CSYS Labs, Inc.

Touch Vision™

TVM2464 Designer's Manual

TVM2464BTC
TVM2464LTC

Touch Vision™

A Magnificent Display of Intelligence!

TVM2464 LCD Modules

C Sys Labs, Inc. 1430 Koll Circle Suite 103, San Jose, CA 95112 (408) 453-5380

Table of Contents

1 General Description

The TVM2464 Touch Vision Module provides a complete user interface in a small, low cost, convenient to use module. It combines a graphics LCD display, touch activated panel overlay, interface electronics, LCD drivers, back-light, and power supply voltage converters. The TVM2464BTC requires a single +5V power supply. The TVM2464LTC, which uses a LED backlight system requires an additional +12V, 750 mA power supply.

The TVM2464 includes a powerful custom programmed micro-processor which acts as an intelligent controller for the entire module and makes interfacing to the TVM2464 very easy. Communication with the TVM2464 is done through a 16 to 20 wire parallel port. Hand shaking signals are provided to make interfacing as simple as possible. A status register is also provided to better accommodate pipe lined communication.

A rich instruction set, for both text and graphics commands is provided. With these commands, the user can freely mix and manipulate graphics and text being displayed. Simple commands can be issued to position the cursor, set up the text window size and set system attributes. Graphics commands are provided to draw rectangles, boxes, lines, vectors or to set individual pixels.

Additional commands can be used to activate a touch panel switch area as well as place a button outline on the LCD display automatically. This makes writing soft buttons (program controlled keys) very easy. The program loads the TVM2464 with the button label and places the button via a single instruction. If enabled, the software automatically sizes the button, places the text in the center of the button, writes the button to the display, and then activates the touch panel in the button's location. When the button is pressed, the on board CPU signals the host CPU that a button has been actuated and makes the button's code available for reading by the host CPU. Other button commands may be used to activate a "phantom button" area, delete a button, or delete all buttons.

The TVM2464 has the ability to simultaneously mix as many as 5 fonts on the screen. Two of the fonts are pre-programmed into the TVM2464 controller. The other three fonts may be custom compiled using a font compiler and down loaded into the TVM2464's font memory. Any font may be used at any time for labels anywhere, including text inside a defined button area.

The TVM2464 has the capability to adjust LCD contrast electronically. This eliminates the need for any hardware contrast adjustments by end users of the equipment. In addition, a course adjustment trimmer is provided on the controller board to accommodate initial factory settings.

An EL back-light panel (including high voltage driver) is included on the TVM2464BTC. Both the EL back-light and the LED back-light in the TVM2464LTC can be turned ON and OFF via an on board switch by issuing a simple command to either module. The EL back-light panel has an expected half life in excess of 10,000 hours.

1.1 TVM2464 Features

* 240 X 64 Super Twist LCD Display
* 3 X 10 Matrix Touch Panel Overlay
* Single +5V Power Supply Operation (TVM2464BTC)
* Low Power Consumption (TVM2464BTC)
* EL or LED Back-light
* Software Controlled Back-light
* Software Controlled Electronic Contrast
* 2 Built in Fonts
* Up to 3 Down Loaded Soft Fonts
* Audio Alarm and/or Key Click
* Freely Mix Text and Graphics
* Text Window Defined via Software
* Automatic Scrolling in Text Window
* Freely Mixed Multiple Font Types
* Automatic Button Generation and Placement
* An Abundance of Graphics Commands
* Efficient 8 Bit Parallel Interface

1.2 Mechanical Description

Many mechanical features have been designed into the TVM2464 which simplify the design of the module into new equipment. Mounting can be accomplished by using 6-32 screws through either holes or slots on the TVM2464BTC or through the slots on the TVM2464LTC. Holes or slots not used for mounting, can be used to fasten additional components to the TVM2464BTC. The bezel opening has been enlarged beyond the actual viewing / touch panel area by .110 inches in all directions to accommodate case overlap of the bezel. The LCD is protected by a glass backed hardcoated polyester touch panel. The dimensional drawing of the TVM2464BTC, Figure 1.1, shows the location of critical components such as the sound transducer and connectors. A component easement area has also been specified which provides adequate clearance of the back side components. The location of the sound transducer is specified to allow design of case openings or additional component relief in the equipment. *Not having enough clearance behind the transducer may degrade sound quality*. The connectors are a standard ribbon cable type having two rows of .100 spaced male headers. Pin one is designated on the connector. The contrast gross adjustment potentiometer, R18, is also shown in the lower middle of the back side view.

The TVM2464LTC is a sandwich consisting of the LCD/Touch Panel assembly, the LED backlight board, and the Touch Vision controller board. The holes located in the corners of the controller board are used to mount the controller board to the LED board via 5/16 inch stand-offs. The LTC bezel is designed to contain an Acrylic diffuser and has built in stand-offs which space the LCD assembly from the LED board. This spacing allows convection cooling of the LEDS. Sufficient air flow must be maintained around the unit, through the use of vents or a fan in your equipment, to allow proper cooling of the unit. The temperature compensation built into the controller board can only be accurate if the LCD and controller are kept at the same temperature. Software contrast adjustments may be required under conditions created by some enclosures.

Mechanical Outlines:

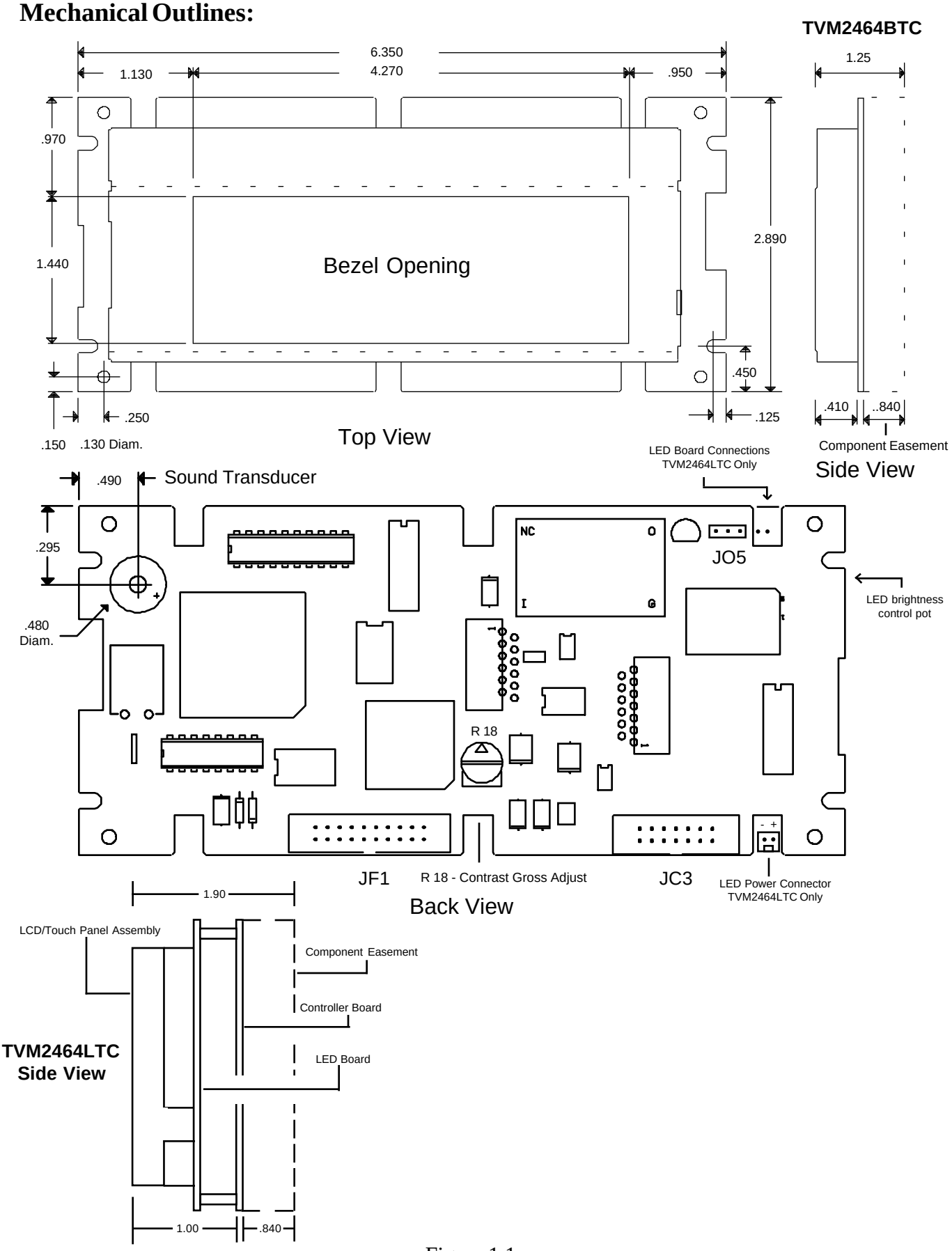


Figure 1.1

1.3 Display and Touch Panel Layout

The LCD display is organized in an array 240 dots wide by 64 dots tall. The upper left hand corner of the display is always 0,0 (X,Y). The lower right is 239,63. All coordinates, whether for cursor placement or for placements of graphics use the same coordinate entry methodology and are always referenced to the upper left corner of the display.

The touch panel is organized as an array, 10 wide by 3 tall, of touch sensitive cells. The upper left hand cell is referred to as 0,0 and the lower right cell is 9,2. Each cell is 20 pixels wide by 16 pixels high. A horizontal space 4 pixels wide and a vertical space 8 pixels wide is left as an easement between each touch panel cell.

