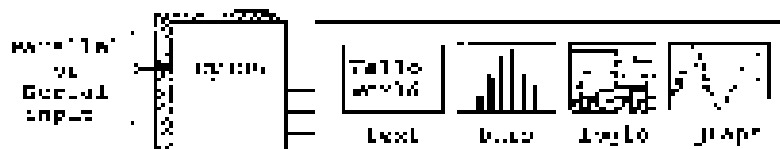


Introduction to Cy325 LCD Windows Controller

The size, cost, and appearance of LCD displays have finally evolved 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 low high-level text and graphics controller chip that works with "instrument-size" LCDs. The primary features supported by the Cy325 include:

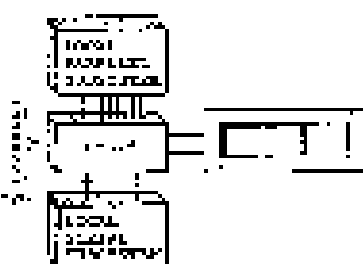
64 x 120 pixels = 8 rows x 20 characters	64 x 120 LCD
64 x 240 pixels = 8 rows x 40 characters	60-100 LCD
128 x 240 pixels = 16 rows x 40 characters	120-150 LCD

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 most stand-alone applications is to simplify the display of messages in windows with sophisticated typography, logic windows, and graphics support.



Instead of having to send low-level pixel-oriented commands, the Cy325 allows you to send high-level level commands with address, hexadecimal, or binary addresses. The 4 common create window, erase window, draw background, and/or windows, 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.

One 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 lines or serial signal lines that allow 6 serial unique functions. The combination of these signal channels allows the Cy325 to link two subsystems (at your choice) together via multiplexed switching between the Cy325 (local) data and the Cy325 serial bus as shown in figure.



LOCAL ADDRESS
LOCAL DATA
LOCAL CONTROL

Internal Mode Register

The CY225 possesses a large total set of internal modes of operation that can be enabled or disabled by setting or clearing a bit in the mode/register mode register. More than two dozen page 218 allow hundreds of variations to be easily specified.

Automatic Logic Waveform Display

The CY225 allows the automatic generation of logic waveforms in any window, via any pins that drive digital logic waveforms in the specified window. Instrument designers can easily produce logic waveforms by simply moving frame pins!

Soft-Key Support in the Cy225

A PC magazine survey (April 12, 1988) found that over 15 per cent of those surveyed thought that a menu driven interface is best for new users of software. Believing that the time is ripe for today's complex instruments, we have provided appropriate features in the cy225. Specifically, the CY225 supports three user-defined windows employed in either standalone operation or in a network, allowing a host computer to display the "meaning" of each key on the DUT. When the host 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 thousands users to make menu choices or otherwise use the soft-keys, with a central host managing the responses and updating the display.

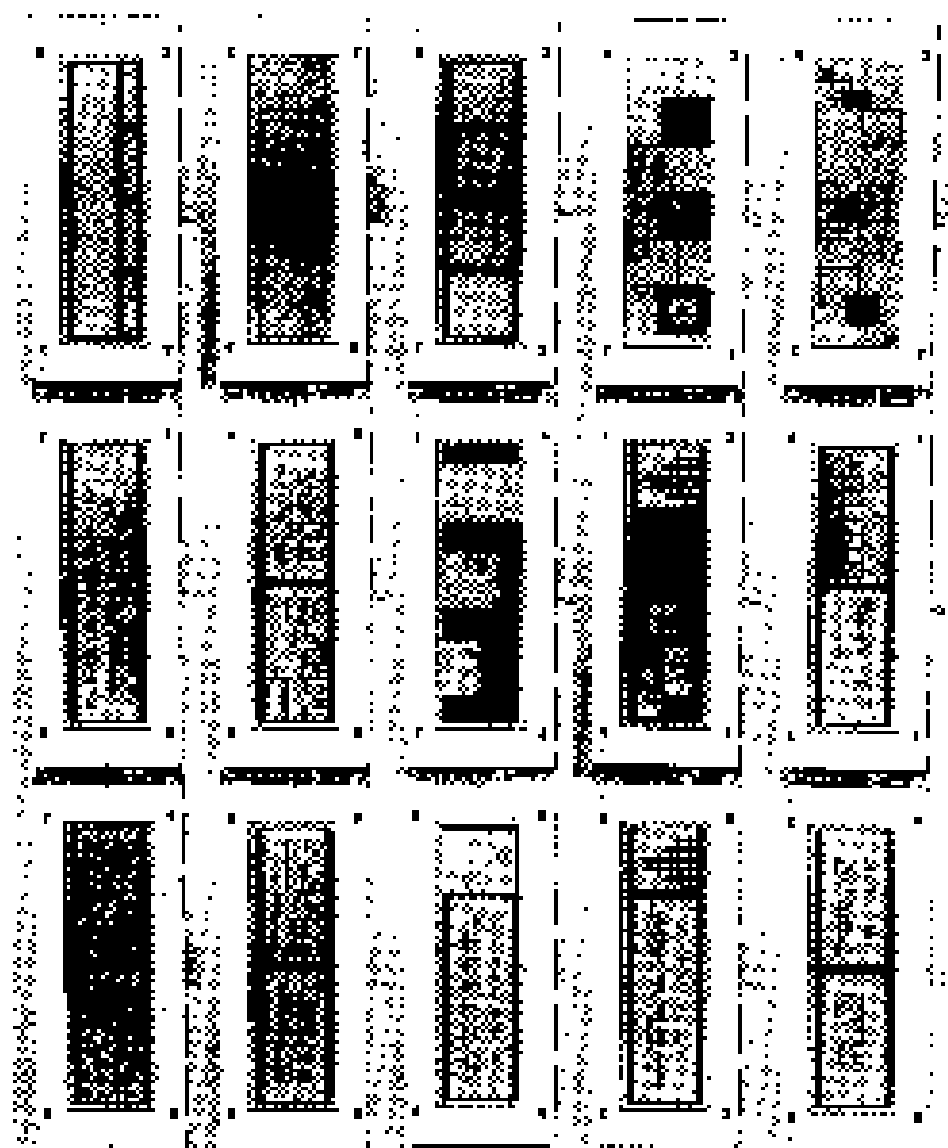
Instrument Design and Other Related Applications

While "display DUT" usually require some 200 pixel displays to be compatible with 305 software, such DUTs are generally too large and too expensive for instruments and benchside units. The CY225 can window controller controls DUTs with formats up to 240x120 pixels (16 lines x 20 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. It is applied to pages of text, as in word processors, etc.

— Cy225 Features —

Serial or 1-bit Parallel interfaces
Scrolling or user-defined windows!
Window definition and manipulation
Window-relative text and graphics!
Logic waveforms built-in to a plot!
Text characters and user defined fonts
Graphics automatically size to window
Communications between serial and parallel!
"Soft-key" support for menu management!
Network support based on Cy223 Network chip.

Example Cy325 Photos



CY325 LCD Windows Controller

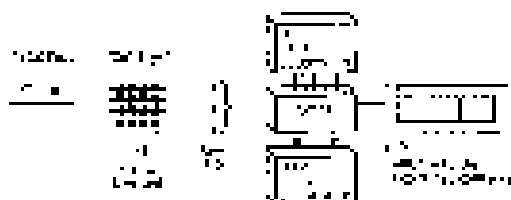
LCD Windows Controller

With an 800-pin parallel interface, the CY325 LCD controller provides an excellent graphics display in multiple windows on a 1280 x 1024 TFT LCD. The CY325 can be used in a wide range of applications, from simple windowing applications to complex multi-window systems.



"Soft Keyboard" Support Menu Management

The CY325 can be used to manage and control a menu system. The menu system can be defined using a menu design tool or a menu design tool. The menu design tool can be used to create a menu system that is easy to use and manage. The menu design tool can also be used to create a menu system that is easy to use and manage.



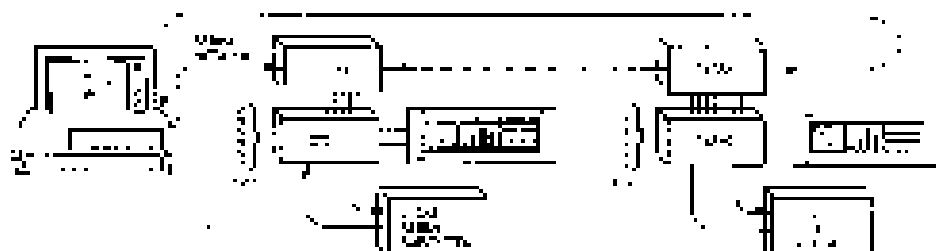
CY325 "System Element"

The CY325 can be used to manage and control a menu system. The menu system can be defined using a menu design tool or a menu design tool. The menu design tool can be used to create a menu system that is easy to use and manage. The menu design tool can also be used to create a menu system that is easy to use and manage.

CY325/CY325A Networked Displays

The CY325 can be used to manage and control a menu system. The menu system can be defined using a menu design tool or a menu design tool. The menu design tool can be used to create a menu system that is easy to use and manage. The menu design tool can also be used to create a menu system that is easy to use and manage.

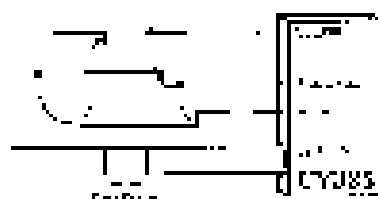
Typical CY325 LCD Display



CY325 Interface to Computer

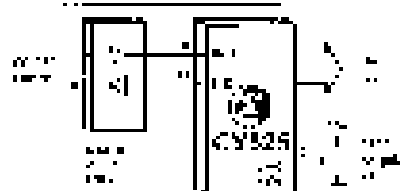
Parallel Interface

For the parallel interface, the CY325 is connected to the computer as shown in Figure 1. The CY325 is connected to the computer as shown in Figure 1. The CY325 is connected to the computer as shown in Figure 1. The CY325 is connected to the computer as shown in Figure 1.



Serial Interface

The CY325 requires only a single RS-232C line to connect to a computer. The flow of data is from the computer to the CY325. The CY325 is connected to the computer as shown in Figure 2. The CY325 is connected to the computer as shown in Figure 2.



CY325 Performance Table

CY325 Model	Line Type	Line Length	Line Rate	Line Rate
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10

The CY325 is a high-speed, low-cost device. It is designed to interface with a variety of computers. The CY325 is a high-speed, low-cost device. It is designed to interface with a variety of computers.

Model	Line Type	Line Length	Line Rate
10	10	10	10
10	10	10	10
10	10	10	10
10	10	10	10

CY325 Commands

The CY325 is a high-speed, low-cost device. It is designed to interface with a variety of computers. The CY325 is a high-speed, low-cost device. It is designed to interface with a variety of computers.

1. Command 1: Set
2. Command 2: Get
3. Command 3: Set
4. Command 4: Get
5. Command 5: Set
6. Command 6: Get
7. Command 7: Set
8. Command 8: Get
9. Command 9: Set
10. Command 10: Get
11. Command 11: Set
12. Command 12: Get
13. Command 13: Set
14. Command 14: Get
15. Command 15: Set
16. Command 16: Get
17. Command 17: Set
18. Command 18: Get
19. Command 19: Set
20. Command 20: Get
21. Command 21: Set
22. Command 22: Get
23. Command 23: Set
24. Command 24: Get
25. Command 25: Set
26. Command 26: Get
27. Command 27: Set
28. Command 28: Get
29. Command 29: Set
30. Command 30: Get
31. Command 31: Set
32. Command 32: Get
33. Command 33: Set
34. Command 34: Get
35. Command 35: Set
36. Command 36: Get
37. Command 37: Set
38. Command 38: Get
39. Command 39: Set
40. Command 40: Get
41. Command 41: Set
42. Command 42: Get
43. Command 43: Set
44. Command 44: Get
45. Command 45: Set
46. Command 46: Get
47. Command 47: Set
48. Command 48: Get
49. Command 49: Set
50. Command 50: Get
51. Command 51: Set
52. Command 52: Get
53. Command 53: Set
54. Command 54: Get
55. Command 55: Set
56. Command 56: Get
57. Command 57: Set
58. Command 58: Get
59. Command 59: Set
60. Command 60: Get
61. Command 61: Set
62. Command 62: Get
63. Command 63: Set
64. Command 64: Get
65. Command 65: Set
66. Command 66: Get
67. Command 67: Set
68. Command 68: Get
69. Command 69: Set
70. Command 70: Get
71. Command 71: Set
72. Command 72: Get
73. Command 73: Set
74. Command 74: Get
75. Command 75: Set
76. Command 76: Get
77. Command 77: Set
78. Command 78: Get
79. Command 79: Set
80. Command 80: Get
81. Command 81: Set
82. Command 82: Get
83. Command 83: Set
84. Command 84: Get
85. Command 85: Set
86. Command 86: Get
87. Command 87: Set
88. Command 88: Get
89. Command 89: Set
90. Command 90: Get
91. Command 91: Set
92. Command 92: Get
93. Command 93: Set
94. Command 94: Get
95. Command 95: Set
96. Command 96: Get
97. Command 97: Set
98. Command 98: Get
99. Command 99: Set
100. Command 100: Get

CY325 Command Format and Data Type

The CY325 is a high-speed, low-cost device. It is designed to interface with a variety of computers. The CY325 is a high-speed, low-cost device. It is designed to interface with a variety of computers.

The CY325 is a high-speed, low-cost device. It is designed to interface with a variety of computers. The CY325 is a high-speed, low-cost device. It is designed to interface with a variety of computers.

Description of Cy325

The Cy325 is a CMOS active integrated circuit that is designed to drive AHD and Toshiba liquid crystal displays based on the T-6959 low level controller. Examples of multi-colored LCRs controlled by the Cy325 include:

Display	Pixel Format	Connector Format
AHD 311A	241 x 64 pixel	1 row x 10 char
AHD 1021	121 x 64 pixel	1 row x 10 char
AHD 061	241 x 121 pixel	15 rows x 20 char

Command and Display Modes

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

Display Mode

The default mode is display mode. In this mode, the Cy325 displays characters from the 16 address available on the parallel channel. These characters are not interpreted (except for ASCII code 1) and are simply displayed on the LCR at the current cursor position.

Command Mode

The Cy325 is placed in the command mode via the ST11-C command. The ST11-C command from the Cy325 will not be displayed, will still arrive in the Cy325, and the command mode. In this mode, the characters are NOT displayed, but are interpreted by the Cy325 according to their meaning, decided in the testing facility.

Command/Display Selection Commands

The ST11-C command causes the Cy325 to select the COMMAND mode in which all received characters are interpreted. Each time displayed as text in the test window, the new value of this command is C00, that is, the ASCII ST11-C command. The ST11-C command can be repeated to the Cy325 via either the parallel or the serial channel and will have the same effect for either channel.

The ST11-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 test window, or at the specified cursor location on the LCR screen. It is described:

- ST11-C selects command mode operation.
- ST11-D selects Display mode operation.

Cy325 Features

System Interfacing:

Serial Interface (RS 232 C) (up to 19200 baud) (Clear-to-Send, Parallel Interface (2-Bus Handshake + Bus Control) (RS 232 Relay) (Selectable) (Cy325 Serial Controller)

Display Mode Features Included:

Display mode contains built-in ASCII font.
Display mode displays user-defined fonts.
Display mode allows special logic for display.

ASCII or Binary mode mode allows:

Font definition
Character line definition
View specification
Box drawing
Pixel plotting (color, interpolation)
Pixel interpolation (optional)
Clipping (optional)
Mode definition
Character definition from default window
Error messages in window
Error graphics in window
Scroll or Map characters in window
Linear Positioning and mode definition
Memory/Parity/Handshake status
ASCII control, hexadecimal, or binary characters
High-level auto-generation
Timing and configuration control
Communication "switch" between Serial and Parallel

Cy325 Serial Interface

The Cy325 accepts serial data with or parity and one stop bit required. The data rate is 1200, specifying the serial rate:

Serial Data Rate	Cy325 Baud Rate
1200	9600
1200	1200
Stopbit	Optional

In adaptive mode, the Cy325 waits for the receiver to get a full receive and a good rate to match the carriage return rate.

The Cy325 CTS line (pin 14) is a receive for clear-to-send serial status line. When this line is low, the Cy325 is ready to accept serial characters on the RX line (pin 10). When this line goes high, serial input should be held off until the internal buffer is unloaded. The CTS line will go low again when the internal buffer has been unloaded and is ready to accept new serial input.

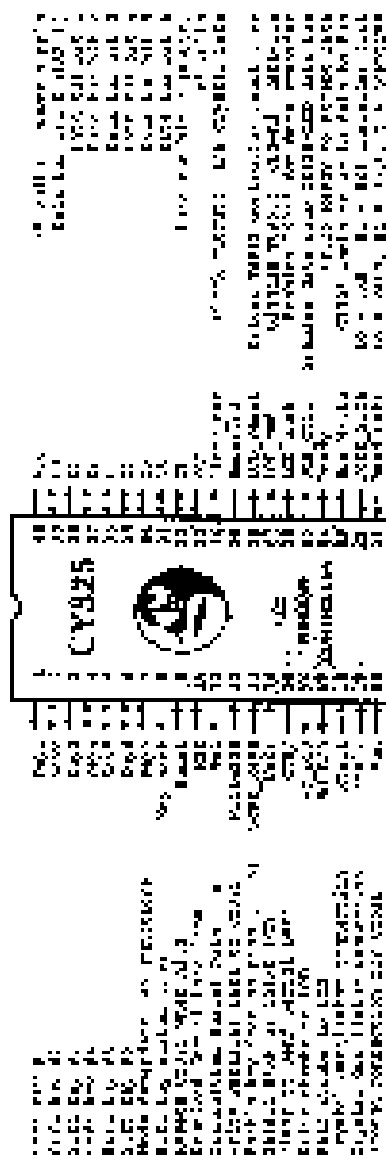
2 CY325 Pin List & Pinout Diagram 2

Cy325 Pin List

Pin #	I/O	Name	Description
1	I/O	Key0	Soft key 41
2	I/O	Key1	Soft key 42
3	I/O	Key2	Soft key 43
4	I/O	Key3	Soft key 44
5	I/O	Key4	Soft key 45
6	I/O	Key5	Soft key 46
7	I/O	Key6	LED on Gx4 module only
8	I	key_gpio	Keyboard status or led enable
9	I	key_irq	Keyboard interrupt when high
10	I	KeyD	Positive Keyboard data lines
11	O	TXD	Transmitted Serial Data Line
12	I	RX1D125	Receive Transmitted Data to Cy325
13	I	gpio_keyin	Keyboard scan pads
14	I	KeyB	Keyboard bit 125 pinout module
15	O	CIS2	Active Low Serial Clock-to-Stand
16	O	CS_LCD	Master strobe to LCD
17	O	rd_LCD	Read strobe to LCD
18	I	crystal	Crystal oscillator connection
19	I	crystal2	Crystal oscillator connection
20	I	cs2	Slave connection to Cy325
21	T	Trans Rate	Tr = 1000, Tr = 0000, I = adaptive
22	I	uart_irqin	Tr = 100, Tr = 0000, I = adaptive
23	O	uart_irqout	Receive strobe to write data to pin
24	O	uart_irq	Receive strobe to write data to pin
25	O	uart_irq2	Enabled Data only System-LCD data bus
26	O	LCD_irqin	Enables 0000, 0000, 0000, 0000
27	O	uart_irqin	Signal's 0000, 0000, 0000, 0000
28	O	uart_irqout	Receive strobe controller strobe line
29	I	uart_irqin	
30	O	clock_out	Clock output: 1/0 crystal rate
31	I	ad0	ad value
32	I/O	ad1	ad value: 0000, 0000, 0000, 0000
33	I/O	ad2	Data bus bit 0
34	I/O	ad3	Data bus bit 1
35	I/O	ad4	Data bus bit 2
36	I/O	ad5	Data bus bit 3
37	I/O	ad6	Data bus bit 4
38	I/O	ad7	Data bus bit 5
39	I/O	ad8	Data bus bit 6 (MSB)
40	I	cs2	15 volts = power supply

Cy325 Pinout Diagram

Cy325 Pin Configuration



The CY325 DCC controller uses several different coordinate systems for your convenience. Pixels are plotted in one system and character in another. These two are shown below for the VGA display:

Character coordinate system: $(x,y) = (0,0)$ to $(1023, 1023)$



Pixel coordinate system: $(x,y) = (0,0)$ to $(1023, 1023)$



VIC = 59
1024 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 DCC display will depend on whether Global or Window Relative coordinates are selected.



Example of Window Relative with Handed

4 CY325 Command Descriptions 4

Cy325 Command Format

In the command mode of operation, an ASCII incoming character (either serial or parallel) are interpreted, the valid command length is as follows:

A M1 M2 ... M1000

where 'A' is any ASCII character, followed by a space, and all through M are relevant parameter values, separated by spaces 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, 18, 50, 99, 121, 128, 12h, 0FAh

A decimal number 0 is represented as either an ASCII decimal character '0' or ASCII hex '00h'. The CY325 uses the latter convention for hex symbols:

'000h' or '000h'

and any leading 0's value from 0 to 99, a leading 0 is needed only if the leading hex character is '0' or '00h'.

Examples of valid numeric arguments include:

0 100 128 12h 000 000h 121 12h 00000000

Invalid numeric examples include: 000 000 000 120.

Binary Command Format

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

B B B B B B ... B

where B is any valid 8-bit value, there are 8B bits and where B is an 8 bit binary data count, followed by the 8 bit binary data value, it is 8h. There is no terminator and there are no blanks or spaces used in the binary data format.

Cy325 Command List

A x,y,x2,y2 data	New drawing command.
C x,y data	Current charactering command.
D x,y,d2,x2,y2 data	Font command command.
E x,y,d2,x2,y2 data	Graphics line drawing command.
E x,y1,x2,y2 data	Photo-graph operation.
I data	Initialize Cy325.
K data	Key in acknowledgment or control.
L data	Large character mode (text) or filled.
M data data	Mode command: data = mode byte, n = 0 to 7.
P x,y data	Plot pixel at (x,y) on LCD.
S data	Set value of n through the Cy325.
V data	Window = selected as current window.
H x1,y1,x2,y2 data	Window command defines character window.
X data	Set "Window Square" (x1,y1) = x,y/1.
Z	Reset command. (Special)
I data	Set data & status.
- data	Default Window status.
Z n data	Query I/O status: register n or Read.
A n data	Transmit n data bytes through I/O.
I n data	Print n binary data bytes in window.
*	Reset: Roll up 16 [Print] and (Special).

