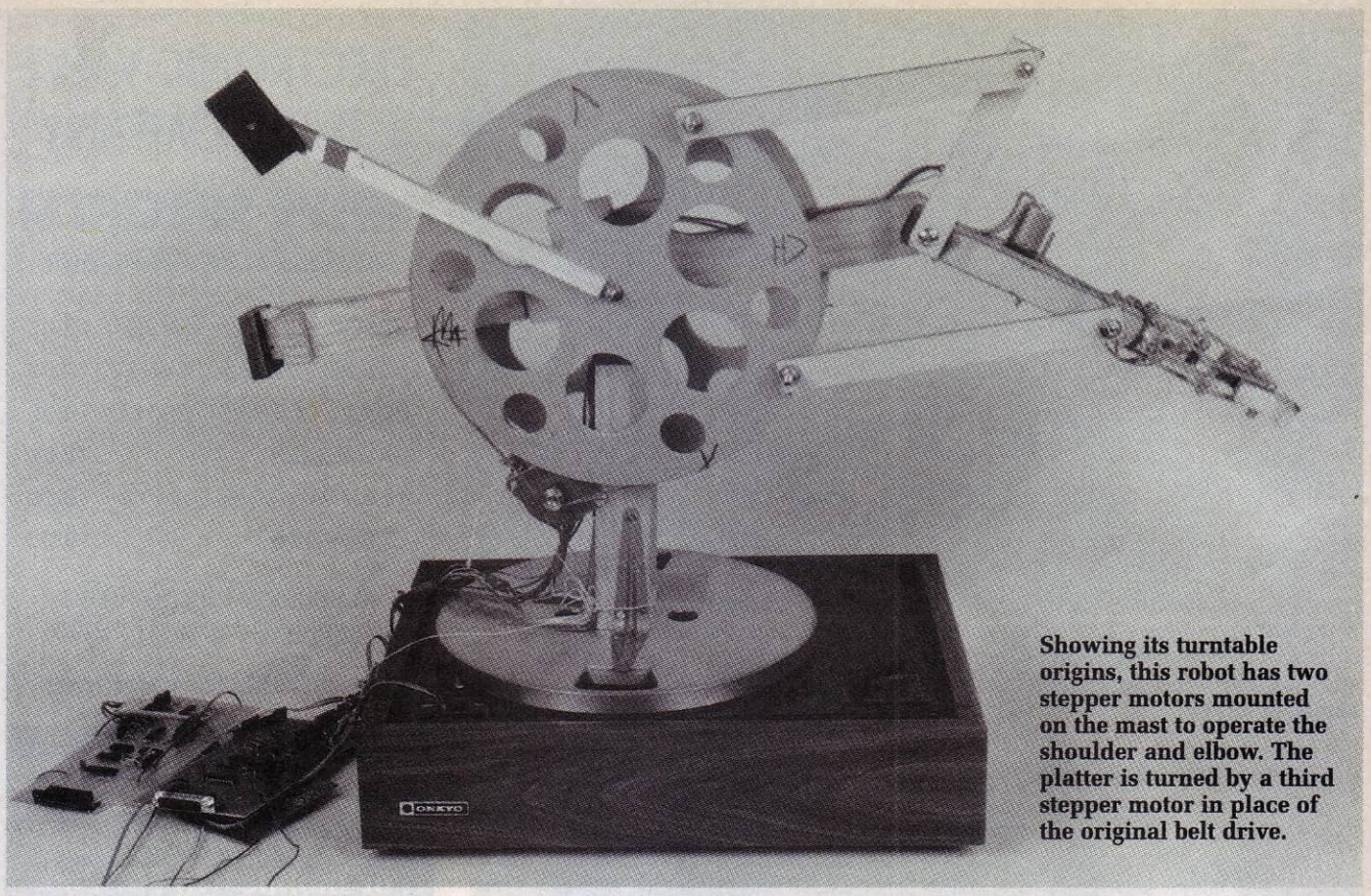


Showing its turntable origins, this robot has two stepper motors mounted on the mast to operate the shoulder and elbow. The platter is turned by a third stepper motor in place of the original belt drive.

Build a PC-controlled robot from surplus parts

What looks like a pile of timber and electronic scrap yet can be controlled by a computer? Answer: a robot based on the stepper motor drivers described in the January 1994 issue of SILICON CHIP. It is a cheap and cheerful introduction to robotics using readily available parts and surplus stepper motors.

By TONY MERCER*

Designed as a practical exercise in robotics and PC control for TAFE students, this robot uses software written in Visual BASIC. It is interfaced to a PC's printer port and, using on-screen menus, is controlled with the keyboard and mouse.

The robot presented here is a demonstration unit only, as an example of what is possible. It is a 3-axis device with a waist, a shoulder and an elbow which has an attached gripper. The waist is made from a record turntable which can be cheaply obtained from a secondhand shop. The shoulder and elbow driver arms are made from discs of customwood 300mm in diameter and belt-driven by two servos. The gripper mechanism is made from 24-gauge galvanised steel scrap and is powered by a small geared motor.

The stepper motor article in the January 1994 issue of SILICON CHIP was quite comprehensive, particularly in regards to stepper motor technology (back issues are available at $7 including postage.) Kits for the stepper motor board are currently available from Altronics in Perth – phone (09) 328 1599.

This project makes use of two of these boards, one to drive two stepper motors and the other to handle a third stepper motor and up to four sole-

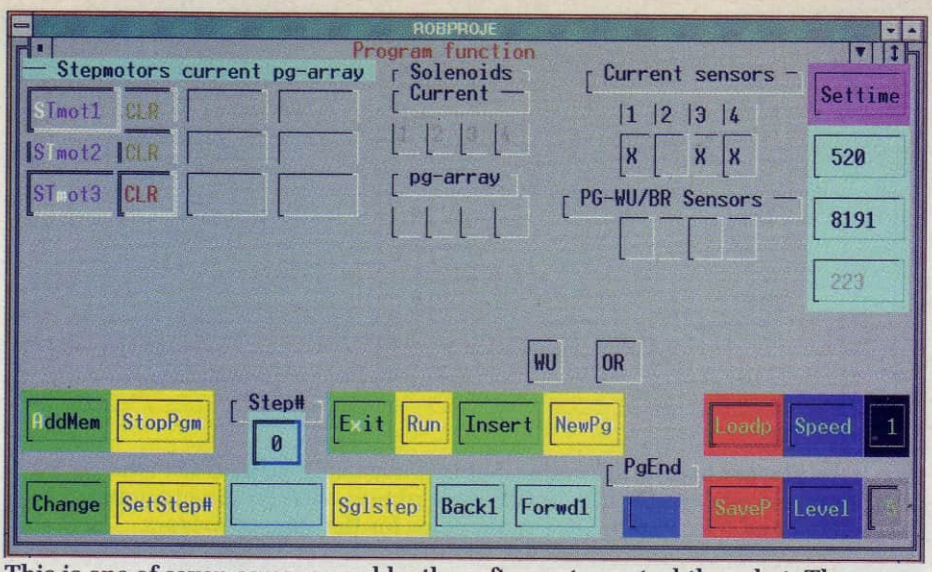
noids. You can also opt to use just one stepper board to drive two stepper motors or one stepper motor and four solenoids. Depending on what approach you take, some changes will be required, as detailed in the section headed "Solenoid Test".

