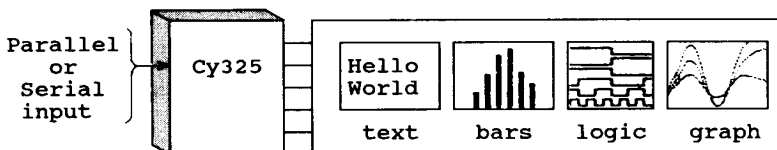


## Introduction to Cy325 LCD Windows Controller

The size, cost, and appearance of LCD displays have finally reached the point where most designers of new instruments and other systems are considering them as the primary user interface. The Cy325 LCD Windows Controller is a new high-level text and graphics controller chip that works with "instrument size" LCDs. The primary LCD sizes supported by the Cy325 include:

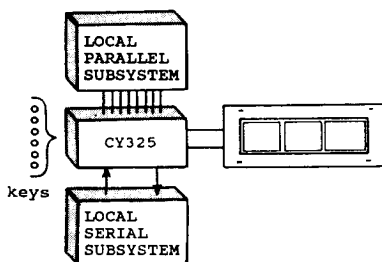
64 x 120 pixels = 8 rows x 20 characters	AND-1021 LCD
64 x 240 pixels = 8 rows x 40 characters	AND-711A LCD
128 x 240 pixels = 16 rows x 40 characters	AND-1091 LCD

In its simplest "stand-alone" application, the Cy325 provides an easy-to-use LCD controller with parallel and/or serial interface. Its primary purpose in such stand-alone applications is to simplify the display of messages in windows with sophisticated bargraph, logic waveform, and graphics support.

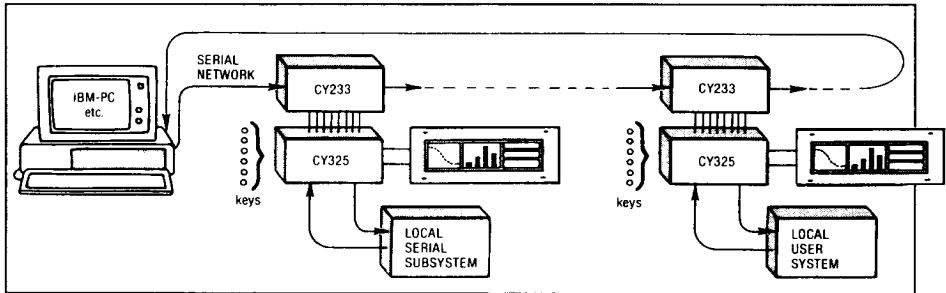


Instead of having to send low-level pixel-oriented commands, the Cy325 allows you to send high-level ASCII commands with decimal, hexadecimal, or binary arguments. These commands create windows, erase windows, draw bargraphs, analog waveforms, etc. While you issue four or five commands to the Cy325, the Cy325 may issue hundreds of commands to the LCD, thus relieving you of the effort involved. The Cy325 also allows you to use simple coordinate systems while the LCD actually requires complicated mapping of pixel coordinate locations into the display coordinate space.

An exceptionally powerful feature of the Cy325 is its ability to accept inputs via either the serial or the parallel channel or **BOTH serial AND parallel**. Another input channel consists of six keys or special signal lines that offer several unique functions. The combination of these signal channels allows the Cy325 to link **two subsystems** (of your choice) together via sophisticated switching between the Cy325 parallel bus and the Cy325 serial bus as shown at right.



Most applications will probably use the Cy325 as a simple display controller and can ignore the deeper levels of functionality that are available in the Cy325. However, in addition to sophisticated control of windows on graphic LCDs, the Cy325 possesses numerous features that qualify it as a **System Element** that can serve as a link between major subsystems, tying together the display function, the user input function, and also providing an interface to your system processor.



## RS-232, Parallel, and Network Interface

The Cy325 provides both TTL parallel and serial interfaces, and also provides a convenient interface to Cybernetic's Cy233 Local Intelligent Network Controller, allowing up to 255 LCDs to be attached to a single serial I/O port (such as COM1 of an IBM-PC).

## Windows Control

The CY325 LCD Windows Controller from Cybernetic Micro Systems is the first LCD controller that is designed to create and manage windows on graphic LCDs (up to 240 x 128 pixels). Any of over 250 default or "built in" windows can be selected via simple commands, or a user defined window can be specified by a single command. Both text and graphics can be written into the current windows with automatic cursor management, clipping, etc. Text and graphics can be independently written, erased, or overlaid in a single window, or text can be written to one window and graphics to another. Windows can be defined within windows. Graphics operations, defined in terms of the "current" window, include automatic histogram generation and logic waveform display. For example, histograms (bargraphs) can be generated by simply specifying the heights of the bars as arguments.

## Window-Relative Pixel Plotting

The Window-relative Pixel Plotting mode allows graphics curves to be plotted in a specified window. For example, the output of an A/D converter can be graphed in a window by simply sending the values to the CY325.

## Internal Mode Registers

The CY325 possesses a large number of special modes of operation that can be enabled or disabled by setting or clearing a bit in the appropriate mode register. More than two dozen mode bits allow hundreds of variations to be easily specified.

## Automatic Logic Waveform Display

The CY325 allows the automatic generation of Logic Waveforms in any window, via six pins that drive digital logic waveforms in the specified window. Digital instrument designers can easily produce logic waveforms by simply driving these pins!

## Soft-Key Support in the Cy325

A PC Magazine survey (April 12, 1988) found that over 95 per cent of those surveyed thought that a **Menu Driven** interface is best for new users of software. Believing that the same is true for today's complex instruments, we have provided appropriate features in the Cy325. Specifically, the CY325 supports from one to sixteen "softkeys" in either standalone operation or in a network, allowing a host computer to display the "meaning" of each key on the LCD. When the user pushes a particular key the result is transmitted to the host. A simple built-in protocol assures that all keys are uniquely identified and acknowledged, even in a network, allowing numerous users to make menu choices or otherwise use the soft-keys, with a central host managing the responses and updating the displays.

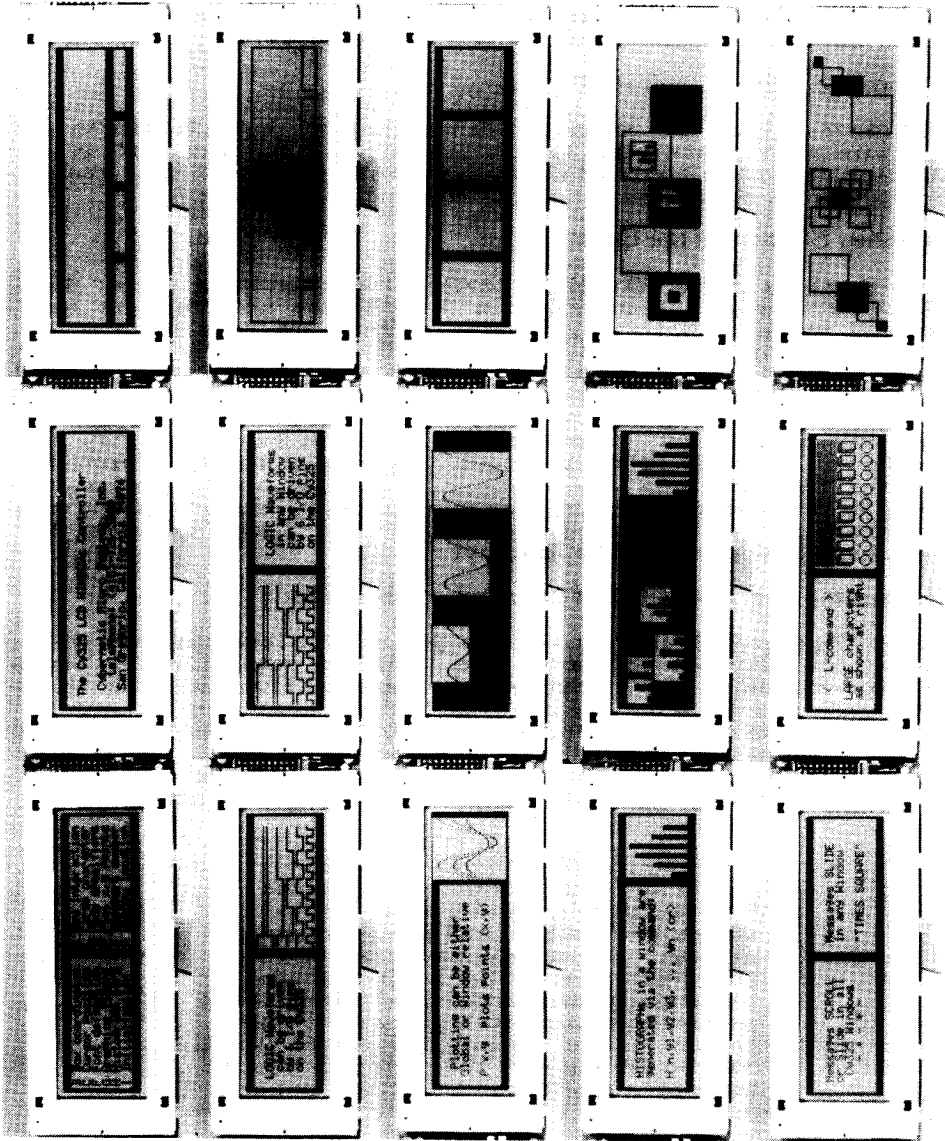
## Instrument Design and Other Intended Applications

While "laptop PCs" usually require 640 x 200 pixel displays to be compatible with DOS software, such LCDs are generally too large and too expensive for instruments and handheld units. The CY325 LCD Windows Controller controls LCDs with formats up to 240x128 pixels (16 rows x 40 characters). This size and resolution are ideal for instruments and display panels designed to display status, waveforms, and to display clear, easy-to-read messages, as opposed to pages of text, as in word processors, etc.

### Cy325 Features

Serial or 8-bit Parallel interfaces  
"Built-in" or user-defined Windows!  
Window declaration and manipulation  
Window-relative text and graphics!  
Logic waveforms built-in (see photo)  
Large characters and user-defined fonts  
Bargraphs automatically size to window  
Communications between Serial and Parallel!  
"Soft-key" support for menu management!  
Network support based on Cy233 Network chip.

# Example Cy325 Photos





# CY325

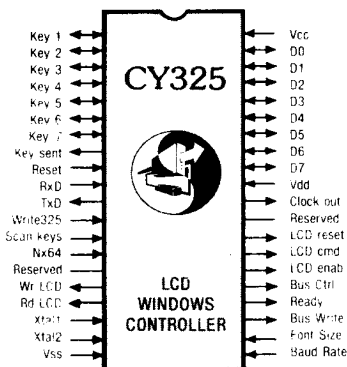
## LCD Windows Controller

The CY325 LCD Windows Controller is a standard 5 volt 40 pin CMOS LSI device designed to provide high level control of "instrument size" LCDs with up to 16x40 characters and 128x240 pixel graphics. The CY325 allows simple parallel 8 bit or serial (RS232 protocol) interface to any standard computer. Up to 256 built-in windows or any number of user defined windows support windows-relative text, plotting, bargraphs, waveforms, etc. Both text and graphics can be written into the "current" window or text can be written to one window and graphics to another. Switching between windows is fast and simple. The CY325 interprets characters in command mode and displays them in display mode.

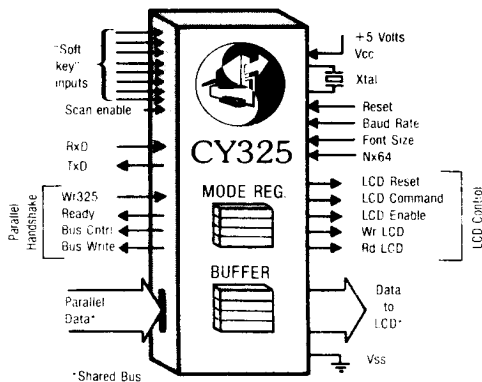
### Standard Features

- Command mode and display mode
- 8 rows x 40 or 16 x 40 characters
- 64 x 240 or 128 x 240 pixel graphics
- 5 x 7 built-in character font
- User definable font selection
- Parallel input with simple interface
- Serial input channel
- CMOS 5 volt 40 pin device
- ASCII decimal, HEX, binary data
- 6 pins drive logic waveforms
- Up to 8 "soft" keys
- 4 x 4 key scan of soft keys
- 256 built-in windows
- Any number of user defined windows
- Bargraphs automatically size to windows
- Pixel plotting is global or window-relative
- Mode registers allow hundreds of modes
- Large characters can be user defined
- Automatic cursor management in windows
- Wrap or scroll text in windows
- Separate text and graphics "planes"
- Network CY325s with CY233 Network chip
- Erase text or graphics in current window
- Vertical or horizontal scroll in windows

### Pin Configuration



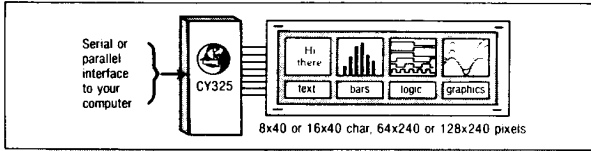
### Logic Diagram



# CY325 LCD Windows Controller

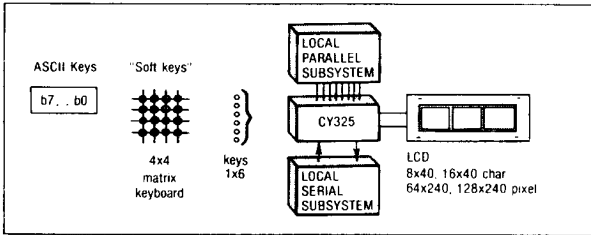
## LCD Windows Controller

With both 8-bit parallel and serial interfaces available, the CY325 provides easy-to-use text and graphics display in multiple windows on an LCD. The CY325 supports sophisticated bargraph, logic waveform and analog waveform displays with simple commands.



## “Soft Keys” Support Menu Management

The CY325 can detect key closures and announce them to the host. The meaning of each “soft” key can be derived from a menu displayed on the LCD, thus providing almost infinite flexibility. The key switches can be ASCII, 4x4 matrix, or switches, depending on the key mode selected. A “positive interlock” with the host is provided.

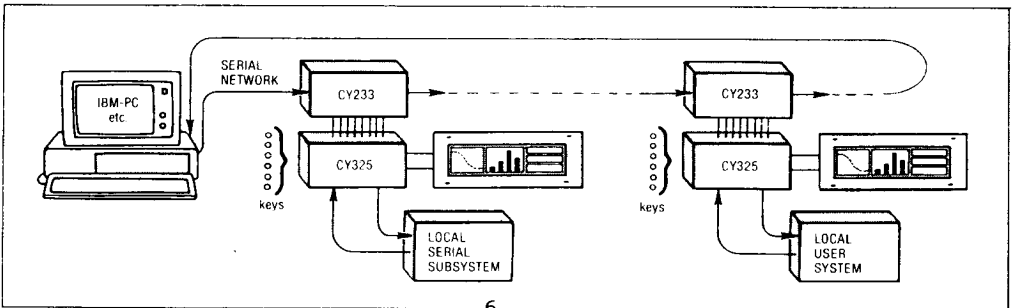


## CY325 “System Element”

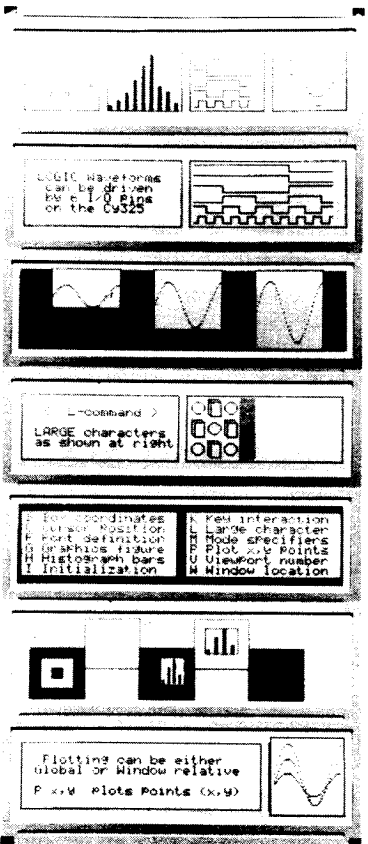
The CY325 can transfer data between the parallel and serial channels and thus link local parallel and serial subsystems to each other as well as to the LCD.

## CY325/CY233 Networked Displays

With built-in interface support for the CY233 Network Controller, the CY325 LCD controller allows up to 255 local stations to communicate with one computer via RS-232 communications.



## Typical CY325 LCD Displays\*

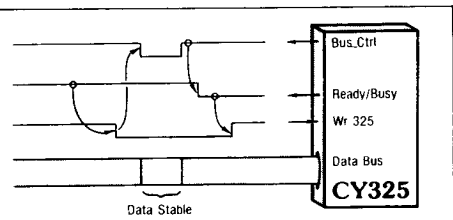


\* AND 711A-ST or Toshiba TLX-711A

# CY325 Interface to Computer

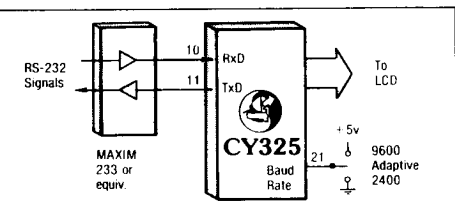
## Parallel Interface

When the master detects the CY325 ready to accept output (Ready=hi) it lowers Wr325 to request a transfer. The CY325 lowers Bus Ctrl to signal the Bus is available, then lowers Ready to signal completion of transfer (busy with data). Bus Ctrl can either be tested in software or simply used to control tri-state outputs.



## Serial Interface

The CY325 requires only an RS232 driver to provide an RS-232 interface. The Baud rate pin (#21) selects 1200 (lo), 9600 (hi), or adaptive (floating) in which two carriage returns are sent to the CY325, which then matches the baud rate.



## CY325 Interface to LCD

CY325 pin #	LCD pin #	Description	CY325 pin #	LCD pin #	Description
40	3	5 volt	39	11	Data 0
20	2	Ground	38	12	Data 1
16	5	WR LCD	37	13	Data 2
17	6	RD LCD	36	14	Data 3
22	19	Font size #	35	15	Data 4
26	7	LCD enable	34	16	Data 5
27	8	LCD command	33	17	Data 6
28	10	LCD reset	32	18	Data 7

The CY325 is designed to work with a variety of different LCDs available from AND Company and from Toshiba, Inc. The AND LCD displays compatible with the CY325 are listed below with their formats.

1091	128 x 240	16 x 40 char	
711 and 711A	64 x 240	8 x 40 char	
1021	64 x 120	8 x 15 char	
1101	32 x 160	4 x 20 char	
1012	128 x 160	16 x 20 char	

# CY325 Commands

All CY325 commands are listed below. While many are self explanatory, the interpretation of some of them is dependent upon the operating mode. A complete description is beyond the scope of this data sheet, however a detailed user manual (100 pages) is available from Cybernetic Micro Systems.

^ C	Command Mode Select
^ D	Display Mode Select
^ K	Klear Current Window
^ N	Select Special Font (Shift Out)
^ O	Select Normal Font (Shift In)
^ W	Window Swap Command
B	Box Command: $B x_1 y_1 x_2 y_2$
C	Cursor Positioning: $C x, y$
F	Font Creation: $F n, d_1 \dots d_7$
G	Graphics Line: $G n, d_1 \dots d_n$
H	Histogram: $H n, Y_1 \dots Y_n$
I	Initialize: $I n$
K	Key Acknowledge: $K n$
L	Large Character Mode: $L$
M	Mode Register Load: $M n, v$
P	Pixel Plot Command: $P x, y$
S	Send Data thru CY325: $S n$
V	Viewport Select: $V n$
W	Window Specify: $W x_1 y_1 x_2 y_2$
Z	Horizontal Scroll Mode: $Z$
/	Negation (prefix to command)
+	Window Status Save
-	Window Status Restore
?	Query Command
@	Transmit Data thru CY325
{	Plot String Command
.	Lower half of 128 (prefix to Viewport)

## CY325 Command Format and Data Types

In the command mode, incoming characters are interpreted as follows:

$$A x_1, x_2 \dots x_n < CR >$$

where 'A' is any ASCII character, followed by a space, and  $x_1$  through  $x_n$  are relevant parameter values, separated by commas and terminated by a carriage return. Parameters can be expressed as ASCII decimal or ASCII Hex numbers ('h' suffix). If the CY325 is placed in Binary Mode, the format is:

$$B N x_1 x_2 \dots x_n$$

B is any valid command, N is a data count followed by N 8-bit binary values  $x_1$  to  $x_n$ .

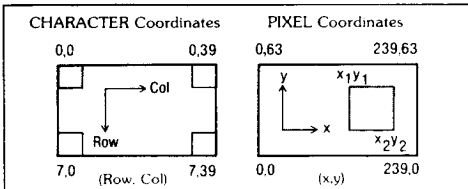
## CY325 Windows, Viewports, and Boxes

WINDOWS possess both character and pixel coordinate systems as shown below. BOXES possess only pixel coordinates. Boxes and windows are specified by defining their top left corner and lower right corner, with window corners expressed in character coordinates and box corners in pixel coordinates. A VIEWPORT is simply a "built-in" window whose size and location is implicitly defined. A viewport is identified by a number. The relevant commands are:

V n ↓	n = viewport ID number
W x <sub>1</sub> y <sub>1</sub> x <sub>2</sub> y <sub>2</sub> ↓	x, y = character coordinates
B x <sub>1</sub> y <sub>1</sub> x <sub>2</sub> y <sub>2</sub> ↓	x, y = pixel coordinates

## CY325 Coordinate Systems

The CY325 uses character and pixel coordinate systems. Text origin is upper left (like IBM-PC) while graphics origin is lower left (like algebra). Although defined in global (LCD based) coordinates, the current box or window establishes a local pixel coordinate system with the (0,0) origin at the lower left corner.



## CY325 Mode Registers

The CY325 operational mode depends on numerous mode bits grouped in registers. The two dozen mode bits are described in the Cybernetics CY325 user manual. Mode registers can be loaded via the 'M reg. value j' command and queried via the '?' command.

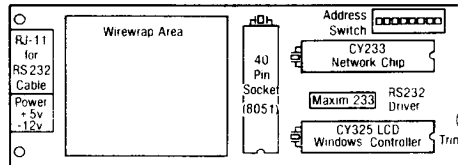
CY325 Mode Register #	Description
0 Window Status	Primary Status Registers
1 LCD Mode	
2 Key Mode	
3 Communications	
4 Key Image	Current Window
5 Viewport #	
6 Cursor Row	Character Cursor
7 Cursor Col	
8 Top Row	
9 Left Col	
10 Bot Row	
11 Right Col	Current Window Graphic Coordinates (Global)
12 x <sub>1</sub> Left Pixel	
13 y <sub>1</sub> Top Pixel	
14 x <sub>2</sub> Right Pixel	
15 y <sub>2</sub> Bottom Pixel	
16 Save View #	Window Save Status Registers
17 Save Row	
18 Save Col	
19 Save Status	

## CY325 Prototyping Support

A prototyping kit for the CY325 is available from Cybernetic Micro Systems. The CYB-003 will support one CY325, and one CY233 Local Network Control and also provide an 8051 socket and wirewrap a along with all connectors and relevant hardware. The board format is identical to the AND 711 LCD form

Space reserved for photographs

## CYB-003 Functional Diagram:



CY325 Electrical Specifications		SYM	PARAMETER	MIN	MAX	UNIT	REMARKS
<b>ABSOLUTE MAXIMUM RATINGS:</b>							
Operating Temperature	Storage Temperature	V <sub>CC</sub>	Level Supply Current	0	14	mA	
		V <sub>BE</sub>	Input High Level	0	5	V	(3.5V for XTAL) RES1
		V <sub>BE</sub>	Input Low Level	-0.5	0.5	V	
		I <sub>CC</sub>	Static Bus Leakage	0	1	µA	High Impedance State
		V <sub>OH</sub>	Output High Level	2.4	5	V	I <sub>OH</sub> = 80 µA
		V <sub>OL</sub>	Output Low Level	0.45	1	V	I <sub>OL</sub> = 1.6 mA
		f <sub>OSC</sub>	Crystal Frequency	3.5	12	MHz	See Clock Circuit
<b>DC &amp; OPERATING CHARACTERISTICS:</b>							
		V <sub>CC</sub>	Supply Voltage	5	5.5	V	



## Description of Cy325

The Cy325 is a CMOS 40 pin integrated circuit that is designed to drive AND and Toshiba Liquid Crystal Displays based on the T-6963 low level controller. Examples of T6963-based LCDs controlled by the Cy325 include:

Display	Pixel format	Character format
AND 711A	240 x 64 pixel	8 row x 40 char
AND 1021	120 x 64 pixel	8 row x 15 char
AND 1091	240 x 128 pixel	16 row x 40 char

## Command and Display Modes

The Cy325 operates in two primary modes, Command and Display.

### Display Mode:

The default mode is display mode: on powerup the Cy325 displays characters input to it either serially or in parallel. These characters are not interpreted ( except for <cr> = CRLF ) but are simply displayed on the LCD at the current cursor position.

### Command Mode:

The Cy325 is placed in the command mode via the Ctrl-C command. The Ctrl-C (03h) input to the Cy325 will not be displayed, but will switch the Cy325 into the command mode. In this mode the characters are NOT displayed, but are interpreted by the Cy325 according to their meaning described in following sections.

## Command/Display Selection Commands

The Ctrl-C command causes the Cy325 to enter the **Command mode** in which received characters are interpreted, rather than displayed as text in the text window. The hex value of this command is 03h, that is, the ASCII Ctrl-C character. The Ctrl-C command can be issued to the Cy325 via either the parallel or the serial channel and will have the same effect for either channel.

The Ctrl-D command causes the Cy325 to enter the **Display mode** of operation in which all printable ASCII characters received either serially or via the parallel channel are displayed in the text window, or at the specified cursor location on the LCD screen. To summarize:

**Ctrl-C**      selects Command mode operation.

**Ctrl-D**      selects Display mode operation.

# Cy325 Features

## Popular Interfaces:

Serial Interface ( RS-232 timing ) with Clear-To-Send  
Parallel Interface ( 2-wire handshake + Bus Control )  
RS-232 Network compatible ( Cy233 Network Controller )

## Display Mode Features include:

Display mode contains built in ASCII font.  
Display mode displays user defined fonts.  
Display mode allows special large font display.

## ASCII or Binary commands allow:

Font definition  
Graphic line definition  
Window specification  
Box drawing  
Pixel plotting:    Global ( LCD-relative )  
                  Local ( Window-relative )  
                  Clipping (optional)  
Mode definition  
Viewport selection from default windows.  
Erase characters in Window  
Erase Graphics in Window  
Scroll or Wrap characters in Window  
Cursor Positioning and mode definition  
Save/Restore Window status  
ASCII-Decimal, Hexadecimal, or Binary parameters  
Histo-graph auto-generation  
"Times-Square" horizontal scroll  
Communications "Switch" between Serial and Parallel

## Cy325 Serial Interface

The Cy325 accepts 8-bit serial data with no parity and one stop bit required. The Baud\_Rate pin (#21) specifies the serial rate:

Baud_Rate Pin	Cy325 Baud Rate
HI	9600
LO	2400
floating	adaptive

In **adaptive** mode, the Cy325 waits for two successive <cr> characters and adjusts its rate to match the carriage returns rate.

The Cy325 **CTS** line (pin 15) is an active low **Clear-To-Send** serial status line. When this line is low the Cy325 is ready to accept serial characters on the **RxD** line (pin 10). When this line goes high, serial input should be held off while the internal buffer is unloaded. The **CTS** line will go low again after the internal buffer has been unloaded and is ready to accept new serial input.

