CSYS Labs, Inc.

Touch Vision™

TVM24128 Designer's Manual

TVM24128 V1.1

Touch Vision™

A Magnificent Display of Intelligence!

TVM24128 LCD Modules

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

Table of Contents

Table of Contents continued...

1 General Description

The TVM24128 Touch Vision Module provides a complete user interface in a compact, convenient to use module. It contains a graphics LCD display, touch panel overlay, interface electronics for both RS232 serial and parallel communication, auxiliary I/O port, plus, LCD drivers, EL back light, and power supply voltage converters. Because the TVM24128 contains on board voltage converters, only a +5V power supply is required.

The TVM24128 includes a powerful micro-processor which acts as an intelligent controller for the module making interfacing to the TVM24128 very easy. Communication with the TVM24128 is done through either a 20 wire parallel port or via an RS232 serial port. Hand shaking, using software or hardware signals is provided to simplify interfacing to host processors. A status register is also provided to better accommodate pipe lined communication. An ASCII language similar to HPGL is used in ordering the TVM24128 to execute commands.

A rich instruction set, including both text and graphics commands is inherent and with these commands, the user can freely mix and manipulate any graphics and text being displayed. 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. Four level gray scale palets can be individually set for graphics and text. In addition, two separate display planes exist which can be uniquely controlled but simultaneously displayed.

Additional commands can be used to activate a touch panel switch area as well as place a button outline on the LCD display automatically. The host can load the TVM24128 with a label and then place the button using 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 main CPU that a button has been actuated and makes the button's code available for reading by the host. Other button commands are used to activate a "phantom button" area, delete a button, or delete all buttons.

The TVM24128 has the ability to simultaneously mix as many as 5 resident fonts on the screen. Two of the fonts are pre-programmed into the TVM24128 controller. The other three fonts may be custom compiled using a font compiler and down loaded into the TVM24128 font memory. Any font may be used at any time for labels anywhere, including text inside a defined button area.

LCD contrast can be electronically adjusted by using software commands issued to the TVM24128. 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 also built into the TVM24128. The back light can be turned ON and OFF via an on board switch by issuing the appropriate command to the TVM24128. The EL panel has an expected life in excess of 10,000 hours.

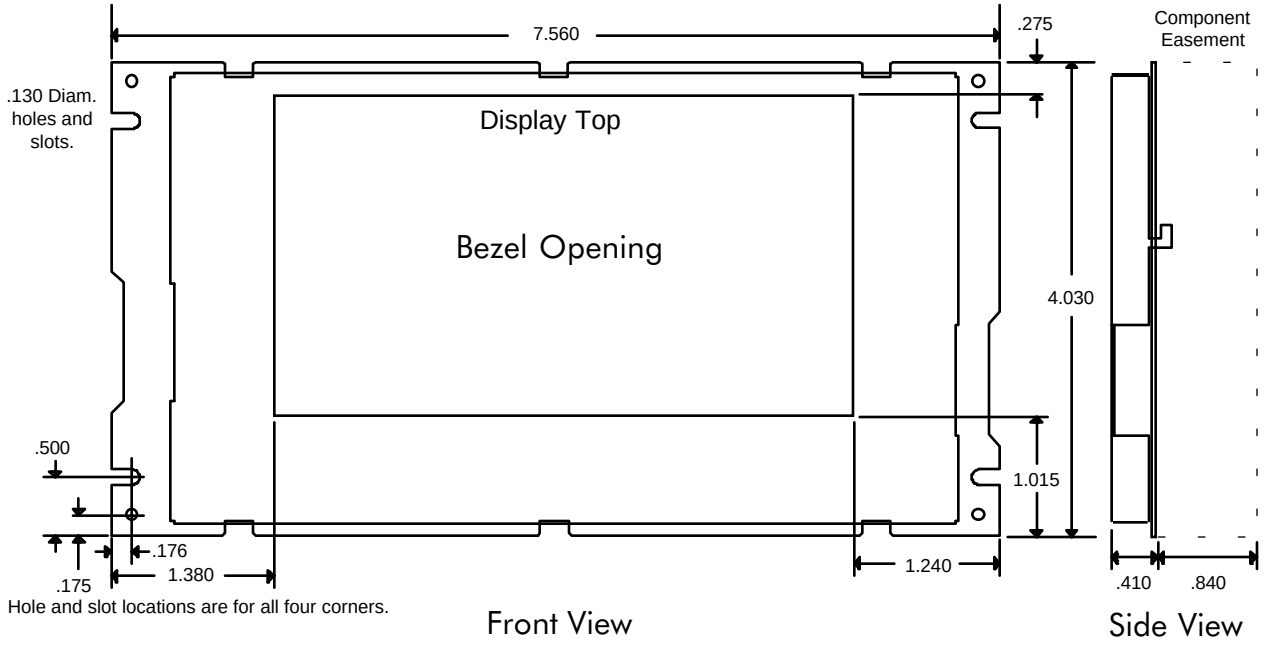
1.1 TVM24128 Features

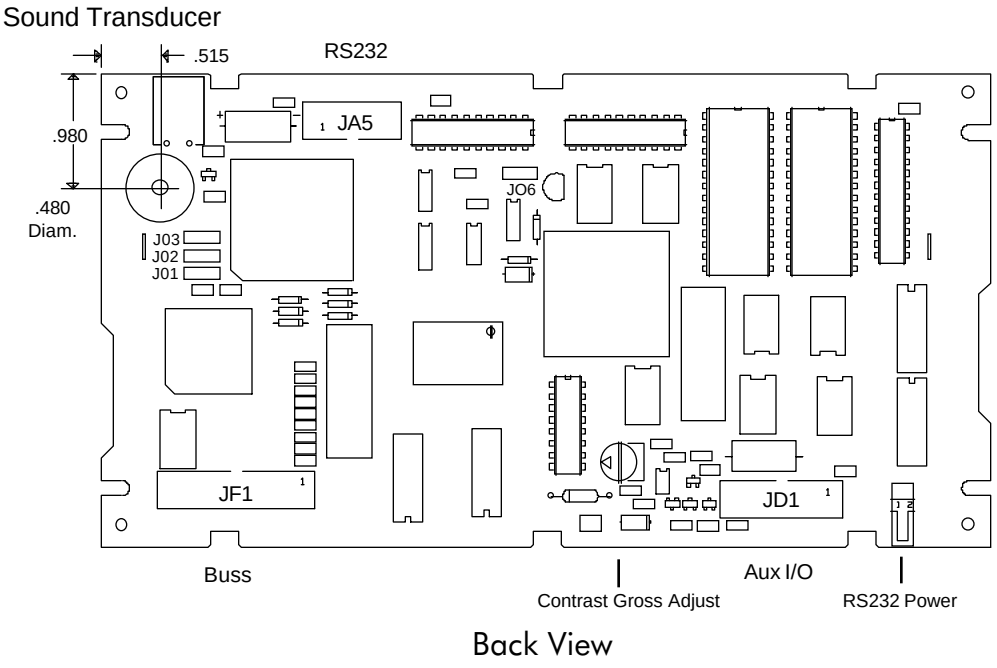
- 240 X 128 Super Twist LCD Display
- 6 X 10 Matrix Touch Panel Overlay
- Single +5V Low Power Supply Operation
- 4 Level Gray Scale Capability
- Software Controlled Long Life EL Back Light
- Compact Size
- Software Controlled Electronic Contrast
- 2 Built in Fonts
- Up to 3 Down Loaded Soft Fonts
- Audio Alarm and/or Key Click
- 4 Programmable Auxiliary Digital Outputs
- 8 Auxiliary Digital Inputs
- Automatic Scrolling in Text Window
- Freely Mixed Multiple Font Types
- Auto Button Generation and Placement
- An Abundance of Graphics Commands
- 8 Bit Parallel and RS232 Serial Interfaces

1.2 Mechanical Description

Many mechanical features have been designed into the TVM24128 which simplify the design of the module into new equipment. Mounting can be accomplished by using 6-32 fasteners through either holes or slots or a combination thereof. Holes or slots not used for mounting, can be used to fasten additional components to the TVM24128. 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 TVM24128, 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 is also shown in the lower middle of the back side view. The potentiometer is factory set to provide a satisfactory range of electronic contrast adjustment.

