# ScreenKey Console
# Low-Level Interface

## Programmer's Guide 5.1

November, 2006

Firmware versions:

5.1 (8-Bit)

1.0 (16-Bit)

**www.ScreenKeys.com**

## ScreenKey Low-Level Interface

Programmer's Guide

Information in this document is subject to change without notice. The latest revisions may be accessed on the SKI Web site.

Web: www.ScreenKeys.com

Technical Support is available

via Email support@screenkeys.com

via Web: www.screenkeys.com



DISCLAIMER:

SKI reserves the right to revise data file formats and functionality at any time.

# Table of Contents

I N T R O D U C T I O N

# ScreenKey Low-Level Interface Overview

## Introduction

ScreenKey controller boards shorten development timeframes by allowing developers to concentrate on the higher level aspects of the application rather than spending time and effort on ScreenKey driver hardware and low-level software to handle internal ScreenKey registers, key scanning, character sets, etc.

This document describes the command set supported by SKI's ScreenKey consoles and controller boards. Application developers can use this interface to control a ScreenKey console with simple and straightforward commands, e.g. display text "abc" on the ScreenKey at row 1, column 1.

It is relatively straightforward to develop a simple interface to drive a ScreenKey console from a host or single-board computer using this command set. This enables ScreenKey technology to be utilised quickly and efficiently in any host environment.

If the console is used on a Windows platform then a range of software drivers are available to further reduce the developer's burden by removing the need to deal with the PC's communications port. In this instance the developer does not have to understand the low-level command set. Windows drivers are described in their own documentation set.

Writing software to handle the low level interface between the host computer and the ScreenKey console requires an experienced programmer. This document assumes that the reader has the necessary level of expertise.

## Console Specifications

SKI provides a range of different ScreenKey consoles. Each console can support a different set of devices, including ScreenKeys, fixed keys, magnetic card readers (MCR), keylock (also called modelock), beeper, etc.

This document presents the **full** low-level interface command set. Each ScreenKey console only implements the relevant commands dependent on the actual devices that it has integrated. Please refer to the specific console specification or technical reference manual for details of the hardware devices that it supports.

## 8-Bit/16-Bit Variations

ScreenKey consoles are designed using either an 8-bit 80C320 CPU (**e.g. OEM-5400**) or else using a pseudo 16-bit 80C251 CPU (**e.g. SK-7000**). This distinction is important as the command set implemented by each differs slightly.

**Note** 

In this document pointing hands in the left margin are used to indicate differences between the 8-bit and 16-bit firmware implementations. For example, if the relevant information only applies to the 8-bit firmware this will be stated, similarly if the section only applies to the 16-bit version. Otherwise, if there is no hand used, the information applies to both versions.

 An upward pointing hand indicates that the whole chapter only applies to the description below the pointing hand.

 A rightward pointing hand indicates a difference in that section if the hand is pointing to a section or a difference in the paragraph if the hand is pointing to a paragraph.

## Interfaces

All ScreenKey consoles implement a standard RS-232C 4-wire hardware handshaking protocol to communicate with the host computer.

*Note: SKI propose to release consoles with USB interfaces in the future.*

## Hardware Connection

Each ScreenKey Console requires two separate connections:

- Serial data connection to the host computers RS-232C port
- Power connection to a suitable power supply

Some consoles are supplied with a separate power bracket that allows the console to utilise a standard host internal power supply.

*Note: Some consoles use 5V dc while other consoles require 12V dc. It should be noted that approx 1Amp @ 5V dc is required per panel of 12 ScreenKeys attached to a ScreenKey Console.*

The relevant specification or technical reference manual gives full details of cable connections and power requirements for each ScreenKey Console.

## Software Updates

From time to time, SKI will issue new firmware updates. These can be downloaded into the consoles memory using either the Update ROM command, see page 37 (8-bit firmware), or the Download File command, see page 57 (16-bit firmware).

*Note: The latest 8-bit ScreenKey Consoles support flash memory updates which are stored on the onboard EEPROM. The SK-7000 does **not** support this feature. System designs should incorporate the ability to always download the latest firmware version on start-up of the console.*

## Sample Code

Sample code, in C, is available to illustrate how to interface to the ScreenKey Console. Sample code can be accessed from the SKI Web Site (www.ScreenKeys.com).

## Disclaimer

SKI reserves the right to revise this specification, at any time. New firmware releases are identified by a Version Number, which can be accessed electronically via the low-level interface. Code written to interface to a ScreenKey Console should check the Firmware Version No. (see the ROM-ID Message on page 69).

H A R D W A R E

# ScreenKey Console Initialization

## Power-on Self Test (POST)

When power is applied to the ScreenKey Console it carries out an internal hardware self-test. This test includes RAM, interrupt system, processor operation, etc. If all is well it follows the Initialisation Sequence described below.

**8-bit** ☞ If all is well the console will beep for 1/10$^{th}$ of a second and follow the Initialization Sequence described below. If there is a problem the console issues a continuous Beep. This indicates that a hardware failure has occurred.

