

CONTROLLER

DESCRIPTION

The *CONTROLLER* is to enable up to six motors to be controlled using position feedback.

The central control element for the *CONTROLLER* is a Picaxe-40X microcontroller.

Other devices may also be connected to the unused inputs and outputs.

The *CONTROLLER* is suitable for many applications, such as the *ROBOT ARM*.

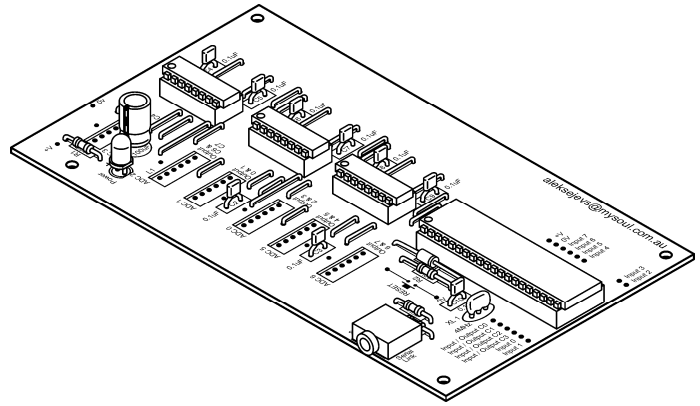


Figure 1 Controller PCB Assembly

THE PROJECT

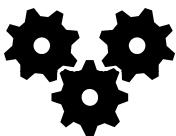
To carry out the project, the student must:

- Design and build a suitable system Assemble the printed circuit board, connect the wiring, motors and position feedback potentiometers.
- Program the Picaxe microcontroller and adjust the program parameters.

INVESTIGATION

This project provides a number of different aspects of the *CONTROLLER* for investigation. Some ideas are listed below.

- Investigate attaching additional input and output devices.
- Investigate adding more "intelligence" or different sequences to the program.
- Investigate other devices that could use up to six bi-directional motors with position feedback.



Issued: August 2009

SCORPIO TECHNOLOGY VICTORIA PTY. LTD.

A.B.N. 34 056 661 422

17 Inverell Ave., Mt. Waverley, Vic. 3149

Tel: (03) 9802 9913 Fax: (03) 9887 8158

www.scorpiontechnology.com.au

1. COMPONENTS REQUIRED

1.1. COMPONENTS SUPPLIED

This unit is designed for use in conjunction with the *ROBOT ARM* unit and component kit. The following components are supplied in the kit:

- 1 x Printed Circuit Board (PCB)
- 1 x Picaxe-40X (Microcontroller)
- 3 x L293D (Motor Driver IC)
- 1 x IC Socket (40 pin)
- 3 x IC Socket (16 pin)
- 1 x 4MHz Resonator - 3 Pin
- 1 x LED (Red) 5mm
- 1 x Stereo Socket - 3.5mm
- 9 x 0.1uF Capacitor (monolithic or equivalent)
- 1 x Resistor - 220 Ohms (Red-Red-Brown-Gold)
- 2 x Resistor - 10k Ohms (Brown-Black-Orange-Gold)
- 1 x Resistor - 22k Ohms (Red-Red-Orange-Gold)
- 1 x Capacitor - 100uF (electrolytic)
- 1 x 1N4004 Diode (or equivalent)
- 1 x Sliding Switch (small)
- 5 x Potentiometer 25k - Linear (B) - Splined Shaft

NOTE: This document describes the use of a Picaxe-40X. Other 40 leg Picaxe microcontrollers may be used, but changes may need to be made to the program and/or hardware.

NOTE: This project requires the use of small DC motors and a suitable mechanism to control. These are not supplied in the kit.

1.2. ADDITIONAL REQUIREMENTS

1.2.1. REQUIRED COMPONENTS

- Electric hook-up wire in assorted colours.
- Tinned copper wire for PCB assembly wire links.
- Assorted screws, nuts and washers.

1.2.2. COMPUTER REQUIREMENTS

- To install the PICAXE programming editor software requires a PC running Windows 95 or later with approximately 20MB free space. Any PC that runs the Windows operating system will work in textual 'BASIC' mode, however a Pentium 4 processor or later is recommended for graphical flowcharting. (We have also used an iMac running OSX and Windows XP.)
- A PC with 9-pin serial (RS-232) interface. (The PC may require an USB to RS-232 adapter.)
- The PICAXE editor can be downloaded from www.rev-ed.co.uk or from www.picaxe.co.uk.
- A PICAXE serial interface cable. This can be purchased from a number of suppliers or may be constructed as per the instructions in this document.

1.2.3. LED INDICATORS ON MOTORS (OPTIONAL)

These components are available from Scorpio Technology or from electronics suppliers:

- 6 x LED (Red) 5mm or 3mm
- 6 x LED (Green) 5mm or 3mm
- 6 x Resistor - 220 Ohms (Red-Red-Brown-Gold)

1.2.4. DISCONNECT CONTROLLER FROM THE PROJECT (OPTIONAL)

The connectors allow the *CONTROLLER* to be disconnected from project. These components are available from electronics suppliers.

- 6 x 0.1" Pin Plug - 6 pin
- 6 x 0.1" Header Socket - 6 pin

2. HOW THE CIRCUIT WORKS (THEORY)

2.1. ABOUT PICAXE MICROCONTROLLERS

The PICAXE* is a type of IC (Integrated Circuit) called a microcontroller, which is another name for a single chip computer. The PICAXE has similar features to a normal PC: CPU (central processing unit), RAM (random access memory), ROM (read only memory), I/O (input/output) lines, timers and A/D (analogue to digital) converters.

* PICAXE is a trademark of Revolution Education Ltd.

The Flash memory (EEPROM - Electrically Erasable Programmable Read Only Memory) in a PICAXE allows it to be reprogrammed many times (typically at least 100,000). This means that you can develop a program and constantly check the effects of changes.

A PICAXE program is created using an easy to learn version of the BASIC programming language (our preferred method) or using flowcharting software (with a limited command set).

NOTE: The PICAXE is supplied containing 'bootstrap' code that enables you to download your program using the serial cable. Do not substitute the PICAXE with a blank PIC microcontroller or any other integrated circuit.