Cy325 Ctrl-Key Command Summary

		value
Ctrl-3	Command Mode Entry	0x6
Ctrl-9	Display Mode Entry	0x9
Ctrl-8	Play current Window	= 0x8
Ctrl-4	Reset to zero	= 0x4
Ctrl-5	Shift with = select special font	= 0x5
Ctrl-9	Shift = normal color	0x9
Ctrl-9	Window Set (to zero)	0x9

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), the Cy325's is the Ctrl-F command (Binary 011).

Ctrl-F Command Mode Entry

The Ctrl-F command switches to the display mode from the command mode.

Ctrl-D Display Mode Entry

The Ctrl-D command applies to the "current" window; selected via the set window command as described by the set window command.

Ctrl-X Reset current window

The Ctrl-X character (001) is detected in either Command or Display mode and immediately causes the "current" character display window to switch to the normal, only text characters. (For window graphics information see the initialization command.)

Ctrl-B Shift Out select special font

The Ctrl-B command should be issued in Cy325 display mode and initiates the Shift Out function. That is, it selects the special font and displays the P, M, B, C, characters, etc., as the Ctrl, Alt, third, character, etc., of the special user defined font that the Cy325 'P' command allows.

Ctrl-C Shift In normal character

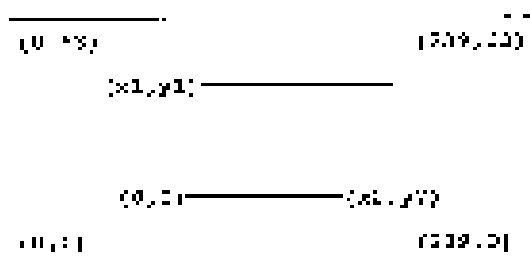
The Ctrl-C command should also be issued in the Cy325 Display mode and shifts back to the normal ASCII character font. The Ctrl-C command is the inverse of the Ctrl-B command. Both of these commands can be used as often as desired. That is you can switch between normal characters and special characters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

Ctrl-Y Shift X toggle

The Ctrl-Y 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 set window command). The window action is also swapped as well as described later. That one window may be displaying large characters while the other is displaying normal characters. Ctrl-Y 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 vectors in the pixel coordinate system:



`B x1,y1,x2,y2 <opt>` = Box drawing command.

The new command will draw a box at the LDD if the parameters `x1,y1,x2,y2` satisfy the geometrical relations: $(x1 < x2 \wedge y1 < y2)$ and $(y1 < y1 + Yscr)$, where `Yscr` is 64 or 128, depending on the particular LDD. These points are also used informally 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 'B' command defines the 'current' box parameters that will be used by any related command. This is true even when the box command is preceded by the negation symbol, '!'. Which defines the box outline to be erased. That is, '`!B x1,y1,x2,y1 <opt>`' erases the box outline, but the current box parameters are retained in exactly the same way as if replaced by a newer box or window declaration.

A box establishes a local pixel coordinate system but has no frame coordinates associated with it. Windows and Histograms, which also define local pixel systems, always contain an integral series of elementary squares 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 LDD screen. The last Box, Window 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
- 'X' Mode

The Cursor Positioning Command, 'C'

The 'cursor' command uses two arguments to specify the row and column 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` *move cursor to row, column*

For an 80 x 25 LCD screen, 40 sheet characters coordinate system is used below. The top left character is located at (1,1) while the bottom right character position is (25,40). Last is the bottom right coordinate is located at row 2, col 39. (The 1001 LCD has 15 rows, 0 to 14.)



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 grid (i.e. "no-window" mode is set), the unconnected cursor location will often lead to confusing results, since after a character is displayed, the Cx125 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 its autoconnect mode is turned off (see "edit" command).

Although windows are displayed using local pixel coordinate systems, and the Cx125 maintains a character-based cursor within the current window, there is no local pixel or window to pixel associated with a window. Thus cursor coordinates have nothing to do with windows. In global character coordinates, that is, they are relative to the window-relative.

The Font Command, 'F'

The 1979-1980 fiscal year defined term to be developed by the ICR, for award of the grant, is as follows:

```

      Fm, d1,d2, .. ,d8 rows      full width of printed
      charn                 n      character code (0..63) being specified
      z1 = top level of character font
      z2 = 2nd level of character font
      z3 = bottom level of character font

```

July 2 : moved to the new house

Each character in the 711A DCE is defined as a fixed set of on/off pixels on a 7x7 grid, depending on the 'font size' selection (p. 22). Each 'font' is 8 or 16 pixels wide and there are 9 rows per character. The following table illustrates a font command defining the second of four characters in the code 104, i.e. 'TAT' (p. 24).

٢٠١٩

Cont. Number	Row #	Date	Time
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	8	8

■ 400 ■ 1500' 100' 500' 1000' 1500' 2000'

After the above last command has been issued to the 7-726, and the input has been returned to the display area, the special character 'A' can be selected by issuing the Ctrl-H command. The character 'A' will point to one of the 80 columns (0-79) and will cause the code '1000' shown above to be displayed at its current position. The character 'A' will always invoke the same character code until either the Ctrl-G command shifts back to the home point, the return is taken, or the next character is used to redefine the code appropriately. In fact, if the Ctrl command accepts character codes from 0 to 63 as it does character fonts are being defined. ASCII characters 'b', 'c', 'd', 'e' are used to invoke the special characters Section three are easily generated in each program language and/or hardware. Special characters may also be changed after the first set up by using a full 3 hex character code to define the special font. In this case, fonts 0-15 correspond to character codes 104-119, 'defail' or ('b', 'm', 'h'), hexadecimal.

The Graphics Line Download Command.157

Graphic lines (rows of pixels) can be sent to the Gyl25 via the Graphics Line command, 'G', which has the following format:

“...and” and “only” have the same meaning

51664

n : number of data points in group, $n = 1, 2, \dots$

but help of us, this is

$\frac{1}{2} = 0.5$

As in the Font command, the data bytes are read as 8 bits, but the selection of the font bits determined how many bits (4 or 8) are actually mapped into the graphics line. The graphics line is formed by concatenating the horizontal groups of pixels to form a horizontal graphics line that is a multiple of 2 (or 4) pixels wide. For example, using 8-bit fonts:

```

2 3, 3, 5, 8 -xyz- will concatenate the segments
      /   |   \
     xy  xz  yz

```

It is found that the average number of visits per year is 1.5, with a standard deviation of 0.5. The number of visits per year follows a Poisson distribution. The probability of a patient visiting the doctor at least once a year is 0.75.

If the "Graph On" flag is set, it is ON the day is incremented above the graphed line is drawn. The graphed volume number is always increased, therefore the next Graphed 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 graphed pattern can be defined. This is done by issuing the following: the top of the volume is set to 1, the following example assumes that the top of the book is 100:

6. **How many people are in the family?**

1. **Introduction**

2017. 12. 16. 15:00

92.3% 92.3% 92.3% 92.3% 92.3%

and the

6. **Guiding light**

COPYING

QPMH "	... 2% ... " 2. ...
QPMH +1	... 2% ... 2. ...
QPMH -1	... 2% ... 2. ...
QPMH -2	... 2% ... 2. ...

2022 chapters have
 not shown on the 2022
 member list as a result of
 the above corrections.

The 'G' command with n = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, specifies the graphics row and column place. G10 is used in plotting the next graphics line. The column values of yRow and col are irrelevant. Therefore this command must be used to place the first one of the desired iteration.

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 `Cy10` will specify the width of each bar in the histogram is drawn to evenly space the display, however the values must be presented in the vertical direction, as is being presented to the `Cy125`. It is necessary to define the window or box before using the `H` command. The 'H', 'H', or 'H' command define the left, top, right and bottom sides of the window in which the histogram will appear. The `Cy125` 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 present the values presented to the `Cy125`.

Histogram command format:

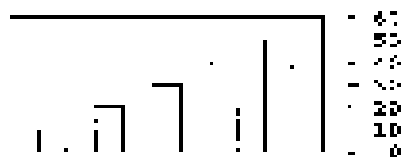
```
H [x1, y1, x2, y2] n x1 y1 x2 y2
where
n = number of bars to be displayed n = 1..9
x1 = x1 of 1st bar
y1 = y1 of 1st bar
x2 = x2 of last bar in display
y2 = y2 of last bar in display
```

Histogram example:

First, define the box via the `BOX` command, `H BOX, 0, 20, 200, 200`. This command defines a box 199 pixels in size (200-0), and an pixels high (200-0). The `Cy125` will use the value 199 for automatic horizontal scaling, and the user should use the value 20 to correlate the values presented in the histogram command. For example, if the largest bar value is 20, then all bars should be scaled by 10/200, that is, the bar height divided by the largest data value. Assume that after the above calculations are done, the histogram command is issued as follows:

```
H 0 0 200 200 5 0 0 200 200
```

This commands the `Cy125` to display 5 bars in the box defined box with values 1, 20, 50, 10, 60.



When the `H` prefix precedes the histogram command, the specified box is ignored instead of drawn. The bars will be drawn only up to the height specified in the command. (A value the value window uses the `W` command (see Initiative Commands).

The Initialize Command, 'I'

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

Cy925 Initialize Commands

I 0,xxx	= initialize register Cy925; 2 class version
I 1,xxx	erase the characters in current window
I 2,xxx	erase the graphics information in window or row
I 3,xxx	= erase the entire screen (currently Apple II only)
I 4,xxx	= restore Cy925 defined default font
I 5,xxx	= set a font-mode variation (only Apple II)
I 6,xxx	= select 8x8 font size (scale per 12 high)
xxx	ASCII or binary mode (default ASCII)
I 7,xxx	= set a font style (only Apple II)
I 8,xxx	= select Fast Map operation (default only)
I 9,xxx	= send value in' and Serial port
I 10,xxx	= and value in' and Parallel port
I 11,xxx	= display value in' as Logical Hexform

The 'I 0,xxx' command is treated as a "random" event, that is, the Cy925 will "pick up" in the row number as when the hardware reset line is pulsed high. Since the processor starts in the display mode, the 'I 0' command will leave the Cy925 in the display mode.

The 'I 1' command will erase all text in a window (defined via 'W' or 'R'). 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 x,y,z,xxx to find a character row in window x,y,z,xxx. Copy text into window x,y,z,xxx. This window is unchanged. When the 'I 2' command is used with 'I 3' the entire screen is "erased". That is, the current graphics window is "filled" with CH or some pixels. Note that these pixels do not interfere with any data in the window. It simply "covers" the text until removed (for a "corrected" screen).

The 'I 3' command erases all LCD RAM, including the external RAM storage area. All information, both text and graphics, is erased from the LCD. It is a predefined font in an LCD. The user must download the special font again after 'I 3' is issued.

The 'I 4' command reloads LCD RAM with the valid 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 scrolling of pins 1-16 by "pulling" pin 16 low from "tri-state" low cycle. Pin 16 can be pulled low from outside the device, maintaining the same effect. If the pin is from, then it stays pulled in low and 20 lines columns in high, enabling and disabling the row-wise operation (also made use of by 'I 6').

The 'I 6' command enables a row character font by pulling pin 16 high from "tri-state". The value can be achieved by pulling pin 16 high from outside. The LCD will stop pin 16 columns which set or fail last. The value pin 16 can be low or high and pin 16 can control the font size. From two pins can be low together and driver externally or controlled by the 'I 5' and 'I 6' commands. It will collect 64 font and 'I 6' will have 48 pixel font size.

The 'I 7' command collects the future row operation of the system while the 'I 8' command returns the cycle to its default state of operation. 'I 7' can be formed in binary: 27 49 01 01.

The 'I 8' command requires parameter for those value is supplied to pins 1-16 for cycle. If those pins are not in low state in one of the 80-800 order of operation, then they will be used to generate purpose value of lines for any user defined purpose. The 'I 8' prefix has no meaning for this command.

The 'I 9' command sets the "Test Bus" mode of operation. Normally this mode is insignificant, however bus structures will support it about half the time in "test bus" mode. From that test bus systems such as DDC 21 or system which have a DDC21 language is supported, it is usually necessary to know the effect in the system "test bus" mode. The 'I 9' command contains the "Test Bus" operation used by systems which cannot support the bus control pin 16 then the approximately 15 microseconds generated by the Test Bus mode of operation.

The 'I 10' command outputs the 8-bit value of the parameter 16 on the Serial port as 8 bits, no parity, one stop bit, using the word rate in effect at the time the command is issued.

The 'I 11' command outputs the 3 bit value of the parameter 16 to the Serial port using the handshake protocol specified in a later section. 'I 11' and 'I 12' are independent of the parameter command is provided in, on the other transfers through the DDC21.

The 'I 12' command uses binary value 16 on the Serial port to the system window. This command does not require any command to be issued on the other means of generating logic value.

The Key Command, 'K'

The Key command, 'K', with a parameter, provides an acknowledgement from the host to the Cy25. The parameter may be one of the options on 'Get Keys'.

The Key command has the format:

K nkey>

where 'n' is the number of the key that you want.

When the 'K n' command is preceded by the '!' prefix the action is to deliver pin 'n' only, where 'n' = 1..31. If Scan mode is enabled this has the same effect as a key acknowledging 'line n'. When Scan mode is disabled the 'K n' command merely puts pin 'n' low with no other effect. In this case the 'K n' command will return pin 'n' again. Thus when the Scan mode of operation is disabled, the 'K' command and the inverse can be used to provide seven 120 lines that can be set and cleared by command, for what ever purpose the user desires. (See the controller document, CY25, for additional programmed I/O.)

The Large Command, 'L'

The large command sets the data bit in the character window from either the large character set (large mode). The characters displayed in this window are either large pixels or their pixels depending on whether the LCU is supporting 6x1 or 8x1 characters. The large characters are effectively formed from four special characters, using the special character font. This allows either the use of large alphanumeric characters for greater visibility, and also allows special characters and symbols to be compared to be entered and displayed. The large characters advance the cursor one row within the character window in the same manner that normal characters behave. However only 16 large characters can be defined at one time.

No large characters are loaded into the Cy25. The following example will show how to create the large characters shown below. These large characters can be loaded by either ASCII decimal or ASCII alphanumeric codes.

ASCII code		Character description	Symbol char
121	121	The character	
122	122	Circle	
123	123	Left arrow	

The Mode Command, 'M'

The mode command can be used to set various bits in the Cy332 (a 32-bit device) (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.

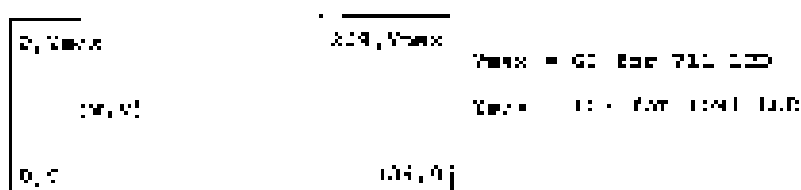
H 2,7 3000 Mode command: set mode register 2 to 7

The mode command sets all 8 bits to zero.

The Plot Command, 'P'

The Plot command operates in pixel space and specifies the pixel coordinates of the point to be plotted. The register of the plot command, obtained by prefixing the 'P' option, chooses the pixel at the specified coordinates.

```
P x,y coord      Plot pixel at (x,y) on LCD.
/P x,y coord      Same pixel as (x,y) on LCD.
```



"Global" and "Local" Pixel Plotting

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

Global coordinates (when used) are based on the centre of the 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 could be "added" 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 origin for pixel plotting. Local pixels will appear on the screen relative to this location. Since negative parameters could automatically be "outside" the window, they are not allowed, however some positive values can cause the pixel to be plotted "outside" of the current window. This can be prevented by setting the clipping mode bit, thus hit "clipped" pixels that fall outside the current window and prevent them from being plotted.

For more information relevant to pixel plotting, see the Box, Viewport, Window, Mode and Plot-Style keywords.

Plot commands handle only one pixel per command. If a string is plotted each is required to graph the output of an Analog-to-Digital converter. The stringing command should be used.

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 system. This is equivalent to loading the window command with a set of default parameters characteristically of the particular window.

The 'V' command defines a new current window. The current window defines the 'active' control of a left graphic and text-header.

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

These remain current until a new WINDOW is set or a viewport command is issued. A new command changes the Graphics window plot coordinates without altering the Text window character location or top, left, bot, right.

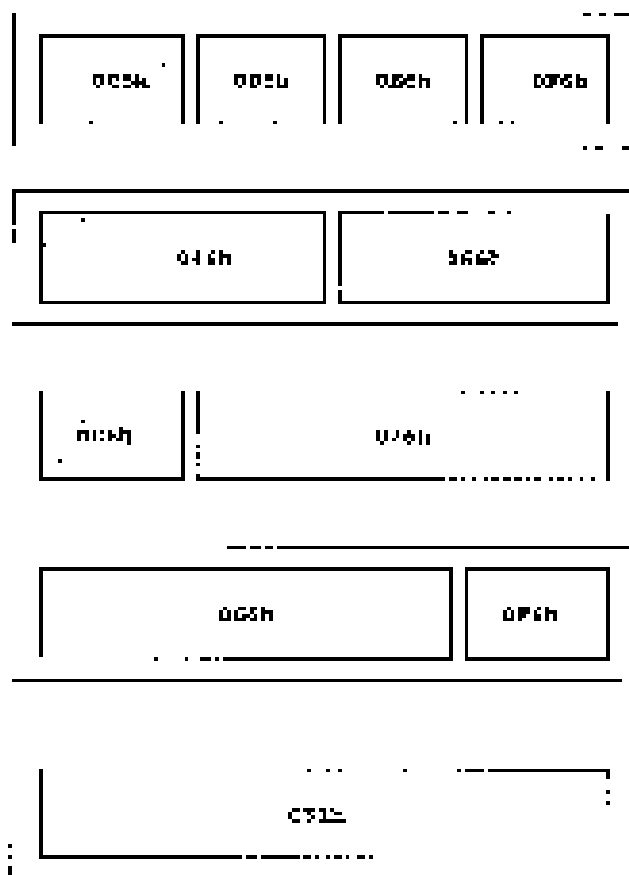
If the default window provided in the 'V' command does not meet your needs, then the Window command defined below will allow you to specify any window by supplying an appropriate set of parameters.

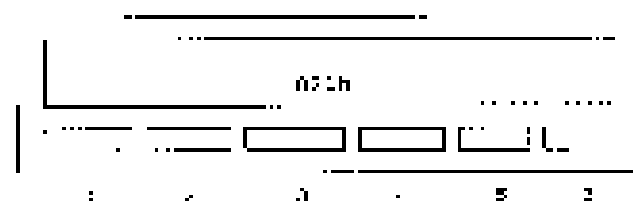
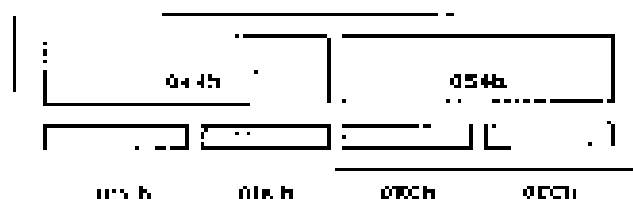
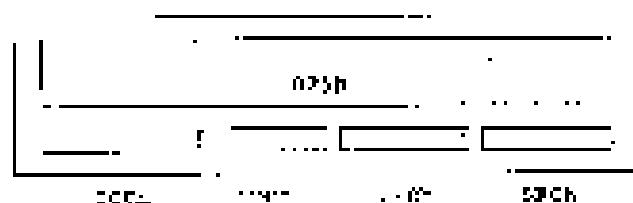
There is one major difference between the use of the Viewport command and the use of the Window command. The Viewport command will select one of a number of default windows that are described in a following section. These windows produce a viewport number and also possess fixed corner locations that are uniquely identified by this number. You can save the corner locations of a particular window, if it is necessary only to save the viewport number that identifies the viewport. These windowed 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 current location and also without losing those other status information relevant to the window. By definition, an action characterized by a specific form is being used. Since the saved window is identified only by the number, it is impossible to save user windowed windows. For this reason, you should attempt to use windows from the default set of plots, to avoid giving up the very useful save, restore, and swap functions.

Default Windows

The C525 provides a set of about 100 built-in windows. The way in which they are selected by the Transport command, "Win". The address they occupy is specified as a decimal or a hexadecimal value but is maintained internally as an 8-bit value that is decoded to generate the particular window.

Typical examples of default windows are shown below:





Cy825 Built-in Viewports

the `VIEW` provides over 25 built-in viewports that can be selected via the `VIEW` command where `VP` is the viewport number, and `N` is the viewport number. The default viewport is viewport number zero, which covers the entire screen of a 100 by 100 I/O. Viewport zero contains a frame of an `ADDRESS` (in `ADDRESS` mode) and contains 64x240 pixels. The 0,0 coordinate is located at the upper left corner while the 10,0 pixel coordinate is located at the bottom left corner of the screen.

All of the viewports are defined as viewport zero, that is, they contain lower than 64x240 coordinates. A window placed in the bottom left describes a given viewport as well as, such a window that its horizontal and vertical coordinates are implied by the code. One might ask why a coded viewport should be used instead of simply defining a window via the `W` command with the top left and bottom right corner coordinates supplied as arguments. The answer lies in the system. It is to have the current window placed on a 'stack' while switching to a new current window, thus to 'restore' the saved window status. The status consists of window number, character codes, window coordinates (both character and pixel) and several mode bits that qualify the window. Take the description of the `VIEW`, `VIEW`, and `VIEW` commands for more information on saving and restoring windows and the section on Cy825 mode bits for more information. The primary reason to use the built-in viewports instead of simply defining windows as they are needed is to place operations in two windows without having to maintain window status for each window. If the status is maintained internally only for numbered windows. It now proceeds to describe the window numbering system.

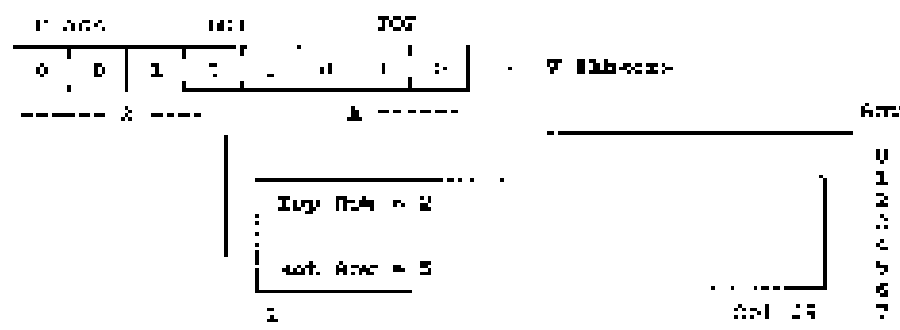
The Viewport Numbering System

The window system is maintained internally as an 8x16 matrix 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 exist horizontally. The relationship is such that (-1) class 1 window will open the 16th column horizontally. For example three class 2 windows will open the screen at four class 3 windows, or one class zero window.

