CYBERNETIC • MICRO SYSTEMS •

CY550

High Performance Stepper System Controller

CY550MAN.022 15 SEP 98 EEK KDG KM MA PRINTED IN U.S.A.

Cybernetic Micro Systems, Inc. software products are copyrighted by and shall remain the property of Cybernetic Micro Systems, Inc. Duplication is subject to a license from Cybernetics. Cybernetic Micro Systems, Inc. reserves the right to improve design or performance characteristics. Cybernetic Micro Systems, Inc. assumes no responsility for the use of any circuitry other than circuitry embodied in Cybernetic products. No other circuit patent licenses are implied.

Information furnished by Cybernetic Micro Systems, Inc. is believed to be accurate and reliable. However, no responsibility is assumed by Cybernetic Micro Systems, Inc. for its use, nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent right of Cybernetic Micro Systems, Inc. Further, Cybernetic Micro Systems, Inc. reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation to notify any person or organization of such revision or changes; and Cybernetics assumes no responsibility for any errors which may appear in this document and makes no commitment to update the information contained herein.

Because of the wide variety of applications in which this product may be used, Cybernetic Micro Systems makes no claim as to the product's fitness for any given application. It is therefore the user's responsibility to ensure the safety and viability of using this product in his application. This product is not recommended for use in critical life sustaining applications.

The following are trademarks of Cybernetic Micro Systems, Inc:

Bin-ASCII
Analog-ASCII
ASCII-Analog
CY232
CY233-LINC
CY250
CY300
CY325
CY327
CY360
CY480
CY500
CY512
CY525
CY545
CY550
CY600
CY750

Cybernetic Micro Systems, Inc.

P.O. Box 3000
San Gregorio, California 94074
Tel: (650) 726-3000
Fax: (650) 726-3003
www.ControlChips.com
info@ControlChips.com

Table of Contents

SECTION 1
Introduction to the CY550

SECTION 2
Overview of Pin Functions

SECTION 3
Command Interfaces

SECTION 4
Command Summary

SECTION 5
Motor Control Commands

SECTION 6
Bit Function Commands

SECTION 7
External Memory Support Commands

SECTION 8
Extended I/O Support

SECTION 9
Program Branch Commands

SECTION 10
Operating Mode Command

SECTION 11
Miscellaneous Commands

SECTION 12
CY550 Example Circuits

SECTION 13
External Memory and Extended I/O Support

SECTION 14
CY550 Output Display Support

SECTION 15
CY550 Prototyping Support

SECTION 16
Timing and Control

SECTION 17
CY550 Step Rate Information

SECTION 18
Electrical Specifications

SECTION 19
Circuits and Examples

SECTION 20
Getting your CY550 Running

CY550 Stepper System Controller

Over a decade ago (1979) Cybernetic Micro Systems invented the first Single Chip Stepper Motor Controller, the **CY500**. Since then we have continued to design improved versions: the **CY512** (1981), the **CY525** (1983), and in 1988 we introduced our **fourth generation device, the CY545 Stepper System Controller.** Unlike all currently available Stepper Motor Controllers, the CY545 was offered as a **Stepper Motor System Controller**, offering completely new features such as:

- **27,000 Steps/Sec !**
- **16 million step motions !**
- **Serial or Parallel interface**
- **LED display interface**
- **LCD display interface**
- **Thumbwheel switch interface**

In 1993 we bring you our **fifth generation device, the CY550 High Performance Stepper System Controller**. Based on technology similar to the CY545, the CY550 offers enhancements and additional features, such as:

- **Live command interfaces during stepping**
- **On-the-fly motion profiling**
- **Enhanced acceleration characteristics**
- **Enhanced Serial interface**
- **Signed absolute positions, +8 million to -8 million**
- **Expanded, memory mapped, user I/O, up to 1/2 million bits !**

Since the CY550 is designed for use in an interactive environment, it does not support thumbwheel switches or a direct LCD display interface, although the LCD display functions can be mimicked by display strings. The special LED display interface is still supported. Also, the maximum step rate is about 19,900 steps/sec with a 16 MHz clock.

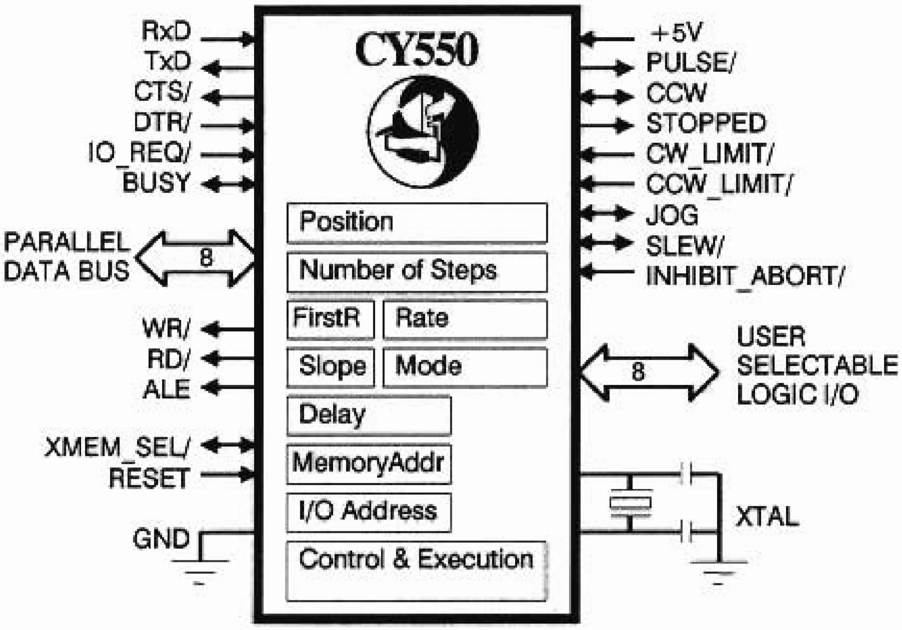
The CY550 can be used to implement very sophisticated, highly interactive motion control systems, at a very modest cost.

The live command interfaces allow a host computer to interact with the CY550 even while it is stepping. It is possible to make on-the-fly queries of current position or status, step rate changes, I/O bit manipulations, and to perform many other functions!

Alternatively, the external memory support of the CY550 allows you to design a stand-alone system which still retains the live features of the interactive system! The CY550 can continue to execute commands from the local memory while it is stepping, making stand-alone, system-interactive control possible without additional processors.

Basic Features

- Single 40 pin 5 volt CMOS integrated circuit.
- Simple ASCII letter based commands and parameters.
- Simple application for basic stepper motor functions.
- Operation with a host computer or stand alone.
- Live command interface during stepping for interactive control functions.
- Built-in Display interface for alpha-numeric LED



Motion Control Functions

- Programmable step rates from 15 steps/sec to 19,900 steps/second provide an extremely large dynamic range.
- Linear acceleration curves, with smoothing, for optimum performance.
- Partial accelerations make short moves in the least time.
- Separate parameters for starting rate, final rate, and acceleration values allow totally programmable motions.
- Relative moves of +/– 16 Mega steps from current position. Absolute moves within a +8 to –8 Mega step range.
- Continuous moves with no specific number of steps, allow acceleration from the starting rate to the final rate, followed by an indefinite run length. On-the-fly stepping parameter changes are possible during continuous moves.
- Internal step position updated during all motions.

Motor Support Signals

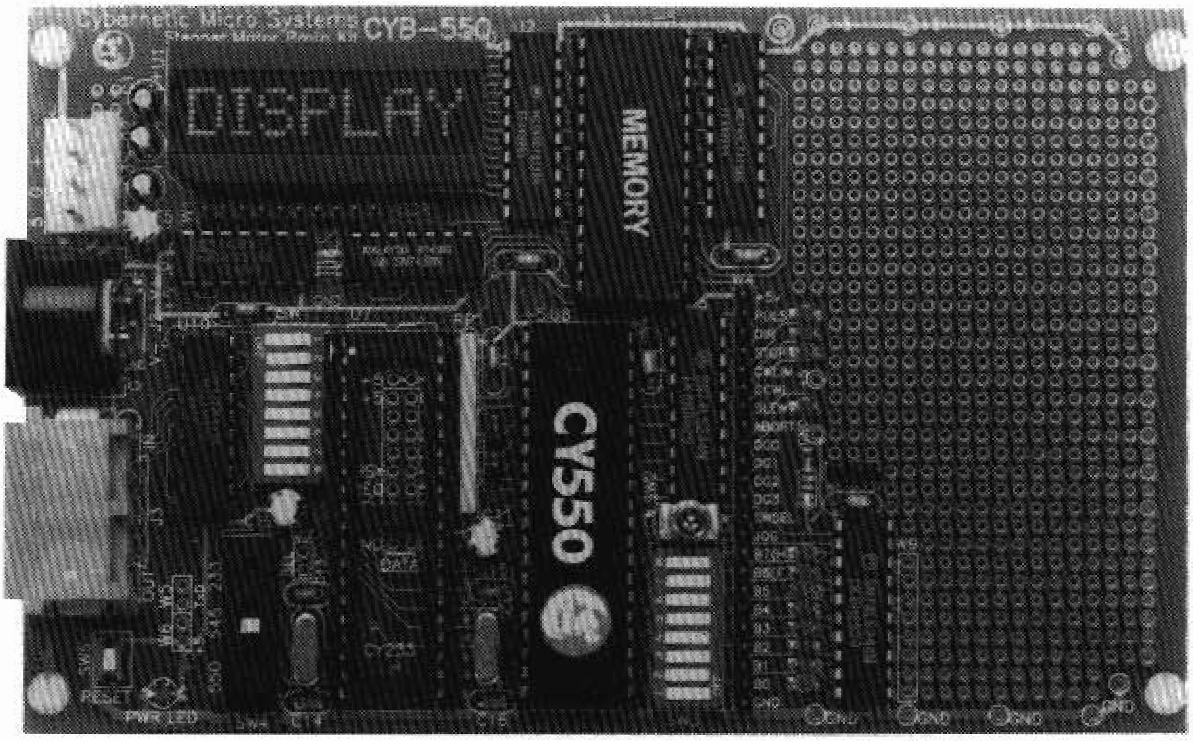
- Step Pulse and Direction signals work with standard power driver modules.
- Internal or external direction control.
- Forced ramp down and abort signal for emergency or externally controlled end of motion.
- Separate CW and CCW limit signals inhibit stepping beyond one limit, but allow stepping in the opposite direction.
- Moving/not moving signal for use as Motion Complete or to switch stepper power driver between high and park power.
- Externally operated jog mode, with direction control and start/stop control from one signal, all at programmable manual control step rates between 1 and 250K steps/sec.
- Automatic home sensor seek, with backlash compensation.

General Command Functions and Special Features

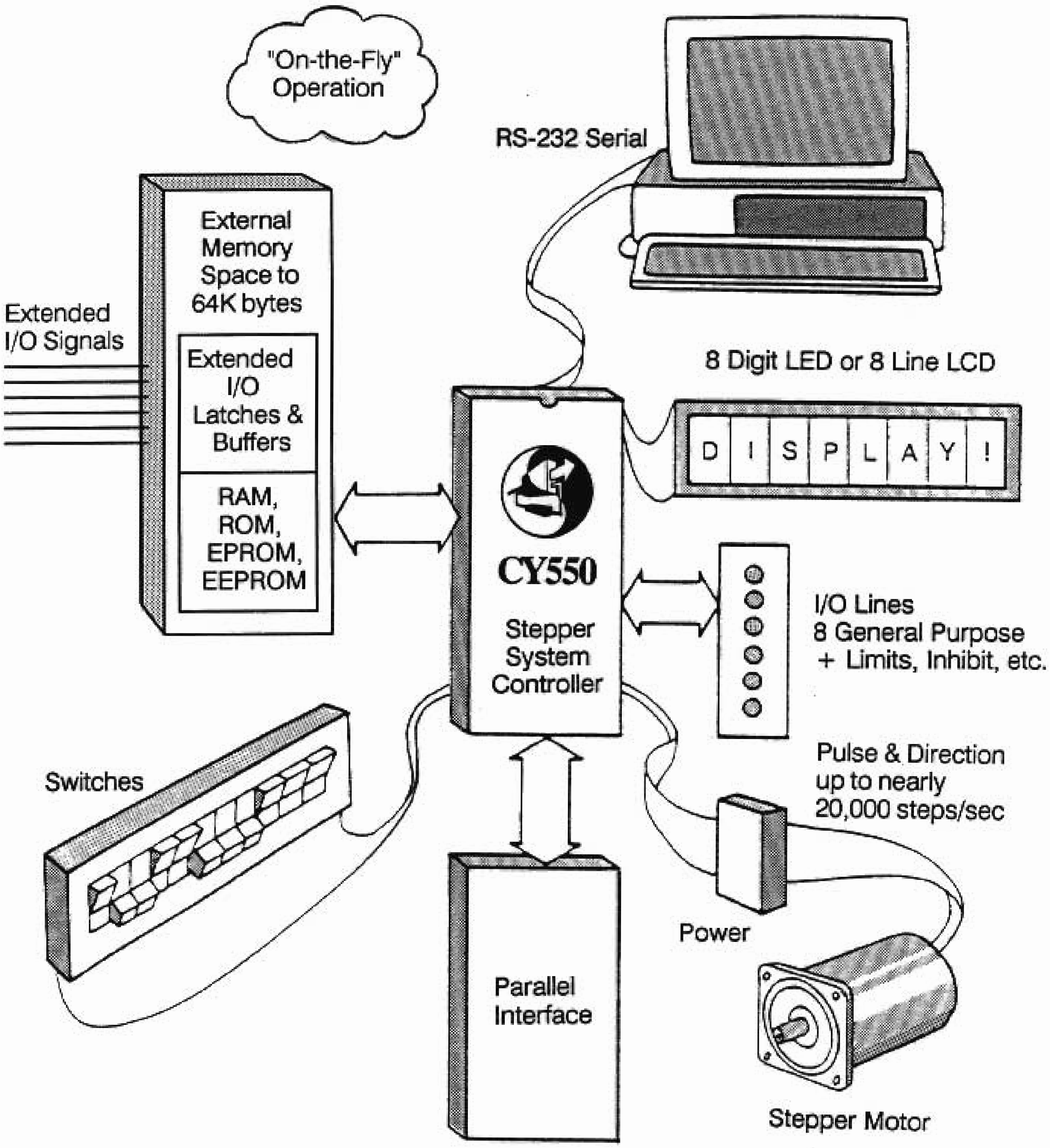
- Parallel command interface with two line handshake, compatible with CY233 network communications controller.
- Enhanced Serial command interface with fixed or adaptive baud rate selection, and in-band or handshake flow control.
- Support for up to 64K of optional external command memory implemented as RAM, ROM, EPROM, or EEPROM.
- Selectable display for output messages from standard serial, standard parallel, or CY233 interfaces, plus special support for a parallel HP HDSP-211x LED display.
- User controlled, multi-purpose I/O lines, for:
  - set and clear bit functions
  - test bit and branch functions
  - wait for bit value functions
  - automatic home seek functions
- Extended I/O line functions or logic flags in memory mapped 64K external memory space.

Prototyping Board Support

The CYB-550 Prototyping board implements most of the available features of the CY550 Stepper System Controller. A more complete description may be found in the chapter on Prototyping Support.



CY550 Stepper Motor System Controller



2 Overview of Pin Functions 2

CY550 Pin Description

Pin		Mnemonic	Function
1	(O)	**PULSE/**	Step pulse output, one pulse per step
2	(I/O)	**CCW**	Step direction

These pins run the stepper power driver. A high level on the direction signal indicates counter clockwise direction, while a low level indicates clockwise direction. The Pulse signal is normally high, going low at the beginning of each step.

3	(O)	**STOPPED**	Motion status, low while stepping

This signal may be used to indicate motion complete, by its high level. It may also control power selection for the stepper driver, switching between high power for stepping, and parking power while stopped. This signal pulses with each step during Jog and Home functions.

4	(I)	**CW_LIMIT/**	Clockwise step limit reached
5	(I)	**CCW_LIMIT/**	Counter Clockwise step limit reached

If either signal is low, it indicates a limit has been reached. The CY550 will not continue stepping in that direction until the limit signal is removed. Normal stepping is allowed in the opposite direction.

6	(I/O)	**JOG**	Manual stepping control

This signal is the manual stepping control input. When it is floating, the CY550 will not jog. However, if the signal is connected to ground, the CY550 will jog in the CW direction, and if it is connected to +5 volts, the CY550 will jog in the CCW direction. The jog step rate is derived from the First Rate parameter and the CY550 Rate Table, by dividing the selected rate by 20. This gives a range of about 1 step per second to about 250 steps per second. The input signal is always enabled, and is tested by the CY550 while it is not executing another command.

7	(I/O)	**SLEW/**	Slew indicator signal
8	(I/O)	**INHIBIT_ABORT/**	External motion control

This signal group indicates and controls motion status. **SLEW/** indicates when the maximum selected step rate has been achieved. If externally driven low at the beginning of a motion, it also selects the continuous step mode. **INHIBIT_ABORT/** can hold off motion at the start or force an early down ramp, then stop motion when the CY550 has ramped down to the starting rate.

CY550 Pin Description (cont.)

Pin	Mnemonic	Function
9 (I)	RESET	CY550 hardware reset, active high pulse
10 (I)	RxD	Received serial data into CY550
11 (O)	TxD	Transmitted serial data from CY550

These are TTL level serial data signals that can be used to issue commands to or get replies from the CY550. Addition of an RS-232 voltage level driver/receiver, such as the Maxim MAX233, allows any RS-232 device, such as the IBM PC COM1 port to talk directly to the CY550.

12 (O)	CTS/	Clear To Send

This signal is active low when the CY550 is ready to receive characters into the RxD line. If hardware flow control is being used, the host system should stop transmitting when **CTS/** goes high.

13 (I)	IO_REQUEST/	Parallel handshake input signal

This signal is driven low when a parallel command is issued to the CY550. It is used as part of the parallel command handshake. If the parallel command function is not used, this signal may select a fixed serial baud rate as follows:
0 = 300 baud, 1 = 2400 baud, F = 9600 baud

14 (I/O)	XMEM_SEL/	External local memory select

This signal goes low when the CY550 reads or writes to the local external memory space. Data and addresses are generated by the parallel data bus. Up to 64K bytes of external memory are supported. When this signal is driven low at power-up or reset, the CY550 will not test for the Auto Start key sequence or automatically execute commands from the external memory. This signal is also used in selecting the extended, memory-mapped I/O functions. This line should be tied low if no memory is used or if multiple CY550s share a common parallel data bus.

15 (I/O)	BUSY/	Parallel handshake acknowledgement

BUSY/ is used with **IO_REQUEST/** to implement the two line parallel command handshake. This signal indicates that the CY550 has accepted a parallel data character from the device driving the **IO_REQUEST** signal. If the parallel handshake is not used, this pin may be tied low to select fixed baud rates through the **IO_REQUEST** signal.

CY550 Pin Description (cont.)

Pin	Mnemonic	Function
16 (O)	**WR/**	Write strobe
17 (O)	**RD/**	Read strobe

These strobes are active while the CY550 is reading from the external display, or memory, or while it is writing to the external display or memory.

Pin	Mnemonic	Function
18 (I)	**XTAL2**	Crystal connection
19 (I)	**XTAL1**	Crystal connection

An external crystal or clock source is connected to these pins, with a value between 3.5 MHz and 16 MHz. For serial communications, an 11 MHz crystal will give standard baud rates.

Pin	Mnemonic	Function
20 (I)	**VSS**	Power supply ground.
21 (I/O)	**USRB0**	User selectable function, bit 0
22 (I/O)	**USRB1**	User selectable function, bit 1
23 (I/O)	**USRB2**	User selectable function, bit 2
24 (I/O)	**USRB3**	User selectable function, bit 3
25 (I/O)	**USRB4**	User selectable function, bit 4
26 (I/O)	**USRB5**	User selectable function, bit 5
27 (I/O)	**USRB6**	User selectable function, bit 6 or FPL/ or DTR/
28 (I/O)	**USRB7**	User selectable function, bit 7 or HP_SEL/

This group of signals is used for multiple, user selectable functions, including bit set and clear, bit test and branch, wait for signal value, and auto home functions. Any function may be performed on any bit. Bits 6 and 7 also have fixed alternative functions of CY233 FPL control or serial Data Terminal Ready, and local parallel HP display selection, if these features are used with the CY550. If the application does not use a CY233, DTR, or HP display, these bits may be used for any other functions.

CY550 Pin Description (cont.)

Pin	Mnemonic	Function
29 (O)	**RESERVED**	Reserved signal, not used by CY550
30 (O)	**ALE**	Address latch enable

This signal is used to demultiplex the lower byte address from the data bus during external memory, display, or extended I/O operations.

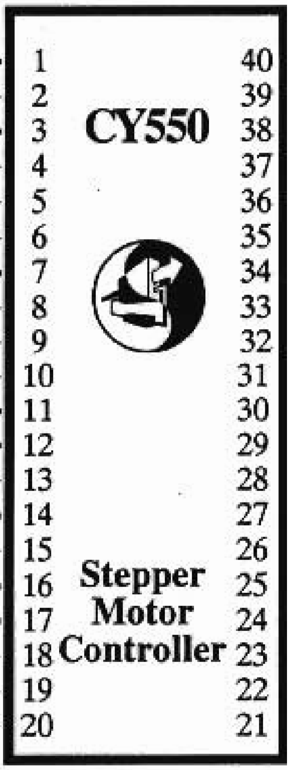
31 (I)	**TEST**	Internal test signal, connect to Vcc.
32 (I/O)	**D7**	Parallel data bus, bit 7, MSB
33 (I/O)	**D6**	Parallel data bus, bit 6
34 (I/O)	**D5**	Parallel data bus, bit 5
35 (I/O)	**D4**	Parallel data bus, bit 4
36 (I/O)	**D3**	Parallel data bus, bit 3
37 (I/O)	**D2**	Parallel data bus, bit 2
38 (I/O)	**D1**	Parallel data bus, bit 1
39 (I/O)	**D0**	Parallel data bus, bit 0, LSB

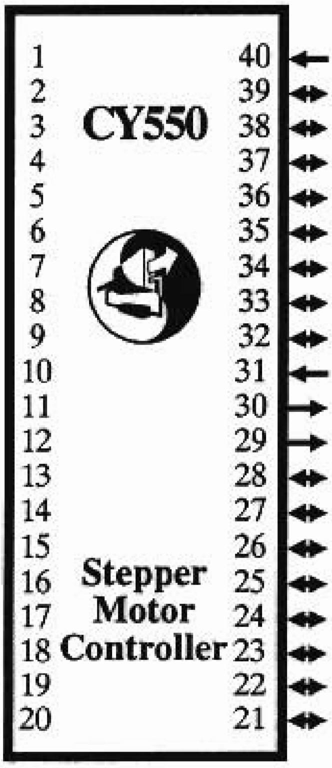
This group is the parallel data bus, used to read or write parallel data. During access to external memory, extended I/O, or display, the lower byte of the device address is presented on these lines just prior to the data transfer. Parallel commands are also accepted by the CY550 on these lines, using the handshake control signals. These lines are open drain, so external pull-up resistors are required when they are used as outputs.

40 (I)	**VCC**	+ 5 Volt power supply input.

CY550 Pin Configuration

Step pulse output, one pulse per step	PULSE/	←	1
Step direction	CCW	↔	2
Motion Status, low while stepping	STOPPED	←	3
Clockwise step limit reached	CW_LIMIT/	→	4
Counter Clockwise step limit reached	CCW_LIMIT/	→	5
Manual stepping control	JOG	↔	6
Slew indicator signal	SLEW/	←	7
External motion control	INHIB_ABORT/	→	8
CY550 hardware reset, active high pulse	RESET	→	9
Received serial data into CY550	RxD	→	10
Transmitted serial data from CY550	TxD	←	11
Serial Clear to Send, active low	CTS/	←	12
Parallel handshake input signal	IO_REQ/	→	13
External local memory select	XMEM_SEL/	←	14
Parallel handshake acknowledgement	BUSY/	↔	15
Write strobe	WR/	←	16
Read strobe	RD/	←	17
Crystal connection	XTAL2	→	18
Crystal or external clock circuit	XTAL1	→	19
Power supply common	VSS	→	20





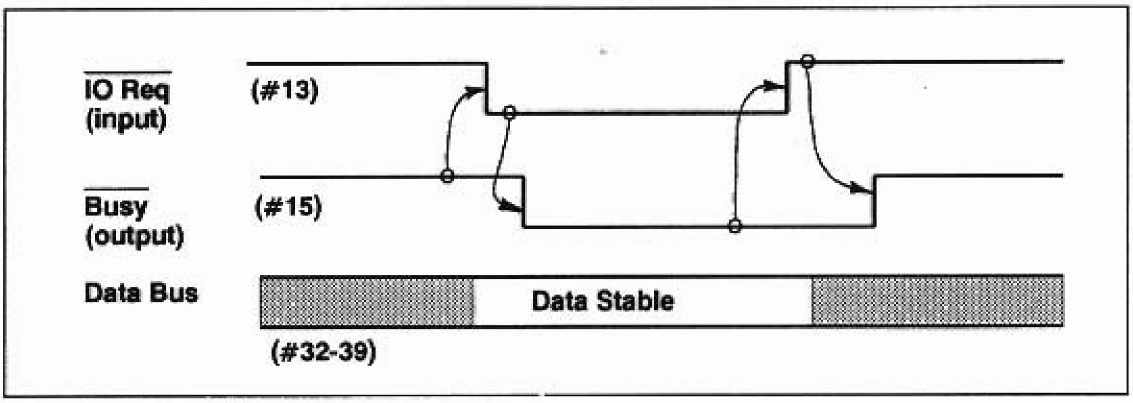
40	←	VCC	+5 volt power supply input
39	↔	D0	Parallel data bus, bit 0, LSB
38	↔	D1	Parallel data bus, bit 1
37	↔	D2	Parallel data bus, bit 2
36	↔	D3	Parallel data bus, bit 3
35	↔	D4	Parallel data bus, bit 4
34	↔	D5	Parallel data bus, bit 5
33	↔	D6	Parallel data bus, bit 6
32	↔	D7	Parallel data bus, bit 7, MSB
31	←	TEST	Internal test signal, connect to Vcc
30	→	ALE	Address latch enable
29	→	RESERVED	Reserved signal, not used by CY550
28	↔	USERB7	User selectable function, bit 7 or HP_SEL/
27	↔	USERB6	User selectable function, bit 6 or FPL/ or DTR/
26	↔	USERB5	User selectable function, bit 5
25	↔	USERB4	User selectable function, bit 4
24	↔	USERB3	User selectable function, bit 3
23	↔	USERB2	User selectable function, bit 2
22	↔	USERB1	User selectable function, bit 1
21	↔	USERB0	User selectable function, bit 0

Command Interfaces

The CY550 supports two basic command interfaces, a parallel interface and a serial interface. These signals are similar to functions on other CYxxx controllers from Cybernetic Micro Systems.

Parallel Interface

The parallel interface uses two handshake lines, IO_REQUEST and BUSY. When you wish to send a command character to the CY550, you first check that it is not busy; the BUSY signal should be high. You then place the character to send on the parallel data bus lines. Be sure to drive all 8 data lines. Next, you drive the IO_REQUEST signal low, indicating that a character is available. Now, wait for the CY550 to go busy, with a low level on the BUSY signal, indicating that the CY550 has read the command character. Finally, you remove the character from the data bus and drive the IO_REQUEST line high again.



$\overline{\text{IO Req}}$ (input)
(#13)
$\overline{\text{Busy}}$ (output)
(#15)
Data Bus
Data Stable
(#32-39)

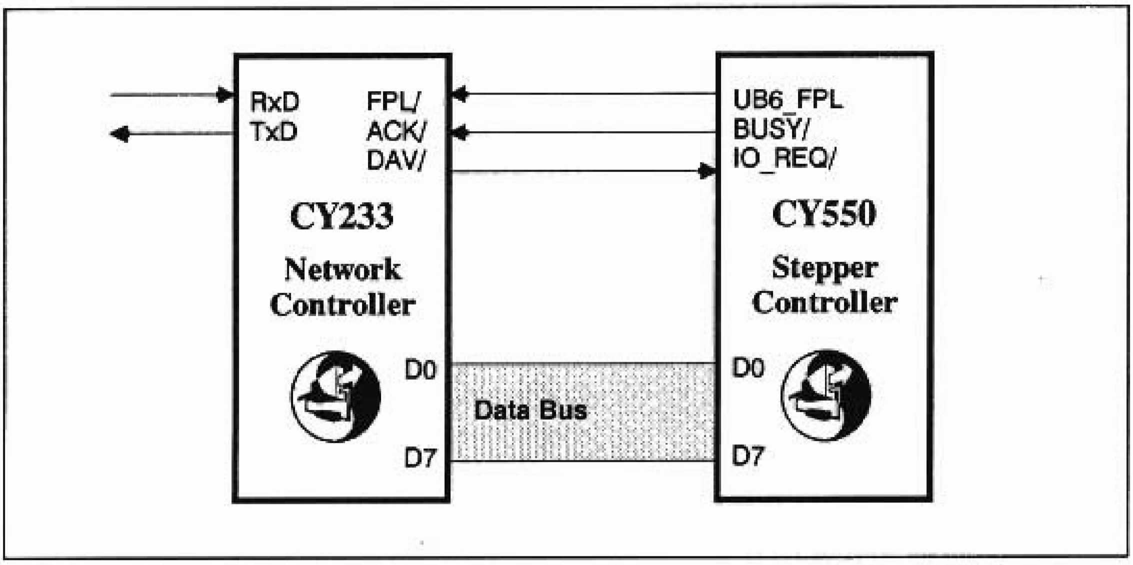
This function is repeated until all command characters have been issued to the CY550. In direct command mode, the CY550 executes every command as it is received. This means that the CY550 will go busy for a longer period of time with the last character of a command, while the command is actually executed. For long time delays or some other commands, this time could be many seconds.

Since the CY550 retains a live command interface during stepping, it will be busy momentarily at the beginning of a motion, long enough to perform some internal setup and actually begin stepping. It will then indicate ready again, with BUSY going high. At this point, you may issue more commands to the CY550, even though it is still stepping the motor!

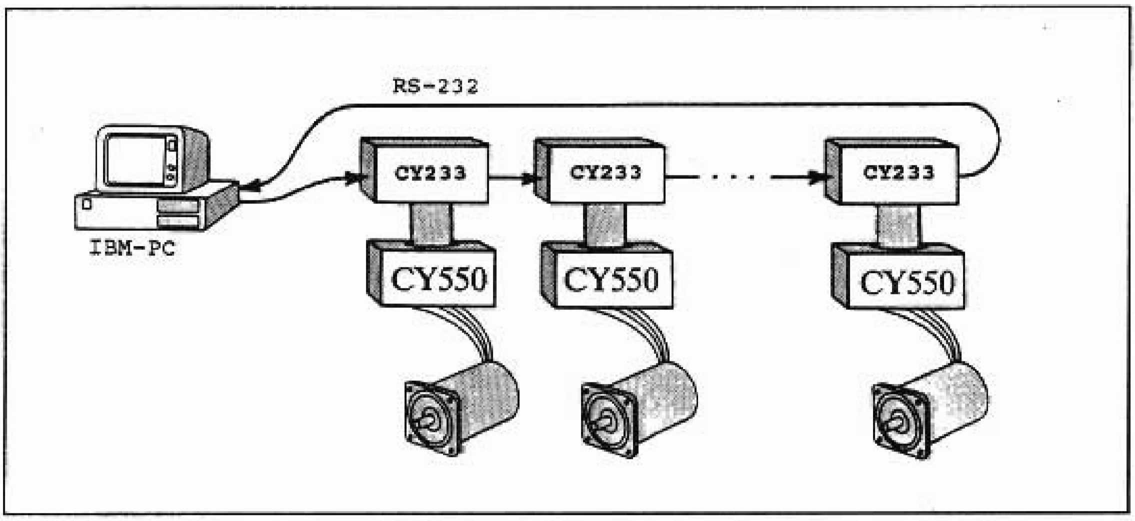
This parallel interface is compatible to that of other CYxxx control chips. It is also compatible with the CY233 Network Controller, so you can command several CY550s from a computer serial port, using a CY233 as a network "front end" to each CY550.

CY550 Interface to CY233 Network Control Chip

For a CY233 based interface, the DAV signal from the CY233 is connected to the IO_REQUEST signal of the CY550, and the ACK signal of the CY233 is connected to the BUSY signal of the CY550. In addition, the FPL signal of the CY233 is connected to USRB6 of the CY550, which shares the FPL function on this line. An example applications schematic, provided with this manual, shows the connections. The relevant portion is reproduced below.



The CY233 Network Control chip allows you to connect up to 255 devices to a single RS-232 serial communications line such as an IBM-PC COM1 or COM2 port, with unique addresses assigned to each device. The CY233 Network is ideal for distributed systems where central control is required but very high speed communication is unnecessary.



The CY550 Serial Interface

The CY550 also provides a direct serial command interface, which may be connected to a host computer or terminal. Since the CY550 signals are all TTL voltage levels, external RS-232 line drivers and receivers must be provided, to translate the RS-232 voltage levels to the CY550 TTL levels. A Maxim MAX-233 does this function in a single chip, operating at +5 volts. This interface is also shown on the applications schematic.

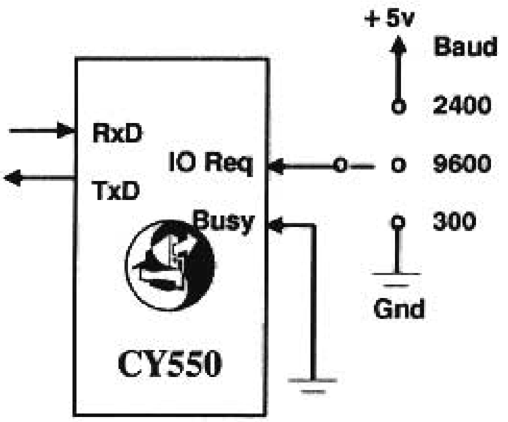
The serial interface may be operated in one of two ways, with a fixed baud rate, selected at power up, or with an adaptive baud rate, selected by two carriage return characters from the host.

The fixed baud rate mode is selected by tying the BUSY line low, so the CY550 will read it as a zero value on power up. The CY550 IO_REQUEST line value will determine the baud rate as follows:

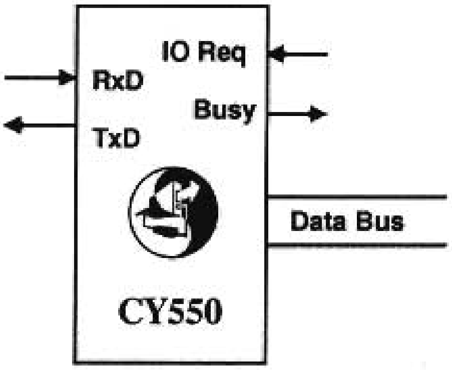
F	**9600**
1	**2400**
0	**300**

When the IO_REQUEST line is left floating, 9600 baud is selected. When tied high, 2400 baud is selected, and when tied low, 300 baud is selected. For these fixed rates to be correct, the CY550 must be operated with an 11 MHz crystal clock. Other values will scale the baud rate linearly by the difference in crystal frequencies.