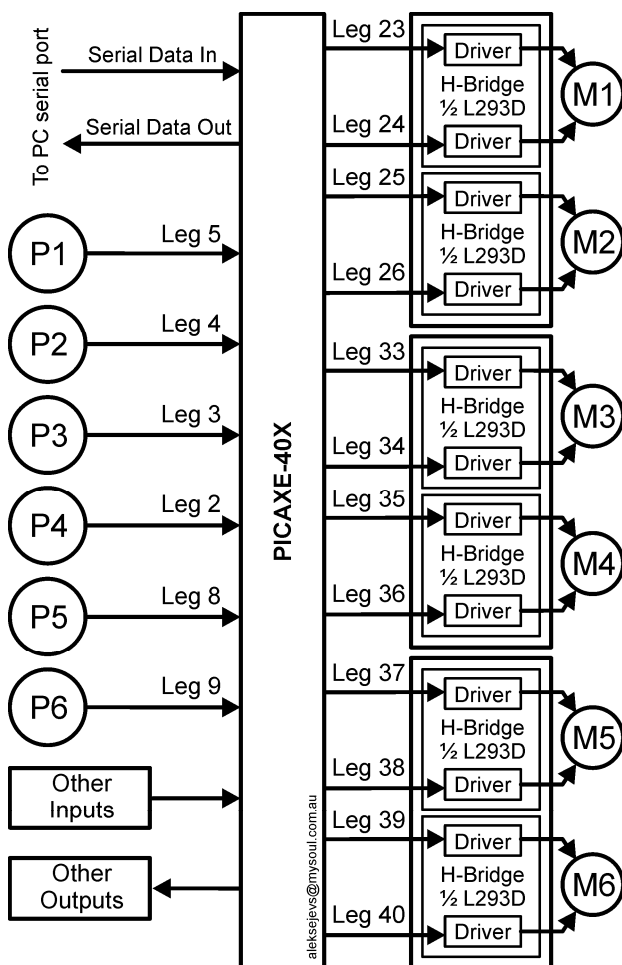


Figure 2 Block Diagram

2.2. CONTROL SYSTEM OVERVIEW

The *CONTROLLER* uses a Picaxe-40X microcontroller, three motor driver ICs and position measuring potentiometers to control the motors.

- The *CONTROLLER* uses potentiometers (variable resistors) as position sensing elements (P1 to P6).
- A motor drives each axis (M1 to M6). Through a gearing arrangement, each motor in is physically linked with the rotation of a potentiometer. The wiper arm of each potentiometer outputs a voltage (between the power rails) that is proportional to the angle of its axis. Within the Picaxe, a digital-to-analogue converter measures the voltage on each potentiometer. The program uses the resultant numerical value, which represents the angle of rotation.
- Within the program, each motor is commanded to move in a preset sequence of defined positions. While a motor is not at its currently required position, it is commanded to move towards the required position. When the desired position is reached, the sequence moves to the next step.
- The program may be modified as required.

2.3. CIRCUIT OVERVIEW

NOTE: PICAXE documentation refers to "Input Pins" and "Output Pins", which are not the same as the physical pins on a device. To avoid confusion, in this document "leg" means a physical pin of an integrated circuit and "pin" means a logical input or output.

- A program can be downloaded into the Picaxe memory to control the motors in response to various hardware and software parameters. The student may customise the program and attach other input/output devices as desired. This document does not detail this type of customisation – refer to the Picaxe Programmer online help for ideas.
- When the program runs, it compares the desired position of the axis that is currently moving with its target position. When the target position is reached, the program then moves the next axis in the program.
- Responding to the high/low voltage sequence on its inputs, each L293D opens/closes the electronic switches in its H-bridge circuits. Two motors can be independently driven backwards and forwards by each L293D. Therefore the Picaxe can command the motors to turn backwards, forwards and stop as required.

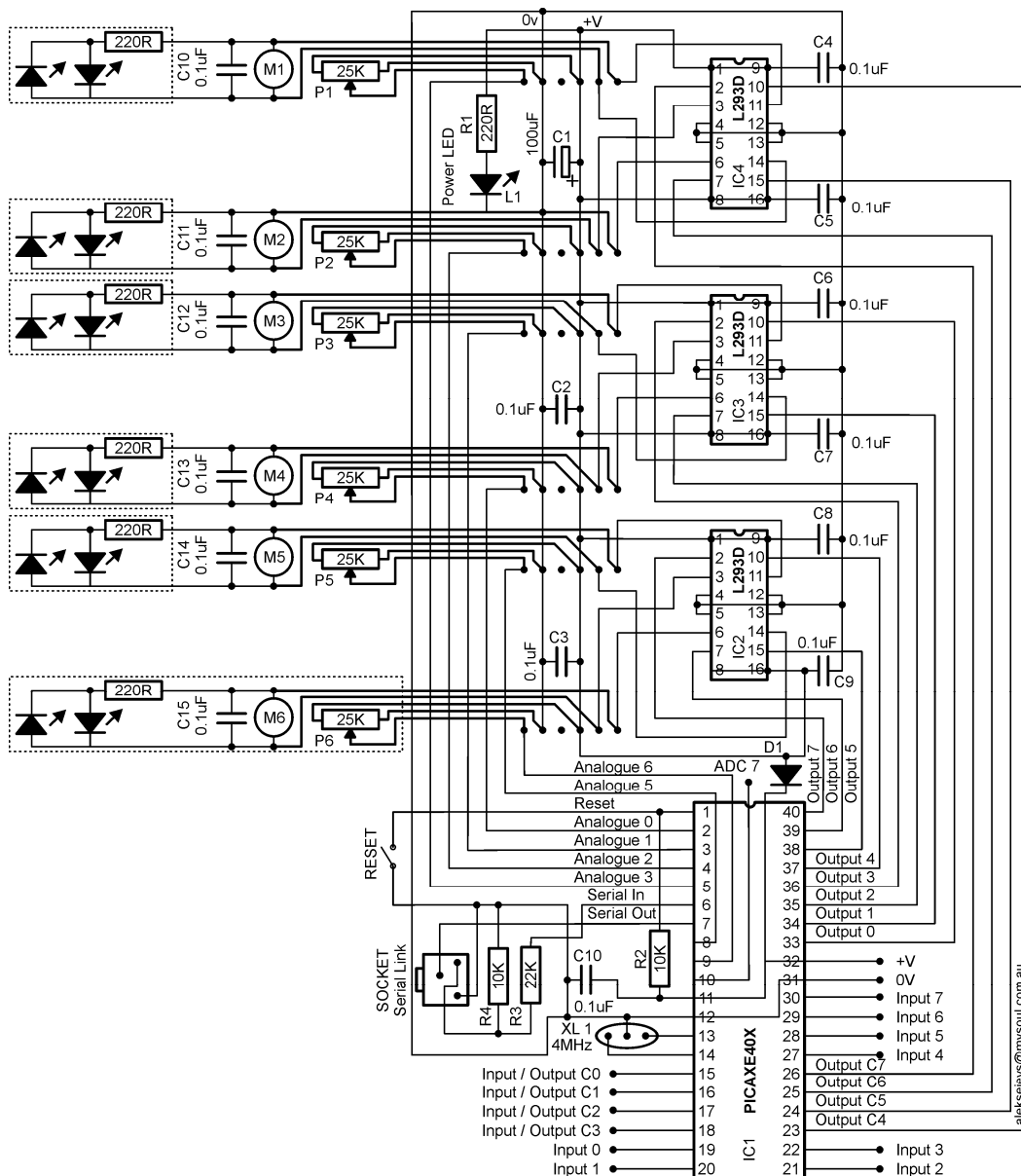


Figure 3 Circuit Diagram

2.4. PICAXE (IC1)

The function of each leg of the Picaxe is summarised in Table 1.