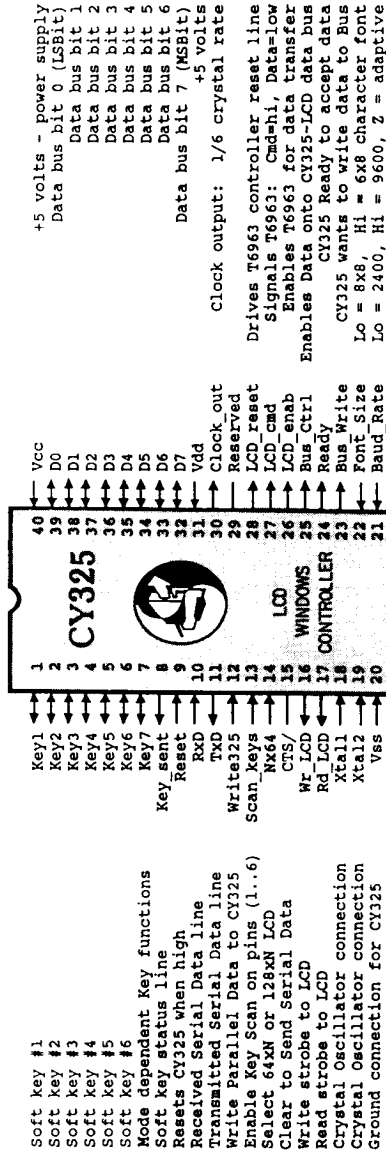
# 2 CY325 Pin List & Pinout Diagram 2

## Cy325 Pin List

Pin #	I/O	Name	Description
1	IO	Key1	Soft key #1
2	IO	Key2	Soft key #2
3	IO	Key3	Soft key #3
4	IO	Key4	Soft key #4
5	IO	Key5	Soft key #5
6	IO	Key6	Soft key #6
7	IO	Key7	ASCII or 4x4 matrix only
8	O	Key_sent	Softkey status or 4x4 matrix
9	I	Reset	Resets Cy325 when high
10	I	RxD	Received Serial Data line
11	O	TxD	Transmitted Serial Data line
12	I	WWrite325	Write Parallel Data to Cy325
13	I	Scan_Keys	Enable Scan Mode
14	I	Nx64	Select 64 or 128 pixel depth
15	O	CTS/	Active Low Serial Clear-To-Send
16	O	Wr_LCD	Write strobe to LCD
17	O	Rd_LCD	Read strobe to LCD
18	I	xtal1	Crystal Oscillator connection
19	I	xtal2	Crystal Oscillator connection
20	I	Vss	Ground connection for Cy325
21	I	Baud_Rate	Lo = 2400, Hi = 9600, Z = adaptive
22	I	Font_Size	Lo = 8x8, Hi = 6x8 character font
23	O	Bus_Write	Cy325 wants to write data to Bus
24	O	Ready	Cy325 Ready to accept data
25	O	Bus_Ctrl	Enables Data onto Cy325-LCD data bus
26	O	LCD_enab	Enables T6963 for data transfer
27	O	LCD_cmd	Signals T6963: Cmd=hi, Data=low
28	O	LCD_reset	Drives T6963 controller reset line
29	..	reserved	
30	O	clock_out	Clock output: 1/6 crystal rate
31	I	Vdd	+5 volts
32	I/O	D7	Data bus bit 7 (MSBit)
33	I/O	D6	Data bus bit 6
34	I/O	D5	Data bus bit 5
35	I/O	D4	Data bus bit 4
36	I/O	D3	Data bus bit 3
37	I/O	D2	Data bus bit 2
38	I/O	D1	Data bus bit 1
39	I/O	D0	Data bus bit 0 (LSBit)
40	I	Vcc	+5 volts = power supply

# Cy325 Pinout Diagram

## CY325 Pin Configuration



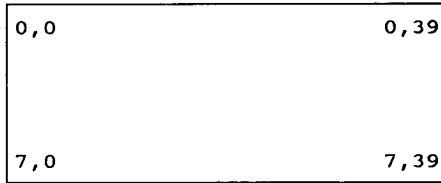
# 3

## CY325 Co-ordinate Systems

# 3

The Cy325 LCD controller uses several coordinate systems for your convenience. Pixels are plotted in one system and characters in another. These two are shown below for the 711A display:

Character coordinate system: ( row, col )

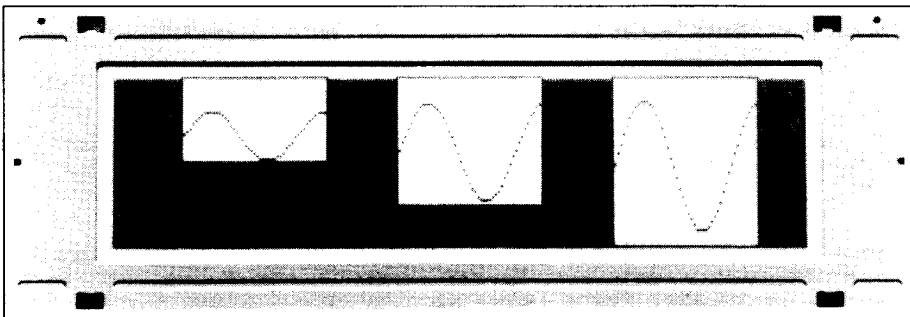


Pixel coordinate system: (x,y) = ( col, line )



711 - 63  
1091 - 127

The pixel coordinate system establishes a "sense" or "handedness" based on the origin (0,0) being located at the lower left corner. The actual location of the origin on the LCD display will depend on whether Global or Window-Relative coordinates are selected.



Example of Window relative Sine waves

# 4 CY325 Command Descriptions

4

## Cy325 Command Format

In the Command mode of operation, in which incoming characters (either serial or parallel) are interpreted, the usual command format is as follows:

A x1, x2, ..., xn <cr>

where 'A' is any ASCII character, followed by a space, and x1 through xn are relevant parameter values, separated by commas and terminated by a carriage return character.

### ASCII Parameter Arguments

The parameters are input as ASCII decimal numbers or hexadecimal numbers, although they are converted internally to binary numbers for use by the Cy325. All numbers are integers and the normal range is from 0 to 255. Examples of numbers include:

1, 12, 56, 99, 128, 33h, 3Fh, 0FAh

An 8-bit argument can be represented as either an ASCII decimal integer or an ASCII hex integer. The Cy325 uses the Intel convention for hex symbols:

'0xxH' or '0xxh'

xx is any 8 bit hex value from 00 to FF. A leading '0' is needed only if the leading hex character is non-numeric, ( 'A'..'F' ).

Examples of valid numeric arguments include:

3 3eh 3Eh 3EH 3eH 0Eh 127 45 03eh 045

Invalid numeric examples include: E3h 0E3 e3H ABh.

### Binary Command Format

If the Cy325 has been placed in the Binary command mode the format is as follows:

B N x1 x2 x3 ... xN

where B is any valid 8-bit command (same as ASCII) and where N is an 8-bit binary data count, followed by the 8-bit binary data values, x1 to xN. There is no terminator and there are no blanks or commas used in the Binary data format.

## Cy325 Command List

B x1,y1,x2,y2 <cr>	Box drawing command.
C x,y <cr>	Cursor positioning command.
F n,d1,d2,...,dn <cr>	Font download command
G n,d1,d2,...,dn <cr>	Graphics line download command
H n,y1,...,yn <cr>	Histo-graph generation
I n<cr>	Initialize options
K n<cr>	Key #n acknowledgement or control
L <cr>	Large character mode 16x12 or 16x16
M n,v <cr>	Mode command sets mode byte n to v
P x,y <cr>	Plot pixel at (x,y) on LCD.
S n <cr>	Send value of n through the Cy325.
V n <cr>	ViewPort n selected as current window
W x1,y1,x2,y2 <cr>	Window command defines character window
Z <cr>	Set "Times Square" horizontal scroll
/	Negate Command. (prefix)
+ <cr>	Save Window Status
- <cr>	Restore Window status
? n <cr>	Query LCD status register n or field
@ n data	Transmit n data bytes through Cy325
{ n data	Plot n binary data bytes in window
*	Lower Half of 128 pixel LCD (prefix)

### Cy325 Ctrl-Key Command Summary

		value
Ctrl-C	Command Mode Entry	= 03h
Ctrl-D	Display Mode Entry	= 04h
Ctrl-K	Klear current Window	= 0Bh
Ctrl-M	same as <cr>	= 0Dh
Ctrl-N	Shift Out - select special font	= 0Eh
Ctrl-O	Shift In - normal char	= 0Fh
Ctrl-W	Window swap command	= 17h

# Cy325 Ctrl-Key Command Descriptions

## Command/Display Mode Switch Commands

The non-printable command that switches the Cy325 from the Display mode (in which characters are simply displayed) to the Command mode (in which characters are interpreted, not displayed) is the Ctrl-C command (binary 03h).

**Ctrl-C**      Command Mode Entry

The Ctrl-D command switches to the Display mode from the Command mode.

**Ctrl-D**      Display Mode Entry

The Ctrl-K command applies to the "current" window ( selected via the 'V' ViewPort command or specified by the 'W' Window command.)

**Ctrl-K**      Klear current window

The Ctrl-K character (0Bh) is detected in either Command or Display mode and immediately erases the "current" character window. Graphics information is not erased, only text characters. ( For erasing graphics information see the Initialize command. )

**Ctrl-N**      Shift Out - select special font

The Ctrl-N command should be issued in Cy325 Display mode and performs the "Shift Out" function, that is, it selects the special Font and displays the `@', `A', `B', `C', characters, etc., as the first, second, third, characters, etc., of the special user defined font that the Cy325 'F' command allows.

**Ctrl-O**      Shift In - normal characters

The Ctrl-O command should also be issued in the Cy325 Display mode and Shifts back to the normal ASCII character font. The Ctrl-O command is the inverse of the Ctrl-N command. Both of these commands can be used as often as desired, that is, you can switch between normal characters and special characters in the same window as often as you like. Cursor actions and other behavior is the same for normal and special characters, only their appearance on the LCD is different.

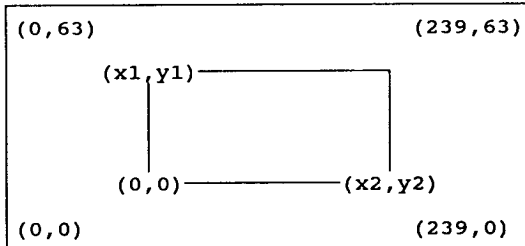
**Ctrl-W**      Window swap command

The Ctrl-W command should also be issued in the Cy325 Display mode. It allows you to swap between the "current" window and a window that has been previously "saved" (via the '+' command.) The window status is also swapped as will be described later, thus one window may be displaying large characters while the other is displaying normal characters. Ctrl-W effectively maintains two "current" windows without constant redefinition.



## The Box Command, 'B'

The 'Box' command uses four arguments to specify the upper left and lower right corners of the box, specified as points in the pixel coordinate system:



**B x1,y1,x2,y2 <cr>** = Box drawing command.

The Box command will draw a box on the LCD if the parameters  $x1, y1, x2, y2$  satisfy the geometrical relations  $\{ 240 > x2 > x1 \}$  and  $\{ y2 < y1 < Ymax \}$ , where  $Ymax = 64$  or  $128$ , depending on the particular LCD. These parameters are saved internally for use by other commands, such as the Histogram command. Although there can be many boxes showing on the screen at one time, the last box command defines the "current" box parameters that will be used by any related commands. This is true even when the box command is preceded by the negation symbol, '/', which causes the box outline to be erased, that is, '/B x1,y1,x2,y1 <cr>' erases the box outline but the current box parameters are retained internally until replaced by a newer box or window declaration.

A Box establishes a local pixel coordinate system but has no text coordinates associated with it. Windows and viewports, which also define local pixel systems, always contain an integral number of characters spaces and thus create text windows. At any time there is only one "current" box or window even though there may be previously defined boxes or windows still displayed on the LCD screen. The last **Box**, **ViewPort** or **Window** command issued determines the 'current' values of  $x1, y1, x2, y2$ .

The box coordinates (x1, y1, x2, y2) define a "graphics window" that is affected by the following commands:

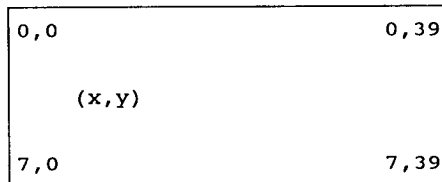
- 'B' Box
- 'I' Initialize
- 'V' Viewport
- 'W' Window
- 'M' Mode

## The Cursor Positioning Command, 'C'

The 'Cursor' command uses two arguments to specify the row and col that the cursor is to be moved to, specified in the character coordinate system. The format of the cursor positioning command is as follows:

`C row, col <cr>` Position cursor at row, column

The 711 LCD (8 rows, 40 char) character coordinate system is shown below. The top left character is located at (0,0) while the bottom right character position is (7,39), that is the bottom right character is located at row 7, col 39. (The 1091 LCD has 16 rows, 0 to 15.)



The cursor can be placed at any character location in the LCD screen, not just in the text window. However, if the text cursor is positioned outside of the text window while the auto-increment mode is set, the incremented cursor location will often lead to confusing results, since after a character is displayed, the Cy325 will attempt to advance the cursor within the current window. If the cursor is to be placed outside of the text window, it is recommended that the auto-increment mode be switched off (see 'Mode' command).

Although windows and viewports define local pixel coordinate systems, and the Cy325 manipulates the character cursor within the current window, there is no local character coordinate system associated with a window. Thus cursor coordinates used with the 'C'-command are always in global character coordinates, that is, they are LCD-relative not Window-relative.

## The Font Command, 'F'

The Cy325 allows special user defined fonts to be downloaded to the LCD. The format of the Font command is as follows:

**F n, d1,d2, .. ,d8 <cr>** Font download command

where

n = character code (0..63) being specified

d1 = top 'row' of character font

d2 = 2nd 'row' of character font

..

d8 = bottom 'row' of character font

<cr> = command terminator

Each character in the 711A-LCD is defined as a 6x8 or as an 8x8 pixel cell, depending on the "Font\_size" selection (pin 22). Each 'row' is 6 or 8 pixels wide and there are 8 rows per character. The following example illustrates a font command defining the special font associated with character code 1, that is, 'A'.

**F 1,1,3,7,15,5,2,5,3 <cr>**

Cell 'Font'	Row #	dec	hex
. . . . . x	row 1	1	1
. . . . x x	row 2	3	3
. . . x x x	row 3	7	7
. . x x x x	row 4	15	0Fh
. . . x . x	row 5	5	5
. . . . x .	row 6	2	2
. . . x . x	row 7	5	5
. . . . x x	row 8	3	3

where: x = pixel 'on' and . = pixel 'off'

After the above font command has been issued to the Cy325, and the Cy325 has been returned to the display mode, the special character font can be selected by issuing the Ctrl-N command. The character 'A' corresponds to code 1 ('B'= 2, 'C'= 3, etc.) and will cause the cell 'font' shown above to be displayed at the cursor position. The character 'A' will always invoke the same character font until either the Ctrl-O command shifts back to the normal font, the Cy325 is reset, or the Font command is used to re-define the font corresponding to character 1. The Font command accepts character codes from 0 to 63 as 'n' when character fonts are being defined. ASCII characters '@', 'A', 'B'..'Z' are used to invoke the special characters because they are easily generated in most computer languages and/or keyboards. Special characters may also be displayed while in the normal font mode by using a full 8-bit character code to access the special fonts. In this case, fonts 0..63 correspond to character codes (164..227), decimal or (0A4h..0D3h), hexadecimal.

## The Graphics Line Download Command, 'G'

Graphic lines (rows of pixels) can be sent to the Cy325 via the Graphics Line command, 'G', which has the following format:

```
G n, d1,d2, ,dn <cr> Graphics line download command
```

where

n = number of data bytes in command, n = 1..9

d1 = 1st byte of graphics data

dn = nth byte of graphics data

As in the Font command, the data bytes are sent as 8 bits, but the selection of the Font size determines how many bits (6 or 8) are actually mapped into the graphics line. The graphics line is created by concatenating the horizontal groups of pixels to form a horizontal graphics line that is a multiple of 6 (or 8) pixel segments. For example (using 6x8 font):

```
      G 3, 3, 5, 8 <cr> will concatenate the segments
      /      |      \
....XX  ...X.X  ..X...
```

to draw the graphics line:

```
....XX...X.X...X...
```

If the 'Row\_Auto\_Inc' mode bit is ON the row is incremented after the graphics line is drawn. The graphics column number is always untouched, therefore the next Graphics Line command will define a line directly beneath the current line, and this will continue as long as such commands are issued. In this way a graphics picture can be painted 'line by line' proceeding from the top of the picture toward the bottom. The following example assumes that the Row\_Auto\_Inc mode bit is ON:

```
G 3, 3, 5, 8 <cr>          series of four
G 3, 3, 4, 8 <cr>          Graphics Line
G 3, 3, 8, 5 <cr>          Commands...
G 3, 3, 7, 3 <cr>
```

gRow n	....XX...X.X...X...	Four graphics lines are drawn on the LCD screen as a result of the above commands.
gRow n+1	....XX...x....x...	
gRow n+2	....XX..x.....x.x	
gRow n+3	....XX...xxx....xx	

The 'G' command with n = 0: **G 0, gRow, gCol <cr>** specifies the graphics row and column (gRow,gCol) cursor for starting the next graphics line. The initial values of gRow and gCol are undefined, therefore this command must be used to place the first row at the desired location.

## The Histogram Command, 'H'

The Histogram command will accept a series of values and draw a histogram, or bar-chart, in a specified window or box. The Cy325 will compute the width of each bar in the histogram in order to evenly space the display, however the values must be pre-scaled in the vertical direction before being presented to the Cy325. It is necessary to define the window or box before using the 'H' command. The 'B', 'V', or 'W' commands define the left, top, right and bottom sides of the window in which the histogram will appear. The Cy325 uses the distance (in pixels) between the left and right sides to compute the width of each bar. The user must use the distance (in pixels) between the top and bottom to prescale the values presented to the Cy325.

Histogram command Format:

```
H n, y1, y2, .., yn <cr>
```

where

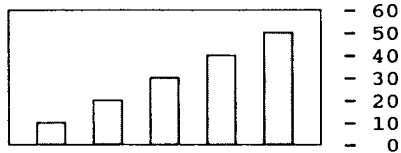
- n = number of bars to be displayed n = 1..9
- y1 = height of 1st bar
- ..
- yn = height of nth bar in display

Histogram Example:

First, define the box via the Box command, B 80, 62, 230, 2 <cr>. This command defines a box 150 pixels wide (230-80) and 60 pixels high (62-2). The Cy325 will use the value 150 for automatic horizontal scaling, and the user should use the value 60 to prescale the values presented in the histogram command. For example, if the largest bar value is 250, then all bars should be scaled by 60/250, that is, the box height divided by the largest data value. Assume that after the above calculations are made, the histogram command is issued as follows:

```
H 5,10,20,30,40,50 <cr>
```

This commands the Cy325 to display 5 bars in the pre-defined box with values 10,20,30,40,50.



When the '/' prefix precedes the Histogram command, the specified bars are erased instead of drawn. The bars will be erased only up to the height specified in this command. To erase the entire window use the 'I 2' command (see Initialize commands).

## The Initialize Command, 'I'

The 'I' command performs initialization functions, specified by a parameter. Initialize commands are used for infrequent operations that don't merit their own command letter. 'I' is also used to set special mode bits that are not included in the normal mode registers because the consequences of an incorrect setting are inconvenient to deal with, such as switching from ASCII to Binary operation by mistake.

### Cy325 Initialize Commands

I 0<cr>	= Initialize (reset Cy325) & clear screen
I 1<cr>	= erase the characters in current window
I 2<cr>	= erase the graphics information in window or box
I 3<cr>	= erase the entire screen (affects spcl font)
I 4<cr>	= restore Cy325 defined default font
I 5<cr>	= enable Key-Scan operation (mode dependent)
I 6<cr>	= select 6x8 font size (pulls pin 22 high)
I 7<cr>	= ASCII / Binary switch (default ASCII)
I 8,n<cr>	= output 8-bit value of N to pins 1..8
I 9<cr>	= select Fast_Bus operation (default off)
I 10,n<cr>	= send value 'n' out Serial port
I 11,n<cr>	= send value 'n' out Parallel port
I 12,n<cr>	= display value 'n' as Logic Waveform

The 'I 0' command is treated as a hardware reset, that is, the Cy325 will 'power up' in the same manner as when the hardware reset line is pulsed high. Since the powerup state is the Display mode, the 'I 0' command will leave the Cy325 in the display mode.

The 'I 1' command clears the character window (defined via 'V' or 'W'). Characters in the window are erased, however graphics and characters outside of the window are unaffected.

The 'I 2' command uses the current value of x1,y1,x2,y2 to define the graphics box or window to be erased. Any text information in this window is unchanged. When the inversion prefix, '/', is used with 'I 2' the erase is "inverted", that is, the current graphics window is "filled" with ON or dark pixels. Note that these pixels do not interfere with any text in the window, it simply "covers" the text until removed (for a "curtain raising" effect).

The 'I 3' command erases all LCD RAM, including the special font storage area. All information, both text and graphics, is erased from the LCD. If a user-defined font is in use, then the user must download the special font again after 'I 3' is issued.

The 'I 4' command reloads LCD RAM with the Cy325 default special font. It is employed following 'I 3' command or when it is desired to return to the default (built-in) special font.

The 'I 5' command enables scanning on pins 1..6 by "pulling" pin 13 low from "inside" the Cy325. This pin can be pulled low from outside the Cy325, achieving the same effect. If the pin is free, then I 5<cr> pulls it low and /I 5<cr> returns it high, enabling and disabling the Key-Scan operation (see also mode register #2).

The 'I 6' command selects the 6x8 character font by pulling pin 22 high from "inside". The same can be achieved by pulling pin 22 high from outside. The AND 711A LCD pin 19 selects either 6x8 or 8x8 font, thus Cy325 pin 22 can be tied to 711A LCD pin 19 to control the font size. These two pins can be tied together and driven externally or controlled by the 'I 6' and '/I 6' commands. 'I 6' selects 6x8 font and '/I 6' selects 8x8 pixel font size.

The 'I 7' command selects the binary mode operation of the Cy325 while the '/I 7' command returns the Cy325 to its default ASCII mode of operation. '/I 7' must be issued in Binary: 2F 49 01 07.

The 'I 8,n' command requires parameter 'n' whose value is output to pins 1..8 on the Cy325. If these pins are not being used in one of the Key-Scan modes of operation, then they can be used as general purpose control lines for any user defined purpose. The '/' prefix has no meaning for this command.

The 'I 9' command sets the "Fast\_Bus" mode of operation. Normally this mode is insignificant, however bus transfers will complete in about half the time in "Fast\_Bus" mode. Note that for slow systems such as dICE-51 or systems driven by a BASIC language interpreter, it is usually necessary to leave the Cy325 in its default "Slow\_Bus" mode. The '/I 9' command restores the "Slow\_Bus" mode of operation used by systems which cannot respond to the Bus\_Control pin within the approximately 15 microseconds required by the Fast\_Bus mode of operation.

The 'I 10,n' command outputs the 8-bit value of the parameter 'n' to the Serial port as 8 bits, no parity, one stop bit, using the Baud rate in effect at the time the command is issued.

The 'I 11,n' command outputs the 8-bit value of the parameter 'n' to the Parallel port using the handshake protocol specified in a later section. I 10 and I 11 are independent of the port the command is received on, unlike other transfers through the Cy325.

The 'I 12,n' command uses binary value n to create logic waves in the current window. This command does not require Key scanning to be enabled, unlike the other means of generating logic waves.

## The Key Command, 'K'

The Key command, 'K', with parameter, provides an acknowledgement from the host to the Cy325. Its use is explained in the section on 'Soft-keys'.

The Key command has the format:

**K n<CR>**




where 'n' is the number of the key that was hit.

When the 'K n' command is preceded by the '/' prefix the effect is to drive pin 'n' low, (where n = 1..7). If Scan mode is enabled this has the same effect as a key activating line n. When Scan mode is disabled the '/K n' command merely pulls pin 'n' low with no other effect. In this case the 'K n' command will return pin 'n' high. Thus when the Scan mode of operation is disabled, the 'K' command and its inverse can be used to provide seven I/O lines that can be set and cleared by command, for what ever purpose the user desires. ( See the Initialize command, 'I 8,n', for additional programmed I/O.)

## The Large Command, 'L'

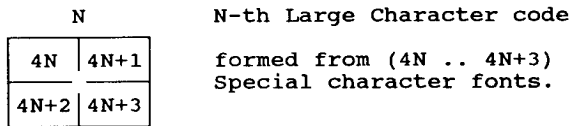
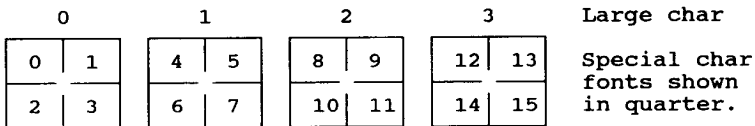
The Large command sets the mode bit in the current window that selects the Large Character Display mode. The characters displayed in this window are either 12x16 pixels or 16x16 pixels depending on whether the LCD is supporting 6x8 or 8x8 characters. The large characters are effectively created from four special characters, using the special character font. This allows either the use of larger alphabetic characters for greater visibility, and also allows special characters such as Chinese or Japanese to be defined and displayed. The larger characters advance the cursor and wrap within the current window in the same manner that normal characters behave, however only 16 large characters can be defined at one time.

No large characters are "built-in" to the Cy325. The following example will show how to create the large characters shown below. These large characters can be invoked by either ASCII decimal or ASCII alphabetic codes.

- ASCII - code	Character description	Symbol shown
'0'	'@'	Checkerboard 
'1'	'A'	Circle 
'2'	'B'	'3-D'-Box 



Special font characters (created via the 'F' command) are used to construct Large characters, with four special font characters per large character. The first special character forms the top left quarter of the large character, the second forms the top right quarter, the third forms the lower left quarter and the last provides the lower right quarter. Then the next special font character forms the top left quarter of the next large character, and so on. The first four large characters are shown below in terms of their constituent special character fonts:



The large characters shown on the previous page can be created via the following sequence of Font commands:

```

;--- checker board squares (4 quadrants)

F 0,055h,0AAh,055h,0AAh,055h,0AAh,055h,0AAh <cr>
F 1,055h,0AAh,055h,0AAh,055h,0AAh,055h,0AAh <cr>
F 2,055h,0AAh,055h,0AAh,055h,0AAh,055h,0AAh <cr>
F 3,055h,0AAh,055h,0AAh,055h,0AAh,055h,0AAh <cr>

;--- 4 quadrants of circle / 4 character circle ---

F 4,0,0,0,3,4,8,16,16 <cr>           ; top left curve
F 5,0,0,0,0F0h,8,4,2,2 <cr>         ; top right curve
F 6,16,16,8,4,3,0,0,0 <cr>          ; bot left curve
F 7,2,2,4,8,0F0h,0,0,0 <cr>         ; bot right curve

;--- 4 quadrants of "3-D box"

F 8,0,1Fh,18h,17h,14h,14h,14h,14h <cr> ; top left
F 9,0,0F8h,4,0FEh,2,2,2,2 <cr>         ; top right
F 10,14h,14h,14h,14h,14h,0Ch,7,0 <cr>   ; bot left
F 11,2,2,2,2,2,2,0FEh,0 <cr>           ; bot right

;--- nested size boxes

F 12,0,0,0,0Ch,0Ch,0,0,0 <cr>           ; little box #0
F 13,0,0,0Eh,0Ah,0Ah,0Eh,0,0 <cr>      ; little box #1
F 14,0,1Eh,12h,12h,12h,12h,1Eh,0 <cr>  ; little box #2
F 15,3Fh,21h,21h,21h,21h,21h,21h,3Fh<cr>; little box #3

```

## The Mode Command, 'M'

The Mode command can be used to set various bits in the Cy325 mode registers ( described in another section ). The command has two arguments. The first argument is the address of the mode register, and the second argument is the value to be written into the selected mode register.

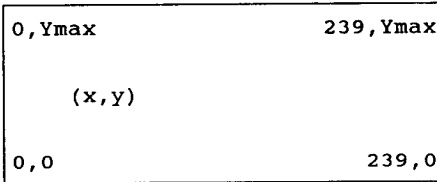
**M n,v <cr>** Mode command: set mode byte n to v

The mode command sets all 8 bits at once.

## The Plot Command, 'P'

The Plot command operates in pixel space and specifies the pixel coordinates of the point to be plotted. The negation of the plot command, obtained by prefixing the '/' symbol, erases the pixel at the specified coordinates.

**P x,y <cr>** Plot pixel at (x,y) on LCD.  
**/P x,y <cr>** Erase pixel at (x,y) on LCD.



Ymax = 63 for 711 LCD

Ymax = 127 for 1091 LCD

### "Global" and "Local" Pixel Plotting

The type of coordinate system used to plot pixels is qualified by mode bits that specify either "global" or "local" coordinates.

**Global** coordinates (shown above) are based on the entire LCD screen with the origin (0,0) at the lower left of the screen. In global mode pixel plotting, the locations of the windows have no effect on a particular plot command. Since negative coordinate parameters would be "below" or "to the left of" the LCD screen they are not allowed.

**Local** coordinates use the lower left corner of the "current" window or viewport as the (0,0) origin for pixel plotting. Local pixels will appear on the screen relative to this location. Since negative parameters would automatically be located "outside" of the window, they are not allowed, however even positive values can cause the pixel to be plotted "outside" of the current window. This can be prevented by setting the **Clipping** mode bit. This bit 'clips' pixels that fall outside the current window and prevents them from being plotted.

For more information relevant to pixel plotting, see the **Box**, **Viewport**, **Window**, **Mode** and **Plot-String** commands.

Plot commands handle only one pixel per command. For multiple pixel plotting such as required to graph the output of an Analog-to-Digital converter the Plot-String command should be used.

## The Plot-String Command '{'

The Plot command, 'P x,y<cr>', described previously plots a single pixel located at coordinates (x,y). Often it is desirable to plot a string of points in a curve. If the x-value simply increments by one and the y-value contains the variable information, then the 'P' command is needlessly verbose. For example, assume that our system is reading A-to-D converter values sequentially and we wish to display these in a window, where the x-axis represents the time axis. The Plot-String command, invoked by '{', is designed to handle this case. Since the A-to-D values are always binary, it makes no sense to convert to hex or decimal, so the arguments of the Plot-String command will be binary. The format of the Plot-String command is as shown below:

```
'{ n y1 y2 y3... yn
```

where '{' is the command character ('{' = 7Bh) and n is the number of points to plot. The n values y1 .. yn are the 8-bit values of the y-ordinate that will be plotted sequentially in the current window. The x-ordinate corresponding to y1 is the x-value of the left side of the current window plus one. The x-value is incremented with each pixel plotted, so that the pixels are plotted from left to right in the current window (or box). This allows you to read voltages or other 8-bit values and display them in sequential fashion with almost no effort involved. Note that the Plot-String command uses the Global/Local and Clipping flags set for the current window to determine the exact plotting behavior to be employed. You should always be aware that Clipping will prevent any values that fall outside of the window from showing up, and therefore this option, while producing neat looking displays, may actually cause the visual loss of some data. Clipping can be turned off or on by setting the appropriate mode bit (see Mode Register description).