Fixed Serial Baud Rate
(No parallel interface)



Adaptive Serial Baud Rate
(plus parallel interface)



An alternative method of selecting the baud rate is through a parallel Mode command. For this command, 9 baud rate selections are available, from 300 to 57600, plus the self adaptive rate. Details of the Mode command are described in a later section.

Note that the selection of fixed baud rate mode through the BUSY and IO_REQUEST lines, gives up the use of the parallel handshake interface, since the lines are used to select the baud rate. This is fine if only the serial interface will be used. However, if both interfaces must be used on the same CY550, the baud rate may only be set adaptively, or through a parallel Mode command.

The adaptive serial mode is chosen by default, when the BUSY signal is left floating, so the CY550 can drive it at power up. In this mode, the serial baud rate is not set until the CY550 receives two carriage return codes. Be sure to send these characters after power up or any reset (hardware or software). Once the two carriage returns have been received, normal CY550 commands may be sent.

When working with the serial interface, the best crystal frequency for the CY550 is 11 MHz. This will provide the largest baud rate operating range, with the least baud rate error. Internal timer resolution of the CY550 will limit the available rates, especially at the high end. At 11 MHz, standard rates up to 57600 baud may be selected. Note that other CY550 functions, such as the step rates and time delays, have been calibrated for a 12 MHz clock source. This means these functions will run somewhat slower when 11 MHz is used. Changes to the relevant parameter values can compensate for some of this difference. Alternatively, a 12 MHz clock will work fine, if the selected adaptive baud rates are always 4800 baud or less.

When operating the CY550 at 16 MHz, internal timer resolution still limits the reliable operation of the serial interface to 4800 baud or less. The error at higher baud rates may be too great for consistent communications.

However, there is a crystal frequency of 14.7456 MHz at which the CY550 can be operated in the adaptive mode. With this frequency, the internal resolution of the CY550 baud rate timer allows it to adapt to all standard baud rates from 300 to 19,200 baud, plus two higher, non-standard rates of 38,400 and 76,800 baud.

Also, the CY550 serial interface works on a fixed format character, which is always 8 data bits, with no parity bit, and one stop bit. All 8 data bits are used to interpret the command character values, so be sure to send the proper 8 bit codes for the various ASCII command characters.

Finally, the CY550 implements two optional flow control functions, hardware signal or character based. The hardware flow control uses the Clear to Send signal (CTS/) and the optional Data Terminal Ready (DTR/), which are used by the CY550 and host system to indicate when characters may be sent or not.

No additional characters should be sent to the CY550 while CTS/ is off (TTL high). Once the CY550 has processed characters from its internal serial buffer, it will bring CTS/ low again, indicating that it is ready for more serial data. The CTS/ signal is always available from the CY550. The polarity of the CY550 CTS/ signal allows it to be connected directly to your host system CTS RS-232 signal, through a standard RS-232 line driver. No additional logic is required.

As an option, selected by a Mode command, the CY550 can also use a Data Terminal Ready signal (DTR/). This function is shared with User Bit 6, which is also the CY233 FPL/ signal. Since the signal has multiple functions, the DTR/ function is disabled at power on or any restart. It is enabled when any Mode command is processed, with MBit 1 set.

When DTR/ is enabled, the signal is driven by the host, and a TTL low level at the CY550 indicates that the host is able to accept serial data, similar to the CTS/ low indicating that the CY550 is able to accept serial data. The CY550 will transmit any serial data as long as DTR/ is low. If DTR/ goes high, the CY550 will stop sending serial data until it goes low again. Note that the CY550 will not process any other commands while it is waiting for DTR/ to go low again.

The CY550 also implements an optional character based flow control, using the X-ON (11h) and X-OFF (13h) ASCII control characters. This function is enabled by a Mode command with MBit 3 set. This "in band" flow control is enabled for both directions.

When first enabled, the CY550 assumes that transmission and reception are enabled in both directions. When one side wants the other to stop transmitting, it sends an X-OFF character. The other side should now suspend transmissions until an X-ON character is sent, allowing transmissions to resume.

4 Command Summary 4

CY550 Command Summary

Cmd	Parameter	Function	Page
A	Pos24	At position, sets current step position	5-3
B	Bit	Bit set or clear of user selectable bits	6-1
C		Set Continuous step mode	5-3
D	Del16	Delay for specified milliseconds	11-2
E		Enter following commands to external memory	7-1
F	Rate	Specify First step rate	5-1
G		Go step, relative mode	5-4
H	Bit	Seek Home, using specified bit	6-5
I		Initialize CY550, perform software reset	11-1
J	Addr	Jump to byte address of current memory page	9-1
K	Addr16	Set input address pointer for extended I/O reads	8-4
L	Cnt,Addr	Loop to byte address for specified count	9-1
M	Addr16	Set output address pointer for extended I/O writes	8-4
N	Num24	Set Number of steps for relative motions	5-3
O	Mode16	Set Operating mode of CY550	10-1
P	Pos24	Step to specified absolute Position	5-4
Q		Quit entering commands to external memory	7-1
R	Rate16	Specify slewing step Rate	5-1
S	Slope	Specify acceleration Slope value	5-1
T	Bit,Addr	Loop to address Til bit matches value	6-3
U		Reserved command	
V		Wait for current motion to finish	5-6
W	Bit	Wait for specified bit to match value	6-3
X		eXecute external memory commands	7-2
Y	Addr16	Set external memory address pointer	7-1
Z	Cnt16,Addr	ZillionLoop to byte address for 16 bit count	9-1
+		Select CW direction for relative motions	5-3
-		Select CCW direction for relative motions	5-3
/		Negate prefix used with Bit commands	6-1
?	Cmd	Query specified command parameter value	11-3
0		Stop execution of commands from memory	7-2
\	Num24	Wait for the specified number of steps	5-6
]	Pos24	Wait for the specifed absolute position	5-6
^		Stop the current motion	5-6
#	Value	Set the value of the I/O byte register	8-4
!		Perform extended I/O read	8-6
%		Perform extended I/O write	8-6
[	Addr,Cnt,D1,...,Dn	Special HP Display support command	11-8
"String"		Display all characters between quotes	11-6

The preceding list of CY550 commands includes the command letter and argument structure. Arguments without a suffix should be single byte numeric values up to 255, while arguments with a "16" suffix may be two byte numbers up to 65535, and arguments with a "24" suffix may be three byte numbers up to 16777215. Commands are entered as the single upper case ASCII letter, followed by a space, and the argument value. Multiple arguments may be separated by a space or comma. The command ends with a carriage return character. Commands without arguments simply use the command letter, followed directly by the carriage return.

Command Formats

The CY550 supports several different command formats, which can be split into two major categories, **ASCII** format and **Binary** format. Most examples in this manual will illustrate ASCII formats, and we believe this will be the most popular mode of operation. The main advantage to Binary format is a savings in the number of characters needed to represent a parameter value, so Binary format commands will usually take fewer characters than ASCII format commands. However, Binary format commands are more difficult for humans to read, and if the CY550 and host computer become unsynchronized in Binary mode, due to a faulty data count, it is extremely difficult to resynchronize the two without a hardware reset. The Binary format requires all commands to be issued with exact byte counts and arguments, with no extraneous characters.

ASCII Command Format

ASCII format commands all start with the ASCII command letter that selects that command. If the command also has parameters, the command letter must be followed by a single space, then the parameter value. Multiple parameters, such as those used by the Loop command, may be separated by a space or comma. Our examples will generally use a comma. All ASCII commands end with the ASCII carriage return character, shown as "< cr >" in the following examples. The CY550 considers the carriage return to specify the end of the command, and will expect a new command letter following the carriage return.

Some host systems may automatically add a line feed character "< lf > " before or after the carriage return. This will not cause any problems, so long as you stay in the ASCII command mode. The CY550 will ignore any line feed characters during command processing. However, no line feed should be included in any mode command that switches the CY550 to Binary command mode, and once in Binary mode, all command bytes must be sent exactly as required by the command, with no extraneous characters.

Commands without parameters, such as the Go command, use only the command letter, followed immediately by the carriage return. For example:

```
G<cr>                   Go command, no parameter
I<cr>                   Initialize command, no parameter
```

Commands with parameters may use two formats for the parameter values in the ASCII mode, a decimal format, consisting of the digits 0 to 9, or a hex format, consisting of the digits 0 to 9, letters A to F, and an "H" suffix. Hex parameters must start with a digit. For example:

S 135<cr>	Slope command, single decimal parameter
S 87H<cr>	same parameter value, but in hex form
S ABH<cr>	Illegal parameter, no digit to start
S 0AB<cr>	Illegal parameter, no H suffix
S 0ABH<cr>	Parameter value ok
D 12345<cr>	16 bit parameter value in decimal
D 1A3BH<cr>	16 bit parameter value in hex
L 35,27<cr>	Loop command, multiple parameters
L 23H,27<cr>	Parameter formats may be mixed

Parameter values may be entered with any number of digits, but the command will limit the value to a single byte if the range of the parameter is only one byte. For example:

S 0ABCH<cr>	Value is taken as 0BCH = 188
S 350<cr>	Value is taken as 94 = 350 - 256

Sixteen bit parameters are limited to 65535 (0FFFFH), and eight bit parameters are limited to 255 (0FFH), while the 24 bit parameters, used for position and step counts, are limited to 16777215 (0FFFFFFH).

Arguments may be signed or unsigned integers, which the CY550 internally converts to 2's complement binary values. While signed arguments may be used for any commands, they are only appropriate for the commands that specify an absolute position, such as **A**, **P**, and **]**. If negative arguments are used for other commands, the CY550 will treat them as large unsigned positive values, and generally this will not result in the desired behavior. Signed arguments should be avoided for commands other than those using absolute positions.

For absolute positions, positive signed arguments are entered without any sign, as shown in all examples above. For example:

P 45678<cr>	Positive absolute position
] 3A5C2H<cr>	Positive hex format

Negative value arguments are specified with a leading minus sign, as shown below:

P -75231<cr>	Negative absolute position
A -0BA9CH<cr>	Negative hex format

The range of signed absolute positions is +8 million to -8 million, so they may be up to seven decimal digits. When the CY550 displays the current position, the sign is always included. However, you should not enter positive arguments with a plus sign. Only the minus sign is accepted as valid.

Also, signed arguments are only used in the ASCII command format. In the Binary format, you must use the 2's complement equivalent binary value of the desired argument.

Binary Command Format

The Binary format still starts with the ASCII command letter for each command, which you may think of simply as an 8-bit command code. However, the format for parameters is quite different in Binary mode. The command letter is always followed immediately by a binary data count, representing the number of data bytes needed by the command. For commands without parameters, such as as the Go command, the data count is a binary value zero (not the ASCII character "0"). For commands with parameters, the data count represents the number of bytes needed to specify the parameter(s). This byte count must match the expected size of the parameters. For example, the Number command must always have a data count of three, for its 24 bit parameter, even if the most significant byte values are zero.

The Quit command, which ends the entry of commands into external memory, should not be followed by any data count. This command is immediately followed by the ASCII letter of the next desired command. Also, the Stop command must use the binary value zero, not the ASCII character "0", followed by a data count which is also zero.

All multi-byte parameters must be entered least significant byte first. For example, the Zloop command has a 16 bit count, followed by an 8 bit address, so it has a data count of three, with the data byte sequence being the LSByte of the count, the MSByte of the count, then the branch address. More examples are shown below.

The data bytes are shown as hex values, with a space separating each one for clarity, but they are issued to the CY550 as single 8 bit values, without the spaces. For example, the "D" command shown below as "D 02 3B 1A", would be sent to the CY550 as four bytes, the command letter "D" (044h), the data count 02, and the two bytes of the 16 bit argument, sent LSByte first, 3B, then 1A.

G	00	No argument, byte count zero
S	01 87	Single byte argument
D	02 3B 1A	Two byte argument, value 1A3BH
N	03 11 22 33	Three byte argument, value 332211H
Z	03 21 04 92	Multiple arguments, Z 0421H,92H

The following sections describe each of the CY550 commands. The commands have been organized into groups of related functions.

Motor Control Commands

The motor control commands are used to set the operating conditions for running the stepper motor, and to actually move the motor. Various commands set relevant parameter values, and two commands cause actual motor motion. In addition, the JOG signal may be used to move the motor manually at any time.

Set Stepping Rate

Three commands set up the basic step rate functions for the CY550. They are:

F Rate	Set initial step rate	[default = 3]
S Slope	Set acceleration slope	[default = 0 (user **must** set >0)]
R Rate16	Set maximum step rate	[default = 100h (256)]

The rate parameter for **FirstRate** is a single byte value that selects the initial step rate for the motor. This parameter is a pointer into the CY550 Rate Table, shown in a later section of this manual. The actual step rate in steps per second comes from a combination of the Rate Table entry and the operating frequency of the CY550. The step per second values of the Rate Table have been computed for a crystal frequency of 12 MHz, with other popular frequencies also shown. The use of other frequencies causes a linear scaling of the rates, and may be derived by multiplying the given rates at 12 MHz by $F_{cy}/12$, where F_{cy} is the actual crystal frequency (in MHz) used for the CY550.

The slope parameter for the **Slope** command determines how quickly the CY550 accelerates and decelerates while stepping. It is also a single byte value, and should range from 1 to 255. Slowest accelerations are selected by small values of slope, while fast accelerations use larger values for slope. In practice, you must experiment with the slope parameter value to determine the optimal acceleration for your specific application. Initial values in the 175 to 225 range might be tried, with adjustments made as required.

Acceleration and deceleration are symmetrical, and are controlled by the single slope parameter. For small values of slope, the CY550 can take a very large number of steps to go from the initial rate to the maximum rate. If the distance to travel is too short to allow the CY550 to reach the maximum rate, it will perform a partial acceleration, reaching the highest possible rate under the current slope parameter, before having to decelerate to stop at the target position.

Also, if the acceleration time is long enough, the CY550 performs a special smoothing function during acceleration, providing the smoothest and finest possible acceleration increments for the given parameters. This function, combined with an improved acceleration curve, significantly reduces the mid-frequency resonance and stalling effects of a motor, and extends the values of slope over which a system can be operated.

The step rate parameter for the **Rate** command specifies the maximum step rate the CY550 will use for the current motion. **Unlike the FirstRate and Slope parameters, the Rate parameter is a 16 bit value.** Also, this value directly defines the maximum step rate in CY550 cycle counts. Since CY550 cycles represent an elapsed time value, the Rate parameter value is inverted in behavior, that is, larger values of the Rate parameter represent slower step rates.

One CY550 cycle time is defined as twelve divided by the oscillator frequency, $12/F_{cy}$. For example, at 12 MHz, one CY550 cycle time is 12/12 MHz, or 1 usec. At 11 MHz, one cycle is 12/11000000, or about 1.091 usec.

A Rate parameter of 100 would specify a slew rate of 100 cycles per step. At 12 MHz, this would be 100 usec per step, for a step rate of 10,000 steps per second. Similarly, a Rate parameter value of 5000 would specify 5000 usec per step, or 200 steps per second.

A general formula for deriving the slewing step rate in steps/second, from the Rate parameter and oscillator frequency is:

F_{cy} (Hz) / (12 * R) = steps/second

For example, with a Rate parameter of 100 and an oscillator frequency of 11 MHz, the step rate value would be:

11000000 / (12 * 100) = 9167 steps/second

Similarly, if you know the desired step rate and oscillator frequency, you can find the required Rate parameter value as follows:

R = F_{cy} (Hz) / (12 * steps/sec)

Note that the actual value of R to use would be the closest integer to the above calculation, since the Rate parameter must be an integer value.

The allowed range of values for the rate parameter is 65535 to 67, which gives a dynamic range of about 15 steps/sec to about 19,900 steps/sec at 16 MHz. The CY550 cannot step at rates faster than the 67 cycle value. If a Rate command has a parameter value smaller than 67, the CY550 will not use that value, and will set the Rate Value Out of Range error bit in the error status register. The previous value of Rate will be retained in this case.

If a maximum stepping rate defined by the Rate parameter is less than the stepping rate defined by the FirstRate parameter, the CY550 will not accelerate. It will run at the FirstRate.

Set Count and Direction

Three commands set relative mode values for the CY550. They are:

N Num24	Set number of steps
+	Select CW direction
-	Select CCW direction

The **Number** command specifies the number of steps the CY550 is to take in the relative step mode. It uses a 24 bit parameter, so the CY550 can travel + or - 16 million steps from its current position in the relative mode. No stepping occurs when the Number command is issued, the command simply defines the number of steps to take, and this value is stored in an internal N register of the CY550.

The + command selects the clockwise direction for stepping. This command causes the CCW signal to go low.

The – command selects the counter-clockwise direction for stepping. The CCW signal will go high when this command is used.

Externally Forced Direction

Stepping direction may be controlled externally, by driving the CCW pin with the desired level. If this function is used, the – command should be issued, so the CCW signal is high. It can then be pulled low externally to force clockwise stepping.

If the CCW pin is being driven low by the CY550, it is a stronger sink device, and will require a very strong external signal to force the pin into the high state for CCW stepping. However, if the CCW pin is held high by the CY550, from a – command, it can easily be driven low for CW stepping.

Set Step Mode

Two CY550 commands control step operation modes for the CY550. They are:

A Pos24	Set current step position
C	Select continuous step mode

The **At** command sets the current position kept by the CY550 to the value of its parameter. This number is a 24 bit signed value, so any position from -8388608 (-800000H) to 8388607 (7FFFFFH) may be specified. Normally, this command is used after stepping to a known mechanical position, such as a "home" position sensor, and can calibrate the CY550 to make this position always be the same number.

Note that with a signed absolute position, the CY550 will consider position zero at the "middle" of its travel range, and can travel in either direction around zero.

While the CY550 is stepping, the internal current position is always updated. The position is incremented during CW stepping, and decremented during CCW stepping. You may query the CY550 for its current position at any time. Also, if MBit 4 of the mode command is set, the CY550 will automatically display its position while stepping.

The position value is not too important for relative mode stepping, in which the direction and number of steps are specified. However, it is very important for absolute mode stepping, in which a target position is specified, and the CY550 computes the direction and number of steps required to reach the target.

The **Continuous** command sets a special step mode in the CY550. In this mode, the CY550 will accelerate from the initial rate to the final rate, and then run continuously. It will not ramp down or stop at any particular position or number of steps.

In order to stop a continuous motion, the INHIBIT_ABORT signal must be pulled low, forcing the CY550 into a deceleration mode, and then be kept low to stop the CY550. The CY550 will stop, after it has decelerated to the starting rate, if the INHIBIT_ABORT signal is still low.

This may be done by driving the INHIBIT_ABORT signal externally, or by sending the Stop Motion command, ^ <cr>, to the CY550 while it is stepping.

The continuous mode is set by the C command, but stepping does not begin with this command. The continuous mode may also be selected by external signal. This is done by forcing the SLEW signal low as the CY550 starts stepping.

A continuous mode motion must be specified before each continuous motion the CY550 takes. When the move is over, the CY550 reverts to the incremental mode until another C command is issued, or until the SLEW line is driven low at the beginning of a motion.

Moving the Motor

Finally, two commands are used to actually make the motor move, based on the currently defined parameters. They are:

G	Step in a relative mode
P Pos24	Step to the specified target position

The **Go** command selects relative mode stepping. The CY550 will step based on the last Number of steps specified and the selected step direction. A continuous mode step motion may also be started by this command, if a previous C command has been processed.

Absolute mode stepping is selected by the **Position** command. This command both specifies the desired target position and starts the motion. When the target is specified, the CY550 will compute the number of steps to take and the direction of motion, based on the current position register.

The argument to the Position command is the signed absolute target position. If the signed target position is greater than the current position, the CY550 steps in the CW direction. If the signed target position is less than the current position, the CY550 steps in the CCW direction.

Recall that the current position is defined by the At command, and is updated with every step. It is a 24 bit value, ranging from –8 million to +8 million. The Position command limits stepping to a number of steps within this range.

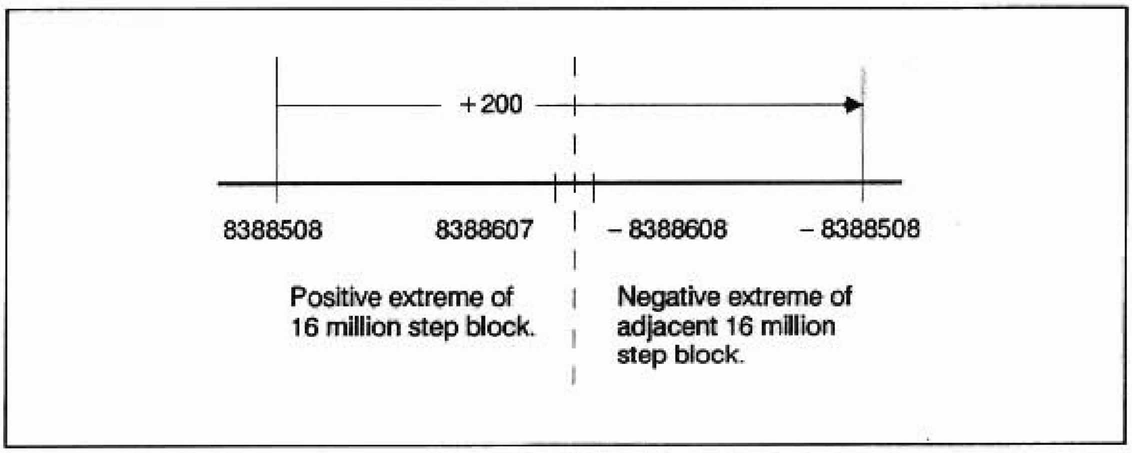
Position Counter "Wrap-around"

If a relative motion causes overflow or underflow of the current position register, an absolute move would now work within the new block of 16 million steps. For example, if the current position is 8388508, and the following commands are executed:

```
N 200<cr>

+<cr>

G<cr>
```

The motor will step 200 steps in the CW direction, and the current position register will overflow to a value of –8388508, which is at the negative extreme of a new block of 16 million steps!

This behavior allows stepping to regions beyond the 16 million step block supported by the CY550. As long as you can work within a block of 16 million steps, relative mode stepping can be used to move from one block to the next, with the host computer keeping track of which block the CY550 is currently located in.


+200
8388508
8388607
– 8388608
– 8388508
Positive extreme of 16 million step block.
Negative extreme of adjacent 16 million step block.

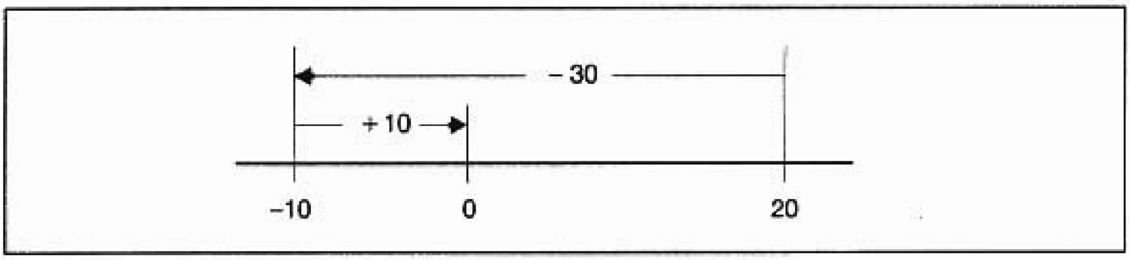
Also, the signed position behavior of the CY550 is beneficial in most cases. If the current position is 00000020, and the following commands are executed:

```
N 30<cr>

-<cr>

G<cr>
```

The motor will step 30 steps CCW, and the current position will be a value of −10. If you now command:

```
P 0<cr>
```

The motor will step the desired 10 steps in the CW direction, stopping at position zero. The combination of large relative moves, which allow position overflow into adjacent blocks of 16 million steps, and signed position functions within a single block, makes for a powerful combination of stepping performance.



On-the-fly Motion Commands

One of the major features of the CY550 is the ability to process commands while a motion is in progress. Most of the CY550 commands can be issued during stepping, and they will perform correctly. However, a few commands cannot be processed on-the-fly. This section will discuss some commands specifically designed to aid on-the-fly operations, followed by a more general discussion of on-the-fly functions.

Several motion related commands are useful in on-the-fly situations:

V	Wait for the end of the current motion
] **Pos24**	Wait for the specified absolute position
\ **Num24**	Wait for the specified number of steps
^	Stop the current motion

The **V** command will force the CY550 to wait for the current motion to finish before executing any other commands. This effectively turns off the on-the-fly functions. The CY550 will not process any additional commands until the motion is over.

This command is useful in situations where you must wait for the motor to stop before performing some other operation, such as manipulating some User I/O bits.

After the V command has been processed, the CY550 will not accept any more commands until stepping has finished. This is true of commands from the host as well as commands from the local memory. The only function still performed during the V command is the live display of the current position, if enabled by MBit 4 being set by a previous mode command.

If the CY550 is not stepping when this command is processed, the command is ignored. Also, note that ALL command processing halts during stepping, including processing of the "^" stop motion command and "?" query commands!

Event/Position Coordination

The] command, which causes the CY550 to wait for the specified absolute position to occur, also halts all other command processing until that position is achieved.

In processing the Wait-for-Absolute-Position command, the CY550 will sample the current position, but not perform any other functions during the stepping, including the display of position on-the-fly, enabled by MBit 4. The CY550 will continuously compare the current position to the desired position, and once that position has been reached, normal command execution will continue.

This command is useful for **coordinating some other activity with the motion**, such as activating an I/O signal with the Bitset command, at a specified absolute position.

Since the CY550 stops all other command processing, then continues with commands after the position has been reached, you must account for the processing time of the following command when the] command is used. That is, the CY550 will wait for the specified position to be achieved. It will then execute the next command, where the execution includes parameter processing and command setup. At high step rates, it may be necessary to compensate for the command execution time by specifying a position a few steps earlier than that desired for the following action. This timing is best determined empirically, as it will vary with the exact step rate and mechanism for getting the next command, either through the serial or parallel command interfaces, or from the local external memory.

The Wait-for-Absolute-Position command has no effect if the CY550 is not stepping when it is issued.

A command related to the absolute wait is the Wait-for-Relative-Number-of-Steps-to-Occur, \. Note that this command uses the ASCII backslash character "\" (5Ch), which is different from the complement prefix used by commands with I/O bit number arguments, "/" (2Fh).

Internally, the CY550 executes the \ command by combining the argument value with the current position when this command is executed. The result is an absolute target position to wait for. Once this target position is computed, the \ command behaves the same as the] command, waiting for the computed position to be reached.

Similar timing considerations apply to the \ command as to the] command, and all other command processing stops until the computed target position is reached.

The \ command is very useful for detecting an event, such as a User Bit I/O signal, then delaying for a fixed number of steps before taking any other action. If the signal may occur at a variable position, but the interval between the signal and the next event must be fixed, this command may be used to achieve that function.

Another useful command in on-the-fly operations is the stop motion command, ^ < cr >. This command may be issued to stop any type of motion, either incremental or continuous, before the CY550 would normally stop.

When the ^ command is issued through the command interfaces, or executed from the local external memory, the CY550 will force itself into a down ramp mode, and stop stepping when the first rate is achieved. This is done by internally pulling the INHIBIT_ABORT signal, pin 8, low, and has the same effect as if that signal had been pulled low and held low externally.

Note that the ^ command is a normal CY550 command, and has the complete command structure, including the carriage return in the ASCII command mode, or the zero binary count in Binary command mode. The CY550 will not act until both bytes of the command have been received.

This command has no effect if the CY550 is not stepping.

Also note, if on-the-fly functions have been suspended for any reason, for example, processing the \ or] commands, the CY550 will not accept the ^ command! This command is not given any different priority than other on-the-fly commands.

General On-the-fly Information

The CY550 on-the-fly operations work best either with a highly interactive host system, or with a program of CY550 commands, loaded and executed from the local external memory. Both modes of operation allow commands to be processed during motions. In fact, any external memory based programs must be written with this in mind, since the CY550 will continue to read and execute commands after a motion has started. Be sure to use the wait for end of motion command, V, if you do not want the following commands to occur until the current motion is over!

The CY550 is capable of processing most commands on-the-fly. However, there are a few restrictions, generally due to the effect such commands could have on the current motion.

Some commands will not be processed on-the-fly at all, since they will interfere with the stepping function. These commands are the direction commands (+ and -), the step commands (G and P), the A command, the H command, and the Jog signal functions. If any of the above commands is issued on-the-fly, the CY550 will wait for the current motion to finish before executing the new command. This is equivalent to issuing the V command before any of the above commands.

Note that this will halt all other on-the-fly commands until the motion is over. For example, it is not possible to issue a change in direction on-the-fly, followed by a stop stepping command, ^. The direction command will halt on-the-fly command processing, so the stop stepping command will not be processed by the CY550.

This behavior is generally not a problem for programs written to execute from the local external memory. However, you must be careful to not issue these commands from a "live" interactive host, since they may "lock up" some desired on-the-fly operations.

Jogging is simply disabled during stepping, but does not suspend the execution of any on-the-fly commands.

For incremental motions, specified with a number of steps or target position, you may not issue the Slope or FirstRate commands during stepping. These commands will have the same behavior as described above, halting on-the-fly command processing until the end of the current motion. The FirstRate and Slope values must remain constant during incremental motions, resulting in symmetrical acceleration and decelaration for such motions.

While it is possible to issue the Rate command on-the-fly for incremental motions, resulting in a change in the slewing rate of the CY550, we do NOT recommend the use of this command during incremental motions. Under certain combinations of values and timing, it is possible for the CY550 to miss the desired ending position, and continue stepping beyond the intended stopping point. This is especially true for slewing Rates that are close to the FirstRate in value.

If you must use the Rate command on-the-fly for incremental motions, you must also verify that the CY550 stepping behavior will be acceptable under all circumstances in which the on-the-fly rate change will be used. This will require extensive testing to confirm proper behavior in your application.

We believe the majority of applications that require on-the-fly Rate changes will also use continuous motions instead of incremental motions. For continuous motions, Rate, Slope, and FirstRate commands may all be issued on-the-fly, changing any of the stepping parameters as desired.

Other CY550 commands do not have on-the-fly restrictions, and may be used freely during motions. This makes it possible to design a highly interactive system, with on-the-fly queries and I/O bit manipulation, as well as conditional testing and decision making!

The **Til** command is only used in an external memory program, since it will jump or fall through, depending on the bit value. The address of the jump is a single byte, limiting the jump to an address on the same 256 byte page as the command after the Til command. The TIL command may jump to itself.

The **Wait** command simply waits for a match of the bit(s) against the parameter value. When there is no match, the CY550 stays in the Wait command, and continually tests the signals until a match occurs. Once a match is detected, the CY550 will continue with the next command. No other commands are processed while the CY550 is waiting for the bit match to occur.

The **Wait** command is normally used in an external memory program, to synchronize the CY550 with some external control or status signal. For example, move a part into place, activate some operation by changing a user bit, wait for a signal back from the operation indicating operation complete, then remove the part.

The following example illustrates the use of **Wait** and **Til** commands for the operation described above:

```
Y 100<cr>             Program start address
E<cr>                 Following instructions go into memory

W 1<cr>               Wait til USRB1 = 1

+<cr>
G<cr>                 relative motion in CW direction
V<cr>                 wait for motion to finish
/B 2<cr>              drive USRB2 = 0 as signal
W 11h<cr>             Wait til USRB1 = 0

-<cr>
G<cr>                 return same number of steps
V<cr>                 wait for motion to finish
B 2<cr>               drive USRB2 = 1 again
T 10h,100<cr>         test USRB0 and jump unless it is 0
0<cr>                 then fall thru and stop the program
                      by returning to command mode

Q<cr>                 End of program definition
Y 100<cr>             Reset address
X<cr>                 eXecute the above program
```

Step/Test/Seek Home Command

Finally, one command uses the bits to control special motions, and cause stepping during the bit testing. It is:

H Bit	Step to Home, using specified signal

The values of the parameters for this command are treated the same as the **Til** and **Wait** commands. That is, for the **Home** command, the bits are tested for a match to the parameter value.

As the CY550 steps the motor for this command, it will run at a constant step rate, with no acceleration or deceleration. The desired step rate is chosen by the **First Rate** parameter, set by the **F** command. The actual step rate is a derivative of the rates in the **CY550 Step Rate Table.** For the **Home** command (and the Jog pin function), the CY550 will divide the selected rate by 20. When operating at 12 MHz, this gives a range of about 1 step per second to about 250 steps per second for this command.

No on-the-fly functions are available during the seek to home.

The Home command (and the Jog pin function) will cause the Stopped signal to pulse with each step.

The **Home** command starts by testing the specified bit pattern for a match with the external signals. If there is a match, the CY550 will begin by stepping in the CCW direction, and will continue in this direction until there is no longer a match.

When there is no match between the bit parameter and the external signals, the CY550 steps in the CW direction. It will continue to step in this direction until the signals match the parameter value. At this point the CY550 stops, and sets the current position to zero.

Thus, the **Home** command steps the motor to seek the edge between a match and no match condition on the home signal. The CY550 then stops at the first step where a match occurs, and is always stepping in the CW direction looking for this match. This mechanism should always seek the same mechanical position, since any directional backlash is compensated by using the same CW step direction in seeking the final home position.

If the CY550 runs into one of the limits while looking for the home signal, it will abort the home command and stop stepping. The current position is not changed to zero in this case.

7 External Memory Support Commands 7

External Memory Support Commands

Several CY550 commands provide support for the optional external memory. They control the addressing, data entry, and execution of commands from the memory. In addition, the external memory address space can be used for extended I/O functions, discussed in the next chapter.

External Memory Address Pointer

The first command is used to set the address pointer. It is:

Y Addr16	Set external memory address pointer

The 16 bit argument to the Y command is placed into the memory address register of the CY550. Other commands may use this value for their functions.

The Y command does not change modes or other features of the CY550, so when issued as a direct command, it simply sets up the address pointer. The CY550 will then wait for the next command. However, **when the Y command is executed from external memory**, redefining the address pointer causes the CY550 to execute commands from the new address, so **the Y command becomes a 16 bit jump command.** This dual function allows you to specify the working address when defining the memory contents, and jump to any location of the memory when running from the memory contents.

Writing Commands into Memory

The two commands that control the entry of data into memory are:

E	Enter following commands to memory
Q	Quit entering commands to memory

The **Enter** command is issued to the CY550 in the direct command mode. Once this command is executed, the CY550 changes to the program entry mode. All following commands will be written to the external memory. The CY550 will not execute any of the commands, it simply writes them into the memory.