**16-bit** ☞ 16-bit CPU consoles halt execution and display the error number on the top-left ScreenKey.

## Initialization Sequence

After successful completion of the POST, the ScreenKey console continues with the initialisation sequence:

**8-bit** ☞

- The EEPROM is accessed (if present) and checked for firmware. If present this is downloaded into RAM, and the keyboard resets itself. If there is no firmware loaded into the EEPROM initialization continues.
- The most recent character set is downloaded into RAM.
- The EEPROM is scanned for downloaded images.

**All** ☞

- Set up and enable all interrupts and timers
- ScreenKeys: Display a blank, green screen.
- Beeper: issues a single beep.

The initialisation sequence continues with the ScreenKey console waiting for contact from the host computer.

If successful the ScreenKey console is now in *On-line mode i.e.* it is waiting to receive commands from the host or send ScreenKey console keypresses to the host.

If a key is pressed when power is being applied to the console, it will enter into *Diagnostics mode*.

**8-bit** ☞

## Diagnostics Mode (8-bit CPU)

Diagnostics Mode is used to check that the ScreenKey console is functioning correctly. The procedure is as follows...

1. Connect the power lead and switch on power. Press and hold any one of the ScreenKeys when power is applied (or within 5 seconds of applying power).
2. The top three ScreenKeys display the test selection: “Key No”, “Color” and “Pixel”.
3. The bottom left key displays “Exit”. Press this to exit *Diagnostics Mode* and return to normal operation.
4. Selecting “Key No” allows the key scanning to be tested. In this test, pressing any key displays the internal key number. On each iteration, the backlight of the ScreenKeys is cycled through the different selections. Press the lower left “Exit” key twice to return to the top menu.
5. Selecting “Color” allows the LED backlighting on the ScreenKeys to be tested. Press the lower right key (showing the text “Color” repeatedly to cycle through the 16 available colours. This test is used to see if there are any unusual colors that would indicate a ScreenKey LED failure. Press the lower left “exit” key once to return to the top menu.
6. Selecting “Pixel” allows the individual pixels of the ScreenKeys to be examined. Press the lower right key “Pixel” to repeatedly cycle through a selection of different pixel patterns. This test is used to see if any pixels are faulty. Press the lower left “exit” key once to return to the top menu.
7. Exit Diagnostics Mode by pressing the lower left “Exit” key when in the top menu.

**16-bit** ☞

## Diagnostics Mode (16-bit CPU)

To enter Diagnostics Mode Mode on 16-bit CPU consoles:

1. Connect the power lead and switch on power. Press and hold any one of the ScreenKeys when power is being applied.
2. The first menu page of diagnostics tests is displayed on the top line of ScreenKeys. Press the ScreenKey to activate the test referenced, e.g. select "Key Press" test to check if all ScreenKeys are operational.
3. Pressing any key in "Key Press" test displays the key number on the ScreenKeys (check this against the Key Numbering Diagrams on page 86).
4. In "Key Press" test, all ScreenKeys cycle through the different backlight colors.
5. Press the "Exit" key to stop the selected test (depending on the test the "Exit" key may have to be pressed twice).
6. In the top level diagnostics menu, the bottom row of ScreenKeys includes a next and back key (lower right and left ScreenKeys). Use these keys to cycle between the different diagnostics menus.
7. The "ScreenKeys" test displays pixel patterns and also cycles through the available backlight colors. Press the lower left ScreenKey to change the pixel pattern, the lower middle ScreenKey to change the backlight color, and the lower right ScreenKey to exit the test.
8. The other diagnostics tests are self-explanatory.

# Hardware Reset

**8-bit** ☞

Some consoles include the ability to perform a hardware reset using the serial interface.

The DTR output line from the PC is used to reset the ScreenKey console. If DTR is set Hi the ScreenKey console is held reset. When a PC is first switched on the DTR line is usually Hi holding the console in reset state. To perform a hardware reset programmatically, the driver software must first set DTR line Hi (output a 0 to bit 0 of the Modem Control Register) to ensure the console is held Reset then, say, 100ms later, set the DTR line Lo (output a 1 to bit 0) to remove the Reset from the ScreenKey console.

# The ScreenKey Console Serial Interface

## Serial Interface

The ScreenKey console presents a 4-wire RS232 serial interface to communicate between it and a host computer with minimal risks of data loss or data corruption.

