Stepping motor control: another application of microcomputers

R. Cassinis*

Stepping motors are becoming even more important in many fields of mechanical machines, including industrial robots. Although interfacing between stepping motors and digital control systems is usually considered quite an easy task, suitable drives must be used in order to achieve top performance from the motors. The aim of this paper is to show how a microcomputer-based drive allows higher operating speeds than conventional drives, and, in many cases, the optimization of motor movements. The drive consists of only three chips, and is connected to a standard power amplifier. The paper deals with all the problems (programming, interfacing, reliability etc) that are related to the system. Experimental results, obtained on the SUPERSIGMA robot, are also included

As a part of the research on robotics that is being carried out at Milan Polytechnic Artificial Intelligence Project, a system was developed to drive the stepping motors employed in the SUPERSIGMA robot (Figs 1–3) which may be applied virtually anywhere a stepping motor has to be used. A general knowledge of the control system of the robot[1,2] may help in understanding the interface of the drives to higher level hardware, but it should be noted that both hardware interfacing and communication protocol may easily be changed according to need.

The aim of the paper is then to describe this system and its features which are well above those offered by similar commercial devices. Special emphasis is given to the firmware that was implemented which allows high flexibility, thus making the system suitable for very different applications. The paper also describes in detail the hardware structure of the system and the experimental results that were obtained.

The problem

The system being described is a part of SUPERSIGMA robot which has been designed mainly for assembly operations. An accurate analysis has led to the conclusion that these operations, when performed on a cartesian coordinate robot, do not need coordination between motors. The system was then designed as a set of completely separate microcomputers, each one driving a single motor. Linear interpolation between two or more motors is however possible. A short overview of the global system will be given.

Since it is impossible for a single computer to directly control all the devices of SUPERSIGMA, the intelligence that drives the robot has been split into two levels: the higher one (Central Control Unit or CCU) has complete control of the lower one (microcomputers). The communication between the CCU and the microcomputers is in the form of commands from the CCU towards the microcomputers and of data and status signals in the opposite direction.

The communication protocol was chosen so that it would minimize hardware requirements. The need to minimize hardware suggests the use of a bus structure which reaches all microcomputers (Fig 4). The CCU executes the robot program, which is written in suitable high-level languages by sending commands and receiving data from microcomputers. SUPERSIGMA employs stepping motors for moving the arms: although these

*Milan Polytechnic Artificial Intelligence Project, Piazza Leonardo da Vinci 32, I-20133 Milan, Italy

