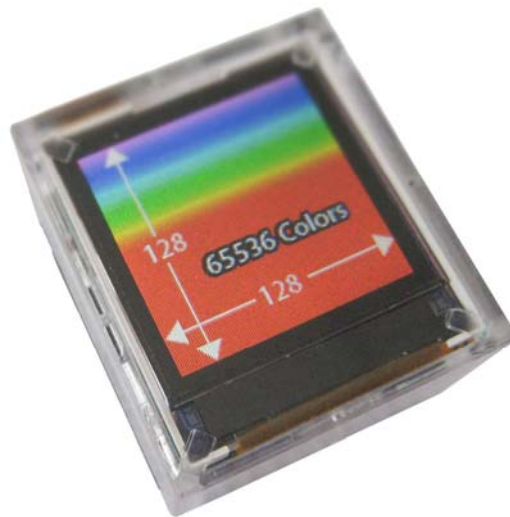




# TFT128 SCREENKEY DATASHEET

Order Part Number:

**TFT128-x**



**SK Interfaces Ltd.**

Unit 11, Keypoint Business Park,  
42 Rosemount Park Drive,  
Ballycoolin Road,  
Dublin 11, Ireland.

Tel: +353-1-8855 075

Fax: +353-1-8855 095

Please visit our website [www.screenkeys.com](http://www.screenkeys.com) for additional information.

## Version Control

<b>Version</b>	<b>Date</b>	<b>Notes</b>
0.4	8 February, 2008	First draft release  Please note that the contents of this datasheet may change prior to the first 1.0 release
0.9	7 August, 2008	Updated with preliminary release information
1.0	19 December, 2008	Updated with information for 1.0 release of product

## Contents

Version Control.....	2
Contents .....	3
1. Technical Description .....	5
2. Typical Uses and Application .....	5
3. Dimensions .....	6
3.1. Drill Masks.....	7
3.2. Recommended Flex-tail Connector .....	9
3.3. Orientation.....	9
3.4. “No Solder” Connection .....	10
3.5. Multiple ScreenKey Placement.....	10
4. Switch Specification.....	11
4.1. External Switch Monitoring .....	11
4.2. Internal Switch Monitoring.....	11
5. ScreenKey Specifications.....	12
6. Electrical Interface .....	13
6.1. Single-ScreenKey Wiring Diagram .....	14
6.2. Backlight Control Options .....	14
7. Bitmap Graphics .....	15
7.1. Portrait Bit-mapping .....	16
7.2. Landscape Bit-mapping .....	16
8. Text Display .....	17
8.1. Factory Default Character Set .....	19
8.2. Downloadable Fonts .....	20
8.3. Reserved Font Memory .....	20
9. Serial Peripheral Interface.....	21
9.1. SPI Transmission .....	22
9.2. Multiple Byte Transmission .....	23
10. High-Level Command Mode.....	24
10.1. Command Structure .....	24
10.2. ScreenKey Status Byte .....	25
10.3. Error Recovery & Resync .....	26
10.4. Data Information <u>from</u> ScreenKey.....	27
10.5. Inactivity Timeout .....	27
10.6. Communications Flowchart.....	28
10.7. Example Communication Scenarios .....	29
10.8. Screen Refresh Rate .....	30
10.9. High-Level Command Set.....	31
Command 01h – Key Mode Reset.....	31

Command 09h – Switch Acknowledge .....	32
Command 10h – Set Orientation .....	33
Command 11h – Set Color .....	34
Command 12h – Set Cursor Position .....	35
Command 13h – Clear Display .....	35
Command 14h – Replace Color.....	36
Command 15h – Set Backlight .....	37
Command 20h – Display Text.....	38
Command 21h – Display 256-Color Graphic .....	39
Command 22h – Display Full-Color Graphic .....	40
Command 23h – Display 16-Color Graphic .....	41
Command 24h – Draw Rectangle.....	42
Command 25h – Draw Circle.....	43
Command 26h – Set Flash .....	44
Command 30h – Download Font .....	45
Command 31h – Download Color Palette Table .....	46
Command 32h – Download Graphic.....	47
Command 33h – Recall Graphic.....	48
Command 34h – Report Free Graphic Memory .....	49
Command 35h – Report Memory Contents .....	50
10.10. Programming Example .....	51
11. High-Speed Mode .....	53
11.1. Activating High-Speed Mode .....	53
11.2. Sending Graphical Data.....	54
11.3. Frame Synchronization .....	54
11.4. Screen Refresh Rate .....	54
11.4. Exit High-Speed Mode .....	55
12. Order Information .....	56
13. Contact Information .....	56
Important Notice .....	57
Warranty Disclaimer.....	57
Copyright Notice.....	57
General Notice .....	57

## 1. Technical Description

The TFT128 ScreenKey is a push-button keyswitch with built-in full-color high resolution TFT graphics display. The display is backlit with a white LED and the TFT liquid crystal display has a resolution of 128\*128 pixels where each pixel color can be chosen from a palette of 65,536 different colors.

The ScreenKey is an intelligent device with an onboard microcontroller and graphical display controller. The ScreenKey fully implements a 4-wire SPI interface with an expanded command set allowing simple ASCII control or full bitmap graphic display, or any combination of the two. The key supports downloadable content and is designed to minimize the bandwidth requirements from a host controller, particularly when used in a multi-ScreenKey environment.

The ScreenKey supports two operating modes:

- *High-level command mode* offers commands to simplify application integration by using text display, clear display, report keypress, etc. This mode reduces the maximum frame-rate for the key.
- *High-speed mode* disables all the command processing and implements a simple graphical only interface. This mode delivers approx 10 frames per second as the frame refresh rate.

The key clicks into place without soldering for simple assembly and easy field replacement. A flex-tail cable is used to make connection to the target electronics via a PCB connector.

## 2. Typical Uses and Application

The multi-function TFT128 ScreenKey, with its high resolution LCD display and 65,536 color support, is suited for any application requiring a man-machine interface. ScreenKey technology is ideal for many different markets and applications where multi-functional input with graphical or text output is required, including:

- Media and Broadcasting
- Audio/Visual Studio and Production Equipment
- Industrial controls
- Point-of-Sale, Point-of-Information
- Medical Devices
- Automotive Industry
- Aerospace
- Financial Services / Stock Trading
- Air Traffic Control
- Telecommunications
- etc.

The advantages of ScreenKeys are that they are simple to integrate into hardware, and software control of the integrated LCD is very simple and straightforward. This allows for the easy integration of the switch into products without extensive development efforts.

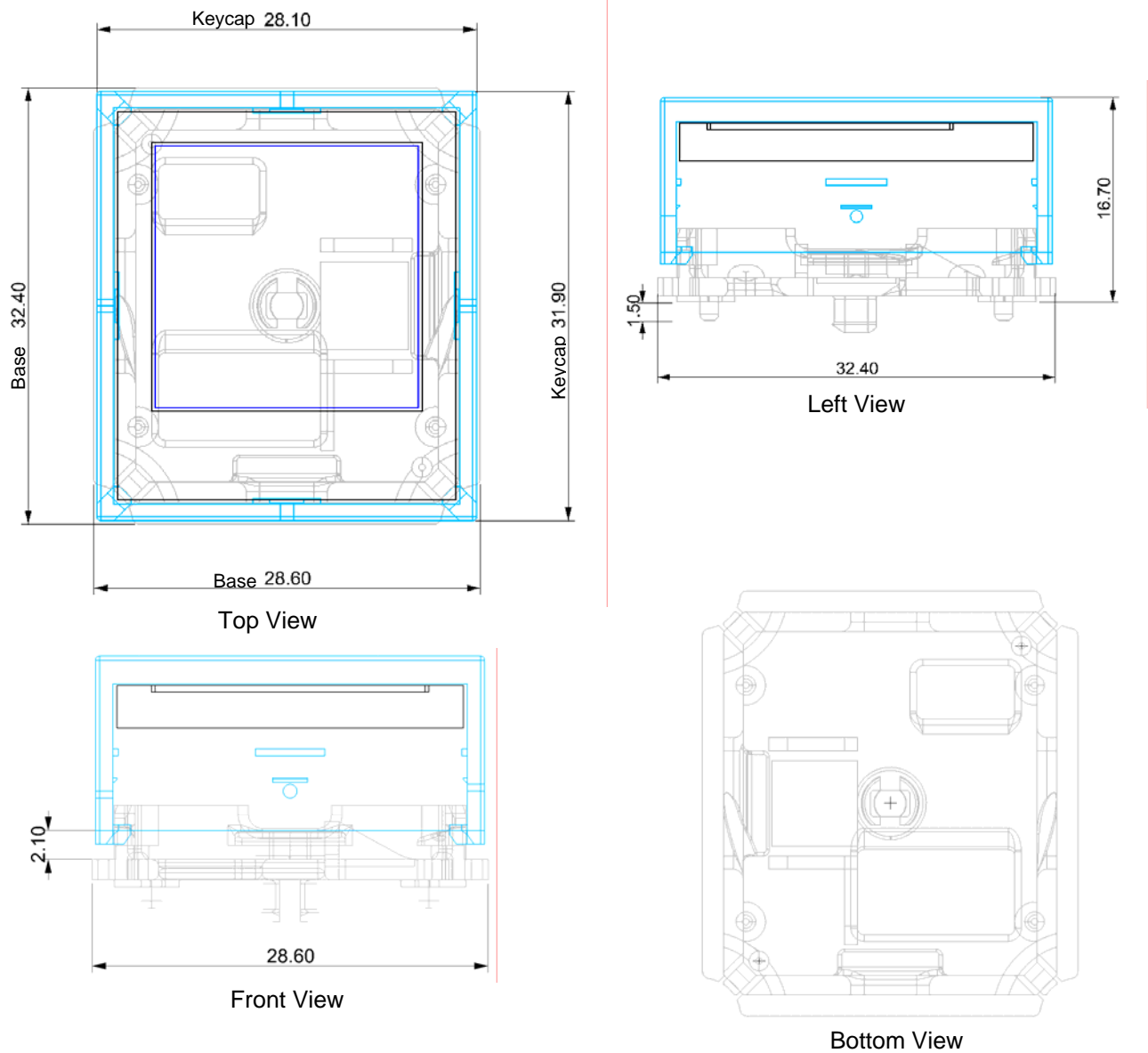
- Industry standard SPI interface
- 3.3V logic and power supply
- No external display controller required
- Greater signal effectiveness through full color display
- Display text and/or graphics with 128\*128 pixel resolution
- Fast data transmission with SPI transmission rates up to 10MHz
- Displayed image is maintained as long as power is applied (no refresh required)
- Excellent display with high contrast and wide viewing angles
- Landscape or portrait orientation
- Keyswitch lifetime of >3 million operations
- Intuitive user guidance through menu systems

### 3. Dimensions

The overall ScreenKey dimension is 28.60mm wide and 32.40mm long.

The height of the ScreenKey in its resting position is 16.70mm from the mounting pcb.

The switch travel distance is 2.10mm. The switch height when pressed is 14.60mm.



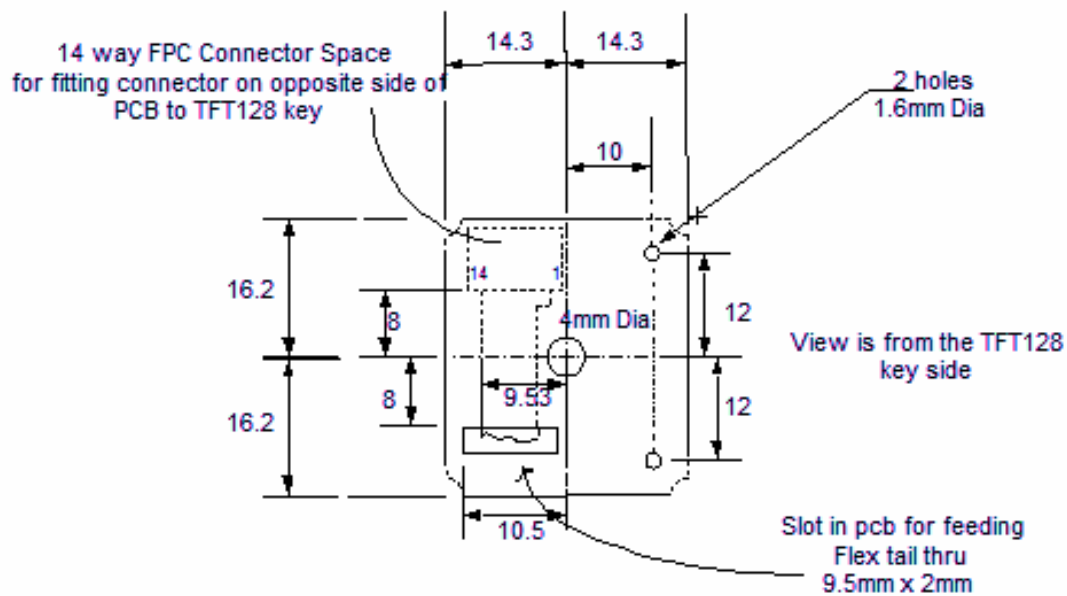
Drawing: Dimensions of TFT128 ScreenKey in mm (all dimensions +/-0.2mm).