Figure 1.2 below details the LCD display with the touch panel overlay.

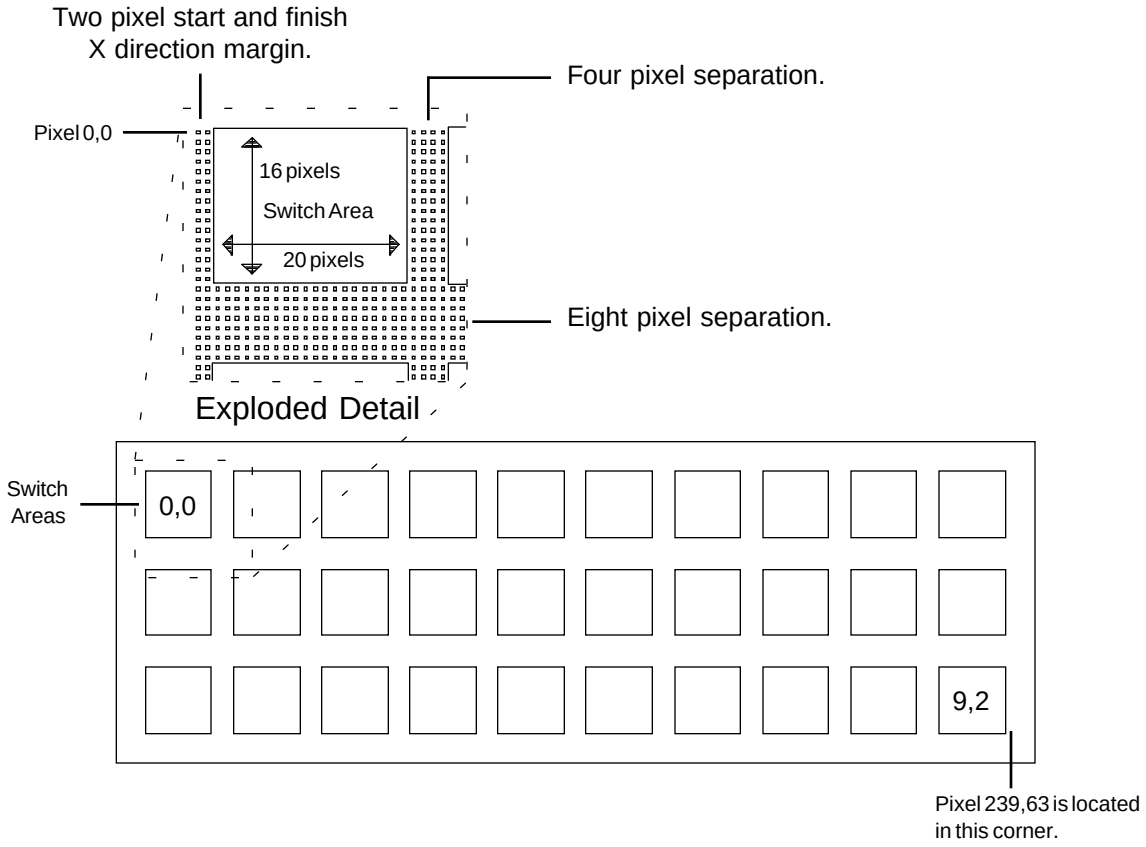


Figure 1.2

1.4 System Interface

Connecting the TVM2464 to the main micro-processor is done via a 20 pin ribbon cable which is connected to JF1 at the bottom of the module. Power and ground are provided to the module through this cable. The TVM2464 uses an 8 bit instruction/data bus as well as 2 address lines. There are also several control signals and status lines. Table 1.1 lists the interface signal names and locations.

Pin #	Function	Description	Type
1	VSS	VSS Power connection	Power
2	RESET/	Module Reset, Negative	Out
3	DEN/	Module Enable, Negative	In
4	DRD/	Read, Negative	In
5	DWR/	Write, Negative	In
6	DIBF	Input Buffer Full, Positive	Out
7	DOBF/	Output Buffer Full, Negative	Out
8	ERROR	Module Error, Positive	Out
9	KEYPRESS	Key Pressed Flag, Positive	Out
10	DA0	Address 0	In
11	DA1	Address 1	In
12	D0	Data 0	I/O
13	D1	Data 1	I/O
14	D2	Data 2	I/O
15	D3	Data 3	I/O
16	D4	Data 4	I/O
17	D5	Data 5	I/O
18	D6	Data 6	I/O
19	D7	Data 7	I/O
20	VCC	Power	Power

Table 1.1

Instructions for the TVM2464 are always written to address 0. If the instruction requires additional data, the data is written to address 1. This is done to facilitate data strings of arbitrary length such as text input or down loading of fonts. String data is always terminated by writing the next instruction to address 0. Instructions having data of an arbitrary amount are referred to as having “String” data.

Any instruction requiring more data before it can execute may be aborted by writing the next instruction to the instruction register.

Some instructions return data from the TVM2464. This returned data is always read from address 0. The status register may be read at any time from address 3. Table 1.2 summarizes the address mapping.

DA1	DA0	Write	Read
0	0	Instruction	Data
0	1	Data	
1	0		
1	1		Status

Table 1.2

Table 1.3 shows the signals needed to interface to the TVM2464 and resulting actions.

DEN/	DWR/	DRD/	DA0	DA1	Action
1	x	x	x	x	No Action
0	1	1	x	x	No Action
0	0	1	0	0	Write Instruction
0	0	1	1	0	Write Data
0	0	1	0	1	Illegal
0	0	1	1	1	Illegal
0	1	0	0	0	Read Data
0	1	0	1	0	Illegal
0	1	0	0	1	Illegal
0	1	0	1	1	Read Status Register

Table 1.3

1.4.1 Signal Description

The following describes the TVM2464 interface signals in more detail.

1.4.1.1 RESET/

A logic 0 level on this pin causes a reset of the module. If this pin is left open, the internal RC reset network on the module will cause a reset. It should be noted that the internal capacitor in the TVM2464 will hold down the RESET/ line and any external circuitry tied to the this pin. Fig. 1.3 shows the internal reset circuit.

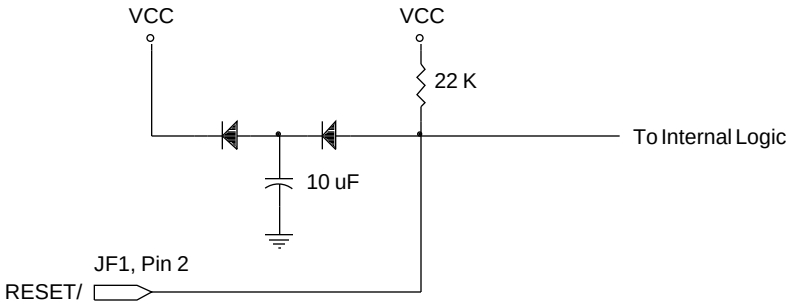


Fig. 1.3

1.4.1.2 DEN/, DRD/, DWR/

A logic 0 level on DEN/ and DWR/ will cause a write to the TVM2464. A logic 0 level on DEN/ and DRD/ will cause a read from the TVM2464. Please refer to Table 1.3 for DA0 and DA1 addressing information.

1.4.1.3 DIBF (Input Buffer Full)

The TVM2464 uses an 8255 PPI device to act as an I/O buffer to the controller. Instruction and data information is always written to this buffer. The I/O buffer can hold information for the next instruction while the controller processes the current instruction so there is one level of pipe lining of data to the module. This is an important point to remember when interfacing the module. DIBF provides a signal to indicate if the TVM2464 can accept data. A logic 0 indicates that the TVM2464 is ready for the next instruction or data. A logic 1 indicates that the input buffer to the controller is full and the module is busy. Therefore the DIBF signal

does not indicate that the controller has completed execution of the current instruction, it only indicates that the input buffer is empty and that new information may be written. DIBF can also be read through the status register.

1.4.1.4 DOBF/ (Output Buffer Full)

A logic 0 on DOBF/ indicates that the controller has placed data into the I/O buffer to be read by the external micro-processor. As soon as the data is read, the DOBF/ line returns to a logic 1. DOBF/ can also be read through the status register. DOBF/ will stay set until the data is read from the TVM2464 or until the reset line is pulled low. i.e. if a user program requests information, but never takes this information, the DOBF/ flag will remain set. When the next request for information is made, the main program will detect the DOBF/ flag set and immediately retrieve the previous instruction's information. To avoid this problem, be sure to always read the output register when the DOBF/ flag is set .

1.4.1.5 ERROR

Because of the intelligent nature of the TVM2464, some instructions may cause an error if they can not be executed. One cause may be providing the instruction with data that produces an internal error. An example would be trying to place a button on top of another button. The second button would not be placed and the ERROR flag would be set to a logic 1. It will stay set until the next instruction is executed. The ERROR flag has different meanings depending on the instruction being executed. Please refer to Section 2 for the specific ERROR flag meaning for each instruction. ERROR can also be read through the status register.

1.4.1.6 KEYPRESS

The KEYPRESS flag indicates that a button has been pressed and that the external micro-processor may now read the "Button Code". This code indicates which button has been pressed. The "Button Code" is assigned to a button when it is placed on the display. Please refer to section 3.6.1 for more information. KEYPRESS can also be read through the status register.

1.4.1.7 DA0, DA1

DA0 and DA1 are used to address the different module registers. Please refer to Table 1.3 for more information.