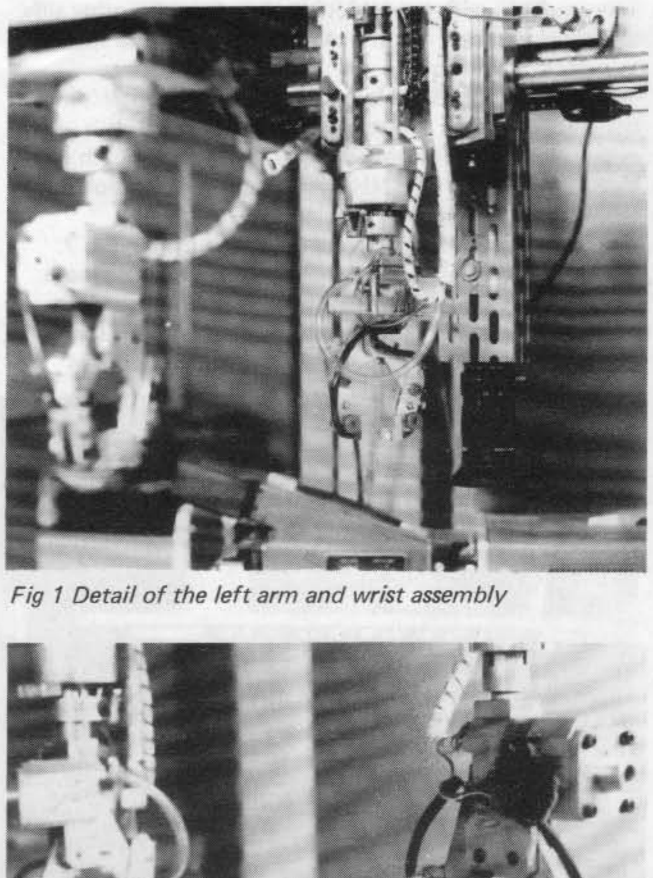

Fig 1 Detail of the left arm and wrist assembly

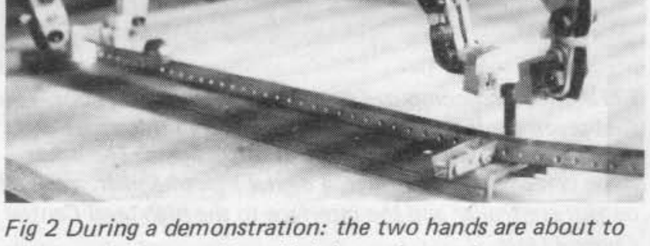

Fig 2 During a demonstration: the two hands are about to lift the whole assembly which is on the work-top

motors do not require feedback, and stability problems do not arise, they are quite difficult to drive because they need to be carefully accelerated to their working speed in order to avoid loss of synchronisation between the rotor and the stator's magnetic field.

The same care must be taken for decelerating and stopping them. For the purpose of obtaining the best performance from each motor, the speed-time curve must have a smooth transition between acceleration and constant speed since this is a critical region where losses of synchronisation may occur (shaded areas in Fig 5). It is, however, desirable to have the possibility of using various acceleration and deceleration curves to accommodate, for instance, different loads. The need to obtain such complex and often parametric curves was the main reason why it was decided to use microcomputers.

The heart of each stepping motor drive is an INTEL 8748 single-chip microcomputer. 8748 is an 8 bit microprocessor with 2.5 μs cycle time which contains 1 kbyte EPROM, 64 byte RAM and 27 I/O lines in addition to on-board oscillator, clock and timer circuits.

The 8748 only needs a 5V low current (about 100 mA) power supply and is designed to be an efficient controller as well as an arithmetic processor. The logical block architecture is shown in Fig 6. The motor drive program is written in EPROM and I/O lines are used to connect 8748 with the MICROBUS and, on the other side, to a programmable counter which produces pulse trains that drive the motor.

The INTEL 8253 is a programmable counter/timer designed for use as a microcomputer peripheral. It is organized as three independent 16 bit counters, each with a count rate of up to 2 MHz. All modes of operation are software programmable.

The internal structure of 8253 is shown in Fig 7. COUNTER 0 receives from CLKO a 2 MHz square wave signal and produces (OUTO) a pulse train that is sent to the power amplifier used to drive the stepping motor. COUNTERS 1 and 2 are used to count the number of

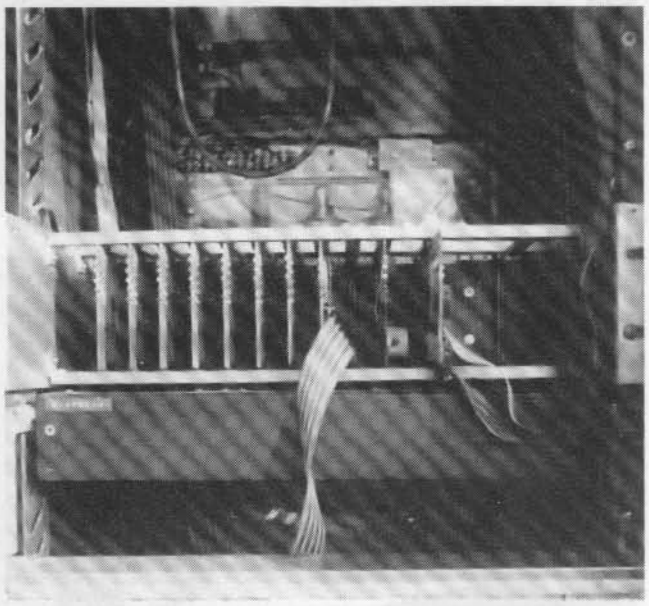

Fig 3 The microcomputer array which drives the robot. The first seven cards on the left are described in this paper and drive the seven stepping motors of Supersigma. The other three cards are, left to right, a digital input handler, a digital output controller and the interface to the high-level Central Control Unit

