

# **GRAPHIC DISPLAY VERSION 3.1**

**a.k.a.**

# **GDISP31**

**A Comprehensive Functional Specification  
of an intelligent controller for a Hyundai  
HG25504NG-01 256x128 LCD Graphics Display  
Module**

**Duane Becker  
September 27, 2005**

# Table Of Contents

DESCRIPTION .....	3
EMBEDDED FEATURES .....	3
HARDWARE INTERFACE DESCRIPTION .....	3
<i>Hardware Pinout</i> .....	4
<i>Timing Diagram of a Typical Byte Transfer</i> .....	4
<i>Interface Timings</i> .....	5
<i>Electrical Specifications</i> .....	5
MEMORY MAP OF THE HYUNDAI DISPLAY CARD.....	5
COMMAND SUMMARY .....	8
COMMAND DESCRIPTIONS .....	9
<i>Z - Mode Selection</i> .....	9
<i>D - Display Selection</i> .....	10
<i>E - Edit Pages Selection</i> .....	11
<i>S - Switch The Active Pages</i> .....	11
<i>C - Clear and Home The Text Page</i> .....	12
<i>W - Wipe Clean The Graphics Page</i> .....	12
<i>L - Line</i> .....	12
<i>F - Fat Line</i> .....	14
<i>N - Dashed Line</i> .....	14
<i>U - Unbroken Chain Of Lines</i> .....	15
<i>P - Position Text Pointer</i> .....	16
<i>T - Text</i> .....	16
<i>B - Box</i> .....	18
<i>I - Initialize</i> .....	19
<i>H - Home</i> .....	20
<i>O - SetDot</i> .....	20
<i>A - Alarm</i> .....	20
<i>K - Cursor</i> .....	21
<i>R - Restore The Cursor Area</i> .....	21
<i>G - Graphics Symbol Instantiate</i> .....	22
<i>Q - Double-Size Symbol Instantiate</i> .....	24
<i>M - Make Graphics Symbol</i> .....	25
<i>Y - Yank Graphics Into A User Symbol</i> .....	26
<i>X - XOR Symbols</i> .....	27
<i>z - Graphic Virtual Text Characters Block Size Command</i> .....	27
<i>t - Graphic Virtual Text Characters String Plotting</i> .....	29
<i>p - Graphic Virtual Text Characters Cursor Position</i> .....	29
<i>V - Video Block Transfer</i> .....	30
<i>e - Ellipse</i> .....	33
COMMAND SUMMARY CARD.....	34
GRAPHICS SYMBOL SET FOR GDISP3 CONTROL CARD .....	35
GRAPHICS SYMBOL LISTING .....	36
CONTROLLER CARD COMPONENT LAYOUT .....	39
GDISP3 SCHEMATIC.....	39
VOLTAGE INVERTER COMPONENT LAYOUT .....	40
VOLTAGE INVERTER SCHEMATIC .....	40

## **Description**

This manual describes the functional operation of the GDISP3 front end control card for a Hyundai HG25504NG-01 display panel (available from AllElectronics). The front end card and firmware of the control card simplifies the interface and use of the display panel. The panel natively accepts byte reads, byte writes, and configuration commands, but supports no high-level commands. This front-end card handles all the work needed to configure and control the panel and provide high-level drawing and text tools to speed development of a project. A parallel interface port with -STROBE, -ACK, and +BUSY signals allows the card to be controlled with a printer port or any parallel interface.

The display module itself is black dots on white background, and requires about -15 Volts on its LCD backplane voltage pin. The response time of the display is pretty slow, about 200 milliseconds to any change, so rapidly changing display data will not show up well. This implies that dynamically moving cursors might be difficult to track. You can use the user-symbols to implement a cursor trail if desired.

## **Embedded Features**

- ⇒ 256 dots wide, 128 dots tall
- ⇒ Contrast Control
- ⇒ Single 5-Volt operation
- ⇒ Standard Parallel Port Interface
- ⇒ Two operational modes: 2 text pages + 1 graphics page (2T+G), or 2 graphics pages (2G)
- ⇒ Can select which text page or graphic page is displayed
- ⇒ Can enable or disable text and graphics display independently
- ⇒ Independent control of which pages are selected for editing
- ⇒ Many built in text, copy, and drawing tools
- ⇒ Built in bitmap transfer command
- ⇒ Sixteen user-definable graphic symbols, 240 predefined graphic symbols
- ⇒ 5×5 pixel, fast, sprite based graphic cursor
- ⇒ Built in audible alarm (1/4 second beep duration)
- ⇒ GDISP3.EXE allows testing the smart controller's functions from a PC.
- ⇒ CONVBMP.EXE allows converting a black and white bitmap to a video transfer command data sequence.

## **Hardware Interface Description**

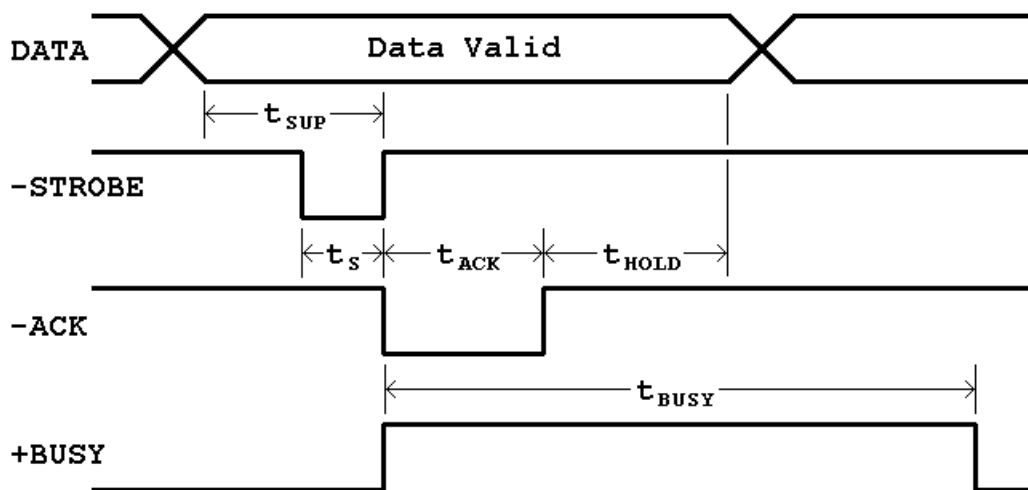
The parallel-port connection to the controller has eight data lines, a -STROBE line, +BUSY line, and a -ACK (acknowledge) line. The data lines are unidirectional and send data to the display. When pulsed low, the -STROBE line commands the display interface that new data is available on the data lines. The -ACK line drops and the +BUSY line rises as soon as a low to high transition on -STROBE occurs. When the control card has read the bus data, it resets the -ACK line to a high, indicating to the host that the data has been accepted and the data lines may be released. The +BUSY line indicates to the host that the controller is busy processing the last command or data byte. The host must first ensure that the +BUSY line is low before strobing the next command or data byte transmission. The +BUSY line

stays at a high level until the data byte or command is completely processed.

### Hardware Pinout

Interface Pin Number of J1	Pin Name	Pin Type
1	+5 Volt Power Supply	Power
2	+BUSY	Output
3	-ACK	Output
4	-STROBE	Input
5	Data Bit 0	Input
6	Data Bit 1	Input
7	Data Bit 2	Input
8	Data Bit 3	Input
9	Data Bit 4	Input
10	Data Bit 5	Input
11	Data Bit 6	Input
12	Data Bit 7	Input
13	Ground	Power Return

### Timing Diagram of a Typical Byte Transfer



## Interface Timings

Parameter	Minimum	Maximum	Units	Description
$t_{SUP}$	0	---	ns	Data setup before rising edge of -STROBE
$t_S$	30	---	ns	-Strobe width
$t_{ACK}$	1.5	3.0	$\mu s$	Time for controller to acknowledge a strobe
$t_{HOLD}$	0	---	ns	Data hold after -ACK rising
$t_{BUSY}$	---	0.8	s	Time to execute command (command dependent)

The normal process of sending a byte of data is:

- 1) Ensure that the +BUSY line is at a low state (not Busy)
- 2) Apply a data byte to bus
- 3) Drop the -STROBE line
- 4) Raise the -STROBE line
- 5) Wait until the -ACK line has returned to a high state.
- 6) Tristate the data bus (optional)

The usual time that you have to wait for the -ACK line to return high after issuing a -STROBE rising edge is two microseconds.

You may also use just the +BUSY line as a command and data throttling signal without using the -ACK line. That process is presented here:

- 1) Ensure that the +BUSY line is low (not busy)
- 2) Apply the data byte to bus
- 3) Drop then raise the -STROBE line
- 4) Wait at least three microseconds for the control card to grab the data
- 5) Tristate the data bus (optional)

The standard PC printer port supports the +BUSY/-ACK protocol and the +BUSY only protocol.

## Electrical Specifications

Parameter	Minimum	Maximum	Units	Description
$V_{CC}$	4.75	5.25	Volts	Power supply voltage
$I_{CC}$	70	85	milliAmps	Power supply current (See Note 1)
$V_{IH}$	2.6	$V_{CC}+0.3$	Volts	Input high level
$V_{IL}$	-0.2	0.8	Volts	Input low level
$F_{ALARM}$		2.3	KHz	Alarm tone frequency (See Note 2)
$t_{EDGE}$	0	1	$\mu s$	Signal rise and fall times

Note 1: When the alarm is sounding, the maximum specification applies. When the alarm is silent, the minimum specification applies.

Note 2: There is no minimum or maximum for alarm frequency. The value indicated is the typical value.

## Memory Map Of The Hyundai Display Card

The display module has 8192 (8K) bytes of memory on-board. The smart controller card allocates this memory in two ways, depending on the operational mode that is selected. This diagram describes how the 8K of memory on the display module is allocated by the GDISP3 control card's

firmware. Normally, the user does not need to know about the internal memory allocation of the display, since the control card provides the high-level command interface to the user.

Addresses (Hexadecimal)	Mode 0 (2T+G)	Mode 1 (2G)
0000	Text 0	Graphics 1
....		
01FF		
0200	Text 1	
....		
03FF		
0400	CGRAM (not used)	
....		
0FFF		
1000	Graphics 0	Graphics 0
....		
1FFF		

You may use the Mode command to select Mode 0 or Mode 1, and the Display and Edit commands to set other parameters allowing you to edit and display different pages. Mode 0 supports two text pages, and one graphics page. Mode 1 supports two graphics pages and no text page. You can, however, instantiate symbols and text onto a graphics screen. The Graphics 0 page is available in both modes. The smart card handles all the byte and bit addressing based upon simple screen position parameters in the commands.

Each of the two paired pages in each mode can be edited or displayed independently. That is, you can edit one while the display shows the other, or you can edit the one you're actually looking at. You can even edit a page and display neither page.

For example, in mode 0, you have graphics page 0, and text pages 0 and 1. Graphics page 0, the only graphics page in mode 0, is always available for editing, but may be selectively displayed or hidden. The two text pages are independently edit-selectable and displayable. You may select either or neither text page to display, and you may select which text page is active for editing (whether displayed or not).

The text pages are 32-characters across and 16-characters tall. Each character sits inside an 8x8 pixel box. The graphics pages are 256-pixels wide by 128-pixels tall. The text characters are logically ORed with the graphics drawings so that graphics shows through the whitespaces of the text characters. Erasing text characters does not alter the graphics data since they held in separate areas of the display memory.

In mode 0, you are limited to editing only graphics page 0, but may enable or disable its display. You have free selection about which text page to

edit, and which text page (or neither) to display. This allows you to formulate a new text page while displaying the other, if you so desire.

