07-Apr-2002 03:05:23 2002-04-07

Introduction

Overview of Manual

Background

Getting started

If you have just recived your TeachMover, you probably will want to start operating it right away. You can - but there are a few simple things you must do first. *This chapter will give you the minimum information you need to quickly get started using and programing the TeachMover arm.* After you've completed this chapter, be sure to read chapter 3-5 (on mechanical & electrical construction of the arm) before proceding to chapter 6, where *all* the teach control operating instructions are given in detail. Propper utilization of all the arm commands requires an understanding of how the arm itself is constructed.

Mechanical check-out

Connecting the power supply

Plugging in the arm

Trial operation

Trial programing

How the TeachMover is built

Major components

The TeachMover's major structural components are shown in Figure 3-1 which is a duplicate of Figure 2-2. The microprocessor card is housed in the *base*. The teach control cable and the D.C. power cord extend from the rear of the base. The body swivels relative to the base on a hollow shaft attached to the base. The shaft is called the *base joint*.

Six stepper motors whith gear assemblies are monted on the *body* and control each of the six joints. The power wires for the motors pass from the computer card in the base through a hollow shaft to the body. This arrangement provides a direct cable-drive system.

The *upper arm* is attached to the top of the body and rotates relative to the body on a shaft called the *shoulder*

joint. Similarly, the forearm is attached to the upper arm by another shaft known as the *elbow joint*.

Finally, the *hand*, also called the *gripper*, is attached to the forearm by the two *wrist joints*. Two separate motors operate the wrist joints to controle the pitch and roll of the hand.

The TeachMover arm has a lifting capacity of one pound when fully extended, and a resolution (the smallest amount the arm can be made to move) of 0.011 inches. The end of the hand can be positioned anywhere within a partial sphere whith a radius of 17.5 inches, as shown in Figure 3-2. The maximum speed is from 2 to 7 inches per second, depending on the load (weight of the object being handled.) Detailed performance characteristics of the TeachMover are given in Table 3-1; this table is reproduced i Appendix F for easy reference.

In general, the base, the body, and all the extension members are hollow sheet-metal parts which are light in weight but strong. All members are connected to each other by means of shafts, or axles, passing through bushings mounted on the members.

Performance characteristics

Cabel drive system

Most robot arms have at least some of the drive motors mounted on the extensin members (forearm, upper arm, hand). Unfortunately this adds to the weight of those members, and means that the other motors - those that drive the extension members - need to be larger and more expensive than would otherwise be required.

If you look for motors on the TeachMover's extension members, you won't find any. That's because all six drive motors are mounted in the body. This minimizees the weight of the extension members and keeps the motor workload requirements as low as possible. To reduce the number of moving parts, all six drive gears are mounted on the same shaft.

A unique system is employed to manipulate the arm members. From the drive system in the body, aircraft-quality cables extend to the base, upper arm, forearm, and hand, as you can se by examining the TeachMover or by looking at Figure 3-3. This cable design is an adoptation and refinement of the "tendon technolgy" used in air-craft, high speed printers, and other types of equipment. Note that each cable is wound around the hub of the drive gear. This serves not only to provide a take-up drum for the cable, but also gives the proper gear-reduction ratios for each of the six drives.

Now, let's look briefly at how each of the cable drives is constructed. As you read about each mechanical part, it's a good idea to locate that part on your TeachMover. You need not be an expert in all aspects of the cable drive system, but some basic knowledge of how the cables work can prove extremely valuable later on.

As you look down on the hubs of the drive gears, you can see a set screw on each hub pinning the cable tightly in a grove cut into the hub. As the drive gears turns, the hub is pulling, winding, one-half of the cable while unwinding the other half. (See Figure 3-3)

Cabel drive details

Base drive

The *base drive* causes the body to rotate on the vertical base axle by driving a large pulley mounted on the base. Two smal pulleys, located at the bottom of the body, changes the base cable direction so that the cable feed is tangent to the surface of both the drive drum and the base pulley. You can see how these pulleys work if you rotate the body manually with power turned off. Note the termination of the cable in a clamp fastened by two screws on one side of the base.

Shoulder drive

Now hold the body in place and move the upper arm to see how the *shoulder cable* causes the upper arm to rotate on the horizontal shoulder axle. Note how this cable passes around a drive pulley on the shoulder shaft, then terminates on the upper arm housing. At the termination point, you'll notice two screws. These screws are used to maintain the cable under tension, as is explained in part C, below (Cable Adjustments).

Now rotate the shoulder joint again and notice that the shooulder rotation always causes equal and opposite elbow and wrist rotation so that the orientation of the hand remains unchanged. This feature is built into the TeachMover's cabling design to make sure that the hand can hold a glass of liquid while the shoulder rotates without spilling the liquid.