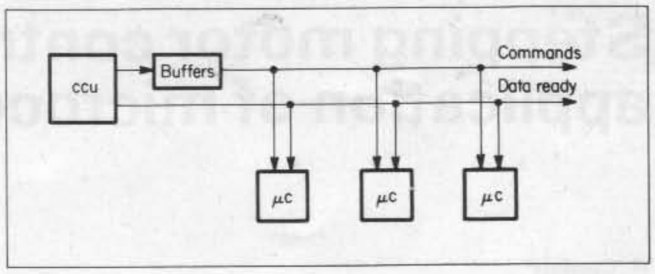

Fig 4 Microbus commands path

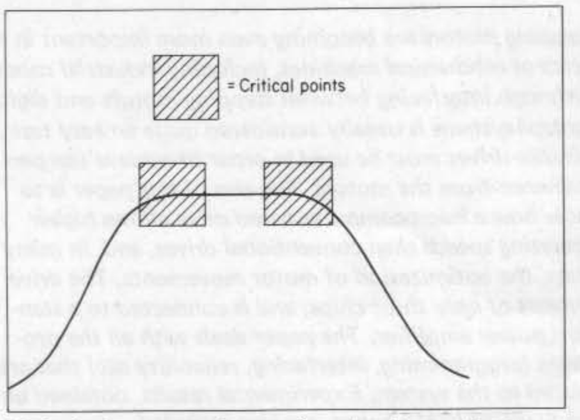

Fig 5 Typical movement curve for stepping motors

pulses generated by counter 0, ie the number of steps performed by the motor from the beginning of the movement. Two different counters were needed because the internal organization of 8253 does not allow the counter that is used to count steps for long movements to be used when very short movements (one or two steps) are required. The two counters are then programmed and read in two different ways and the appropriate one is selected for each movement. Unlike traditional counters, no problems arise if the counter must be read during the counting operation.

Once the counter 0 has been loaded with the number N supplied by the microcomputer, it generates a pulse train whose frequency is

$$V = \frac{F_{in}}{N+1} \qquad (1)$$

where F_{in} is the input frequency. The output signal will remain at the same frequency until a new number is loaded.

In our system, the output frequency from 8253 counter 0 is then

$$V = \frac{2 \times 10^6}{N+1}; N = \text{INT} \quad [(\frac{2 \times 10^6}{V} - 1) + 0.5] \qquad (2)$$

where INT is the function that yields the integer part of its argument.

Fig 8 shows the general block structure of the motor drive system. As it was shown, the communication path between the CCU and the microcomputers is implemented in a very simple way. From a hardware point-of-view, the connection is also very simple (Fig 9). One of the three on-chip bidirectional ports of 8748 is used for MICROBUS connection; no interface is needed between the chip and the transmission line. A testable input (TO) is directly connected to /DATA READY line and one bit of I/O port number 3 is connected to /ANSWER READY line. Two output bits (P23 and P24) are used to drive status lines. A complete description of the communication protocol is given[1,3].

The connection of each microcomputer to MICROBUS is always the same, no matter what function is assigned to the microcomputer. It is also possible to build all stepping motor drive boards in the same way, regardless of the functions that are assigned to them. The only differences between microcomputers are of course in the firmware running in them.

Transducers that feed motors are of standard Olivetti production and require two signals to be fed to their input: the first one indicates the desired sense of rotation and the second one causes the motor to step every time a pulse is applied to it. The microcomputer and the programmable interval timer generate the appropriate pulse trains.

Firmware

The hardware described in the previous section must now be programmed in order to achieve the expected performance. It must be said that firmware was developed for this specific application, and is not completely general. Many simple modifications may, however, be done to tailor the system to other applications.

The program running on the microcomputer may be divided into three main blocks: command reception, ramp generation and auxiliary functions. Command reception is performed by a very simple routine that receives commands from MICROBUS and transmits appropriate parameters to the main program. Commands are two to five bytes long, according to the function to be performed. The meaning of each byte is explained in Table 1. Since all microcomputers are connected in parallel to MICROBUS the first byte is used to allow each microcomputer to understand whether a command is directed to itself or not. The second byte specifies the function to be performed and the direction of the movement (if applicable). The third and fourth byte specify the number of steps to be performed. Two bytes are sufficient, since no movements in SUPERSIGMA exceed 2^{16} steps. The fifth byte indicates the speed of the movement, and is used only by some special functions which will be described in the following.