Note: Although the ASCII '{' character is used as the Plot-string opcode, the arguments are entered as binary numbers, not as ASCII decimal arguments. The Cy325 temporarily switches to a mode in which it expects binary arguments, plots each pixel with the specified y-coordinate, and then resumes ASCII command operation after all binary data points have been entered. For example the plot-string command that will plot five points with y values:

```
5,10,15,20,25
```

would be issued to the Cy325 as the 8-bit codes:

```
7B 05 05 0A 0F 14 19
 /  |
 '{ 'n' data points
```

# 7 CY325 Viewports and Windows 7

## The Viewport Command, 'V'

The Viewport command uses a single argument to select one of the default 'viewports' or 'windows' that are "built-in" to the Cy325. This is equivalent to issuing the Window command with a set of default parameters characteristic of the particular window.

The 'V' command defines a new current window. The current window defines the "active" coordinates, both graphic and text-based.

Graphics:        x1, y1, x2, y2  
Text:            top, left, bot, right

These remain current until a **Box**, **Window**, or another **Viewport** command is issued. A **Box** command changes the Graphics window pixel coordinates without altering the Text window character coordinates (top,left,bot,right).

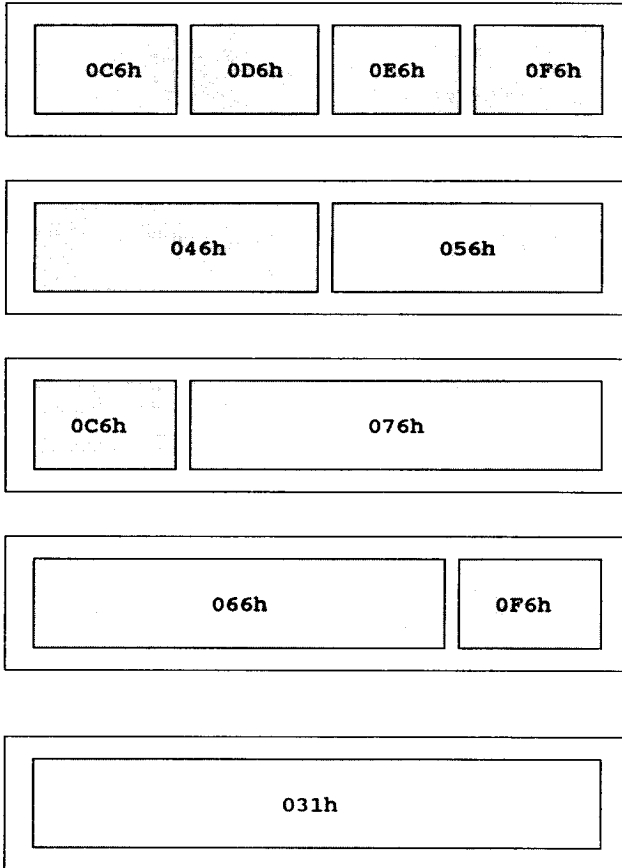
If the default windows provided in the Cy325 do not meet your needs, then the **Window** command defined below will allow you to specify any window by supplying an appropriate set of parameters.

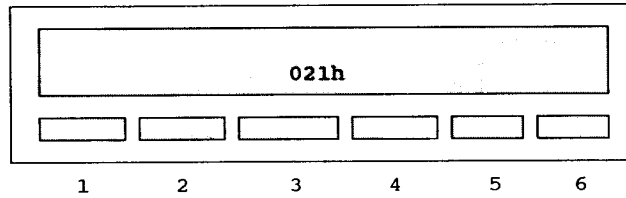
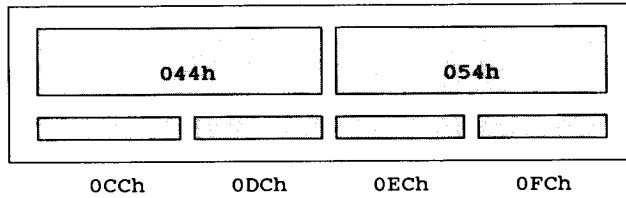
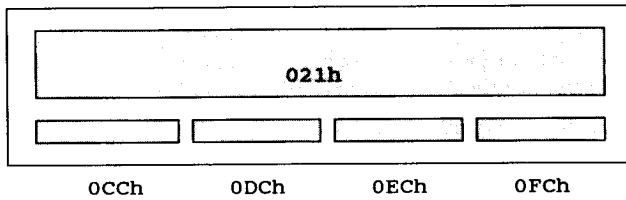
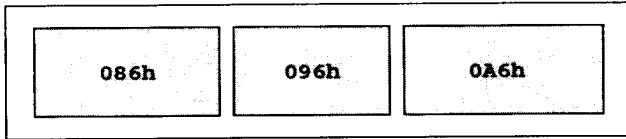
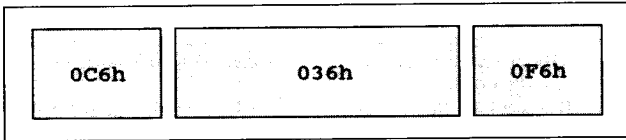
There is one major difference between the use of the Viewport command and the use of the Windows command. The Viewport command will select one of a number of default windows that are described in a following section. These windows possess a viewport number and also possess fixed corner locations that are uniquely identified by this number. Thus to save the corner locations of a particular window, it is necessary only to save the viewport number that identifies the viewport. User defined windows are not identified by any such window number. As described later in this manual, it is possible to "save" and "restore" viewports and also to "swap" viewports. These actions allow you to leave the "current" window and switch to a new "current" window, perform some action, then return to the saved window without "losing your place", that is, without losing the cursor location and also without forgetting other status information relevant to the window, such as whether or not large characters or a special font is being used. Since the saved window is identified only by its number, it is impossible to save user defined windows. For this reason, you should attempt to use windows from the default set if possible, to avoid giving up the very useful save, restore, and swap functions.

## Default Windows

The Cy325 provides a set of about 250 default windows that may be selected by the Viewport command, 'V n'. The argument 'n' may be specified as a decimal or a hexadecimal value but is maintained internally as an 8-bit value that is decoded to generate the particular viewport.

Typical examples of default windows are shown below:





## Cy325 Built-in Viewports

The Cy325 provides over 250 built-in viewports that can be selected via the 'V n' command where 'V' is the viewport command and 'n' is the viewport number. The default viewport is viewport number zero, which covers the entire screen of a 711 or 711A LCD. Viewport zero contains 8 rows of 40 characters (in 6x8 font mode) and contains 64x240 pixels. The (0,0) character location is in the upper left corner while the (0,0) pixel coordinate is located at the bottom left corner of the screen.

All other viewports are subsets of viewport zero, that is, they contain fewer than 8x40 characters or 64x240 pixels. The number that describes a given viewport is coded in such a manner that its horizontal and vertical coordinates are implied by the code. One might ask why a coded viewport should be used instead of simply declaring a window via the 'W' command with the top left and bottom right character coordinates supplied as arguments. The answer lies in the Cy325s ability to 'save' the current window status on a 'stack' while switching to a new current window, then to 'restore' the saved window status. The status consists of window number, character cursor, window coordinates (both character and pixel) and several mode bits that qualify the window. (See the description of the '+', '-', and 'Ctrl-W' commands for more information on saving and restoring windows and the section on Cy325 mode bits for relevant info.) Thus the primary reason to use the built-in viewports instead of simply defining windows as they are needed is to allow operations in two windows without having to maintain window status for each window. Window status can be saved internally only for numbered windows. We now proceed to discuss the window numbering system.

### The Viewport Numbering System

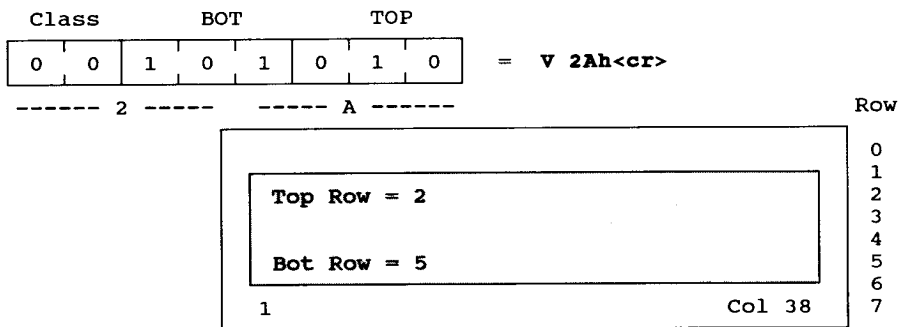
The window number is maintained internally as an 8-bit number that consists of two or more bit fields. The two most significant bits can be thought of as the "class" of the window. The window class generally relates to the number of windows of the class that can span the LCD horizontally. The relationship is such that (n+1) class n windows will span the LCD screen horizontally. For example three class 2 windows will span the screen or four class 3 windows, or one class zero window.



The coding of classes 1, 2, and 3 is different from class 0 coding, which we will describe first.

### Class Zero Viewport Coding Scheme

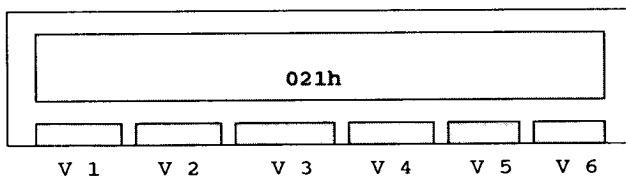
Class zero windows extend from column one to column 38, that is, they span the entire horizontal screen of the 711 LCD. The window frame extends into column zero and column 39 therefore these columns are not available for characters. The class bits are both zero for such windows. The remaining six bits in the viewport number are divided into two 3-bit fields, the TOP field and the BOT field, which are used to specify the vertical extent of the window. Each 3-bit number specifies the appropriate character row that defines the top or bottom of the window. An example is shown below.



There are two special cases of class zero windows. The case in which the bottom is above, or less than, the top is discussed below.

### Class Zero with TOP > BOT

The general case in which the TOP value exceeds the BOT value is illegal although these combinations may have meaning in a future controller. The special case in which BOT = 0 and TOP > 0 defines six 'key' windows as shown below:

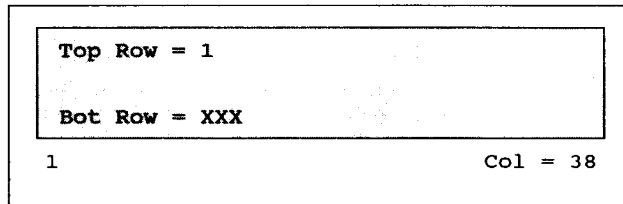


These 'key' windows may be used for any purpose but are especially well suited for use with the Cy325 'soft keys' described elsewhere in this manual.

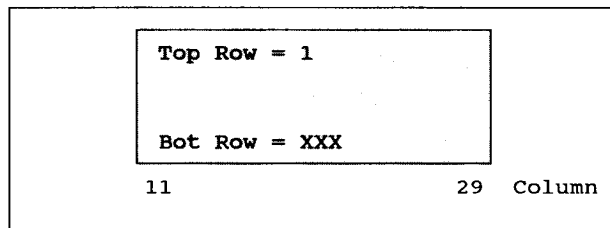
## Class Zero Viewport with TOP=BOT

The second special case is that in which the TOP and BOT fields have the same value. For this case the TOP becomes one and the BOT has the specified value. How does this differ from the class zero window with TOP = 1 and BOT = the value? It differs as described below. Consider the class zero window with TOP = 1 and BOT = x. This window will span the LCD from col 1 to col 38 and will vertically range from row 1 to row x. What is different when TOP = x and BOT = x? The difference is in the horizontal span. In this very special case the window is only one half the screen wide but is centered in the LCD as shown below:

Class		BOT			TOP		
0	0	X	X	X	0	0	1



Class		BOT			TOP		
0	0	X	X	X	X	X	X



Note that there is room on either side of the class zero window for higher class windows that fit in only one quarter of the LCD screen.

Coding for 128 pixel deep displays: The Top=Bot case uses Row 1 for the top row and 2\*Bot as the bottom row, when the LCD display is 128 pixels deep.

## Viewport Classes 1, 2, and 3

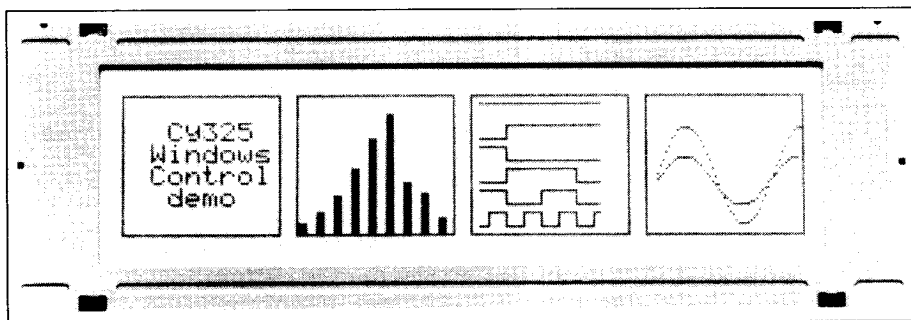
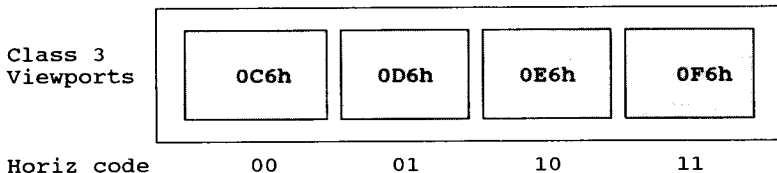
The class zero coding scheme described above has the following fields:

Class		BOT			TOP		
0	0	B	O	T	T	O	P

All other Cy325 window classes have the following coding:

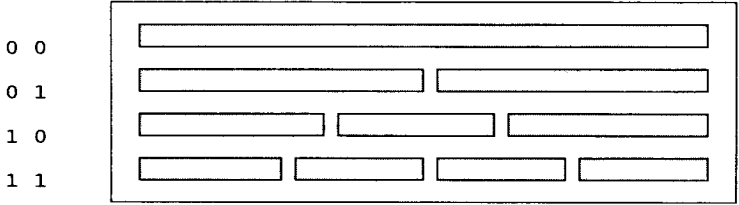
Class		Horiz		Vertical			
x	x	h	h	v	e	r	t

As described previously, the class number generally relates to how many windows it takes to span the LCD screen. For example class three windows allow four viewports to coexist horizontally across a screen as shown below.



From the above example it can be seen that the horizontal coding defines the horizontal position for those classes in which multiple windows span the screen. Thus the 4-bit field consisting of the class field concatenated to the horiz field uniquely specifies the horizontal coordinates of the window. A summary diagram is shown below:

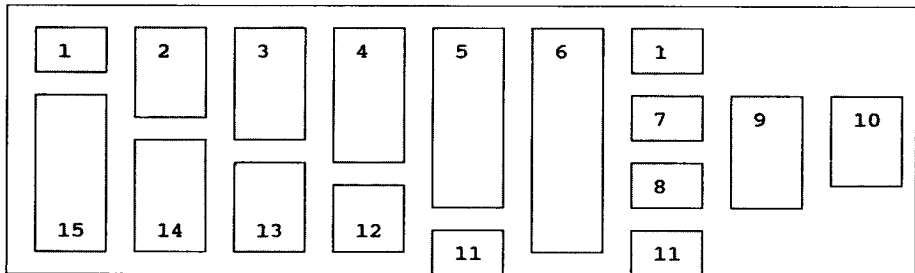
Class



The most significant four bits of the viewport number uniquely define the horizontal extent of a window for classes 1, 2, and 3.

### Vertical Coding for Viewport Classes 1, 2, and 3

As explained above, the high four bits (class + horiz) of the viewport number define the horizontal position and extent of the viewport. The low four bits comprise the vertical field and uniquely determine the vertical extent of the window, as shown in graphical form and in tabular form below.



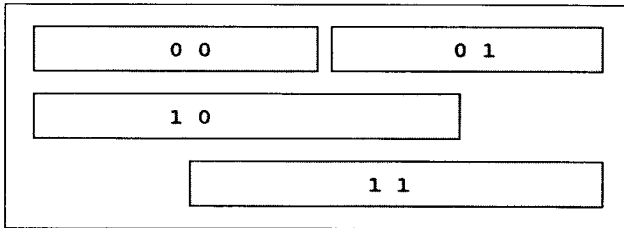
## Vertical Coding Table

Vert Field	TOP row	BOT row
0000	0	7
0001	1	1
0010	1	2
0011	1	3
0100	1	4
0101	1	5
0110	1	6
0111	3	3
1000	5	5
1001	3	5
1010	3	4
1011	7	7
1100	6	6
1101	5	6
1110	4	6
1111	3	6

### Special Class 1 Horiz Coding

Class one viewports allow a max of two windows horizontally per screen, however there are four possible positions implied by the two bit Horiz field. These bits are decoded in a special way as indicated in the following figure.

Class		Horiz		Vertical			
0	1	x	x	y	y	y	y

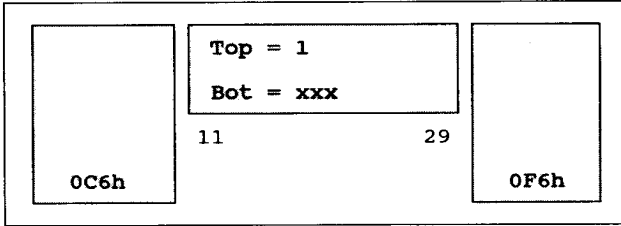


By properly specifying viewport numbers according to the 8-bit coding scheme described above, you can generate over 250 unique windows, with literally tens of thousands of non-overlapping windows on a 711 LCD screen. Since windows can be overlapped, you can generate hundreds of thousands of window combinations and save and restore any current window while you work in another.

# Examples of Cy325 'built-in' Viewports

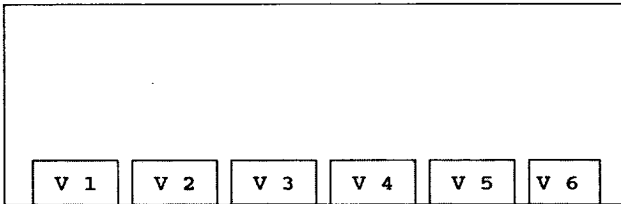
## Class 0 Examples:

Class		Horiz			Vertical		
0	0	B	O	T	T	O	P



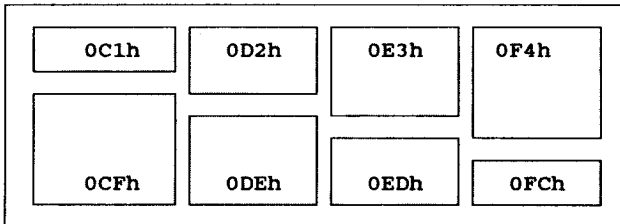
Bot=Top	'V'
001 001	9
010 010	12h
011 011	1Bh
100 100	24h
101 101	2Dh
110 110	36h
111 111	3Fh

## Class 0 "Key" Windows 1 .. 6:



## Class 3 Example:

Class		Horiz		Vertical			
1	1	x	x	V	E	R	T



## The '\*' Prefix for Viewports on (Nx128) - Pixel LCDs

"Built-in" or default windows are based on ( N x 64 )-pixel LCDs. The Cy325 also works with those ( N x 128 )-pixel LCDs such as the AND 1091 display in which the display screen is logically organized as an "upper" half and a lower "half". On power up the upper half is selected for Viewport commands. When the Nx64 pin of the Cy325 is pulled low then the '\*' prefix in front of the 'V n' command causes the viewport to be created in the lower half of the display screen. The appearance of the window will be identical to the same window in the upper half screen but displaced downward by half a screen.

The '\*' applies only to the 'V' command. If the 'W' command is used to create windows then the window may take any valid location and size on the LCD. In particular the Window command 'W' can create windows that are located partly in the upper half of the screen and partly in the lower half.

**Reserved for Photo**

## The Window Command, 'W'

The 'Window' command uses four arguments to specify size of the text window, specified in the character coordinate system. The parameters are separated by commas and the command is terminated by the carriage return. The general format of the Window command is as follows:

```
W x1,y1,x2,y2 <cr>
```

where

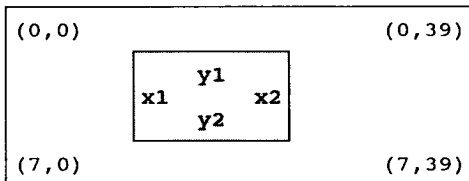
```
x1 = left char col  
y1 = top char row  
x2 = right char col  
y2 = bottom char row
```

that is,

```
W left, top, right, bottom <cr>
```

The window coordinates are inclusive, that is, characters will normally be written into the left, top, right, and bottom row/col as defined by the parameters.

Character coordinate system: ( row, col )



Note: Bottom Row:

```
711 LCD = (7,0)  
1091 LCD = (15,0)
```

The cursor auto-increment mode and the scroll/wrap modes always use the currently specified window coordinates.

The Window Status Save and Restore commands described below use a viewport parameter, View#, to identify the "current" window. This parameter is not included in the Window command, 'W' therefore user specified windows cannot be saved and restored via the Cy325 '+' and '-' commands. This must be accomplished external to the Cy325 (that is, in the host or driver code) if so desired.



## Window Commands and Characteristics

Any window defined by the W-command will be affected by the following commands:

'V n' = define current window to be default window #n.  
'I n' = fill or erase current window.  
'L' = Large characters displayed in current window.  
'Z' = Horizontal scroll ("Times Square") in window.  
'B x1,y1,x2,y2' = define graphics window  
( decouple from text window )  
Ctrl-K = Klear or erase "current" window.  
Ctrl-N = Shift-Out (select special font in window)  
Ctrl-O = Shift-In (select normal font in window)  
'/V n' = create Viewport #n but erase frame around it.  
'\*V n' = create Viewport in lower half screen,  
( '\*' works only with Nx128-pixel LCDs )

The following commands do NOT work on windows defined by 'W':

'+' = Save current window status.  
'-' = Restore current window status.  
Ctrl-W = Swap the current window with the saved one.  
'\*' = Select lower half screen (128 deep LCDs)

## Graphics and Text Windows

The current window defined by the Viewport command consists of a graphics window superimposed over a text window. Most Cy325 commands operate on either the contents of the graphics window or the contents of the text window in orthogonal fashion, that is, text operations do not usually affect graphics pixels and graphic operations are generally independent of text in a window.

The Box command redefines the "current graphics window" without altering the (row, column) coordinates (left,top,right,bot) that define the "current text window". The Box command can be used to "decouple" the current graphics and text windows so that they no longer are superimposed but can have any relation on the screen including non- and partially-superimposed. The Box command only supports graphics, not text.

## The Window Save and Restore Commands: '+' and '-'

The Cy325 LCD Controller allows the use of multiple windows. In many applications it may be desirable to leave the "current" window and create a new window, or update information in another window, then resume operations in the "current" window. The Save, '+' and Restore, '-' commands accomplish this by saving the View# of the current viewport (as defined by the 'V' command) as well as the current cursor location and the current Window status. The Cy325 will allow other windows to be invoked and used in a variety of ways, including the use of the following:

- 'V' - select another viewport to be "current"
- 'W' - specify a new "user-defined" current window
- 'B' - create a new graphics "Box"
- 'I' - erase current window, etc.
- 'Ctrl-K' - erase current window. (issued in Display mode)
- 'Ctrl-W' - Window swap - used after Save '+' command to swap the current window status with the saved window status. The current window is saved and the previously saved window will become the "current" window. This command is issued in Display mode, and can simply be embedded in a string of characters that are sent to the Cy325. The swapped windows can have different modes selected, thus the appearance of the characters may change. Note also that local cursors are swapped so that one does not "lose his place" when going from one window to another.

After the above commands or combinations of commands have been issued and the user wishes to return to the "original" current window, the '-' command is issued, followed by a <cr>. This will set the current window equal to the saved View# and restore the cursor to the place it was located when the '+' command was invoked. Note that this is all the '-' command does. In particular, it does not restore any text, so that if the original text has been erased or altered, the '-' command will not restore it.

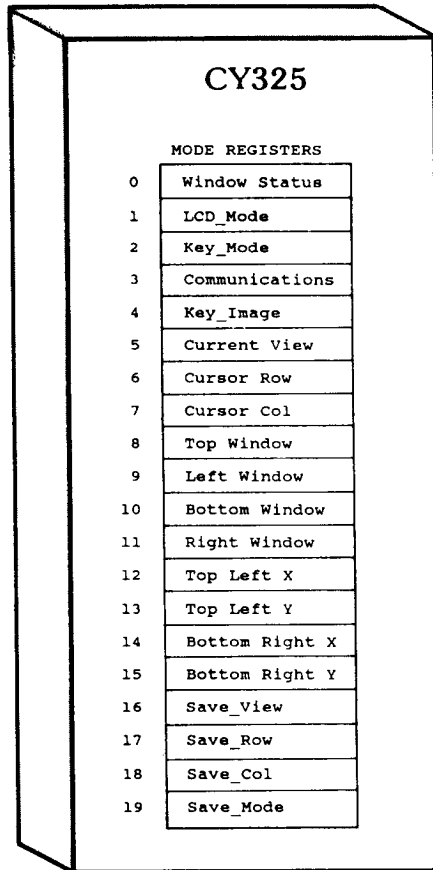
## The 'Negate' or Slash Command, '/'

The slash '/' in front of command will negate the command, for example:

- /B erases the Box drawn by the B command.
- /H n,.. erase histogram bars.
- /K n pulls pin 'n' low, where  $n = (1..7)$ .
- /L negates the Large character mode, restoring the normal character mode to the current window.
- /P x,y erases the pixel set by P x,y <cr>.
- /V n erases the outline of the specified viewport.
- /Z disable Horizontal scroll mode in current window.
- /I 2 fills (negation of erase) graphics window.
- /I 5 disables the Key-Scan operation.
- /I 9 disables the Fast\_Bus operation.

### Cy325 Mode Registers and Mode Bits

The Cy325 modes of operation are specified and selected by the values of mode bits in a set of mode registers. The contents of the mode register can be read or written as described below. The set of mode registers is shown below:



MODE REGISTERS	
0	Window Status
1	LCD_Mode
2	Key_Mode
3	Communications
4	Key_Image
5	Current View
6	Cursor Row
7	Cursor Col
8	Top Window
9	Left Window
10	Bottom Window
11	Right Window
12	Top Left X
13	Top Left Y
14	Bottom Right X
15	Bottom Right Y
16	Save_View
17	Save_Row
18	Save_Col
19	Save_Mode

# Cy325 Mode Register Description

The Cy325 uses the following special internal mode registers to determine its operating mode.

## Primary Status Registers

Mode Register #0	Window Status register
Mode Register #1	LCD Mode register
Mode Register #2	Key-Mode register
Mode Register #3	Communication modes
Mode Register #4	Key_Image

## Current Window number

Mode Register #5	current window (viewport)
------------------	---------------------------

## Character Cursor Position

Mode Register #6	character ROW cursor position
Mode Register #7	character COL cursor position

## Window Coordinates

Mode Register #8	top side of window
Mode Register #9	left side of window
Mode Register #10	bottom side of window
Mode Register #11	right side of window

## Box Coordinates

Mode Register #12	x1	top left x
Mode Register #13	y1	top left y
Mode Register #14	x2	bot right x
Mode Register #15	y2	bot right y

## Window Status Save registers

Mode Register #16	save_view	( Reg #5 saved )
Mode Register #17	save_cROW	( Reg #6 saved )
Mode Register #18	save_ccOL	( Reg #7 saved )
Mode Register #19	save_mode	( Reg #0 saved )

## Mode Commands:

The Cy325 mode registers can be written to using the Cy325 Mode Command, 'M'. The mode command requires at least two arguments; the register address and the register value. Examples of the mode command are:

- M 0,17h <cr>** - Enable text scroll, auto-increment, vertical bars, and local-clip-mode.
  
- M 1,7 <cr>** - Enable blinking cursor, and enable the text plane only. Disable the graphics plane and all of the auto-increment functions. One-line cursor.
  
- M 2,2 <cr>** - select Logic Waveform mode and de-select Soft-key operation. Note that pin 13 must be pulled low externally or via 'I 5<cr>' in order to enable scanning.
  
- M 0,37h <cr>** - select window scrolling (instead of wrapping) and auto-increment the cursor after each character is written to the window. Select the one-line cursor (instead of 8) and select Local (instead of Global) pixel plotting, with Clipping = ON. Finally, select Large characters.

## The Mode Query Command, '?'

The 8-bit contents of mode register n can be read via the Query command, '? n <cr>'. The response to the Query command will be question mark character followed by the Hexadecimal value in the specific mode register, followed by the lower case 'h', terminated by a <cr> character. Thus, assume that Mode register 0 has the three least significant bits on and all other bits off. The Query command will be issued as:

```
? 0<cr>
```

The response to this query will be:

```
?07h
```

Note that the Query response is output to the channel that received the Query, that is, if a Query is received on the serial channel then the query response is output to the serial channel, while a Query received on the Bus will output to the bus.