The coding of classes 1, 2, and 3 is different (see class 1 coding, which we will describe first).

Class Zero Viewport Coding Scheme

Class zero windows extend from column 00 to column 14, that is, they span the entire horizontal extent of the 15 LCU. The window frame extends into column 00 and column 03 therefore two columns are not available for characters. The class bits are 000 zero for each window. 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 characters are lost if lost the top or bottom of the window. An example is shown below.



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

Class Zero with TOP>BOT

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

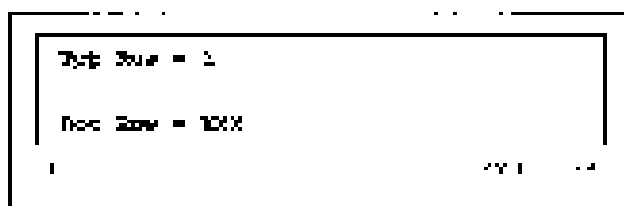


These 'key' windows may be used for any purpose but are originally well suited for use with the Cy323 'soft key' windowed position in this panel.

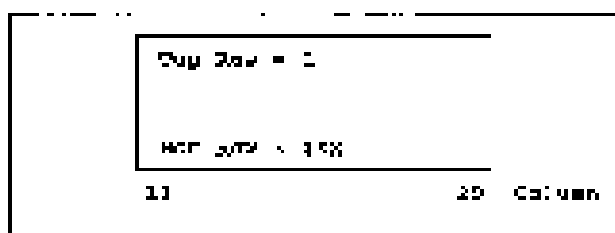
Class Zero Viewport with TOP=BOT

The previous special case is that in which the TOP and BOT lines have the same value. For this case the TOP parameter now 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 = 5. This window will cover the top four and one half rows of the screen starting from row 1.1 to row 5.4. How is different when TOP = 5 and BOT = 5. 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 row as shown below:

Class	BOT				TOP	
0	1	5	5	5	1	1



Class	BOT				TOP	
0	0	5	5	5	5	5



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 screen.

Coding for ISA pixel map displays the 1% BOT case and the 1 for the top row and 1-400 5.1 to 5.4 for row, when the BOT display is the pixel map.

Viewport Classes 1, 2, and 3

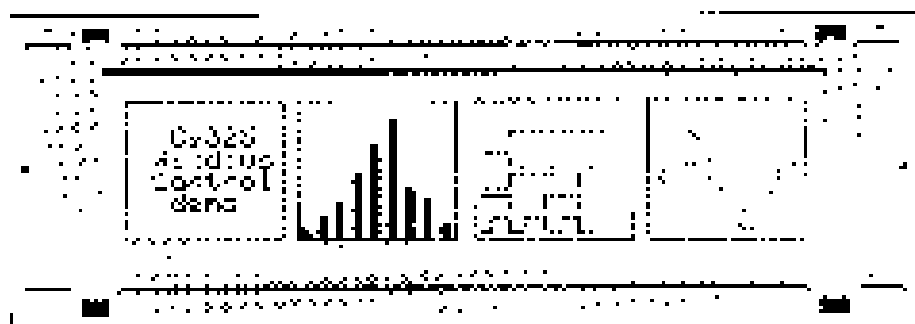
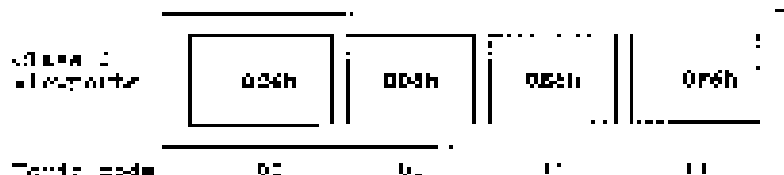
The class zero coding scheme described above has the following format:

Class	Ext	Top
0 0 0	0 1 1	0 0

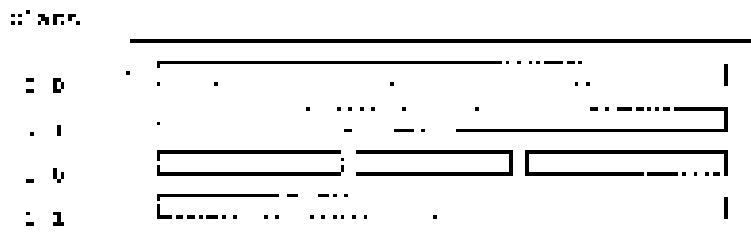
All other 1970 window classes have the following coding:

Class	Main	Vertical
0 0 0	0 1 1	0 0 0 0 0

As described previously, the class number generally relates to how many windows it allows to open the on screen. For example class three windows allow four windows to be opened horizontally across a screen as shown below.



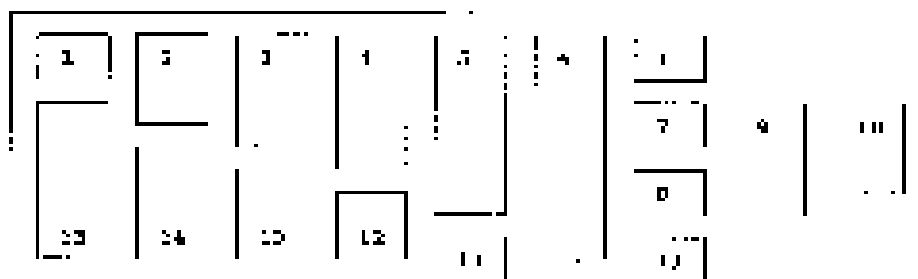
Now, the above example of window class 1, which from the horizontal coding defines the horizontal position for those classes in which multiple windows span the screen. The low four bits consisting of the class field concatenated to the bits field uniquely specifies the horizontal coordinates of the window. A binary diagram is shown below:



The low four bits of the bits field of the viewport number uniquely specify 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 (bits 4-7) of the viewport number define the horizontal position and extent of the viewport. The low four bits specify the vertical flow and uniquely determine the vertical extent of the window, as shown in graphics' flow and in table 1 of figure 4.



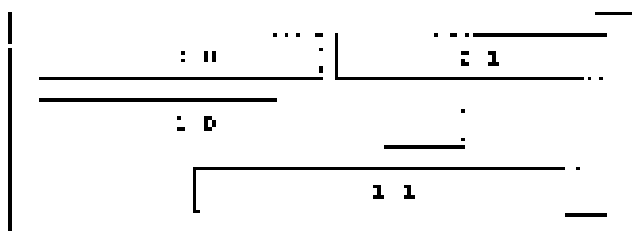
Vertical Coding Table

Left Plane	Top Row	Bit Low
0000	0	0
0001	0	1
0010	0	2
0011	0	3
0100	1	4
0101	1	5
0110	1	6
0111	1	7
1000	0	8
1001	0	9
1010	0	10
1011	0	11
1100	1	12
1101	1	13
1110	1	14
1111	1	15

Special Class 1 Matrix Coding

Class one windows allow a mix of two windows horizontally per row, however there are four parallel positions implied by the two bit plane field. The bits are treated in a special way as indicated in the following figure.

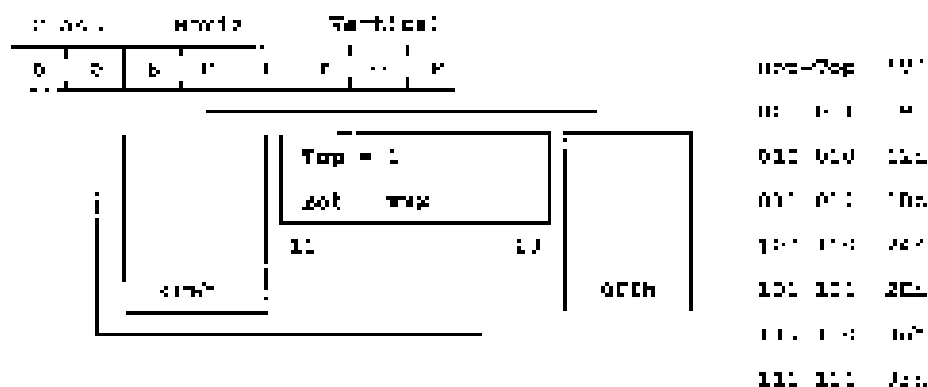
Class	Topic	Vertical					
0	1	2	3	4	5	6	7



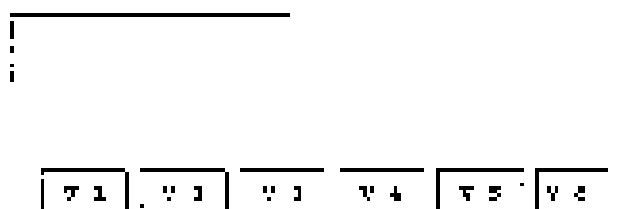
By properly identifying a specific number of columns by the window coding table derivation above, you can generate over 150 unique windows. With literally tens of thousands of non-overlapping windows on a 768x1024 screen, Class windows can be overlapped, you can generate hundreds of thousands of window combinations. It's easy and vector. Any given Class window you want is unique.

Examples of Cy325 'built-in' Viewports

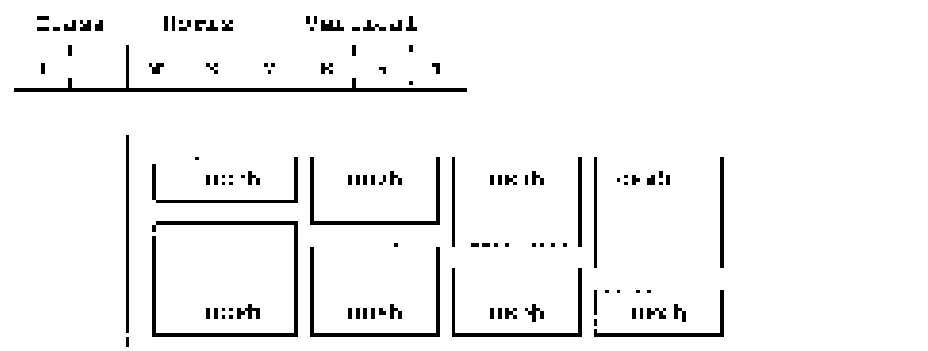
Class 0 Examples



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



Class 3 Example



The 'H' Prefix for Viewports on (Nx128) - Pixel LCDs

'built-in' or default windows are based on 128 x 128 pixel units. The Cx225 also works with 640 x 128 pixel units such as the AG-100 display in which the display screen is logically organized up in "upper" half and a "lower" half". On power up the upper half is selected for viewport windows. When the Hold pin of the Cx225 is pulled low, i.e. the 'H' prefix is "on" at the 'W' command window 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, except that it is covered by half a screen.

