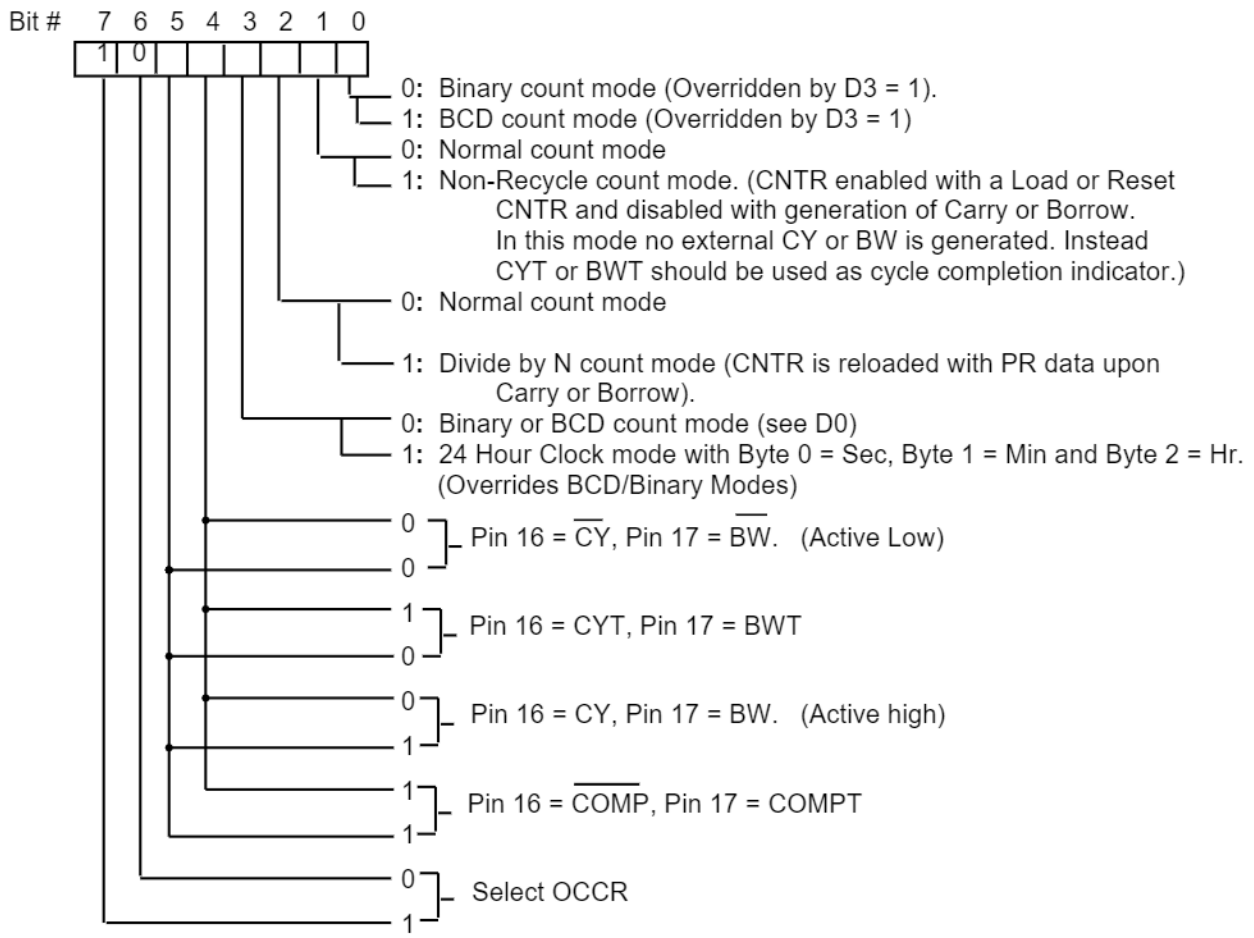
ICR (Input Control Register). Initializes counter input operating modes.
Accessed by**:** WRITE when $C/\overline{D} = 1$, $\overline{CS} = 0$.



0: Input A = Up count input, Input B = Down count input
1: Input A = Count input, Input B = Count direction input (overridden in quadrature mode) where B = 0 selects up count mode and B =1 selects Down count mode.
(**NOTE**: During counting operation B may switch only when A = 1.)
0: NOP
1: Increment CNTR once (A/B = 1, if enabled)
0: NOP
1: Decrement CNTR once (A/B = 1, if enabled)
0: Disable inputs A/B
1: Enable inputs A/B
0: Initialize Pin 4 as CNTR Reset input (Pin 4 = $\overline{\text{RCTR}}$)
1: Initialize Pin 4 as Enable/Disable gate for A/B inputs (Pin 4 = $\overline{\text{ABGT}}$)
0: Initialize Pin 3 as CNTR load input (Pin 3 = $\overline{\text{LCTR}}$)
1: Initialize Pin 3 as OL load input (Pin 3 = $\overline{\text{LLTC}}$)
1: } Select ICR
0:

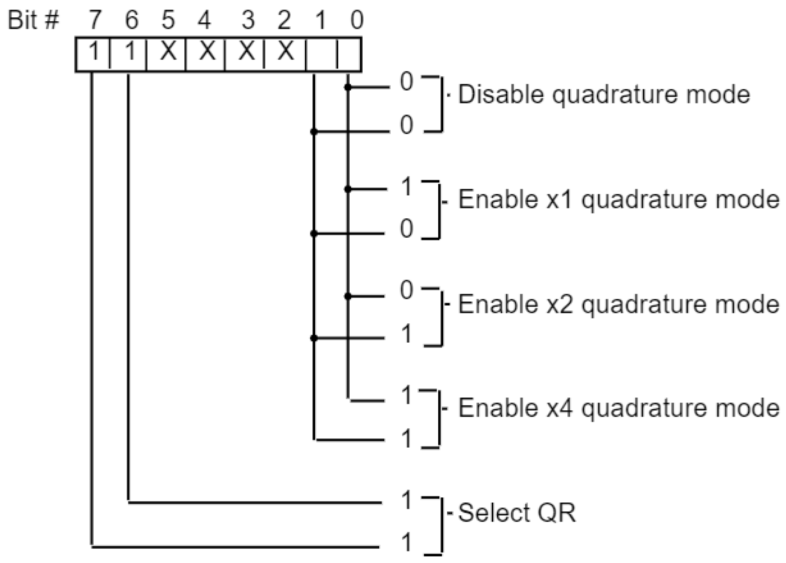
NOTE: Control functions may be combined.

OCCR (Output Control Register) Initializes CNTR and output operating modes.
Accessed by **:** WRITE when C/$\overline{D}$ = 1, $\overline{CS}$ = 0.



0**:** Binary count mode (Overridden by D3 = 1).
1**:** BCD count mode (Overridden by D3 = 1)
0**:** Normal count mode
1**:** Non-Recycle count mode. (CNTR enabled with a Load or Reset CNTR and disabled with generation of Carry or Borrow. In this mode no external CY or BW is generated. Instead CYT or BWT should be used as cycle completion indicator.)
0**:** Normal count mode
1**:** Divide by N count mode (CNTR is reloaded with PR data upon Carry or Borrow).
0**:** Binary or BCD count mode (see D0)
1**:** 24 Hour Clock mode with Byte 0 = Sec, Byte 1 = Min and Byte 2 = Hr. (Overrides BCD/Binary Modes)
0 0 Pin 16 = $\overline{CY}$, Pin 17 = $\overline{BW}$. (Active Low)
1 0 Pin 16 = CYT, Pin 17 = BWT
0 1 Pin 16 = CY, Pin 17 = BW. (Active high)
1 1 Pin 16 = $\overline{COMP}$, Pin 17 = COMPT
0 1 Select OCCR

QR (Quadrature Register). Selects quadrature count mode (See Fig. 7)
Accessed by**:** WRITE when C/$\overline{D}$ = 1, $\overline{CS}$ = 0.



0 0 Disable quadrature mode
1 0 Enable x1 quadrature mode
0 1 Enable x2 quadrature mode
1 1 Enable x4 quadrature mode
1 1 Select QR

X = Don't Care

7166-062394-4

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.
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. When B is programmed as 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	Volts
Operating Temperature	T_A	0 to +70	^oC
Storage Temperature	T_{STG}	-65 to +150	^oC
Supply Voltage	V_{DD} - V_{SS}	+7.0	Volts

DC Electrical Characteristics. (All voltages referenced to V_{SS}.
$T_A = 0^\circ$ to 70°C, V_{DD} = 4.5V to 5.5V, fc = 0, unless otherwise specified)

Parameter	Symbol	Min. Value	Max.Value	Unit	Remarks
Supply Voltage	V_{DD}	4.5	5.5	Volts	-
Supply Current	I_{DD}	-	350	µA	Outputs open
Input Low Voltage	V_{IL}	0	0.8	Volts	-
Input High Voltage	V_{IH}	2.0	V_{DD}	Volts	-
Output Low Voltage	V_{OL}	-	0.4	Volts	4mA Sink
Output High Voltage	V_{OH}	2.5	-	Volts	200µA Source
Input Current	-	-	15	nA	Leakage Current
Output Source Current	I_{SRC}	200	-	µA	V_{OH} = 2.5V
Output Sink Current	I_{SINK}	4	-	mA	V_{OL} = 0.4V
Data Bus Off-State Leakage Current	-	-	15	nA	-

TRANSIENT CHARACTERISTICS (See Timing Diagrams in Fig. 2 thru Fig. 7, V_{DD} = 4.5V to 5.5V, T_A = 0˚ to 70˚C, unless otherwise specified)

Parameter	Symbol	Min.Value	Max.Value	Unit
Clock A/B "Low"	T_{CL}	20	No Limit	ns
Clock A/B "High"	T_{CH}	30	No Limit	ns
Clock A/B Frequency (See NOTE 1)	fc	0	20	MHz
Clock UP/DN Reversal Delay	T_{UDD}	100	-	ns
$\overline{\text{LCTR}}$ Positive edge to the next A/B positive or negative edge delay	T_{LC}	100	-	ns
Clock A/B to $\overline{\text{CY}}$/$\overline{\text{BW}}$/$\overline{\text{COMP}}$ "low" propagation delay	T_{CBL}	-	65	ns
Clock A/B to $\overline{\text{CY}}$/$\overline{\text{BW}}$/$\overline{\text{COMP}}$ "high" propagation delay	T_{CBH}	-	85	ns
$\overline{\text{LCTR}}$ and $\overline{\text{LLTC}}$ 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
$\overline{\text{WR}}$ pulse width	T_{WW}	60	-	ns
$\overline{\text{RD}}$ to data out delay (C_L = 20pF)	T_R	-	110	ns
$\overline{\text{CS}}$, $\overline{\text{RD}}$ Terminate to Data-Bus Tri-State	T_{RT}	-	30	ns
Data-Bus set-up time for $\overline{\text{WR}}$	T_{DS}	15	-	ns **(see Note 3)**
Data-Bus hold time for $\overline{\text{WR}}$	T_{DH}	30	-	ns **(see Note 3)**
C/$\overline{\text{D}}$, $\overline{\text{CS}}$ set-up time for $\overline{\text{RD}}$	T_{CRS}	0	-	ns
C/$\overline{\text{D}}$, $\overline{\text{CS}}$ hold time for $\overline{\text{RD}}$	T_{CRH}	0	-	ns
C/$\overline{\text{D}}$ set-up time for $\overline{\text{WR}}$	T_{CWS}	15	-	ns **(see Note 3)**
C/$\overline{\text{D}}$ hold time for $\overline{\text{WR}}$	T_{CWH}	30	-	ns **(see Note 3)**
$\overline{\text{CS}}$ set-up time for $\overline{\text{WR}}$	T_{SWS}	15	-	ns **(see Note 3)**
$\overline{\text{CS}}$ holdtime for $\overline{\text{WR}}$	T_{SWH}	0	-	ns **(see Note 3)**
Quadrature Mode:				
Clock A/B Validation delay (See NOTE 2)	T_{CQV}	-	160	ns
A and B phase delay	T_{PH}	208	-	ns
Clock A/B frequency	f_{CQ}	-	1.2	MHz
$\overline{\text{CY}}$, $\overline{\text{BW}}$, $\overline{\text{COMP}}$ pulse width	T_{CBW}	75	180	ns

NOTE 1: A) In Divide by N mode, the maximum clock frequency is 10 MHz.
B) The maximum frequency for valid CY, BW, CYT, BWT, COMP, COMPT is 10 MHz.

NOTE 2: In quadrature mode A/B inputs are filtered and required to be stable for at least T_{CQV} length to be valid.

NOTE 3: All $\overline{\text{WR}}$ specifications are critical for proper operation of LS7166

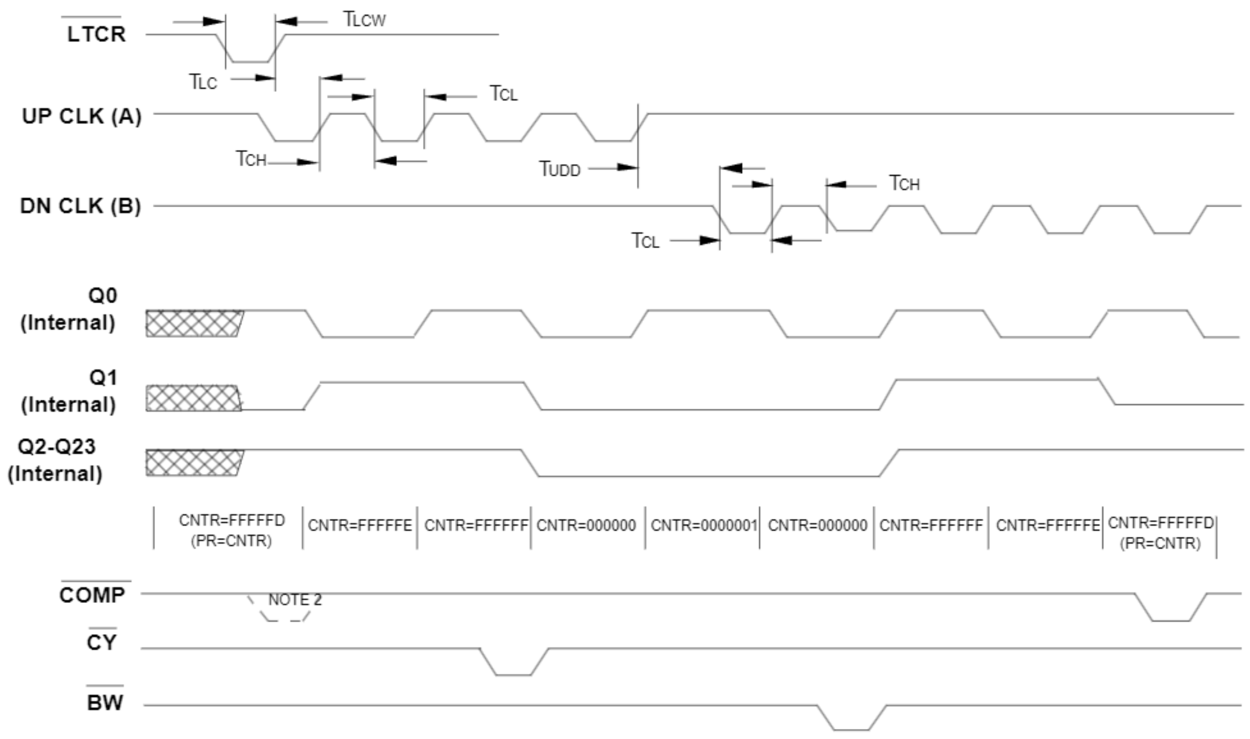


FIGURE 2 . LOAD COUNTER, UP CLOCK, DOWN CLOCK, COMPARE OUT, CARRY, BORROW

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.

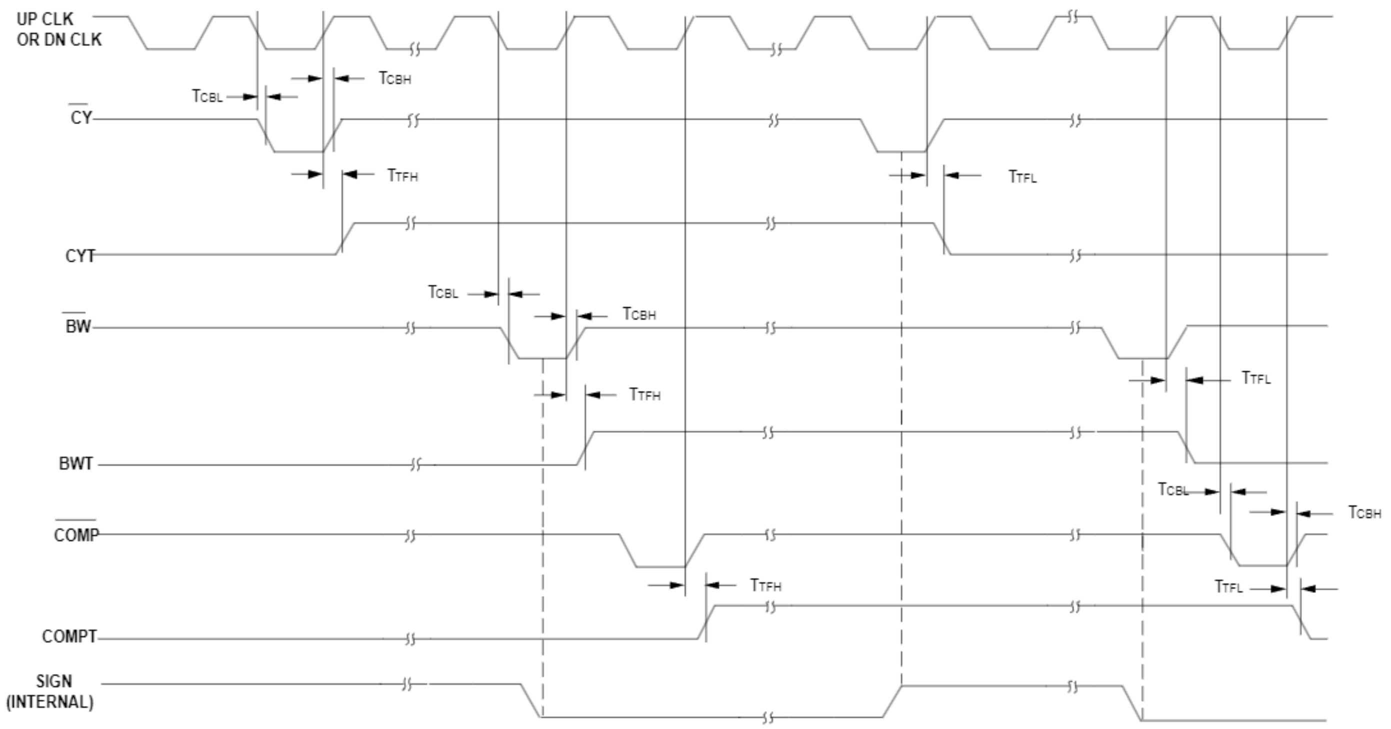


FIGURE 3. CLOCK TO $\overline{CY}/\overline{BW}$ OUTPUT PROPAGATION DELAYS

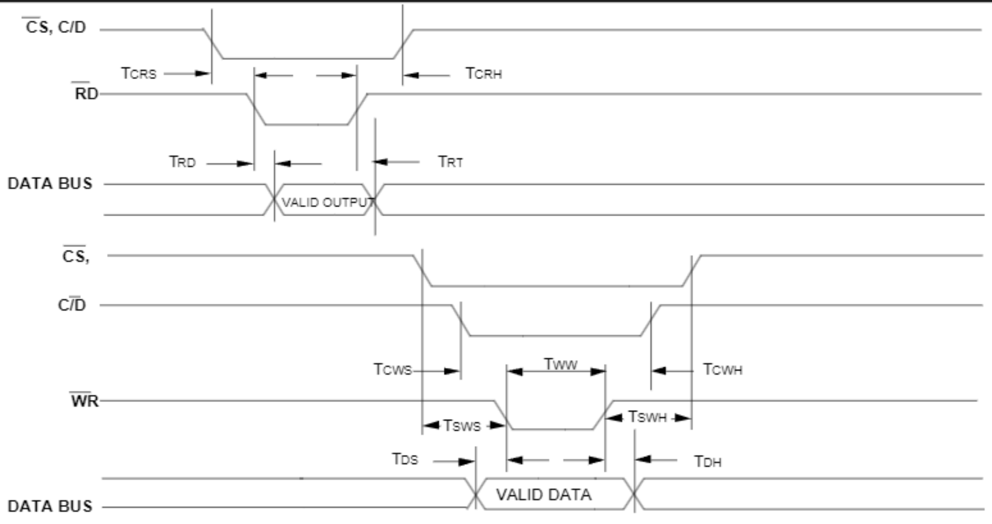
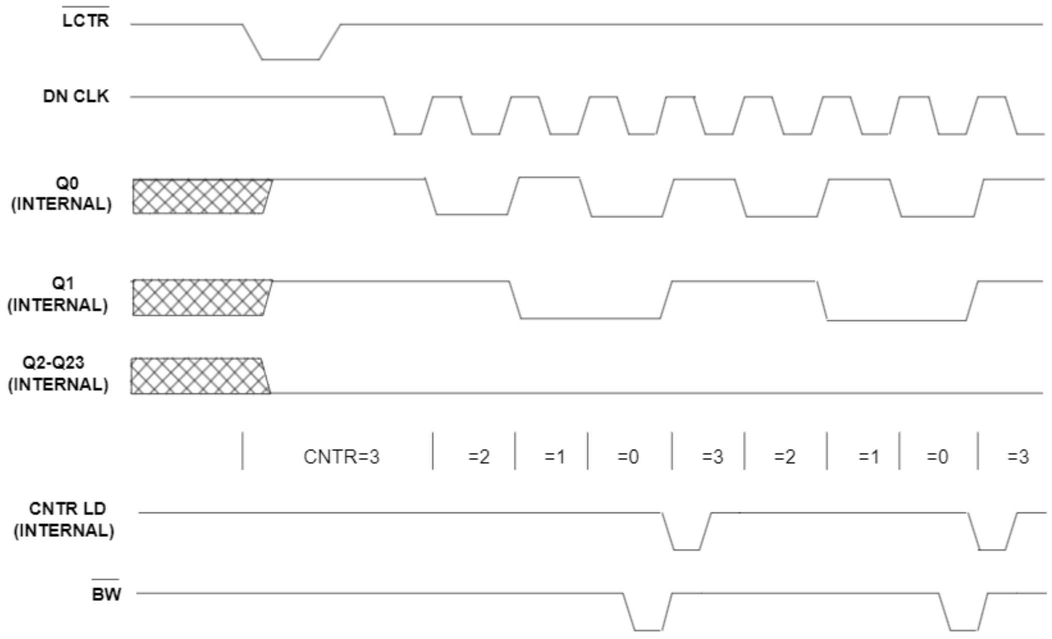


FIGURE 4. READ/WRITE CYCLES



NOTE: EXAMPLE OF DIVIDE BY 4 IN DOWN COUNT MODE

FIGURE 5. DIVIDE BY N MODE

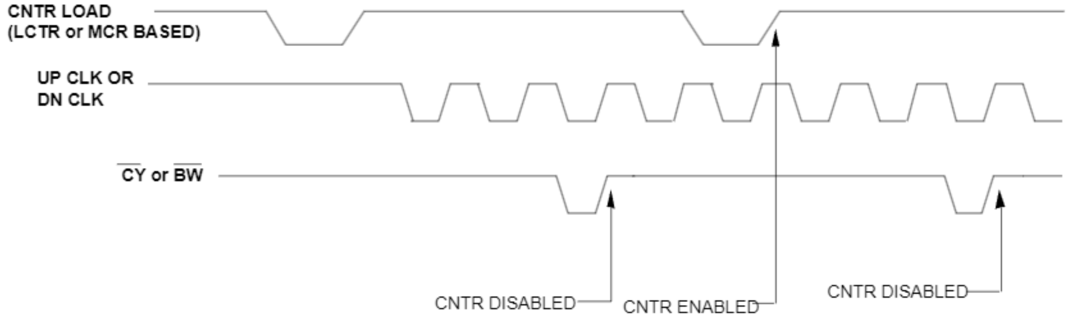


FIGURE 6 . CYCLE ONCE MODE

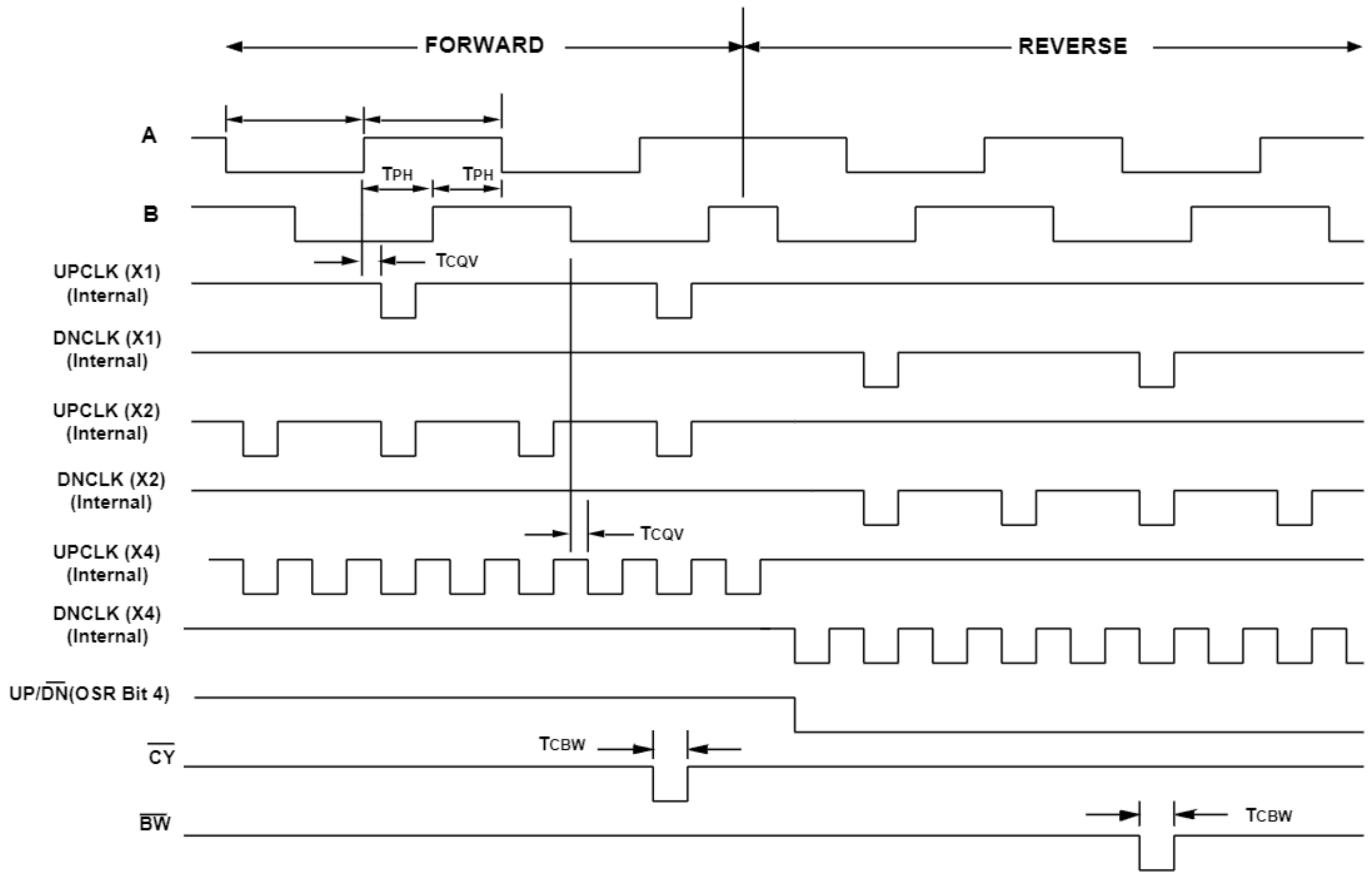


FIGURE 7.