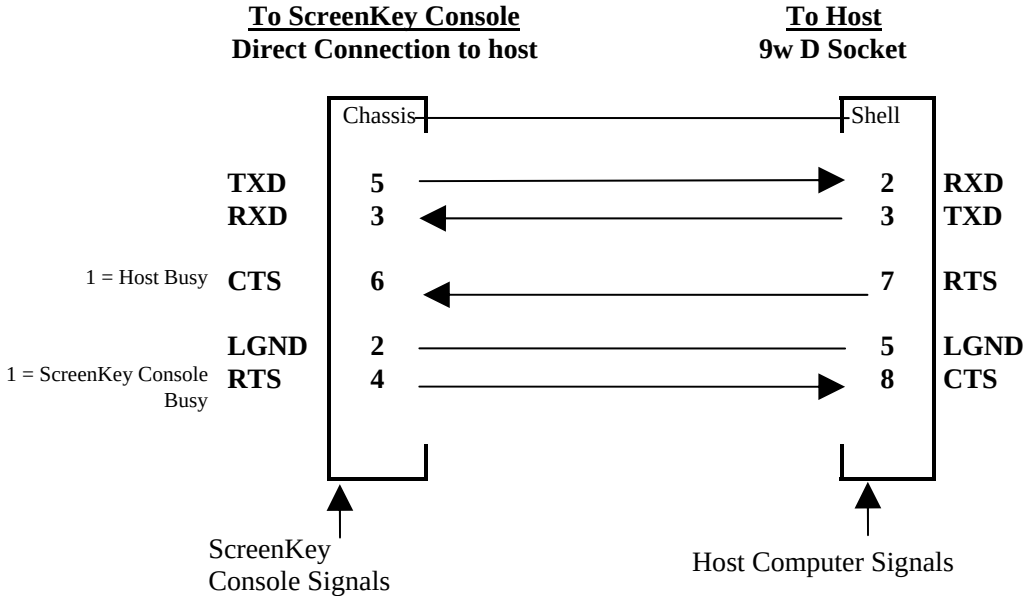
Two lines are used for transmit (Tx) and receive (Rx). Another two lines are provided for hardware handshaking (flow control).

The console implements a standard RTS/CTS method of serial data flow control.

*Note: Earlier versions of consoles from Rapid Technology (RTI) used a non-standard interface method referred to as 'blipping'. This required special cable wiring. Please contact SK Interfaces for information on this interface for cable requirements and how to implement this unique software protocol.*

## Serial Cable Wiring Diagram

The data cable shown below demonstrates the connection of a ScreenKey console to an RS232 9way D-type socket. The pin out of the cable is...



# Serial Interface Protocol

The ScreenKey console is set to communicate at 57600 baud, 1 Start Bit, 8 Data-Bits, 1 Stop Bit and No Parity at full duplex.

The Host Computer-to-ScreenKey Console link utilises a standard "RTS/CTS" protocol. The host computer must check the status of its CTS line to see if the ScreenKey console is "Ready", before sending each byte. Similarly, the ScreenKey console checks the status of its CTS line before it sends data back to the host.

The ScreenKey console sets its CTS to indicate “busy” when its internals buffers are close to full. The host must not send data when the console sets its CTS to busy as data may be lost.

The command protocol is structured using packets. The ScreenKey console ‘waits’ for all bytes of a command to be received before processing the command. Typically the console will wait up to 2 seconds between bytes. If there is no serial activity within this period then the console will reject any bytes received until then and return to waiting for a new command.

C O M M A N D S

# ScreenKey Console Commands

The host computer sends commands to the ScreenKey console. A command consists of...

- Command No. byte
- Command No. byte XOR'ed
- zero, one or more data bytes
- Checksum byte

### Command No.

The Command No. identifies the Command.

### Command XOR

The XOR'ed byte is used for validation.

### Checksum Byte

All commands (except Command 29h - Update ROM – 8-bit CPU consoles only) have a checksum byte appended. The sum of all bytes in the command, including the checksum byte, should add up to A5h.

# List of Commands

| | | | |
|---|---|---|---|
| 3Bh | Set ScreenKey Initialisation Registers (debug) | 8-bit | |
| 40h | Download Reset | 16-bit | |
| 41h | Download Information Request | 16-bit | |
| 42h | Download Update Init | 16-bit | |
| 43h | Download Data Update | 16-bit | |
| 44h | Download Update Complete | 16-bit | |
| 45h | Download Dump Data | 16-bit | |

# Command 01h - Software Reset

**Format:**

| | |
|---|---|
| Command No. | 01h |
| Command XOR | FEh |
| Checksum | A6h |

**Functionality:**

This Command resets the ScreenKey console. The ScreenKey console will go through the normal initialisation process. The **ROM ID Message** (see page 69) will be sent to the host. The ROM ID contains the version number of the ROM.

This is a software reset. If the ScreenKey console is ignoring commands from the host for some reason then this command will be ignored as well.

The Software Reset can be used to cause the ScreenKey Console to exit the built-in diagnostics.

The host should send a Reset command at initialisation.

If the the host has a new firmware file to download it should do so using the appropriate download commands:

- Use **Command 29h** (see page 37) to download to an 8-bit CPU ScreenKey console
- Use **Command 40h** through **Command 45h** to download to a 16-bit CPU ScreenKey console. This requires command/event interaction as described in pages 57 to 62 below.