Elbow drive

The *elbow cable* causes the elbow to rotate on the horizontal elbow axle. Note that this cable first passes around an idler pulley on the shoulder axle, then around a drive pulley on the shoulder axle, then around a drive pulley of the same diameter attached to the elbow axle. The cable terminates at a tension mechanism on the forearm housing. Rotate the elbow manually and you'll notice that the wrist rotates the same amout in the opposite direction, thus maintaining hand orientation.

In rotating the elbow manually (that is with the power off), you may have noticed somthing else: when the elbow rotates, the hand opens and closes. Designing cabling to prevent this from happening mechanically would have added undesirable complexity. Instead, a built-in software routine automatically decouples elbow rotation from hand closure. To see this in action, turn the power on and press one of the E keys. Notice that now the elbow moves *without* opening or closing the hand.

Note: When the TeachMover is in serial interface mode, this automatic decoupling is inoperative, and, once again, rotating the elbow will effect hand closure. It is, however, easy to provide for the decoupling yourself when you write a host computer program. A simple formula to accomplish this is given later when the discussion of the commands available for your use in serial interface mode.

Wrist drive

The *right and left wrist cables* cause the hand to "roll" and "pitch" relative to the forearm. These cables together control the wrist joint (Figure 3-4). Both cables pass around idler pulleys on the shoulder axle and the elbow axle, then around the hubs of bevel gears located on the wrist axle. Tension is maintained in both cables by means of turnbuckles (to find them, look inside the forearm housing.)

Note how the two bevel gears on the wrist axle mesh with the output gear on the hand axle. This configuration forms a diffrential gear set.

To see how this diffrential works, turn the drive gears so the left and right wrist cables both move in the same

direction. You can see the wrist gears control the pitch of the hand. Turning the drive gears so the wrist cables move in opposite directions controls the roll of the hand.

The TeachMover's built in software automatically coordinates left and right wrist motions to produce the amount of pitch or roll you specify with the P and R keys on the teach control. When the TeachMover is in serial interface modde, you cannot specify pitch and roll directly, but only the amonut of motion of the left and right wrists. These motions can, however, be coordinated by a simple formula to produce the desied amount of pitch or roll.

Hand drive

The hand cable system is shown in Figure 3-5. Attached to the output gear of the differential gear set, the hand housing holds two pairs of links, and each pair of links terminates in the *gripper*. The housing, the links, and the gripper are attached to each other by small pins. Torsion springs located on the pins attach the links to the hand housing and provide the return force to open the hand as the hand cable is slackend.

Note: The length of the hand varies slightly with hand opening. For most applications, the amont of variation is negligible. For high-precision work, however, it may be necessary to take the variation into account. A formula for this is given at the end of Appendix D.

The *hand cable* is attached to the hand drive drum located in the body. The cable passes over an idler pulley located on the shoulder axle, and then over an idler pulley mounted on a sensing bracket found inside the upper arm housing. This sensing bracket is also called the *grip micro switch*. This switch goes on when the cable tension increases just beyond the point where the hand closes on an object (or on itself).

As we'll see later, the status of the grip switch can be used in programing for conditional branching. When the grip switch is activated, the green "CLOSED" light in the hand-held teach control goes on. Try it. If you close the gripper on an object and the green light does not go on, then check to see whether the hand cable has come off the shoulder idler pulley, the elbow idler pulley, or the sense bracket (grip switch) pulley. If it has, then simply replace it (refering, if necessary to the Hand Cabling Diagram on Page 3.13), and try again. If the green "CLOSED" light goes on *before* the gripper actually closes, then try tugging on the hand cable a few times, or adjusting the cable tension (see Section C, Cable Tension Adjustments, below.)

Attached to the other end of the tension spring, and in line with the hand cable, you'll notice two separate link-drive cables. These cables pass over two guide pulleys in the wrist yoke and then through the center of the hollow hand axle. When the cables emerge from the hand axle, they pass over separate idler pulleys mounted in the base of the hand. They then pass around idler pulleys mounted on the inner links of the hand, and return to terminate on the shafts of the two pulleys mounted on the hand base. This arrangement forms two block-and-tackle devices that augment the gripping force of the hand. The use of identical cabling on both links provides for symmetrical hand closure.

The tension spring mounted in series with the hand-cable drive assembly permits gripping force to be built up by the position-controlled drive motor once the hand has closed. Figure 3-6 shows: (1) The relationship between hand opening and drive motor steps (this is the line sloping *down* to the right), and (2) the relationship between gripping force and drive motor steps once the hand has closed (this is the line sloping *up* to the right). Note that the maximum gripping force is 3 lbs; this occurs approximately 100 motor steps beyond closure. Figure 3-6 is reproduced in Appendix F for ready reference.

Cabel tension adjustment

After a period of extended use or after an extreme overload, tension adjustments may slip. The relaxed half of a cable should not have noticeable slack. If a cable does develop slack, then the cable needs to be tightened.