### 3.1. Drill Masks

The TFT128 may be mounted with the flex-tail connector on the same PCB side as the key (component side) or with the flex-tail connector on the reverse side of the pcb from the ScreenKey (solder side). See section 3.2 below for more information.

#### Drill Mask for Solder-side Fitting of FPC Connector

The following drawing shows the drill mask and flex-tail placement if solder side mounting of the FPC connector is used:

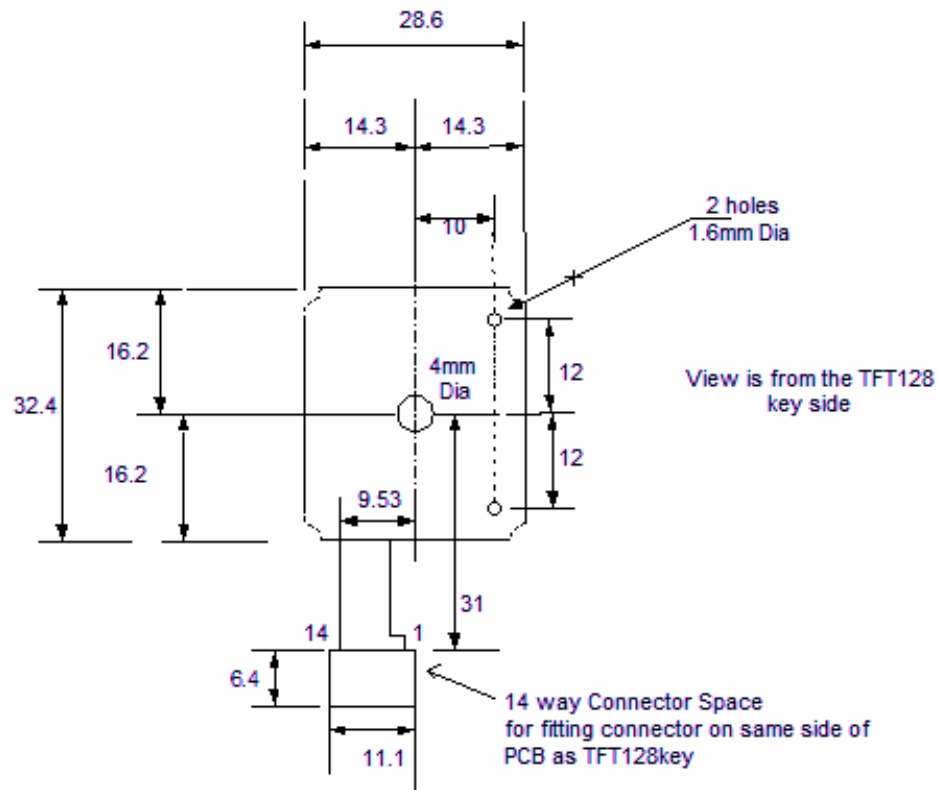


Drawing: Drill mask for solder-side mounting of FPC connector (reverse side of PCB to key).

If a ScreenKey grid layout is required, then the flex-tail connectors **must** be fitted on the PCB reverse side (solder-side) and slots provided to feed the flex-tail through the PCB.

**Drill Mask for Component-side Fitting of FPC Connector**

The following drawing shows the drill mask and flex-tail placement if component-side (same-side) mounting of the FPC connector is used. **This is only possible if a single of row of ScreenKeys is used.**

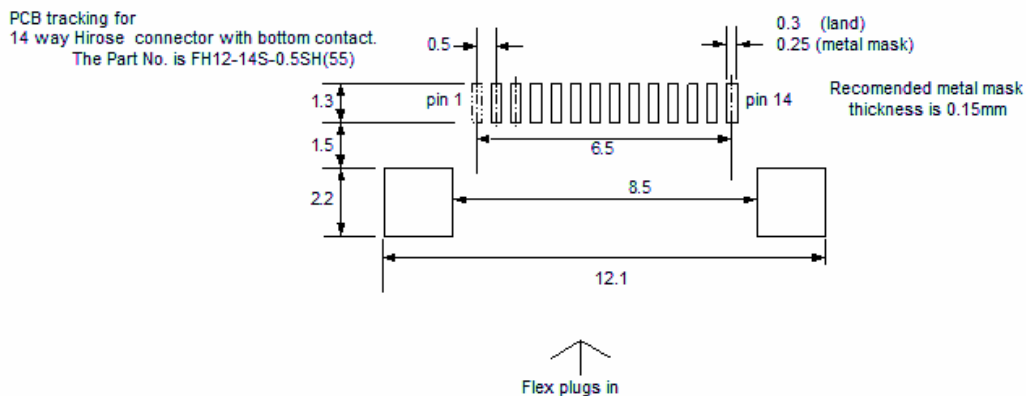


- Note: 1. The recommended FPC connector is a Zif 14 way Hirose connector with bottom contact. The Part No. is FH12-14S-0.5SH(55). The connectors overall footprint is 11.1mm x 6.4mm
- 2. All dimensions are in mm

Drawing: Drill mask for component-side mounting of FPC connector (same side of PCB to key).

**Recommended PCB Tracking for FPC Connector**

The diagram below shows the recommended tracking for the FPC connector (if using the recommended FPC connector – see section 3.2 below).



Drawing: PCB tracking layout for recommended FPC connector

### 3.2. Recommended Flex-tail Connector

The flex-tail cable used in the TFT128 is a 14-way 0.5mm pitch single-sided gold contact.

The recommended FPC connector for use with the TFT128 is a ZIF bottom contact connector:

**Hirose (HRS) FH12-14S-0.5SH(55)**

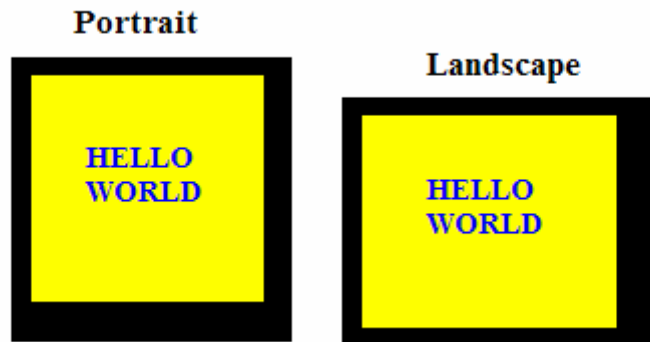
This connector is suitable for both types of ScreenKey mounting.

*Note:*

*The TFT128 is **not** supplied with a flex-tail connector as standard.*

### 3.3. Orientation

The TFT128 can be mounted either in portrait (vertically) or horizontally (landscape), depending on the user and application preference.



Drawing: TFT128 may be mounted either in portrait or landscape orientation

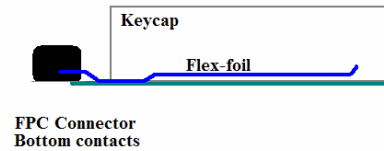
### 3.4. “No Solder” Connection

The TFT128 electrically connects to a PCB via a flex-tail connector.

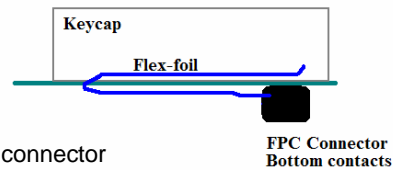
The flex-tail exits from the TFT128 via a cut-out in its base:



#### Top-side Mounting



#### Bottom-side Mounting



Drawing: Flex-tail cable path to connector

Two different assembly options are possible.

1. The flex-tail cable may be fed through a slot in the PCB to a connector mounted on the underside of the PCB. This provides the simplest and easiest connection for hand assembly, particularly when multiple keys are fitted in close proximity.

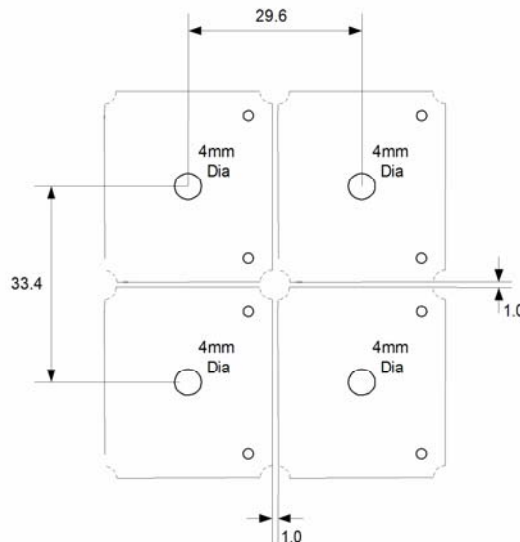
*Note that the flex-tail gold fingers will be on the **bottom** of the cable as it enters the connector in this configuration.*

2. An alternative method of mounting is to place the flex-tail connector on the same PCB side as the ScreenKey itself. This is not suitable if a ScreenKey grid layout is required.

*Note that the flex-tail gold fingers will be on the **bottom** of the cable as it enters the connector in this configuration.*

### 3.5. Multiple ScreenKey Placement

When multiple TFT128 ScreenKeys are fitted in a grid or row/column arrangement, at least 1mm spacing must be reserved on each side of the ScreenKey.

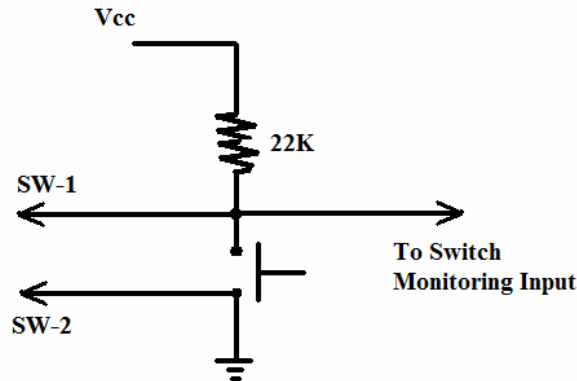


## 4. Switch Specification

Switch detection can be handled in one of two ways to suit the user or application preference.

### 4.1. External Switch Monitoring

The flex-tail contacts expose two switch contacts to the driving circuitry. This is a push-to-make electrically isolated switch.



To use the switch in this manner, connect SW-1 to Vcc via a 22K resistor and also to the external switch monitoring circuitry. Connect SW-2 directly to Gnd.

In its normal or unpressed mode, the switch monitoring circuit will detect logic high (Vcc). When pressed this value will be logic low (0V) until the switch is released.

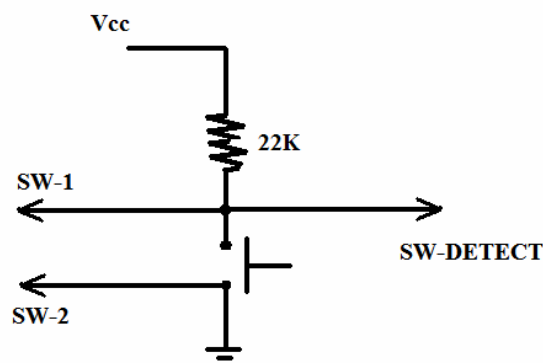
### 4.2. Internal Switch Monitoring

The TFT128 ScreenKey can monitor the internal switch mechanism and report key presses in its *Return Status Byte* for SPI transactions. This can be useful if the user prefers not to implement an external switch detection circuit.

The switch mechanical parameters are exactly the same as in the table in section 4.1.

Electrically, the switch circuitry should be wired as shown below. This is almost identical to section 4.1, except that the switch monitoring is performed by the ScreenKey via SW-DETECT on the FPC.

Note that *Internal Switch Monitoring* is only available if the switch is used in *high-level command mode*. In *high-speed mode*, the SSB is not returned to the host and *external switch monitoring* must be employed.