QUADRATURE MODE INTERNAL CLOCKS

7166-030392-9

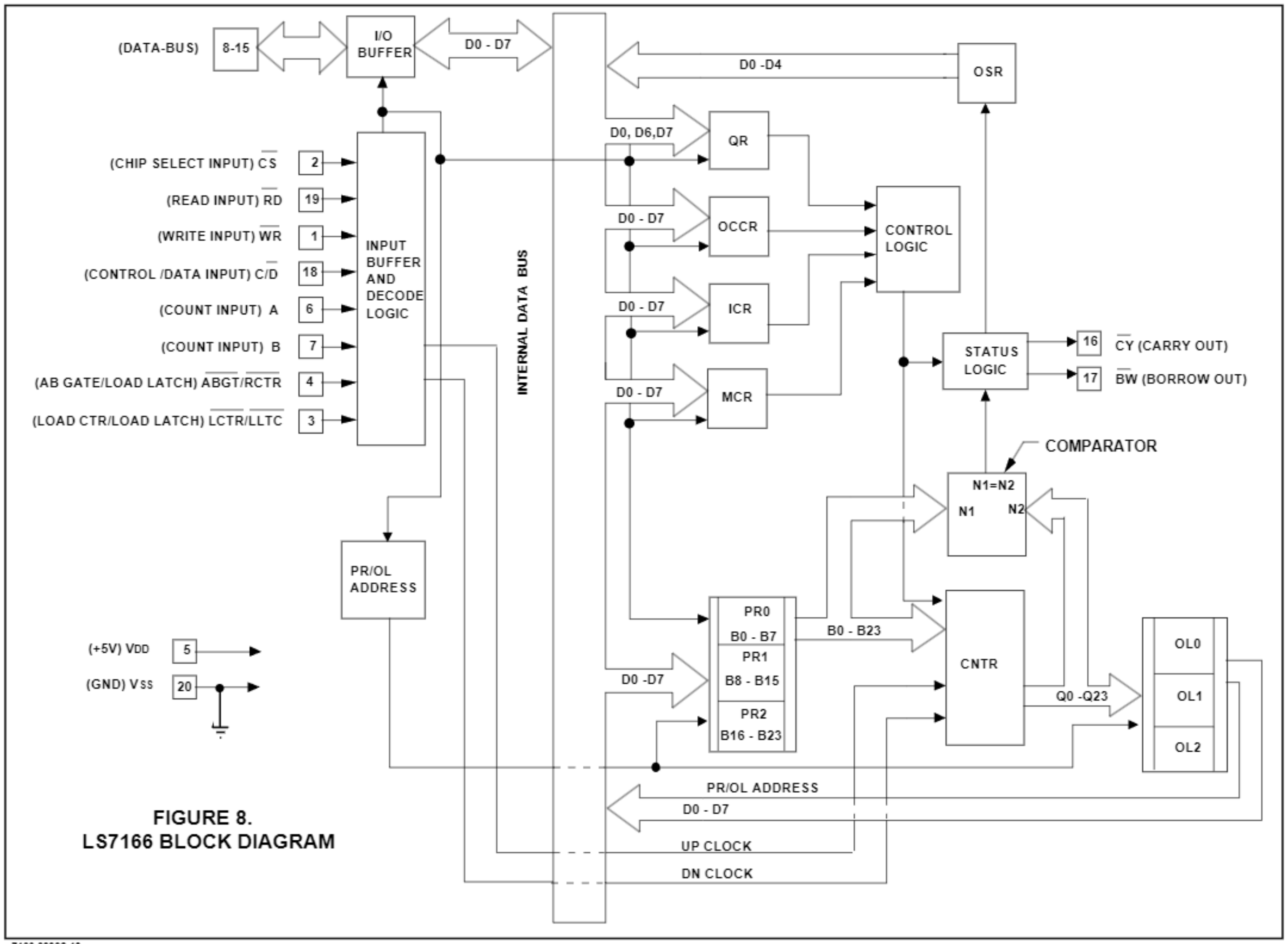
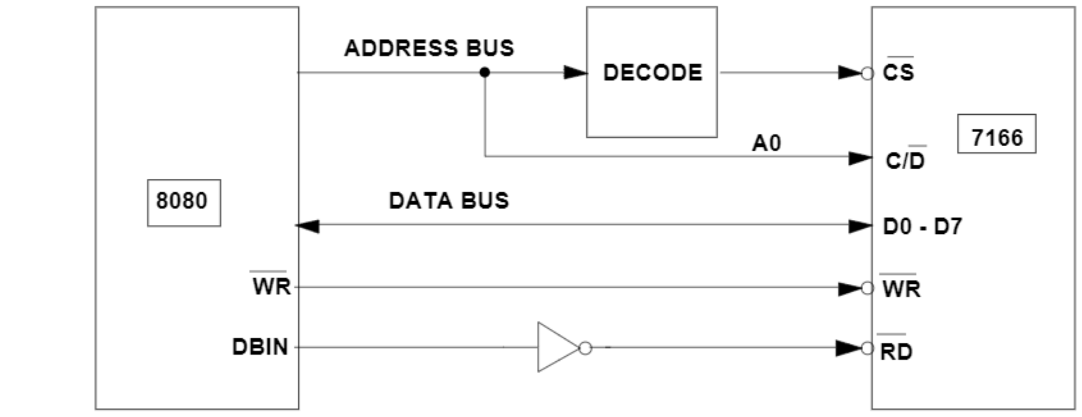
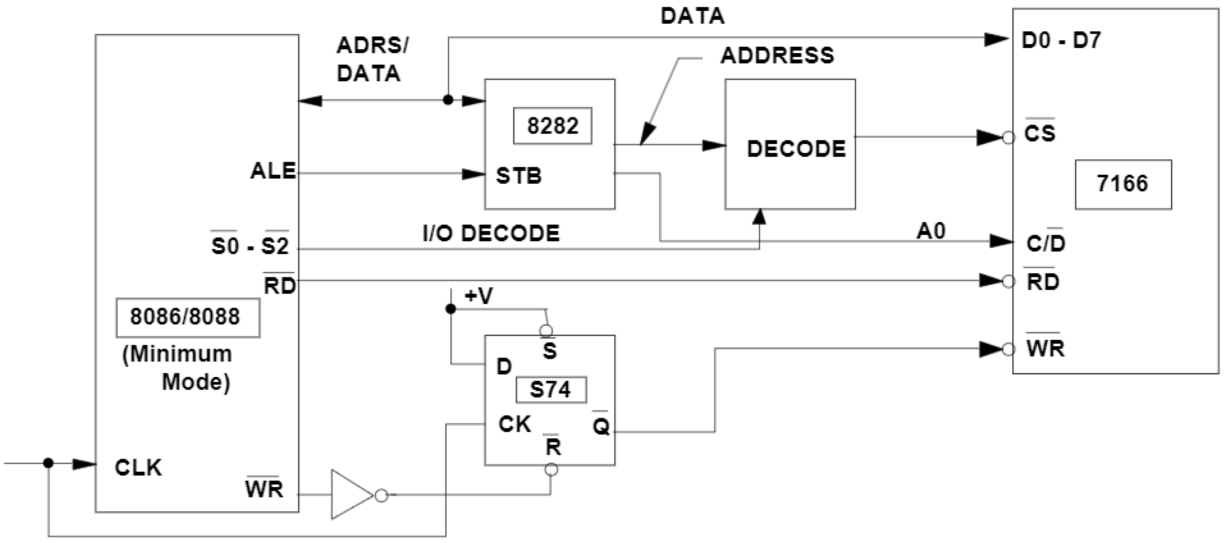


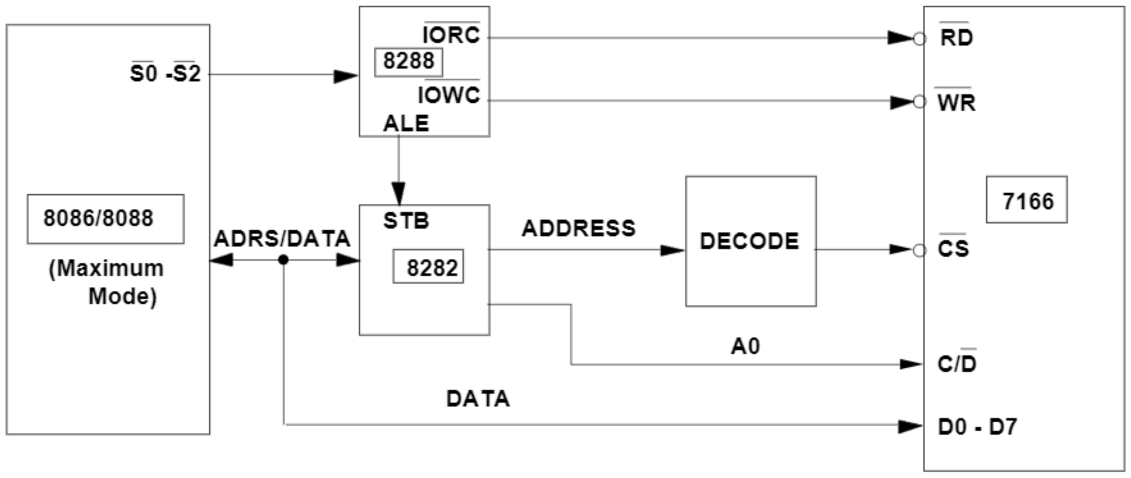
FIGURE 8.
LS7166 BLOCK DIAGRAM

7166-03392-10

LS7166 INTERFACE EXAMPLES

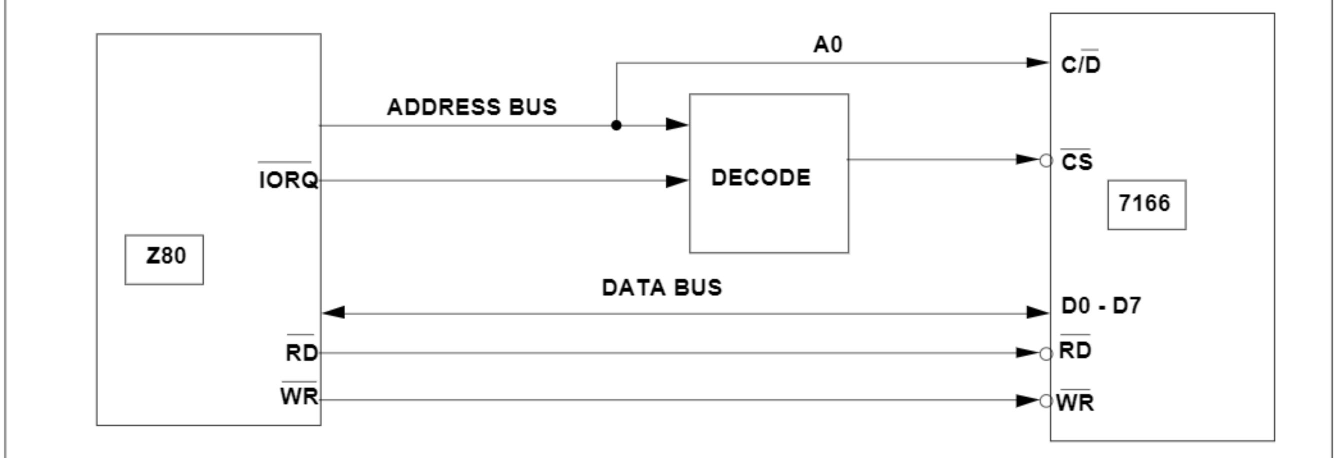


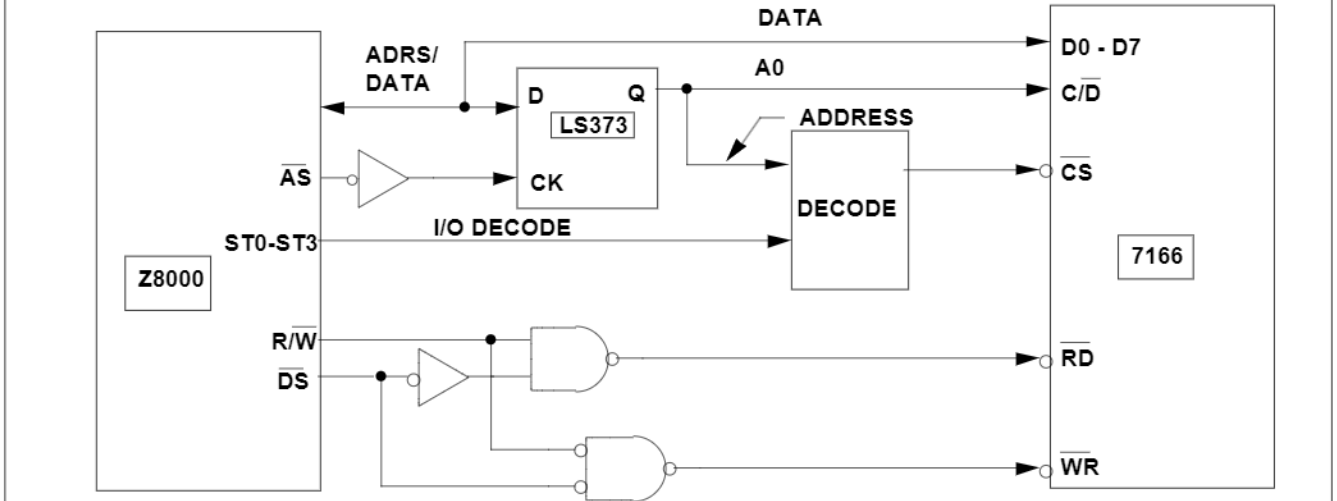


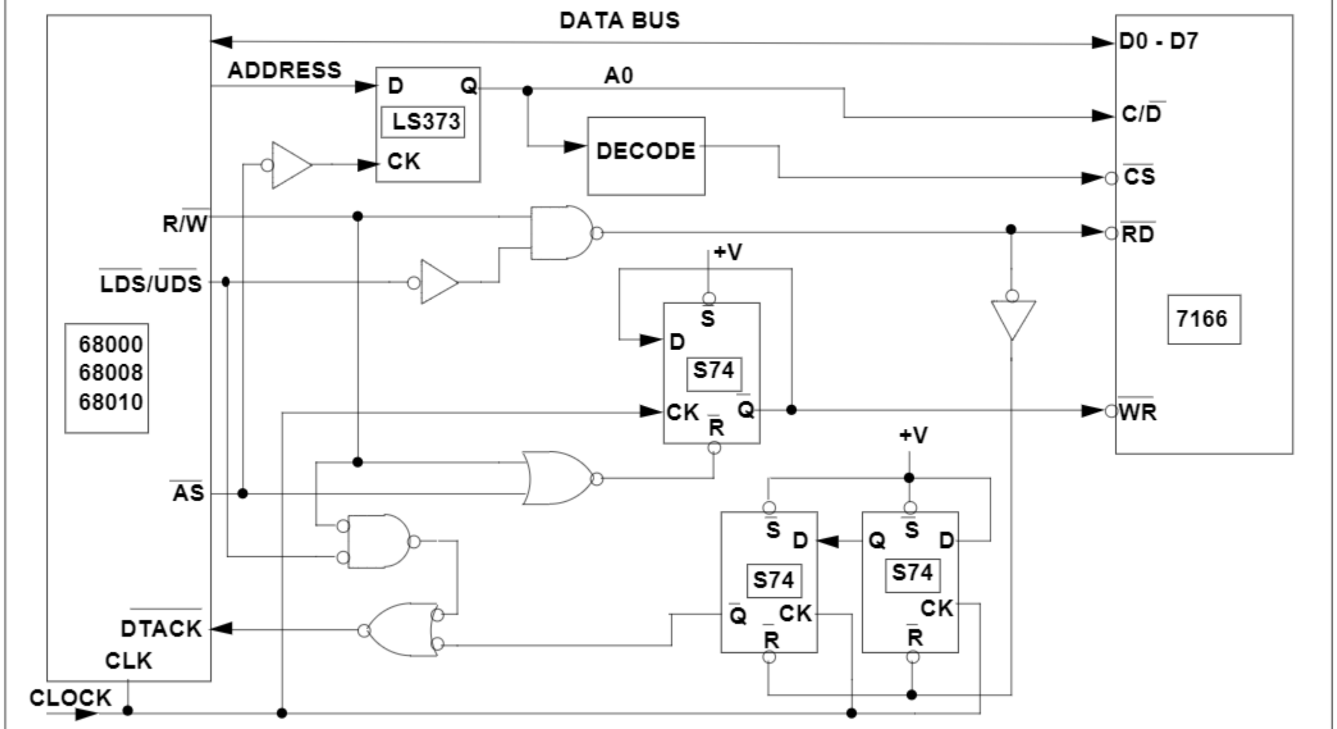


7166-03392-11

LS7166 INTERFACE EXAMPLES







7166-03392-12

INTEGRATED CIRCUITS

DATA SHEET

PCA9535

16-bit I^{2}C and SMBus, low power I/O port with interrupt

Product data 2003 Jun 27

Philips
Semiconductors

PHILIPS

16-bit I^{2}C and SMBus, low power I/O port with interrupt PCA9535

FEATURES

- Operating power supply voltage range of 2.3 V-5.5 V
- 5 V tolerant I/Os
- Polarity inversion register
- Active LOW interrupt output
- Low stand-by current
- Noise filter on SCL/SDA inputs
- No glitch on power-up
- Internal power-on reset
- 16 I/O pins which default to 16 inputs
- 0 to 400 kHz clock frequency
- ESD protection exceeds 2000 V HBM per JESD22-A114, 200 V MM per JESD22-A115, and 1000 V CDM per JESD22-C101
- Latch-up testing is done to JESDEC Standard JESD78 which exceeds 100 mA
- Offered in three different packages: SO24, TSSOP24, and HVQFN24

DESCRIPTION

The PCA9535 is a 24-pin CMOS device that provide 16 bits of General Purpose parallel Input/Output (GPIO) expansion for I^{2}C/SMBus applications and was developed to enhance the Philips family of I^{2}C I/O expanders. The improvements include higher drive capability, 5 V I/O tolerance, lower supply current, individual I/O configuration, and smaller packaging. I/O expanders provide a simple solution when additional I/O is needed for ACPI power switches, sensors, pushbuttons, LEDs, fans, etc.

The PCA9535 consist of two 8-bit Configuration (Input or Output selection); Input, Output and Polarity inversion (Active HIGH or Active LOW operation) registers. The system master can enable the I/Os as either inputs or outputs by writing to the I/O configuration bits. The data for each Input or Output is kept in the corresponding Input or Output register. The polarity of the read register can be inverted with the Polarity Inversion Register. All registers can be read by the system master. Although pin-to-pin and I^{2}C address compatible with the PCF8575, software changes are required due to the enhancements and are discussed in Application Note AN469.

The PCA9535 is identical to the PCA9555 except for the removal of the internal I/O pull-up resistor which greatly reduces power consumption when the I/Os are held LOW.

The PCA9535 open-drain interrupt output is activated when any input state differs from its corresponding input port register state and is used to indicate to the system master that an input state has changed. The power-on reset sets the registers to their default values and initializes the device state machine.

Three hardware pins (A0, A1, A2) vary the fixed I^{2}C address and allow up to eight devices to share the same I^{2}C/SMBus. The fixed I^{2}C address of the PCA9535 is the same as the PCA9554 allowing up to eight of these devices in any combination to share the same I^{2}C/SMBus.

ORDERING INFORMATION

PACKAGES	TEMPERATURE RANGE	ORDER CODE	TOPSIDE MARK	DRAWING NUMBER
24-Pin Plastic SO	-40 to +85 °C	PCA9535D	PCA9535D	SOT137-1
24-Pin Plastic TSSOP	-40 to +85 °C	PCA9535PW	PCA9535PW	SOT355-1
24-Pin Plastic HVQFN	-40 to +85 °C	PCA9535BS	9535	SOT616-1

Standard packing quantities and other packing data are available at *www.philipslogic.com/packaging*.
I^{2}C is a trademark of Philips Semiconductors Corporation.
SMBus as specified by the Smart Battery System Implementers Forum is a derivative of the Philips I^{2}C patent.

PIN CONFIGURATION — SO, TSSOP

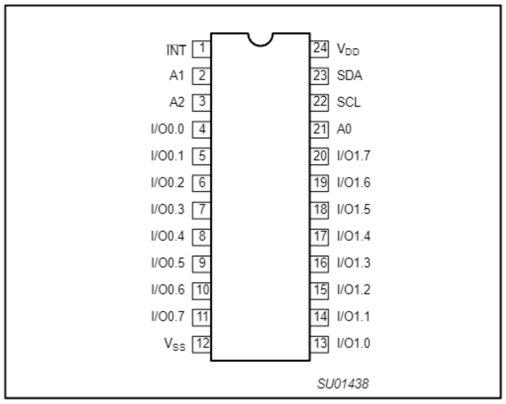


Figure 1. Pin configuration — SO, TSSOP

PIN CONFIGURATION —HVQFN

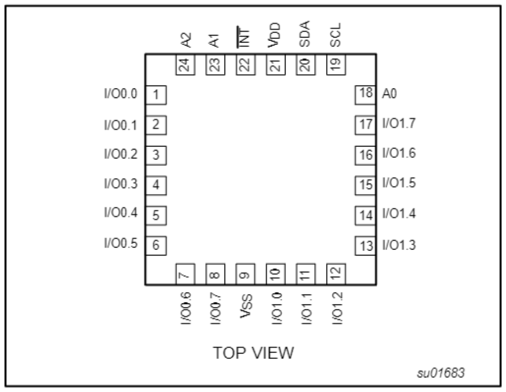


Figure 2. Pin configuration — HVQFN

PIN DESCRIPTION

SO, TSSOP PIN NUMBER	HVQFN PIN NUMBER	SYMBOL	FUNCTION
1	22	$\overline{\text{INT}}$	Interrupt output (open drain)
2	23	A1	Address input 1
3	24	A2	Address input 2
4-11	1-8	I/O0.0-I/O0.7	I/O0.0 to I/O0.7
12	9	V_{SS}	Supply ground
13-20	10-17	I/O1.0-I/O1.7	I/O1.0 to I/O1.7
21	18	A0	Address input 0
22	19	SCL	Serial clock line
23	20	SDA	Serial data line
24	21	V_{DD}	Supply voltage

BLOCK DIAGRAM

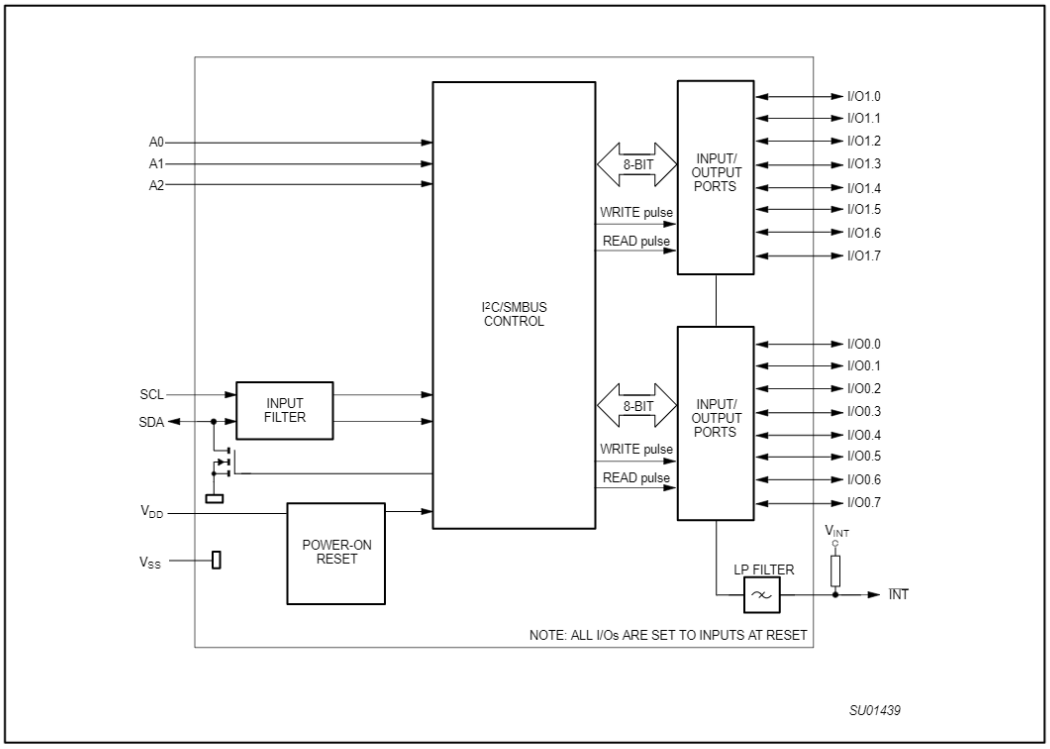
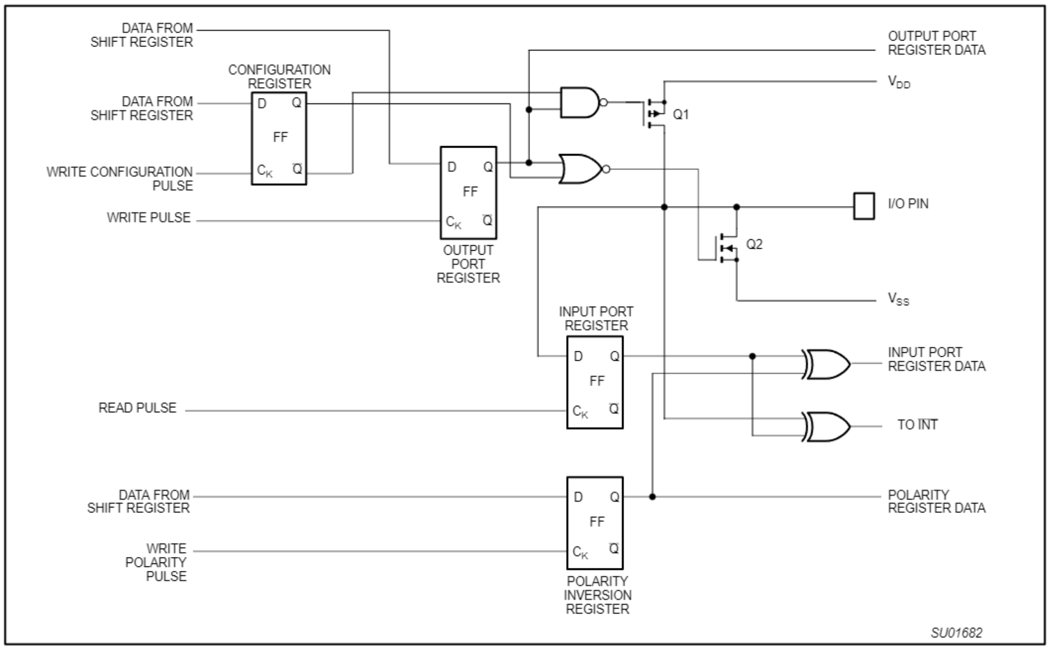


Figure 3. Block diagram

SIMPLIFIED SCHEMATIC OF I/Os



NOTE: At Power-on Reset, all registers return to default values.

Figure 4. Simplified schematic of I/Os

I/O port

When an I/O is configured as an input, FETs Q1 and Q2 are off, creating a high impedance input. The input voltage may be raised above V_{DD} to a maximum of 5.5 V.

If the I/O is configured as an output, then either Q1 or Q2 is on, depending on the state of the Output Port register. Care should be exercised if an external voltage is applied to an I/O configured as an output because of the low impedance path that exists between the pin and either V_{DD} or V_{SS}.

REGISTERS

Command Byte

Command	Register
0	Input port 0
1	Input port 1
2	Output port 0
3	Output port 1
4	Polarity inversion port 0
5	Polarity inversion port 1
6	Configuration port 0
7	Configuration port 1

The command byte is the first byte to follow the address byte during a write transmission. It is used as a pointer to determine which of the following registers will be written or read.

Registers 0 and 1 — Input Port Registers

This register is an input-only port. It reflects the incoming logic levels of the pins, regardless of whether the pin is defined as an input or an output by Register 3. Writes to this register have no effect.

Registers 2 and 3 — Output Port Registers

bit	O0.7	O0.6	O0.5	O0.4	O0.3	O0.2	O0.1	O0.0
default	1	1	1	1	1	1	1	1
bit	O1.7	O1.6	O1.5	O1.4	O1.3	O1.2	O1.1	O1.0
default	1	1	1	1	1	1	1	1

This register is an output-only port. It reflects the outgoing logic levels of the pins defined as outputs by Register 6 and 7. Bit values in this register have no effect on pins defined as inputs. In turn, reads from this register reflect the value that is in the flip-flop controlling the output selection, NOT the actual pin value.

Registers 4 and 5 — Polarity Inversion Registers

bit	N0.7	N0.6	N0.5	N0.4	N0.3	N0.2	N0.1	N0.0
default	0	0	0	0	0	0	0	0
bit	N1.7	N1.6	N1.5	N1.4	N1.3	N1.2	N1.1	N1.0
default	0	0	0	0	0	0	0	0

This register allows the user to invert the polarity of the Input Port register data. If a bit in this register is set (written with ‘1’), the Input Port data polarity is inverted. If a bit in this register is cleared (written with a ‘0’), the Input Port data polarity is retained.

Registers 6 and 7 — Configuration Registers

bit	C0.7	C0.6	C0.5	C0.4	C0.3	C0.2	C0.1	C0.0
default	1	1	1	1	1	1	1	1
bit	C1.7	C1.6	C1.5	C1.4	C1.3	C1.2	C1.1	C1.0
default	1	1	1	1	1	1	1	1

This register configures the directions of the I/O pins. If a bit in this register is set (written with ‘1’), the corresponding port pin is enabled as an input with high impedance output driver. If a bit in this register is cleared (written with ‘0’), the corresponding port pin is enabled as an output. At reset the device’s ports are inputs.

POWER-ON RESET

When power is applied to V_{DD}, an internal power-on reset holds the PCA9535 in a reset state until V_{DD} has reached V_{POR}. At that point, the reset condition is released and the PCA9535 registers and SMBus state machine will initialize to their default states.

DEVICE ADDRESS

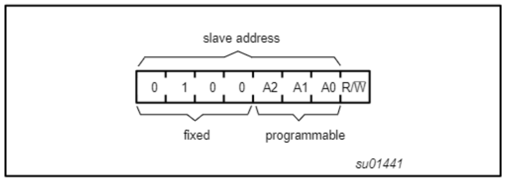


Figure 5. PCA9535 address

BUS TRANSACTIONS

Writing to the port registers

Data is transmitted to the PCA9535 by sending the device address and setting the least significant bit to a logic 0 (see Figure 5 for device address). The command byte is sent after the address and determines which register will receive the data following the command byte.

The eight registers within the PCA9535 are configured to operate as four register pairs. The four pairs are Input Ports, Output Ports, Polarity Inversion Ports, and Configuration Ports. After sending data to one register, the next data byte will be sent to the other register in the pair (see Figures 6 and 7). For example, if the first byte is sent to Output Port (register 3), then the next byte will be stored in Output Port 0 (register 2). There is no limitation on the number of data bytes sent in one write transmission. In this way, each 8-bit register may be updated independently of the other registers.

Reading the port registers

In order to read data from the PCA9535, the bus master must first send the PCA9535 address with the least significant bit set to a logic 0 (see Figure 5 for device address). The command byte is sent after the address and determines which register will be accessed. After a restart, the device address is sent again but this time, the least significant bit is set to a logic 1. Data from the register defined by the command byte will then be sent by the PCA9535 (see Figures 8 , 9, and 10). Data is clocked into the register on the falling edge of the acknowledge clock pulse. After the first byte is read, additional bytes may be read but the data will now reflect the information in the other register in the pair. For example, if you read Input Port 1, then the next byte read would be Input Port 0. There is no limitation on the number of data bytes received in one read transmission but the final byte received, the bus master must not acknowledge the data.

Interrupt Output

The open-drain interrupt output is activated when one of the port pins change state and the pin is configured as an input. The interrupt is deactivated when the input returns to its previous state or the input port register is read (see Figure 9). A pin configured as an output cannot cause an interrupt. Since each 8-bit port is read independently, the interrupt caused by Port 0 will not be cleared by a read of Port 1 or the other way around.

Note that changing an I/O from an output to an input may cause a false interrupt to occur if the state of the pin does not match the contents of the Input Port register.