The 'H' applies only to the 'W' command. If the 'W' command is used to create windows in 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 test 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 <drv>
```

where

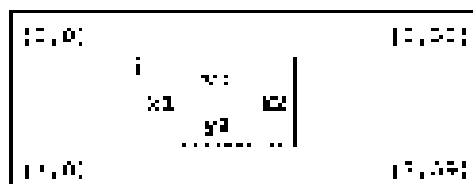
```
x1 = left char col  
y1 = top char row  
x2 = right char col  
y2 = bottom char row
```

that is,

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

The window size in characters is measured from the top-left corner, with increasing by number into the right, top, right, and bottom edges as defined by the parameters.

Character coordinate system: (row, col)



Notes: Window size:
Y1: LID = (0,0)
Y2: LID = (10,0)

For further information, look at the scrolling/scroll modes always use the currently specified window coordinates.

The Window Scrolling Mode and Position commands described below use a window parameter, <drv>, to identify the "test window". This parameter is not input to the driver command. 'S' (scrolls) and 'P' (position) commands can be used and combined with the 'W' (window) and 'S' (scrolls) commands. This may be accomplished externally to the S&P (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:

- (A) `W` = define current window to be default window.
- (B) `W` = fill or empty current window.
- (C) `W` = large characters displayed in current window.
- (D) `W` = small characters ("Times Square") in window.
- (E) `W` = define graphics window.
[displays low level window]
- (F) `W` = close or erase "current" window.
- (G) `W` = shift out current operation out of window.
- (H) `W` = shift in current normal text in window.
- (I) `W` = move viewport in current frame around in.
- (J) `W` = create viewport in lower half of screen.
['*' works only with Halftone pixel mode]

The following commands will do the work of windows defined by 'W':

- (A) `W` = save current window status.
- (B) `W` = restore current window status.
- (C) `W` = Swap in or out of window with the saved one.
- (D) `W` = Select lower half screen (100 deep LCDs).

Graphics and Text Windows

The current window defined by the `W` command becomes a graphics window superimposed over a text window. Most `W` commands operate on the contents of the graphics window or the contents of the text window or, at least, the two, but the text operations do not usually affect graphics pixels and graphics operations are generally independent of text in a window.

The `W` command defines the location of each window without altering the pixel column coordinates. It is important to note that define low "current text window". The `W` command can be used to "disconnect" the current graphics and text windows so that they no longer are superimposed but can have any relation on the screen including none and particularly no overlap. The `W` command only supports graphics, not text.

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

The Apple II RS Controller allows the use of multiple windows. A single application can display as many as 16 windows. To leave the "current" window and create a new window, to update information in another window, to perform operations in the "background" window, the Save, '+', and Restore, '-', commands accomplish this by saving the state of the current window (as defined by the 'V' command) as well as the current cursor location and the current window ID. Thus, the Apple II will allow other windows to be invoked and used in a variety of ways, including the use of the following:

- (1) - select another viewport to be "current"
- (2) - specify a new "background" window, window
- (3) - create a new group for "new"
- (4) - create another window, etc.

(Cmd. 2) - Create second window. (placed in Display mode)

(Cmd. -R) - Window Save - used after Save '+' command to swap the current window starting with the saved window number. 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 substituted in a string of characters that are sent to the C-128. If 16 saved windows can have different names selected, the 16th position of the characters may change. Note also that "local" cursors are swapped so that they do not "lose their place" when going from one window to another.

After the above commands or combinations of commands have been learned and the user wishes to return to the "original" current window, the '-' command is issued, followed by a zero. This will set the current window equal to the saved state and restore the cursor to the place it was located when the '+' command was executed. Note that this is all the '+' command does. In particular, it does not do any text, so that if the original text has been moved or deleted, the '-' command will not restore it.

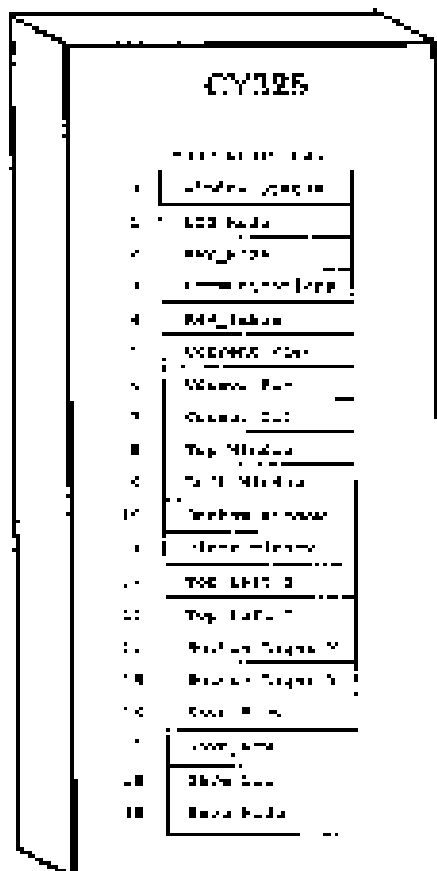
The 'Negate' or Slash Command, '/'

The slash '/' in front of command will 'negate' the command. For example:

- /Z causes the box drawn by the Z command.
- /H n, ... draws a horizontal line.
- /X n pull point to end, where n = 1 to 10.
- /C negates the large character mode, restoring the normal character mode to the current window.
- /W, y shows the plot for y = any value.
- /V n causes the outline of the specified viewport.
- /X disables horizontal scroll mode in current window.
- /I 2 fills (positive) or erases (negative) window.
- /I 0 disables the key-board operation.
- /I 1 disables the keyboard operation.

Cy325 Mode Registers and Mode Bits

The Cy325 modes of operation are specified and selected by the value in each of the 16 mode registers. The contents of the mode registers can be read or written as described below. The set of mode registers is shown below:



Cy325 Mode Register Description

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

Primary window registers

Mode Register #0	Window Status register
Mode Register #1	LED Mode register
Mode Register #3	Key-Mode register
Mode Register #4	Communication mode
Mode Register #5	Key_Power

Current Window number

Mode Register #6	Current Window (0-9999)
------------------	-------------------------

Character Cursor Position

Mode Register #8	Character XMR cursor position
Mode Register #9	Character YMR 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

Key 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 #18	save_time	Sec 45 saved
Mode Register #19	save_date	Sec 51 saved
Mode Register #20	save_mn	Key 11 saved
Mode Register #21	save_key	Key 40 saved

Mode Commands:

The Cy205 mode registers can be written to using the Cy205 Pin 10 command, 'W'. The mode commands require at least two arguments: the register address and the register value. Examples of the mode commands are:

- M 0.17b zero Enable automatic, autoincrement, vertical blank, and level clip mode.
- M 1.7 zero - Enable blanking for all text and video on the text plane only. Disable the graphics plane and all of the auto increment functions. See line number.
- M 2.2 zero - Set the text scan format and direction. See operation.
Note that pin 17 must be pulled low externally or via "I Force" in order to enable scanning.
- M 0.17h zero - Set video clipping (turn on or clipping) and autoincrement the cursor after each character is written to the window. Select the one-line cursor (instead of 4) and select text (instead of global) pixel plotting. With clipping on. Finally, select large characters.

The Mode Query Command, 'Q'

The 1-bit contents of mode register n can be read via the Query command, 'Q n zero'. The response to the Query command will be question mark character followed by the hexadecimal value of the specific bit's position, followed by the label name (see table below). For example, if bit 1 of mode register 0 has the value 1 then the three least significant bits of mode register 0 will be output as:

7 zero

The response to this query will be:

000h

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

Detailed Mode Register Bit Descriptions

Mode Register 10 - window status register

W0.0	- scroll_flag	: scroll = 1, wrap = 1
W0.1	- auto_line	: line wrap occurs after write
W0.2	- text_wrap	: also horizontal wrap (bit0)
W0.3	- global	: global to local plotting
W0.4	- clip_flag	: clip pix to outside local window
W0.5	- large_char	: 1st cell char (12x10-pixels)
W0.6	- clip_line	: box "Times Square" scrolling
W0.7	- special_char	: spec font: 1's = 1, 0's = 2

Mode Register 11 - I/O mode register

M1.0	- blinking	: enable cursor blink attribute
M1.1	- cursor_enable	: enable cursor display only
M1.2	- text_display	: enable text display plane
M1.3	- graphics_on	: enable graphics display plane
M1.4	- prog_update	: increment graphics row after load
M1.5	- display_line	: carry next-increment flag
M1.6	- wrap_line_flag	: normally off
M1.7	- full_cursor	: 4 line of scr, clip last line

Mode Register 12 - Key mode flags

M2.0	- shift_keys	: shift-keys (1's) pins 1.0-3
M2.1	- logic_keys	: pins 1.4 - digital logic (1/0)
M2.2	- cursor_keys	: pins 1.4 = up/down/left/right
M2.3	- aux_1_keys	: pins 1.1 = ASCII key inputs
M2.4	- aux_2_keys	: pins 1.1 = 1st matrix 10k input

Mode Register 13 - communications flags

M3.0	- send to Bus	: enable to input flag
M3.1	- send to Ctrl	: enable to input flag
M3.2	- key to Tx	: send keys to Tx
M3.3	- key to Rx	: send keys to Rx
M3.4	- auto_send	: auto key/display to Tx
M3.5 .. M3.7	- reserved	

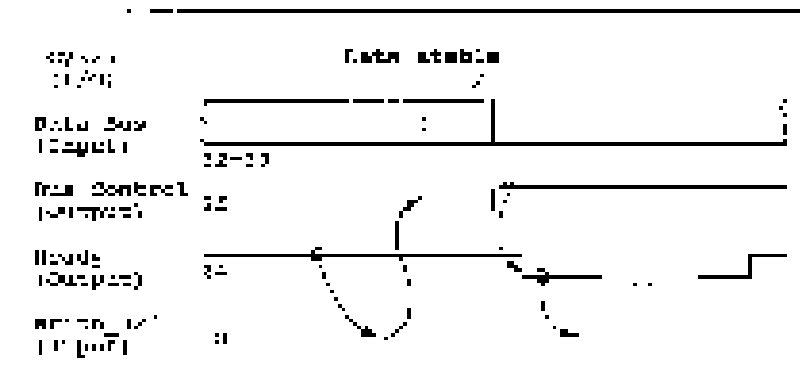
All remaining mode register bits contain byte values with no specific meaning associated with individual bits. These mode registers are set to 0x0 by default values.

Power-on Defaults of Mode Registers

M0	00h	0000	0110
M1	30h	0001	0110
M2	00h	0000	0000
M3	00h	0000	0100

CY325 Bus Interface and Handshake

The Cy325 provides a serial and a parallel interface to the 8080 display with a simple two wire parallel handshake augmented by a bus control signal useful as an output enable signal for getting CY data into the 8080 bus. The 8080 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 runs until the ready line is asserted positive from ground level and then enters the data transfer.

After the host has determined that the Cy325 is ready to accept 8-bit data, it drives the 75kΩ 325 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 address line and drives the 50kΩ 325 line low to the 8080 bus. This 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 fetch the data at the same velocity putting the data on the bus. The host always holds the bus in a high-impedance state, when the Cy325 bus is used to input parallel data and also to communicate with the VCB. By observing the bus control protocol, the host is assured that it will not place data on the bus while operations or communications are processing. After the system has read the data from the bus, it releases the control bit and drives READY low to indicate to the host that the data is ready. The host should drive the READY 325 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 repeating the 8080 line again for the next transfer.

Description of the CY325 Bus - Cstl Signal Line

The 16-bit parallel data bus is shared between the user circuit and the host interface. The Cstl signal line is used to control the busy and ready lines when attempting to write data to the parallel bus. The Ready line signals that the CY325 has finished the previous operation and is ready internally to accept a new command or data character. The low fast Cstl line signals that the CY325 is no longer communicating with the bus, and the user can place data on the bus without disturbing CY325-LED operation. The Cstl signal may be connected to a host data output enable if a buffer latch is used between the host system and the CY325.

Description of the 'Fast-Bus' Mode of Operation

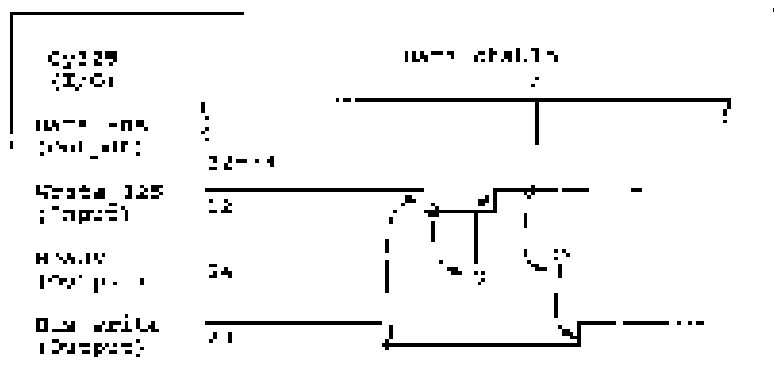
After asserting the Bus Cstl signal low, the CY325 expects to find valid data on the bus within approximately two microseconds. In a similar fashion, the CY325 expects the data to be removed from the bus approximately two microseconds after the Bus Cstl is returned high. If Bus Cstl is used as a output enable on a latch or is introduced to a CY325 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 wait to Bus Cstl, in which case the system will likely hang up. For this reason the CY325 supports a 'Slow-Bus' mode as described above, and also a 'Fast-Bus' mode of operation, designed to support slow speed systems such as BASIC on the Cyronetics stick-it design. In the Slow-Bus mode of operation, the system with data on the bus must wait until before loading the data, and, to terminate the transfer, wait until the bus has returned to the all ones state (00000000) before completing the transfer. In the Slow-Bus mode, timing is effectively disabled, and the status of the data bus is used to affect the handshake. This is the default or the power-on state of the system. The Fast-Bus mode of operation can be selected by loading the initialize Fast bus command (10000000). 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 use case of microprocessor direct handshake is important to the user, should the Fast-Bus mode be enabled. As a final note, the CY325 provides pull-up resistors on the data bus in order to see all ones when a tristate user circuit is driving the bus.

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

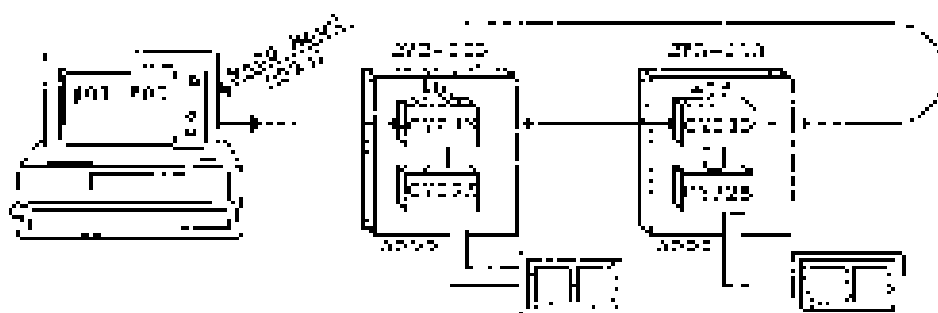
Fast control devices, such as microprocessors, could observe the Bus Cstl signal line for changing state and pull the bus. Slow devices, such as an LED that is simply blinking, should wait until the ready line goes low (busy) before removing the data and pulling the bus high.

Cy325 Output-to-Bus Handshake

The Cy325 can write data to the external bus. The Cy325 pulls the Bus Ready signal up to V_{CC} to indicate a transfer. The host must respond by lowering the address line (pin 22) to announce that it is ready to accept data. At the time the Cy325 then places the data on the bus and announces the presence by pulling the ready line low. When the host releases signal, the host has accepted the data and the Cy325 then drives all bus lines high, and releases $\overline{\text{READY}}$ high, terminating the transaction. If the Cy325 output is a string of bytes, the Bus Ready signal is held low until the last character has been accepted. This allows Cy325 Network Control chips to transmit a single message, rather than a string of one byte messages, even with a limited bit reservation.



The bus write capability, together with the Symmetrical Cy203 Network Control chip, allows multiple Cy325s to communicate over a single serial I/O channel, such as the IBM PC COM channel.



Cy325 "Soft Keys"

"Soft" keys are keys whose meaning is determined by labels appearing on a display instead of by symbols attached to the keys themselves. For example, some DOS 3.3 programs use the Function keys [F1] to [F10] as soft keys by displaying a key label on the bottom line of the user 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 far more likely that menu interfaces are preferred by the percent of users of our equipment. And, certainly, all of your customers will initially be new users of your instrument.

SoftKey Operation

The Cy325 supports the major elements of soft key operation: the LCD display of menu choices and the detection of the user's key-presses. Since the actual messages and interpretation of the keys are instrument dependent, the issue of just placing a unique key identifier code to the instrument processor. The instrument processor is responsible for handling the messages, interpreting the keys, and taking appropriate action.

SoftKey Protocol

The most important aspect of Soft Key operation is simply that no keypresses be lost or confused. This is accomplished by the Cy325 via a positive interlock method. After detecting a keystroke and translating this information to the host processor, the Cy325 "locks" until 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 some interlock with the user as well. In the primary softkey mode (10.0 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 released and acknowledged. The Address Register, located at the LED, thus informing the user that the request for action was seen and recognized. If host communication operations the host computer will change the message on the LCD display to some appropriate location before releasing the keys via the key acknowledgement.

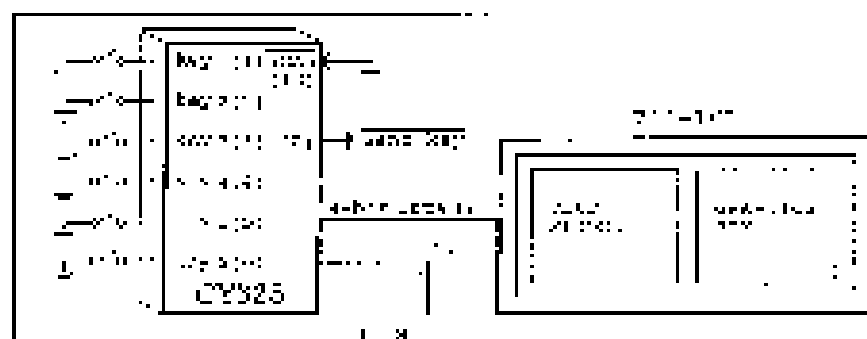
Cu325 Primary Soft-Key Operation

The CY225 STD Controller came out up to six work-days in its primary soft key mode of operation. The 6 work-days allow the DCL screen to display up to six messages and to associate any of six push-button driven signal lines with each message. A facility to set at least six lines can be selected via the 'Y' command where 'Y' corresponds to the number of the line below the window. Each such command will position the cursor at the front of each line, in preparation for writing a message into it. The messages are stored along the bottom of the display and the push button which is physically located in a way that the association of each button with its corresponding message is immediately clear. The STD can be associated with each of the six work-days.