1.2.1 Mechanical Outlines:



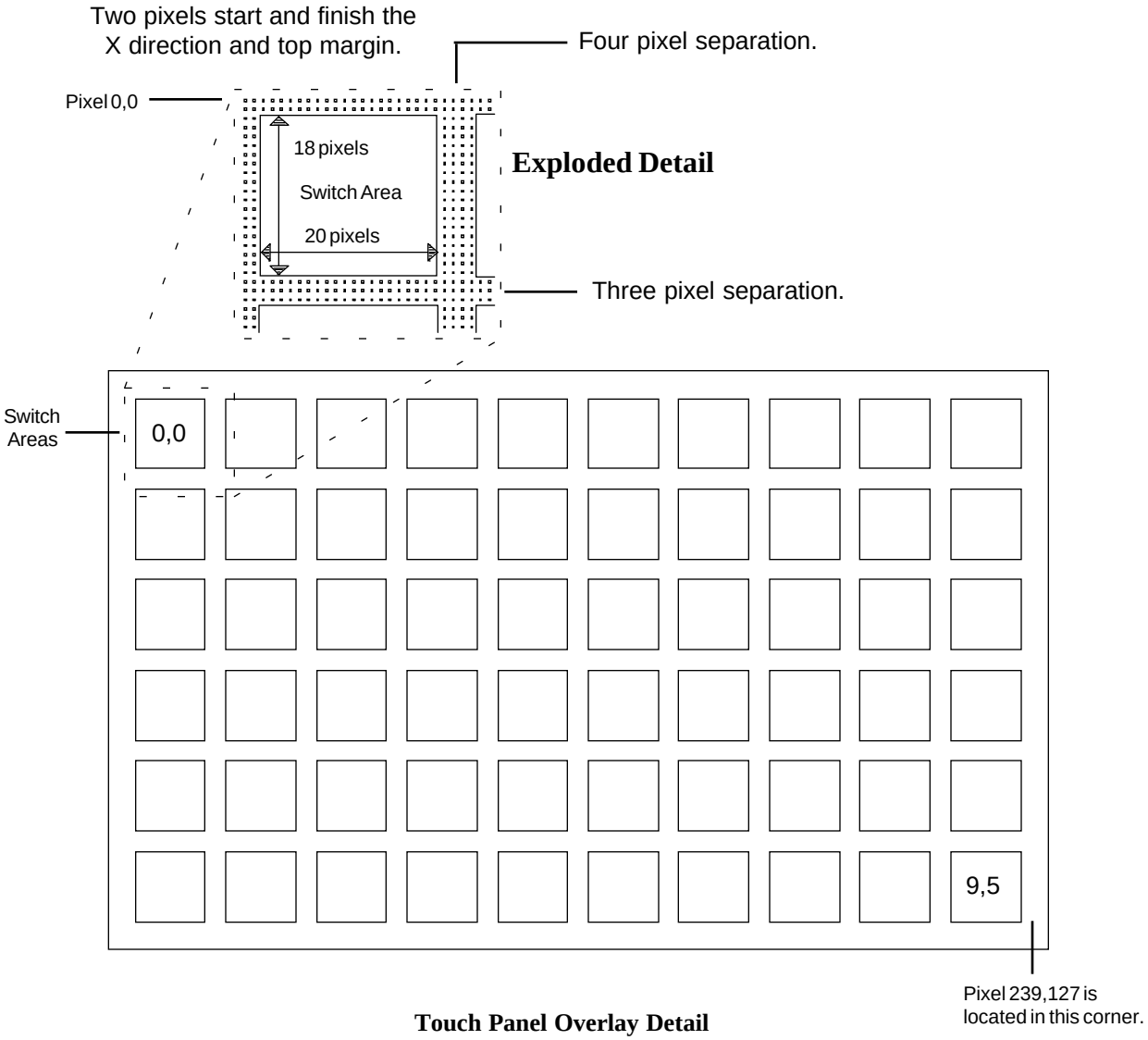


Mechanical Dimensions
Figure 1.1

1.3 Display and Touch Panel Layout

The LCD display is organized in an array 240 dots wide by 128 dots tall. The upper left hand corner of the display is always 0,0 (X,Y). The lower right is 239,127. 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 6 tall, of touch sensitive cells. The upper left hand cell is referred to as 0,0 and the lower right cell is 9,5. Each cell is 20 pixels wide by 18 pixels high. A horizontal space 4 pixels wide and a vertical space of 3 pixels is left as an easement between each touch panel cell.



Touch Panel Overlay Detail

Figure 1.2

1.4 Parallel Interface

Connecting the TVM24128 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 TVM24128 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

JF1 Connections

Table 1.1

Instructions for the TVM24128 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 TVM24128. 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

Address Mapping

Table 1.2

Table 1.3 shows the signals needed to interface to the TVM24128 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

Interface Signals

Table 1.3

1.4.1 Signal Description

The following sections describe the TVM24128 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 TVM24128 will hold down the RESET/ line and any external circuitry tied to the this pin. It is important that sufficient time be allocated to allow the TVM24128 to complete reset and start up prior to receiving instructions from the host processor. Fig. 1.3 shows the internal reset circuit.

Reset Circuit Diagram

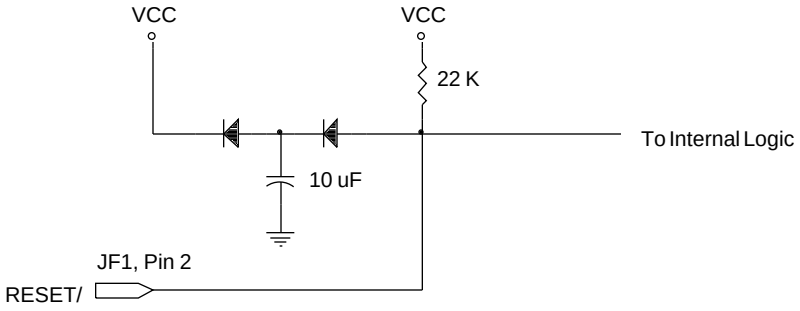


Fig. 1.3

1.4.1.2 DEN/, DRD/, DWR/

A logic 0 level on DEN/ and DWR/ will cause a write to the TVM24128. A logic 0 level on DEN/ and DRD/ will cause a read from the TVM24128. Please refer to Table 1.3 for DA0 and DA1 addressing information.

1.4.1.3 DIBF (Input Buffer Full)

The TVM24128 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 TVM24128 can accept data. A logic 0 indicates that the TVM24128 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 TVM24128 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 TVM24128, 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 3.7.6 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 TVM24128 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.

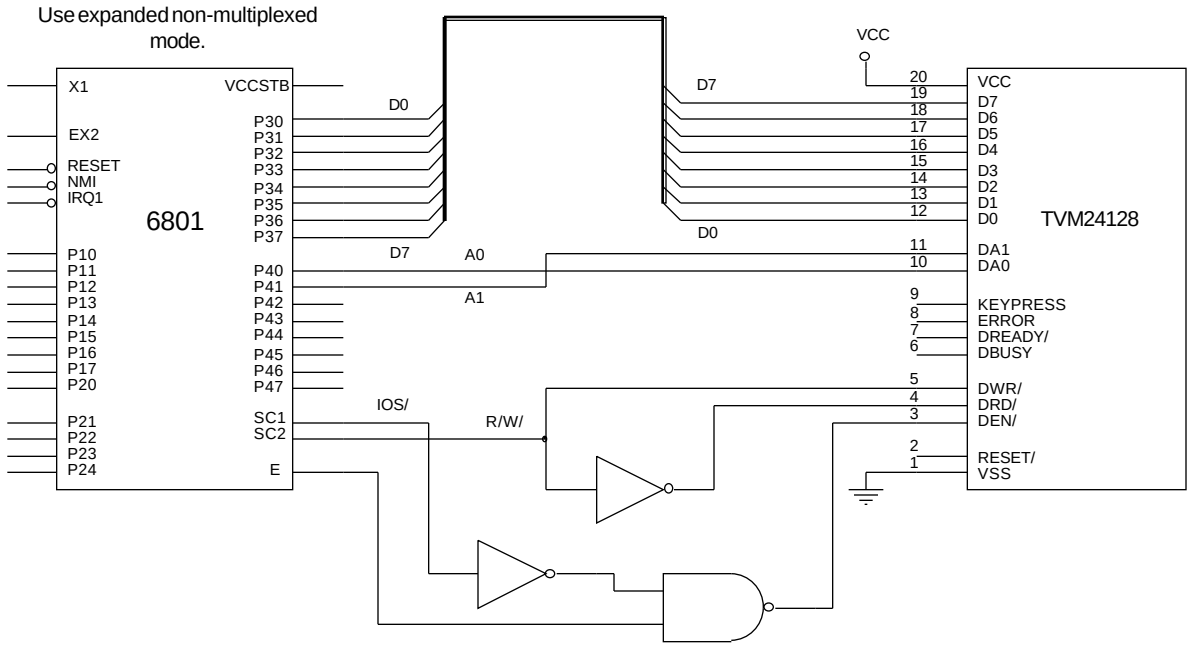
Bit	Function
0(LSB)	DIBF
1	DOBF/
2	ERROR
3	KEYPRESS
4	CPUBUSY
5	0
6	0
7(MSB)	0