| Used For | Function | Leg | Leg | Function | Used For |
|----------------------|-----------------|-----|-----|-----------|----------|
| Reset | Reset | 1 | 40 | Output 7 | Motor 6 |
| Potentiometer 4 | Analogue 0 | 2 | 39 | Output 6 | Motor 6 |
| Potentiometer 3 | Analogue 1 | 3 | 38 | Output 5 | Motor 5 |
| Potentiometer 2 | Analogue 2 | 4 | 37 | Output 4 | Motor 5 |
| Potentiometer 1 | Analogue 3 | 5 | 36 | Output 3 | Motor 4 |
| Stereo Socket | Serial In | 6 | 35 | Output 2 | Motor 4 |
| Stereo Socket | Serial Out | 7 | 34 | Output 1 | Motor 3 |
| Potentiometer 5 | Analogue 5 | 8 | 33 | Output 0 | Motor 3 |
| Potentiometer 6 | Analogue 6 | 9 | 32 | +V | +V |
| Spare Analogue Input | Analogue 7 | 10 | 31 | 0V | 0V |
| +V | +V | 11 | 30 | Input 7 | Spare |
| 0V | 0V | 12 | 29 | Input 6 | Spare |
| Resonator | Resonator | 13 | 28 | Input 5 | Spare |
| Resonator | Resonator | 14 | 27 | Input 4 | Spare |
| Spare | Input/Output C0 | 15 | 26 | Output C7 | Motor 2 |
| Spare | Input/Output C1 | 16 | 25 | Output C6 | Motor 2 |
| Spare | Input/Output C2 | 17 | 24 | Output C5 | Motor 1 |
| Spare | Input/Output C3 | 18 | 23 | Output C4 | Motor 1 |
| Spare | Input 0 | 19 | 22 | Input 3 | Spare |
| Spare | Input 1 | 20 | 21 | Input 2 | Spare |

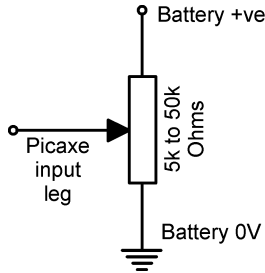
Table 1 Picaxe Leg Functions

- Leg 1 is used to reset the Picaxe. Normally this leg is tied high (to +V via 10k resistor). When it is brought low (to 0V), the Picaxe is reset. The program then restarts from the first line. This resistor must be present for reliable operation.
- Leg 2, Leg 3, Leg 4, Leg 5, Leg 8 and Leg 9 are connected to the wiper arm of potentiometers and are used as inputs to the respective analogue-to-digital converter.
- Leg 6 (serial data in) and Leg 7 (serial data out) are used only when transferring a program from a serial port on your PC (COM1: to COM4:) to the Picaxe. The 22k (W3) and 10k (R4) resistors must be present for reliable operation. Do not substitute other resistor values.
- Leg 10 is a spare analogue input.
- Leg 11 and Leg 32 are connected to the positive terminal (+6V) of the power supply (batteries).
- Leg 12 and Leg 31 are connected to the negative terminal (0V) of the power supply (batteries).
- Leg 13 and Leg 14 are connected to the outer legs of the resonator. (Middle leg is 0V.)
- Leg 15, Leg 16, Leg 17 and Leg 18 can be used as either inputs or outputs.
- Leg 19, Leg 20, Leg 21, Leg 22, Leg 27, Leg 28, Leg 29 and Leg 30 are spare inputs.
- Leg 23 and Leg 24 are connected to the L293 driving Motor 1.
- Leg 25 and Leg 26 are connected to the L293 driving Motor 2.
- Leg 33 and Leg 34 are connected to the L293 driving Motor 3.
- Leg 35 and Leg 36 are connected to the L293 driving Motor 4.
- Leg 37 and Leg 38 are connected to the L293 driving Motor 5.
- Leg 39 and Leg 40 are connected to the L293 driving Motor 6.

2.4.1. MOTOR OUTPUTS

- All of the motor outputs are digital (0V or +V).
- Two output legs on the Picaxe are required to drive one motor.
- Each of the motor outputs is directly connected to an input on an L293D motor driver IC. A total of six motors can be driven from the circuit without modification.
- To incorporate a seventh motor, use Analogue 7 (Leg 10) and two other outputs (Leg 15 to Leg 18). An additional L293D (or another H-bridge circuit) will be needed.
- A red/green pair of LEDs and resistor (optional) may be connected to each motor's terminals used to indicate which direction a motor is currently being driven. These LEDs may be useful during faultfinding.

2.4.2. POTENTIOMETER INPUTS



- All of the potentiometer inputs are analogue. Because the outer arms of each potentiometer are connected to 0V and +V, the voltage that appears on the wiper arm (driven by the motors), is between 0V and +V.
- Each of the potentiometers is associated with an analogue-to-digital converter. In the Picaxe, an analogue-to-digital converter translates an analogue voltage to a numerical value that can be used by the program.

Figure 4 Potentiometer

2.4.3. OTHER INPUTS

To obtain information from the outside world, switches and sensors can be added to unused input pins on the Picaxe. You should add these devices only after the basic design has been built and tested because you will need to modify the program. Some ideas for inputs include:

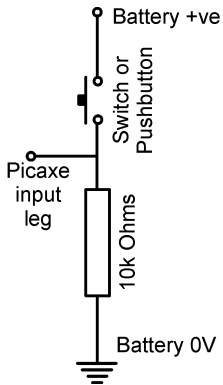


Figure 5 Switch or Pushbutton

- A push-button switch (normally open) may be connected to the reset input. When the reset switch is pressed, the program starts from the beginning.
- A toggle switch for the program to select between two programmed sequences.
- A microswitch can be used to create a "limit" switch for an axis.
- Emergency stop switch to halt the project when the button is pressed. Another switch could be used to detect if a "guard" or "fence" is opened. A separate button should be used to restart the sequence.
- Keypad (4x4 matrix or 4x5 matrix, as per telephone/calculator keypad) to provide direct control of motors. The concept could be extended as a "teaching" pendant.

2.4.4. OTHER OUTPUTS

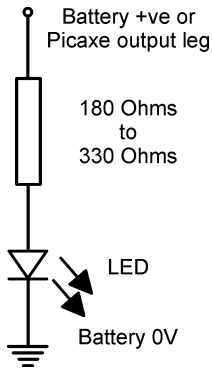


Figure 6 LED

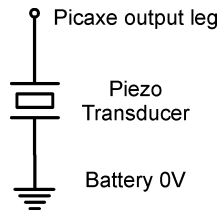


Figure 7 Piezo

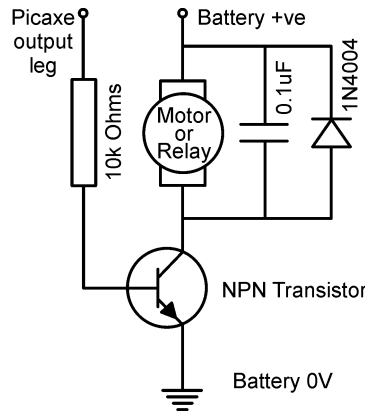


Figure 8 Motor or Relay

Devices may be added to the *CONTROLLER* using available Picaxe output pins. You will need to modify the program to provide suitable output. Some possibilities include:

- LED, such as a bar graph or a seven-segment display.
- Piezo transducer to create sound output.
- Motor (single direction) or relay.
- LCD module for text output.