## 5. ScreenKey Specifications

Dimensions:

Description	Values
Overall Dimension (L x W)	32.4 x 28.6 mm
Keycap Dimensions (L x W)	31.9 x 28.1 mm
Base Dimensions (L x W)	32.4 x 28.6 mm
Overall Switch Height	16.7 mm
Clear Window Size ( X x Y )	22.1 x 22.1 mm

Environmental:

Description	Values
Operating Temperature	-20° ..... +70° Celsius
Storage Temperature	-30° ..... +80° Celsius
Humidity	max. 90% relative at 60° Celsius
Life Cycle	5 – 7 years (life cycle from date of manufacture and may be reduced by exposure to excess humidity, temperature and ultra-violet light)

Electrical:

Description	Values
Operating voltage (Vcc)	3.0v – 3.3V dc
Max supply voltage	3.6V
Max input voltage	Vcc + 0.5V
Current Consumption	27mA (typical)
Logic Input Low (max)	0.2*Vcc – 0.1V
Logic Input High (min)	0.7*Vcc

Display:

Description	Values
LCD Glass	Thin Film Transistor (TFT) / Transmissive
Pixel Field Size ( X x Y )	19.584 x 19.584 mm
Pixel Matrix ( X x Y )	128 (RGB) x 128
Viewing Direction	6 o'clock
Viewing Angle	45 deg (min)
Response Times (typical)	15ms "white-to-black" / 35ms "black-to-white"
Contrast Ratio	Min 120 / Typ 170 (white/black)
Backlight	White LED (half-life of 50,000 hours)

Internal Switch:

Description	Values
Volume Resistance	< 200 Ohm
Insulating Resistance	> 100 MOhm
Contact bounce time	< 20 ms
Key Travel	2.1 mm
Operating Force	1.0 to 1.4 Newton
Durability	>3 Million operations
Circuit Current	5 mA max. @ 5V dc

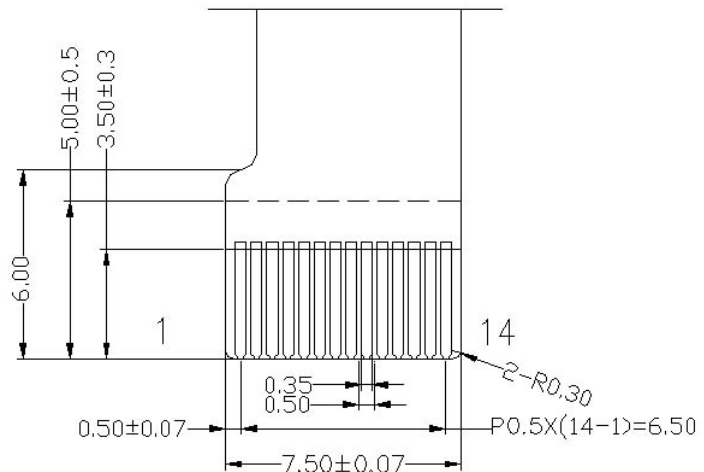
Plastic Materials:

Description	Material	UL Listing
Keycap	Polycarbonate GE Lexan 201R-111	UL 94 V2
Key base	Polycarbonate GE Lexan 201R-111	UL 94 V2

## 6. Electrical Interface

Electrical connection is made to the TFT128 via the 14-way flex-tail cable.

1	Vcc
2	Vcc – LED
3	GND – LED
4	D- (not used)
5	D+ (not used)
6	GND
7	MOSI
8	SCK
9	MISO
10	SS
11	Not connected
12	SW-DETECT
13	SW-1
14	SW-2



Drawing: Flex-tail cable connections and orientation

The supply voltage ( $V_{cc}$ ) is 3.0V to 3.3Vdc. Contacts 3 and 6 must **both** be connected to the supply 0V. Contact 1 must be connected to  $V_{cc}$ . Contact 2 controls the +V supply to the backlight LED (see section 6.2 below for details on how the backlight LED may be controlled).

MOSI/SCK/MISO and SS are the 4-wire SPI interface.

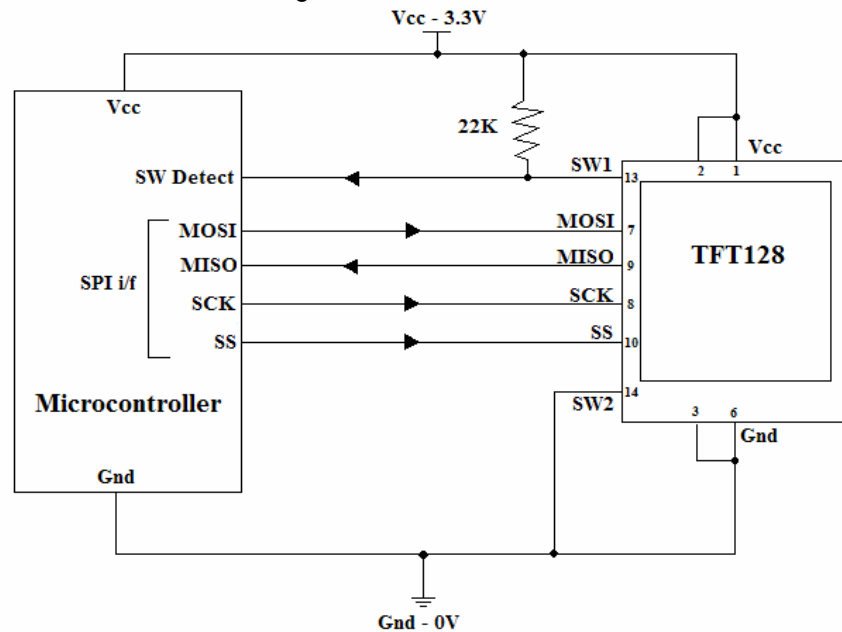
SW-1 and SW-2 are the electrically isolated switch contacts if external switch monitoring is required.

SW-DETECT is the internal switch monitoring input. If used, the SW-1 and SW-2 lines must be wired as shown in section 4.2.

D+/D- are a USB interface that can be used to update the internal firmware in the ScreenKey. A USB user interface may be made available in the future using these pins. These pins should **not** be connected.

## 6.1. Single-ScreenKey Wiring Diagram

The following circuit provides an example of how to connect a single TFT128 ScreenKey to a microcontroller with external switch monitoring:



## 6.2. Backlight Control Options

The TFT128 incorporates an internal white LED backlight. Power to this backlight is supplied via the flex-tail contacts 2 (+Vcc) and 3 (Gnd).

Typically, contact 2 should be directly connected to the main supply Vcc on contact position 1. The backlight Gnd contact (position 3) **must** always be connected to the same Gnd plane as the main supply, e.g. contact position 6.

The TFT128 allows users to adjust the backlight brightness via a SPI command when used in high-level command mode.

However, when used in high-speed mode the backlight cannot be controlled via software and is set to be always on. In this mode, it is possible for users to apply external hardware control of the backlight via contact 2 (Vcc – LED). Applying 3.3V sets the maximum brightness level. The brightness can be dimmed by reducing applied voltage downwards.

## 7. Bitmap Graphics

The display in the TFT128 supports bitmapped graphics where each pixel can display any color from a 65,536 color palette.

To define a single pixel, a two byte word is used where red and blue are specified by 5 bits each and green is specified using 6 bits:

R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B
Red					Green						Blue				
CH (MSB byte)										CL (LSB byte)					

For example,

F800 hex	Red
07E0 hex	Green
001F hex	Blue

The display resolution is 128\*128 pixels, i.e. 16,384 pixels. Using 2 bytes per pixel to define each color requires 32,768 bytes to display one full bitmap on a TFT128 ScreenKey. Clearly this requires significant bandwidth and ways to reduce this requirement are used wherever possible.

For ease of use and reduced processing overheads, the TFT128 also permits graphic updates using a form of the Windows BMP 256-color graphic format. This uses a 256 color palette where each pixel is represented by a single byte reference or index into this palette. This allows a full bitmap image to be defined using 16,384 bytes with a color palette table of 512 bytes (256 colors \* 2 bytes).

It is possible to download graphics to the TFT128 where full 16-bit color information is provided per pixel.

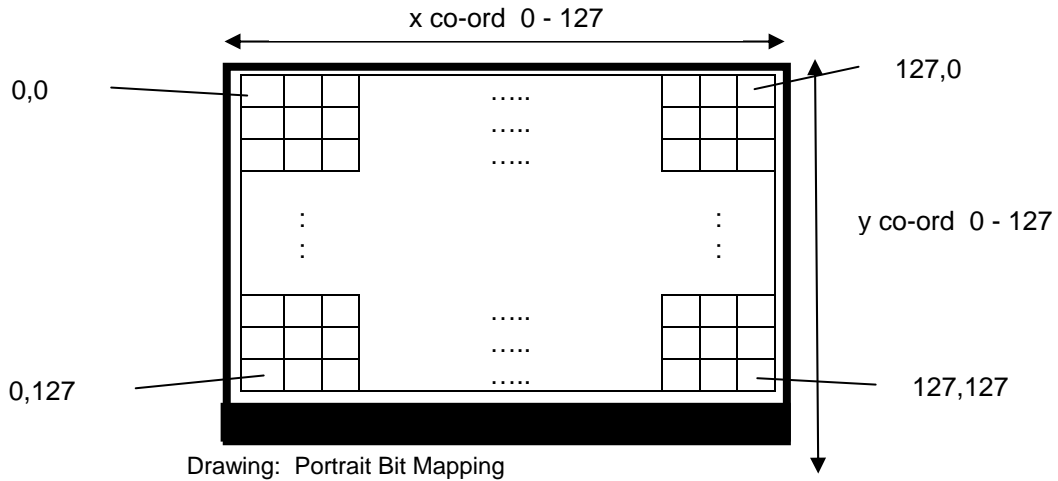
The TFT128 ScreenKey also supports a downloadable 256 color palette and downloadable graphics images that reference this 256 color palette.

All graphic commands and images are designed or positioned using an x,y co-ordinate system (see below). The 'x' co-ordinate is always the horizontal direction from the users perspective, referenced at 0 from the top left pixel, whether the key is in portrait or landscape mode. The 'y' co-ordinate is the vertical direction also referenced to 0 at the top left pixel position.

### 7.1. Portrait Bit-mapping

The visual top left corner of the display is referenced as position  $x=0, y=0$  (0,0), where  $x$  is the horizontal direction and  $y$  is the vertical direction.

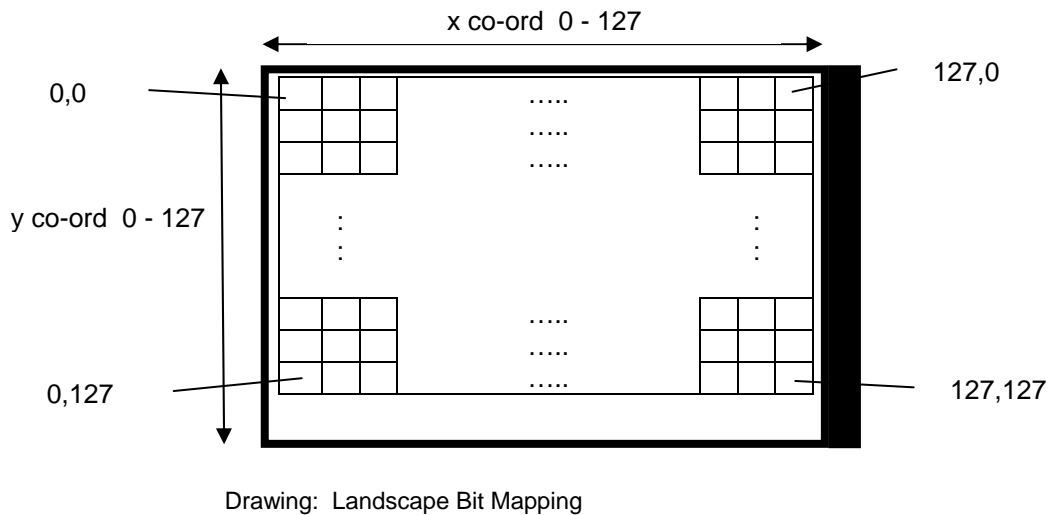
The display has 128 pixels horizontally and 128 pixels vertically.



### 7.2. Landscape Bit-mapping

In landscape mode, the visual top left corner of the display is referenced as position  $x=0, y=0$  (0,0), where  $x$  is the horizontal direction and  $y$  is the vertical direction.

The display has 128 pixels horizontally and 128 pixels vertically.



## 8. Text Display

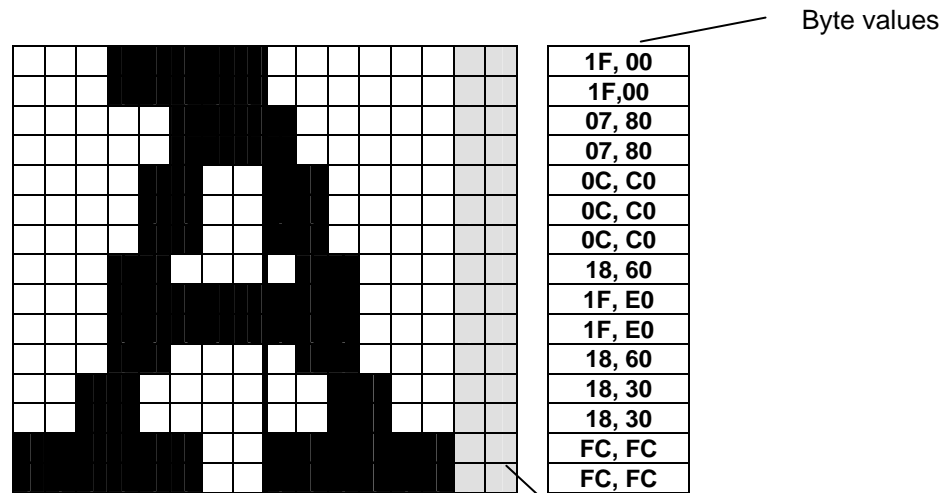
To enable fast integration and to lower bandwidth requirements, the TFT128 ScreenKey supports ASCII text display from simple commands (in *high-level command mode* only). For example, an application may send the ASCII string "Hello World" and the TFT128 will handle all font and character manipulations to generate the bit-mapped graphic image to represent this string.