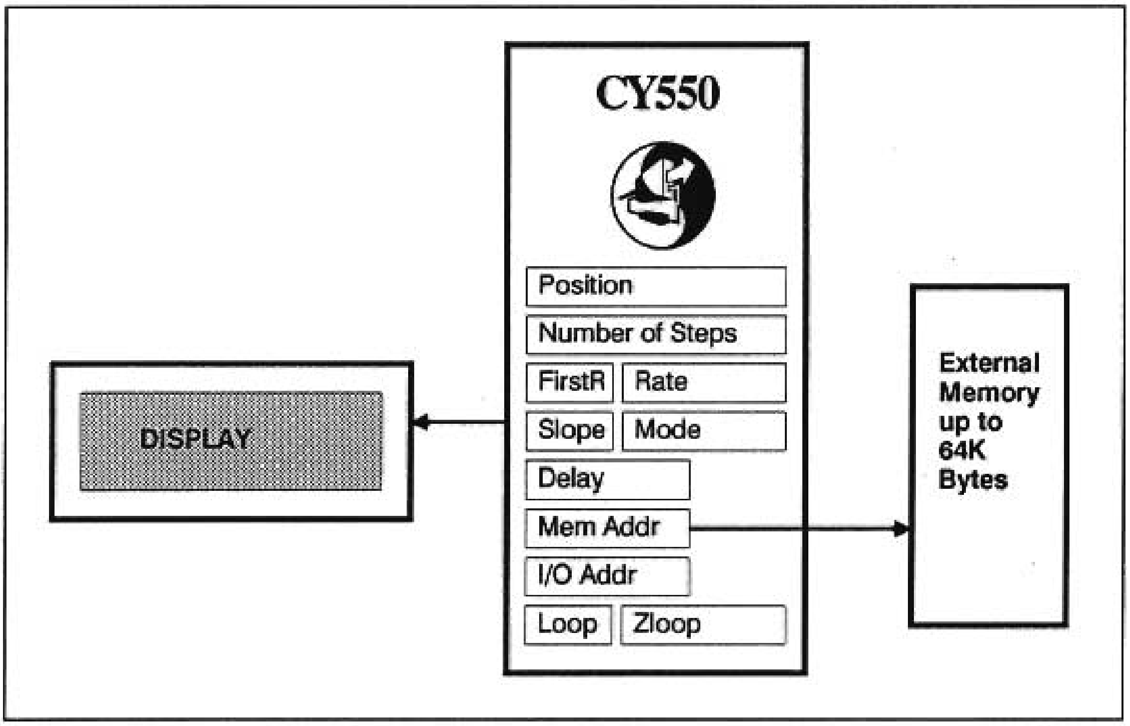
> **Commands will be written to memory starting at the current address set by the Y command, and the address pointer will increment once after writing each command character.**

Note that some communications devices/software will add a linefeed to a carriage return, which will not affect the operation of the CY550. However, this additional character will increment the memory address pointer, which will affect Jump and Loop loactiona if not anticipated.

The memory contents consist of a string of characters, and are identical to the characters issued to the CY550 in direct command mode. Every command character sent to the CY550 will be recorded in memory exactly as it is sent to the part.

The memory writing function continues until the CY550 receives a **Quit** command. It then returns to the direct command mode, in which every command is executed immediately as it is received. The address pointer value now contains the address where the next character would have been written, and the difference between the starting value and ending value will indicate the amount of memory required for your program. Before running the program, the Y command should be used to redefine the pointer value for the start of the program.

Note that the carriage return character is optional after the Quit command in the ASCII command mode, and a data count is not used after the Quit command in the Binary command mode.



Running From External Memory

Two commands are involved in running programs from the external memory. They are:

X	Execute external memory commands
0	Stop external memory command execution

The **Execute** command changes the CY550 from the direct command mode to the program execution mode. Your program will begin running from external memory, at the current address in the address pointer. The **X** command would normally be preceded by a **Y** command to set the program starting address.

In the program execution mode, the CY550 will begin reading command characters from the memory, incrementing the address pointer after every read operation. When an entire command has been read into the part, that command is executed similarly to directly issued commands.

When the current command is finished, the CY550 will read the next command from memory and execute it. This process continues until the CY550 finds a stop command.

The **Zero** command (ASCII character 0, not binary value zero) is the **stop execution** command. When this command is executed, the CY550 changes from program execution mode back to direct command mode. It then waits for the next command to be issued by the host system. Note that in the Binary command mode, the binary value zero should be used for the stop execution command, followed by a zero value data count.

You may also issue commands to the CY550 while it is running a program from external memory. The CY550 will process these commands between the commands read from the memory. However, no commands will be processed, either from the memory or from the command interface, if the CY550 is stepping and the on-the-fly operations have been suspended by a previous command, such as the **V** wait for end of motion command. In these cases, new commands will be processed when the motion finishes.

The ability to issue direct commands while the CY550 is running an external memory program provides another degree of "live" interaction, allowing you to change parameters or modify the execution flow by the direct commands. It is even possible to halt a program, by issuing the **Zero** command!

> **Note:** A special feature of the CY550 is the ability to automatically run an external program at power-up, or whenever the CY550 is reset. See the "Auto-Start" feature in Section 13.

External Memory Programming Example

From this discussion, you can see that one command controls the basic memory addressing function, while two pairs of commands control the definition of memory contents and execution of memory contents. A simple example is shown below:

```
F 15<cr>              Define some parameters in direct mode
S 200<cr>
R 300<cr>

    . . .             Additional commands are possible here

Y 50<cr>              Set memory pointer to address 50

E<cr>                 Start program entry at location 50
```

The following is written to memory, but not executed

```
N 400<cr>             First command in external memory
+ <cr>
G<cr>                 Take 400 CW steps
V<cr>                 Wait for end of motion
D 1000<cr>            Delay for 1 second (1000 msec)
N 350<cr>             Reset number of steps
-<cr>
G<cr>                 Take 350 CCW steps
V<cr>                 Wait for end of motion
D 750<cr>             Delay for 0.75 seconds
0<cr>                 Stop, last command in external memory

Q<cr>                 End of program definition

    . . .             Other direct mode commands may be here

Y 50<cr>              Reset address pointer to prog start
X<cr>                 Execute. Begin running the program
```

The program in the external memory will run after the X command, and will continue until the Stop command is found. The program takes two relative motions, with different numbers of steps, and in different directions. When the Stop command is seen, the CY550 goes back to direct command mode and waits for the next command. The program could be repeated by sending the Y and X commands again, since the program continues to reside in the external memory.

If non-volatile external memory is used, such as an EEPROM, the program will be stored in the memory even when power is removed from the CY550 system. So, once a program is defined, it may be used over and over, by simply pointing to the program address and sending the Execute command.

More hardware details about external memory support are provided in a later section.

Extended I/O Support

In addition to using the CY550 external memory region for program storage and execution, as described in the previous chapter, it can also be used for extended I/O functions.

Direct I/O functions are performed through the User Bit signals, as described in the chapter on Bit Function Commands. These commands provide direct access to the User Bit I/O signals for setting, clearing, and testing functions.

However, in many applications, there is a need for additional I/O signals, beyond the six or eight lines provided by the CY550 directly. Through the extended I/O commands and the address mapping of the external memory space, the CY550 provides you with a potentially very large number of additional I/O signals, up to 1/2 million (64K x 8)!

These additional I/O signals are mapped into the external 64K memory space supported by the CY550, and in most applications, will share that space with the local external memory. Some of the memory space will be used for program storage and execution, while some is used for extended I/O functions.

The CY550 also implements three internal resources and a number of commands in support of the extended I/O. There is an input or read address pointer, an output or write address pointer, and an internal I/O byte register, used for the input and output data transfers. These are in addition to the memory address pointer manipulated by the Y command.

The separate input and output address pointers allow separate addresses for input reading and testing or output setting, without the need to reload a single address pointer between operations. They may also be used in block copy operations, moving the contents of non-volatile memory, such as EEPROM, to output registers or even to volatile data RAM during some initialization sequences.

The input and output address pointers and the I/O byte register are only used by the extended I/O commands. Other CY550 commands will not change the contents of these registers.

The extended I/O functions enable the 64 Kbyte external memory space of the CY550 to be split into four functional units, all of which are optional. Any particular application may use any combination of functions, or none at all, if the CY550 is directly commanded from the host system, and only uses the direct I/O provided by the User Bits!

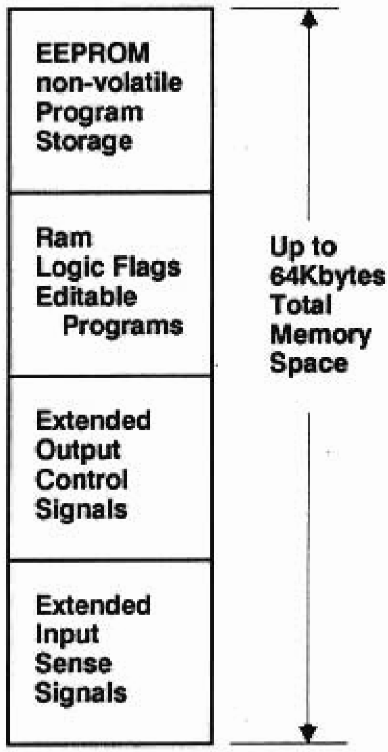
CY550 External Memory Space Architecture

The CY550 can address one external memory space of 64 Kbytes in size. This space can be split into four functional units.

EEPROM. The first use of the CY550 external memory space is for storage of command sequences and programs, as described in the previous chapter. The memory for this function can be RAM, (EP)ROM, or EEPROM, depending on the requirements for non-volatility and rewriting during operation. In most cases, it will be a non-volatile memory, with EEPROM being the most convenient, since it allows the user to define and save new programs, rewrite them on occasion, and retain them during power cycling.

This memory space uses the Y, E, Q, and X commands to set up the memory addresses, define the command sequences, and provide access to the programs.

CY550 Extended Memory Space



The CY550 External Memory Space may be split into four logical sub spaces.

RAM. With the input and output pointers, and the I/O byte register, it is possible to add a second, volatile working RAM to the address space! This memory can be used for temporary storage of values or program sequences that must be changed frequently. It may also be used for program logic flags, which can influence the execution sequence of CY550 programs, based on the values of the flags.

The CY550 can now make some of the changes without the host system rewriting the entire command sequence. It is possible to easily edit parameter values, for example, without rewriting the entire program.

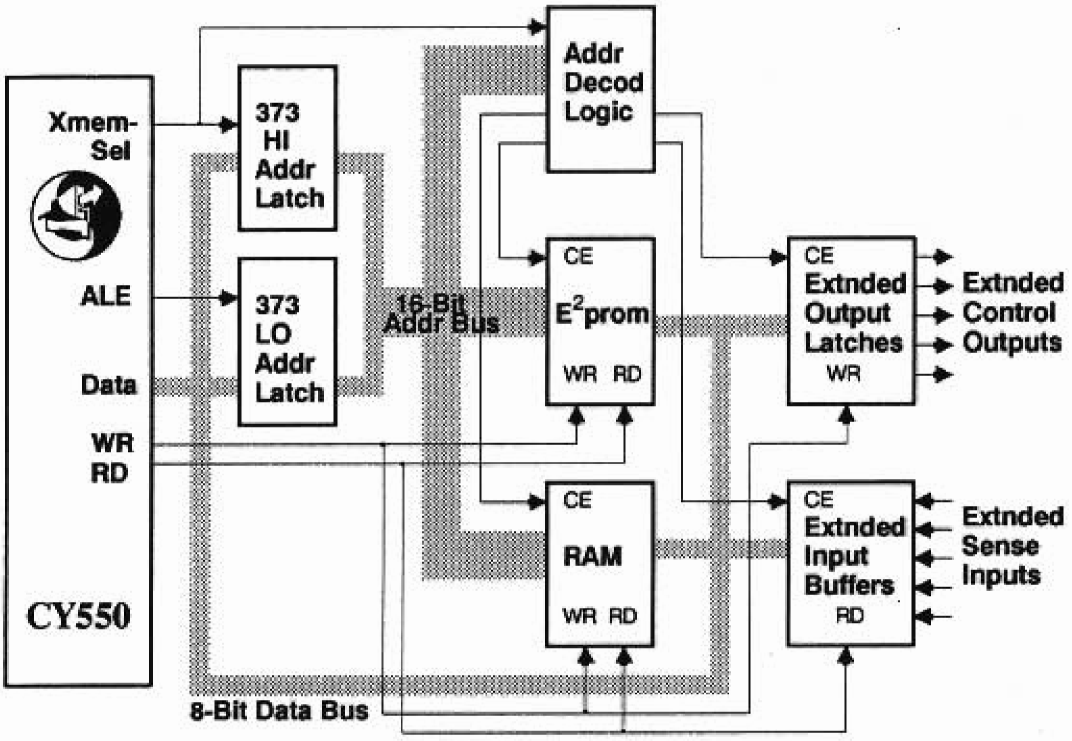
Also, some of the RAM can be used for logical flags, that can be read, written, and tested as part of the normal instruction execution flow of the CY550. This provides a significant extension to the User Bits available for testing and program execution flow modification.

Extended Output. A third use for the 64 Kbyte memory space is for memory mapped output functions. If some of the available address space is allocated to registers or latches instead of memory, any output functions can manipulate real output signals, providing a significant extension to the User Bits available directly on the CY550. Use of this function only requires some address decoding logic plus the latches to hold the output values. Up to 8 bits can be manipulated with a single output operation.

Extended Input. Similarly, the fourth use for the external memory space is for memory mapped input functions. By substituting input buffers for some of the memory, input functions can access real external signals, and through the I/O byte register, CY550 programs can make conditional jumps based on the values read back. Up to 8 signals can be read from one input location, and each bit can be tested independently.

The only penalties involved in the use of the extended I/O functions are the two step process for manipulating the I/O, and the address decoding required to use the I/O function within the external memory address space of the CY550. Where the User Bit signals are accessed directly, the extended I/O signals must be accessed indirectly, through the input or output pointers, and the I/O byte register. However, this process extends the number of I/O signals the CY550 can support enough to satisfy the needs of even the most demanding I/O applications.

CY550 Extended Memory Block Diagram



Extended I/O Commands

The CY550 implements several commands specifically to support the extended I/O functions. The first commands set the address values of the input and output address pointers. They are:

K Addr16 Set external input address pointer
M Addr16 Set external output address pointer

Both commands use 16 bit address values, so the entire 64 Kbyte external memory address space can be accessed. Each command loads the argument value into the relevant pointer register.

I/O Byte Register

The internal I/O byte register is used to transfer data between the extended I/O signals of the external memory space and the internal functions of the CY550. It is the source of output write functions and the destination of input read functions. The register is not accessible on any CY550 pin signals, it is simply an internal working register, similar to those that hold parameter values, such as Rate or Slope.

This register can be loaded by two functions, inputting a value from the extend I/O signals, or using the value load command. The value load command is:

# Value Load value into I/O byte register

The argument to the # command is an 8 bit value, and is loaded directly into the I/O byte register. The I/O byte register contents may be used in many ways.

In addition to the input and output functions, the I/O byte register is accessible to two of the CY550 Bit Function commands, the Bit set command, and the Test command.

The B command can be used to set or clear individual bits of the I/O byte register, as well as the User Bit signals. When the bit number argument has a value from 20h to 27h, one of bits 0 to 7 of the I/O byte register is set, and when it has a value from 30h to 37h, one of bits 0 to 7 of the I/O byte register is cleared. For example:

B	**23h<cr>**	**Set bit 3 of I/O byte register**
/B	**23h<cr>**	**Clear bit 3 of I/O byte register**
B	**36h<cr>**	**Clear bit 6 of I/O byte register**
/B	**36h<cr>**	**Set bit 6 of I/O byte register**

Similarly, the T command uses argument values from 20h to 27h to test I/O byte register bits 0 to 7 for one, or argument values from 30h to 37h to test I/O byte register bits 0 to 7 for zero. This allows the extended I/O signals, whether real external signals from the extended I/O space, or logic flags from an external RAM space, to be used to control program execution flow, similarly to the direct User Bits! For example:

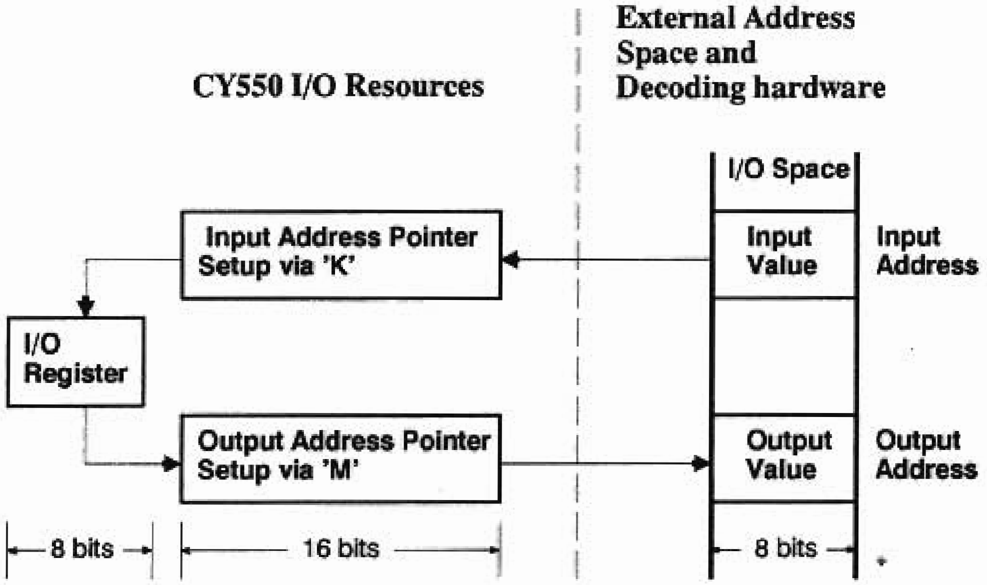
```
T    25h,123<cr>        Test bit 5 of I/O byte register for 1
```

If bit 5 of the I/O byte register is clear (0), the above T command will jump to location 123 of the current memory page, since the bit does not match the specified value. If the bit is set (1), the T command does not jump, but continues with the command following the T command.

This action is identical to that when testing the CY550 User Bits directly, except that it uses the value of the I/O byte register! Before performing such a test, you would normally load the I/O byte register with the appropriate value, by performing an input operation.

Note that bit numbers 20h to 27h and 30h to 37h also refer to the internal I/O byte register if used with the Wait or Home commands. However, the I/O byte register should **never** be used with these commands, since the bits of the I/O byte register cannot change during the execution of the W or H commands. Only use the W and H commands with arguments that specify the CY550 User Bit signals, which can change during the wait or home functions. Otherwise, the CY550 might never finish the execution of the wait or home commands, and will not execute any other commands.

CY550 External Memory Space Architecture and I/O



CY550 I/O Resources
External Address Space and Decoding hardware
I/O Space
Input Address Pointer Setup via 'K'
Input Value
Input Address
I/O Register
Output Address Pointer Setup via 'M'
Output Value
Output Address
8 bits
16 bits
8 bits

The I/O Register is loaded indirectly through the Input Pointer when the I-command is executed, and the contents of the I/O Register are output indirectly through the Output Pointer when the %-command is executed.

Extended Input and Output

To complete the extended I/O functions, the CY550 implements two commands for actually performing the input (read) and output (write) operations between the external memory space and the internal I/O byte register. The commands are:

!	Input a value to I/O byte using input pointer
%	Output a value from I/O byte using output pointer

Each command uses the value of the appropriate pointer to generate the external memory space address. The ! command transfers a byte from the external memory space to the I/O byte register, while the % command transfers a byte from the I/O byte register to the external memory space. Note that if the address corresponds to a memory mapped I/O location, the ! or % command performs input or output with extended I/O signals. Otherwise, the input or output is done to memory.

The output operation generated by the % command is a non-verified output write. That is, the CY550 simply writes the data out. It does not attempt to read back the value to verify it, as is done with program definitions, using the Y, E, and Q commands. Since output may be done to latches, the CY550 does not assume it can read the value back.

Since the writes are not verified, the % commands should not be used with addresses of EEPROM memory, which may require several milliseconds to write. The % commands should only be used with RAM or extended output latches. If a write is made to EEPROM, any executing external program will probably not function correctly, due to the EEPROM being "busy" writing the output data while the CY550 attempts to read the next program command.

> **Auto-Increment of Pointers:** Also, if MB5 of the mode register (set by the **O** command) has been set, the ! and % commands will increment the value of their pointer. Otherwise, the pointer value is not changed. This is an automatic increment function, and allows a sequence of locations to be read or written without loading the pointer value for each location.
>
> Note that MB5 controls the auto increment of both pointers, but only one pointer will be incremented for each instruction. The input pointer is incremented after the ! command inputs the value at the current input pointer address, while the % command outputs the value to the current output pointer address, then increments the output pointer.

These extended I/O functions provide a significant extension to the I/O capabilities of the CY550, compared to only using the CY550 User Bit signals. However, when using the extended I/O functions, there is generally a two step process involved. First, the address pointer must be properly set, then the I/O operation must be performed, using the I/O byte register.

To write a value to a memory mapped output latch, the output address pointer must be set to the address of the latch. Then the I/O byte register must be loaded with the value to write. Finally, the % command is used to actually write out the value.

To test an extended I/O bit, the input address pointer must be set to address the desired byte. The ! command is then used to read the byte into the I/O byte register. Now, the T command may be used to test the desired bit, and conditionally jump in the CY550 program, based on the bit value.

The CY550 User Bits provide a more direct access, but are limited in number, while the extended I/O bits require more commands, but provide significantly more signals to control.

Program Branch Commands

Jump and Loop

Several CY550 commands control program branching and flow while the CY550 is executing a program from external memory. These commands normally are used only within a program, not as direct CY550 commands. The CY550 will however execute the commands in the direct mode, but the result may be different than if the commands were run from a program. The commands are:

J Addr	Jump to byte location of current page
L Cnt,Addr	Loop to byte address for count times
Z Cnt16,Addr	Zloop to byte address for count times

In addition to the above commands, the Y command, when executed from a program, provides a 16 bit jump that can reach any address of the external memory.

The above listed commands all use an 8 bit address value. This restricts the range of addresses to the 256 bytes of the current memory page. When a jump is taken, the lower byte of the current address pointer is replaced by the 8 bit address parameter value, and program execution continues at this new 16 bit address. When one of the above commands lies on a page boundary, the address pointer is incremented before the command is executed, so the branch would be taken into the new page of the memory, not the previous page that started the command. Care must be used with these page relative jumps, to be sure the target location can be reached by the specified command. In general, the branch command and target must be on the same 256 byte page.

The **Jump** command simply provides a short version of the branch function. It can reach any address on the current memory page.

The **Loop** command provides a repeated branch, with a specified number of iterations. Both parameters are byte values, so a loop may be repeated 255 times, and the branch must be to an address on the current page.

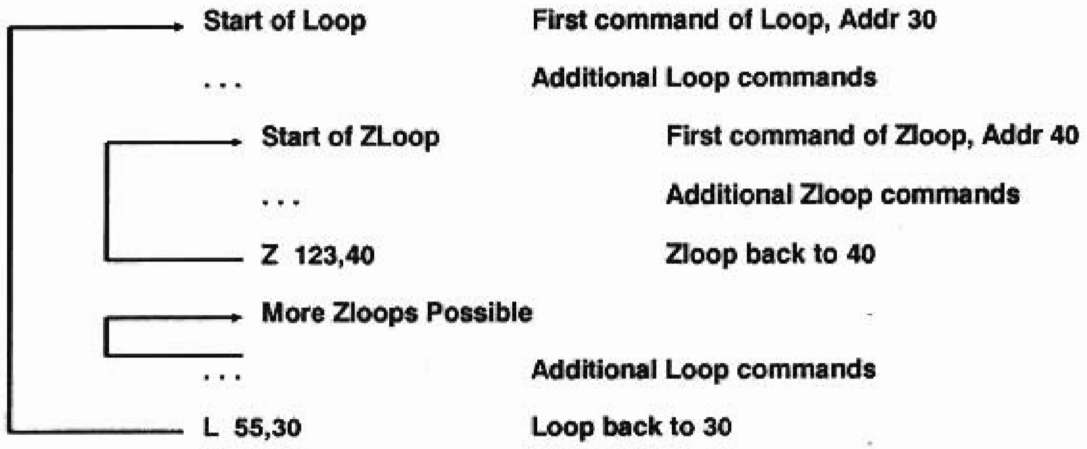
The **Zloop** command also provides a repeated branch. However, the repeat count is a 16 bit value, so the Zloop may be repeated up to 65535 times. The branch is still restricted to an address on the current page.

For both loop commands, the instruction following the loop is executed when the loop count is exhausted. The loop count is decremented before being tested for zero. If the decremented count value is zero, the instruction following the loop is executed. Otherwise, the branch address is used to specify the next instruction to execute.

Nested Loops

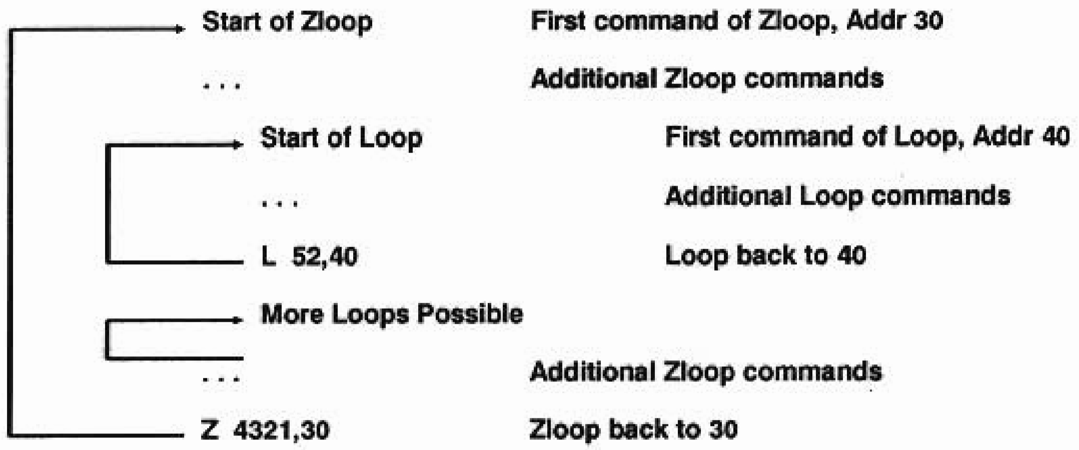
The combination of the Loop and Zloop commands provides two levels of nesting to loops. Only one of each type of loop may be active at a time. That is, a number of loops may reside within one zloop, or a number of zloops may reside within one loop. However, one loop may not reside within another loop, and one zloop may not reside within another zloop. This is shown below:

Zloops within a Loop



```
Start of Loop          First command of Loop, Addr 30
...                    Additional Loop commands
  Start of ZLoop                 First command of Zloop, Addr 40
  ...                            Additional Zloop commands
  Z 123,40                       Zloop back to 40
  More Zloops Possible
...                    Additional Loop commands
L 55,30                Loop back to 30
```

Loops within a Zloop



```
Start of Zloop         First command of Zloop, Addr 30
...                    Additional Zloop commands
  Start of Loop                  First command of Loop, Addr 40
  ...                            Additional Loop commands
  L 52,40                        Loop back to 40
  More Loops Possible
...                    Additional Zloop commands
Z 4321,30              Zloop back to 30
```

Recall that the CY550 also supports a branch command using one of the bit instructions, the Til command. It tests the User Bit I/O signals or I/O byte register against the specified bit parameter value, and jumps if the values do not match. Otherwise execution continues with the instruction following the Til command.

Operating Mode Command

Another command is used to define the **Operating mode** of the CY550. It is:

O Mode16 Set CY550 operating mode

The parameter of the Mode command is 16 bits. The lower 8 bits of the argument value are stored in an 8 bit internal CY550 mode register. These bits are only accessable by the Mode command, not through any external I/O lines, such as the User Bits. Each bit of the Mode command register controls a CY550 feature, as shown below. The upper 8 bits overwrite the Error Status bits, as explained later in this chapter.

MB7 Select ASCII command mode if set, else Binary
MB6 Select serial baud rate if set
MB5 Enable Input and Output pointer auto increment if set
MB4 Display position live during all stepping

MB3 Enable XON/XOFF serial flow control if set
MB2 Select HP Display if set & MB0 = 1
MB1 Enable DTR/ function on USRB6 if set
MB0 Select parallel display if set, else serial

The **Mode** command will define the values for all 8 bits of the register at once, so you must be careful not to affect the value of one feature while changing the value of another. The default value for the mode register is 80H at power up. This selects the ASCII command mode with standard serial display output for the query and string commands.

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	0	0

= 80h, default mode

The most significant bit selects between ASCII command mode and Binary command mode. When the bit is set, ASCII mode is selected, and when the bit is clear, Binary mode is selected. When you use Binary command mode, you must first issue an O command in ASCII mode that turns off the upper bit, to put the CY550 into Binary mode. All commands following the O command are then in the special Binary format:

O 0<cr> ASCII command to select Binary mode

... All following commands in Binary format

When MB4 of the mode register is set, the CY550 will display the current position while stepping from any command, including the Home command, stepping from a JOG signal input, and during normal, commanded motions. When the bit is set, the CY550 will constantly output new position values while any motion is in progress.

The position outputs will go to the currently selected active output, either the serial, parallel, or HP displays, as selected by other bits of the O command, and explained below. It is possible for the CY550 to generate large amounts of output when this mode is enabled. If using the standard serial or parallel interfaces, be sure your host system can handle the volume of output from the CY550!

Note that the position may not be reported on every step, if the step rates are too fast for the position report. For example, if the position is reported through the serial interface at 9600 baud, then the CY550 can only send one character about every millisecond. At step rates above 1000 steps per second, the CY550 will take at least one step in the time it takes to send only one character of the position display! Thus, the CY550 may take many steps before the next display report can be sent.

In the normal commanded motions (G and P commands), the selected step rates are not effected by the live position display. The position display occurs as a background function to the stepping motion. However, every position display will be accurate for the position being reported. That is, the CY550 will not erroneously report a partial position from one step due to taking additional steps during the report. The position is sampled at one point, then the position display is formed from that sample and output.

When the step rate becomes too fast for the CY550 to perform the position display as a background function, due to not being able to sample the position successfully, the position reports will not occur until the step rate slows down again. This will occur at the highest rates of stepping, generally with Rate parameter values below 75, and indicates that there is not enough time between steps for the CY550 to sample the current step position.

Live position displays occur in the same format as they would from a Query (?) command. See the chapter on Miscellaneous Commands for a more complete discussion of the Query command.

The two bits MB0 and MB2 select what device will be used to display query outputs, string value outputs, and live position displays from the CY550. Two basic display choices are a serial display or a parallel display. The least significant bit, MB0, selects between these two options.

When a serial display is selected, by MB0 being zero, the standard serial interface is used for display output functions to the host. This is the power on default option.

When a parallel display is selected, by MB0 being one, MB2 selects between a standard parallel output, and a special Hewlett Packard LED display.

When MB2 is zero, the standard parallel output is selected. This uses the IO_REQUEST and BUSY signals to control the reading of information from the CY550. Also, the CY550 will drive the FPL signal low (shared function with USRB6) when it has something to output. This protocol will work with a standard two-line handshake or with the parallel functions of a CY233 Network controller. The FPL function of USBR6 is automatically enabled when the standard parallel output is selected.

Note that this handshake allows the host system to control the speed at which the CY550 will generate output data, since it will output one character per handshake. The handshake timing is driven by the host system.

B7	B6	B5	B4	B3	B2	B1	B0	
				x	0	0	0	Standard Serial
				x	0	x	1	Standard Parallel
				x	1	x	1	HP LED Parallel

When MB2 is one, the special HP LED display is selected. In this case, all output from the CY550 goes to the local HP LED display, and a special signal, HP_SEL (shared function with USRB7), is generated while the CY550 is writing to the display. More information about support for this display is provided in a later section. This display is well suited for the live position updates selected by MB4 being set to one.

When a Mode command is issued with MB1 set to one, the CY550 enables the DTR/ function on User Bit signal USRB6, pin 27. This signal also shares the FPL function for the standard parallel interface, or can act as a normal I/O bit. Care should be used in assigning alternate functions to this pin. If it must be used for DTR/ functions, other uses should not be shared on this pin!

When the DTR/ function is enabled, the CY550 will only transmit serial characters when the signal is low. If DTR/ is high, it means the host cannot accept serial data, so the CY550 waits before transmitting a new character. Note that up to one full character may be transmitted by the CY550 after the DTR/ signal goes high, due to buffering of the serial characters by the CY550 UART circuitry.

Once the serial transmissions have been suspended, the CY550 will wait indefinitely for DTR/ to go low again. No other command processing will occur if the CY550 is trying to send a serial character, but is being held off by a high DTR/.

When MB3 of the mode register is set, the CY550 XON/XOFF communications protocol is enabled. This protocol provides "in band" flow control using the XON (11h) and XOFF (13h) control characters. Both the host and the CY550 may use these characters to control the transfer of serial data between the host and CY550.

When the XON/XOFF protocol is first enabled, the CY550 assumes that transmission and reception is enabled in both directions. It will not wait for an XON character before sending out serial data.

If the CY550 receive buffer is filling up, it will send an XOFF character to the host. This is similar to the CTS/ signal turning off, and is sent at the same time as the CTS/ signal is changed. The host should stop transmitting characters when the CY550 sends the XOFF character. Once the CY550 has processed the received characters, and is ready for more serial data, it will send the XON character, indicating to the host that more characters may be sent.

Similarly, the host can send an XOFF character to the CY550 to suspend transmissions from it, and resume transmissions with the XON character.

In both directions, the XON and XOFF characters may be sent by devices even if their normal transmissions are suspended. In the CY550, the XON and XOFF characters are used only by the communications protocol, and are never seen as command characters. Note that this protocol works best in the ASCII command mode, since the binary values for the XON and XOFF characters would not be passed into the CY550, and would effect the communications protocol instead of being used as data characters!

Mode bit MB5 is used to enable the auto incrementing of the input and output address pointers after performing an extended I/O input read or output write. When MB5 is one, auto incrementing is enabled.

Only one pointer will be incremented, depending on the operation being performed. The input pointer will be incremented after a read, and the output pointer will be incremented after a write.

The auto increment function makes it easy to implement a block copy operation within a loop. It also makes it easy to perform a scan and test function, without having to set the pointer values for each input or output.

Finally, if MB6 is one, the CY550 uses the lower four bits of the mode byte value to select the serial baud rate. This allows a parallel command to set the serial rate to a desired value, without tying the I/O Request and Busy lines to a particular state, and allows a subsequent parallel command source to set up the CY550 for serial communications, such as showing messages on a serial display.

Also, if a local external memory is used, and an "Auto-Start" program is enabled, that program can set the baud rate to a specified value, using a Mode command with MB6 set to one.

For this function, the CY550 does not store the mode byte value in its internal Mode register, so if MB6 is set, **no** other operating modes of the CY550 are affected, only the baud rate selection. Bits MB0 to MB3 are treated as a binary value, selecting the baud rate as shown in the table below:

MB3-MB0	Selected Rate @ 11 MHz
0	300
1	600
2	1200
3	2400
4	4800
5	9600
6	19200
7	57600
8	Self Adaptive (up to 57600 baud)
9 - F	Undefined

When crystal frequencies other than 11 MHz are used, the fixed rates will be scaled linearly by the crystal frequency, while the self adaptive rate will attempt to compensate for the new crystal frequency. For most crystal frequencies above 11 MHz, the CY550 can adapt successfully to baud rates of 4800 baud or less, with a small enough error to insure reliable operation of the serial interface. However, at 14.7456 MHz, the CY550 can adapt to all baud rates up to 19,200, plus two non-standard rates of 38,400 and 76,800 baud.

The CY550 will set the baud rate during the execution of the Mode command, and the new rate will be in effect until a hardware reset occurs or another Mode command is issued, with MB6 set. When the adaptive rate is selected, the CY550 must receive two carriage returns before the operating baud rate will be defined again, so this selection may only be used if serial data is coming into the CY550. No transmissions will occur until the baud rate is again defined.

Error Status Bits

As mentioned earlier, the Mode command argument is actually a 16 bit value. The lower 8 bits define the value for MB0 to MB7, and set the CY550 modes, as described above. The upper 8 bits overwrite the Error Status bits, allowing the host to clear these bits after the CY550 reports an error. A normal argument to the Mode command would only specify the lower 8 bit value, leaving the upper 8 bits as zero. This would clear any Error Status bits previously set by the CY550.