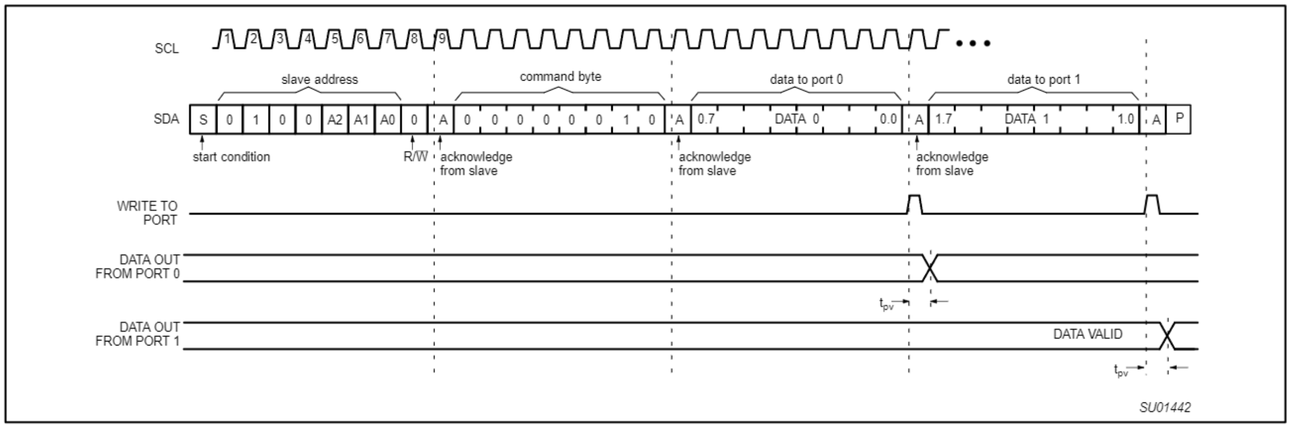


Figure 6. WRITE to output port registers

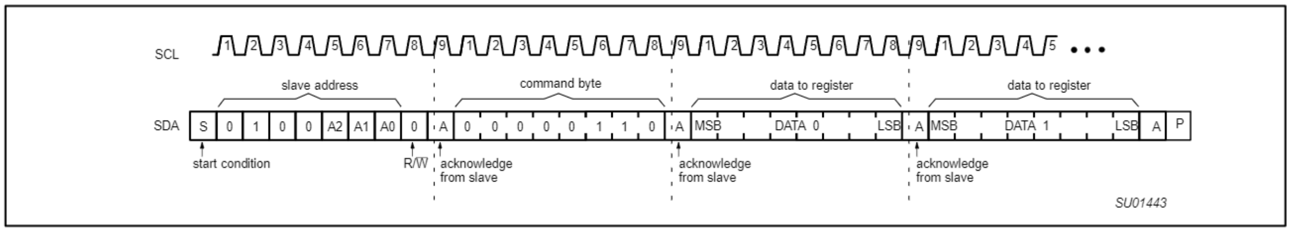
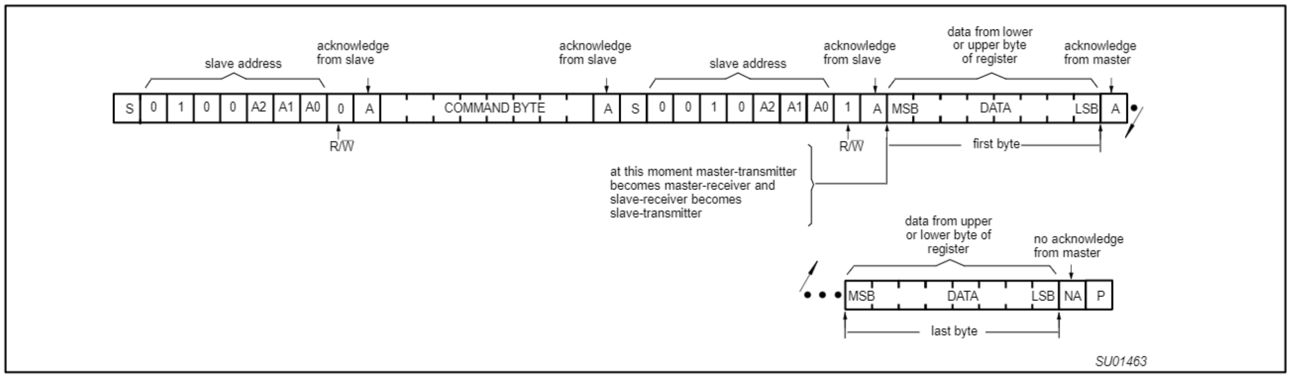
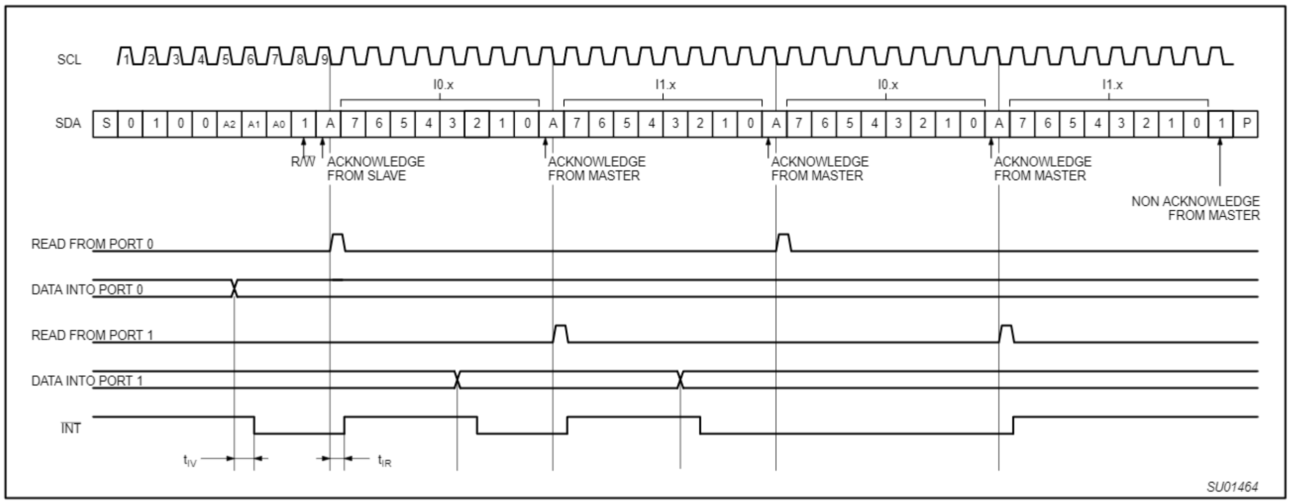


Figure 7. WRITE to configuration registers


slave address
acknowledge from slave
acknowledge from slave
slave address
acknowledge from slave
data from lower or upper byte of register
acknowledge from master
S 0 1 0 0 A2 A1 A0 0 A
COMMAND BYTE
A S 0 0 1 0 A2 A1 A0 1 A MSB DATA LSB A
R/W
R/W
first byte
at this moment master-transmitter becomes master-receiver and slave-receiver becomes slave-transmitter
data from upper or lower byte of register
no acknowledge from master
MSB DATA LSB NA P
last byte
SU01463

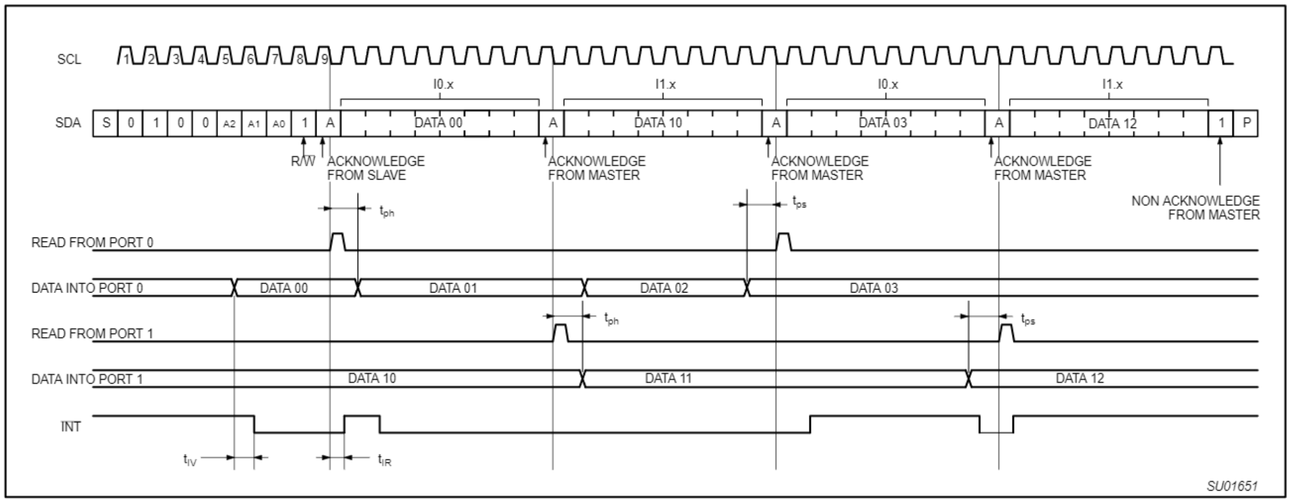
NOTE: Transfer can be stopped at any time by a STOP condition.

Figure 8. READ from register



NOTES: Transfer of data can be stopped at any moment by a STOP condition. When this occurs, data present at the latest acknowledge phase is valid (output mode).
It is assumed that the command byte has previously been set to 00 (read input port port register).

Figure 9. READ input port register — scenario 1



NOTES: Transfer of data can be stopped at any moment by a STOP condition. When this occurs, data present at the latest acknowledge phase is valid (output mode).
It is assumed that the command byte has previously been set to 00 (read input port port register).

Figure 10. READ input port register — scenario 2

TYPICAL APPLICATION

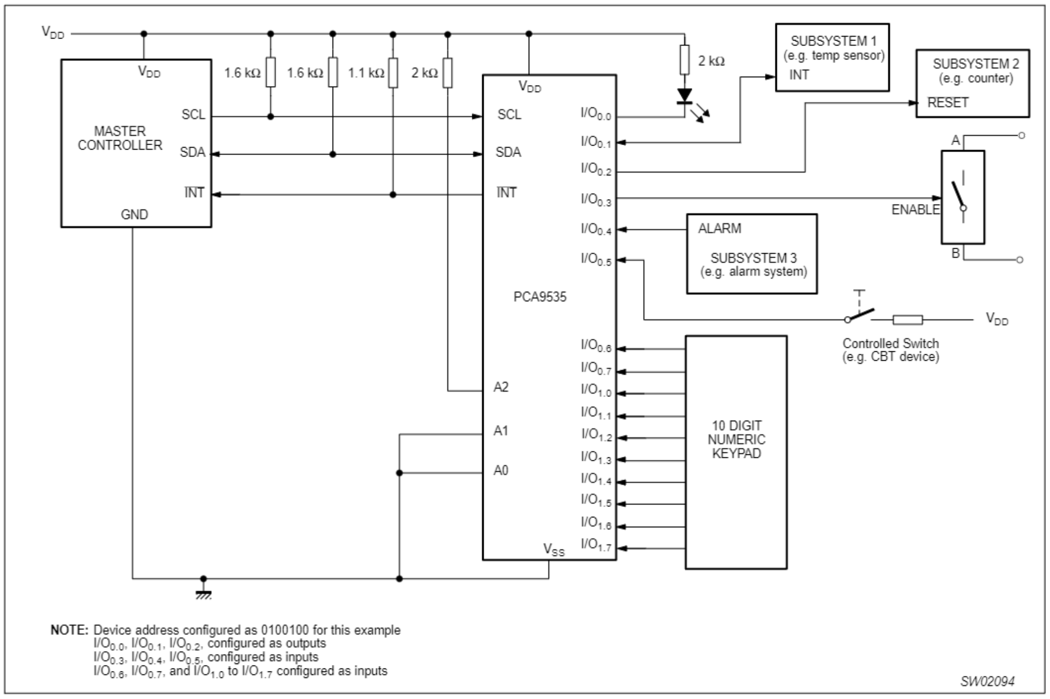

V_DD
MASTER CONTROLLER
SCL
SDA
INT
GND
1.6 kΩ
1.6 kΩ
1.1 kΩ
2 kΩ
PCA9535
A2
A1
A0
V_SS
I/O0.0 … I/O1.7
2 kΩ
SUBSYSTEM 1 (e.g. temp sensor) INT
SUBSYSTEM 2 (e.g. counter) RESET
ENABLE
A
B
ALARM
SUBSYSTEM 3 (e.g. alarm system)
Controlled Switch (e.g. CBT device)
10 DIGIT NUMERIC KEYPAD
NOTE: Device address configured as 0100100 for this example
I/O0.0, I/O0.1, I/O0.2, configured as outputs
I/O0.3, I/O0.4, I/O0.5, configured as inputs
I/O0.6, I/O0.7, and I/O1.0 to I/O1.7 configured as inputs
SW02094

Figure 11. Typical application

Minimizing I_{DD} when the I/O is used to control LEDs

When the I/Os are used to control LEDs, they are normally connected to V_{DD} through a resistor as shown in Figure 11. Since the LED acts as a diode, when the LED is off the I/O V_{IN} is about 1.2 V less than V_{DD}. The supply current, I_{DD}, increases as V_{IN} becomes lower than V_{DD} and is specified as ΔI_{DD} in the DC characteristics table.

Designs needing to minimize current consumption, such as battery power applications, should consider maintaining the I/O pins greater than or equal to V_{DD} when the LED is off. Figure 12 shows a high value resistor in parallel with the LED. Figure 13 shows V_{DD} less than the LED supply voltage by at least 1.2 V. Both of these methods maintain the I/O V_{IN} at or above V_{DD} and prevents additional supply current consumption when the LED is off.

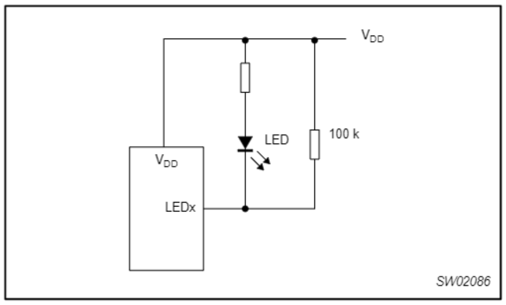


Figure 12. High value resistor in parallel with the LED

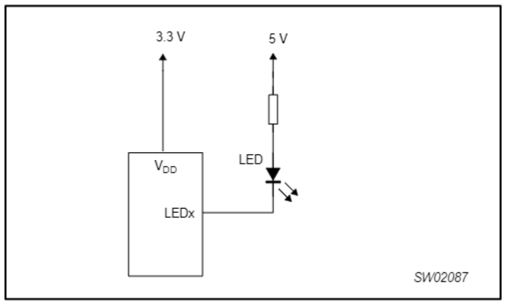


Figure 13. Device supplied by a lower voltage

ABSOLUTE MAXIMUM RATINGS

In accordance with the Absolute Maximum Rating System (IEC 134)

SYMBOL	PARAMETER	CONDITIONS	MIN	MAX	UNIT
V_{DD}	Supply voltage		-0.5	6.0	V
$V_{I/O}$	DC input current on an I/O		V_{SS} - 0.5	6	V
$I_{I/O}$	DC output current on an I/O		—	± 50	mA
I_I	DC input current		—	± 20	mA
I_{DD}	Supply current		—	160	mA
I_{SS}	Supply current		—	200	mA
P_{tot}	Total power dissipation		—	200	mW
T_{stg}	Storage temperature range		-65	+150	°C
T_{amb}	Operating ambient temperature		-40	+85	°C

HANDLING

Inputs and outputs are protected against electrostatic discharge in normal handling. However, to be totally safe, it is desirable to take precautions appropriate to handling MOS devices. Advice can be found in Data Handbook IC24 under "*Handling MOS devices*".

DC CHARACTERISTICS

V_{DD} = 2.3 to 5.5 V; V_{SS} = 0 V; T_{amb} = -40 to +85 °C; unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNIT
Supplies						
V_{DD}	Supply voltage		2.3	—	5.5	V
I_{DD}	Supply current	Operating mode; V_{DD} = 5.5 V; no load; f_{SCL} = 100 kHz; I/O = inputs	—	135	200	µA
I_{stbl}	Standby current	Standby mode; V_{DD} = 5.5 V; no load; $V_I = V_{SS}$; f_{SCL} = 0 kHz; I/O = inputs	—	0.25	1	µA
I_{stbh}	Standby current	Standby mode; V_{DD} = 5.5 V; no load; $V_I = V_{DD}$; f_{SCL} = 0 kHz; I/O = inputs	—	0.25	1	µA
V_{POR}	Power-on reset voltage	No load; $V_I = V_{DD}$ or V_{SS}	—	1.5	1.65	V
input SCL; input/output SDA						
V_{IL}	LOW-level input voltage		-0.5	—	0.3 V_{DD}	V
V_{IH}	HIGH-level input voltage		0.7 V_{DD}	—	5.5	V
I_{OL}	LOW-level output current	V_{OL} = 0.4V	3	—	—	mA
I_L	Leakage current	$V_I = V_{DD} = V_{SS}$	-1	—	+1	µA
C_I	Input capacitance	$V_I = V_{SS}$	—	6	10	pF
I/Os						
V_{IL}	LOW-level input voltage		-0.5	—	0.8	V
V_{IH}	HIGH-level input voltage		2.0	—	5.5	V
I_{OL}	LOW-level output current	V_{OL} = 0.5 V; V_{DD} = 2.3-5.5 V; Note 1	8	8-20	—	mA
		V_{OL} = 0.7 V; V_{DD} = 2.3-5.5 V; Note 1	10	10-24	—	mA
V_{OH}	HIGH-level output voltage	I_{OH} = -8 mA; V_{DD} = 2.3 V; Note 2	1.8	—	—	V
		I_{OH} = -10 mA; V_{DD} = 2.3 V; Note 2	1.7	—	—	V
		I_{OH} = -8 mA; V_{DD} = 3.0 V; Note 2	2.6	—	—	V
		I_{OH} = -10 mA; V_{DD} = 3.0 V; Note 2	2.5	—	—	V
		I_{OH} = -8 mA; V_{DD} = 4.75 V; Note 2	4.1	—	—	V
		I_{OH} = -10 mA; V_{DD} = 4.75 V; Note 2	4.0	—	—	V
I_{IH}	Input leakage current	V_{DD} = 5.5 V; $V_I = V_{DD}$	—	—	1	µA
I_{IL}	Input leakage current	V_{DD} = 5.5 V; $V_I = V_{SS}$	—	—	-1	µA
C_I	Input capacitance		—	3.7	5	pF
C_O	Output capacitance		—	3.7	5	pF
Interrupt $\overline{INT}$						
I_{OL}	LOW-level output current	V_{OL} = 0.4 V	3	—	—	mA
Select Inputs A0, A1, A2						
V_{IL}	LOW-level input voltage		-0.5	—	0.8	V
V_{IH}	HIGH-level input voltage		2.0	—	5.5	V
I_{LI}	Input leakage current		-1	—	1	µA

NOTES:
1. The total current sunk by all I/Os must be limited to 200 mA.
2. The total current sourced by all I/Os must be limited to 160 mA.

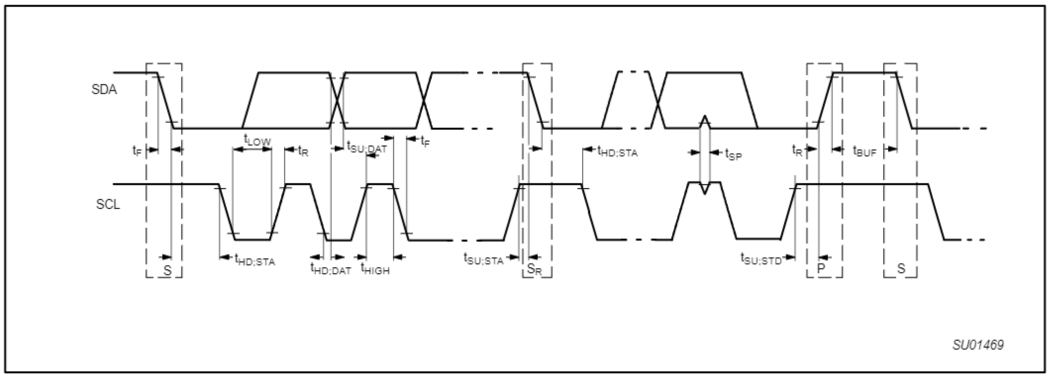
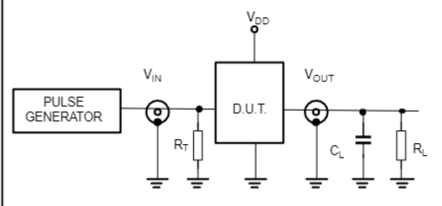


Figure 14. Definition of timing

AC CHARACTERISTICS

SYMBOL	PARAMETER	STANDARD MODE I²C BUS		FAST MODE I²C BUS		UNITS
		MIN	MAX	MIN	MAX	
f_{SCL}	Operating frequency	0	100	0	400	kHz
t_{BUF}	Bus free time between STOP and START conditions	4.7	—	1.3	—	µs
$t_{HD;STA}$	Hold time after (repeated) START condition	4.0	—	0.6	—	µs
$t_{SU;STA}$	Repeated START condition setup time	4.7	—	0.6	—	µs
$t_{SU;STO}$	Set-up time for STOP condition	4.0	—	0.6	—	µs
$t_{VD;ACK}$	Valid time of ACK condition[2]	0.3	3.45	0.1	0.9	µs
$t_{HD;DAT}$	Data in hold time	0	—	0	—	ns
$t_{VD;DAT}$	Data out valid time[3]	300	—	50	—	ns
$t_{SU;DAT}$	Data set-up time	250	—	100	—	ns
t_{LOW}	Clock LOW period	4.7	—	1.3	—	µs
t_{HIGH}	Clock HIGH period	4.0	—	0.6	—	µs
t_F	Clock/Data fall time	—	300	$20 + 0.1C_b$ [1]	300	ns
t_R	Clock/Data rise time	—	1000	$20 + 0.1C_b$ [1]	300	ns
t_{SP}	Pulse width of spikes that must be suppressed by the input filters	—	50	—	50	ns
Port Timing						
t_{PV}	Output data valid	—	200	—	200	ns
t_{PS}	Input data set-up time	150	—	150	—	ns
t_{PH}	Input data hold time	1	—	1	—	µs
Interrupt Timing						
t_{IV}	Interrupt valid	—	4	—	4	µs
t_{IR}	Interrupt reset	—	4	—	4	µs

NOTES:
1. C_b = total capacitance of one bus line in pF.
2. $t_{VD;ACK}$ = time for Acknowledgement signal from SCL LOW to SDA (out) LOW.
3. $t_{VD;DAT}$ = minimum time for SDA data out to be valid following SCL LOW.
4. t_{PV} measured from $0.7V_{DD}$ on SCL to 50% I/O output.



TEST CIRCUIT FOR OUTPUTS

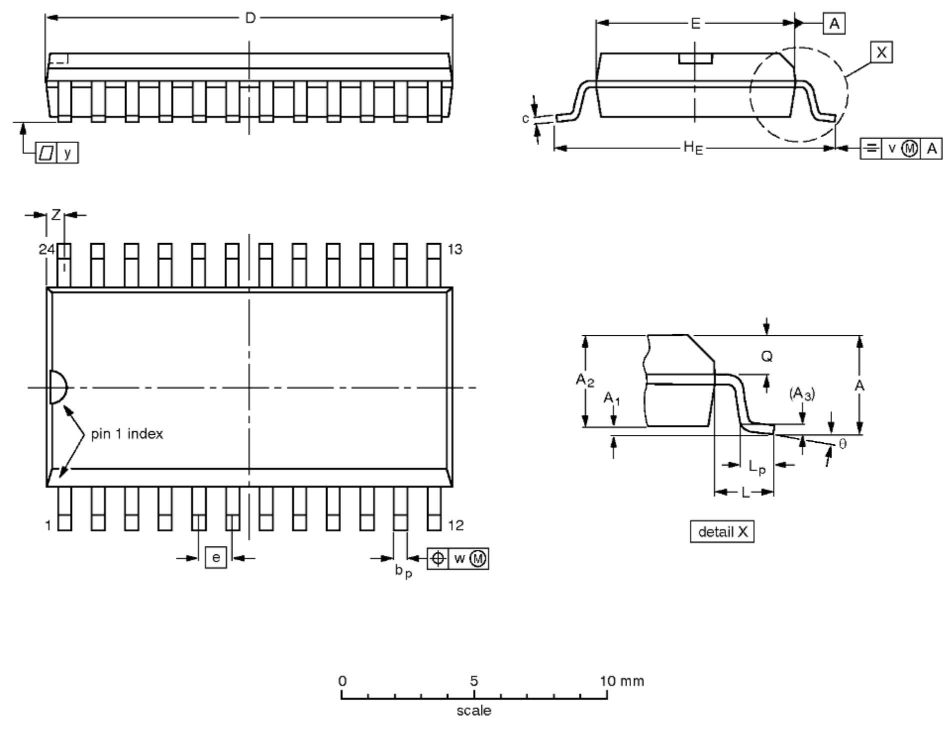
DEFINITIONS

R_L =	1 kΩ
C_L =	50 pF
R_T =	Termination resistance should be equal to Z_{OUT} of pulse generators.

su01760

Figure 15. t_{PV} set-up conditions

SO24: plastic small outline package; 24 leads; body width 7.5 mm **SOT137-1**



DIMENSIONS (inch dimensions are derived from the original mm dimensions)

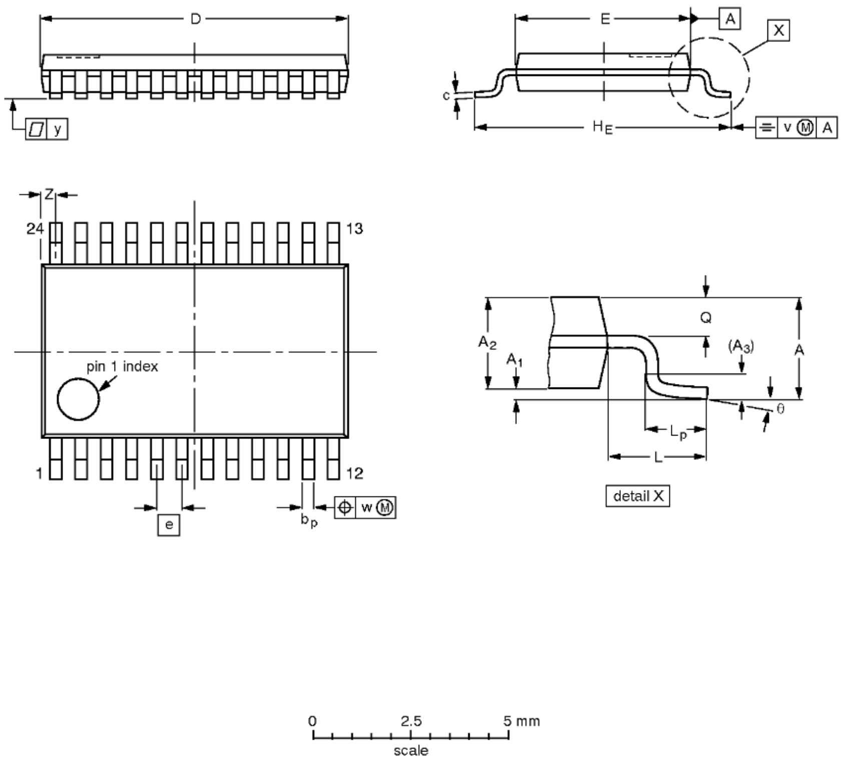
UNIT	A max.	A_1	A_2	A_3	b_p	c	D (1)	E (1)	e	H_E	L	L_p	Q	v	w	y	Z (1)	θ
mm	2.65	0.3 0.1	2.45 2.25	0.25	0.49 0.36	0.32 0.23	15.6 15.2	7.6 7.4	1.27	10.65 10.00	1.4	1.1 0.4	1.1 1.0	0.25	0.25	0.1	0.9 0.4	8° 0°
inches	0.1	0.012 0.004	0.096 0.089	0.01	0.019 0.014	0.013 0.009	0.61 0.60	0.30 0.29	0.05	0.419 0.394	0.055	0.043 0.016	0.043 0.039	0.01	0.01	0.004	0.035 0.016	

Note

1. Plastic or metal protrusions of 0.15 mm (0.006 inch) maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	JEITA			
SOT137-1	075E05	MS-013				~~99-12-27~~ 03-02-19

TSSOP24: plastic thin shrink small outline package; 24 leads; body width 4.4 mm **SOT355-1**



DIMENSIONS (mm are the original dimensions)

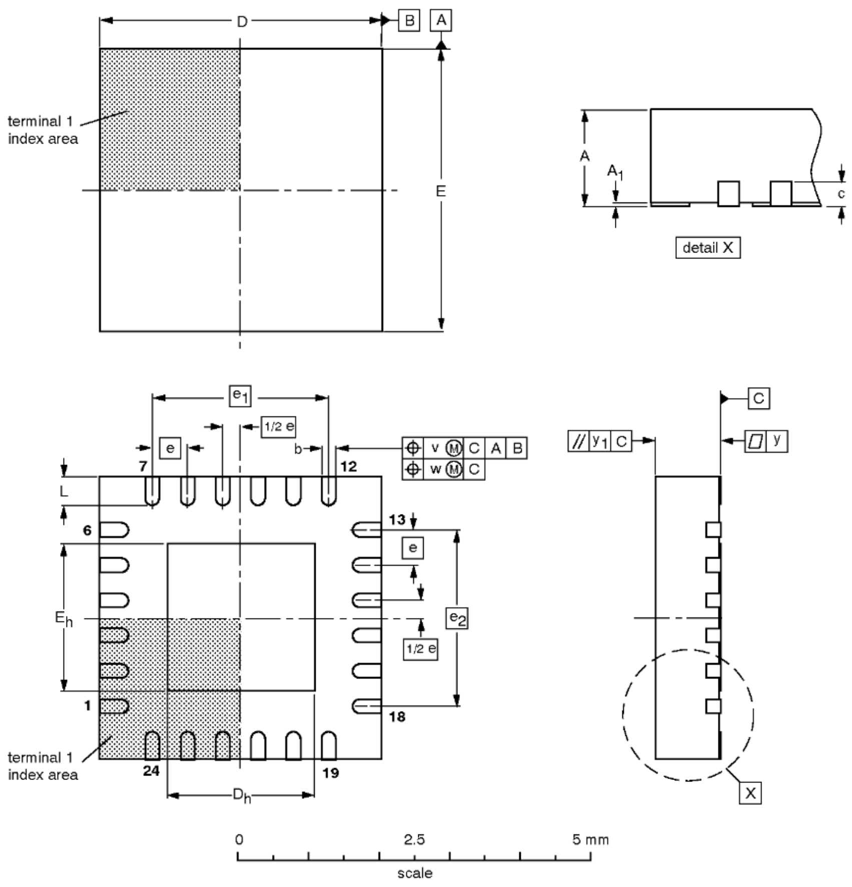
UNIT	A max.	A_1	A_2	A_3	b_p	c	D(1)	E(2)	e	H_E	L	L_p	Q	v	w	y	Z(1)	θ
mm	1.1	0.15 0.05	0.95 0.80	0.25	0.30 0.19	0.2 0.1	7.9 7.7	4.5 4.3	0.65	6.6 6.2	1	0.75 0.50	0.4 0.3	0.2	0.13	0.1	0.5 0.2	8° 0°

Notes

1. Plastic or metal protrusions of 0.15 mm maximum per side are not included.
2. Plastic interlead protrusions of 0.25 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	JEITA			
SOT355-1		MO-153				~~99-12-27~~ 03-02-19

HVQFN24: plastic thermal enhanced very thin quad flat package; no leads; 24 terminals; body 4 x 4 x 0.85 mm **SOT616-1**



DIMENSIONS (mm are the original dimensions)

UNIT	$A^{(1)}$ max.	A_1	b	c	$D^{(1)}$	D_h	$E^{(1)}$	E_h	e	e_1	e_2	L	v	w	y	y_1
mm	1	0.05 0.00	0.30 0.18	0.2	4.1 3.9	2.25 1.95	4.1 3.9	2.25 1.95	0.5	2.5	2.5	0.5 0.3	0.1	0.05	0.05	0.1

Note

1. Plastic or metal protrusions of 0.075 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	JEITA			
SOT616-1	- - -	MO-220	- - -			~~01-08-08~~ 02-10-22

REVISION HISTORY

Rev	Date	Description
_1	20030627	**Product data (9397 750 11681); ECN 853-2430 30019 dated 11 June 2003. Initial version**

Data sheet status

Level	Data sheet status[1]	Product status[2] [3]	Definitions
I	Objective data	Development	This data sheet contains data from the objective specification for product development. Philips Semiconductors reserves the right to change the specification in any manner without notice.
II	Preliminary data	Qualification	This data sheet contains data from the preliminary specification. Supplementary data will be published at a later date. Philips Semiconductors reserves the right to change the specification without notice, in order to improve the design and supply the best possible product.
III	Product data	Production	This data sheet contains data from the product specification. Philips Semiconductors reserves the right to make changes at any time in order to improve the design, manufacturing and supply. Relevant changes will be communicated via a Customer Product/Process Change Notification (CPCN).

[1] Please consult the most recently issued data sheet before initiating or completing a design.

[2] The product status of the device(s) described in this data sheet may have changed since this data sheet was published. The latest information is available on the Internet at URL http://www.semiconductors.philips.com.

[3] For data sheets describing multiple type numbers, the highest-level product status determines the data sheet status.

Definitions

Short-form specification — The data in a short-form specification is extracted from a full data sheet with the same type number and title. For detailed information see the relevant data sheet or data handbook.

Limiting values definition — Limiting values given are in accordance with the Absolute Maximum Rating System (IEC 60134). Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the Characteristics sections of the specification is not implied. Exposure to limiting values for extended periods may affect device reliability.

Application information — Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors make no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Disclaimers

Contact information

For additional information please visit
http://www.semiconductors.philips.com. Fax: **+31 40 27 24825**

For sales offices addresses send e-mail to:
sales.addresses@www.semiconductors.philips.com.

Date of release: 06-03

Document order number: 9397 750 11681

Let's make things better.

Philips Semiconductors

PHILIPS

SCORBOT-ER IX

User's Manual

2nd Edition

Catalog # 100066 Rev.B

ESHED ROBOTEC

WARNING!

**The SCORBOT ROBOT is DANGEROUS
and can cause severe injury.**

USE WITH EXTREME CAUTION.

**Set up a protective screen or
guardrail
around the robot to
KEEP PEOPLE AWAY
from its working range.**

Copyright ©1996, 1999 by Eshed Robotec

Catalog #100066 Rev.B

ISBN 965-291-068-6

(March 1996) March 1999 Reprinted/PDF version

ESHED ROBOTEC INC.
444 East Industrial Park Drive
Manchester, NH 03109 USA
Tel: 1-800-777-6268
Tel: (603) 625-8600
Fax: (603) 625-2137

Table of Contents

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.

