1   INTRODUCTION TO THE CY525   1

HISTORY OF STEPPER CONTROL
Boxes to Boards to Chips

The first stepper motor controllers consisted of bulky boxes controlled by switches and buttons. Step rates were set in hardware, as were the acceleration and deceleration characteristics. Switches were used to set the number of steps and direction of stepping. Buttons were used to actually start the motion. These controllers were obviously meant for manual operation. They were very expensive, very heavy, and very large when compared to the motors to be controlled.

In the next stage of controller design, the functions of the controller boxes were designed onto single PC boards. These significantly reduced the cost and packaging requirements, but did not increase the capability of the controller. One important benefit of this design was the ability to simulate switch inputs electronically, allowing another machine to command the controller.
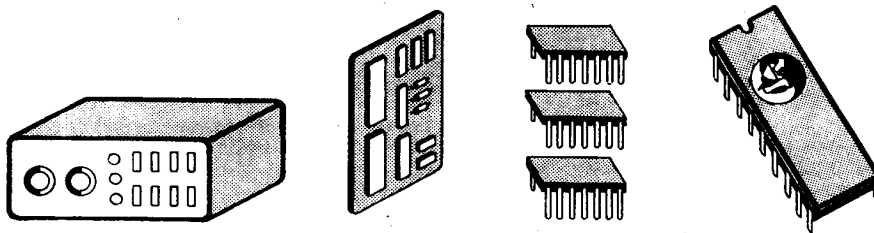

Figure 1.1 Controller evolution, boxes to boards to chips.

About this time the low cost computer came into fashion. It became a natural source of input signals to run the stepper motor controller, providing a pulse train to the translator module. If the loading of the motor was such that acceleration and deceleration was required, then the computer provided the timing of the pulses to affect the acceleration. If, further, the position control required complex motions that were relative to either the current location or to some absolute coordinate system, then the computer also provided these calculations. If the sequence of motions was to be synchronized with other external events, the computer provided such synchronization.

At this point, the control of motion became a non-trivial problem, and the programming of a computer to provide this control represented a major design effort. If not one, but many, motors were to be controlled, the problem became even worse, and quickly exceeded the capabilities of the low cost computer.

In 1979 Cybernetic Micro Systems introduced the first intelligent single chip Stepper Motor Controller — the CY500

In the early days of microprocessors, it was generally true that motor experts were not computer experts, and vice versa. Thus the primary rationale behind the CY500 was to provide an interface chip and to say to both sets of experts:

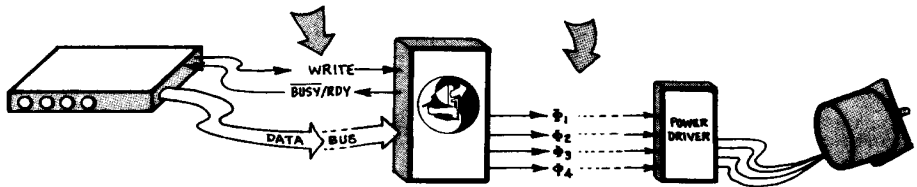"Hook your computer here, and your stepper drivers here."

Figure 1.2 The CY500 was the first single chip controller that provided all the functions necessary for high level control of stepper motors (introduced 1979).

In addition to a simple interface, the CY500 also offered high level functions designed for precision positioning, specifically, the ability to move a specified number of steps (relative move) and also the ability to move to a specified location (absolute move). Thus the user could tell the CY500 to:

$$\text{"take n steps"} \qquad (\ 0 < n < 64K \ )$$

or

$$\text{"go to position p"} \qquad (\ 0 < p < 64K \ )$$

In order to allow easy use with high level components (keyboards & displays) and high level languages like BASIC, the CY500 accepted ASCII commands and ASCII-decimal values. The commands consist of a single alphabetic character followed by a decimal number if applicable. For example, the command to step to position 36003 is sent to the CY500 as

P 36003⟩

where ⟩ is the ASCII carriage return character, ⟩ = 0Dh.

For convenient interface to machine language or binary arithmetic systems, the CY500 also accepts binary values for parameters if the ASCII/Bin pin is held low.

6

Program Storage on Chip

As powerful as the high level commands are, it is always possible to achieve even more powerful control via a **sequence** of commands with conditional branching. In order to accomodate such a sequence, the CY5xx Stepper Control chips provide "on chip" program storage, capable of storing from ten to fifty instructions, depending upon the chip and the instructions. This allows the user to write programs that perform tests and react appropriately.

Conditional Tests

Because of the nature of most stepper applications, emphasis has been placed on testing **external events** (as opposed to testing the results of internal calculations, as is done in general purpose computers.) Thus the CY5xx devices can:

      1. Loop til an external event occurs
      2. Wait for an external event to occur.
      3. Loop for a specified count of events

In addition to sensing external signals, the CY5xx controllers also generate both specific and general purpose control signals to be used by external subsystems. The ability to sense, signal, and synchronize, provides a very powerful subsystem "building block" for systems involving motion.

Intelligent Motion Control

Although stepper motors may operate with considerable angular velocity or step rate, most of them must begin (and end) stepping at a low step rate. Thus a high level controller should provide acceleration and deceleration, and this should be generally transparent to the user.

By "transparent to the user" we mean that the controller should determine when to stop accelerating and also when to begin decelerating to stop at the desired position. The CY500 allows a "slope" parameter to specify the acceleration.
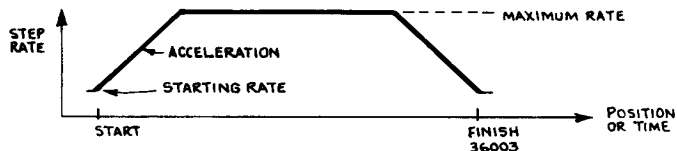

Figure 1.3 Illustrating a typical move for a stepper motor when a position command is given to the CY500.

7

Finally almost all stepper motors are used as system elements and it is often desirable to synchronize instructions with input and output lines for easy TTL interfacing.

As the first intelligent stepper motor controller, the CY500 was well received by customers and won a "Product of the Year" award from Electronic Products Magazine in 1980. However, as is the case with most first products, the CY500 was not perfect. Although it had been designed with precision control of positioning in mind, many users wanted faster stepping. The CY500 would step at 3360 steps/sec.

Thus in 1981, Cybernetics introduced the first second-generation stepper motor controller, the CY512. The CY512 was 90% pin- and instruction-compatible with the CY500 but offered new features:

- First, the CY512 was faster than the CY500. With the use of feedback, the CY512 would step almost 8000 steps/sec. Without feedback, the CY512 would step approximately 5000 steps/sec.

- Second, the CY512 could be queried. For example, the user could ask the CY512 for its position and receive the answer as either a 16-bit binary number or a 5 digit decimal position. The user could also examine the CY512 parameters such as slope, rate, etc.

- Third, the CY512 could determine the direction (CW/CCW) it must move to go from its current (arbitrary) position to a specified target position, unlike the CY500 which must be "pointed" in the right direction before the position command was executed. Also unlike the CY500, the CY512 exhibits its internal direction on an external pin.

- Fourth, a "Loop count" instruction was implemented in the CY512 and its I/O structure generalized somewhat.

These changes were welcomed and most users select the higher cost CY512 over the CY500 because it offers better value.

However...

As proud as we were of the CY512 and as happy as most users were, we still had numerous customers who wanted:

1. Still faster rates, up to 10K steps/sec.

2. More linear ramp (related to faster)

3. Unlimited stepping

4. More complex rate curves.

5. Miscellaneous other requests.

Each of these will be discussed briefly:

1. Faster....with the emphasis on productivity, many people need not only intelligent, easy-to-control motion, but FAST motion. The realistic upper limit of the CY512 was 5K steps/sec, while many motors will go 10K steps/sec (many will not!).

2. More linear RAMP....This feature really relates to usable speed. The CY512 would step at up to 8000 steps/sec, but most motors would not accelerate fast enough to keep up with the CY512. The reason for this is the non-linear acceleration curve of the CY512.
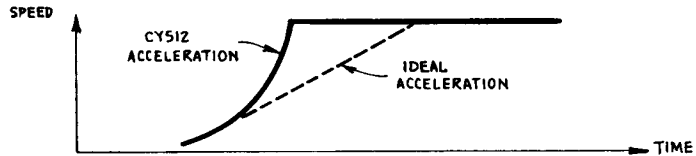
3. Unlimited stepping......64K steps of travel is not enough for some people, so this limit in the CY512 forced such applications to take multiple moves with automatic acceleration for each move.
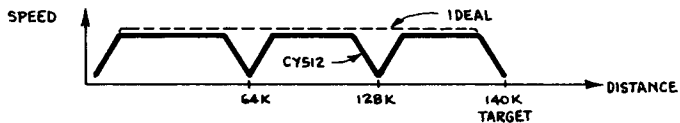
4. As applications become more sophisticated, many people need velocity curves of the type shown below:
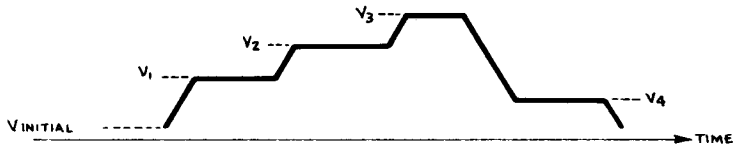

5. Finally, there were numerous miscellaneous requests such as the ability to turn off all phases without altering any internal parameters, etc.

The response to this market feedback is described next:

THE CY525 INTELLIGENT RAMPING STEPPER MOTOR CONTROLLER
The Third Generation Stepper Motor Controller

First - it is FASTER - a true 10000 steps/sec (@11.37MHz)

Second - it is LINEAR (that's how we got it to go faster!)
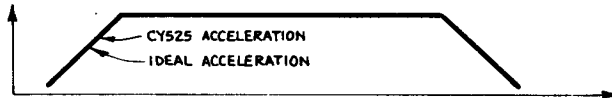
Third - You can change rates-on-the-fly!

Fourth - You can take an unlimited number of steps - run FOREVER!

Fifth - You can read out (binary) position-on-the-fly, that is, you can ask the CY525 where it is, while it's stepping!

Sixth,   Seventh, Eighth, etc.-Of course we allow you to turn the phases off (with or without a pulse) and have cleaned up other features (to allow use with slow BASIC interpreters whose tardy write strobes once terminated CY512 stepping). And we now let you use labels in a program so you won't have to count bytes for jumping, looping, etc.

Last - from R = 3 to R = 40, the rates are specified in units of 100 steps/sec. Thus to go 300 steps/sec (@11MHz) the user specifies R = 3; to go 2900 steps/sec, the user specifies R = 29 etc. Above R=40 (4000 steps/sec) the rates change by approximately 60 steps/sec per rate increment.

As the final feature, the CY525 can be plugged into your present CY512 socket and it will work. Of course to take full advantage of the "on-the-fly" features you will need two latches, etc., but if you just want to go faster and smoother, just pull out the CY512 and slip in the CY525! (note: see ABORT changes.. page 63).

That's the brief history of single chip intelligent motor control from the CY500 in 1979 to the CY512 in 1981 and the CY525 in 1984. Of course we have been busy offering a variety of support chips and circuits such as the CY512/Kit prototyping board, the CYB-002 prototyping kit capable of controlling two CY525s, and the CYB232 board that allows up to 255 stepper motors to be controlled over one serial RS232 line.

AND we have software for the IBM-PC and XT!

Two Primary Modes: Independent vs Closely Coupled

The CY525 can be operated in two primary modes (although the modes can be mixed.) The first is the "independent" mode and is identical to the CY500 and CY512 operation. In this mode the parameters are specified and then a motion command is issued or a program of instructions is executed. The interaction of the controller with the master computer is typically limited to signaling either:

>   1. the motion is complete, or

>   2. the program has terminated.

This high level control relieves the master computer of worrying about the details of the motion or motion sequence.

The CY525 offers a second, "closely coupled" mode of operation that provides extreme flexibility at the expense of some attention on the part of the master computer. This mode provides the ability to change the step rate "on the fly" and to read the current position "on the fly", i.e., while the motor is stepping. An example of the motion that is possible in this mode is shown in the figure below.
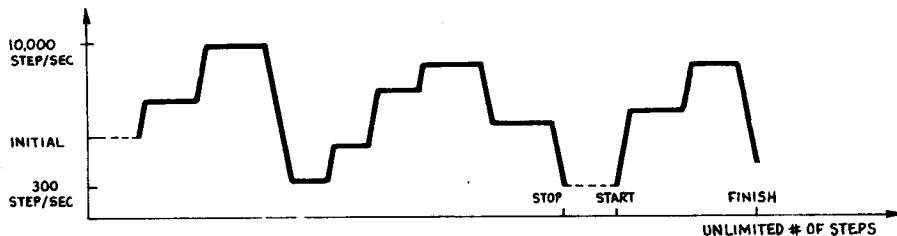

Figure 1.4   Example of complex motion possible in "closely coupled" mode of operation.

NOTE: Although it is possible to mix the two primary modes of operation, this should be done with caution. The two modes are designed to be consistent and coherent as separate modes and there are some inconsistencies that result when mixing the modes. These do not prevent useful mixtures, but require understanding of the differences for successful execution. For example, attention should be given to both the SLEW line and the ABORT line when mixing the modes.
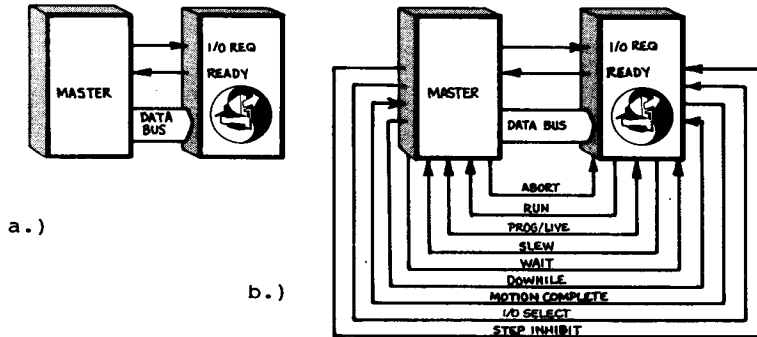
Figure 1.5 The CY525 can operate in two primary modes:

     a.)   "Independent" mode in which commands and programs execute in the CY525 independently of the master computer.

     b.)   "Closely Coupled" mode in which the master computer closely interacts with the CY525 while it is executing commands and programs.

     c.)   These modes can be mixed, with care.

ASCII Mode Allows Simple Prototyping

The CY525 is an ASCII-programmable, peripheral controller chip designed to control stepper motors using an instruction sequence that may be stored internally in a program buffer. This feature allows the user to program the device with an ASCII keyboard and vastly simplifies prototype development and experimentation. When the user decides that the control sequence is correct, the ASCII keyboard is replaced by a computer output port, and the motor can be brought on-line. Of course, the computer can be used initially in those systems in which keyboard programming is impractical, but most applications can usually benefit from the immediacy of the keyboard during the development phase. In this mode the user simply types a command on the keyboard and the controller takes the appropriate action. In the Command Mode, the controller simply executes the command. In the Programming Mode, the command is stored in sequence in the on-chip program buffer for later execution.

Stored Program Peripheral Controller

The Cybernetic Micro Systems CY525 Intelligent Ramping Stepper Motor Controller offers the user stored program capability. This feature significantly increases the power of the device and, as a consequence, decreases the amount of host time and software required to perform a given task. Stored program devices operate in three basic modes:

Command execution mode

The CY525 executes commands as they are received.

Program entry mode

The CY525 stores commands in an internal program buffer for later execution.

Program execution mode

The CY525 executes the commands that were previously stored in the program buffer.

Thus, in addition to the Command Execution mode common to all peripheral controllers, the stored program controller can be placed in a Program Entry mode in which the sequence of commands is entered and stored in the program buffer, and then the device can be placed in the Program Execution mode in which stored sequences of commands are executed.

In many applications the user will find that the CY525 can function as a stand-alone device, completely independent of the host processor, except for program loading. In most of these applications, it may be possible to generate custom devices that load the desired program upon power-up and are triggered by external hardware. The user can then employ these custom controllers in stand-alone applications with no host. See PROM Stand-alone example in Section 10.

13

ARCHITECTURE OF THE CY525
STEPPER CONTROLLER

The CY525 architecture
may be partitioned
into several func-
tional subsystems:

1. Input data
   subsystem

2. Output data
   subsystem

3. Program para-
   meter storage

4. Mode flags and
   pins

5. Program storage
   buffer, 60 bytes

6. Instruction
   selection,
   decoding,
   and control
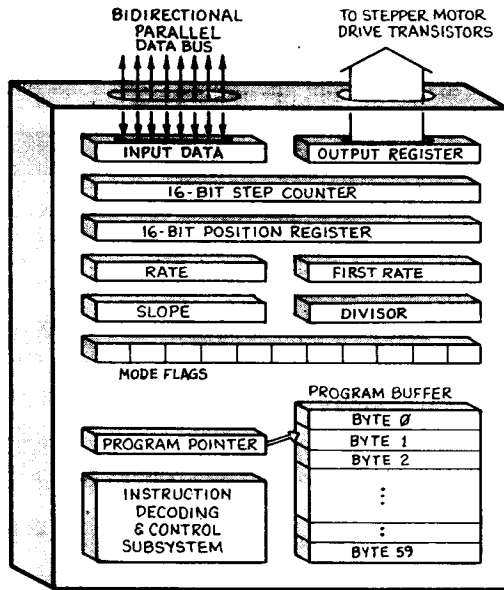   mechanisms.

7. Position
   Register

Figure 1.6 Schematic diagram of the architecture of the CY525 Intelligent Ramping Stepper Motor Controller.

Input and Output Data Subsystems

The input data subsystem accepts commands and the output data subsystem holds the output control signals to the stepper drive circuitry and includes the associated direction and pulse timing lines.

Program Parameter Storage

The program parameter storage subsystem is used to store the step rate parameters, ramp rate parameter, and to maintain a 16-bit position register. The position register is incremented (or decremented) when stepping in the clockwise (or CCW) direction. The position register is used when **absolute** position commands are specified. The 16-bit step counter is used when **relative** commands are employed. The contents of the position register change with every step, while the step counter register contents remain unchanged until a specific command is used to change them.

14

Mode Flags and Pins

The mode flags and mode select pins are used during command execution to perform the appropriate action or to interpret data or input signals correctly.

Program Storage Buffer

The CY525 contains a program buffer that allows the user to store a sequence of instructions that can be executed upon command. This provides all of the benefits of stored program execution that have made computers such powerful tools.

Instruction Decoding and Control

This subsystem performs the actual execution of commands.

Position Register

The CY525 contains a 16-bit position register that can be read to determine the current location. The CY525 will accept relative and absolute position commands; however, the position register always indicates absolute position. In the CY525, this register can be read while the CY525 is stepping.

Absolute vs Relative Position

The **G** command operates in the relative position mode, in which total travel is specified relative to the current position via the Number (of steps) command, **N n}**, where $0 < n < 65535$. In this mode an internal counter is decremented for each step and stepping continues until the count reaches zero (or another Halt condition is detected). If the Position mode command, **P p}**, is received, the target position "p" is interpreted as absolute position with respect to the zero location declared by the Athome command. The CY525 calculates the stepping direction and the number of steps to take to reach the specified target position.

1. The relative mode is selected by the G command.

2. The absolute mode is selected by the P command.

When an actual stepping operation is in progress, both a number of steps value and a target position are used to internally execute the step command. The current position register is updated in both the relative and absolute modes, so motions may be mixed between the modes. In relative mode, the target position will be calculated, and in absolute mode, the number of steps to take will be calculated. After that, the two modes use the same internal stepping routines. Acceleration and deceleration work in both stepping modes, with the CY525 also calculating the position at which to start deceleration in order to return to the starting step rate when the motion is completed.

2   OVERVIEW OF PIN FUNCTIONS   2

COMMUNICATION WITH CY525

Commands can be issued to the CY525 using a parallel data format. In parallel operation, complete handshaking operation occurs via the use of a Busy/Ready line on the CY525, and the I/O Request strobe line from the host. The handshake protocol is shown below.
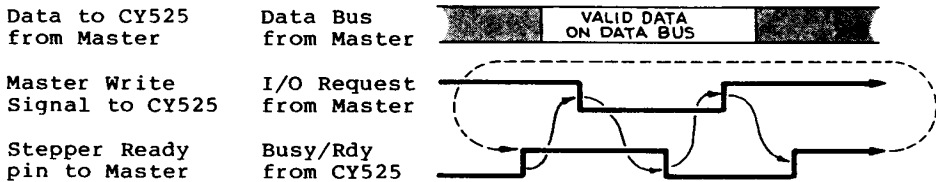
Data to CY525 from Master	Data Bus from Master	VALID DATA ON DATA BUS
Master Write Signal to CY525	I/O Request from Master	
Stepper Ready pin to Master	Busy/Rdy from CY525	

Figure 2.1   Handshaking protocol for CY525 parallel input.

KEYBOARD PROGRAMMABLE DEVICE

The CY525 Stepper Motor Controller offers Hi-Level Language programming with an ASCII keyboard. This design allows the user maximum utility via the closest possible coupling and facilitates interactive prototype development and debugging. Note that the stored program capability makes it possible in many cases to perfect the operation of the stepper motor completely decoupled from the host computer. In such cases, the host processor is required to do little more than load the programs at appropriate times. Of particular importance in many applications is the dynamic stability of the system. By programming a range of test conditions through the keyboard, the designer may exercise the system over broad ranges and thus characterize the system dynamically. Of course, any designer with access to an easy-to-use, interactive host computer can achieve everything that the keyboard user can, and more. Lacking such systems, the designer will appreciate the extreme power of keyboard programming during prototyping phases, thus postponing until final systems integration the slower, costlier, host computer programming associated with all host-controlled controller devices.
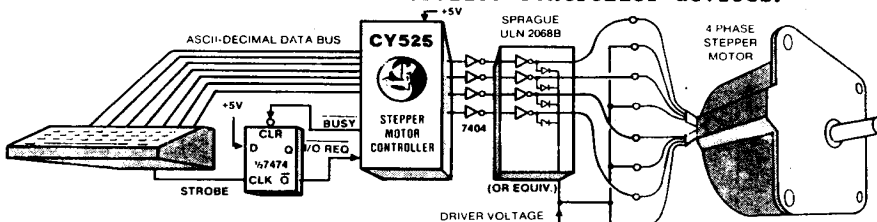

Figure 2.2   Simplest Prototype Development System.

16

SYNCHRONIZATION MECHANISMS

Most stepper motors are employed as parts of functional systems.
These systems often must synchronize the behavior of the various
subsystems to each other or to a real-world occurrence, such as
an operator input. The CY525 has been designed with both signal
emitters and detectors to allow easy synchronization of the
device to neighboring (interacting) subsystems.

The motor interface for the CY525 is very simple, consisting of
six output signals. Since the controller is designed for four
phase motors, there is a signal line for each phase. The
patterns necessary to operate the motor, including sequencing for
proper direction, appear on the phase outputs. A simple L/R type
driving circuit may be connected directly to the phase outputs,
so the motor can be run from the controller signals.
Alternatively, the user could drive a more sophisticated pulse-
to-step translator, using the CY525 Pulse and Direction outputs.
The Pulse line gives one pulse at the beginning of each step,
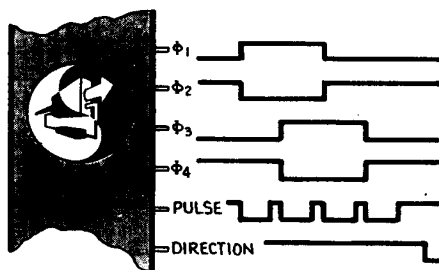while the Direction line always indicates the current stepping
direction.

Figure 2.3 Motor Interface

The computer or data interface of the CY525 is also very simple.
Commands and parameters are passed from the command source to the
CY525 on an eight bit, bidirectional data bus. Direction of data
is determined by the level on the I/O Select line, allowing the
CY525 to not only receive commands, but also to be interrogated
for the current values of its parameters and contents of the
program buffer. Data transfer between the command source and the
CY525 is controlled by a standard two-line handshake protocol.
The master processor waits for the CY525 Busy/Ready line to go
high, indicating that the CY525 is ready for the next command
byte. Data may then be placed on the bus, and data available is
indicated by a high-to-low transition of I/O Request. Data
should remain stable until the CY525 indicates data accepted by a
high-to-low transition of the Busy/Ready line. During this busy
time, the CY525 is processing the character just received. The
master processor should then raise I/O Request and wait until the
CY525 is ready for the next data byte. Data transfers from the
CY525 to the master processor are handled in a similar way, with
the master requesting the next byte using I/O Request, and the

CY525 indicating data available using Busy/Ready. The simplicity of the data transfer handshake, combined with the ASCII command structure of the CY525, allows the commanding device to be any of a number of things, including a microprocessor or other computer, a keyboard for manual command entry, or a ROM for fixed, stand-alone applications. The keyboard is especially useful during prototype development or system characterization.
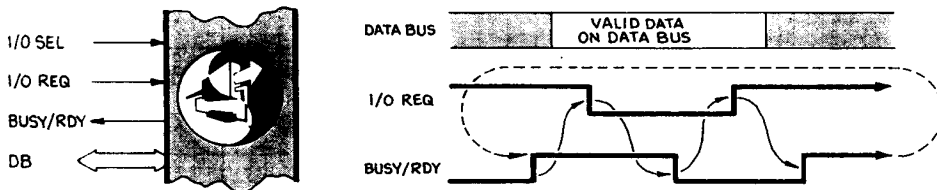

Figure 2.4   Data interface and handshake waveform

Since most stepper motors are parts of functional systems, requiring that various parts of the system stay synchronized with each other, the CY525 has been designed with a number of secondary input and output control lines. These signals may be used to modify and control the stepping behavior of the device, or indicate certain conditions within the controller. Two inputs control the stepping behavior directly. While Step Inhibit is high, the controller will not step. Stepping is resumed when the signal goes low again. This signal may be used to halt a motion under emergency conditions, or to slow the step rate if the motor cannot keep up. The Abort signal is used to cause a deceleration to the starting rate (Abort), at which point the motion may be stopped. Two other inputs modify the way a program is executed. The Wait line is used to suspend a program until the signal level on that line is in a certain state. Commands allow the program to wait for either a high level or a low level, making it possible to synchronize on either transition of the line. The Dowhile input is used with the conditional loop command. While the line is low, the CY525 will loop back to the specified program location, repeating the program section over and over. When the line goes high, the controller will continue with the rest of the program.
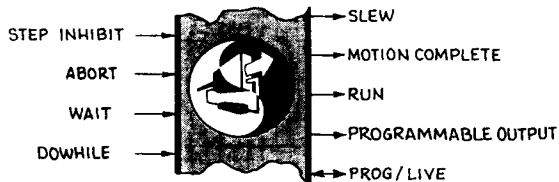

Figure 2.5   Secondary control inputs and outputs

18

The CY525 also provides a number of output signals which may be used by other parts of the system. While stepping, the Slew line indicates that the CY525 has reached the maximum programmed step rate, and is not accelerating or decelerating. When the CY525 has stepped for the number of steps specified, the Motion Complete signal indicates the end of the current motion. Run is used to indicate that a program is executing. In addition, the CY525 provides an uncommitted output, Programmable Output, which the user may apply as needed. The level on this output is controlled by two commands, one for a high output, and the other for a low output.

CY525 PINOUT DIAGRAM

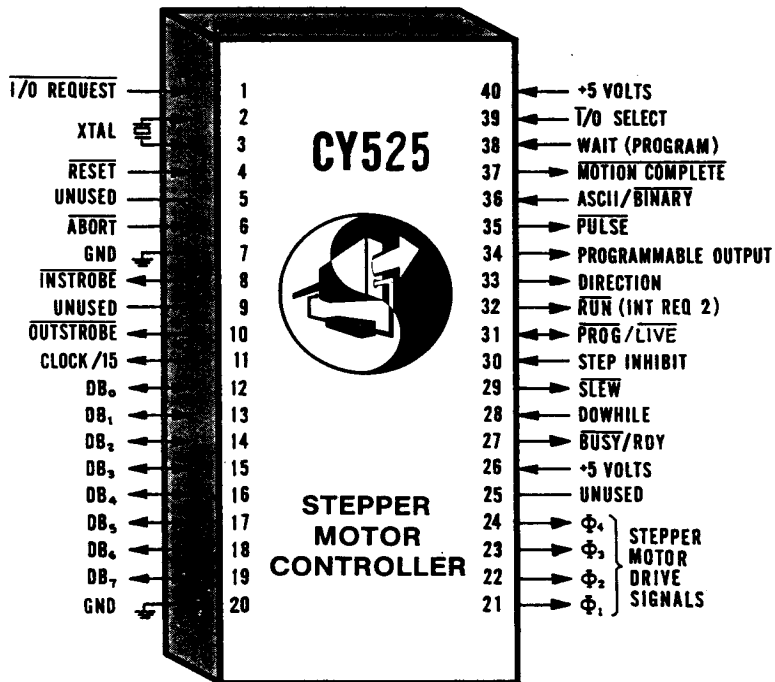The CY525 pinout diagram is shown below, followed by the table of pin definitions.

$\overline{\text{I/O REQUEST}}$	1		40	+5 VOLTS
XTAL	2	**CY525**	39	$\overline{\text{I/O}}$ SELECT
$\overline{\text{RESET}}$	3		38	WAIT (PROGRAM)
UNUSED	4		37	$\overline{\text{MOTION COMPLETE}}$
$\overline{\text{ABORT}}$	5		36	ASCII/$\overline{\text{BINARY}}$
GND	6		35	$\overline{\text{PULSE}}$
$\overline{\text{INSTROBE}}$	7		34	PROGRAMMABLE OUTPUT
UNUSED	8		33	DIRECTION
$\overline{\text{OUTSTROBE}}$	9		32	$\overline{\text{RUN}}$ (INT REQ 2)
CLOCK/15	10		31	$\overline{\text{PROG}}$/LIVE
DB$_0$	11		30	STEP INHIBIT
DB$_1$	12		29	$\overline{\text{SLEW}}$
DB$_2$	13		28	DOWHILE
DB$_3$	14		27	$\overline{\text{BUSY}}$/RDY
DB$_4$	15		26	+5 VOLTS
DB$_5$	16	**STEPPER**	25	UNUSED
DB$_6$	17	**MOTOR**	24	Φ_4
DB$_7$	18	**CONTROLLER**	23	Φ_3
GND	19		22	Φ_2
	20		21	Φ_1

STEPPER MOTOR DRIVE SIGNALS

Figure 2.6 CY525 Pin definition

19

TABLE I	CY525 PIN DESCRIPTION	
DESIGNATION	PIN#	FUNCTION
VCC	40	+5 volt power supply.
VDD	26	+5 volts.
VSS	7,20	circuit GND potential.
Xtall-Xtal2 (input)	2,3	inputs for crystal or external clock (not TTL). See Clock Circuits section.
Clk/15 (output)	11	This output represents the crystal frequency divided by fifteen. The pulse width is at least 300 nanoseconds.
Reset (input)	4	initializes controller to power-up state.
DB0-DB7	12-19	bidirectional parallel data bus.
I/O Select	39	indicates direction of data on the data (input) bus. Low = input to CY525. Hi = output from CY525, which can only be generated if CY525 has received "V" command. Also used to read position while stepping, i.e., on the fly.
I/O Request (input)	1	strobe to initiate command input when writing to CY525. Initiate data output when reading from CY525. Interpretation of pin 1 is a function of I/O Select (pin 39). May be used while stepping to change the step rate on the fly.
Busy/Ready (output)	27	handshake line for command data input. Host must wait until Ready state is indicated by a high level before transferring command or data to CY525. If Run (pin 32) is low, the Prog line (pin 31) must be used to enter Live commands while a program is executing.
Instrobe (output)	8	occurs during data input. The data on the bus must be valid until the trailing edge of Instrobe occurs.
Outstrobe (output)	10	Trailing edge indicates valid data output by CY525 on data bus.
ASCII/Binary (input)	36	selects ASCII-decimal or binary mode of operation.

| TABLE I | | CY525 PIN DESCRIPTION | (continued) |

DESIGNATION	PIN#	FUNCTION
Prog/Live (output/input)	31	indicates program entry mode. Commands are entered and saved, but not executed, while pin 31 is low. May be used as input to enable Live commands to be executed while a program runs.
Run (Int Req 2) (Program Complete) (output)	32	indicates program execution mode. Commands cannot be entered while program is executing (pin 32 = low) unless the Prog line (pin 31) is used.
Motion Complete (Int Req 1) (output)	37	signal to interrupt host at end of stepping or when position is available.
Wait (input)	38	program Waits for this pin to go LOW when "Until" command is executed, and waits for a High signal when "Wait" command is executed.
Dowhile (input)	28	is tested by "T" command. Program will branch to specified target if low, else it will execute next instruction.
Direction (output)	33	indicates current stepping direction and is affected by +, -, and "P" commands (Hi = CW, low = CCW).
Pulse (output)	35	low when step begins, hi when step ends.
Slew (output)	29	goes low when steady stepping rate has been achieved. Will return high when ramping begins.
Abort (input)	6	Low during stepping causes the CY525 to begin ramping down to the initial step rate. If held low, the CY525 Aborts stepping at the bottom of the ramp. If the Abort line is returned high during the downramp, the CY525 ramps down and continues stepping to target position at the initial step rate.
Step Inhibit (input)	30	inhibits stepping while held high.
Programmable Output	34	user programmable output pin.
$\phi1-\phi4$ (output)	21-24	stepper drive signals.
Unused	5,9,25	must remain disconnected.

21

CY525 MOTION STATUS SIGNALS

The CY525 provides three status signals that provide information about the motion during the execution of instructions that cause the motor to move. These three status lines are:

1.	$\overline{\text{BUSY}}$/READY	(pin 27)
2.	$\overline{\text{SLEW}}$	(pin 29)
3.	$\overline{\text{MOTION COMPLETE}}$	(pin 37)

Each of these will be discussed separately.

Busy/Ready

When a motion command (P or G) is issued to the CY525 the BUSY/READY line goes low to acknowledge the terminating carriage return character and remains low throughout the motion, returning high at the end of the motion. When a P or G is executed while a progam is running, the BUSY/READY line goes low prior to the first step and remains low until the last step is taken.

Slew

The SLEW line (pin 29) indicates constant velocity motion when it is low. If the CY525 requires acceleration (R not equal to F) the SLEW line is initially high, going low when the specified step rate is reached. When the CY525 begins decelerating at the end of travel, the SLEW line returns high. In the normal mode of operation the SLEW signal is low only when the CY525 is stepping at the maximum specified rate. In the continuous step or HALT mode of operation (initiated via the H command) the SLEW signal is low when the specified velocity is reached and goes high every time a new velocity is specified, remaining high while the device accelerated or decelerates to the new velocity, then returning low until another change in velocity is specified (on the fly).

Motion Complete

The MOTION COMPLETE signal (pin 37) indicates that the last step of the current motion has occurred. This signal may be used to interrupt the master computer. (Note that MOTION COMPLETE also goes low for a few microseconds when the position is read on the fly).