1.4.1.8 D0 - D7

D0 through D7 forms the 8 bit bi-directional data buss to the module.

1.4.2 Status Register

In some applications you may prefer to read a status register rather than use "hardware hand

shaking". A status register is provided on the TVM2464 which stores the "hand shaking" signals previously described. The status register can be read at address 3 at any time without affecting the writing or reading of instructions or data. See Table 1.4 below for more information.

The status register has a signal, CPUBUSY, that can be used to indicate whether or not the CPU is executing the current instruction. This flag is set high when an instruction is loaded into the instruction register (address 0) and stays set high until the instruction is completed.

Bit	Function
0(LSB)	DIBF
1	DOBF/
2	ERROR
3	KEYPRESS
4	CPUBUSY
5	0
6	0
7(MSB)	0

Table 1.4

1.4.3 Interface Examples

Shown below are interface examples for the TVM2464.

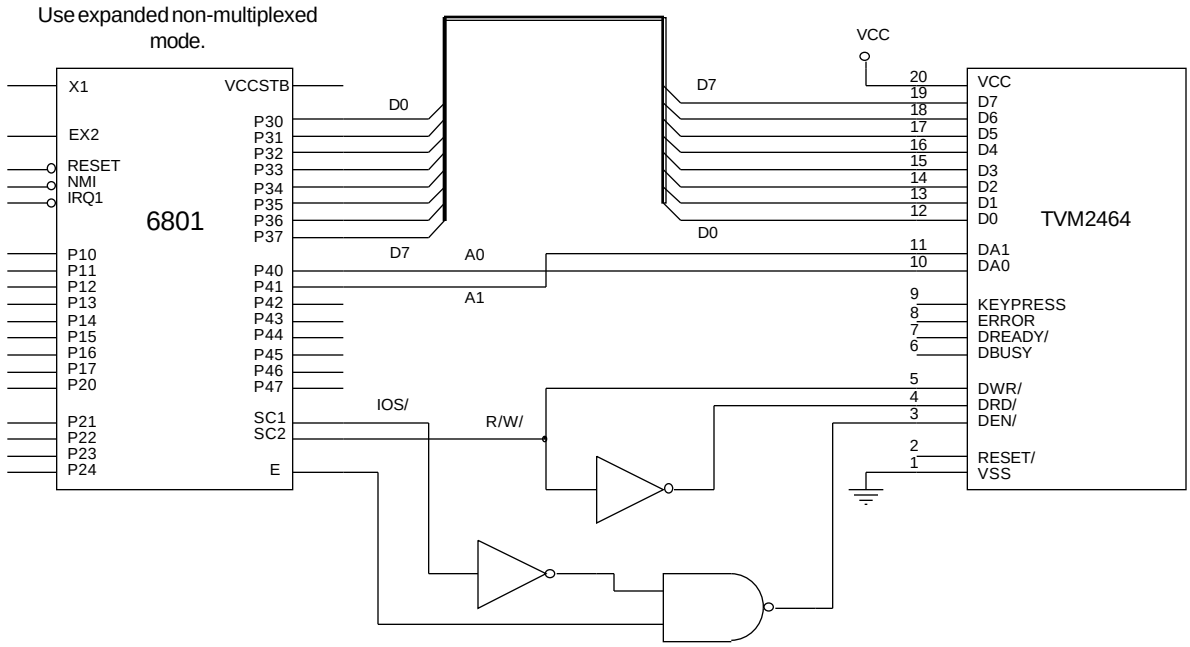


Fig. 1.4

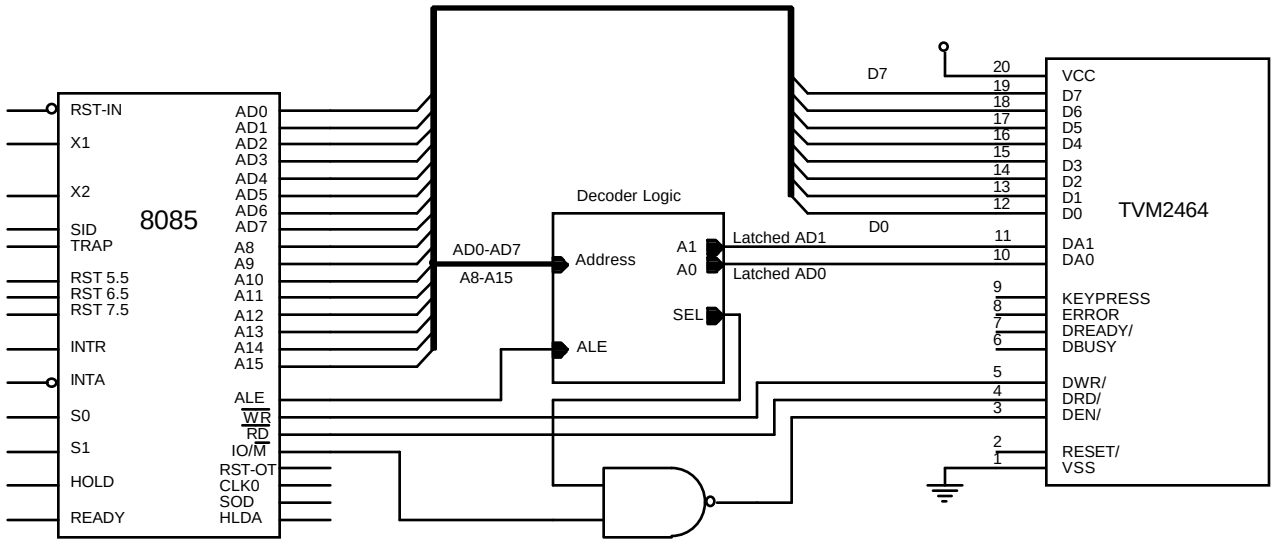


Fig. 1.5

1.4.4 Auxiliary Keyboard Connector

For applications using the TVM2464 without the touch panel or with additional external keys, a separate keyboard interface connector is provided. Any keyboard matrix of up to 3 by 10 keys is compatible with the TVM2464. Fig. 1.6 below shows a schematic for an external keyboard.

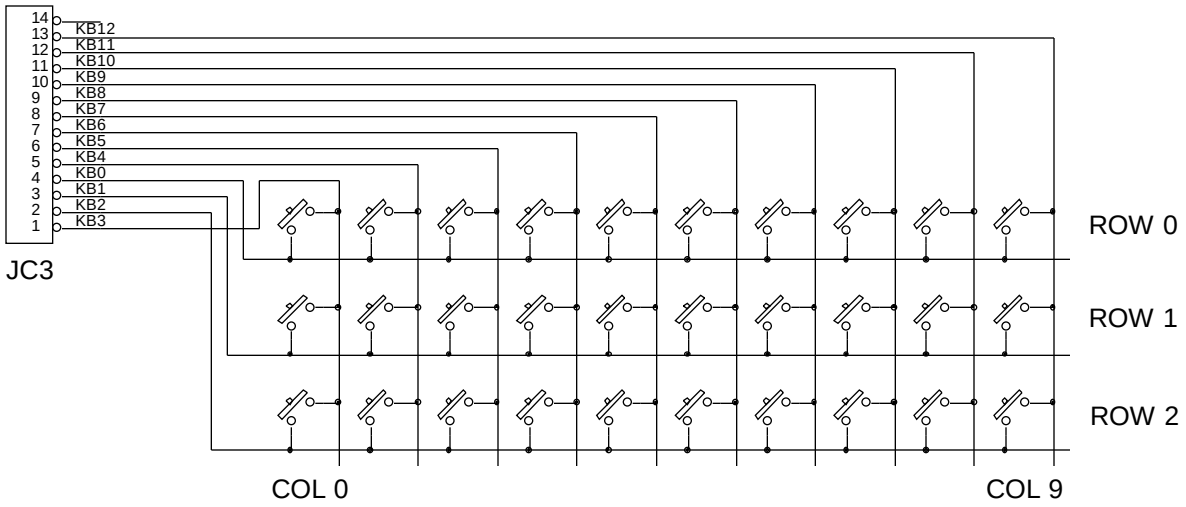


Fig. 1.6

1.4.5 EL Back-light

An Electroluminescent back-light is a standard part of the TVM2464. An on board voltage inverter is included to provide the necessary power for the EL panel. A circuit is also provided to allow the controller to turn the panel ON and OFF using a TVM2464 command. To use the EL panel in the standard configuration, short pins 2-3 on header JO5. This provides the on board 5V power supply to the inverter module. In applications desiring a separate power supply for the inverter, remove the jumper from JO5 and connect ground to pin 1 and inverter power to pin 2. The maximum inverter input voltage is 5.5V.

1.4.6 LED Backlight

The LED backlight requires a 12 Volt power supply capable of providing at least 350 mA of current. The power connection for the LED backlight is located next to the touch panel auxiliary connector JC3 (See figure 1.1). **For proper operation be sure to connect the ground side of the 12 Volt power supply to the ground side of your 5 Volt power supply.** This connection is not made within the TVM2464LTC in order to avoid ground current paths which could impair operation when using long cables to connect control signals to connector JF1. Also note that EL transformer is removed and that the control signal for the LED backlight is passed via a jumper between header JO5 and the long header pin nearest JO5 coming from the LED backlight board. The second long pin on the LED backlight board is connected to the Bezel for grounding purposes. It does not connect to the LED power ground connection.

The LED backlight current can be adjusted via a trim pot located on the upper right of the LED board (back view, See figure 1.1). The current is factory set at a nominal 350 mA. If higher current is desired, forced air cooling may be required depending on orientation and enclosure design.