In mode 1, you have two independent graphic windows, and full control of which (or neither) is displayed, and which one is selected for editing. This independent control allows you to compose a new graphic screen while the other one is being displayed, if so desired.

## Command Summary

In this document, "page" refers to the text or graphics page of data. This is not necessarily the same page that is being displayed since you can edit a graphic or text page of data that may or may not be currently displayed. This allows building new pages of data while the old page is still being displayed, which can be used to provide snap updates to the display while hiding the display building process.

Command Character	Command Description
Z	Mode Selection (to select 2T+G or 2G operating modes)
D	Display selection (which pages are displayed)
E	Edit selection (which graphic and text pages are editable)
S	Switch the pages being displayed and those being edited
C	Clear the text page selected for edit and home the character pointer
W	Wipe clean the graphics page selected for editing
L	Draw a line from one coordinate to another
F	Draw a Fat line from one coordinate to another
N	Draw a dashed line with adjustable dash lengths
U	Unbroken chain of lines on screen
P	Position the text pointer to a row and column
T	Send text characters to the text page selected for editing
B	Draw an opposite-corner specified rectangular box with various attributes
I	Initialize the entire display to reset conditions
H	Send the text pointer to the home position (upper left)
O	Dot command sets or clears any specified dot
A	Alarm sounds for 1/4 second
K	Place a graphics cursor on the currently edit-selected graphics page while saving the graphics dots for later restore
R	Restore the graphics cursor pixels turned on by the Kursor command
G	Graphic symbol instantiating to the graphics page
Q	Double-sized symbol instantiation to the graphics page
M	Make symbol - allows defining your own graphic symbol
Y	Yank graphic symbol allows yanking an 8x8 symbol off graphics screen to a user symbol
X	XOR two symbols together into a user symbol
z	set the horizontal and vertical graphic text spacing
t	Plot text and symbol strings to the graphics layer
p	Position the virtual graphic text pointer to a row and column
V	Video block transfer
e	Ellipse command
Note: The following letters are not used by the controller for commands: J, abcd fghijklmnopqrstuvwxyz	

## Command Descriptions

Some of the controller commands are single ASCII characters and some include additional data bytes that supply values and data needed by the command. Each space-separated element in the command format represents one byte, either an ASCII character or an 8-bit byte. The times for execution are based on using a 24 MHz crystal for the processor and include the time spent sending the commands to the controller over the parallel bus.

### Z - Mode Selection

Command Format:            Z mode

There are two modes of operation. The command is first activated by sending the ASCII character "Z" (5Ah). Then send either a zero byte (00h) for mode 0 or any nonzero byte for mode 1.

Mode 0 supports two selectable text screens (T0 and T1) and one graphics screen (G0). The selected text screen is transparently ORed with the graphics screen dots on the display, but are in physically separate portions of memory. In mode 0, you can independently select which of the two text screens to edit, and which of the two text screens to display. The graphics screen (G0) is automatically selected for editing and display. You may turn the selected text screen and/or the graphics 0 screen on or off at will, independently of each other. Mode 0 is useful if you don't change graphics screens too much, and need to independently incorporate text into the display without having to redraw the graphics screen.

Mode 1 supports two graphics screens (G0 and G1) and no text screens. You may select which graphics screen is displayed (or none of them), and you may select which graphics screen is selected for editing. You may edit either the displayed or the nondisplayed screen. Mode 1 is useful if you are frequently redrawing the graphics data, and want to be able to draw the new screens while hidden, and present the finished screen to the user only when its composition is complete, thus eliminating the on-screen drawing distraction.

When you select mode 0, the following things occur:

1. Both text screen layers are cleared to all spaces
2. Text page 0 is selected for editing and for display, and the text screen is turned on for display (although it is set to all blank spaces).
3. The text pointer is reset to row 0 and column 0 (upper right).
4. Graphics page 0 is selected for editing and display, and is enabled for display, but it is not cleared.

When you select mode 1, the following things occur:

1. The text screens are disabled
2. Graphics page 1 is cleared to all whitespace.
3. Graphics page 0 is selected for editing and display, and is enabled for display, but is not cleared.

The Z mode selection command executes in 56 milliseconds.

## D - Display Selection

Command Format:            D n

This command allows selection of which graphics page and which text page is selected for LCD screen display, and whether the selected graphics and/or text pages are displayed. Either, both, or none of the selected pages may be displayed. The first byte is the ASCII character capital "D" (decimal value 68, hex value 44h). The second byte selects the pages and their on/off status.

The display selection parameter byte (n) is composed of four bit-fields:

7	6	5	4	3	2	1	0
0	0	0	0	Graphics Turned On	Graphics Page Selector	Text Turned On	Text Page Selector

Graphics Turned On - when 1, the graphics page selected by bit 2 is shown on the display. When 0, the graphics page display is turned off

Graphics Page Selector - This selects which graphics page (0 or 1) is allowed to be displayed.

Text Turned On - when 1, the text page selected by bit 0 is shown on the display.

Text Page Selector - This selects which text page (0 or 1) is allowed to be displayed.

Note that in mode 1 (2G), the text control fields are ignored since there is no text window. In mode 0 (2T+G), the Graphics Page Selector bit is ignored and is forced to 0 since there is only graphics page 0 in mode 0. Examples:

Command	Mode 0 (2T+G) Behavior	Mode 1 (2G) Behavior
"D" 00001010b	Display graphics page 0 and text page 0	Display graphics page 0
"D" 00001110b	Display graphics page 0 and text page 0 (the second "1" is ignored in mode 0)	Display graphics page 1 (there are no text pages in mode 1)
"D" 00001011b	Display graphics page 0 and text page 1	Display graphics page 0 (there are no text pages in mode 1)
"D" 00000101b	Displays neither graphics nor text pages	Displays neither graphics page
"D" 00001111b	Display text page 1 and graphics page 0 (no graphics page 1 in mode 0)	Display graphics page 1 (no text pages in mode 1)
"D" 00001100b	Displays graphics page 0 (mode 0 only has one graphics page)	Display graphics page 1
"D" 00001000b	Display just graphics page 0	Display just graphics page 0

You can use this command to FLASH the display, graphics and/or text, on and off by first displaying the data, then turning the displayed page off, then repeating at your own desired rate. The display select command executes in 275 microseconds.

## E - Edit Pages Selection

Command Format:           E n

This command allows you to select which graphics page and which text page are allowed to be edited. This is a two byte command. The first byte is the ASCII character "E". The second byte selects which pages are editable:

7	6	5	4	3	2	1	0
0	0	0	0	0	0	Graphics Edit Selector	Text Edit Selector

A "0" in a selector bit directs editing into page 0 of the graphic or text page. A "1" in a selector bit directs editing into page 1 of the graphic or text page. Editing consists of any operation that changes the page composition (like adding characters, clearing, inverting, etc.). Examples:

Command	Command Description
"E" 00000000b	Enable edit of graphics page 0 and text page 0
"E" 00000001b	Enable edit of graphics page 0 and text page 1
"E" 00000010b	Enable edit of graphics page 1 and text page 0
"E" 00000011b	Enable edit of graphics page 1 and text page 1

When an edit selection is made that is not supported by a particular mode, (such as selecting a text page to edit when in graphics only mode), the selection bit is ignored. The Edit select command executes in 100 microseconds.

## S - Switch The Active Pages

Command Format:           S

This command simplifies the editing and display control of two screens of data. If you are dynamically changing the screen, you can display one set of pages, while you edit the other set of pages. When you are done creating the second set of pages, you can SWITCH, and the selected pages for display will be flipped and the selected pages for editing will also be flipped. This is the easiest way to swap between creating and displaying two pages, and allows display of a stable picture while you edit and create a second picture. Of course, you can edit and display the same pages if you desire to watch the creation of the data dynamically. This command takes no arguments. Just send the character "S" (capital S in ASCII). In mode 0 (2T+G), the switch command only toggles which text screen is displayed and edited. In mode 1 (2G), the switch command only toggles which graphics screen is displayed and which one is selected for editing. NOTE: Switching from one text page to the other does not alter the character pointer (row and column) for next text. You may desire to use one of the cursor altering commands to reset the text pointer before sending characters to the switched into text page. The "H" home, "P" position, and "C" clear text commands all are likely candidates.

Example 1 (commands are in sequence):

```
"Z" 00000000b Select mode 0 (display and edit G0 and T0)
"E" 00000001b edit G0 and T1
```

... text and drawing commands go into pages G0 and T1 while G0 and T0 are being displayed ...

```
"S"          Switch so that page T1 is displayed and T0 is selected
              for editing.
```

Example 2:

```
"Z" 00000001b Select mode 1 (display and edit G0)
"E" 00000010b Edit G1
```

... graphic drawing commands go into page G1 while G0 is being displayed...

```
"S"          Switch so that page G1 is displayed and G0 is selected
              for editing.
```

The switch command executes in 172 microseconds.

## C - Clear and Home The Text Page

Command Format: C

Send the single byte command ASCII "C" (capital C), and the text page currently selected for editing (not necessarily the displayed one) is cleared to all spaces. This command also resets the text row and column pointers to 0,0 (the top row, leftmost column). If the edit selected text page is ALSO selected for display, you will see the clear take place immediately. You may clear the NON displayed text page if it's set up for editing. The clear text command executes in 8.1 milliseconds.

## W - Wipe Clean The Graphics Page

Command Format: W

Send the single byte command ASCII "W" (capital W), and the graphics page currently selected for editing (not necessarily the displayed one) is cleared to whitespace. This command does not clear out any actively stored "K" cursor saved command. The non-edit-selected graphics page is not cleared. If the currently selected-for-edit graphics page is also selected to be displayed, you will see the screen wipe take place immediately. You may clear the non-displayed graphics page if it's set up for editing. This does not clear any text placed using the "T" text command. Since mode 0 has only one graphics page, the WIPE command will always target graphics page 0 in mode 0. The wipe command executes in 54 milliseconds.

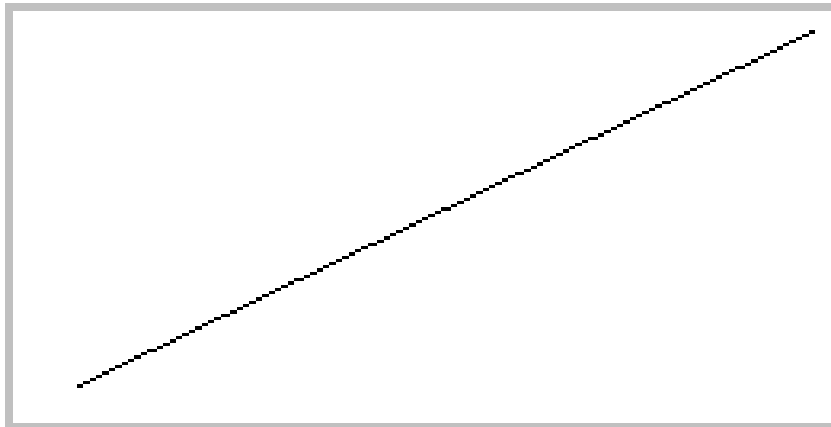
## L - Line

Command Format: L x1 y1 x2 y2

This draws a single-pixel wide line from the first X,Y coordinate pair to the second X,Y coordinate pair. The LCD is 256-pixels wide and 128-pixels tall. The range of values allowed for Y are from 0 to 127 inclusive. You **should not** send a Y value outside of this range for **any** graphics command using dot coordinates. Since the graphic area is 256-pixels wide, and one byte exactly covers the range of 0 to 255, the value of the X parameters can be any byte value. To draw a standard thin line in your currently edit selected page, send the character "L" (capital L), then the two bytes representing the X,Y location of one end, and two more bytes representing the other X,Y endpoint. The screen coordinate system is set up to be mathematically correct where the lower left dot is located at X=0,Y=0 and the upper right dot is located at X=255,Y=127. No error checking is performed. If you specify a Y value larger than 127, other portions of the display memory might be affected. Both endpoints should be in the graphics page field. For example:

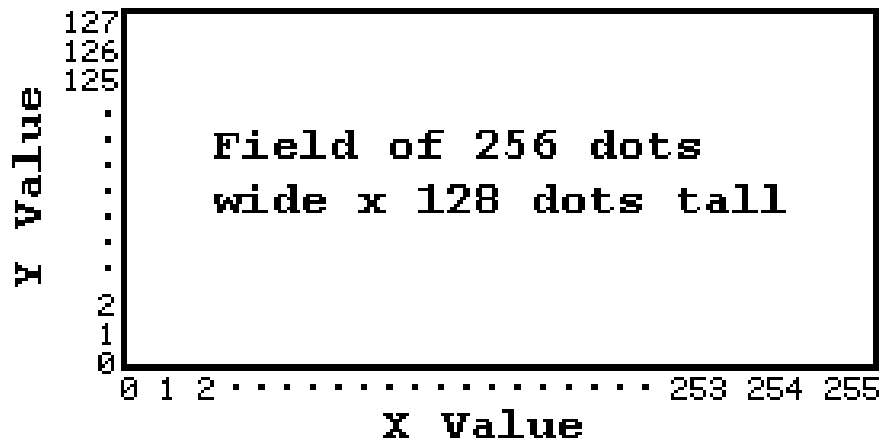
```
"L" 00010100b 00001011b 11111010b 01101111b
      14h      0Bh      FAh      6Fh
      20 dec   11 dec   250 dec  111 dec
```

That 5 byte command will draw a line from 20,11 to 250,111 .