These signals are shown in the following figure for a typical
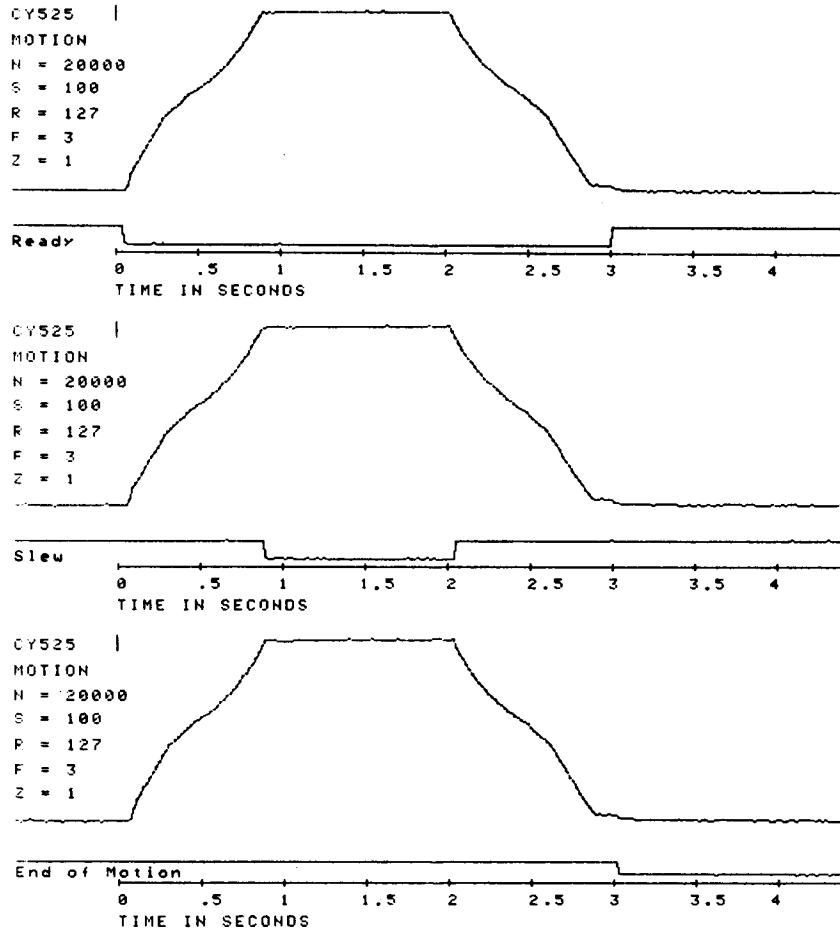motion. An 11 MHz crystal is used.

Figure 2.7   Illustrating the three motion status lines for a
              typical CY525 motion command.

3 OVERVIEW OF COMMAND LANGUAGE 3

BIN-ASCII™ FEATURE

The CY525 user-orientation has been accomplished without the expense of complicating the host programming job. For example, the ASCII-decimal integers typed by the user at the keyboard may not be readily available in the host programming language. For this reason the CY525 can be placed in a **binary** mode in which binary number parameters are used instead of ASCII-decimal. This allows any computer with binary integer arithmetic to send commands and binary information to the controller. **The CY525 is placed in either the binary or the ASCII-decimal mode via a mode-select input pin setting.**

The use of ASCII instruction and ASCII-decimal integer parameters allows the user to type commands in familiar high-level language formats, as shown below:

        N 738)   ; set Number of steps = 738
        G)   ; Go (begin stepping)

where "N" is the ASCII command specifying NUMBER of steps to take. The ASCII space character is shown as a space, and the decimal number "738" is then entered, followed by the carriage return key, ")"= 0DH which terminates the commands. The GO command is entered as "G)". The controller then steps the motor for 738 steps. Other parameters, such as rate, may be specified in similar fashion.

Although the use of ASCII-decimal numbers is ideal for the user employing BASIC or other languages that can output ASCII-decimal numbers, it is, of course, desirable that the controller accept binary number parameters from binary computations. For this reason, the CY525 Stepper Controller may be placed in a BINARY mode via a strap, or mode-select, pin. In this mode, all numbers are interpreted as binary data (as are all commands). See the section on binary data mode for details.

PROGRAMMABLE WITH HIGH LEVEL LANGUAGE

The primary advantage of all hi-level languages is their problem-oriented nature, as opposed to the device-oriented nature of machine languages. A secondary characteristic is their ASCII representation, and a third characteristic of most hi-level languages is their use of the ASCII-decimal numbering system as natural numbers. In all of these aspects, the CY525 qualifies as **a single chip Hi-Level Language Device.** The combination of hi-level language and ASCII-keyboard programmability is designed to maximize user ease and convenience.

Every instruction entered in the ASCII decimal mode of operation consists of one of the following forms:

1. Alphabetic ASCII character followed by the "$\downarrow$" (RETURN) key.

2. Alphabetic ASCII character followed by space, then ASCII decimal number parameter, then "$\downarrow$" = 0DH.

Examples of type one are as follows:

NAME	COMMAND	INTERPRETATION
Bitset	B$\downarrow$	Set programmable output line
Clearbit	C$\downarrow$	Clear programmable output Line
eXecute	X$\downarrow$	eXecute program
Enter	E$\downarrow$	Enter program mode

Examples of type two are as follows:

NAME	ASCII COMMAND	INTERPRETATION
Absolute	A a$\downarrow$	Declare absolute position
Number	N n$\downarrow$	Declare number of steps to be taken (relative)
Rate	R r$\downarrow$	Declare maximum rate parameter
Slope	S s$\downarrow$	Declare ramp rate
Firstrate	F f$\downarrow$	Declare initial step rate
Position	P p$\downarrow$	Declare target position (absolute)

TABLE II		CY525 COMMAND SUMMARY
ASCII	NAME	INTERPRETATION
A	Absolute	Set current location as specified
B	Bitset	Set programmable output line high
C	Clearbit	Reset programmable output line low
D	Delay	Time delay for specified milliseconds
E	Enter	Enter program code
F	Firstrate	Set initial step rate
G	Go	Begin relative stepping operation
H	Haltmode	Set continuous step mode of operation
I	Initialize	Turn off step drive lines, reset controller
J	Jump	Go to specified program buffer location
L	Loop	Repeat program segment for specified count
N	Number	Set number of steps to be taken (relative)
O	Offset	Set next stepper drive signal value
P	Position	Set and step to target position (absolute)
Q *	Quit*	Quit entering program code, re-enter command mode. *Never followed by "}"
R	Rate	Set step rate parameter
S	Slope	Set ramp rate for slew mode operation
T	Branch Til	Branch "Til" dowhile line goes high
U	Until	Stop execution until wait line is low
V	Verify	Verify internal buffer contents
W	Wait	Stop executing until wait line is high
X	eXecute	Begin program execution
Z	divisor	Divide slope by divisor parameter
+	CW	Set clockwise direction
−	CCW	Set counterclockwise direction
Ø	Command	Stop program execution, enter command mode
$	Label	Marker for "jump to" and "loop to" commands

DESCRIPTION OF COMMANDS

The command format shown in the following descriptions indicates the way commands are stored in the program buffer, as well as showing the binary values of the command letters. Note that in Binary mode the user must insert a data count between the command letters and parameters, if any. See section 5 on Binary Data Mode. Also note that 16 bit parameters (number and position) are entered least significant byte first in the Binary mode. In the ASCII mode, command letters are separated from parameters by a single space, and the parameters are entered as ASCII decimal numbers. ASCII mode commands are terminated by a carriage return, as indicated in the leftmost column of the description,

A a) ABSOLUTE

0100 0001		3 bytes
a7	a0	LSbyte
b7	b0	MSbyte

The ABSOLUTE instruction defines the current position. This position is set to the value specified by the 16-bit argument. Absolute positions are used by all POSITION commands. The ABSOLUTE command may be used at any time to define or redefine any position coordinate.

B) BITSET

| 0100 0010 | 1 byte |

This instruction causes the programmable output pin (#34) to go HIGH. This is a general-purpose output that may be used in any fashion.

C) CLEARBIT

| 0100 0011 | 1 byte |

This instruction causes the programmable output pin (#34) to go LOW. The user can signal locations in a program sequence to the external world via B and C instructions.

D d) DELAY

0100 0100		3 bytes
a7	a0	LSbyte
b7	b0	MSbyte

The DELAY command will time delay for the number of milliseconds specified by its argument. The delay is calibrated in milliseconds, using an 11 MHz crystal. Other frequencies will require a linear scaling for the actual delay time. Since this is a 3 byte command (16 bit argument) the delay time can range between 1 msec and about 65.5 sec at 11 MHz. This command is useful in programming a delay time between stepping motions.

E) ENTER

| 0100 0101 | 1 byte |

This instruction initiates the program entry mode of operation. Commands following the ENTER command are saved in the program buffer in sequence until "Q" is entered. The PROG line (# 31) goes low to indicate this mode.

F f) FIRSTRATE | 0100 0110 | 2 bytes
                            | a7   a0 |

The argument, f, is a number from 0 to 127 and is used to specify the initial step rate for every move.

G) GO | 0100 0111 | 1 byte

The GO command causes the stepper motor to step as specified by the rate, direction, etc., commands entered prior to the GO command. Stepping will be in the relative mode, with the number of steps defined by the N command.

H) HALTmode | 0100 1000 | 1 byte

The HALTmode command initiates the continuous run mode. In this mode the CY525 begins stepping when the next P or G instruction is executed, ramps up to the specified rate and then continues to run until HALTed by a low signal on pin 6, the ABORT pin. This allows the CY525 to take an unlimited number of steps instead of the usual 64K step limit. During this mode the step Rate may be changed "on-the-Fly" and the current position may be read "on-the-fly". The low ABORT line will cause the CY525 to begin ramping down until it reaches the FIRSTrate and then stop (if the ABORT is still low). Haltmode is reset after the motion stops, returning the CY525 to the default mode in which the target or number of steps is specified.

I) INITIALIZE | 0100 1001 | 1 byte

The INITIALIZE command causes the CY525 to enter the command mode. **None of the distance or rate parameters are altered.** Any commands following "I" will be executed with the parameters specified prior to "I". The INITIALIZE command, when encountered during program execution, halts the program execution and returns the system to the command mode. This command de-energizes the stepper motor coils, erases the program, if any, sets the direction to CW.

J j) JUMP | 0100 1010 | 2 bytes
                            | a7   a0 |

The JUMP command will branch program execution to the program buffer location specified by the argument, which represents the byte number in the program buffer, starting with zero. Program execution continues from the specified byte number after the jump is executed. When the JUMP command is issued from the Command mode, it enables the Run (Program execution) mode, and begins executing from the specified byte number. **It is the user's responsibility to insure that the number specified with the JUMP command is the correct value for the desired starting point.** The first byte after the "E" command is location zero. The byte count

28

specified in this section determines the number of bytes
used by each instruction. Note that "J 0)" is equivalent to
"X)". In order to simplify the use of the Jump (and other
branching instructions) the CY525 allows the use of LABELs.
An instruction is given a label (A thru E) by preceding the
instruction with the label character followed by a $. See
description of $ at end of this section.

L c,a) LOOP

0100 1100		3 bytes
a7	a0	COUNT
b7	b0	LOCATION

The LOOP command uses the first argument as a repetition
count, and the second argument as a jump location. Each
time the LOOP command is executed, the count is decremented
by one. If the count is nonzero after the decrement,
program execution will jump to the specified address, which
is the second argument of the command. The jump address
specifies the location to which execution will loop, and the
count represents the number of times the loop is to be
repeated. When the count reaches zero, program execution
continues with the instruction immediately following the
LOOP command. In ASCII mode, the two arguments may be
separated by either a comma, or a single space. LOOP
commands may not be nested one inside the other. Labels may
be used with the LOOP command, as with the JUMP command.

N n) NUMBER

0100 1110		3 bytes
a7	a0	LSbyte
b7	b0	MSbyte

The NUMBER command is used to specify the number of steps to
be taken in the "Relative" mode of operation. The argument
may be any number from 1 to 64K-1 (65,535). Note that this
parameter is stored as 2 bytes in the program buffer.

O o) OFFSET

0100 1111		2 bytes
a7	a0	

The OFFSET command specifies the next step pattern to appear
on the STEPPER MOTOR DRIVE SIGNALS, pins 21-24. This
command may be used to synchronize these outputs with the
motor when the desired pattern is known from the motor's
position. The OFFSET command can also be used to turn OFF
all of the 4 phases, by specifying a value greater than 3.
If the argument is greater than three, but less than 128,
then all 4 phases are driven high, and the pulse line (pin
35) does not change. If the argument is greater than 127,
the pulse line is brought low and then goes high as all 4
phase lines go high.

29

P p) POSITION

0101 0000		3 bytes
a7	a0	LSbyte
b7	b0	MSbyte

The POSITION command declares the "Absolute" mode of operation. The argument is treated as the target position relative to position zero. The ABSOLUTE command is used to define any position. Stepping to the target position begins when the POSITION command is executed. No "G" command is required. Direction to the target position is also determined and set automatically.

Q QUIT (Programming) | 0101 0001 | 1 byte

NOTE: The QUIT command is self-terminating, and should NOT be followed by the Linend "}" or data count. Such termination may result in incorrect operation.

The QUIT command causes the CY525 to exit the "Programming" mode of operation, wherein instructions are stored in the program buffer in the order received; and causes the CY525 to return to the "Command" mode of operation, in which instructions are executed as they are received.

R r) RATE

| 0101 0010 | | 2 bytes |
| a7 | a0 | |

The RATE instruction sets the rate parameter that determines the step rate. The rate parameter, r, varies from 0 to 127 corresponding to step rates from 275 to 9675 steps/sec (with an 11 MHz crystal). For rate parameters between 3 and 40 the rate is linear and is specified in units of 100 steps/sec! For rate parameters between 41 and 127 the rate is still highly linear but the increments are approximately 60 steps/sec. For crystals other than 11 MHz the step rate should be multiplied by fc/11 Mhz, where fc is the crystal frequency.

S s) SLOPE

| 0101 0011 | | 2 bytes |
| a7 | a0 | |

The SLOPE or slew mode of operation is used when high step rates are required and the initial load on the motor prevents instantaneous stepping at such rates. In such cases, the load is accelerated from rest to the maximum rate and then decelerated to a stop. The user specifies the distance of total travel (via "N" instruction), the maximum rate (via "R") and the slope rate ("S") or change in rate from step to step. The CY525, starting from rest or from a specified starting rate, will increase the rate in such a fashion that a maximum acceleration (determined by the value of the slope parameter) is not exceeded. For very slow accelerations the slope may be stretched by a slope divider, specified by the argument of the Z command.

T t) loop TIL

0101 0100
a7   a0

2 bytes

The "T" command provides a "Do...While..." capability to the CY525. This command tests pin 28 and, if low, it branches to the specified target instruction. Note that the target can be either a number or a label. If pin 28 is high, the instruction following the T command is fetched and executed.

U) wait-UNTIL

0101 0101

1 byte

The wait-UNTIL instruction is used to synchronize the program execution to an external event. When the "U" instruction is executed, the WAIT pin (pin #38) is tested. When the WAIT pin goes low, the next instruction is fetched from the program buffer and execution proceeds.

V v) VERIFY

0101 0110
a7   a0

2 bytes

The VERIFY command allows interrogation of the internal CY525 buffers, including the rate, slope, number of steps, and current position registers. This command is also used to examine the current contents of the CY525 program buffer. The parameter "v" specifies which internal register group is to be read. VERIFY should only be executed from the command mode. To read data during the RUN mode, the Prog/Live line must be used. Note that position can be read "on-the-fly" while the motor is stepping, but this is NOT done with a Verify command! See "Verify Mode" in Section 6 for details.

W) WAIT

0101 0111

1 byte

The WAIT instruction is the opposite of the U command. WAIT tests the WAIT pin (pin #38) for a high level. The program will stop until the pin is high, then it will continue with the next command. Note that the U command may be used to detect the falling edge of the WAIT line signal, and the W command may be used to detect the rising edge. Thus, it is possible to synchronize program execution to either one or both of the transitions.

X) EXECUTE

0101 1000

1 byte

This instruction causes the CY525 to begin executing the stored program. If no program has been entered, the controller will stay in the Run mode unless a Stop Operation is executed (via the Ø command). If the program exists, the controller will begin execution of the first instruction in the program buffer. If the EXECUTE command is encountered during program execution, it restarts the program (however, the initial parameters and modes may have been redefined later in the program) and may be used for looping or cyclic repetition of the program.

31

Z z) Divisor | 0101 1010 | 2 bytes
                               | a7   a0 |

The DIVISOR command provides slower accelerations by dividing the slope by the value of the argument of Z.

+) CLOCKWISE | 0010 1011 | 1 byte

Direction is set to clockwise by this command. Relative mode steps are taken in the direction last specified, so they will be clockwise until the direction is changed by the "-" or "P" commands. The current direction is always indicated on the DIRECTION line (pin # 33).

-) CCW | 0010 1101 | 1 byte

Direction is set to counter-clockwise by this command. Comments under CLOCKWISE also apply to the "-" command.

0) COMMAND | 0011 0000 | 1 byte

The CY525 is placed in the command mode and the next command is executed as it is received. Programs should be terminated by a "0" command, returning the CY525 to command mode at the end of program execution.

n$ Label Designator

The CY525 allows the use of the ASCII characters A through E (in ASCII mode) followed by the ASCII $ symbol as labels for instructions in a program. There should be no spaces between the $ sign and the adjoining characters. The labels can be used as branch targets for "JUMP" and "LOOP" instructions and thus relieve the user of the need to count bytes to determine the target address. Labels accompanying Jump and Loop are initially stored in the program buffer as the characters A-E, but are then replaced with the actual address location when the program is executed. Label designators are not allowed in binary mode.

        Example:

        A$C)   ; label A for instruction C
          B)   ; the BITSET instruction
          :
          :
        J A)   ; Jump to instruction with Label A$

The labels can be used in any order, and up to five labels can be used in one program. The two-character labels do not take up any space in the program buffer.

COMPARISON TO THE CY512

The following summary of differences between the CY512 and the CY525 is provided for those users of the CY512 who are upgrading to the CY525.

The following instructions are new or modified for the CY525. All of the remaining instuctions are identical to the CY512.

A a⟩	(a = 0..64k)	set current position = a
D d⟩	(d = 1..64k)	delay d milliseconds (11MHz Xtal) whereas the CY512 delay command was eXpend.
F f⟩	(f = 0..127)	set FIRST rate = f
H⟩	no argument (This was the halfstep command in the CY512. There is no half-step in the CY525.)	set HALT mode, in which the CY525 will ramp from initial rate F to maximum rate R and then run continuously an unlimited number of steps until the Abort line (pin 6) is asserted low. Note that the position register contents will "wrap around" at 64K but there will be no other effect.
O o⟩	(o = 0..255)	o = (0,1,2,3) --set output phase and turn on phase outputs. o > 3 -- drive all 4 phase lines high but do not change internal phase value.
R r⟩	(r = 0..127)	specifies maximum rate. The curve is more linear than CY512 and parameter smaller than the CY512 (which had up to r=256).
S s⟩	(s = 1..255)	specifies acceleration (slope) 255 = maximum acceleration 1 = minimum acceleration
T t⟩	(t = 0..48)	specifies target to branch to when test is satisfied.
X⟩	no argument	execute the stored program, whereas the CY512 run command was Doitnow.
Z z⟩	(z = 1..255)	"slope divider" – divides acceleration by "z"

Additional Changes in CY525

1. **Read-on-the-Fly:** Binary position readout is now possible while the step motor is moving. The position appears on the bus as two binary bytes. The high byte appears first and can be latched by the Outstrobe (pin 10) and the low byte follows immediately and can also be latched by the Outstrobe. The position is requested by keeping I/O Req high and raising I/O Select.

   The Motion Complete line (pin 37) will be high when the initial strobe (pin 10) signals that the high byte is on the bus and then will go low and remain low while the low byte is strobed onto the bus via pin 10. The Motion Complete line will then return high and remain high until another position readout occurs or until the motion is complete.

   Note that since the Motion Complete line is often used to interrupt the host, this is simply an extension in the sense that it can now be used to announce the availability of the readout information as well as the end of motion. If used this way, it is up to the user to interpret its meaning.

2. **Labels:** Commands with targets can use labels. A label consists of one of the letters **A** thru **E** followed by a $. A program can use from one to five labels.

   Example: A$P 40⤸
                    C⤸
                    P 0⤸
                    B⤸
                    J A⤸   (jump to location A)

3. **Change rate while moving:** The CY525 can use the I/O Req line to send a new rate while the device is slewing. The values sent to the CY525 must be 8-bit binary numbers in the appropriate range. The CY525 can be in either binary or ASCII mode.

4. **Issue commands while a program is running:** The PROG/LIVE line (pin 31) can be pulled low while a program is running. This tells the CY525 to finish the currently executing instruction and wait for a new instruction to appear using the standard handshake. If several instructions are to be issued, the PROG/LIVE line should be held low. It should be brought high before the ⤸ = 0Dh character is issued for the last instruction. Note that the PROG/LIVE line is still held low by the CY525 when the part is in the Program Entry mode, i.e., following the command **E**⤸ and remains low until this mode is terminated by the **Q** command.

This feature allows the user to change parameters while a program is executing. Note that this is different from the new "On-th-fly" Position readout and rate changes that occur while the device is actually stepping. To issue "on-the-fly" commands, the I/O Req or I/O Sel lines are made active while the device is stepping. To issue instructions, these two are unchanged until after the PROG/LIVE line is activated (low) and the RDY line is brought high.

5. **Changes in the ABORT function:** (see ABORT section.. page 63)

6. **The PULSE output timing is different,** with the pulse line going high for only 2.5 microseconds (@ 11 MHz) between steps. Also, when the step inhibit pin is tested to lengthen a step time (delay the next step), the pulse output will be low while the CY525 is inhibited. This has no effect on those applications which use only the phase outputs, i.e., in these cases the CY512 and CY525 response to the inhibit pin will be identical.

7. **The BUSY/READY line** (pin 27) of the CY525 goes low at the beginning of each motion and remains low until the motion has been completed. The CY512 BUSY/READY line did not go low for motion instructions executed within a program. This feature is useful for monitoring motion in a program and also allows handshaking for LIVE commands.

8. **The CY525 allows WAIT and UNTIL commands to be terminated** by bringing the I/O-request line (pin 1) low. Since these instructions monitor an external line, the master computer can restore execution in the case in which the external line is "stuck", thus preventing the system from hanging up.

9. **When a motion command is given,** the CY525 will not initiate stepping until the I/O-request line has returned high. This prevents the I/O-request from being incorrectly interpreted as a request to change the step rate on the fly, i.e., while the motor is moving. The CY512 could begin stepping before the I/O-request line returned high. This could cause a software abort to occur if a slow BASIC interpreter left the I/O-request line low too long. Note that the CY525 software abort differs from the CY512 software abort!! (see the ABORT discussion in section 6)

10. **The CY525 will accelerate** even if the travel distance is too short to reach the specified step rate. This provides optimal motion for all distances. The CY512 would step at the slowest rate if it could not reach the specified rate before having to begin decelerating to stop at the specified location.

11. In the CY512, the X-command represents a time delay and the D-command runs the stored program. The CY525 is reversed: the **X-command executes the program** and the D-command represents a time delay. This change provides the CY525 greater compatibility with the entire CYxxx family of chips.

4 DETAILED EXAMPLES OF COMMANDS 4

RESET COMMAND (INITIALIZE)

The "I" or Initialize command resets all pointers to the **power-up state** and restores the flags to this state. Specifically, the program is erased and the command mode entered. The direction is clockwise (CW). Note that this command de-energizes the stepper coils (phase outputs all go high). If this effect is undesirable, an external latch should be used to latch the four stepper control outputs using the pulse line (pin 35) to clock the latch. See Figure 4.1.
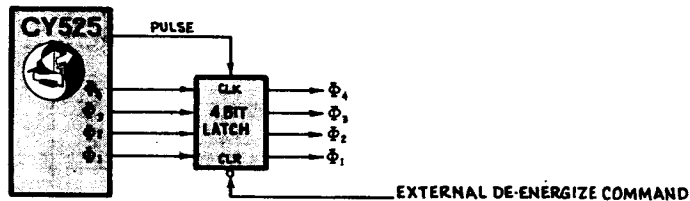
Figure 4.1   An external latch on the stepper control outputs prevents de-energizing of stepper drive coils when CY525 is reset via hardware or software and also allows an external control line to de-energize the coils independently of the CY525.

PROGRAM EXECUTION MODE

Once a program is entered into the program buffer, it may be executed with a run or EXECUTE ("X") command. This code has been assigned the ASCII value "X" = 58H. It is the last command to be entered before program execution. It is a normal command in the sense that it is terminated with a carriage return, "}" = ØDH.

ABSOLUTE POSITION

The "Absolute" command, A, is used to declare the current position, assigned the value of the 16-bit argument. Thus the current position can be specified as any location from Ø to 64K. All absolute movements, affected by the POSITION command, P, are referenced to this position. On power-up, the absolute position is random. Therefore, the A command should be used to define a coordinate system, before the absolute position commands are utilized.

PROGRAM LOOPING, ITERATION

One consequence of stored program execution is the use of program loops or program repetition. If the EXECUTE command of the CY525

is included as a program instruction, the program executes again beginning with the first instruction (but using the latest value of parameters set before the EXECUTE instruction was encountered). In this fashion, rather complex sequences of motions may be repeated without host intervention or interruption. Conditional looping may be accomplished with a "Do While" type instruction that continues looping until a condition is fulfilled. This may be combined with the JUMP command, for unconditional branching to various routines in the program buffer. Finally, a subsection of a program may be repeated a specific number of times by use of the LOOP instruction.

Unconditional Program Looping

If the EXECUTE command, "X", is encountered in the program entry mode, it is stored in the program buffer with the rest of the program. When this instruction is encountered during the program execution, its effect is to begin program execution again, and therefore may be used to achieve cyclical looping if desired. However, program execution may be aborted via the RESET line.

Conditional Program Looping

The ability to repeatedly branch to a specified instruction in a program until an external event occurs provides a unique "Do...While..." capability for the CY525. The "T" command (loop TIL) is used as shown in the following example.

N n)	set number of steps = n
E)	enter program mode
A$+)	label location "A" and set CW direction
R r1)	set rate = r1
G)	begin stepping
-)	set CCW direction
R r2)	set new rate = r2
G)	go (same "n")
T A)	goto A if pin 28 is LO
B)	set programmable output
0)	return to command mode
Q	quit progam mode
X)	begin executing program
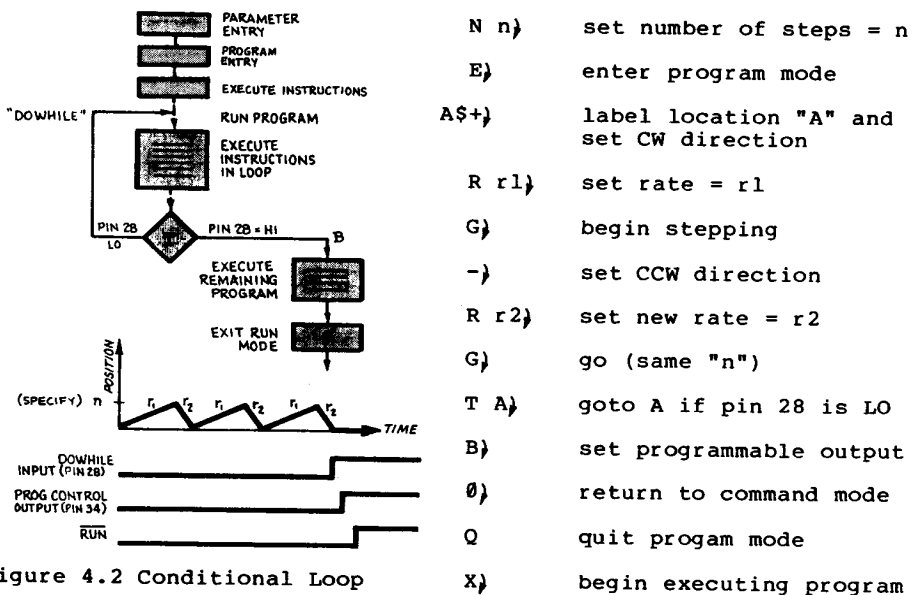
Figure 4.2 Conditional Loop

37

Conditional Loop Embedded in an Unconditional Loop

"Do (the preceding program) While (Pin 28 is low)", then proceed
to execute the remaining program instructions. Note that the
program can (but need not) end with an EXECUTE instruction to
provide a conditional loop inside of an unconditional loop:

R rl)	set rate parameter
E)	enter program mode
C$N nl)	label location "C" and set first distance
+)	set CW direction
G)	take nl steps
-)	set CCW direction
N n2)	distance parameter
G)	take n2 steps
T C)	loop to C TIL pin #28 = HI
B)	set output HI (pin 34)
R r2)	set new rate
X)	repeat program
Q	exit program mode
X)	begin executing program
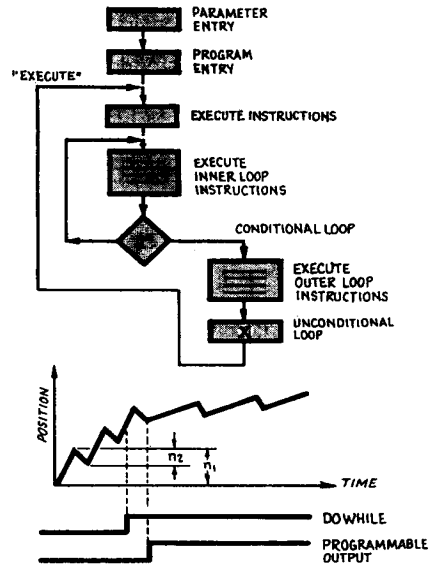
Figure 4.3
Conditional Loop Embedded
in an Unconditional Loop.

The Wait line allows a program
to be suspended until an
external event occurs. As long
as the Wait line is in one
state, the program continually
tests the line without executing
any other instructions. When
the line changes to the state
being waited for, the program
continues with the commands
following the wait instruction.
Figure 4.4 illustrates the wait
Until command, for which the
wait line must be low to
continue.
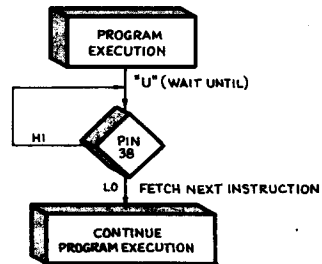
Figure 4.4   Wait UNTIL command use.

38

JUMP Command Use

Unconditional branching to any location in the program buffer may be accomplished by the Jump command. The single argument of the command specifies the program buffer location at which program execution will continue. This number is simply the byte number within the program buffer, with the first location designated as byte zero. Since the program buffer is 60 bytes long, arguments for the Jump command range in value from 0 to 59. The actual value used should correspond to the beginning of the instruction which is to be executed next. The byte numbers may be determined by adding the number of bytes used for each of the previous commands in the buffer. Byte counts are specified in Description of Commands in Section 3.

Note that the CY525 allows the use of labels, using
the characters A thru E to specify a branch target. A
label is defined by using the label character
followed by the $ symbol preceding the target
instruction. Up to 5 labels may be used in a program.

The following example contains a Jump command used to repeat a section of the program. The program steps to the home position at high speed, then repeats motions of 75 steps at a lower speed, waiting for a synchronizing signal before taking each motion. The Jump command may also be used in the Command mode, to start program execution at a location other than the first command of a program. This example assumes the home position has been previousy defined, using the "Absolute" command.

E)	enter program mode
C)	lower programmable out pin
R 127)	define fast step rate
P 0)	step quickly to home position
+)	CW direction for next steps
N 75)	take 75 steps per motion
R 101)	define slower step rate
A$B)	raise programmable out pin
W)	wait for high on pin 30
C)	lower programmable out pin
G)	take 75 steps
J A)	repeat from "B" command
Q	quit program mode
S 20)	define acceleration slope
F 3)	define first stepping rate
X)	begin running program
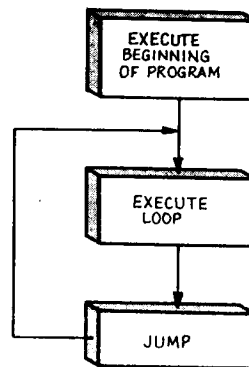
Figure 4.5 JUMP Command Use

WELDING MACHINE EXAMPLE

Suppose we want a row of six equally spaced welds on a piece of metal. The welder should be turned on by the CY525 programmable output line when in position, and be turned off when finished. After completing six such welds, it will return as quickly as possible to its starting position, and wait for the next workpiece to come into position. It will then weld the next six spots, and continue in this manner until there are no more pieces to weld, at which time the program will stop and the CY525 will return to the Command mode.

```
R 120}  ⎫
S 80}   ⎬  define stepping parameters
F 9}    ⎭

A 0}       declare current position as home

E}         enter and save the following program

  B$N 20}    take 20 steps between welds
    +}       CW direction
    W}       wait until workpiece ready
  C$G}       go 20 steps
    C}       activate welder
    D 1000}  delay 1000 msec (1 sec) to weld
    B}       turn off welder
    L 6,C}   repeat six times from G command
    P 0}     return to home position after 6 welds
    T B}     repeat program until no more pieces
    0}       stop program, return to command mode

  Q          exit program mode

  X}         begin executing program
```

CY525
PROGRAM
BUFFER

4E	N
14	} 20
00	
2B	+
57	W
47	G
43	C
44	D
E8	} 1000
03	
42	B
4C	L
06	6
05	C
50	P
00	} 0
00	
54	T
00	0
00	0

Figure 4.6 Loop Command Use.
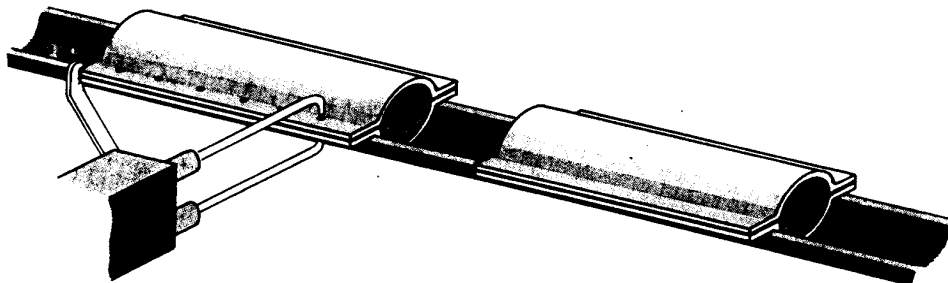
Figure 4.7 Welding Example.

In the welding example, relative mode stepping is used to move the welder from one spot to the next, while absolute mode stepping is used to bring the welder back to the home position, ready to start the next workpiece. The Wait line is used to indicate that a workpiece is in position, and the Dowhile line indicates when there are no more pieces to weld. A real application may be more complex than what is illustrated, but the program indicates the level of problems which the CY525 can solve without help from a host computer. The program uses 20 of the 48 bytes available in the CY525 program buffer.

Note that the use of Labels B$ and C$ allow a branch to be made without the necessity of counting bytes to determine the actual location of the target instruction in the Porgram buffer. The labels can be any of the allowed five label characters in any order. Also note that labels do not require any extra bytes in the program buffer.

TABLE III			OPERATIONAL MODE SUMMARY
MODE DESCRIPTION	MODE 0	MODE 1	MODE SELECTION VIA
Data Type	ASCII-dec	Binary	(pin 36=Hi/Lo (ASCII/Bin)
Position Type	Relative	Absolute	N command selects relative. P command selects absolute.
Step Range	64K max	Unlimited Stepping	defaults to 64K mode. H command selects unlimited.
Gated operation	Triggered	non-trigger	pin 30 low if no triggering. Step on HI-to-LO transition. Inhibit stepping if held HI.
Execution mode	Command	Program	X command executes program. Ø re-enters command mode. (see section on LIVE commands entered while program is executing)
On-the-Fly operation while stepping	Change Rate	Read Position	I/O Req pulled low to change Rate. I/O Sel pulled high to read Position.

5 BINARY DATA MODE OF OPERATION 5

BINARY DATA MODE

To facilitate microprocessor control using binary arithmetic, the CY525 can be placed in the BINARY data mode of command execution by applying a low voltage to pin 36. The possibility of the QUIT command occurring in the binary data necessitates the use of a data count sent after each command byte. In binary mode, the QUIT command, "Q" = 51H, may be inadvertantly transmitted, since some of the binary Position or Rate data may assume this value. For this reason it is **necessary to specify the number of binary data bytes to be sent to the CY525.** In this mode, the data count and data values are specified in binary form, while the command letters retain their equivalent ASCII values.