Tension adjustments are provided at the folowing locations:

DRIVE	LOCATION
Base	Right side of the base (see Figure 3-7)
Shoulder	Right side of the upper arm
Elbow	Left side of the forearm
Wrist	Turnbuckle inside forearm

The adjustment procedure is as follows:

1. *Base, shoulder, elbow:*
    - Losen both tension screws.
    - Pull firmly (2-3 lbs.) on one tension screw to tension the cable as shown in Figure 3-7, and tighten the other screw with the other hand.
    - Release tension and tighten both screws.

    Be careful not to put excessive tension in the cable. Thight cables can cause the motor to slip with a loss of orientation between the microprocessor and the arm position.

2. *Wrist*:

    Each turnbuckle may be tightend or loosened as required to achive proper tension. As with the other adjustments, be careful not to tension the cables so much that the motors slip.

How the stepper motors operate

Fundamentals

Each of the cable drives is controlled by a stepper motor. The motors used have 4 coils, each driven by a power transistor. The drive is digital, with the transistors either turned on or turned off to obtain the desired pattern of currents, a rotating magnetic field is obtained inside the motor that causes the motor to rotate in smal increments or steps. More information on stepper motor control can be found in references (3) and (5).

Stepper motors are not the only kind of motors used in robot arms. Some arms use servo motors with electronic feedback loops for precise position confrol. Unlike stepper motors, these servo motors cannot develop slippage. This advantage must be weighed against the servo motor's far greater cost.

Keeping cost as low as possible is one reason we chose to use stepper motors for the TeachMover. Another reason is that stepper motors are easier to control from a computer than are servo motors.

Now, in order to turn a stepper motor in the TeachMover, a particular sequence of binary phase patterns is output to the desired motor, one pattern per step. In order to change motor direction, the order in which the phase patterns are output is simply reversed. The particular phase patterns used in the TeachMover generate a sequence known as "half-stepping;" the steps are half the size specified by the motor manufacturer. (The

motors used to drive the TeachMover are specified by the manufacturer at 48 steps per revolution, but are actually stepped at 96 steps per revolution.) Compared to full stepping, half-steppingproduces smoother slow-speed motions, reduces the power requirement, and improves the arm resolution by a factor of two.

The relationship between motor steps and actual joint rotation is given in Table 4-1. (The relationship between motor steps and hand opening was given in Figure 3-6.)

Table 4-1
Motor steps and joint rotations

Motor	Joint	Steps per degree	Steps per radian
1	Base	19.64	1125
2	Shoulder	19.64	1125
3	Elbow	11.55	672
4	Right wrist	4.27	241
5	Left wrist	4.27	241

Speed-Torque considerations

The torqe output (lifting capacity) of the stepper motors used on the TeachMover varies with their speed. At slow speeds, maximum torque is obtained. Above a critical high speed motors suddenly slip, and no torque is obtained. (Motor slippage can cause a discrepancy between where the arm is and where the computer program thinks it is, and this may result in unpredictable perrformance.)

The torque required by the motors of the TeachMover also depends on the configuration of the arm and the load held in the hand. This relation is a complex trigometric expression involving the lengths and weights of all the arm members. Instead of solving such an expression before each arm movment to determine the maximum allowable speed, it is simpler to program for the worst case.

The worst case is when the members of the arm are at maximum horizontal extension, requiring the maximum motor torque. All other configurations will require less motor torque. With the arm fully extended but with no load, the torque on all the motors is the same (by design) and motor speed can be as high as 400 half-steps per second, as indicated by the "no-load" point in Figure 4-1. Above this speed the motors will slip, and the torque will be zero. With the arm carrying the *maximum* rated load (that is, with the arm lifting 16 ounces) the torque on all the motors (except the base motor, which does not lift) is approximately equal, and the maximum speed without slippage is 99 half-steps per second; this is shown as the "full-load point" in Figure 4-1. At half rated load (8 ounces), maximum speed without slippage is 206 half-steps per second. These figures will become important later, when we discuss the commands you can use to control the speed of the TeachMover arm.

To perform a task as fast as posible without risking slippage, the following suggestions may prove helpfull:

- Lowering a load may be done at no-load speed-even if the arm is holding a load - provided the sholder, elbow and wrist do not raise.
- Swieling a load about the base joint may always be done at no-load speed.
- Opening the hand, or closing the hand until the contact point is reached, may always be done at no-load speed.

However special care must be exercised in selecting the proper speed when:

- raising a load with any joint.
- developing a gripping force once the gripper has closed on an object or on itself.

Motor control

Moving the arm from one position to another often requires rotation of more then one joint. In such cases, the motion can, in principle, be accomlished in either of two ways: the joints can be rotated sequentially or simultaneously. For example, as shown in Figure 4-2, motion of the arm from A to C can be accomplished through separate, sequential motions of the elbow and then the shoulder (A to B, then B to C), or through a coordinated motion in which the elbow and shoulder joints move simultaneously (A to C).