To estimate a line draw time, use the benchmark of 157 microseconds for each dot in the long dimension. Thus the line 0,0 to 255,127 takes 256 dots x 157 microseconds = 40.3 milliseconds to complete.

**Graphic Dots Page Layout**

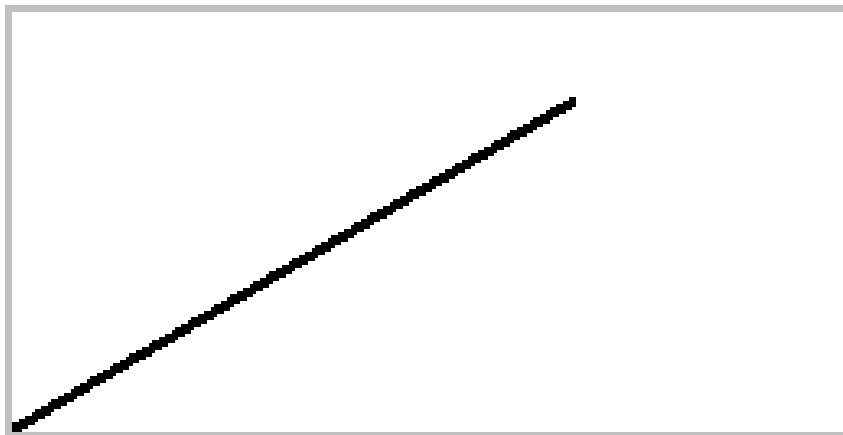


## F - Fat Line

Command Format:        F   x1   y1   x2   y2

This performs the same basic function as the "L" line command, except that the line drawn is 3 pixels fat and the drawing time is almost three times as long as the single line.

No error checking is performed on the Y endpoints. You should not send a Y value greater than 127. The graphics screen is 256 dots wide in the X dimension, so any value is permitted for X. The Y axis is limited to coordinate values from 0 to 127. Essentially, this command draws three lines side by side, offset in the Y axis if the major length is along X, and offset in the X axis if the major length is along Y. If any of the outer lines has dots that fall off the screen, those dots will not be drawn. If you draw a fat line along an edge of the screen, it will only be two dots wide since the outermost line is offscreen and is not drawn. Since this routine draws lots more dots, it takes longer to draw a fat line, so use fat lines judiciously. For example, the command "F" 0 0 172 100 results in the line shown here:



This command executes in 109 milliseconds for a 256-dot long line or about 426 microseconds per dot for the longest axis length.

## N - Dashed Line

Command Format:        N   dashlong   x1   y1   x2   y2

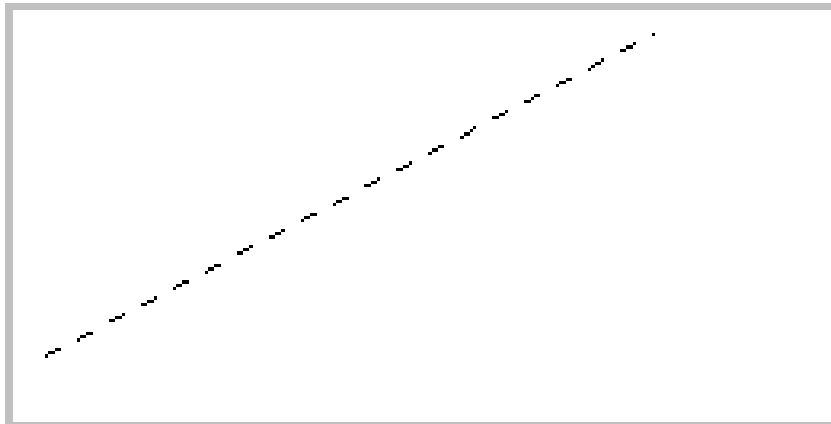
This performs the same basic function as the "L" line command, except that the line is drawn in dashes and is drawn faster than a regular line since only half as many dots are turned on. The portion of the line that is not drawn (between the dashes) is not cleared off the graphic space. In other words, the line gaps are drawn transparently, so other data can show through. The value 'dashlong' represents how many dots are used for dashes and non-dashes and is valid from 1 to 15. In the following situations, a simple solid line is drawn instead of a dashed line:

- a) If the value 'dashlong' = 0 or 'dashlong' > 15
- b) If the longest dimension (x or y) of the line is less than three dots long.
- c) If the longest dimension of the line is less than or equal to the length of the dash specified.

No error checking is performed on the Y endpoints and they should be from 0 to 127. Since the panel is 256 dots wide, any byte value is allowed for the X coordinates.

The dashes are adjusted in length for diagonal lines so that the dash length is kept approximately the same regardless of the angle of the line. A dashed line always has a full dash at the starting end (x1,y1), and a partial dash or a single dot at the trailing end (x2,y2). For example:

"N" 5 10 20 200 120 will draw a dashed line from 10,20 to 200,120 using dashes that are 5 dots long, and looks like this:



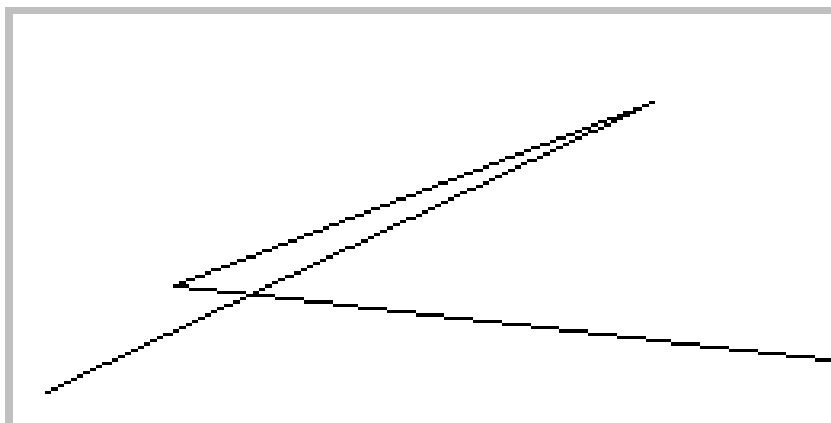
This command executes in 26 milliseconds for a line of length 256 dots in the long axis.

### U - Unbroken Chain Of Lines

Command Format: U x1 y1 x2 y2 x3 y3 ... xn yinvalid

This command draws a chain of line segments, each successive point joined to the previous segment. Send the "U" character to start the chain, then send the first x,y pair. Each successive x,y pair sent thereafter draws that next line segment, and prepares for the next in the chain. To end the command, send an X,Y pair with an invalid Y value (Y>127).

The command "U" 10 10 200 100 50 43 255 20 0 128 draws this:  
(X1 Y1 X2 Y2 X3 Y3 X4 Y4 Xn Yin)



This command draws the lines at the same rate as normal lines, namely about 157 microseconds per dot of the longest dimension.

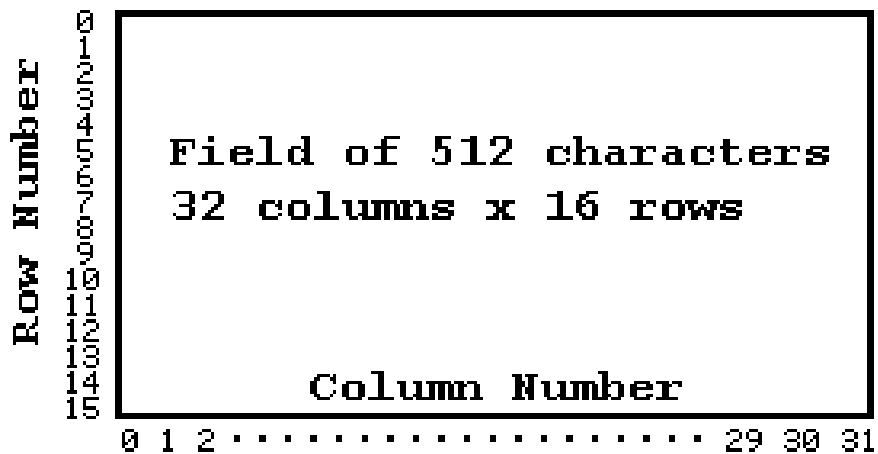
### P - Position Text Pointer

Command Format: P row col

This command positions the text address pointer to row (0 to 15) and column (0 to 31) so the next text command will place text characters at that location on the text page selected for editing. The rows range from 0 at the top to 15 on the bottom, and the columns range from 0 on the left to 31 on the right. There is no blinking cursor on the text page. Use the unsigned byte value for row and col. No checking is performed to the values for row and col. Invalid values will produce unexpected results. An example of using the position command follows:

```
"P" 00001010b 00000011b This three-byte command places the text
      0Ah      03h      pointer on row 10 (11th row from top) in
the      10d      3d      third column (fourth character from left).
```

Text Page Layout



The position command completes in 108 microseconds.

### T - Text

Command Format: T n Char1 Char2 ... Charn

The text command accepts and displays 'n' characters following the count byte (n) to the currently positioned row and column position in the selected-for-edit text page. Send regular ASCII characters. If you are in mode 1, with two graphic planes and no text plane, the command just purges the expected number of characters after the 'n' count byte, and does not alter the display or change the row or column pointers. If you are in mode 0, the mode that supports text pages, the row and column text pointers auto-increment and wrap from the end of a line to the beginning of the next line, and from the last character on the last line back to the upper left corner. There is no support for a "carriage return" character. It is the application's responsibility to use the "P" position pointer command. The display module's internal character generator is used to produce characters.

The display map of text character code to appearance is shown below. Note that this symbol map is different than the graphics symbol map shown later in this document.