Commands are issued by first sending the command letter, which has the same value as in ASCII mode. This is followed by a binary value data count. The data count represents the number of data bytes to follow the command byte. If the command is a single letter with no parameter, such as "B", "C", or "H", the data count will be zero, indicating the end of the command. This is similar to sending the command letter and a carriage return in ASCII mode. Note that the data counts are not ASCII characters, they are binary values. Commands with parameters in the range of 1 to 255 will have a data count of binary 1, since these values can all be specified in a single byte. Rate, Slope, etc. listed in Table IV are in this category. The data count is then followed by the single byte which is the binary value desired for that parameter. Commands such as Loop, Number, etc., listed in the table, will have a data count of binary 2, since their parameters cannot be specified in a single byte. The data count is then followed by the two bytes which represent the 16-bit value for the parameter, or the two parameters used by the Loop command. Note that 16-bit values are sent least significant byte first, while the Loop command parameters are sent as count then address, the same order as specified in the ASCII mode. All commands **except QUIT** are of the form:

TABLE IV	ALLOWED BINARY COMMANDS AND DATA COUNT FOR EACH		
COMMAND BYTE	**COUNT**	**DATA BYTE 1**	**DATA BYTE 2**
B,C,X,E G,H,I,U W,+,-,0	0	....	....
F,J,O,R S,T,V,Z	1	Firstrate, Target, etc.	....
A,L,N,P,D	2	Number of Steps,Target Position,etc. Least significant byte first	

42

Note that the QUIT command is not followed by a data count in the Binary mode, just as it is not followed by a carriage return in the ASCII mode. Also, it is possible to load an entire program with a single byte count value. To do this, issue the ENTER command with a data count value of zero, followed by the first command character of the program. Instead of following this character by the normal data count, use a count equal to the remaining total characters of the program, up to, and including the 0 command. Do not include the QUIT command in the count. The Q command should then be issued separately, ending the program entry mode and reverting to command mode. When this method of program loading is used, the 0 command must have a binary value of zero, not the ASCII character "0". The program may also be loaded as separate commands, with a normal data count for each command. The following example illustrates both options for loading a program in Binary mode:

ASCII Command	Binary with Separate Data Counts	Binary with Single Data Count	CY525 Buffer Contents
E)	{ 45..........45		
	{ 00..........00		
R 127)	{ 52..........52..........52		
	{ 01..........0C		
	{ 7F..........7F..........7F		
N 300)	{ 4E..........4E..........4E		
	{ 02		
	{ 2C..........2C..........2C		
	{ 01..........01..........01		
+)	{ 2B..........2B..........2B		
	{ 00		
G)	{ 47..........47..........47		
	{ 00		
N 750)	{ 4E..........4E..........4E		
	{ 02		
	{ EE..........EE..........EE		
	{ 02..........02..........02		
-)	{ 2D..........2D..........2D		
	{ 00		
G)	{ 47..........47..........47		
	{ 00		
0)	{ 30..........00..........00		
	{ 00		
Q	{ 51..........51		

The program buffer in the CY525 will contain the same 13 bytes no matter which byte sequence is used. In ASCII mode, it takes 31 characters to define the program, including the "E" and "Q" commands. In Binary mode, with a separate data count for each command, the program may be defined in 24 bytes. By using a single data count for the program, this number may be further reduced to 17 bytes. Note that the Binary mode values and the program buffer contents are shown as hex numbers.

43

INTERNAL PROGRAM STORAGE

The CY525 program buffer can contain 60 bytes of program commands and data. The Description of Commands contains the length of each command and is summarized in Table V. Note that program parameters set in the command mode do not require any space in the program buffer. If the internal storage is exceeded, the effects on operation will be unpredictable. For optimal operation, the CY525 is treated as a co-processor, with "subroutines" loaded and executed using Interrupt Req #2 (pin 32) to inform the host when a given routine has finished executing.

TABLE V		INSTRUCTION LENGTHS AND PARAMETER CHARACTERISTICS		
COMMAND	**BYTES**	**PARAMETER**	**RANGE**	***TIME μsec**
ABSOLUTE	3	absolute position	0-64k	375+c+v
BITSET	1			375
CLEARBIT	1			375
DELAY	3	msec time delay	1-65535	100+c+v
ENTER	1			400
FIRSTrate	2	start rate	1-127	240+c+v
GO	1			660+v
HALTmode	1			375
INITIALIZE	1			375
JUMP	2	target A-E or	0-59	100+c
LOOP	3	count and address	1-255, 0-59	150+c
NUMBER	3	travel distance	0-65535	180+c
OFFSET	2	drive signal output	0-3,off	80+c
POSITION	3	target location	0-65535	550+c+v
QUIT	*			175
RATE	2	rate parameter	0-127	80+c
SLOPE	2	acceleration	1-255	80+c
TIL	2	branch target A-E or	0-59	380
UNTIL	1			380+v
VERIFY	2	buffer pointer	0-3	125+c
WAIT	1			380+v
EXECUTE	1			380
DIVISOR	2	slope divisor	1-255	180+c
+	1			380
−	1			380
Ø	1			600

*NOTES:
   Command execution times are for command mode. Program mode
      times are much shorter.
   The "L" & "T" commands are normally used only in a program.
   c = ASCII to Binary parameter conversion time.
   v = variable time depending on parameter values.
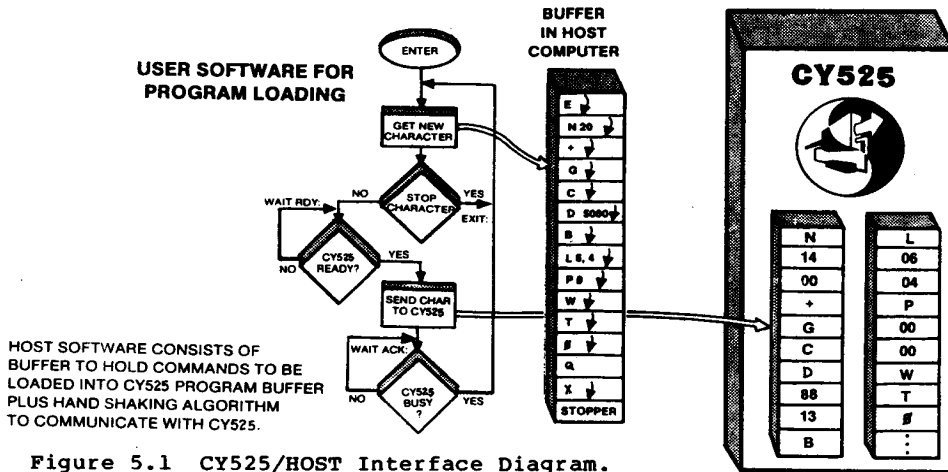
Figure 5.1   CY525/HOST Interface Diagram.

The following illustration shows the internal structure of the
CY525, including the data paths between the various parts of the
device.   Note that all parameters are stored in registers which
are separate from the program and command buffer, so parameters
which do not change may be defined in the Command mode, and
require no space in the program buffer.
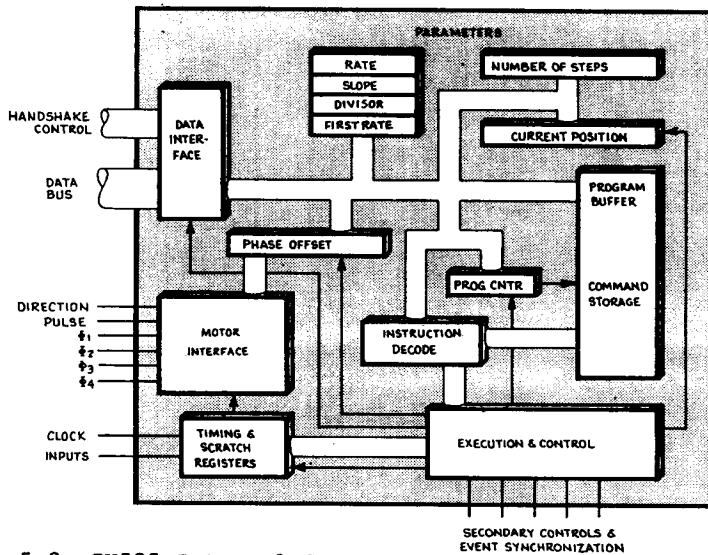
Figure 5.2   CY525 Internal Structure.

INTERFACE EXAMPLE

In the following, it will be assumed that an 8080/8085 transmits data to the CY525 via output port 0DCH. The string of ASCII commands is stored at BUFFER and terminated by a terminal symbol 0FFH. The D-E register pair will be used to access the character string.

```
                    ;
                    ;
0081  11A800  LXI    D,BUFFER
              GETCHAR:
0084  1A      LDAX   D
0085  FEFF    CPI    0FFH
0087  CA9300  JZ     QUIT
008A  13      INX    D
008B  4F00    MOV    C,A
008D  CD9400  CALL   SNDCHAR
0090  C38400  JMP    GETCHAR
                    ;
              QUIT:
0093  CF      RST    EXIT
                    ;
              SNDCHAR:
0094  DBB5    IN     0B5H
0096  E601    ANI    1
0098  CA9400  JZ     SNDCHAR
009B  79      MOV    A,C
009C  E67F    ANI    7FH
009E  D3DC    OUT    0DCH
              GOTIT:
00A0  DBB5    IN     0B5H
00A2  E601    ANI    1
00A4  C2A000  JNZ    GOTIT
00A7  C9      RET
              BUFFER:
                    ;
                    ;
```

ASCII COMMANDS | 8080 BUFFER FOR ASCII MODE | 8080 BUFFER FOR BINARY MODE | RESULTING CY525 PROGRAM BUFFER

E)
Enter Prog Mode — 'E' 45, ')' 0D

R (27)
Set Rate Parameter to 127

F 3)
Set Firstrate of 3

X)
(Loop)

X)
Execute Stored Prog

ASCII	8080 ASCII	8080 Binary
'E'	45	'E' 45 / 00 00
')'	0D	
'R'	52	'R' 52 / 01 01
' '	20	
'1'	31	7F 7F
'2'	32	
'7'	37	
')'	0D	
'F'	46	'F' 46 / 01 01
' '	20	03 03
'3'	33	
')'	0D	
'X'	58	'X' 58 / 00 00
')'	0D	
'X'	58	'X' 58 / 00 00
')'	0D	

RESULTING CY525 PROGRAM BUFFER:
4E, 01, 02, 52, 7F, 46, 03, 58

11 BYTES

22 BYTES

29 BYTES

9 COMMANDS

Figure 5.3   8080/8085 Interfaced to Stepper Motor through CY525 Stepper Motor Controller.

46

6 READ-OUT OPERATIONS 6

VERIFY MODE OPERATION

The Verify mode of operation allows the user to examine the internal register contents of the CY525. This is useful in determining the current state of the CY525, and in verifying parameters before or after critical operations, especially if communications between the CY525 and the host system are enacted in an electrically noisy environment.

The internal contents are divided into four groups, as specified by the parameter in the Verify command. Before reading the contents of the CY525, the user must issue a Verify command to set the internal pointer to the desired group. The contents are then read back one byte at a time, using the sequence described in CY525 Timing and Control Information. As each byte is read out, the internal pointer is advanced to the next byte value, allowing the specified group to be read back by repeated single-byte transfers. Note that I/O SELECT, pin # 39, should be high while the group is being read.

TABLE VI	VERIFY GROUP SUMMARY		
PARAMETER	GROUP	# of BYTES	DESCRIPTION
0	POSITION	2 or 5	current position Binary or ASCII
1	PROGRAM	0 to 48	program buffer contents
2	STATUS	6	pointers and internal flags
3	PARAMETER	6	N, S, R, F, Z parameter values

V 0)

As indicated in Table VI, issuing the Verify command with a parameter value of 0 will set the internal pointer to the current position. The position can then be read back as either a two-byte quantity in Binary mode, or a five-byte quantity in ASCII mode. The mode in which the Verify command is issued will determine the format of the position output. The binary quantity will be presented most significant byte first, and the ASCII quantity will be presented most significant digit first. Internally the position is always maintained in binary, so in ASCII mode, it is converted to the ASCII-decimal equivalent before being output by the CY525. The ASCII position is always a five digit integer quantity, with leading zeroes as required. Note that position is the only quantity converted to ASCII decimal when in ASCII mode. All other Verify outputs are presented in binary, independent of the current command mode. If the current position value is 750, the five ASCII characters would be "00750" in left to right order. The binary mode equivalent would be 02EE, sent as two bytes, first the 02, then EE.

V 1)

With a Verify parameter of 1, the CY525 will output the
contents of the program buffer. The maximum program size is 48
bytes, representing the longest program which may be read back.
The actual number of bytes which have any meaning will depend
on the length of the current program. Output starts with
location 0 of the buffer, the front of the program. Commands
are stored in the program buffer with the format indicated in
Description of Commands except that the 0 command has a binary
value of zero. Note that two-byte parameters are stored least
significant byte first in the program buffer. Several examples
in this manual illustrate the program buffer contents. All
would be read back in the order shown, from the front of the
program through the end. If additional bytes are read back,
they may not have any meaning, since most programs will not use
the entire buffer.

V 2)

The Status group, accessed with a verify parameter of 2,
consists mostly of pointers and flags used by the internal
operations of the CY525. This group is provided mainly for
device testing, and is not expected to be of general interest,
except for one flag, which occurs in the sixth byte of the
group. If the fifth bit (DB4) of the sixth byte is high, the
current direction is set CCW. Direction is CW if the bit is
low. Note that the sense of this bit is opposite that of the
DIRECTION line, pin # 33.

V 3)

The final group accessed by the Verify command is the parameter
group, pointed to when the Verify argument is 3. This is a
six-byte group, consisting of the parameters which may be
specified by the user. The first parameter output is the
number of steps, as specified by the N command. This is a two-
byte value, with the most significant byte output first. Next
is the slope parameter, as specified by the S command. This is
followed by the current rate, set by the R command. The initial
rate, as specified by the F command, and then the slope divisor
specified by the Z command, is output. SLOPE, RATE, FIRSTrate
and DIVISOR are all single-byte values. With N 513), S 25),
R 27), F 3), and Z 1), the bytes read back would be 02, 01, 19,
7F, 03, and 01 (HEX), for N, S, R, F, and Z respectively.

The timing sequence of Figure 7.1, in CY525 Timing and Control
Information, was generated by sending the command "V 1)", and
then reading the first four bytes of the program buffer. The
other groups may be accessed in an identical manner, by
substituting the desired group number in place of the 1 in the V
command. An example subroutine for reading back a desired number
of bytes is shown in Figure 6.1. The routine is written in 8080
Assembly Language. It assumes that the V command has already

been sent. A routine such as the SENDPARALLEL subroutine, shown
in Figure 11.6 could be used to send the desired V command. The
RCVBYTE routine is entered with the B register set to the number
of bytes to read, and the DE register pair pointing to a RAM
buffer which will hold the data.

```
                    ;
                    RCVBYTE:    ;READ CY525 IN VERIFY MODE, V CMD ALREADY SENT
                                ;B = # OF BYTES TO READ, DE = BUFFER POINTER
003B DBED           IN      STATUS
003D E620           ANI     READY       ;LOW IF BUSY
003F CA3B00         JZ      RCVBYTE     ;WAIT FOR READY
                    ;
0042 3E03           MVI     A,IOSELOUT
0044 D3EF           OUT     A4CNTRL     ;I/O SELECT SET HIGH FOR READ BACK
                    ;
                    NEXTCHAR:
0046 3E00           MVI     A,IOREQ
0048 D3EF           OUT     A4CNTRL     ;I/O REQUEST LOW TO REQUEST A BYTE
                    ;
                    WAITDATA:
004A DBED           IN      STATUS
004C E620           ANI     READY       ;LOW WHEN BUSY
004E C24A00         JNZ     WAITDATA    ;BUSY MEANS CY525 HAS OUTPUT A BYTE
                    ;
0051 DBEC           IN      DATA        ;READ THE BYTE
0053 12             STAX    D           ;SAVE IN BUFFER
0054 13             INX     D           ;POINT TO NEXT BUFFER LOCATION
                    ;
0055 3E01           MVI     A,NOIOREQ
0057 D3EF           OUT     A4CNTRL     ;I/O REQUEST HIGH TO ACK BYTE RECEIVED
                    ;
                    WAITCLR:
0059 DBED           IN      STATUS
005B E620           ANI     READY
005D CA5900         JZ      WAITCLR     ;WAIT FOR READY AGAIN
                    ;
0060 05             DCR     B           ;CHAR COUNT
0061 C24600         JNZ     NEXTCHAR    ;MORE BYTES TO READ IF NOT ZERO
                    ;
0064 3E02           MVI     A,IOSELIN
0066 D3EF           OUT     A4CNTRL     ;I/O SELECT SET LOW FOR NEXT COMMAND
0068 C9             RET
                    ;
                    ;
```

Figure 6.1 Verify mode read subroutine.

Reading Position on the Fly

The Verify instruction can be used in the command mode (or as a live command) to find out the current position of the CY525. This is especially useful if a sequence of "relative" moves have been executed (since for "absolute" moves, the current position should be the most recently specified target!) However, in many cases it is desirable to know the position of the motor during the stepping operation. The CY525 offers this capability, although, unlike the VERIFY query, some external circuitry is required (unless a truly fast computer is available). Using two latches and some NOR gates, the current position can be latched and read as follows:

1. The I/O SEL line (pin 39) is brought high while the CY525 is stepping. This requests the READ-on-the-fly operation.

2. The OUTSTROBE (pin 10) is pulsed with the high position on the data bus.

3. The INTERRUPT REQUEST (pin 37) goes low to signal that the high position is on the data bus, and remains low.

4. The OUTSTROBE is again pulsed to signal that the low position byte is on the data bus and can be latched by the trailing edge of the pulse.

5. The INTERRUPT REQUEST (pin 37) goes high to complete the cycle.

6. The I/O SEL should be returned low within approximately 50 microseconds after the interrupt request goes high.

7. The 16-bit position can be read from the latches.

8. If the I/O SEL is high, the CY525 will output the new position with each step.
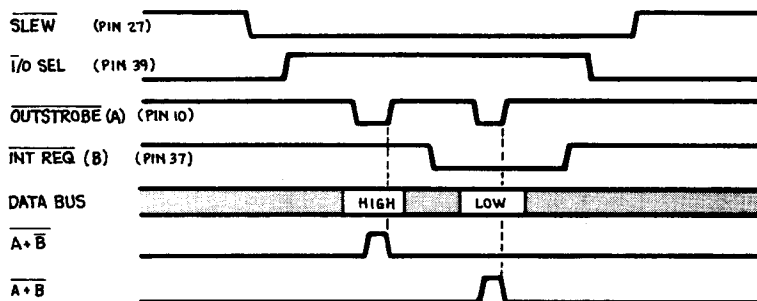
Figure 6.2 Timing Sequence for Reading Position on the fly.

Circuit to Latch Position On-the-Fly

An example circuit using popular 74LS373 8-bit latches is shown
in the following figure.
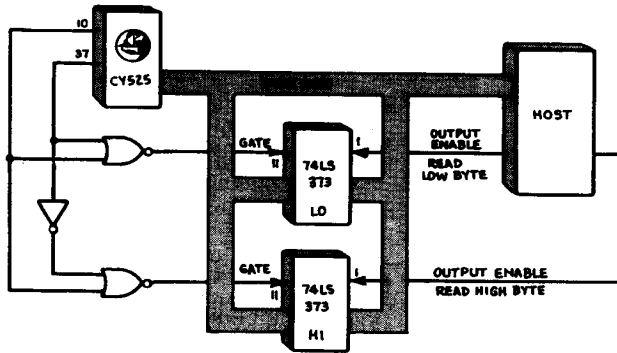
Figure 6.3   Typical circuit for latching position on the fly.

In the example circuit, the position is latched into the LS373s
using the high going pulse derived from signals appearing on
CY525 pins 10 and 37 as shown in the previous figure. After the
data is latched, the CY525 returns its bus to the high impedance
state and the latches can be read back onto the same data bus and
into your computer. This can be done by enabling the LS373
outputs using low going signals generated by your computer. The
sequence is shown below:
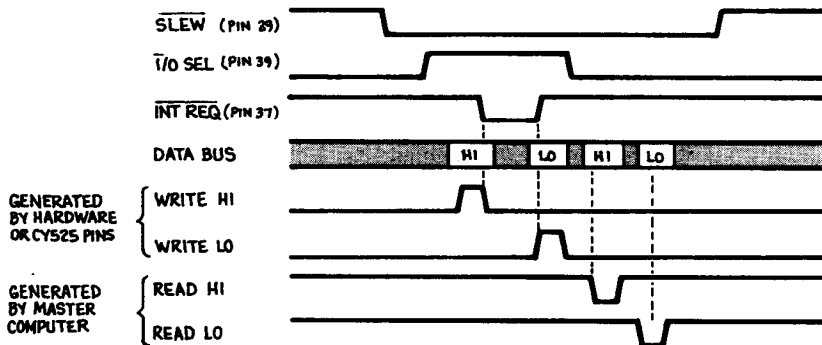
Figure 6.4   Waveforms for circuit shown in figure 6.3.

The exact timing for a CY525 with and 11 MHz crystal is shown
below:
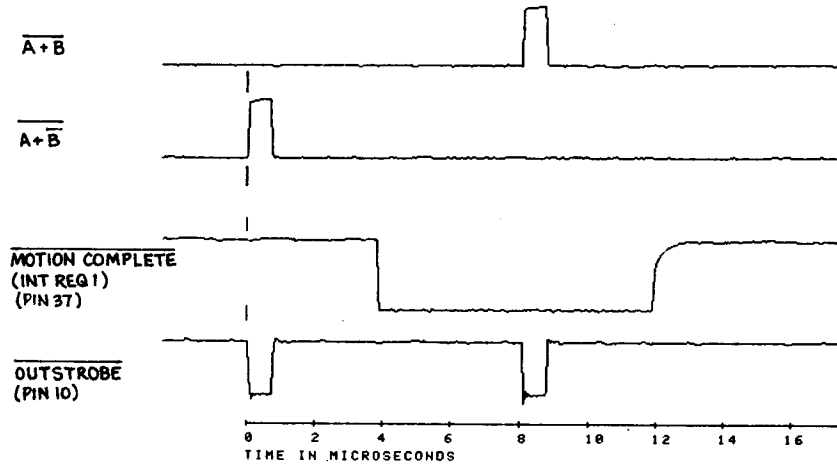
Figure 6.5   Timing for position readout (with 11 MHz crystal).

A typical data bus line is shown below. In this case the line is
initially in high impedance (A), then the line goes high and
presents a stable "1" when OUTSTROBE (pin 10) rises (B).   The
line remains in this state until going low (C) for the second
position byte.   The low is held until latched by the trailing
edge of the second OUTSTROBE, and then the bus returns to tri-
state (D).
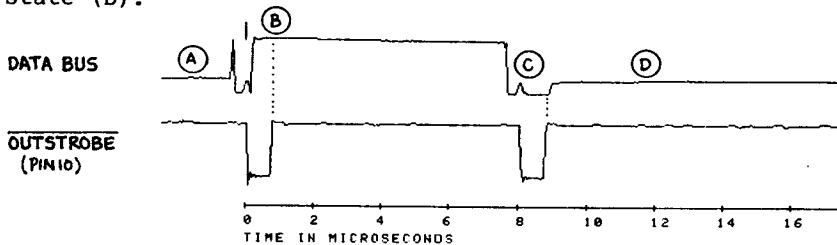
Figure 6.6   Typical data bus line during position readout (11MHz).

Important note: If the position is read while the CY525 is
SLEWING, (i.e., while pin #29 is low), there is no effect on the
step period, that is, the READ operation is transparent and has
no effect on the motor. If the readout occurs while the CY525 is
RAMPING, then the READ operation will increase the step period by
20 microseconds (with an 11 MHz clock). This is generally of no
consequence, however, the effect is maximum during the few steps
at the top of the ramp, that is just before or just after the
motor is slewing.

Latch Control Signals

The circuit shown in figure 6.3 uses two 74LS373 latches to capture the 16-bit position that the CY525 produces in response to raising the I/O SEL line (pin 39). Figure 6.7 below shows the timing for a typical "read on the fly" operation. These signals appear on the pins of the CY525. The READ request is indicated by raising pin 39 and the write pulses appear on the OUTSTROBE pin (10). The signal on pin 37 distinguishes between the high and low byte of the 16-bit location.
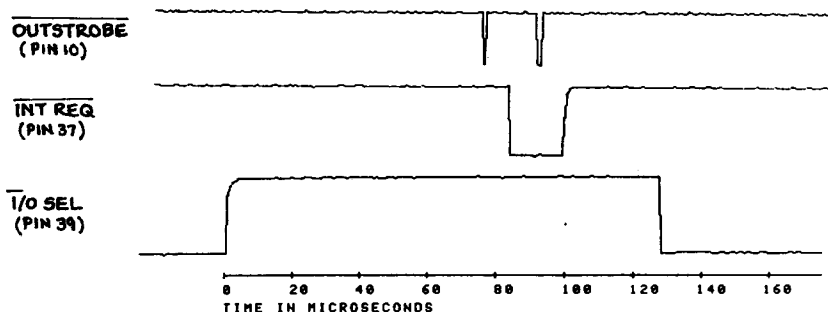
Figure 6.7 CY525 signals for READ on the fly. (11 MHz).

The next figure illustrates typical timing for the latch control signals that enable the inputs and the outputs of the 74LS373s. The first two positive going strobes latch the data into the 373 and are derived from the signals on pins 10 and 37 as shown in figure 6.3. The last two low going pulses enable the output of the 74LS373s and are generated by the master computer. (11 MHz).
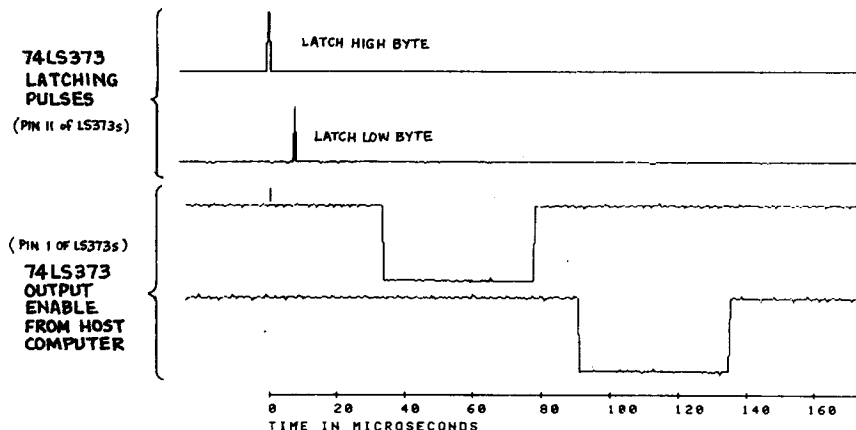
Figure 6.8 Latch control signals for circuit of figure 6.3

53

7 TIMING AND CONTROL INFORMATION 7

CY525 HANDSHAKE TIMING INFORMATION

With the parallel interface to the CY525, the user must wait for
the CY525 BUSY/RDY line (pin #27) to be high before applying I/O
REQUEST strobe to pin #1. Note that no data set-up time is
required, so the data may appear on the data bus at the same time
that the I/O REQUEST write strobe goes low. This is especially
convenient in the ASCII mode as bit 7 of the ASCII data byte can
be used to generate the write strobe (figure 10.8). The data is
read into the CY525 from the data bus by a low going read strobe,
INSTROBE, appearing on pin #8. The data should be valid at the
trailing edge of INSTROBE. INSTROBE may be used to enable the
data onto the data bus from an external device. The data may be
removed at any time following the occurrence of INSTROBE, however
**the I/O REQUEST line should be held low until BUSY/ RDY acknow-
ledges the transfer by going low.** The simplest interface ignores
INSTROBE and uses BUSY/RDY only (Figure 2.1). I/O SELECT must be
low while commands are being sent to the CY525.

Timing for the Verify mode, in which the internal contents of the
CY525 may be examined, is similar to that described above. In
order to read the internal contents, the I/O SELECT line must be
high. This will put the CY525 in an output mode. When the
BUSY/RDY line is high (ready), the user should strobe I/O REQUEST
(pin # 1) low. The CY525 will then write the next byte value
onto the data bus. This is indicated by a low going write
strobe, OUTSTROBE, appearing on pin #10. The data will be
latched and valid on the trailing edge of OUTSTROBE, which may be
used to latch the byte into the user's input port. After
OUTSTROBE, the CY525 will go busy, indicating that the data is
available on the bus. Data will remain valid until I/O REQUEST
is again set high by the user. Note that the user may read the
data directly, after the CY525 goes busy, before raising I/O
REQUEST. When the CY525 detects I/O REQUEST high, the data bus
will be put back into a high impedance state, and data will no
longer be valid. This operation will be indicated by a second
OUTSTROBE pulse. The CY525 will then indicate ready again,
awaiting the next command or another verify read, as indicated by
the I/O SELECT line. See Figure 7.1 for the waveforms.

To enable all CY525 features, the user should connect all 8 lines
of the data bus to his I/O ports, and generate I/O REQUEST and
I/O SELECT as separate lines. I/O SELECT should be changed only
while the CY525 is ready (BUSY/RDY is high), and may be used to
determine the direction of data on the data bus (low=into CY525,
high=out from CY525). Since the data bus is bidirectional, the
user must turn off (tri-state) the command output port during the
Verify mode, allowing the CY525 to drive the data bus lines.
This may be done by I/O SELECT, or the command port could be tri-
state at all times except during the INSTROBE pulse. The typical
for a Verify command and response is shown in Figure 7.1 below.
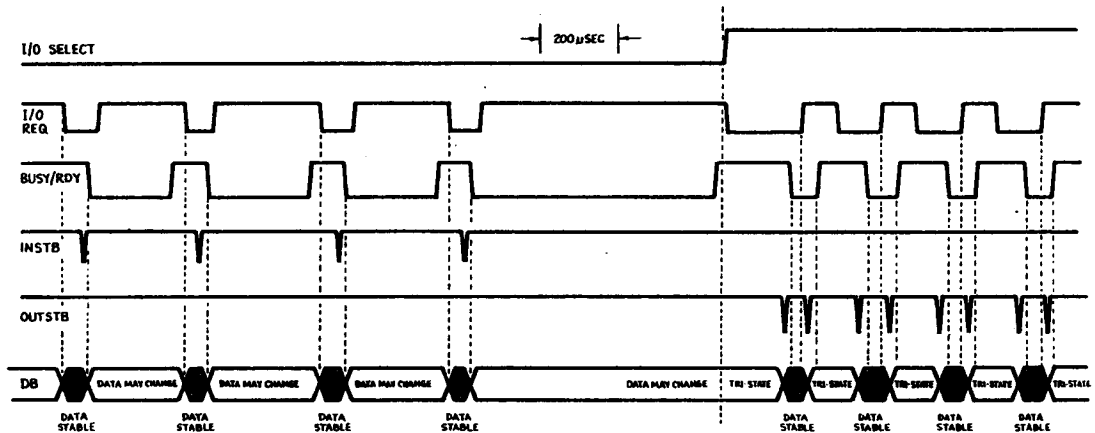
Figure 7.1 CY525 Verify Command Timing Sequence using 6 MHz xtal

CY525 Interface Timing

The CY525 can receive direct commands or execute a sequence of commands stored internally in an on-chip program buffer. The following example's illustrate typical timing for various actions related to command execution. Specifically, we will consider the entry and execution of a very short program. We will send the following commands to the CY525:

E)	; Initiate the Program-Entry mode
	; (Bit set)...drive pin 34 to TTL high
	; (Clear bit)...drive pin 34 to TTL low
	; Stop executing program, re-enter command mode
Q	; Quit entering the above command mode
X)	; eXecute the stored program

Each of the above commands is given to the CY525 by testing the BUSY/RDY line (pin 27) of the CY525. When this pin is high (RDY) the command is placed on the data bus and the I/O REQ (pin 1) of the CY525 is pulled low to signal the presence of data in the bus. When the CY525 accepts this data, the BUSY line is pulled low to acknowledge the transfer. At this point the data may be

removed from the bus and the (active low) I/O REQ signal returned high. After the CY525 has processed the data, the RDY line will return high. The RDY line will not return high until the I/O REQ has been removed. The sequence of handshakes for the entire program shown above is depicted in figure 7.2 below.
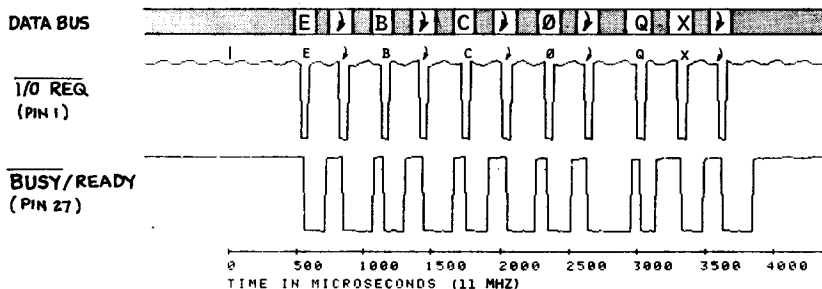
Figure 7.2. Handshake timing for example program.

Note that above each I/O REQ strobe, the ASCII character being transferred is shown on the bus. There are eleven characters transferred to the CY525 in approximately 3300 microseconds. As may be seen the spacing is very regular, therefore the transfer time per character is about 300 microseconds. These times are for an 11 MHz crystal and should be scaled by f/11 for a crystal of frequency f. If we expand the timing diagram above and look at the first transfer only, we see that the CY525 actually completes the transfer of the character (E) and returns to the READY state within 200 microseconds. The extra 100 microseconds before the I/O REQ is pulled low (to signal the ⟩ on the bus) is actually delay caused by the master computer used in this example. If a faster computer were used, this could be decreased so that the minimum character transfer time of 200 micro-seconds per character is approached. This timing is shown below in figure 7.3.
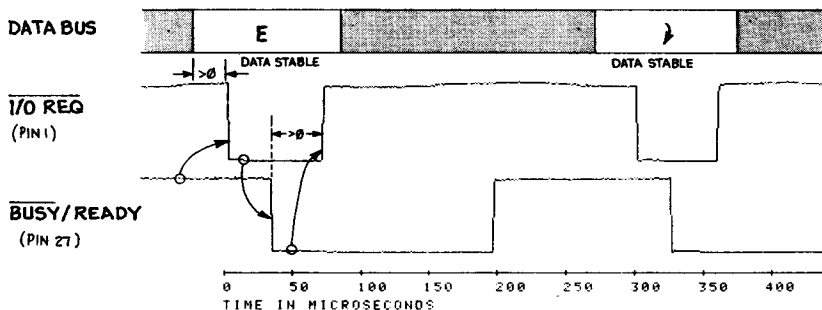
Figure 7.3 I/O REQ and BUSY/RDY timing diagram.

As discussed elsewhere, if ASCII numbers are entered, there is an additional processing time after the carriage return while the decimal-to-binary conversion occurs before the BUSY pin returns high (Ready). This time is absent in the binary data mode.

Timing of Secondary Signals

The handshake signals I/O REQ and BUSY/RDY are considered primary signals and must be observed. There are several secondary signals that provide convenience but are not strictly necessary. For example the PROG (pin 31) will go low upon receipt of the E⟩ command to indicate that the next commands will be ENTERed and stored in the program buffer instead of being executed as they are entered. The Q command (with no carriage return) QUITs the program-entry mode and returns the PROG line high. This is shown in figure 7.4 below:
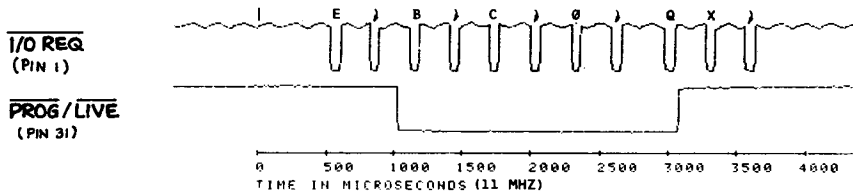
Figure 7.4   Illustrating PROG line timing.