The main portion of the program is devoted to ramp generation. As emphasised above, the greatest care must be taken to generate smooth accelerations and decelerations and these must be suited to the number of steps to be performed. The nature of the circuit is such that, in order to accelerate and decelerate the motor, suitable numbers N must be loaded in the programmable rate generator at appropriate time intervals. These numbers may be generated either by an iterative algorithm that computes each number or by scanning a table of pre-computed numbers. The latter method was chosen because it allows changes to the acceleration curve to be made easier.

Table 1 Meaning of each byte of a command

Byte	Contains
1	Number of the microcomputer to which the command is directed
2	Op-code
3	MSB of the number of steps to perform (if required)
4	LSB of the number of steps to perform (if required)
5	Speed (if required)

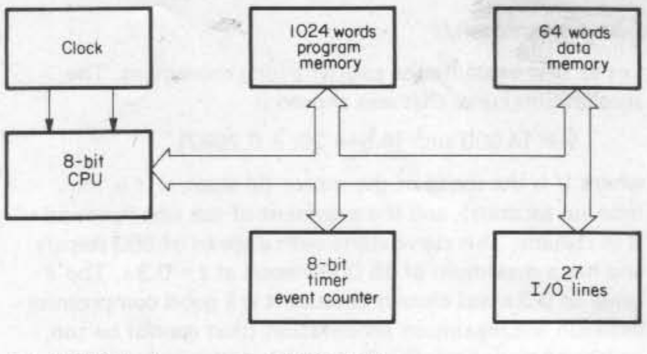

Fig 6 8748 microcomputer diagram

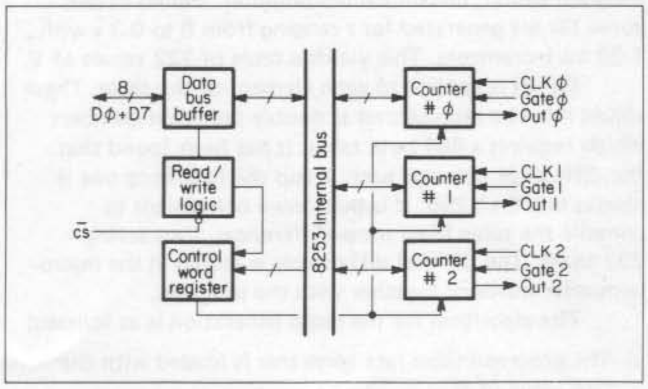

Fig 7 8253 programmable timer-counter block diagram

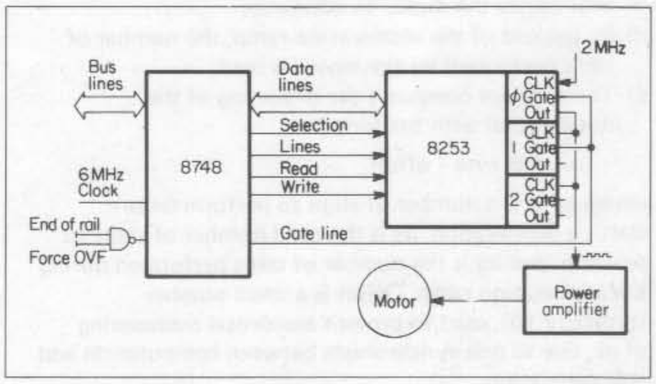

Fig 8 Stepping motor drive block diagram

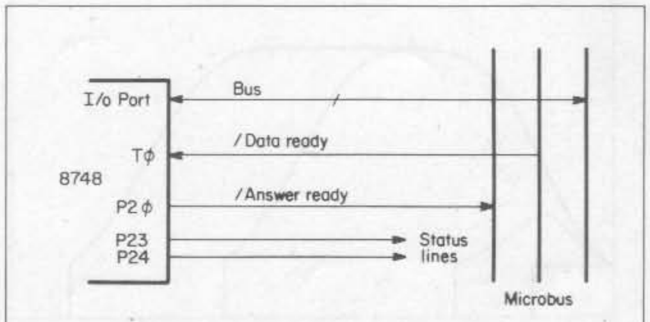

Fig 9 Microcomputer connection to MICROBUS

Long movements

Let us first examine the case of a long movement. The acceleration curve that was chosen is

$$V = 14\,000 \sin^2 (4.544\,26t + 0.2097)$$

where V is the speed of the motor (in steps/s), t is the time (in seconds), and the argument of the sine function is in radians. This curve starts with a speed of 650 steps/s and has a maximum of 15 000 steps/s at $t = 0.3$ s. The value of 0.3 s was chosen because it is a good compromise between the maximum acceleration (that cannot be too high) and resonance phenomena (that require a not-too-long ramp).