Status Register Bit Functions

Table 1.4

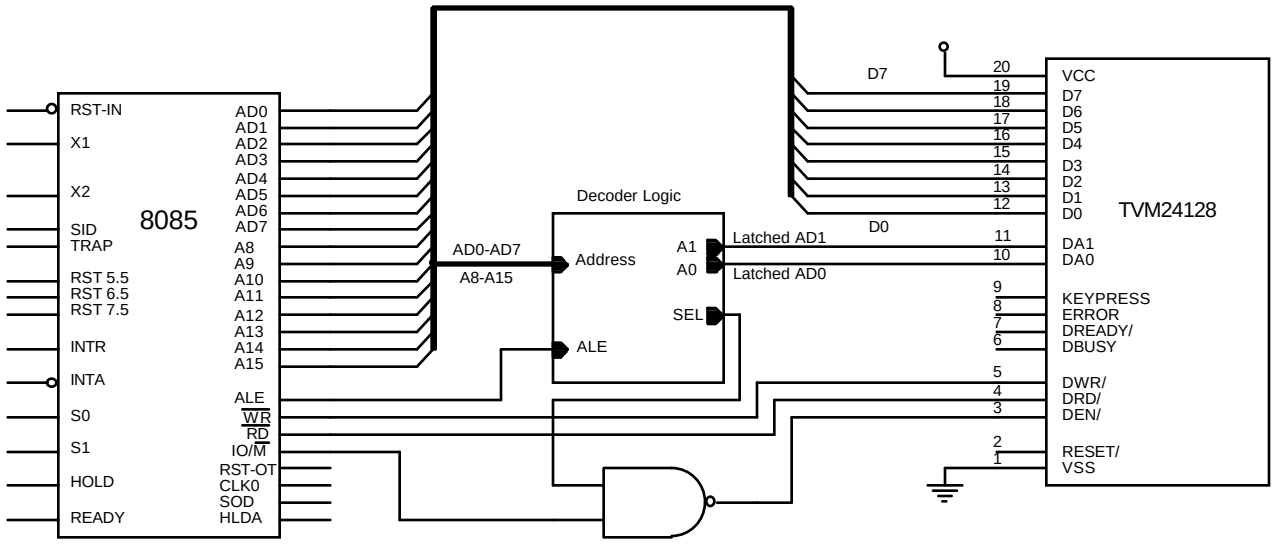
1.4.3 Parallel Interface Examples

Shown below are examples of parallel interfaces for the TVM24128.



6801 Parallel Interface Example

Fig. 1.4



8085 Parallel Interface Example

Fig. 1.5

1.4.4 EL Back Light

An Electroluminescent back light is a standard part of the TVM24128. 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 TVM24128 command. To use the EL panel in the standard configuration, short pins 2-3 on header JO6. 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 JO6 and connect ground to pin 1 and inverter power to pin 2. The maximum inverter input voltage is 5.5V.

1.4.5 Auxiliary Control Port

The auxiliary control port, labeled JD1 on the back view of the TVM24128, enables access to 4 open collector NPN outputs and 8 CMOS digital inputs via the Write Aux Port and Read Aux Port instructions. The inputs are normally pulled up to Vcc through 22K Ohm resistors. The pin functions of JD1 are listed below. A logic one sent to an output turns the appropriate open collector transistor on to pull to ground.

Pin Number	Description	Pin Type
1	Vss	Power
2	Aux In 7	Input
3	Aux In 6	Input
4	Aux In 5	Input
5	Aux In 4	Input
6	Aux In 3	Input
7	Aux In 2	Input
8	Aux In 1	Input
9	Aux In 0	Input
10	Aux Out 3	Output
11	Aux Out 2	Output
12	Aux Out 1	Output
13	Aux Out 0	Output
14	Vcc	Power

JD1 Auxiliary Port Connections

Table 1.5

1.5 RS232 Serial Interface

In addition to a parallel interface, the TVM24128 includes an RS232 interface which uses a programming language called TVSGL (Touch Vision Serial Graphics Language). With only a few exceptions, all commands and data are given as simple ASCII strings, similar to those produced by a terminal. As an example, to move the cursor to a new position using the parallel interface you would issue the SetXY command plus give the position in binary code to the input port. Using TVSGL, the command would be the simple ASCII string CS50,100 which would position the cursor at the 50/100 display co-ordinate.

Special characters, described in detail later, are used to control the hand shaking protocol and status information.

1.5.1 RS232 Command Structure

Serial commands are two characters long. After a serial command is issued, data is provided by giving the ASCII string equivalent of the numeric value required. Data values are separated by a comma (,) or a space ( ). A carriage return can also be used to separate instructions and data. The following examples are valid means of issuing the same command:

CS 50,100
CS50,100
CS 50 100
CS
50
100

Commands can be given in a free format with only a few rules governing syntax. The TVM24128 software will lexically analyze the input string and interpret the instruction prior to execution.

1.5.2 RS232 Command Options and Attributes

Some commands require attributes that are embedded into the parallel instruction OP code. When using the serial interface, these attributes are presented separately following the two letter serial command code. The example below illustrates this:

FD4,5,7,0,48,data,data,data,data ...
or FD4 5 7 0 48 data data data data ...

Down load to font 4, SizeX = 5, SizeY = 7, Descenders = 0, Offset = 48

1.5.3 Special RS232 Binary Data

Some command data is most efficiently sent to the TVM24128 by transmitting binary data directly to the module. This approach is convenient, for instance, when down loading font data. Because binary data can represent any value including special ASCII control characters, a special mode can be entered into to allow sending of this data without conflicts. This mode is entered into by sending an exclamation point "!" prior to any binary data. The module will then receive the next 256 data bytes as binary data without exception. Font data, loaded via this technique, is done so by quantizing the data into packets of 256 bytes. When less than 256 bytes are to be sent, the remainder of the packet must be padded with zeros. The example below shows this:

FD4,5,7,0,48!<256 byte packet>!<256 byte packet>

1.5.4 Quoted Strings

ASCII string data such as that given when using the Input String instruction must be contained within quotes. Any printable ASCII character can be placed within the quotes. The following are examples of strings:

String as given to module	String as printed
"This is a string"	This is a string
"This \"This\" is quoted"	This "This" is quoted
"This \\\"This\"\\ is quoted"	This \"This"\ is quoted

Note that a back-slash escapes the quote allowing it to be printed and not delimit the string. A back-slash is printed by sending two consecutive back-slashes.

To print:	a ",	provide a \"
To print:	a \,	provide a \\

1.5.5 Returned Serial Data Format

Some commands require data to be returned from the TVM24128 to the host. These data strings are always proceeded by a dollar ($) sign. The following example shows a query to return the length of a button area:

Sent to module:	BS5	Ask length of button 5
Returned by module:	$2\n	It is 2 cells long.

The following asks for the current XY position:

Sent to module:	CS
Returned by module:	$50,100\n

The \n character represents a carriage return. It is always appended at the end of a returned string.

1.5.6 RS232 Instruction Errors

Some commands can cause an error condition upon execution. In this case the module will return the pound sign "#" followed by the instruction string that failed. The following example queried the module for the length of a button that had not yet been placed:

Send to module:	BS 5
Returned by module:	#BS$2\n

Asked for length of button with code 5. What returned was the error (#) plus the length information which would be two based on buffer information.

1.5.7 Serial data buffer

The TVM24128 contains a 1024 byte serial data buffer. Commands and data which are sent to the module are stored in this buffer prior to being executed. Thus, commands and data can be sent faster than the module can execute each instruction. When using high baud rates and without safeguards, this buffer could eventually overflow causing a loss of data. To prevent buffer overflow, the TVM24128 supports both the RTS/CTS hardware hand shaking protocol and the XON/XOFF software hand shaking protocol. The buffer hand shaking is determined through the use of the @H, @L, @X and @R instructions described later.

1.5.8 Default Hand Shaking Protocol

The TVM24128 default serial protocol is RTS/CTS with XON/XOFF software protocol disabled. Typically, the host system software would setup the module operating modes and protocol early in the program. It should be noted however, that regardless of what protocol is selected or enabled, the module will respond to commands and return appropriate data. The hand shaking protocol is used only to manage the contents of the serial data buffer. Some applications may not require hand shaking protocol to be used at all.