		Character Code Bits 3 to 0 (lower nibble)															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Character Code Bits 7 to 4 (upper nibble)	0																
	1																
	2		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
	3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
	4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
	6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
	8																
	9																
	A		␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
	B	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
	C	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
	D	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
	E																
	F																

You may send a count value (n) of from 0 to 255. The count value of 0 is treated as 256 characters. This can be confusing if you are manually sending characters since the unit will stop responding to commands until all 256 characters are received. If a program is handling the display, you may freely use the '0' count to mean 256 characters. The number of character codes sent after the count byte MUST equal the count byte to terminate the command. For example, to send the word BARBAROUS you must use the command:

```
"T" 00001001b "B" "A" "R" "B" "A" "R" "O" "U" "S"
      09h
      9 decimal
```

No end of string character is sent, as the command parser automatically quits accepting text characters after 'n' text bytes are received. If you send extra characters, they will be processed as if they were commands. Thus, do not send more text characters than the count parameter indicates or undesired commands may be interpreted.

The text command executes in 4.32 milliseconds for a 32 character string. A general formula for completion time is:

125 + (count x 131) microseconds.

## B - Box

Command Format:            B mode x1 y1 x2 y2

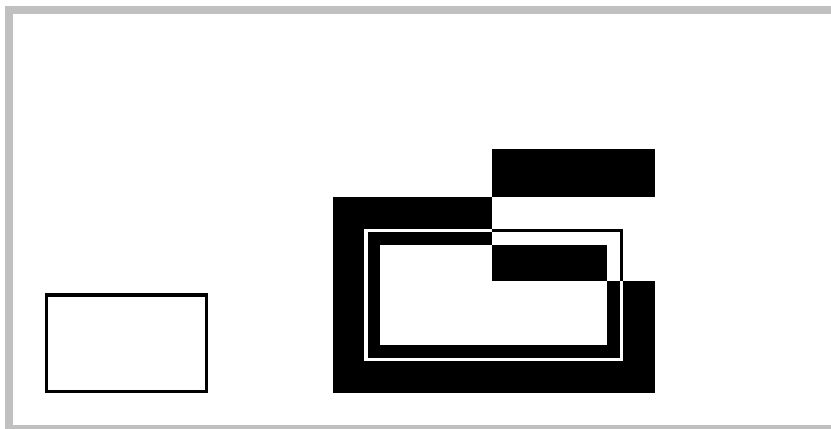
This six-byte command allows you to draw a rectangle or rectangular area into the currently selected-for-edit graphics page. Specify any two opposite corners x1,y1 and x2,y2. The corners do not have to be sorted in any way. The smart card handles the sorting of the corners. The 'mode' selector byte specifies the kind of box to be drawn.

Mode	Type of Box	Description
00h	Black lines	The outline of box drawn in black.
01h	White lines	The outline of box is drawn in white.
02h	Filled area	The entire box area is set to black.
03h	Cleared area	The entire box area is set to white.
04h	Invert area	The entire box area is inverted (black-to-white and white-to-black).

As an example, starting with a blank graphics screen, the following boxes are added in order (decimal values are shown for the byte-sized parameters).

Command	Description
B 0 10 10 60 40	Black outline box from 10,10 to 60,40
B 2 100 10 200 70	Filled black box from 100,10 to 200,70
B 1 110 20 190 60	White-lined box 10 pixels inside the black box
B 3 115 25 185 55	Clear the black area inside the white-lined box, leaving a 4 pixel boundary (4 pixels still blackened).
B 4 150 45 200 85	Invert a portion of the figure plus some of the background screen

The resulting display of the above sequence of commands is shown below.



The lines and fills used for a box opaquely overwrite any graphics data already on the screen. The invert-fill operation simply inverts the black and white already on the screen. None of the box operations affect any text characters that might be displayed on a text page. Symbols that are instantiated on the graphics display can be overwritten, since they lie in the graphics memory area when instantiated. The box-invert function is useful for "flashing" any rectangular section of the graphics screen. Simply invert the box, wait a short time, and repeat the inversion-wait cycle to flash the area in normal/inverse modes.

Completion times for the box command vary according to the type of box attributes and the edge boundaries along the X-axis. Here are some examples of completion times.

Type of Box	X x Y Box size	Completion Time (milliseconds)
0 = Black Outline	256 x 128	120
1 = White Outline	256 x 128	120
2 = Black Filled	256 x 128	118
3 = White Cleared	256 x 128	118
4 = Inverted	256 x 128	590
4 = Inverted	16 x 16	8.6
4 = Inverted	8 x 8	3.1

## I - Initialize

Command Format: I

The Initialize command resets the LCD hardware display parameters, reinitializes the text and graphics selection parameters, homes the text character pointer, and clears all graphic and text pages, and puts the display in mode 0. It is roughly equivalent to a power on reset. The initialize command also sends a power-on-reset to the display itself. This allows software to correct any kind of fatal startup error in the display module itself or to quickly return the display to the power-up condition. The power-up defaults and the initialize command defaults are identical and are:

- ⇒ Operational Mode 0 selected (2T+G)
- ⇒ Graphics page 0 enabled for display and editing
- ⇒ Text page 0 enabled for display and editing

- ⇒ Text row and column pointers are reset to 0 (upper left)
- ⇒ Both text pages and the graphic page is cleared out
- ⇒ The stored sprite Cursor is invalidated.
- ⇒ All user-defined graphic symbols are cleared out
- ⇒ The graphic virtual text block is set to 6 wide x 8 tall
- ⇒ The graphic virtual text cursor is set to row 0, column 0

The initialize command is a good command to use if a control program changes operating modes and screens altogether. The initialize command does not override an errant text command or long video block transfer. If you fail to send the specified number of bytes following a text or video block command, the text command will continue to receive up to 256 characters before accepting any more commands, and the video block might require up to 4096 bytes of video. If you want to force an initialize command, regardless of what previous commands are being executed, the safest way is to send 4096 bytes of value 128 (80h), then send the initialization command. This will purge any possible text or video command of all characters expected, or terminate any incomplete line-chain command, or terminate any other command requiring parameters.

Execution of this command takes 197 milliseconds.

## **H - Home**

Command Format:           H

This command resets the text pointer to the top row and the leftmost column. This command is equivalent to    P 00h 00h . This command does not clear any text from the screen. The home command completes in 50 microseconds.

## **O - SetDot**

Command Format:           O dotvalue x y

The setdot command is started by first sending the character capital O (0h). This command sets the dot located at graphics location x,y on or off based on the value of dotvalue. If dotvalue is zero, then the dot will be turned off (whitespace). If the dotvalue is nonzero, then the dot will be turned on (black dots). The valid range for Y is from 0 to 127. No error checking is performed on the Y value. If you try to turn on a dot at a Y ordinate higher than 127, something else on the screen might be affected. As for all graphics commands, the valid range for X is 0 to 255, fully representing the one-byte X parameter. The setdot command completes in 346 microseconds.

## **A - Alarm**

Command Format:           A

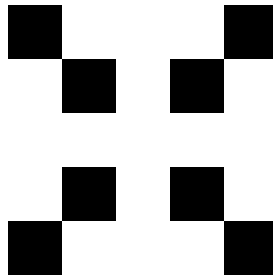
The alarm tone and duration are fixed by hardware, and do not use controller cycles, making for a quick-completing command. The controller is ready for new commands even while the alarm is still sounding. The duration of the alarm is fixed at 1/4 second, and cannot be extended by repeating the alarm command until the alarm has silenced.

Completion of the alarm command takes 45 microseconds. The alarm will continue to sound after this time for the full 1/4 second.

## **K - Kursor**

Command Format:           K x y

Yes, we know cursor is spelled with a C, but C was already taken for Clear. This command puts a graphics cursor at x,y on the selected-for-editing graphics page and saves the eight original dots where the cursor sits for use by the restore command. The cursor is a small 5-pixel by 5-pixel "X" with the middle point (x,y location) missing.



The old value of the eight dots surrounding the x,y location and the actual location of the cursor are saved for use by the restore command to return those dots to their original state when the cursor area is restored. Detailed usage of the cursor is described in the restore command description below. The dots of the cursor are placed on the screen using logical "OR"ing of the dots. Thus, the cursor will not be seen if it is placed onto heavily drawn areas of the screen. If the cursor is instantiated off screen, no dots will be drawn and the restore Kursor will be invalidated. If the cursor is instantiated near the edge of the screen, only those dots that fall inside the screen boundaries are drawn and saved for the restore command. The Kursor command completes in 1.7 milliseconds.

## **R - Restore The Kursor Area**

Command Format:           R

Since the K cursor command saves the eight cursor dots original settings, and the location where the cursor is placed, no arguments are required for the restore command. If a valid K-cursor is stored, then the "R" restore command removes the cursor from the screen, and restores the eight original dots to their appearance before the "K" cursor was added.

The normal use of the K and R commands is to provide a quickly moved and nondestructive cursor pointer on a graphics screen. Typically the following sequence of programming is used to implement a pointer cursor:

```

                                Get pointer location x,y from user (mouse, joystick, etc.).
                                Save x,y in oldx,oldy.
CYCLE:    Use K cursor command to place cursor at oldx,oldy.
                                Wait until pointer location changes or an exit from the
                                pointer cursor routine is invoked.
                                Use R restore command to remove cursor from display.
                                If exiting from pointer cursor routine, goto END:
                                Put the new pointer position in oldx,oldy.
                                Goto CYCLE:
END:      Exit routine.

```

This algorithm will only erase and update the cursor on the screen when its position actually changes. This creates a smoother display with fewer draw cycles to the display.

Another way of generating a restorable cursor is by using the YANK and XOR commands. This technique is described in those sections.

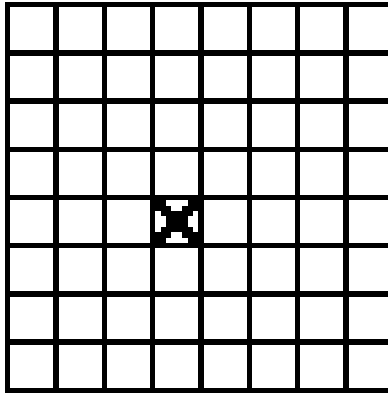
The restore cursor command completes in 1.7 milliseconds.

## G - Graphics Symbol Instantiate

Command Format:            G sel opaque inverse x y

This command places the graphic symbol selected by the byte "sel" to the location x,y on the current edit graphic screen with opaqueness and inversion options. There are 240 predefined symbols and 16 user-defined symbols. The predefined symbols are accessed using a 'sel' value of from 16 to 255 decimal (10h to FFh), and the user symbols are accessed using a 'sel' value of from 0 to 15 decimal (00h to 0Fh). Each predefined symbol and each user-defined symbol is eight pixels wide and eight pixels tall. The 'opaque' flag in the command determines if whitespace in the symbol overwrites already drawn black dots on the screen. If the 'opaque' flag is set to any nonzero value (01h through FFh), then the whitespace overlays existing graphic screen data. If the opaque flag is set to zero (00h), then the whitespace is NOT drawn onto the screen, and existing dots on the display remain intact. Since text characters on a text page are in a separate memory space, the opaque setting will not blank out text characters. The 'inverse' flag determines if the symbol data is inverted when sent to the screen. If you set it to zero, the symbol is placed on the screen in normal mode (where set bits in the symbol are black dots). If you set 'inverse' to any nonzero byte value, the symbol is placed in inverse video on the screen. The inverse option is applied to the symbol BEFORE the opaqueness option. This means that if you were to apply inverse and transparency to symbol 236 (ECh), an all black block, the inverse would first turn all the black dots to white dots, then the transparency would prevent the white dots from overlaying the underlying graphics.

If you try to instantiate the symbol off screen, NOTHING is displayed and the command will immediately complete. If you instantiate the symbol such that a portion of the symbol falls off the display, only the dots that fall inside the screen are drawn. The x,y location you specify designates the "center" of the symbol area. In the symbol map below, the "X" specifies the center reference point for the symbol.