Soft-Key Hardware

For citation: [author], [year], [journal], [volume], [issue], [pages].

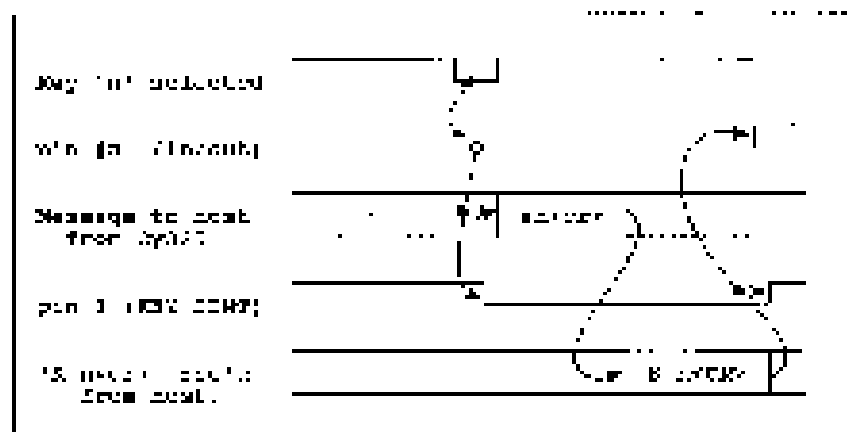
LOC	Pin #	Description
101	1	key #1 input
101	2	key #2 input
101	3	key #3 input
101	4	key #4 input
101	5	key #5 input
101	6	key #6 input
101	7	randomize key
101	8	key sent signal' action loc
101	10	key sent to signal



Soft-Key Protocol

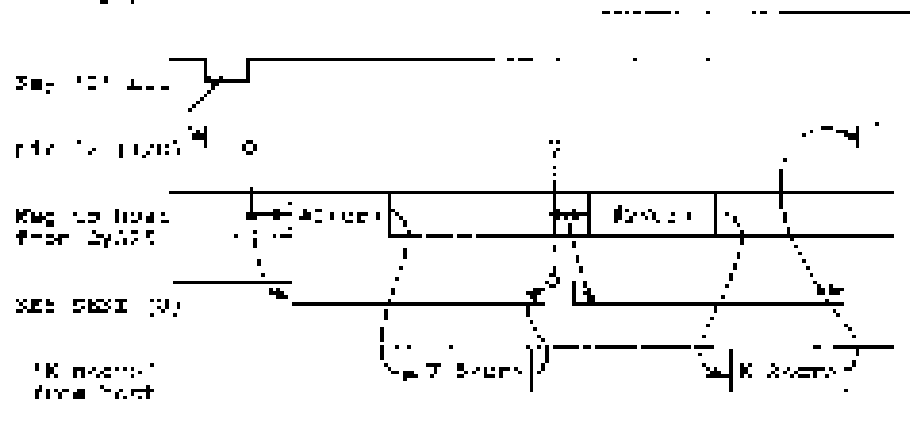
The Soft-Key Searching mode is dynamically enabled when pin 13, the active low *Ready/Busy* line is pulled low. The system will periodically scan the key input lines, pin 1, pin 2, and detect any low level key line such as that associated with a pushbutton. Note that there are already TTL signal lines and can be used with any compatible signal sources. The following protocol is readily understood, proved, to support all physical pushbuttons directly corresponding to ASCII characters.

When one of the six key lines is pulled low, the Cy855 will send the key number (ASCII) to the host, confirmed by the ASCII '1' output, and generated by the carriage return, *carriage*. The Cy855 also enables the *KEY READY* line, pin 8, low to signal that a key is being used, and holds the specific key line low. These signals may be used to control LCD displays. The *KEY READY* line inhibits key scanning and remains low until the host has acknowledged the receipt of key data via the message *16* message. When the Cy855 receives the *16* message, it releases the key line and also the line associated with key *carriage* (pin 10). If the acknowledged pin was the same as the active-low pin, then the Cy855 resumes key scanning (if the *READY* line is still held low). If the wrong pin is acknowledged, then the *KEY READY* line will be raised, ending key scanning. For the original key line will still be held low by the Cy855 until the wrong line was released and this will cause a second (and so on) report to the host. Thus each key requires a sensitive and accurate response (acknowledgments) from the user before restoring the key input to the ready state.



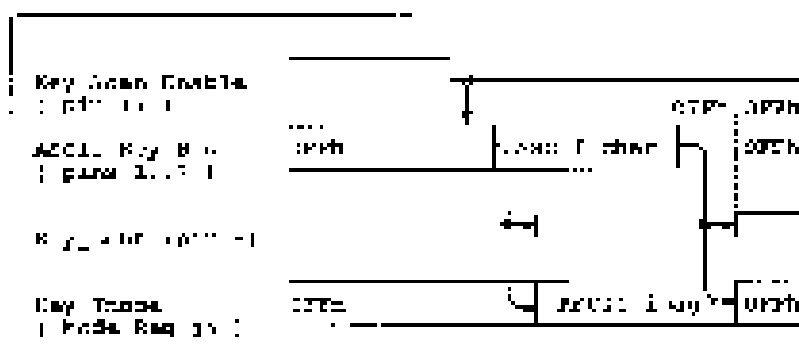
Host Response Key Command

The key scan operation is done that a host or master processor is in the system, since the Cy322 is not designed to take care of the keys. Therefore the key response must be placed in either the bus or the serial output channel. These key 'data' channels, as described in the next section, are enabled by mode bits on a user accessible 16-bit pushbutton. If either, or both, of the data channels is enabled, the Cy322 responds to key scalling time to low as to send the message 'I, host, to the master'. For power pushbutton operation, the master device acknowledges this message by sending the 'I, master' command (making sure the Cy322 is in command mode). The following illustrates the case in which key B (pin 2) is low, sending the message 'please' over the 'data' channel, the host responds with 'I, master', which is the incorrect response to acknowledge key B. As soon as this, the key B pin 2 is pulled up by the 'I, B' response, while the key B (pin 2) remains low. Then the SET SENT line returns high, sounding an alarm is raised, when pin 2 is detected low, the Cy322 assumes that key B is holding it low and issues another 'please' message to the 'data' channel. In this way the soft-keys 'know' on being properly acknowledged, when the host finally sends the correct acknowledgment message, 'I, B', the Cy322 solves key B (pin 2) and the KEY SENT line, (pin 1). At this point, assuming that the key line is still held low, key scanning is re-enabled and the host can be correct. In most applications this assumes that the host processor has processed the 'I, B' message according to its interpretation, accordingly all the key-scanning status messages 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 acknowledgments. Note also that the key B line can be used to hold an 'I, B' 'on' from the time the key is hit until the host acknowledges the key message. This channel signal to the host is considered to be a very significant feature of the soft key processor.



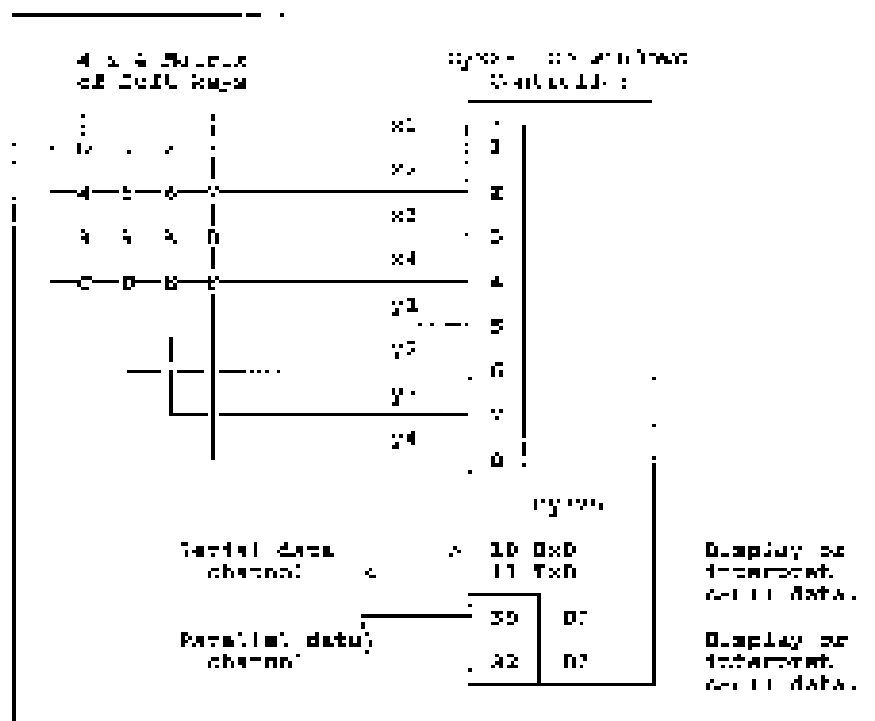
ASCII Soft-Key Protocol

The ASCII soft-key operation is initiated by setting bit 5 of mode register 40. This can be done via the mode command, M 5,0000. In this mode the communication line (pin 10), when pulled low, will cause the soft-key pin, 1/V, to be sent into the key latch (mode register 40) and the ASCII value on the pin 20 to be transmitted to the host computer, based on communication logic in mode register 40. The Key Send Line (pin 0) is pulled low to acknowledge the soft-key and signal that the key is being transmitted. This signal can be used to remove the soft-key bus to the sleep state, that is pin 1/V goes low or can be left to terminate the soft-key input after the Key Send Line goes low. In this mode the Key Send Enable line can be left active low or can go to high when key sent goes high. Regardless of the operation of the `Soft_Sleep` flag, the soft-key bus must be returned high between key strokes so that the bus is in both between two valid ASCII code combinations.



Soft 4x4 Key Scan Operation

The CP-125 supports six matrix key methods as plus 1-3 as shown below. The 1st soft-key operation is selected by entering out 4 of each register: 1 via the Mode command, M 11000000. The mode is changed by pulling the 8 and 9 keys (line 4th 11) key, either via the 1 3 mode command or via hardware external to the matrix. The CP-125 then begins an action over all the 128 keys, looking for a key closure that connects an 'I' line to a 'P' line. When such a closure is detected, the CP-125 transmits the 'I' number and then the appropriate hex key code ('0'-'F') to the host. Since the Key Sent line #1 is used for the address matrix scan, it is not available as an acknowledge signal line or in the other soft-key operational modes. Soft key keys are cleared by pulling the acknowledge acknowledge key command, 'K 000000'.



The key lock operation is inhibited during I/O operations. Although this is normally of such brief duration that the suspension of the key lock could not be detected, it may become noticeable during periods of high contention.

4x4 Shift-Key Matrix Table

	X7	X6	Key#
	111	110	1
	110	110	2
	101	110	3
	111	110	4
	111	101	5
	10	101	6
	101	101	7
	111	101	8
	111	100	9
	100	100	10
	101	100	11
	011	100	12
	110	011	13
	100	011	14
	101	011	15
	011	011	16

4x4 Matrix Scan Timing Considerations

The active scan of the 4x4 key matrix requires time on the part of the CY25. The 4x4 key scanning time diminishes the time that is available for scanning other functions. For example, assume that large characters are being input to the screen. Scanning the large character requires reading 4 serial characters for the 120 ms each transfer takes time. Normally, the CY25 can receive and output large characters continuously at 9600 baud, however when the 4x4 key matrix scan is enabled, the CY25 cannot sustain continuous operation at this rate. The CY25 can support large characters during 4x4 key scanning, but would not not be able to do so there must be a delay time included between each character. In the case of data input mode the received characters are moved in a buffer internal to the CY25, and buffer overflow will occur when long operations on the 120 ms of large character display, key fill or screen, or window scrolling, exists with 4x4 key scanning and high input data rates. If previous input is used, the READY pin (pin 34) should hold off the host long enough to avoid this problem. The output or ready line for serial is clear-to-send. When CTS pin 16 is low, the CY25 is ready to accept serial data to the buffer. When CTS goes high, serial input must be held off to prevent a possible buffer overflow.

Another possible solution to the key timing problem is to disable the 4x4 key scanning via the 4x4 scan command, then transfer high speed serial data to the CY25, then re-enable the key scanning via the 4x4 scan 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 globally the same, except that the globalised soft-key-key acknowledge is 'X 0000' which requires all eight scanning lines high and so enables the 4x4 key scan operation. Since no key-specific acknowledgement exists, it is impossible for the 4x18 to know that the right key has been acknowledged, so the return message 4x4 key is always 0x1 in the normal soft-key mode of operation.

Absolute Soft Key Verification via Query

As discussed in the section on soft-key approval, the 4x18 keys enabled by mode 8x17, 8x17 mode register, 8x17; bit 2 (00001) 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 assuming that the answer key will eventually be given, can allow the host to get a false key 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, 17, can be used to read the key_id register, (mode register 17) and independently verify which key was pressed before taking any other action on the key message.

The 17 Query command will query the Key trace register, and will return an ASCII string of the form 'XXXXXX' where XXX is the 3 character hexadecimal code for the 4x18 key image in the register, followed by the 4x18 value that is appended by the 4x18, then terminated by carriage return, 0Dh. The soft-key message and verification query messages are shown below for the 4x18-key:

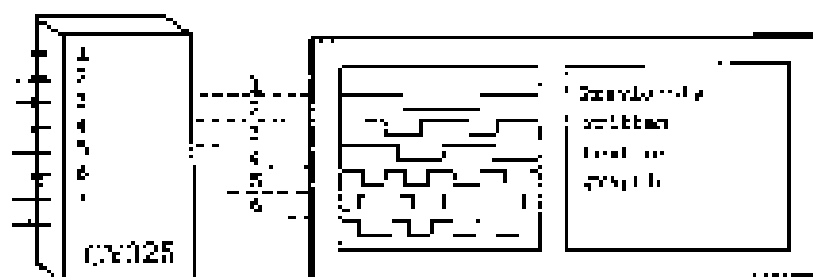
Soft-key value	Soft-key message	Key_id message	Key_image pattern
1	'000000'	'0000'	00111110
2	'000000'	'0000'	01111101
3	'000000'	'0000'	00111111
4	'000000'	'0000'	01110011
5	'000000'	'0000'	01101111
6	'000000'	'0000'	00111111

After the soft-key 1 in row 1 has been correctly acknowledged via the appropriate 'X 0000' command from the host, the result of a following Query of the Key trace, via the 17 Query command should be '000000'.

11 CY325 Logic Waveform Display 11

CY325 Logic Display Mode

The CY325 will display data changed in memory when the Enable SCAP line (pin 13) is pulled low. When the Logic pin (pin 14) is pulled high, via the H key, the command, the Logic key, the mode is enabled. In this mode, eight pins (pins 1 through 8) are scanned and the changes on these pins are displayed in the 'current' window. Only one pin may be selected in each of two columns wide and should be at least six rows deep. If both pins are used, then none of the waveforms will not be displayed. The CY325 will read the pins and reproduce the TTL level of each pin on a separate row with the latch value entered in the right side of the display and the last value of the pin on the left side each row frame displayed. An example logic display is shown below:



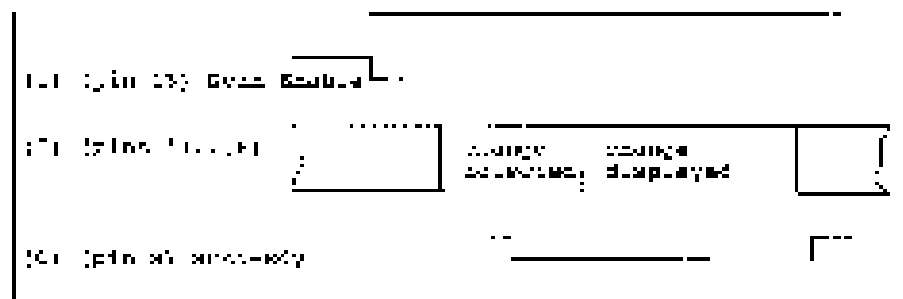
Pin 7 on the CY325 is also enabled, however only six waveforms are displayed. Pin 7 can be used to advance the waveform through any of the six displayed waveforms displaying that pin is currently selected the waveform to show "latch passing" without requiring any of the six active lines to change.

The waveforms can be scanned by the command 'latch' without actually leaving the hardware pins. It is not necessary that the scanning mode is enabled for this command. The six digits of the binary value are used to drive the six logic waveforms in the current window.

Finally, note that the CY325 hardware pins can also be pulled low (pin 1) and the 'latch' command (if they are not pulled high or low externally) and also by the 'latch' command. These commands have the effect of driving the waveform pins from inside the CY325 and will be recognized in the logic waveform display window if the logic waveform mode of operation is selected and enabled. If the pins are actually used externally to pull the signals to pull them low from inside, no further operations should be done if you plan to use these commands.

Buffering the Logic Waveform Signals

The Cc325 Logic Display function operates in either unbuffered or buffered mode. In unbuffered mode, the changing signals are connected directly to Cc325 pins A1 through A4, and the changes are reflected as they occur. This mode, if necessary, limits the rate of change of the displayed logic due more to the limitations of human vision, but also to the rather slow scan rate of the Cc325 (to be specified). For most applications the Cc325 serves only as a display, with the actual data collection and storage handled by a master processor that uses the Cc325 as a slave. In pin 21 and the A0A0 line from 11 to 10 connect to the Cc325, which can change to 00 played in the window per channel. The handshake sequence is shown below:

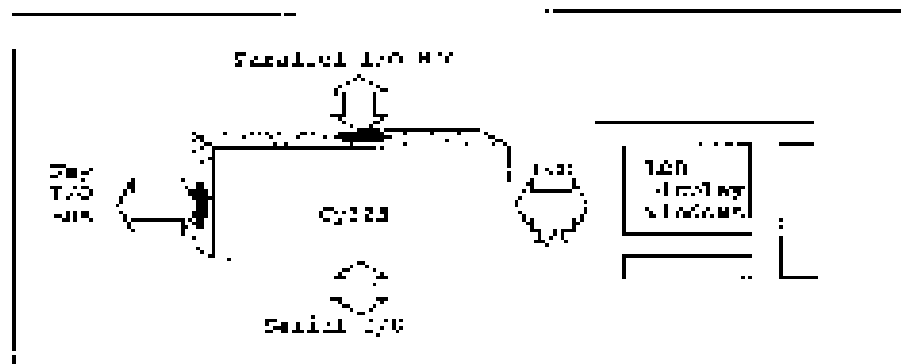


Since it is desired to capture high-speed logic signals and display them on the Cc325, you might use a FIFO or other device capable of saving a sequence of high-speed logic signals and use your host processor to read the FIFO and present the buffered signals to the appropriate pins on the Cc325.



The CY325 as a Switching Element

The CY325 provides a variety of message switching capabilities, based on switch-logic defined in the Communications Mode Register (CMR) and supported by several CY325 instructions. The four I/O channels associated with the CY325 are shown below:



The four major CY325 data channels are:

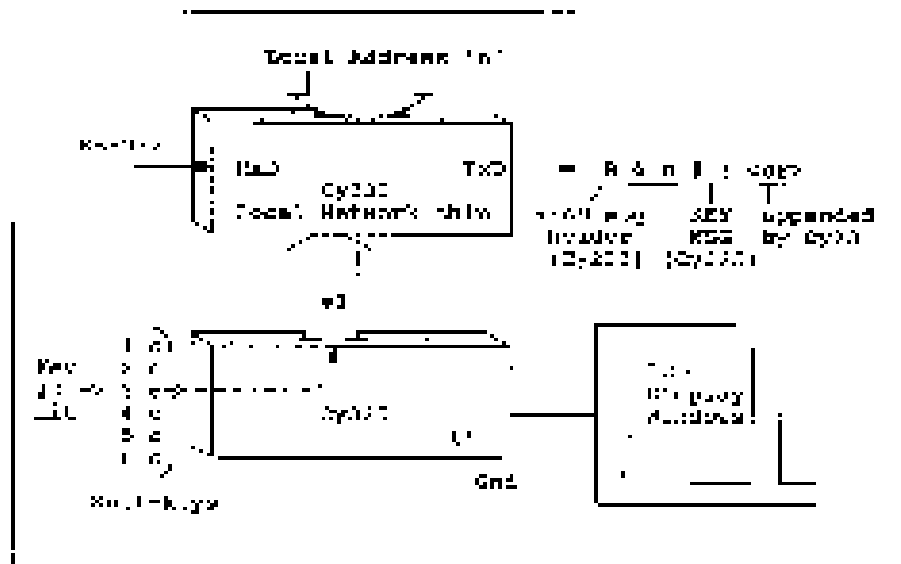
- Two Data Ports
- Parallel I/O Port
- Serial Data Port
- I/O Data Port

As indicated, some 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, but all of them will be considered in a switching matrix table.

The Shift-Key Input Channel

Case 1: No. 11 Air Switch Enabled

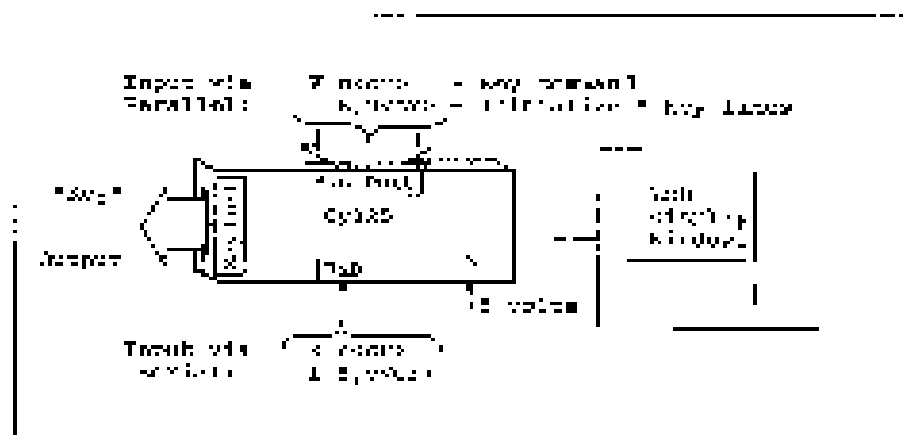
The Key channel connects to the output pins (1..3) on the card. Either 8, 16 or 32 lines are used, depending on whether single byte-key ASCII keys, or 484 matrix keyboard are selected. In all cases the key inputs are received at the key pins and output to either the parallel or serial data channels. Each key inputs one **KEY** output to the host. The choice of parallel or serial output is determined by the Communications mode register, specifically by the `KeyToBus` flag (KS3) on the Key to bus flag pin. If the Key to bus flag is 0H, a key directly off, such as the "0" character is output to the parallel data bus, followed by the ASCII hexadecimal key identifier code, 01111011 for single switches or 0111101110 for 484 matrix mode key switches. If the parallel data pin is interfaced to a Cyrix interface chip, the device will prefix a READ message transfer (RT) followed by the address to the message (RTA) and then transmit the message with a checksum. The other is shown below:



Key Channel Outputs

Normally the anti-key channel is in an input channel, whose summary is wanted at 01 01 01 01 (over) command or by pulling pin 11, the `key_Enable` line low externally. The key activation is interpreted according to the Key Mode register (12) and the output message is stored by the setting of the command in the Communications Mode register (10). Activated Key line is held active low by the `key_Enable` (over) command. If the Key Mode command is active, it shifts the Key line as de-activated (assuming that the user has released the key).

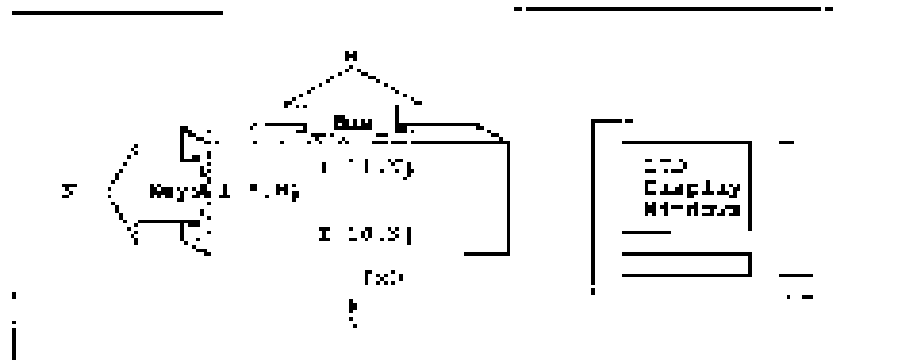
If the Key Scan is disabled via the `key_Enable` or by pulling pin 10 high, the input capabilities of anti-channel are essentially disabled. The key port can be used as to send the output via either pin 10 or the '1' channels. The '1X mode' command (over line 12) and 'K mode' returns to high. The initial key port command, '0 0 mode', sends the shift value, 0, to the eight key pins, thus entering a data byte to the key port.



The Initialize "Output" Commands

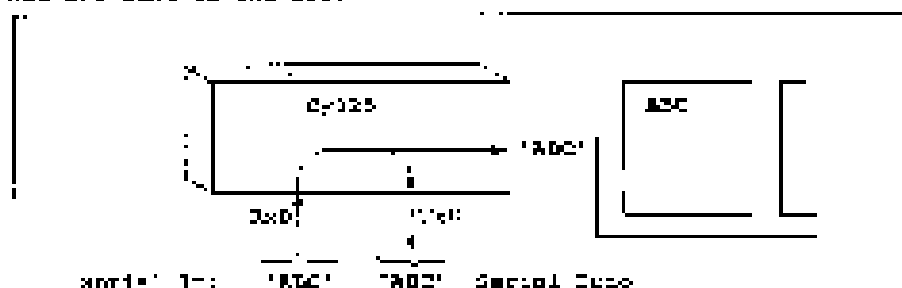
There are three special Initialize commands that output data to three different parts of the Cy325. The command can be forced to the Cy325 via the parallel or the serial channel. The output channel is completely determined by the command, regardless of the channel over which the command was received. (In contrast to the 'send' and 'transmit' commands to be discussed later).

- I 0,0000: Send binary value of 'in' to Sys Data (pin 1..8)
- T 0,0000: Send binary value of 'in' to Serial port (pin 12)
- T 0,0000: Send binary value of 'in' to parallel bus (22..25)



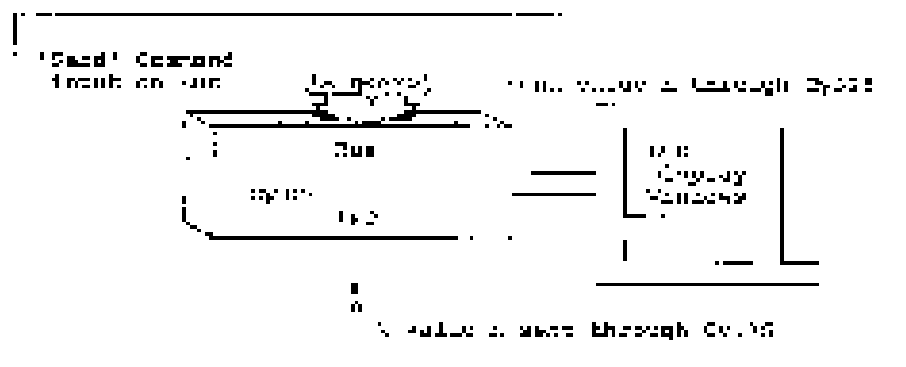
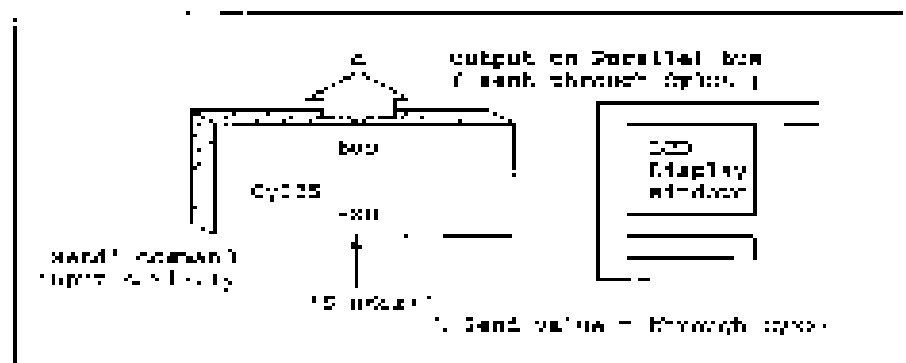
The Serial Echo

The Communications Port Register (10) contains an 'Echo enable' switch or mode bit that causes all displayed characters received on the input serial channel (E30) to be echoed out the output serial channel (pin 12) when the bit is set (to 1, 00000001). There is no corresponding echo bit for parallel data. The serial echo does NOT allow control characters, only Displayable characters that are sent to the LCD.



The Send Command

The Commands discussed above have guaranteed proper regardless of which channel the command was received on. The 'Send' command (and the 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 Cybex, regardless of where the data came from. Data input output directly 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:

```

N bytes
Hex-Val
N = ASCII control or Hex-Val + 1 for N value
  
```

The arguments of the Send command is up to 255 ASCII string numeric values, while Hex-Val may be equivalent to output.

The Transfer Command, '@'

The Transfer Command, like the Read Command, is designed to transfer characters through the Cx225. 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 Command is very somewhat analogous to the Read and Plot String Commands. The Plot String Command plots a string of pixels, those values are entered in binary form. Similarly, 'Send' takes a single argument, identified as a decimal or hex ASCII parameter, and sends the 16-bit equivalent value to the port position 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 Cx225. The data count is followed by data bytes in binary form, each of which is sent through the Cx225 and the data count documented. The format of the Transfer Command is as follows:

@ n al .. an hex

where

l = count (space) : 01 (ch)

n = 16-bit binary data count

al = first 8-bit data byte to be transferred

an = 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 subject to the same 'a' bus with appropriate handshaking. If the speed control is turned in favor to the serial parallel bus, the data will be sent to the Cx225 serial port, Cx0, 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 of the output port are always 8-bit binary numbers, regardless of the input format of the argument.

An example in which the Transfer Command sends data from the remote serial input (Rx) to a Cybit Network chip interfaced to the parallel bus of the system is provided in the next chapter.

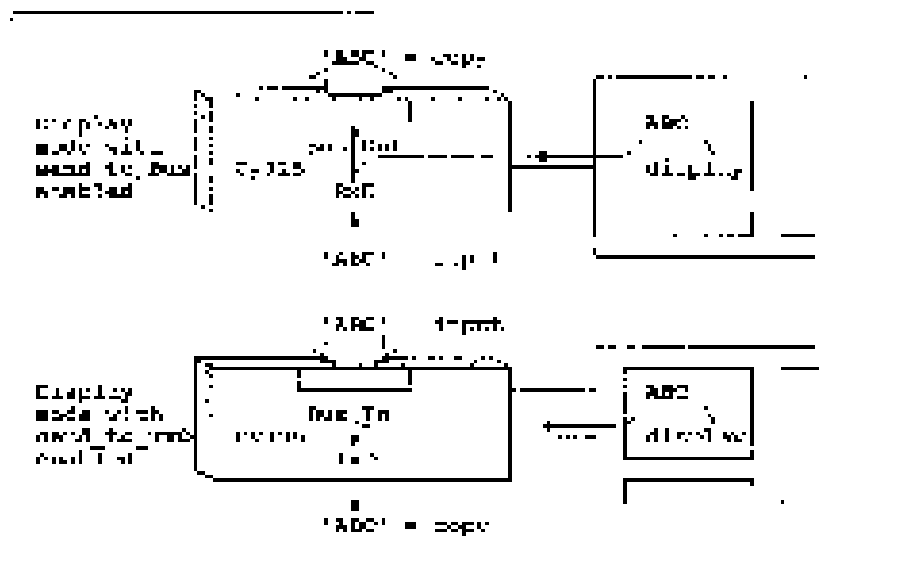
Copy Switch Functions: Send_to_Bus and Send_to_Tx0

The Send and Transfer commands pass their arguments through the Cy225. Their argument values are output to the channel opposite that which the commands are received on. This behavior can be generated such that all displayable characters are copied to the channel opposite that which they are received on. All 'ctrl' characters that are displayed on the LV can also be output to the parallel bus, and all displayable characters that are received on the parallel bus can be output serially on Tx0. The mode bits in the communications Mode Register that control this switching behavior are the R20 and R21:

R20 Send to Bus copy serial display to parallel bus

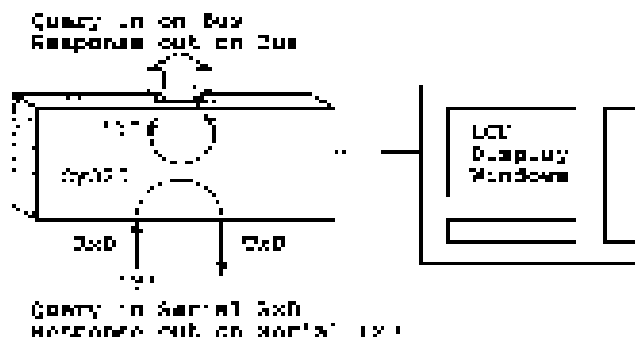
R21 Send to Tx0 copy parallel display to serial out

Only displayable characters are transferred through the Cy225. 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 Cy225 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 Tx0 will be copied to the opposite channel that they are sent. Either or both of the copy bits, R20 and R21, 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, '3'

The query command, '3', is designed to extract information from the system. The information consists of either the value in a specified mode register, or the 16 character or character string located on the LCD screen at the current character cursor address. The Query response depends on the input channel (00) with the Query was received, however the same is opposite to that of the Read and Transfer commands. While these commands are designed to send information through the D125, the Query command is designed to extract information from the system. From the data output in response to a Query command the 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:

'3' nnnn'

where

'3' 3000 Command character
n hex character value
nnnn command termination

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

- n = 1, n = 2 respond with 4-bit value in Mode Register no.
- n = 20 respond with 3 bit character code of the character located on LCD screen at position specified by current character address.

Query Command Response Formats

The response to a query command is unique to the task involved in which the Query command is received. The format of the response to a query of one of the mode registers is:

7 0 x 8 0000

where the 71 prefix identifies the response as a query response, the 00 hex indicates that the 01 prefix identifies both the start of the 0x 00000000 value (not the hexadecimal equivalent of the mode register contents), with a redundant '0' confirming this fact, terminated by a carriage return. All mode register query responses always have this format, even when the Query Command is not in column 0.

When the Query command argument is 20, the command is not a mode register query, but a request to the 2x00 to read the character on the TCH screen located at the current cursor location. If it is, then the 2x00 will output the ASCII value of the character read from the 2x00 located at the 2x00's ASCII equivalent. It is not the ASCII value of Character Query, determined by the setting of the "Query auto read" mode bit (M12) in the 1x0 mode register bit.

If the Query argument bit is not, the character Query command simply reads the character at the cursor location and transmits:

7 0 x 00

where

- 7 = query prefix
- 0 = ASCII character read at cursor
- 00 = terminator

If the Query auto read bit is ON, the character query command reads the character at the cursor location, and sends it out if it is the ASCII blank (octal 000), 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 edge of the window is reached.

The character read terminates when any of the above cases occur. The message that was read from the 2x00 then is formatted with the format:

7 0 x 00 00 00 00 00

where

- 7 = Query prefix
- 00, 00, 00 = ASCII character value, on 2x00
- 00 = ASCII blank (octal 000) from 2x00
- 00 = terminator

If the response is to a 2x00 the "End" header will be received.

Cy325 Switching Function Summary

The preceding transaction treated the various switching functions separately; this section summarizes the various *key* commands, paths, and receiver switching information. The switching functions of the Cy325 are designed to provide data paths through the four information channels associated with the Cy325.

Data Path:	Key Data Path
	Special Data Path
	Serial Data Path
	LED Data Path

The *switch* switching functions depend upon *key* and *switch* 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:

<i>Cy325</i> commands:	'0' n, xl ... xn Transfers data Cy325 to opposite path 'K channel' Send thru Cy325 to opposite path '1 channel' send key path for changes 'I 1,channel' Output value 1 to Key path 'I 10,channel' Output value 1 to rxn path 'I 11,channel' Output value 2 to Bus path '2 channel' Output mode reg n to rxn path
------------------------	--

The mode registers that are associated to *Cy325* data switching are the Key Mode Register 12 and the Communications Mode Register 13. These registers are shown below:

Mode Register 12 - Key Mode flags

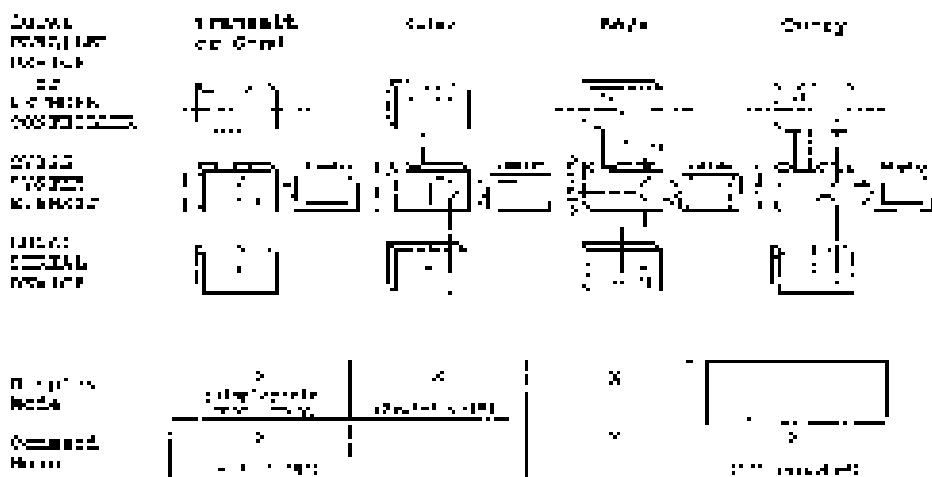
M2.0 - soft keys	: pins 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8
M2.1 - "begin" channel	: pins 1.1, 6 = digital logic 1/0
M2.2 - "end" keys	: pins 1.1, 6 = up/down left/right
M2.3 - ASCII keys	: pins 1.1, 7 = ASCII key inputs
M2.4 - Key matrix	: pins 1.1, 8 = key matrix (16 keys)

Mode register 13 - communications flags

M3.0 - send to Bus	: quality of output flag
M3.1 - send to TxR	: quality of input flag
M3.2 - Key to TxR	: send keys to TxR
M3.3 - Key to RxR	: send keys to RxR
M3.4 - RxR serial	: echo RxR display to TxR

CyS25 Message Switching - Major Options

The following diagram illustrates the data paths through the system. The effect of the mode bits in the Communications mode register are shown as follows for the two paths. If the rate bit is enabled (RSE), the switch will be "closed", while if the rate bit is disabled (RSE), the switch will be "open".



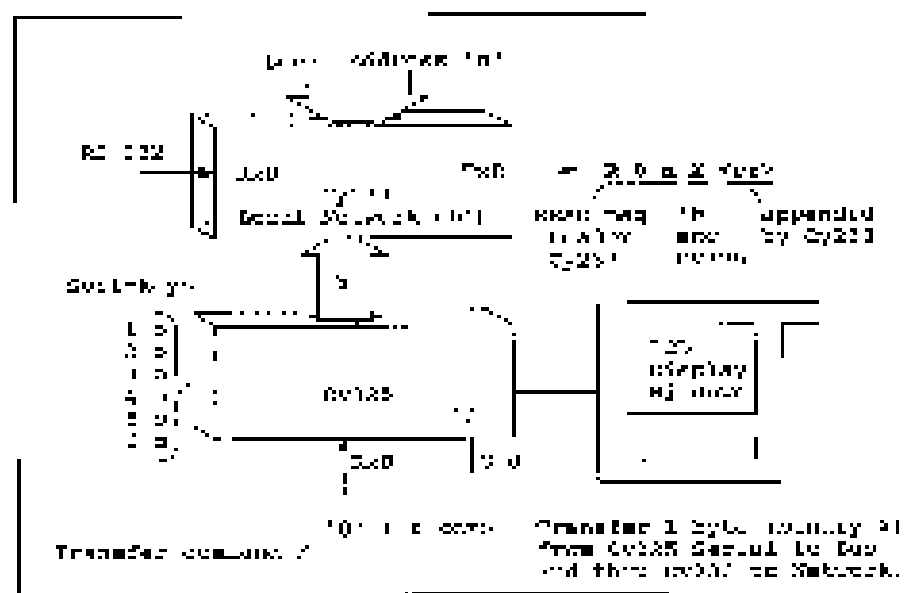
CvJ25 Switching Matrix

[illegible]

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 1200 to 2400 baud computers to a "host" area network. All such links can be implemented in a number of ways, but the Cy325 uniquely supports Systematic House Systems (SHS) Local Intelligent Network Controller, that consists of a 4000 baud rate RS-232C capable of supporting up to 125 network addresses. The serial communication channel such as the DCH1 port of an IBM-PC is available. The interface between the Cy325 and the Cy233 is done either as a box diagram out in schematic form or a following page.



The Cy325 connects to the Parallel bus of the Cy233 and supports 2400 baud operation with its 1200 baud baud rate. The Cy325 can also accept Serial information to be displayed on its bus and send some the network to the host computer. The Cy325 transfer command is used to transfer data from the serial port of the Cy325 (RS-232C) to the Display Monitor that is connected to the Cy233 Network Controller.

Brief Description of Cy233

The Cy 233 is a Micro System Cy233 is a device to the CHOC device providing addressing capability to peripheral ports. Each Cy233 in a network is given a unique address by tying the address lines high or low as per requirement. The Cy233 supports ASCII binary, or hex message format. To write the ASCII message string low to a Cy233 with address set to 00 00 00 00:

00 00 00 00 0001

The Write buffer, the Cy233, the Cy233, to write a message from the serial network to the peripheral device (Cy233) attached to the Cy233. The address selection 0001 0001 will actually write the message. The 'Address' is written to the Cy233. The message return location for the message. To read a message (0001) over the network, the Cy233 handshake with the Cy233 via the buffer, which is compatible, protocol. The Cy233 accepts the message from the Cy233 attached to its peripheral bus and sends the message message (with source address) to the serial network as follows:

00 00 00 00 0001

Transfer Command Example:

Assume that data is to be transferred from the serial side of a Cy233 to the serial side of a network composed of Cy233. Network controllers, each connected to the data bus of a Cy233 via various controllers. The Transfer command, 00, received on the serial Cy233 channel will transfer bytes 0001 at a time, over the parallel bus to the local Cy233 which will then provide a Read signal consisting of 0001 0001 followed by the local address specified as 00 00 00 00 0001. The data byte is then sent as the message data and the Cy233 appends a message return message (0001). Thus the example from the command:

00 00 00 00 0001 (written address = 0001)

would be transferred three data bytes, 00, 00, and 00 to the Cy233 with appropriate handshaking. The Addresses are binary numbers, hence a three ASCII equivalent is 0001 to convert the actual address input to the Cy233 would have the value:

00 00 00 00 0001 (Addresses shown in hex ASCII)

A Cy233, with local address 00, upon receiving three three data bytes would verify three messages and transmit them to the serial line as follows:

00 00 00 00 0001 (00 is local Cy233 address)
00 00 00 0001
00 00 00 0001

Soft Key Acknowledgement via Query in a Network

The use of the query command for receipt verification of such softkeys was discussed in the section on Soft-keys. The query can be used for the same purpose in a Cy325-Cy326 network as will be described here. The soft-keys are enabled by using the `KEY ON` via M-2, from and are individually acknowledged. For applications where it is imperative that all keys be correctly interpreted, it is possible to verify every softkey before taking any action on the soft-key message.

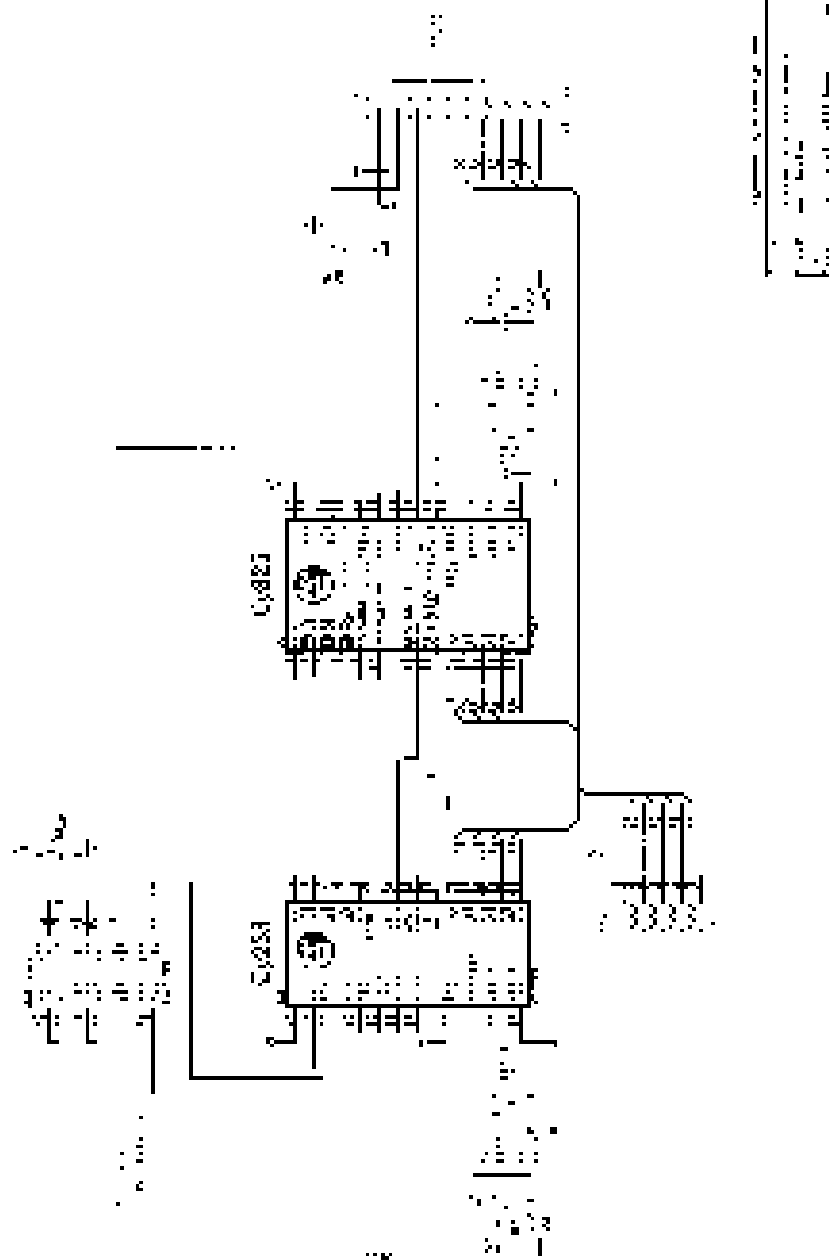
The 'Q' query command will query the Key Image register, and will return an ASCII string of '1'- '0's 'xxxxxx' where xxx is the 2-character hex-ASCII code for the soft-key image in the register, followed by the 'Q' suffix that is appended to the query, then terminated by the carriage return, CR. The soft-key messages and verification query messages are shown below for the six soft-keys. The messages go across the network serial channel, but are after they have been taken through parallel data bus into the Cy325 and have been output as a serial message, with a modified Read message format consisting of an odd '1' followed by a two byte hex-ASCII address. Answer in the following that the servers 'xxx' with address 3 is attached to the Cy325 as controller, the 'x' generating the soft-key messages.

Cy325 soft-key address	Server Soft Key Message	Cy325 Verify message	Return Key Image value
1	'0011xxxx'	'10011xxxxx'	01111110
2	'00111xxx'	'100111xxxx'	01111111
3	'0011xx0x'	'100112xxxx'	01111011
4	'0011xx1x'	'100113xxxx'	01111011
5	'00115xxx'	'100115xxxx'	01111111
6	'00114xxx'	'100114xxxx'	01111111

After the Soft Key Ack message has been correctly acknowledged via the acknowledgment 'X' message command line (see list), the query can be followed query of the Key Image, via the 'Q' query command, shown in the following.

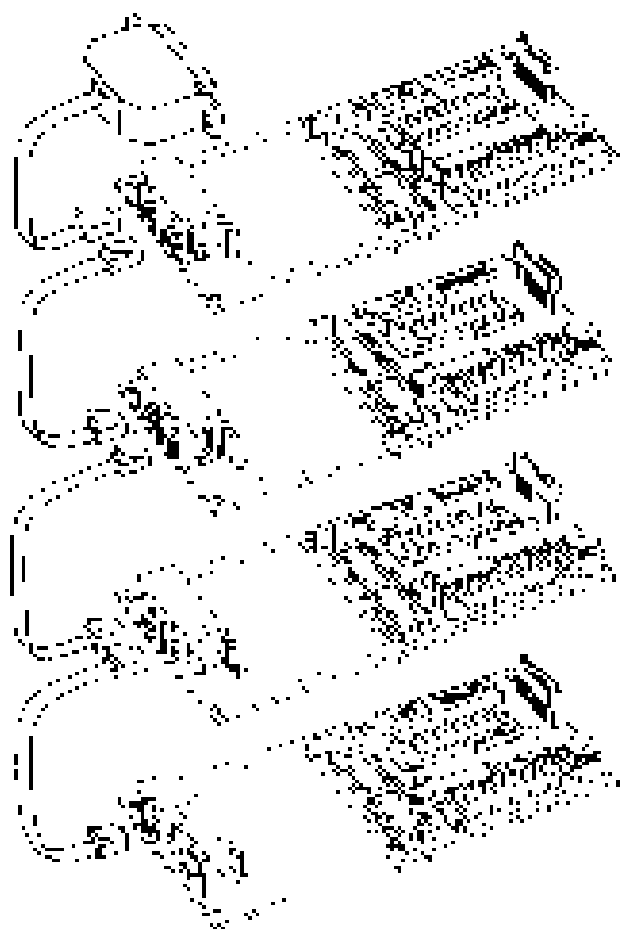
Notice that the messages appended to the network serial bus are sufficiently needed to uniquely identify every soft-key in the network. The 'Q' header and two terminated properly leave each message, while the network address, 02, uniquely identifies which Cy325-Cy326 station is generating the message, and the '1n' data identifies softkey 'n', while the '1xxx' information identifies the response to the query of the Key Image data register. More information of the Cy325 is available from Cycom, Inc.

Cy325 - Cy298 Schematic Diagram



Cyb-003 Prototyping Board in a Network

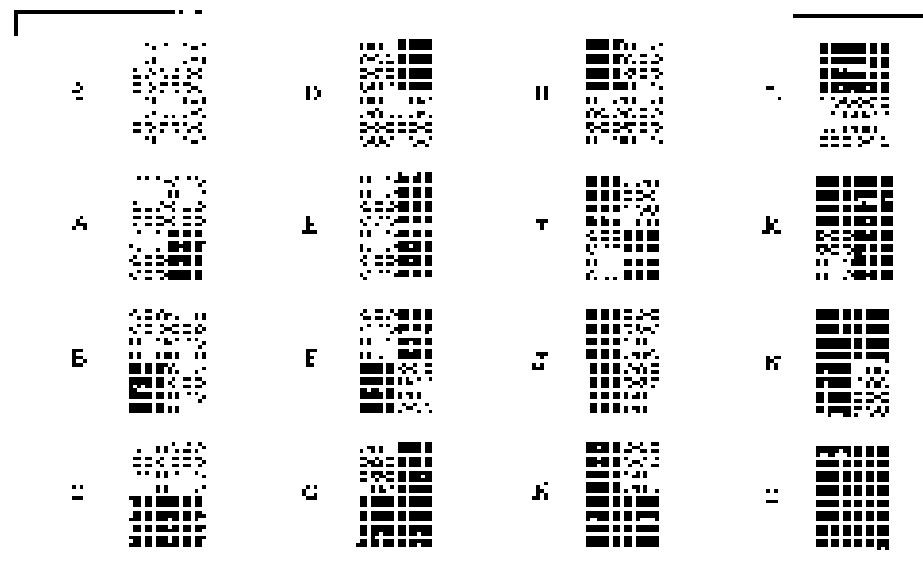
The Cyb-003 can be used in a local network. Connections between boards can be via active cables or direct cables with termination-type tasks. The direct cables can carry both signals and power or clock signals.



14 CY325 Built-in 'Special' Font 14

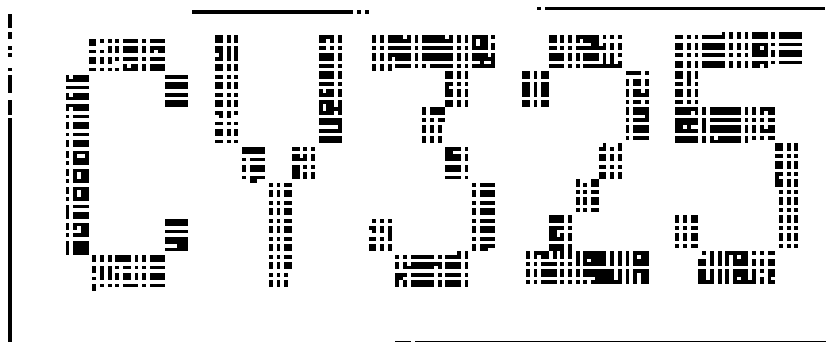
The CY325 allows the user to design any number of special fonts and download them, character by character, via the Font command. When the CY325 powers up (or after T-0) it will contain fifteen special font characters that can be used to form a large and unique character set. The first one, which is thirty feet high when displayed, is required for some wall display displays. These characters are much larger than the 'large' characters that are invoked via the 'L' command. 'Large' characters are formed from one stroke of the special font characters, while 'large' characters are usually formed by two strokes. For the CY325, twelve large characters are built-in, entitled with common abbreviations: window, wrap, stop, wait. 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 'paint' a character. For the giant character, then, the first stroke of one is written as a second stroke of another, forming the character, and so on.

The built-in special font effectively provides "giant pixels" that use 640 normal pixels to draw. Each special font character will contain four pixels. If the character is drawn, the pixel size will be 160 pixels by 160 pixels. The character is identical to the "normal" character, but the pixel size is 160 pixels by 160 pixels. The following combinations are shown below. These can be used to build any character in the same manner that small pixels are used to draw characters. The primary difference is that characters will be giant pixels and visible from across a room.



Examples of 'Special' Font

The default Apple font contains 100 glyphs. Four large pieces of font "character" are given the size of 12, 14, 16 and 18. Four characters use the special font patterns for the large sizes. Then 14 characters are given to create many super and sub. The 100 glyphs show how to use the given letters "00000".

[illegible]

The "shift out" code, 0x1B, allows special font patterns to be selected via 1-bit ASCII character codes. Special font patterns can be selected via 1-bit character codes, although the 1-bit shift code, 0x1B, is not corresponding to 01 in 164 or 165. The ASCII default special character can be selected via the character table ASCII value 0x1B. There is no way to set the 1-bit character code with value 0x1B. A code, 0x1B, although the value is justified in this form, the character code is set to the shift out code, which character code, 0x1B, "shift out" may need to be set to the default font, the 1-bit code can be used to print the default font. You can value the special font while retaining the default and by using the 01 command with character value 0x1B or 01. 01 commands with character value 0x1B are used to set 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 instrument and other systems are considering them as the primary user interface. The new "instrument size" are distinguished from "laptop PC" LCDs in several ways. While "laptop PCs" usually require 600 x 300 pixel displays to be compatible with DOS software, such 1200 are generally too large and too expensive (over \$1000) for instruments, handheld units, and point-of-sale terminals.

Instrument size LCDs provide 240 x 64 pixel graphics (rows x 40 characters), or smaller, costing around \$60 in quantity, they are ideal for instruments and display panels designed to display values, messages, and to display a few, high-contrast images as opposed to pages of text, as in word processors, etc. They are smaller in size than PC type LTMs and are available in superalumin 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 few that exist are 1201 and 1202. The LCD controllers are extremely difficult to program. So the good news that super attractive graphics 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, striving to provide the best possible user interface, will have to take the long hard instrument design and instead develop many sub-routines to developing routines which display text, plot traces, draw boxes, create icons, build bargraphs, show logic waveforms, support serial communications, and so forth. In other words, instrument designers will not be able to concentrate the effort in order to utilize the attractive new displays.

To minimize such "dead-end" situation, Datacube Micro Systems designed the new 1203 LCD Windows Controller specifically for instrument designers producing point-of-sale terminals. Along with the goal of reducing display design from a good fraction of a man year down to only one week, the 1203 can simply provide serial input to the 1201 or 1202, serve as the major system element, bring together the display functions, the user input function, and the providing of in values to the system processor. The 1203 even has built-in support for RS-172 based local area networks, even allowing a chain of instruments or point-of-sale terminals to be linked to an RS-172 bus channel or equivalent.

Text and Graphics in Windows

While text display is generally not too difficult to do now, the simple task of plotting a curve on an LCD can be a fairly undertaking, involving the solution of several equations applied to transform from the device coordinate system in a box of a window, to the display coordinate system of the user, then to the display coordinate system seen by the hardware, which is often x - y reversed, and byte oriented. The programmer has to write operations to handle 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 clearly additional knowledge for the designer, somewhat analogous to the need to calculate values of $\tan$ values based on various specifications every time you simply want to read on the accelerometer. Thus, to track back the cycle period in 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 coordinate work areas or "windows" is both functionally and aesthetically desirable. Therefore the system has been designed to enable partitioning windows on graphics LCDs. Each window possesses its own coordinate system with its origin located on the lower left corner of the window as depicted. A maximum window can be open on a screen at any time, the "current" or "active" window defines the current coordinate system. Pixel plotting operations map into the window system and the cycle was performed intelligently based on this window id system.

Any of our default or built in windows can be selected via single commands, or a user defined window can be specified by a single command. Both text and graphics can be written into the current window with automatic window management, allowing, for text and graphics to be independently written, moved, or overlaid in a single window, or even 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, involve automatic program generation and logic control of display. For example, photographs (imagery) can be generated by simply specifying the heights of the array as appropriate.

A short summary of typical features includes:

- Customized and display window of operation
- Support of 3 bit parallel interface
- Write-in or user defined window
- Window-relative text and graphics
- Large waveforms (write-in (raw photo)
- Large numbers and user-defined forms
- Graphics overlaid by other windows
- Communications between several unit (serial)
- "soft-key" support for menu management
- Window support based on Cycle Hardware map.

Menu-based Instrument Programming

In addition to graphic information display, VCB-based instruments often provide menu-based programming functions, so the system elements "learn-key" themselves to provide menu programming capability. The CY225 supports up to six "softkeys" which can be assigned the instrument computer to display the "meaning" of each key on the VCB. When the user pushes a particular key the result is transmitted to the instrument via a serial built-in protocol which is then a) displayed, uniquely identified, and acknowledged, b) allowing the user to take some choice or otherwise use the softkeys, with the system computer managing the response and updating the display.

Examples of Instrument Display Design

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

1. Multiple waveforms
2. Graph outputs
3. Histograms
4. Special icons or symbols

While each of these graphics displays is complex and difficult to implement using the low-level capabilities available with today's VCB, the CY225 allows user-defined hi-level commands that perform these various display functions. Other one command to the Cyax will generate hundreds of commands to the low-level built-in VCB controller, hundreds of commands that the instrument designer no longer need be concerned with.

In order to demonstrate how the CY225 can make your life easier, we will design two prototype instruments. The first will use a CY225 to capture hi-speed digital signals and display logic waveforms on a window on the VCB, while the second instrument will be used as an investigative transducer, with the goal of displaying low transducer output on an VCB window. One instrument will use a serial interface to the CY225 and the other will use an 8-bit bus parallel interface to the Cyax.

Instrument #1: Serial Interface to Logic Display:

We begin with a very simple design (Figure 1) in which the Cy12 is used primarily to display local windows and windows. The system manager will use its own host to interface locally to the Cy12. Windows will be transferred serially to the Cy12 to open windows and display messages. While the local windows will drive six 120 pins of the Cy12. Assume that the instrument will be an 8051, although any processor will do. The instrument system is shown will perform the following tasks:

- ```

1 setup the Cy125 operator's table
2 update the manager window
3 display messages in the window
4 enable the logic waveform window
5 capture logic bus data using FIFO hardware
6 with a the Cy125 Waveform Area
7 and the logic waveform area in the Cy125

```

The first task consists of reformatting the C-125 to hold data for manual communications and also for the C-125 in the logic conveyor mode of operation. To put it on the C-125 is pulled in. The head data will be set to 0000, at least for now, and if pin 21 is in writing the C-125 effectively responds to logic conveyor returns and selects the head bit. If the carriage returns are transmitted when the C-125 automatically powers up with logic conveyor mode selected that will not work. However, if there has been a change, then a logic command can be applied to select conveyor operation. Note that the pin used to input logic conveyor can be used for other purposes with an "auto-pay" operation.

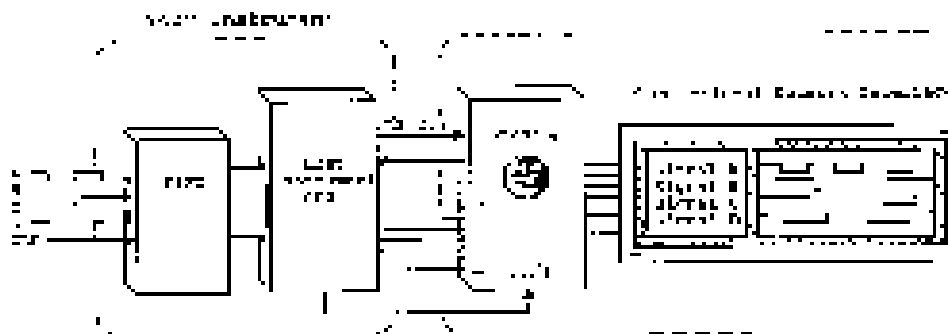


Figure 1. A. 3440 system with serial interface to the DTH.

The message window can be created in two different ways. Since the CY255 necessitates 256 'built-in' windows, it is usually easiest to simply create an appropriate window or viewport by sending the request in command 10 (page 1), where 10 is the ASCII code for 0x0021 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 '0 0000000' creates the message window that is one fourth of the screen wide and six lines deep. Before sending the viewport command the system must be placed in the command mode by setting the control character (031h).

After the message window has been created, we wish to display the six messages in the window. We send the Ctrl-F command (06h) to return the system to the display mode of operation, then we simply send the six messages:

```
Signal 1 error
Signal 2 error
Signal 3 error
Signal 4 error
Signal 5 error
Signal 6 error
```

The next task is to create the viewport window and this is done by sending the ctrl-c command (03h) after command 10 and by sending the 0x000000 command to select the window for the logic simulation. At this point the EBC screen looks like:



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

One benefit of this is that systems designers possess the instantaneous necessary for measurement systems design, while we are simply trying to relieve them of the burden of display design at the low level by offering a high level display controller. This we know for the following: First the instrument was supposed to let all logic users know that they were going to use the EBC in parallel.

After sending six (or more) logic signals in parallel, the instrument EBC simply places the six desired signals on the system waveform pins (i.e. 00001) and leaves the scan plane's (see pg. 11) to tell the users to accept and display these signals in the waveform window, leaving that to a low level side of the window. The scan window can be left low or pulled high to stroke such signals as signals into the CY255. If there is no change in the signals, pin 7 on the logic can be used to strobe the

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

In this way an additional sequence of waves can appear on the right of the waveform window and with at the left of the window, or any time the instrument CPU can divide the screen, change windows, and change waveforms, then return to the waveform window, and continue the waveform display operation. Any of the cycle's 255 built-in windows or any user-specified window can be used to display waveforms, in fact two or more separate waveforms at once can be action by carefully controlling the sequence of commands. If the window is less than six lines deep, the signal will automatically truncate the display and only show the portion of lines that fit in the window. Various methods of display can be placed over the waveforms that do not have relation to the window while the waveforms alone function as waveforms. The window can at any time be moved by commands from the system CPU.

Upon completion the design of software, further work is in worth providing the commands necessary to achieve the display shown in Figure 1:

|                                                   |                                   |
|---------------------------------------------------|-----------------------------------|
| Select waveform window:                           |                                   |
| Ctrl-C                                            | enter command mode                |
| 9 0000000                                         | select waveform window (from 255) |
| Ctrl-D                                            | enter display mode                |
| Display message:                                  |                                   |
| W  at 1 wave                                      | send ASCII messages               |
| W  at 2 wave                                      |                                   |
| Signal 3 wave                                     |                                   |
| Signal 4 wave                                     |                                   |
| Signal 5 wave                                     |                                   |
| Signal 6 wave                                     |                                   |
| Return waveform window:                           |                                   |
| Ctrl-C                                            | enter command mode                |
| 9 0000000                                         | select waveform window (from 255) |
| Display Waveforms:                                |                                   |
| p 00000000 00000000 00000000 00000000             |                                   |
| 00000000, waveform data from window using bit 15. |                                   |

That's all. Five commands!

Unlike the above procedure for generating waveforms display with the low level approach, the necessity of the pixel level. Months of work are reduced to an hour of coding time or less.

## Instrument #2: Parallel Interface to System Element

Having seen how simple it is to obtain sophisticated displays in a short time, we will now design Instrument number 2 to utilize some of the C6325's built-in system functions. Instead of the serial interface, the system processor will use an analog-to-digital converter to the system element. The system element will then be available to the system processor as a general DAST 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 a "back-key" input or a pin of the system 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.

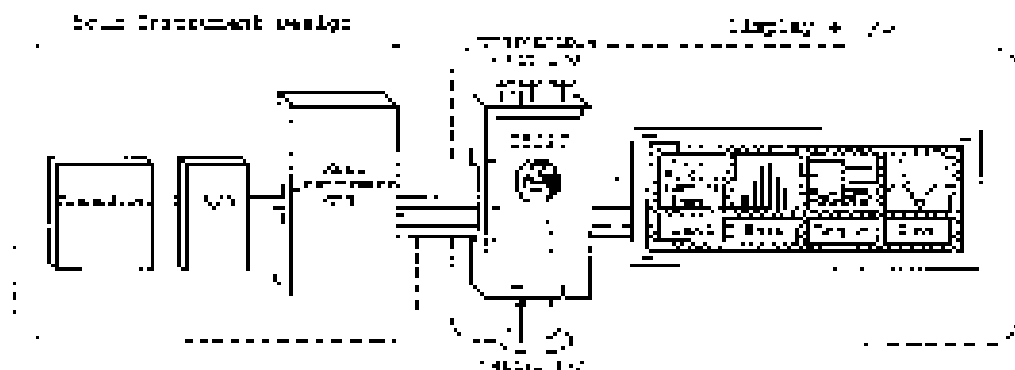


Figure 3. Example Instrument number 2 uses a parallel interface to the system element and generates four display windows.

The features that we will concentrate on are the following:

1. The parallel interface to the C6325
2. The generation of waveforms
3. The display of analog waveforms
4. The use of the system's processor I/O lines.

## Parallel Interface to the Cy325

The 4-bit bus interface uses a ready signal and a write line with a 1-to-0 control signal to enable the Cy325 data onto the Cy125 bus. Since 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 support it, approximately two times as fast. However, the default operation is designed to allow bus technology to be completed under software control of the system processor.

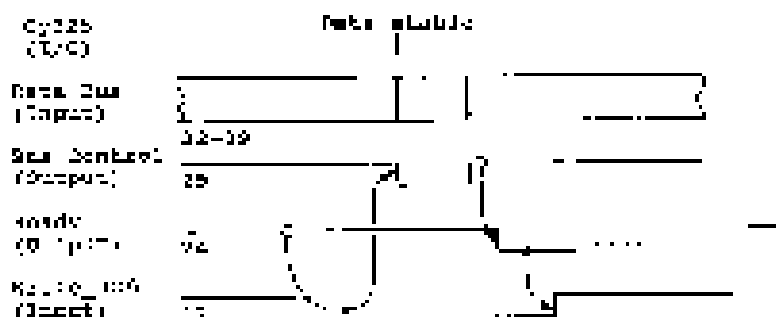


Figure 4. Parallel data transfer signals and timing sequence.

The instrument processor shows the Ready line (pin 22) to see if the Cy125 is ready to accept a character on the Data Bus. If READY is high the instrument CPU drives the M325\_106 line (pin 13) low to request a transfer. When the Cy125 successfully writes the CPU data into (pin 20) low, the CPU places the 4 bit data on the bus. The Cy125 acknowledges receipt of the data by returning the M325\_106 line high and driving the Ready line low. The CPU then checks the bus and waits until the Ready line returns high to repeat the sequence for the next transfer.

After the parallel interface has been implemented, the Cy125 is controlled by sending commands to it in serial fashion. The Cy125 goes on in the Display mode of operation with a default window equal to the video screen. Any characters sent to the Cy125 will then be converted to ASCII characters and displayed on the screen accordingly. While this is very convenient for the simplest systems, the usual starting procedure will be to issue the serial command to enter the command mode and then to create the appropriate window and send up messages. Serial control commands will enter command display mode. In command mode characters are interpreted in display mode characters are simply displayed at the current video location in the current window. The cursor is managed automatically so that characters that hit the right side of a window automatically appear on 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 wrap/scroll flag in the window status register.

## Automatic Bargraph Generation

Since bargraphs are universally understood diagrams, they provide a very effective means of displaying information. For this reason the GY25 supports a Bargraph or Histogram command that can use the following format:

```
[H] n, y1, y2, s1, ..., sn (x)
```

where

- H is the Histogram command words (ASCII 'H' = 48h) followed by at least one space (10h).
- n is the decimal or hexadecimal number of arguments, followed by a comma.
- y1 is the first y-coordinate value (vertical or horizontal) followed by a comma.
- yn is the last argument value, followed by a space.

The 'H' command will plot the y-values, how many bars are to be plotted in the current window, and specify the height of each bar. The GY25 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 specification mode bit in the window status register. An ASCII character can be drawn vertically in one window and horizontally in another. The command sequence:

```
enter command mode
H 003H000
 4,1,6,16,26,36,47,57,68,79,89
 \
 p = a bar 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 recorder, we now invert/reverse the window-relation display of analog waveforms. The GY25 provides the primary PLOT command, a Plot-Point command and a Plot-String command. The Plot-Point command has the format P x,y,xxx where (x,y) is the coordinate of the pixel to be plotted. For plotting multiple pixels such as those obtained by a serial sampling or an A/D converter, the Plot-String command is preferable.

The Plot-String command has the following format:

```

where (1) n = y, 32, 72, ..., 255
 (2) is the ASCII Plot-string mode.
 n = is the binary number of points to follow.
 yn = is the binary value of the n-th data point.

```

To plot an analog waveform in the current window, the instrument SMU simply sends the "P" character (ASCII 80) 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,  $y_1$ , will be plotted at the local coordinate  $(1,y_1)$  while the second will appear at  $(2,y_2)$  until the nth data point appears at  $(n,y_n)$ . The host can sample a value and send it back at once or sample over the points as they are sampled. However the 16125 expects all a point to be sent before another can and will be forced. The Plot String uses a 16125 code instead of a 16124 replacement.

The DCE screen, shown in Figure 6, illustrates the results of the above operations (including a other logic waveform). Less than a dozen commands were required to produce this display!

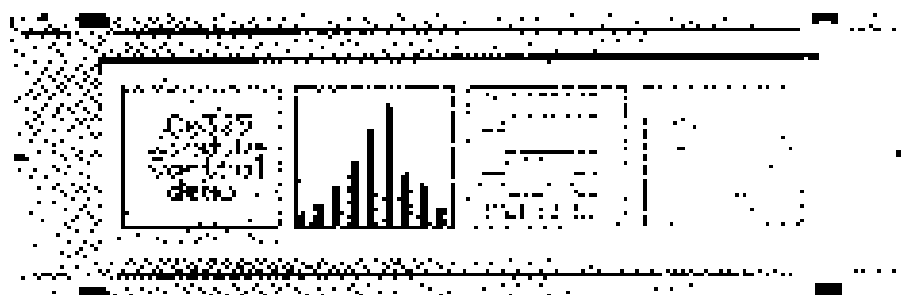


Figure 6. An example display achieved with a dozen commands.



Figure 7. Illustrating two 16124 analog waveform displays.

The CY125 possesses eight pins that are multi-purpose. They can be I/O or control with a maximum of 255 I/O's. In a following article that runs from page 245, System can be connected to a serial network, connected to an IBM-PC's COM1 channel. In the current examples, these multi-purpose pins simply provide eight general purpose I/O lines. CY125 commands exist that will allow the user to individually set or reset each of the 8 I/O lines and also to write or read 8 bits to the bus (8 parallel).

## Instrument Design Summary

The two example instruments designed show how far we've come in the way of doing an instrument program and is depicted in Figure 2. Some things provided as a rich interface and on the user with some sophisticated displays. Acquiring 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 more detailed and specific measurements require the use of general types of displays that are designed to appear somewhat analogous and logic in reading operations.

The CY125 possesses several other display functions that are very useful. In particular, the ability to create several forms for icons or other objects on specific layers. As in the other display cases, only one CY125 command is required to define a new symbol or icon, and the CY125 can even create special double-size symbols as a combination of four such user-defined symbols. Still another useful feature supported by the device is the following. Each window can be considered an instance of a display plane, with a layer 0 and 1 plane, and one of the two planes can be manipulated in several ways within a window.

Finally, the CY125 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. With CY125, the user can do it all, even hundreds of specializations to be easily specified.

## Network Interface and Soft-Keys

The CY125 provides both PC parallel and serial interfaces. In addition, the CY125 provides a convenient interface to Cybarratic's Cybarratic Host Interface Network Controller. This allows up to 255 users to be connected to a single serial I/O port (one of COM1 or an IBM-PC's). The CY125 is designed to support up to 255 "clients" in such a network. This allows a host computer to display the "meaning" of each key on the bus. When the user pushes a particular key, the result is transmitted to the host. A special keyboard protocol ensures that all keys are uniquely identified and "known" to the host. The following numerous modes of operation were chosen as standard and the softkeys, with a central host managing the responses and updating the displays. This feature allows applications such as character array entry, etc., to be implemented as key codes.

## Application Note 2:

### The CY325 In a CY293 Based Local Area Network







```

;-----

```

```

 mov [dword ptr 0], 0

```

```

;-----

```

```

mov Ebx, 0
mov Ebx, 0
mov Ebx, 0
mov Ebx, 0
mov Ebx, 0
mov Ebx, 0

```

```

;-----

```

```

mov Ebx, 0
mov Ebx, 0
mov Ebx, 0
mov Ebx, 0
mov Ebx, 0
mov Ebx, 0
mov Ebx, 0
mov Ebx, 0
mov Ebx, 0
mov Ebx, 0
mov Ebx, 0
mov Ebx, 0
mov Ebx, 0
mov Ebx, 0
mov Ebx, 0

```

```

; open window
; display mode
; shift out

```

```

; shift out

```

```

; new viewport
; display mode
; shift out

```

```

; shift out

```

```

;-----
; perform simulation on the first and main
; panels in the window
;-----

```

```

 mov [dword ptr 0], 0

```

```

mov Ebx, 0
mov Ebx, 0
mov Ebx, 0
mov Ebx, 0

```

```

;-----
; setup panel 1 sub
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0

```

```

;-----
; setup panel 2 sub
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0
; mov Ebx, 0

```











```

size 10 setup:
db 1E10,10,10,0,1E10,0,1E10,0 ; setup window 0
db 10,0,0,0,10,0,0,1E10 ; draw box on 0

size 11 setup:
db 0000
db 10,00,00,00,00,1E10,1E10 ; box command

size 12 setup:
db 0000
db 10,10,00,00,00,1E10,1E10 ; box command

size 000 setup:
; setup for 'lines'
db 0000 ; 1E10,00,1E10,00 ; fill 100 screen
; 10,00,00,1E10,00 ; setup window
; 0000 ; enter clipping mode
; 1E10
; 1E10,10,00,00,00,1E10,1E10
; Global of window 0000,00
; 1E10
; 1E10,10,00,00,00,1E10,1E10
; 1E10

; 0000,1E10,00 ; draw graphics
; 10,10,00,00,00,1E10,1E10 ; draw box
; 1E10,00 ; move graphics to 100

size 1 setup:
db 0000
db 10,00,00,00,00,1E10,1E10 ; draw box

size 000 setup:
db 0000
db 10,00,00,00,00,1E10,1E10

```



```

do_show_optv:
:
small black screen
row Optv: #Row=0
small row n delay
row Optv: #Row=0
small row n delay
row Optv: #Row=0
row n delay
:
do_show_xi:
:
small black screen
row Optv: #Row=0
small row n delay
row Optv: #Row=0
small row n delay
row Optv: #Row=0
small row n delay
row Optv: #Row=0
row n delay
:

```

```

=====
send_low_string : do tail call
=====
; R0 : zero offset from pointer
; R1, R2, R3 : get into this pointer
; R4 : pointer pointer
;
; send send to LCD : send by 0 to 0xFF
;
; R0, R1, R2, R3, send the string : do tail call
=====
; send_low_string
;
; R0 : zero offset from pointer
; R1, R2, R3 : pick up from this ptr
; R4, R5, R6, R7, R8, R9, R10 : receive all 15
; R11 : return value according all message
;

send_Ctrl0:
mov r1, 01 : Ctrl-0 command
; R0, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10
; R11

send_Ctrl1:
mov r1, 01 : Ctrl-0 display
; R0, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10
; R11

send_low:
; send_low_string
; R0, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10
; R11
; send send to LCD

```

[illegible]





```

Row 1:
db 0x00,0x00,0x00
db 0x00,0x00
db 0x00,0x00,0x00,0x00,0x00,0x00
db 0x00,0x00,0x00
db 0x00,0x00,0x00
db 0x00,0x00,0x00
db 0x00,0x00,0x00
db 0x00,0x00,0x00
db 0x00,0x00,0x00
db 0x00,0x00,0x00
db 0x00,0x00,0x00
db 0x00,0x00,0x00

```

```

Row 2:
db 0x00,0x00,0x00
db 0x00,0x00,0x00
db 0x00,0x00,0x00,0x00,0x00,0x00
db 0x00,0x00
db 0x00,0x00,0x00,0x00,0x00,0x00
db 0x00,0x00
db 0x00,0x00,0x00
db 0x00,0x00
db 0x00,0x00,0x00
db 0x00,0x00,0x00
db 0x00,0x00,0x00
db 0x00,0x00,0x00
db 0x00,0x00,0x00
db 0x00,0x00,0x00

```

```

Row 3:
db 0x00,0x00,0x00,0x00
db 0x00,0x00,0x00
db 0x00,0x00,0x00
db 0x00,0x00,0x00,0x00

```

```

Row 4:
db 0x00,0x00,0x00,0x00
db 0x00,0x00,0x00,0x00
db 0x00,0x00,0x00,0x00
db 0x00,0x00,0x00,0x00

```

```

Row 5:
db 0x00,0x00
db 0x00,0x00,0x00,0x00
db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00

```

```

Row 6:
db 0x00,0x00
db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
db 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00

```









### Line Keys ::

6 G. : data points

[illegible]

**Abstract**

## Appendix A:

### Electrical Specifications

#### Absolute Maximum Ratings

ambient temperature under bias.....0°C to 70°C  
Storage Temperature.....-55°C to +125°C  
Voltage on any pin with respect to GND.....no more than  
Power Dissipation.....0.2 Watts

TABLE 1. ELECTRICAL CHARACTERISTICS

(TA = 0°C to 70°C, VDD = 1.8V TYP)

| SYM | PARAMETER          | HTS | MAX CNTS | REMARKS                   |
|-----|--------------------|-----|----------|---------------------------|
| VDD | pin supply voltage | 1.8 | V        |                           |
| VTH | input high level   | 1.5 | VDD      | 0 (0.5V for HSTL, HSTL2)  |
| VIL | input low level    | 0.5 | 0.4      | V                         |
| ICD | data bus leakage   | 10  | uA       | high impedance state      |
| VOH | output high level  | 2.4 | V        | IOL = -10 uA              |
| VOL | output low level   | 0.4 | V        | IOL = 10 uA               |
| FCY | crystal frequency  | 3.3 | 12       | MHz<br>see clock circuits |

#### Electrical Conventions

All I/O pins signals are listed in 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 "N" after the pin name, i.e., NCT.

All I/O pins except the pins that 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 GND pin drives it both high and low.

The data bus is bidirectional. All is 3V3T/3.3V using non-saturating modes. Some data bus signals are positive logic.

## Reset Circuitry

The Sestart pin (pin 14) must work as held high when powerup or powerdown initialization (the Cy225). This may be accomplished via the use of a 4.7  $\mu$ F capacitor, as shown in Figure 1. Sestart must be high for a brief after powerup, otherwise, on powerup, once the Cy225 is running, Sestart need only be high for about 10 msec (100  $\mu$ s normally).

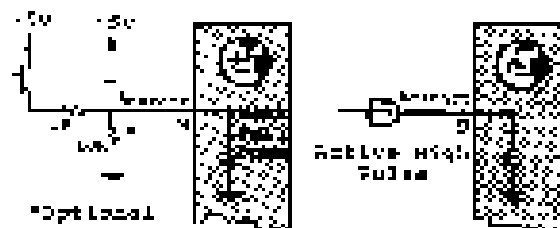
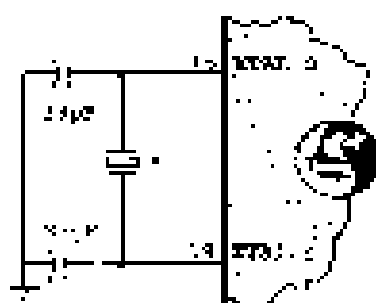


Figure 1 a) Sestart Circuitry, b) External Reset.

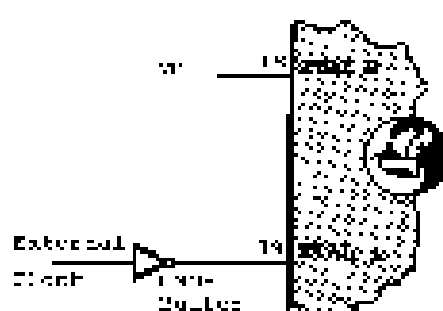
## Clock Circuits

The Cy225 may be operated with crystal or external clock circuits. All timing discussed in this manual assumes a 11.059 MHz surface resonant crystal. From 11.0 MHz, such as a 74C9100A PLLC or equivalent will work fine. The crystal requires an 11 MHz clock in order to generate the standard baud rates, although any crystal in the 11.059 MHz range can be used with the adaptive baud rate, 230400, the timing resolution limits of the Cy225.