# Detailed Mode Register Bit Descriptions

## Mode Register #0 - Window Status register

M0.0	- scroll_flag	; scroll = 1, wrap = 0
M0.1	- auto_inc	; inc char cursor after write
M0.2	- vert_bars	; else horizontal strips (histo)
M0.3	- global	; Global=1, Local=0 pixel plotting
M0.4	- clip_flag	; clip pixels outside local window
M0.5	- large_char	; 2x2 cell char (12x16-pixels)
M0.6	- slide_flag	; for "Times Square" scrolling
M0.7	- special_char	; spcl font: 'A' -> 1, 'B' -> 2

## Mode Register #1 - LCD Mode register

M1.0	- blinking	; enable cursor blink attribute
M1.1	- cursor_enable	; enable cursor display mode
M1.2	- text_display	; enable text display plane
M1.3	- graphics_on	; enable graphics display plane
M1.4	- row_auto_inc	; increment graphics row after load
M1.5	- Qry_auto_inc	; Query auto-increment flag
M1.6	- skip_home_flag	; (normally off)
M1.7	- full-cursor	; 8 line if set, else one line

## Mode Register #2 - Key mode flags

M2.0	- soft_keys	; soft-keys [1..6] = pins 1..6
M2.1	- Logic_waves	; pins 1..6 = digital logic [1/0]
M2.2	- Cursor_keys	; pins 1..4 = up-down-left-right
M2.3	- ASCII_keys	; pins 1..7 = ASCII key inputs
M2.4	- Key_matrix	; pins 1..8 = 4x4 matrix (16 keys)

## Mode Register #3 - Communications flags

M3.0	- send_to_Bus	; qualify by input flag
M3.1	- send_to_TxD	; qualify by input flag
M3.2	- Key_to_TxD	; send keys to TxD
M3.3	- Key_to_BUS	; send keys to BUS
M3.4	- echo_serial	; echo RxD-Display to TxD
M3.5 .. M3.7	- reserved	

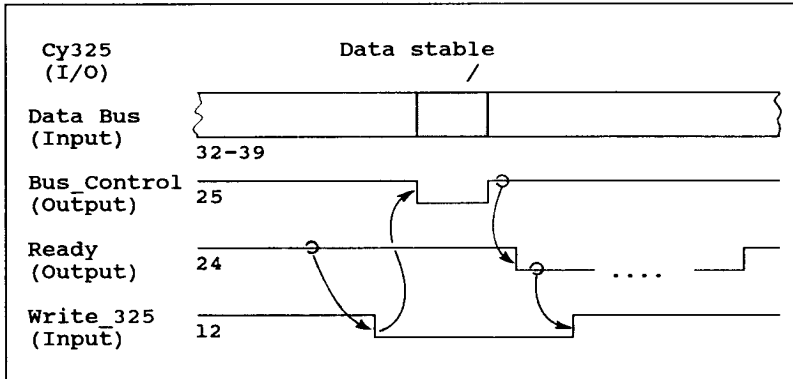
All remaining mode registers contain byte values with no specific meaning associated with individual bits. These mode registers are set or read as 8-bit values.

## Power-on defaults of Mode Registers

#0	06h	0000	0110
#1	3Fh	0011	1111
#2	02h	0000	0010
#3	04h	0000	0100

## CY325 Bus Interface and Handshake

The Cy325 provides a serial and a parallel interface to the LCD display with a simple two wire parallel handshake augmented by a Bus Control signal useful as an Output Enable signal for gating the data onto the data bus. The INPUT handshake sequence is shown below:



The Parallel Input handshake sequence is as follows:

The host or master processor checks the Cy325 READY line to see if the Cy325 is ready to accept a character on the data bus. The host must wait until the READY line is asserted positive true. ( 'TRUE' = 'ON' = 'HI' = '1' = 5 volts ).

After the host has determined that the Cy325 is ready to accept 8-bit data, it drives the WRITE<sub>325</sub> line to the Cy325 low to signal that data is ready to be input to the Cy325. The Cy325 detects the falling edge of the WRITE<sub>325</sub> line and drives the Bus\_Control line low. The Bus\_Control line is used to enable the data from the host onto the Cy325 data bus. The data is read into the Cy325 within approximately 25 microseconds of the falling edge of the Bus\_Control line. This allows software to test the control line before actually putting the data on the bus. The host normally holds the bus in a hi-impedance state, since the Cy325 bus is used to input parallel data and also to communicate with the LCD. By observing the bus control protocol, the host is assured that it will not place data on the bus while Cy325<-->LCD communications are proceeding. After the Cy325 has read the data from the bus, it returns Bus\_Control hi and drives READY low to acknowledge the transfer. The host should drive the Write<sub>325</sub> high when the Cy325 drives Ready low and remove the data from the bus within 10 microseconds, then wait until the READY line goes high before lowering the WRITE line again for the next transfer.



## **Description of the CY325 Bus\_Ctrl Signal Line**

The Cy325 parallel data bus is shared between the user circuit and the LCD interface. For this reason the user must observe the Ready and Bus control lines when attempting to write data to the parallel bus. The Ready line signals that the Cy325 has finished its previous operation and is ready internally to accept a new command or data character. The low Bus\_Ctrl line signals that the Cy325 is no longer communicating with the LCD, and the user can place data on the bus without clobbering Cy325-LCD transfers. The Bus\_Ctrl may be connected to a host latch output enable if a buffer latch is used between the host system and the Cy325.

## **Description of the 'Fast-Bus' Mode of Operation**

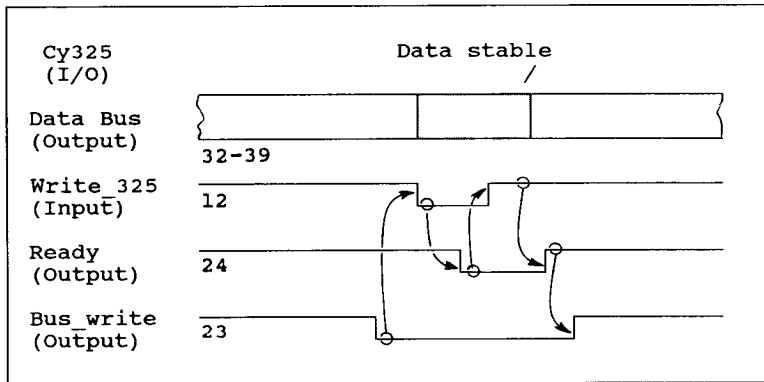
After asserting the Bus\_Ctrl signal low, the Cy325 expects to find valid data on the Bus within approximately ten microseconds. In a similar fashion, the Cy325 expects the data to be removed from the bus approximately ten microseconds after the Bus\_Ctrl is returned high. If Bus\_Ctrl is used as an output enable on a latch or to interface to a Cy233 Network chip, there is no problem with these times, however if the host computer uses software such as a BASIC language interpreter to control the system, the response time of the software may be insufficient to react to Bus\_Ctrl, in which case the system will likely hang up. For this reason the Cy325 supports a Fast-Bus mode as described above, and also a non-Fast-bus mode of operation, designed to support slow speed systems such as BASIC or the Cybernetics dICE-51 debugger. In the Slow-bus mode of operation, the Cy325 waits until it sees data on the bus (not all ones) before reading the data, and, to terminate the transfer, waits until the bus has returned to the all ones state (Bus=0FFh) before completing the transfer. In the Slow-bus mode, timing is effectively disabled, and the state of the data bus is used to effect the handshake. This is the default or the poweron state of the Cy325. The Fast-bus mode of operation can be selected by issuing the Initialize-Fast-bus command; I 9<cr>. In most cases there is no user discernable difference between the two modes of operation, and the user should simply use the Cy325 in the default Slow-bus mode. Only if the few tens of microseconds faster handshake is important to the user, should the Fast-bus mode be enabled. As a final note, the Cy325 requires pullup resistors on its data bus in order to see all ones when a tri-state user circuit is driving the bus.

## **Differences in "Slow-Bus" and "Fast-Bus" Control of CY325**

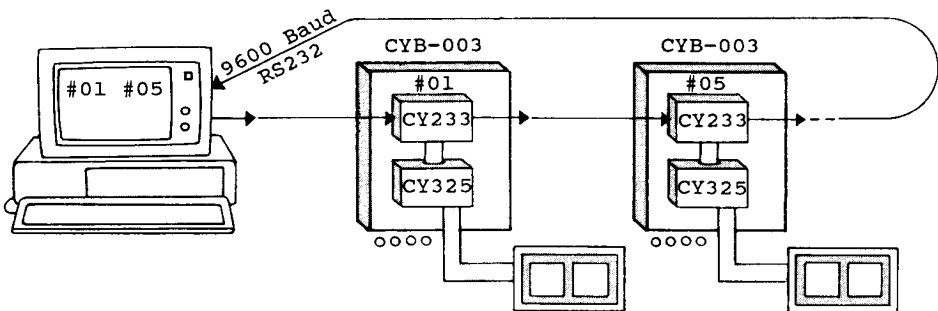
Fast control devices, such as microcomputers, should observe the Bus\_Ctrl signal line for placing data on and off the bus. Slow devices, such as an ICE that is single stepping, should wait until the Ready line goes low (busy) before removing the data and driving the bus high.

## Cy325 Output-to-Bus Handshake

The Cy325 can write data to the parallel bus. The Cy325 pulls the Bus\_Write signal on pin #23 low to initiate a transfer. The host must respond by lowering the Wr\_325 line (pin #12) to announce that it is ready to accept data on the bus. The Cy325 then places the data on the bus and announces its presence by pulling the Ready line low. When the Wr\_325 returns high, the host has accepted the data and the Cy325 then drives all bus lines high, and returns Bus\_write high, terminating the transfer. If the Cy325 output is a string of bytes, the Bus\_Write signal is held low until the last character has been output. This allows Cy233 Network Control chips to transmit a single message, rather than a string of one byte messages, each with a header and terminator.



The bus write mechanism compatibility with the Cybernetics Cy233 Network control chip allows multiple Cy325s to communicate over a single serial I/O channel such as the IBM-PC COM1 channel.



## Cy325 "Soft Keys"

"Soft" keys are keys whose meanings are determined by labels appearing on a display instead of by symbols attached to the keys themselves. For example many IBM-PC programs use the Function keys (F1 .. F10) as soft keys by displaying a key legend on the bottom line of the CRT display.

Soft keys are particularly useful for "menu" based programming where the user of an instrument is given a limited selection of choices displayed on the LCD. If only valid choices are displayed, the user can never make a mistake!. It is for this reason that menu interfaces are preferred by 95 percent of new users of computers. And remember, all of your customers will initially be new users of your instrument.

### Soft-Key Operation

The Cy325 supports the major elements of soft-key operation; the LCD display of menu choices and the detection of the users keystrokes. Since the actual messages and interpretations of the keys are instrument dependent, the Cy325 simply passes a unique key identification code to the instrument processor. The instrument processor is responsible for handling the messages, interpreting the keys, and taking appropriate action.

### Soft-Key Protocol

The most important aspect of Soft-key operation is simply that no keystrokes be lost or confused. This is accomplished by the Cy325 via a positive interlock mechanism. After detecting a keystroke and transmitting this information to the host processor, the Cy325 'locks out' the keyboard until the proper acknowledgement of the key is received from the host.

Although the Cy325 maintains a positive interlock with the host processor, it is also desirable to provide such an interlock with the user as well. In the primary softkey mode (1..6 keys) this can be accomplished by associating an LED with each key. When the user strikes a key the LED is lit and the Cy325 holds the LED ON until the key is serviced and acknowledged. The acknowledgement turns off the LED, thus informing the user that the request for action was seen and recognized. In most menu/soft-key operations the host computer will change the message on the LCD display in some appropriate fashion before releasing the keys via the key acknowledgement.

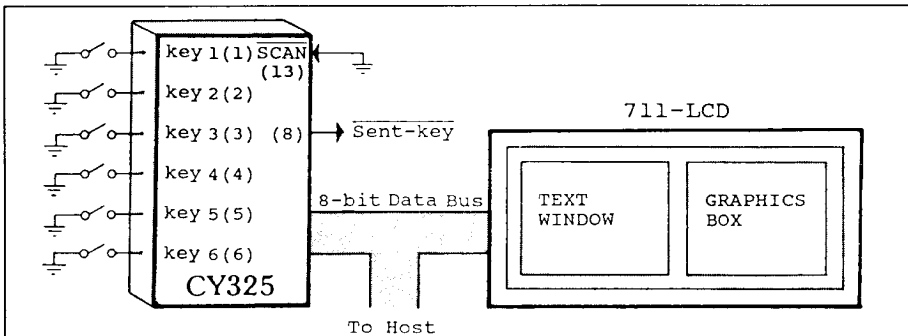
## Cy325 Primary Soft-Keys Operation

The Cy325 LCD Controller supports up to six soft-keys in its primary soft-key mode of operation. These soft-keys allow the LCD screen to display up to six messages and to associate one of six push-button driven signal lines with each message. A "built-in" set of "key" windows can be selected via the 'V n' command where 'n' corresponds to the number of the key below the window. Each such command will position the cursor at the front of each box, in preparation for writing a message into the box. The messages are aligned along the bottom of the display and the push buttons can be physically located in a way that the association of each button with its corresponding message is immediately clear. One LED can be associated with each of the six soft-keys.

## Soft-Key Hardware

The following signal lines support the Cy325 primary Soft-keys:

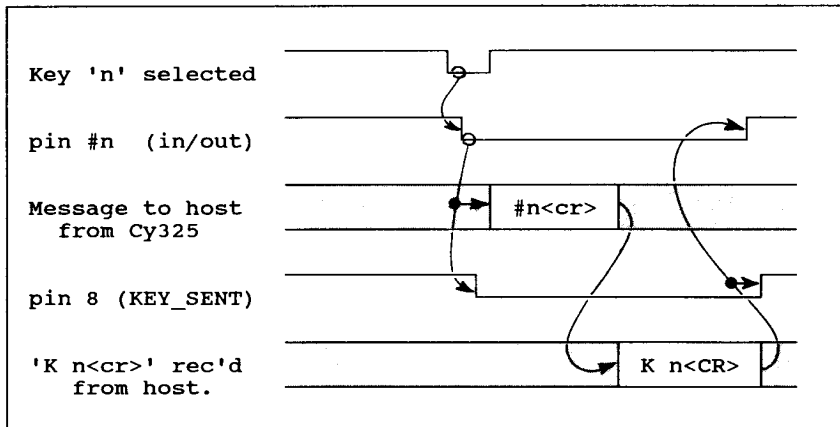
I/O	Pin #	Description
(I)	1	key #1 input
(I)	2	key #2 input
(I)	3	key #3 input
(I)	4	key #4 input
(I)	5	key #5 input
(I)	6	key #6 input
(I)	7	reserved key
(O)	8	key_sent signal active low
(I)	13	Scan Enable signal



## Soft-Key Protocol

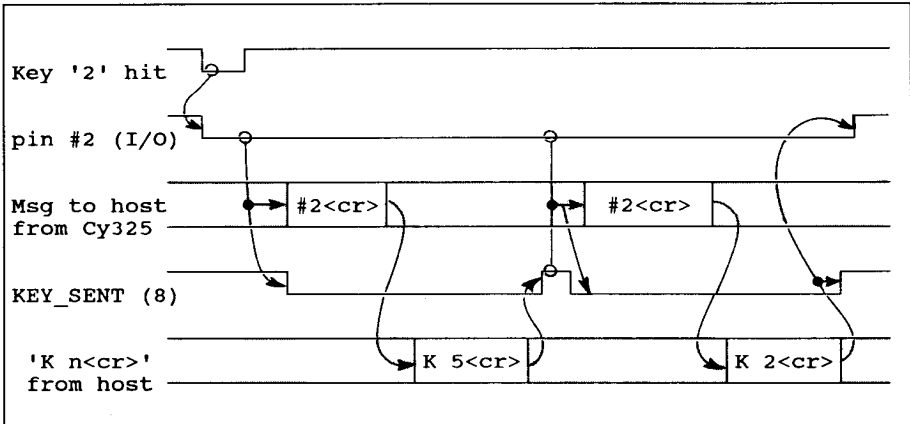
The Soft-key Scanning mode is dynamically enabled when pin 13, the active low Scan\_Enable line is pulled low. The Cy325 will periodically scan the key input lines, pins 1..6, and detect any low level key line such as that associated with a pushbutton. Note that these are simply TTL signal lines and can be used with any suitable signal sources. The following protocol is readily understood, however, in support of physical pushbuttons closely corresponding to LCD messages.

When one of the six key lines is pulled low, the Cy325 will send the key number, (1..6), to the host, prefixed by the ASCII '#' symbol, and post-fixed by the carriage return, <cr>. The Cy325 also drives the KEY\_SENT line, pin 8, low to signal that a key is being sent, and holds the specific key line low. These signals may be used to control LED drivers. The KEY\_SENT line inhibits key scanning and remains low until the host has acknowledged the receipt of key number n via the message 'K n<cr>'. When the Cy325 receives the 'K n'-command, it raises the KEY\_SENT line and also the line associated with key #n, that is, pin n. If the acknowledged pin was the same as the activated pin, then the Cy325 resumes key scanning (if the SCAN line is still held low). If the wrong pin is acknowledged, then the KEY\_SENT line will be raised, enabling key scanning, but the original key line will still be held low by the Cy325 (since the wrong line was raised) and this will cause a second (and so on) report to the host. Thus each key requires a positive and accurate response (acknowledgement) from the host before restoring its key input to the ready state.



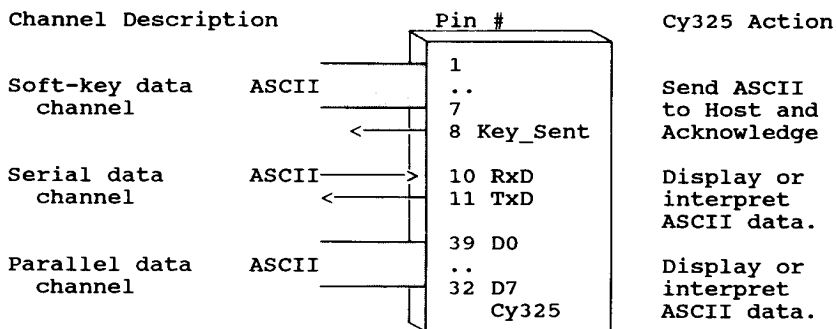
## Host Response Key Command

The key scan operation assumes that a host or master processor is in the system, since the Cy325 is not designed to make use of the keys. Therefore the key response must be placed on either the bus or the serial output channel. These key 'echo' channels, as described in the next section, are enabled by mode bits in a user accessible mode register. If either, or both, of the echo channels is enabled, the Cy325 response to key #n pulling line #n low is to echo the message, '#n<cr>' to the master. For proper pushbutton operation, the master should acknowledge this message by sending the 'K n<cr>' command (making sure the Cy325 is in command mode). The following illustrates the case in which key #2 pulls pin 2 low, sending the message '#2<cr>' over the 'echo' channel. The Host responds with 'K 5<cr>', which is the incorrect response to acknowledge key #2. As seen below, the KEY\_SENT line is raised by the 'K 5' response, while the key #2 line (pin 2) remains low. When the KEY\_SENT line returns high, Scanning mode is resumed. When pin 2 is detected low, the Cy325 assumes that key 2 is holding it low and issues another '#2<cr>' message to the 'echo' channel(s). In this way the soft-keys 'insist' on being properly acknowledged. When the host finally sends the correct acknowledgement message, 'K 2', the Cy325 raises key #2 (pin 2) and the KEY\_SENT line, (pin 8). At this point, assuming that the SCAN line is still held low, key scanning is re-enabled and new keys can be sensed. In most applications this assumes that the host processor has processed the '#n' message according to the interpretation associated with the corresponding window message at the time when the key was depressed. It also assumes that the host has updated the messages displayed before releasing the keys via the proper acknowledgement. Note also that the key n line can be used to hold an LED 'on' from the time the key is hit until the host acknowledges the key message. This visual signal to the user is considered to be a very significant feature of the soft-key protocol.



## Soft ASCII-Key Operation

The Cy325 Soft-key operation can support ASCII 7-Bit keys on pins 1..7. These ASCII characters are distinct from the ASCII data that is sent to the Cy325 over the parallel or serial channels. While ASCII data input via parallel or serial channels can be either displayed on the LCD or interpreted as commands, ASCII data that is input via pins 1..7 can only be transferred to the host computer as a soft-key input. The Cy325 makes no use of this ASCII information other than to pass it through to the host computer. The three input data channels are shown below:



The ASCII soft-key operation detects a seven bit ASCII code on keys 1..7, and sends this code to the system, after prefixing the '#' character. If the key\_to\_TxD bits is enabled, the Cy325 will append a <cr> to the message. If the key\_to\_Bus is enabled, only the '#' and the ASCII code will be sent. If a Cy233 is connected to the Cy325, the Cy233 will append a <cr> to the message (and also prefix the Read header and address.) The following example illustrates several aspects of ASCII Soft-key operation.

Commands to Cy325:

```

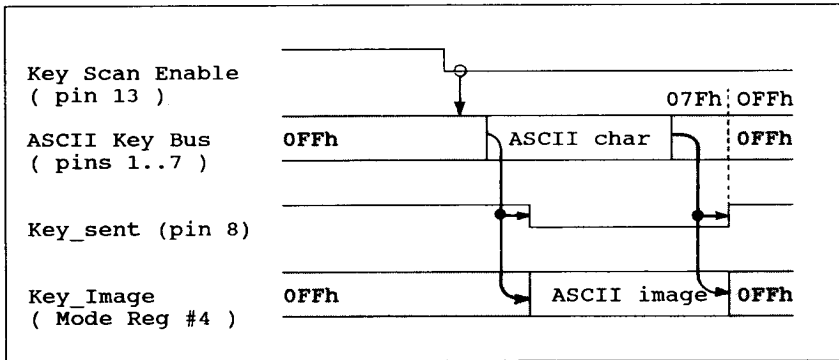
M 2,8<cr>      Command to select ASCII Soft-key operation
M 3,0Ch<cr>    Enable Key_to_Bus AND Key_to_TxD
I 5<cr>        Enable Soft-key Scan operation
I 8,45h<cr>    Force 45h on bus internally !
  
```

```

Response from Cy325 on:  TxD      Bus      Cy233(#2)
                        #E<cr>    #E      R02#E<cr>
  
```

## ASCII Soft-Key Protocol

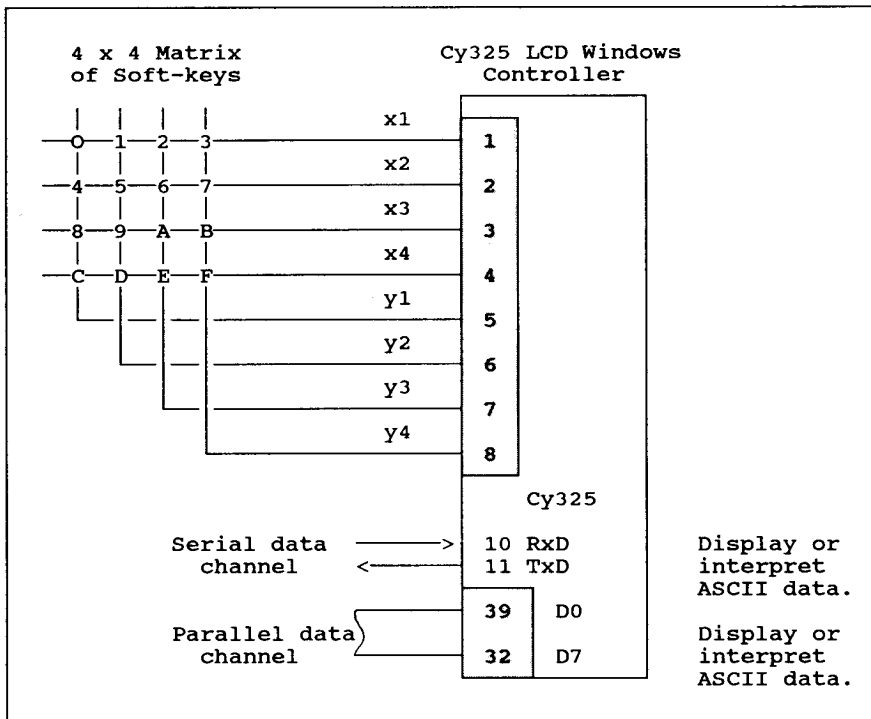
The ASCII soft-key operation is selected by setting bit 3 of mode register #2. This can be done via the mode command, **M 2,8<cr>**. In this mode the Scan\_Enable line (pin 13), when pulled low, will cause the soft-key pins, 1..7, to be read into the Key\_image (mode register #4) and the ASCII value on the bus will be transmitted to the host computer, based on communication flags in mode register #3. The Key\_Sent line (pin 8) is pulled low to acknowledge the ASCII soft-key and signal that the key is being transmitted. This signal can be used to restore the soft-key bus to the high state, that is pins 1..7 must be returned high to terminate the soft-key input after the Key\_Sent line goes low. In this mode the Key Scan Enable line can be left active low or can return high after Key\_Sent goes high. Regardless of the operation of the Scan\_Enable line, the Soft-key bus must be returned high between key strokes so that the bus reads 07Fh between two valid ASCII code combinations.





## Soft 4x4 Key Scan Operation

The Cy325 supports 4x4 matrix key switches on pins 1..8 as shown below. The 4x4 Soft-key operation is selected by setting bit 4 of mode register 2, via the mode command, **M 2,10h<cr>**. The mode is enabled by pulling the Scan\_Enable line (pin 13) low, either via the **I 5<cr>** command or via hardware external to the Cy325. The Cy325 then begins an active scan of the 4x4 matrix, looking for a key closure that connects an 'x' line to a 'y' line. When such a closure is detected, the Cy325 transmits the '#' character and then the appropriate hex key code ( '0'..'F' ) to the host. Since the Key\_Sent line (pin 8) is used in the active matrix scan, it is not available as an acknowledge signal line as in the other Soft-key operational modes. Soft-4x4-keys are cleared by sending the generalized acknowledge Key command, **'K 0<cr>'**.



The 4x4 key scan operation is inhibited during LCD operations. Although this is normally of such brief duration that the suspension of the key scan would not be noticed, it may become noticeable during graphic erase or fill operations.

## 4x4 Soft-Key Matrix Table

Yn	Xn	Key#
1110	1110	0
1101	1110	1
1011	1110	2
0111	1110	3
1110	1101	4
1101	1101	5
1011	1101	6
0111	1101	7
1110	1011	8
1101	1011	9
1011	1011	A
0111	1011	B
1110	0111	C
1101	0111	D
1011	0111	E
0111	0111	F

## 4x4 Matrix Scan Timing Considerations

The active scan of the 4x4 key matrix requires time on the part of the Cy325, and this key handling time diminishes the time that is available for handling other functions. For example, assume that large characters are being input to the Cy325. Manipulating large characters requires sending 4 normal characters to the LCD and each transfer takes time. Normally, the Cy325 can receive and display large characters continuously at 9600 baud, however when the 4x4 key matrix scan is enabled, the Cy325 cannot sustain continuous operation at this rate. In order to support large characters during 4x4 key scanning, the baud rate must be reduced or there must be a delay time included between each character. In the serial data input mode the received characters are saved in a buffer internal to the Cy325, and buffer overflow will occur when long operations on the LCD (such as large character display, box fill or erase, or window scrolling) combine with 4x4 key scanning and high input data rates. If parallel input is used, the Ready line (pin 24) should hold off the host long enough to avoid this problem. The equivalent Ready line for serial is **Clear-to-Send**. When **CTS** (pin 15) is low, the Cy325 is ready to accept serial data in its buffer. When **CTS** goes high, serial input must be held off to prevent a possible buffer overflow.

Another possible solution to the 4x4 timing problem is to disable the 4x4 key scanning via the **/I 5<cr>** command, then transfer high speed serial data to the Cy325, then re-enable 4x4 key scanning via the **I 5<cr>** command.

## Soft Key Acknowledgement in 4x4 Scan Mode

The 4x4 Soft-key operation does not allow individual keys to be acknowledged. All 4x4 Soft-keys are cleared by the same command. The generalized 4x4 soft-key acknowledge is 'K 0<cr>' which restores all eight scanning lines high and re-enables the 4x4 key scan operation. Since no key-specific acknowledgement exists, it is impossible for the Cy325 to insure that the right key has been acknowledged, so the Cy325 never repeats 4x4 key messages as in the normal soft-key mode of operation.

## Absolute Soft Key Verification via Query

As discussed in the section on Soft-key Protocol, the Soft-keys enabled by mode bit #0 of mode register #2 ( via M 2,1<cr> ) are individually acknowledged. If the incorrect key is acknowledged, the correct key will be repeated, and this sequence will continue until the correct key is acknowledged. This sequence, although assuring that the correct key will eventually be seen, can allow the host computer to falsely assume that the initial (incorrect) key acknowledgement was in fact correct and that the repeated key is a new key, rather than a correction message. For applications where it is imperative that all keys be correctly interpreted, it is possible to verify every Soft-key before acknowledging it. The Query command, '?', can be used to read the Key\_image register, (mode register #4) and independently verify which key was pressed before taking any action based on the key message.