To examine the execution time for this program, we expand the portion of the timing diagram that follows the Q command, which terminates program entry. The X⟩ command which immediately follows the Q begins the program execution. The timing diagram shown in Figure 7.5 below presents the Q, X and ⟩ transfers. As discussed above, the Q causes the PROG line to return high. The X⟩ command causes the program to begin executing as signaled by the RUN line (pin 32) going low. As seen below, this occurs apparently one hundred fifty microseconds after receipt of the X.
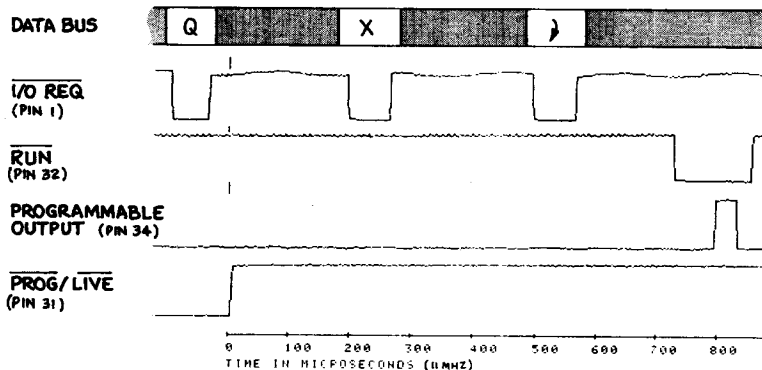
Figure 7.5   Timing for PROG, RUN, and PROGRAMMABLE OUTPUT.

57

The short program consists of the commands **B**, **C**, and **0**. The **B** causes the PROGRAMMABLE OUTPUT (pin 34) line to go high and the **C** causes it to return low. The **0** command terminates program execution and causes the RUN line to return high. The CY525 is now back in the command mode. The program still resides in the CY525 on-chip program buffer and can be executed an indefinite number of times by sending a succession of **X** commands. A new program can be entered by sending another **B** command. The last portion of the above diagram is expanded and shown below. It can be seen that the **B** command causes the control pins to go high with 60 microseconds of the RUN lines going low, however the **C** command returns control low with 40 microseconds and the **0** command raises the RUN lines to terminate program execution in less than 40 μsec.
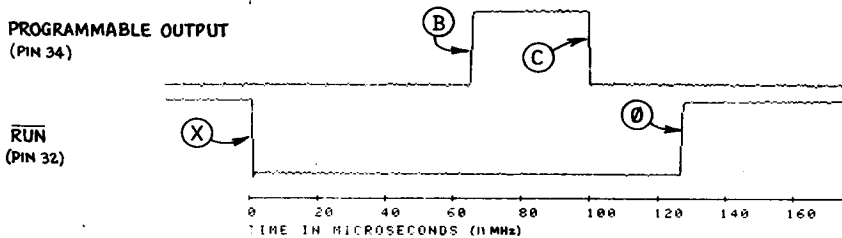
Figure 7.6 Timing for program execution of sample program.

Timing for Command Execution

The **B** and **C** instructions executed in the example program above can also be executed directly in the command mode. The timing for these commands is shown below in figure 7.7.
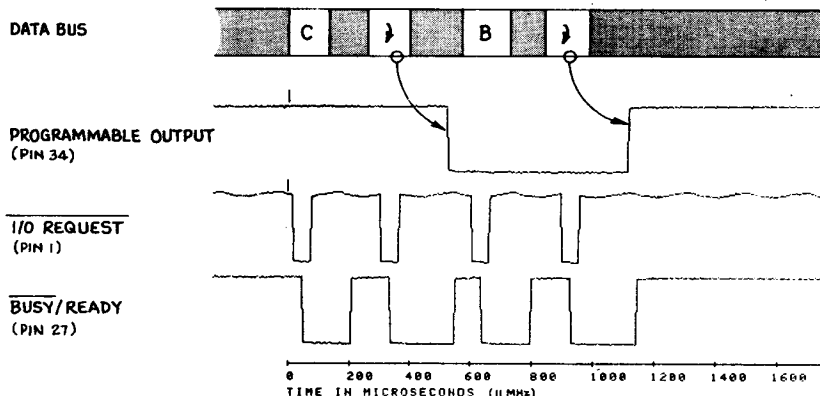
Figure 7.7. Timing for **B** and **C** command execution.

STEP INHIBIT PIN

In the triggered mode of operation, the GO command initiates the stepping sequence if the Step Inhibit pin is low. If the Step Inhibit pin (pin #30) is high, the controller simply waits for a low level on this pin and then takes the step. The level of the pin is tested during the time out loop of every step.
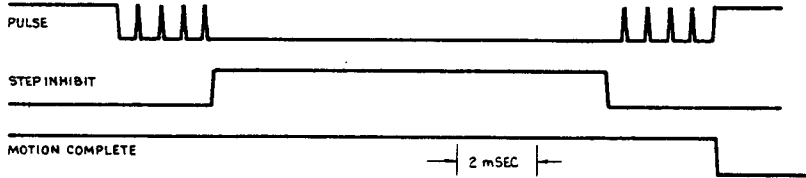
Figure 7.8 Step Inhibit Timing with 6 MHz xtal.

DIRECTION CONTROL

In the Absolute Mode of operation, in which target positions are specified with the "P" command, direction is determined automatically. If the specified position is greater than or equal to the current position, the direction is set to clockwise (CW), and if the specified position is less than the current position, the direction is set to counter-clockwise (CCW). The current setting of direction is indicated by the DIRECTION pin (pin # 33). HIGH corresponds to CW, and LOW corresponds to CCW. In Absolute Mode, the DIRECTION pin is set just before stepping.

In the Relative Mode of operation, in which the number of steps to take from the current position is specified with the "N" command, and stepping is activated with the "G" command, the user may control direction with the "+" (CW) and "-" (CCW) commands. The current setting of direction is still indicated by the DIRECTION pin, and stepping will occur in the direction last set when the "G" command is issued. The system powers up in the clockwise direction.   Note that the direction commands are separate commands; they are terminated by the carriage return character,)=0DH. Thus, to specify 100 steps in the counter clockwise direction it is necessary to send two commands:
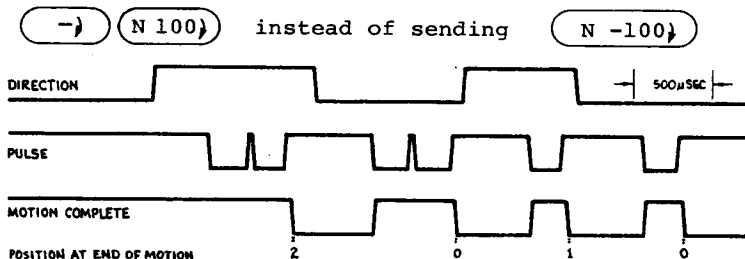
Figure 7.9 Direction indication. Program: P 2) P 0) +) G) -) G) using a 6 MHz xtal.

LIVE COMMANDS
(While a Program is Executing)

The CY525 has two features that allow control while the CY525 is:

        1.   executing a program
        2.   generating step signals

These are quite different operations. The ability to change step rates on the fly, that is, while the motor is stepping, is discussed in another section. This section discusses live commands, that is, commands that are sent to the CY525 while a program is executing, as indicated by the RUN line (pin 32) in a low state.

If it is desired to send commands to the CY525 while a program is running, the PROG/LIVE line (pin 31) must be pulled low. This informs the CY525 that the master computer has a command for it. After the CY525 finishes executing the current instruction in a program, it will enter a mode in which any valid commands will be accepted and executed. For example, the PROGRAMMABLE OUTPUT line (pin 34) may be set or reset (via B or C) or any parameter may be changed. Multiple commands may be issued, as long as PROG/LIVE is held low.

To exit the live mode, the PROG/LIVE line is returned high after the last command has been entered, but before the last carriage return code (0DH) has been sent. This is illustrated in the following figure for three commands.
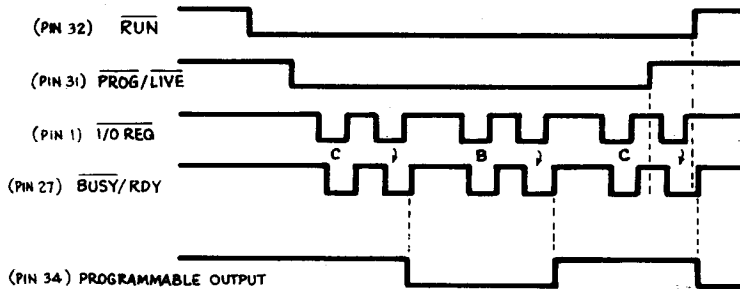
Figure 7.10   Illustrating live commands sent while the CY525 is executing a program.

It should be understood that the BUSY/READY line (pin 27) is to be observed as usual. In the CY525, the READY line is pulled low for every motion command (P or G) and returned high at the end of motion. The I/O REQ line (pin 1) should not be brought low until the READY line is high. If the I/O REQ is made low before the motion is completed, it may be interpreted as a request to change

60

the step rate on the fly ! Also, since PROG/LIVE (pin 31) is bi-directional, the CY525 will drive PROG/LIVE low during program entry mode, so the user should only drive this pin with an open collector gate or equivalent.

Once the CY525 sees the PROG/LIVE line is low, you must issue one command before program execution will continue. The PROG/LIVE line is tested between commands and if low, the CY525 will wait for a command input. After the command is executed, the line is tested for another live command. If the line has returned high, the program will continue, otherwise the CY525 waits for another command. Note that the 0 command will stop program execution, which brings the RUN line high and goes back to end mode.

CONTINUOUS STEP (HALT Mode)

The CY525 has a provision for unlimited stepping (unlike the CY512 which is limited to 64K). This mode is enabled via the H instuction. H stands for Halt mode, since the CY525 will step until Halted by the ABORT line (pin 6). The H command does not initiate stepping. It merely sets an internal flag that is tested during execution of a motion command. Note also that the H instruction must be issued for every continuous stepping command, since the Halt mode flag is reset when the continuous motion is aborted. The following example illustrates this point.

```
N 1000)    set number of steps = 1000
G)         take 1000 steps and stop
H)         set continuous step mode
G)         ramp up and step until halted by the ABORT
G)         take 1000 steps and stop
```

Note the ABORT line is used to HALT the continuous stepping that occurs when a GO command (G) is preceded by the Halt mode command (H). The ABORT line should be brought low and held low until the MOTION COMPLETE signal is pulled low (pin 37).

If the ABORT line is returned high before the MOTION COMPLETE signal goes low, then unpredictable results will occur. The CY525 in the non-continuous mode, will continue stepping at the First-rate until the target position is reached. In the continuous mode, the internal bookkeeping may be scrambled, and the target may be anywhere.

To summarize:

Normal motions of the CY525 terminate at the specified target position (or after N steps).

Continuous step operation terminates when the ABORT line is pulled low and held low until the MOTION COMPLETE signal (pin 37) goes low. There is no target position in this mode.

WRAP AROUND POSITION

When the CY525 is placed in continuous step operation, the internal position counter is not disabled, but continues to keep track of the position (modulo 64K). By reading the position on the fly, a master computer can readily keep track of the position exactly. Note that at 10,000 steps/sec it takes approximately 6 seconds for the position to wrap around, thus providing plenty of time for a master computer to monitor (by sampling) the state of the CY525. There is no external indication of the occurrence of wrap around, so the monitoring computer must perform simple arithmetic tests to detect it.

ABORT OPERATIONS

The CY525 uses the ABORT line to cause the motor to ramp down from its current rate to the Firstrate (F) and either stop or continue to the target, depending on the condition of the ABORT line at the bottom of the ramp. Additional abort pulses will be ignored until motion is completed. Several cases are described in the following. Only Case III should be used in the continuous step mode.

Abort Line Descriptions

Case I.   Normal Motion

In this case the Abort line is left high (+5V) the entire time and the CY525 ramps up from the specified starting rate, slews at the specified maximum rate, and ramps down to stop at the specified final location. The slew line (pin 29) is used to distinguish accelerated motion (up & down) from constant velocity (slewing).
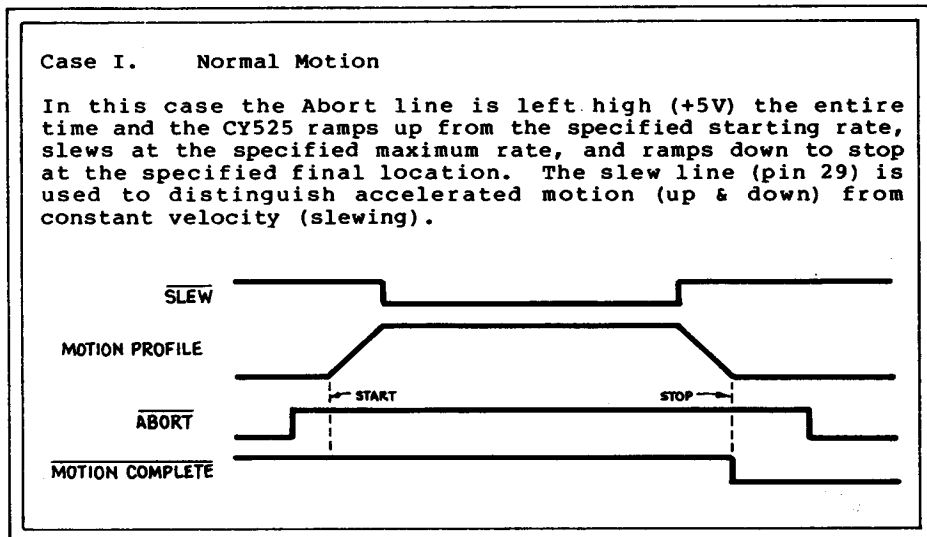
Case II.   ABORT disabled

In the special case that the abort line is initially low
(0v), the ABORT action is disabled for the duration of the
normal motion. Note that stepping can be inhibited via pin
30, but cannot be aborted. This case is provided for those
people who use a limit switch to abort motion. Without this
case, you could never move away from the limit switch.
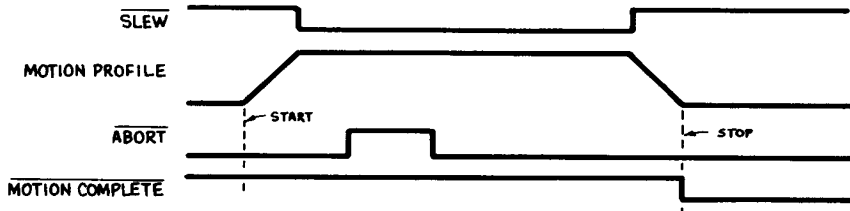
Case III.   ABORT and STOP

This case illustrates the ability to abort the CY525 motion
with the normal deceleration to the starting rate (Firstrate
= F) and then, upon finding the ABORT line still low, STOP
and signal motion complete. The Abort line can be brought
high any time following the Motion Complete signal.
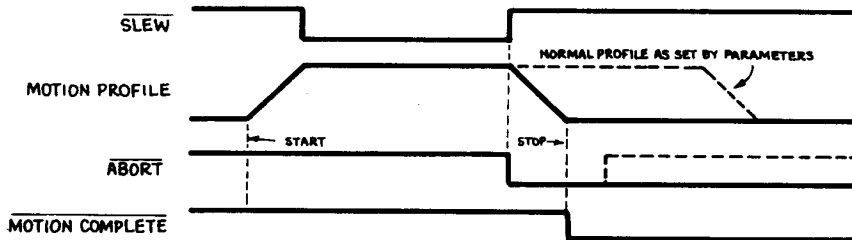
Case IV.   ABORT and go to TARGET

In this case, the Abort line is brought low to terminate
slewing but is removed before the motion reaches the initial
step rate F (set by FIRSTRATE). Upon reaching the starting
rate and finding the Abort line high, the CY525 continues
moving at the initial step rate until it reaches the
specified target location and then stops and issues the
Motion Complete signal.

SLEW

MOTION PROFILE

START   STOP

ABORT

MOTION COMPLETE
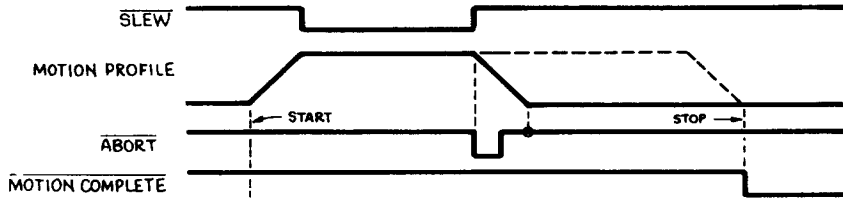
Case V.   ABORT while RAMPING

This case is included to show that the normal motion can be
aborted before the slew rate is reached, that is, while the
CY525 is still accelerating the motor. Depending on whether
ABORT is returned high or not, the motion will stop at the
bottom of the down ramp or proceed at the slow rate to the
target position.

SLEW

MOTION PROFILE

START   STOP

ABORT

MOTION COMPLETE
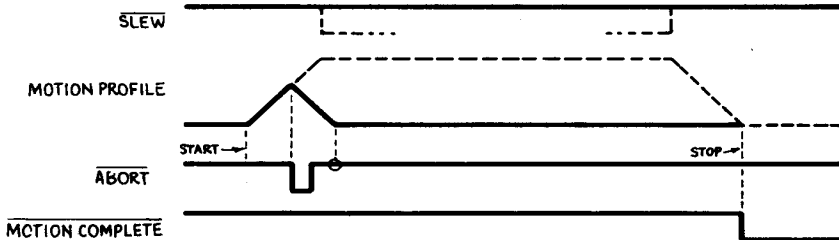
Case VI.   Precautions about ABORTing Continuous Motion

When continuous run mode is selected (via H), a large number
of steps should be entered, for example N 64000).   Any
operation occurring within this range will abort normally as
shown in the CASE examples.   Note however, that if the
"target position" is detected while the CY525 is ramping down
due to an ABORT, the CY525 will abruptly stop "on target"!
For extremely long continuous motion, this behavior will
occur modulo 64K.

an example is shown below:
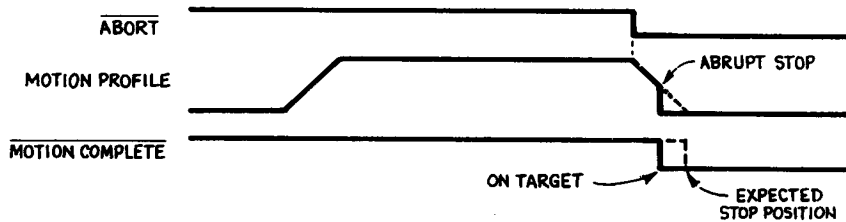
If an application is expected to encounter this situation,
and if an abrupt stop cannot be tolerated, the following
procedure should be used:

Instead of issuing an ABORT signal, change the rate on-the-
fly to the desired minimum step rate. The CY525 will begin
ramping down as if an ABORT had been issued. When the SLEW
line goes low at the bottom of the ramp, issue the ABORT.
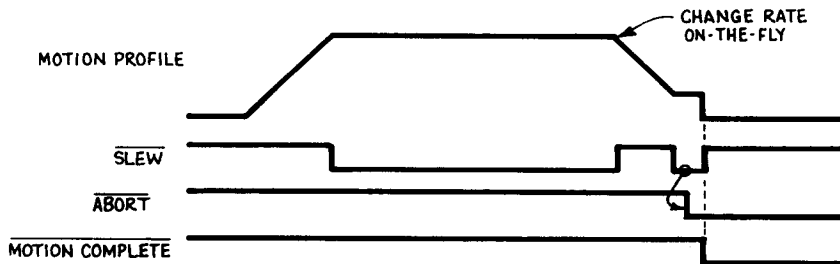
ABORT RELATED SIGNALS

The ABORT line (pin 6) is a level sensitive line that causes the CY525 to begin down ramping when it detects a low level on this pin.   The level sensitive nature of this pin is shown in the following figure.
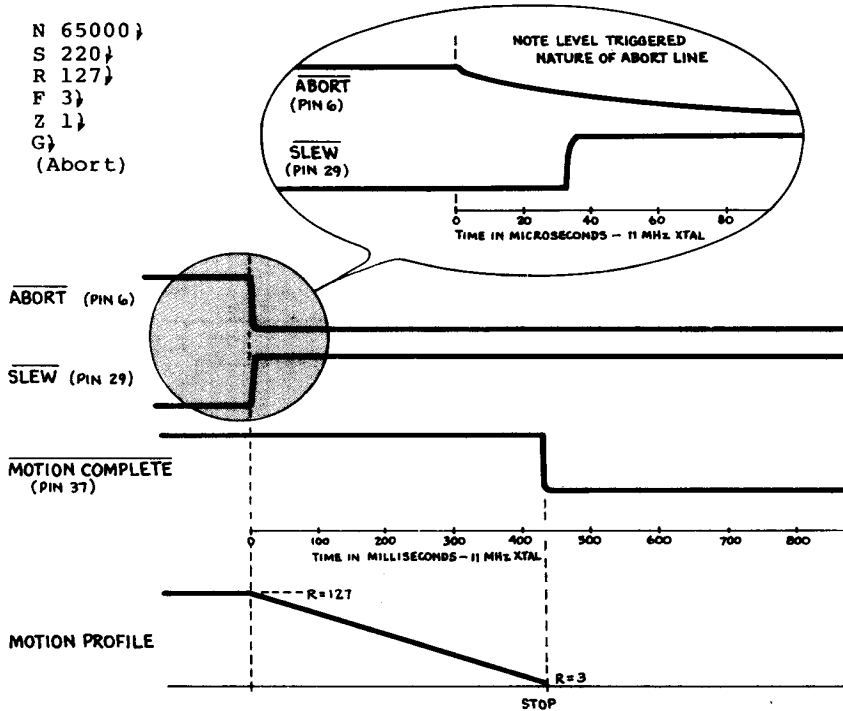
Figure 7.11 The level sensitive ABORT line can be used to trigger down ramping of the CY525.

THE SLEW LINE

The SLEW line (pin 29) goes low to indicate that the CY525 is stepping at a steady rate. Thus it is high during both up ramp and down ramp excursions. The normal mode of operation (as in continuous stepping) causes the SLEW line to go low when the maximum step rate is reached. It returns high when the CY525 begins ramping down. This ramp down can result from three different causes:

> 1. normal deceleration to the target position.
> 2. the ABORT line has been pulled low.
> 3. the step rate has been changed on the fly.

The abort operation can occur in either normal operation or in the continuous stepping mode selected by the H command. The last case should occur only in the continuous step mode of operation.

In normal operation, the SLEW line is low only while the CY525 is stepping at the maximum rate. Note that if this motion is aborted (via pin 6 going low), the device will ramp down to the First-rate (specified by the F command) and if the ABORT line has been returned high, will step at this steady rate until the target position is reached. Even though the device is stepping at a steady rate, the SLEW line will remain high. To repeat, for normal stepping, the SLEW line indicates stepping at the maximum rate. This operation is shown in the figure below:
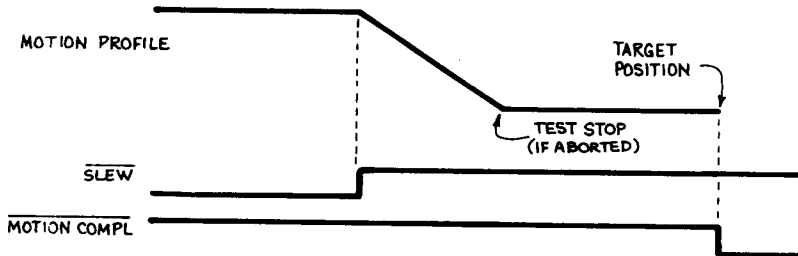
Figure 7.12 SLEW line behavior during abort in normal mode.

Slew Operation in Continuous Step Mode

The meaning of slew is more general in the continuous step mode of operation. In this mode, SLEW going low indicates that the CY525 has completed ramping and is moving at a steady rate, not necessarily the maximum rate! Thus, when the rate is changed on the fly, the CY525 raises the slew line while accelerating or decelerating to the new rate, then lowers it when the new rate is reached. This is shown in the following figure with 11 MHz xtal.
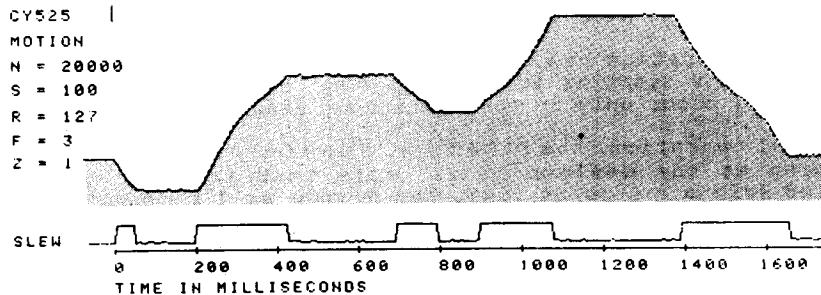
Figure 7.13 Illustrating the operation of SLEW in continuous step mode when rate changes are executed on the fly.

RATE CHANGES ON THE FLY

The CY525 offers a major new feature -- the ability to specify rate changes on the fly, that is, while the motor is stepping! This allows much more complex motion profiles and much closer control of motor movement. In order to make use of this feature, the CY525 should be placed in the continuous step mode of operation by preceding the G command with the H command (H stands for Halt mode, since in this mode stepping must be halted via the ABORT line (pin 6) as opposed to the normal mode of operation in which stepping halts at the target position. There is no target position in the continuous step mode.)

In order to change the step rate on the fly, two conditions should be met:

1. continuous step mode should be invoked via **H**.

2. the SLEW line should be low, indicating steady stepping (not ramping).

Once these condition are met, the step rate can be changed as
follows:

1. place the new 8-bit binary rate on the CY525 bus.
2. lower the I/O REQ line (pin 1) to the CY525.
3. raise the I/O REQ when the SLEW line goes high.

The CY525 will now ramp up or down as appropriate until the new
rate is reached, then continues stepping at this rate until
changed again or aborted.

Rate Change Timing

When the CY525 is slewing in the continuous step mode, a new rate
can be entered as described above. Although this entry can be
synchronized with the pulse line (pin 35), this is unnecessary
and does not offer any advantage. The normal method of changing
rates while slewing consists of simply placing the new (binary)
rate parameter on the CY525 data bus and asynchronously pulling
the I/O REQ line low. Since the I/O REQ line is checked only
once per step, there is an indeterminate delay from the time
I/O REQ is lowered until the SLEW line goes high to acknowledge
it. This is illustrated in the following figure. At the left of
the figure, the CY525 is slewing at a low rate, therefore each
step is longer and the average response time to I/O REQ is
accordingly longer. This is observed by the initial wider I/O
REQ pulse that is held until SLEW goes high. The CY525 then
ramps up to the new high step rate and lowers the SLEW line when
it reaches the new rate. We now place a low rate on the bus and
lower I/O REQ again. Since the step period is much shorter at
the high rate than it was at the low rate, the CY525 detects the
I/O REQ much sooner (on the average) and responds by raising SLEW
and ramping down to the new low rate.

Note that since I/O REQ is lowered asynchronously, these are
typical responses. Of course, if the I/O REQ happens to occur
just before it is tested, then the response time would be short
even at low rates.
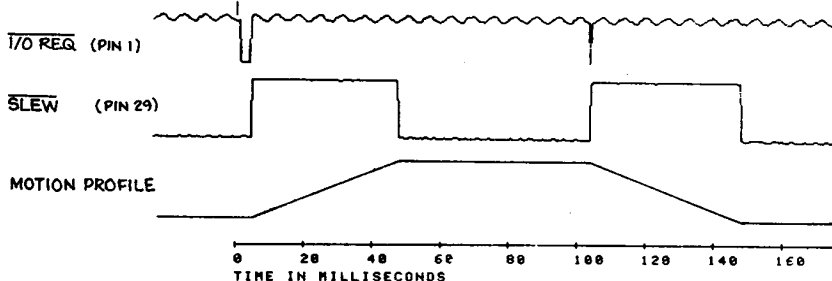
Figure 7.14 Illustrating typical timing for changing rates on the
fly using an 11 MHz xtal.

69

Rate Change Examples

The figure below shows several motion profiles obtained by changing the rate on the fly. It can be seen that the acceleration curve between any two specified rates is simply the section of curve that would be found between those two rates when the CY525 accelerates from minimum to maximum velocity (and the corresponding deceleration segment.) Note the behavior of the SLEW line.
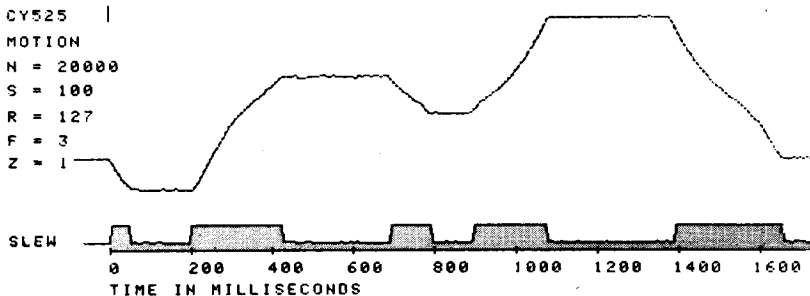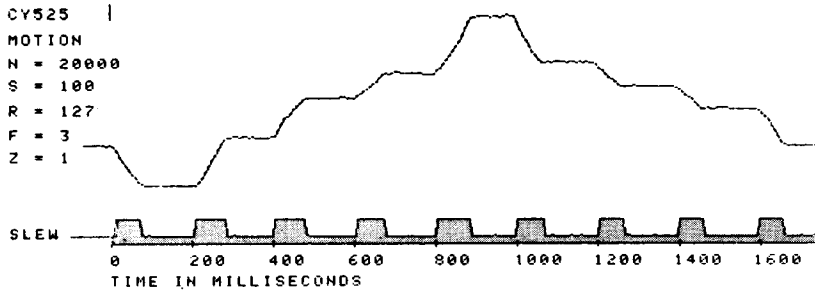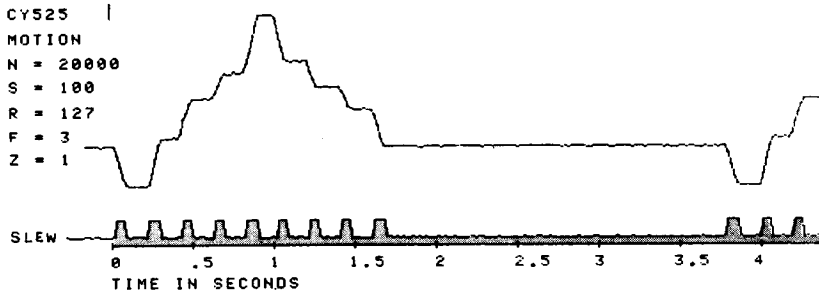
Figure 7.15   Typical profiles for "on the fly" changes.

70

Typical Response Times

If the CY525 is stepping with R = 3 (300 steps/sec @ 11MHz) then the step period is 3333 microseconds and the average response will be about one half of this, i.e., 1600 microseconds. If the rate is R = 127 (9675 steps/sec @ 11MHz) then the step period is 103 microseconds and the average response time will be 50 microseconds. The two scope traces shown below typify this timing. The I/O REQ goes low to signal that the new rate is on the bus. The SLEW line goes high to indicate that the CY525 has begun accelerating or decelerating to the new rate. (In any case, the maximum response time to the I/O REQ line is the current step period.)
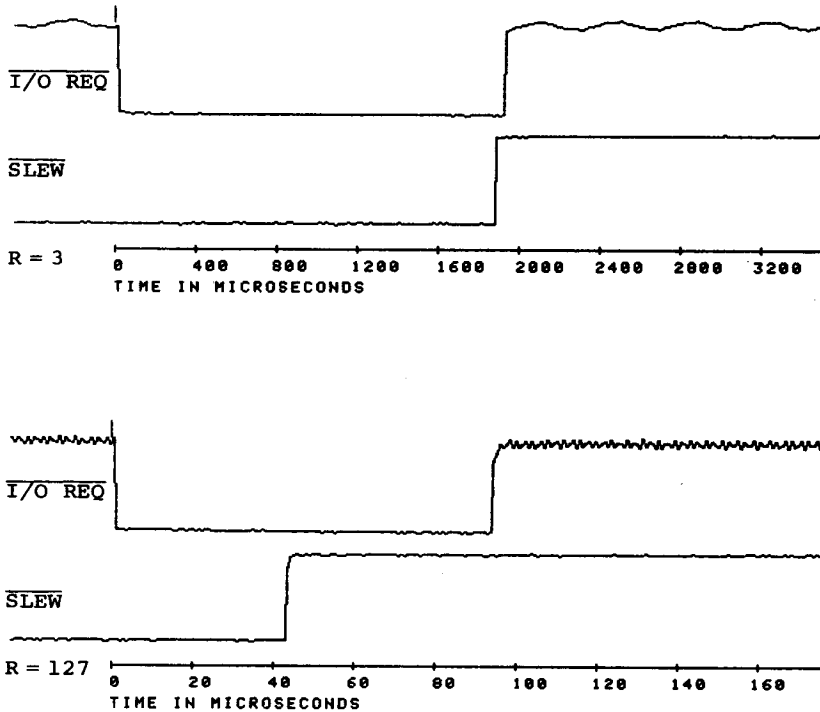
Figure 7.16   Typical response times for changing the rate.

SPECIAL CONSIDERATIONS

As mentioned previously, the CY525 should be placed in the continuous step mode (via the H command) in order to change the step rate while the motor is stepping. If this is not done, unpredictable behavior will occur, including the possibility of an abrupt halt from any step rate.

If the Continuous run mode is to be used, then the Number-of-Steps command, N, should be used with a large argument, say 64K or so. This is to ensure that the CY525 begins slewing. If a very large Z factor is used, it is possible that the CY525 will not reach the slew state in 32K steps, in which case, the CY525 will not enter the Continuous run mode. This is a very unlikely case, and one which can be avoided by using N 64000} and relatively small values of Z, say < 10. The exact value of Z depends on the value of the slope parameter, S. Although this limitation is very unlikey to occur in most applications, you should be aware of it. If for some reason, your system halts unexpectly in the Continuous run mode, you should increase N and decrease Z.

Example of commands for initiating Continuous Stepping:

```
N 65000}      ; set large value of N
Z 1}          ; set small value of Z
S s}          ; set appropriate value of slope
F f}          ; set appropriate First rate
R r}          ; set appropriate Max rate
H}            ; issue HALT-mode command
G}            ; begin stepping....
```

A second consideration is the following. When the CY525 is slewing (at any rate) and is given a new rate (by lowering I/O REQ, with the rate on the bus) the CY525 will ramp up to the new rate if it is higher than the current rate or down to the new rate if it is lower. The effect of this is to temporarily establish a new value of R and F to last until either another new rate is presented or the current motion is terminated. If the new rate is higher than the current rate, then the new rate replaces the maximum rate. If the new rate is lower than the current rate, then the new rate replaces the first rate. This has important consequences! When the ABORT line is pulled low to end the motion, the CY525 ramps down to the first rate and stops. In the second case described above, the CY525 is already slewing at the first rate, therefore it will abruptly halt with no ramping!

To clarify the above situation, let us consider the following two cases in detail:

CASE I.   Begin stepping with F=3 and R=80, then change rate
           while stepping to a new rate of 60. Since this is
           lower than 80, the CY525 ramps down to the 60,
           which becomes the effective new first rate. (F=3
           is lost for the duration of this motion but will
           still apply to the next motion command.) If the
           ABORT line goes low, the CY525 comes to a crash
           stop from a rate of 60, without ramping down as is
           probably necessary.