The TFT128 incorporates a factory default 256 character fixed size raster font. The font is a standard "Courier" style pattern and uses a 14 column x 15 row pixel grid for each character.

The TFT128 maintains a *cursor position* which is initially set to 0,0. This relates to the top left pixel of the character to be displayed. After each font character is displayed, a set number of pixels are skipped for the inter-character spacing (inter-character spacing is set to zero pixels for the factory default font).

As characters are drawn horizontally across the screen, the TFT128 continually monitors the remaining horizontal space. If there is insufficient pixels to draw a new character (14 pixels required for the default factory font) then the cursor position is mapped to the left most position on the next character line below. A pre-set number of pixels is skipped between lines and this is called the inter-line spacing. The factory default font's inter-line spacing is set to three pixels.

For example, the factory default "A" character is defined as follows:



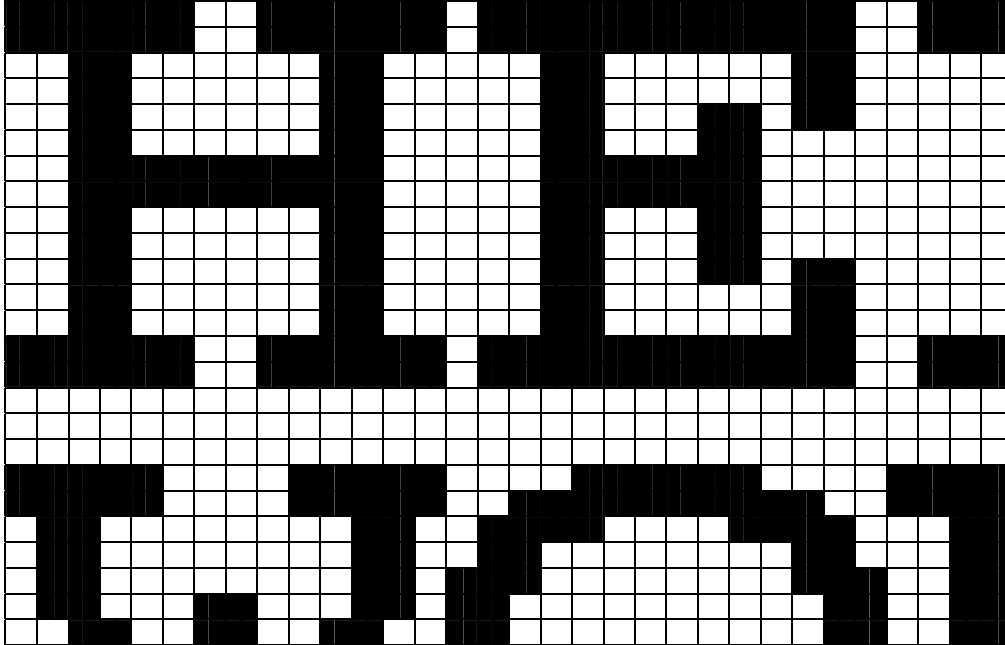
Factory default font:

- 2 bytes per row
- 14 pixels per row
- 30 bytes per character
- 0 pixels inter-character spacing
- 3 pixels inter-line spacing
- 256 number of characters in font
- 0 first character offset

Last 2 LSB's **not** used

The drawing below describes how the text strings are placed on the TFT display (using factory default font).

The string "HELLO" is requested on the first character line of the display and the text string "WORLD" on the second character line of the display.



The 3 pixel inter-line spacing is shown in the gap between the bottom row of pixels in the first line of characters and the top row of pixels in the second line of characters.

The factory default font has a zero inter-character spacing. This is because the majority of the character designs leave the first (leftmost) column of pixels clear, i.e. a default one pixel inter-character spacing is built into the 14x15 grid. There are some characters which do not leave the first column clear, i.e. the extra wide characters such as W, H, K, M etc. It is possible that these characters will 'touch' the character directly to their left. This is accepted in order to fit more characters per line.

### 8.1. Factory Default Character Set

The character set implemented by the factory default font is based primarily on the ISO 8859-15 (Latin alphabet no. 9) character set:

		High nibble															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Low nibble	0				0	@	P	`	p	-	=		°	À	Ð	à	ð
	1			!	1	A	Q	a	q			i	±	Á	Ñ	á	ñ
	2			"	2	B	R	b	r	┌	┐	¢	²	Â	Ò	â	ò
	3			#	3	C	S	c	s	└	┘	£	³	Ã	Ó	ã	ó
	4			\$	4	D	T	d	t	└	┘	€	Ž	Ä	Ô	ä	ô
	5			%	5	E	U	e	u	└	┘	¥	µ	Å	Õ	å	õ
	6			&	6	F	V	f	v	└	┘	Š	Œ	Æ	Ö	æ	ö
	7			'	7	G	W	g	w	└	┘	Š	·	Ç	×	ç	÷
	8			(	8	H	X	h	x	└	┘	š	ž	È	Ø	è	ø
	9			)	9	I	Y	i	y	└	┘	©	¹	É	Ù	é	ù
	A			*	:	J	Z	j	z	└	┘	ª	º	Ê	Ú	ê	ú
	B			+	;	K	[	k	{	▲	☺	«	»	Ë	Û	ë	û
	C			,	<	L	\	l		▶	☼	↳	Œ	Ì	Ü	ì	ü
	D			-	=	M	]	m	}	▼	♀		œ	Í	Ý	í	ý
	E			.	>	N	^	n	~	◀	♂	®	ÿ	Î	Þ	î	þ
	F			/	?	O	_	o		×	✓	—	¿	Ï	ß	ï	ÿ

Additional characters have been inserted into the “C1 Control Code” space, i.e. positions 80 hex through to 9F hex. These additional characters are useful to easily construct simple images using text commands.

There are no printable characters in positions 0 through 1F hex. User defined downloadable fonts may use this area to embed specific character designs as desired.

## 8.2. Downloadable Fonts

The TFT128 ScreenKey supports user downloadable fonts (see section below on High-Level Command Set).

When a new font is downloaded into the TFT128 memory the factory default font is deleted and cannot be recovered. The factory default font may be re-established by downloading it again via the interface command set.

When designing downloadable fonts the following information must be provided:

### *Number of characters in font*

The default factory font has 256 characters defined. To save space, sometimes character sets of 128 characters are used by leaving out the upper 128 characters. To further reserve space, the font may be designed as a reduced number of characters, e.g. 96 or even less. The only requirement is that all characters must be contiguous in the character set.

### *Offset to first character*

When a reduced size font is used (i.e. less than 256 characters) then the offset to the first font is required. This is useful if the first number of characters is not required, e.g. in printer fonts the first printable character is usually "space" which begins at position 32. To save space, the first 32 characters may be discarded in which case this offset is required to identify the starting position of the font.

### *Inter-line spacing*

Displayed text becomes difficult to read if there is insufficient spacing between characters and lines. This value defines the gap in pixels to use between the bottom of one line of text and the top row of the next line of text.

### *Inter-character spacing*

This value defines the automatic gap to place between characters horizontally. As demonstrated with the factory default font, this gap may be created in the design of the character layouts and so may be set to zero.

### *Bytes per character*

Each font is simply a contiguous collection of data bytes. This value defines the number of bytes that are assigned to one character in the font. The total size of the font must be the total of "bytes per character" X "number of characters in font".

### *Bytes per row*

This value defines the number of bytes that is used to define the pixels for a single row for a character in this font. The total number of rows for each character is "bytes per character" divided by the "bytes per row".

### *Pixels per row*

Not every bit in the data bytes for a row are used, e.g. the factory default font uses 14 bits. This value defines the number of used bits and begins with the MSB bit of the first data byte.

### *Note:*

*The TFT128 does not implement a specific baseline for handling characters which extend below the baseline, e.g. "g", "j", etc. If required, this must be handled in the inherent font design.*

## 8.3. Reserved Font Memory

Downloadable fonts are very flexible and can vary in size and content based on the parameters described above. The only limitation is the maximum physical memory size of the font.

The TFT128 reserves a memory block of 8kbytes for font usage. Downloadable fonts cannot exceed 8,196 bytes in size (contiguous data bytes).

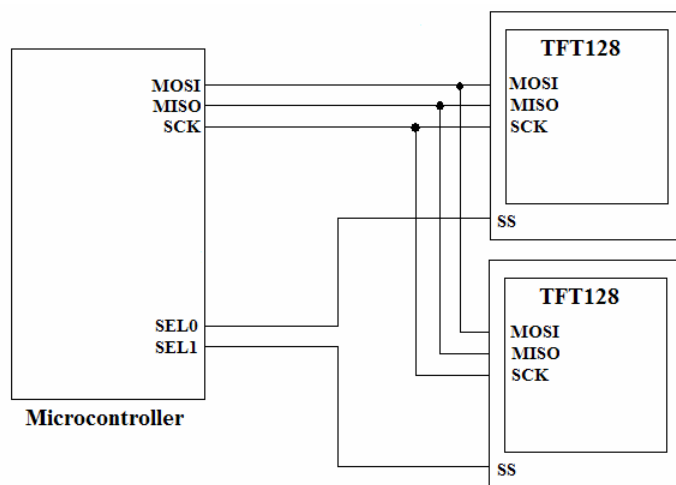
## 9. Serial Peripheral Interface

The TFT128 operates as a slave device on a Serial Peripheral Interface bus (SPI). It supports full-duplex, synchronous, serial communication with a master device at clock frequencies up to 10 MHz.

Four SPI interface signals are used:

- MOSI Master Out Slave In (data IN to ScreenKey)
- MISO Master In Slave Out (data OUT from ScreenKey)
- SCK Clock (controlled by Master)
- SS Slave Select (ScreenKey select line)

Multiple ScreenKey devices can be connected to a SPI bus as follows:



In this configuration, the MOSI, MISO and SCK signals are common to all devices. A unique SS line is required from the master to individually select one ScreenKey at a time.

All data is transmitted as 8-bit words with the most significant bit (MSB) transmitted first and the least significant bit (LSB) transmitted last.

The master device controls the clock. The TFT128 ScreenKey supports clock frequencies up to a maximum of 10MHz.

The TFT128 is a complex device supporting multiple high-level commands. Each command has a different execution time and/or different data throughputs. To support these differences, the SPI Master must listen to the return data path via MISO, to know when the slave is busy and when a command has been fully executed (unless in *high-speed mode*). Additionally, the ScreenKey has a maximum data throughput that requires the SPI Master to implement an inter-byte delay.

A minimum inter-byte delay is recommended to achieve maximum throughput for all high-level commands. If using 256-color graphic commands, a frame refresh rate of approximately four frames per second is achieved with these values.

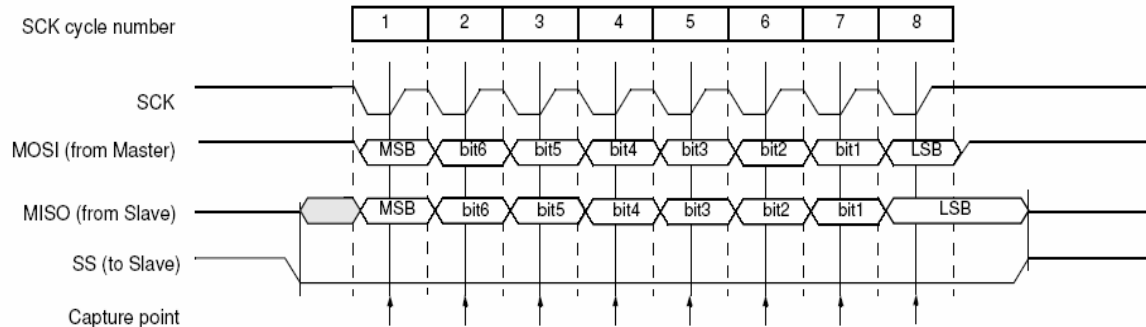
A high-speed mode is offered in which there is no data returned on MISO and the inter-byte delay is reduced significantly. This mode enables approximately ten frames per second for full color (16-bit) display. High-speed mode only provides a direct paint feature and does not allow access to the high-level commands such as text drawing, image downloads, etc. **In this mode, the MISO line may be ignored and can be left disconnected if preferred.**

External switch monitoring must be implemented if using high-speed mode.

## 9.1. SPI Transmission

The bus is controlled by the master device. The ScreenKey should first be selected ( $SS=0$ ) by the master before transmission commences. Only one ScreenKey should be selected at a time, otherwise bus conflicts may occur on the MISO line causing data corruption.