1.5 Electrical Specifications

Absolute Maximum Ratings:

Storage Temperature	-20C to +60C
Operating Temperature	0C to +40C
V_{CC} with respect to Ground	-0.5V to 7.0V
All other pins	(VCC + 0.5)V to (GND - 0.5)V

DC Electrical Characteristics:

V_{CC} = 5.0V +/- 10% unless otherwise specified.
T_A = 25 C Unless otherwise specified.

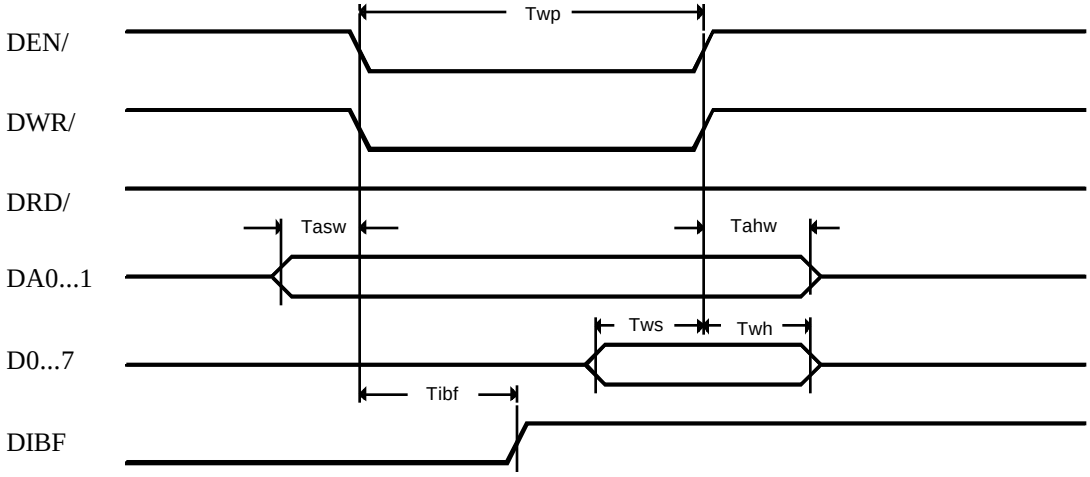
Parameter	Min	Typ	Max	Units	Conditions
Power Supply Voltage	4.5		5.5	Volts	
Supply Current					
No EL or LED backlight		50	75	mA	LED current is
With EL backlight		95	125	mA	adjustable.
I_{ol} Input Low Voltage					
D7-D0,DA1, DA0,DWR/,DRD/ DEN/	-0.5		0.8	Volts	
I_{oh} Input Hi Voltage					
D7-D0,DA1, DA0,DWR/,DRD/ DEN/	2.0		VCC	Volts	
V_{ol} Output Low Voltage					
D7-D0,DBUSY DREADY/,ERROR KEYPRESS			.4	Volts	@I_{ol} = 2.5mA
V_{oh} Output Hi Voltage					
D7-D0,DBUSY	3.0			Volts	@I_{oh} = -2.5mA
DREADY/,ERROR KEYPRESS	Vcc-.4			Volts	@I_{oh} = -100uA
Output Floating Leakage					
D0-D7			+/-10	uA	V_{in} = Vcc or 0V
RESET/ Active Low			.5	Volts	22K Pull Up
RESET/ Inactive Hi	4.5			Volts	
Key Board Strobe					
KB0-KB2			0.5	Volts	@I_{ol} = 50mA
Key Board Column Input Low					
KB3-KB12			0.8	Volts	100K Pull Up
Key Board Column Input Hi					
KB3-KB12	4.0		Open	Volts	100K Pull Up
External EL Supply Voltage	3.0		5.5	Volts	
External EL Supply Current			65	mA	

AC Electrical Characteristics:

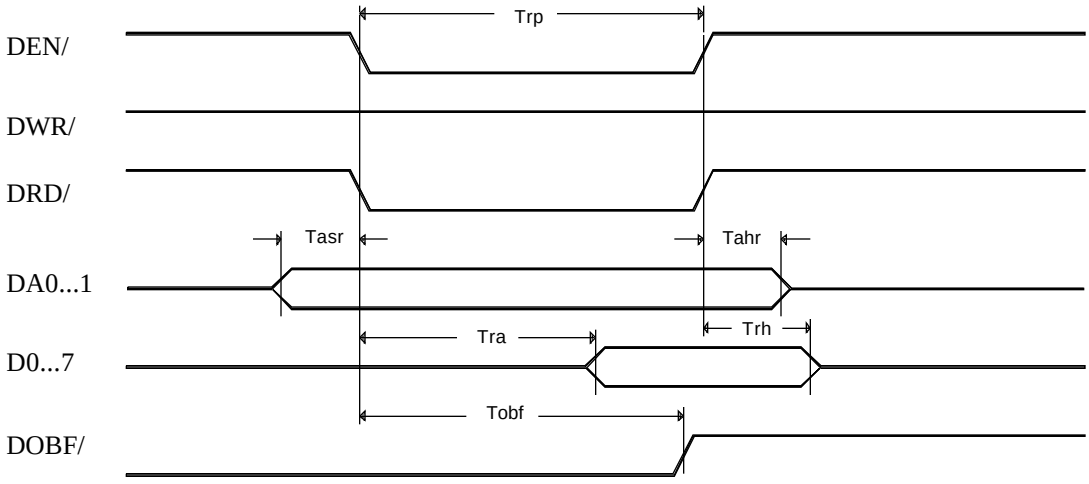
V_{CC} = 5.0V +-10% unless otherwise specified
T_A = 25 C Unless otherwise specified

Symbol	Parameter	Min	Max	Units
T_{wp}	Write Pulse Width	100		nS
T_{asw}	Address Setup Write	0		nS
T_{ahw}	Address Hold Write	25		nS
T_{ws}	Write Data Setup	0		nS
T_{wh}	Write Data Hold	175		nS
T_{ibf}	Write to IBF High		100	nS
T_{rp}	Read Pulse Width	100		nS
T_{asr}	Address Setup Read	0		nS
T_{ahr}	Address Hold Read	25		nS
T_{ra}	Read Data Access		125	nS
T_{rh}	Read End to Data Tri-state		85	nS
T_{obf}	Read to OBF High		100	nS

Timing Waveforms:



Write Cycle Timing



Read Cycle Timing

2 Instruction Set

This section describes the instruction set for the TVM2464 module. There are a total of 41 instructions divided into 8 different types. Each instruction is described in detail showing the op-code, required data and range of data, error handling if applicable and options.

2.1 Symbol Notation and Instruction Definitions

2.1.1 Symbol Notation

Table 2.1 defines the symbols used when describing the TVM2464 instruction set.

SizeX	Value 1 to 8	Font Size X
SizeY	Value 1 to 64	Font Size Y
Offset	Value 0 to 255	Ascii Offset
Descenders	Value 0 to SizeY	Font # Descenders
Pitch	Value 0 to 239	Font Pitch X
Height	Value 0 to 63	Scroll Height Y
Data	Value 0 to 255	General Data
String...	Value 0 to 255	Arbitrary number of data elements
Xpos	Value 0 to 239	X Cord position
Ypos	Value 0 to 63	Y Cord Position
AX	Value 0 to 239	CordA X
AY	Value 0 to 63	CordA Y
BX	Value 0 to 239	CordB X
BY	Value 0 to 63	CordB Y
Length	Value -127 to +127	Length of Hor. or Vert. Line
BLength	Value 1 to 10	Phantom Button Length
a,b,c,d	Bit attributes	Described per instruction
xxx...	Instruction imbedded data	
[RData]	Returned data	
ERROR	Error flag	
KeyCode	Value 0 to 255	Button code
Position	Coded Button Position	

Table 2.1

2.1.2 Instruction Definitions

The following instruction definitions are organized as shown below

A. *Instruction Name*

B. *Coding*

This section describes the instruction coding. Coding is listed as follows:

76543210,<>,<>,<>,<...>

7-0 lists the binary data encoding for the instruction. <> refers to optional or instruction dependent data. <...> refers to string data of arbitrary length.

C. *Coding Options*

This section describes coding options for the instruction. This may include instruction attributes or data imbedded into the instruction.

D. *Description*

This section describes the function performed in detail.

2.2 Font Instructions

Name:	Select Font
Coding:	00000xxx
Coding Options:	xxx = binary value from 0 to 4
Description:	Selects a new font for all subsequent text input. Font 0 and 1 are hard coded into the controller software. If font 2-4 are selected and they have not been previously loaded the ERROR flag is set.

Name:	Down Load Font
Coding:	00001xxx,SizeX,SizeY,Offset,Descenders, String...
Coding Options:	xxx = binary value from 2 to 4
Description:	This instruction down loads a new font into the specified font memory. If the specified font number is not 2, 3 or 4 the ERROR flag will be set and the instruction will be ignored. SizeX is the font size in the X direction. Size Y is the font size in the Y direction. Offset is subtracted from the Ascii code to offset the value when indexing into the font memory. Please refer to section 3.4 for more information. Descenders indicates the number of pixels a character can extend below the line. In should be noted that SizeY includes the Descenders. String... is the binary font information. It is of arbitrary length. A maximum of 8192 bytes of font information can be loaded into a font memory. Data after this will be ignored.