2.4.5. EXTERNAL COMMUNICATIONS

It is possible to connect the Picaxe to a PC using a two-way serial data link. Unfortunately, you cannot use the programming port except for using the Debug command.

- The Debug command may be useful to obtain real-time information, such as axis position, while debugging your software/hardware.
- To communicate with your PC using serial communications, you will need to use a suitable interface circuit (such as a MAX232 integrated circuit) and obtain or create suitable software for your PC. Please note that this could substantially increase the scope of your project.

2.5. MOTOR DRIVER L293D (IC2, IC3 & IC4)

The L293D motor driver (IC2) contains two H-bridge circuits. Each of the "driver" blocks contains transistors that are configured as electronic switches. Each leg's function is summarised in Table 2.

| Used For | Function | Leg | Leg | Function | Used For |
|--------------|----------|-----|-----|----------|--------------|
| +V | 5V | 1 | 15 | 5V | +V |
| From Picaxe | In 1 | 2 | 15 | In 3 | From Picaxe |
| To Motor "A" | Out 1 | 3 | 14 | Out 3 | To motor "B" |
| 0V | 0V | 4 | 13 | 0V | 0V |
| 0V | 0V | 5 | 12 | 0V | 0V |
| To Motor "A" | Out 2 | 6 | 11 | Out 4 | To motor "B" |
| From Picaxe | In 2 | 7 | 10 | In 4 | From Picaxe |
| +V | V+ | 8 | 9 | 5V | +V |

Table 2 L293D Leg Functions

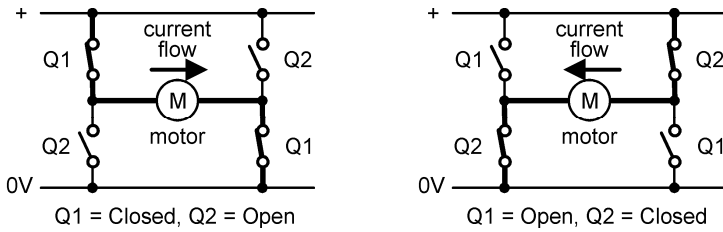


Figure 9 H-Bridge Circuit Concept

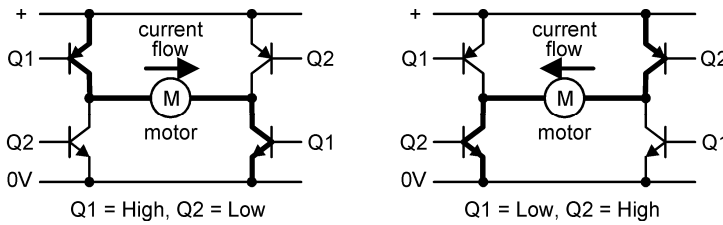


Figure 10 Typical H-Bridge Circuit

- Changing the voltage applied to the input pins changes the direction of current flow through the motor.
- Q1 low, Q2 low - motor stop
- Q1 high, Q2 low - motor forward
- Q1 low, Q2 high - motor reverse
- Q1 high, Q2 high - motor stop
- The motor driver IC is simpler to use and fault-find than an equivalent circuit built using individual transistors and resistors. To view H-Bridge circuits constructed from transistors, see the "Radio Controlled Vehicle", "Wanderer" and "Seeker" teaching units on the Scorpio Technology website.

2.6. POTENTIOMETERS

Each of the five potentiometers is used to measure the angular position of an axis. The potentiometers must be of a "linear" type; a "logarithmic" type is not suitable. The actual resistance value is not critical, since a relative voltage between the power rails (0V and +6V) is being measured. A suitable resistance range is 5k to 50k.

2.7. CAPACITORS

- Capacitor C1 (100uF) smooths battery power caused by motor switching.
- Capacitors C2 to C9 (0.1uF) smooth the power supply close to the ICs.
- Capacitors C10 to C15 (0.1uF) reduce electrical motor noise that reaches the Picaxe.

2.8. POWER

- Power switch SW1 is used to control power to the circuit.
- A red LED (light emitting diode) (L1), in series with a 220R resistor is used to indicate the presence of power on the PCB assembly. The LED intensity may change during operation.
- Diode D1 is used to drop the voltage to the Picaxe and to provide reverse voltage protection.

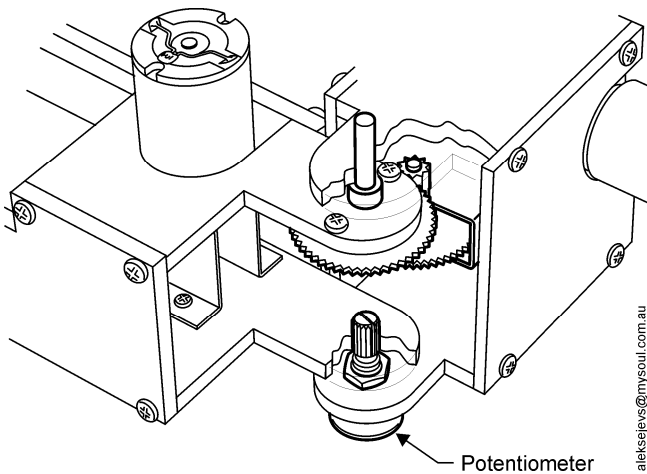


Figure 11 Typical Rotating Axis

3. MECHANICAL CONSIDERATIONS

3.1. PLANNING

Before starting construction, plan and lay out all the components using a suitable computer program or on a sheet of paper. Look at your project as a complete unit, and not just as separate parts.

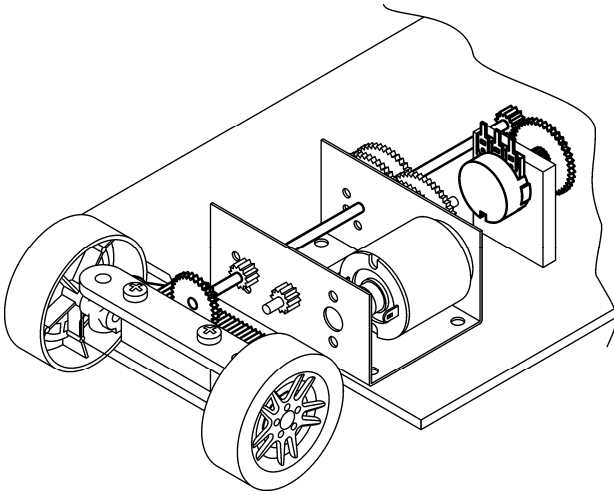


Figure 12 Typical Linear Axis

3.2. POSITION FEEDBACK

3.2.1. ROTATING AXIS

- In a typical light-duty application, the feedback potentiometer can be used as an axle and a position sensing device.
- If the axis rotation is more than 270 degrees, use a suitable gearing system to link the axis with the potentiometer.