Data transmission is described in the drawing below:



Drawing: SPI bus transmission format

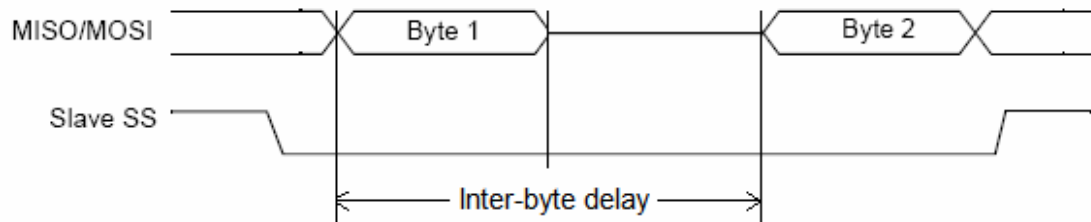
The TFT128 ScreenKey implements “mode 3” of the SPI transmission format, i.e. clock polarity (CPOL) high and clock phase (CPHA) high. The idle state of the clock line (SCK) is high and the data is sampled by the ScreenKey when the clock returns to the idle state.

To transmit a byte from the master to a ScreenKey, first the master ensures the SCK line is in its idle state (high). Next the master selects the ScreenKey by driving SS low. The clock transitions from high to low, and the master drives the MSB of the data byte on the MOSI line. At the same time the ScreenKey drives the MSB of its return data byte on the MISO line. The master then transitions the clock line back to idle (low to high) and the data is sampled by the ScreenKey and master on the MOSI and MISO lines respectively.

This is repeated for the remaining bits through to the LSB. Once all eight bits have been sent/received, the clock line remains at idle and the master deselects the ScreenKey by driving SS high.

## 9.2. Multiple Byte Transmission

When sending a package of information, i.e. multiple bytes, the master may leave the SS line low and continue to clock successive data words observing the inter-byte delay, see diagram below.



Drawing: Multiple byte transmission over SPI bus

The actual inter-byte delays for the two different operating modes are:

<b>Operating Mode</b>	<b>Inter-byte Delay</b>
High-Level Command	15.5 microseconds
High-Speed	3.3 microseconds

## 10. High-Level Command Mode

In *High-Level Command Mode* the TFT128 ScreenKey supports an extensive command interface that is designed to minimize the bandwidth requirements necessary to drive multiple ScreenKeys within the same panel.

For example, many switch displays will use text prompts extensively. High-Level Command Mode offers a text display command in which a simple ASCII string is sent to the ScreenKey and the font manipulation and rendering is handled within the key itself. Similarly, simple commands are provided to quickly blank the display to a certain color or to display small graphics in sub-windows, etc.

To support this feature, the TFT128 implements a 'soft' handshaking protocol over SPI. This is necessary as different commands can take longer to execute, other commands operate on the data as it is received while yet other commands buffer the incoming data before executing the instruction.

### 10.1. Command Structure

The master controls the TFT128 ScreenKey by issuing a command packet. The ScreenKey reports its status for each command byte transmitted. Where required, the ScreenKey can return data information if requested by the master.

Command packets sent by the master conform to the following packet structure:

<i>Header</i>				<i>Data</i>	<i>Trailer</i>		
CMD ID	XOR	LEN (MSB)	LEN (LSB)	... data ...	0x55	0xAA	0x00
byte 1	byte 2	byte 3	byte 4	byte 5 ... byte n	byte n+1	byte n+2	byte n+3

CMD ID	unique command identifier (00 – FF)
XOR	exclusive-or value of the CMD ID (00-FF)
LEN	length of data following (0000 - FFFF)
data	8-bit data bytes as necessary
trailer	3 bytes must always be sent after each <i>data</i> section (0x55 0xAA 0x00)

## 10.2. ScreenKey Status Byte

While data bytes are clocked into the ScreenKey on the MOSI line, the return information clocked out of the ScreenKey on the MISO is the **ScreenKey Status Byte**.

The ScreenKey Status Byte (SSB) is an 8-bit byte transmitted MSB first:

bit 7	SW <i>0 = switch not pressed</i> <i>1 = switch pressed</i>
bit 6	Reserved – returns 0
bit 5	Reserved – returns 0
bit 4	Reserved – returns 0
bit 3	ONLINE <i>1 = TFT128 present and online</i> <i>0 = invalid reply from ScreenKey</i>
bit 2	CMDOK <i>0 = normal state during data transfer</i> <i>1 = Command accepted and processed by ScreenKey</i>
bit 1	NACK <i>0 = Data accepted</i> <i>1 = Command/data rejected by ScreenKey</i>
Bit 0	BUSY <i>0 = Byte accepted OK</i> <i>1 = ScreenKey busy, resend current byte</i>

Note:

The ScreenKey Status Byte is **not** returned when operating in high-speed mode.

### SW (SWITCH DETECT)

The SW bit is used when *internal switch monitoring* is employed. The ScreenKey continually monitors its internal switch mechanism and reports a switch depression by setting the SW bit. This bit remains set until the master issues a *Switch Acknowledge* command. The SW bit is then cleared until the next key press is detected.

### BUSY (SCREENKEY BUSY)

During receipt of a command packet, the ScreenKey may require the master to delay the next byte transmission. The ScreenKey sets the BUSY bit to indicate this to the master. The master should continue to resend its current byte until the WAIT bit is cleared by the ScreenKey. The inter-byte delay period must be adhered to between byte resends.

Commands with longer execution times may set the BUSY bit during receipt of the *trailer* bytes. The master may proceed to service another key and return later to resend the byte to the key that requested a WAIT.

**CMDOK (COMMAND OK)**

After receiving a full command packet including the three trailing bytes, the ScreenKey will set the CMDOK bit to indicate that the command has been fully received and executed. If CMDOK bit is **not** set in the SBB returned on the third trailing byte then the command has not been received correctly and the master should resend.

If CMDOK is not set when expected, the master should resync with the ScreenKey using the resync sequence described below.

**NACK (COMMAND NOT ACKNOWLEDGED)**

The NACK bit may be set at any time during command reception. This indicates that the ScreenKey found an error or unacceptable value during command reception and has rejected the command. The ScreenKey automatically returns to waiting for the start of the next command packet. The master must decide to resend the command if required.

If NACK is set at any time, the master should resync with the ScreenKey using the resync sequence described below.

**ONLINE (SCREENKEY PRESENT AND ONLINE)**

The ONLINE bit should always be set. It may be used to detect the presence of a TFT ScreenKey on the SPI bus and to confirm that the ScreenKey is operating normally and is in high-level command mode. If this bit position returns 0 (when not in high-speed mode) then the device is not a ScreenKey or it is not operating correctly.

*Refer to communications flowchart below that describes how these bits are used to control command transmission.*

*Note that during font, graphic and color downloads the SPI interface will automatically issue 0xFF relies to the master while it stores the downloaded information into its flash memory (approx every 128 bytes). The SPI master software should recognize this as a BUSY reply from the ScreenKey and continue to reissue the last byte as per the normal busy response.*

### **10.3. Error Recovery & Resync**

In the event of a command transmission failure, the master should resync with the ScreenKey. The master can send zero-value bytes (0x00) before starting a new command to resync with the ScreenKey. The ScreenKey returns it's SSB and the master can confirm that the BUSY, CMDOK and NACK bits are all returned to 'normal', i.e. all off (0).

While the ScreenKey is not in a command reception sequence, it ignores preamble zeroes and returns its SSB to each. Receipt of a non-zero value is taken by the ScreenKey as the CMD ID of the next command.

#### **10.4. Data Information from ScreenKey**

Where commands request the ScreenKey to return information, the master must complete the full command packet including the three trailer bytes.

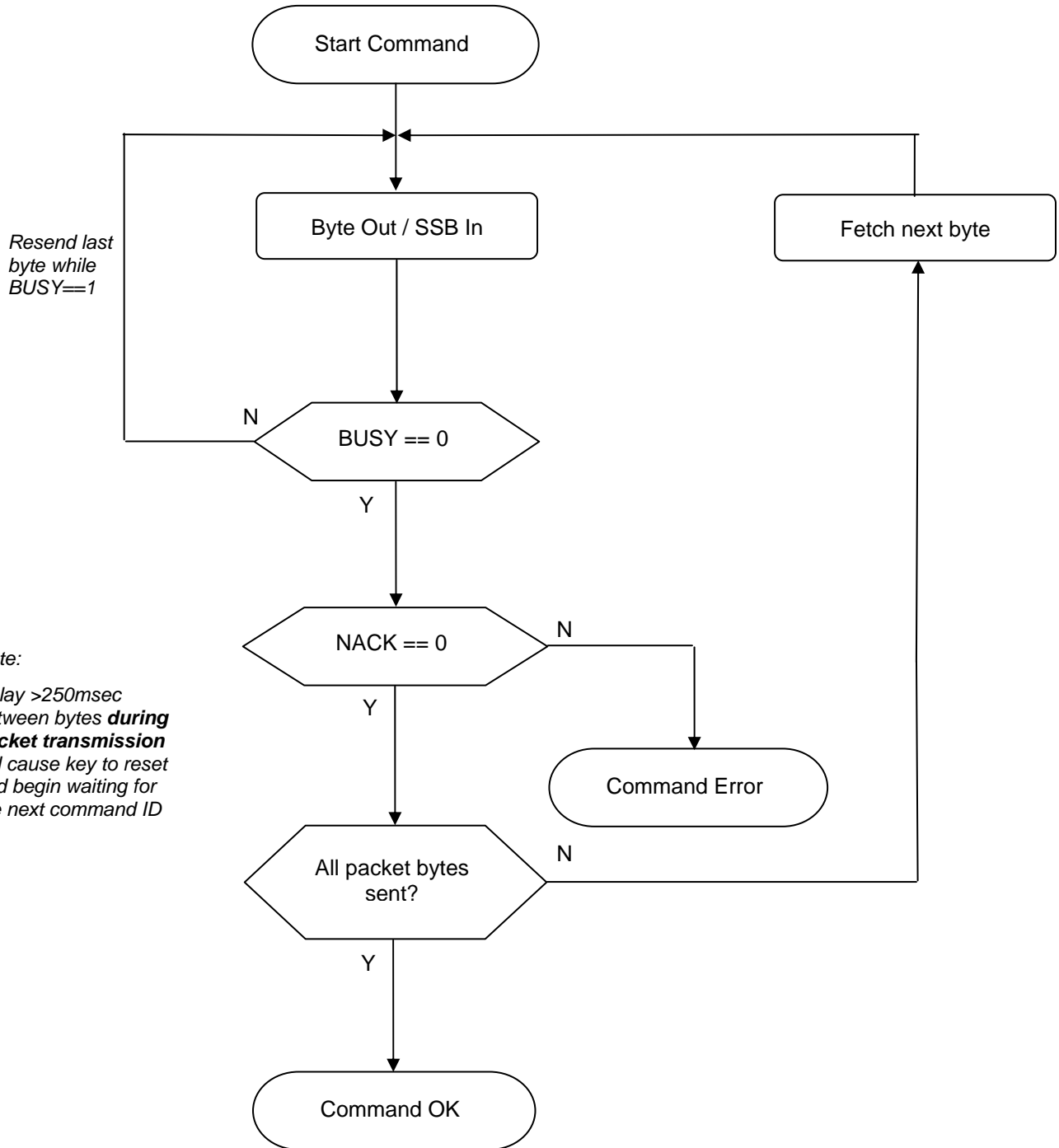
Before returning to normal processing, the master must then issue zero bytes (00hex) to the ScreenKey to trigger the return of the requested information from the ScreenKey. The master must issue the correct number of zero bytes to match the requested number of return information for that command.

#### **10.5. Inactivity Timeout**

During command packet transmission it is possible for the master to “disappear” and fail to send the full command packet or for the slave ScreenKey to believe in error that the master has not sent the full command.

This situation is detected by the inactivity timeout. Once a new command packet is initiated, the ScreenKey begins a 250 millisecond timer after each received byte. If the next byte is not received in this period, then the key discards any information received in this partial packet and it returns to waiting for a new CMD byte.

10.6. Communications Flowchart



## 10.7. Example Communication Scenarios

### COMMAND SENT AND EXECUTED SUCCESSFULLY

<i>Timeslot</i>		<b>1</b>	<b>2</b>	<i>...</i>	<b>n-2</b>	<b>n-1</b>	<b>n</b>
		CMD	XOR	<i>data</i>	0x55	0xAA	0x00
<b>BUSY</b>	0	0	0	0	0	0	0
<b>NACK</b>	0	0	0	0	0	0	0
<b>CMDOK</b>	0	0	0	0	0	0	1

The master sends the CMD byte and checks BUSY in SSB return. The ScreenKey keeps the NACK bit clear while it continues to accept data. The master sends the data bytes while always checking the BUSY bit (BUSY is set by the key to indicate that the master should wait before sending a new byte). All commands must send the three trailer bytes (0x55, 0xaa, 0x00) after the command data bytes. If all is OK with the ScreenKey, then it will set the CMDOK bit as the final 0x00 is sent from the master.