Apart from being able to actuate motors and solenoids, it is also possible to connect up to five different sense lines. Four sense lines are used in this project. An individual sensor line will have a 10kΩ resistor pulling it to the +5V rail on the driver board – see Fig.1. In this case, the line will normally be high or a "1". If the line is brought to ground, it will be low or a "0". This can be done using a switch or an open-collector transistor, as shown in Fig.2.

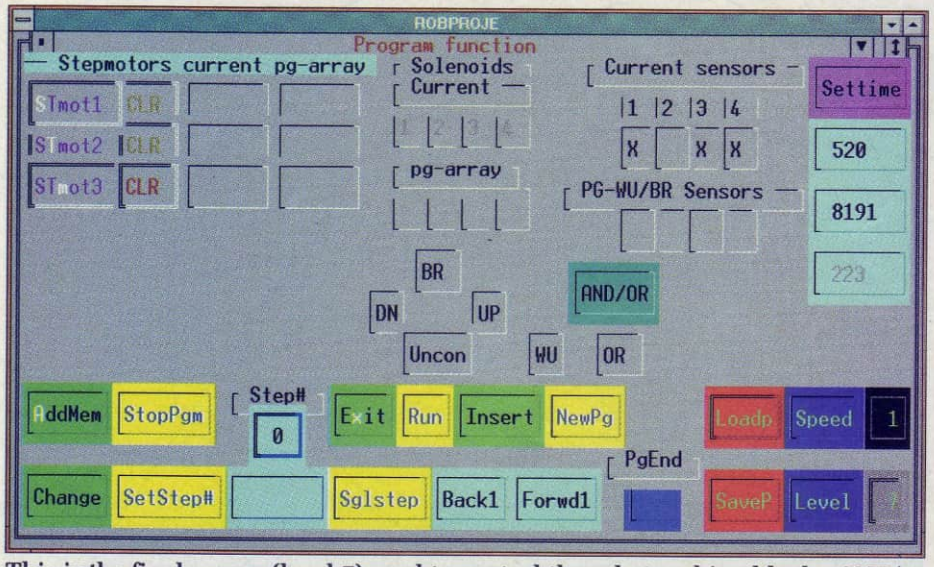
We can not only step the motors, and be confident of their final positions, but can also sense the result of these actions or any other sensory input we might be interested in. It may be that we are using the robot to lift something from the flat car of a model train, for example, but not until the flat car is in position. The robot software can be programmed to wait until this happens and then to proceed from there.

We can also use the solenoid outputs to turn a DC motor on in either direction and use the sensors to sense when an action has been completed.

The boards are connected in daisy chain fashion via a length of 25-way ribbon cable. A 25-pin IDC male plug is used for the computer connection and two 25-pin IDC female plugs for the stepper drivers. The plugs are attached to the cable as shown in Fig.3.



This is one of seven screens used by the software to control the robot. The different levels add program features in a way which makes it less confusing for the novice (level 5 shown).



This is the final screen (level 7) used to control the robot and it adds the AND/OR function. The various levels are stepped through by clicking on the level box at the bottom righthand corner of the screen.

Software

The software handles the operation of the robot and includes diagnostic screens for testing the stepper boards' operation, as an aid to debugging the system.

As pointed out in the January 1994 article, a stepper motor is designed to rotate a specified distance when a pair

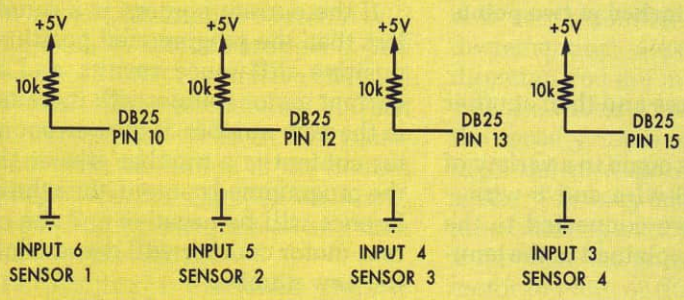


Fig.1: individual sensor lines will have a 10kΩ resistor pulling them to the +5V rail on the driver board. In this case, the line will normally be high or a "1"

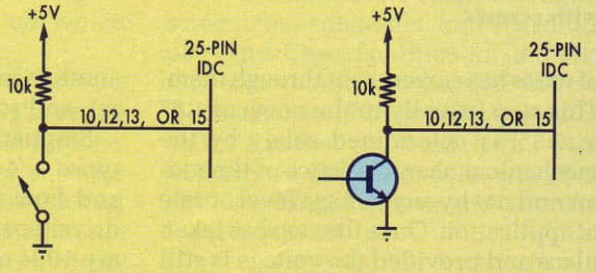


Fig.2: the sensor lines can be pulled low using a switch or an open-collector transistor, as shown here.