1.5.9 Serial Interface Commands

Special commands are provided to allow the host system to control the serial interface of the TVM24128. All of these commands are proceeded by the "@" symbol followed by a single character command and optional data.

@K Key press query

Sending the @K command will cause the TVM24128 to return the current keypress status.

Send to module:	@K	
Returned:	@K1	if key press
Returned:	@K0	if not key press

@E Echo String

In some cases, the controlling system must know what instruction the TVM24128 is executing. A method to obtain this information is required since the TVM24128 can buffer as much as 1K bytes of operational data. If a given instruction is inclined to generate an error, the controlling host needs to know when it is executed. The host can then correlate the returned error codes to a particular instruction. The feedback can be generated by following error prone instructions with an @E command. When the host receives the echo string back from the module, the host knows that any error codes received immediately prior where created by the echoed instruction.

Echo capability enables the host to send a block of serial commands as fast as possible and then wait for an appropriate echo before proceeding. The echo string can be of any length but must contain only numbers or letters (alpha numeric) and must not be quoted. The following shows an example of the @E command:

Sent to module	Received from module
@E100	@E100
@EEchoThis	@EEchoThis

@H @L Set High Threshold, Set Low Threshold

@H and @L are used to set the high and low buffer threshold points. The thresholds are used to determine when to change the CTS (clear to send) signal and when to send the XON/XOFF characters. The default high threshold is set at 800 bytes. The default low threshold is set at 200 bytes. Upon the buffer contents falling below the @L threshold mark, the CTS signal goes high. If XON/OFF is selected, the XON (DC1 or 0x11 hex) character is sent to the host. When the buffer contents rise above the @H threshold mark, the CTS signal goes low. If XON/OFF is selected, the XOFF (DC3 or 0x13 hex) character is sent to the host. Following are examples of the @H and @L instructions:

@H850	Set high threshold to 850 bytes.
@H650	Set high threshold to 650 bytes.
@L400	Set low threshold at 400 bytes.

@X Enable XON/OFF

Sending @X selects the XON/OFF software hand shaking.

@R Disable XON/OFF

Sending @R disables the XON/OFF software hand shaking.

@S @D Enable/Disable KPS

The TVM24128 can automatically send a special code to the host whenever a key is pressed. This signal, called the KPS signal, is the same as DC2 (0x12 hex). If @S is sent to the module,

it enables the module to automatically send the KPS signal when a key is pressed. This allows the host system to interrupt what it is doing and service the keyboard without the host continuously polling the module using the @K command. The @D command disables the KPS feature.

1.5.10 RS232 Interface Description

The RS232 hardware connections are made via a 10 pin shrouded header (JA5). The pins are ordered as is a DCE (modem). The TVM24128 requires a straight through cable connection often referred to as a "NULL MODEM".

1.5.11 Serial Data Protocol

All serial data communication uses 8 bit, NO parity and 1 stop bit protocol.

1.5.12 RS232 Signal Names

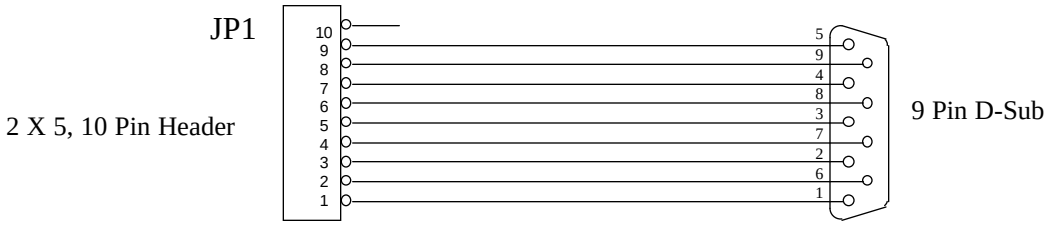
The table below and following figure show the RS232 connector pin designations:

Pin #	Function	Description	Type
1	Ground	Ground connection	
2	Ground	Ground connection	
3	TD	Transmit data	Output
4	RTS	Request to send	Input
5	RD	Receive data	Input
6	CTS	Clear to send	Output
7	N/C	No connection	
8	Ground	Ground connection	
9	Ground	Ground connection	
10	N/C	No connection	

RS232 Connector Pin Functions

Table 1.6

The connector pins are ordered in such a way that a simple adapter can be made converting the 10 pin connector to a 9 pin D-SUB connector typically used on compatible PC's. A wiring diagram for the cable follows:



10 PIN Header to 9 pin D-SUB PC RS232 Cable

Figure 1.6

1.5.13 RS232 Signal Description

Following is a description of the RS232 signals.

TD This is the TVM24128 transmitter output.

RD Receives data from the host processor.

RTS The signal provided by the host system to tell the module it is ready to send data. The module will then raise the CTS signal if the buffer is below the full threshold. As long as RTS stays low, the TVM24128 will not raise the CTS signal. Tieing this line high (+5 V) will permanently enable the CTS signal to indicate the current buffer status.

CTS This signal provided by the TVM24128 tells the host system that it is ready to receive data. If the TVM24128 buffer is below the high threshold when RTS goes high, CTS will go high. Upon the buffer filling above the high threshold, CTS will go low and stay low until the buffer contents fall below the low threshold. If RTS is taken low and high again, the module will re-evaluate the buffer status and raise the CTS signal if the buffer is below the high threshold. CTS can not go high unless RTS is also high.

If software hand shaking is employed, both CTS and RTS can be left disconnected.

1.5.14 Baud Rate Selection

The TVM24128 can operate at any of 8 baud rates. The baud rate is selected by the module at the time reset occurs by reading the baud rate jumpers JO1, JO2 and JO3. A shorting clip between the center pin and pin 3 programs the baud jumper to a logic one. The table below shows the available baud rates and associated jumper combinations:

Baud Rate	JO1	JO2	JO3
300	0	0	0
600	0	0	1
1200	0	1	0
2400	0	1	1
4800	1	0	0
9600	1	0	1
19200	1	1	0
38400	1	1	1

The baud rate is only set during reset.

Table 1.6

1.5.15 Mixed Parallel & Serial Operation

The TVM24128 has the unique ability to operate in a combined parallel and serial mode. Commands and data from both the serial and parallel interface can be received as long as commands do not overlap due to simultaneous transmission by the host or hosts. If the operational data overlaps, the module will mix and match serial and parallel commands and data and execute a composite of both. This will result in some unpredictable behavior.

1.6 Electrical Specifications

1.6.1 Absolute Maximum Ratings:

Storage Temperature: -20°C to +60°C
Operating Temperature: 0°C to +50°C
Vcc with respect to ground: -0.5V to 6.0V
All other pins: Vcc + 0.5V to Gnd - 0.5V

1.6.2 DC Electrical Characteristics:

V_{CC} = 5.0V ± 5% unless otherwise specified.
T_A = 25°C Unless otherwise specified.

Parameter	Min	Typ	Max	Units	Conditions
Power Supply Voltage Supply Current, Back-light OFF Back-light ON	4.75	 175 450	5.25 210 500	Volts mA mA	V_{CC} = 5.0 Volts
V_{IL} Input Low Voltage D7-D0,DA1,DA0,DWR/,DRD/,DEN/,Aux inputs	- 0.5		0.8	Volts	
V_{IH} Input Hi Voltage D7-D0,DA1,DA0,DWR/,DRD/,DEN/, Aux inputs	2.0		V_{CC}	Volts	
V_{OL} Output Low Voltage D7-D0,DBUSY,DREADY/,ERROR,KEYPRESS			.4	Volts	I_{OL} = 2.5 mA
V_{OH} Output Hi Voltage D7-D0,DBUSY,DREADY/,ERROR,KEYPRESS	3.0 V_{CC} - .4			Volts Volts	I_{OH} = -2.5 mA I_{OH} = -100 uA
I_{OZ} Output Floating Leakage D0-D7	± 50		± 300	uA	V_{IN} = V_{CC} or 0 Volts
I_{OL} Output Current Auxiliary Outputs	50			mA	V_{OH} = .5 Volts
$\overline{\text{RESET}}$ (Active Low) $\overline{\text{RESET}}$ (Inactive Hi)	 4.5		.5	Volts Volts	22K Pull Up
External EL Supply Voltage External EL Supply Current	3.0		5.5 200	Volts mA	
RS232 Port Specifications					
Input Voltage Operating Range	- 30		+30	Volts	
Input Threshold Low	0.8		1.2	Volts	
Input Threshold High		1.7	2.4	Volts	
Input Hysteresis	0.2	0.5	1.0	Volts	
Input Resistance	3	5	7	kOhms	
Output Voltage Low		- 9.0	- 5.0	Volts	I_{OUT} = 3.2 mA
Output Voltage High	5.0	9.0		Volts	I_{OUT} = 1.0 mA
Instantaneous Slew Rate			30	V/uS	C_L = 10 pF, R_L = 3-7 kOhms
Transition Region Slew Rate		3		V/uS	C_L = 2500 pF, R_L = 3 kOhms
Output Resistance	300			Ohms	V_{OUT} = ± 2 Volts
Output Short Circuit Current		± 10		mA	