# Command 03h - Request ACK

**Format:**

| | |
|---|---|
| Command No. | 03h |
| Command XOR | FCh |
| Checksum | A6h |

**Functionality:**

This Command requests the ScreenKey Console to send an ACK message. This is use to check if communications between the host and the ScreenKey console are still working.

See also ACK Message on page 72.

# Command 04h - Request KeyLock Status

**Format:**

| | |
|---|---|
| Command No. | 04h |
| Command XOR | FBh |
| Checksum | A6h |

**Functionality:**

Requests the console to send the current KeyLock position. It will return a number as per the Key Numbering Diagram on page 86.

The KeyLock operates like a stuck key.

The status is returned in a Key Close Message.

# Command 05h - Don't send Key Opens

**Format:**

| | |
|---|---|
| Command No. | 05h |
| Command XOR | Fah |
| Checksum | A6h |

**Functionality:**

Request the ScreenKey Console not to return Key Opens. This is the default. See Command 06h.

When a key is pressed both pressing the key (key close) and releasing the key (key open) may result in a message to the host.

# Command 06h - Send Key Opens

**Format:**

| | |
|---|---|
| Command No. | 06h |
| Command XOR | F9h |
| Checksum | A6h |

**Functionality:**

Request the ScreenKey Console to send both the Key Closes and the Key Opens. See Command 05h.

# Command 07h - Disable "Beep" per keypress

**Format:**

| | |
|---|---|
| Command No. | 07h |
| Command XOR | F8h |
| Checksum | A6h |

**Functionality:**

Request the ScreenKey Console not to beep when a key is pressed or the key in the KeyLock is turned.

# Command 08h - Enable "Beep" per keypress

**Format:**

| | |
|---|---|
| Command No. | 08h |
| Command XOR | F7h |
| Checksum | A6h |

**Functionality:**

Request the ScreenKey Console to beep when a key is pressed or the key in the KeyLock is turned. This is the default.

# Command 0Dh - Track-1 mandatory

# Command 0Eh - Track-2 mandatory

# Command 0Fh - Track-3 mandatory

# Command 10h - Track-1 OR Track-2 Mandatory

# Command 11h - Track-1 OR Track-3 Mandatory

# Command 12h - Track-2 OR Track-3 Mandatory

# Command 13h - Track-1 OR Track-2 OR Track-3 Mandatory

**Format:**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Command No. | 0Dh | 0Eh | 0Fh | 10h | 11h | 12h | 13h |
| Command XOR | F2h | F1h | F0h | EFh | EEh | EDh | ECh |
| Checksum | A6h | A6h | A6h | A6h | A6h | A6h | A6h |

**Functionality:**

Specify which, if any, Magnetic Stripe Reader tracks are mandatory.

The ScreenKey Console will make a decision about the data read from the card and issue a good or bad beep accordingly. By allowing the application to tell the ScreenKey Console which track(s) it needs we avoid the situation where the ScreenKey Console has issued a good beep for the card only for the application to reject it.

With one exception, these Commands are “mutually exclusive” i.e. the most recent Command sent to the host overrides earlier Commands sent.

The exception is that Commands 0Dh, 0Eh and 0Fh may be combined with each other. For example, sending Command 0Dh and 0Eh would make Track-1 AND Track-2 mandatory i.e. the ScreenKey Console would reject any card that didn’t have both these tracks on it.

The default is Command 13h--any track or combination of tracks is OK.

# Command 14h - Request ScreenKey Console Beep

**Format:**

| | |
|---|---|
| Command No. | 14h |
| Command XOR | EBh |
| Checksum | A6h |

**Functionality:**

Request the ScreenKey Console to issue a 0.1 second beep.

# Command 17h - Request ROM ID

**Format:**

| | |
|---|---|
| Command No. | 17h |
| Command XOR | E8h |
| Checksum | A6h |

**Functionality:**

Request the console to return the ROM ID. See the ROM ID message on page 69 for details.

# Command 1Ch - Request Configuration Data

**Format:**

| | |
|---|---|
| Command No. | 1Ch |
| Command XOR | E3h |
| Checksum | A6h |

**Functionality:**

Request the ScreenKey Console to return the Configuration Data. See the Configuration Data Message on page 70 for details.

# Command 1Dh - Turn all or specified ScreenKeys ON

**Format:**

| | |
|---|---|
| Command No. | 1Dh |
| Command XOR | E2h |
| Key No. | xxh |
| Checksum | xxh |

**ScreenKey Display Numbering Scheme**

**Functionality:**

Set all pixels on the specified ScreenKey(s) ON. If *Key No*. is zero then all available ScreenKeys are updated otherwise only the Specified ScreenKey is updated. The colour doesn't change.

See the Key Numbering Diagram on page 86 for valid *Key No*. values.

# Command 1Eh - Turn all or specified ScreenKeys OFF

**Format:**

| | |
|---|---|
| Command No. | 1Eh |
| Command XOR | E1h |
| Key No. | xxh |
| Checksum | xxh |

**ScreenKey Display Numbering Scheme**

**Functionality:**

Set all pixels on the specified ScreenKey(s) OFF. If *Key No*. is zero then all available ScreenKeys are affected otherwise only the specified ScreenKey is. The colour doesn't change.