In general, coordinated motion is both smoother and faster than sequential motion. TeachMover firmware is programmed to produce coordinated motion whenever two or more motors are needed to move the arm from one recorded position to the next. To accomplish the coordination, the motor steps are timed so that each motor is pulsed at regular intervals during the full duration of the move. For example, if the shoulder motor is told to move 3 steps and the elbow motor is told to move 21 steps, the resultant timing will be as shown in Figure 4-3.

The TeachMover's motor-control algorithm has another feature: it produces controlled acceleration and deceleration to minimize jerkiness when the arm starts and stops. The velocity profile for motion of a stepper motor at a speed of 450 half-steps per second is shown in Figure 4-4. Note that for relatively short motions, such as the 100-step motion shown in Figure 4-4, the motor might not actually reach the specified speed before it needs to begin decelerating for a smoth stop. (We'll describe how to specify motor speeds in Chapers 6 and 7.)

Although the TeachMover's arm members move along curved paths, motion in a straight line may be approximated by a series of these curved motions. For example, one of the TeachMover's built-in demonstration programs moves the arm along an approximatly straight 10-inch line by means of 11 steps spaced one inch apart. (We'll explain how to run the demonstration programs later.) The segmented approximation has a theoretical error of only 0.018 inches in tracking the desired 10-inch line.

Internal electronics and interface

On-Board computer and memory

If you open up the base of the TeachMover (by gently laying the unit on its side and removing the four screws you'll find on the bottom), you'll see the circuit card that houses all the internal electronics, including the 6502A Microprocessor (Figure 5-1). In technical terms, this micrprocessor is an 8-bit, 2MHz chip. It's the same chip used in the Apple, Atari, & PET computers; it is used in the TeachMover to coordinate all joint motions and handle all input and output.

TeachMover firmware (permanently built-in software) is contained in another chip housing 4K bytes of read-only memory (ROM); this firmware interprets the commands you give the arm, converting these to electrical signals the arm can obey. Also contained in the 4K ROM are two built-in demonstration programs. More on these later.

The circuit card also includes chips containing 1K bytes of random-access memory (RAM); this is enough RAM to let you store an arm-motion program of up to 53 steps. It is possible to "piggy-back" a second set of RAMs on the first, thereby extending your program capacity to 126 steps. See Appendix C for instructions.

Serial ports

In the rear of the base, on either side of the flat cable that goes to the hand-held teach control, you'll see two multi-pin connectors. These are the *serial interface ports* that allow you to connect the TeachMover to a host computer, printer, or terminal. A switch on the coputer card allows you to select the serial transmission speed; eight standard speeds are available, from 110 to 9600 Baud. Further details of serial port operation are given in the chapter on serial interface mode.

User inputs and outputs

The computer card also contains an auxiliary parallel input/output port. This lets you interface the TeachMover to external equipment with a 16-conductor flat ribbon cable. Five TTL compatible user output bits can be set (to 1) or cleard (to 0) under program control to turn other equipment on or off when a given arm motion is complete. Seven TTL compatible user input bits can be used to control an arm sequence when a given external condition is met. The 16 pins and theier functions are given in Figure 5-2 and Table 5-1. We'll explain how to use the input and output bits i Chapter 6.

A block diagram of the TeachMover's electronic circuitry is shown in Figure 5-3.

Table 5-1
AUXILIARY I/O
CONNECTOR

Pin no.	Function
1	+5V-User Power
2	Ground
3	Not Used
4	Input bit 1
5	Output bit 5
6	Input bit 2
7	Output bit 4
8	Input bit 3
9	Output bit 3
10	Input bit 4
11	Output bit 2
12	Input bit 5
13	Output bit 1
14	Input bit 6
15	Ground

16	Input bit 7

Operating the hand-held teach control

Color coded controls

The TeachMover's hand-held teach control (*see Figure 6-1*) performs most of the same functions as do the teach controls on large-scale industrial robots. To provide a wide range of command options yet keep product cost to a minimum, we employed keyboard "overlays" that allow the same set of keys to provide three different kinds of functions. Rather than use an expensive alpha-numeric display to indicate which overlay is in use, we developed a simple system of color-coded lights and key lables.

For example, when you press the red MODE key, the red MODE light goes on, and the words printed in red (TRAIN, STEP, PAUSE, RUN, etc.) apply to the keys.

Pressing certain of these keys (PAUSE, OUT, POINT, JUMP or SPEED) will cause the yellow ENTER light to go on. When this light is on, the yellow numerals next to the keys apply, and you can enter numericals values (exactly how, we'll explain later.) *When the ENTER light is on, pressing the REC button will clear the entered value, allowing you to then enter the correct value*. Pressing the MODE button terminates enter mode.

Finaly, the labels printed on the keys themselves (B, S, E, P, R, G, and REC) apply when the teach control is in TRAIN mode (or in MOVE mode, as you'll see.)