The values of the Error Status bits are reported to the host as part of the mode query command **? O**<cr>. The output of this query corresponds to the Mode command argument value. The least significant byte is the current setting of MB0 to MB7, while the most significant byte is the Error Status bit settings as follows:

ES7	Running a program from external program memory
ES6	Jogging by JOG signal
ES5	reserved
ES4	Adaptive serial mode did not receive a carriage return
ES3	Serial receive buffer overflow
ES2	External memory write error
ES1	First Rate parameter out of range
ES0	Rate parameter out of range

When all error status bits are zero, there is no error to report. The CY550 will set the corresponding bit when it detects an error, and the bit will stay set until it is reset by the host system with a Mode command. A mode query will not clear the error status bits.

Status bits ES6 and ES7 are not error indicators. They indicate a certain state for the CY550. These bits are set and cleared as the CY550 changes state. Note that a Mode command that overwrites these bits will not force a state change, but might erroneously report an incorrect state in a subsequent mode query.

ES0 and ES1 indicate out of range parameter values. The Rate command parameter value is limited to values of 67 or greater. If a value less than 67 is specified, the value will be ignored, and the ES0 bit will be set.

The First Rate parameter is limited to values between 0 and 119. If a value of 120 or greater is specified, the ES1 bit will be set. The CY550 will accept the out of range value, but it will then force the value to 119, the maximum First Rate allowed.

When ES2 is set, it indicates a problem writing to the external memory. The CY550 performs a verify after writing each byte to memory, as part of the time out support for local EEPROM memory. If the value read back does not match the value written within about 15 msec of the write, the CY550 gives up on the write function and sets the ES2 bit.

Note that the verify is only performed on memory writes that define programs, following the Enter command (E<cr>). It is not performed for extended outputs of the % command.

If ES3 is set, it indicates a serial buffer overflow. This is normally due to the host system not reacting to the CTS/ signal or XOFF character (if enabled). The CY550 can be busy during some commands, such as a time delay, or wait for motion to finish, and cannot accept more commands during this time. As the serial buffer fills up, the CY550 will bring the CTS/ signal high, and if enabled, send the XOFF character. The host should stop sending characters at this time, until the CY550 has a chance to process the characters already received. If characters continue to be sent after CTS/ is turned off, the buffer could overflow. The CY550 can accept two more characters after CTS/ turns off. On reception of the third character with CTS/ off, the ES3 bit will be set, and that character, along with any others will be lost.

The ES4 bit indicates a problem with the adaptive baud rate initialization. When adaptive baud rate is selected, the host must send two carriage returns to the CY550 to set the communications baud rate. If the CY550 detects some characters during this time, and tries to set its baud rate, but the character patterns do not match the carriage return code, it will set the ES4 bit. This normally would mean that the baud rate has not been set correctly, and serial communications may not occur successfully. If there is a parallel interface to the CY550, the parallel side could query for this error. However, it may not be possible to send serial commands in this case.

The ES6 bit is a status indicator, which is set when the CY550 is jogging, due to an input signal on the JOG pin. It will be cleared when the CY550 stops jogging.

The ES7 bit is also a status indicator, and it is set when the CY550 begins executing a program from the local external memory. This bit will stay set as long as the program is executing, and is cleared when the CY550 stops running the external program, and reverts to the immediate command mode. A host processor may use this status bit to know when the CY550 has finished running the program commanded by the host.

11 Miscellaneous Commands 11

Initialize Command

Finally, the CY550 has several miscellaneous commands for general control purposes. One command provides a software reset for the CY550. It is:

I	Initialize CY550

When the **Initialize** command is processed, it acts like a power-on reset. The CY550 command mode is changed to direct command mode, all I/O signals are brought high, and the serial baud rate is reset. Any program that was executing from external memory is stopped if an I command is read.

The **Initialize** command or power-on reset also fix the CY550 stepping parameters as follows:

R	0	** undefined value
S	0	** undefined value
F	3	
N	10	

and the current position register is cleared to zero.

The current values of these registers will always be changed back to the defaults when a reset is performed or an I command is executed.

Note that the Rate and Slope parameters are set to zero, but that these are undefined values for the parameters. After any hardware reset or I command, you must specify good Slope and Rate commands to define these parameters properly. The CY550 will not accelerate properly or perform normal motions until the Slope and Rate parameters are set to proper values.

Time Delay Command

The CY550 also provides a time delay function, with:

D Del16 Delay for specified milliseconds

The parameter for the **Delay** command is a 16 bit value, calibrated in milliseconds when running at 12 MHz. This command simply causes the CY550 to wait for the specified time before going to the next command.

When used in an external memory program, the command allows you to pause for a known time between program functions. For example, take a specified motion to move a part into place, then delay for 3 seconds while some operation is performed, then remove the part.

The value of the Delay parameter allows a time delay between 1 millisecond and 65 seconds when the CY550 is running at 12 MHz. Longer delays can be created by repeated delay commands, or delay commands in a loop.

Also note that very slow stepping is possible by a loop that consists of a single-step relative motion followed by a time delay. For example:

```
Y 10<cr>                  Select address 10 to start code
E<cr>                     Enter to memory

    N 1<cr>                          Set to 1 step relative
    +<cr>                            Select CW direction
    G<cr>                            Take a step
    V<cr>                            Wait for step to finish
    D 2000<cr>                       Delay 2 seconds
    L 50,16<cr>                      Loop back to G command, 50 times
    0<cr>                            Stop program

Q<cr>                     Quit entering program to memory
Y 10<cr>                  Reset address
X<cr>                     Run the program
```

The above example will take 50 steps, with a delay of two seconds between steps. Other delay values will provide different step rates. With this technique, it is possible to generate arbitrarily slow step rates from the CY550!

Note that the CY550 suspends "live" command processing during a delay command. Delays may be executed "live" while a motion is in progress, but other commands are not executed during the delay! Also, if the delay is executed during a motion, the delay time will be longer, due to the overhead of stepping during the delay. In this case, you must experiment with the proper delay time parameter to give the desired real time delay. The amount of error due to stepping depends on the step rate being used. Higher step rates require more overhead, and thus smaller values of delay parameter for the same real time delay.

Query Command

Display CY550 Status

Along with display strings, the CY550 allows you to query the values of various working parameters and status. The command is:

? Cmd	Query specified parameter value

The "Cmd" portion of the **Query** command is a single letter, selecting the parameter for display. Possible choices are:

N	Number of Steps value
P	Current position value
R	Final rate value
F	First rate value
S	Slope value
Y	Memory address pointer value
B	User bits input value
G	Motor control signals state
O	Mode and Error Status values
#	I/O byte register value
I	Input (read) pointer value
W	Output (write) pointer value
V	CY550 chip version

When the Query command is received or read from memory, the CY550 displays the parameter value as follows:

```
X=vvvvv<cr>        ie:        S=00220<cr>
```

Where "X" is the parameter letter, and "vvvvv" is the 5 digit ASCII decimal value for the parameter. Parameters that only have 8 bit values will have leading zeros for the unused digits. This is a fixed format output while the CY550 is in the ASCII command mode. Since Position and Number of steps are 24 bit values, these two queries will output an 8 digit number, not the standard 5 digit number.

In the Binary command mode, the "vvvvv" portion is replaced by two bytes that represent the binary value of the parameter, sent most significant byte first. The carriage return is not sent in the Binary command mode. Position and Number are sent as three byte values.

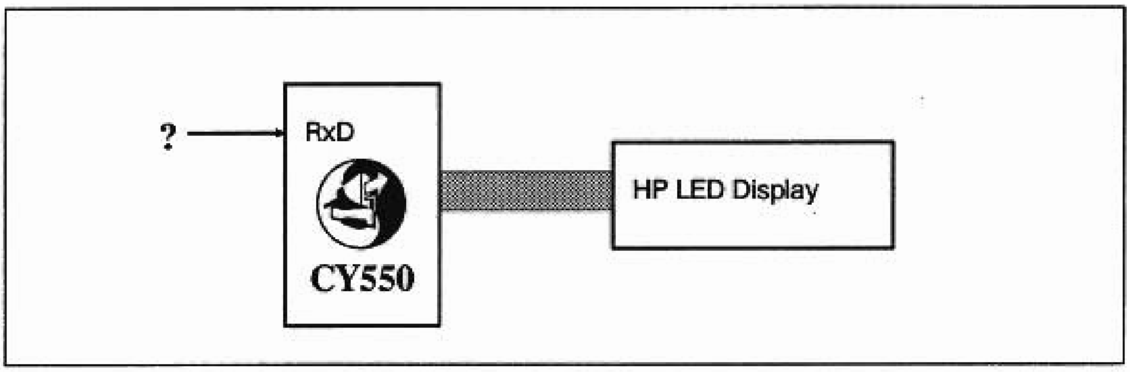
Also, since the CY550 uses signed positions, the position output in the ASCII command mode uses a "+" or "–" sign as the first digit of the display, followed by seven numeric digits. For example:

```
P=+3124589<cr>        or        P=-02475194
```

The sign is always included in the position display, even though you do not specify a " + " sign for positive positions in the Position command argument.

In the Binary command mode, the position display uses the 2's complement binary format for the current position value, sent as three bytes. If the most significant bit is one, it indicates a negative position in Binary format.

The Query command output message is sent to the currently selected display device, so it is possible to send the result to something other than the command source. For example, a host computer could issue the Query command over the serial interface, with the CY550 displaying the result on a local HP display.



If the HP display is used for number of steps (N) or current position (P) queries, the "N = " or "P = " portion of the display output is not used, since the parameter values are already 8 digits, and will just fit onto the display.

Each query command will output the current value of the requested parameter. For the User Bits query, a 16 bit value is returned, with the lower 8 bits corresponding to the CY550 User Bits, and the upper 8 bits corresponding the the CY550 Data bus.

The motor signal query returns the current settings of the CY550 motor control signals, pins 1 to 8. The pin 1 value corresponds to the least significant bit, and pin 8 corresponds to the most significant bit. The bit values reported match the values indicated on the CY550 signal pins, with a logic high corresponding to a 1 in the reported value.

This query allows the host to determine if the CY550 is stepping, what the current stepping direction is, if a limit switch has been hit (and which limit it is), when the CY550 is slewing, and if a signal is driving the INHIBIT_ABORT pin.

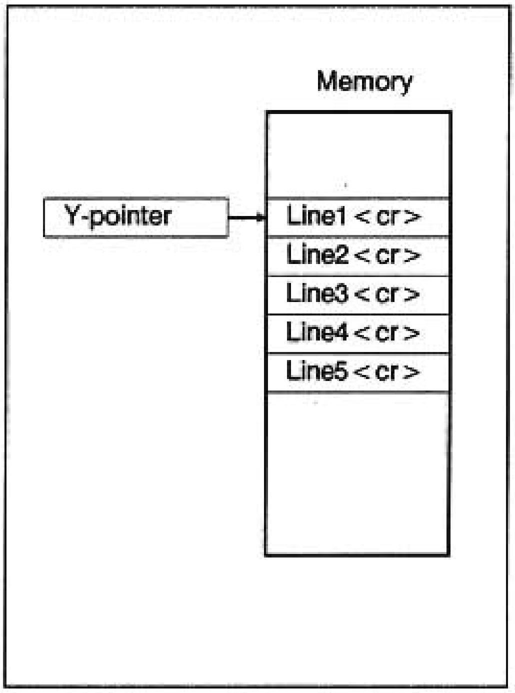
As described in a previous chapter, the mode and error status query returns the current value of the mode register in the lower 8 bits of the value, and the current error status bits in the upper 8 bits of the value. If an error has been detected by the CY550, a corresponding error status bit will be set. Two bits also indicate if the CY550 is running a program from external memory or if the CY550 is jogging.

The version query returns a value corresponding to the version number of the CY550 chip. For the first release, this version number is "05501" in ASCII format. Subsequent releases, if any, will increment the least significant digit value.

Display Memory Contents

One more parameter may be issued for the Query command, the "M" parameter. The format for this form is special, because it also includes a second parameter, which is a count value. For example:

? M,5<cr>



For this Query command, the "M" is selecting a query of the external memory, and the count indicates how much to display.

The memory display starts by showing the value of the memory address pointer, since the query will start at this address (ie: M=00001). This is followed by the actual memory contents, as specified by the count value. Each carriage return read from the memory counts as one line. This format operates in the ASCII command mode.

In the Binary command mode, you may still perform a memory query, and the query command takes two parameters, the "M", and the count. However, in this case, the count is the number of bytes to read from the memory, not the number of lines.

Message Display Command

The CY550 also provides a mechanism for displaying messages that you define. This is done by:

"String" Display all characters between quotes

Normally, these messages would be part of a program in external memory, and can announce specific items on the display. The messages can also be prompts to an operator to turn something on or off at particular points in a program, with the CY550 then delaying, or waiting for a signal before continuing with the program.

The string format is very general. Any character is allowed between the quotes (except another quote), including ASCII control codes and carriage returns. The only character that terminates a display string is the double quote character (ASCII 22H), also used to start the string. This is the only character that may not be part of the string.

Strings are stored in external memory if they are received following an Enter command, just as regular CY550 commands. Every part of the string, including the starting and ending quotes is saved. The next CY550 command immediately follows the ending quote of the string. In Binary mode, strings do not include data counts, they are treated like ASCII mode for the duration of the string.

Visual Program Markers:

A useful feature of CY550 displayable messages is their use as "markers" to visually indicate the status of a program to human operators. This can include informational messages, such as "waiting for input", "paused", and "stepping", and user prompts, such as "Push Start", "Check Stock", and "Restart System".

By combining the messages with commands that wait for User Bit signals, or take conditional actions based on User Bit inputs, a highly interactive control program can be implemented by the CY550.

Message Storage in External Memory

```
        Y  100<CR>
        E<CR>
100         R  200<CR>
106         S  175<CR>
112         F  2<CR>
116         "READY TO STEP <CR>"
132         W  3<CR>
136         P  1234<CR>
143         0<CR>
        Q<CR>
        Y  100<CR>
        X<CR>
```

100	R		2	0	0	<CR>	S	
108	1	7	5	<CR>	F		2	<CR>
116	"	R	E	A	D	Y		T
124	O		S	T	E	P	<CR>	"
132	W		3	<CR>	P		1	2
140	3	4	<CR>	0	<CR>			

Every character received by the CY550, after an Enter command, is written to external memory.

Embedded Control Characters and their Use:

The string command allows any 8 bit characters (except ") to be transmitted as part of the string. This will allow embedded control codes and special "setup" commands to be issued to peripheral devices, such as the CY325 LCD windows controller. For example, the string:

```
"<Ctrl-C>V 0D6h<cr>
 <Ctrl-D>Hello<cr>"
```

puts the CY325 into the command mode (Ctrl-C character), then issues a Viewport command, which defines a currently active window for display. The window selected uses one quarter of the 711 display, from the middle. The Ctrl-D character then places the CY325 back into the display mode, and the message "Hello" is written to the selected window. The carriage return advances the cursor to the line directly below the "Hello" message.

The above sequence is sent as one string message by the CY550, since all the commands and control codes are embedded within one character sequence, between the double quote characters.

Hewlett Packard (HP) LED 8 Digit Display

Finally, the CY550 has a special command for controlling the Hewlett Packard display. The command is:

[Addr,Cnt,D1,...,Dn Special Hewlett Packard display support

This HP Display command allows you to manipulate some special features of the HP display, such as brightness control or custom character fonts. Use of this command is optional, since it changes the defaults as set by the CY550. It is not required for normal operation of the HP display.

HP LED Display Details

The address parameter is the display address used to write the data and control information. For example, 30H is the address for the HP control register. The count parameter is the number of data bytes to write, starting at the specified address. If the count is zero, nothing is written, but the internal display address is changed to the specified first parameter value. When the count is non-zero, the following bytes are written to the display, with the count determining how many bytes are written.

One address is treated specially by this command, address 30H, for the control register. If address 30H is followed by a zero data count, the CY550 default control value will be selected. Otherwise, the value of D1 will become the new control register value, and can be used to change the display intensity or turn on the blinking feature.

For more information about the HP display and CY550 support, see a later section on Display Support and the HDSP-211x data sheets (available from Hewlett Packard). The data sheet defines the registers and functions of the display

Also note that the HP display is limited to 8 characters, so the Query output format automatically changes when the HP display is selected, and the Position or Number parameter values are queried. In these cases, only the 8 digit value is shown, not the parameter letter and equal sign. For example, instead of showing:

```
N = 12345678 < cr >
```

as it would for a normal serial or parallel display, the CY550 simply shows:

```
12345678 < cr >
```

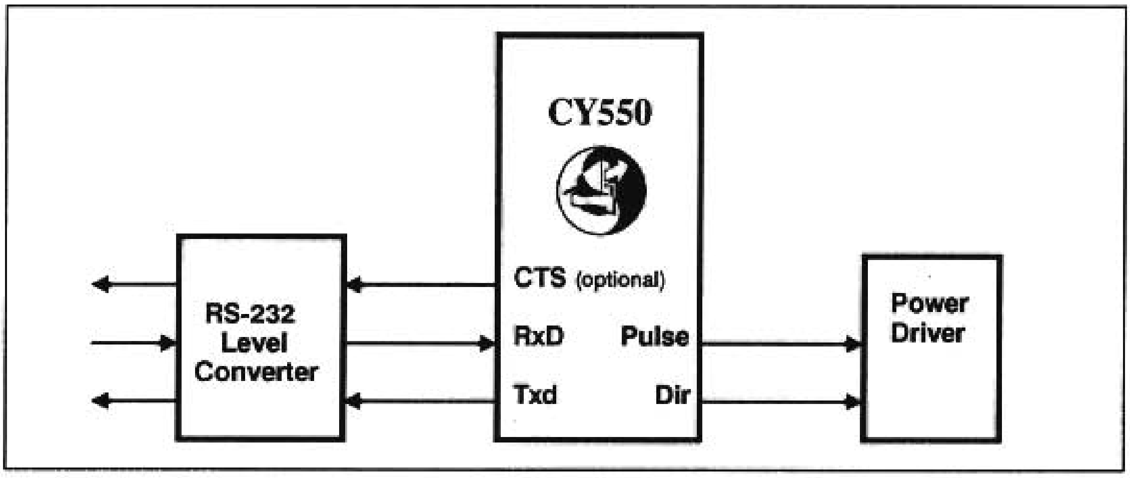
when the ? N or ? P commands are issued for the HP display.

CY550 Example Circuits

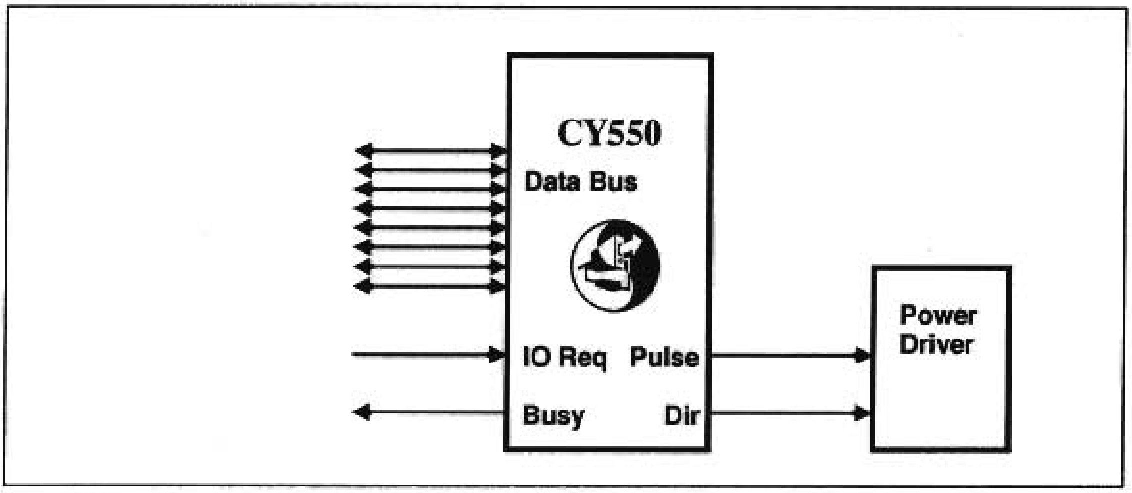
Three Example Cases

The three following circuit diagrams illustrate the wide choice of possibilities for CY550 applications circuits. In the first, only the serial command interface and the pulse and direction lines are used. This represents a minimal CY550 design. The second example is similar, but uses the parallel input bus to send commands to the CY550, with the pulse and direction lines still sent to a power driver. The third diagram illustrates a complete system with RS-232 serial input, external memory, display circuits, and I/O lines, and is implemented on the CYB-550 prototyping board.

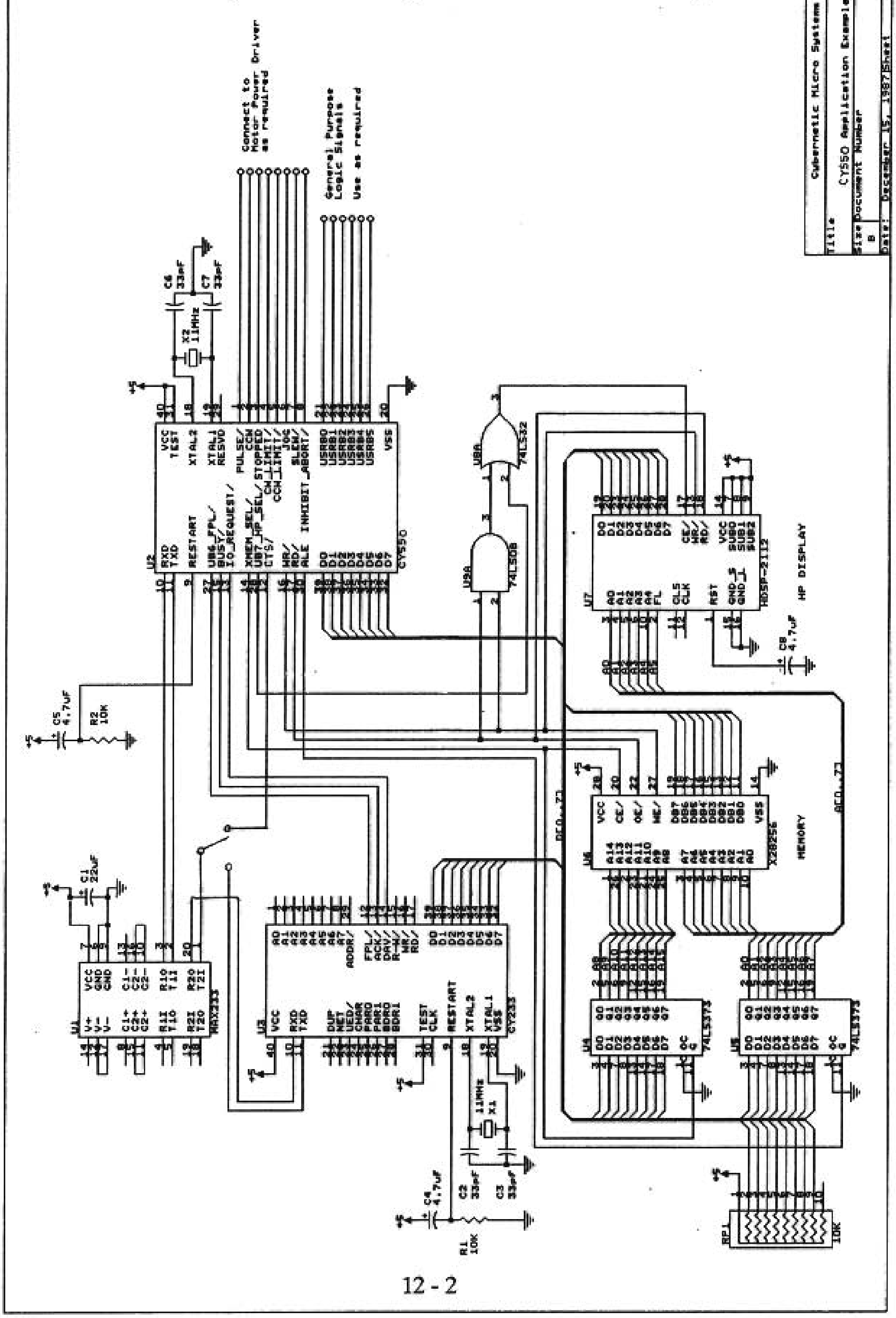
CY550 Example Circuit Diagram - Minimum Serial Configuration



CY550 Example Circuit Diagram - Minimum Parallel Configuration



CY550 Example Circuit Diagram - Maximum Configuration



13 External Memory & Extended I/O 13

Memory Read/Write

The CY550 may be operated with up to 64K bytes of external memory, which hold various sequences of commands to run. A number of CY550 commands are used to support the external memory, allowing you to set the memory address pointer, enter commands into memory, and execute commands from memory.

Some of this external 64K byte memory space may also be used for extended I/O functions, as described in a previous chapter.

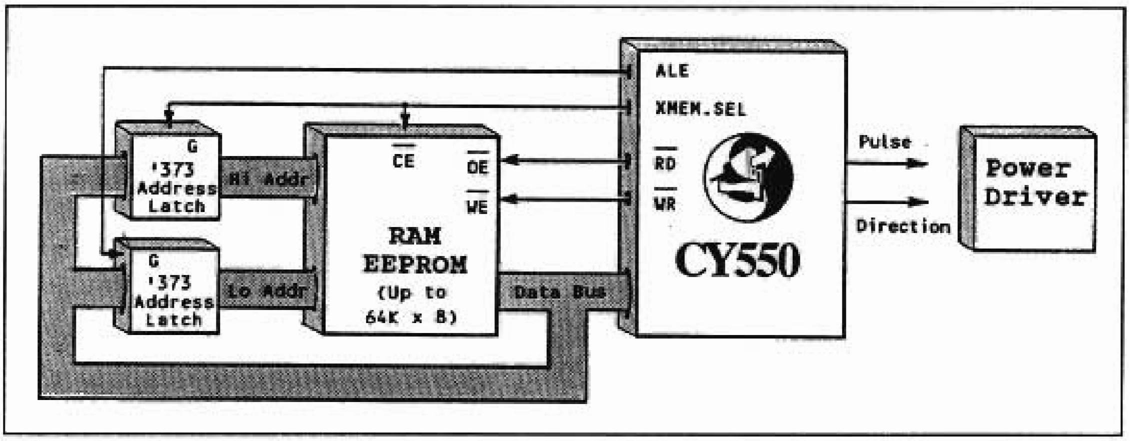
Several types of external memory may be used, including RAM, EEPROM, and EPROM. Note that EPROM memory must be programmed separately from its use in a CY550 application, so the CY550 Enter command and the ability to define the contents of memory are given up when an EPROM is used. The other two forms of memory allow the CY550 to write to memory as well as read it.

When RAM is used, the contents are lost when power is removed, unless some battery back-up power is provided. Normally, the contents of RAM must be redefined whenever the CY550 application is started.

With EEPROM, the memory can still be written by the CY550, especially if a 5 volt only version is used, but the memory contents are not lost when power is removed.

An applications schematic, provided with this manual, illustrates the use of EEPROM as the external data memory for the CY550. The device shown is a Xicor X28256 EEPROM, which has 32K bytes of memory. Smaller devices may be used if less memory is required, and a larger device, with a full 64K bytes of memory could be supported.

Memory types may even be mixed, with some non-volatile memory, and some RAM memory in the same application. In this case, the RAM memory may be initialized



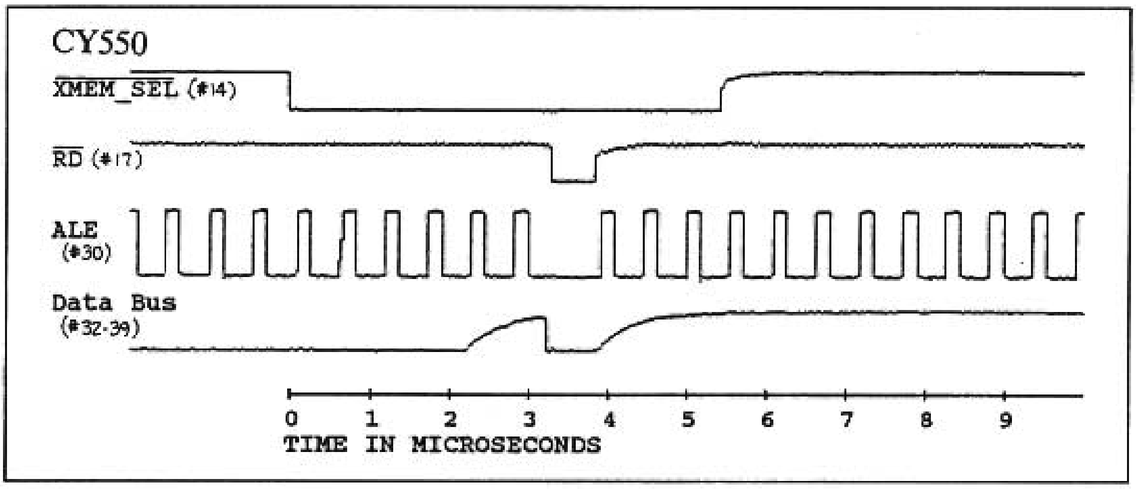
through the extended I/O read and write functions, using block copy functions. The RAM memory may also be programmed through the normal CY550 program definition command sequence, using the Enter and Quit commands.

The CY550 provides all address and strobe timing signals to the memory. Signals involved are the data bus, XMEM_SEL, ALE, WR, and RD. Special use is made of the data bus, which provides all 16 bits of address, plus the actual data transfers. Two external latches are required to hold the address. The example uses 74LS373s for this purpose. The same control lines are used for any memory or extended I/O operation. External address decoding circuitry must be used to distinguish a read or write meant for the memory from one meant for extended I/O, and produce the proper "chip select" signals from the decoded addresses.

When the CY550 reads or writes the external memory space, it first tests that the data bus is floating (FF). If the bus is latched by another process and never returns to a floating state, the CY550 will remain stuck in this test loop. Once the floating state is encountered, the CY550 places the most significant address byte on the data bus and then selects the memory by driving XMEM_SEL low. This both chip-enables the memory system and latches the upper address byte into the 74LS373, which holds address lines A8 to A15. The CY550 then performs a read or write operation.

The read or write is broken into two steps. First the lower address byte is generated on the data bus. This value is latched into the other 74LS373, using the ALE signal. Then the actual data transfer occurs. For a write, the CY550 generates the write data on the data bus, along with a write strobe, WR. For a read, the data bus is floated, and a read strobe is generated, using RD, so the memory or extended I/O circuits can now drive the bus.

When the memory transfer is over, the XMEM_SEL signal goes high again, de-selecting the external memory space. This allows the lower address byte and RD and WR strobes to be used with other hardware as well, without interference from the memory. Only the memory uses the upper address byte. If External memory is not used in the application, then the XMEM_SEL line should be tied low.



As a special feature, the CY550 performs a verify read after writing commands to the local external memory. The verify read is only performed for writes that occur after an Enter command, in which a CY550 command sequence is being defined to the external memory. It is not performed after an extended I/O output function from the % command. You must be sure that all command sequences are written to a valid memory location, so the verify read function can be performed.

The verify read is provided in support of EEPROM memories, that require a long time to perform a write operation, and cannot perform another random write or read while the first one is in progress. Many of these memories support Data/ polling, in which at least one bit of data is inverted while the memory is "busy" writing the byte. When a read is performed, the data read does not match the data written until the write operation is complete. This change in data values indicates when the memory is ready for the next operation.

The CY550 performs the verify read repeatedly until the data matches the value written, or a time out occurs. For RAMs, the match will occur immediately, so the CY550 simply continues at that point. For an EEPROM, the match will occur after the EEPROM write is finished, and the CY550 waits during this time before going on to the next function.

You should be aware of this time period delay when using EEPROM, because it is possible to send serial characters to the CY550 faster than the EEPROM can accept them, when running the serial interface at higher baud rates. Be sure to use the CTS signal to control serial interface character timing in this case, or use a time delay between each character, to prevent serial buffer overflow.

Note that the parallel interface takes care of this time delay on its own, since the CY550 will stay busy while it is writing the character to the memory.

"Auto-Start" Feature for Stand Alone Operation

Another special feature of the CY550 is the use of the external memory to automatically run a command sequence at power-up, or whenever the CY550 is reset.

When the CY550 is reset, it will read the first three locations of the external memory. If the values read are 12H, 34H, and 56H respectively, the CY550 will then switch to the program execution mode, and start reading commands from location 00003 of the memory. This allows the CY550 to work in a stand alone mode, without any commands from a host computer or other source than the memory itself. This function is shown below:

00000	12H	12H in address 0
00001	34H	34H in address 1
00002	56H	56H in address 2
00003	...	First CY550 command in address 3

This feature will only occur if the first three bytes have the proper key codes in them. Any other values in those locations will keep the CY550 in the direct command mode, waiting for commands from a host system before doing anything.

To set up the auto start function, simply record the data bytes 12h, 34h, and 56h, in locations 0 to 2 of the memory, followed immediately by the command sequence desired. For example:

```
Y  0<cr>              Set memory pointer address to zero
E<cr>                 Following sequence is recorded to memory

<Ctrl-R>4V            key sequence 12h, 34h, 56h
R  125<cr>            standard CY550 commands follow now
F  15<cr>

    . . .             additional start up commands

Q<cr>                 End of memory content definition
```

When the CY550 is now reset, it will execute the memory based commands, starting with the "R 125<cr>" command, since the key sequence was found in the first three locations. Note that there is no carriage return after the 56h. The character immediately after the 56h is the command letter of the first command in the auto- start program.

As a special convenience, you may command the CY550 to ignore the auto start command sequence, by holding the XMEM_SEL signal low while the CY550 is reset. If the signal is low, the CY550 will not test the memory contents for the 12h, 34h, 56h key sequence. Instead, the CY550 will set itself into the Command mode, and wait for commands from the serial or parallel interface. Memory access will remain disabled as long as XMEM_SEL remains low. Releasing the XMEM_SEL line after reset will allow memory access without executing the startup sequence.

This is especially useful during program development, where errors in the memory content must be corrected before the CY550 can run the memory based commands. Without this feature, you would be forced to remove the memory and delete the key sequence, before the CY550 could be operated with the memory again.

The sequence for bypassing the Auto Start tests is:

- Drive the XMEM_SEL signal low
- Reset the CY550
- Delay at least 2 msec after reset is released
- Release the XMEM_SEL signal
- Issue any CY550 commands in the direct Command mode

14 CY550 Output Display Support 14

Output Format

As mentioned in the command descriptions, the CY550 can support a number of output devices, selected by the Mode command. A standard serial output is possible, and two types of parallel output are possible.

The CY550 generates output from the string function, for user defined messages, and from the Query command, for current parameter values and status information. The exact format of the output is determined by the selected display and the ASCII or Binary command mode.

For standard serial output, the power-on default, the CY550 simply transmits each character of a string or query on the serial output line, TxD. Every character between the quotes is sent for a string, and the query parameter outputs are:

```
X=vvvvv<cr>        or        X=vvvvvvvv<cr>
```

Where "X" is the parameter letter (N, P, R, etc.), and "vvvvv" is the five decimal digit or eight decimal digit value of the parameter. The query output ends with a single carriage return character.