Some preliminary work must be carried out while preparing the program. This work is done by a FORTRAN program which runs on a minicomputer. Values of the curve (3) are generated for t ranging from 0 to 0.3 s with 1.29 ms increments. This yields a table of 232 values of V.

Eq (2) is applied to each element of the table. These values must be represented as double precision numbers which requires a 464 byte table. It has been found that the difference between each N and the following one is always less than 256: it is then more convenient to compile the table from these differences, thus saving 232 bytes. The table of differences is stored in the micro-computer memory together with the program.

The algorithm for the ramp generation is as follows:

a) The programmable rate generator is loaded with the first value of N
b) The motor is started
c) The table of differences is scanned, subtracting each element from the original value of N once every 1.29 ms. The new value is fed to the rate generator. This causes the motor to accelerate
d) At the end of the acceleration ramp, the number of steps performed by the motor is read
e) The program computes the beginning of the deceleration with the formula:

$$nd = ns - na - \text{offset}$$

where nd is the number of steps to perform before starting deceleration, ns is the total number of steps to perform, and na is the number of steps performed during the acceleration ramp. Offset is a small number (typically 10), used to prevent accidental overcoming of ns, due to non-synchronism between computation and step generation.

(f) The motor is now running at top speed. No action is undertaken until the number of steps nd is reached

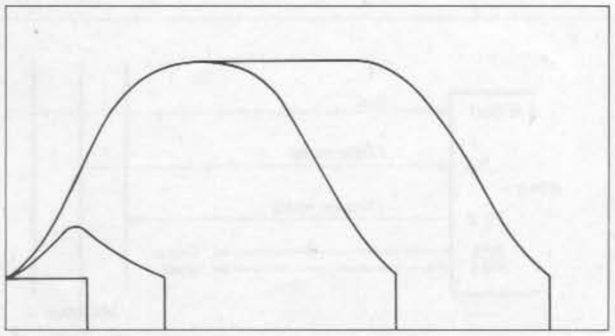

Fig 10 Speed-time curve for various length movements

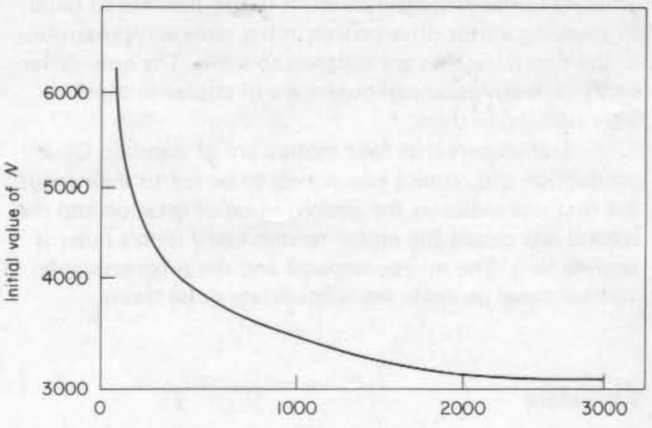

Fig 11 Correspondence between initial values of N *and the length of the resulting ramp*

Table 2 Length of program sections

Section	Bytes
Commands reception	50
Ramp generation	198
Table of differences	232
Additional features	522
Total	1002

(g) The motor is decelerated, reversing step (c), ie adding each value of the table of differences in reverse order
(h) The motor is now running at low speed. The program waits until the step counter reaches the value ns, then stops the motor

Short movements

In the case of short movements (less than about 2600 steps), the method just described may not be used, since under no circumstance may a ramp be changed before it comes to an end. It has been found that, simply by changing the initial value of N and using the algorithm just described, a family of accelerations (and of course of decelerations) may be generated. Fig 10 shows a subset of this curve family.