The thirteen control fuctions

There are 13 diffrent control commands you can give with the teach control. (A concise summary of all teach control commands is given in Appendix F.*)

TRAIN

You can put the teach control into train mode in two diffrent ways:

- Simply turn the unit on. This puts the teach control into TRAIN mode automatically. (Note: turning the unit off erases the current program, and is therefore not recommended during arm training.)
- Press the red MODE key at any time to make the red MODE light go on, then press the key labeled, TRAIN.

Once the teach control is in the TRAIN mode, you can use the arm-motion (joint-control) keys and the REC key to manipulate the arm and record arm positions. (If necessery, refer back to Chapter 2, Section E, Trial Programming, to review how to do this.)

You can program up to 53 steps; these steps are internally numberd 0-52. (By adding additional RAM according to the instructions in Appendix C, you can extend program memmory to 126 steps; these step are internally numberd 0-125.)

Important Point: Pressing the REC key writes the current program step and *then* increments an internal sequence pointer so the TeachMover memory is ready to record the *next* step.

RUN

To run a program, first press the MODE key to exit from TRAIN, then press RUN. To stop a program while it's running, just press STOP. The MODE light will go on, and you can press RUN again, or TRAIN, or any other control key.

CLEAR

This command clears all recorded arm positions and operations from program memmory, and sets the sequence pointer to Step 0. You must use this command before you start entering a new program sequence.

To operate the CLEAR command:

- Press MODE *and hold the key down*.
- Then press CLEAR at the same time.

A word of caution: *Do not use the CLEAR command unless you mean it*! When you use CLEAR as above, your program is erased. A program can be uploaded to a host computer and saved on a disk. We'll explain how to do this in the next chapter.

ZERO

The TeachMover keeps track of the arm position with a set of six internal motor position registers. Each register contains the number of steps one of the six motors has taken since the registers were initialized. These registers are automaticaly initialized, or set to zero, when power is turned on.

In addition to setting all six internal postion registers to zero, the ZERO command also resets the sequence pointer to zero. The ZERO command lets you initialize the position registers at other times as well. As with the CLEAR command, you can activate the ZERO command only if you press the ZERO key *while holding down* the MODE key.

At the starting point for each program, the arm is first placed in a known starting position (such as the on described in Appendix G*,) then the ZERO command is enterd. As the arm moves, the computer keeps a count of the number steps each motor takes.

If you place the arm in the correct initial position, but forget to use the ZERO command prior to starting a recorded program, the arm first moves to reverse the count of all the internal position registers to zero. To avoid this problem it is a good idea to get into the habit of using the ZERO command just before runnig a recorded program.

PAUSE

This command is introduced in Chapter 2, Section E (Trial programing.) To review: If you want to program a pause into a program, perform these steps:

- Press the MODE key (if the MODE light is off.)

- Press the PAUSE key. The yellow ENTER light will come on.
- Enter a numer (0-255) corresponding to the number of seconds you wish the arm to pause. If you make an error in the numerical entry, you can "erase" it by pressing the REC key while the ENTER light is still on. Then enter the correct value.
- Enter the MODE key again. This terminates the ENTER mode.

Important: The PAUSE command is saved as a program step, and the sequence pointer is incremented.

SPEED

This command lets you change the speed of the arm by causing all the subsequent steps and manual teach control motions to be executed at the commanded speed. The SPEED does not get recorded as a program step. Try this:

- Press the MODE key (if the MODE light is of.)
- Press the SPEED key. Yellow ENTER light will come on.
- Enter a number from 0 to 15. Zero represent the slowest speed (it's not zero speed!) and 15 represent the fastest. (The TeachMover is always initialized to speed 5 when you turn it on.) If you make an error in entering the SPEED value, you can erase it by pressing the REC key while the ENTER light is still on. Then enter the correct value.
- Press the MODE key again.

The correspondence between the speed numbers and the number of steps per second of the drive motors are given in Table 7-3.

There is a maximum speed you can drive a motor before causing it to slip. For the worst-case configuration (arm fully extended, therefore requiring maximum torque), the highest without slipping depends on the load the arm is carrying, as indicated in Item F-5.

Under certain conditions, motors can be operated at higher speeds without slipping. In particular:

- If shoulder, elbow, and wrist all descend, the arm may be lowered at the no-load maximum speed (400 half-steps per second, teach control speed 8) even if it is carrying a load. *However, be careful not to exceed the speed given in the table whenever lifting a load with any joint when the arm is at or near full extension.*
- The base joint may be swiveled as fast as speed 7 even if the arm is carrying a load.
- The hand may always be opened as fast as speed number 12.
- The hand may always be closed as fast as speed number 10 until the hand closure contact point is reached. However, once the grip has closed, it is best to stay at or below a speed number of 6 in order to build up gripping force without motor slippage.

STEP

Once all or part of a program has been recorded, it is often useful to move the arm through the program one step at a time. To do this use the STEP command. First press the MODE key, if necessery, then press STEP. This moves the arm to the next programmed position. Press STEP again, and the arm moves to the next position again, and so forth.