### COMMAND SENT AND EXECUTED SUCCESSFULLY BUT WAIT REQUESTED DURING TRANSMISSION

<i>Timeslot</i>		<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>n-1</b>	<b>n</b>	<b>n+1</b>
		CMD	XOR	<i>data</i>	<i>DataX</i>	<i>DataX</i> resent	0x55	0xAA	0x00
<b>BUSY</b>	0	0	0	0	1	0	0	0	0
<b>NACK</b>	0	0	0	0	0	0	0	0	0
<b>CMDOK</b>	0	0	0	0	0	0	0	0	1

The transaction begins as normal but during data transmission the key sets BUSY bit. The master continues resending the current byte (*DataX*) until BUSY is cleared by the key. The application programmer should decide how many time loops to wait before deciding that the transmission has been compromised. After BUSY is cleared the transmission then continues as normal.

### BAD DATA RECEIVED BY KEY

<i>Timeslot</i>		<b>1</b>	<b>2</b>	<i>...</i>	<b>n-2</b>	<b>n-1</b>	<b>n</b>
		CMD	XOR	<i>Data</i>	0x55	0xAA	0x00
<b>BUSY</b>	0	0	0	0	0	0	0
<b>NACK</b>	0	0	0	0	0	0	1
<b>CMDOK</b>	0	0	0	0	0	0	0

The master sends a command packet as in the first example. At the third trailer byte, the ScreenKey reports NACK to the master meaning that the command was unsuccessful. The master must decide to resend the packet or service another key.

**COMMAND NOT FULLY RECEIVED BY KEY**

<i>Timeslot</i>		<i>1</i>	<i>2</i>	<i>...</i>	<i>n-2</i>	<i>n-1</i>	<i>n</i>
		CMD	XOR	<i>Data</i>	0x55	0xAA	0x00
<b>BUSY</b>	0	0	0	0	0	0	0
<b>NACK</b>	0	0	0	0	0	0	0
<b>CMDOK</b>	0	0	0	0	0	0	0

If the master sends the full packet including trailer bytes but the ScreenKey responds without either a NACK or CMDOK, then this indicates that the ScreenKey has missed bytes during the command transmission and it still believes it has more bytes to receive.

The master can handle this scenario by allowing the inter-byte timeout (50msec) to activate and so return the ScreenKey to waiting for the next command. The master should send the zero-preamble before starting the next command to confirm that all is OK.

**MASTER FAILS TO SEND FULL COMMAND PACKET**

If the master fails to complete the sending of a full command packet then the inter-byte timeout will trigger on the ScreenKey. This will discard any partial data received and set the key into waiting for a new CMD byte.

**10.8. Screen Refresh Rate**

Using the 256-color command (see section below), each pixel is defined by a single byte value. A full screen refresh requires 128\*128 bytes, i.e. 16,384 bytes.

Using 15.5 microsecond inter-byte delay, the minimum transmission period for 16,384 bytes is 0.254 seconds.

The screen refresh rate is therefore ~4 frames per second.

## 10.9. High-Level Command Set

### Command 01h – Key Mode Reset

#### Format:

CMD ID	01h		
XOR	FEh		
LEN	00h/02h		
Data	MM	Mode	
	RR	Orientation	

#### Command Data:

MM =	00	Set operating mode to <i>high-level command</i> (default)
	01	Set operating mode to <i>high-speed</i>
RR =	00	Reset to <i>Landscape</i>
	01	Reset to <i>Portrait</i> (default)

#### Return Data:

None.

#### Description:

This command sets the ScreenKey mode:

For *high-level command* mode it restores the key to its power on state, resets all communication states and settings to their startup values and clears the display to its default background color.

For *high-speed* mode, it puts the ScreenKey into high-speed mode but does not reset any variables or other settings. When put into high-speed mode, the active window area is always set to full screen (128 \* 128 pixels).

If high-speed mode is activated with this command then the key treats all further master transmissions as 16-bit color instructions (see section 11 below).

Once high-speed mode is activated, the key remains in this mode until power-off or until the *Exit High-Speed* signal is detected (see section 11).

It is also necessary to define the orientation of the key with this command (see command 10h below).

**Default: The default mode is HIGH-LEVEL COMMAND and PORTRAIT**

## ***Command 09h – Switch Acknowledge***

**Format:**

CMD ID	09h
XOR	F6h
LEN	00h/00h

**Command Data:**

None.

**Return Data:**

None.

**Description:**

This command instructs the ScreenKey that the master has acknowledged the reported switch press. The ScreenKey resets its SW bit in the SSB until a new switch press is detected, where the ScreenKey then sets the SW bit again.

## Command 10h – Set Orientation

### Format:

CMD ID	10h	
XOR	EFh	
LEN	00h/01h	
Data	MM	<i>Orientation</i>

### Command Data:

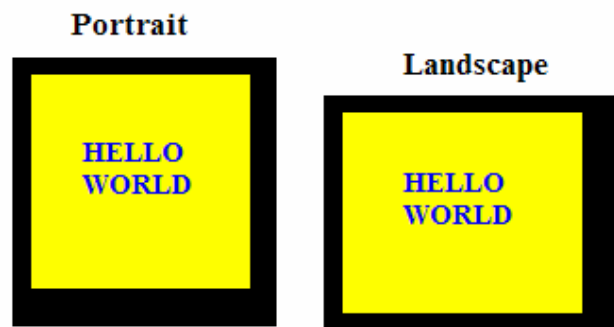
MM =	00	Set display mode to <i>landscape</i>
	01	Set display mode to <i>portrait</i> (default)

### Return Data:

None.

### Description:

The TFT can be physically mounted in a portrait or landscape orientation.



Drawing: Portrait and landscape physical orientations

Normal text and drawing commands are referenced with respect to the top left-hand corner of the key display. Physically, this differs depending on how the key is physically mounted. Once this command is issued, all future text or display commands reference themselves to the relevant top left corner.

Any currently display images or text will continue to be displayed in the same positions after this command has been issued. If used dynamically, this command would usually be followed by the *Clear Display* command.

Issuing this command will reset any flash settings (see Cmd 26h).

**Default:** *The default mode is PORTRAIT*

## Command 11h – Set Color

### Format:

CMD ID	11h	
XOR	EEh	
LEN	00h/03h	
Data	RR	<i>Color reference</i>
	CH	<i>Color value (MSB)</i>
	CL	<i>Color value (LSB)</i>

### Command Data:

RR	Color reference (position in color table 0-15)
CH/CL	16 bit color value

### Return Data:

None.

### Description:

Every pixel can be set to one of 65,536 colors, with each color definable as a 16-bit value:

R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B
Red					Green						Blue				
CH (MSB byte)										CL (LSB byte)					

The TFT128 maintains a Color Reference Table of 16 different colors:

Color Ref	Default Color	RGB Value	Color Ref	Default Color	RGB Value
0	Black	00 00	8	Dark Grey	D6 9A
1	Dark Red	D8 00	9	Red	F8 00
2	Dark Green	07 60	10	Green	07 E0
3	Dark Yellow	FF 80	11	Yellow	FF E0
4	Dark Blue	00 1B	12	Blue	00 1F
5	Dark Magenta	D8 1B	13	Magenta	F8 1F
6	Dark Cyan	0D FB	14	Cyan	07 FF
7	Light Grey	F7 9E	15	White	FF FF

This command is used to redefine the colors in the above table. Text commands reference the colors in this table for their foreground and background colors.

Simple 16-color graphics can also be drawn with reference to this color palette.

## Command 12h – Set Cursor Position

### Format:

CMD ID	12h	
XOR	EDh	
LEN	00h/02h	
Data	XX	'x' co-ordinate
	YY	'y' co-ordinate
CKSUM	CC	

### Command Data:

XX	'x' co-ordinate for new cursor position (0-127)
YY	'y' co-ordinate for new cursor position (0-127)

### Return Data:

None.

### Description:

Text commands display text on screen beginning at the current cursor position. The cursor position is updated after each character has been displayed.

This command allows the user to set the cursor position to a new co-ordinate so that finite text positioning is possible.

**Default:** *The default cursor position is 0,0*

## Command 13h – Clear Display

### Format:

CMD ID	13h	
XOR	ECh	
LEN	00h/01h	
Data	RR	Color reference

### Command Data:

RR	Index into Color Reference Table
----	----------------------------------

### Return Data:

None.

### Description:

This command blanks the full display to the specified color from the Color Reference Table. Issuing this command will reset any flash settings (see Cmd 26h).

**Default:** *The default power-on screen color is WHITE*

## Command 14h – Replace Color

### Format:

CMD ID	14h	
XOR	EBh	
LEN	00h/09h	
Data	TT	Type of color reference
	RH/RL	Color to replace
	NH/NL	New color (16-bit value)
	X1	Top left 'x' co-ord
	Y1	Top left 'y' co-ord
	X2	Bottom right 'x' co-ord
	Y2	Bottom right 'y' co-ord

### Command Data:

TT	Type of color reference: 0 = index into Color Palette Table (256 colors) 1 = actual 16-bit color RGB value 2 = index into Color Reference Table (16 colors)
RH/RL	Color to be replaced (RH = high byte always 00h if type = 0 or 2)
NH/NL	New color to use (16-bit color value)
X1,Y1	'x,y' co-ordinates of top left position (0-127,0-127)
X2,Y2	'x,y' co-ordinates of bottom right position (0-127,0-127)

### Return Data:

None.

### Description:

This command can be used to replace a specified color with a new color within a specified rectangular area of the display.

The rectangular area is defined by the x1,y1 and x2,y2 co-ordinates (top left and bottom right respectively).

Two bytes are used to define a color and the value in these bytes can be interpreted differently, depending on the value of TT:

#### **TT = 0 => Color values are indexes into the 256-color Color Palette Table**

*The color high byte (RH) will always be 00h*

*The color low byte (RL) will contain a value between 00h and FFh which is an index into the Color Palette Table*

#### **TT = 1 => Color values are actual 16-bit RGB color**

*The high/low byte combination define the full RGB value*

#### **TT = 2 => Color values are indexes into the 16-color Color Reference Table**

*The color high byte (RH) will always be 00h*

*The color low byte (RL) will contain a value between 00h and 0Fh which is an index into the Color Reference Table*

The color specified by Nh/NL is **always** the actual 16-bit RGB color value.

*Note: The active window area is automatically returned to 128\*128 after completion of this command.*

## Command 15h – Set Backlight

### Format:

CMD ID	15h	
XOR	EAh	
LEN	00h/02h	
Data	BB	<i>Backlight brightness setting</i>
	TT	<i>Backlight timeout</i>

### Command Data:

BB	Backlight brightness setting, 00h = off, FFh = full
TT	Backlight timeout setting in minutes (0 = always on)

### Return Data:

None.

### Description:

The TFT128 includes an integrated white LED backlight. This can be turned on or off and the brightness adjusted with this command. This is useful to put a panel into “sleep” mode, e.g. for energy conservation.

The ScreenKey automatically enters sleep mode after a preset timeout period and the backlight is turned off. The default timeout period is 20 minutes but this can be changed from always on (value = 0) to any period up to 255 minutes.

The brightness can be adjusted between off (00h) and full brightness (FFh).

**Default Values: Backlight ON full brightness; timeout = 20 minutes**

## Command 20h – Display Text

### Format:

CMD ID	20h	
XOR	DFh	
LEN	00h/xxh	
Data	RR	<i>Fore and back colors</i>
	BB	<i>Bit operation</i>
	...text...	<i>ASCII text</i>

### Command Data:

RR	High nibble is Color Reference Table of foreground color for text Low nibble is Color Reference Table of background color for text
BB	Bit operation to perform
text	ASCII text to display

### Return Data:

None.

### Description:

The TFT128 has an integrated fixed size font that allows ASCII text to be displayed quickly and easily. The ASCII string passed in this command is painted from left to right, starting at the current cursor position. Characters are drawn to the end of the line and continued from the left most position on the next line. This is continued until the last character position on the display is reached. All further text is discarded.

The starting cursor position is defined as 0,0 or the top left pixel position.

The foreground and background color of the text is defined in the first passed data byte (RR), where the high nibble refers to the text background color and the low nibble refers to the text foreground color. Each nibble is an index into the Color Reference Table.

b7	b6	b5	b4	b3	b2	b1	b0
Background color index				Foreground color index			

The bit operation (BB) defines how to paint the font character onto the display:

<i>Value</i>	<i>Description</i>
00h	Paint text with referenced colors from RR byte, including painting "off" pixels with specified background color
01h	Paint font "on" pixels with specified foreground color but do not perform any action for font "off" pixels
02h	Paint "on" pixels from font as 1's complement of current pixel display color. Do not perform any action for font "off" pixels.

Note that this command automatically stops any flash settings setup with Cmd 26h.

## Command 21h – Display 256-Color Graphic

### Format:

CMD ID	21h	
XOR	DEh	
LEN	xxh/xxh	
Data	XX	'x' co-ord for graphic display
	YY	'y' co-ord for graphic display
	WW	Width of graphic
	HH	Height of graphic
	...graphic...	Bitmap data

### Command Data:

XX,YY	'x,y' co-ordinates of top left position for graphic placement (0-127)
WW	Width of graphic in pixels (0-127)
HH	Height of graphic in pixels (0-127)
graphic	Graphic bitmap data (WW*HH bytes)

### Return Data:

None.