11.059 MHz for Standard Baud Rates

CRYSTAL



Any 11.059 MHz clock with 100pF capacitor

EXTERNAL

Figure 2 Clock Circuits for Cy225

# Appendix B: ASCII-Decimal-HEX Conversion Table

| DEC | HEX | DEC | HEX | DEC | HEX | ASCII | DEC  | HEX |      |
|-----|-----|-----|-----|-----|-----|-------|------|-----|------|
| 0   | 00h | 51  | 33h | 102 | 66h | 153   | 99h  | 225 | 201h |
| 1   | 01h | 52  | 34h | 103 | 67h | 154   | 9Ah  | 226 | 202h |
| 2   | 02h | 53  | 35h | 104 | 68h | 155   | 9Bh  | 227 | 203h |
| 3   | 03h | 54  | 36h | 105 | 69h | 156   | 9Ch  | 228 | 204h |
| 4   | 04h | 55  | 37h | 106 | 6Ah | 157   | 9Dh  | 229 | 205h |
| 5   | 05h | 56  | 38h | 107 | 6Bh | 158   | 9Eh  | 230 | 206h |
| 6   | 06h | 57  | 39h | 108 | 6Ch | 159   | 9Fh  | 231 | 207h |
| 7   | 07h | 58  | 3Ah | 109 | 6Dh | 160   | A0h  | 232 | 208h |
| 8   | 08h | 59  | 3Bh | 110 | 6Eh | 161   | A1h  | 233 | 209h |
| 9   | 09h | 60  | 3Ch | 111 | 6Fh | 162   | A2h  | 234 | 20Ah |
| 10  | 0Ah | 61  | 3Dh | 112 | 70h | 163   | A3h  | 235 | 20Bh |
| 11  | 0Bh | 62  | 3Eh | 113 | 71h | 164   | A4h  | 236 | 20Ch |
| 12  | 0Ch | 63  | 3Fh | 114 | 72h | 165   | A5h  | 237 | 20Dh |
| 13  | 0Dh | 64  | 40h | 115 | 73h | 166   | A6h  | 238 | 20Eh |
| 14  | 0Eh | 65  | 41h | 116 | 74h | 167   | A7h  | 239 | 20Fh |
| 15  | 0Fh | 66  | 42h | 117 | 75h | 168   | A8h  | 240 | 210h |
| 16  | 10h | 67  | 43h | 118 | 76h | 169   | A9h  | 241 | 211h |
| 17  | 11h | 68  | 44h | 119 | 77h | 170   | AAh  | 242 | 212h |
| 18  | 12h | 69  | 45h | 120 | 78h | 171   | ABh  | 243 | 213h |
| 19  | 13h | 70  | 46h | 121 | 79h | 172   | ACH  | 244 | 214h |
| 20  | 14h | 71  | 47h | 122 | 7Ah | 173   | ADh  | 245 | 215h |
| 21  | 15h | 72  | 48h | 123 | 7Bh | 174   | AEh  | 246 | 216h |
| 22  | 16h | 73  | 49h | 124 | 7Ch | 175   | AFh  | 247 | 217h |
| 23  | 17h | 74  | 4Ah | 125 | 7Dh | 176   | B0h  | 248 | 218h |
| 24  | 18h | 75  | 4Bh | 126 | 7Eh | 177   | B1h  | 249 | 219h |
| 25  | 19h | 76  | 4Ch | 127 | 7Fh | 178   | B2h  | 250 | 21Ah |
| 26  | 1Ah | 77  | 4Dh | 128 | 80h | 179   | B3h  | 251 | 21Bh |
| 27  | 1Bh | 78  | 4Eh | 129 | 81h | 180   | B4h  | 252 | 21Ch |
| 28  | 1Ch | 79  | 4Fh | 130 | 82h | 181   | B5h  | 253 | 21Dh |
| 29  | 1Dh | 80  | 50h | 131 | 83h | 182   | B6h  | 254 | 21Eh |
| 30  | 1Eh | 81  | 51h | 132 | 84h | 183   | B7h  | 255 | 21Fh |
| 31  | 1Fh | 82  | 52h | 133 | 85h | 184   | B8h  | 256 | 220h |
| 32  | 20h | 83  | 53h | 134 | 86h | 185   | B9h  | 257 | 221h |
| 33  | 21h | 84  | 54h | 135 | 87h | 186   | BAh  | 258 | 222h |
| 34  | 22h | 85  | 55h | 136 | 88h | 187   | B Bh | 259 | 223h |
| 35  | 23h | 86  | 56h | 137 | 89h | 188   | BC h | 260 | 224h |
| 36  | 24h | 87  | 57h | 138 | 8Ah | 189   | BD h | 261 | 225h |
| 37  | 25h | 88  | 58h | 139 | 8Bh | 190   | BE h | 262 | 226h |
| 38  | 26h | 89  | 59h | 140 | 8Ch | 191   | BF h | 263 | 227h |
| 39  | 27h | 90  | 5Ah | 141 | 8Dh | 192   | C0h  | 264 | 228h |
| 40  | 28h | 91  | 5Bh | 142 | 8Eh | 193   | C1h  | 265 | 229h |
| 41  | 29h | 92  | 5Ch | 143 | 8Fh | 194   | C2h  | 266 | 22Ah |
| 42  | 2Ah | 93  | 5Dh | 144 | 90h | 195   | C3h  | 267 | 22Bh |
| 43  | 2Bh | 94  | 5Eh | 145 | 91h | 196   | C4h  | 268 | 22Ch |
| 44  | 2Ch | 95  | 5Fh | 146 | 92h | 197   | C5h  | 269 | 22Dh |
| 45  | 2Dh | 96  | 60h | 147 | 93h | 198   | C6h  | 270 | 22Eh |
| 46  | 2Eh | 97  | 61h | 148 | 94h | 199   | C7h  | 271 | 22Fh |
| 47  | 2Fh | 98  | 62h | 149 | 95h | 200   | C8h  | 272 | 230h |
| 48  | 30h | 99  | 63h | 150 | 96h | 201   | C9h  | 273 | 231h |
| 49  | 31h | 100 | 64h | 151 | 97h | 202   | CAh  | 274 | 232h |
| 50  | 32h | 101 | 65h | 152 | 98h | 203   | CBh  | 275 | 233h |

## Appendix C:

### Typical LCD Specifications



## 2.7 Data Model: a Knowledge Base

## CHUCK BEEGACHEE GUIDE

| Page No. | Page Title | Page No. | Page Title | Page No. | Page Title | Page No. | Page Title | Page No. | Page Title |
|----------|------------|----------|------------|----------|------------|----------|------------|----------|------------|
| 1        | Page 1     | 2        | Page 2     | 3        | Page 3     | 4        | Page 4     | 5        | Page 5     |
| 6        | Page 6     | 7        | Page 7     | 8        | Page 8     | 9        | Page 9     | 10       | Page 10    |
| 11       | Page 11    | 12       | Page 12    | 13       | Page 13    | 14       | Page 14    | 15       | Page 15    |
| 16       | Page 16    | 17       | Page 17    | 18       | Page 18    | 19       | Page 19    | 20       | Page 20    |
| 21       | Page 21    | 22       | Page 22    | 23       | Page 23    | 24       | Page 24    | 25       | Page 25    |
| 26       | Page 26    | 27       | Page 27    | 28       | Page 28    | 29       | Page 29    | 30       | Page 30    |
| 31       | Page 31    | 32       | Page 32    | 33       | Page 33    | 34       | Page 34    | 35       | Page 35    |
| 36       | Page 36    | 37       | Page 37    | 38       | Page 38    | 39       | Page 39    | 40       | Page 40    |
| 41       | Page 41    | 42       | Page 42    | 43       | Page 43    | 44       | Page 44    | 45       | Page 45    |
| 46       | Page 46    | 47       | Page 47    | 48       | Page 48    | 49       | Page 49    | 50       | Page 50    |
| 51       | Page 51    | 52       | Page 52    | 53       | Page 53    | 54       | Page 54    | 55       | Page 55    |
| 56       | Page 56    | 57       | Page 57    | 58       | Page 58    | 59       | Page 59    | 60       | Page 60    |
| 61       | Page 61    | 62       | Page 62    | 63       | Page 63    | 64       | Page 64    | 65       | Page 65    |
| 66       | Page 66    | 67       | Page 67    | 68       | Page 68    | 69       | Page 69    | 70       | Page 70    |
| 71       | Page 71    | 72       | Page 72    | 73       | Page 73    | 74       | Page 74    | 75       | Page 75    |
| 76       | Page 76    | 77       | Page 77    | 78       | Page 78    | 79       | Page 79    | 80       | Page 80    |
| 81       | Page 81    | 82       | Page 82    | 83       | Page 83    | 84       | Page 84    | 85       | Page 85    |
| 86       | Page 86    | 87       | Page 87    | 88       | Page 88    | 89       | Page 89    | 90       | Page 90    |
| 91       | Page 91    | 92       | Page 92    | 93       | Page 93    | 94       | Page 94    | 95       | Page 95    |
| 96       | Page 96    | 97       | Page 97    | 98       | Page 98    | 99       | Page 99    | 100      | Page 100   |

Copyright © 2010 John Wiley & Sons, Inc. All rights reserved.

[illegible]

### Character Guidelines

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 | 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 | 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 | 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 | 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 | 220 | 221 | 222 | 223 | 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 | 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 | 248 | 249 | 250 | 251 | 252 | 253 | 254 | 255 | 256 | 257 | 258 | 259 | 260 | 261 | 262 | 263 | 264 | 265 | 266 | 267 | 268 | 269 | 270 | 271 | 272 | 273 | 274 | 275 | 276 | 277 | 278 | 279 | 280 | 281 | 282 | 283 | 284 | 285 | 286 | 287 | 288 | 289 | 290 | 291 | 292 | 293 | 294 | 295 | 296 | 297 | 298 | 299 | 300 | 301 | 302 | 303 | 304 | 305 | 306 | 307 | 308 | 309 | 310 | 311 | 312 | 313 | 314 | 315 | 316 | 317 | 318 | 319 | 320 | 321 | 322 | 323 | 324 | 325 | 326 | 327 | 328 | 329 | 330 | 331 | 332 | 333 | 334 | 335 | 336 | 337 | 338 | 339 | 340 | 341 | 342 | 343 | 344 | 345 | 346 | 347 | 348 | 349 | 350 | 351 | 352 | 353 | 354 | 355 | 356 | 357 | 358 | 359 | 360 | 361 | 362 | 363 | 364 | 365 | 366 | 367 | 368 | 369 | 370 | 371 | 372 | 373 | 374 | 375 | 376 | 377 | 378 | 379 | 380 | 381 | 382 | 383 | 384 | 385 | 386 | 387 | 388 | 389 | 390 | 391 | 392 | 393 | 394 | 395 | 396 | 397 | 398 | 399 | 400 | 401 | 402 | 403 | 404 | 405 | 406 | 407 | 408 | 409 | 410 | 411 | 412 | 413 | 414 | 415 | 416 | 417 | 418 | 419 | 420 | 421 | 422 | 423 | 424 | 425 | 426 | 427 | 428 | 429 | 430 | 431 | 432 | 433 | 434 | 435 | 436 | 437 | 438 | 439 | 440 | 441 | 442 | 443 | 444 | 445 | 446 | 447 | 448 | 449 | 450 | 451 | 452 | 453 | 454 | 455 | 456 | 457 | 458 | 459 | 460 | 461 | 462 | 463 | 464 | 465 | 466 | 467 | 468 | 469 | 470 | 471 | 472 | 473 | 474 | 475 | 476 | 477 | 478 | 479 | 480 | 481 | 482 | 483 | 484 | 485 | 486 | 487 | 488 | 489 | 490 | 491 | 492 | 493 | 494 | 495 | 496 | 497 | 498 | 499 | 500 | 501 | 502 | 503 | 504 | 505 | 506 | 507 | 508 | 509 | 510 | 511 | 512 | 513 | 514 | 515 | 516 | 517 | 518 | 519 | 520 | 521 | 522 | 523 | 524 | 525 | 526 | 527 | 528 | 529 | 530 | 531 | 532 | 533 | 534 | 535 | 536 | 537 | 538 | 539 | 540 | 541 | 542 | 543 | 544 | 545 | 546 | 547 | 548 | 549 | 550 | 551 | 552 | 553 | 554 | 555 | 556 | 557 | 558 | 559 | 560 | 561 | 562 | 563 | 564 | 565 | 566 | 567 | 568 | 569 | 570 | 571 | 572 | 573 | 574 | 575 | 576 | 577 | 578 | 579 | 580 | 581 | 582 | 583 | 584 | 585 | 586 | 587 | 588 | 589 | 590 | 591 | 592 | 593 | 594 | 595 | 596 | 597 | 598 | 599 | 600 | 601 | 602 | 603 | 604 | 605 | 606 | 607 | 608 | 609 | 610 | 611 | 612 | 613 | 614 | 615 | 616 | 617 | 618 | 619 | 620 | 621 | 622 | 623 | 624 | 625 | 626 | 627 | 628 | 629 | 630 | 631 | 632 | 633 | 634 | 635 | 636 | 637 | 638 | 639 | 640 | 641 | 642 | 643 | 644 | 645 | 646 | 647 | 648 | 649 | 650 | 651 | 652 | 653 | 654 | 655 | 656 | 657 | 658 | 659 | 660 | 661 | 662 | 663 | 664 | 665 | 666 | 667 | 668 | 669 | 670 | 671 | 672 | 673 | 674 | 675 | 676 | 677 | 678 | 679 | 680 | 681 | 682 | 683 | 684 | 685 | 686 | 687 | 688 | 689 | 690 | 691 | 692 | 693 | 694 | 695 | 696 | 697 | 698 | 699 | 700 | 701 | 702 | 703 | 704 | 705 | 706 | 707 | 708 | 709 | 710 | 711 | 712 | 713 | 714 | 715 | 716 | 717 | 718 | 719 | 720 | 721 | 722 | 723 | 724 | 725 | 726 | 727 | 728 | 729 | 730 | 731 | 732 | 733 | 734 | 735 | 736 | 737 | 738 | 739 | 740 | 741 | 742 | 743 | 744 | 745 | 746 | 747 | 748 | 749 | 750 | 751 | 752 | 753 | 754 | 755 | 756 | 757 | 758 | 759 | 760 | 761 | 762 | 763 | 764 | 765 | 766 | 767 | 768 | 769 | 770 | 771 | 772 | 773 | 774 | 775 | 776 | 777 | 778 | 779 | 780 | 781 | 782 | 783 | 784 | 785 | 786 | 787 | 788 | 789 | 790 | 791 | 792 | 793 | 794 | 795 | 796 | 797 | 798 | 799 | 800 | 801 | 802 | 803 | 804 | 805 | 806 | 807 | 808 | 809 | 810 | 811 | 812 | 813 | 814 | 815 | 816 | 817 | 818 | 819 | 820 | 821 | 822 | 823 | 824 | 825 | 826 | 827 | 828 | 829 | 830 | 831 | 832 | 833 | 834 | 835 | 836 | 837 | 838 | 839 | 840 | 841 | 842 | 843 | 844 | 845 | 846 | 847 | 848 | 849 | 850 | 851 | 852 | 853 | 854 | 855 | 856 | 857 | 858 | 859 | 860 | 861 | 862 | 863 | 864 | 865 | 866 | 867 | 868 | 869 | 870 | 871 | 872 | 873 | 874 | 875 | 876 | 877 | 878 | 879 | 880 | 881 | 882 | 883 | 884 | 885 | 886 | 887 | 888 | 889 | 890 | 891 | 892 | 893 | 894 | 895 | 896 | 897 | 898 | 899 | 900 | 901 | 902 | 903 | 904 | 905 | 906 | 907 | 908 | 909 | 910 | 911 | 912 | 913 | 914 | 915 | 916 | 917 | 918 | 919 | 920 | 921 | 922 | 923 | 924 | 925 | 926 | 927 | 928 | 929 | 930 | 931 | 932 | 933 | 934 | 935 | 936 | 937 | 938 | 939 | 940 | 941 | 942 | 943 | 944 | 945 | 946 | 947 | 948 | 949 | 950 | 951 | 952 | 953 | 954 | 955 | 956 | 957 | 958 | 959 | 960 | 961 | 962 | 963 | 964 | 965 | 966 | 967 | 968 | 969 | 970 | 971 | 972 | 973 | 974 | 975 | 976 | 977 | 978 | 979 | 980 | 981 | 982 | 983 | 984 | 985 | 986 | 987 | 988 | 989 | 990 | 991 | 992 | 993 | 994 | 995 | 996 | 997 | 998 | 999 | 1000 |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|

## Appendix D: 6 x 8 Character Worksheet

□ □ □ □ □ □ 1 \_\_\_\_\_  
 □ □ □ □ □ □ 2 \_\_\_\_\_  
 □ □ □ □ □ □ 3 \_\_\_\_\_  
 □ □ □ □ □ □ 4 \_\_\_\_\_  
 □ □ □ □ □ □ 5 \_\_\_\_\_  
 □ □ □ □ □ □ 6 \_\_\_\_\_  
 □ □ □ □ □ □ 7 \_\_\_\_\_  
 □ □ □ □ □ □ 8 \_\_\_\_\_

□ □ □ □ □ □ 1 \_\_\_\_\_  
 □ □ □ □ □ □ 2 \_\_\_\_\_  
 □ □ □ □ □ □ 3 \_\_\_\_\_  
 □ □ □ □ □ □ 4 \_\_\_\_\_  
 □ □ □ □ □ □ 5 \_\_\_\_\_  
 □ □ □ □ □ □ 6 \_\_\_\_\_  
 □ □ □ □ □ □ 7 \_\_\_\_\_  
 □ □ □ □ □ □ 8 \_\_\_\_\_

□ □ □ □ □ □ 1 \_\_\_\_\_  
 □ □ □ □ □ □ 2 \_\_\_\_\_  
 □ □ □ □ □ □ 3 \_\_\_\_\_  
 □ □ □ □ □ □ 4 \_\_\_\_\_  
 □ □ □ □ □ □ 5 \_\_\_\_\_  
 □ □ □ □ □ □ 6 \_\_\_\_\_  
 □ □ □ □ □ □ 7 \_\_\_\_\_  
 □ □ □ □ □ □ 8 \_\_\_\_\_

□ □ □ □ □ □ 1 \_\_\_\_\_  
 □ □ □ □ □ □ 2 \_\_\_\_\_  
 □ □ □ □ □ □ 3 \_\_\_\_\_  
 □ □ □ □ □ □ 4 \_\_\_\_\_  
 □ □ □ □ □ □ 5 \_\_\_\_\_  
 □ □ □ □ □ □ 6 \_\_\_\_\_  
 □ □ □ □ □ □ 7 \_\_\_\_\_  
 □ □ □ □ □ □ 8 \_\_\_\_\_

□ □ □ □ □ □ 1 \_\_\_\_\_  
 □ □ □ □ □ □ 2 \_\_\_\_\_  
 □ □ □ □ □ □ 3 \_\_\_\_\_  
 □ □ □ □ □ □ 4 \_\_\_\_\_  
 □ □ □ □ □ □ 5 \_\_\_\_\_  
 □ □ □ □ □ □ 6 \_\_\_\_\_  
 □ □ □ □ □ □ 7 \_\_\_\_\_  
 □ □ □ □ □ □ 8 \_\_\_\_\_

□ □ □ □ □ □ 1 \_\_\_\_\_  
 □ □ □ □ □ □ 2 \_\_\_\_\_  
 □ □ □ □ □ □ 3 \_\_\_\_\_  
 □ □ □ □ □ □ 4 \_\_\_\_\_  
 □ □ □ □ □ □ 5 \_\_\_\_\_  
 □ □ □ □ □ □ 6 \_\_\_\_\_  
 □ □ □ □ □ □ 7 \_\_\_\_\_  
 □ □ □ □ □ □ 8 \_\_\_\_\_



# 8 x 8 Character Worksheet

כ כ כ כ כ כ כ כ 1  
 נ נ נ נ נ נ נ נ 2  
 כ כ כ כ כ כ כ כ 3  
 כ כ כ כ כ כ כ כ 4  
 כ כ כ כ כ כ כ כ 5  
 ח ח ח ח ח ח ח ח 6  
 ל ל ל ל ל ל ל ל 7  
 ח ח ח ח ח ח ח ח 8

ח ח ח ח ח ח ח ח 1  
 ל ל ל ל ל ל ל ל 2  
 כ כ כ כ כ כ כ כ 3  
 כ כ כ כ כ כ כ כ 4  
 כ כ כ כ כ כ כ כ 5  
 ח ח ח ח ח ח ח ח 6  
 נ נ נ נ נ נ נ נ 7  
 כ כ כ כ כ כ כ כ 8

כ כ כ כ כ כ כ כ 1  
 ח ח ח ח ח ח ח ח 2  
 נ נ נ נ נ נ נ נ 3  
 כ כ כ כ כ כ כ כ 4  
 כ כ כ כ כ כ כ כ 5  
 ח ח ח ח ח ח ח ח 6  
 כ כ כ כ כ כ כ כ 7  
 ח ח ח ח ח ח ח ח 8

ח ח ח ח ח ח ח ח 1  
 ח ח ח ח ח ח ח ח 2  
 כ כ כ כ כ כ כ כ 3  
 ח ח ח ח ח ח ח ח 4  
 ח ח ח ח ח ח ח ח 5  
 ל ל ל ל ל ל ל ל 6  
 ח ח ח ח ח ח ח ח 7  
 ל ל ל ל ל ל ל ל 8

ח ח ח ח ח ח ח ח 1  
 ל ל ל ל ל ל ל ל 2  
 כ כ כ כ כ כ כ כ 3  
 ח ח ח ח ח ח ח ח 4  
 ל ל ל ל ל ל ל ל 5  
 כ כ כ כ כ כ כ כ 6  
 ח ח ח ח ח ח ח ח 7  
 כ כ כ כ כ כ כ כ 8

ח ח ח ח ח ח ח ח 1  
 כ כ כ כ כ כ כ כ 2  
 ח ח ח ח ח ח ח ח 3  
 נ נ נ נ נ נ נ נ 4  
 ח ח ח ח ח ח ח ח 5  
 ח ח ח ח ח ח ח ח 6  
 ח ח ח ח ח ח ח ח 7  
 ח ח ח ח ח ח ח ח 8

[illegible]