The predefined symbols are shown in a table later in this document. These symbols, and the characters that are contained within the symbol map, are completely different in nature than the internal character generator that paints the text pages of the display. The classic ASCII character set is duplicated in the symbol map so that you can instantiate characters onto the graphics plane as well as the other graphic symbols, and any of the 16 user-defined symbols. As an example, starting with a blank screen, the following symbol commands are performed.

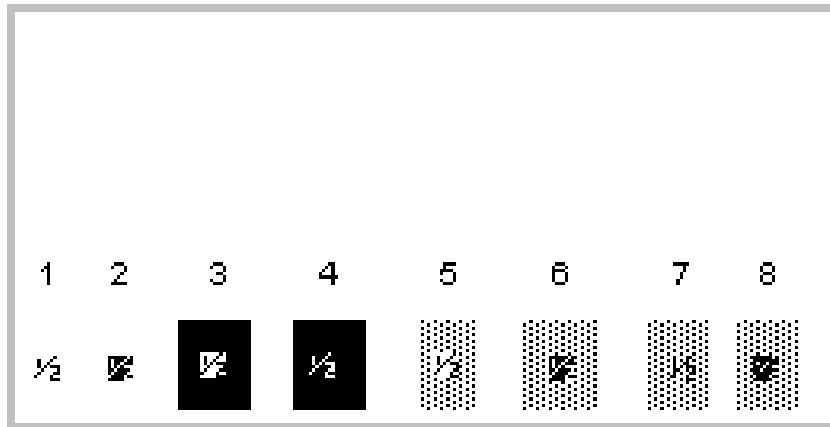
```
G 18 1 0 21 10      Dog Symbol
G 127 1 0 30 10    Don't Symbol
G 40 1 0 39 10     Goblet
G 246 1 0 48 10    Car Symbol
```

The resulting display appears (Dogs Don't Drink and Drive):



Another example shows the effects of the inverse and opaque options under different conditions. The most vibrant examples for stationary use are opaque noninverted on white, and opaque inverted on black. The ones that use transparency are good to use if the symbols are going to be moving around and the user needs to follow their motion. The examples below are described here:

1. The 1/2 symbol onto whitespace opaquely without inversion.
2. The 1/2 symbol onto whitespace opaquely with inversion.
3. The 1/2 symbol onto blackspace opaquely without inversion.
4. The 1/2 symbol onto blackspace opaquely with inversion.
5. The 1/2 symbol onto a gridded space opaquely without inversion.
6. The 1/2 symbol onto a gridded space opaquely with inversion.
7. The 1/2 symbol onto a gridded space transparently without inversion.
8. The 1/2 symbol onto a gridded space transparently with inversion.



This command completes in different times depending upon the options selected and the number of white dots in the symbol. White (blank) dots in a symbol are not drawn in transparent mode, and shorten the drawing time for mostly white symbols. Heavily black symbols (or inverted mostly-white symbols) take longer in transparent mode since all the dots will be drawn on the graphics screen. The formula for computing the approximate completion time is:

$2100 + (153 \times N)$  microseconds (N is the number of dots drawn or cleared)

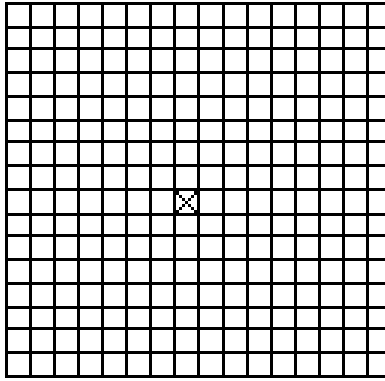
For example, placing the blank character opaquely on the screen takes  $2100 + (153 \times 64) = 11,900$  microseconds since 64 dots are forced off. Conversely, placing the blank character transparently on the screen only takes 2100 microseconds, since no dots are turned on OR off.

## Q - Double-Size Symbol Instantiate

Command Format:            Q sel opaque inverse x y

This command is identical to the Graphics Symbol Instantiate command except that the graphic symbol selected is doubled in size. It supports anywhere-on-screen placement, and the opaqueness and inversion options, and functions with any of the 256 selectable symbols. As in the "G" command, the inversion option is exercised prior to the transparency option.

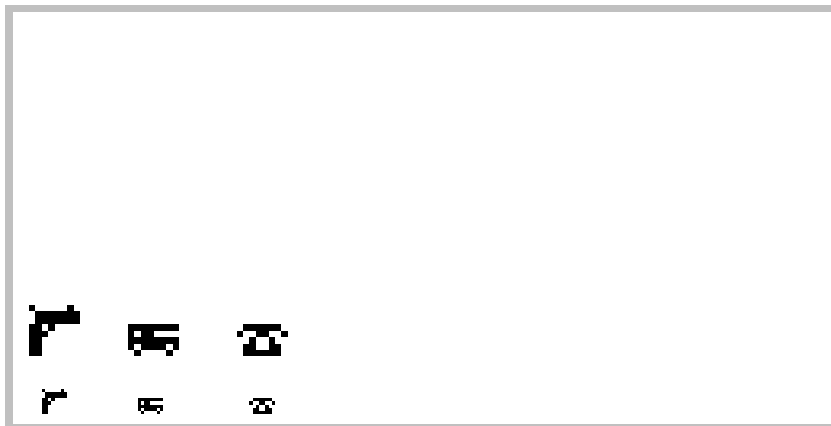
If you try to instantiate the symbol off screen, NOTHING is displayed and the command will immediately complete. If you instantiate the symbol such that a portion of the symbol falls off the display, only the dots that fall inside the screen are drawn. The x,y location you specify designates the "center" of the symbol area. In the symbol map below, the "X" specifies the center reference point for the symbol.



Remember that each double symbol occupies a sixteen-by-sixteen area of dots and takes about four times as long to complete as an equivalent normal-sized symbol. This command completes at different rates depending on how many dots are drawn or cleared. The formula for approximating the completion time is:

$$6000 + (153 \times N) \text{ microseconds} \quad (N \text{ is the number of dots drawn or cleared})$$

For example, placing any double-sized symbol opaquely on the screen takes approximately  $6000 + (153 \times 256) = 45,100$  microseconds. Here is a screen shot of several symbols and their double-sized versions:



### M - Make Graphics Symbol

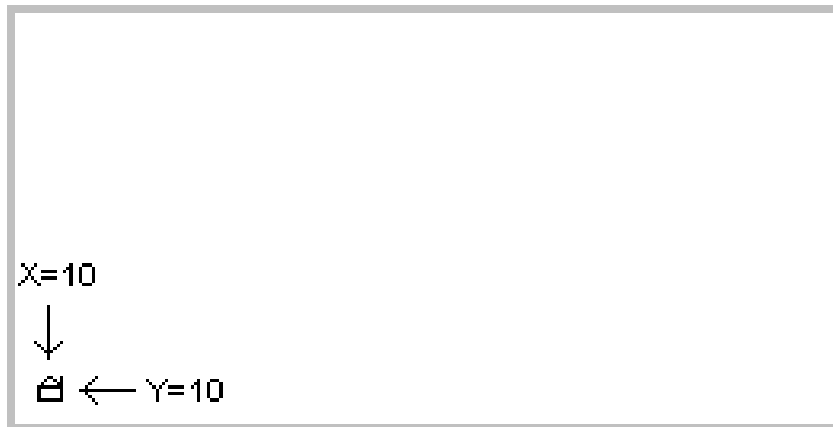
Command Format:            M sel d1 d2 d3 d4 d5 d6 d7 d8

This command allows you to define a custom symbol in the controller. A graphic symbol is limited to an 8x8 bit area (8 bytes worth). There are 16 user-definable symbols selected by the value in 'sel' from 0 to 15 decimal (00h to 0Fh). The symbols from 16 to 255 decimal (10h to FFh) are predefined in system ROM and cannot be changed (but can be copied into a user symbol). To define a symbol, supply the "M" command, the symbol number from 0 to 15, and then 8 bytes of data. The first byte should be the top row of bits. The left most side should be supplied in the most significant bit of each of the 8 bytes. A bit value of '1' will turn on the black dot. As an example, let's build a symbol of a house with a chimney into user symbol selection 0:

			X	X			X	0	0	0	1	1	0	0	1	=	19h	
		X				X	X	0	0	1	0	0	1	0	1	=	25h	
	X						X	X	0	1	0	0	0	0	1	1	=	43h
X	X	X	X	X	X	X	X	X	1	1	1	1	1	1	1	1	=	FFh
X								X	1	0	0	0	0	0	0	1	=	81h
X								X	1	0	0	0	0	0	0	1	=	81h
X								X	1	0	0	0	0	0	0	1	=	81h
X	X	X	X	X	X	X	X	X	1	1	1	1	1	1	1	1	=	FFh

The command to define the example symbol is:

M 00h 19h 25h 43h FFh 81h 81h 81h FFh . When defined and instantiated on a blank screen at location X=10 Y=10, the above definition appears as shown below. Remember that the lower leftmost dot of the screen is at 0,0, not 1,1.



At power on reset and after an initialize command, all the user-defined symbols are set to all-off dots. If you supply a selector higher than 15 (0Fh), the entire command will be ignored, and all the data will be read and discarded.

The center of the symbol is the same as that in the graphics symbol instantiation command, namely the 4th row up, and the 4th row over from the left.

The symbol definition feature allows defining up to sixteen custom graphics symbols for your particular application. This command completes in 346 microseconds.

## Y - Yank Graphics Into A User Symbol

Command Format:            Y   symbnum   x   y

This command reads graphic data off the selected-for-editing screen in an 8x8 dot format centered on point x,y , and stuffs the bytes into the user symbol number 'symbnum' from 0 to 15. X and Y are centered as described in the graphic symbols section above and must be located somewhere on screen. If you specify an offscreen location for Y, the symbol number you specified is cleared. If the center location is on screen, but some of the dots in the yanked symbol would be off screen, then zeros (off dots) are assumed and put into those positions of the user symbol. The yank command is useful for saving an 8x8 area of screen, then using graphics symbol instantiate command to drop copies of that new symbol anywhere you want. It is also useful for implementing a symbol-based cursor. Simply yank the area where you want to

put the cursor, instantiate the graphic symbol of your choice for the cursor, and when it's time to restore the graphics screen and remove the cursor, reinstantiate your yanked symbol back where it was.

The yank command completes in about 8.8 milliseconds.

## **X - XOR Symbols**

Command Format:           X sourcesymbol destinationsymbol

This function is used to combine any source symbol from the available 256 graphics symbols with and into a destination user symbol using the XOR function. Applications include copying a fixed symbol into a user symbol, combining two symbols together, or creating an XOR symbol based cursor.

There are a couple methods to copy any of the 256 graphic symbols into a user symbol. One way is to instantiate the desired source symbol onto a graphic screen, then use the symbol Yank command. To copy a source symbol using the XOR feature, first use XOR command to XOR a user symbol into itself. This will clear the user symbol. Then you can use the XOR symbol to XOR a sourcesymbol into your now-cleared destination user symbol.

To combine two symbols together using XOR, use the above method to get the first desired symbol into one of the sixteen user symbols. Then use the XOR command to exclusive-OR another symbol into the destination user symbol.

If you want to create a symbol based cursor using XOR, first you capture the 8x8 graphic screen target area using YANK. Then you XOR your desired pointer symbol into your yanked symbol. Then you instantiate your modified user symbol onto the graphics area where you originally yanked the screen data from. When you want to restore your screen and remove the 'cursor', you XOR the selected pointer symbol back into your user symbol, thereby returning it to the original yanked screen data. Then reinstantiate your user symbol, returning the screen data back to original. This method results in a cursor that shows up in black or white areas since the XOR operation always selects a dot color opposite to that on the screen.