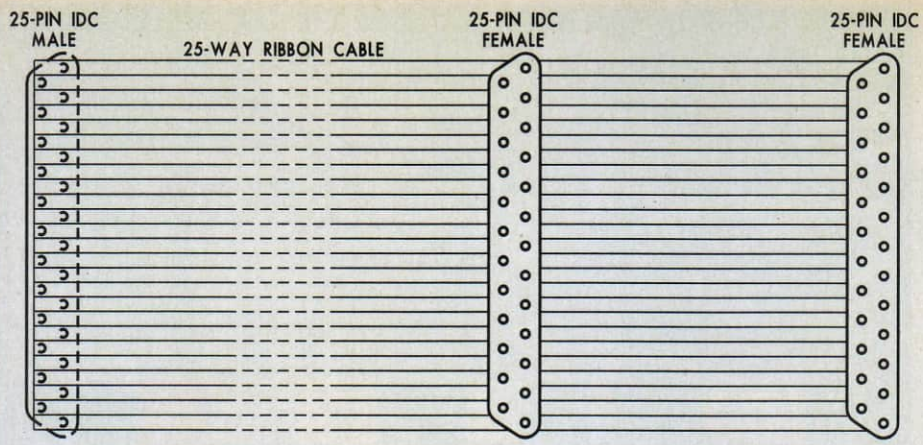


Fig.3: the stepper boards are connected in daisy chain fashion via a length of 25-way ribbon cable. A 25-pin IDC male plug is used for the computer connection and two 25-pin IDC female plugs for the stepper drivers.

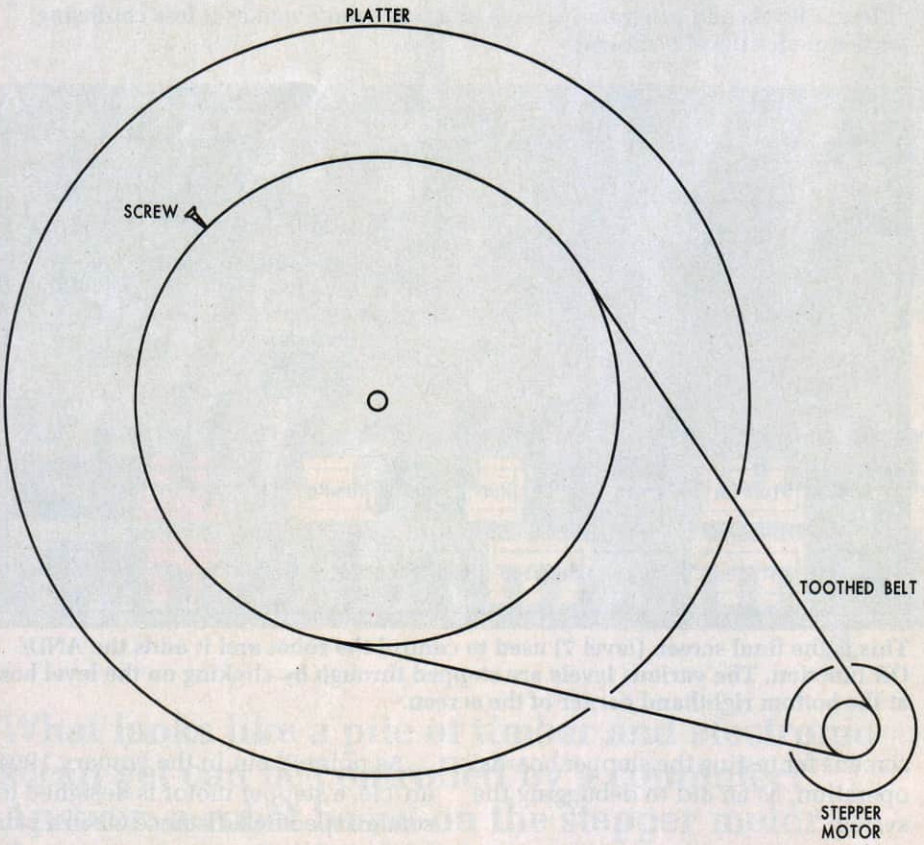


Fig.4: the waist of the robot uses a toothed belt driven by a stepper motor. Since the platter will not need to rotate any more than 270° or so, the belt will not need to wrap around the entire circumference but can be attached at two points with screws.

of wires has current sent through them. This step (usually in the range of 1.8° to 7.5°) is determined solely by the mechanical characteristics of the motor and not by any voltage level or rate of application. Once the step has taken place and provided the voltage is still applied, the motor will be locked in position.

To obtain another step from the motor, a voltage has to be applied to another set of wires and then another set, and so on.

Stepper motors come in a variety of types – 4-wire, 6-wire and 8-wire – and how these are connected to the driver boards is explained in the January 1994 article.

The software moves the robot arms by pulsing the stepper motors a certain number of times. Provided that there is no slippage, the arms should go to the position required. However, stepper motors do have inertia. If a loaded motor is presented with a pulse sequence that is too fast, it will just hunt back and forth. Similarly, if a motor running at maximum speed is suddenly deprived of its pulse train, it will tend to run on.

While no damage is likely, the program will lose vital positional data and think that the arm is somewhere other than where it actually is. The software needs to take care of this.

The way to control the speed of a stepper motor is to vary the rate of its driving pulses. To accelerate the motor to its final running speed, the pulse rate is slow at first and then increases. Deceleration is the reverse procedure.

If the software is instructed to move an arm to a specified position, it needs to know where it is and where it is to go to. To achieve this, the software uses two registers, called the current position and the programmed position. The software compares these two resisters and computes a difference. It will then issue a number of pulses to the designated stepper motor. Once done, it will look at other current and programmed positions and repeat this operation until there are no more differences.

The current register contains only stepper motor positions and solenoid and sensor status. The program position is a 2-dimensional array, one dimension holding the new required position (which when done will become the new current position) and the other dimension a list of all the future positions.

When the software is run, it starts by comparing the contents of its current register with the register contents of the first location in the array. It will look at stepper motor one and if it sees a difference it will take action to reduce this difference to zero.

If the current content is a number less than the programmed position, a positive difference results and the current motor counter will increment to the new number. If the current motor content is a number greater than the programmed content, then the difference will be negative and the current motor counter will decrement to the new number.

When this is done the next motor is interrogated and so forth. When all the motors are positioned, the software will look at the solenoids. As the

solenoids can only be on or off, it will merely turn on those that are required and turn off those that are not.

Next, the input sensors are interrogated. By now it should be clear that the program is running a set of positional data contained in the 2-dimensional array. Each new program position contains a complete set of positional requirements for each of the stepper motors and solenoids. There's a great deal more in the programs, as will become apparent later in this article.

Teaching the robot to move

As noted above, the control program has a series of seven on-screen menus (Level 1-7) and you control the actions of the robot with the keyboard and mouse. When the robot starts, the screen will be in Level 1. From this screen you will be able to manually move the motors, select motor speed, select a higher level, observe the current motor position, set the base motor timing and Exit the program.