In Binary mode, the digit strings are replaced by the two or three byte binary value of the parameter being queried. The "X = " portion is still included in the Binary mode.

Output to Parallel Device

(Including CY233 Network Chip)

The standard parallel output can be used by a parallel connected host or by the CY233 Network Communications Controller. Both connections use the same signals, the parallel handshake control lines IO_REQUEST, BUSY, and FPL (USRB6).

When the CY550 receives a Query command or a string to send, it changes to the output mode internally, and lowers the FPL signal. It then waits for a low level on the IO_REQUEST signal, indicating that the host or CY233 is ready to take a character.

The character to output is placed on the data bus, and the CY550 drives the BUSY line low, indicating that a character is available.

The CY550 will now wait in this state until the host or CY233 drives IO_REQUEST high again. This indicates that the character has been accepted, so the CY550 removes the character from the data bus, and raises the BUSY line again.

This handshake process repeats until all the characters of the Query output or string have been sent. The CY550 then drives the FPL line high again, and goes back into the command mode.

A CY233 interface is included with the "CY550 Applications Example", shown in a previous chapter.

The format of parallel output queries is the same as that for the serial interface, using exactly the same character sequences. This is true for ASCII and Binary modes.

Output to 8 Digit LED Display

The final selection for a display device is a special parallel display, using the Hewlett-Packard HDSP-211x displays. These displays are very compact, LED based, 8 character, dot matrix displays. They have programmable intensity and custom definable character fonts.

The displays connect to the CY550 over the parallel data bus, the lower address byte, and the HP_SEL (USRB7) signal. The RD and WR strobes also provide data transfer timing for these displays.

Typical LED Register Displays

Typical Position Display	+	0	0	0	0	0	0	0
Typical Rate Display	R	=	0	0	1	0	0	
Typical Slope Display	S	=	0	0	2	2	0	
Typical FirstRate Display	F	=	0	0	0	1	4	
Typical Number of Steps Display	0	0	0	0	0	2	0	0
Typical Memory Pointer Value Display	Y	=	0	0	0	0	1	
Typical User Bit Display	B	=	6	5	5	3	5	

The applications example schematic includes the proper connections for the HDSP-211x displays. This connection scheme must be used for proper operation of the displays, because the CY550 assumes the address lines are connected as shown to access the display control registers and data buffers.

When the HP display is selected, any string messages are sent to the display verbatim. Be careful that messages are not longer than 8 characters at a time. Carriage return codes are not displayed, but they reset the character address to the left most display character, and cause the display to be blanked before the next character is written. The blanking function is performed just before writing the next character, not at the time the carriage return is processed.

Query command outputs follow the same format as the standard serial and parallel display outputs, except for the 8 digit values of the Position or Number of steps parameters, which only display the value, so the HP displays can show one complete parameter value at a time. The display is blanked, and a new parameter value is written each time the Query command is used.

The CY550 takes care of any timing involved in running the HP display, and generates the proper control signals. Also, a special command ([ ) allows you to customize the display control for your desired intensity or other functions.

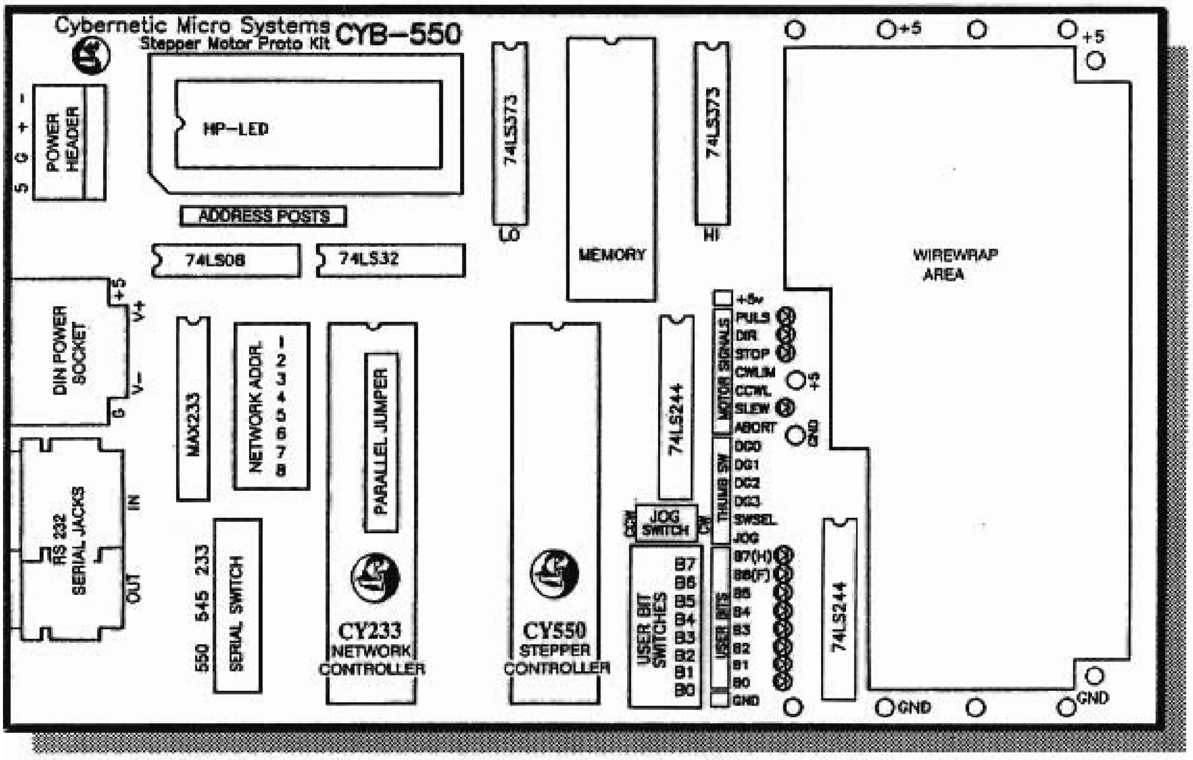
The CYB-550 Prototyping Board

The CYB-550 Prototyping Board implements the Example Circuit - Maximum configuration as discussed earlier, supporting all features of the CY550. It provides the most complete support for the CY550 of the boards discussed here. The CY550 may communicate serially with the host over an RS232 interface using the RJ11 jacks, while parallel communications are supported on a 20 pin header.

Multiple axes or networked controllers are supported using the CY233 Network Controller. The CY233 communicates with the CY550 over the Parallel bus.

This board also contains the interface circuits to support an HP or Siemens 8 character LED, for displaying parameter values, prompts, and messages. Up to 256K bits (32K x 8) of memory are addressable at the memory socket, which can accept Prom, Eprom, or EEprom. It is possible for such a system to operate without the use of a host computer. The CY550 with local memory can contain the routines required to run the system, while the display provides the mechanism for system control by the operator.

A Wirewrap area is provided for implementation of driver circuits, or other custom circuitry. Based on the single height VME format (100mm x 160mm, approximately 4"x 6.3"), the board requires only +5v (@ 650 mA). The RS232 level voltages are generated by the Max233 on the board.



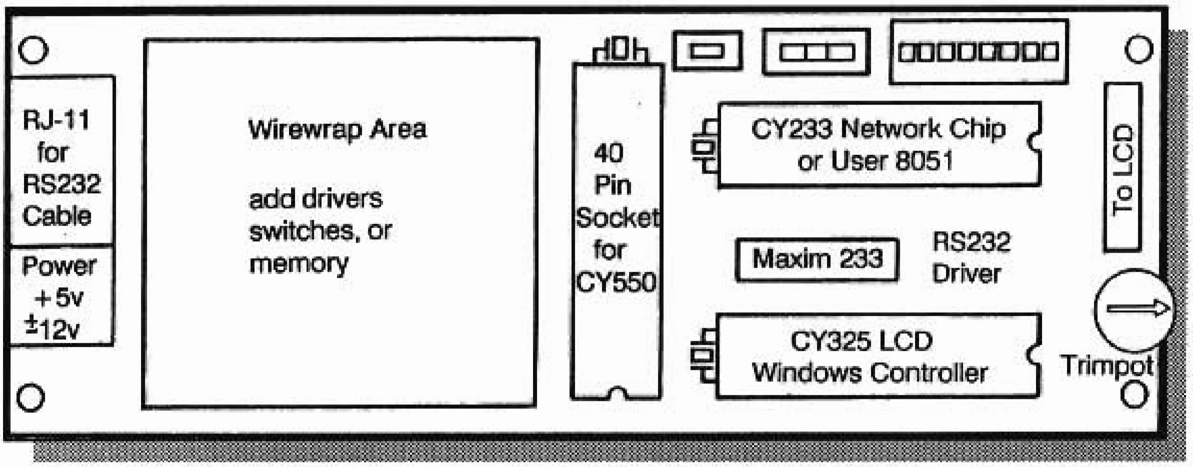
The basic board is provided in kit form, with all parts necessary to assemble a working board. Separate options are available, including an LED Display, Memory EEprom, CY233 Network Controller Chip, power supply, and serial cable.

The CYB-003 Prototyping Board

Those CY550 applications that use the CY325 and liquid crystal display, can be supported by Cybernetic Micro Systems CYB-003 board. This board contains the interface circuits and connections to support the CY325 and display unit. The board is the same size as the popular 711 LCD display, which is 8 lines by 40 characters.

The board contains a socket pattern that will accept the CY550, and allows it to communicate serially with the CY325. Also included are an RS-232 interface and RJ-11 jacks for serial connection to the board. The wire wrap area makes it possible to add the motor driver interface, the local external memory, and other special interfaces for the CY550.

The combination of the CYB-003 board, with the CY550, CY325, and any interface logic, plus the 711 display, makes a very compact, display oriented, stepper controller system. This is especially useful in stand alone applications, where the local memory holds the CY550 stepping routines, and the CY325 plus display are used for the operator interface, status display, and prompting messages.



The CYB-003 can also be used in conjunction with the CYB-550 board using a serial link between the CY550 and the CY325. Using both boards together provides all the features of both boards without additional modification.

Timing and Control

This section contains more detailed timing and applications information for various signal groups of the CY550. Included are the command interfaces, special external hardware support, and motor control signals.

The Parallel Handshake

The CY550 parallel command interface uses three control signals, plus the data bus lines for actual data transfers. The control signals are IO_REQUEST, BUSY, and FPL, which is a shared function with USRB6. IO_REQUEST is an input, driven by the device sending commands to the CY550, while BUSY and FPL are outputs, indicating status of the CY550.

The IO_REQUEST signal is used to control the parallel handshake functions. The signal is normally high when no commands are being sent to the CY550. To issue commands, wait until the BUSY line is high, indicating that the CY550 is ready for commands, then put a command byte on the data bus lines, and drive IO_REQUEST low. Keep the data signals and IO_REQUEST stable until the CY550 responds by driving BUSY low. Note the response time is normally around 25 microseconds (usec), so the CY550 is slower than an I/O peripheral or memory device, and some systems may require a latch on the data bus signals. Once the BUSY signal has gone low, the IO_REQUEST signal should be driven high again, and the data signals may be removed. A typical handshake waveform is shown below.

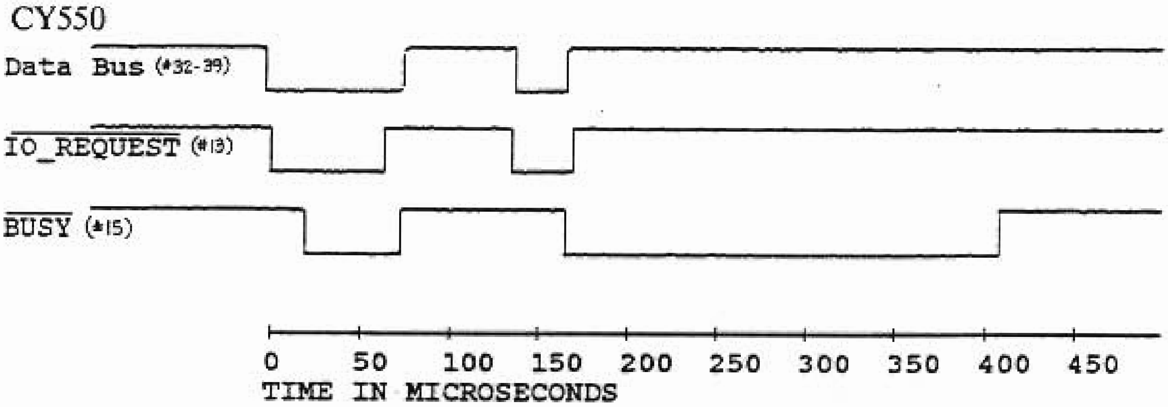


Figure 16.1 Typical Parallel Handshake Waveform.

Every command character must be sent to the CY550 with the above described handshake. Timing for the handshake will vary somewhat, and the CY550 will go busy for longer periods of time after the last character of a command is issued. This represents the actual execution time of the command.

When the command is a query command, the CY550 will generate a response to the query as part of the command execution. If the mode register is set so that query responses come out the standard parallel interface (MB0 = 1 and MB2 = 0), a handshake will also be used for the data generated by the CY550. In this case, the CY550 will drive the FPL signal low to indicate that it wants to generate some output to the parallel interface. The CY550 then waits for the IO_REQUEST signal to go low, indicating that the host system is ready for data. The data byte is written to the data bus, and the BUSY signal is driven low. This indicates that data is available to the host. When the host drives the IO_REQUEST signal high again, the bus is cleared, and BUSY is brought back high also.

In the ASCII command mode, the FPL signal will continue to stay low as long as the CY550 has data to generate. This is an indication that the host should continue reading data bytes from the CY550. The FPL signal will be brought high when the CY550 writes a carriage return code, indicating the end of the response.

In the Binary command mode, data may have any value, so each byte sent out will toggle the FPL line back high again. When there are multiple bytes to send, the FPL signal will be brought low as the CY550 is ready to generate the next data byte value.

This behavior is compatible with the CY233 LINC device, generating one message for query responses in the ASCII mode, and one message per data byte in the Binary mode. A typical query response waveform is shown below.

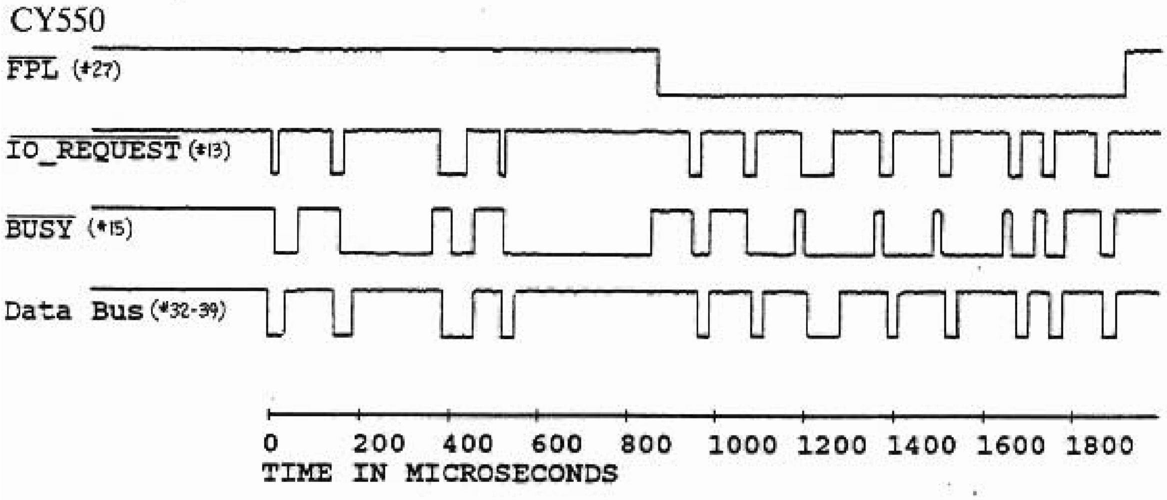


Figure 16.2 Typical Parallel Handshake Response Waveform.

Immediate Commands during External Memory Program Execution

In addition to the ability to execute commands in a live mode during motions, the CY550 is also capable of accepting and executing immediate commands from the host while also executing a program from the local external memory. However, the handshakes for this function are a little more complex than the standard command handshake. More details are provided in this section.

The FPL signal is very important in implementing the handshake protocol for immediate commands, especially if the CY550 is also generating query responses or quoted messages from the local memory program.

The FPL signal, driven by the CY550, can be used to control the direction of data on the CY550 local data bus. When FPL is high, the CY550 can accept command inputs, and when FPL is low, the CY550 has output data for the parallel host.

In the CY550, the FPL signal is driven low when the CY550 has data to output, and it is removed after the host has read the last character, and brought IO_REQUEST high again, but before the CY550 brings BUSY high. This timing is compatible with hardware implemented handshakes, requiring no time delays or complex logic to implement.

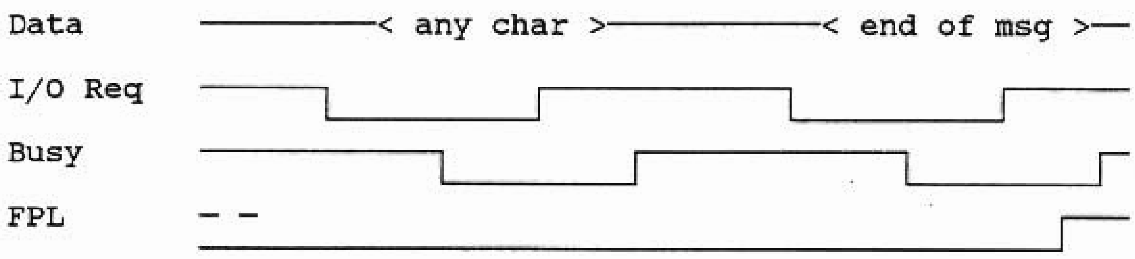


Figure 16.3 The FPL logic waveform.

Notice that the FPL signal is removed at the end of the message, while the CY550 is still Busy, but after the IO_REQUEST has been driven high again.

Also, there is a fundamental handshake change required for the CY550 to accept immediate commands from the host, while reading and executing commands from the local external memory.

If the CY550 is to execute a program from the local external memory, and accept immediate parallel commands at the same time, there is a resource conflict involving the data bus. This bus is used to both read commands from the memory, and read commands from the parallel host, while only one device is allowed to drive the bus at a time. Notice that the CY550 may also drive the bus, while outputting messages to the host or the local HP display! This makes the data bus a very critical resource in CY550 designs.

In order to allow immediate commands to be issued at any time, even while the CY550 is reading data from the local memory, we must implement a revised handshake protocol for the immediate commands, which prevents corruption of data on the data bus. In the CY550, this is done without using any more signals, so no other User Bits are given up to perform this more complex handshake.

Instead of using additional signals, immediate commands are issued with a modified first handshake, involving no character! That is, the host drives IO_REQUEST low, and waits for the BUSY signal to go low, but the data bus is not driven. This allows the CY550 to finish reading and executing any command from the local external memory, that is in progress when the host decides to issue the immediate command.

After the memory command is finished, including possible output of a message or query response, the CY550 will drive BUSY low, with FPL high, to indicate that it is now accepting the host's immediate command. This is similar to a normal command handshake, except that the command letter is not put on the data bus.

Since the data bus is not driven during this first handshake, the CY550 reads an 0FFh data byte (due to the pull-up resistors on the local data bus), and will ignore this byte.

This special, first character handshake is followed by the normal command format, starting with the normal command letter, which the CY550 reads with a normal handshake. After executing the immediate command, the CY550 resumes external memory execution, unless the immediate command was the "0" command, which stops the execution.

Note that the first handshake, without driving the data bus, is only required by the parallel interface during immediate commands, issued while the CY550 is also running a program from the local external memory.

When the CY550 is only executing immediate commands from the host, or when the host uses the serial interface instead of the parallel interface, the "0FFh" first character is not required. However, if you issue a first handshake with no command character (read as 0FFh by the CY550) in these cases, the CY550 will still ignore that character.

The examples below, show the initial handshake of an immediate command during external memory execution. Notice that in the general case, FPL might go low before BUSY goes low. In this case, the CY550 has decided (due to a command from the memory) to output a message to the host rather than accept the immediate command input. This is shown in the first example waveform below. The CY550 is driving the data bus when FPL is low.

The host must accept the output message from the CY550 before continuing with the immediate command in this case. The state of FPL during the first handshake will determine if the CY550 is taking the command from the host (FPL high), or if the CY550 is outputting a message to the host (FPL low).

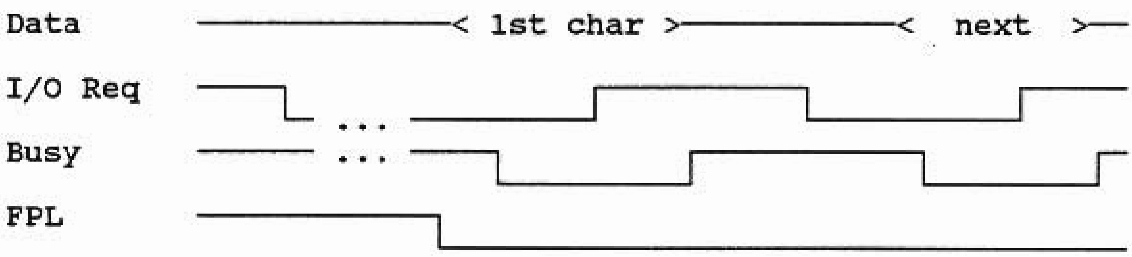


Figure 16.4 Message Output During Immediate Handshake (CY550 drives data)

After the CY550 has issued the output message, and FPL is high again, the host may continue with the immediate command handshake, and the CY550 will now accept and execute the immediate command, as shown in the second example waveform below. Here, the host does not drive the data bus for the first handshake, giving the CY550 the "0FFh" first command character prefix. This is followed by the normal first command character.

Also note that the second example waveform applies to the case where the CY550 accepts the immediate command without first generating some output data.

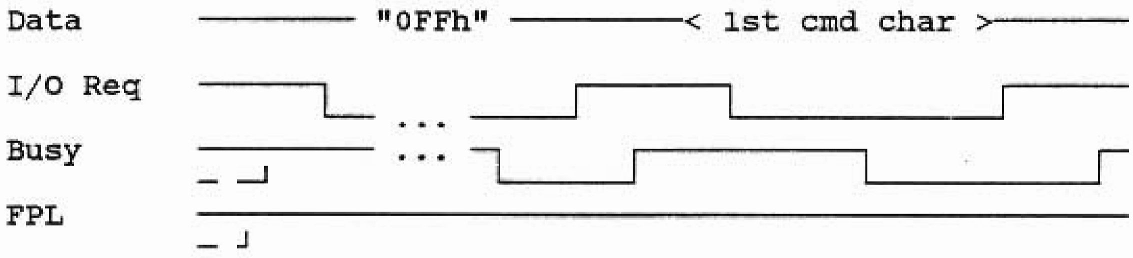


Figure 16.5 Normal Immediate Handshake or Continued Handshake after message.

In addition, the CY550 will only issue an output message if there is a query or quoted message in the memory program. If there is no such command, the CY550 will always accept the immediate command from the host, without the possibility of changing FPL to the output mode during the first handshake of the immediate command.

In this case, the host handshake is less complex, because the host is not required to test the state of FPL during the first handshake with no data. FPL should always be high. However, during the first handshake the data bus must still be left floating, with the "0FFh" command prefix character, so the CY550 can continue to read from the memory, if required to finish the currently executing command.

When using the serial command interface, it is also possible to issue immediate commands to the CY550 while it is executing a program from the local external memory. As with the immediate parallel commands, the CY550 will stop the memory execution, process the immediate command, then continue execution from the memory, unless the immediate command was the "0" command.

Since the CY550 may be in the middle of a command when the host decides to send the immediate serial command, it is best to use the Clear To Send (CTS) protocol or the XON/XOFF protocol to help control this environment. This will prevent overflow of the CY550 serial buffer, since it cannot read the new command in the middle of executing the command read from memory.

The CY550 will continue executing the command from memory until it is complete. Then the CY550 will notice that a serial command has been recieved, and suspend memory execution until that serial command is completed.

If necessary, the CY550 will drive CTS high (off) as the serial receive buffer is filling. The host must suspend sending more serial characters at this time, until CTS is turned on again.

The XON/XOFF protocol works in a similar manner, with the CY550 sending an XOFF character as the serial receive buffer is filling, and an XON character when the CY550 is ready for more characters.

After the immediate serial command has been processed, external program execution will resume, unless the immediate command was the zero command, which halts program execution.

As with the parallel command interface, the CY550 does not mix the command read from memory with characters received over the serial interface. Once the first letter of a serial command has been received, execution from the external memory is suspended until the immediate command is finished.

The Serial Interface

The CY550 serial command interface uses two primary signals, RxD and TxD. Serial data is received on RxD and is sent on TxD. Note that when the parallel interface is not used, IO_REQUEST and BUSY may be tied to select a fixed baud rate for the CY550. Otherwise, the adaptive mode will be used, or a parallel mode command can select a fixed baud rate (either from the local external memory, or from a parallel-connected command source).

In the adaptive case, the CY550 must receive two carriage return codes before receiving or sending any other serial data. It will adjust the baud to match that of the received carriage return codes.

With an 11 MHz crystal, the CY550 can adapt to standard baud rates between 300 and 19,200 baud, plus 57,600 baud. Any integral division of 57,600 baud may be used at this crystal frequency. For other crystal frequencies, the higher rates may be limited by the internal timer resolution of the CY550 circuitry.

The CY550 also implements two forms of flow control, one in hardware and one in software.

Hardware flow control consists of the CTS/ signal, and optionally the RTS/ signal, shared on USRB6. The Mode command is used to enable the RTS/ signal function. Note that when enabled, RTS/ must be driven low at the CY550 to enable it to transmit serial characters. If RTS/ is left floating or driven high, the CY550 will not transmit any serial data, and may halt all command processing, waiting for RTS/ to go low!

The CY550 drives CTS/ low when the internal serial receive buffer is filling up. If the CY550 is capable of processing the incoming commands as fast as the character baud rate, the CTS/ signal will not be turned off. This should be true for baud rates of 19,200 or less, as long as there are no commands that take a longer time to execute, such as a Delay command or a Wait for an input signal.

If the buffer fills up, due to long execution times for some commands, or very high baud rates, the CY550 will turn CTS/ off. It will then stay off until the accumulated characters have been read from the serial buffer.

Software flow control uses the XON and XOFF characters to control the flow of serial data. Since this is an "in band" flow control protocol, no additional signals are required beyond transmit and receive data.

The XON/XOFF flow control is also enabled by a Mode command, and once enabled, functions similarly to the hardware CTS/ and RTS/ flow controls. The CY550 sends XOFF at the same time CTS/ is turned off, and sends XON when CTS/ is turned on again.

Note that the flow control may also be used by the host, suspending CY550 transmissions by sending XOFF to the CY550, and resuming them by sending XON.

When a query command is received, and the CY550 mode register is set to respond out the serial port (Mode bit MB0 = 0), the query response will be sent on the TxD signal after the command is received. Note it is possible to issue the query command on the parallel interface, and have the response sent out the serial side. A typical serial query and response are shown below.

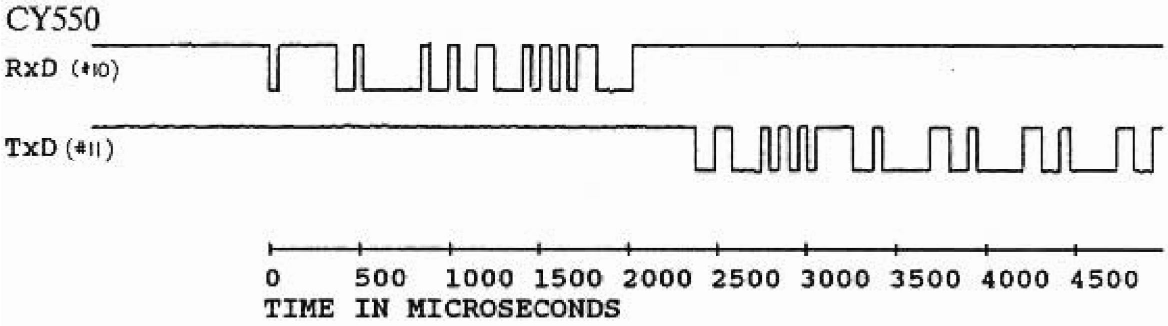


Figure 16.6 Typical Serial Interface Waveform.

External Memory Control Signals

As discussed previously, the CY550 can support up to a 64K byte external memory space, used for storing and executing various command sequences, for implementing logic execution flags, and for extended I/O functions.

The CY550 provides all address and strobe timing signals to the memory. Signals required are XMEM_SEL, ALE, WR, RD, and the data bus lines. Special use is made of the data bus, which provides all 16 bits of address, plus the actual data transfers. Two external latches are required to hold the address. The example shown uses 74LS373s for this purpose.

Additional address decoding, to select between various external memory types, and the extended I/O functions must be provided by the application design. The CY550 uses the same control signals for all external memory or extended I/O read and write operations. Separate "chip select" signals are not provided, they must be generated from external decoding of the memory mapped addresses used by the different memory or I/O circuits.

When the CY550 reads or writes the external memory, it starts by placing the most significant address byte on the data bus. It then selects the memory by driving XMEM_SEL low. This both enables the memory or decoding circuits and latches the upper address byte into the 74LS373 which holds address lines A8 to A15. The CY550 then performs a read or write operation.

The read or write is broken into two steps. First the lower address byte is generated on the data bus. This value is latched into the other 74LS373, using the ALE signal. Then the actual data transfer occurs. For a write, the CY550 generates the write data on the data bus, along with a write strobe, WR. For a read, the data bus is floated, and a read strobe is generated, using RD, so the memory can now drive the bus.

When the memory transfer is over, the XMEM_SEL signal goes high again, de-selecting the external memory space. This allows the lower address byte and RD and WR strobes to be used with other hardware as well, without interference from the memory circuits. Only the memory uses the upper address byte. A typical memory access waveform is shown below.

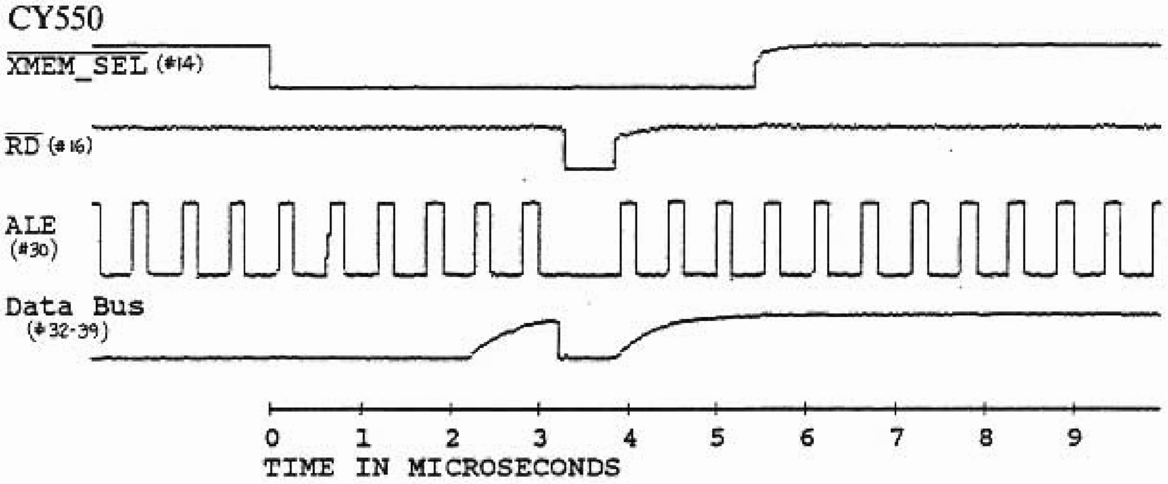


Figure 16.7 Typical External Memory Access Waveform.

As a special feature, the CY550 performs a verify read after writing commands to the program memory. The verify read is not performed for any extended I/O output writes. This function is provided in support of EEPROM memories, that require a long time to perform a write operation, and cannot perform another random write or read while the first one is in progress. Many of these memories support Data/ polling, in which at least one bit of data is inverted while the memory is busy writing the byte. When a read is performed, the data read does not match the data written until the write operation is complete. This change in data values indicates when the memory is ready for the next operation.

The CY550 performs the verify read repeatedly until the data matches the value written, or a time out occurs. For RAMs, the match will occur immediately, so the CY550 simply continues at that point. For an EEPROM, the match will occur after the EEPROM write is finished, and the CY550 waits during this time before going on to the next function. This feature is shown in the waveforms below.

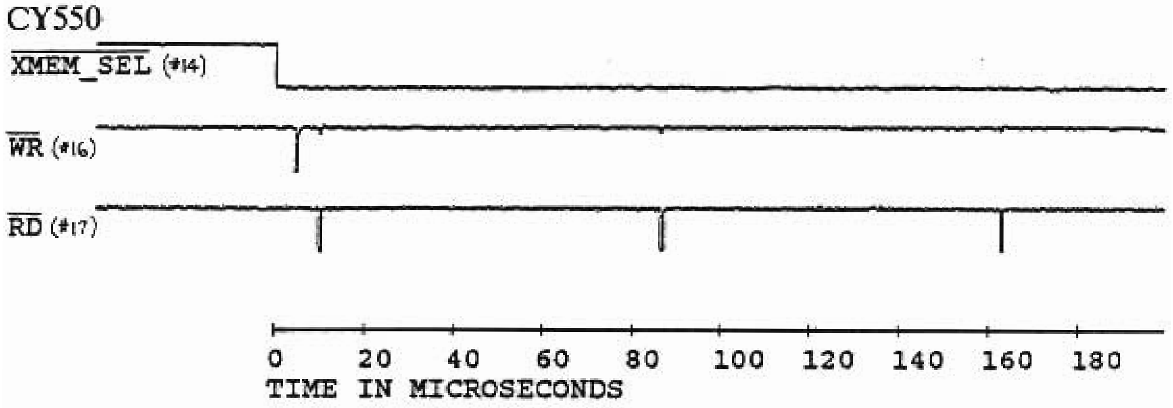


Figure 16.8 External Memory verify read after write.

You should be aware of this time period delay when using EEPROM, because it is possible to send serial characters to the CY550 faster than the EEPROM can accept them, when running the serial interface at higher baud rates. It is best to use one of the flow control protocols in this case, or at least implement a time delay between each character.

Note that the parallel interface takes care of this time delay on its own, since the CY550 will stay busy while it is writing the character to the memory.

Additional care must be taken when using the parallel command interface along with external memory. Since both functions share the CY550 data bus, the parallel command source must be able to release the data bus when the CY550 is commanded to access the external memory. This would be true while writing programs into the memory as well as during program execution.

The host system should only drive the data lines when the CY550 is not busy, and the host must stay off the bus while the CY550 is trying to write data to the memory, or is executing commands from the memory. No special signals are provided in this mode, the CY550 simply performs memory read and write operations until the command sequence causes it to stop, and process only immediate commands again.

The BUSY signal is the only indication when the CY550 might try to write to the memory, and the host must stop driving the data bus as soon as possible after the BUSY signal goes low.