The problem is now the choice of the initial value of N as a function of the number of steps to perform. This may be done observing that the number of steps required for acceleration versus the initial value of N approximates a branch of a square hyperbola as shown in Fig 11. Short movements are then performed by computing the appropriate initial value of N, and following the algorithm described in the preceding paragraph. If the movement is very short (less than 200 steps) it is no use accelerating and decelerating the motor, and the whole movement is performed at a constant speed.

Length of the program

Although the algorithm just described is quite intricate, memory requirements are small enough to fit into the 8748 internal program memory. Table 2 gives the length of each program section.

Additional features

In addition to the most important function (ramp generation), the stepping motor program performs other fundamental though simpler operations.

Emergency braking

Two inputs are used to stop the motor when an emergency situation is detected. This applies in our robot to the end-of-rail and excessive force situations. The connection of the microcomputer to these lines was shown in Fig 8. If an interrupt is sent to the microcomputer while the motor is running, the movement must be stopped as soon as possible, that means following the steepest allowable curve. It must be remembered that a stepping motor may not be stopped simply by suspending the pulse train, since this would most probably cause a loss of synchronism that would increase the stopping time, rather than reduce it. The same algorithm which generates deceleration curves during ramped movements is efficiently employed to produce quick braking, reducing the length of the intervals between loads of *N* values to the rate generator. In fact the only difference between a normal and an emergency deceleration is the slope of the curve.

Origination

Another important operation is the origination of the motor. Since no positional feedback is provided in our robot, it is impossible to know the position of the arms when the machine is first turned on. The only way to know this is to bring each axis to a known position. This operation is automatically performed by motor microcomputers upon reception of a particular command which causes the motor to be moved backward until the limit switch is activated, and forward until it is released. At this point, the position of the arm is known, and it may be traced along successive movements unless losses of synchronism occur.

Status

As previously discussed, when an interrupt signal is received during a movement, the program generates an appropriate quick deceleration. At this point the CCU does not know the number of steps performed by the motor which means the position of the axis is unknown although no loss of synchronism occurred. By means of the status command, the CCU is able to interrogate the interrupted microcomputer about the number of steps performed until emergency stop point.

Other features

Some other functions may be performed by motor microcomputers: for instance, a movement may be performed at constant speed regardless of its length, or speed limits may be imposed. These special functions are indicated by appropriate op-codes in the commands.

Experimental results

Reliability of the system is good, since the number of components employed is very small. In fact the whole control system consists of only three integrated circuits. The algorithm has been tested using a motor Rapid-Syn model 34D-9209FA on a test bench with and without inertial and resistive loads. In these tests the speed of 27 000 half steps per second has been reached without loss of synchronism. This value is well above the 15 000 half steps per second which is the maximum required speed. This excellent performance did not deteriorate when the motor was mounted on the robot.

The only problem still to be solved is the optimization of short movements. This is because the method used for choosing the acceleration curve does not always yield the choice of the optimal one. The solution may be found in the introduction of a more sophisticated mathematical package.

Conclusions

The system described is an experimental device that has proved to be suitable for use in industrial robots, but that may as well be employed in a great variety of applications. The structure of electronics and the components employed are such that the engineering work for industrial production of the system would be relatively simple.

Since the communication protocol between the CCU and the microcomputer may be changed easily, the stepping motor drive may be used with many different control systems. The flexibility, reliability and low cost of this system are then undoubtedly a valid alternative to traditional random logic stepping motor drives.

References

1. **Cassinis R. and Mezzalira L.** A Multi-Microprocessor System for the Control of an Industrial Robot. *Proc. 7th I.S.I.R., Tokyo, 1977*
2. **Cassinis R.** Sensing System in SUPERSIGMA Robot. *Proc. 9th I.S.I.R., Washington, 1979*
3. **Cassinis R.** An Example of a Distributed Intelligence Discrete Process Controller. *Digital Systems for Industrial Automation. Due for publication*

Conference report

23–27 June 1980, Braunschweig, FRG

International Conference on Precision Electromagnetic Measurements

CPEM '80 was the latest in a series of conferences held every other year since 1958 and sponsored by the US National Bureau of Standards, the IEEE and the International Union of Radio Science (URSI). Most of the conferences have been held in the United States but they have become increasingly international in character: those attending come from many parts of the world, from national standards laboratories and from other organisations involved in precision measurements of electrical and magnetic quantities.

The first time the conference was held outside North America was in London in 1974. 1980 marked its