Before setting the position of any of the motors, you first need to set the speed. Because stepper motors are critical of pulse rate, it needs to be carefully set. Each pulse is a result of a series of internal program steps, updating the screen, etc. However, computers operate at different speeds depending on whether they have a 286, 386, 486 or other processor and if they have the turbo facility on or not.

To overcome this variation in computing speed, we need to set a variable in the program. As you view the main program screen you will see a "set timing" button in the upper righthand corner. Clicking on this will cause the program to test its internal timing and produce a number unique to this configuration. When the motors are now actuated you should see a fairly consistent speed.

When the program is started it will automatically set the base speed. You need only alter it if you have changed the status of the turbo facility. You can also change the speed in five increments with the button at the bottom righthand corner. Clicking on it will increment it up to 5 (fastest) and then back to 1 (slowest).

To position a particular stepper motor, click on the one you want and the screen will change to provide further instructions. Briefly, the left mouse button is pressed to move the motor clockwise while the right mouse button is pressed to move in the other direction. Pressing any key on the keyboard will return you to the main menu. While the motor is moving you will see a counter incrementing or decrementing, depending on which direction you are moving.

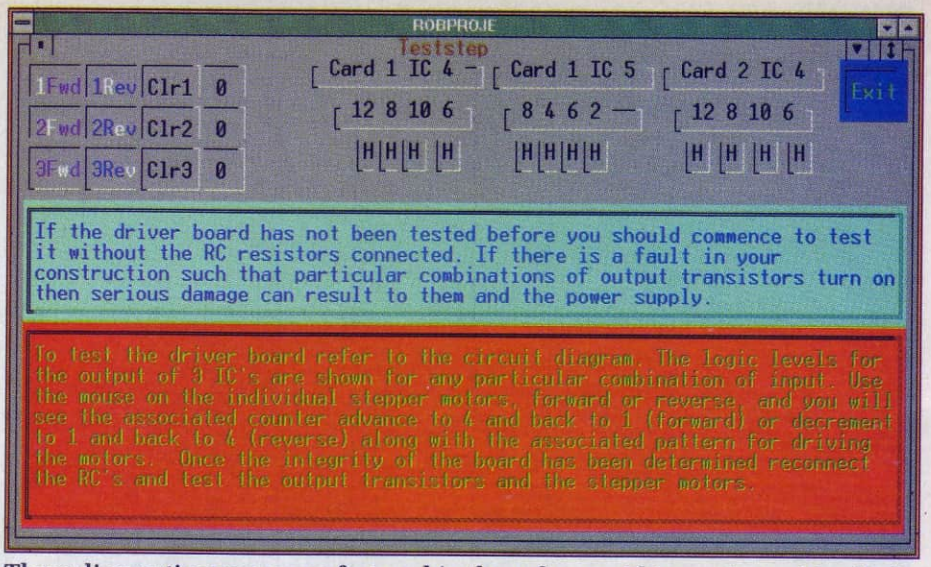
If the motor movement is erratic, the speed you are using might be too high. Select a slower speed and try again. You may also experience erratic motor operation because the load is too high, the voltage applied to the motor too low or the current limiting resistors (if used) on the stepper board are too large.

Once back in the main menu you can either select another motor or you can exit. To leave the program, click on the Exit button and you will return to the beginning menu. To exit altogether, select Exit and you will be returned to the DOS prompt.

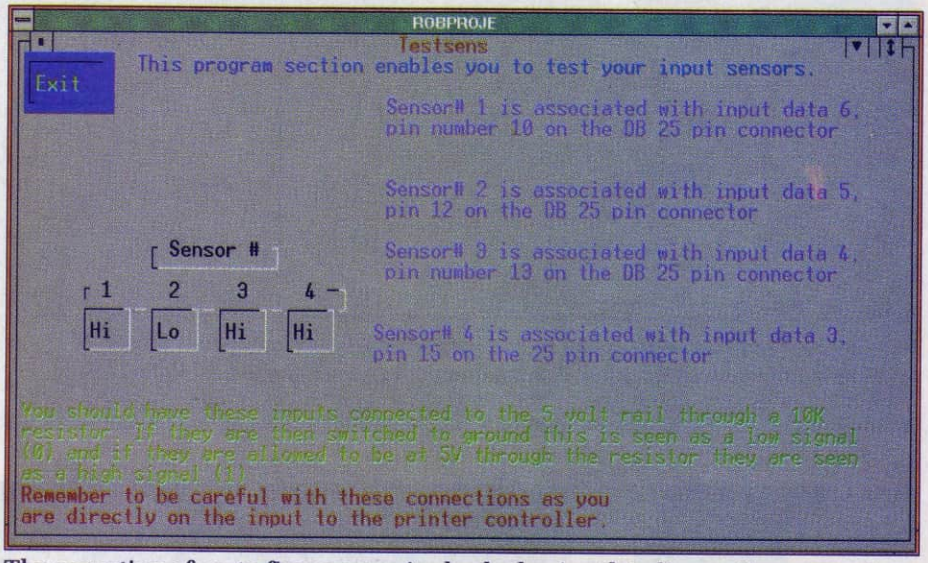
Level 2 adds more functions to the screen: four solenoids, four sensor inputs and Clear facilities for the current step motor locations. If you require a solenoid to actuate, just click on the one that you want. If you want to disable a particular solenoid, click on the solenoid button and you will see it toggle off.

Home position

To the right of the stepper motor button is the CLR button which will clear the contents of the current regis-



Three diagnostic screens are featured in the software. This one is used to check the operation of the stepper motor driver boards which were described in the January 1994 issue of SILICON CHIP.



The operation of up to five sensors is checked using this diagnostic screen. A third screen is used to check the operation of the solenoids.