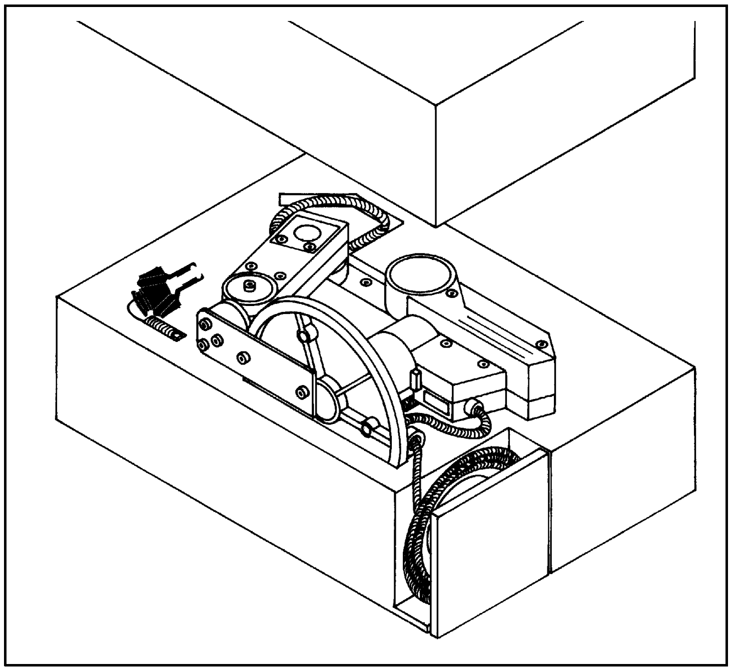


Figure 1-1: SCORBOT-ER IX in Packing

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 Range	Effective Speed	
Axis 1: Base rotation	270°	79°/sec	112°/sec
Axis 2: Shoulder rotation	145°	68°/sec	99°/sec
Axis 3: Elbow rotation	210°	76°/sec	112°/sec
Axis 4: Wrist pitch	196°	87°/sec	133°/sec
Axis 5: Wrist roll	737°	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

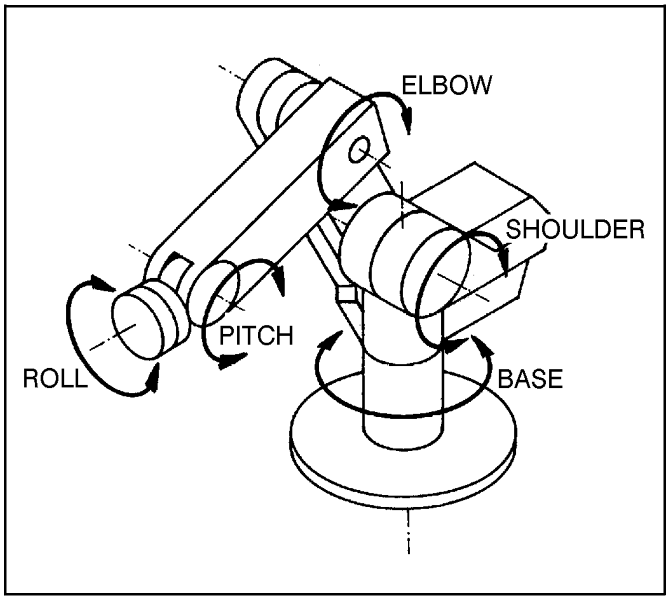


Figure 2-1: SCORBOT-ER IX Joints

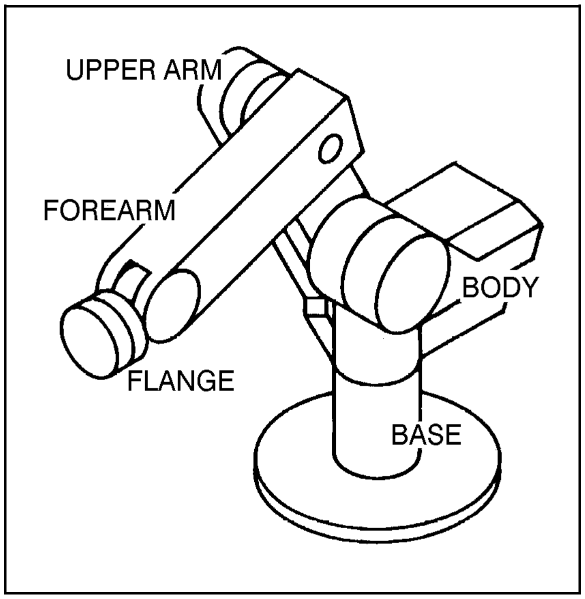


Figure 2-2: SCORBOT-ER IX Links

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.

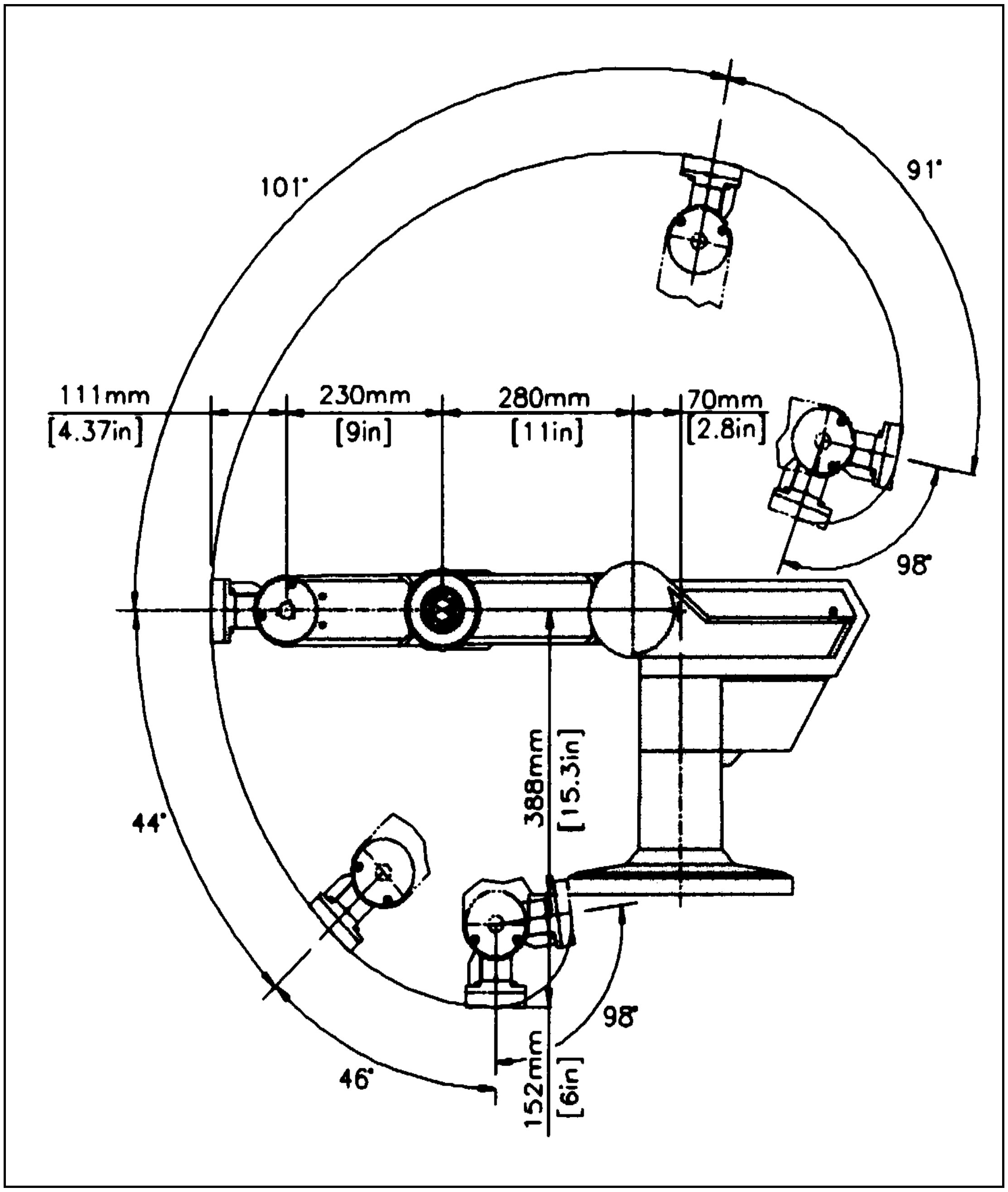


Figure 2-3: Operating Range (Side View)

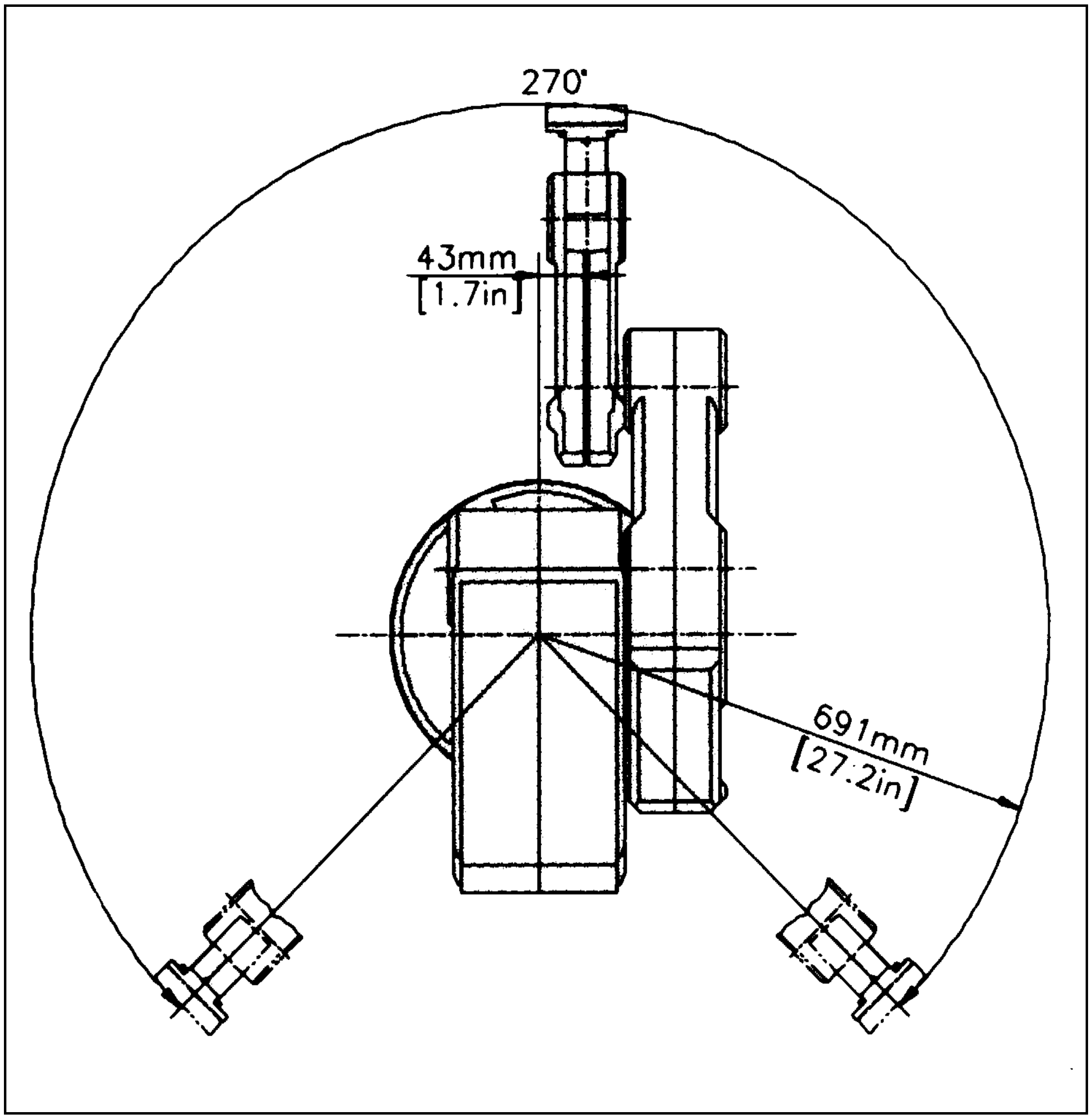


Figure 2-4: Operating Range (Top View)

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.

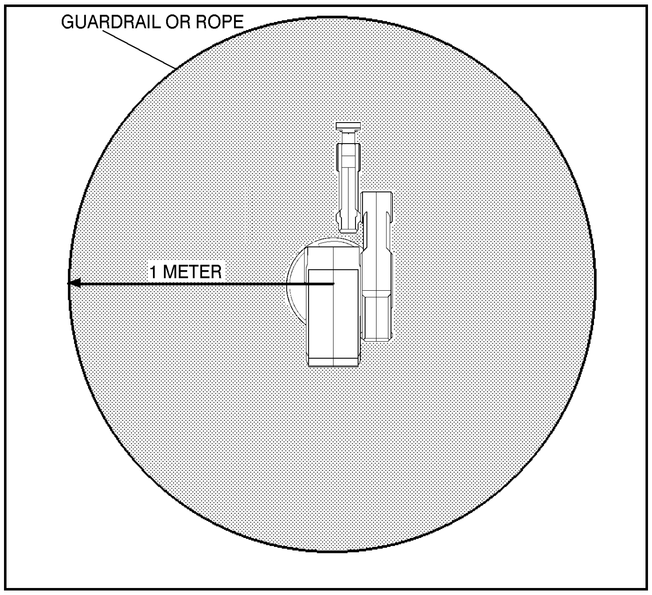


Figure 4-1: Robot Safety Range

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.

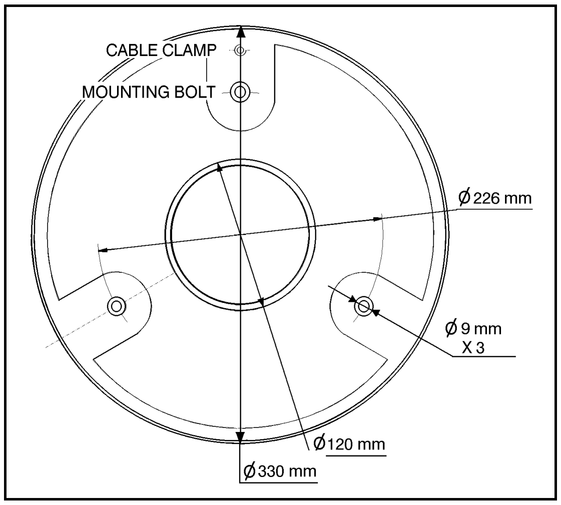


Figure 4-2: Robot Base Layout

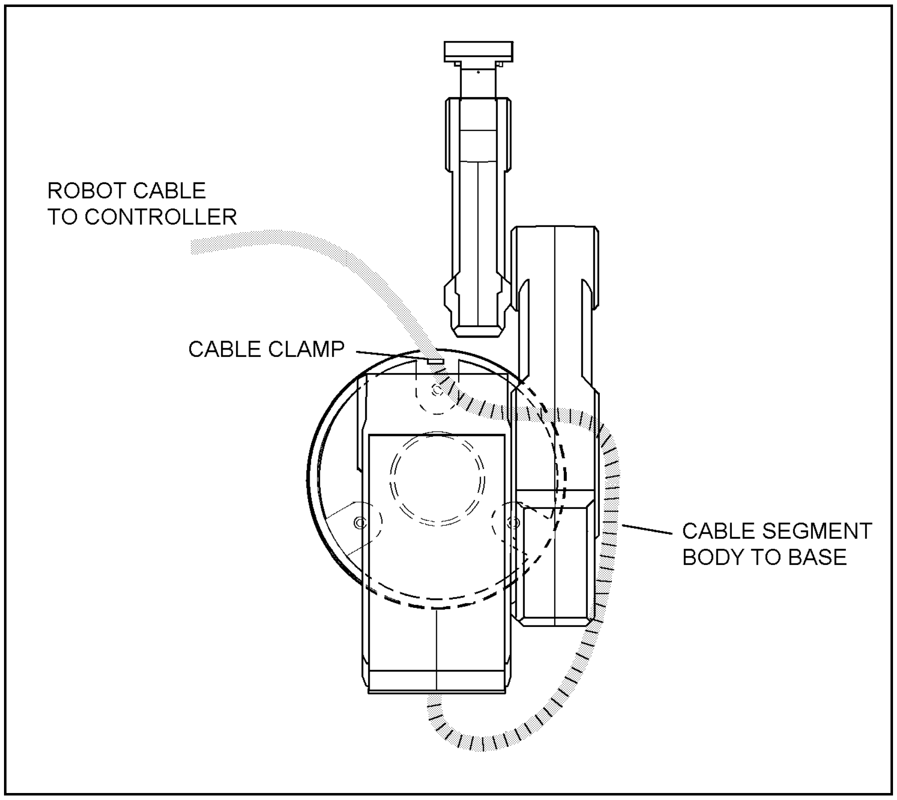


Figure 4-3: Robot Setup

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.

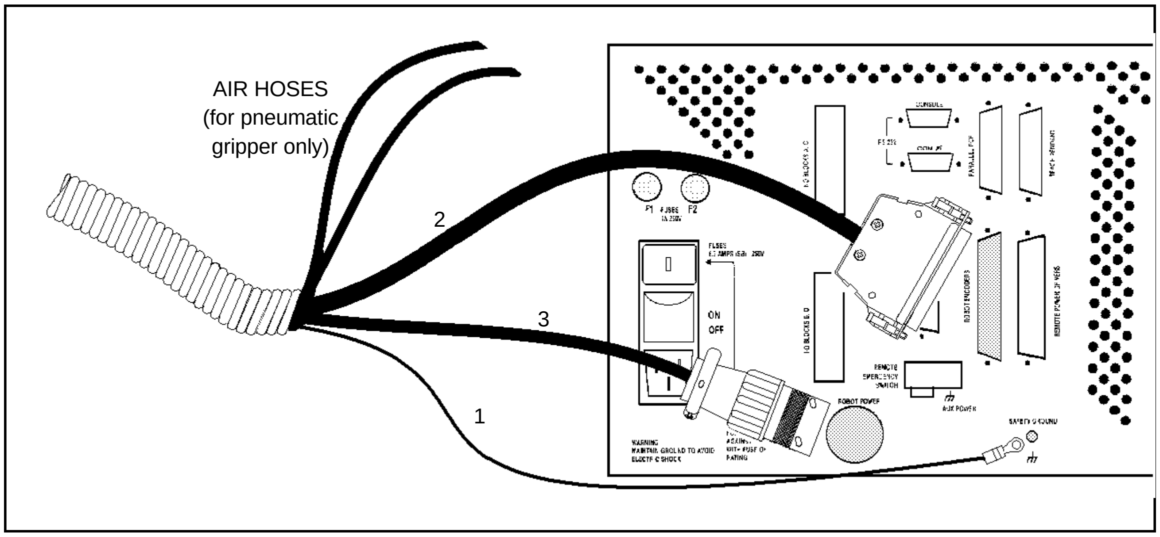


Figure 4-4: Robot—Controller Cable Connections

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.

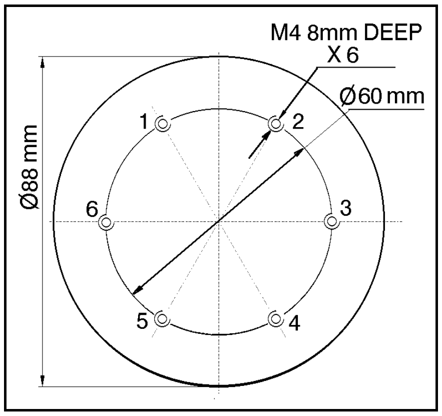


Figure 4-5: Gripper Mounting Flange Layout

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.*

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.

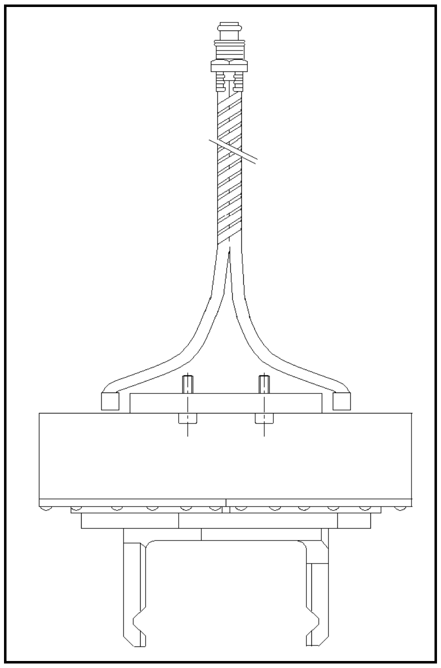


Figure 4-6: Pneumatic Gripper

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

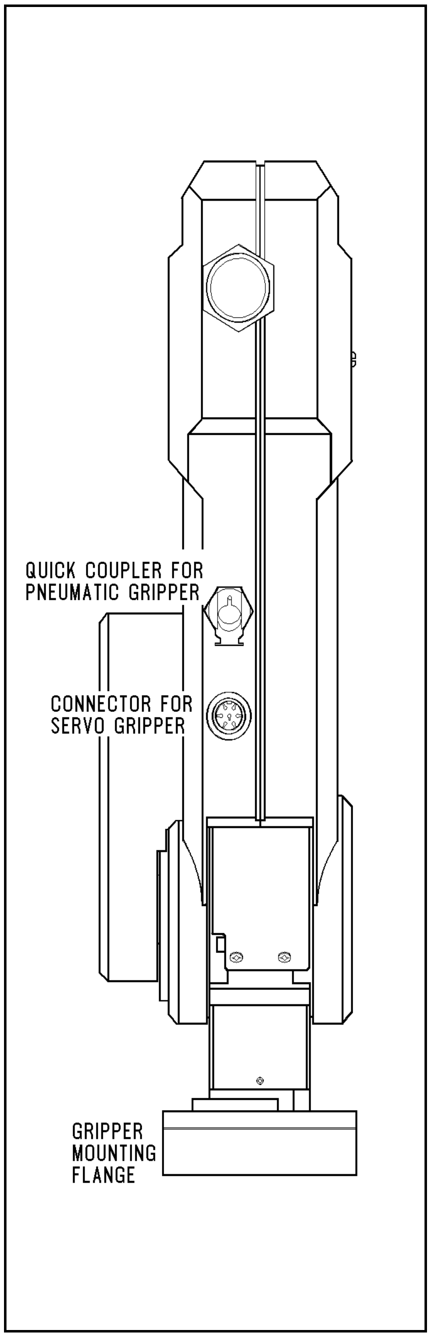

QUICK COUPLER FOR PNEUMATIC GRIPPER
CONNECTOR FOR SERVO GRIPPER
GRIPPER MOUNTING FLANGE

Figure 4-7: Gripper Connectors

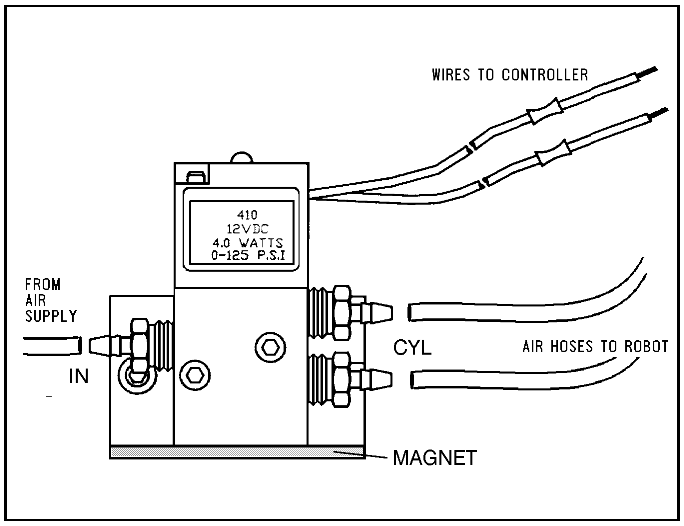


Figure 4-8: Pneumatic Solenoid Valve

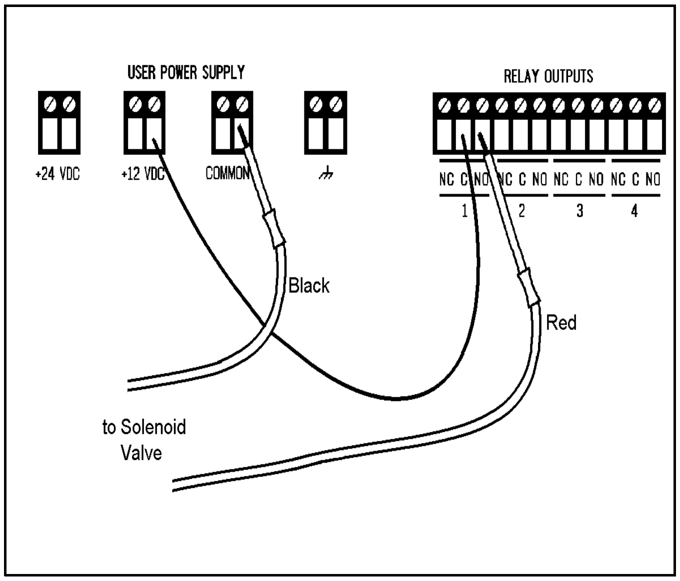


Figure 4-9: Valve—Controller Connections

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.*

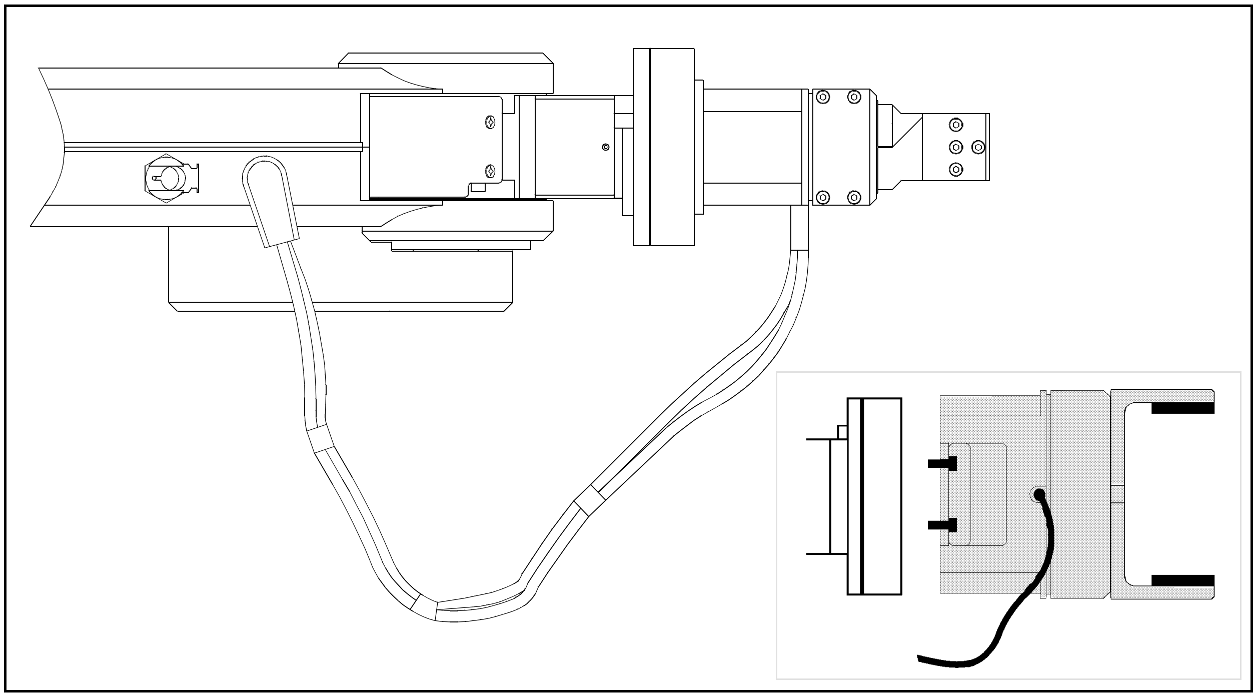


Figure 4-10: Connecting Gripper to SCORBOT-ER IX

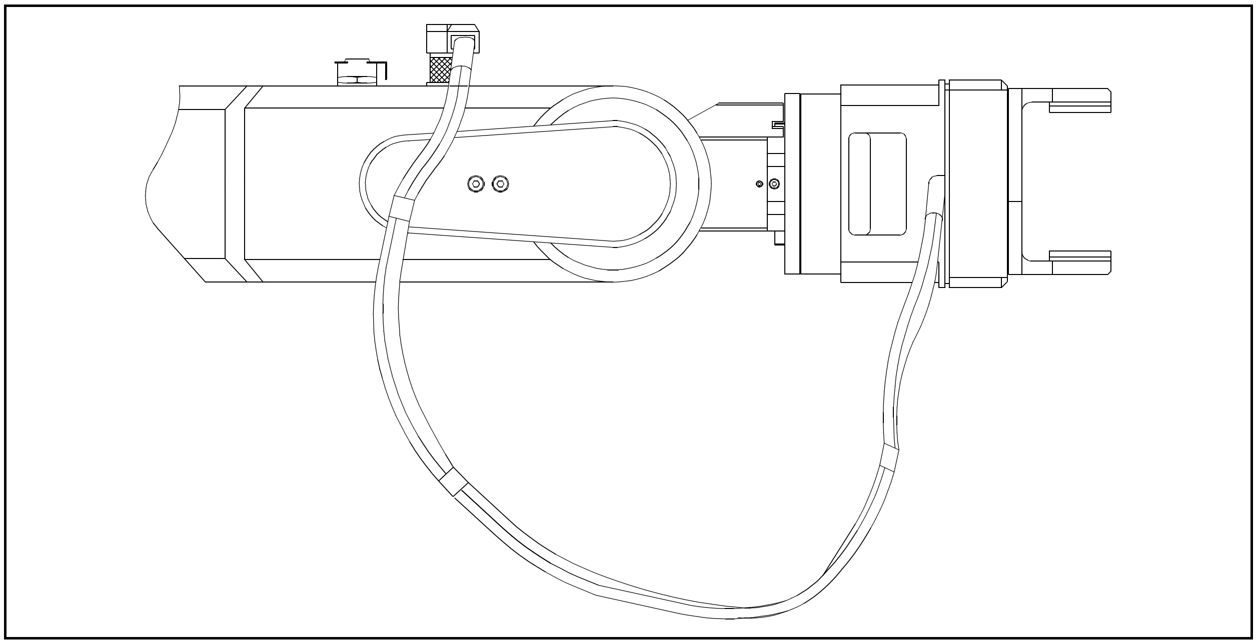


Figure 4-11: Connecting Gripper to SCORBOT-ER IX

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****.*

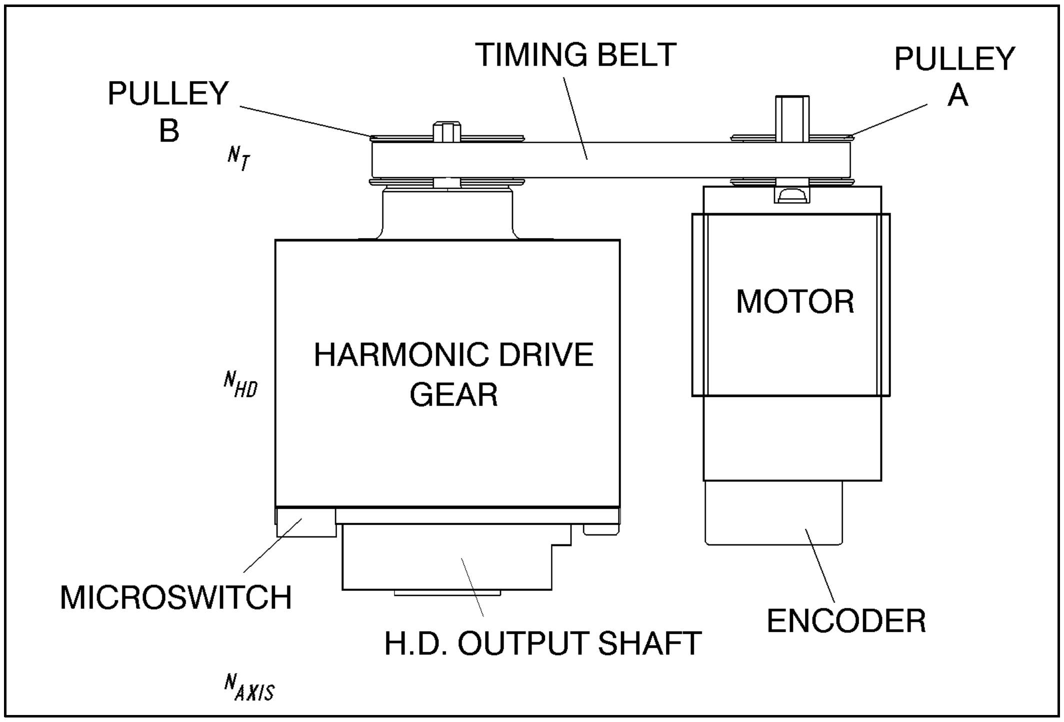


Figure 6-1: The SCORBOT-ER IX Drive System

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.