The '? 4<cr>' command will query the Key\_image register, and will return an ASCII string of the form '?xxh<cr>' where xxh is the 2-character Hex-ASCII code for the 8-bit Key image in the register, followed by the 'h' suffix that is appended by the Cy325, then terminated by carriage return, <cr>. The Soft-key messages and verification query messages are shown below for the 6 soft-keys:

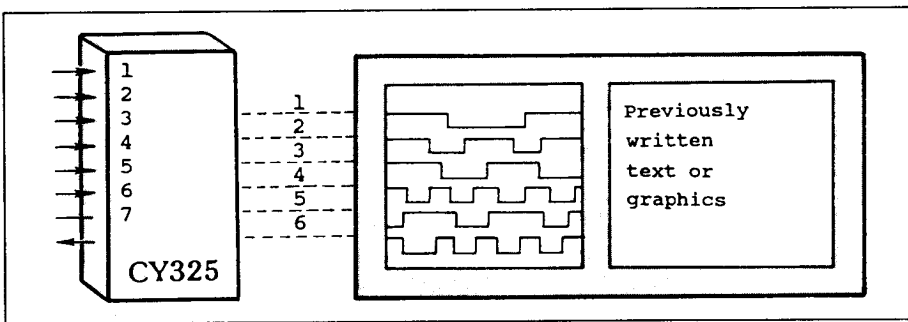
Soft-Key selected	Soft-Key message	Verify message	Key_image pattern
1	'#1<cr>'	'?7Eh'	01111110
2	'#2<cr>'	'?7Dh'	01111101
3	'#3<cr>'	'?7Bh'	01111011
4	'#4<cr>'	'?77h'	01110111
5	'#5<cr>'	'?6Fh'	01101111
6	'#6<cr>'	'?5Fh'	01011111

After the Soft-key #n message has been correctly acknowledged via the appropriate 'K n<cr>' command from the host, the result of a following Query of the Key\_image, via the '? 4<cr>' query command should be '?FFh<cr>'.

# 11 CY325 Logic Waveform Display 11

## CY325 Logic Display Mode

The Cy325 auxiliary data channel is scanned when the **Enable\_SCAN** line (pin 13) is pulled low. When the Logic mode bit (**M2.1**) is set high, via the **M 2,2<cr>** command, the Logic Waveforms mode is enabled. In this mode Cy325 pins #1..#6 are scanned and the changes on these lines are displayed in the 'current' window, which can be any width in excess of two columns wide and should be at least six rows deep. If fewer than 6 rows are used, then some of the waveforms will not be displayed. The Cy325 will read the pins and represent the TTL level of each pin on a separate row with the latest value entered at the right side of the display and the past values sliding to the left with each new change displayed. An example logic display is shown below:



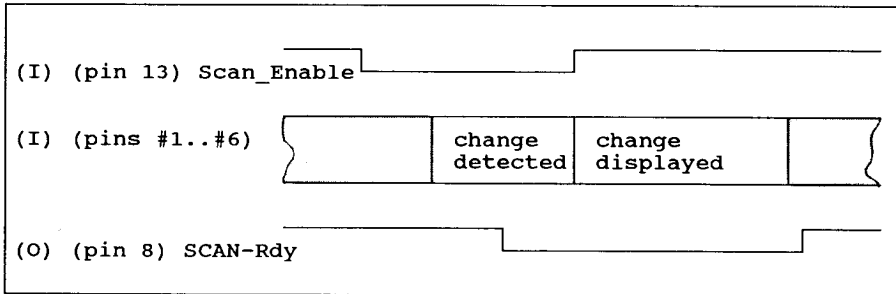
Pin 7 on the Cy325 is also sampled, however only six waveforms are displayed. Pin 7 can be used to advance the waveforms without any of the six displayed waveforms changing. Thus pin 7 effectively "clock's" the waveform to show "time passing" without requiring any of the six active lines to change.

The waveforms can be driven by the command **'I 12,n<cr>'** without actually driving the hardware pins. It is not necessary that the scanning mode be enabled for this command. The six LSBs of the binary value 'n' are used to drive the six Logic waveforms in the current window.

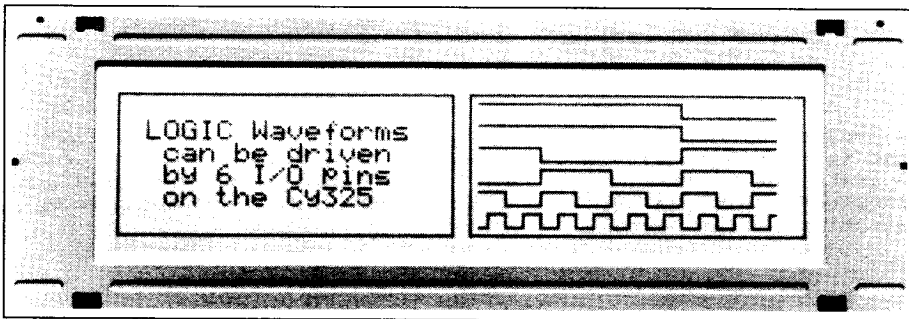
Finally, note that the Cy325 hardware pins can also be driven via the **'K n'** and the **'/K n'** commands (if they are not pulled high or low externally) and also by the **'I 8,n'** command. These commands have the effect of driving the waveform pins from inside the Cy325 and will be registered in the Logic waveform display window if the Logic waveform mode of operation is selected and enabled. If the pins are actually tied directly to Vcc it is impossible to pull them low from inside, so pullup resistors should be used if you plan to use these commands.

## Buffering the Logic Waveform Signals

The Cy325 Logic Display function operates in either unbuffered or buffered mode. In unbuffered mode, the changing signals are connected directly to Cy325 pins #1 thru #6, and the changes are reflected as they occur. This must, of necessity, limit the rate of change of the individual lines, due more to the limitations of human vision, but also to the rather slow scan rate of the Cy325 (to be specified). For most applications the Cy325 serves only as a display, with the actual data collection and storage managed by a master processor that uses the SCAN-Ready line (pin 8) and the SCAN line (pin 13) to handshake with the Cy325, where one change is displayed in the window per transfer. The handshake sequence is shown below:

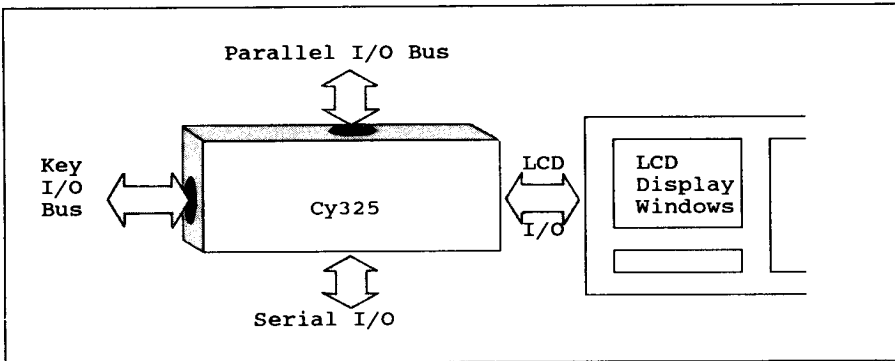


Thus if it is desired to capture hi-speed logic signals and display them on the LCD then you might use a FIFO or other device capable of saving a sequence of high speed logic signals and use your host processor to unload the FIFO and present the buffered signals to the appropriate pins on the Cy325.



### The CY325 as a Switching Element

The Cy325 possesses a variety of message switching capabilities, based on switch-flags located in the Communications Mode Register (#3) and augmented by several Cy325 instructions. The four I/O channels associated with the Cy325 are shown below:



The four major Cy325 data channels are:

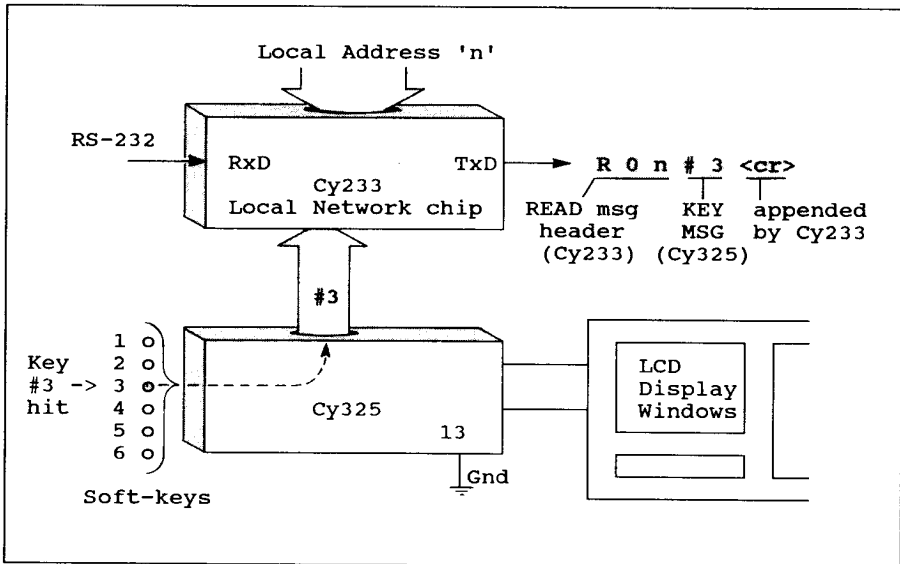
- |                    |
|--------------------|
| Key Data Port      |
| Parallel Data Port |
| Serial Data Port   |
| LCD Data Port      |

As indicated, each of these ports is potentially bi-directional, that is, data can flow into or out of the Cy325 over each of the separate data channels. Each of the data channels will be discussed individually, then all of them will be summarized in a switching matrix table.

# The Soft-Key Input Channel

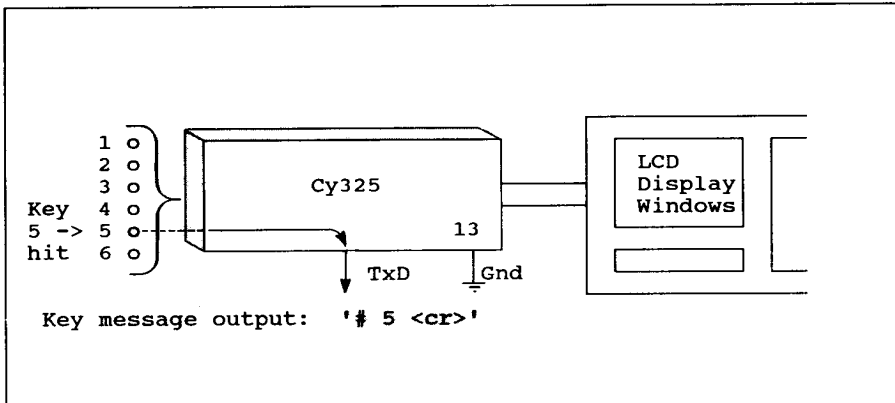
## Case 1: Key\_to\_Bus Switch Enabled

The Key channel consists of the eight pins (1..8) on the Cy325. Either 6, 7, or 8 lines are used, depending on whether simple soft-keys, ASCII soft-keys, or 4x4 matrix soft-keys are selected. In all cases the key inputs are received at the key port and output to either the parallel or serial data channels. Soft-key inputs are never output to the LCD. The choice of parallel or serial output is determined by the Communications mode register, specifically by the **Key\_to\_Bus** flag (M3.3) or the **Key\_to\_TxD** flag (M3.2). If the **Key\_to\_Bus** flag is ON, a key closure will cause the '#' character to be output to the parallel data bus, followed by the ASCII hexadecimal key identifier code, ('1'..'6') for simple switches or ('0'..'F') for 4x4 matrix mode key switches. If the parallel data port is interfaced to a Cy233 Network chip, the Cy233 will prefix a READ message header ( 'R' followed by local address) to the message ('#3') and then terminate the message with a carriage return. The scheme is shown below:



## Case 2: Key\_to\_TxD Switch Enabled

The Key\_to\_TxD switch is enabled via the Mode command, M 3,4<cr>, which sets the Key\_to\_TxD bit (M3.2) of the Communications mode register. The effect of this is to cause all softkey activations to send messages to the Serial Output channel, TxD. The messages consist of the '#' character, followed by the relevant ASCII key identifier ('1'..'6' or '0'..'F'), terminated by the carriage return, <cr>. The message will be transmitted at the same baud rate as the receiver is set for.

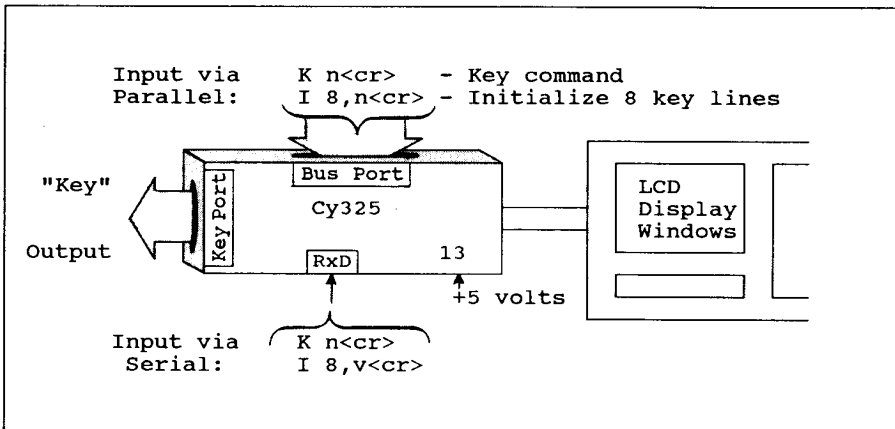




## Key Channel Outputs

Normally the soft-key channel is an input channel, whose scanning is enabled via the 'I 5<cr>' command or by pulling pin 13, the Scan\_Enable line low externally. The key activation is interpreted according to the Key Mode register (#2) and the output message is steered by the setting of the switches in the Communications mode register (#3). Activated key lines are held active low by the Cy325 until receipt of the Key Acknowledge command, 'K n<cr>', at which time the key line is de-activated (assuming that the user has released the key).

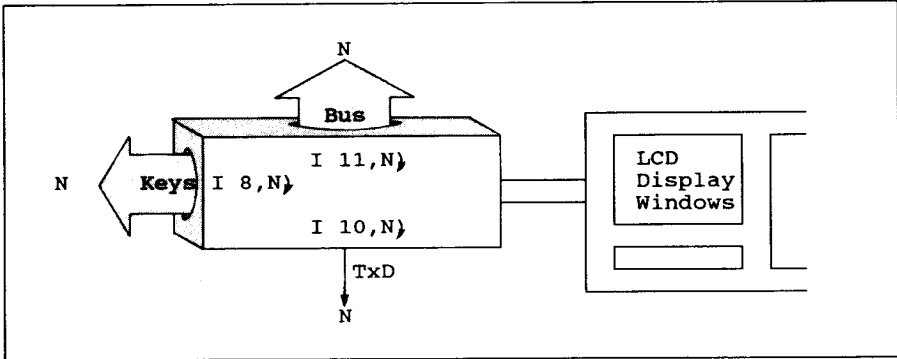
If the Key Scan is disabled via the '/I 5<cr>' or by pulling pin 13 high, the input capabilities of this channel are essentially disabled. The Key port can, however, be used for output via either the 'K' or the 'I' commands. The '/K n<cr>' command lowers line 'n' and 'K n<cr>' returns it high. The Initialize Key port command, 'I 8,n<cr>', sends the 8-bit value, n, to the eight key pins, thus outputting a data byte to the Key port.



## The Initialize “Output” Commands

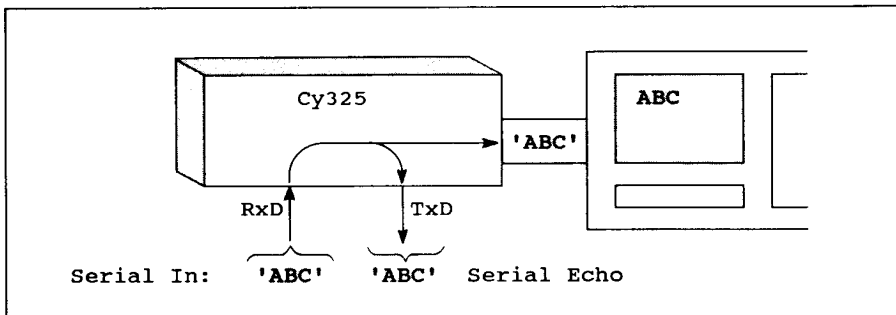
There are three special Initialize commands that output data to three different ports on the Cy325. The command can be issued to the Cy325 via the parallel or the serial channel. The output channel is completely determined by the command, regardless of the channel over which the command was received, (in contrast to the 'Send' and 'Transmit' commands to be discussed later).

I 8,n<cr>    Send binary value of 'n' to Key Port (pins 1..8)  
 I 10,n<cr>    Send binary value of 'n' to Serial port (pin 11)  
 I 11,n<cr>    Send binary value of 'n' to parallel bus (32..39)



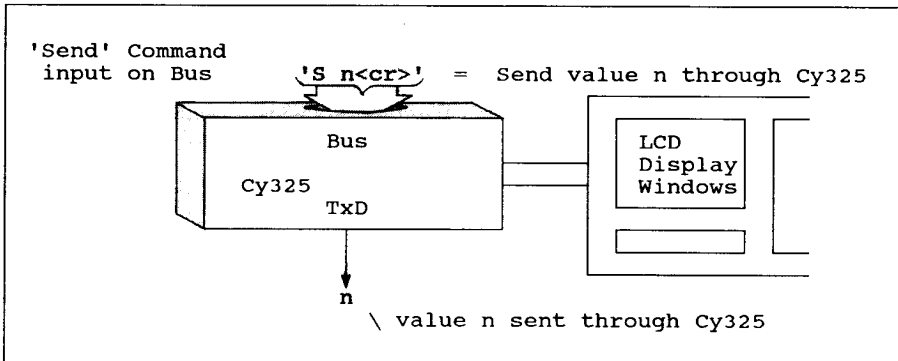
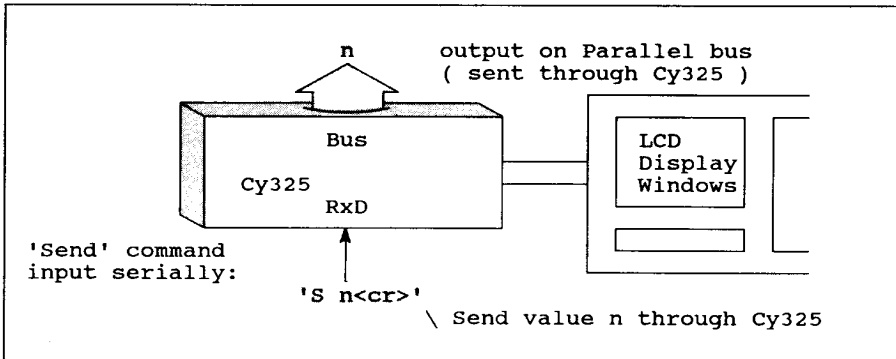
## The Serial Echo

The Communications mode register (#3) contains an 'Echo\_serial' switch or mode bit that causes all displayed characters received on the input serial channel (RxD) to be echoed over the output serial channel (TxD) when the bit is set (via M 3,10h<cr>). There is no corresponding echo bit for parallel data. The serial echo does NOT affect command characters, only displayable characters that are sent to the LCD.



## The Send Command

The Commands discussed above have generated output regardless of which channel the command was received on. The 'Send' command (and two more commands) differs in that the output channel is a function of the input channel on which the command was received. The intent of the Send command is to Send data through the Cy325, regardless of where the data comes from. Thus data input serially will be output to the parallel bus, while data input from the parallel bus will be output on the serial channel.



The Format of the Send Command is as shown:

where **S n<cr>**  
 n = ASCII decimal or hex-ASCII 8 bit value

The argument of the Send command is input as an ASCII string numeric value, while the binary 8-bit equivalent is output.

## The Transfer Command, '@'

The Transfer command, like the Send command, is designed to transfer characters through the Cy325. If the transfer command is issued to the parallel port, the data will be output to the Serial port. If the Transfer command is issued to the serial port the data will be transferred to the parallel port.

The Send and Transfer commands are somewhat analogous to the Plot and Plot-String commands. The Plot command plots one pixel per command, while the Plot-String command plots a string of pixels, whose values are entered in binary form. Similarly, 'Send' uses a single argument, specified as a decimal or hex ASCII parameter, and sends the binary equivalent value to the port opposite that on which the command was received. The Transfer command, '@', has a binary data count that specifies how many bytes are to be transferred through the Cy325. The data count is followed by data bytes in binary form, each of which is sent through the Cy325 and the data count decremented. The format of the Transfer command is as follows:

```
@ n x1 .. xn <cr>
```

where

@ = ASCII opcode ('@' = 40h)

n = 8-bit binary data count

x1 = first 8-bit data byte to be transferred

xn = last 8-bit data byte to be transferred

The diagrammatic representation of the Transfer command is almost identical to that of the Send command. Data transferred from the serial side is output to the parallel bus with appropriate handshaking. If the same Transfer command is input to the Cy325 parallel bus, the three data bytes will be sent to the Cy325 serial port, TxD, with no header and no terminator. While the argument of the Send command is usually an ASCII decimal or hex string, the arguments of the Transfer command are always binary values. The values at the output port are always 8-bit binary numbers, regardless of the input format of the arguments.

An example in which the Transfer command sends data from the Cy325 serial input (RxD) to a Cy233 Network chip interfaced to the parallel bus of the Cy325 is presented in the next section.

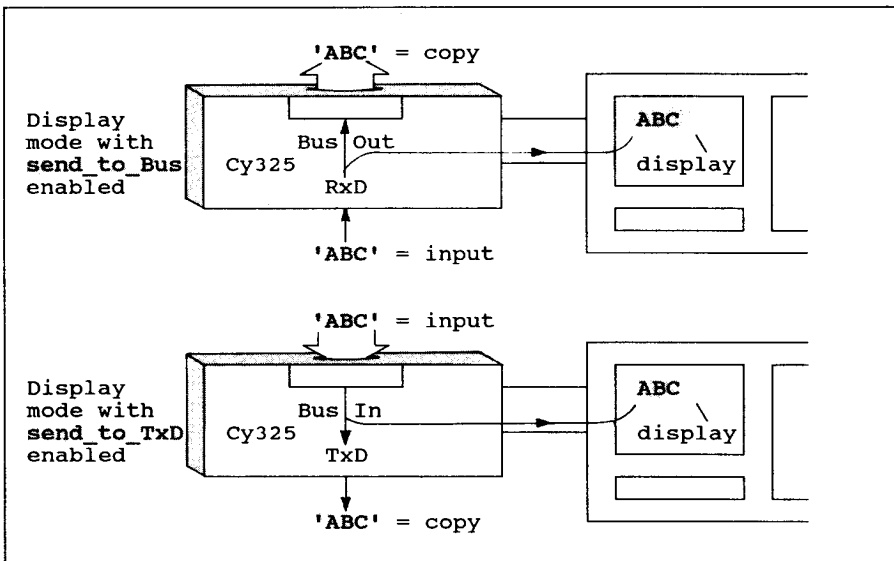
## Copy Switch Functions: Send\_to\_Bus and Send\_to\_TxD

The **Send** and **Transfer** commands pass their arguments through the Cy325. Their argument values are output to the channel opposite that which the commands are received on. This behavior can be generalized such that all displayable characters are copied to the channel opposite that which they are received on. All serial characters that are displayed on the LCD can also be output to the parallel bus, and all displayed characters that are received on the parallel bus can be output serially on TxD. The mode bits in the Communications Mode Register that control this switching behavior are bits M3.0 and M3.1:

M3.0 send\_to\_Bus - copy serial display to parallel bus

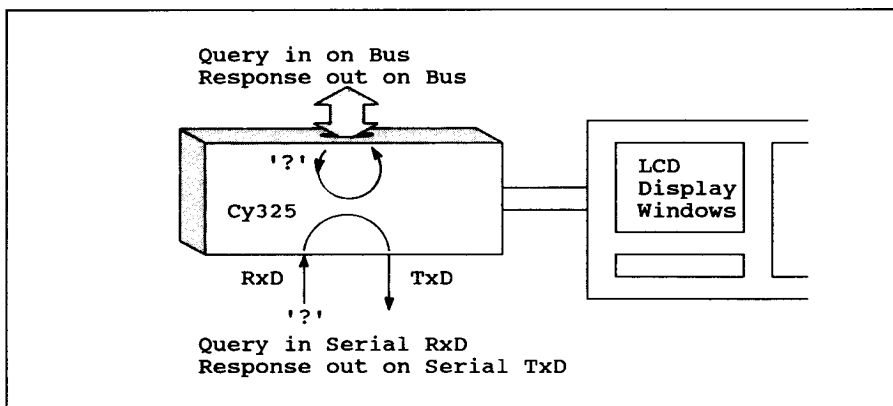
M3.1 send\_to\_TxD - copy parallel display to serial out

Only displayable characters are transferred through the Cy325. No 'Ctrl' characters are transferred out, even though they may be issued in the Display mode. If 'Ctrl' characters are to be transferred through the Cy325 this must be done via the Send or Transfer commands. No commands or command arguments are copied to the opposite channel. Only characters which will be displayed on the LCD will be copied to the opposite channel when these bits are set. Either or both of the 'copy' bits, M3.0 and M3.1, may be set, however the copy action is only from the receiving port to the opposite port. These bits do NOT generate echoes to the receiving port, only copies to the opposite port.



## The Query Command, '?'

The Query command, '?', is designed to extract information from the Cy325. The information consists of either the value in a specified mode register, or the LCD character or character string located on the LCD screen at the current character cursor address. The Query response depends on the input channel over which the Query was received, however the sense is opposite to that of the Send and Transfer commands. While those commands are designed to send information through the Cy325, the Query command is designed to extract information from the Cy325. Thus the data output in response to a Query command is output to the same data channel that the command was input to. A Query entered on the serial channel will respond to the serial channel, while a Query command received on the parallel bus will respond to the parallel bus, as shown below:



The format of the Query command is as follows:

```
'? n<cr>'
```

where

```
'?'      ASCII Command character  
n        ASCII numeric value  
<cr>    command terminator
```

The value that is output in response to the Query command depends on the numeric value of the Query argument as follows:

```
n = { 0 .. 19 }  respond with 8-bit value in Mode Register #n.  
n = 20           respond with 8-bit character code of the  
                  character located on LCD screen at position  
                  specified by current character cursor.
```

## Query Command Response Formats

The response to a Query command is output to the same channel on which the Query command is received. The format of the response to a query of one of the mode registers is:

```
? x x h <cr>
```

where the '?' prefix identifies the message as a query response, (in the same fashion that the '#' prefix identifies soft-keys) and the two ASCII-hex values 'xx' are the hexadecimal equivalent of the mode register contents, with a redundant 'h' confirming this fact, terminated by a carriage return. All mode register query responses always have this format, even when the Cy325 binary command mode is selected.

When the Query command argument is 20, the command is not a mode register query, but a request to the Cy325 to read the character on the LCD screen located at the current cursor position. In this case the Cy325 will output the ASCII value of the character read from the LCD, instead of the two-byte ASCII-hex equivalent. There are two types of Character Query, determined by the setting of the "Query\_auto\_inc" mode bit (M1.5) in the LCD mode register #1.

If the Query\_auto\_inc bit is OFF, the character Query command simply reads the character at the cursor location and transmits:

```
? A <cr>
where
? = Query prefix
A = ASCII character read at cursor
<cr>= terminator
```

If the Query\_auto\_inc bit is ON, the character Query command reads the character at the cursor location, and tests to see if it is the ASCII blank (20h). If not, the cursor is advanced, and the next character is read. This continues until either:

```
a blank character is read, or
fifteen characters are read, or
the right side of the window is reached.
```

The character read terminates when any of the above cases occurs. The message that was read from the LCD is then transmitted with the format:

```
? ABC..FGH _ <cr>
where
? = Query prefix
ABC..FGH = ASCII character string on LCD
_ = ASCII blank (if read from LCD)
<cr> = terminator
```

If the response is to a Cy233 the 'Rnn' header will be prefixed.

## Cy325 Switching Function Summary

The preceding discussion treated the various switching functions separately. This section summarizes the various modes, commands, paths, and relevant switching information. The switching functions of the Cy325 are designed to provide data paths through the four information channels associated with the Cy325.

Data Ports:

Key Data Port Parallel Data Port Serial Data Port LCD Data Port
--

The actual switching functions depend upon mode bits, commands, and often upon the channels over which the commands are input. The Cy325 commands that perform switching action of some sort are the following:

Commands:	'@' n x1 .. xn	Transfer thru Cy325 to opposite port
	'S n<cr>'	Send thru Cy325 to opposite port
	'I 5<cr>'	Scan key port for changes
	'I 8,n<cr>'	Output value n to Key port
	'I 10,n<cr>'	Output value n to TxD port
	'I 11,n<cr>'	Output value n to Bus port
	'? n<cr>'	Output mode reg n to same port

The mode registers that are relevant to Cy325 data switching are the Key Mode Register #2 and the Communications Mode Register #3. These registers are shown below

### Mode Register #2 - Key mode flags

M2.0	- soft_keys	; soft-keys [1..6] = pins 1..6
M2.1	- Logic_waves	; pins 1..6 = digital logic [1/0]
M2.2	- Cursor_keys	; pins 1..4 = up-down-left-right
M2.3	- ASCII_keys	; pins 1..7 = ASCII key inputs
M2.4	- Key_matrix	; pins 1..8 = 4x4 matrix (16 keys)

### Mode Register #3 - Communications flags

M3.0	- send_to_Bus	; qualify by input flag
M3.1	- send_to_TxD	; qualify by input flag
M3.2	- Key_to_TxD	; send keys to TxD
M3.3	- Key_to_BUS	; send keys to BUS
M3.4	- echo_serial	; echo RxD-Display to TxD



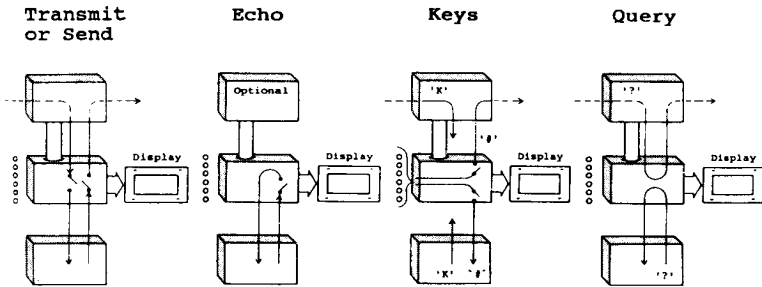
## Cy325 Message Switching - Major Options

The following diagrams illustrate the data paths through the Cy325. The effect of the mode bits in the Communications mode register are shown as switches in the data paths. If the mode bit is enabled (ON), the switch will be 'closed', while if the mode bit is disabled (OFF), the switch will be 'open'.