### Description:

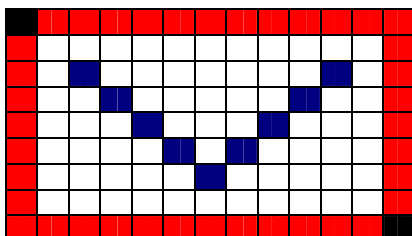
The TFT128 can light a pixel to any of 65,536 colors, as explained in section 9, where each pixel color is defined by a two-byte value. However, to display a full screen image using 2 bytes per pixel would require 32Kbytes of information. To reduce the amount of data required to define graphics, the TFT128 can work with 256 color graphics. In 256-color mode, each pixel is defined by a single byte which is a reference into a *Color Palette Table (CPT)*. The CPT holds the relevant two-byte full color codes for each unique color in the 256-color graphic. This is a similar structure as used in 256-color Windows BMP files.

The Color Palette Table is downloaded into the TFT128 using a separate command. Multiple graphics may then be designed and manipulated using the same palette.

Graphic data is transmitted row by row starting with the top left pixel.

The following abstract graphic (13 pixels wide by 9 pixels high) is defined by the data bytes to the right. These are shown in the order in which they are transmitted, i.e. left to right and top to bottom.

The data bytes shown assume a Color Palette Table based on the default Color Reference Table (see command 11), i.e. black is in position 0 in the CPT, red is in position 9, white in position 15 (0F hex) and blue in position 4.



```
00 09 09 09 09 09 09 09 09 09 09 09 09 09
09 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 09
09 0F 04 0F 0F 0F 0F 0F 0F 0F 04 0F 09
09 0F 0F 04 0F 0F 0F 0F 0F 04 0F 0F 09
09 0F 0F 0F 04 0F 0F 0F 04 0F 0F 0F 09
09 0F 0F 0F 0F 04 0F 04 0F 0F 0F 0F 09
09 0F 0F 0F 0F 0F 04 0F 0F 0F 0F 0F 09
09 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 09
09 09 09 09 09 09 09 09 09 09 09 09 00
```

Total no of  
data bytes  
= 117

Note that this command automatically stops any flash settings setup with Cmd 26h.

## Command 22h – Display Full-Color Graphic

### Format:

CMD ID	22h	
XOR	DDh	
LEN	xxh/xxh	
Data	XX	'x' co-ord for graphic display
	YY	'y' co-ord for graphic display
	WW	Width of graphic
	HH	Height of graphic
	...graphic...	Bitmap data

### Command Data:

XX,YY	'x,y' co-ordinates of top left position for graphic placement (0-127)
WW	Width of graphic in pixels (0-127)
HH	Height of graphic in pixels (0-127)
graphic	Graphic bitmap data

### Return Data:

None.

### Description:

The command allows graphics to be drawn on the TFT128 in full-color, i.e. all 65k colors can be used for each pixel.

Each pixel color is defined by two bytes:

R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B
Red					Green					Blue					
CH (MSB byte)										CL (LSB byte)					

The graphic bitmap data is transmitted in the same order as command 21, i.e. left to right and top to bottom, starting with the top left pixel.

Note that this command automatically stops any flash settings setup with Cmd 26h.

## Command 23h – Display 16-Color Graphic

### Format:

CMD ID	23h	
XOR	DCh	
LEN	xxh/xxh	
Data	XX	'x' co-ord for graphic display
	YY	'y' co-ord for graphic display
	WW	Width of graphic
	HH	Height of graphic
	...graphic...	Bitmap data

### Command Data:

XX,YY	'x,y' co-ordinates of top left position for graphic placement (0-127)
WW	Width of graphic in pixels (0-127)
HH	Height of graphic in pixels (0-127)
graphic	Graphic bitmap data

### Return Data:

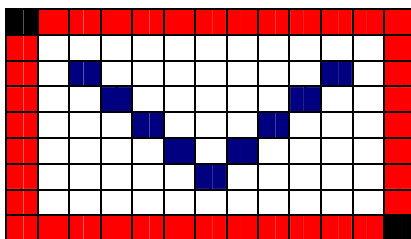
None.

### Description:

Graphical data may be reduced significantly by reducing the number of colors in an image. This is useful to reduce the bandwidth requirements of managing multiple ScreenKeys from the same controller.

The TFT128 includes a 16 color *Color Reference Table* (CRT - see command 11) which can be used as a 16-color palette. 16-color graphics can use this palette where each pixel is defined as a reference into the CRT.

As each pixel reference only requires 4 bits, one byte in the graphic data may be used to define two pixels, e.g.



```
09 99 99 99 99 99 9_
_9 FF FF FF FF FF F9
9F 4F FF FF FF 4F 9_
_9 FF 4F FF FF 4F F9
9F FF 4F FF 4F FF 9_
_9 FF FF 4F 4F FF F9
9F FF FF 4F FF FF 9_
_9 FF FF FF FF FF F9
99 99 99 99 99 99 0x
```

Total no of  
data bytes = 59

Last byte  
includes unused  
lower nibble

Note that this command automatically stops any flash settings setup with Cmd 26h.

## Command 24h – Draw Rectangle

### Format:

CMD ID	24h	
XOR	DBh	
LEN	00h/06h	
Data	X1	<i>Top left 'x' co-ord</i>
	Y1	<i>Top left 'y' co-ord</i>
	X2	<i>Bottom right 'x' co-ord</i>
	Y2	<i>Bottom right 'y' co-ord</i>
	LL	<i>Line color</i>
	WW	<i>Line width</i>

### Command Data:

X1,Y1	'x,y' co-ordinates of top left position (0-127,0-127)
X2,Y2	'x,y' co-ordinates of bottom right position (0-127,0-127)
LL	Index into Color Reference Table (0-15)
WW	Line width (1-127)

### Return Data:

None.

### Description:

This command will draw a rectangle using the specified co-ordinates. The line width can be specified from 1 pixel upwards. The specified co-ordinates are always the outer limits of the rectangle, i.e. line widths greater than 1 pixel extend towards the interior of the rectangle.

The line color (LL) passed with this command is an index into the 16-color Color Reference Table (see command 20).

This command may also be used to draw lines.

Note that this command automatically stops any flash settings setup with Cmd 26h.

## Command 25h – Draw Circle

### Format:

CMD ID	25h	
XOR	DAh	
LEN	00h/05h	
Data	X1	<i>Centre 'x' co-ord</i>
	Y1	<i>Centre 'y' co-ord</i>
	RR	<i>Radius of circle in pixels</i>
	LL	<i>Line color</i>
	WW	<i>Line width</i>

### Command Data:

X1,Y1	'x,y' co-ordinates of centre of circle (0-127,0-127)
RR	Radius of circle in pixels (1-127)
LL	Index into Color Reference Table (0-15)
WW	Line width (1-127)

### Return Data:

None.

### Description:

This command will draw a circle with a radius of RR pixels from the centre point defined by the co-ordinates x1,y1. The line width can be specified from 1 pixel upwards. Line widths greater than 1 pixel extend towards the interior of the circle.

The line color (LL) passed with this command is an index into the 16-color Color Reference Table (see command 20).

Note that this command automatically stops any flash settings setup with Cmd 26h.

## Command 26h – Set Flash

### Format:

CMD ID	26h	
XOR	D9h	
LEN	00h/xxh	
Data	X1	<i>Top left 'x' co-ord</i>
	Y1	<i>Top left 'y' co-ord</i>
	X2	<i>Bottom right 'x' co-ord</i>
	Y2	<i>Bottom right 'y' co-ord</i>
	RR	<i>Flash colors</i>
	X3	<i>Top left 'x' co-ord for text</i>
	Y3	<i>Top left 'y' co-ord for text</i>
	TT	<i>Text color</i>
	...text...	<i>Text string</i>

### Command Data:

X1,Y1	'x,y' co-ordinates of top left rectangle (0-127,0-127)
X2,Y2	'x,y' co-ordinates of bottom right rectangle (0-127,0-127)
RR	High nibble is Color Reference Table of first flash color (0-15) Low nibble is Color Reference Table of alternate flash color (0-15)
X3,Y3	'x,y' co-ordinates for text placement (0-127,0-127)
TT	High nibble is Color Reference Table of first text color (0-15) Low nibble is Color Reference Table of alternate text color (0-15)
<i>text</i>	Text to display in rectangle

### Return Data:

None.

### Description:

Flashing between two colors can be a useful effect for attracting a users attention, e.g. if an alarm condition occurs, or if a default option should be specifically highlighted, etc.

This command encapsulates the parameters required to flash a sub-section of the display between two specified colors and it can specify a text string to display within this region.

The x/y co-ordinates are provided that define the rectangular area to flash. The flash colors are set in RR, where the high nibble and low nibble are color indexes into the Color Reference Table. The display color of the defined rectangular area is continually flashed between these two colors every second. For best visual performance, the rectangular area should be kept relatively small (the full area of the display should not be used).

Usually, it will be required to also display text in this flashing area. The text location and fore/flash color is defined.

Flashing will be disabled as soon as another display command is received. The last displayed color and text will remain on screen and will be painted over by the most recent command. It is recommended to use the Clear Display command when flashing is no longer required.

Note that the flash period is automatically fixed to display each alternate image for 1 second. This period cannot be modified.

## Command 30h – Download Font

### Format:

CMD ID	30h	
XOR	CFh	
LEN	xxh/xxh	
Data	NN	<i>Number of characters in font</i>
	OO	<i>Offset to first character</i>
	LL	<i>Inter-line spacing</i>
	SS	<i>Inter-character spacing</i>
	BB	<i>Bytes per character</i>
	RR	<i>Bytes per row</i>
	PP	<i>Pixels per row</i>
	...chars...	<i>Bitmap data</i>

### Command Data:

NN	Number of characters defined in font (1-255)
OO	Offset to first character in font (0-255)
LL	Number of pixels to skip between text lines
SS	Number of pixels to skip between adjacent characters
BB	Number of bytes used to define a single character bitmap
RR	Number of bytes that define a single row of a character bitmap
PP	Number of pixels used in a single row of a character bitmap
chars	Character bitmap data

### Return Data:

None.

### Description:

Please refer to section 8 above which explains how text commands, fonts and character sets are handled in the TFT128.

The default factory font has 256 characters defined and is based on the “Courier” style layout with the ISO 8859-15 (Latin alphabet no. 9) character set (see section 8.1).

New font styles and character sets may be defined and downloaded into the TFT128 using this command.

The maximum size of a downloadable font is 8,196 bytes, including the font parameters, i.e. number of characters, inter-line spacing, etc.

Only one font can be present at the same time. The factory default font is replaced when a new font is downloaded using this command. Downloaded fonts remain in place after power down. The only way to re-establish the factory default font is to download this font again using this command.

*Note: During font download the TFT128 SPI interface will automatically issue 0xFF relies to the master while it stores the downloaded information into its flash memory. The SPI master software must recognize this as a BUSY reply from the ScreenKey and continue to reissue the last byte as per the SPI interface description in section 10.2.*

*Note: The TFT128 flash memory has a maximum of 100k write cycles.*

## Command 31h – Download Color Palette Table

### Format:

CMD ID	31h	
XOR	CEh	
LEN	02h/00h	
Data	...colors...	Color Palette Table

### Command Data:

*colors*            256 entry Color Palette Table

### Return Data:

None.

### Description:

To reduce the amount of data required to define a graphic image (and reduce the amount of bandwidth to transmit graphic images), the TFT128 can use a separate pallet of colors where the graphic data are references into this table.

The TFT ScreenKey supports 256 color images using a 256-entry Color Palette Table (CPT). Each entry in the CPT is two bytes wide, defining a unique color from the 65,536 colors supported by the display:

R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B
Red					Green						Blue				
CH (MSB byte)								CL (LSB byte)							

Only one CPT is in use at any one time. All 256-color graphic images downloaded use the current CPT for displaying. Changing the CPT after an image has been displayed has no effect on the displayed image. Any subsequent image displays will use this new CPT.

*Note: During color palette downloads the TFT128 SPI interface will automatically issue 0xFF relies to the master while it stores the downloaded information into its flash memory. The SPI master software must recognize this as a BUSY reply from the ScreenKey and continue to reissue the last byte as per the SPI interface description in section 10.2.*

*Note: The TFT128 flash memory has a maximum of 100k write cycles.*

## Command 32h – Download Graphic

### Format:

CMD ID	32h	
XOR	CDh	
LEN	xxh/xxh	
Data	WW	Width of graphic
	HH	Height of graphic
	NN	Graphic ID
	TT	Type of graphic
	...graphic...	Bitmap data

### Command Data:

WW	Width of graphic in pixels (1-127)
HH	Height of graphic in pixels (1-127)
NN	Graphic ID (1-127)
TT	Graphic type – 00 = 256-color, 01 = full-color, 02 = 16-color
graphic	Graphic bitmap data

### Return Data:

None.