3.2.2. LINEAR AXIS

- Position feedback using a rotary potentiometer may be obtained from a linear axis using suitable gearing.
- A linear potentiometer of suitable length may also be used.

3.2.3. POSITION RESOLUTION

- We have found 8 bit ADC resolution (up to 256 discrete locations) to be suitable for a typical rotary or linear axis. (Command: readadc)
- The high resolution ADC mode in the Picaxe can be used to increase the resolution to 10 bits (up to 1024 discrete positions). However, you will need to change the program to use two-byte variables. (Command: readadc10)

4. ASSEMBLY

4.1. MECHANICAL ASSEMBLY

Use the following information to supplement the instructions for assembling the *CONTROLLER*.

- Before starting assembly, solder a 0.1uF capacitor across the terminals of each motor. The purpose of the capacitor is to reduce interference generated by the motor reaching the *CONTROLLER* and external equipment.
- If you wish to mount the LED motor direction indicators on the motors, twist and solder a red LED, a green LED and a 220R resistor (not supplied) between the motor terminals. Connect the LEDs in opposite directions, flat on red LED to "+" terminal, flat on green LED away from "+" terminal. The LEDs will then indicate direction of motor rotation.
- Solder suitable lengths of wire to the motor terminals. If using multi-core alarm cable, allow a sufficient length of wire for each potentiometer.

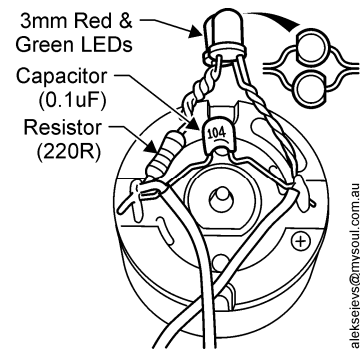


Figure 13 Components on Motor Terminals

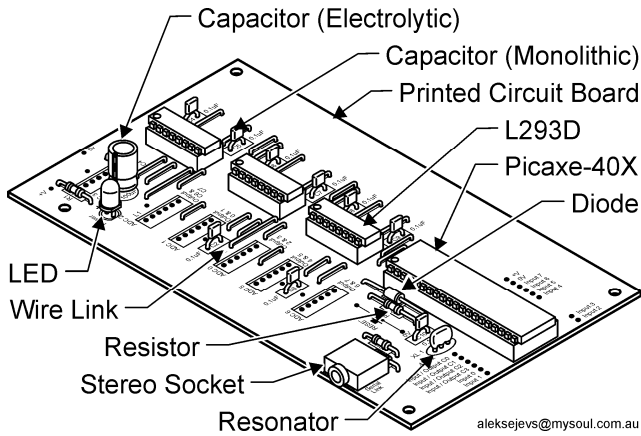


Figure 14 PCB Component Terminology

NOTE: The wires to the *ROBOT ARM* may be soldered directly to the PCB or connected via suitable 6-pin connectors (not supplied). The hole spacing for these connectors is 0.1" (2.54mm).

- Solder the wire links to the PCB. These are shown as straight lines on the overlay. Use tinned copper wire to make the links.
- To the PCB, solder the stereo socket, resistors, diode, monolithic capacitors, IC sockets, LED, electrolytic capacitor and resonator.
- Do not insert the ICs into their sockets yet. These will be inserted during testing.

4.2. PRINTED CIRCUIT BOARD ASSEMBLY

WARNING: Take care in the orientation of IC sockets, LED and electrolytic capacitor. The electrolytic capacitor will be damaged and may cause damage or injury if it is installed in the wrong direction and power is applied.

CAUTION: Unsoldering and replacing damaged or wrongly positioned components will waste time. During soldering, do not overheat the PCB and components.

NOTE: Trim component leads as required after soldering to the PCB.

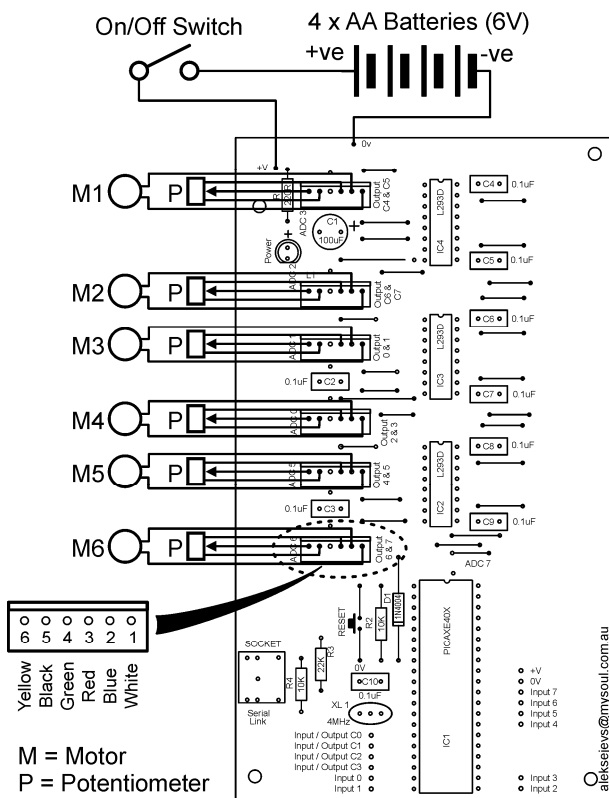


Figure 15 Wiring Diagram

NOTE: The "spare" connection to pin 4 on each connector location allows an additional input or output device, such as a switch or LED, to be connected for each axis using the same length of alarm cable. Connect the track on the PCB to a spare input or output pin on the Picaxe. You will need to modify your program to use the additional device(s).

4.3. WIRING

For wiring details, refer Figure 3 and Figure 15. Use 6-core alarm cable or different coloured wires to assist during faultfinding.

When soldering wires, strip a short piece of insulation from the end of the wire, twist the strands and "tin" them with solder.

- Pass the wiring from the motors and potentiometers, through the *ROBOT ARM* to the PCB.
- If required, use mating connectors to allow the *CONTROLLER* to be disconnected from the project
- Connect the motors and potentiometers as follows (suggested wire colours):
 - Pin 1 = Motor (white)
 - Pin 2 = Motor (blue)
 - Pin 3 = +6V to Potentiometer (red)
 - Pin 4 = Spare (green)
 - Pin 5 = 0V to Potentiometer (black)
 - Pin 6 = Potentiometer Arm (yellow)
- Connect the switch and battery holder (+ve = red, -ve = black).