CASE II.   Begin stepping with F=3 and R=80, then change rate
           while stepping to a new rate of 127. Since this
           is higher than 80, the CY525 ramps up to the 127,
           which becomes the effective new maximum rate.
           (R=80 is lost for the duration of this motion but
           will still apply to the next motion command.)
           When the ABORT line goes low, the CY525 ramps down
           from R=127 to the first rate = 3, then stops if
           the ABORT line is still low.

Note that in the first case, the CY525 halts without ramping,
while in the second, the normal (desirable) ramp down occurs.
These motions are shown in the following figure.
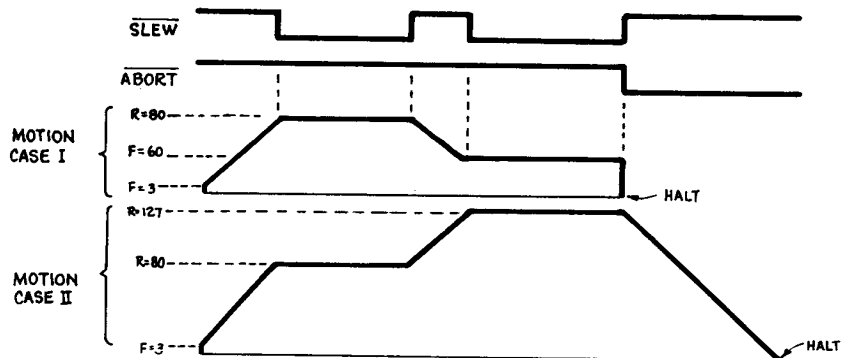
Figure 7.17   Illustrating two cases occurring for aborting motion
               while issuing rate changes on the fly.

73

In the continuous step mode, it is possible to change rates an indefinite number of times. In every case, the last rate change determines the effective values of **F** and **R** (as seen by the ABORT operation). If the last rate change caused the CY525 to ramp up, then the upper rate is changed and the lower rate is not (although it may have been changed by a previous on the-fly-operation). If the last change caused the CY525 to ramp down, then it is effectively at the first rate, and aborting will not ramp down but will stop immediately.

A VERY SPECIAL CASE

As a general rule, any valid rate parameter from 1 to 127 can be issued on the fly (while the motor is stepping). If the special case of R=0 occurs, i.e., if the data bus is held at zero when I/O REQ is pulled low, then the CY525 will immediately stop stepping (with no ramp down). To indicate this special case the SLEW line goes high and then returns low (the CY525 is now slewing at zero steps/sec!) but the MOTION COMPLETE line also goes low to signal this unique state. Internally, the CY525 is still slewing and will continue to slew (at zero rate) until a new rate is presented in the usual manner, by placing the rate on the bus and lowering I/O REQ. The only requirement is that the new rate be greater than the slew rate before R=0 was given. Thus if the CY525 was slewing with a rate of 2 when it was stopped (via 0) then it must be given a rate of 3 or greater to continue stepping (it will ramp up to the new rate). An example of motion is shown in the following figure.
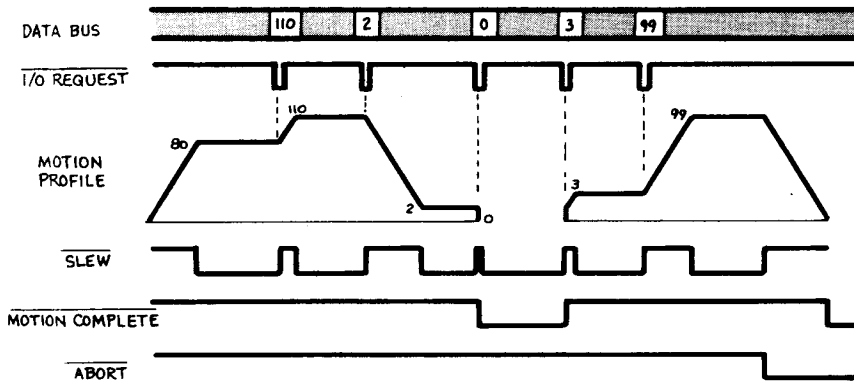
Figure 7.18   Illustrating several rate changes on the fly (including the special case of zero rate) and the corresponding behavior of I/O lines associated with rate changes.

Aborting from the ZERO-SLEW State

If the CY525 is placed in the Zero-SLEW state by issuing a rate of zero while the CY525 is slewing, the ABORT line can be used to terminate this state and prepare the CY525 for normal commands. The behavior of the relevant lines is shown below:
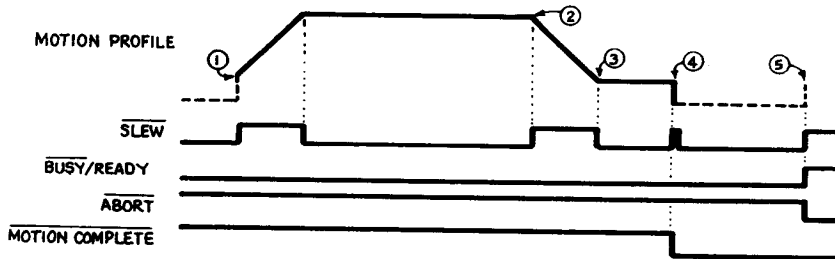
① In response to the G command (issued after H) the CY525 begins ramping from the Firstrate (specified by F) to the maximum rate (specified by R).

② In response to a new, lower, rate issued on the fly, the CY525 begins ramping down until it reaches the specified new rate.

③ When the new rate is reached, the CY525 begins slewing at this rate.

④ When a new rate of zero is issued on the fly, the CY525 immediately halts without ramping down. To indicate that the CY525 is "slewing at zero velocity" the SLEW line returns low but, to indicate the special nature of this state, the Motion Complete line goes low also.

⑤ If, instead of issuing a new rate command, the ABORT line is brought low, the CY525 will raise the READY line, raise the SLEW line, and leave the Motion Complete line low. The CY525 is now ready to accept new commands, including the Verify command.

SUMMARY

The CY525 provides two major operational modes: The normal mode is characterized by specifying all relevent parameters and then forgetting about the CY525 until the motion (or program) is complete.

The continuous step mode of operation allows much more exotic motion profiles but requires close attendence to the CY525 behavior, as well as careful consideration of abort behavior, etc. Choice of mode should of course be based on your system requirements.

PHASE SIGNALS

The tables and waveforms below indicate the sequential values
assumed by the stepper drive signals. The tables also indicate
how the step patterns correspond to the Offset parameter
specified in an OFFSET (O) command. Note that the OFFSET command
will set the internal step pattern pointer to that specified by
the parameter indicated, and the pulses will appear on the drive
signal lines. If the Offset parameter is greater than three, and
less than 128 (3 < o < 128), the phase lines will all go high
with no pulse. They will go high with a pulse on pin 35 for
parameter values greater than 127.

PHASE OUTPUT SIGNALS					
STEP	1	2	3	4	1
$\phi 1$	0	1	1	0	0
$\phi 2$	1	0	0	1	1
$\phi 3$	0	0	1	1	0
$\phi 4$	1	1	0	0	1
OFFSET	0	1	2	3	0
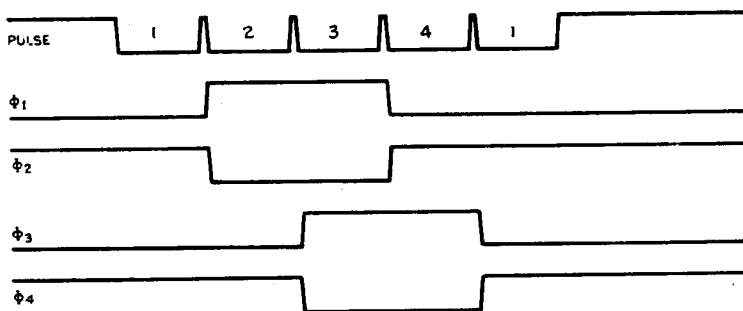
Figure 7.19   Phase Step Control Outputs.

The stepping waveforms indicate that outputs $\phi 1$ and $\phi 2$ are paired
together, as are outputs $\phi 3$ and $\phi 4$. $\phi 1$ and $\phi 2$ should be
connected to opposite ends of the same winding of a four phase
motor, and $\phi 3$ and $\phi 4$ should be connected to the other winding of
the motor. The user should study the waveforms and the motor
requirements carefully to determine the proper connection between
the phase outputs and the wires coming from the motor.

The phase outputs are considered on, or energized, when they are
low (0) and off, or de-energized, when they are high (1).

CY525 RAMP CURVES

The CY525 will operate at 127 different step rates for a given crystal frequency. Prior to issuing the first motion command, the user should specify the initial or first rate via the First rate command, **F**, for example F 3) tells the CY525 to begin stepping at 300 steps/sec. The rate command, **R**, is used to specify the maximum step rate, for example R 127) tells the CY525 that the maximum rate should be 9675 steps/sec. The Slope command, **S**, is then used to specify how rapidly the CY525 accelerates from 300 steps/sec. to 9675 steps/sec. Finally the Slope divisor, **Z**, is set at one for the time being. After these parameters have been specified, the CY525 can be commanded to begin stepping in the relative mode via a G command or to step to an absolute position via the P command. When either of these commands is issued (or executed in a program) the CY525 begins stepping at the first rate and keeps increasing the step rate at a rate determined by the slope command. The following sections describe the slope parameter in detail.

THE SLOPE PARAMETER

The SLOPE command, **S**, allows you to control the acceleration and deceleration performance of the CY525. In general, the greater the loading on a motor, the more time is needed to accelerate the load up to full speed. Control of this variable is provided by specifying the argument of the slope command. The range of this argument is from one to 255 with the larger values corresponding to larger accelerations. Thus the larger the slope parameter, the faster the CY525 reaches its maximum stepping rate.
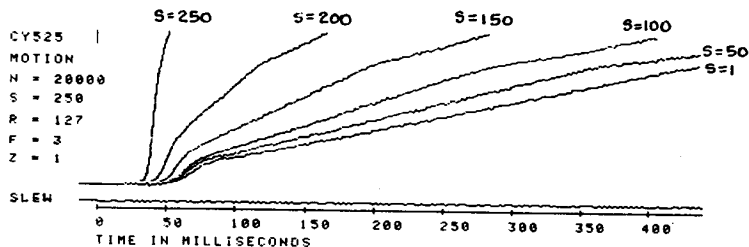
Figure 8.1 Typical velocity-vs-time curves for values of **S**.

How do you choose the correct slope? In most cases, simply start with the minimum slope, S=1, and increase **S** until the system fails to keep up with the CY525. While this method is the most realistic, the following discussion and tables will help calculate accelerations based on slope. We will also calculate ramp times, defined as the time required to accelerate from the first rate to the final rate.

Optimal Acceleration Curves

Before we calculate travel times for particular slopes we will discuss the general behavior of the CY525 as a function of slope. The first observation is that the shape of the acceleration profile is optimal for a rate parameter of 80 corresponding to a maximum velocity of 6000 steps/sec if an 11 MHz clock is used. For rate parameters above 80, the acceleration curve has a slight inflection, however it is not enough to seriously affect the performance, in fact, if the motor can accelerate through the rates from 3 to 40, then it should be able to reach the maximum speed of 10,000 steps/sec.

```
CY525    |
MOTION
N = 10000
S = 100
R = 80
F = 3
Z = 1



SLEW
       0     .5     1    1.5    2    2.5    3    3.5    4
    TIME IN SECONDS
```
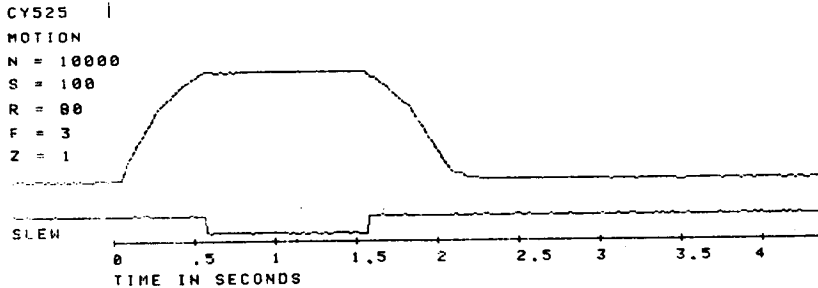
Figure 8.2   Optimal acceleration curve for R = 80.

Acceleration Curve as a Function of R

As seen in the figure below, the shape of the acceleration curve is fixed for a given value of the slope parameter and the rate parameter merely specifies when the CY525 will begin slewing at a constant velocity. The total travel is the same for all of these curves, thus the travel time (horizontal axis) depends on the travel speed, as is obvious from the figure.
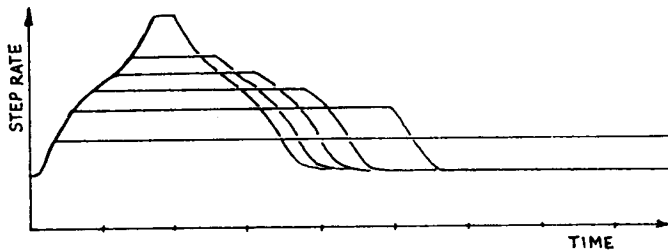
Figure 8.3   A family of curves for various rates with fixed slope parameter and fixed travel distance.

78

In order to calculate times we note that in general, the time required to travel from the first rate, R0, to the maximum rate Rmax is a function of the change in rate dR and the acceleration a(R), i.e.,

$$
R = \int_{t(R0)}^{t(Rmax)} a(R)\ dt = \int_{t(R0)}^{t(Rmax)} \left(\frac{dr}{dt}\right) dt = \int_{R0}^{Rmax} dr = r\ \bigg|_{R0}^{Rmax}
$$

therefore

$$
(Rmax - R0) = \int_{t(R0)}^{t(Rmax)} a\ dt
$$

If we make the simplifying approximation that the acceleration is constant, and also set t(R0)=0, then we have:

$$
(Rmax - R0) = a\ t(Rmax)
$$

or

$$
t = \left[\frac{Rmax - R0}{a}\right]
$$

Since this only works for constant acceleration, we need to examine this aspect. From the rate table and from the curves in figure 8.5, we see that the ramps are approximately linear, or at least piecewise linear, i.e. either:

$\Delta \approx 100$ steps/sec
$\Delta \approx 50$ steps/sec
$\Delta \approx 100$ steps/sec

1st Approximation   2nd Approximation
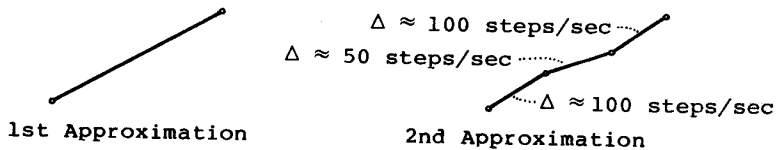
From rate parameters 3 thru 40, the rate changes 100 steps per unit. From 40 thru 100, the rate changes approximately 50 steps/unit. Finally, above rates of 100, the change per unit varies from about 75 to 125 steps. Since the highest rates were determined by constraints inherent in the implementation of the CY525, the lower rates were chosen with several features in mind:

79

First, many customers have asked for meaningful rate
parameters, and the rate parameters from 3 to 40 are
now meaningful in the sense that

$$\text{rate parameter} = \frac{\text{rate}}{100} \quad (\text{steps/sec})$$

Second, in order to use the full range of rates, the
system must be able to accelerate with greater than
100 steps/sec. between successive rates, therefore the
use of 100 at the low end imposes no severe constraints
on the system.

Third, for maximum rates up to about 8000 steps/sec.,
the curve is almost optimal in shape, i.e.:

Fourth, the total travel time is minimized by
accelerating faster at the slower rates as may be seen
in figure 8.4.

Fifth, consistent with the above conditions, the dynamic
range of the CY525 is maximized.
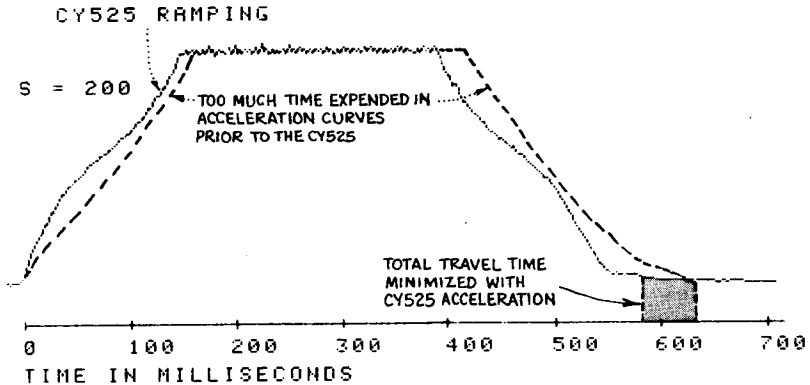
Figure 8.4   Illustrating optimal performance via minimal travel
time associated with CY525 acceleration curves.

80

After this digression on the slope, we see that there are two possibilities for our formula. The first one is to use an approximate constant for the whole curve and the other is to use a piecewise linear approximation to the curve. With this value the formula for acceleration as a function of the slope parameter, is given by

$$\text{accel} = \frac{dV(s)}{dT(s)} \approx \left(\frac{(22988 * D)}{(256 - s)}\right) = a(s)$$

where a(s) is expressed in (steps/sec2). If we now substitute this function into our expression for ramp time,

$$t = \frac{|\ Rmax - R0\ |}{a(s)}$$

we obtain the ramp time:

$$t = \frac{|\ Rmax - R0\ |}{\left(\dfrac{22988 * D}{(256 - s)}\right)}$$

For a clock frequency different from 11 MHz, this time should be scaled by (11/f) where F is the new frequency in MHz. Similarly, if the ramp time is desired for a slope S', different from 80 (steps/sec./unit), then scale by (80/S').

If we consider the case R0 = 300 steps/sec and Rmax = 9675 steps/sec then we can compute ramp times from the above formula for various values of slope parameter, s, as shown below. Also shown are the measured times from the curves shown in figure 8.5.

SLOPE (S)	RAMP TIME (in milliseconds)		ACCELERATION in (steps/sec^2)
	COMPUTED	MEASURED	COMPUTED
1	1299	1350	7211
2	1294	1350	7240
10	1254	1250	7475
25	1177	1200	7961
50	1050	1050	8927
100	795	800	11788
150	540	550	17349
200	285	270	32840
225	158	150	59323
250	30	38	306506

The formulas used in these calculations are summarized below:

$$accel = \frac{22988 * D}{(256-S)} \quad in \quad (steps / sec^2)$$

$$time = \frac{|R1 - R0|}{\left(\dfrac{22988 * D}{(256 - S)}\right)} \quad seconds$$

where   D = "distance" between two adjacent rate parameters in (steps/sec.) i.e., the "incremental slope".

R1 = maximum rate ... (argument of **R** command)

R0 = initial rate ... (argument of **F** command)

S = slope parameter   (argument of **S** command)

Below is a sample BASIC program for the HP-85 desk computer. This program accepts values for D, R1 and R0, and then computes time and acceleration for a sequence of slopes, input one at a time.

```
3000 PRINT "CY525 RAMPs"
3001 PRINT " "
3003 DISP "INPUT D(steps/sec) "
3004 INPUT D
3005 DISP "INPUT MAX RATE , R1"
3006 INPUT R1
3007 DISP "INPUT MIN RATE , R0"
3008 INPUT R0
3009 PRINT "R1=";R1;" R0=";R0;"
     D=";D
3010 PRINT "------------------"
3012 PRINT "S..ACCEL..time"
3014 DISP "INPUT slope S"
3015 INPUT S
3016 A=22988*D/(256-S)
3018 PRINT S;INT(A);INT(1000*(R1
     -R0)/A)
3025 PRINT " "
3030 GOTO 3014
```

```
CY525 RAMPs

R1= 9675   R0= 300   D= 80
-----------------
S..ACCEL..time
  1    7211   1299

  2    7240   1294

 10    7475   1254

 25    7961   1177

 50    8927   1050

100   11788    795

150   17349    540

200   32840    285

225   59323    158

250  306506     30
```

The CY525 performance for typical values of the slope parameter are shown in the following figures. These figures are followed by the CY525 RATE table, which lists the step rates (at 11 MHz) for each of the 127 possible rate parameters.
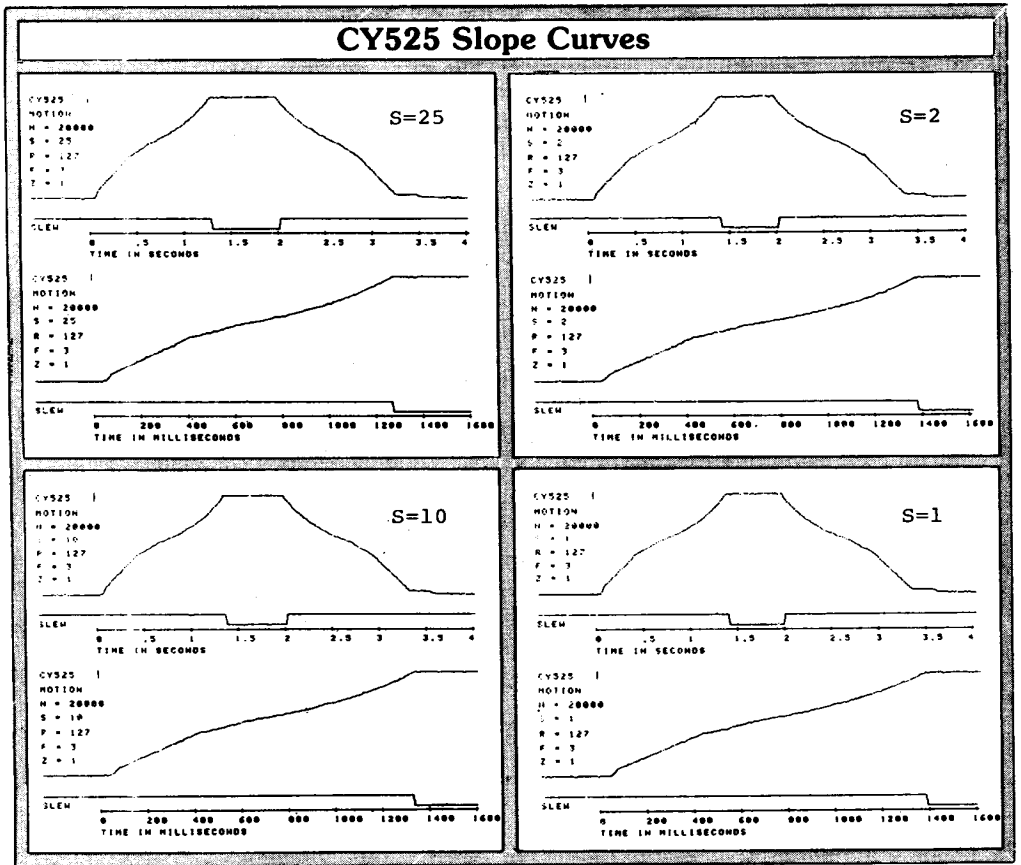
CY525 Slope Curves

S=250

S=150

S=225.

S=100

S=200

S=50
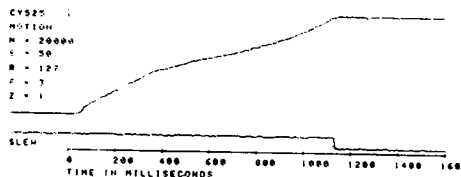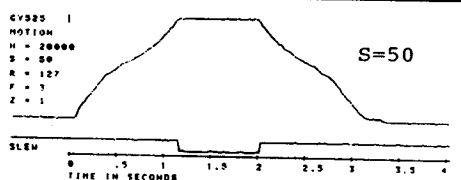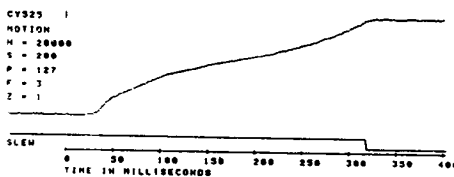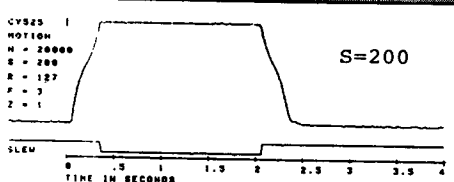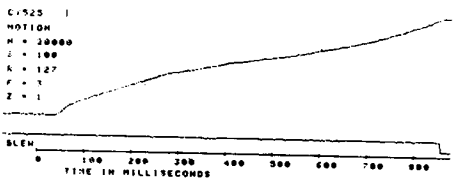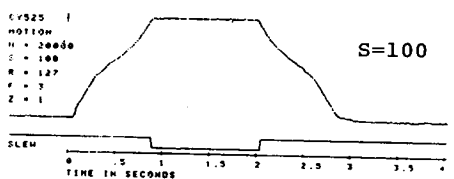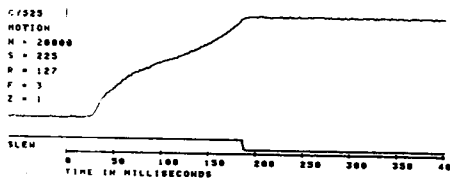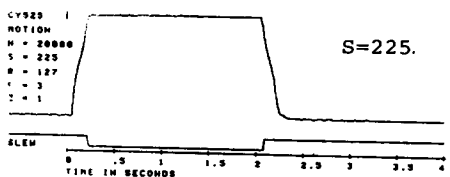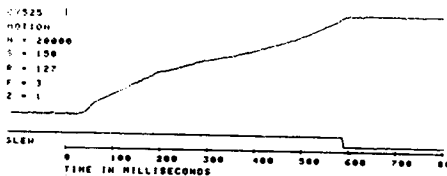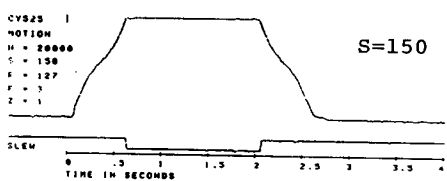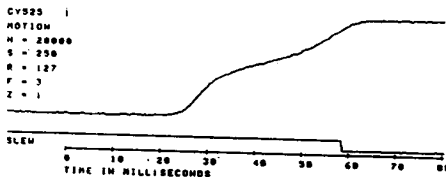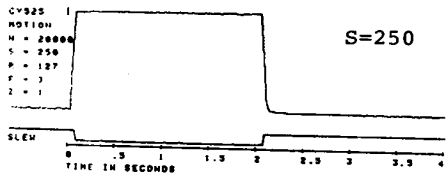
Figure 8.5   Example CY525 slope curves.

84

The Slope Divisor Z

If the acceleration is expressed as dv/dt for z=1, then the average acceleration will be (1/z)*(dv/dt) for any value of Z from 1 to 250. Typical accelerations are shown below. The range of rates from 300 to 4000 steps/sec (with 11 MHz clock) are very linear and the slope divisor calculations are most accurate in this range.
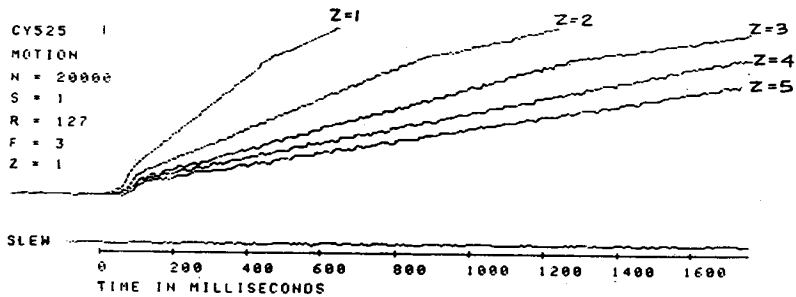

Figure 8.6   Effects of various Z factors using S = 1.

Special Case - Short Slew

If at the end of acceleration, there are fewer than 256 steps to take before starting to decelerate, the CY525 will immediately begin deceleration. No slew steps will be taken. When the CY525 reaches the first rate, it will continue stepping at this rate until the target position is reached.

		TABLE VII	CY525 RATE TABLE			

R	RATE	R	RATE	R	RATE
0	* †	40	4000	85	6231
1	*	41	4062	86	6284
2	*	42	4107	87	6338
3	300	43	4154	88	6393
4	400	44	4201	89	6449
5	500	45	4250	90	6507
6	600	46	4299	91	6565
7	700	47	4350	92	6624
8	800	48	4402	93	6684
9	900	49	4456	94	6745
10	1000	50	4511	95	6808
11	1100	51	4567	96	6871
12	1200	52	4624	97	6936
13	1300	53	4683	98	7002
14	1400	54	4743	99	7070
15	1500	55	4805	100	7138
16	1600	56	4869	101	7208
17	1700	57	4934	102	7280
18	1800	58	5002	103	7352
19	1900	59	5070	104	7427
20	2000	60	5141	105	7503
21	2100	61	5178	106	7580
22	2200	62	5214	107	7659
23	2300	63	5252	108	7739
24	2400	64	5289	109	7822
25	2500	65	5328	110	7906
26	2600	66	5367	111	7992
27	2700	67	5406	112	8080
28	2800	68	5446	113	8169
29	2900	69	5487	114	8261
30	3000	70	5528	115	8355
31	3100	71	5570	116	8451
32	3200	72	5612	117	8549
33	3300	73	5656	118	8650
34	3400	74	5699	119	8753
35	3500	75	5744	120	8858
36	3600	76	5789	121	8967
37	3700	77	5835	122	9077
38	3800	78	5882	123	9191
39	3900	79	5929	124	9307
		80	5978	125	9426
		81	6027	126	9549
		82	6076	127	9675
		83	6127		
		84	6178		

* Rates are slightly less than 300 s/s, subject to change

† R=0 halts stepping in continuous mode

STEP TIMING SIGNALS

The PULSE output (pin #35) may be used as a step timing signal.
When the CY525 is not stepping, this output is high. It goes low
at the beginning of every step, and stays low for the duration of
the step. When the step time is over, PULSE goes high again.
PULSE remains high for about 5 microseconds between steps. This
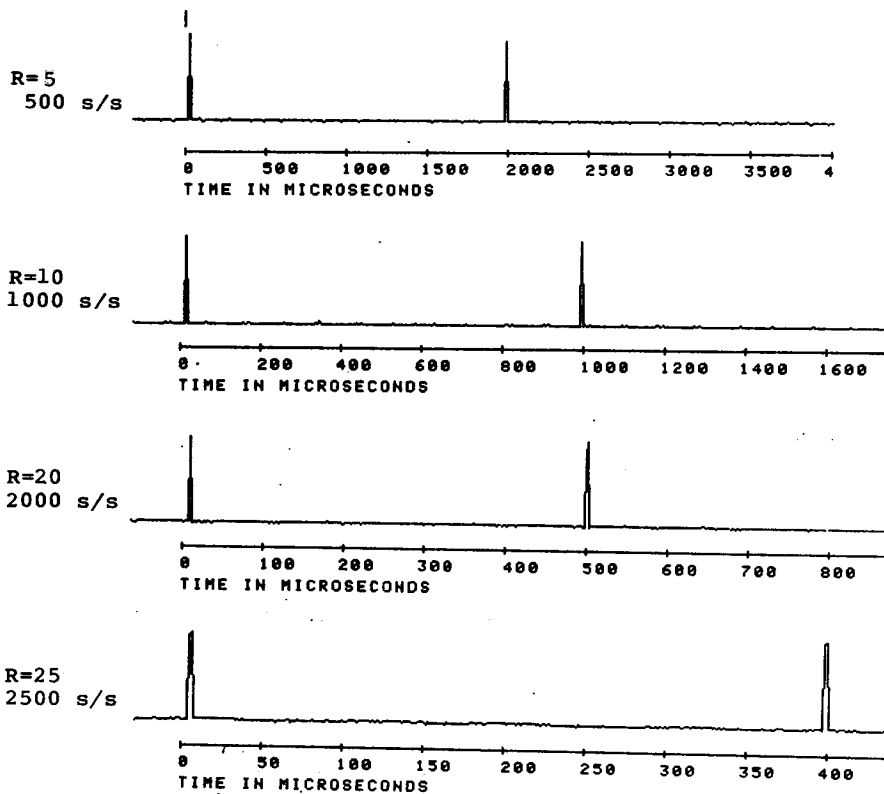time has been included in the rate equation.

R=5
 500 s/s

```
  0     500   1000   1500   2000   2500   3000   3500    4
TIME IN MICROSECONDS
```

R=10
1000 s/s

```
  0 .   200    400    600    800   1000   1200   1400   1600
TIME IN MICROSECONDS
```

R=20
2000 s/s

```
  0     100    200    300    400    500    600    700    800
TIME IN MICROSECONDS
```

R=25
2500 s/s

```
  0      50    100    150    200    250    300    350    400
TIME IN MICROSECONDS
```

Figure 8.7a   Pulse Times for various rates.

R=30
3000 s/s

TIME IN MICROSECONDS

R=40
4000 s/s

TIME IN MICROSECONDS

R=58
5000 s/s

TIME IN MICROSECONDS

R=88
6400 s/s

TIME IN MICROSECONDS

R=98
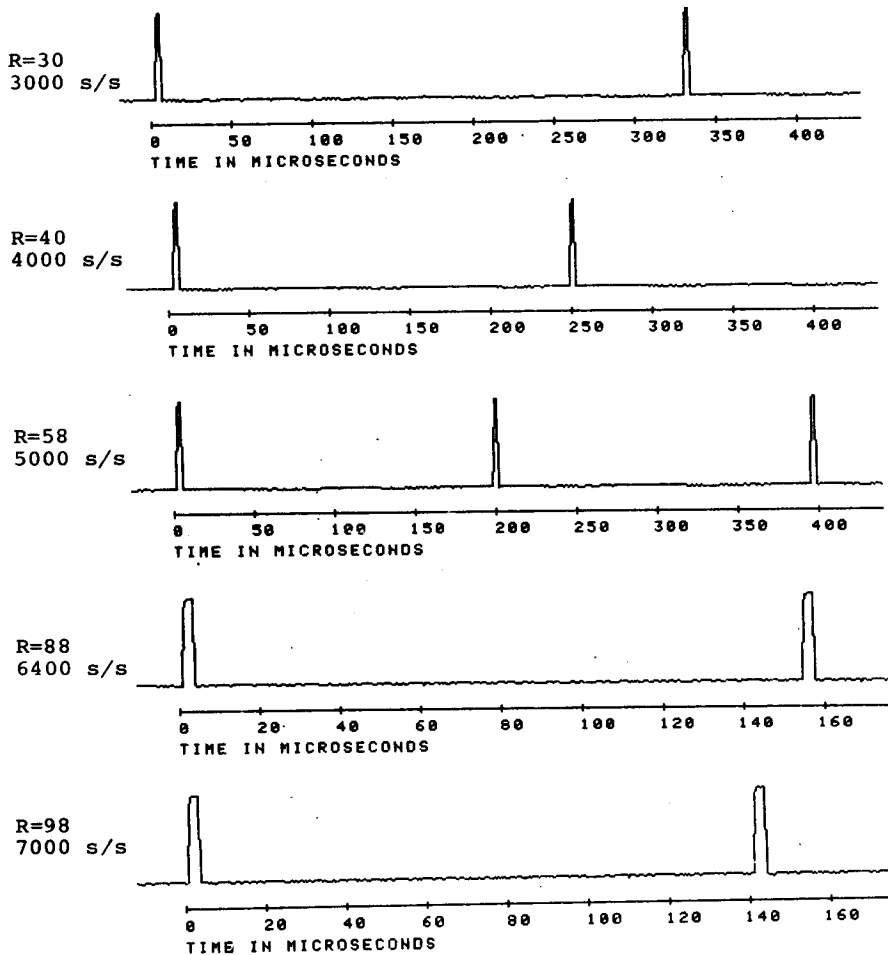7000 s/s

TIME IN MICROSECONDS

Figure 8.7b   Pulse times for various rates

The CY525 will begin ramping from the Firstrate specified as the argument of the F command and ramp up until it reaches the maximum rate specified by the R command. At this point the SLEW line (pin #29) will be brought low indicating that the maximum specified step rate has been attained. The CY525 will continue slewing until it is time to down ramp to hit the target position. The point at which this occurs is determined automatically by the CY525. The SLEW line is then turned off (set high), and the CY525 begins ramping down until the first rate is reached and then travels at this rate until the target position is reached. In most cases only a few steps will be taken at the slow rate.