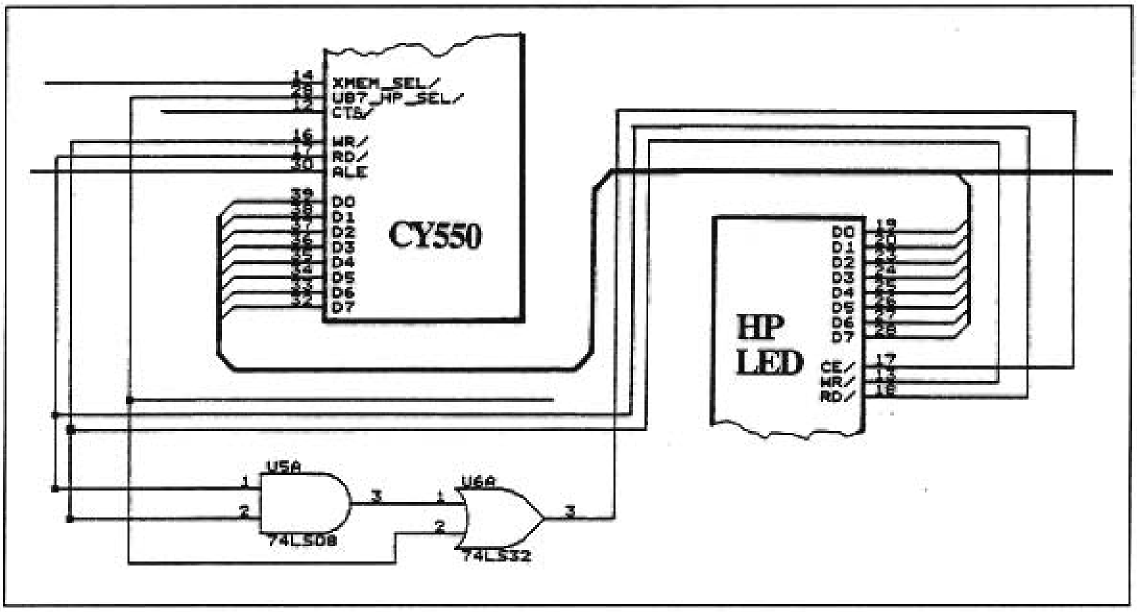
As described earlier, immediate commands may be issued by the host during program execution. In this case, a special handshake is used, in which the data bus is not driven during the first command handshake. This allows the CY550 to finish any memory access required, before handling the immediate command from the host.

Special HP LED Display Control Signals

Along with the normal serial and parallel data outputs, the CY550 also supports a special parallel LED display, using the Hewlett- Packard HDSP-211x displays. These are compact 8 character displays, with programmable intensity, custom characters, and other features. The displays are very convenient and compact, making them useful in a minimal CY550 based system.

The CY550 provides all address and strobe timing signals to the display. Signals required are HP_SEL (shared function with USRB7), ALE, WR, RD, and the data bus lines. For the HP display, only the lower 6 address signals are used, and they are generated from the data bus and ALE signals. The address signals must be demultiplexed by an external latch, such as the 74LS373. If external memory is also being used in the system, the same address latch used for the lower byte of the memory address may be connected to the display. This is shown in the example schematic.

The HP_SEL, RD, and WR signals must be combined as shown to generate the Chip Enable signal for the display. This signal has special timing requirements, and may not be simply connected to the HP_SEL signal.



When the CY550 reads or writes to the display, it starts by driving HP_SEL low. The read or write is then broken into two steps. First the lower address byte is generated on the data bus. This value is latched into the 74LS373, using the ALE signal. Then the actual data transfer occurs. For a write, the CY550 generates the write data on the data bus, along with a write strobe, WR. For a read, the data bus is floated, and a read strobe is generated, using RD, so the display can now drive the bus.

When the data transfer is over, the HP_SEL signal goes high again, disabling the RD and WR strobes for the display. A typical display access waveform follows.

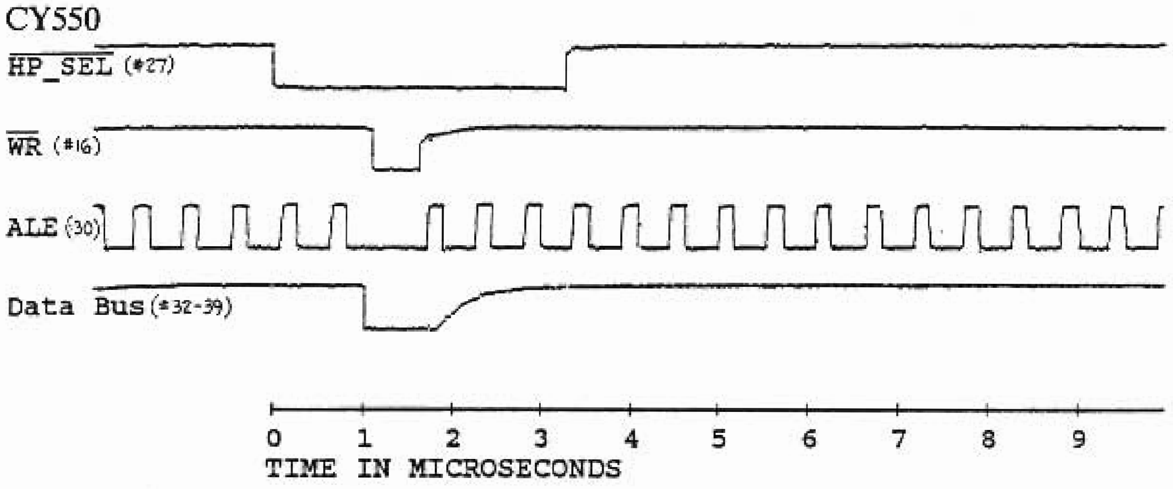


Figure 16.9 Typical HP Display Access Waveform.

Stepper Motor Interface Signals

The CY550 implements eight signals dedicated to the stepper motor, plus the user selectable function bits, which can generate output control signals for special hardware, or be tested for special conditions that guide the CY550 through a program of motions, or be used for home sensor seeking. The extended I/O functions enhance this capability even further.

PULSE and CCW

The two primary control signals for the motor are PULSE and CCW, which provide the step and direction indications to the motor. These signals are connected to the external power driver that actually runs the motor. Every time the CY550 directs the motor to take a step, a signal is generated on the PULSE line.

The power driver should advance the motor by one increment for each pulse. Note that the step increment could be a full step, half step, quad step, or micro step. This will be determined by the application and the power driver. The CY550 does not know or care what the step size is. All position information is treated in terms of the step increment, not in terms of the linear displacement or degrees of rotation of the motor.

The CCW signal is a level active signal that tells the power driver in which direction to step the motor. When high, the driver should step counter clock-wise, and when low, the driver should step clock-wise. Note that the physical rotation of the motor will depend on how the motor windings are connected to the power driver, not on the level of the CCW signal. When this signal is high, the CY550 will decrement the current position for every step taken, and when the signal is low, it will increment the current position for every step taken.

Also note that the CCW signal can be used as an input, to force stepping in the desired direction. In this case, the software direction command should be set to "-", so the CY550 tries to drive the CCW signal high. Now an external circuit can easily override the CY550, and force the CCW signal low, causing stepping in the " + " direction. If this is tried while the CY550 is driving the CCW signal low, a much stronger external circuit will be required to override the CY550, since it has much stonger sink current capability than source current capability. This could ultimately burn out the CCW signal, and should be avoided if possible.

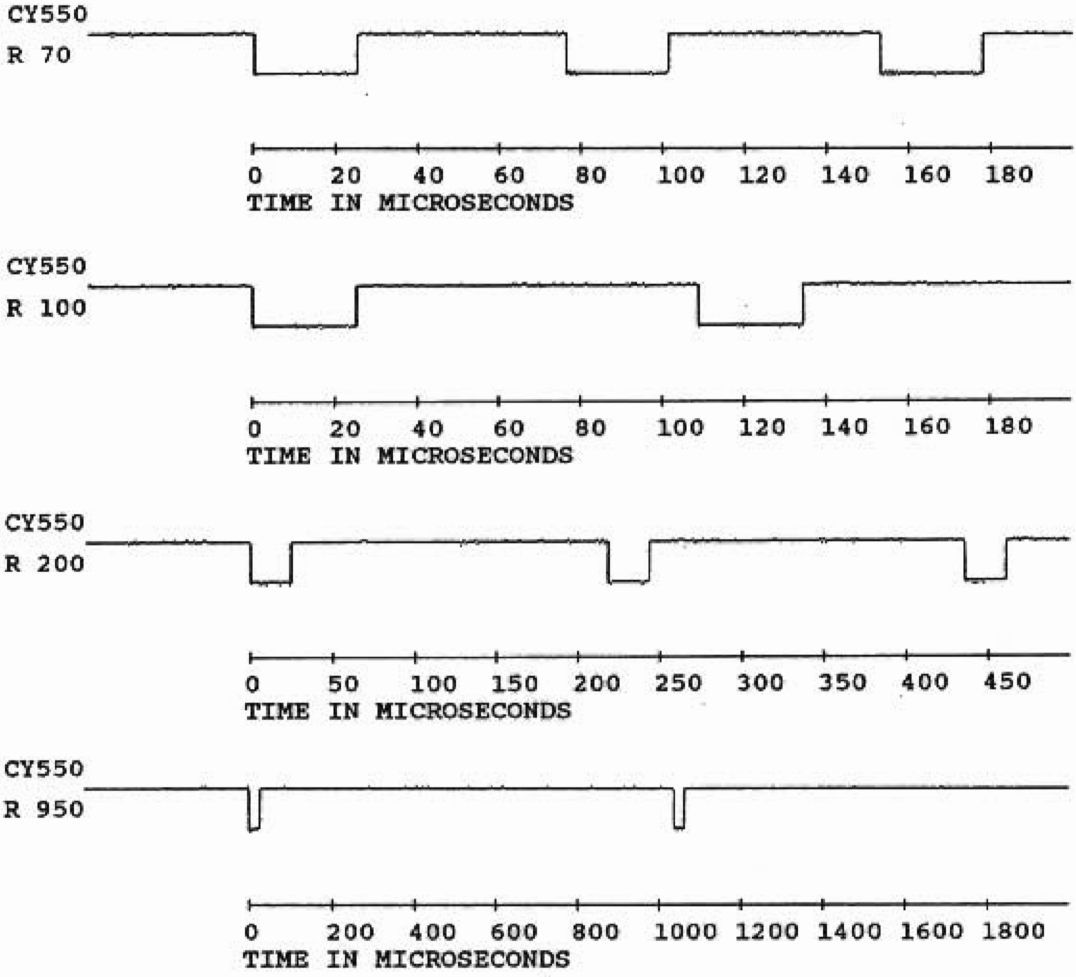


Figure 16.11 $\overline{\text{PULSE}}$ waveforms for various step rates.

Several PULSE signal waveforms are shown above, for different step rates. Notice that the PULSE signal is normally high, and goes low at the beginning of a step. It stays low for a fixed amount of time, which depends on the crystal frequency of the CY550, and then goes back high for the duration of the step time. When it is time for the next step, PULSE goes low again.

Be sure the requirements of your power driver are met by this signal. For some drivers, PULSE should normally be low, and go positive when a step is taken. If you are using such a driver, an external inverter may be required on the CY550 PULSE signal. Otherwise, the power driver may not be properly clocked, especially when stepping direction is changed.

When the step rate parameters are set to enable acceleration, the CY550 will vary the step rate from the starting rate (First Rate parameter) to the final rate (Rate parameter) for each motion it takes. This means the PULSE waveform will vary from the slower rate to the faster rate as the CY550 accelerates the motor up to the slew rate, and then will be stable for the duration of the slewing period. The CY550 will determine when to begin deceleration so that the motor will slow down again to the starting rate as the desired number of steps has expired.

STOPPED

The STOPPED signal is a status output from the CY550, indicating whether the motor is stepping or not. At the beginning of a motion, from either a Go or Position command, the STOPPED signal is brought low. It will continue to stay low for the duration of that motion. At the end of the motion, the STOPPED signal is brought high again. The low to high transition on STOPPED indicates when the motion is over, and the high level indicates that the motor is no longer moving. A typical waveform is shown below.

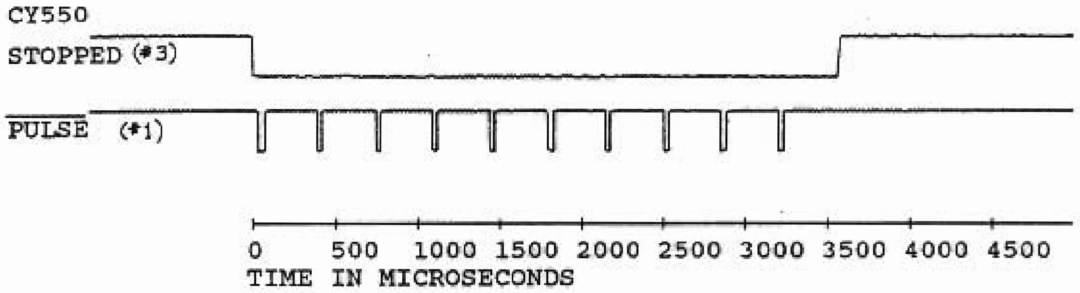


Figure 16.12 STOPPED signal waveform.

The STOPPED signal may be used as a motion complete indication to the host system. It may also control the power selection for the stepper motor power driver, switching automatically between a high power selection while stepping, and a parking power selection while stopped. This can provide maximum torque during motions and provide holding torque while the motor is not moving, in order to prevent overheating of the motor. The CY550 automatically delays after the last step, before bringing the STOPPED signal high again. The delay depends on the First Rate parameter value, and represents one step time at the selected rate. This allows the motor to settle into its final position before the high driving power is removed. Note that the Home command and Jog function will cause the Stopped signal to pulse with each step.

SLEW

Another status output from the CY550 is the SLEW signal, which indicates when the selected step rate (Rate parameter) has been achieved. This signal is normally high, and goes low while the CY550 is stepping the motor without acceleration or deceleration. A typical waveform is shown below. Notice that the time between STOPPED being low and SLEW being low represents the time to accelerate or decelerate the motor.

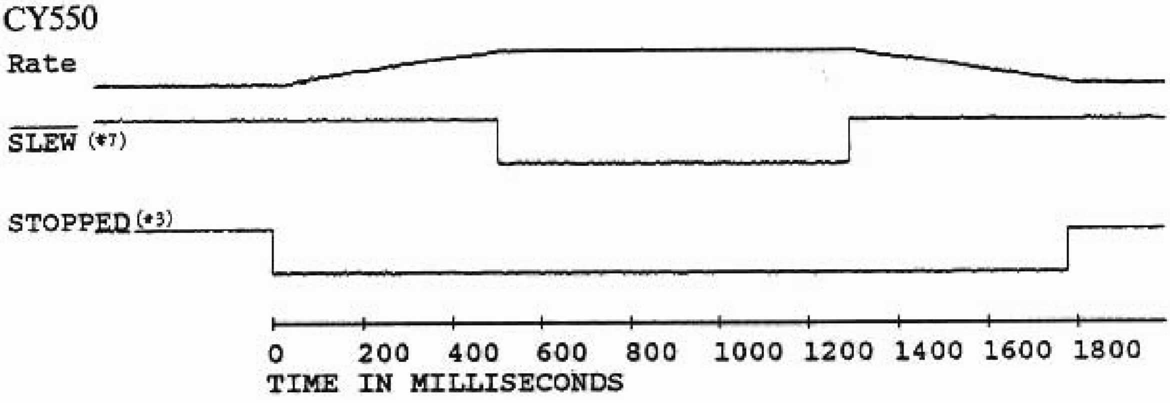


Figure 16.13 SLEW signal waveform.

The SLEW signal may also be used as an input. If this signal is driven low at the beginning of a motion, the CY550 will enter the continuous stepping mode. In this mode, it will accelerate from the starting rate to the slew rate, then run indefinitely at this rate. There are no specific number of steps to move or target position to stop at. However, the current position is still updated during the motion, and may be queried at any time. The CY550 is capable of processing most commands during a continuous move, providing on-the-fly capability for such motions. The SLEW signal is tested just before the CY550 begins the first step of a motion to determine if a continuous motion should be made.

Alternatively, the continuous mode may also be selected by the Continuous command, C. This should be followed by a Go command to actually start the motion.

Stopping Continuous Motions: A continuous motion is ended by either pulling the INHIBIT_ABORT signal low externally or by sending the Stop Motion (^) command to the CY550.

Continuous mode stepping is disabled at the end of every motion, so if a second motion is desired in continuous mode, another Continuous command must be issued, or the SLEW signal must be driven low as the motion starts.

INHIBIT_ABORT

The INHIBIT_ABORT signal is a stepping control input to the CY550, with a number of functions. In general, it is used to override the normal stepping behavior of the CY550, as selected by the CY550 commands.

If this signal is low at the beginning of a motion, started by a Position or Go command, that motion will not start until the INHIBIT_ABORT is brought high again. This allows you to synchronize the motion to an external signal, rather than a CY550 command. Multiple axes of motion could be started simultaneously by using this signal to control the start of stepping. An example waveform is shown below.

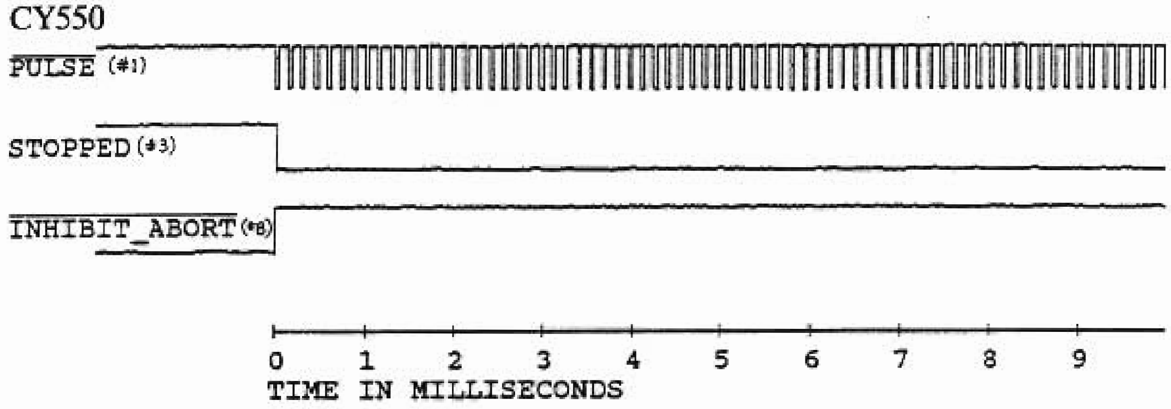
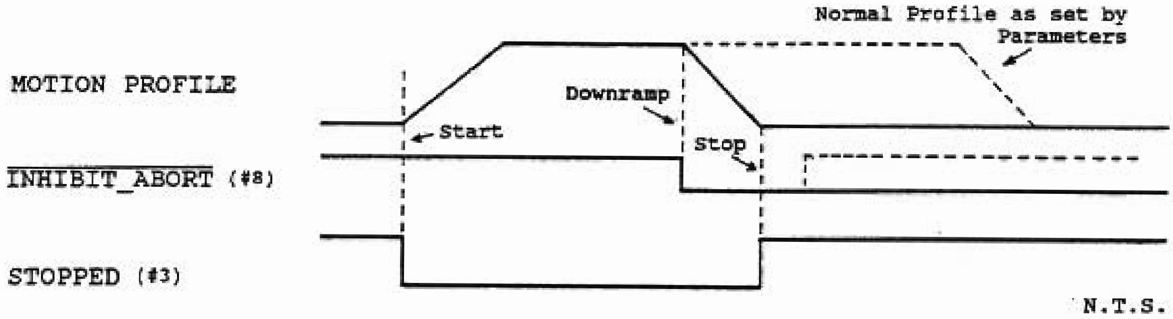


Figure 16.14 INHIBIT_ABORT at the start of motion.

Note that once a Position or Go command is executed by the CY550, it will wait indefinitely for the INHIBIT_ABORT signal to be high. There is no signal to terminate the motion until it has been allowed to start.

Once the motor is moving, the INHIBIT_ABORT signal may be used to slow down and stop a motion before the CY550 reaches the target position. Using the signal in this way provides a controlled early stop to the motion.

If INHIBIT_ABORT is driven low during a motion, the CY550 switches to a deceleration mode. It will decelerate symmetrically to the acceleration, until it achieves the selected initial stepping rate (First Rate parameter). The CY550 will then test the INHIBIT_ABORT signal once for each step, and if it is low, the motion will stop. Otherwise, the motion will continue, at the starting rate, until the target position is reached. To slow down the motor and stop when the initial step rate is reached, simply hold the INHIBIT_ABORT signal low until the CY550 brings the STOPPED signal high again. Several cases are illustrated below.



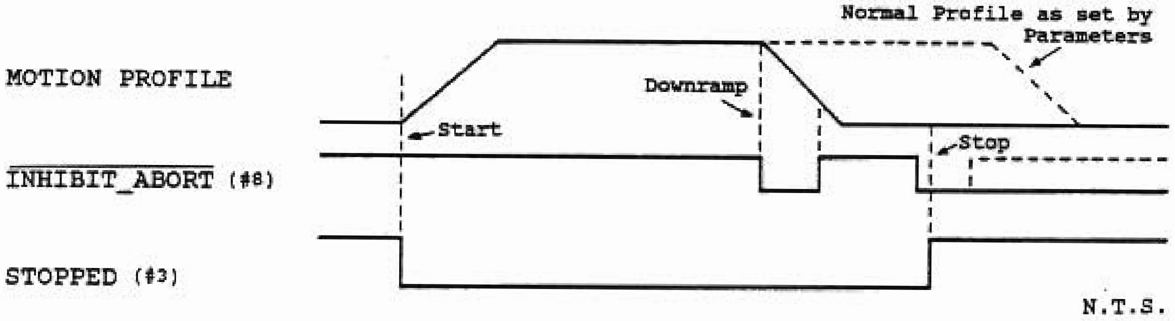


Figure 16.15 Stopping motions with INHIBIT_ABORT.

CW_LIMIT and CCW_LIMIT

The CY550 also provides two inputs for immediately stopping a motion, the CW_LIMIT and CCW_LIMIT signals. If the relevant signal goes low during a motion in that direction, the CY550 immediately stops the motion. There will be no deceleration before the stop, the CY550 simply abandons the motion.

Also, if a limit signal is low at the beginning of a motion, the motion will not be taken. The CY550 immediately aborts the motion without taking any steps. Normal motions are allowed in the opposite direction. Example waveforms are shown below.

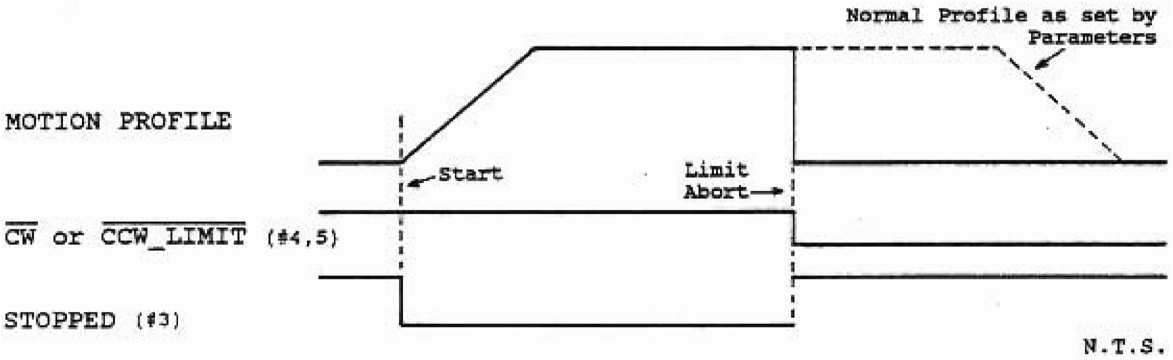


Figure 16.16 CW_LIMIT or CCW_LIMIT signals stop a motion.

*** Warning *** Warning *** Warning ***

Although the CW_LIMIT and CCW_LIMIT signals allow the CY550 to provide some protection to systems that accidentally run into limits, they should NOT be used as the only protection if reaching the limits could endanger operators or damage the equipment.

Unexpectedly hitting a limit implies some system failure, and no component, including the host computer or CY550, can be fully trusted in this situation. Automatic recovery from a limit should be used with extreme caution, since the system could perform the recovery, then run back into the limit again, if the original cause of the failure has not been fixed.

The best protection in cases where the system has failed, is to disable the power to the stepper motor drivers. This will allow the motors to stop moving the load, even if they are commanded to continue stepping. The cause of failure should then be determined and fixed before the motors are energized again.

JOG Mode Operation

The CY550 provides a manual stepper control function through the JOG signal. This signal is tested whenever the CY550 is not executing a command or performing another motion. This is a special signal, because it is checked for one of three different states, and is both an input and an output.

When the line is left floating, so the CY550 can drive it both ways, the jog function is disabled, and stepping only occurs when the CY550 is commanded to take a motion.

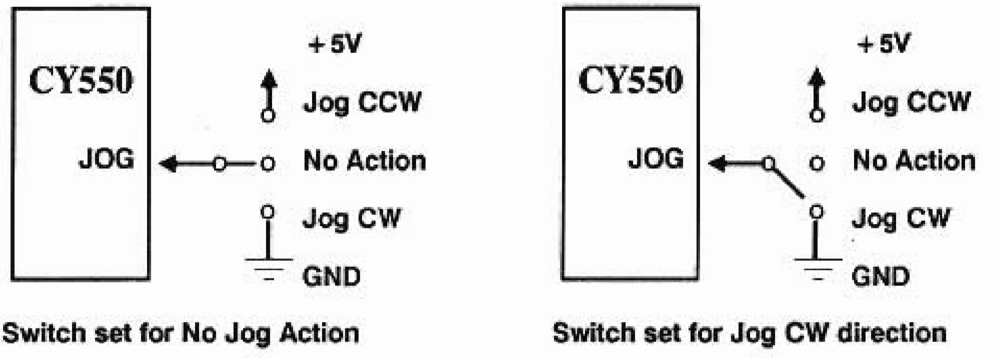
When the line is driven low, the CY550 will jog the motor in the clockwise direction. The CCW signal will change state, if required, to select this direction.

When the line is driven high, the CY550 will jog the motor in the counter clockwise direction. Again, the CCW signal will change state, if required, to select this direction. Note that the signal level on the JOG line matches the stepping direction selected by the CCW signal.

The CCW signal is restored to it's original state when each jog single-step finishes.

Normally, the JOG signal will be connected to a three position, center-off, switch. One side of the switch will be connected to ground, while the other side is connected to +5 volts. Jogging is enabled in each direction by pushing the switch from the center position to one side or the other. This is shown in the following figure:

Switch Connections



When jogging, the CY550 will take one step, then check for any new command inputs. If a new command is available, it will be executed, otherwise the JOG signal will be tested again. When the CY550 is idle, the effect of driving the JOG line is for the CY550 to take repeated single steps. Note that the checking for new commands during jogging allows you to issue CY550 commands between jog steps, providing jog motion and command execution at the same time!

While jogging, the CY550 will use a step rate derived from the parameter of the First Rate command. The initial stepping rate for incremental or continuous motions is taken from the First Rate parameter value, and this rate is divided by 20. Note that the same step rate is used by the Home command. The CY550 Step Rate Table lists the step rates for normal motions. These rates should be divided by 20 to obtain the jogging step rate. With a 12 MHz clock, jog rates from about 1 step per second to about 250 steps per second are possible.

The CY550 will not accelerate or decelerate during jogging, running only at the rate derived from the current value of the First Rate parameter. However, the interactive command execution during jogging does allow you to modify the step rate, by issuing a First Rate command with a new parameter value.

Automatic Position Display

Also, when bit 4 of the Mode Register is set, the CY550 will display the current position during all motions. This may be especially useful for the jog steps, indicating position for manual adjustment. The position is always output to the currently enabled display device, and for jogging, it will affect the step rate. For example, sending the current position to a serial device would require enough time to issue the 11 character position string at the baud rate defined for the serial interface. For normal motions, the step rates are not affected, since the position display occurs independent of the motion itself.

The JOG signal also represents a minimal mechanism for testing the CY550 and motor subsystem. Without any commands, it is possible to move the motor in either direction, verifying that the PULSE and CCW signals are correctly connected to the motor power driver, and that the motor is properly connected. This also verifies that the CY550 is functioning at least in a very basic fashion.

Home Signal

The CY550 supports an additional stepping function, but it uses the user selectable function signals to implement the process. This function is the automatic home sensor seek, and is selected by the Home command.

A single argument to the Home command specifies which user selectable bit or bits to monitor during the step to home. The parameter value is treated the same as for the Til and Wait commands, and is reproduced below:

00H to 07H	Test one of user bits 0 to 7 for 1
08H to 0FH	Test one of data bus bits 0 to 7 for 1
10H to 17H	Test one of user bits 0 to 7 for 0
18H to 1FH	Test one of data bus bits 0 to 7 for 0
80H to 0BFH	Test user bits 0 to 5 for a group match

You have the option of checking any one signal for the home sensor, or using the six signal group of user bits 0 to 5. Normally, a single home sensor would be connected to a single user bit signal. Note that the bits of the I/O byte register, parameter values 20h to 27h and 30h to 37h, should not be used by the Home command! Since these bits are internal to the CY550, they will not change value during the execution of the Home command, meaning the "home" position can never be found through the I/O byte register bits.

This command uses the same step rate as the jog function described above, a division by 20 of the step rate selected by the First Rate parameter value, so the CY550 will step the motor at about 1 to 250 steps per second while seeking the home sensor signal.

The Home command (and the Jog pin function) will cause the Stopped signal to pulse with each step.

The Home command starts by testing the specified bit pattern for a match with the external signals. If there is a match, the CY550 will begin by stepping in the CCW direction, and will continue in this direction until there is no longer a match. The CCW stepping phase is skipped if the initial test indicates no match between the argument bit pattern and the external signals.

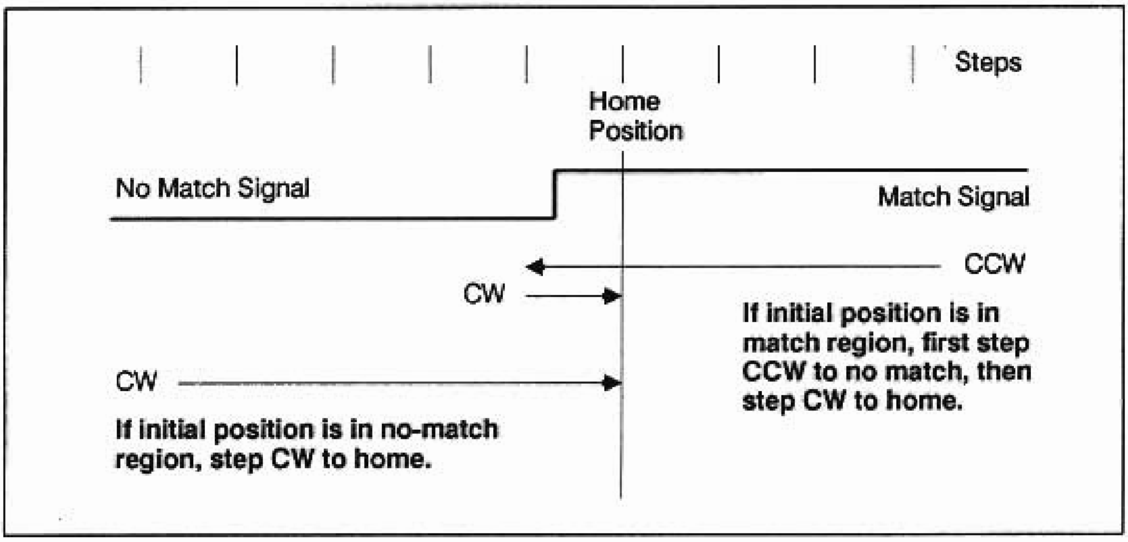
When there is no match between the bit parameter and the external signals, the CY550 steps in the CW direction. It will continue to step in this direction until the signals match the parameter value. At this point the CY550 stops, and sets the current position to zero.

Thus, the Home command steps the motor to seek the edge between a match and no match condition on the home signal. The CY550 then stops at the first step where a match occurs, and is always stepping in the CW direction looking for this match. This mechanism should always seek the same mechanical position, since any directional backlash is compensated by using the same final step direction in seeking home.

If the CY550 runs into one of the limits while looking for the home signal, it will abort the home command and stop stepping. The current position is not changed to zero in this case. As with the jog function, the current position will be displayed after every step, if bit 4 of the Mode Register is set.

Some care must be used in designing the home sensor signal for the CY550, to insure that it initially steps in the correct direction to seek the home signal. If the application involves a circular positioning function, in which the mechanism rotates a wheel or turn-table, there is no special consideration, since the mechanical positions will always repeat after some number of steps. The CY550 can find the home sensor signal no matter what direction it starts to step.

However, if the application involves a linear displacement, the mechanical home position can only be found by stepping in one direction, depending on the relative position between the mechanical home and the current position. In this case, the sensor signal should be designed to indicate the direction to home, as well as the actual home position, found by the change in signal from no match to match conditions. This is illustrated as follows:


Steps
Home Position
No Match Signal
Match Signal
CCW
CW
If initial position is in match region, first step CCW to no match, then step CW to home.
CW
If initial position is in no-match region, step CW to home.

If the home sensor signal cannot be designed to indicate the initial direction to home, the motor must be positioned so that it is on the "correct" side of home before the Home command is used. This will insure that the CY550 always steps towards the sensor in seeking the home position. The cw and ccw limits might be useful in this situation, since you could move to a known limit, then use the home command to step in the opposite direction until the home position is found.

CY550 Step Rate Information

Step rates for the CY550 are specified by two commands. Selections for the starting rate are made by the value of the First Rate (F) parameter, and selections for the slewing rate are made by the value of the Rate (R) parameter.

The First Rate (F) parameter value acts as an index into the CY550 Step Rate Table, which has 120 unique entries, ranging from a parameter value of zero to 119. Thus, for any crystal clock frequency, the CY550 can generate 120 different initial step rates. The CY550 Step Rate Table is shown for a clock frequency of 12 MHz, so the specified rates must be linearly scaled by the actual clock frequency when a different clock is used. The scaling factor is Fcy/12, where Fcy is the operating frequency, in megahertz.

The slewing Rate (R) value is specified directly by the parameter value of the Rate command. It is a 16 bit quantity, and specifies the number of cycles per step at the slew rate. Since the value specifies a cycle count, smaller values of Rate specify faster stepping rates. The Rate parameter may vary between 65535 and 67, for rates of about 15 steps per second to 14,925 steps per second at 12 MHz, or about 19,900 steps per second at 16 MHz.

The Rate parameter is specified in CY550 cycles

Since the Rate parameter is specified in CY550 cycles, you have single cycle resolution over the slewing rate, and for slower rates, can specify exactly the desired rate. One CY550 cycle is about 1 usec, depending on the crystal frequency being used, so this represents the minimum time increment from one rate to the next.

You should not specify Rates faster than 67 cycles. The CY550 ignores any rate values below 67, and sets an error status bit indicating an out of range rate value if smaller values are tried.

Recall that one CY550 cycle time is defined as twelve divided by the oscillator frequency, $12/F_{cy}$. For example, at 12 MHz, one CY550 cycle time is 12/12 MHz, or 1 usec. At 11 MHz, one cycle is 12/11000000, or about 1.091 usec.

A Rate parameter of 100 would specify a slew rate of 100 cycles per step. At 12 MHz, this would be 100 usec per step, for a step rate of 10,000 steps per second. Similarly, a Rate parameter value of 5000 would specify 5000 usec per step, or 200 steps per second.