See the Key Numbering Diagram on page 86 for valid *Key No*. values.

# Command 1Fh - Change Colour on all or specified ScreenKeys

**Format:**

| | |
|---|---|
| Command No. | 1Fh |
| Command XOR | E0h |
| Key No. | xxh |
| Colour Code | xxh |
| Checksum | xxh |

**ScreenKey Display Numbering Scheme**

**Functionality:**

Change the colour on all specified ScreenKey(s). If *Key No*. is zero then all available ScreenKeys are affected otherwise only the specified ScreenKey is. The pixels don't change.

See the Key Numbering Diagram on page 86 for valid *Key No*. values.

See Command 25h - Display Data on ScreenKey for valid *Colour Code* values.

# Command 21h - Display LC16 Data on ScreenKeys

**Format:**

| | |
|---|---|
| Command No. | 21h |
| Command XOR | Deh |
| Key No. | Xxh |
| Graphic Data | xx xx xx xx xx xx xx xx xx xx xx xx ..... xx xx xx |
| Checksum | Xxh |

**ScreenKey Display Numbering Scheme**

**Functionality:**

Display LC16 data on the ScreenKey(s). If *Key No*. is zero then display on all available ScreenKeys--see the Key Numbering Diagram on page 86 for valid *Key No*. values.

LC16 *Graphic Data* is always 64 Bytes and is sent using "Compiled" format (see Command 25h - Display Data on ScreenKey and page **Error! Bookmark not defined.** for details).

This Command is a limited version of Command 25h.

# Command 22h - Download 7x5 Character Set

**Format:**

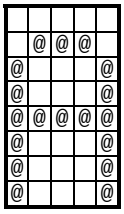
| | |
|---|---|
| Command No. | 22h |
| Command XOR | DDh |
| CharacterSet | xxh xxh xxh xxh xxh<br>xxh xxh xxh xxh xxh<br>-- -- -- --<br>xxh xxh xxh xxh xxh<br>xxh xxh xxh xxh xxh |
| Checksum | xxh |

**Functionality:**

Download a 7x5 Character Set to the ScreenKey Console and enable it. The character set is made up of 256 entries. Each entry represents one character and is made up of 5 bytes.

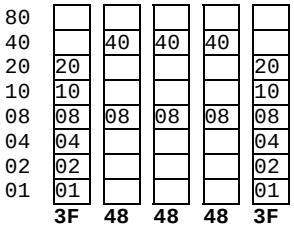
*NOTE: there is no equivalent command to request the ScreenKey Console to return the Character Set that it is using to the host.*

The process of converting the bitmap into 5 bytes can be illustrated by taking the 7x5 representation for a upper case 'A' as an example.



Note: The top row of the character is usually left blank: it provides the separation between the top and bottom lines of text on the ScreenKey display.

Taking each column in turn, perform the sum as illustrated... The result for our sample character is 3Fh, 48h, 48h, 48h, 3Fh.



# Command 23h - Enable Default 7x5 Character Set

**Format:**

| | |
|---|---|
| Command No. | 23h |
| Command XOR | DCh |
| Checksum | A6h |

**Functionality:**

Tell the ScreenKey Console to use the default 7x5 character set. This has the effect of cancelling out Command 22h if one has been issued. The ScreenKey Console reverts to using its built-in Character Set.

# Command 24h - Clear the MSR Bit-Buffers

**Format:**

| | |
|---|---|
| Command No. | 24h |
| Command XOR | DBh |
| Checksum | A6h |

**Functionality:**

This command is used for debugging the MSR handling in the ScreenKey Console.

It clears the area of External RAM which the ScreenKey Console uses for storing data from the MSR.

The idea is that this command is used to reset the buffers to a fresh start, then a card is swiped and then Command 1Ah is used to retrieve the contents of the buffer for analysis.

# Command 25h - Display Data on ScreenKey

**Format:**

| | |
|---|---|
| Command No. | 25h |
| Command XOR | DAh |
| Key No. | xxh |
| Colour | xxh |
| FlashColour | xxh |
| Flags | xxh |
| Length | xxh |
| Display Data | xx xx xx xx xx xx xx xx xx xx xx xx ..... xx xx xx |
| Checksum | xxh |

**ScreenKey Display Numbering Scheme**

**Functionality:**

Display Text or Graphic on a ScreenKey(s). If *Key No*. is zero then all available ScreenKeys are updated--see the Key Numbering Diagram on page 86 for valid *Key No*. values.

*Colour* and *FlashColour* can be any value from 0 to 15 (see below).

The *Flags* are...

| | |
|---|---|
| 40h | 1 = Flashing is Requested |
| 20h | 1 = Graphic, 0 = Text |
| 10h | 1 = Text Centered |
| 08h | 1 = Graphic data has been "Compiled" to pixel layout for faster console processing |

*Colour* always sets the foreground colour of the ScreenKey.