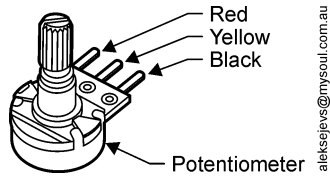


Figure 16 Potentiometer Wiring

4.3.1. POTENTIOMETER WIRING

Connect wires to each of the potentiometers as shown in Figure 16. (You may need to swap the red and black wires during testing.)

4.3.2. MOTOR WIRING

Connect wires to the motors: white to "+" and blue to "-".

5. ELECTRICAL TESTING

- Before applying power, inspect soldering for short circuits and poor "wetting" of component leads or pads.
- Insert four 1.5Volt AA batteries into the battery holder. Move the power switch to "on". Check that the LED illuminates. This shows that power is available.
- If the LED does not illuminate:
 - Check that the battery voltage is above 5.5 volts. (If low, replace the batteries.)
 - Check that the batteries are properly inserted in the battery holder.
 - Check that the LED is the right way around.
 - Check the wiring against the wiring diagram.
- Move the power switch to "off". Check the orientation of the ICs - the end with leg 1 is identified with a notch or dimple at one end. Line up the legs of each IC with its IC socket holes and press down firmly. Do not use the letters/numbers on the IC to identify leg numbers.

NOTE: It may be necessary to bend the IC legs slightly to line them up with the socket holes.

CAUTION: ICs will be damaged if they are installed in the wrong direction or if power supply (battery) connections are reversed.

- Move the power switch to "on". Check that the LED illuminates. This checks that power is available. At this stage nothing will appear to be working.

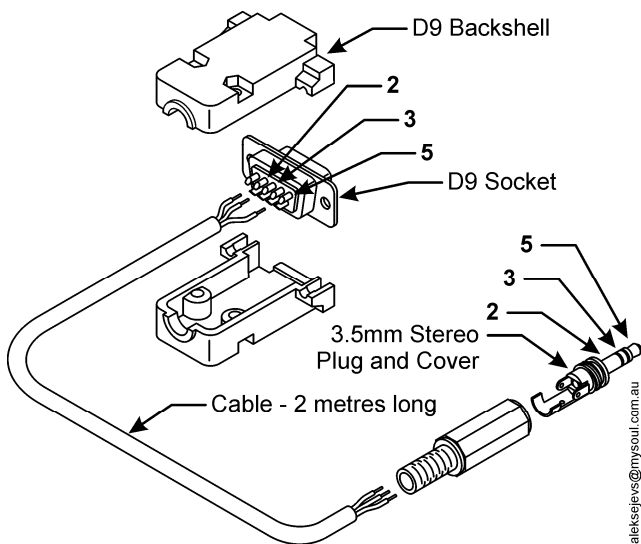


Figure 17 Picaxe Serial Cable Construction (If Required)

5.1. CONSTRUCT PICAXE SERIAL CABLE (IF REQUIRED)

- If required, construct a PICAXE serial cable, as shown.

5.2. INSTALL PICAXE EDITOR

NOTE: PICAXE editor needs to be installed only once. We have used Picaxe editor versions 4.1.10 and 5.1.5.

- Start up and log into your PC. (Some PCs require that you log in as the 'Administrator' to install software. See your system administrator if you do not have sufficient rights to install software.)
- Download and run the installation file from the software page at www.rev-ed.co.uk or from www.picaxe.co.uk.

- Follow the on-screen instructions to install the PICAXE editor.
- If your PC has a 9 pin serial port, insert the PICAXE serial cable into it.
- If your PC does not have a 9 pin serial port, connect a USB to serial adapter and the PICAXE serial cable into a vacant USB port. (Use the Picaxe editor to verify the COM port.)

5.3. START PICAXE EDITOR

- Click Start>Programs>Revolution Education>Programming Editor to start the software.
- If the Options screen does not automatically appear, click the View>Options menu. On the 'Mode' tab select the option for the Picaxe that you are using (this should be one of 28X, 28X1 or 28X2). On the 'Serial Port' tab select the appropriate serial COM port then click OK.
- The PICAXE programming editor software is ready to use.

5.4. EDIT PROGRAM

NOTE: The program listed in this document was created for the five-axis *Robot Arm*. You will need to change the program for your application.

- Copy our program (refer Figure 20) into the PICAXE programming editor software. The program and flowchart are at the end of this document.
- Save the program. You can keep the original file. For changes use different file names.
- Our program is designed to continuously move two objects between three predefined positions. Draw a diagram to describe the sequence of actions as specified in the pair of lookup tables (refer to Program Overview and the Program Listing).

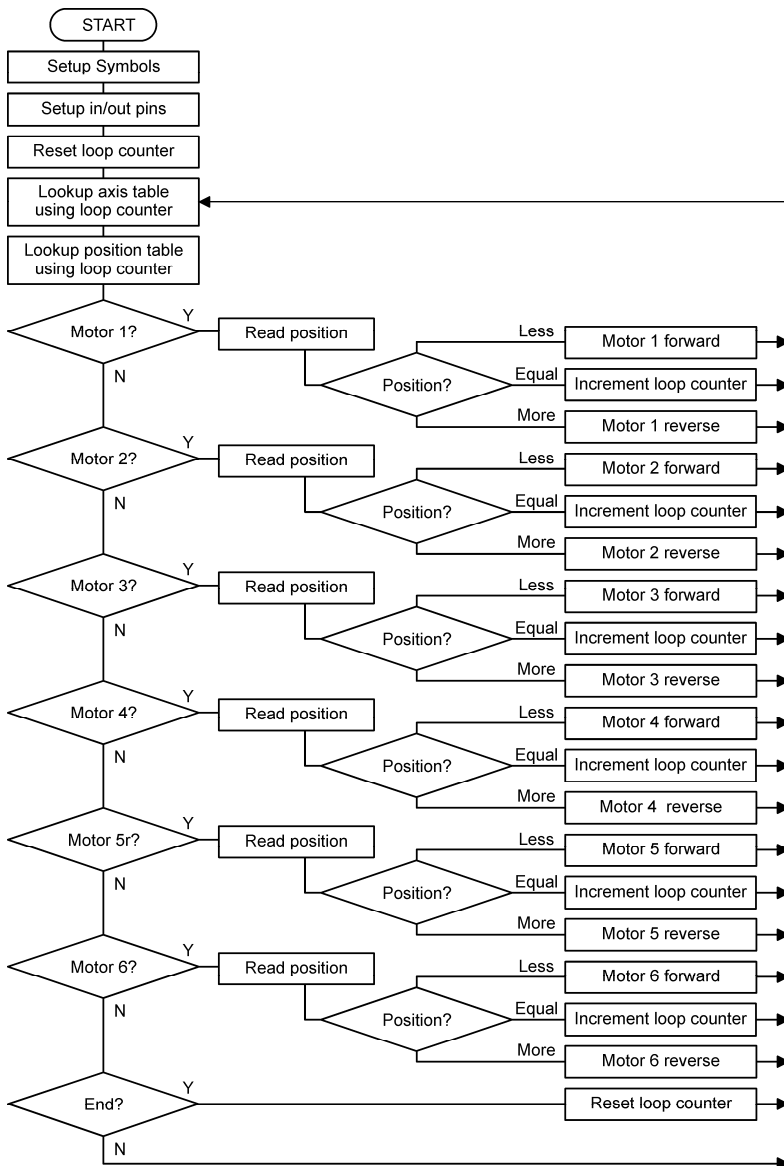


Figure 18 Simplified Program Flow Chart

5.4.1. PROGRAM OVERVIEW

NOTE: For programming language details, from the PICAXE "Programming Editor" help menu, open "PICAXE Manual 2 - BASIC Commands".

