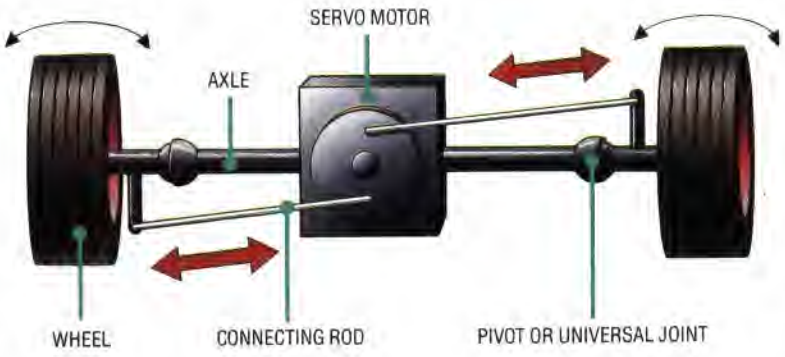
Steering Mechanism



In our Workshop robot, directional control is achieved by independent control of two bi-directional stepper motors. A possible alternative to this arrangement is to have one stepper motor driving the vehicle, and a servo motor to steer the vehicle, in much the same way as a driver steers a car. The diagram shows a servo motor mounted directly over the wheel axle, pushing or pulling connecting rods to the wheels to steer the vehicle. An alternative arrangement would be to use a rack-and-pinion steering mechanism, mounting the pinion gear on the servo motor's spindle

programs. The first listing is for control of a single servo motor connected to one of the user port lines. The event timer is set up using BASIC. An initialising procedure assembles the event-handling routine before the main program runs it by enabling event 5.

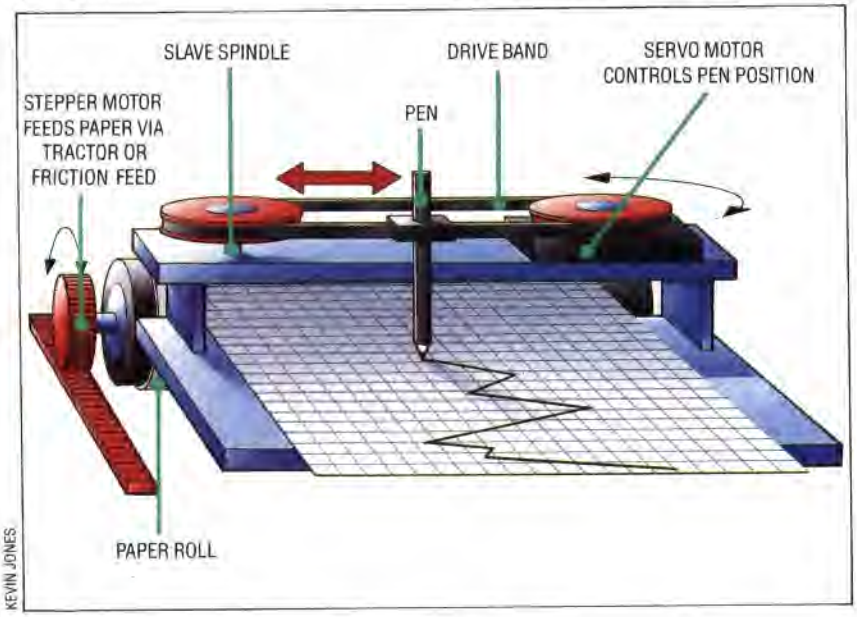
On the BBC Micro, the operating system includes a centisecond user timer. By setting it to 2 centiseconds (two interrupts of 10 milliseconds each) and then using the 'event' vector (see page 464 of the BBC Advanced User Guide), the processor will jump to the pulser code at the correct time. Since the operating system was designed to use the events, the program needs merely to return (RTS) from the subroutine, and not use RTI.

The second listing, for multiple servo control, first uses a BASIC loop to initialise the look-up table with $FF values. If each element of the table was output to the user port in turn, the pulses would all continue for two milliseconds. However, exceptions can be made, and each motor turned off in turn. The exceptions are inserted into the table starting with motor 7, by using an offset (in the X register) proportional to the length of the pulse. Then the table is output to the user port by addressing each element in turn indirectly, this time using post-indexed indirect addressing (where the processor adds the value of the Y register to the address found in a zero page vector).

The bit patterns (exceptions) that must be sent to the user port to control the motors are produced as follows. Binary 01111111 is required to turn off motor 7; likewise 10111111 for motor 6; 11011111 for motor 5; and so forth. These are generated by loading the A register with $FF and clearing the carry flag. Then, as the routine handles each exception in turn, these bits are rotated once to the right. On the first rotation, the carry bit is moved to bit 7, bit 7 moves to bit 6, and so on, with bit 0 replacing the carry bit. The bit pattern required to turn off each motor is temporarily saved on the stack. Each table entry is ANDed with the previous one (as it is output) to ensure that once a pulse is turned off, it remains off.

Having generated the machine code, event 5 is cleared for action. Thereafter, pressing the Shift key along with a number key from 1 to 8 selects one of the motors, while pressing keys 1 to 9 moves the motors into position.

Chart Recorder
A chart recorder can be designed using a servo motor to move the pen back and forth, while a stepper motor feeds the paper underneath the pen. Angular movement of the servo motor spindle is translated into the linear back-and-forth movement of the pen by way of a taut drive band. The pen movement corresponds to changes in a vertical axis variable — for example, temperature or barometric pressure — with the continuous paper feed often corresponding to the passage of time



KEVIN JONES

USING THE LISTINGS

To use the BBC listings, simply type them in, as shown, SAVE them, and then RUN. Both the Single Servo and Multiple Servo listings are run with the Common Initial Routine (lines 10 to 750).

For the Commodore 64, the second algorithm uses the same command keys as the BBC listing. The BASIC Calling program allows the position of each motor to be set independently: the Shift key and a number key from 1 to 8 select the desired motor, and a key from 1 to 9 defines the required position.

If you have an assembler, then type in the source listing and assemble it into an object file that can be subsequently loaded by the BASIC Calling program. Alternatively, type in the BASIC Loader for the machine code and RUN this. Type NEW before loading and running the BASIC Calling program. If you use the BASIC Loader then lines 30 and 40 can be omitted.