LOCAL  
PARALLEL  
DEVICE  
OR  
NETWORK  
CONTROLLER

CY325  
SYSTEM  
ELEMENT

LOCAL  
SERIAL  
DEVICE



Display  
Mode

Command  
Mode

	X (Displayable char. only)	X (Serial only)	X
	X (Send or 'e')		X ('? ' response)

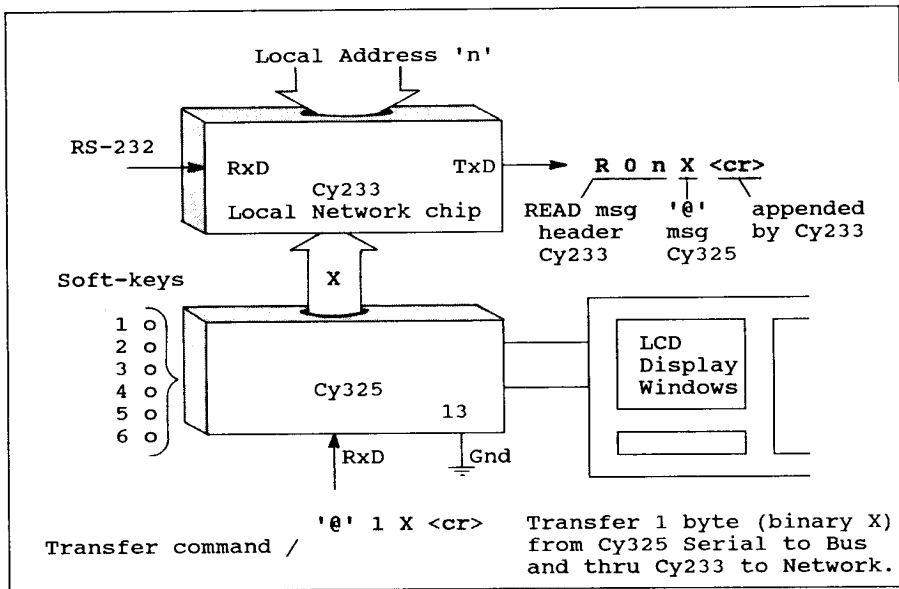
## Cy325 Switching Matrix

	Input	Display Mode		Command Mode	
Soft keys	KEY ↓	key_to_Bus	key_to_TxD	key_to_Bus	key_to_TxD
Transmit Pass-thru copy	Bus		Send to TxD		
	RxD	Send to Bus			
ECHO	Bus				
	RxD		Echo Serial		
Send 'e'	Bus				Send to TxD
	RxD			Send to Bus	
Query	Bus			Automatic	
	RxD				Automatic
	Output →	to Bus		to TxD	

# 13 CY325 in an RS-232 Network 13

## The Cy325-Cy233 Network

In many applications, such as point of sale terminals, it may be desirable to connect a number of LCDs to a host computer via a local area network. Although this can be implemented in a number of ways, the Cy325 directly supports Cybernetic Micro Systems Cy233 Local Intelligent Network Controller, that consists of a +5 volt CMOS 40 pin IC capable of supporting up to 255 network addresses on one serial communications channel such as the COM1 port of an IBM-PC or equivalent. The interface between the Cy325 and the Cy233 is shown below as a box diagram and in schematic form on a following page.



The Cy233 connects to the Parallel bus of the Cy325 and supports Soft-key operation with up to sixteen keys. The Cy325 can also accept Serial information to be displayed or to be sent over the network to the host computer. The Cy325 Transfer command is used to transfer data from the serial port of the Cy325 (RxD) to the parallel bus that is connected to the Cy233 Network Controller.

## Brief Description of Cy233

The Cybernetic Micro Systems Cy233 is a 5-volt 40 pin CMOS device providing networking capability on RS-232-C ports. Each Cy233 in a network is given a unique address by tying its address lines high or low as appropriate. The Cy233 supports ASCII, binary, or hex message formats. To write the ASCII message string 'ABC' to a Cy233 with address set to 03 we send:

```
'W 0 3 A B C <cr>'
```

The Write header, 'W' tells the Cy233 to write a message from the serial network to the parallel device (Cy325) attached to the Cy233. The address determines which Cy233 will actually write the message. The 'ABC<cr>' is written to the Cy325. The carriage return terminates the message. To send a message (XYZ) over the network, the Cy325 handshakes with the Cy233 via its built-in, Cy233-compatible, protocol. The Cy233 accepts the message from the Cy325 attached to its parallel bus and sends the 'Read' message (with source address) over the serial network as follows:

```
'R 0 3 X Y Z <cr>'
```

## Transfer Command Example

Assume that data is to be transferred from the serial side of a Cy325 to the Serial Network, composed of Cy233 Network Controllers, each connected to the data bus of a Cy325 LCD Windows Controller. The Transfer command, '@', received on the serial Cy325 channel will transfer bytes, one at a time, over the parallel bus to the local Cy233 which will then prefix a Read header consisting of ASCII 'R' followed by the local address specified as two hex-ASCII bytes. The data byte is then used as the message data and the Cy233 appends a carriage return (message terminator). Thus the example transfer command;

```
'@' 3 'A' '2' 'B' (quotes indicate ASCII)
```

would be transfer three data bytes, 'A', '2', and 'B' to the Cy233 with appropriate handshaking. The arguments are binary numbers, however their ASCII equivalent is shown in quotes. The actual command input to the Cy325 would have the values:

```
'@' 03 41 32 42 (arguments shown in hex-ASCII)
```

A Cy233, with local address 0x, upon receiving these three data bytes would compose three messages and transmit them on the serial line as follows:

```
'R 0 x A <cr>' (0x is local Cy233 address)  
'R 0 x 2 <cr>'  
'R 0 x B <cr>'
```

## Soft Key Acknowledgement via Query in a Network

The use of the Query command for absolute verification of each soft-key was discussed in the section on Soft-keys. The Query can be used for the same purpose in a Cy325-Cy233 network as will be described now. The Soft-keys are enabled by mode bit M2.0 via M 2,1<cr> and are individually acknowledged. For applications where it is imperative that all keys be correctly interpreted, it is possible to verify every soft-key before basing any action on the soft-key message.

The '? 4<cr>' command will query the Key\_image register, and will return an ASCII string of the form '?xxh<cr>' where xxh is the 2-character Hex-ASCII code for the 8-bit Key image in the register, followed by the 'h' suffix that is appended by the Cy325, then terminated by the carriage return, <cr>. The Soft-key messages and verification query messages are shown below for the six soft-keys. The messages are shown on the network serial channel, that is, after they have passed from the Cy325 parallel data bus into the Cy233 and have been output as a serial message, with a prefixed Read message header consisting of an ASCII 'R' followed by a two byte hex-ASCII address. Assume in the following that the Network Cy233 with address 3 is attached to the Cy325 LCD controller that is generating the soft-key messages.

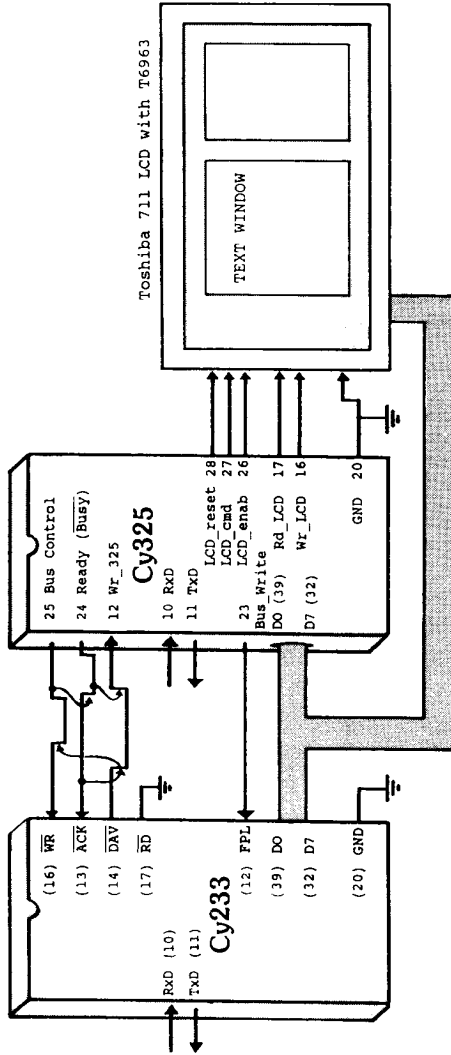
Cy325 Soft-Key selected	Network Soft-Key message	Network Verify message	actual Key_image pattern
1	'R03#1<cr>'	'R03?7Eh<cr>'	01111110
2	'R03#2<cr>'	'R03?7Dh<cr>'	01111101
3	'R03#3<cr>'	'R03?7Bh<cr>'	01111011
4	'R03#4<cr>'	'R03?77h<cr>'	01110111
5	'R03#5<cr>'	'R03?6Fh<cr>'	01101111
6	'R03#6<cr>'	'R03?5Fh<cr>'	01011111

After the Soft-key #n message has been correctly acknowledged via the appropriate 'K n<cr>' command from the host, the result of a following Query of the Key\_image, via the '? 4<cr>' query command should be 'R03?FFh<cr>'.

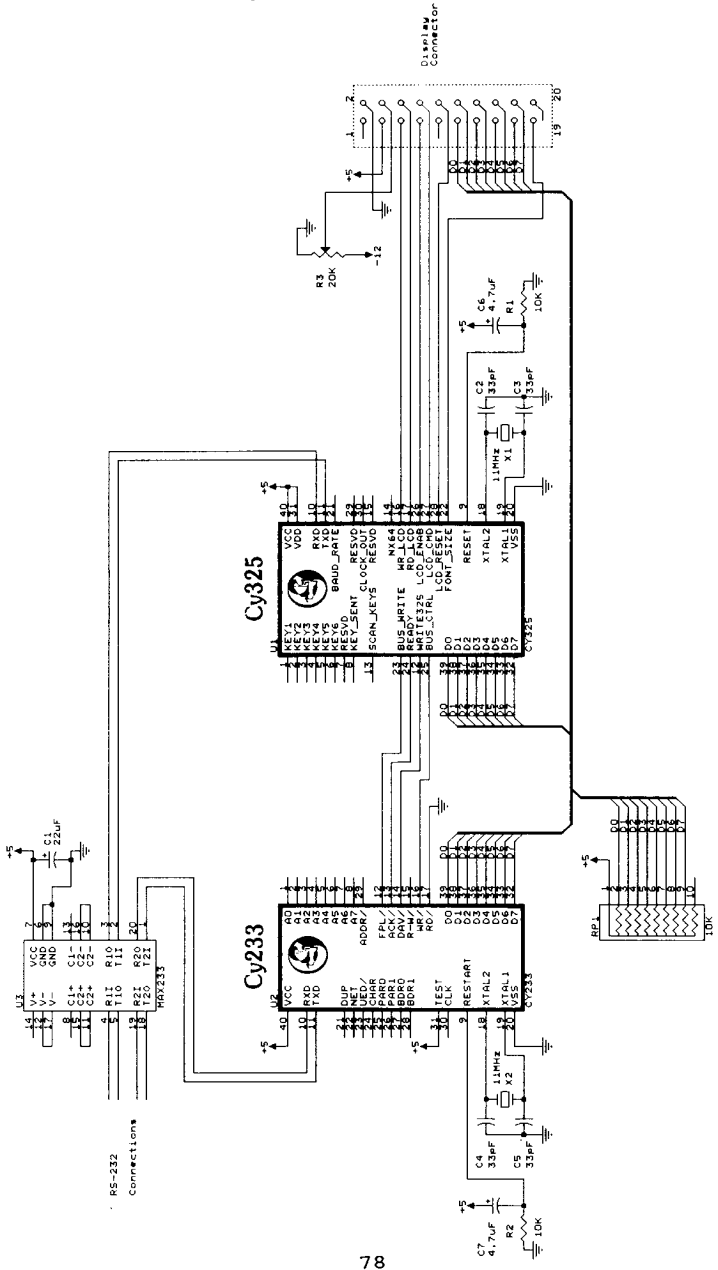
Notice that the messages appearing on the Network serial bus are sufficiently encoded to uniquely identify every soft-key in the network. The 'R' header and <cr> terminator properly frame each message, while the network address, 03, uniquely identifies which Cy233-Cy325 station is generating the message, and the '#n' info identifies soft-key #n, while the '?xxh' information identifies the response to the query of the Key\_image mode register. More information on the Cy233 is available from Cybernetics.

# Cy325 - Cy233 Interface Diagram

## CY233/CY325 Interface



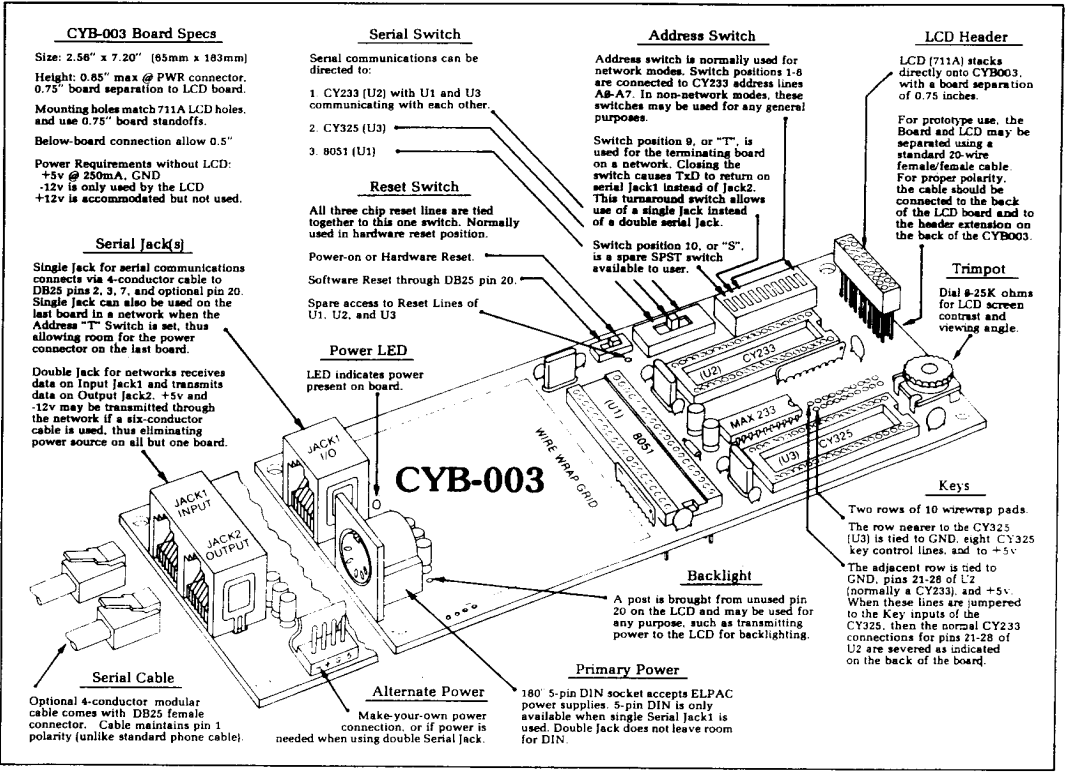
# Cy325 - Cy233 Schematic Diagram



Cybernetic Micro Systems	
Title	CY225 to CY233 Schematic
Size	Document Number
REV	B
Date	Rev. Cl. 15, 1988 Sheet 1 of 1

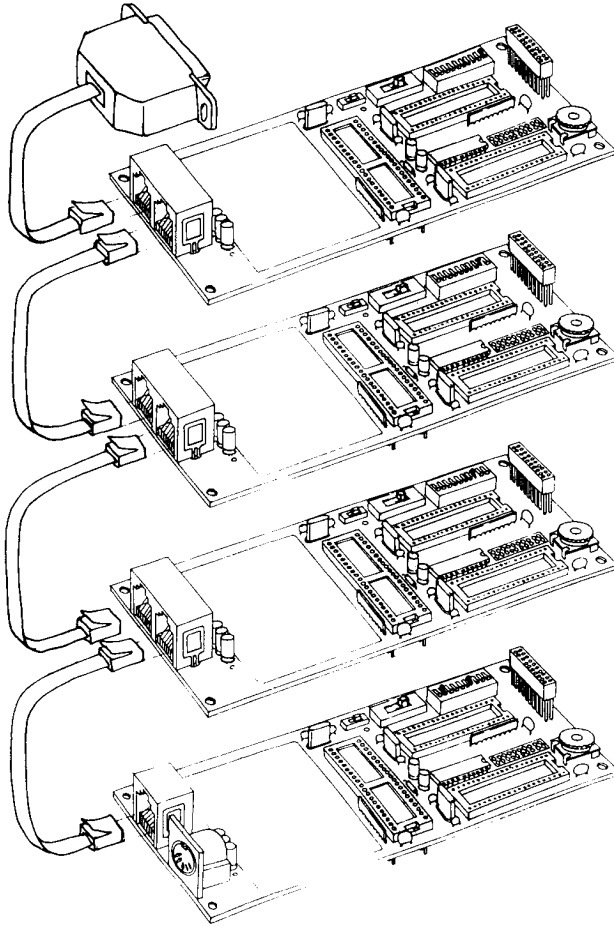
# Cyb-003 Prototyping Board for Cy325 & Cy233

A multi-purpose prototyping board for the Cy325, a Cy233, and an 8051-type 40 pin integrated circuit is available from Cybernetics with a form factor that exactly matches the AND 711A LCD display.



## Cyb-003 Prototyping Board in a Network

The Cyb-003 can be used in a serial network. Connections between boards can be via 4-wire cables or 6-wire cables with telephone-type jacks. The 6-wire cables can carry both signals and power as shown below:

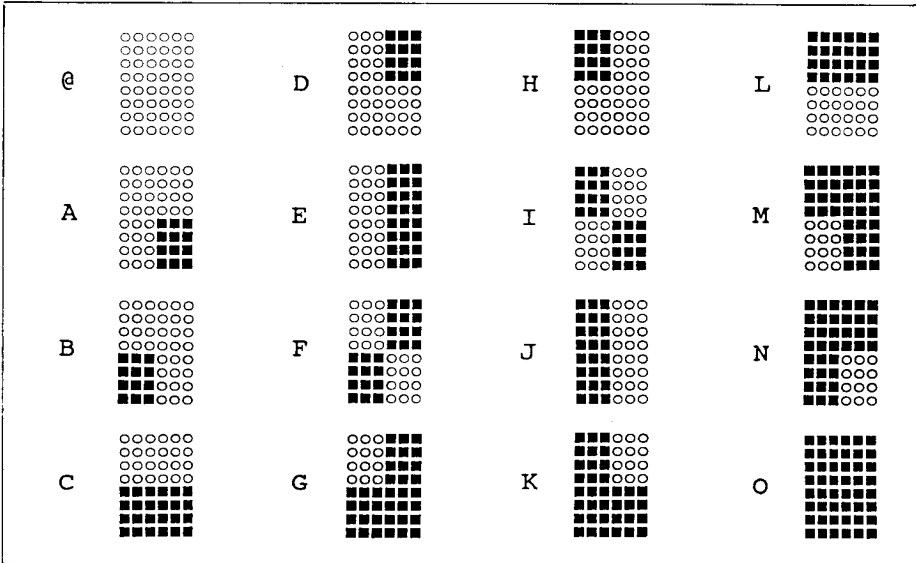




# 14 CY325 Built-in 'Special' Font 14

The Cy325 allows the user to design any number of special fonts and download these, character by character, via the Font command. When the Cy325 powers up (or after I 4) it will contain fifteen special font characters that can be used to form Giant characters that are visible from twenty or thirty feet (ten meters) away, as is required for some wall mounted displays. These characters are much larger than the 'Large' characters that are invoked via the 'L'-command. 'Large' characters are formed from 2x2 arrays of the special font characters, while 'Giant' characters are usually four or more rows high. The Cy325 treats Large characters as discrete entities, with cursor management, window wrap, etc., while Giant characters are managed entirely by the user. Giant characters are created by the user, and will generally be written as a string of special characters that 'paints' a stripe (row) of the giant character, then the next stripe or row is written as a second string, after adjusting the cursor, and so on.

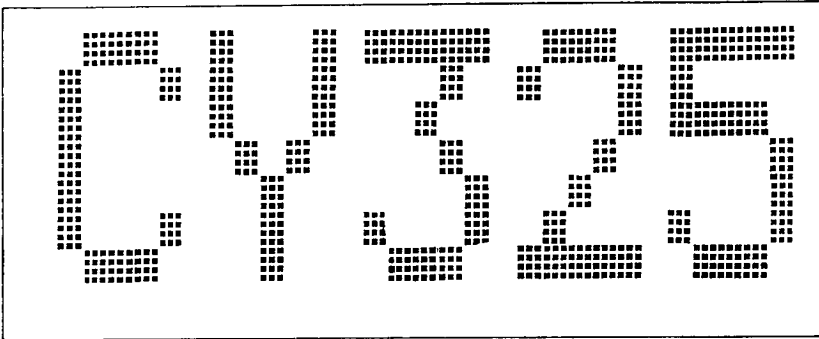
The built-in special font effectively provides "giant pixels" that are 4x4 normal pixels in size, thus a special font character will contain four pixels. Of the sixteen combinations possible with four pixels, the one with all pixels off is identical to the ASCII blank character, 20h. The fifteen remaining combinations are shown below. These can be used to build any characters in the same manner that small pixels are used to create characters. The primary difference is that characters built of giant pixels are visible from across a room.



## Examples of 'Special' Font

The default special font patterns can represent four large pixels to form "characters" four times the size of normal letters. Such characters use 3x4 special font patterns for 6x8 large pixels. Thus 12 characters are used to create each super character. The example below shows how to form the giant letters "CY325".

Ctrl-C	enter Command mode
I 1<cr>	erase characters in current window
Ctrl-D	enter Display mode
Ctrl-N	select special font
<cr>	1st col, 2nd row in window
ALI E@E DLN ALI ELL <cr>	1st & 2nd rows of giant CY325
E@E DBF @DB @@F DLI <cr>	3rd & 4th rows of giant CY325
E@A @EE A@E @F@ A@E <cr>	5th & 6th rows of giant CY325
@LH @D@ @LH DLL @LH <cr>	7th & 8th rows of giant CY325
Ctrl-O	shift back to normal font
<cr>	perform carriage return, line feed
CY325 <cr>	print normal size "CY325" text



The 'shift-out' code, Ctrl-N, allows special font patterns to be selected via 7-bit ASCII character codes. Special font patterns can be displayed via 8-bit character codes, without the Ctrl-N shift code. The 8-bit code corresponding to '@' is 164 or 0A4h. The Cy325 default special characters can be selected via the shift-out 7-bit ASCII characters '@' thru 'O' or via the 8-bit character codes with values 164..179 or 0A4h..0B3h. Although the values are specified in Hex form, the characters must be sent to the Cy325 as one 8-bit character code. If the 'Font' command has been used to redefine the default font, the I 4<cr> command can be used to reload the default font. You can extend the special font while retaining the default set by using the 'F' command with character value greater than 15. 'F' commands with character values below 16 redefine the default font patterns.

## Application Note 1:

### LCD Windows Controller Speeds Instrument Design

The size, cost, and appearance of "instrument size" LCD displays has finally reached the point where most designers of new instruments and other systems are considering them as the primary user interface. The new "instrument" LCDs are distinguished from "laptop PC" LCDs in several ways. While "laptop PCs" usually require 640 x 200 pixel displays to be compatible with DOS software, such LCDs are generally too large and too expensive (\$200) for instruments, handheld units, and point-of-sale terminals.

**Instrument size LCDs** provide 240 x 64 pixel graphics 8 rows x 40 characters, or smaller. Costing around \$60 in quantity, they are ideal for instruments and display panels designed to display status, waveforms, and to display clear, easy-to-read messages, as opposed to pages of text, as in word processors, etc. They are smaller in size than PC-type LCDs and are available in supertwist with back-lighting. While in many respects they are ideal for instruments, they are not readily supported by appropriate LCD controllers. In fact, there is no standard LCD controller, and the very low level controllers available from LCD manufacturers are extremely difficult to program. So the good news that super attractive graphic displays are available is offset by the need for programming one more complex peripheral! Thus, as is often the case, hundreds or thousands of instrument designers, wishing to provide the best user interface, will have to take time away from instrument design and instead devote many man-months to developing routines which display text, plot curves, draw boxes, create icons, build bargraphs, show logic waveforms, support serial communications, and so forth. In other words, instrument designers all over the world will be "re-creating the wheel" in order to utilize the attractive new displays.

To minimize such "wheel re-creation", Cybernetic Micro Systems designed the new Cy325 LCD Windows Controller specifically for instrument designers (including point-of-sale terminals, etc.) with the goal of reducing display design from a good fraction of a man-year down to only man-weeks. The Cy325 can simply provide serial input to the LCD, or it can serve as the major system element, tying together the display function, the user input function, and also providing an interface to the system processor. The Cy325 even has built-in support for RS-232 based local area networks, thus allowing a chain of instruments or Point-of-Sale terminals to be linked to an IBM-PC COM channel or equivalent.

## Text and Graphics in Windows

While text display is generally not too difficult on LCDs, the simple task of plotting a pixel on an LCD can be a major undertaking, involving the solution of several equations required to transform from the desired coordinate system in a box or window, to the display coordinate system as seen by the user, then to the display coordinate system seen by the LCD controller, which is often segmented, and byte oriented, thus requiring bit mask operations on bytes, multiplications and divisions, for every such pixel to be displayed! Since these operations have little or no connection with the purpose of most instruments, they are simply additional headache for the designer, somewhat analogous to the need to calculate carburetor flow rates based on orifice specifications every time you simply want to step on the accelerator. Thus the first task the Cy325 performs for the user is to hide this detail by automatically performing all necessary coordinate transformations, regardless of the position on the LCD display. The partitioning of a display into separate work areas or 'windows' is both functionally and visually desirable, therefore the CY325 has been designed to create and manage windows on graphic LCDs. Each window possesses its own coordinate system with the (0,0) origin located in the lower left corner of the window as expected. While many windows can be open on a screen at one time, the 'current' or 'active' window defines the current coordinate system. Pixel plotting operations map into the current window and the Cy325 even performs 'clipping' based on this window if desired.

Any of 255 default or "built in" windows can be selected via simple commands, or a user defined window can be specified by a single command. Both text and graphics can be written into the current windows with automatic cursor management, clipping, etc. Text and graphics can be independently written, erased, or overlaid in a single window, or text can be written to one window and graphics to another. Windows can be defined within windows. Graphics operations, defined in terms of the "current" window, include automatic histogram generation and logic waveform display. For example, histograms (bargraphs) can be generated by simply specifying the heights of the bars as arguments.

A short summary of Cy325 features includes:

- Command and Display modes of operation
- Serial or 8-bit Parallel interfaces
- Built-in or user-defined Windows
- Window-relative text and graphics
- Logic waveforms built-in (see photo)
- Large characters and user-defined fonts
- Bargraphs automatically size to window
- Communications between Serial and Parallel
- "Soft-key" support for menu management
- Network support based on Cy233 Network chip.

## Menu-based Instrument Programming

In addition to graphic information display, LCD-based instruments often provide menu-based programming functions, so the Cy325 supports "soft-key" operation to provide menu programming capability. The CY325 supports up to six "softkeys" which are keys that use the instrument computer to display the "meaning" of each key on the LCD. When the user pushes a particular key the result is transmitted to the instrument CPU. A simple built-in protocol assures that all keys are uniquely identified and acknowledged, thus allowing the user to make menu choices or otherwise use the softkeys, with the system computer managing the responses and updating the displays.

## Examples of Instrument Display Design

In addition to the simple display of alpha-numeric text, the most common instrument displays consist of:

1. Voltage Waveforms
2. Logic Waveforms
3. Bargraphs
4. Special icons or symbols

While each of these graphic displays is complex and difficult to implement using the low level controllers available with today's LCDs, the Cy325 offers easy-to-use hi-level commands that perform these common display functions. Often one command to the Cy325 will generate hundreds of commands to the low-level (built-in) LCD controller, hundreds of commands that the instrument designer no longer need be concerned with.