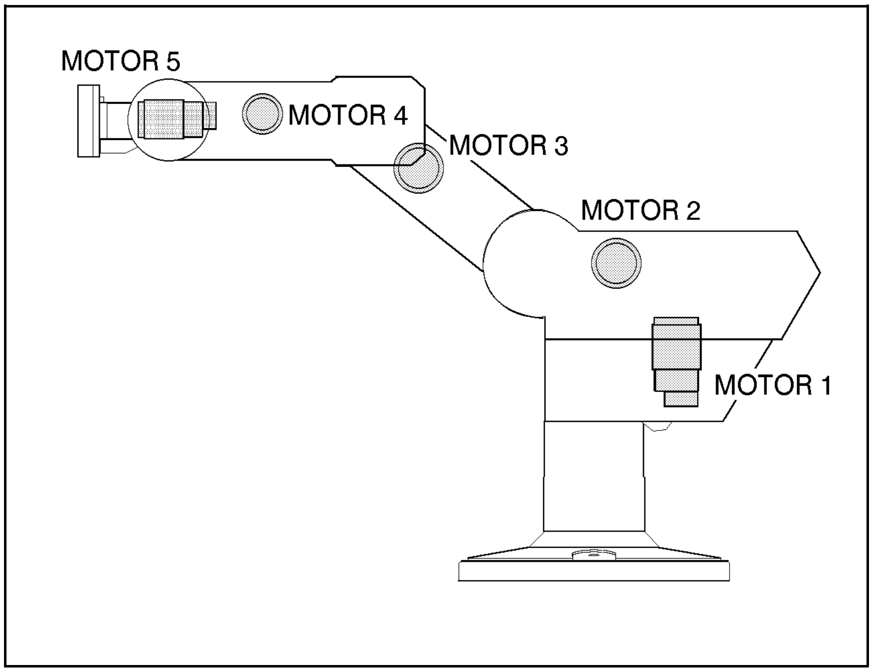


Figure 6-2: Motor Locations in SCORBOT-ER IX

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.

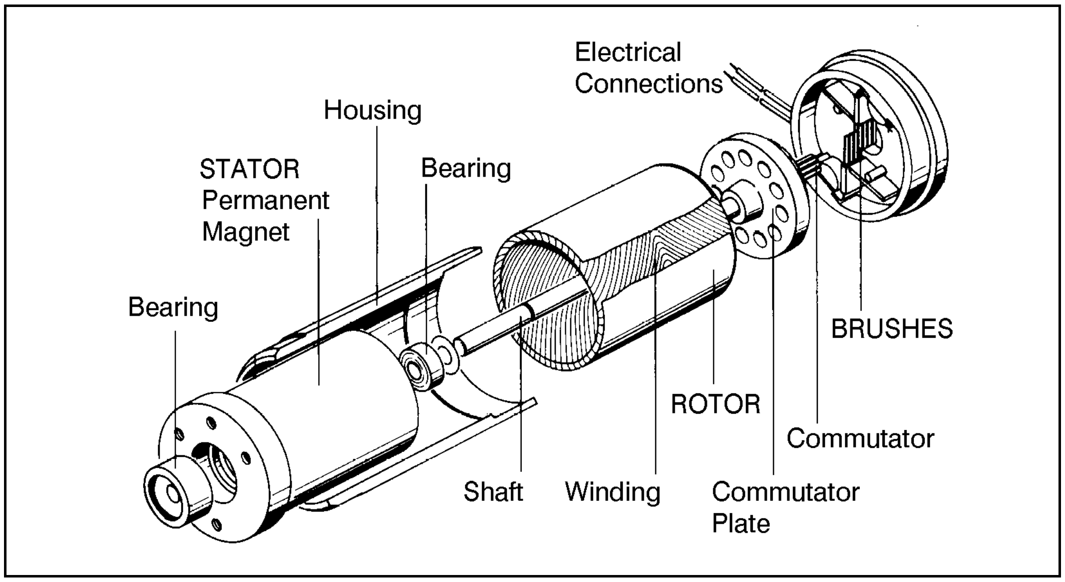


Figure 6-3: Basic Structure of a DC Motor

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

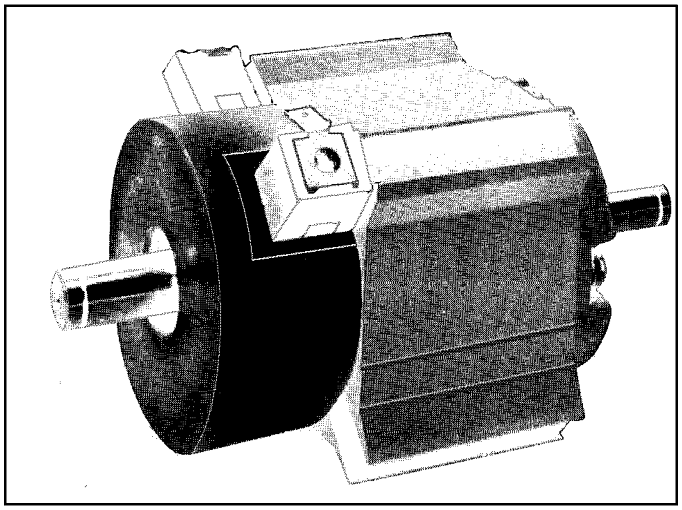


Figure 6-4: Motor on Axes 1, 2 and 3

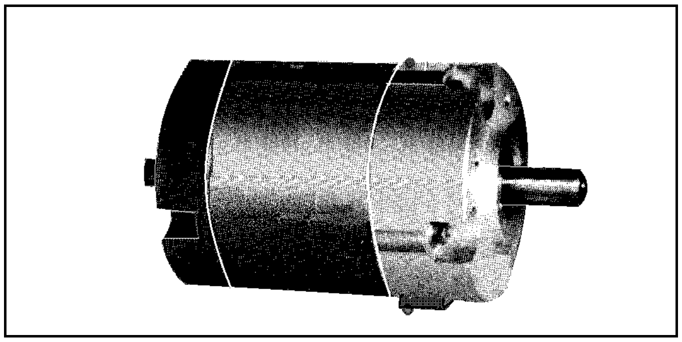


Figure 6-5: Motor on Axes 4 and 5

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.

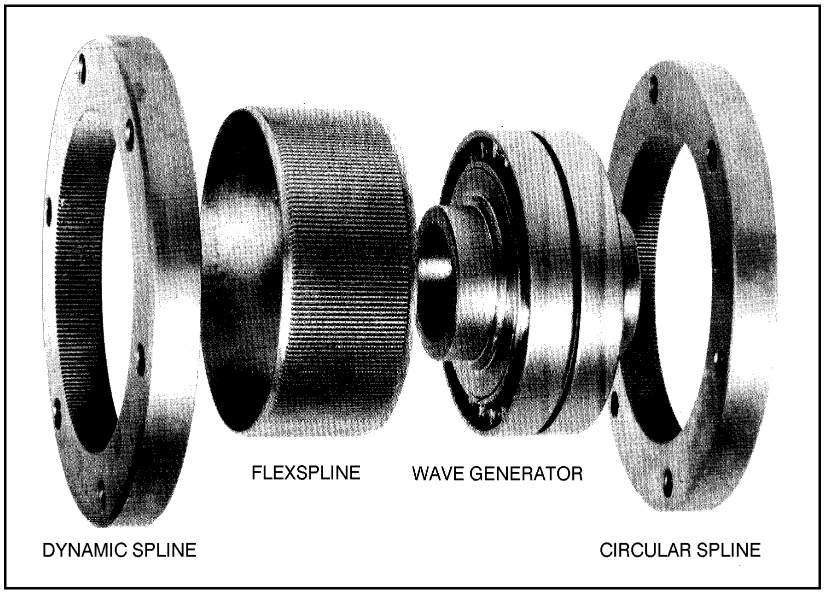


Figure 6-6: Harmonic Drive Structure

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:

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

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

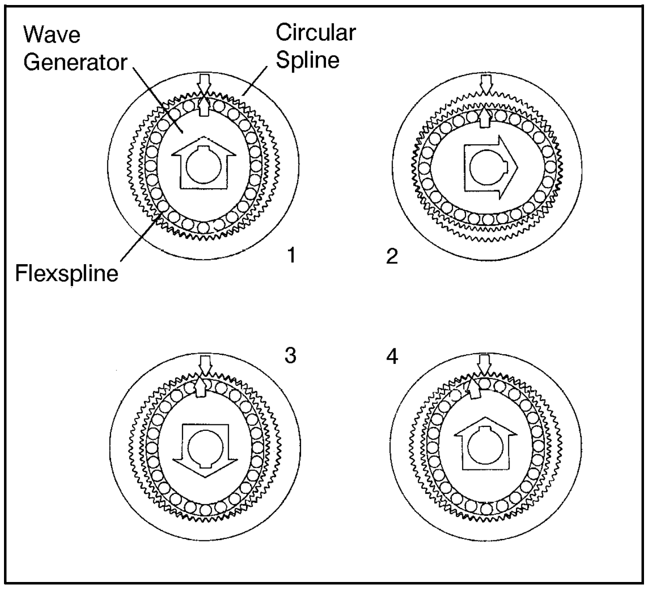


Figure 6-7: Operation of the Harmonic Drive

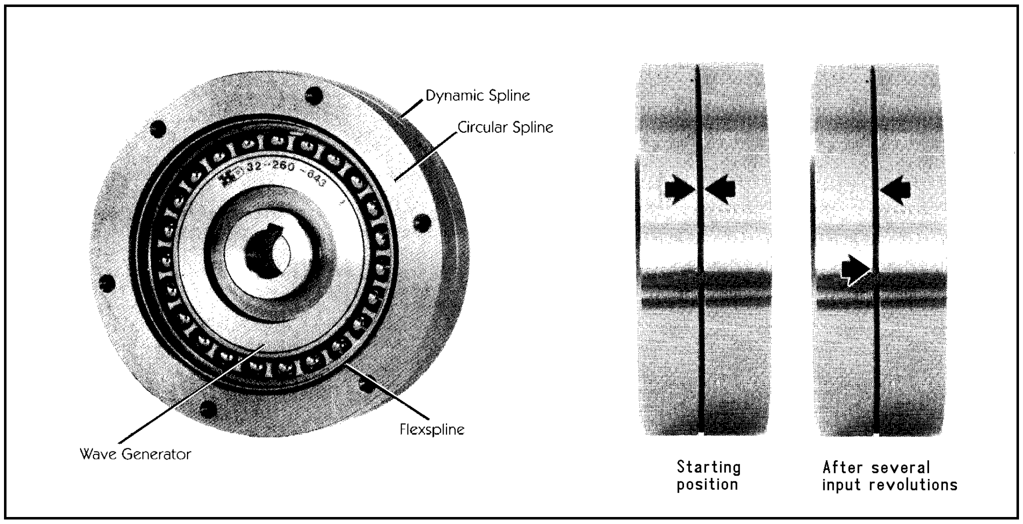


Figure 6-8: Operation of the Harmonic Drive

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.

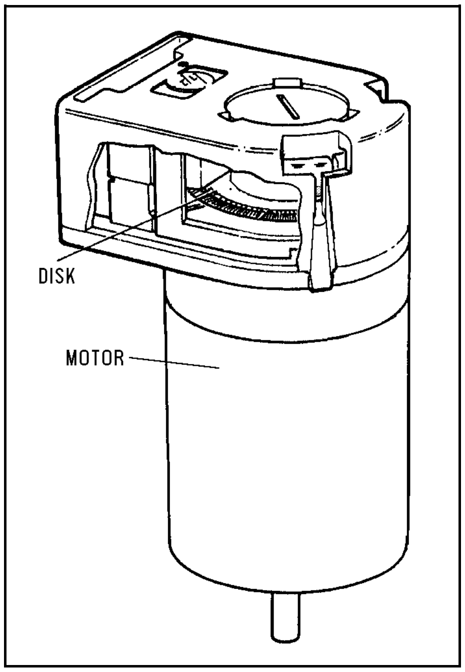


Figure 7-1:
SCORBOT-ER IX Encoder

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.

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.

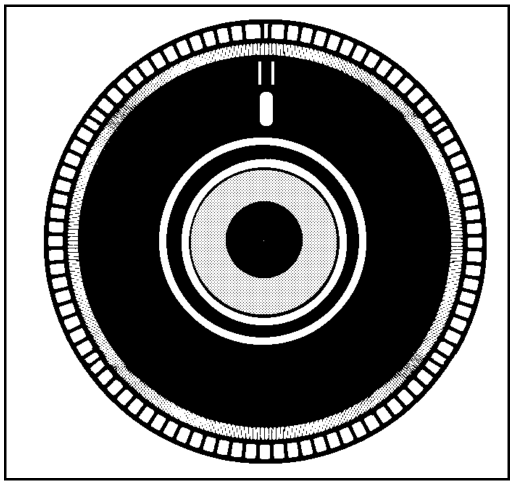


Figure 7-2:
SCORBOT-ER IX Encoder Disk

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.

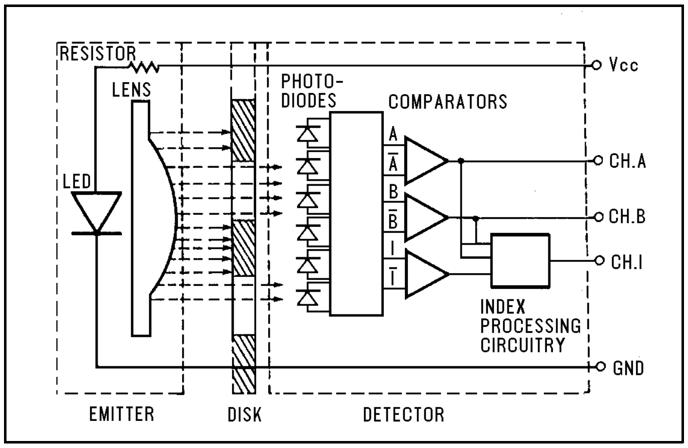


Figure 7-3: Encoder Circuitry

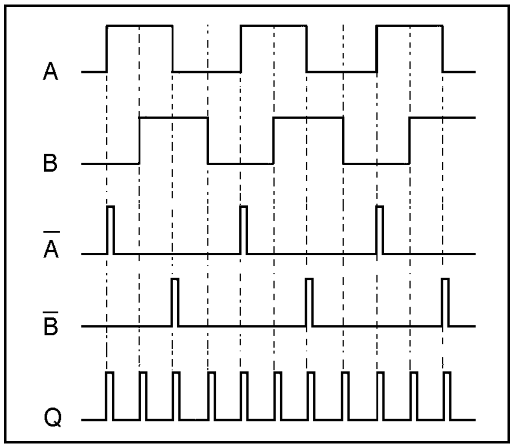


Figure 7-4: Encoder Output Signals

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.

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.

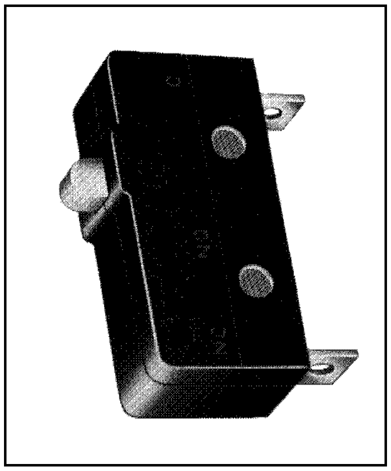


Figure 7-5: SCORBOT-ER IX Limit Switch

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.

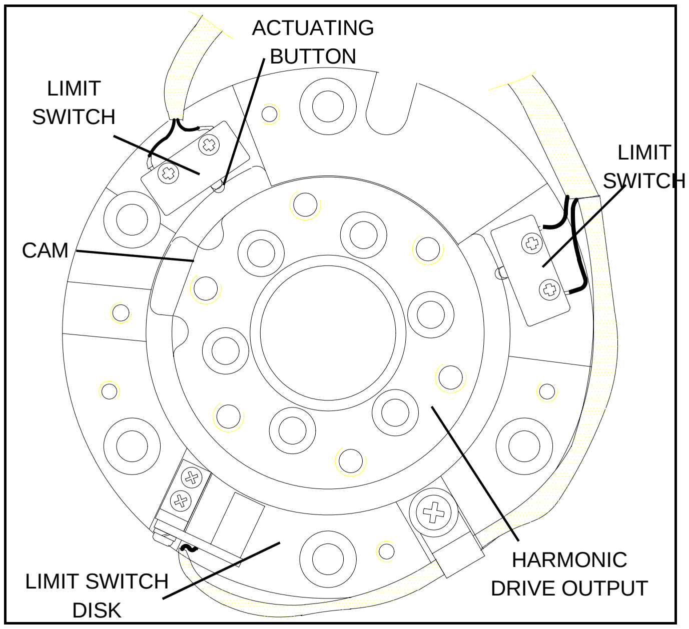


Figure 7-6: Limit Switch Activation

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.

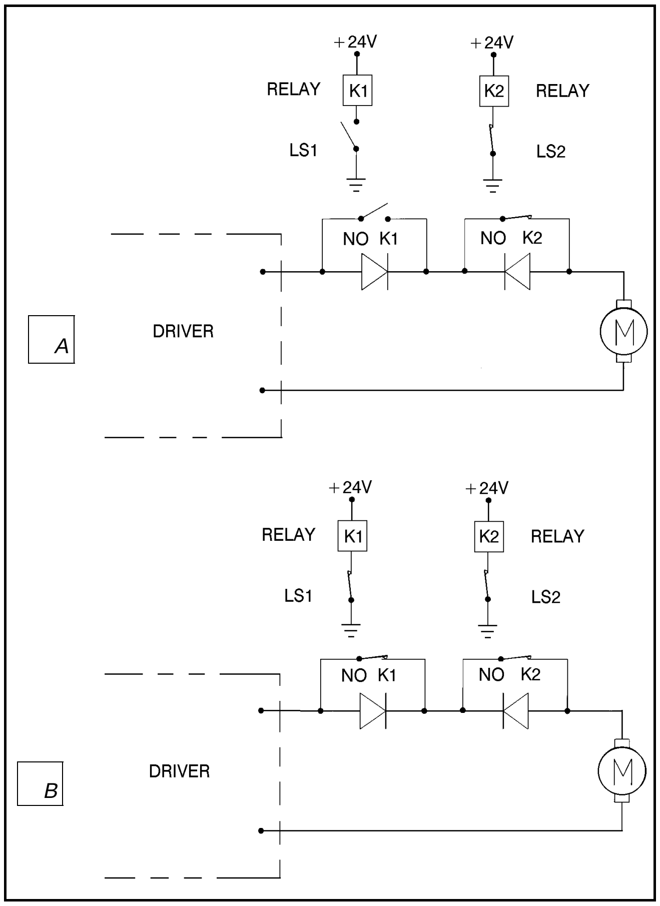


Figure 7-7: Axis Limit Circuit

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.

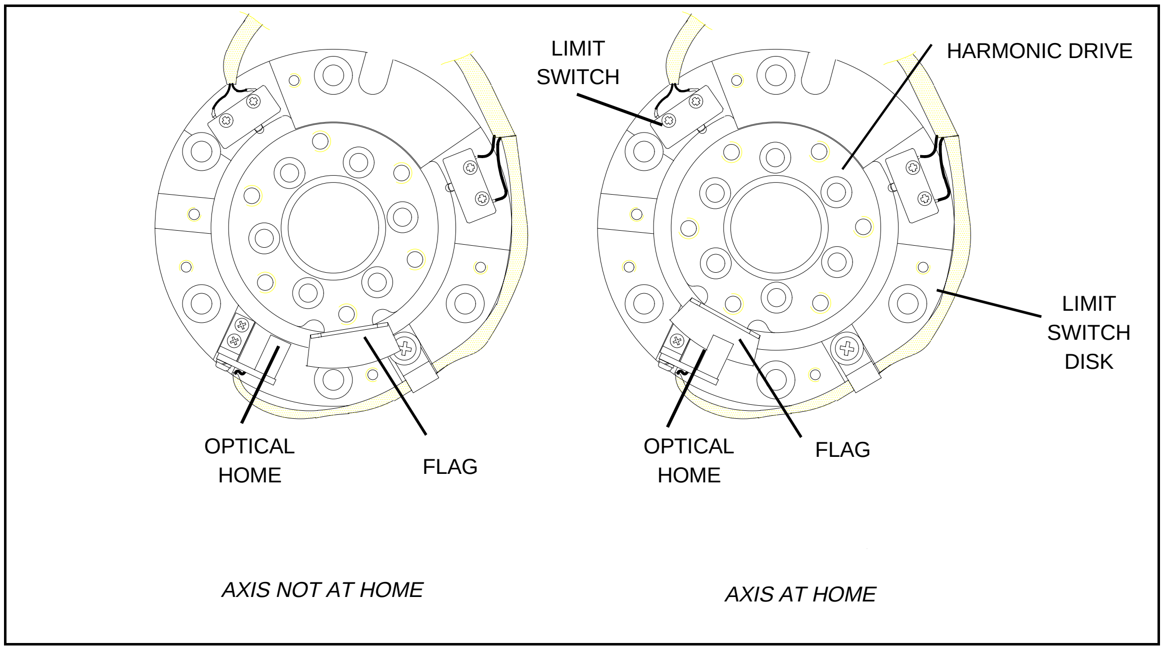


Figure 7-8: Home Switch Activation

CHAPTER 8

Wiring

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

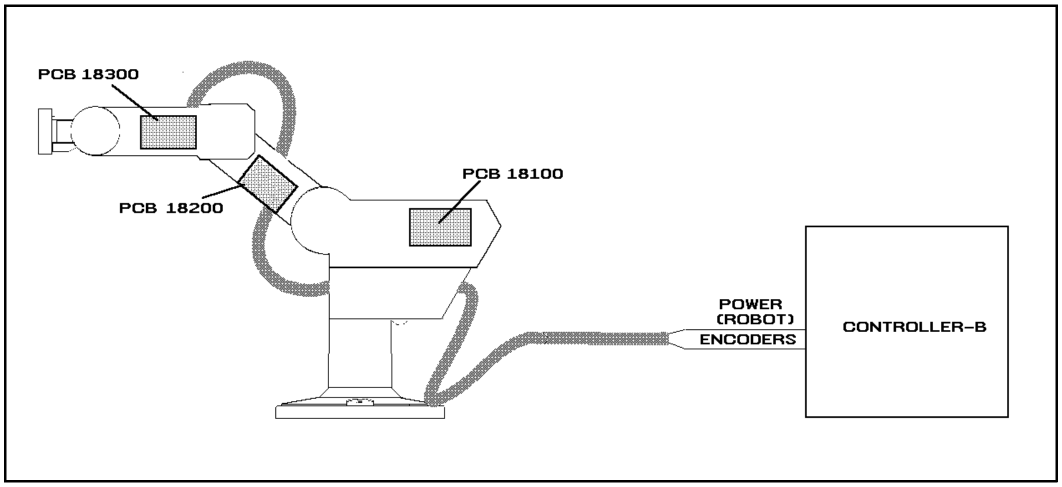


Figure 8-1: SCORBOT-ER IX Cabling

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.

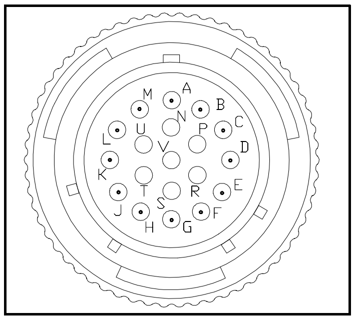


Figure 8-2: Burndy 19 Pin Connector

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.

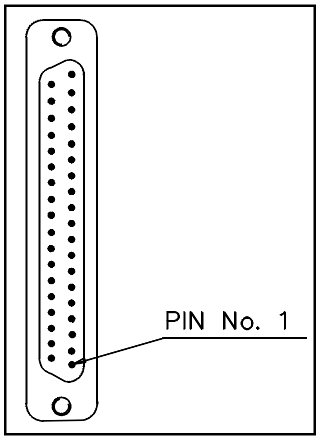


Figure 8-3: D37 Connector

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.

SCORBOT-ER V$_{plus}$
User's Manual

3rd Edition

Catalog # 100016 Rev.C

ESHED ROBOTEC

Read this manual thoroughly before attempting to install or operate the robot. If you have any problems during installation or operation, call your agent for assistance.

Save the original carton and all packing material. You may need them later for shipment.

ESHED ROBOTEC (1982) LTD.
13 Hamelacha St.
Afek Industrial Park
Rosh Ha'ayin 48091, Israel
Tel: (972) 3-900 4111
Fax: (972) 3-903 0411

ESHED ROBOTEC INC.
472 Amherst St.
Nashua, NH 03063, USA
Tel: 1-800-777-6268
Tel: (603) 579-9700
Fax: : (603) 579-9707

email: info@eshed.com
website: www.eshed.com

Table of Contents

List of Figures

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.

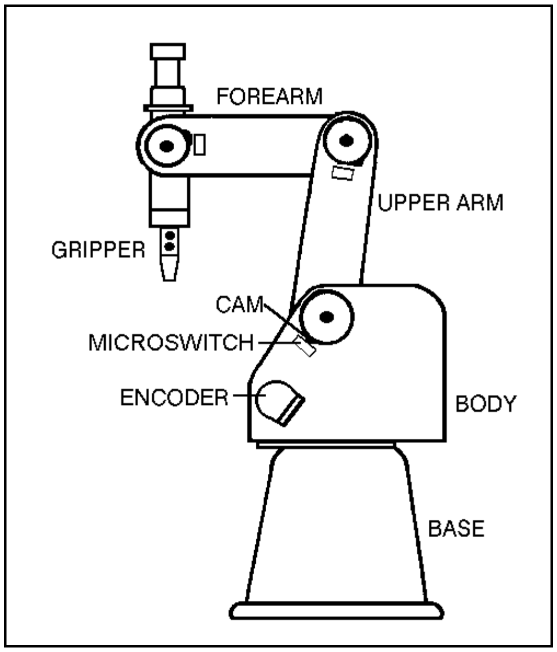


Figure 1-1: Robot Arm Parts

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.

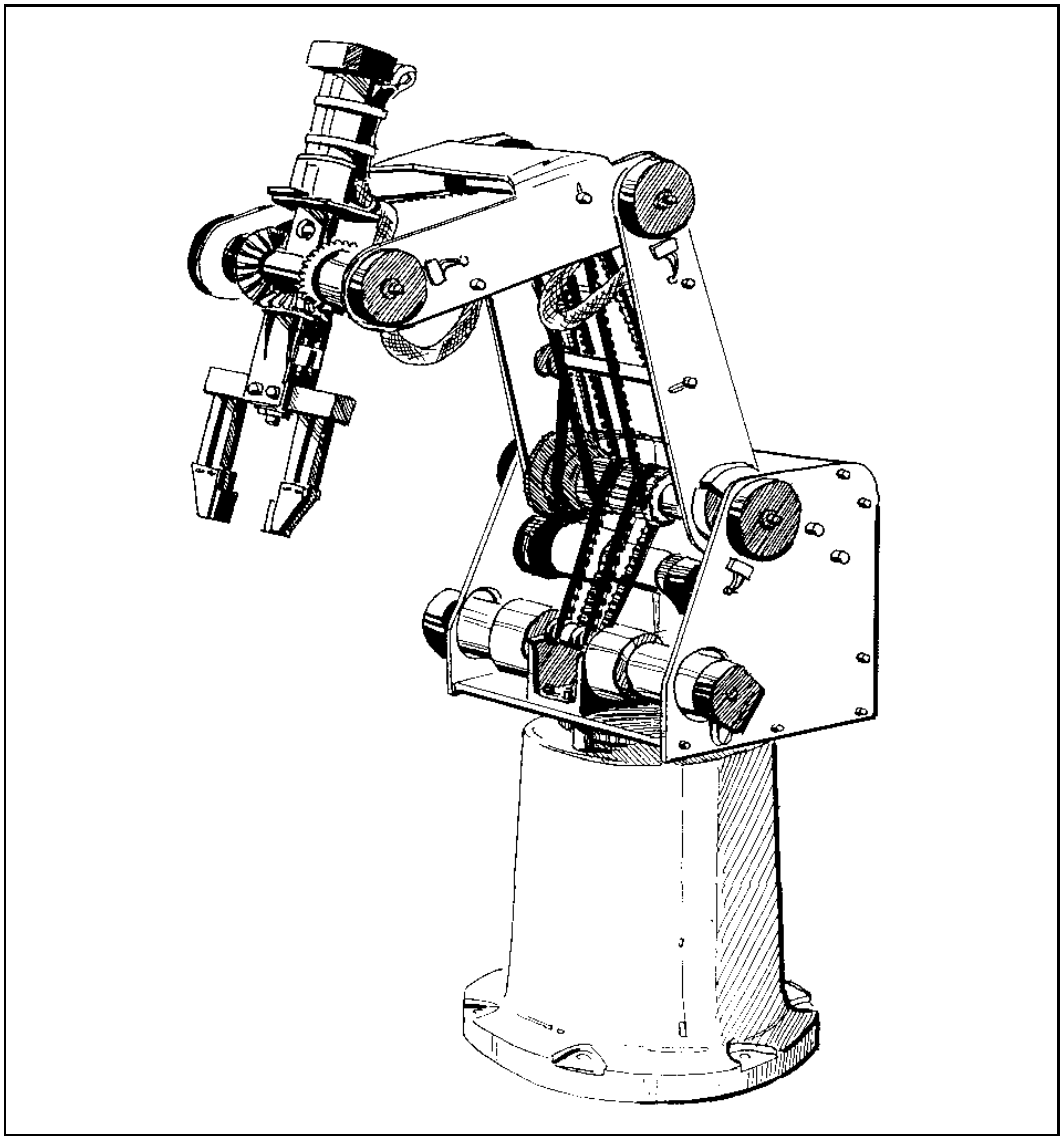


Figure 2-1: SCORBOT-ER Vplus Robot Arm

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

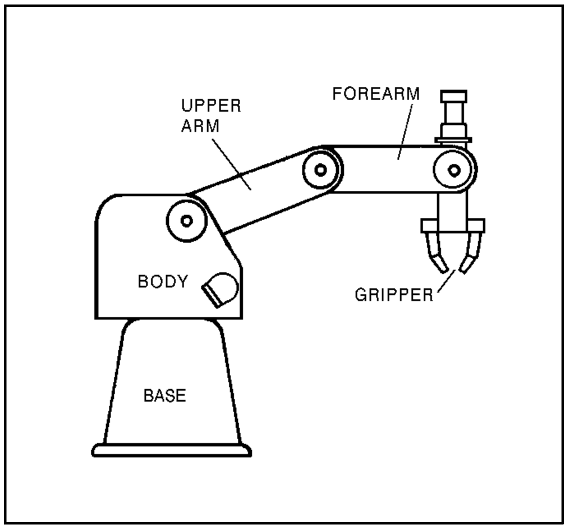


Figure 2-2: Robot Arm Links

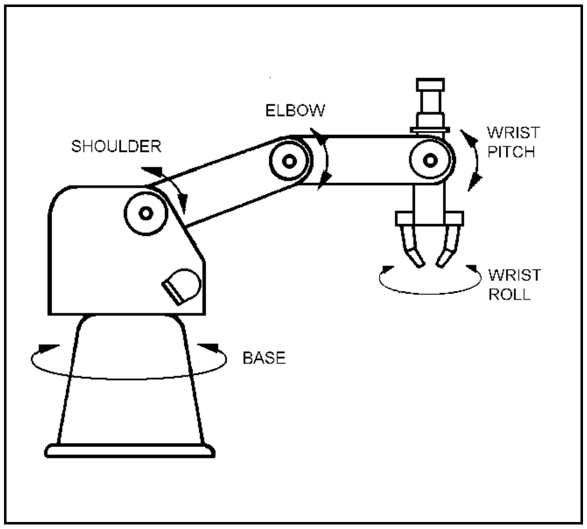


Figure 2-3: Robot Arm Joints

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.