A general formula for deriving the slewing step rate in steps/second, from the Rate parameter and oscillator frequency is:

F_{cy} **(Hz) / (12 * R) = steps/second**

For example, with a Rate parameter of 100 and an oscillator frequency of 11 MHz, the step rate value would be:

11000000 / (12 * 100) = 9167 steps/second

Similarly, if you know the desired step rate and oscillator frequency, you can find the required Rate parameter value as follows:

R = F_{cy} (Hz) / (12 * steps/sec)

Note that the actual value of R to use would be the closest integer to the above calculation, since the Rate parameter must be an integer value.

As an example, the following commands would select a starting rate of 200 steps per second, and a final rate of 10000 steps per second, if the CY550 is operated at 12 MHz.

F 2	**200 steps per second initial rate**
R 100	**10000 steps per second final rate**

If a maximum stepping rate defined by the Rate parameter is less than the stepping rate defined by the FirstRate parameter, the CY550 will not accelerate. It will run at the FirstRate.

The Slope Parameter

When the CY550 is commanded to step through a motion, by either the Go or Position commands, stepping will start at the rate selected by the F parameter. The rate will then increase regularly through the acceleration algorithm, until the final rate, selected by the R parameter is achieved. The CY550 will then slew at this rate until it is time to decelerate back to the selected starting rate as the specified number of steps is taken.

If the travel distance is too short to achieve the final rate, the CY550 will perform a partial acceleration, going to the highest rate possible before decelerating back to the starting rate.

The speed at which the CY550 changes from one rate to the next is selected by the Slope (S) parameter. The value of S can range from 1 to 255, with 1 representing the slowest acceleration, and 255 representing the fastest acceleration.

Also, the CY550 performs an automatic "smoothing" of the acceleration and deceleration functions, based on the values of the rate and slope parameters. This provides the finest acceleration increment possible for the specified parameters, and aids in reducing mid-frequency resonance and stalling effects.

For any particular application, the stepping parameter values, including the slope, are best determined experimentally when you are looking for the operating limits of the system. Each motor size, driver type, and load combination will have different performance limits. In general, the required range of step rates is known for a particular application, and will determine the values for F and R, but the acceleration value is not so easily specified.

Optimal Acceleration Curves

The stepping algorithm used in the CY550 is proprietary, and provides an optimal shape, with constant acceleration at rates below about 3000 steps per second (at 12 MHz.). This easily brings the motor through the critical mid-frequency resonance band. For higher rates, the acceleration becomes near linear, until the cycle resolution of the CY550 dominates the function. The combination of the acceleration curve shape and special "smoothing" function will provide maximum performance for any motor/driver/load combination. An example waveform is shown below:

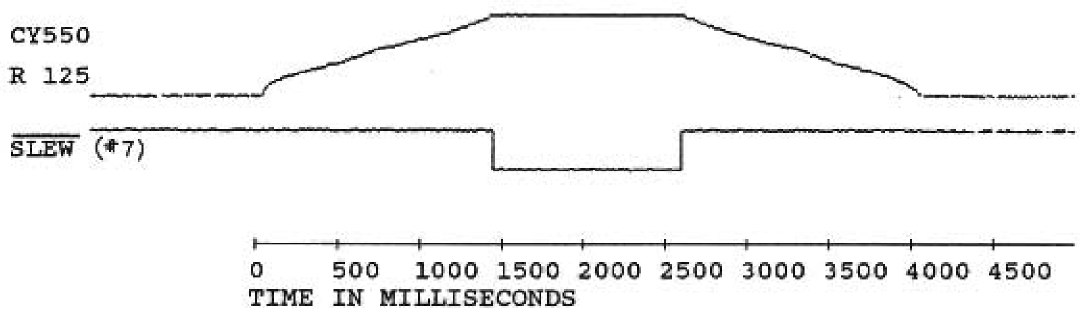


Figure 17.1 Optimal acceleration curve

Acceleration Curve as a Function of R

Another feature of the CY550 acceleration algorithm is that the shape is fixed by the value of the Slope parameter (S). The acceleration curve is always the same for a given value of S. The step rate parameters merely determine where the CY550 starts stepping, and where it stops accelerating and starts slewing in the motion profile. The three parameters F, R, and S, are not coupled by the stepping algorithm, so each may be specified independently. These features are illustrated in the figure below.

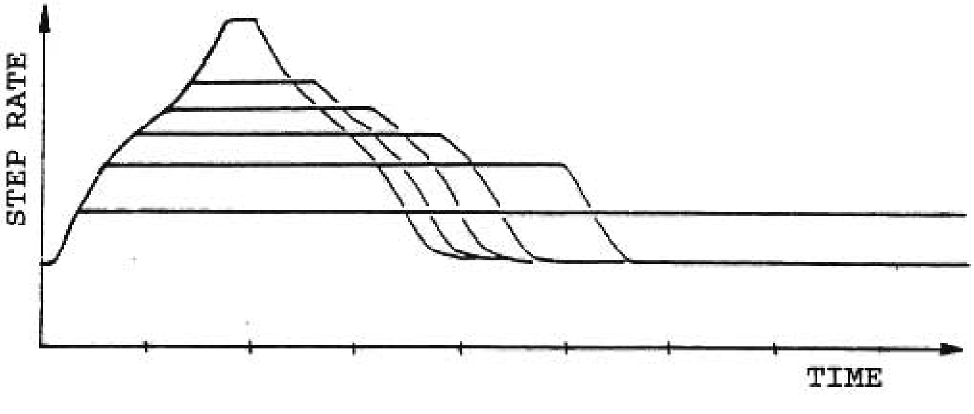


Figure 17.2 A family of curves for various rates with fixed slope parameter and travel distance.

Slope and Elapsed Time

While it is difficult to provide an exact formula for the CY550 acceleration behavior, due to the complex nature of the stepping algorithm, we can provide an approximate formula. This should be used as a general guide only, with actual timing and acceleration values determined experimentally. Do not rely on the results of this formula in the design of your system. The formula is given by:

$$\mathbf{accel} \quad = \quad \mathbf{a(s)} \quad = \quad \frac{\mathbf{dV(s)}}{\mathbf{dT(s)}} \quad = \quad \frac{\mathbf{135{,}000}}{\mathbf{(256 - s)}} \qquad \mathbf{@\ 12\ MHz}$$

where

a(s) is the acceleration in steps/second^2

s is the slope parameter value

also, the ramp time is given by the acceleration and the dynamic range of step rates taken as:

$$\mathbf{t} \quad = \quad \frac{\mathbf{|Rmax - R0|}}{\mathbf{a(s)}} \quad = \quad \frac{\mathbf{|Rmax - R0|}}{\dfrac{\mathbf{135{,}000}}{\mathbf{(256 - s)}}}$$

where

RMax is the maximum step rate in steps/second

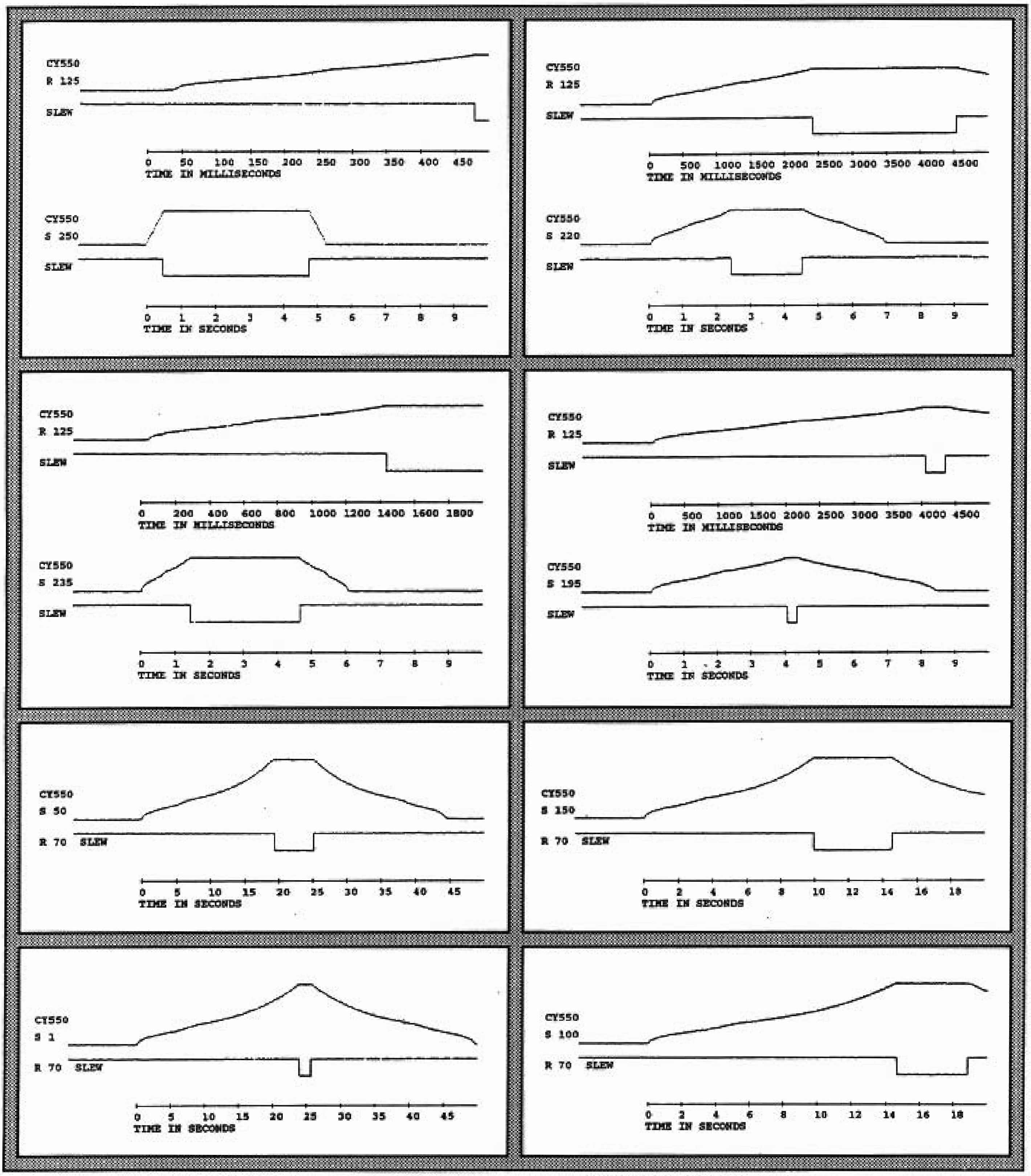
R0 is the initial step rate in steps/second

Since RMax and R0 are given in steps/second, which change with crystal frequency for the same values of the First Rate and Rate parameters, the acceleration must also be compensated by changes in crystal frequency. To scale the above equations for different crystal clocks, use:

$$\mathbf{accel} \quad = \quad \mathbf{a(s)}\ F_{cy} \quad = \quad \mathbf{a(s)}\ F_{12} * \frac{F_{cy2}}{12^2} \qquad \mathbf{@}\ F_{cy}\ \mathbf{MHz}$$

The following figures show typical performance under various values of slope, and may be used as a general guide in selecting initial operating parameters. The figures are followed by the CY550 Step Rate Table, which lists all 120 possible initial step rates when operating the CY550 with a 12 MHz clock.

CY550 Slope Curves



CY550 Step Rate Table @ 12 MHz

Param	Rate	Cycles	Param	Rate	Cycles	Param	Rate	Cycles
F	S/S	/Sec	F	S/S	/Sec	F	S/S	/Sec
F 0	15	R 65576	F 40	2294	R 436	F 80	3413	R 293
F 1	100	R 10000	F 41	2326	R 430	F 81	3448	R 290
F 2	200	R 5000	F 42	2353	R 425	F 82	3484	R 287
F 3	300	R 3333	F 43	2381	R 420	F 83	3521	R 284
F 4	400	R 2500	F 44	2410	R 415	F 84	3559	R 281
F 5	500	R 2000	F 45	2439	R 410	F 85	3597	R 278
F 6	640	R 1562	F 46	2469	R 405	F 86	3636	R 275
F 7	750	R 1334	F 47	2500	R 400	F 87	3676	R 272
F 8	843	R 1186	F 48	2525	R 396	F 88	3717	R 269
F 9	926	R 1080	F 49	2551	R 392	F 89	3759	R 266
F 10	1002	R 998	F 50	2577	R 388	F 90	3802	R 263
F 11	1072	R 933	F 51	2604	R 384	F 91	3846	R 260
F 12	1138	R 879	F 52	2632	R 380	F 92	3891	R 257
F 13	1199	R 834	F 53	2660	R 376	F 93	3937	R 254
F 14	1258	R 795	F 54	2688	R 372	F 94	3984	R 251
F 15	1314	R 761	F 55	2717	R 368	F 95	4032	R 248
F 16	1368	R 731	F 56	2740	R 365	F 96	4082	R 245
F 17	1418	R 705	F 57	2762	R 362	F 97	4132	R 242
F 18	1468	R 681	F 58	2786	R 359	F 98	4184	R 239
F 19	1515	R 660	F 59	2809	R 356	F 99	4237	R 236
F 20	1562	R 640	F 60	2833	R 353	F 100	4274	R 234
F 21	1608	R 622	F 61	2857	R 350	F 101	4310	R 232
F 22	1650	R 606	F 62	2882	R 347	F 102	4348	R 230
F 23	1692	R 591	F 63	2907	R 344	F 103	4386	R 228
F 24	1733	R 577	F 64	2933	R 341	F 104	4425	R 226
F 25	1773	R 564	F 65	2959	R 338	F 105	4464	R 224
F 26	1812	R 552	F 66	2985	R 335	F 106	4505	R 222
F 27	1852	R 540	F 67	3012	R 332	F 107	4545	R 220
F 28	1890	R 529	F 68	3040	R 329	F 108	4587	R 218
F 29	1927	R 519	F 69	3067	R 326	F 109	4630	R 216
F 30	1965	R 509	F 70	3096	R 323	F 110	4673	R 214
F 31	2000	R 500	F 71	3125	R 320	F 111	4717	R 212
F 32	2037	R 491	F 72	3155	R 317	F 112	4762	R 210
F 33	2070	R 483	F 73	3185	R 314	F 113	4808	R 208
F 34	2105	R 475	F 74	3215	R 311	F 114	4831	R 207
F 35	2137	R 468	F 75	3247	R 308	F 115	4854	R 206
F 36	2169	R 461	F 76	3279	R 305	F 116	4878	R 205
F 37	2203	R 454	F 77	3311	R 302	F 117	4902	R 204
F 38	2232	R 448	F 78	3344	R 299	F 118	4926	R 203
F 39	2262	R 442	F 79	3378	R 296	F 119	4950	R 202

18 Electrical Specifications 18

Electrical Specifications

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 V_{cc}+.5V
Power Dissipation...	1.0 watts

DC & Operating Characteristics
(T_A = 0°C to +70°C, V_{CC} = +5V +/-10%)

SYM	PARAMETER	MIN	MAX	UNIT	REMARKS
I_{CC}	pwr supply current		26	mA	
V_{IH}	input high level	1.9	V_{cc}	V	(3.5V for XTAL, RESTART)
V_{IL}	input low level	–.5	0.9	V	
I_{LO}	data bus leakage		10	µA	high impedance state
V_{OH}	output high level	2.4		V	I_{OH} =–60 µA
V_{OL}	output low level		.45	V	I_{OL} = 1.6 mA
F_{CY}	crystal frequency	3.5	16	MHz	see clock circuits

Electrical Conventions

All CY550 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 slash after the pin name, i.e., BUSY/.

All input lines except the data bus include weak pull-up resistors. If the pins are left open, the input signals will be high. The data bus pins must have external pull-up resistors to output a high value. Where appropriate, an input line will be considered in the floating state if the CY550 can drive it both high and low.

The data bus is bidirectional, and is tri-state during nonactive modes. Note that data bus signals are positive logic.

Reset Circuitry

The Restart line (pin#9) must be held high upon power-up to properly initialize the CY550. This may be accomplished by using a 4.7 µFd capacitor, as shown in Figure 18.1. Restart must be high for 10 milliseconds after power stabilizes on power-up. Once the CY550 is running, Restart need only be high for about 10 microseconds (11 MHz crystal).

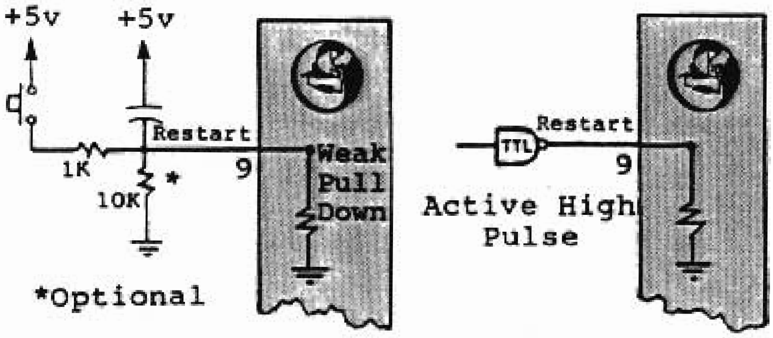
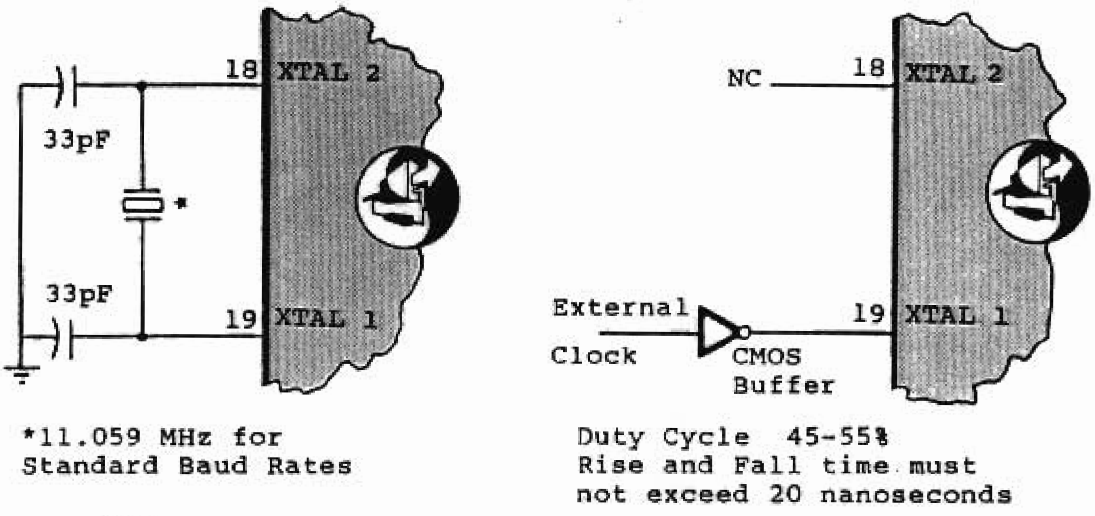


Figure 18.1 a) Restart Circuitry. b) External Restart.

Clock Circuits

The CY550 may be operated with crystal or external clock circuits. These two options are shown in Figure 18.2. All timing discussed in this manual assumes an 11.059 MHz series resonant crystal. Note that 11.0 MHz, such as a CTS Knights MP110 or equivalent will work fine. The CY550 requires an 11 MHz clock in order to generate the standard baud rates, although any crystal in the allowable range can be used with the adaptive mode, within the timing resolution limits of the CY550. Use 7.3728 MHz to make the fastest possible rate a standard 38400 baud, or 14.7456 MHz for 76,800 baud.



CRYSTAL

EXTERNAL

Figure 18.2 Clock Circuits for CY550.

19 Circuits and Examples 19

On-the-fly Command Example

The interactive nature of the CY550 on-the-fly capabilities makes it possible to perform very sophisticated operations. The following example gives a feel for some of the things that are possible. The entire example is based on immediate command mode execution, where the host system issues commands to the CY550, which are executed directly. Similar functions are possible in local external memory based programs.

The host begins by setting the basic CY550 stepping parameter values. The Rate and Slope values must always be defined after any hardware or software reset, since the CY550 resets them to undefined values. The current position is also cleared to zero, and a User Bit is initialized to one.

R 175<cr>	Specify initial slew rate
S 235<cr>	Set slope parameter
F 3<cr>	Set first rate parameter
A 0<cr>	Declare current position as 0
B 2<cr>	Preset User Bit 2 to one

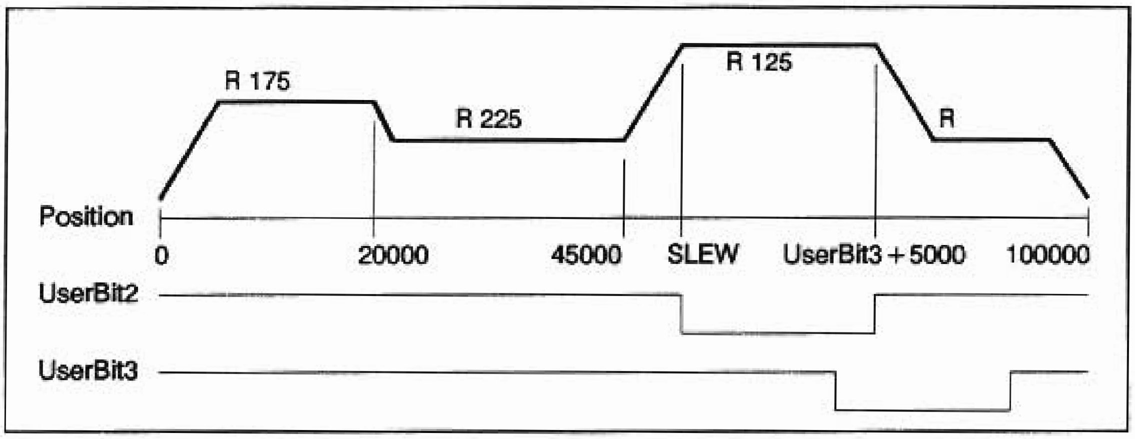
Once the stepping parameters have been defined, the host commands the CY550 to begin a motion. The CY550 begins stepping when the Position command is received, and will step to target position 100000 from the current position, which happens to be zero. This means the number of steps to take equals the target position in this case. This may not be true in general. The CY550 may start from any position, and computes the proper number of steps to take to reach the desired target.

P 100000<cr>	Begin a move

Once the motion has started, the CY550 is ready for more commands! In this example, we wait (hold off further command execution) until position 20000 is reached, then change the Rate parameter value from 175 to 225. Recall that the Rate parameter specifies the number of cycles per step, so the new rate value is slower than the current rate. The CY550 will decelerate to the new rate, starting the deceleration after position 20000.

] 20000<cr>	Wait for position 20000
R 225<cr>	Change slew rate to a slower value

While the CY550 is waiting for position 20000, additional command processing is suspended by the]-command. If the commands are being issued through the serial interface, the CY550 will not read additional command characters from the serial buffer. If commands are being issued through the parallel interface, with IO_REQUEST and BUSY handshaking, the CY550 will stay "busy" until it reaches position 20000. It will then indicate "ready" by driving the BUSY signal high, allowing the next command handshake to occur.



We now wait for position 45000, then again redefine the Rate parameter value, this time to a faster value of 125. This causes the CY550 to accelerate after position 45000.

```
] 45000<cr>        Wait for position 45000
R 125<cr>          Change slew rate to a faster value
```

Next, the host queries the CY550 for its stepper status value. This is used by the host to determine if the CY550 has reached the new slew rate yet. The host may need to repeat the query command until the desired status is read.

```
? G<cr>            Query stepper status lines
```

Once the CY550 is slewing at the new rate, the host commands that User Bit 2 be cleared to zero, possibly signaling some other action at this point in the motion. The CY550 then waits for a reply signal on User Bit 3, and continues for 5000 steps beyond seeing the reply signal. User Bit 2 is then set to one, indicating the end of this action.

```
/B 2<cr>           Clear User Bit 2 to zero
/W 3<cr>           Wait for User Bit 3 to be zero
\ 5000<cr>         Continue 5000 more steps
B 2<cr>            Set User Bit 2 to one
```

The CY550 is then commanded to slow down again and suspend further command processing until the current motion is over.

```
R 225<cr>          Change slew rate to a slower value
V<cr>              Wait for end of motion
```

Notice that all commands after the start of motion are executed while the CY550 is moving the motor.

Variable Stock Mark and Cut

Suppose we want a row of equally spaced marks painted onto some raw stock, which is fed into our machine in continuous strips. The spacing of the marks and the number of marks should be variable for any run of pieces. The stock should be cut into strips with a length that depends on the number of marks and spacing, and the operation should be repeated until the CY550 is signaled to stop.

The capabilities of the CY550 make this problem fairly easy to solve. In particular, this problem is well suited for execution as a program from a local external memory. The command sequence shown below can be used to implement all the functions required. The variable information, for the number of marks and spacing between them, can be specified each time the program is downloaded, or the program can be edited to only change those parameters. The user selectable function bits act as control and test signals to guide the program. When the operation is complete, the CY550 will stop the program and wait for more commands.

In the following command sequence, memory addresses have been shown for your convenience. They are not sent to the CY550 or recorded in the memory. Counting the bytes as the commands are written makes it easier to determine the addresses for program jumps.

The example program consists of several sections. First, some stepping performance parameters are set, and the current position is cleared to zero. This position could also be found by a Home command, if the application requires a move to a known reference position.

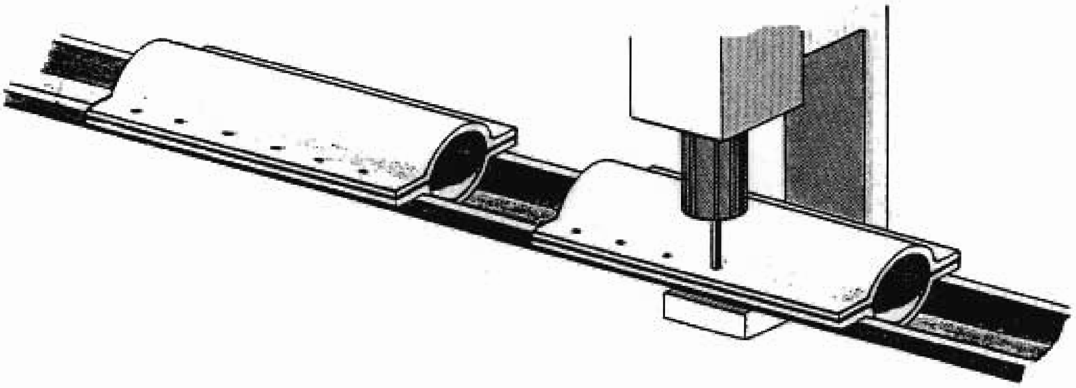
Next, the CY550 waits for a signal that indicates stock is available. This is tested on User Bit 3. When stock is available, the CY550 begins a continuous mode motion. All marks will be painted on-the-fly in one motion, with other commands read and executed from memory as the CY550 is moving the stock! The slewing step rate is specified just before the motion begins, because it will be changed near the end of the program, and must be reset to the indicated value for repeated program operations.

The CY550 advances the stock by 350 steps, which is fixed by the command sequence. Again, by changing the program or editing the command, any other step count could be used. The relative wait command is used to count the number of steps gone by.

Now the CY550 comes to a sequence of commands that form a loop, and paint the marks on the stock.

The loop begins by painting a mark, signaled by User Bit 1. When the bit goes low, the paint sprayer is activated, and when it goes high, the paint sprayer stops. A delay is used between the activation and deactivation to allow time to perform the painting operation. Note that since the delay is performed while the CY550 is stepping, it will be longer than the time specified by the parameter value, due to the overhead of stepping during the delay. The parameter should be adjusted to achieve the desired real delay time, and this value must be determined experimentally.

```
Y  100<cr>                      Select external memory location
E<cr>                           Write the following commands to memory

100          F  3<cr>               Commands will start at address 100
104          S  200<cr>             First set up stepping parameters
110          A  0<cr>               Declare current position as zero
114          W  3<cr>               Wait for Bit 3 high, stock is available
118          R  500<cr>             Set slew rate value
124          C<cr>                  Set continuous move
126          +<cr>
128          G<cr>                  Begin the motion
130          \  350<cr>             Wait for 350 steps
136          /B  1<cr>              Activate paint sprayer with Bit 1 low
141          D  100<cr>             Delay 0.1 sec to paint mark
147          B  1<cr>               Turn off paint sprayer
151          \  520<cr>             Move to next mark position
157          L  15,136<cr>          Repeat marks for specified count
166          /B  1<cr>              Activate paint sprayer for last mark on piece
171          D  100<cr>             Delay 0.1 sec to paint mark
177          B  1<cr>               Turn off paint sprayer
181          R  3350<cr>            Ramp down to rate just above first rate
188          \  350<cr>             Move 350 steps after last mark
194          ^<cr>                  Stop motion now
196          D  250<cr>             Delay 0.25 sec for stock to settle
202          /B  2<cr>              Cut stock off with Bit 2 low
207          D  1500<cr>            Delay 1.5 sec to cut stock
214          B  2<cr>               Turn off cutter
218          D  150<cr>             Wait for cutter to clear work stock
224          R  200<cr>             Set slew rate value
230          P  0<cr>               Return to starting location at high speed
234          V<cr>                  Wait until motion is over
236          T  4,114<cr>           Repeat operation until Bit 4 is high
244          0<cr>                  Stop running the program now

Q<cr>                           Quit saving commands to memory

Y  100<cr>                      Reset memory address to start of program
X<cr>                           Execute the above program
```



The CY550 continues after painting the mark, by advancing the stock to the next mark location. In the program, this is 520 steps from the end of the previous mark. Changing this number will place the marks at different spacing.

Next, the Loop command is executed, which specifies the repeat count, set as one less than the total number of marks desired. This is because the loop first paints a mark, then advances to the next mark location, so the last mark is not painted within the loop. The Loop command jumps back to the instruction that activates the paint sprayer, at the beginning of the loop. This sequence is repeated until the count is exhausted, and the specified number of marks have been painted.

Notice that every character of a command takes one memory location, including the carriage return, and all characters must be counted to compute the target address for the Loop command.

The program then paints the last mark, by repeating the User Bit 1 commands that are used in the loop.

At this point, the CY550 ramps the motor down to a rate near the initial rate of motion. This is done so the stop motion command will stop quickly, with only a step or two taken before stopping. Otherwise, the stop motion would perform a complete down ramp before coming to a stop, and thus step farther than we would like for this example.

During the deceleration, the program also waits for 350 steps to go by, to space the last mark at about the same distance as the first mark. Notice that the 350 step wait occurs during the deceleration, and includes deceleration steps. Specifying the new rate changes the Rate parameter, and starts a deceleration, but the relative wait command is then executed before the deceleration is complete!

Once the 350 steps have gone by, the motion is ended. The CY550 performs a delay to let the stock settle, then activates a cutter with User Bit 2, which cuts the stock that has been marked.

Now the CY550 returns the motor to the starting position, but uses a faster rate, since no other operations are performed during this move. The wait for end of motion command suspends program execution until the starting position is reached. The CY550 then tests User Bit 4, and if the signal is low, the program repeats from the Wait instruction. If the signal is high, the program stops with the Zero command, and the CY550 returns to the command mode.

Driver Circuit Considerations

The CY550 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 CY550. This circuit will take the logical signals generated by the CY550 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 CY550 PULSE and CCW signals. This allows the CY550 to control three and five phase motors as well as the standard four phase designs.

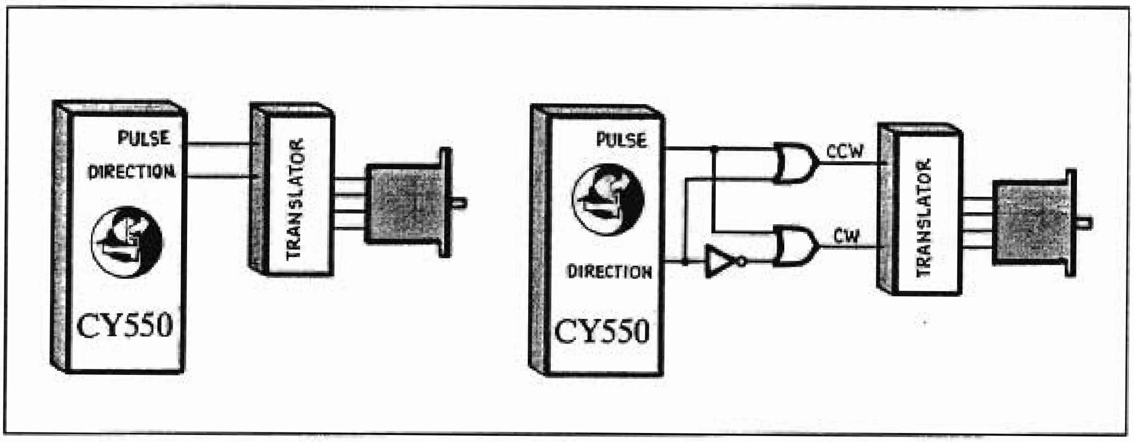


Figure 19.3 CY550 to Translator Driver connections.

If the user opts for his own driver design, the PULSE and CCW lines may be used to drive a counter circuit, which counts up or down once for each pulse, based on the level of the CCW signal. The counter output then drives the address lines of a memory device, such as a PROM, EPROM, or EEPROM, with data outputs that generate the desired motor phase patterns as the counter address steps through the PROM locations. This design is shown here.

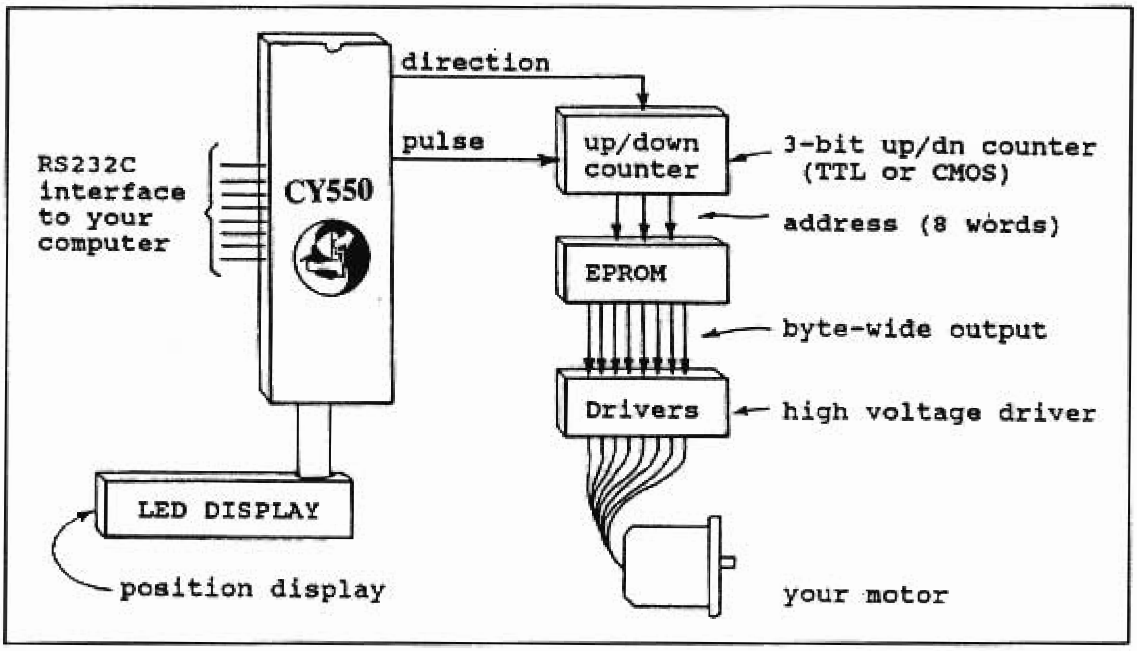


Figure 19.4 Motor phase generation circuit.