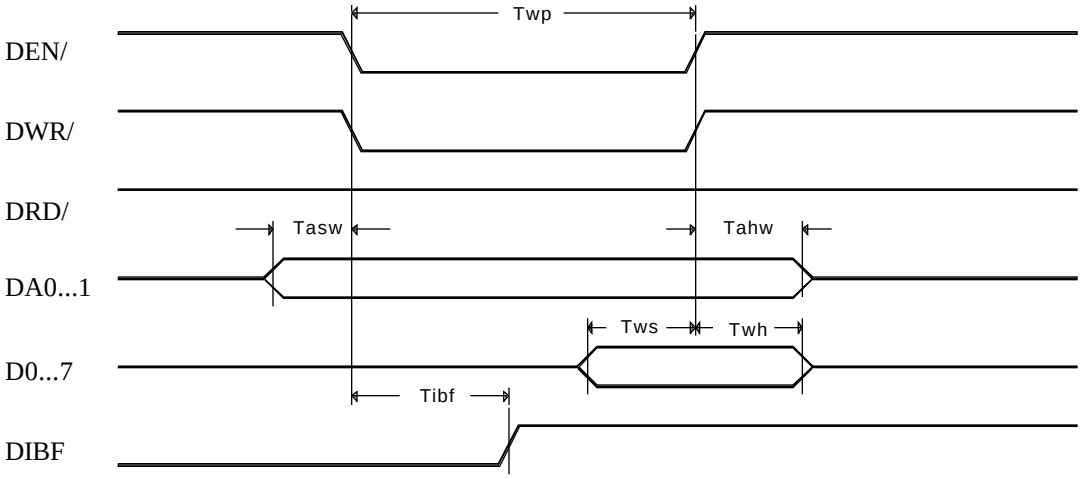
1.6.3 AC Electrical Characteristics

V_{CC} = 5.0V ±5% unless otherwise specified.

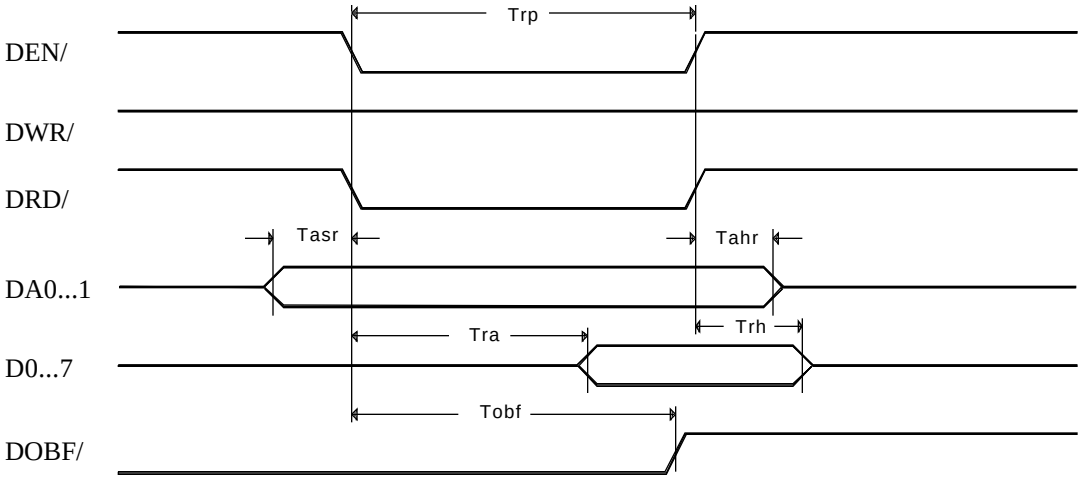
T_A = 25°C Unless otherwise specified.

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 TVM24128 module. There are a total of 60 instructions divided into 9 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 TVM24128 instruction set.

SizeX	Value 1 to 240	Font Size X
SizeY	Value 1 to 128	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 127	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 127	Y Cord Position
AX	Value 0 to 239	CordA X
AY	Value 0 to 127	CordA Y
BX	Value 0 to 239	CordB X
BY	Value 0 to 127	CordB Y
Length	Value -127 to +127	Length of Hor. or Vert. Line
BLength	Value 1 to 10	Phantom Button Length
KeyCode	Value 0 to 255	Button code
a,b,c,d	Bit attributes	Described per instruction
xxx...	Instruction imbedded data	
[RData]	Returned data	
ERROR	Error flag	
Position	Coded Button Position	
{xxx}	ASCII range of xxx bits (0-7)	
(${val},{val}\n)	Serial data returned due to a serial instruction. A $ begins the string and a \n (carriage return) ends the string.	
(![data] [data]...)	Block of 256 bytes of data. A "!" begins each 256 byte block sent.	
AABB	Gray scale palet information.	

Symbol Notations

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

Select Font

Parallel Code: 00000xxx

Serial Code: FS{xxx}

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 high.

Down Load Font

Parallel Code: 00001xxx,SizeX,SizeY,Offset,Descenders, String...

Serial Code: FD{xxx},{SizeX},{SizeY},{Offset},{Descenders},(![data] [data]...) (![data] [data]...)

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.

Set Font Attributes

Parallel Code: 000110ab

Serial Code: FA{ab}

Coding Options: ab = 00, Default, force font into display

ab = 01, OR Font data into display

ab = 10, XOR Font information into display

ab = 11, Forces only "ON" pixels into the 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.

Set Font Gray Palet

Parallel Code: 1011AABB

Serial Code: FG{AABB}

Coding Options:

AA Pixel ON gray palet value
BB Pixel OFF gray palet value

00	White
01	Light Gray
10	Dark Gray
11	Black

Description: This instruction sets the palet value for subsequent input string instructions. AA bits are used to set the ON pixel value and the BB bits are used to set the OFF pixel value.

Set Font Plane

Parallel Code: 0001001x

Serial Code: FP{x}

Coding Options: x = plane number, value is either 0 or 1

Description: This instruction sets the display plane to be used when writing subsequent string instructions.

2.3 Cursor Positioning

SetXY

Parallel Code: 0010000,Xpos,Ypos

Serial Code: CS{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.

ReadXY

Parallel Code: 00100001,[Xpos],[Ypos]

Serial Code: CG(${XPos},{YPos}\n)

Coding Options: None

Description: ReadXY returns the current XY cursor position.

Cursor Up

Parallel Code: 00100010

Serial Code: CU

Coding Options: None

Description: This instruction moves the cursor up scroll value. The cursor will not move if moving will cause it to go outside the text window.

Cursor Down

Parallel Code: 00100011

Serial Code: CD

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.

Cursor Left

Parallel Code: 00100100

Serial Code: CL

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.

Cursor Right

Parallel Code: 00100101

Serial Code: CR

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.

SetX

Parallel Code: 00100110,Xpos

Serial Code: CX{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.

SetY

Parallel Code: 00100111,Ypos

Serial Code: CY{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.

Set Cursor Attributes

Parallel Code: 00010abc

Serial Code: CA{abc}

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

Set Text Window

Parallel Code: 00101000,AX,AY,BX,BY

Serial Code: TW{AX},{AY},{BA},{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 inputted text.

Set Pitch

Parallel Code: 0 0 1 0 1 0 1 0,Pitch

Serial Code: TP{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.

Set Height

Parallel Code: 0 0 1 0 1 0 1 1,Height

Serial Code: TH{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

Input String

Parallel Code: 0 0 1 0 1 1 0 0,String...

Serial Code: IS"Quoted 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. In parallel mode, this text input instruction is terminated at the start of the next instruction (data written to address 0). In serial mode, by a closing quote.

2.6 Graphics Input

Draw Box

Parallel Code: 0 1 0 0 T T T F,AX,AY,BX,BY

Serial Code: GX{TTTF},{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.