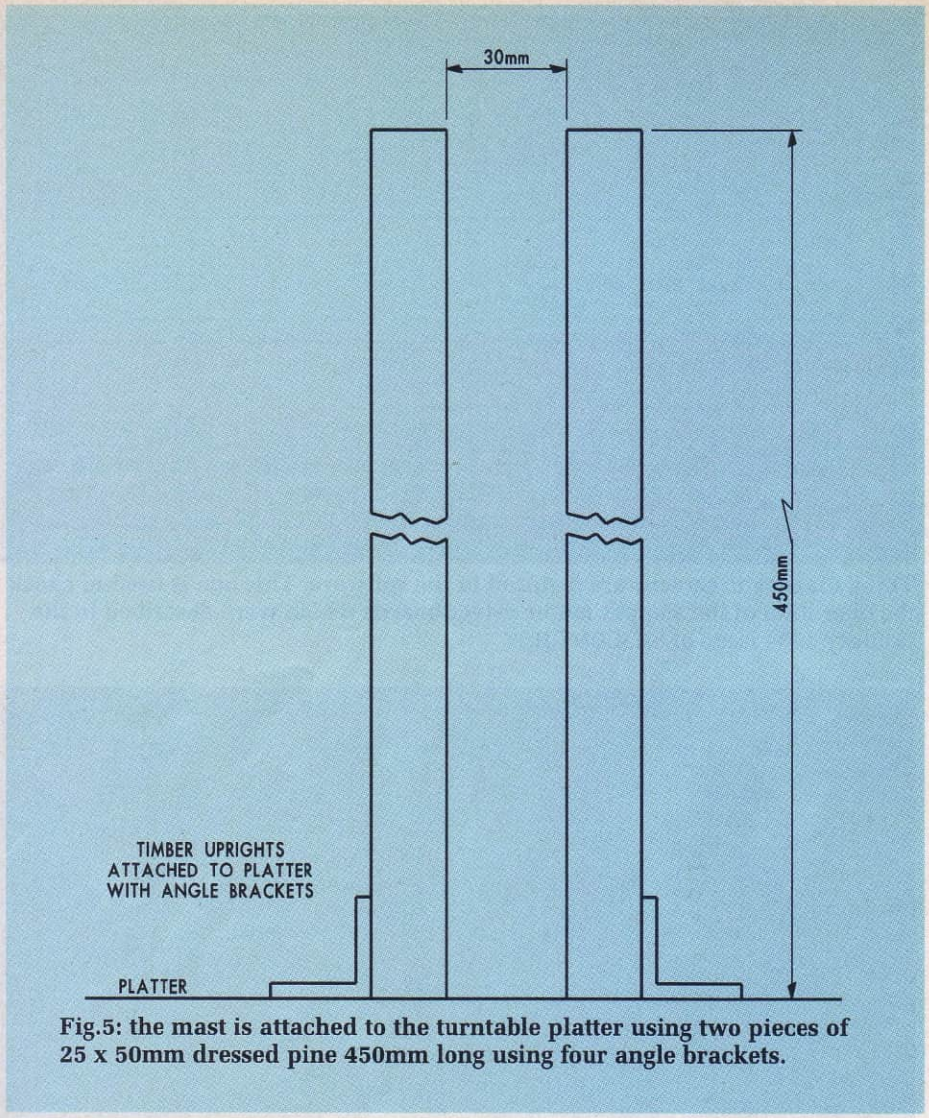
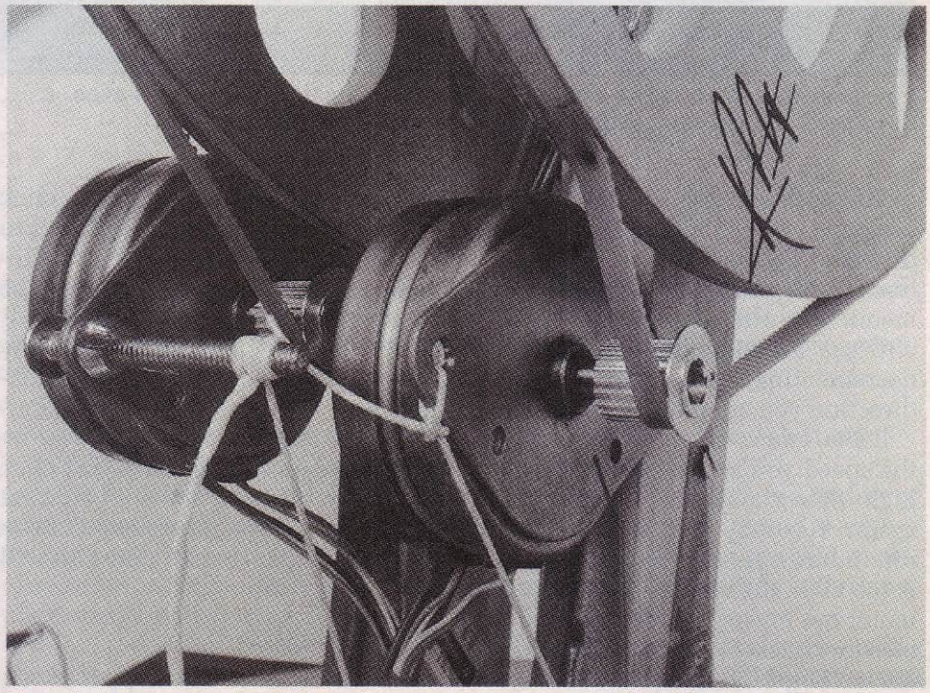


Fig.5: the mast is attached to the turntable platter using two pieces of 25 x 50mm dressed pine 450mm long using four angle brackets.



Two steppers are mounted on the mast to operate the shoulder and elbow discs via toothed belts.

ter for this motor to zero. When you first start to use the program you must "home" the actuated arms. You do this by moving the arms to the mid-point of each arm's travel.

For repeatability, you should mark this "home" position with a pencil. On return to the main menu, you press the clear button and this current location will be "home" or zero. Be careful that you do not clear the register after this as you will confuse the program logic.

Because of this, the clear function is hidden from view until the level control (bottom righthand corner) is incremented to 2 . This is done so that you do not accidentally click on it.

Level 3 adds the facility to store settings in memory. The new buttons are AddMem, NewPg, Run, SetStep and StpPg. NewPg clears the memory prior to a new program. SetStep sets the step number to 1 so that the program can start from here. Stp Pg stops the program but only after all the individual locations have been interrogated and Run commences operations.

You can add the new location(s) to memory, which is what the new positional array is called, by clicking on the AddMem button. The speed information is also loaded. You will see the step number and program end labels increment and the program location change to the current value. Repeat this as often as you need with this and the other motors.

Before writing a new program you should list on a piece of paper the moves you want to take place and include on this the actual numbers for each position, the solenoid and sensor status and branch and wait conditions.