The program (refer Figure 18 and Figure 20) contains various sections, as follows:

- Set up variables for axis positions ("symbol" commands). Adjust these values during testing.
- Initialisation ("let dirs", "let dirsc", "let pins" and "let pinsc" commands)
- The main loop consists of a pair of lookup tables and a counter (first lookup table is to select the axis to move, second one defines the axis position and the counter is used to select which pair of values is used). The axis position is an integer in the range 1-255.
- A "branch" statement that directs the program to jump to the relevant axis that needs to be moved.
- A loop of code for each axis, consisting of two motor outputs and potentiometer input. When the loop reaches the desired position, the program selects the next axis to be moved.

5.4.2. MOTORS AND POSITION FEEDBACK POTENTIOMETERS

- The spare motor and its feedback potentiometer may be connected without any circuit modifications. The motor and potentiometer should be the same as those used in this project.
- The voltage present on the spare analogue input must be between 0V and +V. A potentiometer or a variable resistance, such as an LDR (Light Dependant Resistor), should be suitable. Refer to other Picaxe documentation for circuit details.
- A pair of commands needs to be used to control the motors, as shown in Table 3.

| Axis Name | Potentiometer ADC Input | Motor Reverse Commands | Motor Forward Commands |
|----------------------|-------------------------|-------------------------------|-------------------------------|
| Motor 1 | 0 | pins = %00000100 pinsc = 0 | pins = %00001000 pinsc = 0 |
| Motor 2 | 1 | pins = %00000001 pinsc = 0 | pins = %00000010 pinsc = 0 |
| Motor 3 | 2 | pins = 0 pinsc = %01000000 | pins = 0 pinsc = %10000000 |
| Motor 4 | 3 | pins = 0 pinsc = %00010000 | pins = 0 pinsc = %00100000 |
| Motor 5 | 5 | pins = %00010000 pinsc = 0 | pins = %00100000 pinsc = 0 |
| Motor 6 | 6 | pins = %00100000 pinsc = 0 | pins = %10000000 pinsc = 0 |
| Spare Analogue Input | 7 | - | - |

Table 3 Motors and Position Feedback Potentiometers

- To stop all motors, use the following pair of commands: "pins = 0" and "pinsc = 0".

5.4.3. ADDITIONAL DIGITAL INPUTS AND OUTPUTS

Legs C0, C1, C2, C3 may be defined as either inputs or outputs.

- Input 0, 1, 2, 3, 4, 5, 6, 7 (use the "pins" command)
- Input C0, C1, C2, C3 (use the "pinsc" command)

Up to four additional digital outputs may be connected.

- Output C0, C1, C2, C3 (change the value of the "dirsc" in the initialisation section and use the "pinsc" command)

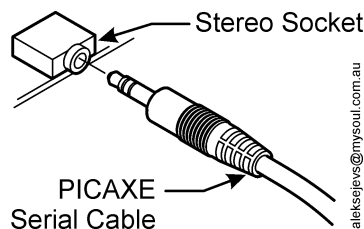


Figure 19 Connect Picaxe Serial Cable

5.5. TRANSFER PROGRAM TO PICAXE

- Move the power switch off.
- Connect the PICAXE serial cable between your PC and the stereo socket on the PCB.
- Move the power switch on. Check that the power LED illuminates.
- Run the PICAXE Programming Editor software and transfer the program to the PCB.
- If required, move the power switch off.
- If required, disconnect the PICAXE serial cable from the stereo socket on the PCB.

6. FUNCTIONAL TEST

NOTE: The functional test can only be performed after the program is loaded into the PICAXE.

- Move the power switch "on".
- The *ROBOT ARM* should go through a sequence of motions.
- If the *ROBOT ARM* does not perform the required sequence, check each axis individually. Create a temporary version of the program. In the "loopa:" section, delete all data from both lookup tables except for three values. Change the first parameter pair to one extent of motion, the second to the opposite extent of motion, and the last pair to "xx". Download the program to the PICAXE and run the program. (Fewer or additional data points may be used.) As required, swap the motor wires and adjust the potentiometer positions by rotating the potentiometer body or changing the data values.

NOTE: The L293D's may become warm with continuous use.

7. FURTHER DEVELOPMENT

We encourage you to change the program after you have determined that our program works with your *CONTROLLER*.

- Develop another sequence of actions for your *ROBOT ARM* to perform. Use a switch to select which routine is used.
- Add a switch input to move your *ROBOT ARM* to a "home" position.
- Add a switch input to "pause" the sequence to simulate an emergency stop situation.
- Add a microswitch (or two) to detect that an object is present in the gripper.
- Add a switch to select between "manual" and "program" modes. Use the 5 switches supplied with the *ROBOT ARM* to select which axis to move.

Congratulations on successfully building and customising your own *ROBOT ARM* under the direction of the *CONTROLLER*!

HAVE FUN!