Name:	Set Font Attributes
Coding:	000110ab
Coding Options:	ab = 00, Default, force font into display ab = 01, OR Font data into display ab = 10, XOR Font information into display
Description:	This instruction determines how the font will be combined with the data that is already in the display area. For example, if the attribute XOR is selected, the font data will reverse whatever data may already be on the display. All subsequent text input will be acted on by the font attribute.

2.3 Cursor Positioning

Name:	SetXY
Coding:	0010000,Xpos,Ypos
Coding Options:	None
Description:	SetXY sets the positions of the upper left corner of the cursor. If the underline cursor is used, the cursor is still positioned such that the font placed will be positioned at the upper left corner of the SetXY position. The cursor can not be positioned outside the text window. If cursor coordinates are provided that place it outside the text window, the cursor is placed as close as possible to the provided location but still in the text window.

Name:	ReadXY
Coding:	00100001,[Xpos],[Ypos]
Coding Options:	None
Description:	ReadXY returns the current XY cursor position.

Name:	Cursor Up
Coding:	00100010
Coding Options:	None
Description:	This instruction moves the cursor up Scroll amount. The cursor will not move if moving will cause it to go outside the text window.

Name:	Cursor Down
Coding:	00100011
Coding Options:	None
Description:	This instruction moves the cursor down by the Scroll amount. The text window will scroll up if the cursor is already at the bottom of the window. In this case text data at the top of the window will be lost.

Name:	Cursor Left
Coding:	00100100
Coding Options:	None
Description:	This instruction moves the cursor left by the SizeY amount. If pitch data is entered, it moves the cursor by the pitch amount. The cursor can not move past the left edge of the text window.

Name:	Cursor Right
Coding:	00100101
Coding Options:	None
Description:	This instruction moves the cursor right by the SizeY amount. If pitch data is entered, it moves the cursor by the Pitch amount. The cursor can not move past the right edge of the text window.

Name:	SetX
Coding:	00100110,Xpos
Coding Options:	None
Description:	SetX positions the cursor to the Xpos. The cursor Y position is not changed. If the Xpos would cause the cursor to move outside the text window, the cursor is positioned as close as possible to the Xpos.

Name:	SetY
Coding:	00100111,Ypos
Coding Options:	None
Description:	SetY positions the cursor to the Ypos. The cursor X position is not changed. If the Ypos would cause the cursor to move outside the text window, the cursor is positioned as close as possible to the Ypos.

Name:	Set Cursor Attributes
Coding:	00010abc
Coding Options:	a b c Attribute - - 0 Cursor OFF 0 0 1 Block Cursor ON 0 1 1 Block Cursor Flash 1 0 1 Underline Cursor ON 1 1 1 Underline Cursor Flash
Description:	This instruction selects the cursor attribute.

2.4 Text Configuration

Name:	Set Text Window
Coding:	00101000,AX,AY,BX,BY
Coding Options:	None
Description:	This instruction defines the area in which text can be placed. It also defines the scroll window if scrolling becomes necessary due to cursor movement or auto line wrapping of inputed text.

Name:	Set Pitch
Coding:	0 0 1 0 1 0 1 0,Pitch
Coding Options:	None
Description:	This instruction sets the pitch for text input. The pitch is the size of the X step that the cursor makes for each character. If Pitch is not defined or if Pitch is set to 0, the cursor X step defaults to SizeX.

Name:	Set Height
Coding:	0 0 1 0 1 0 1 1,Height
Coding Options:	None
Description:	This instruction sets the distance the cursor can move in the Y direction. It also defines the amount the text window is scrolled up if required. If Height is not defined or if Height is set to 0, the cursor can not move in the Y direction using Cursor Up or Cursor Down. Also, the text window can not scroll. Default height is 0 upon reset.

2.5 Text Input

Name:	Input String
Coding:	0 0 1 0 1 1 0 0,String...
Coding Options:	None
Description:	This instruction places text on the display at the cursor location. The cursor is automatically indexed to the right by the SizeX or Pitch amount. If the cursor hits the right side of the text window, the cursor is moved down by the Height amount being positioned at the left side of the text window. If the cursor is at the bottom of the text window, the window will scroll if Height is set. String... data (text) can be any arbitrary length. This text input instruction is terminated at the start of the next instruction (data written to address 0).

2.6 Graphics Input

Name:	Draw Box
Coding:	0 1 0 0 T T T F,AX,AY,BX,BY
Coding Options:	TTT = 0 - 7, Box thickness in pixels F = 0, Clear pixels inside box F = 1, Leave pixels inside box same
Description:	This instruction draws a Box from AX/Y to BX/Y. The box is TTT thick. The thickness extends inside the box coordinates. If TTT is zero, no outline for the Box is drawn.

Name:	Draw Block
Coding:	0 1 1 0 0 0 T T,AX,AY,BX,BY
Coding Options:	T T — Attribute 0 0 — Set pixels OFF 0 1 — Set pixels ON 1 0 — XOR pixels 11 — Checker board pattern

Description: This instruction draws a Block from AX/Y to BX/Y.

Name: Draw Horiz

Coding: 0 1 1 0 0 1 T T,Xpos,Ypos,Length

Coding Options:

T T	Attribute
0 0	Set pixels OFF
0 1	Set pixels ON
1 0	XOR pixels

Description: This instruction draws a horizontal line starting at Xpos/Ypos being Length long. If Length is positive, the line is drawn to the right of Xpos, Ypos. If Length is negative, the line is drawn to the left of Xpos, Ypos.

Name: Draw Vert

Coding: 0 1 1 0 1 0 T T,Xpos,Ypos,Length

Coding Options:

T T	Attribute
0 0	Set pixels OFF
0 1	Set pixels ON
1 0	XOR pixels

Description: This instruction draws a vertical line starting at Xpos/Ypos being Length long. If Length is positive, the line is drwn up from Xpos, Ypos. If Length is negative, the line is drawn down from Xpos, Ypos.

Name: Draw Vector

Coding: 0 1 1 0 1 1 T T,AX,AY,BX,BY

Coding Options:

T T	Attribute
0 0	Set pixels OFF
0 1	Set pixels ON
1 0	XOR pixels

Description: This instruction draws a vector line from AX/Y to BX/Y.

Name: Set Pixel

Coding: 0 1 1 1 0 0 T T,Xpos,Ypos

Coding Options:

T T	Attribute
0 0	Set pixel OFF
0 1	Set pixel ON
1 0	XOR pixel

Description: This instruction sets a single pixel at Xpos, Ypos.

2.7 Button Input

Name: Place Button

Coding: 0 0 1 1 0 0 0 0,KeyCode,Position

Coding Options: None

Description: This instruction places a button at the specified location. It places the string data loaded with the "Load Button Buffer" instruction into the button. It centers the text, sizes the button and then places it. The position is coded as follows:

0 0 R R C C C C

Where RR = Row number 0 - 2
Where CCCC = Column number 0 - 9

The touch panel cells over the display button area are automatically activated. If the button will not fit due to its length or another button being in the way the button will not be placed and the ERROR flag will be set.

Name: Load Button Buffer

Coding: 0 0 1 1 0 0 0 1,String...

Coding Options: None

Description: Text data String... is loaded into the internal buffer memory. Subsequent button placements will use this string data for the text inside the button. This instruction is terminated when another command is issued

---

Name: Get Button Size

Coding: 0 0 1 1 0 0 1 0,KeyCode,[RData]

Coding Options: None

Description: This instruction returns the size of the specified button. The button is specified by KeyCode. [RData] returned is the length of the button in Button Cells. If the KeyCode button does not exist the ERROR flag is set and [RData] returns the size of the button based on data in the Button Buffer. This allows the user to predict the size of a button before placement for display formatting considerations.

Name:	Place Phantom Button
Coding:	0 0 1 1 0 0 1 1,KeyCode,Position,BLength
Coding Options:	None
Description:	This instruction activates the touch panel starting at Position, BLength long. BLength specifies the width of the button in touch panel cells. BLength can range from 1 to 10. Position is defined the same as in "Place Button". The display is not affected. If the button will not fit due to BLength or another button, the phantom button will not be placed and the ERROR flag will be set.

Name:	Delete Button
Coding:	0 0 1 1 0 1 0 0,KeyCode
Coding Options:	None
Description:	The button specified by KeyCode will be removed. The display area beneath the button will be erased and the touch panel will be de-activated. The space left can be re-used for another button. If a button with KeyCode does not exist the ERROR flag will be set.

Name:	Delete All Buttons
Coding:	0 0 1 1 0 1 0 1
Coding Options:	None
Description:	This instruction deletes all buttons.

Name:	Read KeyCode
Coding:	0 0 1 1 0 1 1 0,[RData]
Coding Options:	None
Description:	This instruction returns the KeyCode for the button last pressed (or currently pressed). [RData] contains the key code.

Name:	Set Button Attributes
Coding:	0 0 1 1 1 a b c
Coding Options:	Auto Key Latch: a=0 OFF, a=1 ON Auto Repeat: b=0 OFF, b=1 ON Key Click: c=0 OFF, c=1 ON
Description:	This instruction sets the button attribute for all subsequently

placed buttons.

2.8 Display Control

Name:	Blank Display
Coding:	1 0 0 0 0 0 0 0
Coding Options:	None
Description:	This instruction erases the display. The cursor position is not affected.

Name:	Clear Display
Coding:	1 0 0 0 0 1 1 1
Coding Options:	None
Description:	This instruction is similar to "Blank Display" except it actually re-formats the display RAM and cursor RAM to guarantee that all locations on the LCD display are re-written. The cursor position is not affected.