Once the program is being run and you are debugging it there may be changes that you will want to make. Level 4 has several development tools for this. These comprise six new buttons, as follows:

Single Step (Sglstep) allows the program to execute one complete step and then you will see the contents of the next position that the program will go to; the new stepper motor positions, solenoid and sensor requirements and Branch and Wait Until. Pressing the button again will cause the program to perform these operations and you will be presented with the contents for the next step.

If you want to change any of this,

use the Change button. Clicking on this button brings up a screen that tells you to click on the function that you want to change. The selected function will not alter its state but merely load the new state into the current program step.

Move Forward increments the step number and displays the next lot of contents. No other action will take place. Move Backward does the same thing, only in the reverse direction.

The Insert button allows you to insert new locations into the program. In this case, the program end counter and step number will increment and the insertion will assume the current step number.

Save program

When you want to save the program click on the SaveP button and a copy of the program array will be loaded onto the default disc.

Level 5 adds the Wait Until facility. With this you can stop program execution until a selected condition is sensed. You need to click on the sense input that you want 'high' for the program to continue. At this level you can 'OR' up to four sense lines. The program will advance when any of them become high.

Level 6 adds the Branch function. You will see the current step number appear in the box below the BR button. Click on the UP or DN buttons to tell the software where you want the program to branch to and the Unco button for conditional or unconditional branch. As for the Wait Until function, the conditional branch will occur when any of the sense lines you have selected goes high.

Level 7 adds the AND/OR facility. In the AND case, the Wait Until or Branch will not occur unless all the selected sense lines go high.

In order to help in the debugging of the electronics there are three diagnostic screens: Stepper Motor Test, Solenoid Test and Sensor Test.

Making the robot

As noted previously, the waist is made from a record turntable which can be cheaply obtained from a secondhand shop. Remove the tone arm mechanism and the drive motor which is replaced with a stepper motor. The stepper can be coupled to the platter using a toothed belt, as depicted in Fig.4.

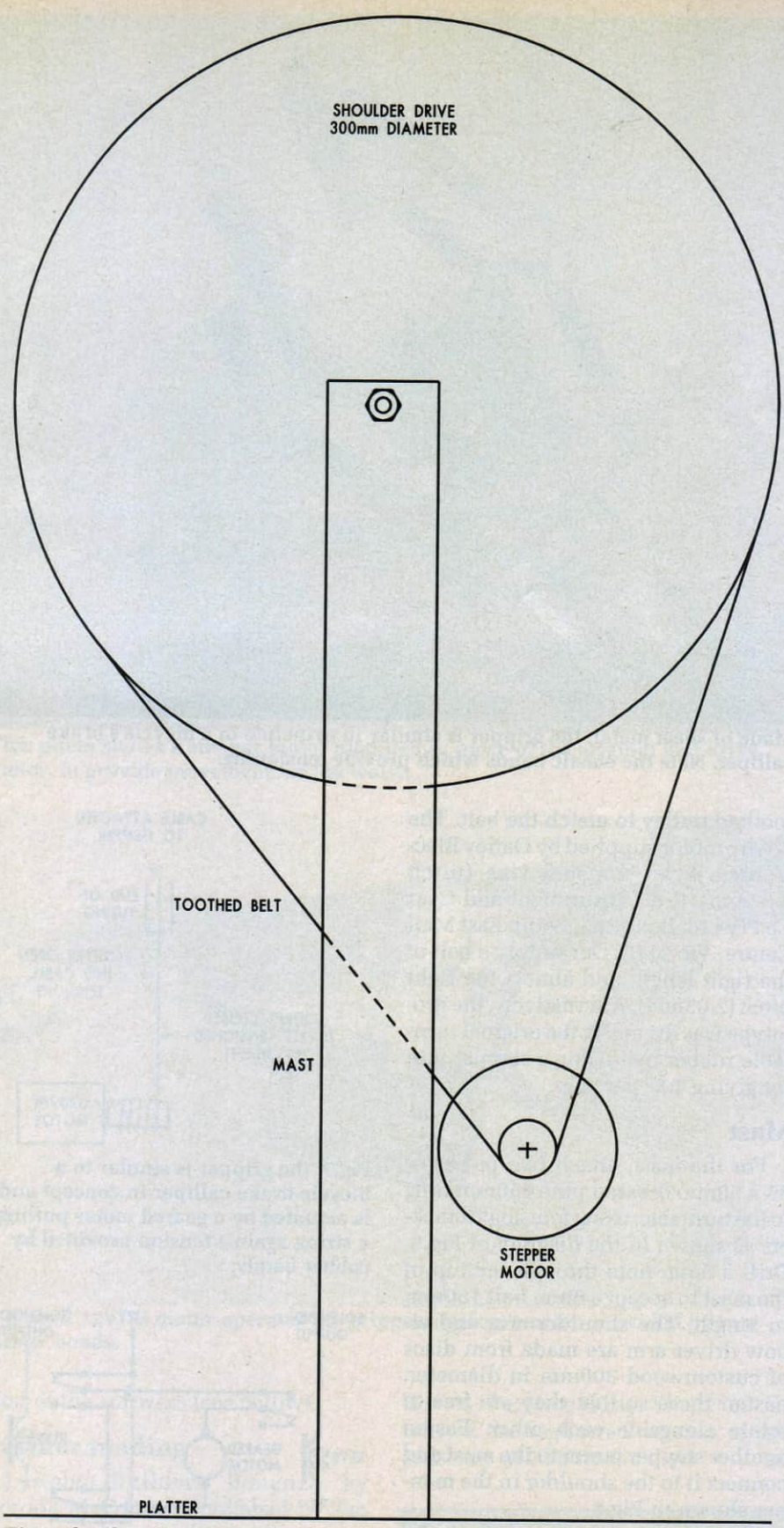
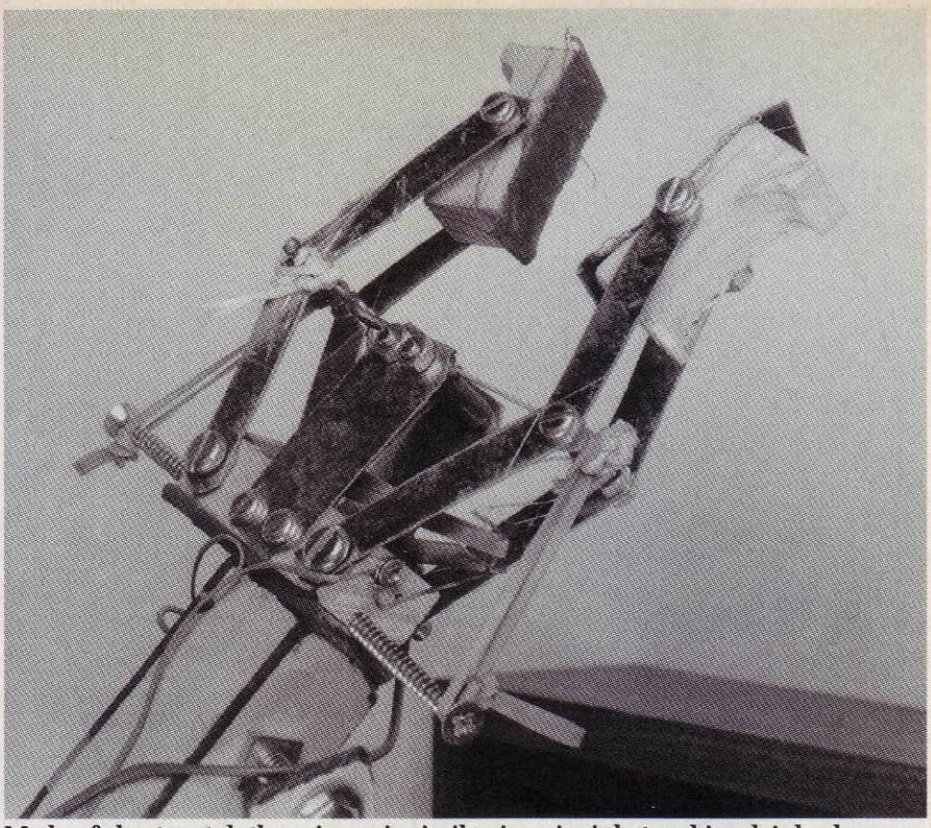