If the number of steps to travel is less than the number required to reach the slew rate and then ramp down again, the CY525 will ramp up to the maximum rate possible that allows an equal ramp down in order to reach the target while traveling at the minimum rate. This assures that the total travel time is minimal for the specified Firstrate, slope, and travel.

For an approximation of how many ramp steps will be taken there are two important parameters:

1. how much time it takes to reach a certain rate
2. how far the motor has moved in that time

To determine the number of steps the motor has traveled during acceleration, simply compute the average step rate, and multiply by the elapsed time. Note that the average rate for a linear function is given by

$$Rav = \frac{|R1-R0|}{2}$$

so the travel distance is

$$Rav * time = \frac{|R1-R0|}{2} * \frac{|R1-R0|}{\left(\frac{22988 * D}{(256-s)}\right)} * steps$$

ABSOLUTE MAXIMUM RATINGS:

Ambient Temperature under bias...........0°C to 70°C
Storage Temperature.....................-65°C to +150°C
Voltage on any pin with respect to GND...-0.5V to +7V
Power Dissipation........................1.5 Watts

TABLE VIII				DC & OPERATING CHARACTERISTICS	
(T_A = 0°C to 70°C, V_{CC} = +5V±10%)					
SYMBOL	PARAMETER	MIN	MAX	UNIT	REMARKS
I_{CC}	pwr supply current		100	mA	
V_{IH}	input high level	2.0	V_{CC}	V	(3.8V for $XTAL_{1,2}$,RESET)
V_{IL}	input low level	-.5	.8	V	(0.6V for $XTAL_{1,2}$,RESET)
I_{LO}	data bus leakage		10	µA	high impedance state
V_{OH}	output hi voltage	2.4		V	I_{OH} = -40 µA
V_{OL}	output low voltage		.45	V	I_{OL} = 1.6 mA
F_{CY}	crystal frequency	1	11	MHz	see clock circuits

ELECTRICAL CONVENTIONS

All CY525 signals are based on a positive logic convention, with a high voltage representing a "1" and a low voltage representing a "0". Signals which are active low are indicated by a bar over the pin name, i.e., $\overline{PULSE}$.

All input lines except pins 1, 6, and 39 (I/O Req, Abort, and I/O Select) include 50K ohm pull-up resistors. If the pins are left open, the input signals will be high.

The data bus is bidirectional, and is tri-state during nonactive modes. Note that data bus signals are positive logic, and all command letters are upper case ASCII.

RESET CIRCUITRY

The RESET (pin #4) line must be held low upon power-up to properly initialize the CY525. This is accomplished via the use of a 1 µFd capacitor as shown in Figure 9.1. RESET must be low for 10 msec after power stabilizes on power-up. Once the CY525 is running, RESET need only be low for about 15 µsec (6 MHz crystal).
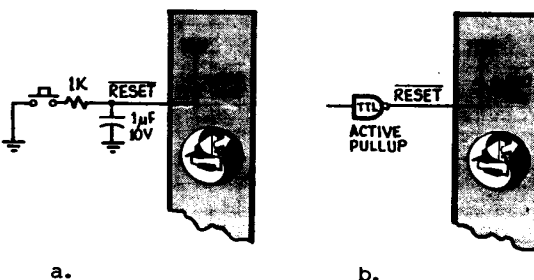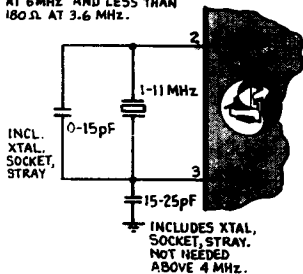

a.   b.

Figure 9.1   a) Reset Circuitry.
             b) External Reset.

CLOCK CIRCUITS

The CY525 may be operated with crystal, LC, or external clock circuits. These three circuits are shown in Figure 9.2. Unless otherwise specified, all timing discussed in this manual assumes a 6MHz series resonant crystal such as a CTS Knights MP060 or Crystek CY6B, or equivalent. The CY525 will operate with any crystal from 1 to 11 MHz, including a standard 3.58 MHz TV color burst crystal. All timing values specified in this manual will be changed by using different crystal frequencies. Time values must be scaled by 6/fcy, and stepping rates must be scaled by fcy/6, where fcy is the crystal frequency in MHz. Note, however, that the "D" command is calibrated for milliseconds at 11 MHz.
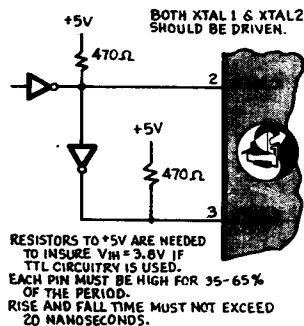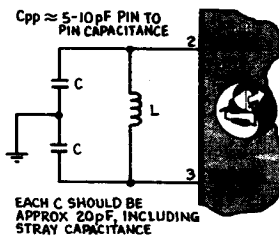

CRYSTAL   LC

EXTERNAL

Figure 9.2   Clock Circuits for CY525.

91

CY512/KIT

Since the CY525 contains all the logic needed to operate a stepper motor, including instruction decoding, parameter maintenance, and step timing, very little external circuitry is required to get a minimal stepper motor subsystem operating. A circuit indicating what is required is shown in Figure 10.2. As a convenience to our customers, Cybernetic Micro Systems has implemented such a circuit on a small printed circuit board. This board is made available as a kit, including all parts necessary to put the board together. A photograph of the CY512/Kit is shown in Figure 10.1.

The design consists of a CY512 (or substitute a CY525) with associated peripheral parts (crystal, socket, capacitors, etc.), buffers with LED indicators on CY525 output signals, switches for CY525 inputs, Abort/Terminate logic for closed loop motor operation, and a simple, unipolar driver circuit for the motor. Three edge connectors make the various signals available to the rest of the system. Signals are divided into a Data Interface, Secondary Control lines, and Motor and Power Supply connections. Finally, about half of the board consists of a wire-wrap area, useful for special data interfaces or motor driver circuits.

The kit supplies all parts needed to assemble the board, with assembly requiring only a few hours. In addition to the kit, the user must have a power supply, a source of commands for the CY525 (keyboard or computer), and a four-phase stepper motor. Complete documentation is provided. The CY512/Kit is ideal for prototyping, allowing first time users of the CY525 to quickly and easily get their part into operation.
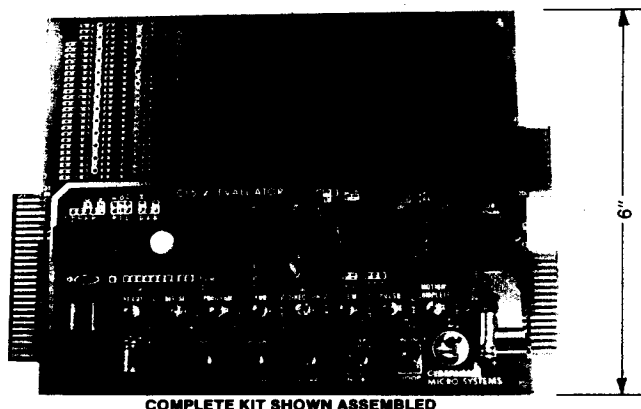

COMPLETE KIT SHOWN ASSEMBLED

Figure 10.1 CY512/KIT board available for prototyping.

TEST DEMONSTRATION CIRCUIT

The circuitry shown in Figure 10.2 provides a simple setup that allows use of an ASCII keyboard to control the CY525. It is generally helpful to use LEDs (light emitting diodes) on all output lines to visually display the state and state transitions. This is especially true when working through the detailed timing diagrams in the back of this manual. The programs used in the timing examples have slow rates specified, so that the transitions are visible to the user if LEDs are used on the outputs. A slower crystal frequency will also help. The CY512/KIT shown in Figure 10.1 is ideal for this purpose.

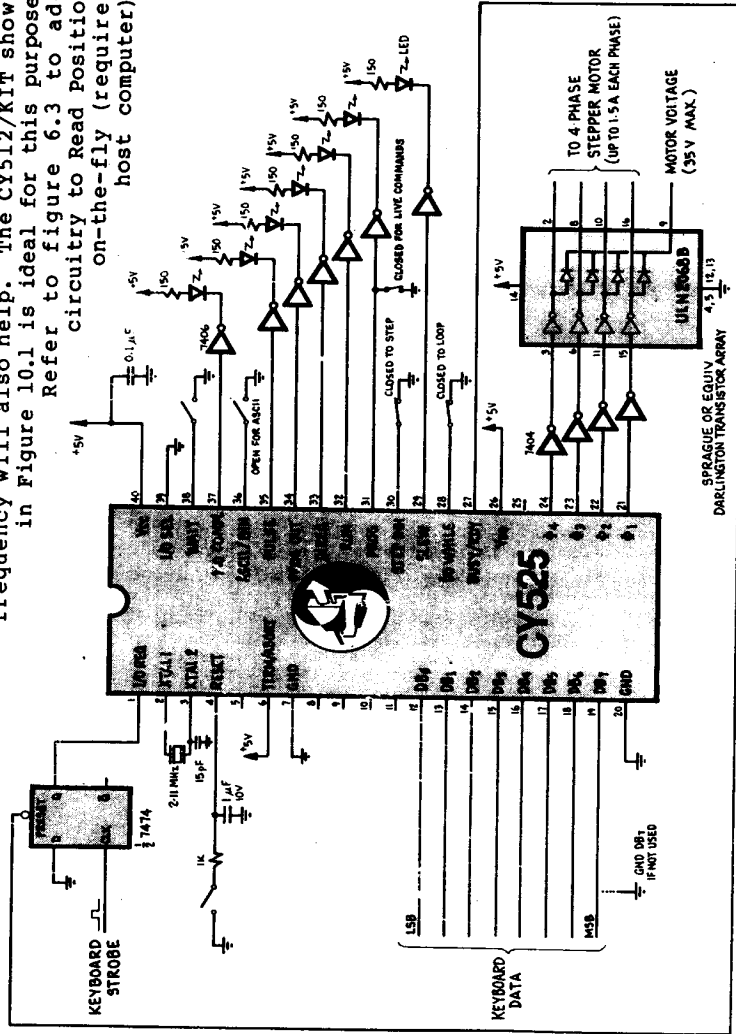Refer to figure 6.3 to add circuitry to Read Position on-the-fly (requires host computer).

Figure 10.2   Test Demonstration Circuit

93

CYB-002 MULTI-PURPOSE CONTROL BOARD

A general purpose prototyping board is available which will allow
the user to easily interface his computer, keyboard, or CRT to
his control application. The CYB-002 board comes ready to
assemble as a kit, with the capability of accepting any two
Cybernetic Micro Systems control chips in any combination. Thus
the board can become a dual axis stepper controller, waveform
synthesizer, programmable controller, printer controller, data
acquisition controller, and the like, with very little additional
effort. Support software will also be available soon.

The core of the CYB-002 is Cybernetic's new Local System
Controller, the CY250, which accepts ASCII commands, and
addresses either of the two target chips via a pass-through mode,
or accepts the data as direct commands to its own program buffer.
Since the CYB-002 is wired to accept an optional EEPROM, then
once programmed, it may also operate as an independent system.
The board has additional circuitry for an optional LCD display
and CY300 display controller, and for a network mode via the
CY232. The CY232 will give the user the option of stringing
boards together in a network with each having the ability to
address up to 256 devices.

User definable switches and LEDs are available for various input
and output signals, and an additional wire-wrapping area allows
the user to customize the board to his particular application--in
the case of the CY525, this could include the motor driver
circuitry. While the board was designed as a prototyping aid for
implementing the CYxxx family of chips, many users find that it
is the ideal solution to their control problems. The CYB-002 is
available with a variety of options: Display with CY300, Network
with CY232, Memory with EEPROM, Keyboard, and Target with any
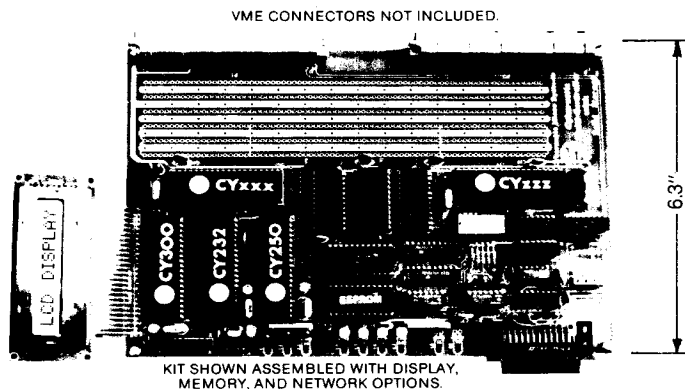CYxxx, as shown in the figure below:

Figure 10.3 CYB-002 Multi-purpose Control Board

DRIVER CIRCUIT CONSIDERATIONS

The CY525 provides the timing and logical signals necessary to control a stepper motor. However, to make a complete system, a driver circuit must be added to the CY525. This circuit will take the logical signals generated by the CY525 and translate them into the high-power signals needed to run the motor.

The user has two choices in the selection of driver circuits. Existing designs, usually in the form of pulse-to-step translators, may be used, or special designs may be created. Translators usually require a pulse and direction input, or two pulse streams, one for CW stepping and one for CCW stepping. The translator takes the pulse inputs and generates the proper four phase outputs for the motor. Note that it is also possible to drive motors with this scheme which are not four phase designs. Since the translator generates the actual motor driver signals, it only requires the pulse timing and direction information generated by the CY525 Pulse and Direction signals. This allows the CY525 to control three and five phase motors as well as the standard four phase designs.
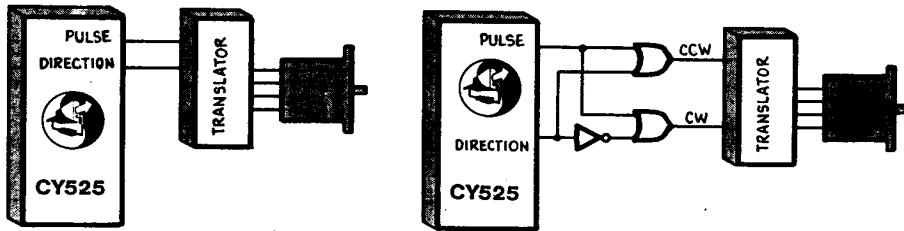

Figure 10.4 CY525 to Translator Driver connections.

If the user opts for his own driver design, the Pulse and Direction lines may be used, or the four phase outputs may directly control the driver circuits. This type of design makes full use of the CY525 signals. The following paragraphs are meant as a guide to various types of driver circuits, but should not be used as final driver designs. Detailed switching characteristics, transient suppression, and circuit protection logic have been omitted for clarity and simplicity.

Unipolar designs are the simplest drivers, and are generally useful when running at less than 600 steps per second. These designs require motors with six or eight leads, since the power supply is connected to the middle of each winding. The end of each winding is pulled to ground through a transistor controlled by one of the phase output lines from the CY525. Motor performance may be improved by adding a dropping resistor between the power supply output and the center tap of each winding. This decreases the field decay time constant of the motor, giving

faster step response. The performance increase is paid for by a higher voltage power supply and heat losses through the dropping resistors. This type of circuit is know as an L/xR circuit, where the x represents the resistor value relative to the winding resistance. An L/R circuit would not have any external resistors, while an L/4R circuit would use a resistor of three times the value of the motor winding resistance. Note that the power supply could be four times the nominal motor value with this circuit. Also note that this circuit requires only a single voltage and one transistor per phase.
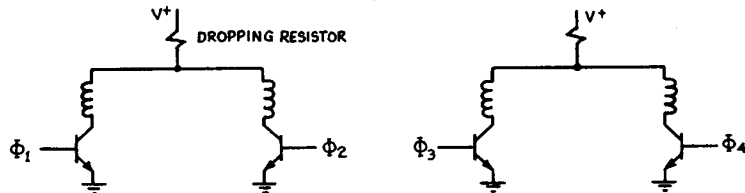

Figure 10.5 Unipolar driving circuits.

The second basic type of driver circuit is the bipolar design. In this case, the motor is driven only from the ends of each winding, with switching logic used to control the direction of current through the winding. These circuits may be implemented with a four lead motor, since only the ends of each winding are needed. Bipolar designs are more efficient in driving the motor, and result in higher performance than the unipolar designs. Two methods of switching the direction of current may be used. With a single voltage power supply, eight transistors are used, two per phase. Transistors are turned on in alternate pairs across each winding to control the current. The second alternative uses only four transistors, but requires a dual voltage power supply. In this case, one side of each winding is connected to ground, and the other side is switched between the positive and negative power supplies. In both designs it is very important to insure that both transistors on one side of the winding are not on at the same time, as this would short the power supply through the transistors, generally destroying the transistors in the process. Protection logic is usually included to insure that one transistor is off before the other is allowed to turn on.
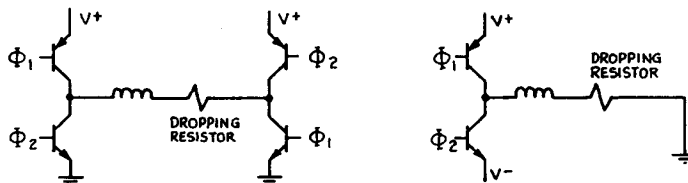

Figure 10.6 Bipolar driver designs.

The most advanced driver designs are variations on the unipolar or bipolar types, although they are generally implemented using the bipolar approach. These drivers are capable of the highest step rates attainable. They work by switching current or voltage through the motor at much higher than the rated value. This is done for only a short period of time, causing the magnetic field in the motor to change very quickly, without exceeding the maximum power dissipation of the motor. As long as the average dissipation does not exceed the motor rating, the motor will perform without problems. Once the maximum limit is reached, the motor may overheat and self destruct. One technique for increasing motor performance would simply apply a high voltage to the motor at the beginning of each step. This makes the motor react very quickly to the change in phase signals. After a short period of time, the voltage is switched to a lower value, allowing the motor to continue its motion without overheating. A second approach, known as a constant current design, senses the amount of current flowing through the winding, and adjusts the voltage applied to the motor to maintain the current at its maximum rated value. At the beginning of a motion, the voltage would be low, with a constant adjustment to a higher value as the motor speed increases, and back EMF decreases the current draw for a fixed voltage level. Another technique, known as chopping, may also be applied to these driver designs. This approach applies a voltage much higher than the rated value for a short period of time. The voltage is then turned off for another time period. This occurs many times per step, with the frequency of switching known as the chopping frequency. This frequency may be controlled by time, switching at a given rate, or it may be controlled by sensing the current flow through the motor, switching at a variable rate. The highest performance drivers are usually designed as bipolar chopper circuits.

The user should consult design guides available from the various motor manufacturers for additional information.

HANDSHAKE PROTOCOL

All commands and data transmitted from the master processor to the CY525 peripheral processor are sent asynchronously with complete handshaking performed. The master processor waits for the CY525 READY line to go HIGH before sending the active LOW I/O REQUEST signal. The data may be placed on the bus at any time prior to the HIGH-to-LOW transition of I/O REQUEST. The data should be stable on the bus until the CY525 RDY line goes LOW, indicating that the transfer has been acknowledged and that the CY525 is BUSY processing the command or data. The master then brings I/O REQUEST to the HIGH state. The next transfer can occur as soon as BUSY/RDY returns HIGH. The sequence described is shown in Figure 10.7.
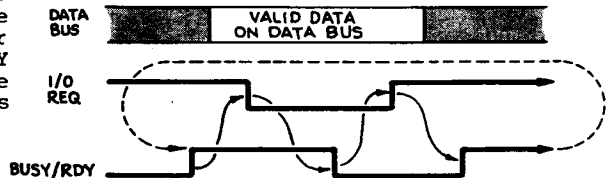
Example 8080/85 Driver: ASCII mode operation Bit 7 of data used as I/O REQUEST strobe, Routine entered with ASCII in C-register.

```
                        SENDCHR:
        0069 DBED       IN      STATUS
        006B E620       ANI     READY
        006D CA6900     JZ      SENDCHR ;WAIT TIL READY
        0070 79         MOV     A,C
        0071 E67F       ANI     7FH
        0073 D3EC       OUT     DATA    ;WITH I/O REQ LOW
                        ;
                        BUSY:
        0075 DBED       IN      STATUS
        0077 E620       ANI     READY
        0079 C27500     JNZ     BUSY    ;WAIT TIL BUSY
        007C 3EFF       MVI     A,0FFH
        007E D3EC       OUT     DATA    ;I/O REQ HIGH
        0080 C9         RET
```

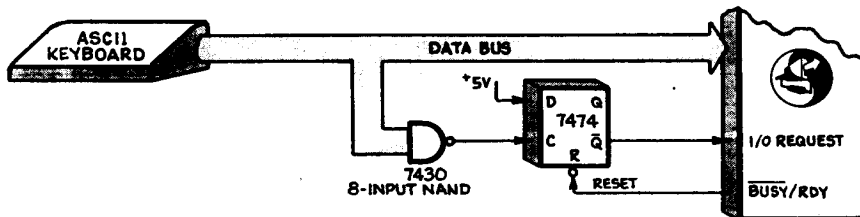Figure 10.7 Data Transfer Handshake Sequence

Figure 10.8 Write Strobe Generator for keyboards without strobe.

In the example shown in Figure 10.9, the CY525 is operating in the PARALLEL ASCII input mode. In this mode, bit 7 is always zero and b7 line of the CY525 data bus may be tied to ground. Since the user will normally transfer bytes of data from memory to the output port, the most significant bit of the data byte may be used to generate the I/O REQUEST strobe, thus allowing only one 8 bit output port to suffice. The "SENDCHR" routine, shown in Figure 10.7, demonstrates the coding used to achieve this. Of course, a separate port line may be used to generate I/O REQUEST, if this is desired. If the CY525 is operated in the PARALLEL BINARY mode, all 8 data bus lines are used, and a separate I/O REQUEST line is required. Note that in the example shown, use is made of the fact that the data and the I/O REQUEST signal may be applied simultaneously in parallel operation. If Verify mode is to be used, all 8 bits of the data bus must be free to operate bidirectionally. In this case, it is generally best to make I/O REQUEST and I/O SELECT separate lines from the data ports. See Timing and Control Information in section 7.
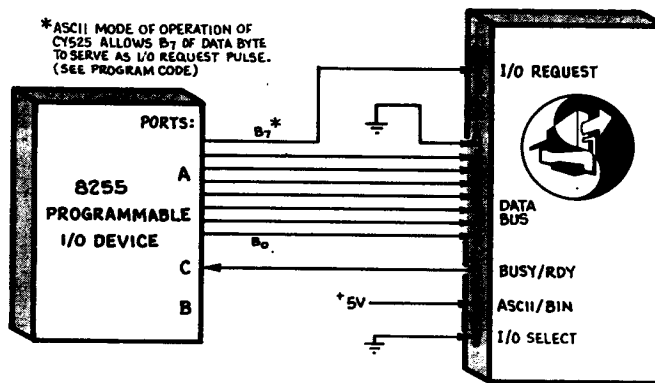

Figure 10.9   Example interface to CY525 using 8255 PIO.

OPERATION OF SEVERAL CY525s USING A COMMON DATA BUS

In systems where multiple CY525s are to be controlled by a host computer it is possible to use one eight-bit port to establish a common data bus for sending instructions to the CY525s. Each of the separate RDY lines (pin 27) of each CY525 must be monitored individually and each I/O REQUEST line (pin 1) must be activated separately. This technique effectively uses the I/O REQUEST line as a chip select (CS). A CY525 will ignore all bus information if its I/O REQUEST line is inactive. Note that On-the-fly operations could restrict the sharing of the data bus between multiple CY525s.
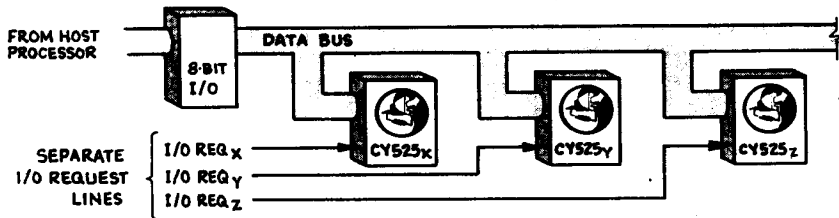
Figure 10.10 CY525s share common data bus by using separate I/O
REQUEST lines for chip select.

SYNCHRONIZATION OF TWO CY525s

Two CY525s, executing the same program, may be synchronized as
shown in Figure 10.11. The master controller can control the
WAIT line of the slave CY525 via the BITSET or CLEARBIT commands.
The slave CY525 is started first, with an EXECUTE command, and
executes a WAIT command and waits until the wait line (pin #38)
is driven low by the CLEARBIT command executed by the master
CY525 when it receives the (second) EXECUTE command. Both CY525s
then proceed to the next
instruction and are
synchronized as shown in Figure
10.11a, to within approximately
10 microseconds. Note that
when the two programs are not
identical, the master can also
wait for the slave to execute
its own CLEARBIT instruction,
and thereby achieve a more
general synchronization.
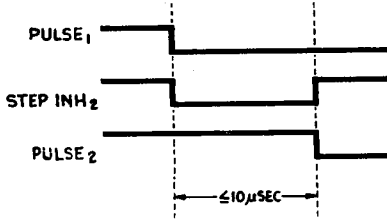
a.) Timing Diagram
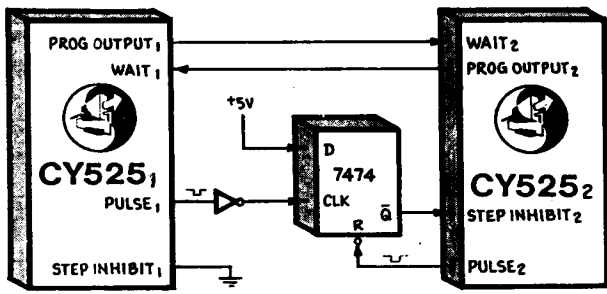
b.) Hardware

c.) Software

Figure 10.11 Synchronization of two CY525s.

COORDINATION OF MULTIPLE CY525s

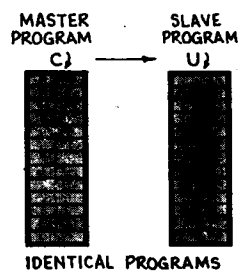Multiple CY525s may be synchronized to each other by use of the Programmable Output line, the Wait functions, Dowhile signal, and time delays. These may also be combined with other signals, such as Direction, Slew, or Motion Complete, used to select the point in the motion when the signal is presented to the waiting controller. Consider a general parts handling function in which the part must be handed off between two controllers. The geometry of the parts and the arms used to carry the parts requires that the hand off be carefully synchronized between the two controllers. The one to receive the part waits at the receiving position until the CY525 which has the part signals that it has arrived. The two arms then move together in a coordinated motion, reaching a point at which the distance between them is a minimum. The part is exchanged and the arms move apart, again in a coordinated motion. Once a certain position is reached, the arms are free to move independently, and continue with their assigned programs. If the motion is repetitious, both controllers can work with the same program, always being resynchronized at the hand off. The following program illustrates such a motion.


```
A 0)     Declare current position as home
R 120)
S 25)    Define stepping parameters
F 5)
P 14)    Move to the receiving position
E)       Define hand off program
U)       Wait for a part to arrive
P 0)     Arms move together to handoff position
+)       Change direction
C)       Activate mechanism to transfer part
D 90)    Delay for part to actually transfer
P 14)    Move apart, back to receiving position
D 90)    Delay for part to stabilize, arms apart
P 108)   Transport part to next handoff position
-)       Low DIR & PROG OUT indicates part arrived
P 122)   Move together with receiving arm
B)       Release mechanism which holds part
D 90)    Delay for part transfer to receiving arm
P 108)   Move apart, back to receiving position
D 90)    Delay for part to stabilize, arms apart
R 20)    Change step rate to slower rate
P 0)     Move empty arm back for next part
P 14)    Stay at the receiving position
R 120)   Change rate back to faster rate
T 0)     Repeat program if Dowhile low
0)       Else stop program
Q        End of program
X)       Execute program
```

Figure 10.12   Synchronized part transfer example.

EXAMPLE PROGRAMS AND WAVEFORMS

The timing sequence for a typical program is shown in Figure 10.12. In this example, the first-rate value, rate, and number of steps are present before entering the program-entry mode via the "E" command. These parameters are chosen to allow easy observation of the outputs using the test/demonstration circuit shown in Figure 10.2. The program entered sets the programmable output (pin #34), then takes three steps, clears the programmable line, and waits for the WAIT line (pin #38) to go low when the wait UNTIL instruction is executed. As shown, the STEP INHIBIT line has gone high, and the CY525 waits for this line to go low before stepping. The three-step motions are done one step at a time, using the LOOP command and a time delay between each step. The time delay is used to create a very slow step rate, which can be more easily observed. If the Dowhile line (pin 28) is low when the Til command is executed, the program will repeat from the beginning. When pin 28 is high, the program stops and the CY525 returns to the Command mode. Two program loops are shown in the waveforms.

```
R 80}
F 10}
N 1}
C}

E}

B}
G}
D 500}
L 3,1}
C}
U}
B}
G}
D 330}
L 3,11}
C}
T 0}
B}

Q

X}
```

Figure 10.13 Sample Program and Timing Diagram.

102

Figure 10.14 provides timing relations for a command sequence that inputs the parameters and executes a "G" command to begin stepping. The I/O REQUEST, BUSY/RDY, and INSTROBE signals are related to the data bus and several outputs are shown as a function of the STEP INHIBIT input.

```
COMMAND MODE INPUT SEQUENCE:

R 80)      set RATE = 80
S 255)     set SLOPE = 255
F 10)      set FIRSTrate = 10
N 4)       set NUMBER of steps = 4
G)         GO, begin stepping
```


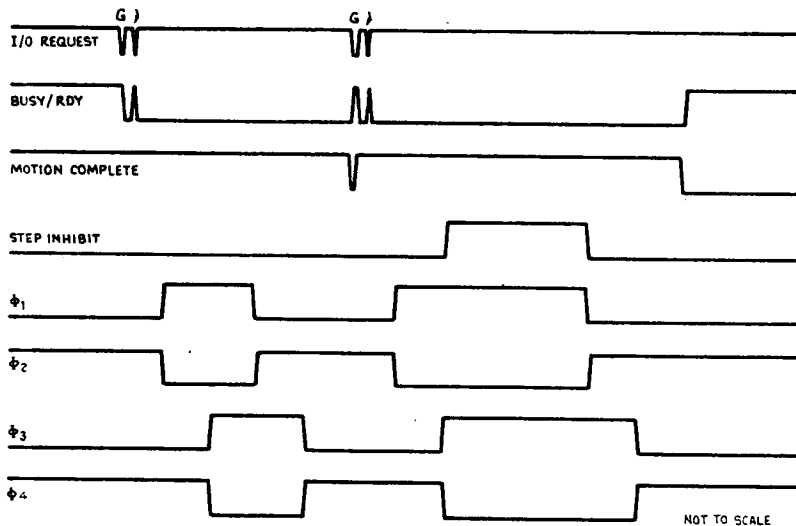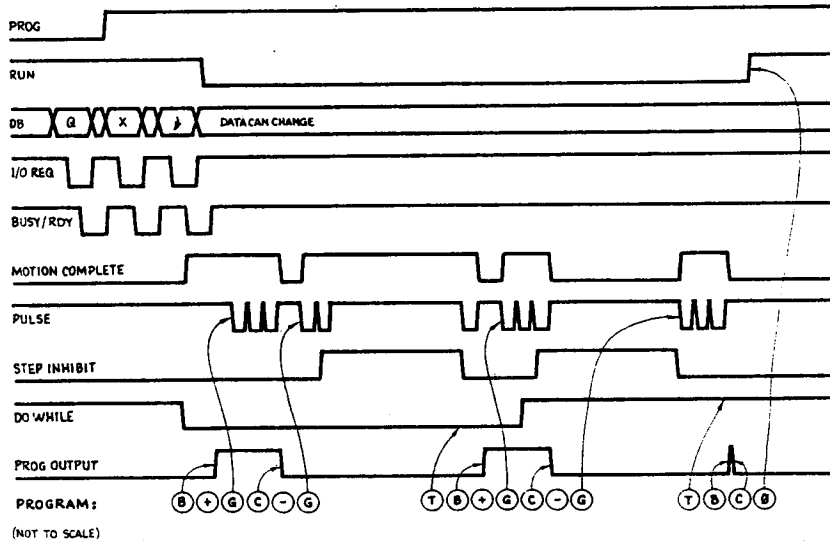Figure 10.14   Timing Diagram for Commands.

The use of the "loop TIL" instruction is illustrated in Figure 10.15.   The PROG/LIVE and RUN outputs are also shown as a function of the "Q" and "X" commands and the "0" instruction. The program loops until the DOWHILE line (pin #28) goes high, then fetches the next instruction.   The effect of the STEP INHIBIT input on the MOTION COMPLETE output is also shown.

```
PROG
RUN
DB        Q    X    J    DATA CAN CHANGE
I/O REG
BUSY/RDY
MOTION COMPLETE
PULSE
STEP INHIBIT
DO WHILE
PROG OUTPUT
PROGRAM:    Ⓑ⊕ⒼⒸ⊖Ⓖ         ⓉⒷ⊕ⒼⒸ⊖Ⓖ         ⓉⒷⒸⓞ
(NOT TO SCALE)
```

PRESET: C)   clear output line   ⎤
            R 90)   set RATE = 90   ⎥
            S 200) set SLOPE = 200   ⎥ P₀ ------- [R] PRESET PARAMETERS
            F 10)   set FIRSTRATE = 10   ⎥
            N 3)   set NUMBER steps = 3   ⎦

ENTER PROG: E)   ⎤
            B)   set output line   ⎥   [P] MAIN PROGRAM LOOP
PROGRAM   +)   set CW direction   ⎥
CODE   G)   GO, begin stepping   ⎥ P₁   TEST EXTERNAL
            C)   clear ouput line   ⎥   [T]   CONDITION AND
            -)   set CCW direction   ⎥ DOWHILE DOWHILE LOOP TIL TRUE
            G)   GO, begin stepping   ⎦

            T 0) repeat above prog Til DOWHILE = HI ⎤ T   EXECUTE
                                                              AFTER TEST
            B)   set output line   ⎤   [P]   PASSED
            C)   clear output line   ⎥ P₂
            0)   exit run mode, enter command mode ⎦

QUIT: Q
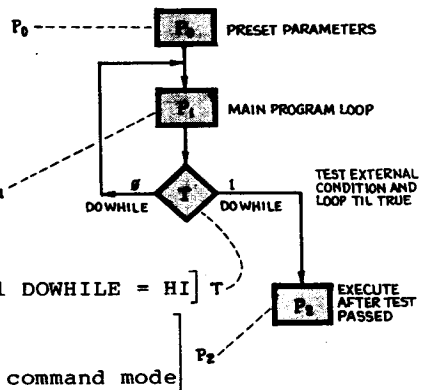EXECUTE: X)   EXECUTE

Figure 10.15   Timing and Control for Program Entry and
               Conditional Looping.

104

RS-232-C RECEIVE ONLY INTERFACE

When the user wishes to communicate with the CY525 over a serial data link, a special data interface, such as the RS-232-C design shown in this section, must be used. The main component of such a design is the UART (Universal Asynchronous Receiver Transmitter), which transforms the serial data from the data link into the parallel form required by the CY525.