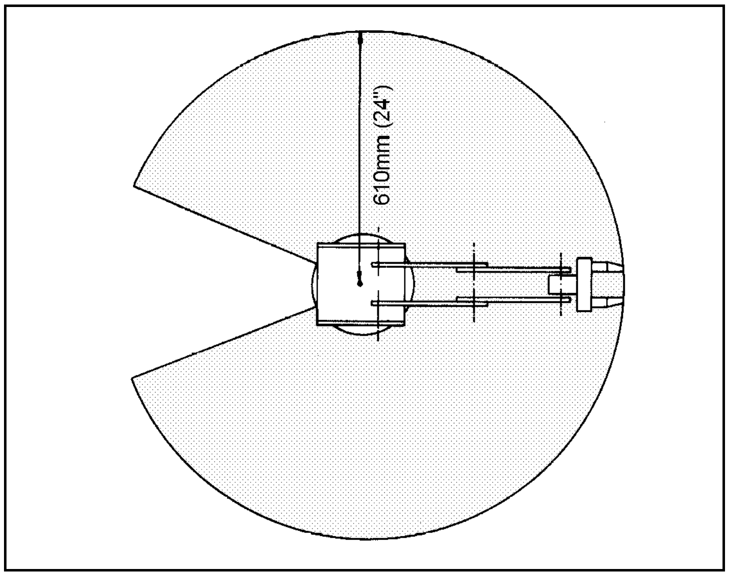


Figure 2-4: Operating Range (Top View)

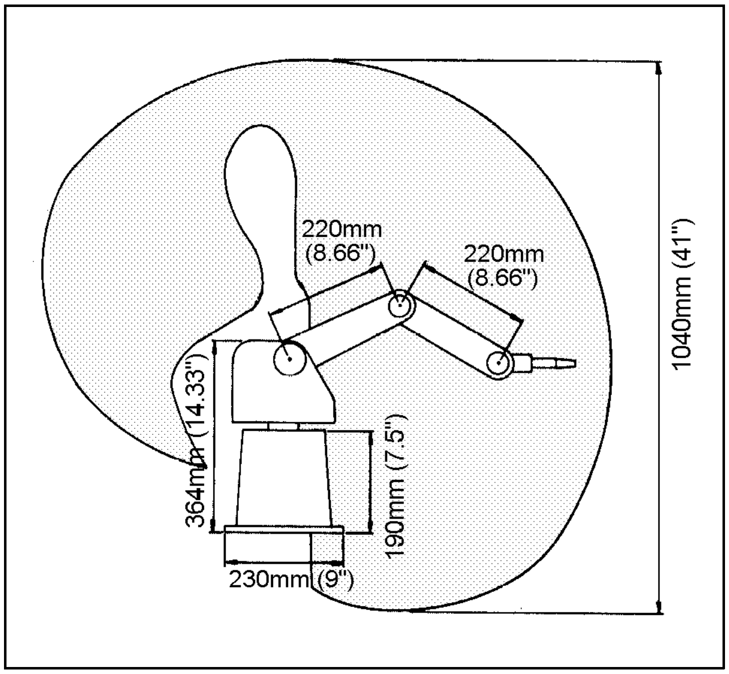


Figure 2-5: Operating Range (Side View)

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.

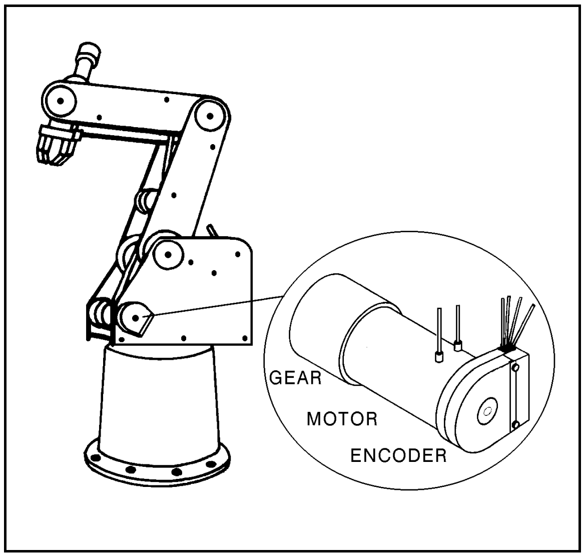


Figure 2-6: Motor

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.

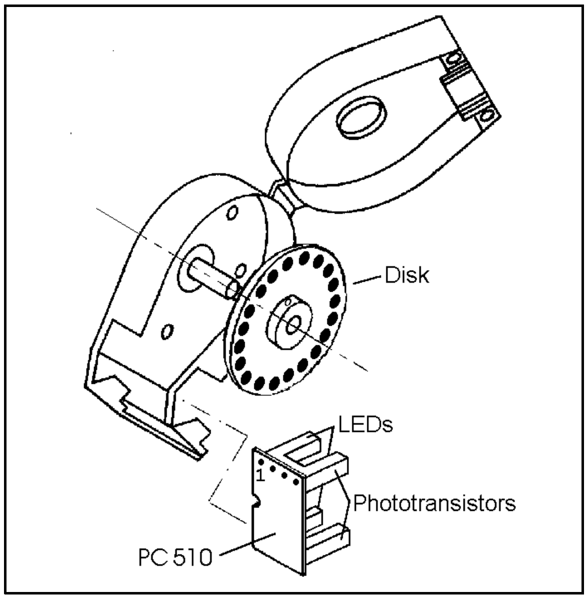


Figure 2-7: Encoder

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.

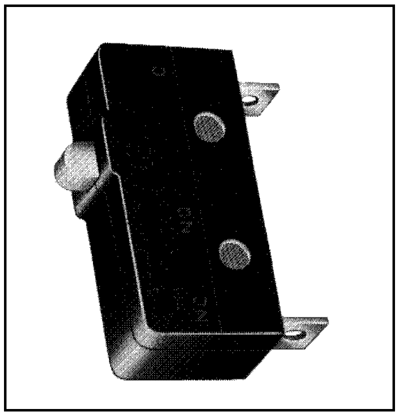


Figure 2-8: Microswitch

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.

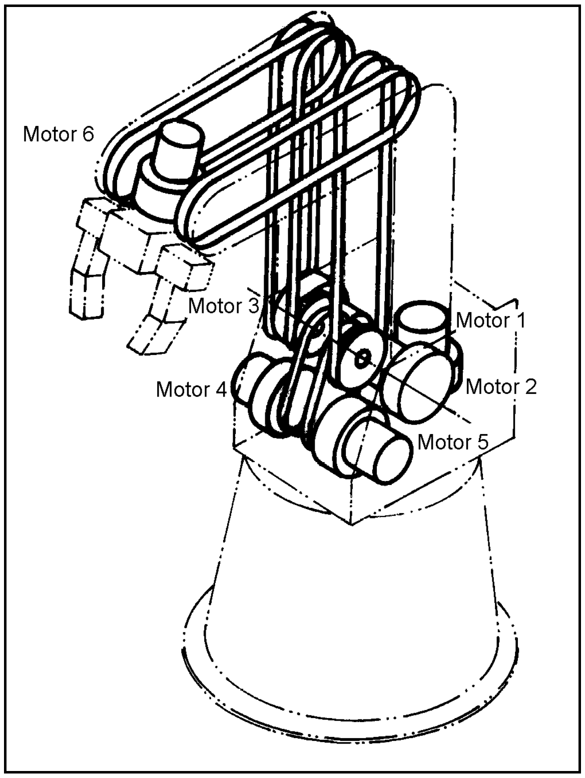


Figure 2-9: Transmissions

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.

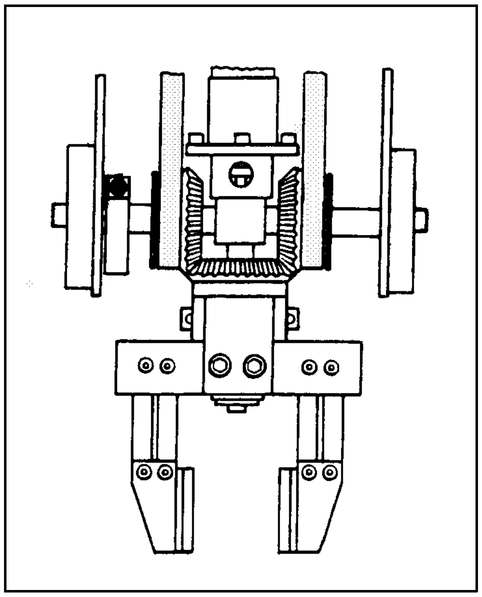


Figure 2-10: SCORBOT-ER Vplus Gripper

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.

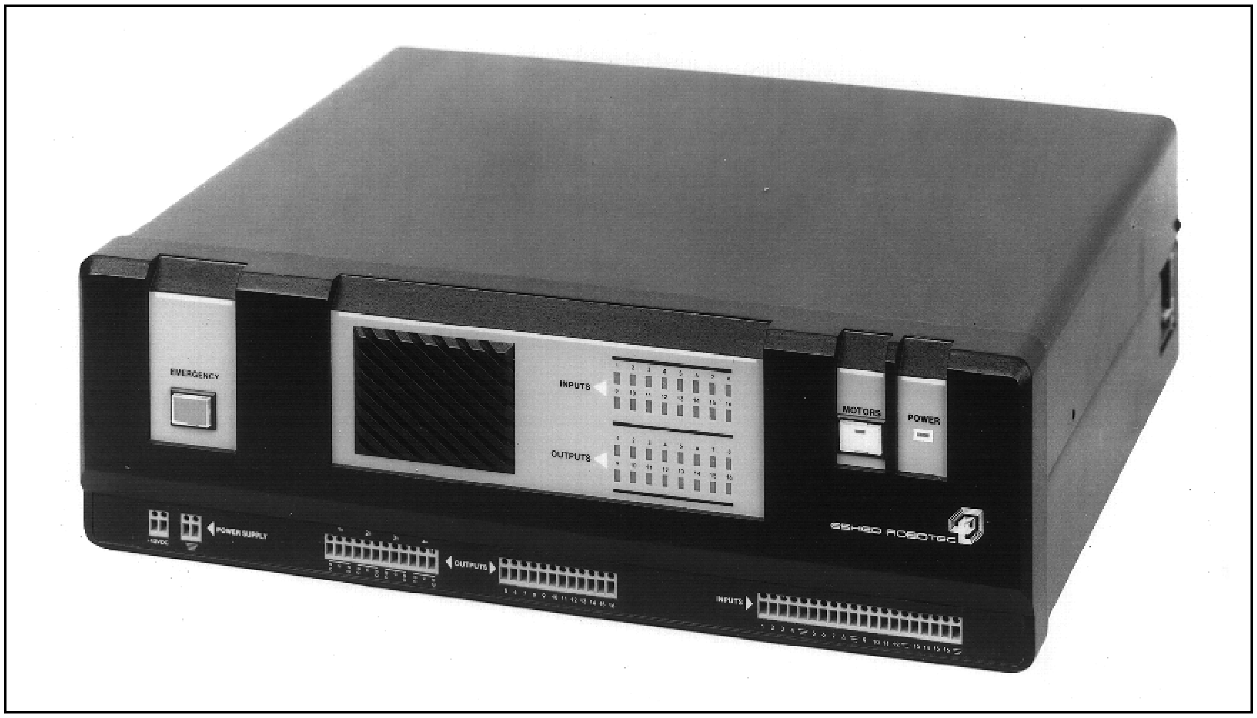


Figure 3-1: Controller-A

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.

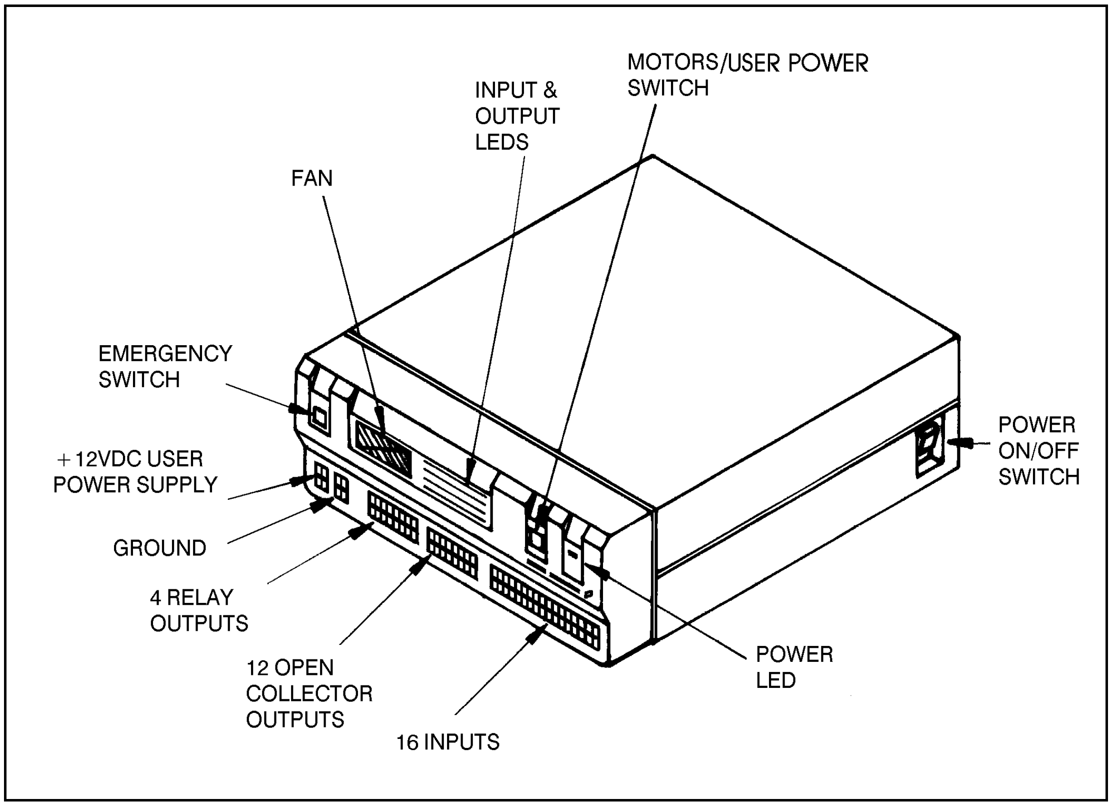


Figure 3-2: Controller Front Panel

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.

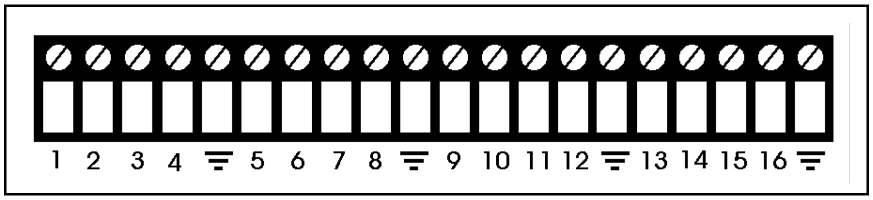


Figure 3-3: Input Terminals

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

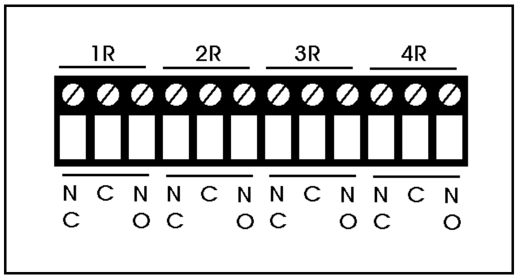


Figure 3-4: Relay Output Terminals

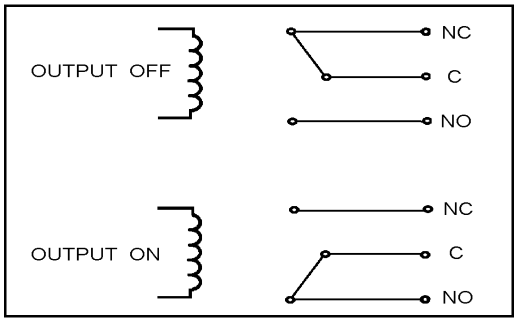


Figure 3-5: Relay Output States

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.

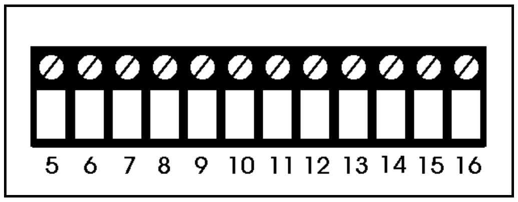


Figure 3-6: Open Collector Output Terminals

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

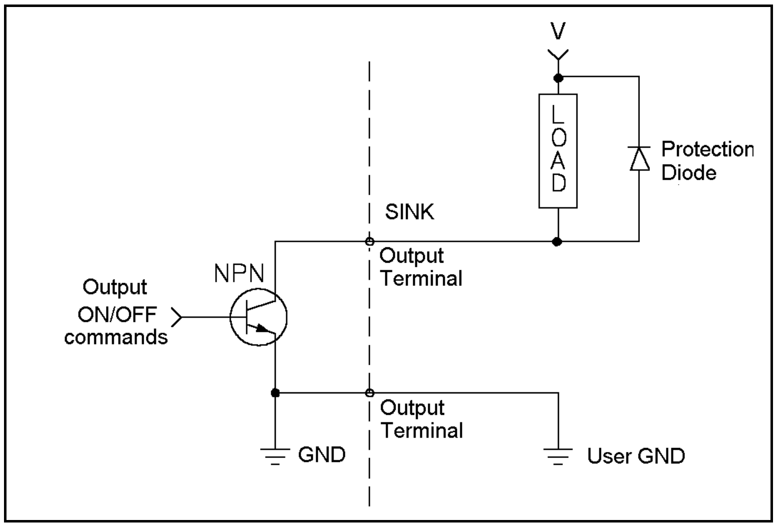


Figure 3-8: Open Collector Output: NPN Mode

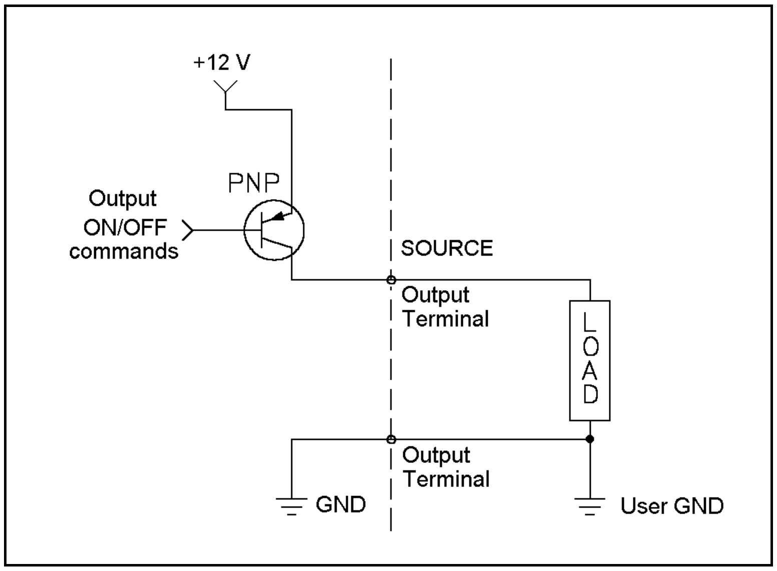


Figure 3-7: Open Collector Output: PNP Mode

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.

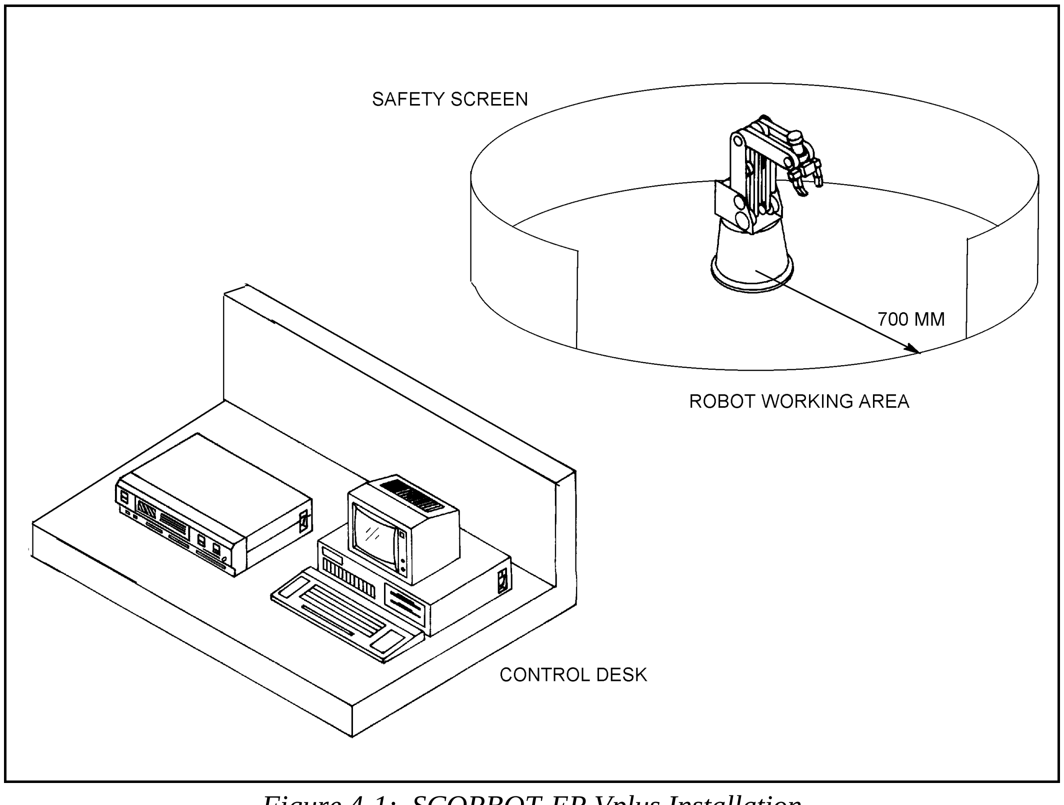


Figure 4-1: SCORBOT-ER Vplus Installation

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.

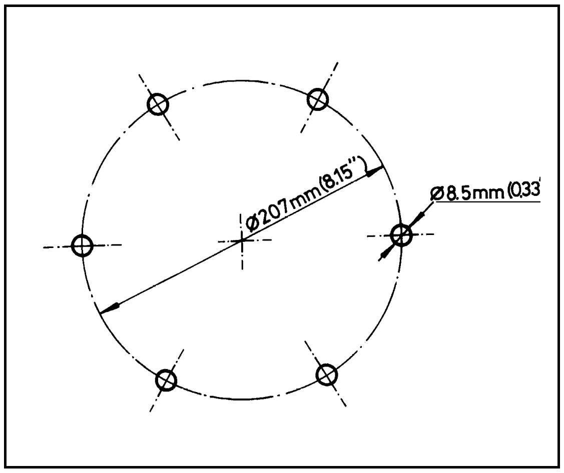


Figure 4-2: Robot Base Plate Layout

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.

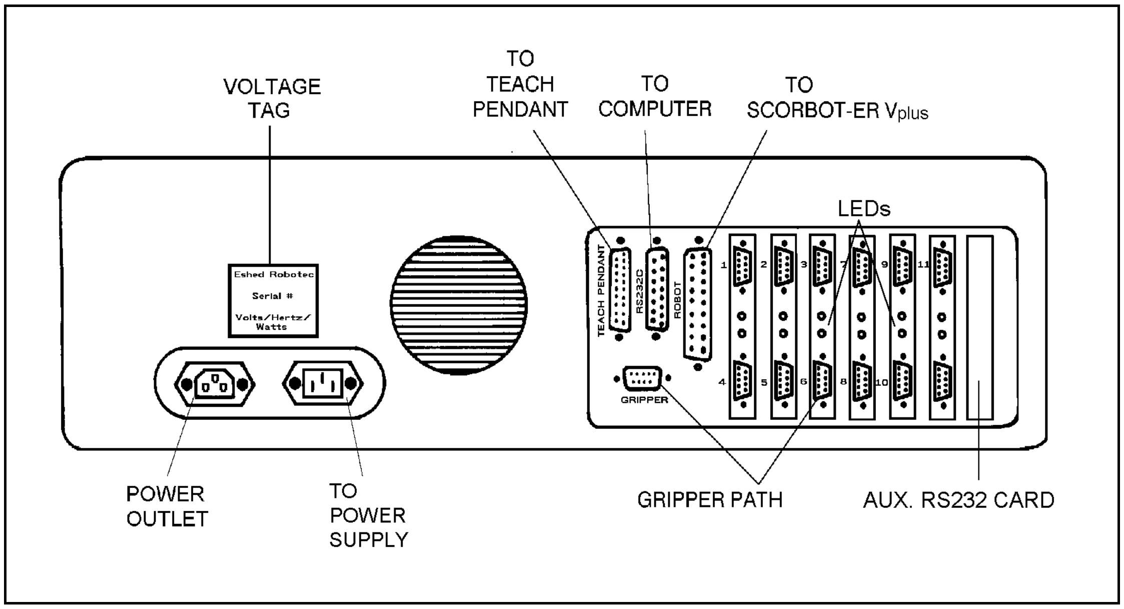


Figure 4-3: Controller Rear Panel

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**.

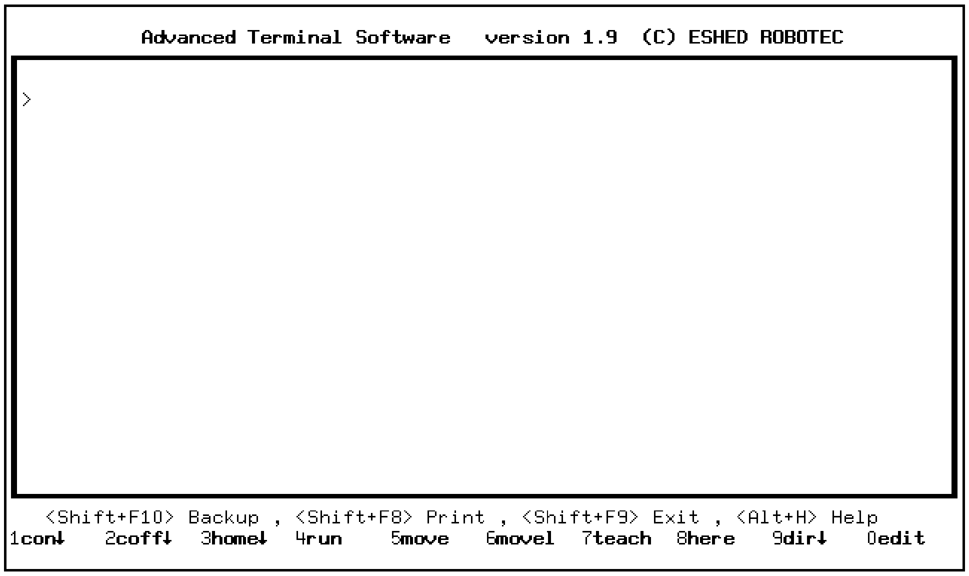
   If the controller is connected to computer port COM1 (default), type:

   ```
   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:



```
         Advanced Terminal Software   version 1.9  (C) ESHED ROBOTEC
>



   <Shift+F10> Backup , <Shift+F8> Print , <Shift+F9> Exit , <Alt+H> Help
1con↓    2coff↓   3home↓   4run     5move    6movel   7teach   8here    9dir↓    0edit
```

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

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

```
Ā               JOINTS
```

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*.

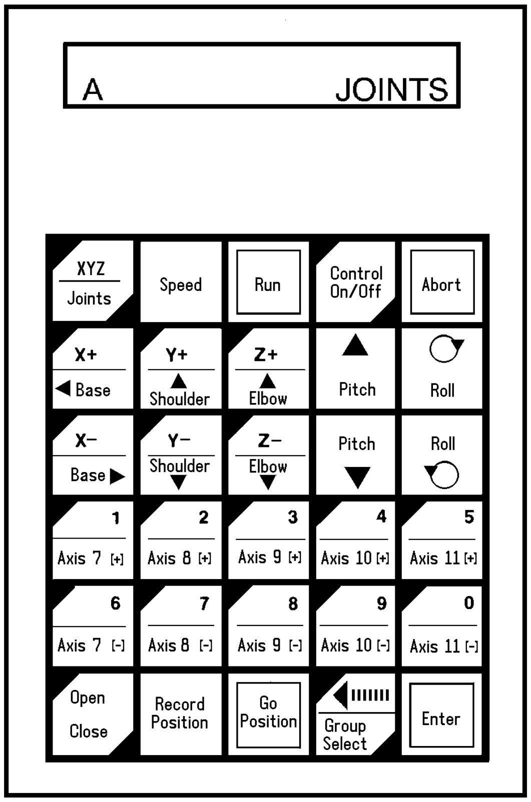
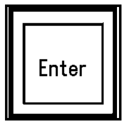


Figure 5-1: Teach Pendant

Keypad Functions

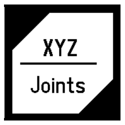
The teach pendant's keypad has 30 color-coded keys. Most of the keys are multi-functional; for example, some keys include both an axis drive command and a numeric function. The controller recognizes the keys from the order in which they are pressed. Thus, the numeric function will be active only if a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

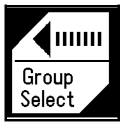


Accepts and/or executes the command which has been entered.

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.

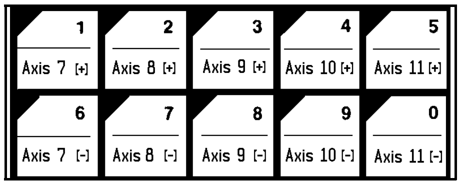
- Enables TP control of a specific axis group.

  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**.

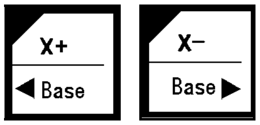
  The Record Position and Speed functions apply only to the currently selected group.

Control
On/Off

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

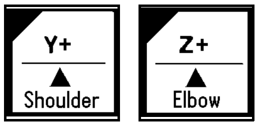


- The **Axis** keys move axes 7 through 11 in two directions.
- The numeric keys are operative if one of the following functions has been activated: **Speed**, **Run**, **Record Position**, **Go Position**, **Group Select**.



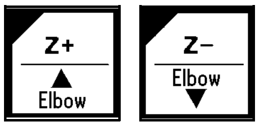
In Joint mode: the **Base/X** keys move the base axis in two directions.

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



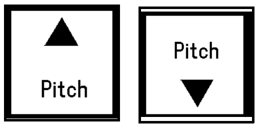
In Joint mode: the **Shoulder/Y** keys move the shoulder axis in two directions.

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.



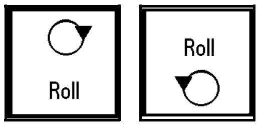
In Joint mode: the **Elbow/Z** keys move the elbow axis in two directions.

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

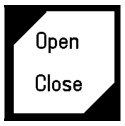


In Joint mode: the **Pitch** keys move the TCP (tip of gripper) up or down, without moving the other axes.

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.

Speed *number* Enter

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**.

Record Position *number* 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.

Go Position *number* Enter

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.

Run ***number*** Enter

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.

Abort

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
On/Off

```
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.