In order to demonstrate how the Cy325 can make your life easier, we will design two prototype instruments. The first will use a FIFO to capture hi-speed digital signals and display logic waveforms in a window on the LCD, while the second instrument will be based on an unspecified transducer, with the goal of displaying the transducer output in an LCD window. One instrument will use a serial interface to the Cy325 and the other will use an 8-bit bus parallel interface to the Cy325.

## Instrument #1: Serial Interface to Logic Display

We begin with a very simple design (figure 1) in which the Cy325 is used primarily to display logic waveforms and messages. The system computer will use its own UART to interface serially to the Cy325. Commands will be transmitted serially to the Cy325 to open windows and display messages, while the logic waveforms will drive six I/O pins on the Cy325. Assume that the instrument CPU is an 8051, although any processor will do. The instrument system processor will perform the following tasks:

1. Setup the Cy325 operating mode
2. create the message window
3. display messages in the window
4. create the logic waveform window
5. capture logic waveforms using FIFO circuits
6. enable the Cy325 Waveform Scan
7. send the logic waveform info to the Cy325

The first task consists of setting the Cy325 baud rate for serial communications and placing the Cy325 in the logic waveform mode of operation. If pin 21 on the Cy325 is pulled hi, the baud rate will be set to 9600, if low, then 2400, and if pin 21 is floating the Cy325 adaptively responds to two carriage returns and selects the baud rate that the carriage returns are transmitted with. The Cy325 automatically powers up with logic waveform mode selected (but not enabled). However, if this has been changed, then a mode command can be issued to select waveform operation. Note that the pins used to input logic waveforms can be used for other purposes such as "soft-key" operation.

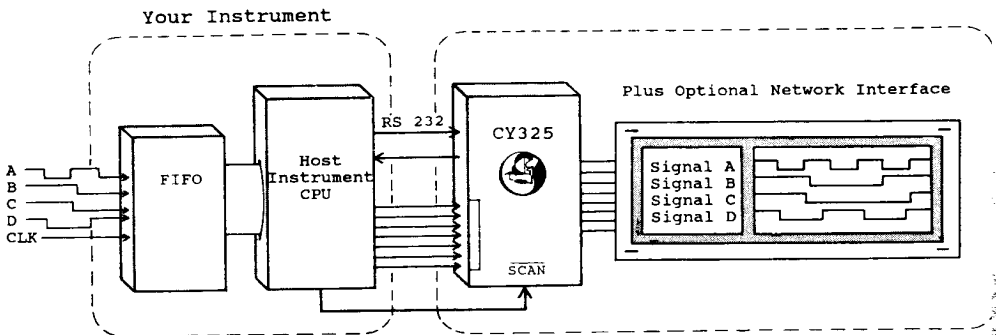


Figure 1. A logic analyzer with serial interface to the LCD.

The message window can be created in two different ways. Since the Cy325 possesses 256 'built-in' windows, it is usually easiest to simply choose an appropriate window or viewport by sending the viewport #n command, 'V n<cr>', where 'V' is the ASCII letter V (56h) followed by a space (20h) followed by either the decimal or hexadecimal number of the window, followed by a carriage return. In the example shown in fig 1 the command 'V 0C6h<cr>' creates the message window that is one fourth of the screen wide and six lines deep. Before sending the viewport command the Cy325 must be placed in the command mode by sending the Ctrl-C character (03h).

After the message window has been created, we wish to display the six messages in the window. We send the Ctrl-D command (04h) to return the Cy325 to the Display mode of operation, then we simply send the six messages:

```
Signal 1 <cr>
Signal 2 <cr>
Signal 3 <cr>
Signal 4 <cr>
Signal 5 <cr>
Signal 6 <cr>
```

The next task is to create the waveform window and this is done by issuing the Ctrl-C command again to enter command mode and by sending the 'V 76h<cr>' command to select the window for the logic waveforms. At this point the LCD screen looks like:

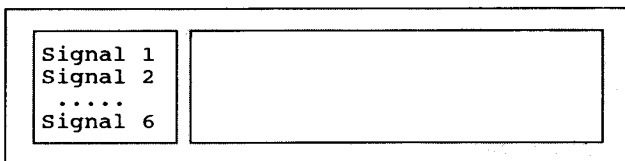


Figure 2. Two viewports have been defined and text messages displayed in the first window.

Our basic premise is that systems designers possess the expertise necessary for measurement systems design, while we are simply trying to relieve them of the tedium of display design at the low level by offering a high level display controller. Thus we assume for the following that the instrument has captured a set of logic waveforms that can be read from the FIFO in parallel.

After reading six (or more) logic signals in parallel, the instrument CPU simply places the six desired signals on the Cy325 waveform pins (1 thru 6) and lowers the Scan\_Enable line (pin 13) to tell the Cy325 to accept and display these signals in the current window, entering them from the right side of the window. The Scan\_Enable line can be left low or pulled high to strobe each change in signals into the Cy325. If there is no change in the signals, pin 7 on the Cy325 can be used to 'clock' the

display, that is, to advance the waveform display one period from right to left with no change in the shape of the waveforms.

In this way an endless sequence of waves can appear at the right of the waveform window and exit at the left of the window. At any time the instrument CPU can disable the scan, change windows and change messages, then return to the waveform window, and continue the waveform display operation. Any of the Cy325's 255 built-in windows or any user specified window can be used to display waveforms, in fact two or more separate waveform windows can be active by carefully controlling the sequence of commands! If the window is less than six lines deep, the Cy325 will automatically truncate the display and only show the number of lines that fit in the window. Various markers or cursors can be placed over the waveforms that do not move relative to the window while the waveforms slide 'under' the marker(s). The markers can of course be moved by commands from the system CPU.

That completes the design of instrument number one. It is worth reviewing the commands necessary to achieve the display shown in figure 1:

```
Select message window:

Ctrl-C           enter Command mode
V 0C6h<cr>      select message window (from 255)
Ctrl-D           re-enter Display mode

Display message:

Signal 1 <cr>    send ASCII messages
Signal 2 <cr>
Signal 3 <cr>
Signal 4 <cr>
Signal 5 <cr>
Signal 6 <cr>

Select Waveform window:

Ctrl-C           enter Command mode
V 76h<cr>        select waveform window (from 255)

Display Waveforms:

place signals on Cy325 pins 1..6
strobe waveform signals into window using pin 13.
```

That's it! Five commands!

Compare the above procedure for generating waveform displays with the low level operations necessary at the pixel level. Months of work are reduced to an hour of coding time or less.



## Instrument #2: Parallel Interface to System Element

Having shown how simple it is to obtain sophisticated displays in a short time, we will now design instrument number 2 to utilize more of the Cy325's built-in system functions. Instead of the serial interface, the system processor will use an 8-bit bus to interface to the Cy325. The Cy325 serial port will then be available to the system processor as a special UART that can be employed to connect other subsystems together or to provide a serial channel to the external world. In addition, the logic pins used in the previous example can either be used as 6 "soft-key" inputs or 8 pins of the Cy325 can be used as general purpose I/O under software control of the system processor. The system as we have described it is shown in figure 3.

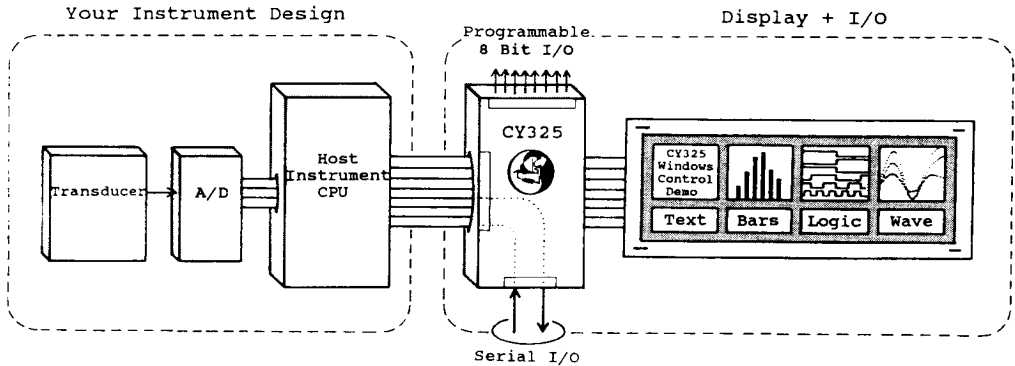


Figure 3. Example Instrument number 2 uses a parallel interface to the Cy325 and maintains four separate windows.

The features that we will concentrate on are the following:

1. The parallel interface to the Cy325
2. The generation of bargraphs
3. The display of analog waveforms
4. The use of the general purpose I/O lines.

## Parallel Interface to the Cy325

The 8-bit bus interface uses a ready signal and a write line with a bus-control signal to enable the system data onto the Cy325 bus where it will be accepted by the Cy325. The timing sequence is shown in figure 4. A fast-bus mode can be selected for systems that can respond in approximately ten micro seconds, however the default operation is designed to allow bus transfers to be completed under software control of the system processor.

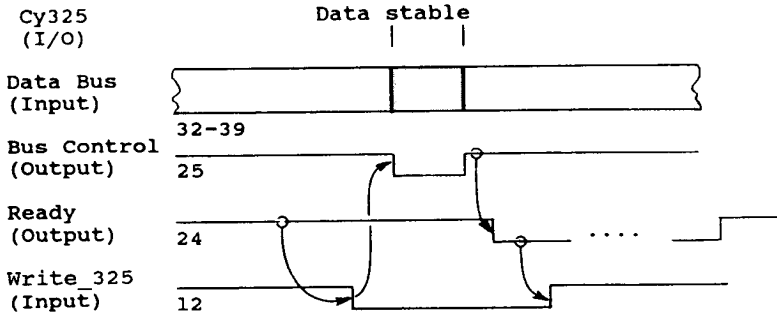


Figure 4. Parallel Input transfer signals and timing sequence.

The instrument processor checks the READY line (pin 24) to see if the Cy325 is ready to accept a character on the data bus. If READY is high the instrument CPU drives the WR\_325 line (pin 12) low to request a transfer. When the Cy325 responds by pulling the BUS\_Ctrl line (pin 25) low, the CPU places the 8-bit data on the bus. The Cy325 acknowledges receipt of the data by returning the BUS\_CTRL line high and driving the READY line low. The CPU should float the bus and wait until the READY line returns high to repeat the sequence for the next transfer.

After the parallel interface has been implemented, the Cy325 is controlled by sending commands as in the serial design. The Cy325 powers up in the Display mode of operation with a default window equal to the whole screen. Any characters sent to the Cy325 will then be considered to be ASCII characters and displayed on the screen accordingly. While this is very convenient for the simplest systems, the usual startup procedure will be to issue the Ctrl-C command to enter the command mode and then to create the appropriate windows and menus or messages. Ctrl-C enters Command mode, Ctrl-D re-enters Display mode. In command mode characters are interpreted, in display mode characters are simply displayed at the current cursor location in the current window. The cursor is managed automatically in that characters that hit the right side of a window automatically advance to the left side on the next line. If the current line is the bottom line, then the text in the window will either wrap around to the top, or scroll up from the bottom, depending on the setting of the Scroll/wrap flag in the window status register.

## Automatic Bargraph Generation

Since Bargraphs are universally understood diagrams, they provide a popular means of displaying information. For this reason the Cy325 supports a bargraph or Histogram command that has the following format:

```
'H n, y1, y2, y3, .. yn <cr>'
```

where

H is the Histogram command opcode (ASCII 'H' = 48h) followed by an ASCII space (= 20h)

n is the decimal or hexadecimal number of arguments followed by a comma.

y1 is the first argument value (decimal or hex) followed by a comma.

yn is the last argument value, followed by a <cr>.

The 'H' command tells the Cy325 how many bars are to be plotted in the current window, and specifies the height of each bar. The Cy325 then scales the independent axis in such a manner that the bars are equally distributed, and draws the bars to the specified height. The bars may be drawn either horizontally or vertically, depending on the value of the appropriate mode bit in the window status register. In fact bars can be drawn vertically in one window and horizontally in another. The command sequence:

```
Ctrl-C              enter command mode
V 0D6h<cr>         select window for bargraphs
H 9,5,9,15,25,36,45,20,16,7<cr>
  \                   \
                           plot 9 bars of specified height
```

will generate the display shown in figure 6.

## The Display of Analog Waveforms

Since our example instrument is presumed to sample voltages from a transducer, we now investigate the window-relative display of analog waveforms. The Cy325 provides two primary PLOT commands, a Plot-Point command and a Plot-String command. The Plot-Point command has the format P x,y <cr> where (x,y) is the coordinate of the pixel to be plotted. For plotting multiple pixels such as those obtained by repeated sampling of an A-to-D converter, the Plot-String command should be used.

The **Plot-String** command has the following format:

where            '(' n y1 y2 y3 ... yn  
                 '(' is the ASCII Plot-String opcode.  
                 n is the binary number of points to follow.  
                 yn is the binary value of the nth data point.

To plot an analog waveform in the current window, the instrument CPU simply sends the '(' character ( = 07Bh ) followed by the binary number of points that are to be plotted. After these two bytes have been received, the binary data points are transmitted. The first data point, y1, will be plotted at the local coordinate (1,y1) while the second will appear at (2,y2) until the nth data point appears at (n,yn). The host can sample n values and send them all at once or simply send the points as they are sampled, however the Cy325 expects all n points to be sent before another command can be issued. The Plot-String uses a data count instead of a <cr> terminator.

The LCD screen shown in figure 6 illustrates the results of the above operations (including another logic waveform). Less than a dozen commands were required to generate this display!

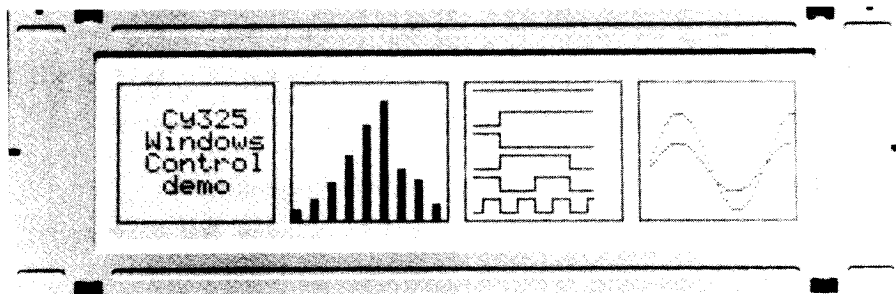


Figure 6. An example display achieved with a dozen commands.

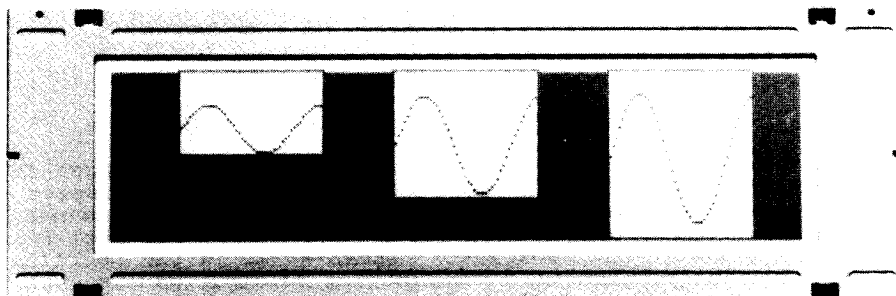


Figure 7. Illustrating several analog waveform displays.

The Cy325 possesses eight pins that are multi-purpose. Their use for 'soft-key' menu operation will be described in a following article that shows how up to 255 Cy325s can be connected in a serial network connected to an IBM-PC's COM1 channel. In the current example, these multi-purpose pins simply provide eight general purpose I/O lines. Cy325 commands exist that will allow the user to individually set or reset each of the 8 I/O lines and also to write an 8-bit byte to the lines in parallel.

## **Instrument Design Summary**

The two example instruments designed above have focused on the ease with which an instrument processor can be coupled to an LCD, using either parallel or serial interface, and on the ease with which sophisticated displays, requiring hundreds or thousands of pixel operations, can be generated using only a few commands. The measurement portions of the instruments have been fairly unspecified, since most digital and analog measurements require the same general types of displays, that is, bargraphs, alphanumeric messages and logic or analog waveforms.

The Cy325 possesses several other display functions that are very useful, in particular the ability to create special fonts for icons or other application specific images. As in the other display cases, only one Cy325 command is required to define a new symbol or icon, and the Cy325 can even create special double-size symbols as a combination of four such user defined symbols. Still another useful feature supported by the Cy325 is the following: Each window can be considered to consist of a graphics plane overlaying a text plane, and each of these planes can be manipulated in several ways within a window.

Finally, the CY325 possesses a large number of special modes of operation that can be enabled or disabled by setting or clearing a bit in the appropriate mode register. More than two dozen mode bits allow hundreds of variations to be easily specified.

## **Network Interface and Soft-Keys**

The Cy325 provides both TTL parallel and serial interfaces. In addition, the CY325 provides a convenient interface to Cybernetic's CY233-LINC Local Intelligent Network Controller. This allows up to 255 LCDs to be attached to a single serial I/O port (such as COM1 of an IBM-PC). The CY325 is designed to support up to six "softkeys" in such a network. This allows a host computer to display the "meaning" of each key on the LCD. When the user pushes a particular key, the result is transmitted to the host. A simple built-in protocol assures that all keys are uniquely identified and acknowledged, thus allowing numerous users to make menu choices or otherwise use the softkeys, with a central host managing the responses and updating the displays. This feature allows applications such as customer order entry, etc., to be implemented at low costs.

## **Application Note 2:**

### **The CY325 in a CY233 Based Local Area Network**

## Application Note 3:

### Demonstration Programs for the CY325

```
$Title ...LCDrvr.250 code

dICE51      = 0

Bseg ; 8051 Boolean Bit segment
;
IO_request  bit P3.4 ; Port 3, bit 4, driven by demo
Busy_Ready  bit P3.3 ; Port 3, bit 3, driven by Cy325
Bus_Control bit P3.6 ; driven by Cy325 - put data on Bus
next_cycle  bit P3.7 ; push button for photographer
Bus_Write   bit P3.2 ; driven by Cy325 to send to demo
;
endS ; end of Boolean Bit declarations

Dseg ; -- declare Data Segment of 8051 memory --
;
Org 30h

Buffer: DS 10h      ; Define Storage for data buffer
buffer_length = 10h
dead_zone data $    ; (buffer overflow debugging)
Time_Counter data $ ; count "clock ticks"
DPLsave data $      ; save Dptr low and hi
DPHsave data $
wave_cnt data $     ; wave port image
wave_loop_cnt data $
Demo_number data $  ; index into Demo table

Org 70h
Stack = $ ; Working Stack area
;
endS ; declare end of Data segment

Fseg ;-- Function Segment of 8051 memory --
;
DATA_port data P0 ; = data bus
key_port data P2
;
endS ; end of function register segment

CtrlC = 3 ; Command enable
CtrlD = 4 ; Display enable
CtrlK = 0Bh ; Klear window
CtrlN = 0Eh ; shift in
CtrlO = 0Fh ; shift out
CtrlW = 17h ; swap windows
CR = 0Dh ; define ASCII carriage return value

Last_Demo = 22 ; number of demos in table
```

```

Cseg  ;-- declare Code Segment of 8051 memory --
;%S   Special Start marker used by ICE-8051

Org 0
;-----

                Start_Code:  ;: 8051 Reset

;-----
mov SP, #Stack-1 ; setup working stack
mov DATA_port, #0FFh ; remove any data from bus
mov key_port, #0ffh ; select logic mode
setb Busy_Ready
mov r4, #100
acall mini_delay ; while Cy325 powers up !
mov Demo_number, #0 ; reset
acall send_CtrlC
mov Dptr, #blank_cursor
acall send_messages
acall delay

Backgnd_Loop:  ; branch thru table and return
;
; acall next_demo
;
sjmp Backgnd_Loop

;-----
; next_demo:
;-----
mov a, Demo_number
inc Demo_number
mov Dptr, #Demo_table
rl a
jmp @A + Dptr

Demo_table:
ajmp do_Cy325_SignOn
ajmp show_Cy325_Cmds
ajmp do_4_Windows
ajmp do_Large_chars
ajmp do_Box_sets
ajmp do_Boxes_2
ajmp do_Histo_stuff
ajmp do_Histo_in_Boxes
ajmp do_Sine_stuff
ajmp do_Sines_in_Boxes
ajmp do_waves
ajmp make_waves_again
ajmp do_pat_and_rub
ajmp do_Cy325_SignOff
mov Demo_number, #0 ; restart demo
ret

```



```

;-----
do_4_Windows:
;-----
acall blank_screen
mov Dptr, #Four_Windows
acall msg_n_delay
acall wave_in_window
mov Dptr, #Sine_setup
acall send_messages
mov Dptr, #Sine_3_setup
acall send_messages
mov Dptr, #Sine_wave_10
acall send_sines
mov Dptr, #Sine_o20_setup
acall send_messages
mov Dptr, #Sine_wave_20
acall send_sines
mov r4, #180
acall mini_delay
ret

Four_windows:
db CtrlC

; display message

db 'v 0C6h',cr
db 'I 2',cr
db CtrlD
db ' ',cr
db ' Cy325',cr
db ' Windows'
db ' Control' ; ,cr
db ' demo',cr

; display histogram

db CtrlC
db 'v 0D6h',cr
db 'I 2',cr
db 'H 9,5,9,15,25,36,45,20,16,7',cr

; display logic waves

db 'v 0E6h',cr
db 'I 2',cr,'$'

; display sine waves

Sine_setup:
db 'v 0F6h',cr
db 'I 2',cr,'$'

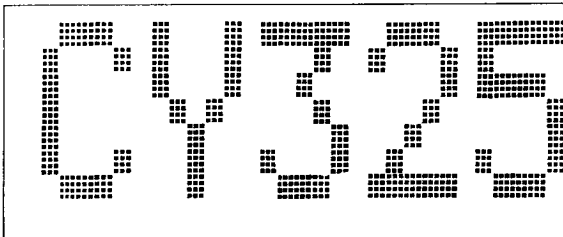
```

```
;-----
```

```
do_Large_chars:
```

```
;-----
```

```
acall blank_screen  
mov Dptr, #Large_char_setup  
acall msg_n_delay  
mov r4, #180  
acall mini_delay  
ret
```



```
Large_char_setup:
```

```
db CtrlC  
db 'v 46h',cr  
db CtrlD  
db CtrlN,CR  
db '@@FLB@H@@@@@@@@E@',CR  
db '@@JCBDJ@DLBKLBMH@',CR  
db '@@J@J@J@FLJ@J@E@B',CR  
db '@@DLHDL@DLHH@H@L@',CR  
db CtrlO  
db ' Giant',CR  
db CtrlC  
db 'v 56h',cr  
db CtrlD  
db CtrlN,CR  
db '@@ALIE@EDLNALIELL',CR  
db '@@@@DBF@DB@@FDLI',CR  
db '@@@@A@@@A@@F@A@E',CR  
db '@@@LH@D@@LHDLLE@LH',CR  
db CtrlO  
db ' CY325',CR, '$'
```

```
; open Viewport  
; Display mode  
; shift out
```

```
; shift in
```

```
; new viewport  
; Display mode  
; shift out
```

```
; shift in
```

```
;-----
```

```
; perform simultaneous Vertical and Horiz  
; Scrolls in two windows..
```

```
;
```

```
;-----
```

```
do_pat_and_rub:
```

```
;-----
```

```
mov Dptr, #Swap_um  
acall send_messages  
mov wave_loop_cnt, #8  
mov Dptr, #Slide_msg
```

```
setup_pat_n_rub:
```

```
mov r4, #16  
acall mini_delay  
mov Dptr, #Slide_msg  
djnz wave_loop_cnt, pat_and_rub  
mov Dptr, #undo_slide  
acall msg_n_delay  
ret
```

```

;-----
pat_and_rub:
;-----
;
;   get next sliding char
;   and send to right/slide
;
clr a
movc a, @a+Dptr
cjne a, #'#', send_em
sjmp setup_pat_n_rub

send_em:
acall send_to_LCD
mov a, #CtrlW      ; Swap Windows !
acall send_to_LCD
;
;   get next scrolling char
;   and send to left/scroll
;
mov a, #(Scroll_msg - Slide_msg)
movc a, @a+Dptr
inc Dptr
acall send_to_LCD
mov a, #CtrlW      ; Swap Windows !
acall send_to_LCD
sjmp pat_and_rub

        Swap_um:
db CtrlC          ; setup window stack !
db 'v 46h',cr     ; enter Command mode
db '/I 2',cr      ; create window
db 'I 1',cr       ; fill window
db 'I 1',cr       ; erase characters
db 'I 2',cr       ; erase graphics in window
db 'M 0,17h',cr   ; set scroll flag
db '+',cr         ; push window onto stack
db 'v 56h',cr     ; create 2nd window
db '/I 2',cr      ; fill window
db 'I 1',cr       ; erase characters
db 'M 0,16h',cr   ; clr scroll flag
db 'I 2',cr,'z',cr ; erase graphics
db CtrlD,cr,'$', ; enter Display mode

;-----
        Undo_slide:
db CtrlC,cr
db CtrlC,'-',cr  ; pop window stack !
db 'M 0,16h',cr ; clr scroll mode (wrap!)
db '/z',cr,'$', ; clear horizontal scroll

```



```

;-----
  do_waves:
;-----
acall blank_screen
mov Dptr, #Wave_Window_1
acall msg_n_delay
mov wave_loop_cnt, #80
sjmp make_waves

make_waves_again:
mov Dptr, #Wave_window_2
acall send_messages

wave_in_window:
mov wave_loop_cnt, #36

make_waves:
inc wave_cnt
mov a, wave_cnt
;
  acall make_wave
;
mov r4, #6 ; select time delay constant
acall mini_delay
djnz wave_loop_cnt, make_waves
ret

```



```

make_wave:
mov A, #'I'
acall send_to_LCD ; send 'I 12,0xxh<cr>'
mov A, #' '
acall send_to_LCD
mov A, #'1'
acall send_to_LCD
mov A, #'2'
acall send_to_LCD
mov A, #','
acall send_to_LCD
mov a, wave_cnt
and A, #03fh ; only drive lower 6 bits
acall send_Param ; convert to hex & send
mov A, #CR
acall send_to_LCD ; end with <cr>
ret

```

```

;-----
; send_Param: ; value in Acc as hex
;-----
mov R2,A ; save value
mov A,#'0'
acall send_to_LCD ; always send 0xxh
mov A,R2
swap A
acall Bintohex ; convert nibble to ASCII hex
acall send_to_LCD
mov A,R2
acall Bintohex ; now LS nibble
acall send_to_LCD
mov A,#'h' ; end of parameter
acall send_to_LCD
mov A,R2 ; restore Acc
ret

```

```

Bintohex: ; convert to ASCII hex
anl A,#0Fh ; lower nibble only
add A,#30h ; numeric conversion
cjne A,#3Ah,$+3 ; CY set if A < 3Ah
jc BintohexRet ; number is 0 to 9
add A,#07h ; else make A to F
BintohexRet:
ret

```

```

Wave_Window_1:
db CtrlC,'v 0',cr ; create window 0
db 'I 1',cr ; erase charactes
db 'v 46h',cr ; create left window
db CtrlD ; enter Display mode
db '
db ' LOGIC Waveforms '
db ' can be driven '
db ' by 6 I/O pins '
db ' on the Cy325 '
db ' ',cr
db CtrlC
db 'v 56h',cr ; create right window
db 'I 2',cr,'$' ; clear graphics

```

```

wave_window_2:
db CtrlC
db 'v 56h',cr,'I 1',cr; erase right window
db CtrlD ; enter Display mode
db ' ; display message
db ' LOGIC Waveforms '
db ' in any Window '
db ' can be driven '
db ' by 6 I/O pins '
db ' on the Cy325 ',cr
db CtrlC,'I 2',cr ; erase window
db 'v 46h',cr ; select left window
db 'I 1',cr,'$' ; clear window

```

```

;-----
do_Sine_stuff:
;-----
mov Dptr, #Sine_Over_setup
acall send_messages
mov Dptr, #Sine_wave_10
acall send_sines
mov Dptr, #Sine_o20_setup
acall send_messages
    mov Dptr, #Sine_wave_20
    acall send_sines
mov Dptr, #Sine_o30_setup
acall send_messages
    mov Dptr, #Sine_wave_30
    acall send_sines
mov r4, #50
ajmp mini_delay
;-----
do_Sines_in_Boxes:
;-----
mov Dptr, #Sine_10_setup
acall send_messages
mov Dptr, #Sine_wave_10
acall send_Sines
mov Dptr, #Sine_20_setup
acall send_messages
    mov Dptr, #Sine_wave_20
    acall send_Sines
mov Dptr, #Sine_30_setup
acall send_messages
    mov Dptr, #Sine_wave_30
    acall send_Sines
mov r4, #15
ajmp mini_delay
;-----

send_Sines: ; enter with Dptr -> Sine args
;-----
mov r6, #52 ; counter for data points to LCD
mov a, #'(' ; 'Plot_string' opcode to Cy325
acall send_to_LCD ; send opcode to Cy325

send_args: ; send count, then data points
clr a ; zero offset from pointer
movc a, @a+Dptr ; fetch byte thru pointer
inc Dptr ; advance pointer
acall send_to_LCD ; send byte to Cy325
djnz r6, send_args ; loop til all data sent
ajmp delay ; delay for observation