Draw Block

Parallel Code: 0 1 1 0 0 0 T T,AX,AY,BX,BY

Serial Code: GB{T T},{AX},{AY},{BX},{BY}

Coding Options:

T T	Attribute
0 0	Force pixel to (BB) palet value
0 1	Force pixel to (AA) palet value
1 0	XOR pixels
1 1	Checker board pattern

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

Draw Horizontal

Parallel Code: 0 1 1 0 0 1 T T,Xpos,Ypos,Length

Serial Code: GH{T T},{Xpos},{Ypos},{Length}

Coding Options:

T T	Attribute
0 0	Force pixel to (BB) palet value
0 1	Force pixel to (AA) palet value
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.

Draw Vertical

Parallel Code: 0 1 1 0 1 0 T T,Xpos,Ypos,Length

Serial Code: GV{T T},{Xpos},{Ypos},{Length}

Coding Options:

T T	Attribute
0 0	Force pixel to (BB) palet value
0 1	Force pixel to (AA) palet value
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 drawn up from Xpos, Ypos. If Length is negative,

the line is drawn down from Xpos, Ypos.

Draw Vector

Parallel Code: 0 1 1 0 1 1 T T,AX,AY,BX,BY

Serial Code: GC{T T},{AX},{AY},{BX},{BY}

Coding Options:

T T	Attribute
0 0	Force pixel to (BB) palet value
0 1	Force pixel to (AA) palet value
1 0	XOR pixels

Description: This instruction draws a vector line from AX,AY to BX,BY.

Set Pixel

Parallel Code: 0 1 1 1 0 0 T T,Xpos,Ypos

Serial Code: GI{T T},{Xpos},{Ypos}

Coding Options:

T T	Attribute
0 0	Force pixel to (BB) palet value
0 1	Force pixel to (AA) palet value
1 0	XOR pixels

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

Set Gray Palet

Parallel Code: 1100AABB

Serial Code: GC{AABB}

Coding Options:

AA	Pixel ON gray palet value
BB	Pixel Off gray palet value
00	White
01	Light Gray
10	Dark Gray
11	Black

Description: This instruction sets the palet value to be used for subsequent graphics instructions.

Set Graphics Plane

Parallel Code: 0111100x

Serial Code: GP{x}

Coding Options: x = plane number, value of 0 or 1.

Description: This instruction sets the display plane to be used for subsequent graphics instructions.

2.7 Button Input

Place Button

Parallel Code: 0 0 1 1 0 0 0 0,KeyCode,0RRRCCCC

Serial Code: BC{KeyCode},{RRRCCCC}

Coding Options: RRR = Row (0-5)
CCCC = Column (0-9)

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 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.

Load Button Buffer

Parallel Code: 0 0 1 1 0 0 0 1,String...

Serial Code: BL"Quoted 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

Get Button Size

Parallel Code: 0 0 1 1 0 0 1 0,KeyCode,[RData]

Serial Code: BS{KeyCode} Returns: (${Size}\n)

Coding Options: None

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

Place Phantom Button

Parallel Code: 0 0 1 1 0 0 1 1,KeyCode,Position,BLength

Serial Code: BH{KeyCode},{RRRCCCC},{BLength}

Coding Options: RRR = Row (0-5)
CCCC = Column (0-7)

Description: This instruction activates the touch panel starting at the position RRRCCCC . 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 by the button placement. If the button will not fit due to an incorrect BLength or due to another button being in the way, the phantom button will not be placed and the ERROR flag will be set.

Delete Button

Parallel Code: 0 0 1 1 0 1 0 0,KeyCode

Serial Code: BD{KeyCode}

Coding Options: None

Description: The button specified by the 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.

Delete All Buttons

Parallel Code: 0 0 1 1 0 1 0 1

Serial Code: BE

Coding Options: None

Description: This instruction deletes all buttons.

Read KeyCode

Parallel Code: 0 0 1 1 0 1 1 0 Returns: [RData]

Serial Code: BK Returns: (${KeyCode}\n)

Coding Options: None

Description: This instruction returns the KeyCode for the button last pressed (or currently pressed). [RData] contains the key code.

Set Button Attributes

Parallel Code: 0 0 1 1 1 a b c

Serial Code: BA{abc}

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.

Set Button Gray Palet

Parallel Code: 1101AABB

Serial Code: BG{AABB}

Coding Options:

AA	Pixel ON gray palet value
BB	Pixel Off gray palet value

00	White
01	Light Gray
10	Dark Gray
11	Black

Description: This instruction sets the palet value to be used for subsequent button placement instructions.

Set Button Plane

Parallel Code: 1110000x

Serial Code: GP{x}

Coding Options: x = plane number, value is 0 or 1.

Description: This instruction sets the display plane to be used for subsequent button placement instructions.

2.8 Display Control

Blank Display

Parallel Code: 1 0 0 0 0 0 0 0

Serial Code: DB

Coding Options: None

Description: This instruction erases the display. The cursor position is not affected.

Clear Display

Parallel Code: 1 0 0 0 0 1 1 1

Serial Code: DC

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.

Dump Display RAM

Parallel Code: 1 0 0 0 0 1 0 0,[RData...]

Serial Code: Not available in serial mode.

Coding Options: None

Description: Display RAM is dumped out of the module. A total of 15360 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.

Load Display RAM

Parallel Code: 1 0 0 0 0 1 0 1,String...

Serial Code: Not available in serial mode.

Coding Options: None

Description: Loads string data into the display RAM. A total of 15360 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.

Move Block Vertical

Parallel Code: 01110100,AX,AY,BX,BY,Distance

Serial Code: DV{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 +/- 127 pixels.

Move Block Horizontal

Parallel Code: 01110101,AX,AY,BX,BY,Distance

Serial Code: DH{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.

Load Gray Dither

Parallel Code: 10000010,GD1,GD2

Serial Code: DD{GD1},{GD2}

Coding Options: None

Description: GD1 and GD2 are 8 bit binary words which determine the dither pattern used when generating gray scale values on the display. Default values have been pre-programmed and in most cases, should be satisfactory. If desired however, the default GD1 and GD2 values can be modified via this instruction in order to modify the density of gray being used for either level. The current value of GD1 represents the light gray dither pattern and GD2 the dark gray pattern.

Reverse Video Mode

Parallel Code: 1000110R

Serial Code: DR{R}

Coding Options: None

Description: This instruction is used to reverse the entire screen in order to better display information when using the transmissive version of the module or to more easily do special effects.

2.9 System Instructions

Soft Reset

Parallel Code: 1 1 1 1 1 1 1 0

Serial Code: SR

Coding Options: None

Description: This instruction causes a soft reset of the controller. All setup variables, the display and attributes are re-initialized.

Set Contrast

Parallel Code: 1 1 1 1 0 0 1 1,Data

Serial Code: SC{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.

Set EL

Parallel Code: 1 1 1 1 0 1 0 a

Serial Code: SE{a}

Coding Options: a=0, EL Off
a=1, EL On

Description: This instruction turns the EL back light on and off.

NOP

Parallel Code: 1 1 1 1 1 1 1 1

Serial Code: SN

Coding Options: None

Description: No operation performed.

Set Beeper

Parallel Code: 1 1 1 1 0 0 0 a

Serial Code: SB{a}

Coding Options: a=0 Beeper OFF
a=1 Beeper ON

Description: This instruction turns the sound transducer on and off.

Read Key Matrix

Parallel Code: 1 1 1 1 0 0 1 0 Returns: [RData...(12 bytes total)]

Serial Code: SK Returns: (${byte 1},{byte 2},... ,{byte 12}\n)

Coding Options: None

Description: This instruction returns the previously stored, current value of the touch panel matrix. There are 12 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:

Touch Panel Column

		Data A		Data B
Row 0	[1]	76543210	[2]	98
Row 1	[3]	76543210	[4]	98
Row 2	[5]	76543210	[6]	98
Row 3	[7]	76543210	[8]	98
Row 4	[9]	76543210	[10]	98
Row 5	[11]	76543210	[12]	98

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 5	0 1 2 3 4 5 6 7 8 9

Write Auxiliary Ports

Parallel Code: 11111000,0000xxxx

Serial Code: SW{xxxx}

Coding Options: None

Description: This instruction writes the auxiliary output port. A logic one written to the port will turn on the output driver which will then sink current to ground. Writing a logic zero will force the open collector output to its high impedance state. The most significant of the xxxx bits sent as data is the Aux Out 3 output. The last bit sent is the Aux Out 0 output.

Read Aux Port

Parallel Code: 11110111 [Aux Data]

Serial Code: SA (${Aux Data}\n)

Coding Options: None

Description: This instruction prompts for the data present at the auxiliary input port. Since there are 8 inputs, 8 bits are returned. The MSB of the code returned is the data present at Aux In 7, the LSB is Aux In 0.