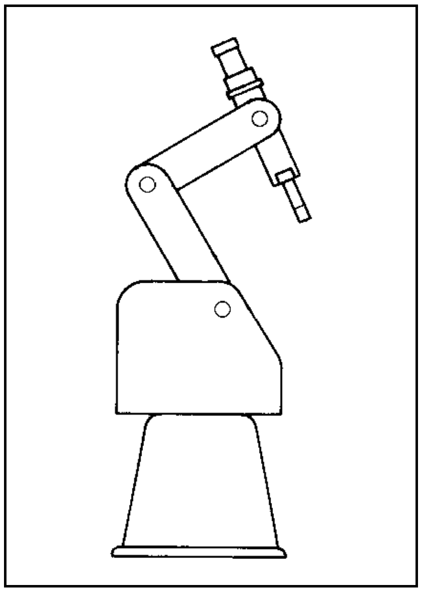


Figure 6-1: SCORBOT-ER Vplus Home Position

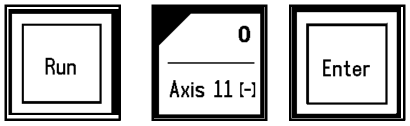
During the homing procedure, the robot joints move and search for their home positions, one at a time, in the following sequence:
shoulder (axis 2), elbow (axis 3), pitch (axis 4), roll (axis 5), base (axis 1), gripper (axis 6).

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.

☞ *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.

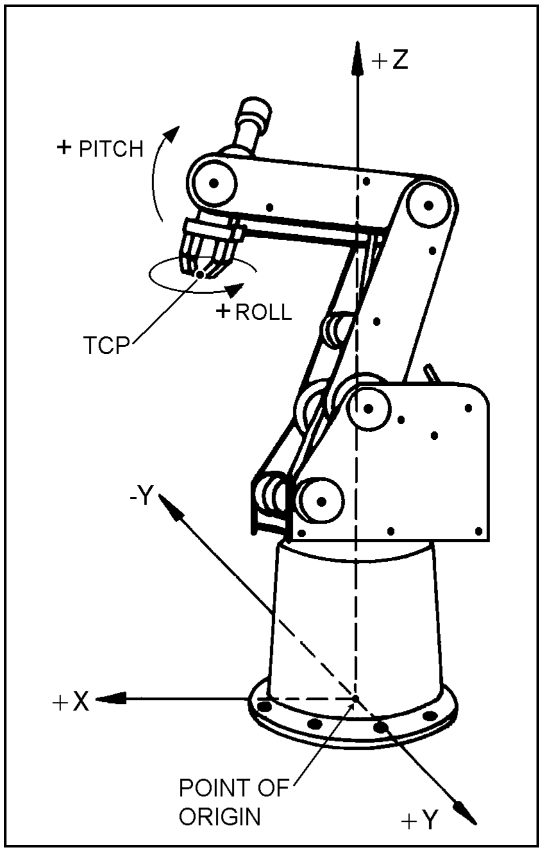


Figure 6-2: Cartesian Coordinates

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: **XYZ / Joints**

```
A               JOINTS
```

Press again: **XYZ / Joints**

```
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 On/Off

```
CONTROL   ENABLED
```

Press again: Control On/Off

```
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).

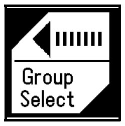
Axis Control Groups

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

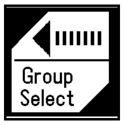
To select the axis control group:

Press:



```
B̄               JOINTS
```

Press again:



```
AXIS . .
```

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.

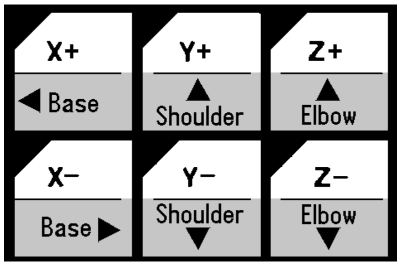
When the coordinate system is set to the Joint mode, the robot responds to movement commands by moving from one defined point to another.

Peripheral axes always move according to Joint coordinates.

TP

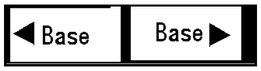
When in XYZ mode, the controller recognizes the Cartesian functions of the teach pendant keys.

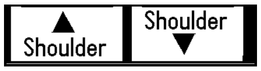
When in Joint mode, the controller recognizes the joint functions (shaded in diagram) of the teach pendant keys.

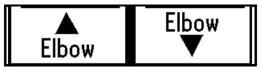


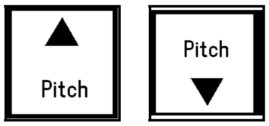
The teach pendant offers the easiest method for moving the robot arm. You simply press an axis movement key, and the robot moves. When you release the key, movement stops.

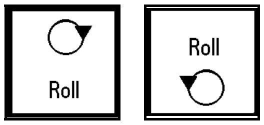
Before you press the keys shown below, make sure JOINTS, Group A, and Control On are active. Move the axes of the robot, in both directions.

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: **X+** **X–**

Press: **Y+** **Y–**

Press: **Z+** **Z–**

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: Open Close

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

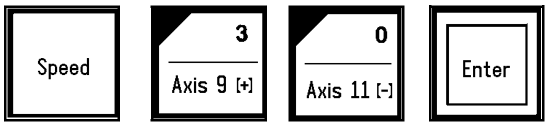
TP

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.

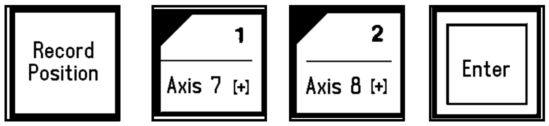
If you accidently record coordinates for position 0, execute the Home program. The homing routine records the proper coordinates for position 0.

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.”

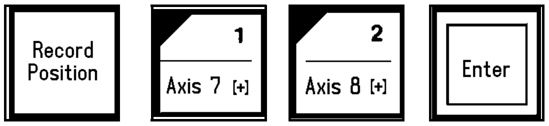
Moving to Recorded Position

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`**

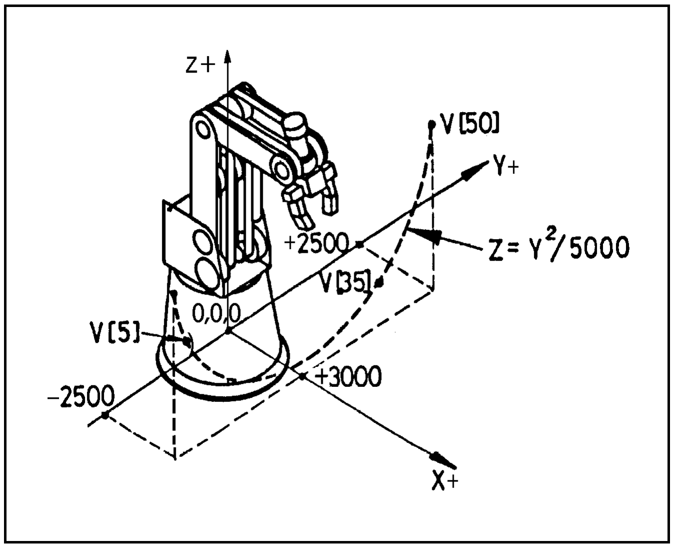


Figure 7-1: Parabola

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 Y^2
  SET       ZV=ZV / 5000        Value of Z will be Y^2/5000
  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 Y^2/5000.
  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                The initial Y coordinate.
FOR       I=1 TO 120            Starts a loop of 120 repetitions.
  SET       YV=YV + 50          Distance between positions:Y=5mm
  SET       ZV=YV * 360         These two operations scale the
  SET       ZV=ZV / 3000        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.
  SET       ZV=1500 SIN ZV      Sin Z is multiplied by a scaling
                                factor of 1500 to produce an
                                acceptable result.
  SET       ZV=ZV + 2000        Offsets the Z value by 200mm to
                                keep movement above table.
  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
```

```
       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.

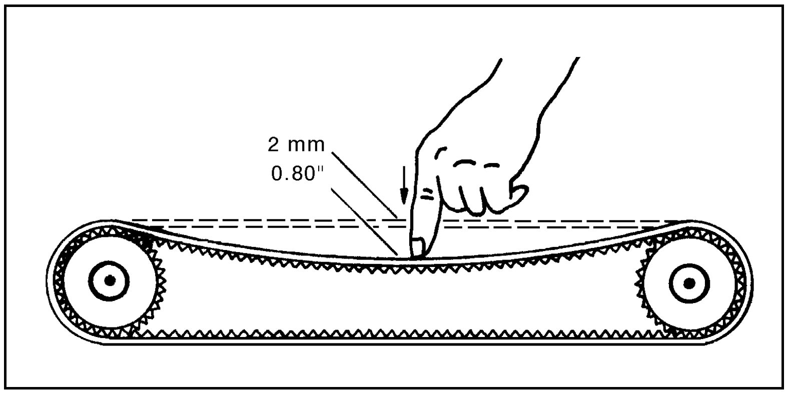


Figure 8-1: Belt Tension

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 exececssive 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.

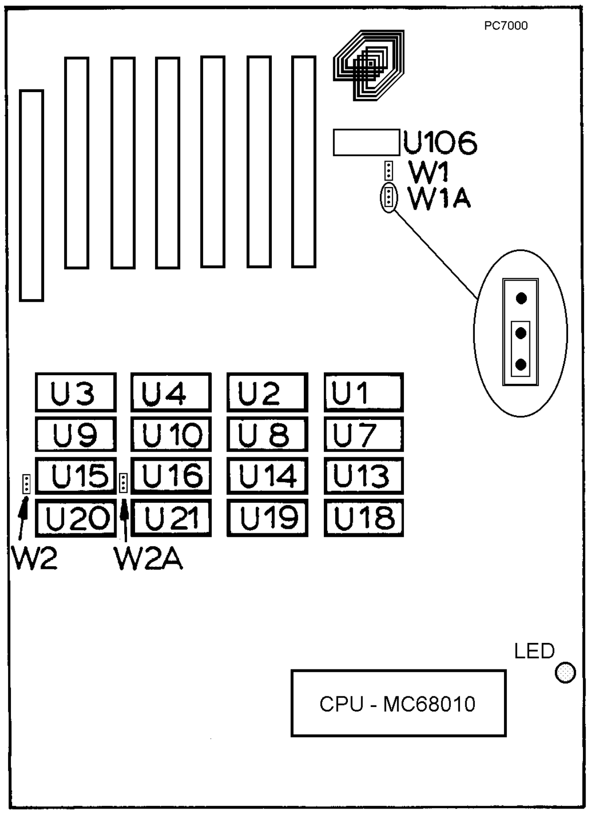


Figure 8-2: Main Board - Memory

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.

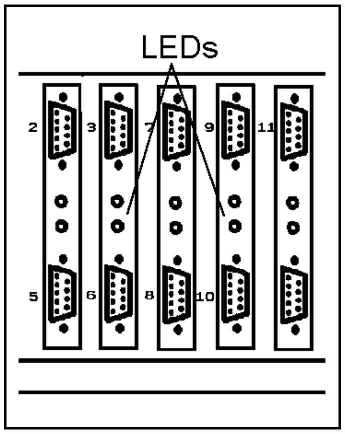


Figure 8-3: Driver Card LEDs

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.)

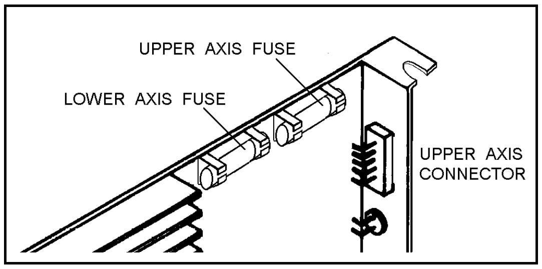


Figure 8-4: Driver Card Fuse

- **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.

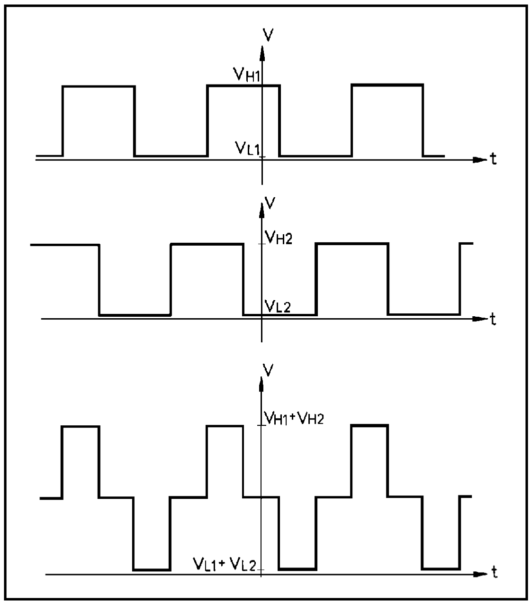


Figure 8-5: Encoder Signals

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.)

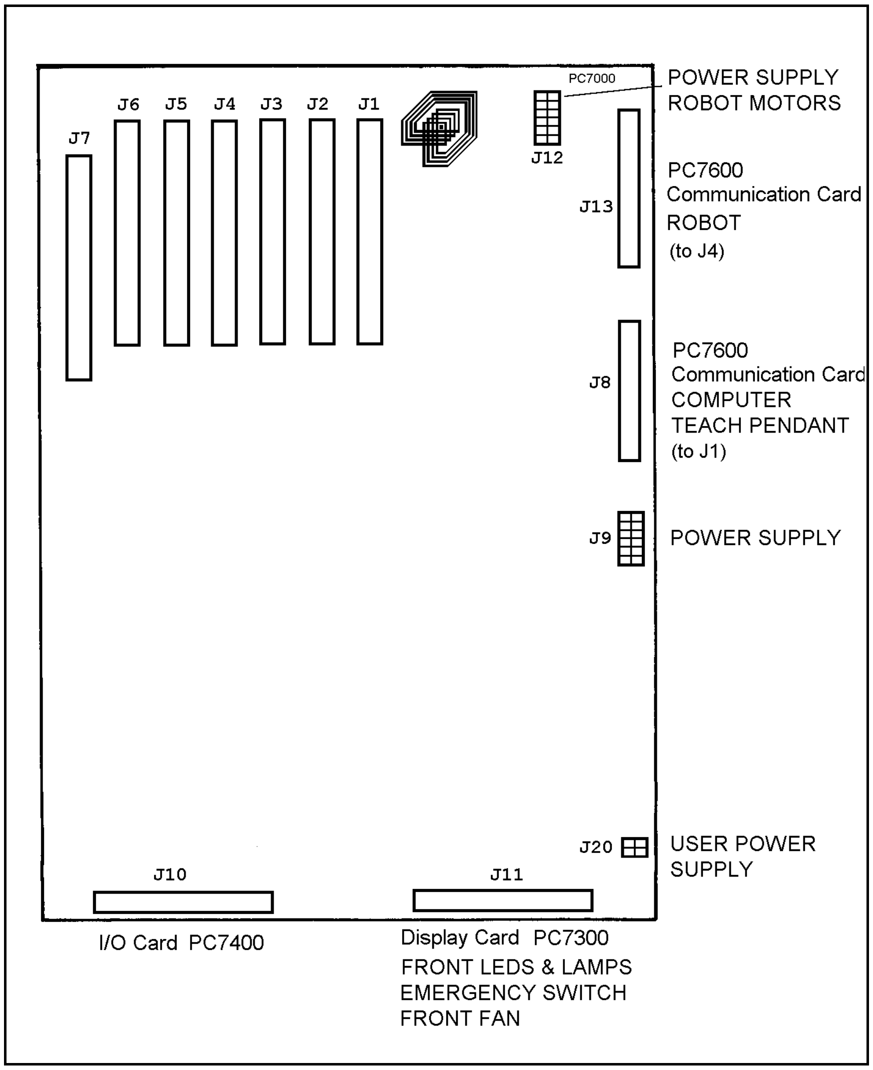


Figure 8-6: Main Board - Connectors

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.

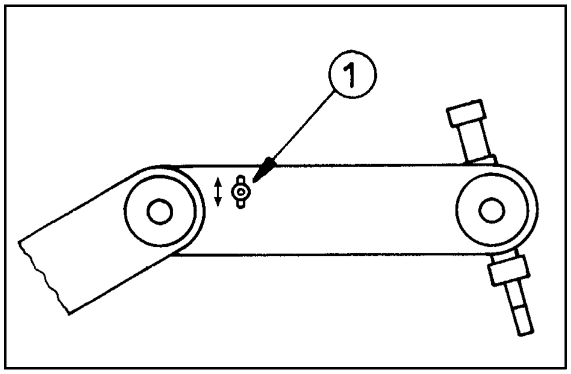


Figure 8-7: Tightening Belts in Forearm

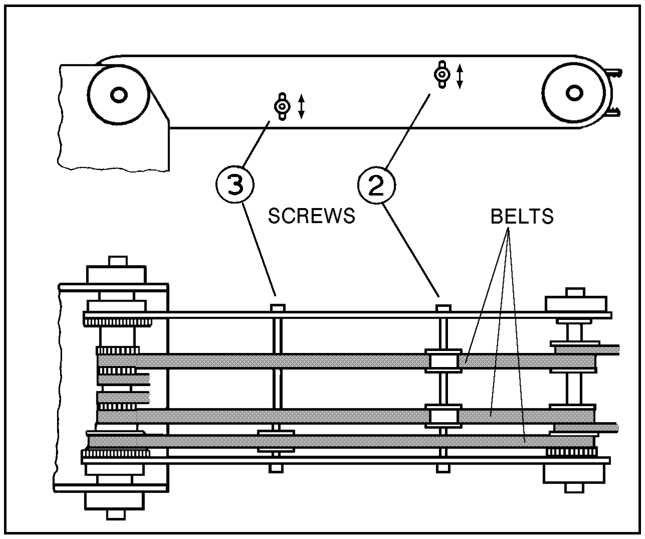


Figure 8-8: Tightening Belts in Upper Arm

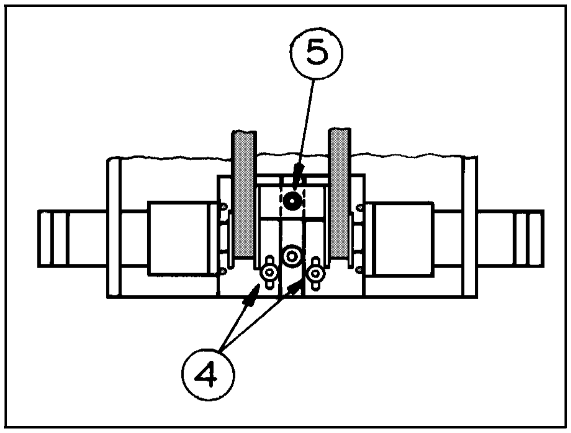


Figure 8-9: Tightening Belts in Robot Base

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.
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.

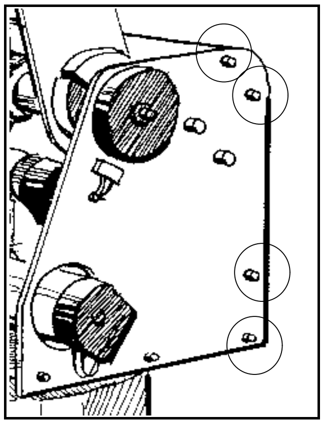


Figure 8-10: Shoulder Cover Screws

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.

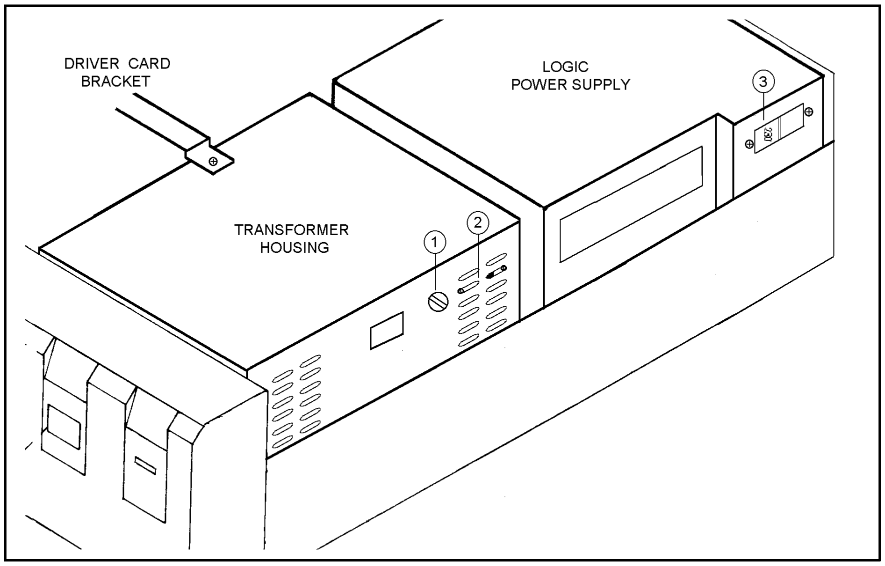


Figure 8-11: Controller Voltage Setting

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

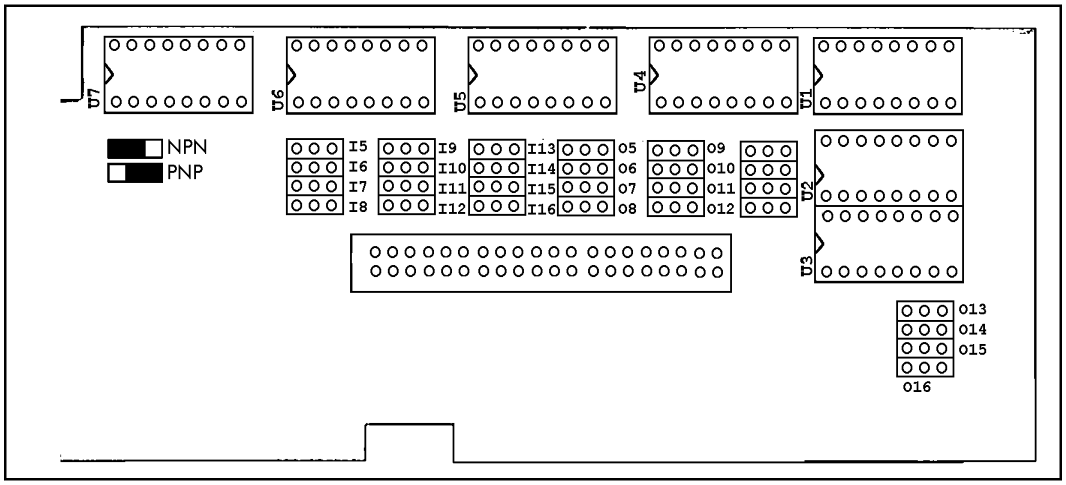


Figure 8-12: I/O Card - Logic Jumpers

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.

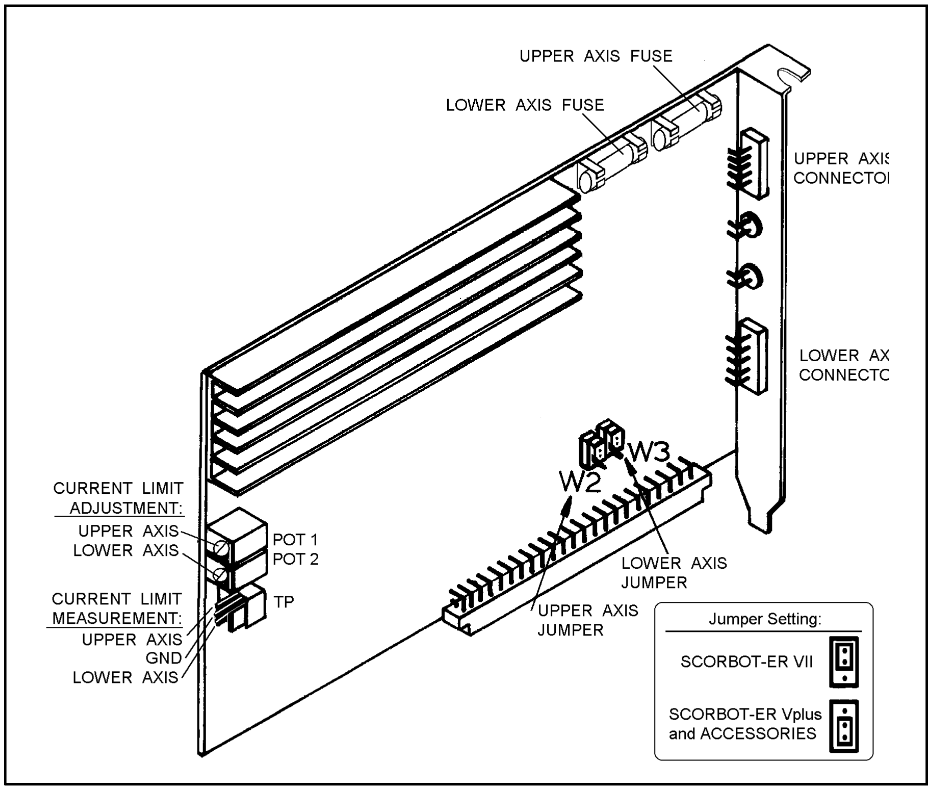


Figure 8-13: Axis Driver Card

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.

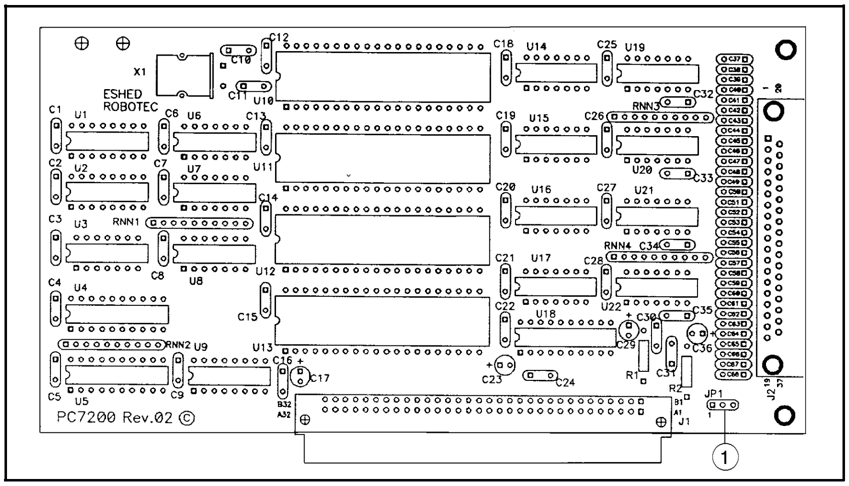


Figure 8-14: Auxiliary RS232 Communication Card

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.