*FlashColour* sets the alternate flashing colour when the “Flashing” bit is set in *Flag*.

*Length* specifies the total number of bytes in *Display Data*.

This command can be used to send data (text or graphics) to LC16 or LC24 ScreenKey consoles. LC16 keys have a resolution of 32 pixels across by 16 rows and LC24 are 36 pixels across by 24 rows. See page 94 for a description of pixel addressing.

If this command is sending Graphic data then the *Length* must be 64 for LC16 graphic images and 108 for LC24 images. If Text is being sent then the *Length* can be 1 to 10 for LC16 ScreenKEys and 1 to 18 for LC24 ScreenKeys.

*Note:*

*LC16 ScreenKeys support two lines text of max 5characters per line.*
*LC24 ScreenKeys support three lines of text of max 6 characters per line.*

The "Compiled" Flag (08h) is used when sending graphics.

The mapping between the graphic data stream and the actual pixel positions on the LCD display of the ScreenKey is different for LC16 and LC24. Page 94 describes these mappings in detail.

LC24 graphic data **must** always be forwarded to the console in "compiled" or "ScreenKey ready" format. This is necessary for efficient handling in the console and to prevent unnecessary delays caused by pixel manipulations.

LC16 graphic data may be forwarded in either compiled or uncompiled ("Logical Pixel Addressing) formats. This is possible because the difference is simply the orientation of the bits within each byte (see page 94 for more detail).

The "Compiled" *flag* is used to indicate the format of the included graphic data.

ScreenKey consoles support either RG (red-green) ScreenKeys or RGB (red-green-blue) ScreenKeys. When specifying the colour of the ScreenKey backlight the following tables should be used:

RG and RGB ScreenKeys:

| Code | Description |
|---|---|
| 0 | OFF |
| 1 | Dark RED |
| 5 | Bright RED |
| 2 | Dark GREEN |
| 10 | Bright GREEN |
| 3 | Dark ORANGE |
| 7 | Bright ORANGE |
| 14 | Greenish ORANGE |
| 9 | Reddish ORANGE |

RGB ScreenKeys only:

| Code | Description |
|---|---|
| 4 | Dark BLUE |
| 6 | Bright BLUE |
| 8 | PINK |
| 11 | Dark PURPLE |
| 12 | Bright PURPLE |
| 13 | TORQUOISE |
| 15 | WHITE |

# Command 27h - Set SLEEP Parameters

**Format:**

| | |
|---|---|
| Command No. | 27h |
| Command XOR | D8h |
| Flag | xxh |
| Delay Lo/Hi | xxh xxh |
| Checksum | xxh |

**Functionality:**

Set the parameters used by the ScreenKey Console to control the sleep mode.

After a specified period of ScreenKey Console inactivity (i.e. no user input) the ScreenKeys will go to sleep--the ScreenKey colour changes to OFF.

Setting *Flag* to 00h disables sleep mode; any non-zero value enables sleep mode.

*Delay* is in units of 1/20th of a second. For example, if you want the ScreenKey Console to go to sleep after 5 minutes of inactivity then set the *Delay* to 6000 (5 * 60 * 20).

# Command 28h - Set FLASH Parameters

**Format:**

| | |
|---|---|
| Command No. | 28h |
| Command XOR | D7h |
| TimeOn Lo/Hi | xxh xxh |
| TimeOff Lo/Hi | xxh xxh |
| Checksum | xxh |

**Functionality:**

Set the parameters used by the ScreenKey Console to control Flashing colours on the ScreenKey display.

The ScreenKey is shown with the normal colour for *TimeOn* time and with the *Flash Colour* (see Command 25h) for *TimeOff* time.

Both *TimeOn* and *TimeOff* are in units of 1/20th of a second.

For example, setting *TimeOn* to 10 and *TimeOff* to 20 will show the normal colour for half a second and the flash colour for one second.

# Command 29h - Update the ROM in the Console

**8-bit** ☞

**Format:**

| | |
|---|---|
| Command No. | 29h |
| Command XOR | D6h |
| Offset HI/LO | xx xx |
| Length HI/LO | xx xx |
| Code Bytes | xx xx xx xx xx xx xx xx xx xx xx xx ..... xx xx xx |
| | xx xx xx xx xx xx xx xx xx xx xx xx ..... xx xx xx |
| | xx xx xx xx xx xx xx xx xx xx xx xx ..... xx xx xx |

**Functionality:**

Command 29h allows the code firmware in ROM to be updated by downloading code into memory in the ScreenKey Console and transferring control to this code. There is no *Checksum* with this command.

SKI from time to time releases an update firmware binary file (usually called EXRAM.BIN) that contains the *Offset*, *Length* and *Code Bytes*--i.e. all bytes needed for this command except the *Command No*. and the *Command XOR*.

*Offset* is the address in ScreenKey Console memory where the code is be stored.

*Length* is the number of bytes in *Code Bytes*.