2.10 Serial Interface Instructions

Name	Description
@K	Queries the module to determine if the KeyPress status flag has been set. Returns @K1 if set, @K0 if not set.
@E{alpha-num string}	Echoes the alpha-num string sent to the module. If @E123NUM is sent, the module returns @E123NUM.
@H{number}	Sets the receive buffer high threshold
@L{number}	Sets the receive buffer low threshold.
@X	Enables XON/XOFF hand-shaking protocol. RTC/CTS continues to work for either state.
@R	Turns off XON/XOFF hand-shaking.
@S	Enables KPS such that when a key is pressed. KPS is DC2 (0x12).
@D	Disables KPS.

Refer to section 1.5.9 of this manual for more detailed information on these commands.

2.11 Instruction Set Summary

	Parallel Instruction	Serial Instruction
Font Selection		
Select Font	00000xxx	FS{xxx}
Down Load Font	00001xxx,SizeX,SizeY,Offset,Descenders String...	FD{xxx},{SizeX},{SizeY},{Offset},{Descenders}, !<256 byte packet>!<256 byte packet> ...
Set Font Attributes	000110ab	FA{ab}
Sel Font Gray Palet	1011AABB	FG{AABB}
Set Font Plane	0001001x	FP{x}
Cursor Positioning		
SetXY	00100000,Xpos,Ypos	CS{XPos},{YPos}
ReadXY	00100001,[Xpos],[Ypos]	CG [${XPos},{YPos}\n]
Cursor Up	00100010	CU
Cursor Down	00100011	CD
Cursor Left	00100100	CL
Cursor Right	00100101	CR
SetX	00100110,Xpos	CX{XPos}
SetY	00100111,Ypos	CY{YPos}
Set Cursor Attributes	00010abc	CA{abc}
Text Configuration		
Set Text Window	00101000,AX,AY,BX,BY	TW{AX},{AY},{BA},{BY}
Set Pitch	00101010,Pitch	TP{Pitch}
Set Height	00101011,Height	TH{Height}
Text Input		
Input String	00101100,String...	IS"Quoted String"
Graphics Input		
Draw Box	0100TTTF,AX,AY,BX,BY	GX{TTTF},{AX},{AY},{BX},{BY}
Draw Block	011000TT,AX,AY,BX,BY	GB{TT},{AX},{AY},{BX},{BY}
Draw Horiz	011001TT,Xpos,Ypos,Length	GH{TT},{XPos},{YPos},{Length}
Draw Vert	011010TT,Xpos,Ypos,Length	GV{TT},{XPos},{YPos},{Length}
Draw Vector	011011TT,AX,AY,BX,BX	GC{TT},{AX},{AY},{BX},{BY}
Set Pixel	011100TT,Xpos,Ypos	GI{TT},{XPos},{YPos}
Set Gray Palet	1100AABB	GG{AABB}
Set Graphics Plane	0111100x	GP{x}
Button Input		
Place Button	00110000,KeyCode,0RRRCCCC	BC,{KeyCode},{RRRCCCC}
Load Button Buffer	00110001,String...	BL"Quoted String"
Get Button Size	00110010,KeyCode,[RData]	BS [${Size}\n]
Place Phantom Butt.	00110011,KeyCode,0RRRCCCC,BLength	BH,{KeyCode},{RRRCCCC},{BLength}
Delete Button	00110100,KeyCode	BD{KeyCode}
Delete All Buttons	00110101	BE
Read KeyCode	00110110,[RData]	BK [${KeyCode}\n]
Set Butn Attributes	00111abc	BA{abc}
Set Butn Gray Palet	1101AABB	BG{AABB}
Set Button Plane	1110000x	GP{x}

Instruction Set Continued...

	Parallel Instruction	Serial Instruction
Display Control		
Blank Display	10000000	DB
Clear Display	10000111	DC
Dump Display RAM	10000100,[RData...]	Not Available
Load Display RAM	10000101,String...	Not Available
Move Block Vert	01110100,AX,AY,BX,BY,Distance	DV{AX},{AY},{BX},{BY},{Distance}
Move Block Horiz	01110101,AX,AY,BX,BY,Distance	DH{AX},{AY},{BX},{BY},{Distance}
Load Gray Dither	10000010,GD1,GD2	DD{GD1},{GD2}
Reverse Video Mode	1000110R	DR{R}
System Instructions		
Soft Reset	11111110	SR
Set Contrast	11110011,Data	SC{Data}
Set EL	1111010a	SE{a}
NOP	11111111	SN
Set Beeper	1111000a	SB{a}
Read Key Matrix	11110010,[RData (12 bytes total)]	SK [${byte 1},{byte 2}, ... ,{byte 12}\n]
Write Aux Port	11111000,0000xxxx	SW{xxxx}
Read Aux Port	11110111 [AuxData]	SA [${Aux Data}\n]

Serial Interface Instructions

The following instructions are used in serial interface applications.

Name	Instruction	Description
Query Keypress	@K	Returns @K1 if Keypress is set, @K0 if not.
String Echo	@E{alpha num string}	Echoes the alpha numeric string sent.
Set Buffer High Thresh.	@H{number}	Sets the receive buffer high threshold.
Set Buffer Low Thresh.	@L{number}	Sets the receive buffer Low threshold.
Enable XON/XOFF	@X	Turns on the XON/XOFF hand-shaking protocol.
Disable XON/XOFF	@R	Turns off the XON/XOFF.
Enable KPS	@S	Enable KPS when a key is pressed. KPS is DC2 (0x12).
Disable KPS	@D	Disables KPS.

3 Using the TVM24128

3.1 Display Control

3.1.1 Display Memory

The TVM24128 controller has a built in display buffer memory that is shared by both the LCD display controller and the on-board microprocessor. All display information is written directly into this buffer and is available for immediate displaying.

3.1.2 Display Planes

The display memory is divided up into two sections with each section corresponding to a graphics plane. These planes hereafter will be referred to as Plane 0 and Plane 1. All of the instructions affecting the display have been designed in a manner which allows the writing of information to only one of these planes at a time. The non-addressed plane is not disturbed. Text, graphics and button functions can be assigned to influence a specific plane thus insuring that text and button operations do not modify previously drawn graphics and likewise, that graphics being drawn do not affect text or buttons being displayed. The LCD controller combines the information contained in each of the two planes prior to sending it to the display by overlaying one plane on top of the other.

3.1.3 Using the Gray Scale Palet

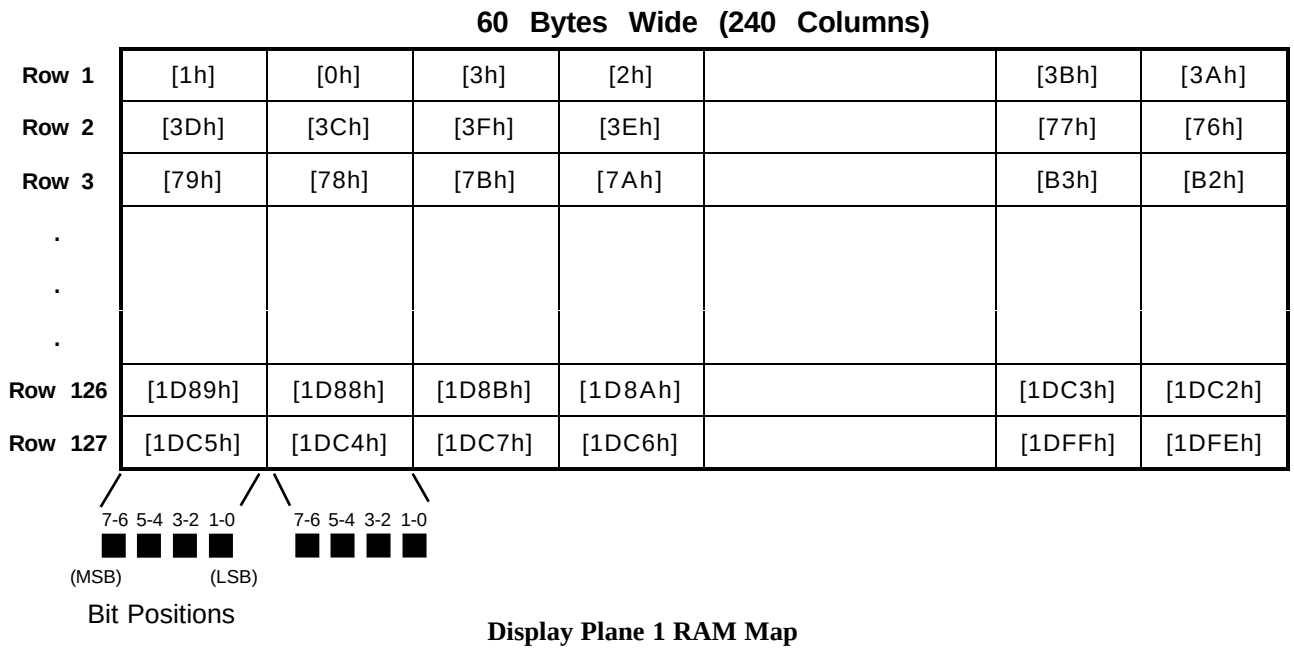
Four levels of gray scale are used when displaying information. The levels available are "Off" (pixels are clear), "Light Gray", "Dark Gray" and "Black". Each pixel of the display is defined using two bits in video RAM which have a value range of between 0 and 3 and which in turn represent the display value from off to black. Raw binary data sent to the module is converted to the actual gray scale information which the display uses via the gray scale palet. Palets can be individually assigned to text, buttons and graphics instructions. In this manner each type of operation is processed using it's own palet to produce the desired combined effect on the display.