Name:	Refresh
Coding:	1 0 0 0 0 0 0 1
Coding Options:	None
Description:	All areas changed since the last refresh are written to the LCD display controller.

Name:	Set AutoRefresh
Coding:	1 0 0 0 1 0 a b
Coding Options:	Instruction Ref: a=0 OFF, a=1 ON String Ref: b=0 OFF, b=1 ON
Description:	This instruction causes an automatic refresh. Option "a" determines if a refresh will occur after each instruction. Option "b" determines if a refresh will occur after String data effecting the display (i.e. Input String) is written. The default for AutoRefresh is 00 (OFF,OFF).

Name:	Dump Display RAM
Coding:	1 0 0 0 0 1 0 0,[RData...]
Coding Options:	None
Description:	Display RAM is dumped out of the module. A total of 1920 bytes are transferred. This instruction will be terminated if another instruction is issued before all data is read. The cursor position or type will have no affect on the data read.

Name:	Load Display RAM
Coding:	1 0 0 0 0 1 0 1,String...
Coding Options:	None
Description:	String data is loaded into display RAM. A total of 1920 bytes of data can be written. Any data written after this is ignored. This instruction will be terminated if another instruction is issued before all data is written however the data written up to that point is placed into display memory.

Name:	Move Block Vert
Coding:	01110100,AX,AY,BX,BY,Distance
Coding Options:	None
Description:	Moves a block of display data defined by AX/Y to BX/Y by the Distance amount. If Distance is positive, the display data is moved up. If Distance is negative, the display data is moved down. Distance is limited to +/- 64 pixels.

Name:	Move Block Horiz
Coding:	01110101,AX,AY,BX,BY,Distance
Coding Options:	None
Description:	Moves a block of display data defined by AX/Y to BX/Y by the Distance amount. If Distance is positive, the display data is moved right. If Distance is negative, the display data is moved left. Distance is limited to +/- 127 pixels.

2.9 System Instructions

Name:	Soft Reset
Coding:	1 1 1 1 1 1 1 0
Coding Options:	None
Description:	This instruction causes a soft reset of the controller. All setup variables, the display and attributes are re-initialized.

Name:	Set Contrast
Coding:	1 1 1 1 0 0 1 1,Data
Coding Options:	None
Description:	The display contrast is set to Data. Data can range from 0 to 31 with 0 being light and 31 being dark. The controller initializes with contrast at 16.

Name:	Set EL or LED
Coding:	1 1 1 1 0 1 0 a
Coding Options:	a=0, Backlight Off a=1, Backlight On
Description:	This instruction turns the backlight on and off.

Name:	NOP
Coding:	1 1 1 1 1 1 1 1
Coding Options:	None
Description:	No operation performed.

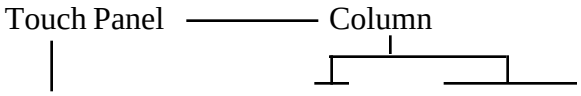
Name:	Set Beeper
Coding:	1 1 1 1 0 0 0 a
Coding Options:	a=0 Beeper OFF a=1 Beeper ON
Description:	This instruction turns the sound transducer on and off.

Name: Read Key Matrix

Coding: 1 1 1 1 0 0 1 0 [RData...]

Coding Options: None

Description: This instruction returns the previously stored, current value of the touch panel matrix. There are 6 bytes returned (noted by [X] in the table below). Two bytes are used for each row of the touch panel. The data is formatted as follows:



Row 0	[1]	98	[2]	76543210
Row 1	[3]	98	[4]	76543210
Row 2	[5]	98	[6]	76543210

For reference, the physical touch panel matrix is organized as follows:

Row 0	0 1 2 3 4 5 6 7 8 9
Row 1	0 1 2 3 4 5 6 7 8 9
Row 2	0 1 2 3 4 5 6 7 8 9

2.10 Instruction Set Summary

The following is a summary listing of all the instructions.

Font Selection	
Select Font	00000xxx
Down Load Font	00001xxx,SizeX,SizeY,Offset,Descenders String...
Set Font Attributes	000110ab
Cursor Positioning	
SetXY	00100000,Xpos,Ypos
ReadXY	00100001,[Xpos],[Ypos]
Cursor Up	00100010
Cursor Down	00100011
Cursor Left	00100100
Cursor Right	00100101
SetX	00100110,Xpos
SetY	00100111,Ypos
Set Cursor Atrib.	00010abc
Text Configuration	
Set Text Window	00101000,AX,AY,BX,BY
Set Pitch	00101010,Pitch
Set Height	00101011,Height
Text Input	
Input String	00101100,String...
Graphics Input	
Draw Box	0100TTTF,AX,AY,BX,BY
Draw Block	011000TT,AX,AY,BX,BY
Draw Horiz	011001TT,Xpos,Ypos,Length
Draw Vert	011010TT,Xpos,Ypos,Length
Draw Vector	011011TT,AX,AY,BX,BX
Set Pixel	011100TT,Xpos,Ypos
Button Input	
Place Button	00110000,KeyCode,Position
Load Button Buffer	00110001,String...
Get Button Size	00110010,KeyCode,[RData]
Place Phantom Butt.	00110011,KeyCode,Position,BLength
Delete Button	00110100,KeyCode
Delete All Buttons	00110101
Read KeyCode	00110110,[RData]
Set Button Attrib.	00111abc
Display Control	
Blank Display	10000000
Clear Display	10000111
Refresh	10000001
Set Auto Refresh	100010ab
Dump Display RAM	10000100,[RData...]
Load Display RAM	10000101,String...
Move Block Vert	01110100,AX,AY,BX,BY,Distance
Move Block Horiz	01110101,AX,AY,BX,BY,Distance
System Instructions	
Soft Reset	11111110
Set Contrast	11110011,Data
Set EL or LED	1111010a
NOP	11111111
Set Beeper	1111000a
Read Key Matrix	11110010,[RData]

3 Using the TVM2464

This section will describe in detail how to use the TVM2464.

3.1 Display Control

The TVM2464 controller has a built in buffer memory that corresponds to the LCD display. In addition, the LCD display controller has a separate pixel memory that is used to refresh the LCD display. Whenever any information is written to the display, it is actually written to the display buffer RAM. This RAM is tied directly to the TVM2464 controller to allow fast access. All manipulations of the display data are done in conjunction with this RAM. This allows the controller to quickly manipulate the pixels, write font data to the buffer, draw boxes, lines, vectors, etc.. After data is written to the display buffer RAM, it is then written to the LCD display controller. The instruction to accomplish this is called *Refresh*.

3.1.1 Auto Refresh

If desired, the TVM2464 can be configured to do an *Auto Refresh* which automatically refreshes the screen after each instruction or string data is received. However, auto refresh should be used with caution. As an example, if many instructions are required to format the screen, auto refresh would begin to slow down the controller. In these cases it would be better to disable auto refresh, write all the information to the display buffer and then refresh the screen all at once.

3.1.2 Erasing the Display

For most applications the *Blank Display* instruction is used to remove data from the LCD display. After erasing, new data can be written and then a *Refresh* instruction can be given.

In some cases, the display buffer RAM may need to be reset. In this case the *Clear Display* instruction should be used. After this instruction is executed, a *Refresh* must be given to guarantee that all of the LCD display pixels are turned off. Information written to the LCD display after the *Clear Display* instruction is given but before a *Refresh* may not be properly written to the LCD display.

3.1.3 Dumping and Loading the Display Buffer

Some applications may require direct access to the display buffer. *Dump Display RAM* allows reading of the pixel data by the external processor for manipulation. The pixel data can then be re-loaded back into the display buffer using the instruction *Load Display RAM*. Both of these instructions will transfer 1920 bytes of information between the TVM2464 and the external micro processor. Figure 3.1 shows a map of how the data is formatted.

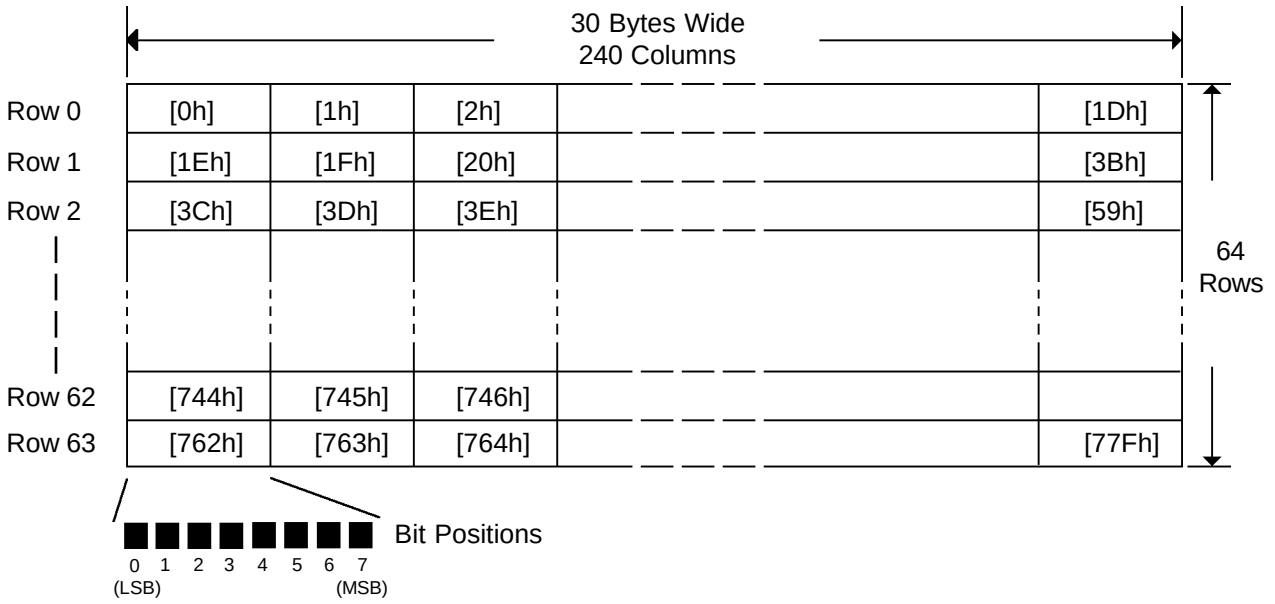


Fig. 3.1

Data is formatted from upper left to lower right. Each byte represents 8 pixels. The LSB bit is the left most pixel and the MSB is the right most pixel. Note that the RAM bits are "turned around" (i.e. other graphics systems have MSB left, LSB right). Data is scanned from left to right, top to bottom. The first 30 bytes are row 1, next 30 are row 2 and so on.