8. THE PROGRAM

```
'Robot Arm Controller
'(C)2009 PVA Tecwrite - Peter Aleksejevs
'Define constants used to define axes in the lookup tables.
symbol sx = 0 'shoulder
symbol ax = 1 'arm
symbol fx = 2 'forearm
symbol wx = 3 'wrist
symbol gx = 5 'gripper
symbol bx = 6 'spare motor
symbol cx = 7 'spare ADC input
symbol yy = 8 'end of data - wait until switch cleared on input - not implemented
symbol xx = 9 'end of data marker = reset counter
'Define positions of axis preset positions: i=initial, 1="left", 2=mid, 3="right".
'All positions values are 0 to 255, as measured by the axis potentiometer.
'Define position 128 at centre of motion - i.e. for shoulder when arm is pointing straight out.
symbol si = 100 'shoulder initial
symbol s1 = 100 'shoulder left
symbol s2 = 128 'shoulder mid
symbol s3 = 156 'shoulder right
symbol ai = 100 'arm initial
symbol a1 = 100 'arm left
symbol a2 = 128 'arm mid
symbol a3 = 156 'arm right
symbol fi = 145 'forearm initial
symbol f1 = 156 'forearm down
symbol f2 = 128 'forearm mid
symbol f3 = 100 'forearm up
symbol ri = 138 'wrist initial
symbol r1 = 100 'wrist left (w1 is a variable)
symbol r2 = 128 'wrist mid (w2 is a variable)
symbol r3 = 148 'wrist right (w3 is a variable)
symbol gi = 135 'gripper initial
symbol g1 = 114 'gripper closed
symbol g2 = 130 'gripper mid
symbol g3 = 140 'gripper open
'Define slow down distances. To disable, change value to 0. Suitable values are in parentheses.
symbol ds = 0 '(8) shoulder deceleration position.
symbol da = 0 '(6) arm slow down distance.
symbol df = 0 '(6) forearm slow down distance.
symbol dr = 0 '(6) wrist slow down distance.
symbol dg = 0 '(5) gripper slow down distance.
symbol pon = 25 'PWM on time for slow down in milliseconds
symbol pof = 2 'PWM off time for slow down in milliseconds (also add delay in routine)
'variables
symbol counter = b1 'Selects which axis & position value pair is currently used
symbol axis = b2 'Specifies current axis
symbol position = b3 'Specifies desired position of current axis
symbol analog = b4 'Read in value from current axis potentiometer
symbol temp = b5 'Temporary variable (used for deceleration calculation)
'The following line needs to be changed if using switches on port C.
let dirsc = %11111111 'Set all C port pins as outputs. Unused pins can be inputs or outputs.
let pins = 0
let pinsc = 0
counter = 0
loopa:
,
'add code to check switch inputs here! i.e. "Pause" switch/button.
,
'Add/remove comments (') at start of lines as required.
'SINGLE AXIS TEST SEQUENCE (change "sx" and position values as required)
'lookup counter,( sx, sx,xx),axis
'lookup counter,( 50,200,xx),position
'MULTIPLE AXIS TEST SEQUENCE
'lookup counter,(sx,sx,ax,ax,fx,fx,wx,wx,gx,gx,xx),axis
'lookup counter,(s1,s3,a1,a3,f1,f3,r1,r3,g1,g3,xx),position
'PICK AND PLACE SEQUENCE
lookup
counter,(sx,ax,fx,wx,gx,gx,fx,wx,ax,sx,fx,gx,fx,wx,fx,wx,wx,sx,ax,fx,gx,fx,sx,ax,fx,gx,fx,sx,ax,fx,g
x,fx,sx,ax,fx,gx,fx,sx,ax,fx,gx,fx,xx),axis
lookup
counter,(si,ai,fi,r1,gi,g3,f2,r2,a1,s1,f1,g1,f2,r1,f3,r3,r2,s2,a2,f1,g3,f2,s3,a3,f1,g1,f2,s1,a1,f1,g
3,f2,s2,a2,f1,g1,f2,s3,a3,f1,g3,f2,xx),position
branch axis,(loop0,loop1,loop2,loop3,loop4,loop5,loop6,loop7,loop8,loop9)
goto loopa 'The program should never get to this line of code
'shoulder
loop0:
readadc 0,analog
temp = position - 3 'position error (low)
```

```

        if analog < temp then rev0
        temp = position + 3
        if analog > temp then for0
        counter = counter + 1
        let pins = 0
        let pinsc = 0
        goto loopa
rev0:
        let pins = %00000100
        temp = position - analog
        if temp > ds then loopa
        pause pon
        let pins = 0
        let pinsc = 0
        pause pof
        goto loopa
for0:
        let pins = %00001000
        temp = analog - position
        if temp > ds then loopa
        pause pon
        let pins = 0
        let pinsc = 0
        pause pof
        goto loopa
'arm
loop1:
        readadc 1,analog
        temp = position - 3
        if analog < temp then rev1
        temp = position + 3
        if analog > temp then fow1
        counter = counter + 1
        let pins = 0
        let pinsc = 0
        goto loopa
rev1:
        let pins = %00000001
        temp = position - analog
        if temp > da then loopa
        pause pon
        let pins = 0
        let pinsc = 0
        pause pof
        goto loopa
fow1:
        let pins = %00000010
        temp = analog - position
        if temp > da then loopa
        pause pon
        let pins = 0
        let pinsc = 0
        pause pof
        goto loopa
'forearm
loop2:
        readadc 2,analog
        temp = position - 3
        if analog < temp then rev2
        temp = position + 3
        if analog > temp then fow2
        counter = counter + 1
        let pins = 0
        let pinsc = 0
        goto loopa
rev2:
        let pinsc = %01000000
        temp = position - analog
        if temp > df then loopa
        pause pon
        let pins = 0
        let pinsc = 0
        pause pof
        goto loopa
fow2:
        let pinsc = %10000000
        temp = analog - position
        if temp > df then loopa
        pause pon
        let pins = 0
        let pinsc = 0

```

```

        pause pof
        goto loopa
'wrist
loop3:
    readadc 3,analog
    temp = position - 3
    if analog < temp then rev3
    temp = position + 3
    if analog > temp then fow3
    counter = counter + 1
    let pins = 0
    let pinsc = 0
    goto loopa
rev3:
    let pinsc = %00010000
    temp = position - analog
    if temp > dr then loopa
    pause pon
    let pins = 0
    let pinsc = 0
    pause pof
    goto loopa
fow3:
    let pinsc = %00100000
    temp = analog - position
    if temp > dr then loopa
    pause pon
    let pins = 0
    let pinsc = 0
    pause pof
    goto loopa
loop4:
    goto loopa 'Analogue input 4 does not exist in hardware!
'gripper
loop5:
    readadc 5,analog
    temp = position - 3
    if analog < temp then rev5
    temp = position + 3
    if analog > temp then for5
    counter = counter + 1
    let pins = 0
    let pinsc = 0
    goto loopa
rev5:
    let pins = %00010000
    temp = position - analog
    if temp > dg then loopa
    pause pon
    let pins = 0
    let pinsc = 0
    pause pof
    goto loopa
for5:
    let pins = %00100000
    temp = analog - position
    if temp > dg then loopa
    pause pon
    let pins = 0
    let pinsc = 0
    pause pof
    goto loopa
'NOTE: Analogue input 6 is available for use as motor feedback or for general purpose use.
loop6:
    readadc 6,analog
    temp = position - 3
    if analog < temp then rev6
    temp = position + 3
    if analog > temp then for6
    counter = counter + 1
    let pins = 0
    let pinsc = 0
    goto loopa
rev6:
    pins = %01000000
    let pinsc = 0
    temp = position - analog
    if temp > da then loopa
    pause pon
    let pins = 0
    let pinsc = 0

```

```

        pause pof
        goto loopa
for6:
    pins = %00100000
    let pinsc = 0
    temp = analog - position
    if temp > da then loopa
    pause pon
    let pins = 0
    let pinsc = 0
    pause pof
    goto loopa
loop7:
'NOTE: Analogue input 7 is available for general purpose use.
'
    readadc 7,analog
    goto loopa
loop8:
    'need to add code here to wait for switch release.
    counter = 0
    goto loopa
loop9:
'NOTE: Resets counter to restart cycle, 'xx' must be last value in lookup table
    counter = 0
    goto loopa

```

Figure 20 Program Listing