Incoming logic zeros are processed independently from logic ones when assigning a gray value. This allows the assignment of any gray value to a logic zero and any gray value to a logic one.

Please refer to the individual gray palet instructions in section two for detailed instruction information.

3.1.4 Dumping and Loading the Display Buffer

Some applications may require direct access to the display buffer. *The instruction, 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 15360 bytes of information between the TVM24128 and the host micro-processor. Figure 3.1 shows a map of how the data is formatted.



Display Plane 1 RAM Map
Fig. 3.1

Data for each of the two planes is formatted from upper left to lower right. Each byte represents 4 pixels. The least significant bit is the right most pixel and the MSB is the left most pixel. Note that the RAM bits as well as the byte pairs are reversed. Data is scanned from left to right, top to bottom. The first 60 bytes are row 1, next 60 are row 2 and so on. The second display plane also follows this same convention but starts at the next available address. The display RAM information is consecutively dumped or loaded for both planes and is ordered by ascending address even though the data position on the display is scrambled.

128 Rows X 60 Bytes = 7680 Bytes per Plane

7680 Bytes X 2 Planes = 15360 Bytes

3.2 Text Window and Display

The TVM24128 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 TVM24128, 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.

Care should be used when drawing graphics through the cursor. It is best to always remove the cursor, draw the graphics and then replace the cursor. This will prevent ghosting of the cursor due to graphics data when the cursor is moved.

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 Effects on Cursor Placement

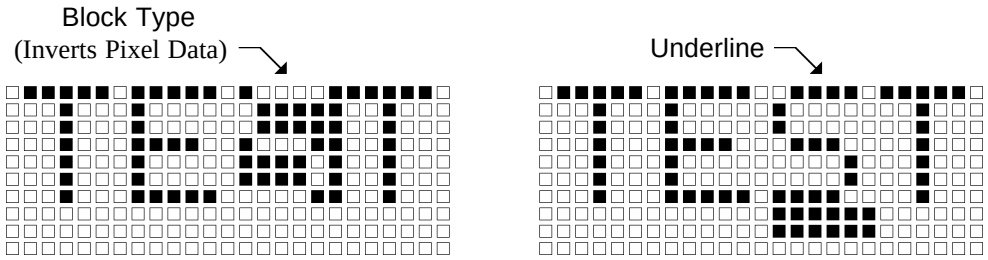
Fonts which use descenders cause the underline cursor to be placed such that the upper left corner of the cursor begins at the cursor Ycord - SizeY + Descenders. This positions the under bar cursor at the exact place in the line at which the descenders start. If a block cursor is used, descenders have 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.



Cursor Styles
Fig. 3.2

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 illustrates the cursor shape and position for both block and underline.

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 TVM24128 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, which can contain one font, can accommodate up to 8192 bytes of information. Each particular character code for a font which

is less than or equal to 8 pixels wide occupies an amount of memory equal to SizeY of font memory. In other words, each character of a 5 by 7 font will take 7 bytes of memory. For fonts having characters greater than 8 pixels wide, the memory required is calculated using the formula;

$$\text{Memory size} = (((\text{SizeX} - 1)/8) + 1) \times \text{SizeY}$$

The maximum character height is 128 pixels. The maximum allowable width is 240 pixels. Thus, a single character can occupy the entire screen.

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 *Input String* 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 forces font data into the display buffer. Pixels which the font defines as on will be forced to a gray value which is determined by the palet logic "One" value. Likewise, pixels the font defines as off will be forced to the palet logic "Zero" value. 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.

If the font attribute is set to force pixels "ON", only the newly enabled pixels will be placed on the display using the palet logic "One" value. All display data outside of this area remains un-touched.

Given the gray palet effects as well as the logical controls available, there is a large variety of display special effects possible. Due to the shear number of options, it's probably best to move on at this point and allow your imagination to take over.

3.6 Graphics Input

A graphics command can place information anywhere in the display field. All coordinates used for graphics operations can range across the entire display. If a coordinate is given that would cause data to be placed outside the display area, 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

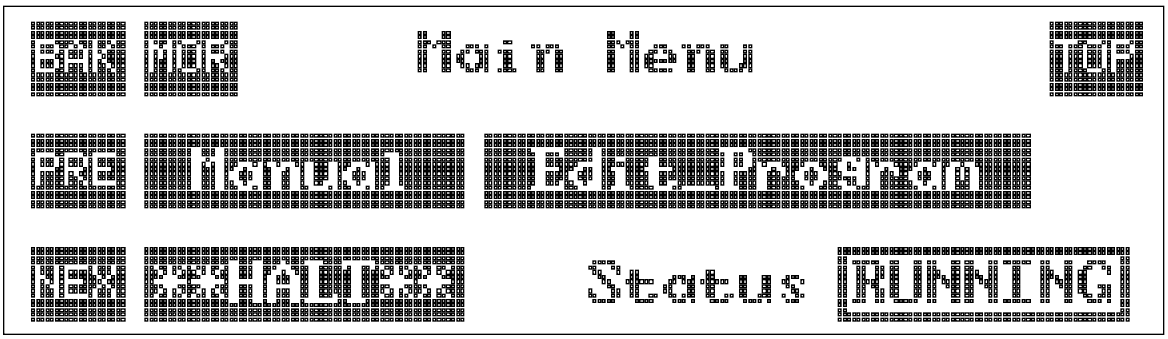
When a button is placed, the button attributes activated using *Set Button Attributes* will determine how it functions. After attributes are set, all subsequent button placements will use these attributes to flavor the button's operation. Buttons with different attributes may be placed onto the display 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 codes for the button attributes.

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 TVM24128 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 18 pixels high, the exact height of a Touch Panel cell. Fig. 3.3 shows a typical display with several buttons.



Example Display
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 Effects 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 TVM24128 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

3.8.1 Soft Reset

In some cases it may be desirable to reset the TVM24128 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 TVM24128 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 is also included on the TVM24128 board for coarse contrast adjustments. This trimmer is set at the factory and should not need adjustment. If adjustment is necessary, the electronic contrast should be set to 16 (the center of its range) before adjusting the trim pot.

3.8.3 EL Back Light Control

The instruction *Set EL* is used to turn the back light ON and OFF.
For more information on external back light control, please refer to section 1.3.6.

3.8.4 Audio Transducer

The *Set Beeper* instruction is used to turn the audio transducer 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.

3.8.6 Auxiliary Port Control

The auxiliary port consists of 8 CMOS inputs and four open collector NPN outputs which can be individually addressed and or read to accommodate supplemental control panel operations and indicators. The outputs are controlled using the *Write Aux Port* instruction and can be used to drive LEDs or to act as strobes for an additional key pad. The *Read Aux Port* instruction is used to read the 8 available inputs. The inputs are useful in capturing switch information other than that presented by the touch panel interface. The auxiliary port is totally independent with respect to the touch panel. Refer to section 2.9 for detailed information on these instructions.

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 TVM24128 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 Example 4.1.

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

Example 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. Each font character or graphic cannot exceed a SizeX of 239 or SizeY of 128.

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 TVM24128 "Designer's Kit Manual" for more information.

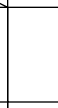
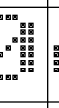
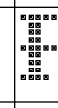
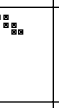
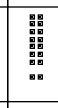
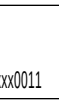
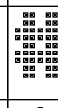
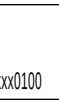
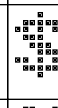
5 Permanent Fonts

The following tables describe the two fonts which are hard coded into the TVM24128 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								

Index

E

F

G

H

I

J

K

L

M

N

P

Q

R

S

T

U

V

W

X