Fig.6: the shoulder arm and elbow driver arm are made from discs of customwood 300mm in diameter. These are driven by steppers and toothed belts.

Since the platter will not need to rotate any more than 270° or so, the belt will not need to wrap around the entire circumference but can be attached at two points with screws.

The stepper motor will need a



Made of sheet metal, the gripper is similar in principle to a bicycle's brake calliper. Note the elastic bands which provide tensioning.

toothed pulley to match the belt. The 6-wire motor supplied by Oatley Electronics is an unusual size (pitch 2.07mm). R & I Instrument and Gear Co Pty Ltd, Box 1302, South East Mail Centre, Vic 3176, can supply a belt of the right length and almost the right pitch (2.03mm). Alternatively, the prototype was driven by the original turntable rubber belt using a stepper motor giving 1.8° per step.

Mast

For the mast, attach two pieces of 25 x 50mm dressed pine 450mm long to the turntable, using four angle brackets as shown in the diagram of Fig.5. Drill a 6mm hole through the top of the mast to accept a 6mm bolt 150mm in length. The shoulder arm and elbow driver arm are made from discs of customwood 300mm in diameter. Fasten these so that they are free to rotate alongside each other. Fasten another stepper motor to the mast and connect it to the shoulder in the manner shown in Fig.6.

The resulting movement of the shoulder will be something less than 180° but this was not found to be a problem. Again the mechanical connection to this will need to be determined by you. Note that there will be more load on this axis than on the waist, considering that we are actually going to lift something.

To this 300mm disc attach a piece of 25 x 50mm dressed pine so that the reach is extended by 150mm. This is powered by a stepper motor in the same manner as for the shoulder. From 150mm pieces of light timber construct a box section as shown and connect the driving arms at right angles. The distance between the outer points and the axis will need to be the same as the dimensions on the 300mm disc.

Now, using light dressed timber, make two driving arms and fasten them to the box section. The driving arm lengths need to be the same as the distance between the centre of the 300mm half and the pivot point of the elbow.

Other methods could be used to mechanically attach the stepper motor to the arm. For instance, a length of threaded rod can be connected to the stepper motor shaft and the arm connected to this via a threaded nut.

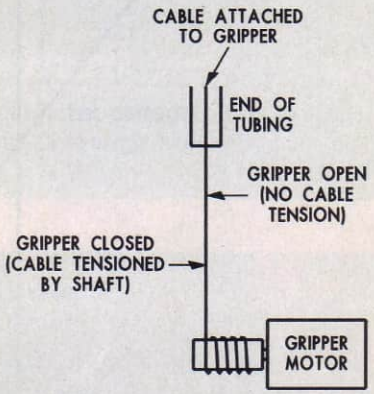


Fig.7: the gripper is similar to a bicycle brake calliper in concept and is actuated by a geared motor pulling a string against tension provided by rubber bands.

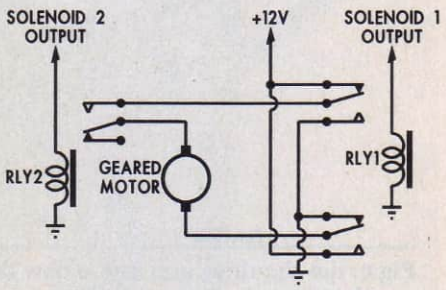


Fig.8: two relays connected to the solenoid outputs provide for control of the gripper motor. RLY2 connects voltage to the DC motor while RLY1 controls the motor direction.

Gripper

The gripper was made from 24 gauge galvanised steel sheet. It is similar to a bicycle brake calliper in concept and is actuated by a geared motor pulling a string against tension provided by rubber bands – see Fig.7. The mass of the gripper and its motor are counterbalanced by weights at the other ends of the arms.

Using a piece of Veroboard and two relays build the circuit of Fig.8. One of the relays, RLY2, is actuated by the solenoid 2 output and will connect a voltage to the DC motor. The other relay, RLY1, is driven by the solenoid 1 output so that the motor direction can be forward or reversed.

Two switches will be attached to the first two sensor inputs; the closed switch to sensor input 1 and the other to sensor 2. One will be 'high' when the gripper is open and when it is closed the other will be 'high'.

Once the robot is completed, you are ready to program it to pick up something by using the manual position and remembering buttons. To use the gripper, the following sequence may be of help.

(1). Select the sensor being used to determine grip closure.

(2). Click on the Wait Until and OR control.

(3). Select the solenoid that governs the direction of motor movement. Whether or not this is set or reset depends on how you have wired the relays and the polarity of the motor drive supply.