The other thing you should know about the STEP command is how to use it for program editing. To change

an already-recorded arm position, simply STEP through the program until you reach the position you wish to change. Switch over to TRAIN mode, move the arm to the correct position, and then press the REC key. This overwrites the old position.

The POINT command provides an alternative method of accessing program steps for editing.

JUMP

This command lets you write rather sophisticated arm-motion programs by allowing for *conditional branching* - one of the most powerful features of classical computer programming. The JUMP command tests the user input bits on the Auxiliary I/O connector (P17) discussed in chapter 5, section C. Here is how it works: When you press the JUMP key, the yellow ENTER light comes on, and you enter not one, but *two* numbers. The MODE key is pressed after each number to allow the computer to store each number separately. A number may have more than one digit requiring you to press more than one key to enter the number.

- The first number identifies the *jump condition*.
- The second number identifies the program step to jump to if the jump condition has been met.

The jump conditions are listed in Item F-6.

Note: When you step through a JUMP command, the usual incrementing of the sequence pointer is slightly modified. For example, let's say you STEP to the command. JUMP 9,7. This unconditional jump sets the value of the sequence pointer to 7. If you press again, step number 7 gets executed, but the sequence pointer is *not* incremented. If the pointer were incremented *first*, as it usually is with the STEP command, then step number 7 would be skipped, and step number 8 would be executed instead. On subsequent pressings of the STEP key, the sequence pointer is incremented as usual.

POINT

In a way, the POINT command is similar to an unconditional JUMP. For example, POINT 12 means go to Step 12 in a program and proceed from there. However, unlike the JUMP command, *POINT does not create a program step*. POINT is used simply to move to a given step in an existing program. It can be invoked even in the middle of program execution by pressing the MODE (STOP) key first.

One of the most useful applications of the POINT command is program editing. Instead of using the STEP command to go to a program step you want to change, just POINT to the corresponding program step number. Then press MODE or TRAIN (or appropriate command,) and enter the new program step. Keep in mind that the STEP command points to the step and execute it. Thus, when you use STEP for editing, the program step you change is the one the TeachMover just executed. However, the POINT command points to the step without executing it.

GRIP

This command will cause the gripper to close 32 half-steps past the point at which the grip switch is activated as the gripper first closes. This builds up about 1 lb. of gripping force.

This command is diffrent from using the G keys in train mode. The G keys will simply command the fingers to open or close to a particular spacing, regardless of whether the hand is holding an object. The GRIP

command, on the other hand (sorry for the pun!), close on an object and then builds up 1 lb. of gripping force regardless of the size of the object. Thus, the GRIP command is usefull when you want the arm to pick up a variety of objects, whereas the G keys are a better choice when you want the arm to sense whether a particular object is present.

Note: Once in a while, the grip switch may fail to operate, or the GRIP command **opens** the gripper instead of closing it. See page 3.14, Hand Drive, for simple remedies.

MOVE

This activates the joint control keys used in TRAIN mode, but does not change the internal position registers or allow a position to be recorded. In other words, if you press MODE, then MOVE, then one of the joint control keys, then REC, you'll find that the REC key has no effect.

The MOVE command proves useful in moving the arm to a known initial position in the event of motor slippage or mekanical interference from an external obstacle. The procedure is:

- Use the ZERO command. (Remember, this sets the internal position registers and the sequence pointer to zero.)
- Use the MOVE command.
- Use the arm-motion keys to achive the correct initial configuration. (For very precise position control, first use the SPEED key to specify a speed of 0.)

Note that this does not change the recorded program in any way; it simply lets you start the program again with the arm in its correct initial position.

In some cases, you may want to reinitialize the arm by returning it to a know position *other* than the initial position. The ZERO command is not useful here. Instead, follow these steps:

- STEP (or POINT then STEP) until the arm reaches the position you wish to use for initializion.
- Use the MOVE command.
- Use the arm-motion keys to achive the correct position.

When you subsequently RUN the program, all settings of the internal position registers are associated with the recallibrated new initialization position.

A similar use for the MOVE command is if an object isslipping from the gripper because the gripper isn't holding it tight enough. Instead of reprogramming, you can just STOP the program, use the MOVE command, use the G key to close the gripper a bit, then RUN. All subsequent settings of the internal position registers will now be associated with the new grip setting.

FREE

FREE is similar to the MOVE command, execpt it turns off all motor currents to allow you to position the arm manualy. In many cases, it is simply a matter preference whether you use MOVE or FREE. However, as mentioned earlier, you will probably find that you get finer position control when you use the MOVE command with a speed setting of 0, rather than moving the arm manually in FREE mode. Try both and see.

OUT

This is the command that lets you turn external equipment on and off based on arm position achived or conditions met. You can also use it to turn on and off various lights on the hand-held teach control.

When you press OUT, the yellow ENTER light will come on, and you must give two numerical entries (pressing the MODE key in between). The first entry is an output number (see below,) and the second is 0 or 1. (In the case of the teach control lights, 0 indicates "off" and 1 indicates "on".) The output numbers are listed in Item F-6.