The data outputs from the PROM are then connected to the power driver circuit itself, generating the proper phase patterns for the motor. Note that such a design not only handles motors of various phases, but can also implement stepping schemes such as half-step or quad-step, as well as the standard full-step pattern.

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 data PROM.

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.

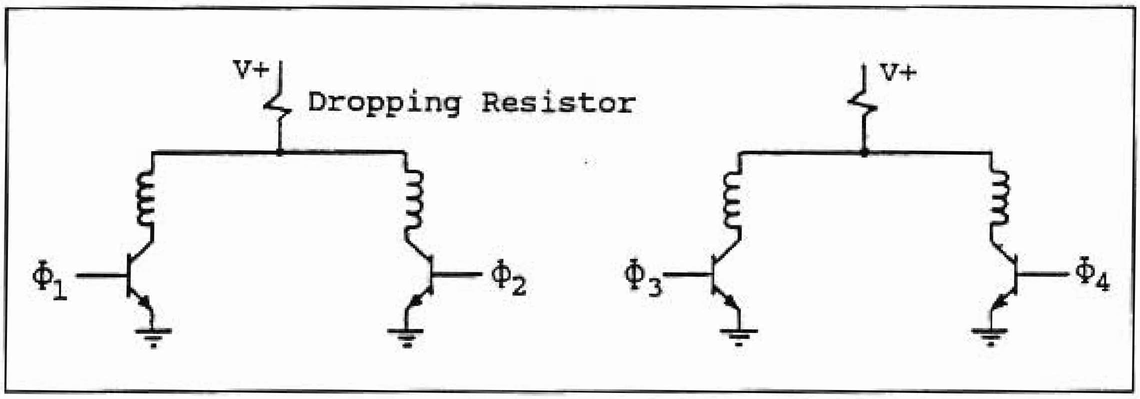


Figure 19.5 Unipolar driving circuits.

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. There is also some gain in torque, since the entire winding is always driven, unlike the unipolar design, in which only half of a winding is used at a time.

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.

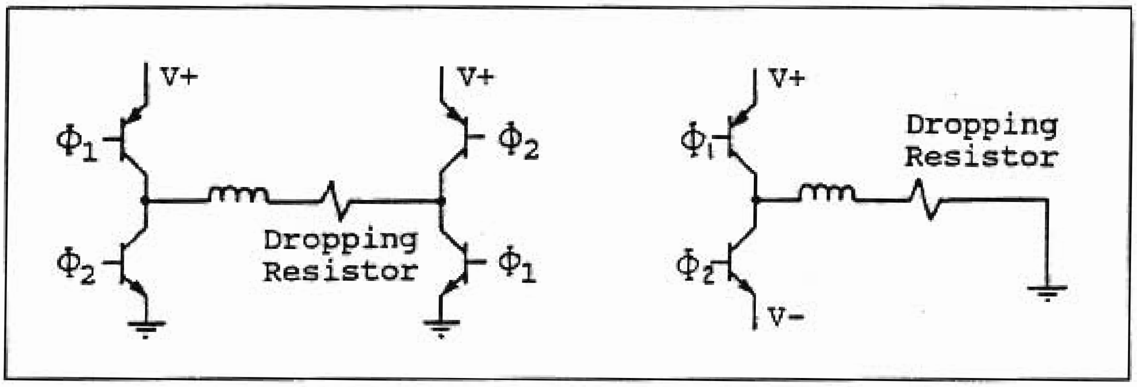


Figure 19.6 Bipolar driver designs.

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 it's step 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. This switching technique controls the power dissipation of the motor, keeping it at maximum torque for any step rate, by adjusting the average applied power. The highest performance drivers are usually designed as bipolar chopper circuits.

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

Interfacing to the 8255 or other PIO

In the example shown in Figure 19.9 below, the CY550 is operating in the parallel ASCII input mode. In this mode, bit 7 is always zero and b7 line of the CY550 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 IO_REQUEST strobe, thus allowing only one 8 bit output port to suffice. Of course, a separate port line may be used to generate IO_REQUEST, if desired. If the CY550 is operated in the parallel BINARY mode, all 8 data bus lines are used, and a separate IO_REQUEST line is required. Note that in the example shown, use is made of the fact that the data and the IO_REQUEST signal may be applied simultaneously in parallel operation. If query commands are 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 IO_REQUEST a separate line from the data ports.

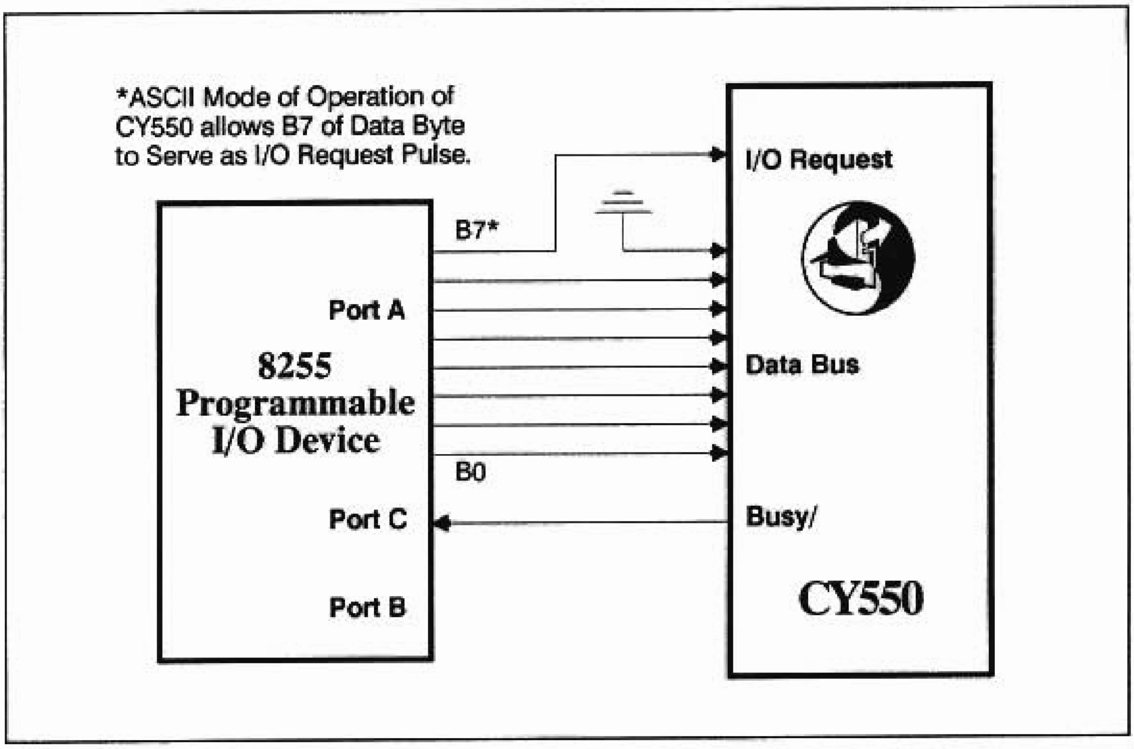

*ASCII Mode of Operation of CY550 allows B7 of Data Byte to Serve as I/O Request Pulse.
B7*
I/O Request
Port A
8255 Programmable I/O Device
Data Bus
B0
Port C
Busy/
Port B
CY550

Figure 19.9 Example Interface to CY550 using 8255 PIO.

Operation of Several CY550s Using a Common Data Bus

In systems where multiple CY550s 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 CY550s. Each of the separate BUSY lines (pin 15 of each CY550) must be monitored individually and each IO_REQUEST line (pin 13) must be activated separately. This technique effectively uses the IO_REQUEST line as a chip select (CS). A CY550 will ignore all bus information if its IO_REQUEST line is inactive.

This case would not allow programs to be executed from local external memory, since each CY550 data bus must be isolated to allow access to the external memory. Access to local external memory must be disabled by grounding each chip's XMEM_SEL line.

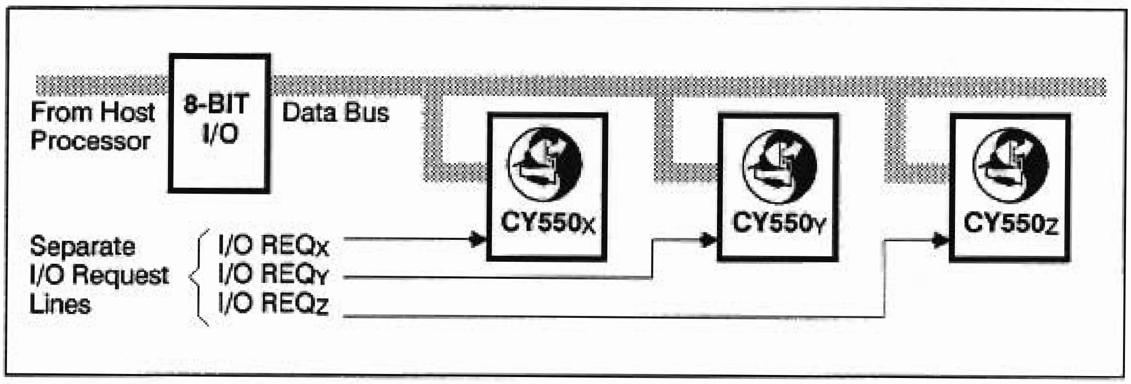
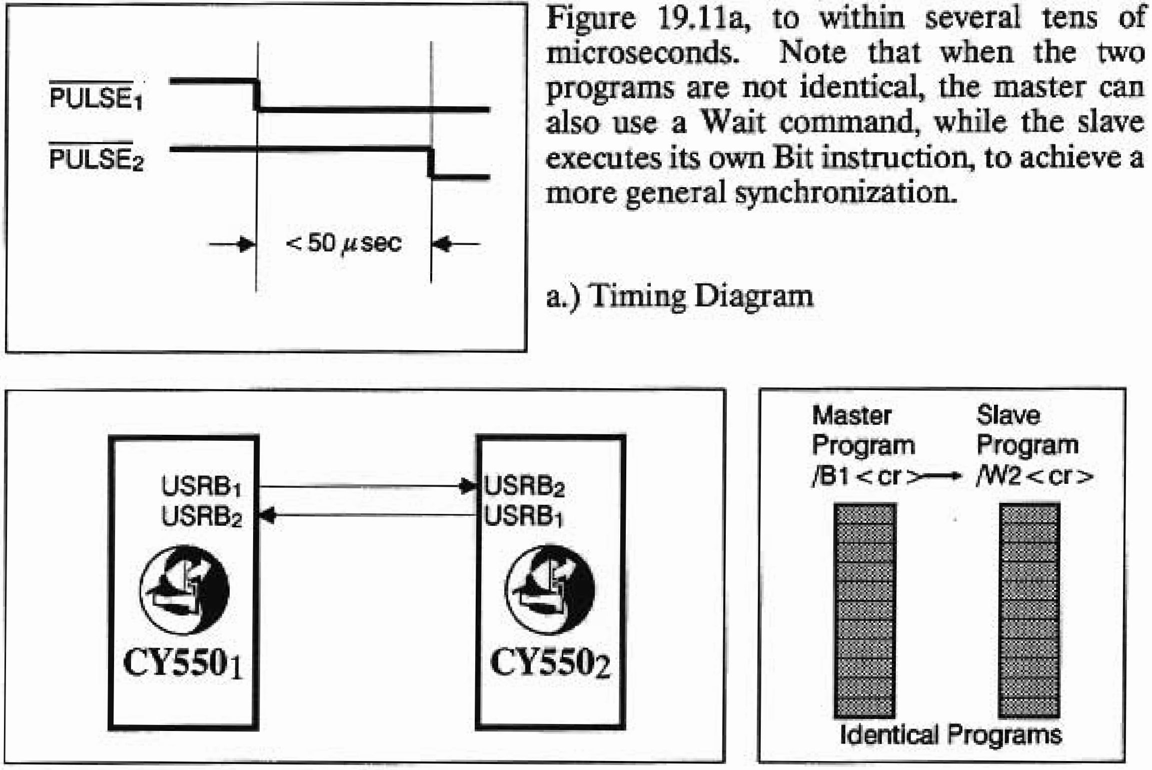


Figure 19.10 CY550s share common data bus by using separate I/O REQUEST lines for chip select.

Synchronization of Two CY550s

Two CY550s, executing similar programs from external memory, may be synchronized as shown in Figure 19.11. The master controller can control a USERBIT line of the slave CY550 via the Bit command, to set or clear a user bit. The slave CY550 is started first, with an Execute command, and executes a Wait command and waits until the USERBIT line is driven low by the Bit command executed by the master CY550, when it receives the (second) Execute command. Both CY550s then proceed to the next instruction and are synchronized as shown in Figure 19.11a, to within several tens of microseconds. Note that when the two programs are not identical, the master can also use a Wait command, while the slave executes its own Bit instruction, to achieve a more general synchronization.



a.) Timing Diagram

b.) Hardware

c.) Software

Figure 19.11 Synchronization of two CY550s.

Coordination of Several CY550s

Multiple CY550s may be synchronized to each other by use of the programmable User Bits, the Wait commands, Til commands, and time delays. These may also be combined with other signals, such as CCW, SLEW, or STOPPED, 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 CY550 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<cr>      Declare current position as home
R 200<cr>
S 225<cr>    Define stepping parameters
F 4<cr>
P 14<cr>     Move to the receiving position
V<cr>        Wait for end of motion
Y 50<cr>     Set external memory address

E<cr>        Define hand off program

    W 16<cr>     Wait for a part to arrive, User Bit 0 low
    P 0<cr>      Arms move together to handoff position
    +<cr>        Change direction after motion is over
    B 17<cr>     Activate mechanism to transfer part, User Bit 1 low
    D 90<cr>     Delay for part to actually transfer
    P 14<cr>     Move apart, back to receiving position
    V<cr>        Wait for end of motion
    D 90<cr>     Delay for part to stabilize, arms apart
    P 108<cr>    Transport part to next handoff position
    V<cr>        Wait for end of motion
    B 18<cr>     Low User Bit 2 indicates part arrived
    P 122<cr>    Move together with receiving arm
    V<cr>        Wait for end of motion
    B 1<cr>      Release mechanism which holds part
    D 90<cr>     Delay for part transfer to receiving arm
    P 108<cr>    Move apart, back to receiving position
    V<cr>        Wait for end of motion
    D 90<cr>     Delay for part to stabilize, arms apart
    R 1600<cr>   Change step rate to slower rate
    P 0<cr>      Move empty arm back for next part
    P 14<cr>     Stay at the receiving position
    R 200<cr>    Change rate back to faster rate
    T 3,50<cr>   Repeat program if User Bit 3 low
    0<cr>        Else stop program

Q<cr>      End of program
Y 50<cr>   Reset starting address
X<cr>      Execute program
```

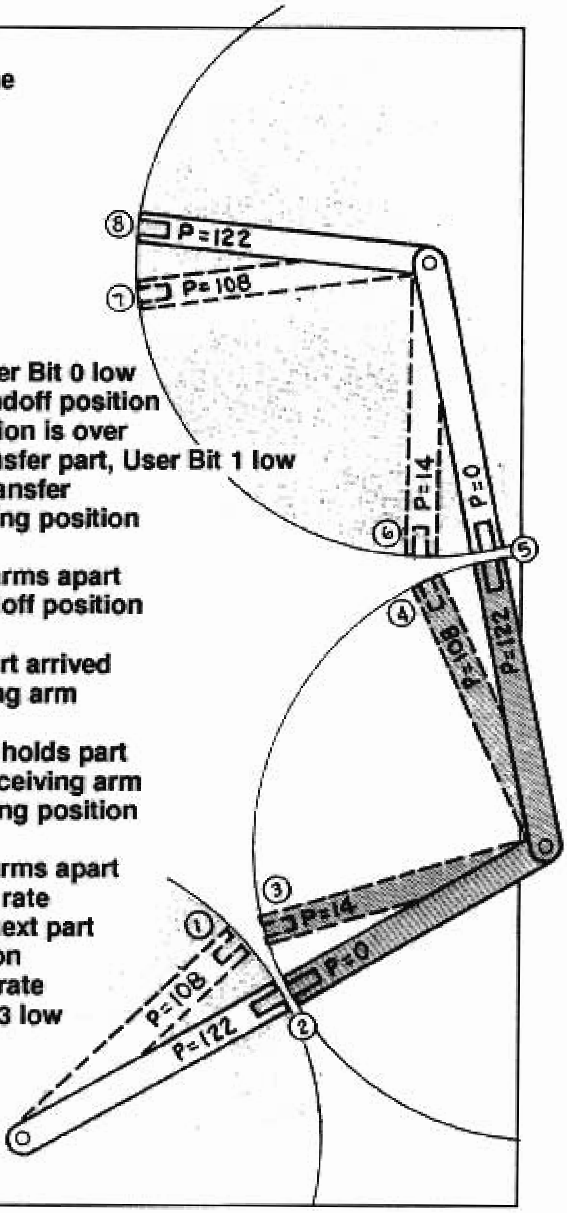
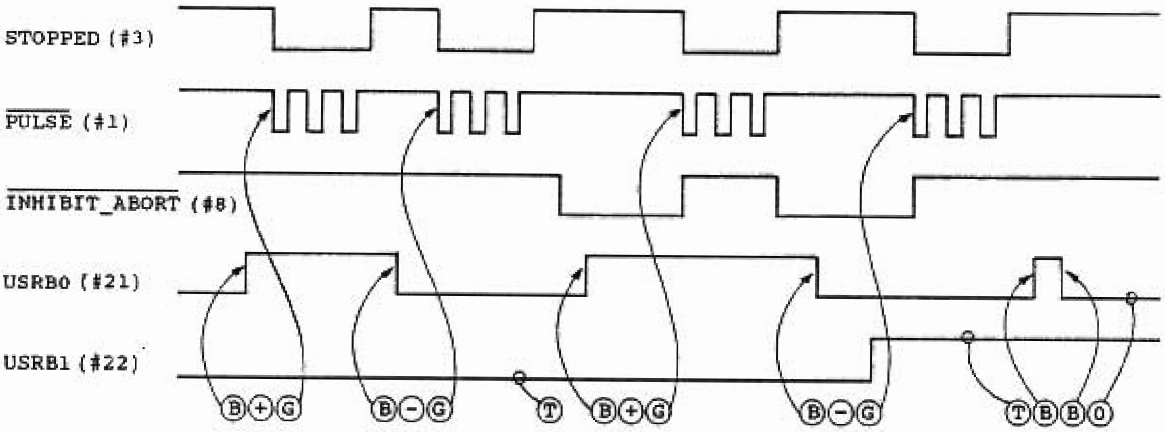


Figure 19.12 Synchronized part transfer example.

Example Programs and Waveforms

The use of the "loop TIL" instruction is illustrated in the following example. The program loops until the USERBIT 1 line (pin #22) goes high, then fetches the next instruction. The effect of the INHIBIT_ABORT input on the STOPPED output is also shown.



```
PRESET:
     B  16<cr>                  clear USER BIT 0 line
     R  250<cr>                 set RATE = 250
     S  200<cr>                 set SLOPE = 200
     F  2<cr>                   set FIRSTRATE = 2
     N  3<cr>                   set NUMBER steps = 3
     Y  100<cr>                 set memory address

ENTER PROG:
                 E<cr>
                 B  0<cr>              set USER BIT 0
PROGRAM          +<cr>                 set CW direction
CODE             G<cr>                 GO, begin stepping
                 V<cr>                 Wait for end of motion
                 B  16<cr>             clear output line
                 -<cr>                 set CCW direction
                 G<cr>                 GO, begin stepping
                 V<cr>                 Wait for end of motion
                 T  1,100<cr>          repeat above prog Til USER BIT 1 = HI
                 B  0<cr>              set USER BIT 0 line
                 B  16<cr>             clear USER BIT 0 line
                 0<cr>                 stop run mode, enter command mode

QUIT:            Q<cr>

EXECUTE:         Y  100<cr>            reset pointer
                 X<cr>                 EXECUTE
```

Figure 19.15 Timing and Control for Program Entry and Conditional Looping.

ASCII and Binary Data Programming Examples

The following command sequence illustrates a simple motion with trigger signals. First we toggle USER BIT 0, then we set the stepping parameters, and finally take five steps. In the ASCII command mode, the command sequence is as follows:

B 16<cr>	USER BIT 0 low
B 0<cr>	USER BIT 0 high
R 220<cr>	set Rate
S 240<cr>	set Slope
F 4<cr>	set First Rate
N 5<cr>	set Number of steps
G<cr>	take five steps
V<cr>	Wait for end of motion

If the same sequence is issued in the Binary command mode, the command letters are the same, but the data values are given to the CY550 in binary form, not as ASCII decimal values. In the Binary command mode, the equivalent sequence is:

42 01 10	USER BIT 0 low
42 01 00	USER BIT 0 high
52 02 DC 00	set Rate 220
53 01 F0	set Slope 240
46 01 04	set First Rate 4
4E 03 05 00 00	set Number of steps 5
47 00	take five steps
56 00	Wait for end of motion

All values are listed as their hexadecimal equivalents, but are sent to the CY550 as single 8 bit data values. Command letters have the same codes as in ASCII, but parameters require a count, followed by the binary parameter value. Notice that multi-byte parameters are sent least significant byte first.

The resulting waveforms are shown below. The CY550 behavior is the same in both ASCII and Binary modes, but the number of data bytes sent to achieve the results is different. Also, Binary mode commands will execute somewhat faster, since the CY550 converts ASCII parameter values into a binary format internally, taking additional command execution time.

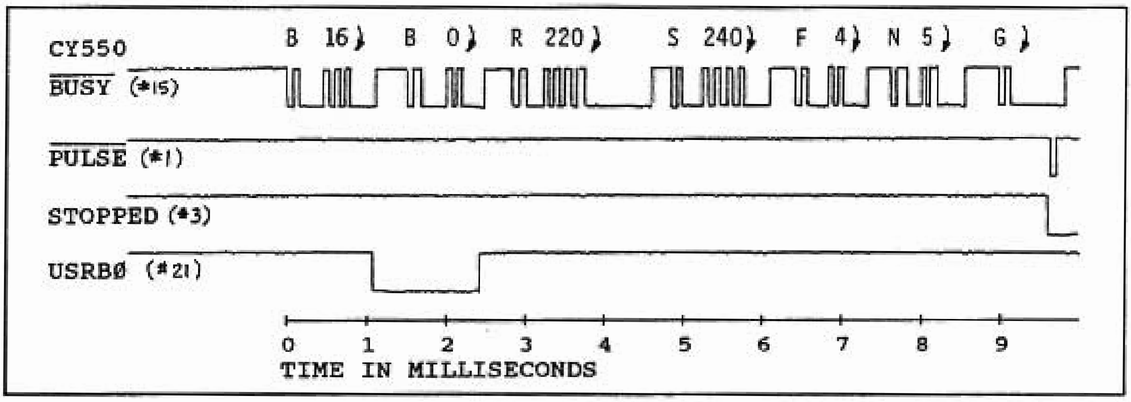


Figure 19.16 Example motion waveforms.

Getting Your CY550 Running

The following checklist will simplify getting your CY550 up and running.

1. Connect pin 20 to ground and pins 31 & 40 to +5 volts.

2. Be sure RESET (pin 9) is high for at least 10 milliseconds after power stabilizes. The CY550 can be reset at any time.

3. Upon proper reset, all outputs should be at logic 1, and the CY550 should be testing the JOG signal (pin 6).

4. Observe the crystal frequency divided by 6 on ALE, pin 30.

The following steps may be used with the Parallel Command Interface.

5. Place the "+" command on the data bus, ASCII value 2Bh

D0 = 1
D1 = 1
D2 = 0
D3 = 1
D4 = 0
D5 = 1
D6 = 0
D7 = 0

6. Lower the IO_REQUEST line, pin 13.

7. Wait for the BUSY line to go low before bringing IO_REQUEST high again. BUSY is pin 15.

8. When IO_REQUEST is brought back high, BUSY will return high.

9. Wait for BUSY to return high, then place the carriage return code on the data bus, value 0Dh

D0 = 1
D1 = 0
D2 = 1
D3 = 1
D4 = 0
D5 = 0
D6 = 0
D7 = 0

10. Generate the IO_REQUEST handshake strobe again, interacting with the BUSY signal.

11. Upon completion of the above sequence, CCW (pin 2) will go low.

The following steps may be used with the Serial Command Interface.

12. Connect your serial port to the CY550 RxD and TxD signals, pins 10 and 11, with the proper voltage translation buffers. CTS may also be connected, and is recommended, if your system can support it.

13. Send two carriage return codes to adapt the CY550 baud rate. The CRT emulation program at the end of this section is useful for this purpose.

14. Send the command "+ <cr>".

15. The CCW signal (pin 2) will go from high to low.

16. Repeat the above command sequence with a "- <cr>" command. The ASCII code for "-" is 2Dh. The CCW signal will return high when this command is sent.

If you have successful parallel or serial communications....

17. If you reach this point successfully, you are commanding the CY550, and should be able to enter any command and obtain the correct response.

18. Suggested sequences:

a. Try to toggle a User Bit signal using "B 0 <cr>" and "B 16 <cr>" for pin 21. Remember the single space between the command letter and parameter value.

b. Change the stepping parameters, using the R, F, S, and N commands, then try stepping with "G <cr>".

c. Refer to examples in this manual for other command sequences.

19. After the initial checkout, use your own command sequences as required by the application. You may need to exercise the external memory interface, extended I/O circuits, or an external display.

BASIC Language CRT Emulation Program

The following BASIC language program emulates a CRT for direct serial communications with the CY550. It starts by sending the two carriage returns to set the CY550 baud rate. CTS monitoring is enabled and used by the serial port driver. The program is a simple terminal emulator, which takes any command entered from the keyboard and sends it to the CY550, while also displaying any serial data sent by the CY550.

```
100 '              BASIC Language
110 '              CRT EMULATOR FOR CY550
120 '              Cybernetic Micro Systems, Inc.
130 '
140 '     This program configures the CY550 for serial commands,
150 '     by sending two carriage returns to set the adaptive
160 '     rate.
170 '     It then becomes a terminal emulator, waiting for keyboard
180 '     input, which it displays on the screen and sends to the
190 '     CY550, while displaying all serial characters received
200 '     from the CY550.
210 '
300 CLS
310 LF$=CHR$ (10)  :  CR$=CHR$ (13)  :  NL$=CHR$ (0)  :  ES$=CHR$ (27)
320 '
330 '     Open the COM1 serial port at 9600 baud, no parity
340 '     Note that CTS is also enabled
350 '
360 OPEN "COM1:9600,N,8,1,CS30000,DS0,CD0" AS #1
370 '
380 '     Send two carriage returns to adapt the CY550 baud rate
390 '
400 PRINT #1,CR$;CR$;
410 '
500 LOCATE 5,5,1
510 PRINT "Ready to Go!"
520 '
530 '     Open the screen for displays
540 '
550 OPEN "SCRN:" FOR OUTPUT AS #2
560 '
570 '     Check for keyboard input.  Display and send any
580 '     keys to COM port.  Stop when Escape key input
590 '
600 A$=INKEY$  :  IF A$=ES$  GOTO  900
610 IF A$< >" " THEN PRINT #1,A$;  :  PRINT #2,A$;
620 '
630 '
640 '     Check for any received data from COM port, and
650 '     display it, with line feeds filtered out.  Loop back
660 '     to check keyboard again when no more received data
670 '
680 WHILE NOT EOF(1)
690   J%=LOC(1)  :  B$=INPUT$(J%,#1)  :  LF%=0
700   LF%=INSTR(LF%+1,B$,LF$)
```

```
710    IF LF%>0 THEN MID$(B$,LF%,1)=NL$  :  GOTO  700
720    PRINT #2,B$;
730 WEND
740 GOTO  600
750 '
900 '      Exit program when Escape key is pressed
910 '
920 CLOSE #1  :  CLOSE #2
930 STOP
```

Stepper Motor Controller Selection Guide

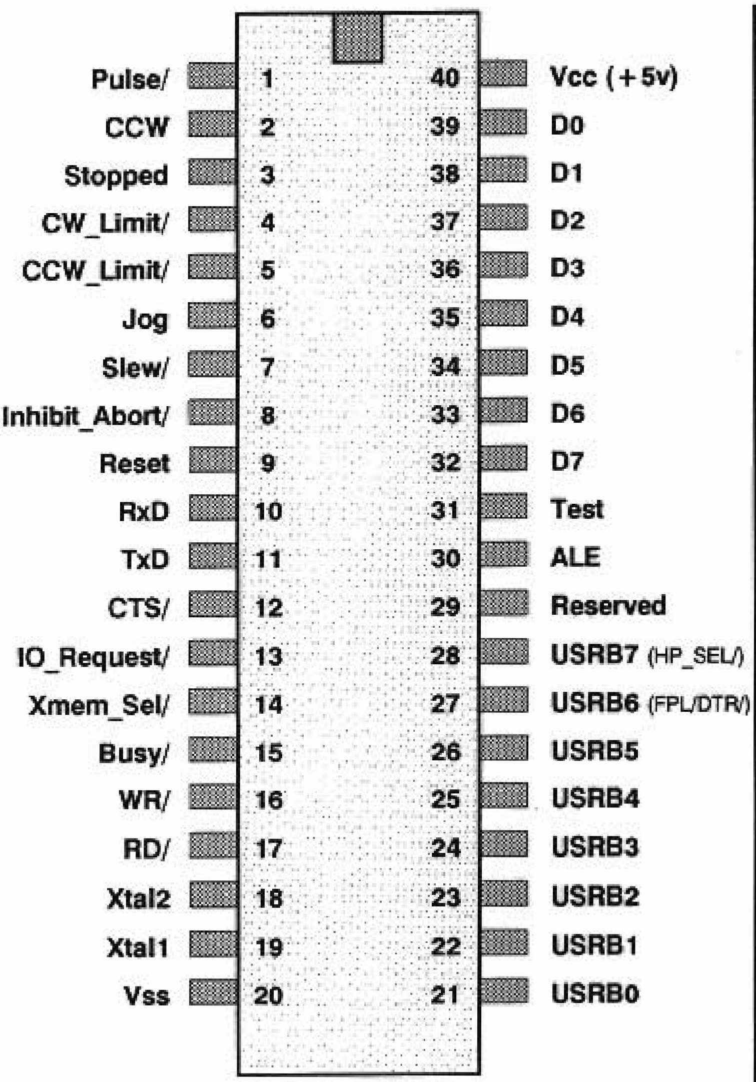
CY 500	CY 512	CY 525	CY 545	CY 550	Function
2K	5K	10K	27K	20K	Max Usable Step/Sec
64K	64K	64K	16M	16M	Max Number of Steps
21	25	26	28	38	Number of Instructions
18	48	60	64K	64K	Program Storage (bytes)
Exp	Exp	Lin	Lin	Lin	Accel (Exponent/Linear)
					Command Interface
			●	●	Serial Interface
●	●	●	●	●	Parallel Interface
●	●	●	●	●	Binary Data Structure
●	●	●	●	●	ASCII Data Structure
●	●	●			Internal Stored Program
			●	●	External Stored Program
●	●	●	●	●	Direct Command Mode
			●		Thumbwheel Input Support
			●	●	External Display Support
	o		●	●	Stand Alone Operation
					Program Features
	●	●	●	●	List Prog Buffer Contents
	●	●	●	●	Display # of Steps Param
	●	●	●	●	Display Accel Parameter
	●	●	●	●	Display Step Rate
	●	●	●	●	Display Current Position
			●	●	Display Val of Ext Inputs
				●	Display Motor Status
●	●	●	●	●	Stored Prog Execution
●	●	●	●	●	Conditional Prog Structure
●	●	●	●	●	Programmable Time Delay
	256	256	64K	64K	Program Repetition Count
	●	●	●	●	Unconditional Branch
1	1	1	16	16+	Programmable I/O Lines
				●	Extended I/O
		●			Program Labels
●	●	●	●	●	Multi-controller Sync
		●	●	●	Live Cmds During Prog Exec
				●	Live Cmds while Stepping

CY 500	CY 512	CY 525	CY 545	CY 550	Function
					Motor Support
	●	●	●	●	Pulse & Direction Output
●	●	●			4-phase Output
●	●	●	o	o	2-phase Compatible
	o	o	o	o	3-phase Compatible
●	●	●	o	o	4 phase Compatible
	o	o	o	o	5-phase Compatible
●	●	●	●	●	Full Step
●	●	o	o	o	Half Step
		o	o	o	Quad Step
			o	o	Micro Step
●	o	o	●	●	Single Step (Jog Mode)
o	o	●	●	●	Continuous Stepping
●	●	●	●	●	Constant Rate stepping
●	●	●	●	●	Ramped Stepping
			●	●	Low Power Standby
			●	●	Motor On/Off Output
o	o	o	●	●	Limit Detection
					Motion Features
o	o	●	●	●	Programmable Start Rate
●	●	●	●	●	Programmable Slew Rate
o	o	●	●	●	Program. Accel/decel Slope
●	●	●	●	●	Software Direction Control
	●	●	●	●	Automatic Direction Finding
●	●	●	●	●	Program. Num of Steps
			●	●	Home Seek Command
●	●	●	●	●	Absolute Position Stepping
●	●	●	●	●	Relative Num of Steps
●	●	●	●	●	Emergency Stop/Abort
	●	●	●	●	Decelerating Stop/Abort
●	●	●	●	●	Step Inhibit Input
	●	o			Closed Loop Rate Control
●			●	●	External Direction Control
●	●	●	●	●	Motion Complete Indicator
	●	●	●	●	Slew Indicator
●	●	●	o	o	Prog Complete Indicator
●			●	●	External Jog Mode
		●		●	On-the-Fly Rate Change
		●		●	On-the-Fly Position Output
				●	On-the-Fly Cmd Execution

●	Yes
o	To a Degree or with Additional Work

CY550 Summary

CY550 Pins



CY550 Commands

A	p	At current step position
B	b	user Bit set or clear
C		Continuous Step mode
D	d	Delay milliseconds
E		Enter External memory prog
F	f	Firstrate
G		Go, step relative
H	h	Home seek
I		Initialize
J	a	Jump to address
K	a	Set read pointer addr
L	c,a	Loop to addr for count
M	a	Set write pointer addr
N	n	Number of Steps
O	o	Mode
P	p	Position for stepping
Q		Quit entering to external mem
R	r	Rate max
S	s	Slope of accel/decel
T	b,a	Til bit matches, loop to addr
U		reserved
V		Wait for step to finish
W	b	Wait for bit match
X		eXecute external commands
Y	a	external memory addr pointer
Z	c,a	ZillionLoop to addr for count
+		CW direction
–		CCW direction
/		Negate or clear bit values
?	cmd	Query command parameter
0		Stop execution of ext memory
\		Wait number of steps
]		Wait for position match
^		Stop motion
#	v	Set I/O byte register value
!		Read from mem to I/O byte reg
%		Write from I/O byte reg to mem
[	a,c,d	HP-LED command string
"chr"		display string between quotes