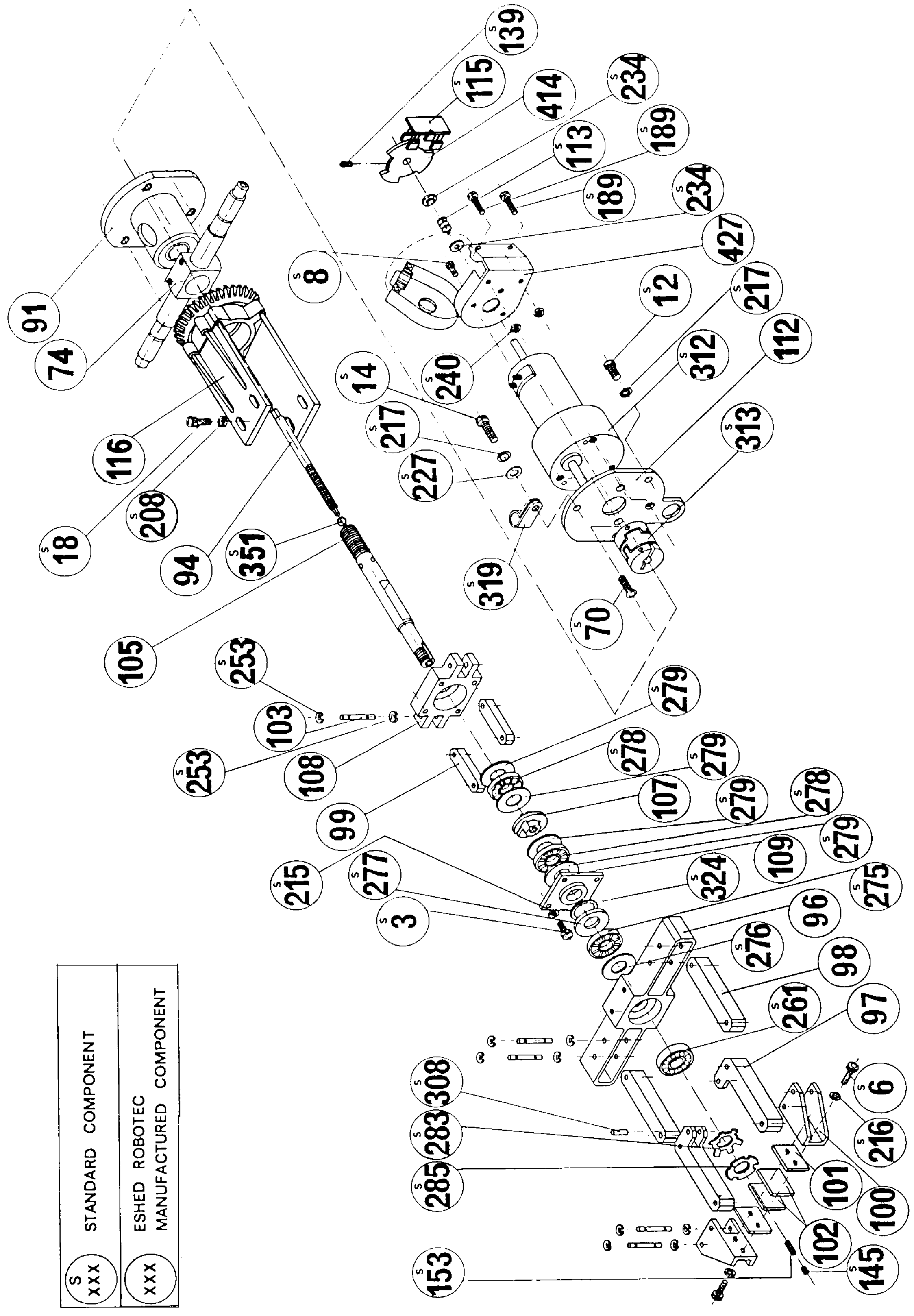


Figure 9-1: Gripper Assembly

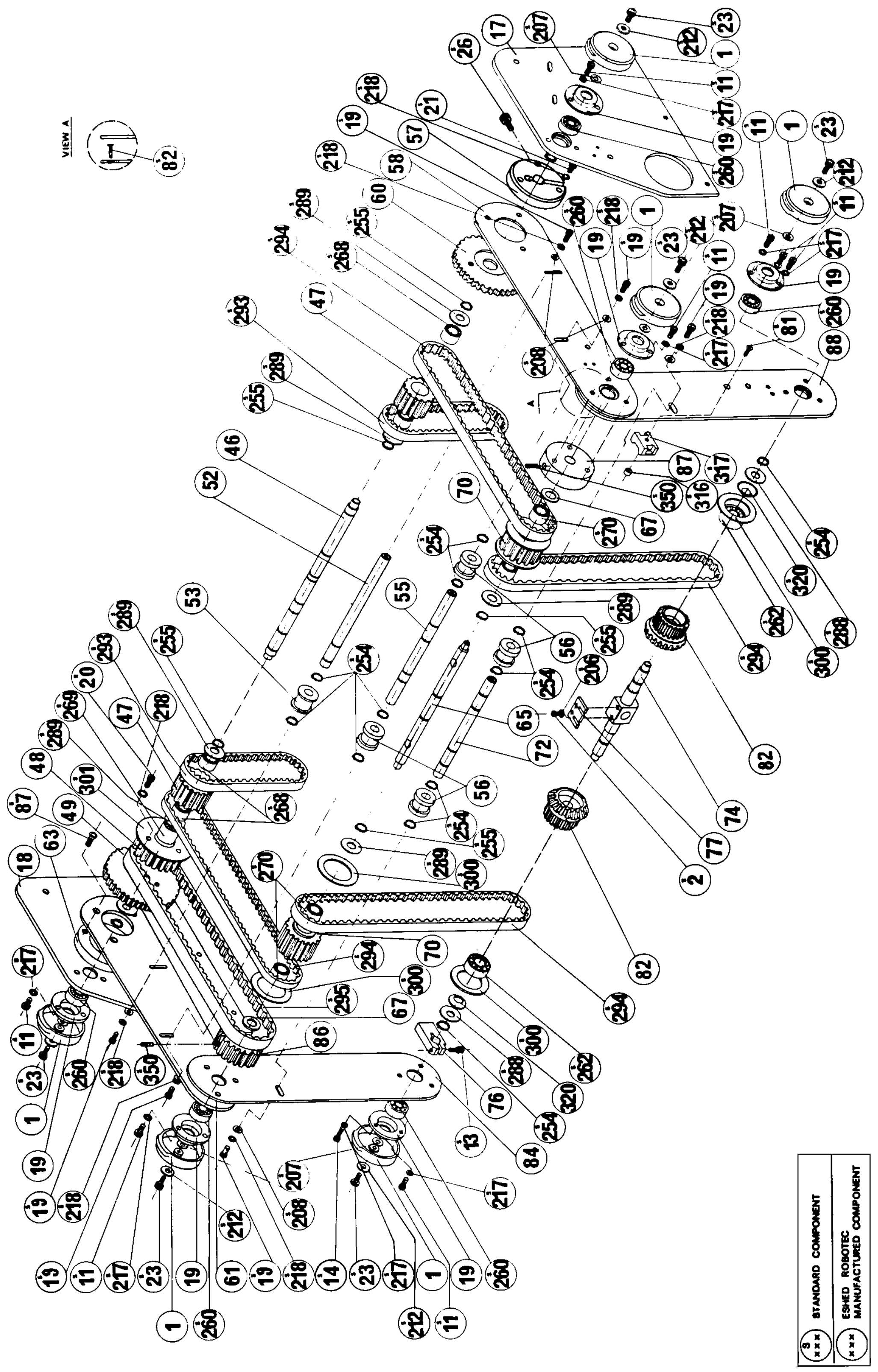


Figure 9-2: Robot Arm Assembly

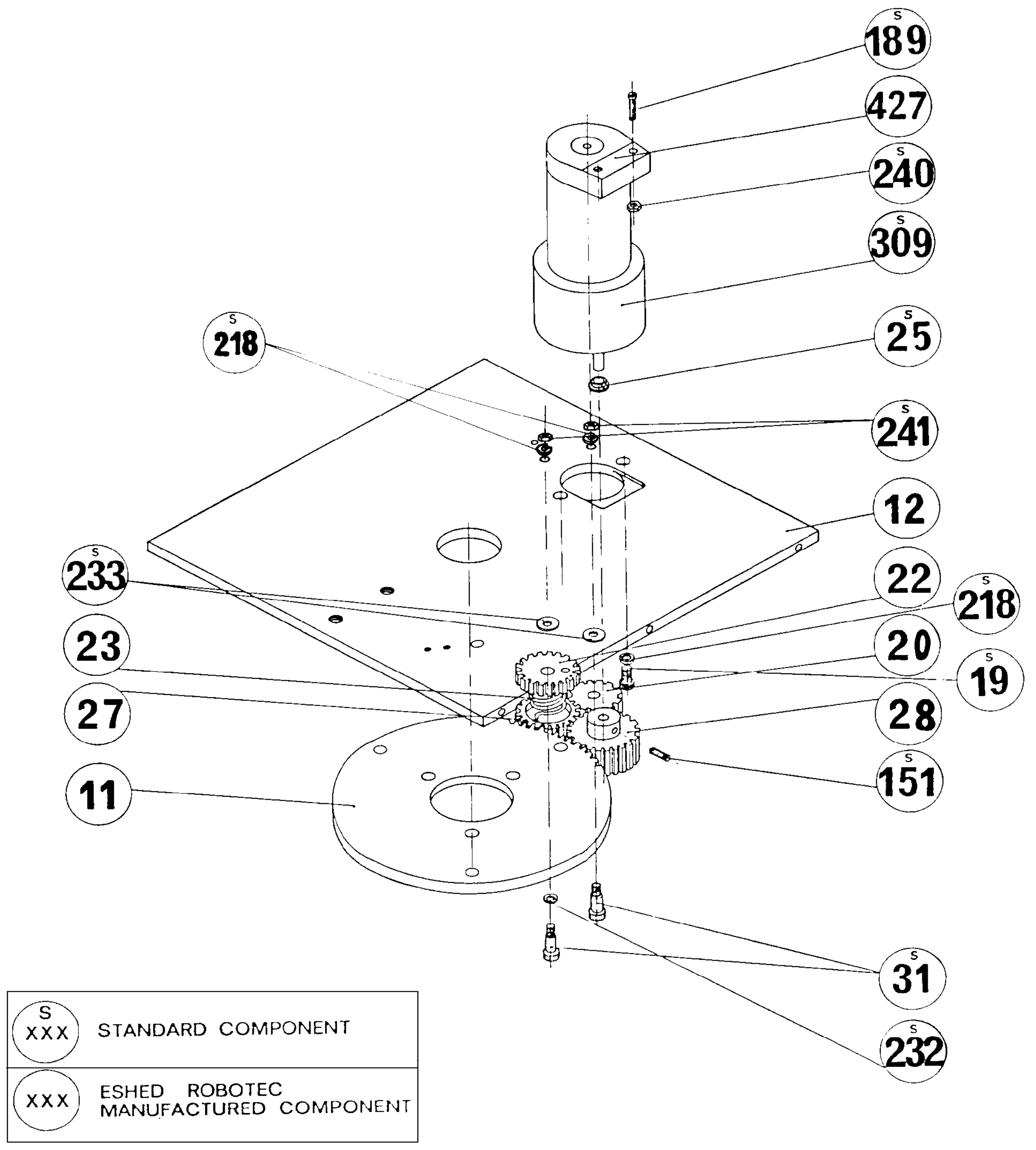


Figure 9-3: Anti-Backlash Assembly

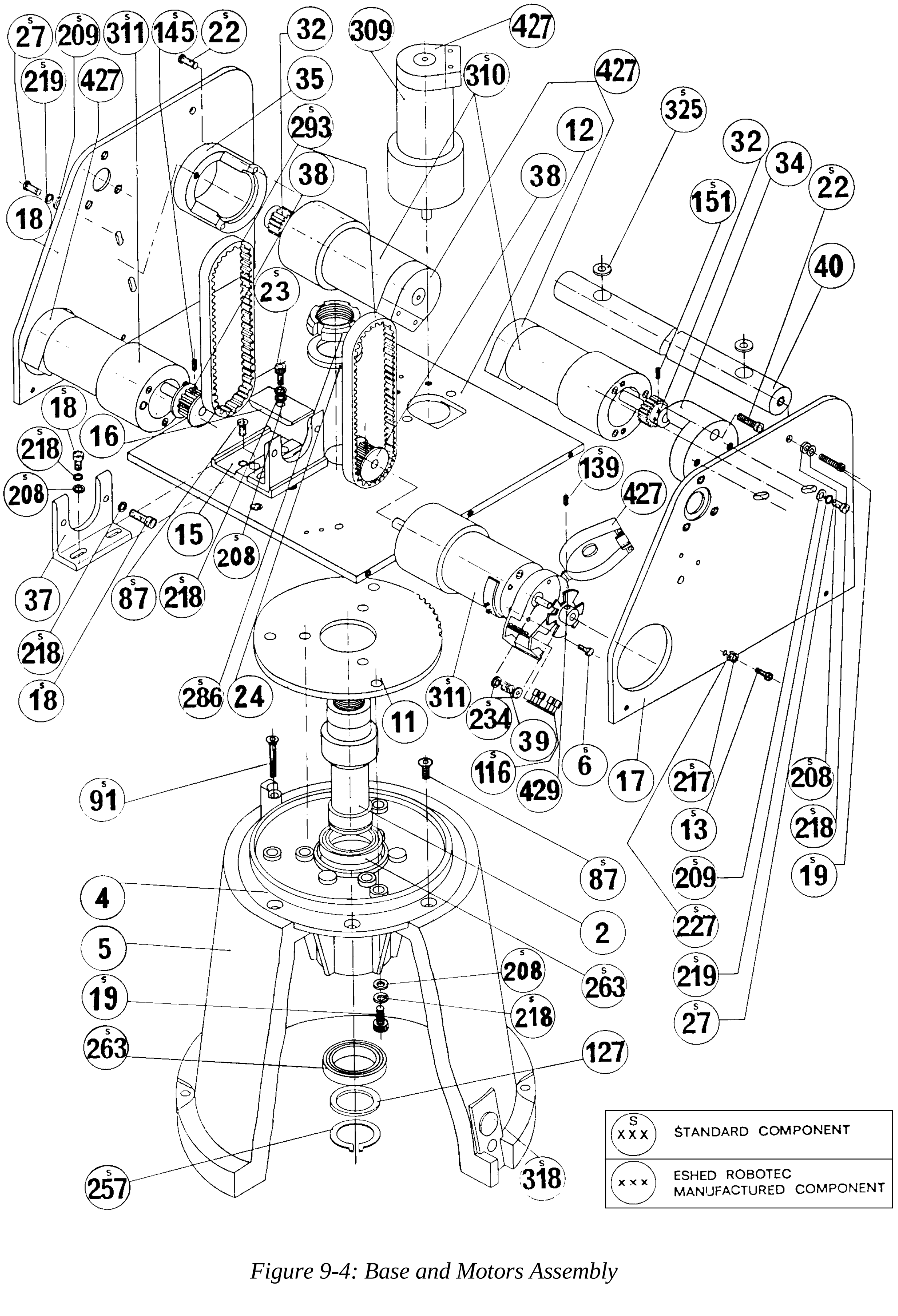


Figure 9-4: Base and Motors Assembly

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*]

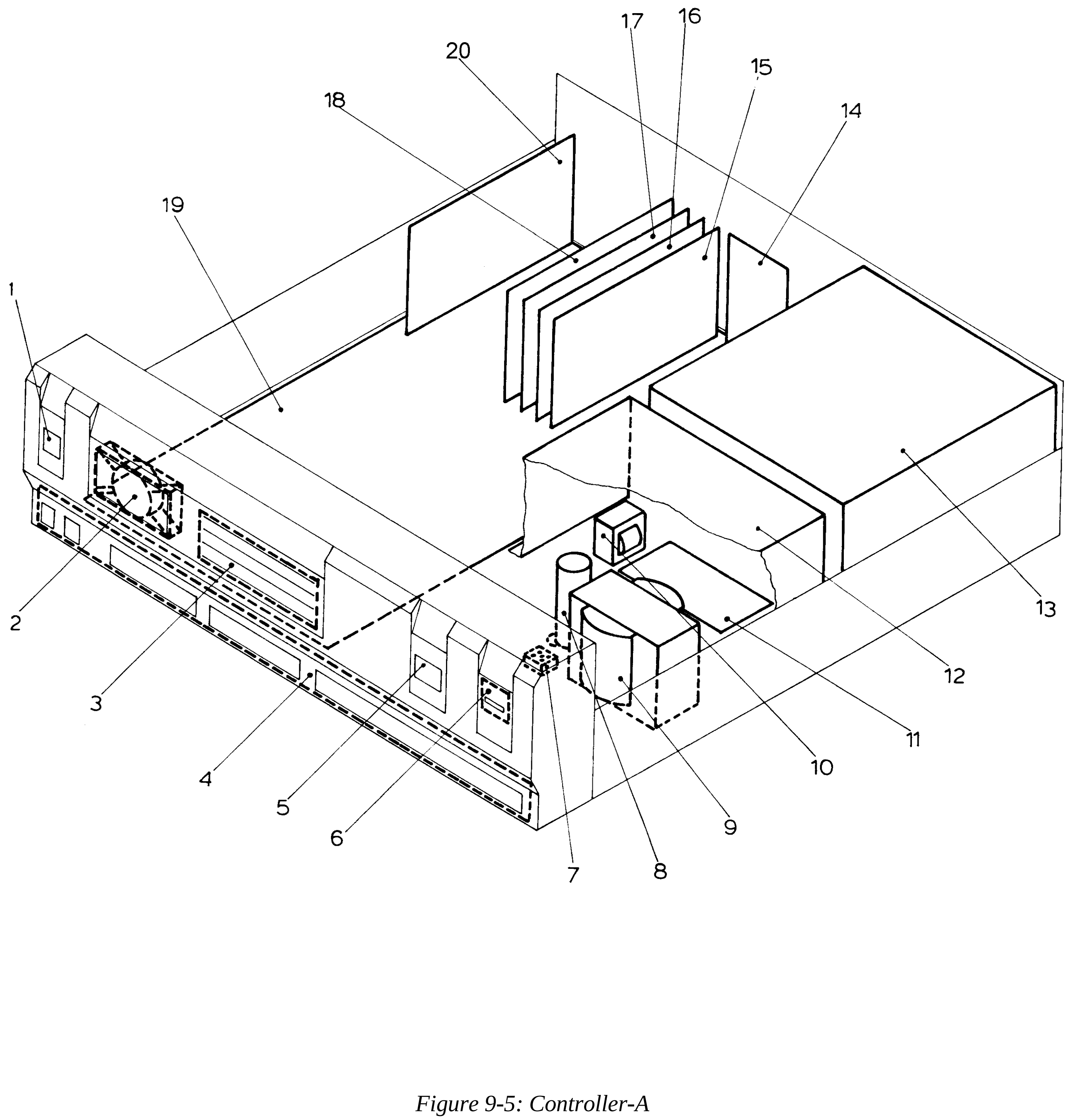


Figure 9-5: Controller-A

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.)

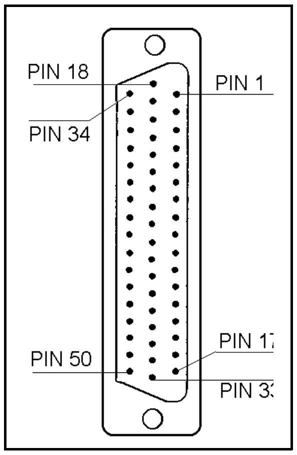


Figure 10-1: Robot D50 Connector

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.

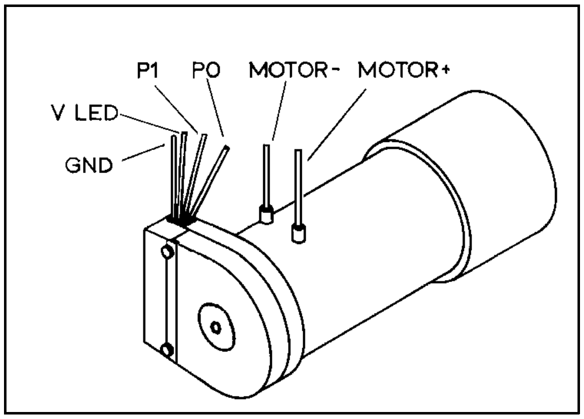


Figure 10-2: Motor Wiring

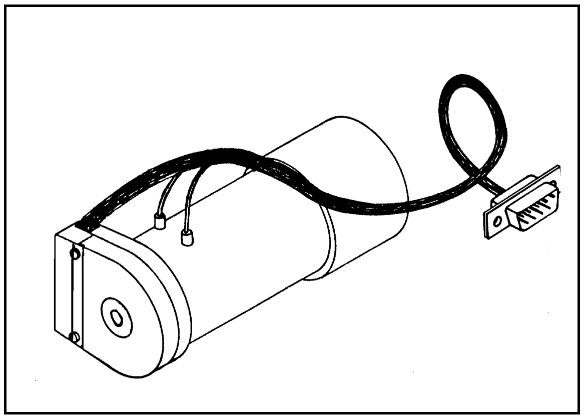


Figure 10-3: Motor with D9 Connector

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 .

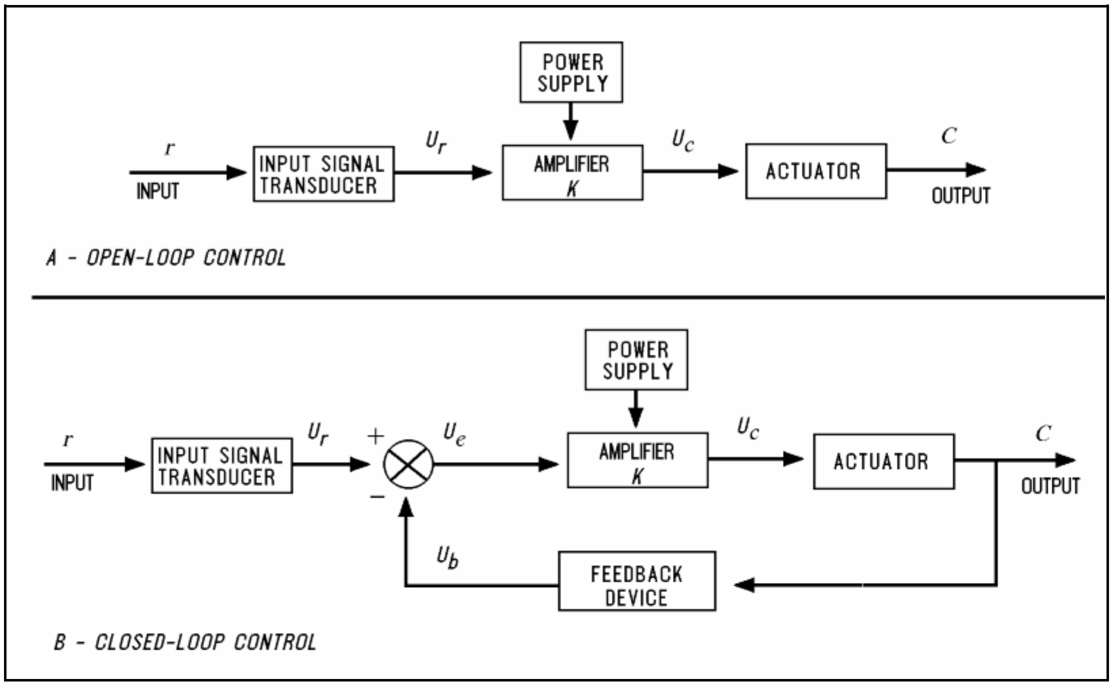


Figure A-1: Open-Loop and Closed-Loop Control

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.

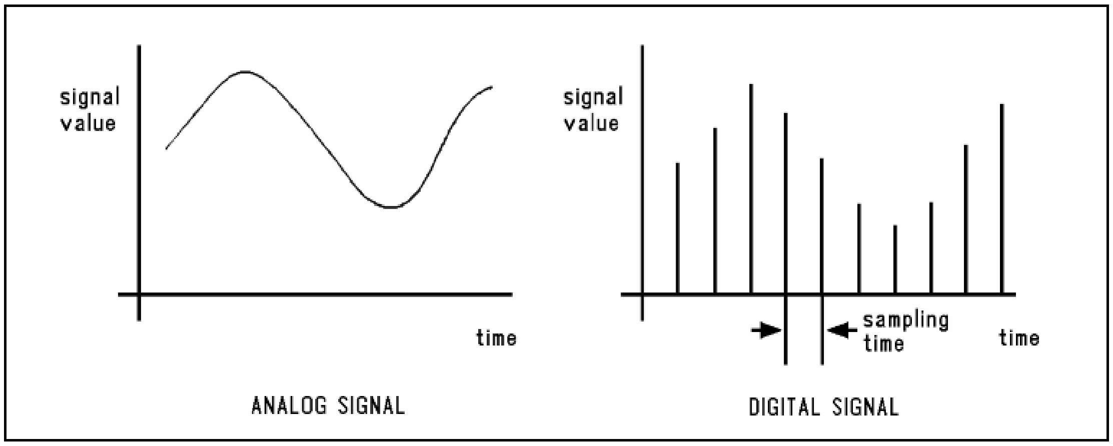


Figure A-2: Analog and Digital Signals

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.

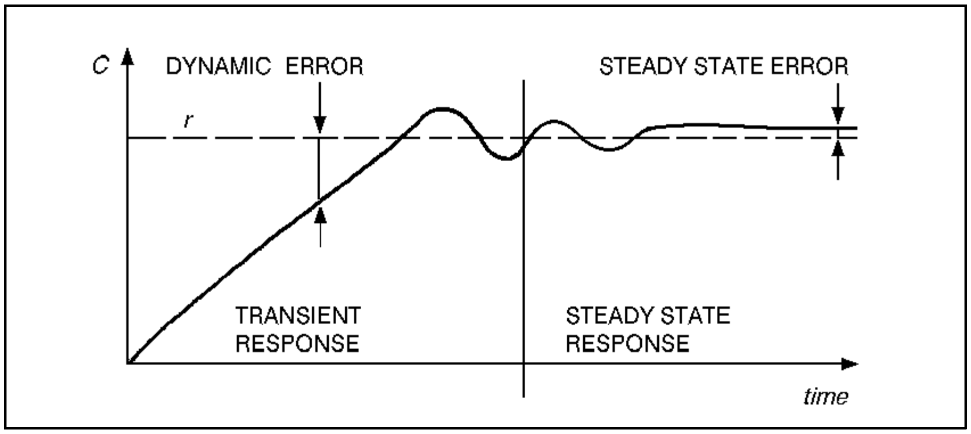


Figure A-3: Transient and Steady States

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.

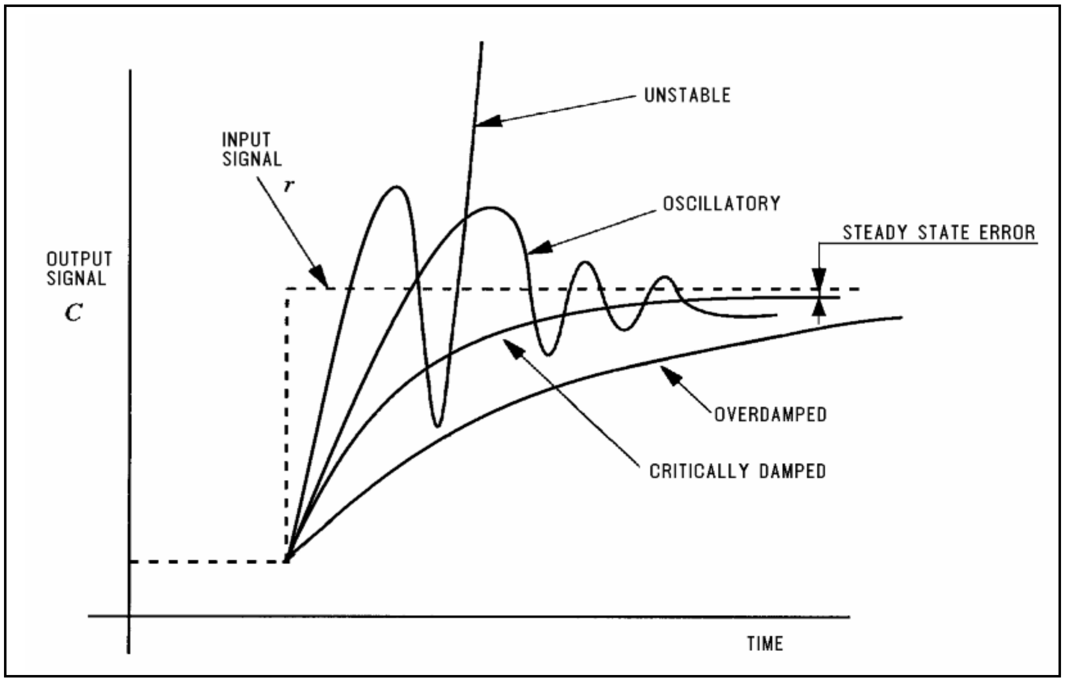


Figure A-4: Transient State Responses

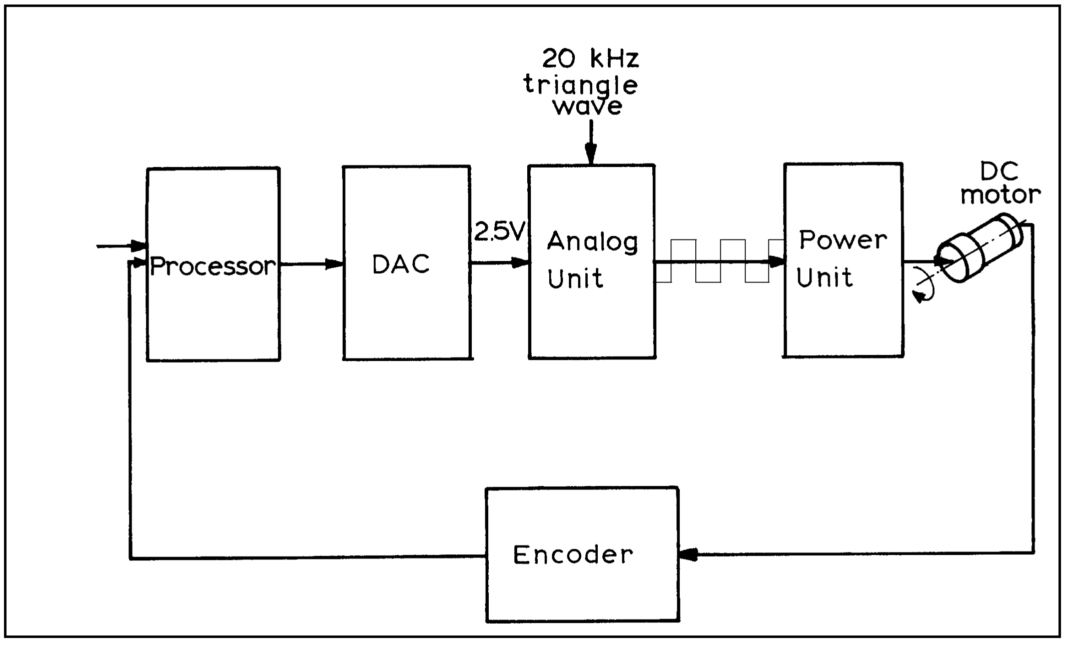


Figure A-5: Controller-A Control Loop

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.

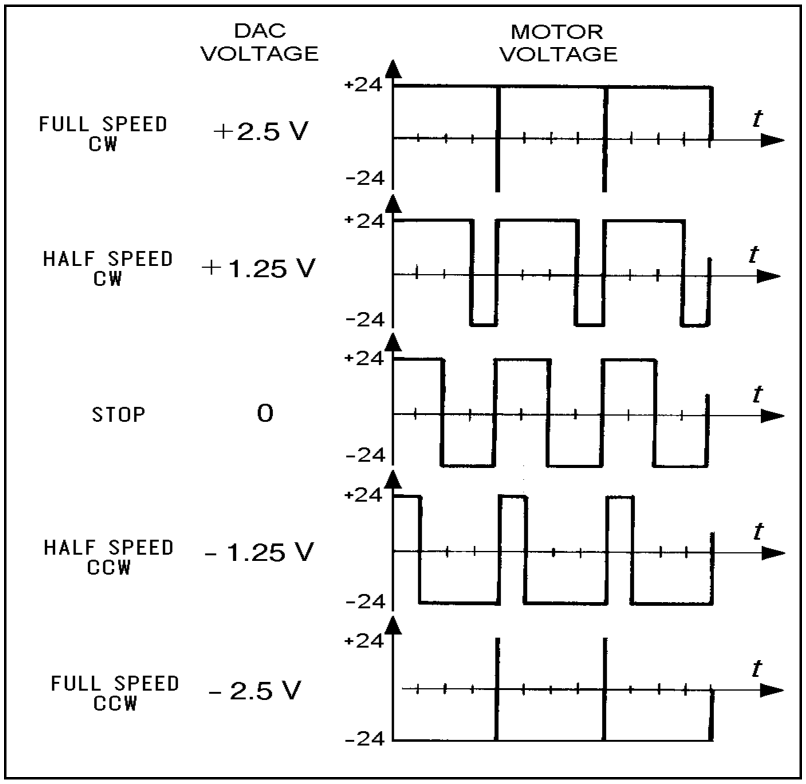


Figure A-6: Controller-A Control Signals

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.

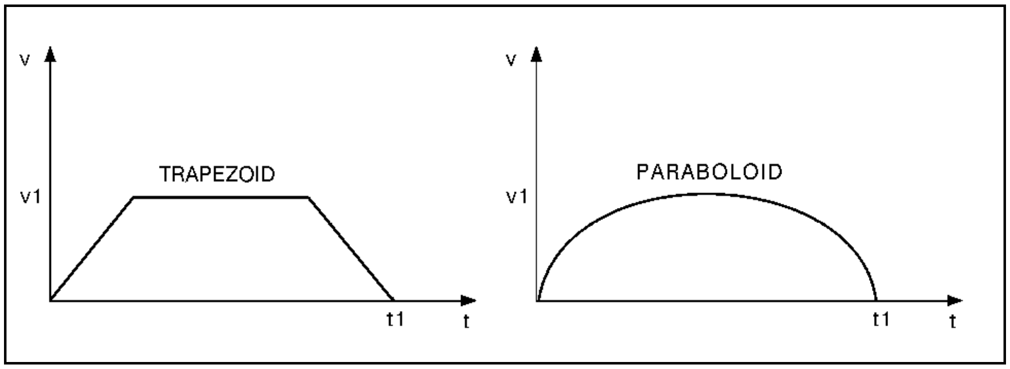


Figure A-7: Trajectory Control Profiles

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.

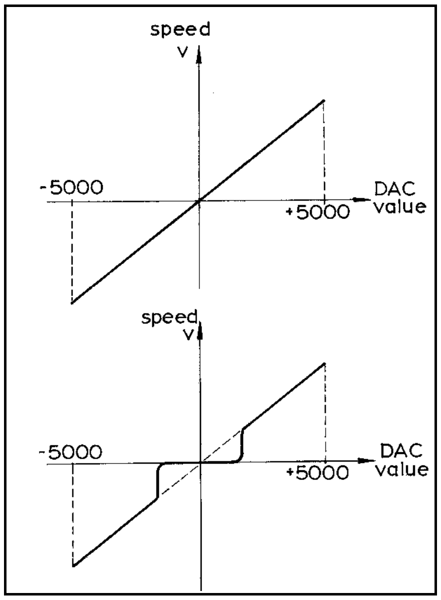


Figure A-8:
Linearity and Non-Linearity

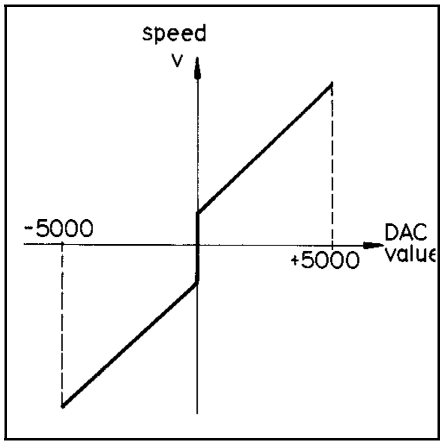


Figure A-9:
Control System Offset

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*.