(4). Select the solenoid that turns the gripper motor on. The motor should start to move. You may like to wait until the sensor indicating the gripper is closed goes high and then turn the power off to the motor. This should prevent any damage to the mechanism if you are not quick enough to add this to memory and then turn the solenoid off.

(5). Add to Memory.

(6). When the the sensor for gripper closure comes on (goes high), the gripper is closed. De-select the solenoid that turns the gripper motor on. The gripper motor should stop. Reapply power again if you had already disconnected it.

(7). Add to Memory.

Position the gripper and load to where you want it to be, using the methods as described before.

Open gripper

To open the gripper the following may prove useful.

(1). Select the sensor that indicates that the gripper is open. Make sure that you de-select the sensor that indicates gripper closure and the other two sensors. Failure to do this will result in no operation when the Wait Until function is set, as you can not have the gripper open and closed at the same time.

(2). Click the Wait Until and OR operation.

(3). Select the solenoid that governs gripper motor direction.

(4). Select the solenoid that causes the gripper motor to operate. The gripper motor should now be running and allowing the gripper to open. Again, you may want to disconnect power as for the closing sequence.

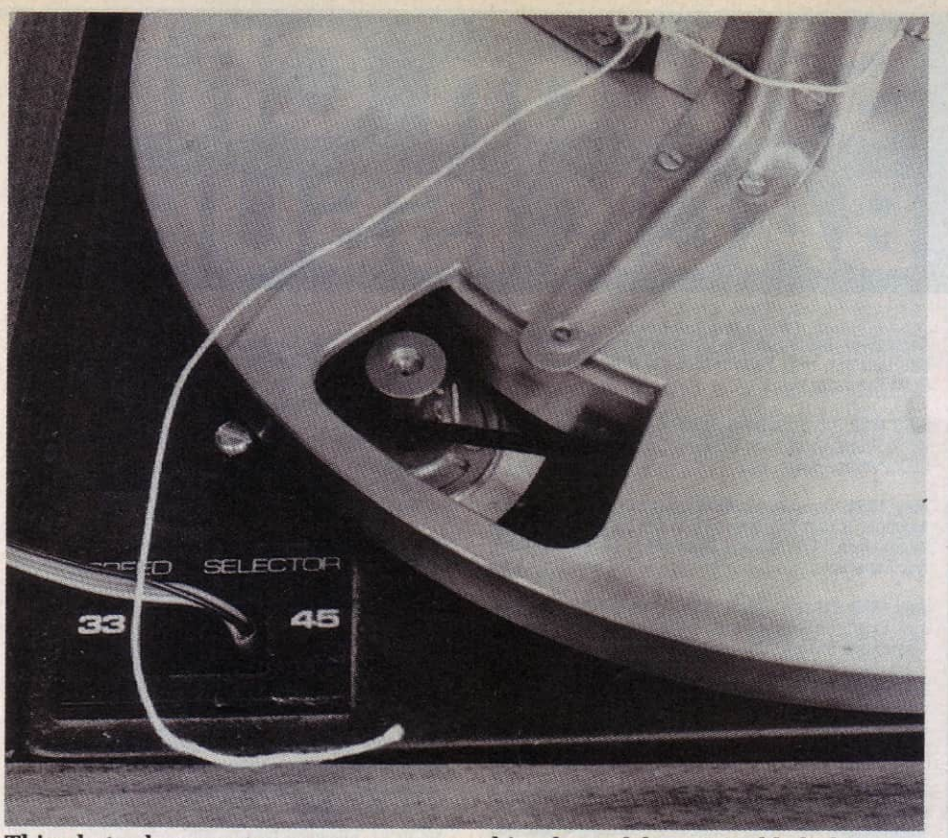
(5). Add to Memory.

(6). When the indicator that shows the gripper has opened comes on de-select the solenoid that powers the gripper motor. Reapply power if you went down this path.

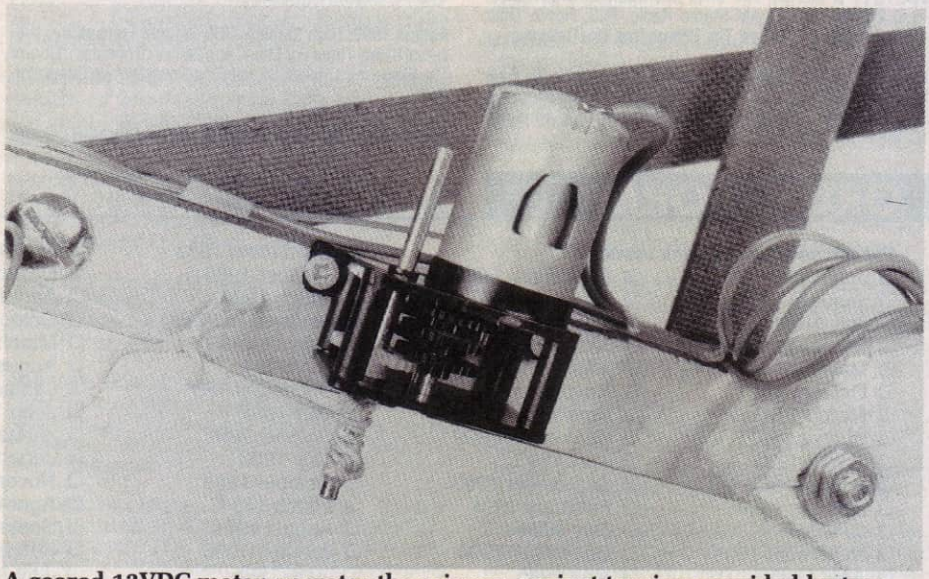
(7). Add to Memory.

Position the gripper for another operation or branch back to repeat the sequence.

A simple gripper open/close function is included in the registered version of the software (see below).



This photo shows a stepper motor mounted in place of the original belt drive motor, to provide movement for the waist.



A geared 12VDC motor operates the gripper, against tension provided by two elastic bands.

Further reading

(1). Robot Builders' Bonanza, by Gordon McCombs. Published by Tab Books.

Software availability

Shareware versions of this software can be obtained by sending $8 to NewTech Education Resources, PO Box 61, Ferntree Gully, Vic 3156. Details of the registered version of the software will be on the disc. **SC**

* Tony Mercer is a lecturer in technology studies at the Hawthorn Institute of Technology and can be contacted during office hours by phoning (03) 9810 3279.