You can only XOR into a user-defined symbol from 0 to 15. As such, the value for destinationsymbol is internally taken modulus 16 to force it into the correct range. The sourcesymbol value can be any symbol from 0 to 255. This means you can combine user symbols as well. Note that if you XOR a user symbol into itself, it results in clearing the symbol to all blank.

The XOR symbol command executes in approximately 275 microseconds.

## **z - Graphic Virtual Text Characters Block Size Command**

Command Format:           z pelswide pelstall

The virtual cursor graphic characters feature allows a virtual character pointer for use on the graphics display. The real text characters are spaced out at 8 pixels per character, due to the shared setting of the horizontal byte spacing of 8 pixels per graphic byte. This can leave large gaps between text characters since there are 32 per line, and usually 3 pixels between characters. This adjustable virtual text character feature allows putting text and symbol strings on a graphic screen with a built-in

cursor pointer to simplify putting differently spaced out characters to the graphic screen, making them more readable.

This size command (lower case "z", hexadecimal 7A, decimal 122) allows describing the width and height of the virtual character block. The characters are dropped onto the graphics screen, and a virtual row and column pointer are incremented, with wrap around, with a character spacing set by pelwide and pelstall. When a virtual graphics character string is placed using the "t" graphics text command, the location of each successive character is adjusted by pelwide pixels across in the horizontal direction, and when it wraps to a new line, the new line is adjusted down by pelstall. You may set pelwide to anything from 0 to 255 pixels. You may set pelstall to anything from 0 to 255 pixels, but any value greater than 128 is forced to 128, since the screen is only 128-pixels tall.

This command **DOES NOT** change the size of the symbols plotted, just the horizontal and vertical spacing of each symbol thus plotted. The standard ASCII set of characters is reproduced in the symbol set, and generally fit inside an eight-pixel tall by five-pixel wide block. Setting pelwide to a value of 06h (6 decimal) will plot symbols with six pixels between their centers, making for one pixel of white space between the typical ASCII characters. Note that some of the symbols (as shown in the symbol table later in this document) take more or fewer pixels to display. If you are anticipating using symbols beyond the ASCII set, be sure to set the horizontal size large enough so as not to overlap the plotted symbols (unless that is your goal).

When the size command is executed, the virtual graphics text row and column pointers are reset to row 0 and column 0. Note that characters in row 0 are always plotted in the fifth dot row down from the top (dot row number 123). Characters in column 0 are plotted centered in the fourth dot column from the left (dot column number 3). This is the logical character center for the topmost fully displayed symbol. If you want to mix the spacings of graphically plotted text strings, you must plot one string using one spacing setting first, then change the block sizes using this command, then adjust the graphic text pointers using the "p" graphics position command (lowercase "p" pointer command), then plot the next character string using the new spacings.

The actual position of the character plotting locations are described in the "p" graphic pointer command. The characters plotted using the graphic text plotting command "t" (lowercase t) are always plotted in a transparent mode, where whitespace will not overwrite existing black dots. This allows symbols and characters surrounded by whitespaces to be plotted closer than eight pixels apart without erasing existing black dots.

The following table describes the number of columns of text for selected pelwide values. In general, the number of columns equals the integer part of the quotient of 256/pelwide. The column numbers used in the "p" position command number from zero up to one less than the number of columns for the selected horizontal spacing.

pelwide	Columns Of Text
6	42 (0 to 41)
7	36 (0 to 35)
8	32 (0 to 31)
9	28 (0 to 27)
10	26 (0 to 25)

The following table describes the number of rows of text for selected pelstall values. In general, the number of rows equals the integer part of the quotient of 128/pelstall. The row numbers used in the "p" position command number from zero up to one less than the number of rows for the selected vertical spacing.

pelstall	Rows Of Text
7	18 (0 to 17)
8	16 (0 to 15)
9	14 (0 to 13)
10	12 (0 to 11)

## **t - Graphic Virtual Text Characters String Plotting**

Command Format:            t charcount ...charcount\_characters...

The "t" (lower case t) text plotting command is used to plot symbol strings to the edit-selected graphics screen. This feature allows mixing character strings and graphics on the graphics layer, AND supports character spacing of a user definable pixel width and height allotted per standard sized character. The built-in text layer characters are fixed at eight pixels horizontal spacing and eight pixels vertical spacing, and might be difficult to read with three white pixel spaces between them horizontally. This virtual text feature allows you to easily plot character strings and symbols at a user-defined spacing on the graphics layer.

The "t" command is followed by a single byte character count from 0 to 255 decimal. A count of zero is interpreted as a count of 256 characters. The characters are plotted transparently, allowing existing black dots to show through white spaces in the symbols. The existing graphic symbols (and the sixteen user-definable symbols) are what get plotted. Following the character count byte, you must send charcount number of bytes, each representing a character or symbol to be plotted. They are plotted from left to right across the screen, with automatic line wrapping according to the block width and block height you set in the "z" size command. The number of characters across is determined by the setting of the pixel spacing width set in the "z" size command. The number of rows of characters from top to bottom is determined by the setting of the pixel spacing height set in the "z" command.

While plotting characters, if the virtual cursor shifts beyond the current row, it wraps back to the left column and down a line. If that new line is below the bottom of the screen, then the virtual cursor wraps back to the top of the screen, just like the real character cursor.

The character and symbol strings plotted using this command are not individually clearable like text in the text layer. These characters are plotted as dots on the edit-active graphics layer. If you have plotted these strings on dedicated space on the graphics layer, you may use the box clear function to erase the character to white space.

This command executes in approximately two milliseconds multiplied by the number of characters being sent.

## **p - Graphic Virtual Text Characters Cursor Position**

Command Format:            p textrow textcol

The "p" (lowercase P) command allows setting the virtual graphics character cursor position. This position indicates the row and column (zero based) to which a graphic text string command "t" (lowercase t) will start putting symbols and characters. The top row is textrow 0, and the leftmost column is textcol 0. As each graphic symbol in the graphic text command is placed, the virtual cursor row and column are incremented from left to right, and then top down, simulating the real character layer cursor pointer. The graphic virtual cursor pointer movement-per-character can be adjusted for horizontal and vertical spacing using the "z" size command.

When the "z" size command is issued, the current values of textrow and textcol are automatically reset to row 0, column 0. If you are changing the spacing of the characters to be plotted, you should also ensure you've set the cursor position to the desired point. This is because switching to a smaller or larger pitch resets the cursor location on the screen.

As described in the "z" size command, the number of rows and columns are affected by the width and height spacing of the characters. If you supply a value for textrow or textcol that falls off the screen according to the current size settings, then the off-screen parameter is forced to its maximum allowable location horizontally or vertically on the screen.

The actual positions of the row and column are defined as a function of the selected pelswide and pelstall parameters in the "z" size command.

The X position of the center of the symbol to be plotted is defined by:

$$X = 3 + (\text{textcol} \cdot \text{pelswide})$$

The Y position of the center of the symbol to be plotted is defined by:

$$Y = 123 - (\text{textrow} \cdot \text{pelstall})$$

### V - Video Block Transfer

Command Format:            V mode x1 x2 y1 y2 data.....

This allows you to move blocks of byte-aligned bitmap data into the graphic display memory quickly. 'mode' allows four different ways of putting the data into memory. Bit 0 of the mode byte is the "invert video flag". Bit 1 of the mode byte is the "opaque video flag".

7	6	5	4	3	2	1	0
0	0	0	0	0	0	Opaque	Invert

Mode Value	Function
0	positive data transparently
1	negative data transparently
2	positive data opaquely
3	negative data opaquely

In positive mode (INVERT=0), a '1' bit in the data stream will turn a black dot on. In negative mode (INVERT=1), a '1' bit is a white or transparent dot.

In opaque mode (OPAQUE=1), both black and white video dots overwrite the existing data on the graphics screen. In Transparent mode (OPAQUE=0), existing dots on the screen will show through the white areas of the new video data.

- **X1** is the leftmost byte column and ranges from 0 to 31.
- **X2** is the rightmost byte column and ranges from 0 to 31 and must be greater than or equal to X1.
- **Y1** is the lowest line number (from 0 to 127) where data will be placed.
- **Y2** is the highest line number where byte data will be placed and must be greater than or equal to Y1.

All X and Y position data is inclusive, meaning that the start and ending ranges will receive data.

**Data...** is the bit block data bytes to be placed on the graphics screen. The number of data bytes required for the command is:

$32 \times (Y2 - Y1 + 1) \times (X2 - X1 + 1)$  (The ones are added because of end-inclusiveness).

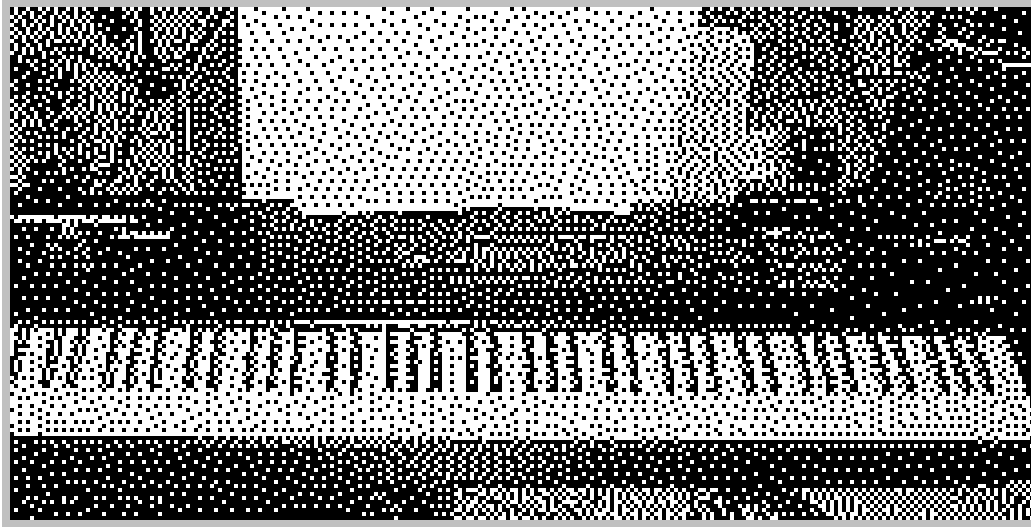
Only whole bytes are placed on byte-aligned dot sets of the screen. No partial bytes are allowed. Therefore, the bitmap must be a multiple of 8 dots wide, but can be any number of dots tall from 1 to 128.

Example: Place a 120-dot wide by 60-dot tall pixelmap onto the screen in inverse video transparently at the lower left corner dot position 16,7 (x,y):

```
"V"  02h  02h  10h  07h  42h  [900 bytes data]
cmd  mode x1   x2   y1   y2   data
      2d   16d  7d   66d
```

- ⇒ X1 is calculated as  $\text{INTEGER}(\text{left dot column}/8)$  or  $16/8 = 2$
- ⇒ X2 is calculated as  $X1 + \text{INTEGER}(\text{width}/8) - 1$  or  $2 + 120/8 - 1 = 16$
- ⇒ Y1 is the same as the desired bottom most line which is 7 in this example.
- ⇒ Y2 is calculated as  $Y1 + \text{tall} - 1$  or  $7 + 60 - 1 = 66$

The left-most desired dot position MUST be a multiple of 8 (0, 8, 16,...). To put a bitmap with a non-multiple-of-eight width in pixels, you must first crop or pad it to a whole number of bytes wide. A program, **convbmp.exe**, is included that converts a black and white bitmap into various data formats suitable for this command. Here is an example bitmap of a piano keyboard as shown on the display:



The best bitmaps to present should be high contrast in nature due to the black and white nature of the presentation. This command's completion time depends on the size of the block transferred and whether opaque or transparent mode is selected. The above full-screen picture transfer times are shown here.