```



```
Sine_10_setup:
db CtrlC,'V 0',cr,'/I 2',cr,'I 1',cr ; setup window 0
db 'B 24,63,74,32',cr,'I 2',cr,'$' ; draw BOX on LCD
```

```
Sine_20_setup:
db CtrlC
db 'B 98,63,148,16',cr,'I 2',cr,'$' ; BOX command
```

```
Sine_30_setup:
db CtrlC
db 'B 2,63,222,1',cr,'I 2',cr,'$' ; BOX command
```

```
Sine_Over_setup: ; setup for 'Sines'
db CtrlC ; , 'V 0',cr,'/I 2',cr ; fill LCD screen
db 'V 66h',cr,'/I 2',cr ; setup window
db CtrlD ; enter Display mode
db ' ',cr
db ' Plotting can be either',cr
db ' Global or Window relative',cr
db ' ',cr
db ' P x,y plots points (x,y)',cr
db ' ',cr
```

```
db CtrlC,'I 2',cr ; erase graphics
db 'B 184,62,238,2',cr ; draw BOX
db 'I 2',cr ; erase graphics in box
```

```
Sine_3_setup:
db CtrlC
db '/B 186,48,236,16',cr,'$' ; draw BOX
```

```
Sine_o20_setup:
db CtrlC
db '/B 186,56,236,8',cr,'$'
```

```
Sine_o30_setup:
db CtrlC
db '/B 186,60,236,4',cr,'$'
```





```
;-----  
do_Box_sets:
```

```
;-----  
acall blank_screen  
mov Dptr, #Boxes_1  
acall msg_n_delay  
mov Dptr, #Boxes_2  
acall msg_n_delay  
mov Dptr, #Boxes_3  
ajmp msg_n_delay  
;-----
```

```
do_Boxes_2:
```

```
;-----  
acall blank_screen  
mov Dptr, #Boxes_4  
acall send_messages  
mov Dptr, #Histo_1  
acall send_messages  
mov Dptr, #In_Box_4  
acall send_messages  
mov Dptr, #Histo_1  
ajmp msg_n_delay
```



```
;-----  
send_the_string: ; til <cr> sent  
;-----
```

```
clr a ; zero offset from pointer  
movc a, @a+Dptr ; get byte thru pointer  
inc Dptr ; advance pointer  
;  
acall send_to_LCD ; send byte to Cy325  
;  
cjne a, #CR, send_the_String ; do til <cr>  
;-----
```

```
send_messages:
```

```
;-----  
clr a ; zero offset from pointer  
movc a, @a+Dptr ; pick up byte thru ptr  
cjne a, #'$', send_the_string ; repeat til '$'  
ret ; return after sending all messages..
```

```
send_CtrlC:  
mov a, #3 ; Ctrl-C_ommand  
sjmp send_n_CR
```

```
send_CtrlD:  
mov a, #4 ; Ctrl-D_isplay
```

```
send_n_CR:  
acall send_to_LCD  
mov a, #CR  
ajmp send_to_LCD
```

```

;-----
msg_n_delay:
;-----
acall send_messages
acall delay
ret

msg_delay_n_blank: ; send message to LCD
acall msg_n_delay ; then delay awhile
; then blank LCD screen.

blank_Screen:
mov Dptr, #LCD_blank ; commands to blank LCD
ajmp send_messages ; then return to caller
;-----
; Delay routines
;-----
Delay:
;-----
mov r4, #15 ; nominal delay constant
;-----
mini_delay:
;-----
IF dICE51
ret
endif
mov r5, #0
mov A,R4
cpl A ; complement count
mov DPH,A ; transfer count to DPTR
mov A,R5
cpl A ; use a 16 bit argument
mov DPL,A
inc DPTR ; count now set

Delay_100_us: ; 100 usec delay loop
mov R7,#46 ; loop count for 100 usec delay
nop ; tweak for 100 cycles per loop

D100_us:
djnz R7,D100_us ; wait 100 usec
inc DPTR
mov A,DPH ; test delay count
orl A,DPL
jnz Delay_100_us ;? More Delay
ret

;-----
show_Cy325_Cmds:
;-----
mov Dptr, #Cy325_Cmds_1
acall send_messages
mov Dptr, #Cy325_Cmds_2
acall send_messages
mov r4,#150
sjmp mini_delay ; then ret

```



```

;-----
;
; IO driver to send (ac) to LCD - Cy325
;
;-----

    send_to_LCD: ; called with data in ac

;-----
jnb Busy_ready, send_to_LCD ;?I/O Req Hi
clr IO_request ; signal LCD we have data

wait_for_BUS:
jnb Bus_Control, wait_for_BUS
;
    mov DATA_port, a
;
wait_for_no_BUS:
jnb Bus_Control, wait_for_no_BUS
;
    mov Data_port, #0FFh
;
wait_data_acked:
;
IF not dICE51
;
    jnb Busy_ready, wait_data_acked
;
endif
;
setb IO_request
ret

;-----
    do_Cy325_signon:
;-----
mov Dptr, #Cy325_setup
sjmp sign_on
;-----
    do_Cy325_signOff:
;-----
mov Dptr, #Cy325_signOff

sign_on: ; display messages on LCD
acall send_messages ; to current window
mov r4, #250 ; time constant for delay
ajmp mini_delay ; go spend time

blank_cursor:
db CtrlC,cr,'M 1,1Ch',cr,'$'

```

```

;-----
do Histo_stuff:
;-----
acall blank_screen
mov Dptr, #Histo_setup
acall msg_n_delay
acall do_Histo_in_4
mov r4, #50
ajmp mini_delay
;-----
do_Histo_in_Boxes:
;-----
mov Dptr, #Histo_close_up
acall send_messages
mov Dptr, #Box_1
acall send_messages
mov Dptr, #Histo_1
acall send_messages
mov Dptr, #Box_2
acall send_messages
mov Dptr, #Histo_1
acall send_messages
mov Dptr, #Box_3
acall send_messages
mov Dptr, #Histo_1
acall send_messages

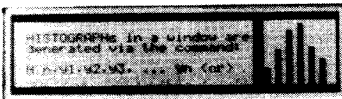
do_Histo_in_4: ; separate entry point !
mov Dptr, #V4_Histo_6
ajmp send_messages
;-----
Histo_setup:
;-----
db CtrlC ; , 'V 0', cr, 'I 2', cr
db 'V 66h', cr ; setup window
db CtrlD
db ' ', cr
db ' HISTOGRAMs in a window are'
db ' generated via the command:', cr
db ' ', cr
db ' H n,y1,y2,y3, ... yn <cr>', cr
;
db CtrlC, 'I 2', cr, 'v 0F6h', cr, 'I 2', cr, '$'

Histo_close_up:
db 'v 0F6h', cr
db 'V 66h', cr, '/I 2', cr, 'I 1', cr, '$'

V_9:
db 'v 31h', cr, '$'

Wset_1:
db CtrlC, 'v 0', cr
db 'v 31h', cr, '$'

```



```
Wset_3:
db CtrlC,'v 0',cr
db '/I 2',cr
db 'v 21h',cr ; was v 10
db 'I 2',cr
db 'v 0CCh',cr
db 'I 2',cr
db 'v 0DCh',cr
db 'I 2',cr
db 'v 0ECh',cr
db 'I 2',cr
db 'v 0FCh',cr
db 'I 2',cr,'$'
```

```
Wset_4:
db CtrlC,'v 0',cr
db '/I 2',cr
db 'v 44h',cr ; was v 19
db 'I 2',cr
db 'v 54h',cr ; was v 20
db 'I 2',cr
db 'v 0CCh',cr
db 'I 2',cr
db 'v 0DCh',cr
db 'I 2',cr
db 'v 0ECh',cr
db 'I 2',cr
db 'v 0FCh',cr
db 'I 2',cr,'$'
```

```
Wset_5:
db CtrlC,'v 0',cr
db 'v 86h',cr
db 'v 96h',cr
db 'v 0A6h',cr,'$'
```

```
Wset_6:
db CtrlC,'v 0',cr
db 'v 0C6h',cr
db 'v 36h',cr
db 'v 0F6h',cr,'$'
```

```
Wset_8:
db CtrlC
db 'v 0C6h',cr
db 'v 76h',cr,'$' ; was v 7
```

```
Wset_9:
db CtrlC
db 'v 66h',cr ; was v 8
db 'v 0F6h',cr,'$'
```

```

Wset_7:
db CtrlC,'v 0',cr
db '/I 2',cr
db 'v 0C6h',cr
db 'I 2',cr
db 'v 0D6h',cr
db 'I 2',cr
db 'v 0E6h',cr
db 'I 2',cr
db 'v 0F6h',cr
db 'I 2',cr,'$'
;-----
Cy325_Cmds_1: ; show in V 5
;-----
db CtrlC
db 'V 0',cr ; window zero
db '/I 2',cr ; darken screen
db 'I 1',cr ; erase chars
db 'v 46h',cr ; left window
db CtrlD ; Display mode
db 'B Box coordinates '
db 'C Cursor position '
db 'F Font definition '
db 'G Graphics figure '
db 'H Histogram bars '
db 'I Initialization ',cr
db CtrlC,'I 2',cr,'$'

;-----
Cy325_Cmds_2: ; show in right window
;-----
db CtrlC ; enter Command mode
db 'v 56h',cr ; setup right window
db CtrlD ; enter Display mode
db 'K Key interaction '
db 'L Large character '
db 'M Mode specifiers '
db 'P Plot x,y points '
db 'V Viewport number '
db 'W Window location ',cr
db CtrlC,'I 2',cr,'$'
;-----
Cy325_setup:
;-----
db CtrlC,'V 0',cr,'/I 2',cr,'I 1',cr
db 'v 31h',cr
db CtrlD
;-----
db ' ',cr
db 'SIMPLIFY and SPEED UP YOUR DESIGN',cr
db ' ',cr
db 'The Cy325 LCD WINDOWS Controller',cr
db ' with Serial and Parallel inputs',cr
;-----
db CtrlC,'I 2',cr,'$'

```



```

;-----
Cy325_signOff:
;-----
db CtrlC,'v 46h',cr,'/I 2',cr,'I 1',cr
db 'v 56h',cr,'/I 2',cr,'I 1',cr
db 'v 31h',cr
db CtrlD
;
;-----
db ' ',cr
db ' The Cy325 LCD WINDOWS Controller',cr
db ' ',cr
db ' Cybernetic Micro Systems, Inc.',cr
db ' telephone: (415)-726-3000',cr
db ' San Gregorio, California 94074',cr
;
;-----
db CtrlC,'I 2',cr,'$'

```

```
;%E
```

```

;-----
LCD_init:
;-----
db CtrlC,'I 0',cr,'$' ; software reset .
;-----
LCD_blank:
;-----
db CtrlC,'I 3',cr ; clear LCD screen
db 'I 4',cr,'$' ; restore default font

```



```

Fill_zero:
db CtrlC,'V 0',cr,'I 1',cr

```

```

Fill_Cmd:
db CtrlC,'/I 2',cr,'$'
;-----

```

```

Box_1:
;-----
db CtrlC ; enter Command mode
db '/B 5,55,45,35',cr ; send BOX command
db 'I 2',cr,'$' ; erase inside the box
;-----

```

```

Box_2:
;-----
db CtrlC
db '/B 25,33,65,9',cr
db 'I 2',cr,'$'
;-----

```

```

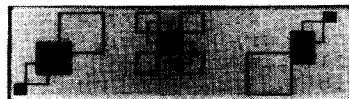
Box_3:
;-----
db CtrlC
db '/B 75,45,105,20',cr
db 'I 2',cr,'$'

```

```

;-----
Boxes_1:
;-----
db CtrlC
db 'B 0,8,8,0',cr
db '/I 2',cr
db 'B 8,24,24,8',cr
db 'B 16,40,40,16',cr
db '/I 2',cr
db 'B 32,63,64,32',cr,'$'
;-----
Boxes_2:
;-----
db CtrlC
db 'B 168,32,200,0',cr
db 'B 200,48,216,24',cr
db '/I 2',cr ; darken inside of box
db 'B 208,56,223,40',cr
db 'B 223,63,232,56',cr
db '/I 2',cr,'$'
;-----
Boxes_3:
;-----
db CtrlC
db 'B 88,63,104,48',cr
db 'B 96,56,112,40',cr
db 'B 104,48,120,32',cr
db '/I 2',cr
db 'B 112,40,128,24',cr
db 'B 120,32,136,16',cr
db 'B 88,32,104,16',cr
db 'B 120,63,136,48',cr,'$'
;-----
Boxes_4: ; 5 boxes dn-up-dn-up-dn
;-----
db CtrlC
db 'B 8,40,48,0',cr
db '/I 2',cr
;-----
; inside 1st box
db 'B 16,32,40,8',cr
db 'I 2',cr
; inside inner box
db 'B 24,24,32,16',cr
db '/I 2',cr
;-----
db 'B 48,63,88,24',cr
db 'B 88,40,128,0',cr
db '/I 2',cr
db 'B 128,63,168,24',cr
db 'B 168,40,208,0',cr
db '/I 2',cr
db '/B 104,32,120,12',cr ; <---*
db 'I 2',cr,'$'

```





```

In_Box_4:
db CtrlC
db 'B 136,56,160,36',cr,'$'

;-----
      Histo_1:
;-----
db CtrlC
db 'H 4,4,14,20,8',cr,'$'

V4_histo_6:
db CtrlC,'v 0F6h',cr
;-----
      Histo_6:
;-----
db CtrlC                ; Command mode
db 'I 2',cr            ; clear window
db 'M 0,16h',cr        ; vertical bars
db 'H 6,13,26,40,48,28,20',cr ; draw Histogram
db 'I 2',cr            ; erase all bars
db 'M 0,12h',cr        ; horizontal bars
db 'H 6,13,26,40,48,28,20',cr ; draw Histogram
db 'I 2',cr            ; erase window
db 'M 0,16h',cr        ; vertical bars
db 'H 6,13,26,40,48,28,20',cr ; draw Histogram
db 'I 2',cr            ; clear window
db 'M 0,12h',cr        ; horizontal bars
db 'H 6,13,26,40,48,28,20',cr,'$'; draw Histogram

```

```

;-----
Sine_Wave_10:
;-----
db 50 ; data points
db 10
db 11
db 12
db 14
db 15
db 16
db 17
db 18
db 18
db 18
db 18
db 18
db 18
db 17
db 16
db 15
db 13
db 12
db 11
db 9
db 8
db 6
db 5
db 4
db 3
db 2
db 1
db 1
db 1
db 1
db 1
db 2
db 2
db 3
db 5
db 6
db 7
db 9
db 10
db 11
db 12
db 14
db 15
db 16
db 17
db 18
db 18
db 18
db 18
db cr

Sine_Wave_20:
db 50 ; data points
db 20
db 23
db 25
db 28
db 30
db 33
db 34
db 36
db 37
db 37
db 37
db 37
db 36
db 35
db 34
db 32
db 30
db 27
db 25
db 22
db 19
db 16
db 13
db 11
db 8
db 6
db 5
db 3
db 2
db 2
db 2
db 3
db 4
db 5
db 7
db 10
db 12
db 15
db 18
db 20
db 23
db 25
db 28
db 30
db 33
db 34
db 36
db 37
db 37
db cr

Sine_Wave_30:
db 50 ; data points
db 30
db 34
db 37
db 41
db 44
db 47
db 49
db 51
db 52
db 53
db 53
db 53
db 52
db 51
db 48
db 46
db 43
db 40
db 36
db 32
db 29
db 25
db 21
db 18
db 15
db 12
db 10
db 8
db 6
db 6
db 6
db 6
db 7
db 9
db 11
db 13
db 16
db 20
db 23
db 27
db 30
db 34
db 37
db 41
db 44
db 47
db 49
db 51
db 52
db 53
db cr
endsS ; declare en
of Code segmen
END Start_Code

```

## Appendix A:

### 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 Vcc+0.5V  
Power Dissipation.....0.2 watts

TABLE I

DC & OPERATING CHARACTERISTICS

(TA = 0°C to 70°C, Vcc = +5V ±10%)

SYM	PARAMETER	MIN	MAX	UNIT	REMARKS
ICC	pwr supply current		18	mA	
VIH	input high level	1.9	Vcc	V	(3.5V for XTAL, RESTART)
VIL	input low level	-.5	0.9	V	
ILO	data bus leakage		10	µA	high impedance state
VOH	output high level	2.4		V	IOH = -60 µA
VOL	output low level		.45	V	IOL = 1.6 mA
FCY	crystal frequency	3.5	12	MHz	see clock circuits

### Electrical Conventions

All Cy325 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., ACK/.

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 Cy325 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 (pin #9) line must be held high upon power-up to properly initialize the Cy325. This may be accomplished via the use of a 4.7  $\mu$ F capacitor, as shown in Figure 1. Restart must be high for 10 msec after power-up stabilizes on power-up. Once the Cy325 is running, Restart need only be high for about 10  $\mu$ sec (11 MHz crystal).

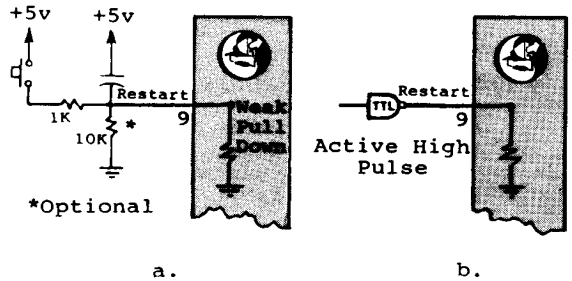
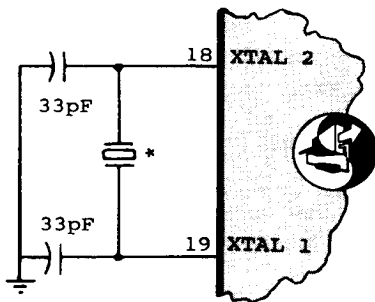


Figure 1 a) Restart Circuitry.  
b) External Restart.

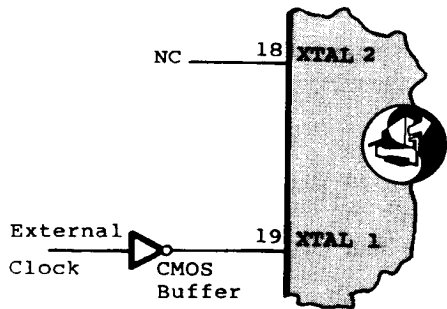
## Clock Circuits

The Cy325 may be operated with crystal or external clock circuits. 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 Cy325 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 Cy325.



\*11.059 MHz for  
Standard Baud Rates

**CRYSTAL**



Duty Cycle 45-55%  
Rise and Fall time must  
not exceed 20 nanoseconds

**EXTERNAL**

Figure 2 Clock Circuits for Cy325

## Appendix B: ASCII-Decimal-HEX Conversion Table

DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC				
0	00h	51	33h	102	66h	153	99h	204	CCh	^C	03	03h
1	01h	52	34h	103	67h	154	9Ah	205	CDh	^D	04	04h
2	02h	53	35h	104	68h	155	9Bh	206	CEh	^K	11	0Bh
3	03h	54	36h	105	69h	156	9Ch	207	CFh	^N	14	0Eh
4	04h	55	37h	106	6Ah	157	9Dh	208	D0h	^O	15	0Fh
5	05h	56	38h	107	6Bh	158	9Eh	209	D1h	^W	23	17h
6	06h	57	39h	108	6Ch	159	9Fh	210	D2h	CR	13	0Dh
7	07h	58	3Ah	109	6Dh	160	A0h	211	D3h	SP	32	20h
8	08h	59	3Bh	110	6Eh	161	A1h	212	D4h	+	43	2Bh
9	09h	60	3Ch	111	6Fh	162	A2h	213	D5h	'	44	2Ch
10	0Ah	61	3Dh	112	70h	163	A3h	214	D6h	-	45	2Dh
11	0Bh	62	3Eh	113	71h	164	A4h	215	D7h	/	47	2Fh
12	0Ch	63	3Fh	114	72h	165	A5h	216	D8h	0	48	30h
13	0Dh	64	40h	115	73h	166	A6h	217	D9h	1	49	31h
14	0Eh	65	41h	116	74h	167	A7h	218	DAh	2	50	32h
15	0Fh	66	42h	117	75h	168	A8h	219	DBh	3	51	33h
16	10h	67	43h	118	76h	169	A9h	220	DCh	4	52	34h
17	11h	68	44h	119	77h	170	AAh	221	DDh	5	53	35h
18	12h	69	45h	120	78h	171	ABh	222	DEh	6	54	36h
19	13h	70	46h	121	79h	172	ACH	223	DFh	7	55	37h
20	14h	71	47h	122	7Ah	173	ADh	224	E0h	8	56	38h
21	15h	72	48h	123	7Bh	174	AEh	225	E1h	9	57	39h
22	16h	73	49h	124	7Ch	175	AFh	226	E2h	?	63	3Fh
23	17h	74	4Ah	125	7Dh	176	B0h	227	E3h	@	64	40h
24	18h	75	4Bh	126	7Eh	177	B1h	228	E4h	A	65	41h
25	19h	76	4Ch	127	7Fh	178	B2h	229	E5h	B	66	42h
26	1Ah	77	4Dh	128	80h	179	B3h	230	E6h	C	67	43h
27	1Bh	78	4Eh	129	81h	180	B4h	231	E7h	D	68	44h
28	1Ch	79	4Fh	130	82h	181	B5h	232	E8h	E	69	45h
29	1Dh	80	50h	131	83h	182	B6h	233	E9h	F	70	46h
30	1Eh	81	51h	132	84h	183	B7h	234	EAh	G	71	47h
31	1Fh	82	52h	133	85h	184	B8h	235	EBh	H	72	48h
32	20h	83	53h	134	86h	185	B9h	236	ECh	I	73	49h
33	21h	84	54h	135	87h	186	BAh	237	EDh	J	74	4Ah
34	22h	85	55h	136	88h	187	BBh	238	EEh	K	75	4Bh
35	23h	86	56h	137	89h	188	BCh	239	EFh	L	76	4Ch
36	24h	87	57h	138	8Ah	189	BDh	240	F0h	M	77	4Dh
37	25h	88	58h	139	8Bh	190	BEh	241	F1h	N	78	4Eh
38	26h	89	59h	140	8Ch	191	BFh	242	F2h	O	79	4Fh
39	27h	90	5Ah	141	8Dh	192	C0h	243	F3h	P	80	50h
40	28h	91	5Bh	142	8Eh	193	C1h	244	F4h	Q	81	51h
41	29h	92	5Ch	143	8Fh	194	C2h	245	F5h	R	82	52h
42	2Ah	93	5Dh	144	90h	195	C3h	246	F6h	S	83	53h
43	2Bh	94	5Eh	145	91h	196	C4h	247	F7h	T	84	54h
44	2Ch	95	5Fh	146	92h	197	C5h	248	F8h	U	85	55h
45	2Dh	96	60h	147	93h	198	C6h	249	F9h	V	86	56h
46	2Eh	97	61h	148	94h	199	C7h	250	FAh	W	87	57h
47	2Fh	98	62h	149	95h	200	C8h	251	FBh	X	88	58h
48	30h	99	63h	150	96h	201	C9h	252	FCh	Y	89	59h
49	31h	100	64h	151	97h	202	CAh	253	FDh	Z	90	5Ah
50	32h	101	65h	152	98h	203	CBh	254	FEh	{	123	7Bh
								255	FFh			

# Appendix C:

## Typical LCD Specifications



LCD Dot Matrix Modules

QUICK REFERENCE GUIDE

Type No.	Number of Characters/Dots	Page No.	Character/Dot Size (mm)	Viewing Area (mm)	Outline Dimension (mm)	Driving Voltage (V)	Power Consum. (mW)	Control
----------	---------------------------	----------	-------------------------	-------------------	------------------------	---------------------	--------------------	---------

### Graphic Displays with Controller/RAM

AND1021	120 W x 64 H	151	0.44 W x 0.56 H	62.5 W x 43.5 H	85 W x 70 H x 20 D	+5/-8.5	35	T6963C	
AND1021-EO		151	(AND1021 - With Electroluminescent Panel Installed)						T6963C
AND682	160 W x 64 H	155	0.41 W x 0.41 H	75 W x 32 H	125 W x 50 H x 18 D	+5/-12	40	T6963C	
AND711A	240 W x 64 H	159	0.49 W x 0.49 H	132 W x 39 H	180 W x 65 H x 12 D	+5/-8.5	60	T6963C	
AND711A-EO		159	(AND711A - With Electroluminescent Panel Installed)						T6963C
AND1013	160 W x 128 H	157	0.56 W x 0.56 H	101 W x 82 H	129 W x 104.5 H x 14 D	+5/-8.5	45	T6963C	
AND1091	240 W x 128 H	161	0.66 W x 0.66 H	179.9 W x 101.5 H	241 W x 125.3 H x 12 D	+5/-8.5	50	T6963C	
AND1101	160 W x 32 H	153	0.46 W x 0.46 H	82 W x 21 H	140 W x 40 H x 12 D	+5/-5	-	T6963C	

### Graphic Displays with Super Twist Nematic with Controller

AND711AST	240 W x 64 H	178	0.49 W x 0.49 H	132 W x 39 H	180 W x 65 H x 12 D	+5/-8.5	-	T6963C
-----------	--------------	-----	-----------------	--------------	---------------------	---------	---	--------

### Character Generator

MSB \ LSB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]
1	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]
2	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]
3	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]
4	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]
5	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]
6	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]
7	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]	[Pattern]

# Appendix D: 6 x 8 Character Worksheet

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	6	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	7	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	8	_____

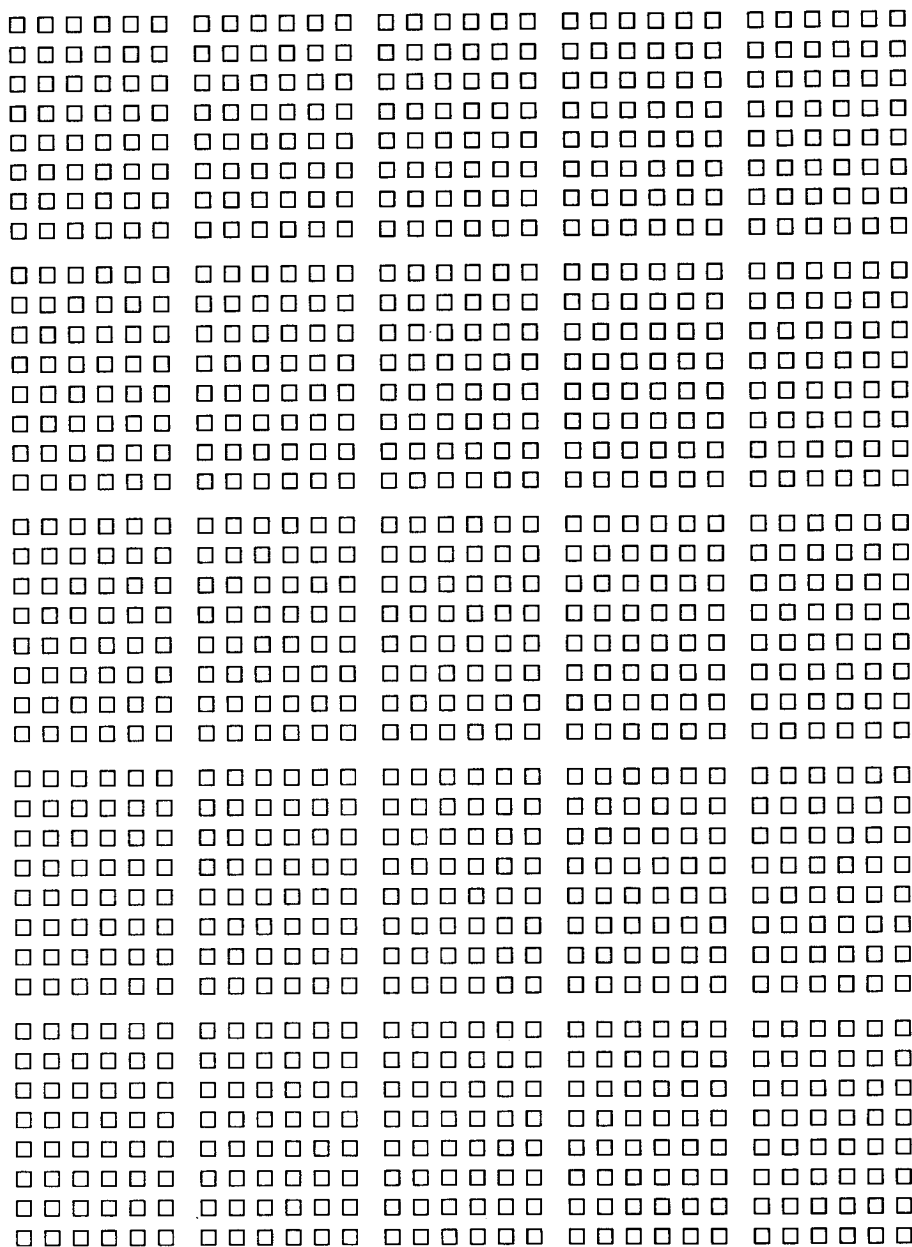
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	6	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	7	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	8	_____

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	6	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	7	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	8	_____

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	6	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	7	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	8	_____

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	6	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	7	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	8	_____

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	6	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	7	_____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	8	_____





# 8 x 8 Character Worksheet

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	6 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	7 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	8 _____

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	6 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	7 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	8 _____

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	6 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	7 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	8 _____

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	6 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	7 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	8 _____

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	6 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	7 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	8 _____

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	6 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	7 _____
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	8 _____