64 Rows X 30 Bytes = 1920 Bytes

30 Bytes x 8 Bits/byte = 240 bits wide

3.2 Text Window and Display

The TVM2464 software creates a text window which is used to define an area of the display for text data. Cursor movements are constrained to be within the text window. Upon initializing the TVM2464, using a hardware reset or *Soft Reset* instruction, the text window is set to the maximum size of the display allowing the cursor and text to be placed anywhere on the display.

Some applications may require a graphics area with a separate text area which can scroll. In this case, the *Set Text Window* instruction is used to define the text and scrolling area. All other areas on the LCD display will not be affected by the text input, cursor movement or scrolling.

It should be noted that ANY pixel data contained in the text window will also scroll and that the text window does not constrain the placement of buttons (even though buttons have text characters inside them).

3.3 Cursor Positioning

The cursor can be positioned using the cursor positioning instructions. The cursor cannot be positioned outside the text window. If cursor positioning outside of the text window is attempted, the cursor will instead be placed as close as possible to the given coordinates while remaining within the text window. The cursor may be stepped up, down, left and right using the cursor positioning instructions.

3.3.1 Cursor Coordinates

The cursor, regardless of whether an under bar or a block, is always positioned in reference to the upper left corner of the character to be placed. As an example, if the cursor was a block, the cursor coordinates specify the upper left position of the block with the character being placed inside that block. If the cursor was an under bar, the cursor is placed below the character such that the same upper left coordinate would cause the character to be placed in exactly the same location as if the cursor was a block. i.e. any given set of co-ordinates places the character in the exact same position regardless of the cursor type.

3.3.2 Descender Affects on Cursor Placement

Fonts which use descenders cause the underline cursor to be placed such that the upper left corner of the cursor begins at the cursor Ycord - SizeY + Descenders. This positions the under bar cursor at the exact place in the line at which the descenders start. If a block cursor is used, descenders have no affect on the cursor positioning. The block cursor would cover the entire font area including the descender lines.

3.3.3 Pitch and Height

The Pitch and Height parameters determine how the cursor steps in the display.

Pitch is the measure in pixels from the start of one character to the start of the next character. If Pitch is not used (set to 0), the controller software will automatically default to the value of SizeX. SizeX is the font width in the X direction. The controller initializes Pitch to 0.

Height is used to determine the size of the vertical step the cursor takes when using *Cursor Up/ Down* instructions. Height also determines how many pixel rows the text window will move with each scroll instruction. If Height is set to 0 the text window can not scroll and the cursor can not be moved vertically. The cursor may still be positioned using *SetX/Y* instructions. Height is initialized to 0.

3.3.4 Cursor Attributes

The cursor can be programmed to appear several different ways. It can be turned ON or OFF, be a block or an underline and can flash or be on continuously. These characteristics are set using the *Set Cursor Attribute* instruction. Figure 3.2 below illustrates the cursor shape and position for both block and underline.

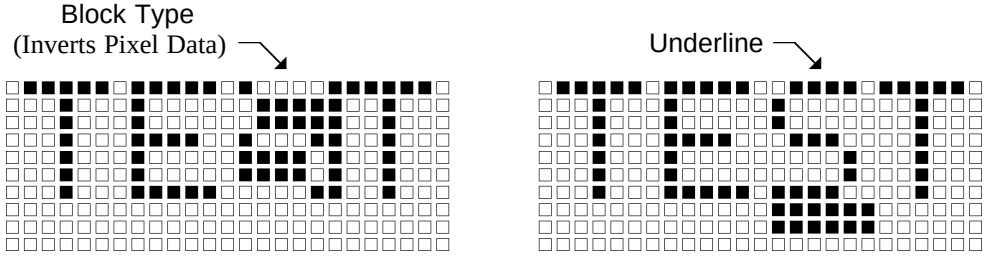


Fig. 3.2

It should be noted that even if the cursor is turned off, it will still operate as if it was on (e.g. it can not be outside the text window). Internally the cursor is always present to mark where the next text data will be located.

3.4 Text Input

The *Input String* instruction is used to input text to the display starting at the current cursor location. The cursor is automatically stepped by the Pitch/Height amount as each character is placed and the text area scrolled if required.

Text is always placed using the currently selected font and font attributes.

3.5 Fonts

The TVM2464 can have up to 5 different fonts resident simultaneously. Fonts, numbered 0 and 1, are hard coded into the controller memory. Fonts, 2, 3 and 4, can be loaded at run time. There are no restrictions in regards to mixing fonts. Text of any font can be placed within the Text Window by issuing the *Select Font* instruction and then writing text data. All fonts can be displayed at the same time.

If font 2, 3 or 4 are selected and the corresponding font has not been loaded into memory, an ERROR will result and the font selection will not occur.

Font 0 is a bold easy to read 7 by 11 font. It has a SizeX of 8 so that a space 1 pixel wide is automatically left between each character. Height should be set at 12 for proper scrolling and cursor movement. It has two pixel descenders.

Font 1 is a smaller 5 by 7 font. It has a SizeX of 6 so that a space 1 pixel wide is automatically left between each character. Height should be set at 8 for proper scrolling and cursor movement. It has no descenders.

3.5.1 Down-loading Fonts

Fonts 2, 3 and 4 are all down-loaded. Each block, containing one font, of font memory can accommodate up to 8192 bytes of information. Each particular character code occupies an amount equal to SizeY of font memory. In other words, each character of a 5 by 7 font will take 7 bytes of memory.

The maximum font width is 8 pixels wide. If a larger font is required it must be divided into strips, each strip being 8 pixels wide. The characters are then displayed by grouping strips together.

When the font is down-loaded, the offset data must be specified. This offset is subtracted from the Ascii code used to represent the character. By using this technique, the font memory does not waste space taken by characters that are not required.

For example, a custom font is to contain the characters 0 to 9, just numbers. In this case, since the Ascii code for zero is 30h, 30h becomes the offset. When the data 30h is written using the *String Input* instruction, an offset of 30h is subtracted from the data causing an index of 0 into the font memory. Thus, the controller places the font data for character “0” into the display buffer.

It should be noted that the specified offset determines the effective start of the font memory and all subsequent characters must be contiguous. In other words if several characters after “9” were not required but even later characters were required, the characters not required would need to be loaded with dummy data to allow proper indexing of the font memory.

3.5.2 Font Attributes

Font attributes determine how the font data is to be combined with other data already in the display buffer. (Refer to section 2.2 for information on the *Set Font Attributes* instruction.)

The default setting causes the font data to be forced into the display buffer. Pixels which the font defines as on will be forced on. Likewise, pixels the font defines as off will be forced off. It should be noted that the size of the area forced is SizeX by SizeY. In other words, if the pitch is set larger than the SizeX, a space will be left unaffected between each character.

If the font attribute is set to OR, the font data will be OR'ed with the current data in the display buffer. This is useful when the font needs to be placed over other graphics data.

If the font attribute is set to XOR, the font data will be XOR'ed with the current data in the display buffer. This is useful for doing a "Reverse Video" display.

3.6 Graphics Input

A graphics command can place information anywhere in the display buffer. All graphics coordinates range across the entire display. If a co-ordinate is given that would cause data to be placed outside the display buffer, it is automatically limited to force it onto the display.

3.7 Touch Panel Commands

The Touch Panel Commands are used to place or remove buttons from the touch panel and LCD display. Each button (or phantom button) has a corresponding key code when it is placed. The KeyCode is given in the placement instruction. From then on, the KeyCode is used to identify which button is pressed, which button to delete, and so on. Their are no restrictions on the value of the KeyCode. The same KeyCode can also be used for several different buttons if desired. Each button placed is treated as if it was a unique button even if it has the same KeyCode.

3.7.1 Button Attributes

When a button is placed, the button attributes activated*using Set Button Attributes* will determines how it functions. After attributes are set, all subsequent button placements will use these attributes to flavor the buttons operation. Buttons with different attributes may be placed onto the display at the same time. A button has three different attributes assigned to it.

Auto Key Latch: If this attribute is turned ON, it causes the KeyPress flag to stay set until the key code is read. If this attribute is OFF, the KeyPress flag will only stay high while the button is pressed. The KeyCode is always latched and will correspond to the last key pressed (or current key) no matter what.

Auto Repeat: This attribute causes the button to repeat. When the button is first pressed, the KeyPressed flag is set immediately. The processor can then read the KeyCode. If the button is held down longer than 0.7 seconds, the KeyPress flag will be set automatically, 10 times per second.