Demonstration programs

Exerciser

Block stacking

Program editing

Experimentation

Operation from a host computer

Connecting the TeachMover arm to a host computer or a terminal greatly extends the unit's capabilities. As we will see, use of the TeachMover's serial interfaces allows you to write programs that specify arm positions by means of Cartesian coordinates, programs that actually measure the position and thicknes of an object, and much more - all without losing the ability to program the arm from the hand-held teach control.

Configuring the serial ports

"Configuring the serial ports" refers to making sure that your computer and the TeachMover can "talk" to one another. This requires taking care of the following:

1. electrical connections
2. transmission rate
3. data format
4. settings for standard interface signals
5. opening the port
6. testing the configuration

Depending on the computer you're using, configuring the serial ports may be straightforward and simple, or it may be complex. We'll start with the most basic steps first, then procced to the more intricate details.

If you follow the procedure given below, and yet the arm won't respond to serial port commands, chances are that some of the details of configuring you computer are not correct. If this seem to be the case, carefully review the user manual that came with your computer or call a costomer service representative employed by the computer manufacturer.

Electrical connections

In the back of the TeachMover's base you'll find two mulit-pin connectors (Figure 7-1*.) These are the two serial ports.

- Signals that enter the left port (P2) always pass through to the right port (P1) unchanged.
- Signals that enter the right port pass through to the left port unchanged, unless the signals are a series of characters begining with an "@" sign and terminating with <CR> (carriage return); these signals are not passed through, but are iterpreted as arm commands. (As we'll se later, one of the serial interface commands lets you specify a diffrent recognition character in place of the "@" sign.)

Thus, to operate the arm from a host computer or a terminal, connect the computer or terminal to the TeachMover's *right* serial port (Figure 7-2 a. and b.*.)

Transmission rates

The TeachMover is shipped with both serial ports configured to operate at a transmission rate of 9600 baud (9600 bits per second), for both send and recive. You can change this rate to any of seven other standard rates by means of three switches located on the TeachMover computer card (Figure 7-6*.) The available rates and the corresponding switch settings are given in Table 7-1. (This table also appears in Appendix F for easy reference.) These switches should be changed when power is off, since the switch settings are read TeachMover firmware on power-up only.

As with the TeachMover, most computers have some means of setting baud rate - either through switches on a circuit card, or via commands that can be issued under the computer's disk operating system (DOS,) or as part of VASIC. The main thing is that both the TeachMover and your computer be configured to operate at the same baud rate; otherwise communication between the two will be impossible.

Data format

The TeachMover uses the following data format:

- word length = 8 bits
- 1 start bit
- 1 stop bit
- no parity bit
- full duplex

Many computers have the above as their "default" format - that is, the format in which data will be transmitted if you do nothing special to configure the format. However, with other computers you will need to configure the format explicitly. Consult your computer manual to learn how to do this.

Standard interface signals

Some computers and terminals require logic levels on certain pins to indicate the following status conditions:

1. Data Terminal Ready
2. Clear to Send

3. Carrier Detect
4. Request to Send

The TeachMover does not use these signals, but does pass them through when it is placed in series between a computer and a terminial.

However, when only a single computer or terminal is connected to the TeachMover (or in some cases even if the TeachMover is placed in Series between a computer and a terminal,) you may need to modify the TeachMover in order to provide these signals.

To find out if this is necessary, consult the user manual for your computer to determine wheter any of the four above-mentioned serial port signals are required. (With som computers, the user has control over whether these signals are required. If this is the case with your computer, then configure the computer so that these signals are *not* required.) If any of these signals *are* required, then, if you have a peripheral in series with the TeachMover, check the user manual that came with that peripheral to see whether the peripheral supplies the required signals for transmitting and receiving data. If it does, then all should be well. If not, then you will need to modify the TeachMover.

The modification procedure is simply to solder a jumper across the appropriate terminals on the TeachMover circuit card as shown in Figure 7-7, using the information given in Table 7-2; the jumpers are labelled W1, W2, W3, W4. (Note: soldering a jumper will have the effect of permanently setting the corresponding signal to logic level 1, or "on.")

Table 7-2
Auxiliary Control Lines

Left Port Pin No.	Description	Right Port Pin No.	Jumper
8	Data Carrier Detect	8	W1
1,7	Ground	1,7	-
3	Transmit from TeachMover	2	-
2	Receive by TeachMover	3	-
4	Request to Send	4	W4
5	Clear to Send	5	W3
20	Data Terminal Ready	20	W2

Opening the port

This step refers to configuring your computer so that commands and data are properly routed from your computer to the TeachMover. If your computer has only one serial port, there may nothing special to do other then use the proper BASIC commands for input and output throgh that port. On some computers, you might also have to use a special routing command or switch setting to transmit to the TeachMover whatever would normally go to a parallel printer. Some computers have several serial ports, and allow you a choice of transmission chanells.