The design shown here is a "receive only" type, meaning that it can only receive data, not transmit. This design will allow the user to send commands to the CY525, but will not allow the Verify mode to work. Bidirectional communication through a UART is very difficult with the CY525, because there is no direct control over the I/O SELECT line or the number of bytes to transmit from the CY525. Those who require the Verify mode must use a more sophisticated design to control the handshake protocol during the verify portion.

As shown in the schematic below, only two signals are needed from the RS-232-C lines. Transmitted Data contains the data sent by the host to the CY525, and Signal Ground is a reference for the data line. Since signals on the RS-232-C interface are not TTL compatible, the transistor circuit connected between Transmitted Data and the UART acts as a converter, generating the TTL equivalent of the data signal for the UART.

The type of UART shown is a single, 40 pin IC. It was chosen because the operating mode is set by connecting the control lines either high or low. Other types of UARTs require a command word to be written to an internal register which controls the mode, something the CY525 is not capable of doing. The type of UART shown is made by several manufacturers, and is readily available. The mode control lines should be connected so that the operating mode of the UART matches that of the host system. This is very important in getting data transmitted properly to the CY525.

Whenever the UART receives a character, the data available line (DAV) goes high. This signal runs I/O REQUEST, indicating to the CY525 that a command character is ready. As the CY525 reads the character, the INSTROBE signal is used to put the character onto the CY525 data bus, by controlling RDE, which brings the received data lines (RD1 to RD8) to their active state. BUSY/READY, connected to RDAV, then resets the DAV signal, clearing the I/O REQUEST. Thus, the standard signals from the UART fully implement the two-line data transfer handshake used by the CY525.

The rest of the circuitry is a baud rate generator. It creates the clock rates needed to operate the UART at most of the common data transfer rates. The 7404 and crystal circuit is an oscillator which runs at 2.4576 MHz. This frequency is an exact multiple of the popular baud rates used. The CD4040 is a CMOS, twelve stage counter. It takes the 2.4576 MHz clock rate and divides it through twelve binary stages, creating one half the frequency of the preceeding stage in each case. The outputs are

labeled with the resulting data baud rate, although the actual signal frequency is sixteen times this rate. The clock inputs of the UART should be connected to the desired rate. It will do an internal divide by sixteen, generating the data rate needed by the interface.
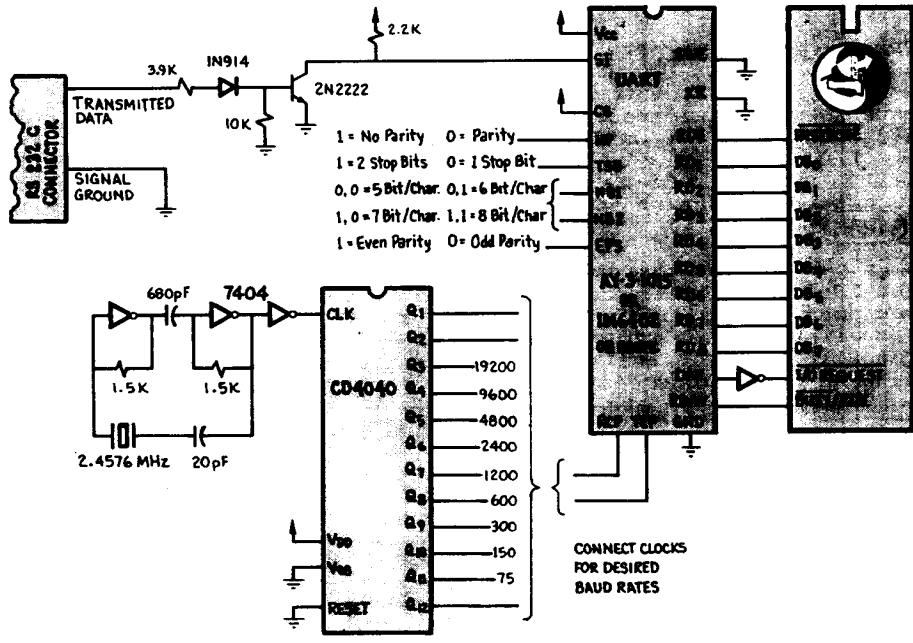

Figure 10.16   RS-232-C Receive-Only Interface Schematic.

RS-232-C TRANSMIT/RECEIVE INTERFACE WITH CY232

The CY232 Parallel/Serial Network controller enables the user to both transmit and receive data from the CY525 parallel device via a serial RS-232-C port. The actual CY232 to CY525 interface is very easy as shown in the schematic below. However, since the CY232 gives the user the ability to address multiple devices on a network, the CY232 address lines should be tied high or low to provide the CY525 with a specific address, and this address should be used when writing to the CY232/CY525. Also, multiple CY525s can be addressed this way by preceding each with a separate CY232 with a different address or by connecting multiple CY525s to a single CY232. In the second case, the CY232 address decoding logic should be combined with the CY232 DAV to generate a unique I/O REQUEST for each CY525 (see also Figure 10.10). The CY232 manual gives complete details on this interface.
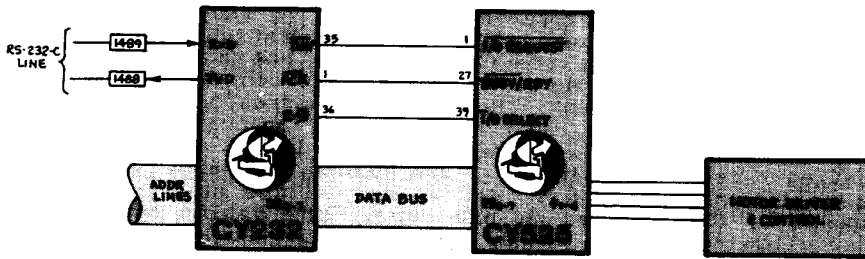

Figure 10.17   CY525 connections to CY232.

PROM STAND-ALONE INTERFACE DESIGN

When the CY525 is to be used in specific applications, with fixed commands or a small number of different programs, the user may eliminate the need for a keyboard, which is prone to typing errors, and the need for a computer, which may not be justified for the application. By programming the CY525 commands into a PROM or EPROM, a stand-alone design may be generated, in which the program may be selected by switch position, and a push button is used to get things going. The BUSY/READY signal from the CY525 is used to advance the address counter of the PROM, and the hardware automatically loads the commands, one byte at a time, until the end of the program is reached. The end of program then inhibits further program loading until the procedure is restarted by setting the address to the front of a program again.

The circuit shown in this section is started by selecting the desired program starting address for the PROM. With the 74193 counters, any address may be chosen by setting the counter inputs and pulsing the load signal low. The schematic shows the load signal controlled from the CY525 RESET, but a separate load switch could be used. The outputs from the counters control the address inputs to the PROM. Each address corresponds to a single CY525 command character, so the PROM should be organized as eight data outputs per address. Many popular PROMs and EPROMs are organized this way, including 2708s, 2508s, and 6309-1s. Enough address lines must be provided to access the number of bytes required by the program or programs. The design shows eight lines, allowing for 256 bytes, but more could be added by simply cascading additional 74193s.

When the starting address is loaded, the PROM will output the first command byte to the CY525, so the data bus will have the byte ready when the CY525 reads it. When the CY525 becomes ready, with a high level on the BUSY/READY line, the 7400 nand gate generates a low output to the CY525 I/O REQUEST line. This will tell the CY525 that a command byte is available. The CY525 will read the byte from the data bus and then go busy, indicated by a low level on the BUSY/READY line. This will generate a high level on I/O REQUEST, indicating that the byte transfer has been completed. The same signal also clocks the 74193 counters, advancing the PROM to the next byte location, and putting the next command byte on the data bus. When the CY525 has finished processing the last command byte, it will go ready again, generating another I/O REQUEST, and causing the CY525 to read the next command byte.

The above procedure continues until the PROM address reaches a value at which the data byte output is all bits high, 0FFH. This will generate a low output from the 7430, which will keep the CY525 READY signal from generating another I/O REQUEST. The circuit stops clocking at this point, and stays frozen with I/O REQUEST high and the 74193 counters set at the address which contains the 0FFH byte value. No more bytes will be transferred

until the address is changed by another load pulse to the 74193. This means that the user must end the program to be loaded into the CY525 with a byte containing the 0FFH. Note that the 0FFH is not read into the CY525, it is only used to stop the circuit from advancing any further. Since 0FFH is not a legal ASCII character, it may be used to end the program without fear that such a value might be part of the program, so long as the CY525 is operated in the ASCII mode. If the CY525 must be operated in the Binary mode, and the program to be loaded must contain an 0FFH data value, some other means of stopping the program must be found. In this case, the best approach would detect the end of program by a unique address from the 74193 counters. This would require the user to place the program in the PROM so that the last program byte occurs at the address just before this end of program address. Note that the same logic now used will work if the last address is 0FFH. In this case, the 7430 inputs connect to the 74193 outputs instead of the data bus. The last byte of the program should be at location 0FEH, one before 0FFH, since the byte at location 0FFH would not be read by the CY525. With this scheme, the starting address of the program would depend on the length of the program, and must be set properly before the load pulse is given to the 74193. The design shown in the schematic allows the starting address to be fixed, with the end indicated by the 0FFH data byte value.
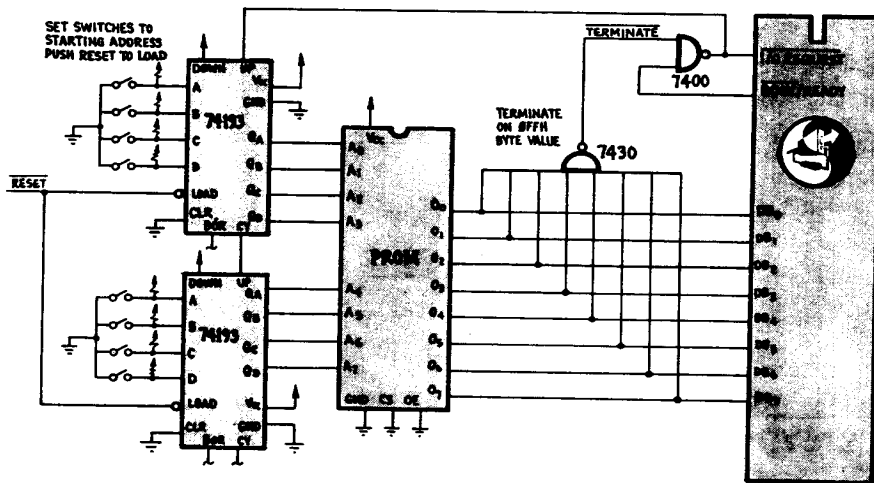

Figure 10.18   PROM Stand-alone Interface

EEPROM STAND-ALONE INTERFACE DESIGN

The CY250 Local System Controller will allow the user to interface the CY525 to an EEPROM for easy storage of often used programs and for a stand alone system. The CY250 accepts serial or parallel commands and can address either of two CY525s via a pass-through mode, or accepts data as direct commands to its own program buffer. Alternately, the command sequences may be defined once and sent to the EEPROM, where the various command sequences are stored as named procedures, with the CY250 taking care of the EEPROM operation, space allocation, and name directory. This allows frequently used programs to be remembered by name and recalled whenever they are needed. For stand-alone operation, the CY250 has an "auto recall" feature which calls a specified routine from the EEPROM on power up or reset. This EEPROM interface has been implemented on the CYB-002 board shown in figure 10.2. More details on the EEPROM interface may be found in the CY250 manual and the CYB-002 manual.
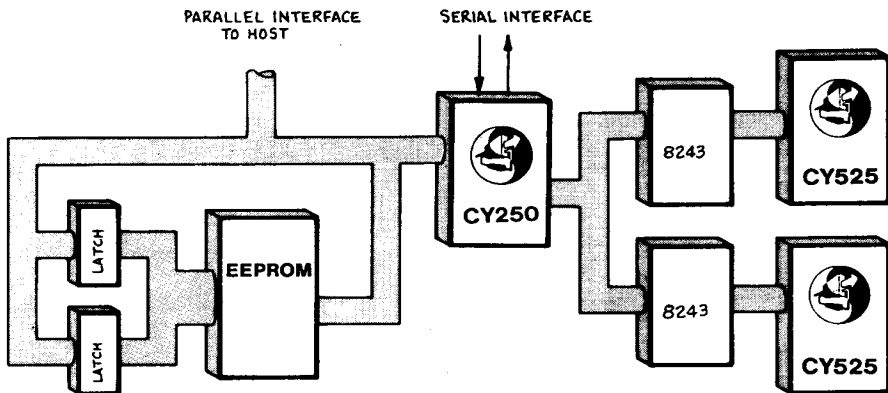
Figure 10.19 CY525 interface to EEPROM through CY250.

110

COMPUTER CONTROL OF CY525

The ability to control all of the CY525 control inputs and
monitor all of the CY525 outputs allows the designer to exercise
the maximum control over the device. The following sections
present information that may be used as a guide to interfacing
the CY525 to a computer via the use of programmable I/O devices
such as the Intel 8255. The programs are written for the 8080
microprocessor, but the general scheme will, of course, work with
any computer using two parallel 8-bit output ports and one
parallel 8-bit input port. For Verify mode, the data bus port
may be bidirectional, or replaced by a tri-statable output port
and another input port. The setup is as shown below:
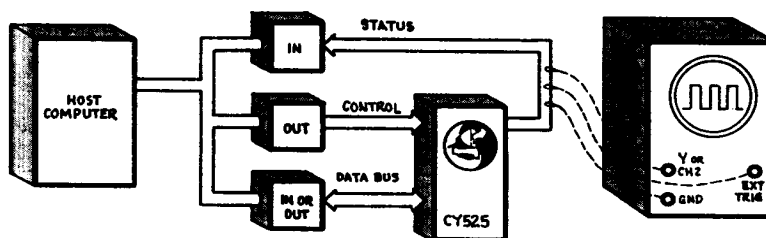

Figure 11.1   Example setup for Test/Display/Control of CY525
              Stepper Controller.

By using a loop in the host computer (or in the CY525) the user
can achieve a repetitive operation of the CY525 that allows easy
display of CY525 signals on a standard oscilloscope. The use of
externally triggered horizontal sweep circuits to synchronize the
scope display is particularly convenient. The MOTION COMPLETE
(INT REQ 1) output (CY525 pin 37) and the PROGRAMMABLE OUTPUT
(pin 34) serve well as external triggers.

ENTER/QUIT PROGRAM MODE

A key feature of the CY525 is the capability to accept and
execute sequences of instructions; i.e., stored programs. The
device powers-up in the "Command" mode of operation in which
valid instructions are executed as they are received. If the
ENTER command, "E", is given, the device initializes the relevent
(internal) pointers and prepares to accept the program entered.
All commands received prior to the receipt of the "Q" command are
stored in the program buffer in the order in which they are
received. Each command is entered just as in the command mode;

111

that is, the opcode is entered followed by either the "Linend" character "⟩" (carriage return) or a delimiter and parameter string terminated with the "⟩". **The only command NOT terminated with a Linend (0DH) is the QUIT command, "Q"=51H. The Linend should not be used immediately following the "Q" character.** The escape (QUIT) command terminates the program entry mode of operation, and returns the system to the command execution mode.

The maximum efficiency in use of the CY525 may be gained by presetting parameter values before entry and execution of the program. All parameter values have their own storage registers, so they need not occupy program buffer space, if the values stay constant during program execution. The host program may treat the CY525 program as a "Co-routine" that can be passed a set of parameters and invoked via the EXECUTE command. The host can then sample the RUN output (pin #32) or utilize this output in an interrupt mode to detect program completion and load new parameters **or programs**, as appropriate. This mode of operation is particularly well suited for inclusion in multi-tasking systems, when two or more CY525s are controlled by a single host.
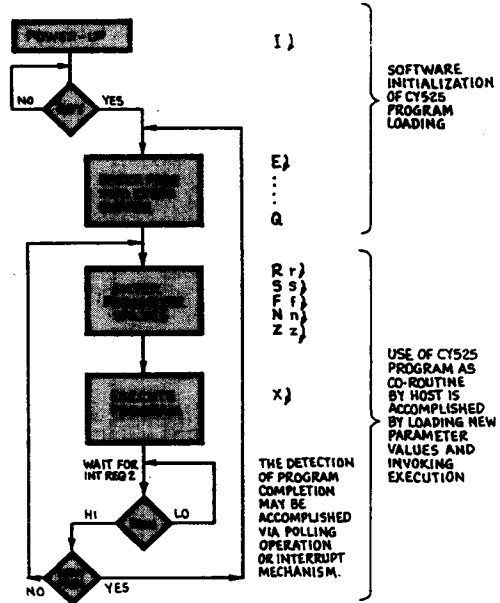
Fig. 11.2 CY525 used as "co-routine"

CY525 STAND-ALONE APPLICATIONS

The CY525 receives data and commands from an 8-bit data bus. The source of data in most cases will be from an ASCII keyboard during prototype development and a microcomputer in the final system. The CY525, of course, does not know or care where the commands and data actually come from. This means that as long as the handshake protocol is properly implemented, the commands can be stored in a ROM, PROM, or EPROM and can be sequenced to control the CY525 with no host processor at all. For certain limited repertoire machines and stand-alone applications, this may be a very practical solution. A conceptual diagram of this type of system is shown in Figure 11.3. See also PROM Stand-Alone Interface Design in section 10.
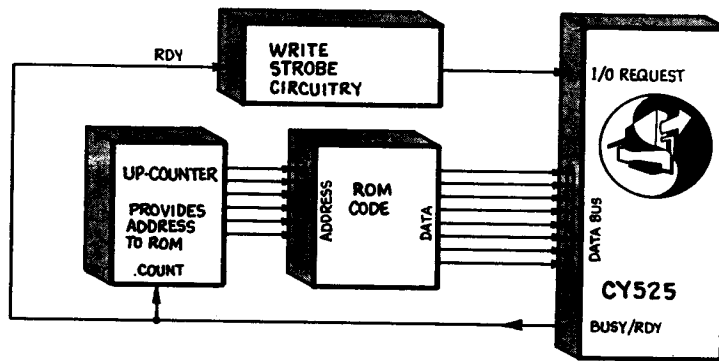
112

Figure 11.3 The CY525 can receive commands and data from a ROM sequencer for many stand-alone applications not requiring a host microcomputer.

PROGRAMMING EXAMPLES

The following pages illustrate several programming examples, including waveforms and program listings. Programs are all written in 8080 Assembly Language, but the comments should allow those readers who are not familiar with the 8080 to understand what the various subroutines are doing. The programs were used on an SDK80 board, with the CY525 included in the wire wrap area.

We start with an equate table, indicating how the CY525 was connected to the SDK80 I/O signals. The names assigned to the various signals are used in the other routines. The table is followed by a Binary mode example, with the data buffer, BINBUFFER, showing the exact data bytes sent to the CY525 in this program. All bytes except the 0FFH at the end of the table are sent by the SENDPARALLEL program, which is shown next. This routine implements the basic data transfer between the SDK80 and the CY525, illustrating an example of the handshake protocol needed to transfer the bytes. It may be used in either Binary or ASCII mode. The ASCII mode example, which follows the SENDPARALLEL program, sends the same commands to the CY525 in ASCII mode as the Binary mode example shown previously, with ASCIIBUFFER containing the ASCII data bytes sent by this program. Finally, another Binary mode example is used to generate a repeating oscilloscope waveform.
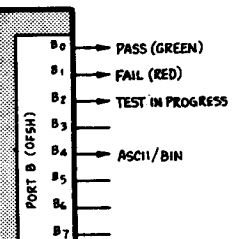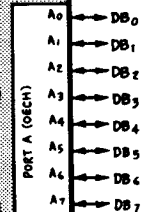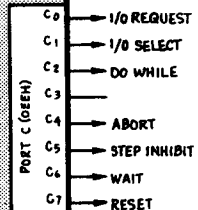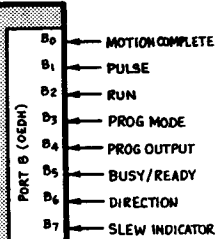
| TABLE IX | EQUATE TABLE AND 8255 PORT ASSIGNMENTS |

The CY525 is connected to 8255-A4 ports 0ECH
to 0EEH on the Intel SDK80 board.

8255 PIO

```
            ;
            ; EQUATE TABLE
            ;
00F5 = LEDS     EQU 0F5H  ;TESTING LEDS ON PORT B=F5H
            ;
0001 = GREEN    EQU 1     ;PASS
0002 = RED      EQU 2     ;FAIL
0004 = YELLOW   EQU 4     ;IN PROGRESS
            ;
00EC = DATA     EQU 0ECH  ;PORT A IS DATA BUS ON A4
            ;
00ED = STATUS   EQU 0EDH  ;CY525 STAT READBACK ON PORT B
            ;
0001 = MOTION   EQU 1     ;B0--MOTION COMPLETE
0002 = PULSE    EQU 2     ;B1--PULSE OUTPUT
0004 = RUNBAR   EQU 4     ;B2--RUN MODE PIN
0008 = PROGBAR  EQU 8     ;B3--PROGRAM MODE PIN
0010 = BITOUT   EQU 10H   ;B4--PROGRAMMABLE OUTPUT
0020 = READY    EQU 20H   ;B5--BUSY/READY PIN
0040 = DIRECT   EQU 40H   ;B6--DIRECTION INDICATOR PIN
0080 = SLEW     EQU 80H   ;B7--SLEW INDICATOR PIN
            ;
00EF = A4CNTRL  EQU 0EFH  ;CY525 INPUTS RUN FROM PORT C
            ;
0000 = IOREQ    EQU 0     ;C0--LOW I/O REQUEST
0001 = NOIOREQ  EQU 1     ;C0--HIGH FOR NO REQUEST
            ;
0002 = IOSELIN  EQU 2     ;C1--LOW FOR COMMAND INPUT
0003 = IOSELOUT EQU 3     ;C1--I/O SELECT HI FOR VERIFY
            ;
0004 = DOWHILE  EQU 4     ;C2--LOW DOWHILE TO LOOP
0005 = NOLOOP   EQU 5     ;C2--HIGH FOR NO LOOP
            ;
            ;
            ;
            ;
0008 = ABORT    EQU 8     ;C4--LOW FOR ABORT
0009 = NOABORT  EQU 9     ;C4--HIGH FOR NORMAL STEPPING
            ;
000A = TRIGGER  EQU 0AH   ;C5--LOW TO ALLOW STEPPING
000B = STPINH   EQU 0BH   ;C5--HIGH STEP INHIBIT
            ;
000C = LOWAIT   EQU 0CH   ;C6--LOW ON WAIT PIN
000D = HIWAIT   EQU 0DH   ;C7--HIGH ON WAIT PIN
            ;
000E = RESETLO  EQU 0EH   ;C7--HARDWARE RESET ON LOW
000F = NORESET  EQU 0FH   ;C7--HIGH TO RUN CY525
            ;
            ;
000D = CR       EQU 0DH   ;ASCII CARRIAGE RETURN CODE
            ;
```

PORT B (0EDH):
- B_0 — MOTION COMPLETE
- B_1 — PULSE
- B_2 — RUN
- B_3 — PROG MODE
- B_4 — PROG OUTPUT
- B_5 — BUSY/READY
- B_6 — DIRECTION
- B_7 — SLEW INDICATOR

PORT C (0EEH):
- C_0 — I/O REQUEST
- C_1 — I/O SELECT
- C_2 — DO WHILE
- C_3 —
- C_4 — ABORT
- C_5 — STEP INHIBIT
- C_6 — WAIT
- C_7 — RESET

PORT A (0ECH):
- A_0 — DB_0
- A_1 — DB_1
- A_2 — DB_2
- A_3 — DB_3
- A_4 — DB_4
- A_5 — DB_5
- A_6 — DB_6
- A_7 — DB_7

PORT B (0F5H):
- B_0 — PASS (GREEN)
- B_1 — FAIL (RED)
- B_2 — TEST IN PROGRESS
- B_3 —
- B_4 — ASCII/BIN
- B_5 —
- B_6 —
- B_7 —

8255 PIO

BINARY DATA PROGRAMMING EXAMPLE

The binary data mode is illustrated by the programs and timing diagrams that follow:

```
                    ;
                    TESTBINARY:
        00A8  11C300  LXI     D,BINBUFFER
        00AB  CDD700  CALL    SENDPARALLEL
                    ;
                    RDYERROR:
        00AE  DBED    IN      STATUS
        00B0  E620    ANI     READY
        00B2  C2E803  JNZ     ERROR     ;FALSE READY
                    ;
                    TSTINTREQ1:
        00B5  DBED    IN      STATUS
        00B7  E601    ANI     MOTION
        00B9  CAAE00  JZ      RDYERROR
                    ;
        00BC  3E01    MVI     A,GREEN
        00BE  D3F5    OUT     LEDS
        00C0  C3A800  JMP     TESTBINARY
                    ;
                    ;
                    BINBUFFER:
        00C3  4300    DB  'C',0        ;CLEAR CY525 PIN 34
        00C5  4200    DB  'B',0        ;SET PIN 34 HIGH
        00C7  520164  DB  'R',1,100    ;SET RATE = 100
        00CA  5301FE  DB  'S',1,254    ;SET SLOPE = 254
        00CD  460103  DB  'F',1,3      ;SET FIRSTRATE = 3
        00D0  4E020500 DB 'N',2,5,0    ;SET 5 STEPS
        00D4  4700    DB  'G',0        ;GO FOR 5 STEPS
        00D6  FF      DB  0FFH         ;STOPPER
                    ;
```

In the command mode, the BUSY/RDY output remains low after a GO command is received until the CY525 finishes the last of the "N" steps specified. This is indicated by the END-of-MOTION (INTREQ1) output (pin 37). The RDY line returns high approximately 30 microseconds after the INTREQ1 goes low. INTREQ1 rises when the next command is sent to the CY525.
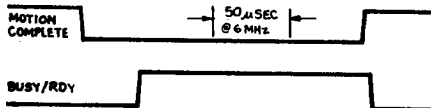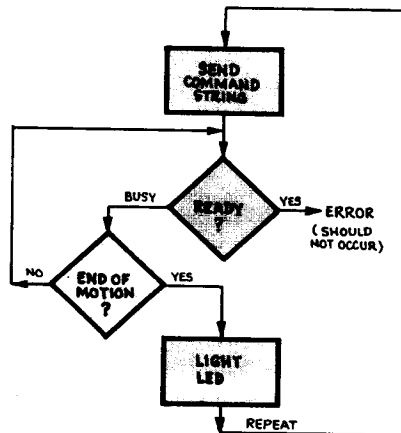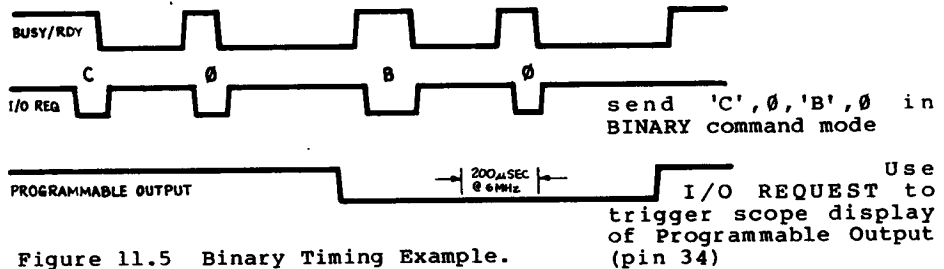

Figure 11.4   End-of-Motion Timing.

send 'C',Ø,'B',Ø in BINARY command mode

Use I/O REQUEST to trigger scope display of Programmable Output (pin 34)

Figure 11.5 Binary Timing Example.

HANDSHAKE SUBROUTINE

The parallel ASCII data is sent to the CY525 Stepper Motor Controller using the 8080 SENDPARALLEL code shown below. In this system the I/O REQUEST strobe is generated via a separate programmable control line and is removed after the data is acknowledged by the CY525.

```
                ;
                SENDPARALLEL:   ;ROUTINE TO SEND COMMAND BYTES TO CY525
                                ;DE = POINTER TO BYTE STRING, 0FFH IS STOPPER
00D7 1A         LDAX    D               ;GET NEXT BYTE FROM BUFFER
00D8 FEFF       CPI     0FFH            ;IS IT STOPPER?
00DA C8         RZ                      ;RETURN IF STOPPER, ALL BYTES SENT
00DB 13         INX     D               ;UPDATE POINTER
00DC 4F         MOV     C,A
00DD CDE300     CALL    SENDCHAR        ;SEND THIS BYTE TO CY525
00E0 C3D700     JMP     SENDPARALLEL    ;KEEP LOOPING TIL STOPPER
                ;
                ;
                SENDCHAR:       ;OUTPUT CHAR IN C TO CY525
00E3 DBED       IN      STATUS
00E5 E620       ANI     READY           ;LOW IF BUSY
00E7 CAE300     JZ      SENDCHAR        ;WAIT FOR READY
                ;
00EA 79         MOV     A,C
00EB D3EC       OUT     DATA            ;PUT CHAR ON DATA BUS
                ;
00ED 3E00       MVI     A,IOREQ
00EF D3EF       OUT     A4CNTRL         ;LOWER I/O REQUEST, DATA IS AVAIL
                ;
                WAITBSY:
00F1 DBED       IN      STATUS
00F3 E620       ANI     READY
00F5 C2F100     JNZ     WAITBSY         ;WAIT FOR BUSY (CY525 GOT DATA)
                ;
00F8 3E01       MVI     A,NOIOREQ
00FA D3EF       OUT     A4CNTRL         ;RAISE I/O REQUEST
00FC C9         RET
                ;
```

Figure 11.6 Example command output subroutine.

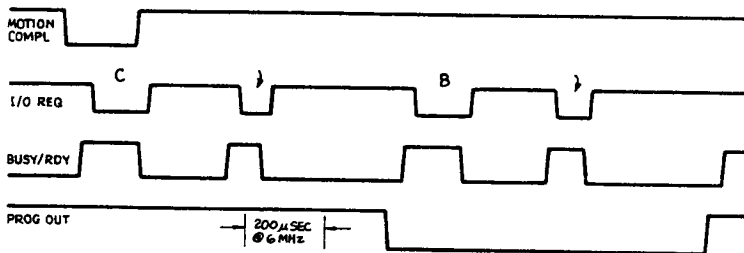# ASCII DATA PROGRAMMING EXAMPLE

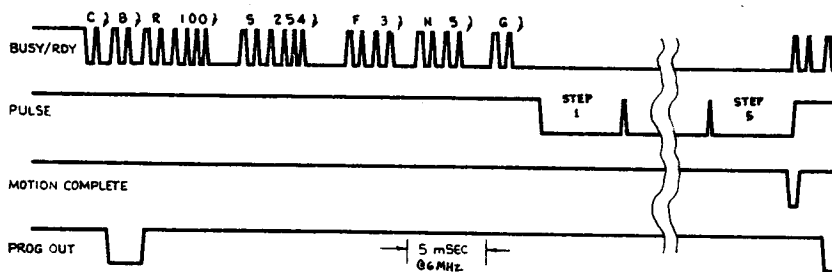
Figure 11.7   Expanded Handshake Timing Diagram.

Figure 11.8   Complete Timing for Sample Program.

```
              TESTASCII:      ;ASCII MODE
00FD 111801   LXI     D,ASCIIBUFFER
0100 CDD700   CALL    SENDPARALLEL    ;SEND THE COMMANDS IN THE BUFFER
              ;
              RDYLOOP:
0103 DBED     IN      STATUS
0105 E620     ANI     READY
0107 C2E803   JNZ     ERROR           ;SHOULD BE BUSY STEPPING
              ;
010A DBED     IN      STATUS
010C E601     ANI     MOTION
010E CA0301   JZ      RDYLOOP         ;WAIT FOR MOTION COMPLETE
              ;
0111 3E01     MVI     A,GREEN
0113 D3F5     OUT     LEDS            ;LIGHT GREEN LED
              ;
0115 C3FD00   JMP     TESTASCII       ;LOOP FOR SCOPE DISPLAY
              ;
              ASCIIBUFFER:    ;COMMAND STRING FOR CY525
0118 430D     DB 'C',CR       ;CLEAR PROGRAMMABLE OUTPUT (PIN 34)
011A 420D     DB 'B',CR       ;PIN 34 HIGH
011C 52203130 DB 'R 100',CR   ;SET RATE = 100
     300D
0122 53203235 DB 'S 254',CR   ;SET SLOPE = 254
     340D
0128 46203330D DB 'F 3',CR    ;SET FIRSTRATE = 3
012C 4E20350D DB 'N 5',CR     ;SET FOR 5 STEPS
0130 470D     DB 'G',CR       ;GO FOR 5 STEPS
0132 FF       DB 0FFH         ;STOPPER FOR SENDPARALLEL ROUTINE
              ;
```

Figure 11.9   Sample program.

117

OSCILLOSCOPE DISPLAY EXAMPLE
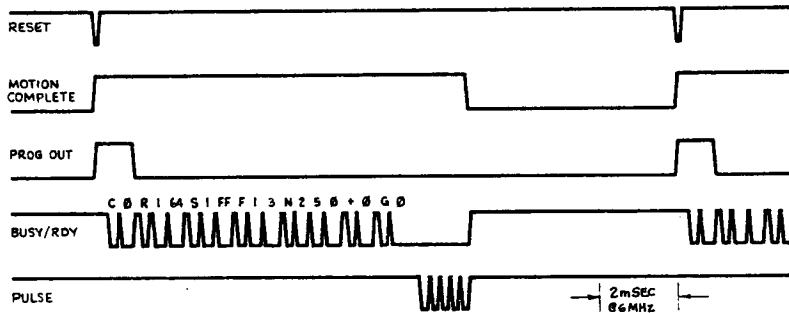

Figure 11.10   Timing for Program Shown Below.

The 8080 (or equivalent) sends the following commands and binary
data to the CY525:

```
    ---        reset CY525 using pin 4
'C' 0          clear programmable output (pin 34)
                   (used to trigger scope display)
'R' 1 64H  set rate parameter = 064H
'S' 1 FFH  set slope parameter = 0FFH
'F' 1 3    set First rate = 3
'N' 2 5 0  set number of steps = 5
'+' 0          set CW direction (redundant)
'G' 0          begin stepping
```

After sending the above commands, the host computer polls the
MOTION COMPLETE output (pin 37) and, upon finding it active,
after the 5th step has been taken, the host delays a fixed time
interval and then loops back, resets the CY525 and repeats this
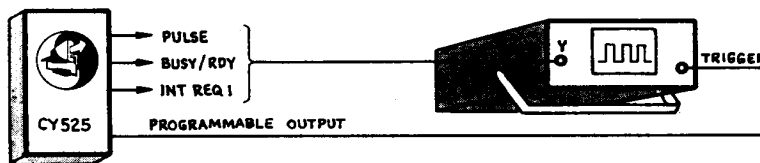process.   The programmable output may be used to trigger an
oscilloscope.

Figure 11.11   Test Setup.

118

IEEE-488 INTERFACE TO THE CY525

Using only a few SSI TTL gates, the CY525 can be made to work as a "LISTENER" on the IEEE-488 or GPIB (General Purpose Interface Bus). This section describes the timing and control involved in the GPIB interface and identifies the CY525 signal names with the appropriate GPIB signals. This implementation represents a simple GPIB interface. If a more complete bus interface is required, especially in a multi-instrument environment, the user should employ a separate GPIB interface device between the CY525 and the bus. This would allow the user to assign device addresses and communicate in both directions, using the CY525 Verify mode. A suitable GPIB interface device would be Fairchild's 96LS488.