Key Click: If this attribute is ON, a short beep or “click” will occur each time a key is pressed. This is very useful audio feedback to the user since a touch panel has little if any tactile feedback. If Auto Repeat is set, the “click” will also repeat accordingly.

Please refer to section 2.7 for details on coding for the button attribute.

3.7.2 Placing a Button

A button is placed by using the *Place Button* instruction. The text placed within the button is determined by using the *Load Button Buffer* instruction. The TVM2464 software determines the required size of the button based on the selected font, pitch and the number of characters. A block is placed into the display buffer at the desired button location with the text characters for the button centered in the block using an XOR attribute to reverse the text. The button width is always rounded up to align exactly with a Touch Panel cell and the button is always 16 pixels high, the exact height of a Touch Panel cell. Fig. 3.3 shows a typical display with several buttons.

Fig. 3.3

The touch panel area over the displayed button is automatically activated. When the button is pressed, the button's KeyCode is latched as previously explained.

If a button is placed inside the text window, the button outline block will scroll up when the text window scrolls. The activated area of the touch panel does not change however. Therefore it is recommended that buttons in text areas be deleted before allowing any scrolling activity.

3.7.3 The Affects of Descenders On Button Placement

The Place Button command automatically centers the button text vertically within the button block. When using descenders, the text is centered properly with the descender lines going below the base line of the button text making the text displayed within the button appear to be more accurately centered.

3.7.4 Phantom Buttons

In some cases, it may be necessary to activate the touch panel area without showing a button location on the display. For example, when a graphic is placed beneath the button area. The*Place Phantom Button* instruction is used to place phantom buttons just as you would place a displayed button. A phantom button is just like a normal button in all respects, except that nothing is placed on the LCD display. The button attributes also apply to phantom buttons.

3.7.5 Get Button Size

Because the TVM2464 software automatically determines the size of the button, it may be necessary to find out how big the button is. The *Get Button Size* instruction is used to obtain this information. The instruction returns the length of the button in button cells (not pixels). If the button is not placed, the error flag is set and the value returned is based on the anticipated size of the button dependent on the number of characters in the Button Buffer. This allows external software to determine the size of a button before it is placed. This may be necessary if automatic placement algorithms are used to generate menu type touch screens. The demonstration program *tvm.exe* makes extensive use of automatic placement.

3.7.6 Read KeyCode

When the KeyPress flag goes high, it flags the external micro-processor to read the KeyCode. The *Read KeyCode* instruction is used to retrieve the KeyCode information. After the KeyCode is read, the KeyPress flag will drop. If the KeyCode is read again, the same value will be returned until another button is pressed.

3.7.7 Deleting Buttons

The instructions *Delete Button* and *Delete All Buttons* are used to remove and deactivate buttons. Delete Button will delete the button whose KeyCode is given. *Delete All Buttons* will delete all displayed or phantom buttons. If several buttons have been placed that have the same KeyCode, the *Delete Button* instruction will remove those buttons in the order they were placed one per instruction.

If a phantom button is deleted, the LCD display under the button will not be effected. It is up to the external micro-processor and software to effect any desired change to the LCD display in this case.

3.8 Other System Commands

This section will detail the miscellaneous system commands of the TVM2464.

3.8.1 Soft Reset

In some cases it may be desirable to reset the TVM2464 without having to pull the RESET/ line down. The *Soft Reset* instruction causes the controller software to restart in the same manner as if the reset line was activated. All of the initialization routines in the controller will execute, all of the buttons are cleared and the LCD display is erased. Attributes are reset, the font is set to 0, and the font memory is cleared. Also the contrast is set to 16 and the EL Back light is turned off.

3.8.2 LCD Contrast Control

The TVM2464 has the ability to control the LCD contrast electronically. The *Set Contrast* instruction is used to provide a fine contrast control adjustment of the display. A trimmer pot, R18, is also included on the TVM2464 board to do coarse contrast adjustments. This adjustment is set at the factory and should not need adjustment. If adjustment of this trimmer is necessary, the electronic contrast should be set to 16 (the center of its range) before R18 is set.

3.8.3 Backlight Control

The instruction *Set EL or LED* is used to turn the backlight ON and OFF.

For more information on external back light control, please refer to section 1.4.5.

3.8.4 Beeper

The *Set Beeper* instruction is used to turn the beeper ON and OFF.

3.8.5 Reading the Key Matrix

Some applications may need to read the key matrix directly. The *Read Key Matrix* instruction is used to accomplish this. The touch panel key matrix is automatically scanned 10 times per second and the data from this is stored internally in the controller. This data can be accessed with this instruction. Please refer to section 2.9 for more information on this instruction.

4 Custom Fonts

This section describes the building of custom font data and using the fnt2bin font compiler.

4.1 Overview

The source file which needs to be created to build a custom font is an ASCII file which can be generated using any standard text editor. The font source file must be named *file.fnt* where the name *file* is specified by the user. After the source file is created, it can be converted to a binary format using the program *fnt2bin.exe*. This program produces the files, *file.bin* and *file.asc*. *file.asc* is an ASCII file that may be useful as a source to other assemblers or compiler programs. *File.bin* can be down loaded directly from the applications program *tvi.exe*, provided in the designer kit, into the TVM2464 module.

4.2 Font File Format

Source files for fonts, to be compiled, are very simple. The standard file format used is shown below in Fig 4.1.

```
#SizeX,#SizeY,#Offset
#Descenders
...       <— Font body starts here
...
```

Fig 4.1

Note that the first two lines must be formatted as shown. After these lines, a free format can be used (with certain restrictions).

Comments are designated the same way as if in “C” language and can be placed anywhere in the file. A comment would consist of /* *This is comment* */. The /* and */ characters indicate a comments beginning and end. All text located between these characters is ignored by the compiler.

The “$” sign denotes the start of font character information. Starting on the line following the “$” sign, a graphical representation of the character is given using dots (.) and ones (1). A “one” is used to represent a pixel that is ON. For each character, there must be “SizeX” columns and “SizeY” rows of font information before the next $ sign occurs. You’ll find several font file examples in the following section.

4.3 Font File Examples

A 5x7 basic font.

```
5,7,48
0
/* This specifies a 5X7 font with no descenders */
/* 48 specifies an offset corresponding to the */
/* ASCII value for "0" */
$         /* Beginning of the font */
/* 0 */
.111
1...1
1..11
1.1.1
11..1
1...1
.111
$
/* 1 */
..1
.11
..1
..1
..1
..1
.111
$
/* 2 */
.111.
1...1
....1
...1.
..1..
.1...
11111
```

The previous example would require a Pitch of at least 6 to be specified to prevent characters from running together. An alternate way to specify the above font would be to change the first line from “5,7,48” to “6,7,48”. This would create a one pixel space between each character provided that the Pitch was not specified (Pitch must be set to 0).

Next is a 7x11 font using a 2 pixel descender. This font is designed to leave a one pixel space between each character (if Pitch is set to 0).

```
8,11,65
2
/* 65 is offset for "A" */
$
/* A */
...1...
..111..
.11.11.
11...11
11...11
1111111
11...11
11...11
11...11        /* This is the character base line */
.......        /* Descender not required for A */
.......
```

Note that even though the descender lines are not needed for this particular character, they must be specified.

Following is another example character which uses descender pixels to build a lower case letter.

```
8,11,103
2
/* 103 is code for "g" */
$
/* g */
.......
.......
.......
.1111 1
11..111
11..111
11...11
.111111
.....11
11...11        /* Descender lines */
.11111.
```

Just as characters can be placed next to each other to form words, a font may can be designed to create a graphic when slices of the graphic are created (in place of characters) and then placed together. Each slice cannot exceed a SizeX of 8 or SizeY of 64.

4.4 Running *fnt2bin.exe*

Once the source file is generated, *fnt2bin* must be run to convert the source file into a binary file by typing:

fnt2bin filename

Two files, *filename.bin* and *filename.asc* will be created. *Filename.bin* can be loaded into the demonstration program, *tvi.exe*, using the "L" command. Refer to the TVM2464 "Designer's Kit Manual" for more information.

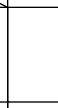
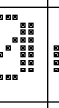
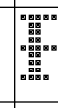
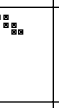
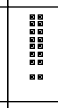
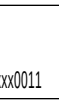
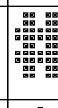
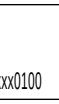
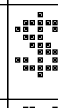
5 Permanent Fonts

The following tables describe the two fonts which are hard coded into the TVM2464 module.

5.1 Font 1 - small 5 x 7 pixels

4 MSB's / 4 LSB's	0010	0011	0100	0101	0110	0111
xxxx0000						
xxxx0001						
xxxx0010						
xxxx0011						
xxxx0100						
xxxx0101						
xxxx0110						
xxxx0111						
xxxx1000						
xxxx1001						
xxxx1010						
xxxx1011						
xxxx1100						
xxxx1101						
xxxx1110						
xxxx1111						

5.2 Font 0 - large bold 7 x 11 pixels

4 MSB's / 4 LSB's	0010	0011	0100	0101	0110	0111	1000	1001
xxxx0000								
xxxx0001								
xxxx0010								
xxxx0011								
xxxx0100								
xxxx0101								
xxxx0110								
xxxx0111								
xxxx1000								
xxxx1001								
xxxx1010								
xxxx1011								
xxxx1100								
xxxx1101								
xxxx1110								
xxxx1111								