Download will reset the ScreenKey Console. This means that the ScreenKey Console will do the standard initialisation which includes returning the new ROM ID and version number to the host.

By comparing the version number found in the ROM ID message with the version number held in EXRAM.BIN it is possible to determine whether there is a need to update the ROM or not. The version number is held in EXRAM.BIN at offset 2E/2F hex (major at 2E / minor at 2F). The number is stored in ASCII, e.g. 2E/2F = '52' means version 5.2.

*Notes:*

*ROM 5.0 does **not** support softwareupdates via command 29h. Please contact SKI to have a replacement ROM shipped to you if you have this version.*

*When Downloading to a ROM 4.0 and earlier, the ScreenKey Console beeper is turned on at the start; off at the end. ROM 4.1 and later ScreenKey Consoles dispense with this beep.*

# Command 2Ah - Disable specified key attribute(s)

**Format:**

| | |
|---|---|
| Command No. | 2Ah |
| Command XOR | D5h |
| Key Number | xxh |
| Flags | Xxh |
| Checksum | Xxh |

**Logical Key Numbering Scheme**

**Functionality:**

Function 2Ah turns off the specified attribute(s) of the specified key. The attributes that can be controlled are Beep, Data Returned, ScreenKey Display, and Repeat Key.

This Command is related to Commands 07h and 08h (Disable/Enable Beep per keypress) and also Command 2Ch which sets the Typematic rate for the repeating keys.

For example, use Command 07h to disable beeps per keypress for all keys and then use Command 2Bh to enable beeps for selected keys.

The *Flags* are...

| | |
|---|---|
| 80h | 1 = Disable Beep attribute (0 = no change) |
| 40h | 1 = Disable Return Data Attribute (0 = no change) |
| 20h | 1 = Disable ScreenKey Display Attribute (0 = no change) |
| 10h | 1 = Disable Repeat Key Attribute (0 = no change) |

The default values are: Beeps on, do Return Data, do display ScreenKeys, don't repeat keys.

The command 2Ah and 2Bh is intended to work with ScreenKeys and fixed keys only, not with the KeyLock. If a value above 93 (the highest valid key number) is given in the *Standard Key Number* parameter then the command is applied to all keys on the console as well as to the KeyLock. The problem is that if a K*ey Number* greater than 93 is used to set repeat, the KeyLock will also repeat, which means it will generate KeyLock changes constantly.

# Command 2Bh - Enable specified key attribute(s)

**Format:**

| | |
|---|---|
| Command No. | 2Bh |
| Command XOR | D4h |
| Key Number. | xxh |
| Flags | xxh |
| Checksum | xxh |

**Logical Key Numbering Scheme**

**Functionality:**

Function 2Bh turns on the specified attribute(s) of the specified key. The attributes that can be controlled are Beep, Data Returned, ScreenKey Display and Repeat Key.

This Command is related to Commands 07h and 08 (Disable/Enable Beep per keypress) and also Command 2Ch which sets the Typematic rate for the repeating keys.

For example, use Command 07h to disable beeps per keypress for all keys and then use Command 2Bh to enable beeps for selected keys.

The *Flags* are...

| | |
|---|---|
| 80h | 1 = Enable Beep attribute (0 = no change) |
| 40h | 1 = Enable Return Data Attribute (0 = no change) |
| 20h | 1 = Enable ScreenKey Display Attribute (0 = no change) |
| 10h | 1 = Enable Repeat Key Attribute (0 = no change) |

The default values are: Beeps on, do Return Data, do display ScreenKeys, don't repeat keys.

The command 2Ah and 2Bh is intended to work with ScreenKeys and fixed keys only, not with the KeyLock. If a value above 93 (the highest valid key number) is given in the *Standard Key Number* parameter then the command is applied to all keys on the console as well as to the KeyLock. The problem is that if a K*ey Number* greater than 93 is used to set repeat, the KeyLock will also repeat, which means it will generate KeyLock changes constantly.

# Command 2Ch - Set REPEAT KEY Parameters

**Format:**

| | |
|---|---|
| Command No. | 2Ch |
| Command XOR | D3h |
| Delay Lo/Hi | xxh xxh |
| Speed Lo/Hi | xxh xxh |
| Checksum | Xxh |