GPIB HANDSHAKE SIGNALS

The "TALKER", or device desiring to send 8 bits of data to the CY525 over the data bus, uses the DAV (Data AVailable) signal that corresponds to the I/O REQUEST line on the CY525. Before lowering the DAV line, the TALKER must test the NRFD (Not Ready For Data) line. This line corresponds to the CY525 BUSY/RDY line. When this line is low, the LISTENER (CY525) is Not Ready For Data. When the TALKER finds the NRFD line high, it can assert (lower) its DAV write line to the CY525. Thus far, the interface is identical to the standard CY525 handshake. The third handshaking signal is an acknowledge line from the listener named NDAC (Not Data ACcepted). This line must initially be low and is raised to indicate that the data has been accepted by the CY525. The NDAC line is tested by the TALKER to determine whether or not the LISTENER has accepted the data. The CY525 BUSY/RDY line actually acknowledges the data transfer by going low, thus by **inverting** the RDY line, an NDAC signal can be generated. This completes the three line handshake required for the GPIB.
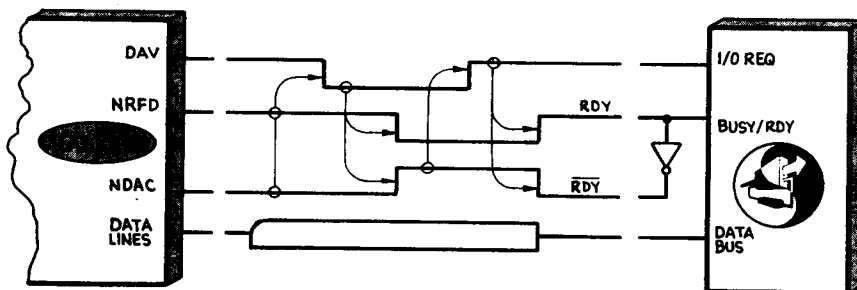

Figure 12.1   CY525/GPIB Interface.

DATA LINES

DAV (DATA AVAILABLE)

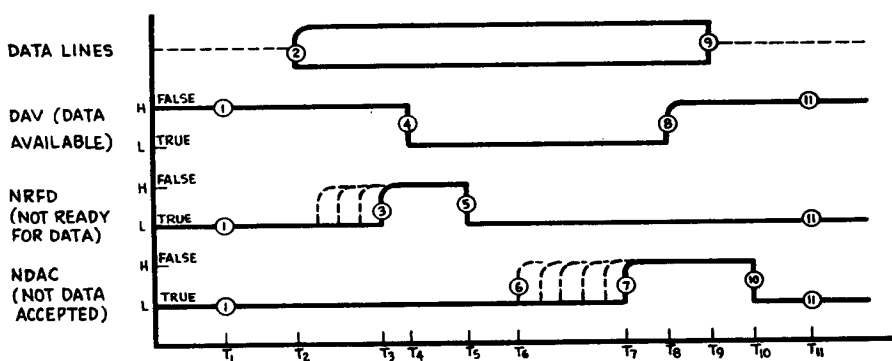NRFD (NOT READY FOR DATA)

NDAC (NOT DATA ACCEPTED)

Figure 12.2   GPIB Handshake Signals.

The flowchart for the TALKER that controls the CY525 is shown in Figure 12.3.   This procedure can be implemented simply using any microprocessor and describes the manner in which most GPIB interface devices function.
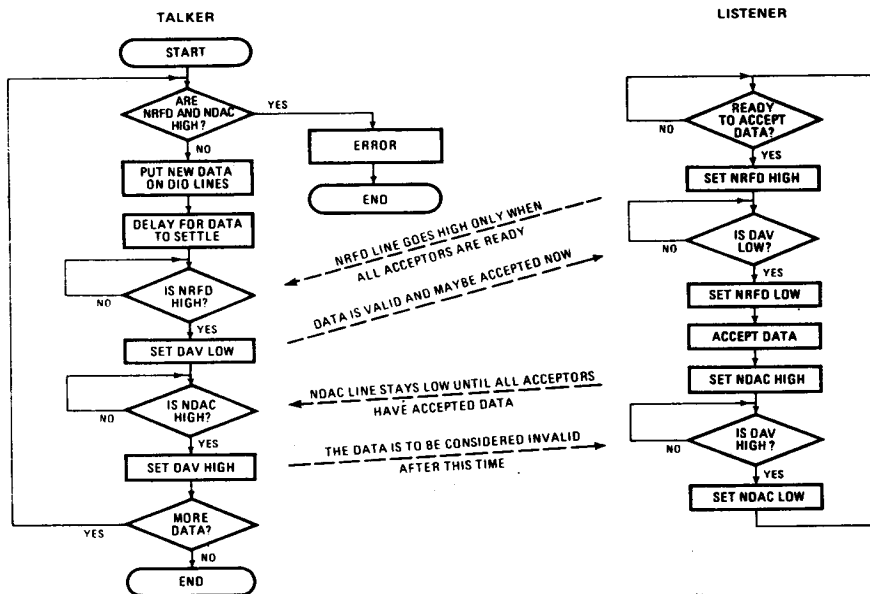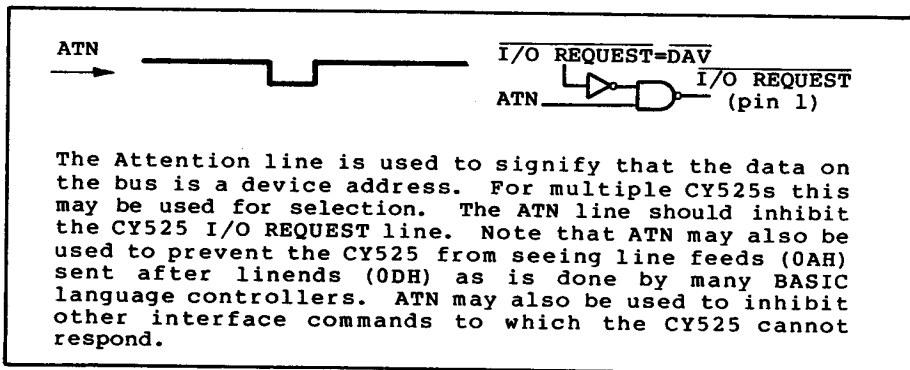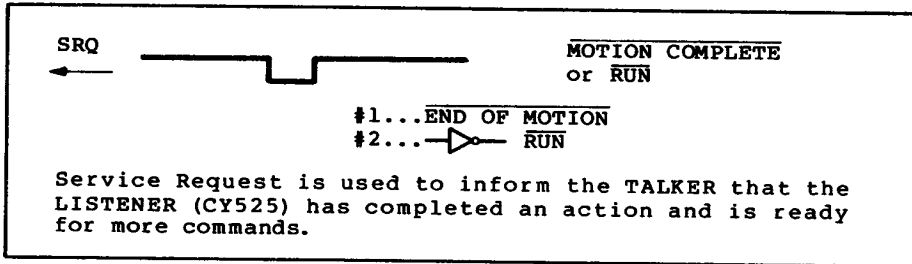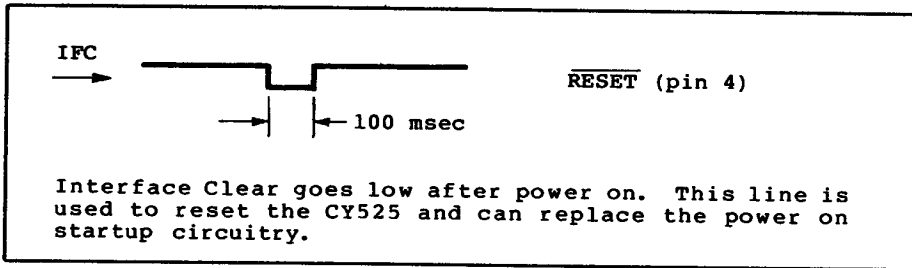

Figure 12.3   TALKER/LISTENER handshaking procedure.

120

GPIB INTERFACE MANAGEMENT SIGNALS

In addition to the three line handshake, there are several other
control lines defined by the IEEE-488 interface specifications.
These are described below and identified with appropriate CY525
signal lines.

IFC

$\overline{\text{RESET}}$ (pin 4)

100 msec

Interface Clear goes low after power on. This line is
used to reset the CY525 and can replace the power on
startup circuitry.

SRQ

$\overline{\text{MOTION COMPLETE}}$
or $\overline{\text{RUN}}$

#1...$\overline{\text{END OF MOTION}}$
#2...—▷o— $\overline{\text{RUN}}$

Service Request is used to inform the TALKER that the
LISTENER (CY525) has completed an action and is ready
for more commands.

ATN

$\overline{\text{I/O REQUEST}} = \overline{\text{DAV}}$

$\overline{\text{I/O REQUEST}}$
ATN _____ (pin 1)

The Attention line is used to signify that the data on
the bus is a device address. For multiple CY525s this
may be used for selection. The ATN line should inhibit
the CY525 I/O REQUEST line. Note that ATN may also be
used to prevent the CY525 from seeing line feeds (0AH)
sent after linends (0DH) as is done by many BASIC
language controllers. ATN may also be used to inhibit
other interface commands to which the CY525 cannot
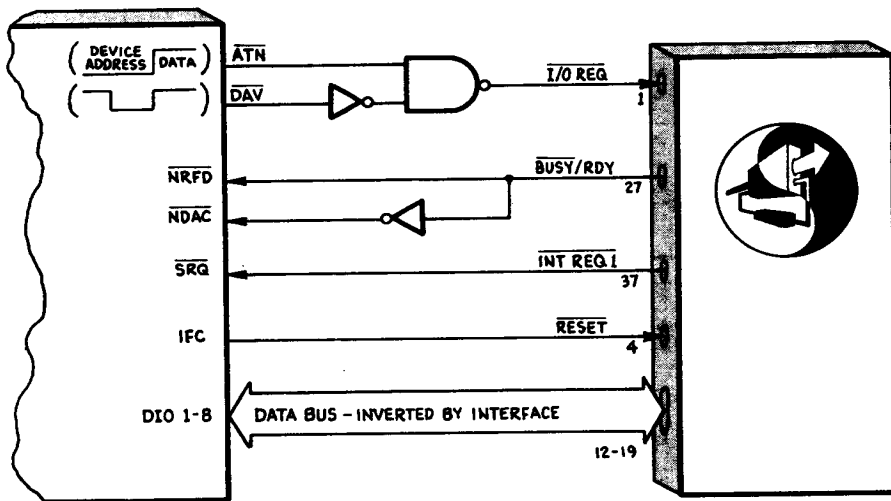respond.

Figure 12.4   Simple IEEE-488/CY525 Interface.

In some systems the REN (Remote ENable) and EOI (End Or Identify)
IEEE-488 control signals may be useful. For further information
on the IEEE-488 interface the reader is referred to the following
references:

        IEEE STANDARD 488 - 1978
                available from
                IEEE Service Center
                445 Hoes Lane
                Piscataway NJ   08854   USA

        PET and the IEEE 488 BUS
                by Fisher and Jensen, 1980
                Osborne/McGraw Hill
                630 Bancroft Way
                Berkeley CA   94710   USA

GPIB SCHEMATIC EXAMPLE

The following pages illustrate the logic used in an actual project which connected the CY525 to the IEEE-488 bus, using the Fairchild 96LS488. The schematics indicate general data flow, handshake control logic for bidirectional data transfers, and interrupt logic to control stepping and detect when a limit has been reached. Such a design supports many functions of the GPIB, and allows several CY525s or other GPIB instruments to reside on the same bus. This design is included with the permission of Christopher R. Hansen of the Mayo Foundation.
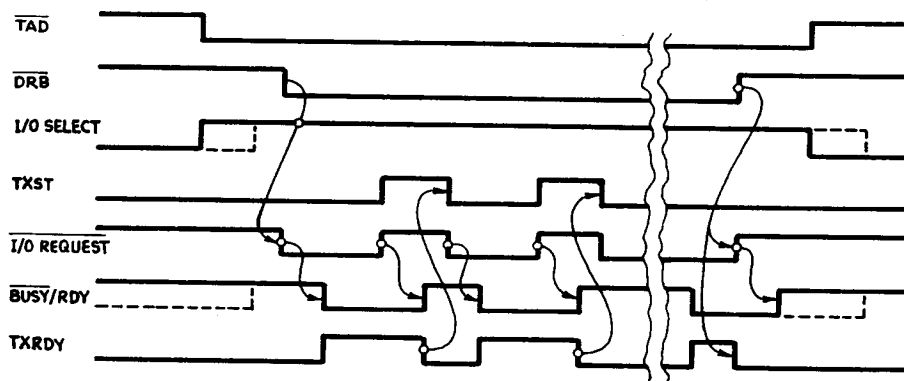
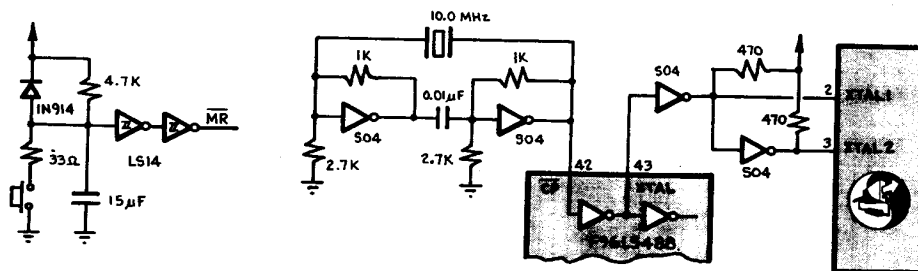Figure 12.5   Timing for Talker addressed.

Figure 12.6   Clock and Reset Logic

The figure above shows the implementation of the clock inputs for both the CY525 and the 96LS488 from a single crystal, and a manual reset. It also shows the timing for the handshake circuits when the CY525 is asked to output its values from a Verify command.
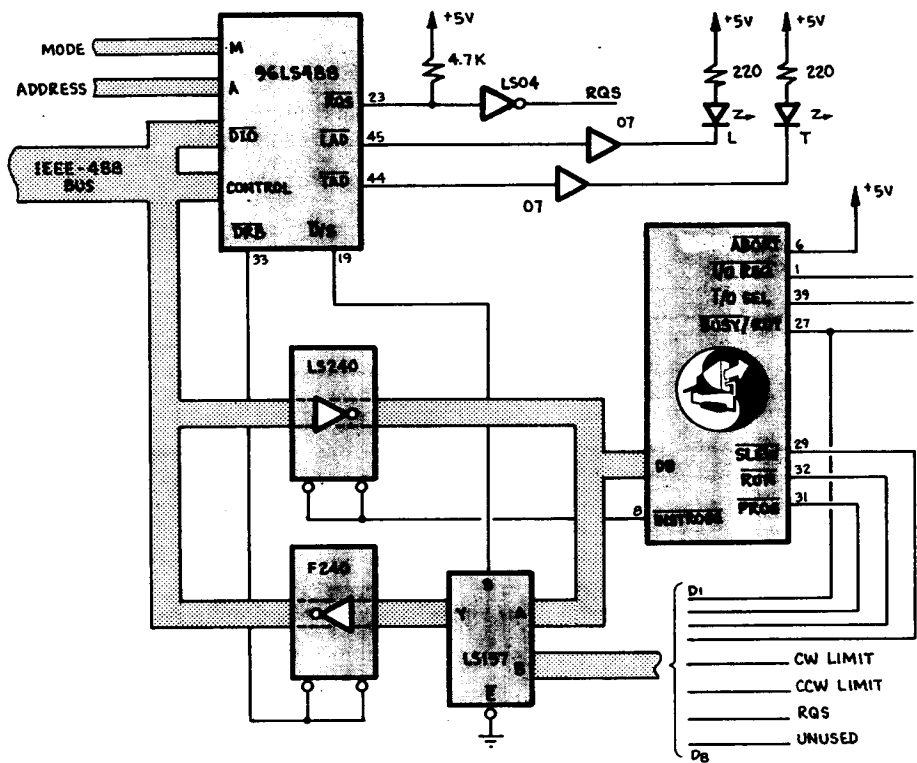
Figure 12.7   Data Paths

The data path schematic illustrates general data flow between the
GPIB, the 96LS488, and the CY525. All control signals from the
GPIB connect directly to the 96LS488, which interprets the
bus commands and controls the handshake logic to the CY525. The
eight data lines connect to the 96LS488, and through 74LS240
buffers to the CY525. The buffers invert the data signals
between the CY525, which uses positive logic, and the GPIB, which
uses negative logic. Two modes are used to read back information
from the CY525. To read the internal parameters, using the
Verify mode, a normal GPIB read operation is performed, by making
the 96LS488 and CY525 the bus talker. To read back the states of
various CY525 control lines, the 96LS488 is asked to perform a
status read as a result of a poll command. Note that any eight
of the CY525 signals may be connected to the status port.
Internal CY525 data and the status information are multiplexed by
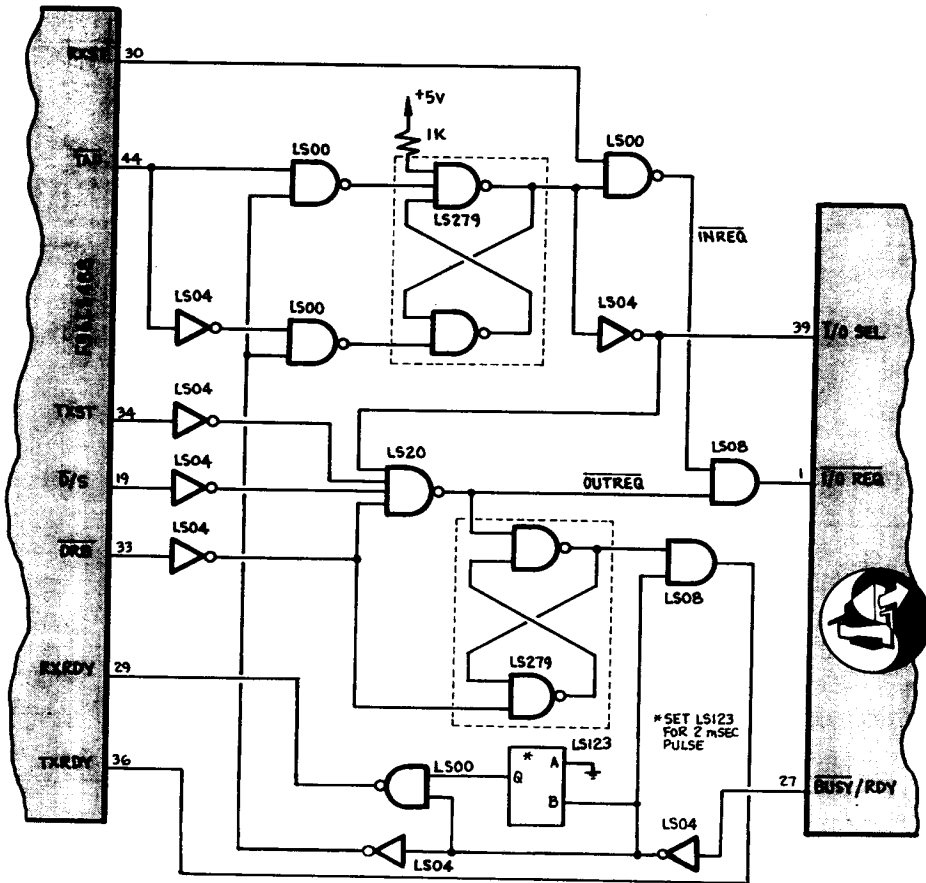the 74LS157.

124

Figure 12.8   Handshake Control Logic

The handshake control logic is the key to connecting the CY525 to the 96LS488. This logic converts the appropriate signals between the simple handshake of the CY525, and the more complex handshake performed by the 96LS488. By combining the various signals from the 96LS488, the proper values for I/O Select and I/O Request are generated. Much of this logic distinguishes between the Listen mode (sending commands to the CY525) and the Talk mode (sending data from the CY525). This enables the Verify command to be used with the GPIB design. The CY525 Busy/Ready signal is used to complete the handshake between the CY525 and the 96LS488.
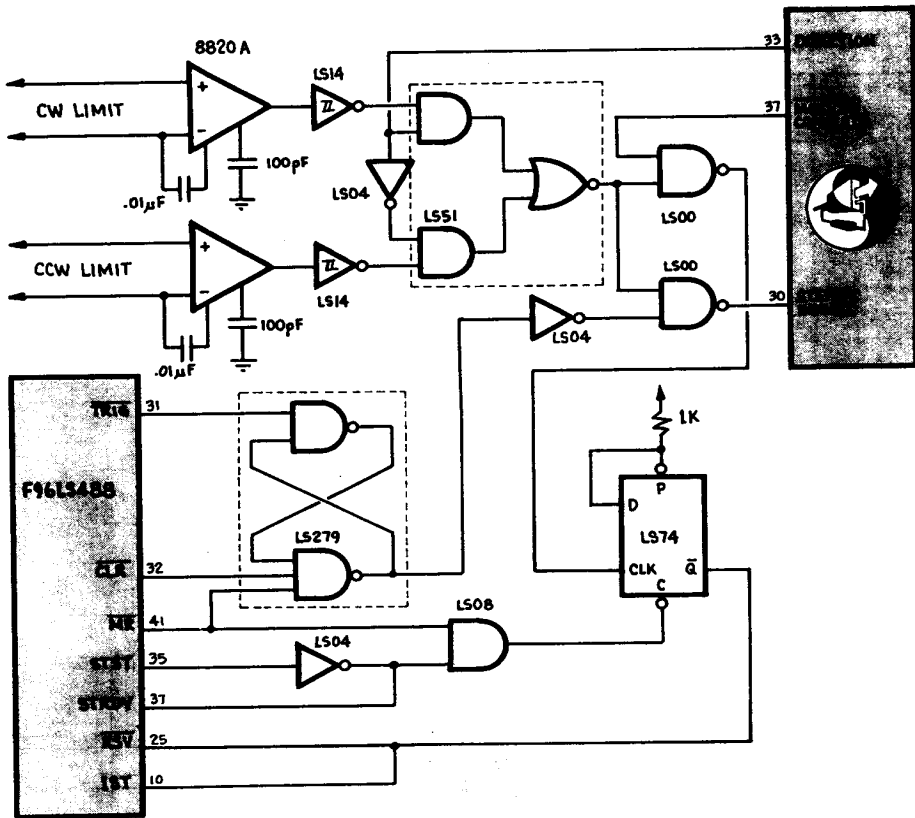
Figure 12.9   Interrupt Logic

The interrupt logic illustrates two functions.   In the first, the
GPIB may be used to control the start of stepping.   The 96LS488
Trig, CLR, and MR signals are combined and connected to the CY525
Step Inhibit.   This allows the GPIB master controller to stop any
stepping operations by issuing a Device Clear command.   Stepping
may also be synchronized to other events by allowing the master
to start the stepping using a Device Trigger command.   The second
function of the interrupt logic is to monitor the stepping
operation, and warn the master controller when a limit has been
reached.   By monitoring the limits, a Service Request can be
generated when a limit is reached.   The Step Inhibit is also
controlled by this process, keeping the CY525 from inadvertently
stepping too far.   Service Requests are also generated by the
CY525 Motion Complete signal.

126

13 GETTING YOUR CY525 RUNNING 13

The following checklist will simplify getting your CY525 up and running.

1. Connect pins 7 & 20 to ground and pins 26 & 40 to +5 volts.

2. Be sure that pin 39 is low and pin 6 is high.

3. Set pin 36 high (ASCII mode), set pin 30 low for now.

4. Be sure RESET (pin 4) is low for at least 10 milliseconds after power stabilizes. The CY525 can be reset at any time.

5. Upon proper reset all outputs should be at logic 1 (>3V).

6. Observe the RDY line (pin 27) to be sure it is high.

7. Observe CLK/15 (pin 11 ⎍⎍⎍ 400 KHz with 6 MHz Xtal).

8. Place the "CLEARBIT" command "C" (=43H) on the data bus.

              DB0 = 1
              DB1 = 1
              DB2 = 0
              DB3 = 0
              DB4 = 0
              DB5 = 0
              DB6 = 1
              DB7 = 0

9. Lower the I/O REQUEST line (pin 1).

10. Wait for RDY (pin 27) to go low before bringing I/O REQUEST high. If using I/O REQUEST strobe circuitry that generates a low write signal when an ASCII character is placed on the bus, be sure that your software detects **low** RDY line (Busy) before looking for High RDY. If you are using a debounced keyboard this should not be a problem.

11. When I/O REQUEST is brought back high, RDY will return high.

12. Wait for RDY to return high before placing the RETURN code (♪=0DH) on the data bus.

        pin 12... DB0 = 1
                  DB1 = 0
                  DB2 = 1
                  DB3 = 1
                  DB4 = 0
                  DB5 = 0
                  DB6 = 0
        pin 19... DB7 = 0

13. Generate the low I/O REQUEST strobe until RDY goes low, then return I/O REQUEST high, as before.

14. Upon completion of the above sequences of steps, the Programmable Output (pin 34) will go low.

15. Repeat steps 8 through 13, replacing "C" (=43H) with "B" (=42H). This "BITSET" command will cause the Programmable Output (pin 34) to return high. All other outputs (except RDY) should have remained high during the above procedure.

16. Repeat steps 8 through 13, replacing "C" (=43H) with "-" (=2DH). This is the "CCW" command. The result of this command will be to bring the DIRECTION Line (pin 33) low.

17. Following the CCW command with a CW command ("+" =2BH) will again raise the DIRECTION line.

18. If you have reached this point successfully you should be able to enter any of the commands and obtain the correct responses.

19. Suggested sequences:

    a. enter "E}" followed by "Q" and observe the PROG/LIVE (pin 31) go low with "E}" and return high with "Q".

    b. with STEP INHIBIT (pin 30) low, enter "A 0}" followed by "P 1}". The STEPPER MOTOR DRIVE SIGNALS, pins 21-24, will be activated, and PULSE, pin 35, will go low and return high, indicating the duration of the step. The drive signals will change from step to step as the above sequence is repeated.

    c. raise the STEP INHIBIT line and enter the single step sequence as in "b" above. Nothing will happen on PULSE, or the stepper control lines, until the STEP INHIBIT line is lowered.

    d. refer to Figures 10.13, 10.14, and 10.15. Enter these commands as listed and observe the outputs. Note that LEDs on the relevant outputs are very useful.

20. After initial checkout is accomplished using ASCII input, the user may place pin 36 low to select Binary. Read the manual carefully for differences in the two modes.
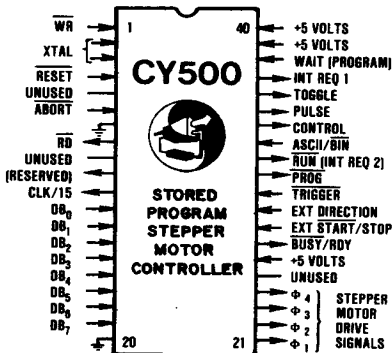
CY500

STORED PROGRAM
STEPPER MOTOR CONTROLLER

The CY500 stored program stepper motor controller is a standard 5 volt, 40 pin LSI device configured to control any 4-phase stepper motor. The CY500 will interface to any computer using asynchronous parallel TTL input and provides numerous TTL inputs and outputs for auxiliary control and interfacing. The CY500 allows programming with an ASCII keyboard for prototype development and allows sequences of hi-level type commands to be stored internally in a program buffer and be executed upon command. The TTL outputs sequence the stepper drive circuits that consist of standard power transistors or transistor arrays.

STANDARD FEATURES

- ASCII-DECIMAL OR BINARY COMMUNICATION
- SINGLE 5 VOLT POWER SUPPLY
- HI-LEVEL LANGUAGE COMMANDS
- STORED PROGRAM CAPABILITY
- HALF-STEP/FULL-STEP CAPABILITY
- ABSOLUTE/RELATIVE POSITION MODES
- PROGRAMMABLE VIA ASCII KEYBOARD
- 3300+ STEPS PER SECOND (6 MHz XTAL)
- PROGRAMMABLE OUTPUT LINE
- TWO INTERRUPT REQUEST OUTPUTS

- HARDWARE/SOFTWARE DIRECTION CONTROL
- HARDWARE/SOFTWARE START/STOP
- 'ABORT' CAPABILITY
- SINGLE/MULTIPLE STEP INSTRUCTIONS
- RAMP-UP/SLEW/RAMP-DOWN MODE
- 24 INSTRUCTIONS IN SET
- TRIGGERED OPERATION
- 'DO-WHILE' COMMAND
- 'WAIT-UNTIL' COMMAND
- SEVERAL SYNC INPUTS AND OUTPUTS

PIN CONFIGURATION

LOGIC DIAGRAM

Cybernetic Micro Systems
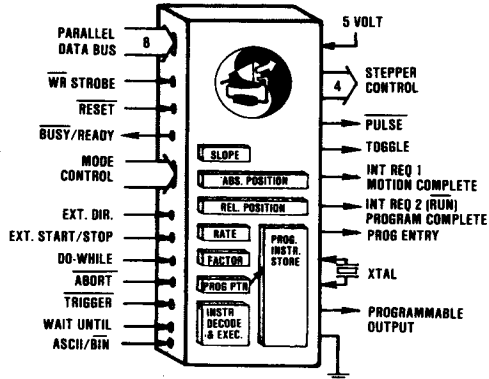
CY512

## INTELLIGENT POSITIONING STEPPER MOTOR CONTROLLER

WITH ASCII/BINARY POSITION READOUT & AUTOMATIC DIRECTION FINDING

The CY512 intelligent positioning stepper motor controller is a standard 5 volt, 40 pin LSI device configured to control any 4-phase stepper motor. The CY512 will interface to any computer using parallel TTL input and provides numerous TTL inputs and outputs for auxiliary control and interfacing. The CY512 allows sequences of hi-level type commands to be stored internally in a program buffer and be executed upon command. The TTL outputs sequence the stepper drive circuits that consist of standard power transistors or transistor arrays. When absolute position commands are executed, the CY512 automatically determines whether it is necessary to move CW or CCW to reach the specified target position.

STANDARD FEATURES

- ASCII-DECIMAL OR BINARY COMMUNICATION
- SINGLE 5 VOLT POWER SUPPLY
- 25 HI-LEVEL LANGUAGE COMMANDS
- STORED PROGRAM CAPABILITY
- HALF-STEP/FULL-STEP CAPABILITY
- ABSOLUTE/RELATIVE POSITION MODES
- PROGRAMMABLE VIA ASCII KEYBOARD
- 8000± STEPS PER SECOND (11 MHz XTAL)
- PROGRAMMABLE OUTPUT LINE
- TWO INTERRUPT REQUEST OUTPUTS
- MORE LINEAR RAMP THAN CY500
- HIGHER RATE RESOLUTION THAN CY500
- PROGRAMMABLE DELAY

- SOFTWARE DIRECTION CONTROL
- HARDWARE/SOFTWARE START/STOP
- 'ABORT' CAPABILITY
- AUTOMATIC DIRECTION DETERMINATION
- RAMP-UP/SLEW/RAMP-DOWN
- VERIFY REGISTER/BUFFER CONTENTS
- STEP INHIBIT OPERATION
- 'DO-WHILE' AND 'WAIT-UNTIL' COMMANDS
- 'JUMP TO' COMMAND
- SEVERAL SYNC INPUTS AND OUTPUTS
- 'SLEWING' INDICATION OUTPUT
- 'TERMINATE' STEP LINE FOR MAX ACCELERATION
- LOOP COMMAND WITH REPETITION COUNT

PIN CONFIGURATION

LOGIC DIAGRAM

Cybernetic Micro Systems

130
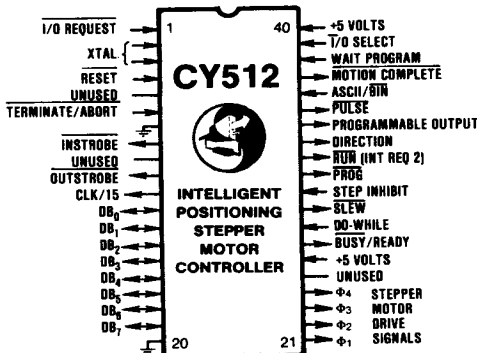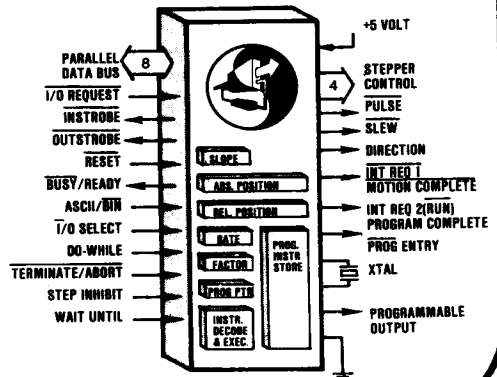
ASCII-DECIMAL TO HEX CONVERSION TABLE

DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	ASCII	HEX
0	00	51	33	102	66	153	99	204	CC	CR	0D
1	01	52	34	103	67	154	9A	205	CD	SP	20
2	02	53	35	104	68	155	9B	206	CE	+	2B
3	03	54	36	105	69	156	9C	207	CF	,	2C
4	04	55	37	106	6A	157	9D	208	D0	–	2D
5	05	56	38	107	6B	158	9E	209	D1		
6	06	57	39	108	6C	159	9F	210	D2	0	30
7	07	58	3A	109	6D	160	A0	211	D3	1	31
8	08	59	3B	110	6E	161	A1	212	D4	2	32
9	09	60	3C	111	6F	162	A2	213	D5	3	33
10	0A	61	3D	112	70	163	A3	214	D6	4	34
11	0B	62	3E	113	71	164	A4	215	D7	5	35
12	0C	63	3F	114	72	165	A5	216	D8	6	36
13	0D	64	40	115	73	166	A6	217	D9	7	37
14	0E	65	41	116	74	167	A7	218	DA	8	38
15	0F	66	42	117	75	168	A8	219	DB	9	39
16	10	67	43	118	76	169	A9	220	DC		
17	11	68	44	119	77	170	AA	221	DD	A	41
18	12	69	45	120	78	171	AB	222	DE	B	42
19	13	70	46	121	79	172	AC	223	DF	C	43
20	14	71	47	122	7A	173	AD	224	E0	D	44
21	15	72	48	123	7B	174	AE	225	E1	E	45
22	16	73	49	124	7C	175	AF	226	E2		
23	17	74	4A	125	7D	176	B0	227	E3	F	46
24	18	75	4B	126	7E	177	B1	228	E4	G	47
25	19	76	4C	127	7F	178	B2	229	E5	H	48
26	1A	77	4D	128	80	179	B3	230	E6	I	49
27	1B	78	4E	129	81	180	B4	231	E7	J	4A
28	1C	79	4F	130	82	181	B5	232	E8		
29	1D	80	50	131	83	182	B6	233	E9	K	4B
30	1E	81	51	132	84	183	B7	234	EA	L	4C
31	1F	82	52	133	85	184	B8	235	EB	M	4D
32	20	83	53	134	86	185	B9	236	EC	N	4E
33	21	84	54	135	87	186	BA	237	ED	O	4F
34	22	85	55	136	88	187	BB	238	EE		
35	23	86	56	137	89	188	BC	239	EF	P	50
36	24	87	57	138	8A	189	BD	240	F0	Q	51
37	25	88	58	139	8B	190	BE	241	F1	R	52
38	26	89	59	140	8C	191	BF	242	F2	S	53
39	27	90	5A	141	8D	192	C0	243	F3	T	54
40	28	91	5B	142	8E	193	C1	244	F4		
41	29	92	5C	143	8F	194	C2	245	F5	U	55
42	2A	93	5D	144	90	195	C3	246	F6	V	56
43	2B	94	5E	145	91	196	C4	247	F7	W	57
44	2C	95	5F	146	92	197	C5	248	F8	X	58
45	2D	96	60	147	93	198	C6	249	F9	Y	59
46	2E	97	61	148	94	199	C7	250	FA	Z	5A
47	2F	98	62	149	95	200	C8	251	FB		
48	30	99	63	150	96	201	C9	252	FC		
49	31	100	64	151	97	202	CA	253	FD		
50	32	101	65	152	98	203	CB	254	FE		
								255	FF		