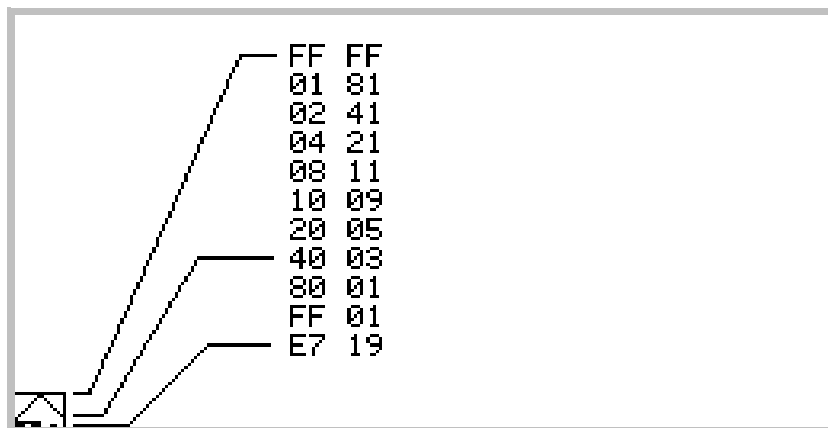
Opaqueness	Transfer Time in Milliseconds
Transparent	800
Opaque	552

Opaque transfer times may be approximated though the equation:

$$135 \text{ microseconds} \times \text{height of picture}(\text{lines}) \times \text{width of picture}(\text{column})/8$$

The byte block order is always sent in the order leftmost data to rightmost data, row by row from the bottom to the top, as described in the figure below. The picture in the bottom left corner was created by the command (shown in hexadecimal):

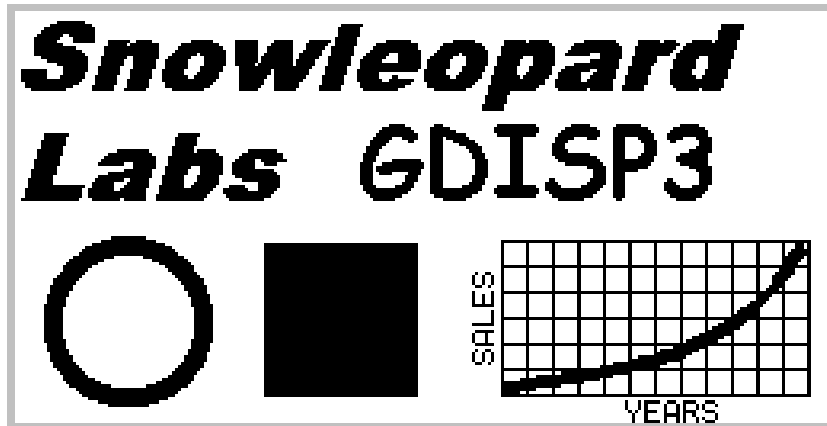
```
56 02 00 01 00 0A E7 19 FF 01 80 01 40 03 20 05 10 09 08 11 04 21 02 41 01 81 FF FF
V POS X1 X2 Y1 Y2 DATA.....
OPQ
```



The mini picture in the lower left corner is 16 dots wide (2 bytes wide) and 11 dots tall. The leftmost byte column (X1) is 00 (from 0 to 31, left to

right), and the rightmost column (X2) is 01. The bottom most destination row is 00 (from 0 at the bottom to 127 (7Fh) at the top), and the topmost destination row is 0Ah (10 decimal). Ten since all four edges are inclusive in the range (11 rows ranging from 0 to 10). Then 22 data bytes follow, going from left to right, row by row from the bottom up. For more information about the use of the program **convbmp.exe** to package black and white images into data suitable for the Video Block transfer command, run the convbmp.exe program without specifying any filename.

Here is another example of a bitmap that is not a photograph, but is perfectly suited for display:



## e - Ellipse

Command Format:            e inv cx cy rx ry

This command (lower case "e") will draw or erase an ellipse given the centerpoint (cx,cy), the x axis radius (rx), the y axis radius (ry), and an invert flag (inv). If 'inv' is zero, then the ellipse is drawn with black dots. If 'inv' is nonzero, then dots along the ellipse outline are erased to white. In either case, the ellipse outline is drawn opaquely, overwriting existing graphics data. The middle of the ellipse is transparent, not affecting anything already on the graphics page.

The value for CX must be from 0 to 255 decimal. CY must be from 0 to 127 decimal. RX must be from 0 to 127 decimal, and RY must be from 0 to 63 decimal. The entire ellipse must fall inside the screen area. Prechecks are performed upon the incoming data, and if any of the edges are predicted to fall offscreen, none of the ellipse is drawn. If both the x and y radius are zero, this command will default to just setting or clearing (if INV is set) the dot at cx,cy. Setting one of the two radii to zero results in drawing a line on the screen, although it is a slow method of drawing a single line.

If you set the RX and RY radii to the same value, a circle is drawn, since a circle is really a special type of ellipse of eccentricity 1.

Execution of this command takes approximately 189 microseconds per dot of circumference. For circles, it works out to RADIUS x 1.2 milliseconds.

## Command Summary Card

Z modevalue	Select mode 0 (2T+G) or mode 1 (2G)
D [0 0 0 0 gd gp td tp]	gd = graphics display on gp = graphics display page select td = text display on tp = text display page select
E [0 0 0 0 0 0 g t]	g = graphics edit page selected t = text edit page selected
S	Switch edit and display pages
C	Clear text page and home the pointer
W	Wipe graphics page
L x1 y1 x2 y2	Line draw from x1,y1 to x2,y2
F x1 y1 x2 y2	Fat (triple wide) line draw from x1,y1 to x2,y2
N dash x1 y1 x2 y2	Dashed line of 'dash' dash length from x1,y1 to x2,y2
U x1 y1 x2 y2 ...xn yninvalid	Unbroken chain of lines command (send Y>127 to stop the chain)
P row col	Set text pointer to row,col (0,0 is upper left)
T count TEXT	Put 'count' text characters to the text page
B mode x1 y1 x2 y2	Box of opposite corners x1,y1 and x2,y2. Modes: 0=blackline 1=whiteline 2=blackarea 3=whitearea 4=invertarea
I	Initialize command resets the display to defaults
H	Home the text pointer to row 0, column 0
O dotvalue x y	Set dot x,y to dotvalue
A	Alarm sounds for 1/4 second
K x y	Place graphics cursor to x,y saving affected dots
R	Restore the dots from the 'K' cursor
G sel opaq inv x y	Graphics symbol 'sel' to x,y with opaque and inverse video options
Q sel opaq inv x y	Double-sized graphics symbol 'sel' to x,y with opaque and inverse video options
M sel 8*databytes	Make graphics symbol 'sel' + 8 data bytes
Y sel x y	Yank into user symbol 'sel' from the graphics screen centered at x,y
X src dest	XOR symbol number src into user symbol number dest.
z pelswide pelstall	Set the graphic text spacing to pelswide horizontally and pelstall vertically.
t count TEXT	Put 'count' graphic symbols to the graphics page at the graphics virtual row and column pointer.
p row column	Set graphics virtual text pointer to row,col (0,0 is upper left)
V opq*2+inv x1col x2col y1line y2line data	Video bit block transfer
e inv cx cy rx ry	Draw ellipse, optionally in inverse video, centered at cx,cy, with X and Y radii xradius and yradius.

**Graphics Symbol Set for GDISP3 Control Card**

(For example, the symbol 8Ch is a goblet.)

SYMBOL SET FROM FILE GPS.TXT

F0																
E0																
D0																
C0																
B0																
A0																
90																
80																
70																
60																
50																
40																
30																
20																
10																
00																
	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F

## Graphics Symbol Listing

HEX INDEX	DECIMAL INDEX	DESCRIPTION
00	000	USER 0
01	001	USER 1
02	002	USER 2
03	003	USER 3
04	004	USER 4
05	005	USER 5
06	006	USER 6
07	007	USER 7
08	008	USER 8
09	009	USER 9
0A	010	USER 10
0B	011	USER 11
0C	012	USER 12
0D	013	USER 13
0E	014	USER 14
0F	015	USER 15
10	016	CAMEL
11	017	DEER
12	018	DOG
13	019	FEMALE
14	020	MALE
15	021	FISH
16	022	OSTRICH
17	023	SMILEY
18	024	ANCHOR
19	025	BOAT RAMP
1A	026	BOUY
1B	027	SHIPWRECK
1C	028	CARDCLUBS
1D	029	CARDDIAMONDS
1E	030	CARDHEARTS
1F	031	CARDSPADES
20	032	SPACE
21	033	EXCLAMATION
22	034	DOUBLEQUOTE
23	035	POUND
24	036	DOLLAR
25	037	PERCENT
26	038	AMPERSAND
27	039	TIC
28	040	OPENPARENTHESSES
29	041	CLOSEPARENTHESSES
2A	042	ASTERISK
2B	043	PLUS
2C	044	COMMA
2D	045	MINUS
2E	046	PERIOD
2F	047	SLASH
30	048	0
31	049	1
32	050	2

HEX INDEX	DECIMAL INDEX	DESCRIPTION
33	051	3
34	052	4
35	053	5
36	054	6
37	055	7
38	056	8
39	057	9
3A	058	COLON
3B	059	SEMICOLON
3C	060	LESSTHAN
3D	061	EQUALS
3E	062	GREATERTHAN
3F	063	QUESTION
40	064	AT
41	065	A
42	066	B
43	067	C
44	068	D
45	069	E
46	070	F
47	071	G
48	072	H
49	073	I
4A	074	J
4B	075	K
4C	076	L
4D	077	M
4E	078	N
4F	079	O
50	080	P
51	081	Q
52	082	R
53	083	S
54	084	T
55	085	U
56	086	V
57	087	W
58	088	X
59	089	Y
5A	090	Z
5B	091	LEFTBRACKET
5C	092	LEFTSLASH
5D	093	RIGHTBRACKET
5E	094	CARAT
5F	095	UNDERSCORE
60	096	LEFTTIC
61	097	a
62	098	b
63	099	c
64	100	d
65	101	e