### Description:

The TFT128 has limited storage available for downloaded images (max 10 images). Using downloaded images can improve response times in multi-key systems by removing the need to constantly download new images to changing circumstances. Downloaded images are easily recalled using a simple command.

When downloading images, the height and width of the image must be provided as well as the type. Different image types have different storage requirements and pixel mappings:

256-color image	One byte per pixel	Each byte is index into Color Palette Table
Full-color image	Two bytes per pixel	Two bytes are actual RGB color value
16-color image	One nibble per pixel	Each nibble is index into Color Reference Table

Refer to the display Image commands for full descriptions of the actual structure of image data for each type.

Each download must have a unique numeric graphic ID as this number is used in Command 33 to recall the downloaded image. It is important to ensure that there is sufficient memory space to store the graphic image before downloading. The available remaining space is obtained using Command 34 (see below).

*Note: This command does **not** immediately display the downloaded image. Command 33 must be used to display the image that is downloaded with this command.*

*Note: During graphic downloads the TFT128 SPI interface will automatically issue 0xFF relies to the master while it stores the downloaded information into its flash memory. The SPI master software must recognize this as a BUSY reply from the ScreenKey and continue to reissue the last byte as per the SPI interface description in section 10.2.*

*Note: The TFT128 flash memory has a maximum of 100k write cycles.*

## Command 33h – Recall Graphic

### Format:

CMD ID	33h	
XOR	CCh	
LEN	00h/05h	
Data	XX	'x' co-ord for graphic display
	YY	'y' co-ord for graphic display
	NN	Graphic ID

### Command Data:

XX,YY	'x,y' co-ordinates of top left position for graphic placement
NN	Unique graphic ID

### Return Data:

None.

### Description:

This command is used to recall images that have been downloaded previously using Command 32. The *Graphic ID* (specified in Command 32) is used to uniquely identify the image to be recalled.

Note that this command automatically stops any flash settings setup with Cmd 26h.

## Command 34h – Report Free Graphic Memory

### Format:

CMD ID	34h
XOR	CBh
LEN	00h/00h

### Command Data:

None.

### Return Data:

Data	MLh	<i>LSB of available free memory</i>
	MHh	<i>MSB of available free memory</i>

### Description:

This command is used to check that the ScreenKey has sufficient free space to accept a new graphic download. The ScreenKey returns an integer value (2 bytes) with the amount of free space remaining in the key.

*Note: The TFT128 has approx 9,200 bytes available for general purpose user storage.*

## Command 35h – Report Memory Contents

### Format:

CMD ID	35h	
XOR	CAh	
LEN	00h/04h	
Data	X1	<i>Top left 'x' co-ord</i>
	Y1	<i>Top left 'y' co-ord</i>
	X2	<i>Bottom right 'x' co-ord</i>
	Y2	<i>Bottom right 'y' co-ord</i>

### Command Data:

X1,Y1	'x,y' co-ordinates of top left position (0-127,0-127)
X2,Y2	'x,y' co-ordinates of bottom right position (0-127,0-127)

### Return Data:

Data	CHh	<i>MSB of pixel color</i>
	CLh	<i>LSB of pixel color</i>

### Description:

This command requests the actual graphic data in the specified region (i.e. x1,y1 to x2,y2). The ScreenKey reports this information pixel by pixel, starting with the top left pixel and moving left to right and top to bottom until all pixel information has been reported. The return information is two bytes per pixel which reports the exact pixel color as follows:

R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B
Red					Green					Blue					
CH (MSB byte)										CL (LSB byte)					

This command may be used to extract some specific graphic from the ScreenKey display. This could then be manipulated and then rewritten to the same location using command 22.

*Note: The maximum allowed requested number of return bytes is 512. Multiple commands may be issued if more than 512 bytes are required.*

## Command 36h – Report Product Version

### Format:

CMD ID	36h
XOR	C9h
LEN	00h/00h

### Command Data:

None

### Return Data:

Data	PP	<i>Product Type</i>
	VV	<i>Version Number</i>

### Description:

This command requests the product type and version number. The ScreenKey reports this information as two bytes:

PP	Product Type	<i>always 01h for TFT128</i>
VV	Version Number	<i>current version 10h</i>

## 10.10. Programming Example

This example describes how to display the text "Hello World" with blue foreground color and yellow background on a TFT128 in portrait mode.

Issue reset to key on power up, set orientation to portrait

```
0x01 0xFE 0x00 0x02 0x00 0x01 0x55 0xAA 0x00
```

*Always issue key reset after power on*

Clear display to "yellow" background color

```
0x13 0xEC 0x00 0x01 0x0B 0x55 0xAA 0x00
```

*Clear Display uses Color Reference Table (yellow = 11 dec = 0B hex)*

Set cursor position to 26,37

```
0x12 0xED 0x00 0x02 0x1A 0x25 0x55 0xAA 0x00
```

*Reset positions cursor position to 0,0 but need text to be centered on screen*

*Need to set new cursor position to ensure text is located as desired*

Display text "Hello"

```
0x20 0xDF 0x00 0x07 0xCB 0x01 0x48 0x65 0x6C 0x6C 0x6F 0x55 0xAA 0x00
```

*Best performance is achieved if only the "on" pixels are painted on screen*

*Use bit operation 01 to achieve this*

Set new cursor position to 26,73

```
0x12 0xED 0x00 0x02 0x1A 0x49 0x55 0xAA 0x00
```

*Normal cursor position updates to next character or beginning of new line*

*Sometimes cursor position needs to be explicitly set for desired formatting*

Display text "World"

```
0x20 0xDF 0x00 0x07 0xCB 0x01 0x57 0x6F 0x72 0x6C 0x64 0x55 0xAA 0x00
```

*Use bit operation 01 to only write "on" pixels for best performance*

This example demonstrates how WAIT is handled:

Set new cursor position to 26,73

```
0x12 0xED 0x00 0x02 0x1A 0x49 0x49 0x49 0x55 0x55 0x55 0xAA 0x00
```

BUSY set by key so current byte resent until BUSY=0

BUSY set by key so current byte resent until BUSY=0

## 11. High-Speed Mode

In *High-Speed Mode* the TFT128 ScreenKey offers a refresh rate of over 10 frames per second. It does not offer any of the high-level commands as described above. High-speed mode provides direct access to the graphic area of the ScreenKey display and all rendering is the responsibility of the application or system integrator.

In High-Speed Mode, there is no data or status information returned from the ScreenKey. The data that is placed on MOSI line by the master appears directly on the MISO line in the next byte transfer sequence. If only using the key in this mode, the MISO line may be left unconnected.

### 11.1. Activating High-Speed Mode

By default, on power-on, the TFT128 starts in high-level command mode. To activate high-speed mode, it is necessary to issue the following command sequence:

CMD ID	Header			Data		Trailer		
	XOR	LEN (MSB)	LEN (LSB)	mode	orientation	byte1	byte2	byte3
0x01	0xFE	0x00	0x01	0x01	0xRR	0x55	0xAA	0x00

The value of 0xMM defines the required operating mode of the key:

- 0x00 High-Level Command Mode
- 0x01 High-Speed Mode

The value of 0xMM defines the orientation of the display (see section 3.3 above):

- 0x00 Landscape orientation
- 0x01 Portrait orientation

The orientation defines the position of the top left pixel position which is the starting point from where the the graphic display will be populated.

During transmission of this command, the ScreenKey will return the ScreenKey Status Byte until the final trailer byte (byte3) is sent. Thereafter the data received from the master will be reflected back on the MISO line.

An inter-byte delay of 15 microseconds should be observed while transmitting this sequence.

## 11.2. Sending Graphical Data

As already stated, high-speed mode only supports direct display of graphical data on the ScreenKey's display. Each full screen image requires:

$$128 \text{ pixels wide by } 128 \text{ pixels high by } 16\text{-bit pixel color (2 bytes)} = 32,768 \text{ bytes}$$

The start of a new frame display command is detected by the SS (Slave Select) line being activated. This resets the cursor position to the top left pixel position (referenced to the defined orientation).

Each byte transferred to the key updates the 16-bit color definition for each pixel in turn, i.e. first the top line is drawn left to right, then the next line (left to right) and so on until the last line is drawn (see also section 7).

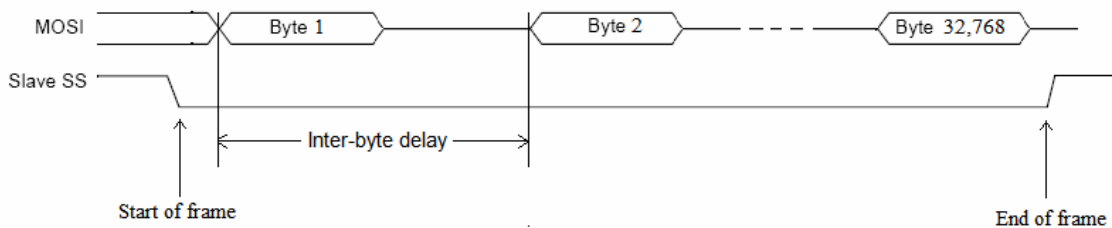
The SS line should only be raised after all bytes for the current display frame have been sent.

## 11.3. Frame Synchronization

As there is no return feedback from the key, it is critical that the ScreenKey and master have a method for synchronization of frames. This is achieved by the SS (Slave Select) line.

The ScreenKey identifies when the SS line is raised (deselecting the current key) and recognises this as an end of frame. The cursor position is automatically returned to the top left pixel position in preparation for the next frame transmission.

The master must ensure that the SS line is raised and lowered between each frame transmission:



Drawing: High-speed full display frame transmission

Between frames, the SS line must be held high for a minimum period of 5 microseconds.

## 11.4. Screen Refresh Rate

Each pixel is defined by a two-byte value (16-bit color). A full screen refresh requires  $128 \times 128 \times 2$  bytes, i.e. 32,768 bytes.

Using 3.3 microsecond inter-byte delay, the minimum transmission period for 32,768 bytes is 0.108 seconds.


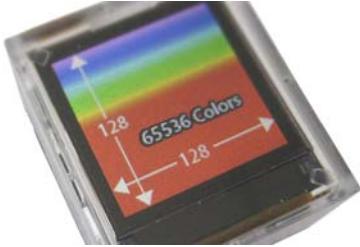
The screen refresh rate is therefore ~10 frames per second.

### ***11.4. Exit High-Speed Mode***

It is possible to exit high-speed mode and return to high-level command mode. To do this, the master should raise the SS line (deselect the ScreenKey), then lower the SS line (reselect key) but do not send any data. Then again raise the SS line to deselect the key. The ScreenKey is then restored to high-level command mode and will respond to normal commands including returning the SSB.

Always issue 00h bytes to the key before sending a new command to ensure the SSB is being returned correctly.

## 12. Order Information

Order No.	Description
TFT128-B	TFT128 ScreenKey with full-color 128*128 display resolution – BLACK 
TFT128-C	TFT128 ScreenKey with full-color 128*128 display resolution – CLEAR 

## 13. Contact Information

For further information on TFT ScreenKeys, RGB and LC Series ScreenKeys and other information, including technical documentation, datasheets, user manuals, software downloads, development and prototyping tools, please visit our website at: [www.ScreenKeys.com](http://www.ScreenKeys.com) or email us at [info@screenkeys.com](mailto:info@screenkeys.com).

**SK Interfaces Ltd**  
**Unit 11, Keypoint Business Park,**  
**42 Rosemount Park Drive,**  
**Ballycoolin Road, Dublin 11, Ireland.**  
**Tel: +353-1-88 55 075**  
**Fax: +353-1-88 55 095**

## Important Notice

### *Warranty Disclaimer*

SK INTERFACES LIMITED GRANTS NO WARRANTY WITH RESPECT TO THIS DATA SHEET, NEITHER EXPLICIT NOR IMPLIED, AND IT IS NOT LIABLE FOR DIRECT OR INDIRECT DAMAGES. SOME STATES DO NOT GRANT THE EXCLUSION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES AND, THEREFORE, THIS STATEMENT MAY NOT BE VALID IN SUCH CASES.

### *Copyright Notice*

© 2008 - Copyright SK Interfaces Limited. All rights reserved. No part of this publication may be copied, photocopied, reproduced, translated or reduced to any electronic medium or machine readable form without the expressed written consent from SK Interfaces Limited.

This data sheet has been produced with all due care. However, since errors cannot be excluded, SK Interfaces Limited does not grant any warranty or accept any legal responsibility or liability in any form for erroneous statements herein.

### *General Notice*

This data sheet is intended for technically qualified personnel trained in the field of electronics.

The knowledge and technically correct implementation of the content of this data sheet are required for problem free installation and safe operation of the described product. Only qualified personnel has the required know how to implement the specifications given in this data sheet.

For clarity, not all details regarding the product or its implementation, installation, operation, or maintenance have been included. Should you require additional information, please contact SK Interfaces Limited or visit our website at [www.ScreenKeys.com](http://www.ScreenKeys.com).