In all cases, consult your computer manual to find out what is required.

Testing the configuration

Once steps 1-5 above are completed, you are ready to test the serial configuration. This is best accomplished by issuing an "@CLOSE" command. This command closes the gripper until the grip switch is activated. We'll go into the details of this and all the other commands in the next section, but for now it is important to know how the TeachMover responds to arm commands in general.

Serial interface commands

Ten diffrent commands can be issued to the TeachMover over the serial lines. (A concise summary of all ten commands are given in Appendix F*.)

Note:

- It is a good idea to familariarize yourself whith all the teach control commands (Chapter 6) before reading about the serial interface commands in this chapter.
- All commands can be abbrevated to an "@" sign plus the first three characters--@CLO for @CLOSE, etc.
- All characters and numeric values are decimal ASCII (industry-standard character format).
- Once a serial command is executed, the the teach control is left in TRAIN mode, with two exceptions:
  - @RESET leaves it in MODE mode.
  - @RUN simply runs until another command stops it.
- However, the indicator lights will remain as they were *before* the serial command was executed. (Example: If MODE light is on, and then, say, an @CLOSE command is executed, the Teach Control will then be in TRAIN mode but with the MODE light still on.)

- If you wish to change the status of the indicator lights, you can use the @STEP command with all the parameters set to zero except the "OUT" value (see below.) No other serial command affects the status of the lights (except the closed light which always indicates the state of the gripper switch.)

  The ten commands are as follows. Details of syntax and usage begin on the next page. Sample programs using these commands are given in part C of this chapter.

  - @STEP
  - @CLOSE
  - @SET
  - @RESET
  - @READ
  - @ARM
  - @DELAY
  - @QDUMP
  - @QWRITE
  - @RUN

@STEP

The @STEP command is analogous to using the teach control SPEED command, the TRAIN command and the OUT command all at once.

The @STEP command causes all six of the stepper motors to move simultaneously. The syntax of this command is:
`@STEP <SP>,<J1>,<J2>,<J3>,<J4>,<J5>,<J6>,<OUT><CR>`
where:

<SP> gives the speed of motion,
<J1> to <J6> are the number of half-steps that each of the six motors are to be moved,
<OUT> specifies the bit pattern to go to the user outputs, and
<CR> signifies carriage return.

As explained above, the arm respondes with `[0<CR>]` if you have made a syntax error, and `[1<CR>]` if the arm executes the command. However, if the STOP key gets pressed before the specified motion is completed, the arm returns `[2<CR>]` instead. (This is discussed further at the end of the "@CLOSE" command, below.)

Now here are the details.

Speed Value, SP

The speed value, SP, is related to the motor speed in half-steps per second by the folowing formula:
Motor Speed = 1843200/(|SP - 255| * 256)

Tabel 7-3 gives the correspondence between motor speed, teach control speed numbers, and serial interface speed values (SP.) Note that although only 16 speeds are possible using the hand-held teach control, you can specify 246 diffrent speeds (SP = 0, 1, 2, ..., 245) in serial interface mode. Table 7-3 is reproduced in Appendix F for ready reference.

It's a good idea before using the @STEP command to review the maximum speeds that you can drive the motors without causing them to slip. You will find these speeds in a table in Appendix F, Table 5; the maximum speeds are given in three diffrent ways: half-steps per second, teach control speed numbers, and serial interface speeds values (SP.)

TABLE 7-3
Stepping Rates

Teach Control Speed Number	Serial Port Speed Value	Half-Steps Per Second
0	0	28
1	111	50
2	159	74
3	183	99
4	205	141
5	221	206
6	232	300
7	236	360
8	238	400
9	239	424

10	240	450
11	241	480
12	242	514
13	243	554
14	244	600
15	245	655

Motor Steps, J1-J6

The magnitude of each of the quantities J1-J6 indicates the number of half-steps the motor should driven. The sign of each number indicates the direction; positive directions are indicated in parantheses as follows:

- J1 - Base Swivel (counter-clockwise)
- J2 - Shoulder Bend (downwards)
- J3 - Elbow Bend (downwards)
- J4 - Right Wrist (downwards)
- J5 - Left Wrist (downwards)
- J6 - Hand (open)

Important Point: Unlike operation with the hand-held teach control, using the @STEP command does not uncouple the elbow, <J3>, from the hand, <J6>. Moreover, you cannot specify "pitch" and "roll" directly, but only the number of half-steps for the base, shoulder, elbow, pitch, roll, and grip are given by B, S, E, P, R, and G, respectively, then the motion command you would use is simply:

```
@STEP <SP>,B,S,E,(P-R),(P+R),(E+G),<OUT><CR>
```

If you specity an unequal number of half-steps for each joint, then TeachMover firmware will automatically coordinate the timing so as to produce smooth simultaneous motion of the motors. For example, if the elbow motor is told to move 16 steps and the shoulder motor 3 steps, the resultant timing is as ilustrated in Figure 7-8.