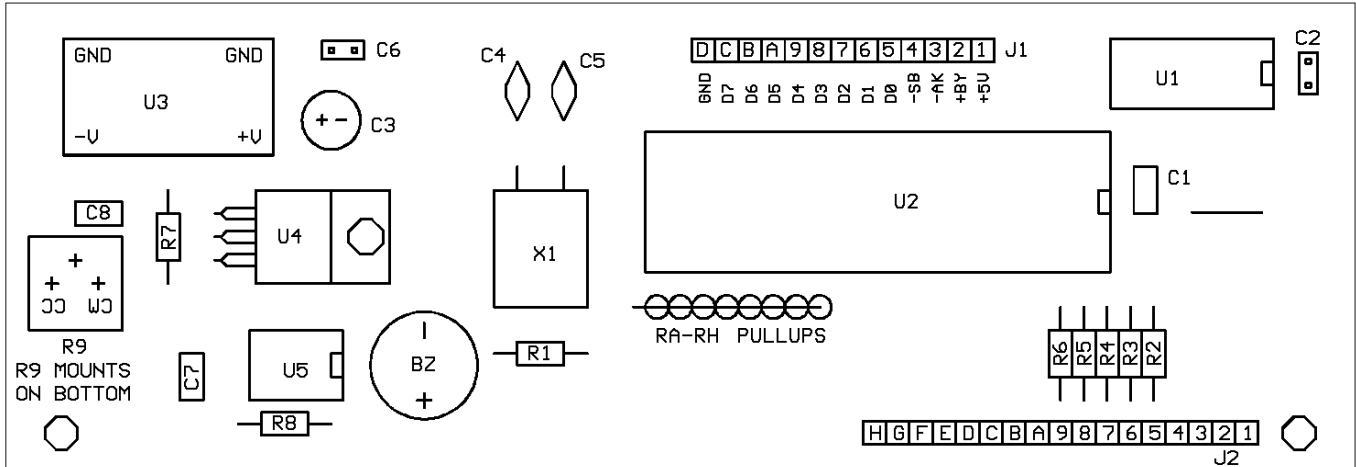
HEX INDEX	DECIMAL INDEX	DESCRIPTION
66	102	f
67	103	g
68	104	h
69	105	i
6A	106	j
6B	107	k
6C	108	l
6D	109	m
6E	110	n
6F	111	o
70	112	p
71	113	q
72	114	r
73	115	s
74	116	t
75	117	u
76	118	v
77	119	w
78	120	x
79	121	y
7A	122	z
7B	123	LEFTBRACE
7C	124	VBAR
7D	125	RIGHTBRACE
7E	126	TILDA
7F	127	NO
80	128	BARN
81	129	CHURCH
82	130	COURTHOUSE
83	131	HOUSE
84	132	SCHOOL
85	133	SHELTER
86	134	BATTERY
87	135	CAPACITOR
88	136	COMPUTER
89	137	DISKETTE
8A	138	POWER PLUG
8B	139	COFFEE CUP
8C	140	GOBLET
8D	141	GOLDEN ARCHES
8E	142	KNIFE SPOON
8F	143	PICNIC TABLE
90	144	MICRO
91	145	OHM
92	146	PI
93	147	SIGMA
94	148	EYEGLASSES
95	149	HOSPITAL
96	150	RED CROSS
97	151	MEDICAL WORMSTK
98	152	COMPACT DISK
99	153	MUSICAL NOTE
9A	154	BARBELL
9B	155	BED
9C	156	BOWLING
9D	157	BRIDGE

HEX INDEX	DECIMAL INDEX	DESCRIPTION
9E	158	CAMERA
9F	159	BANK
A0	160	BLKSMALLCHKRS
A1	161	WHTSMALLCHKRS
A2	162	BLKBIGCHKRS
A3	163	WHTBIGCHKRS
A4	164	DOOR
A5	165	ENVELOPE
A6	166	HANDGUN
A7	167	LADDER
A8	168	RECYCLING
A9	169	SHIRT
AA	170	SP/SHOPPING
AB	171	TOXIC
AC	172	WINDOW
AD	173	MOUNTAIN
AE	174	PINETREE
AF	175	TENT
B0	176	+/-
B1	177	1/2
B2	178	1/4
B3	179	<<
B4	180	<=
B5	181	>=
B6	182	>>
B7	183	APPROXIMATELY
B8	184	DEGREES
B9	185	DIVIDE BY
BA	186	DOT AND
BB	187	IDENTICAL
BC	188	INTEGRAL
BD	189	SQUAREROOT
BE	190	TIMES
BF	191	CHECKMARK
C0	192	ARROW-EAST
C1	193	ARROW-NE
C2	194	ARROW-NORTH
C3	195	ARROW-NE
C4	196	ARROW-WEST
C5	197	ARROW-SW
C6	198	ARROW-SOUTH
C7	199	ARROW-SE
C8	200	TRIANGLE-EAST
C9	201	TRIANGLE-NE
CA	202	TRIANGLE-NORTH
CB	203	TRIANGLE-NE
CC	204	TRIANGLE-WEST
CD	205	TRIANGLE-SW
CE	206	TRIANGLE-SOUTH
CF	207	TRIANGE-SE
D0	208	UP/DOWN
D1	209	LEFT/RIGHT
D2	210	RIGHT TURN
D3	211	LEFT TURN
D4	212	CEMETARY
D5	213	CHRISTIANCROSS

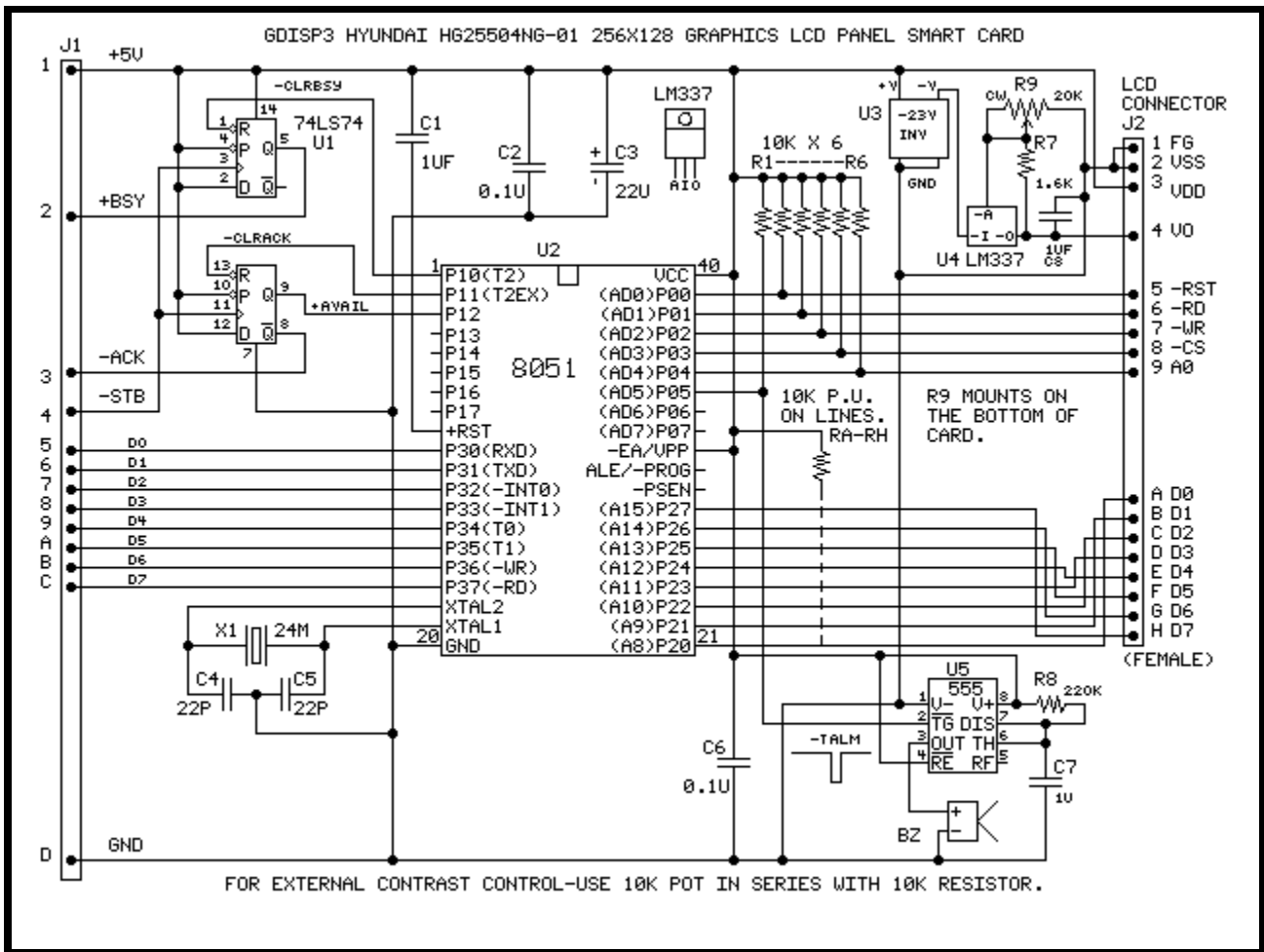
HEX INDEX	DECIMAL INDEX	DESCRIPTION
D6	214	MENORA
D7	215	TOWNCENTER
D8	216	TOLL
D9	217	FUEL
DA	218	INTERSTATE
DB	219	PARKING
DC	220	RR-CROSSING
DD	221	ROUTESYMBOL
DE	222	STOPLIGHT
DF	223	STOPSIGN
E0	224	TELEPHONE
E1	225	TELEPHONE POLE
E2	226	TIRE
E3	227	TP/TURNPIKE
E4	228	XT/EXIT
E5	229	CIRCLE-X
E6	230	QUESTION BOX
E7	231	CIRCLE HOLOW SML
E8	232	CIRCLE SOLID SML
E9	233	CIRCLE SOLID BIG
EA	234	SQUARE HOLOW SML

HEX INDEX	DECIMAL INDEX	DESCRIPTION
EB	235	SQUARE HOLOW BIG
EC	236	SQUARE SOLID BIG
ED	237	5 POINT STAR
EE	238	SIX POINT STAR
EF	239	FOOTBALL
F0	240	PUTTER
F1	241	HAMMER
F2	242	WRENCH
F3	243	AIRPLANE
F4	244	BICYCLE
F5	245	BUS
F6	246	CAR
F7	247	POLICE CAR
F8	248	CRANE
F9	249	FIRETRUCK
FA	250	FLATBED TRUCK
FB	251	PICKUP TRUCK
FC	252	POWERBOAT
FD	253	ROCKET
FE	254	SAILBOAT
FF	255	TOW TRUCK

## Controller Card Component Layout

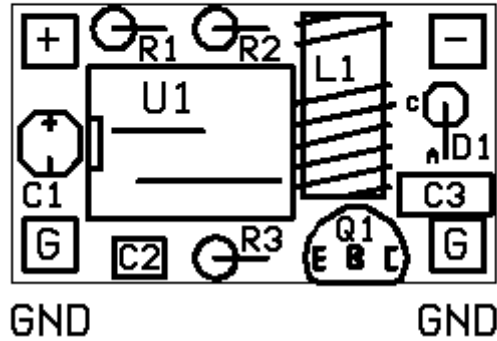


## GDISP3 Schematic

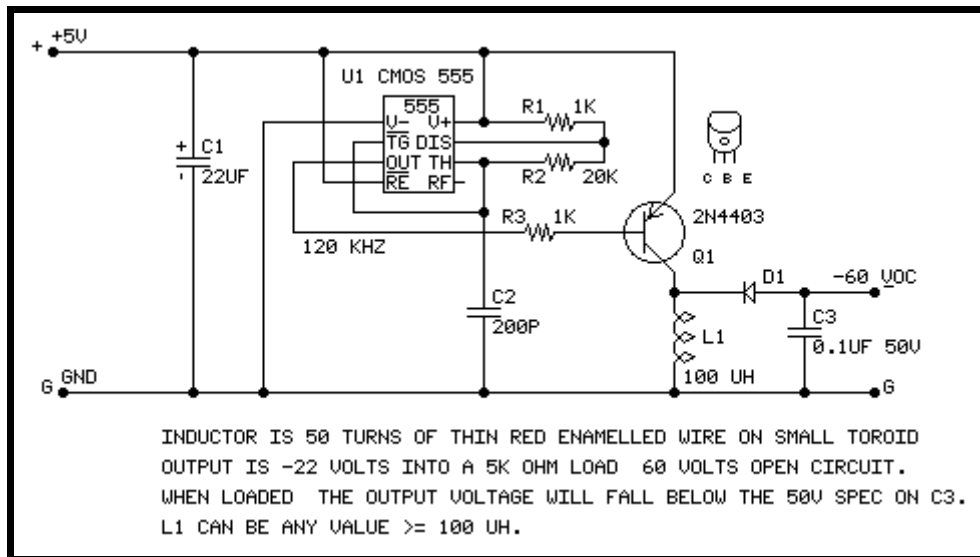


## Voltage Inverter Component Layout

NEGBST - NEGATIVE  
VOLTAGE BOOSTER.  
+5V -23V



## Voltage Inverter Schematic



© 2004 by Duane K. Becker, Snowleopard Labs

The architecture, design, software, firmware, and features of the GDISP3 card remain the sole property of Duane K. Becker.