**Functionality:**

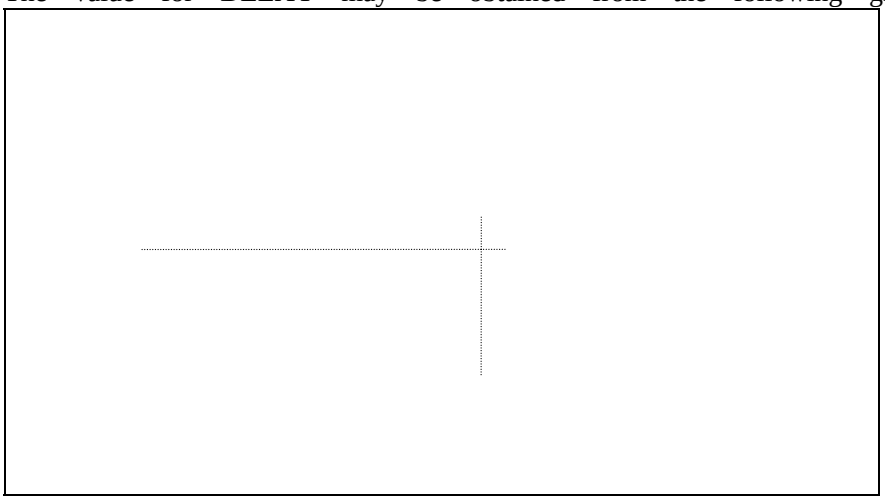
Set the parameters used by the ScreenKey Console to control the Typematic Rate and Delay of a Repeating key.

This Command is related to Commands 2Ah and 2Bh (Disable/Enable the repeatability for each individual key).

For example, use Command 2Bh to enable of a particular key as repeatable, then use Command 2Ch to adjust the repeat rate for the keys.

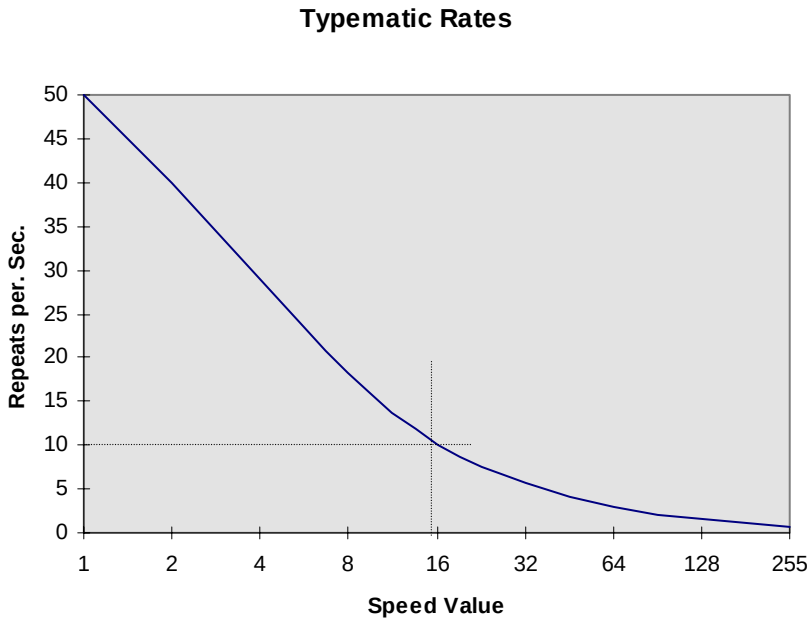
This function mimics the standard Repeat Key operation of a QWERTY console. This has two variables, the first is the initial DELAY before the key starts repeating, while the second variable is the SPEED at which the key repeats i.e. the Typematic Rate.

The value for DELAY may be obtained from the following graph



For example, a Delay value of 128 gives a delay of approximately two thirds of a second between the initial key press and the first repeat. This is the default value. Any value will be accepted but the valid values are 1 to 255.

The value for the SPEED at which the keys are returned to the host may be obtained using the following graph:




For example, a Speed value of 16 gives 10 repeats per second. This is the default value. Any value will be accepted but the valid values are 1 to 255.

# Command 2Dh – Turn OFF LED Indicators(s)

**Format:**

| | |
|---|---|
| Command No. | 2D |
| Command XOR | D2 |
| LEDIndicatorBits | Xxh |
| Checksum | Xxh |

**Functionality:**

This command turns OFF the specified LED Indicator (see also Command 2Eh).

The *LEDIndicatorBits* are (as they correspond to the LED Indicators on an IBM Console with Wait as the left-most LED Indicator)...

| | |
|---|---|
| 01h | 1 = Wait |
| 02h | 1 = OffLine |
| 04h | 1 = Message Pending |
| 08h | 1 = <UnNamed> |

Set the bit(s) in *LEDIndicatorBits* to '1' to turn OFF the LED Indicator(s).

The default values are: All LED Indicators off.

If this command is sent to a console that does not have LED Indicators then they will be ignored. No error message will be returned and no error beep will sound.

# Command 2Eh – Turn ON LED Indicator(s)

**Format:**

| | |
|---|---|
| Command No. | 2E |
| Command XOR | D1 |
| LEDIndicatorBits | xxh |
| Checksum | xxh |

**Functionality:**

This Command turns ON the specified LED Indicator (see also Command 2Dh).

Set the bit(s) in *LEDIndicatorBits* to '1' to turn ON the LED Indicator(s).

See Command 2Dh for more information.

# Command 2Fh - Disable specified key / KeyLock attribute(s)

**Format:**

| | |
|---|---|
| Command No. | 2Fh |
| Command XOR | D0h |
| Key No. | xxh |
| Flags | xxh |
| Checksum | xxh |

**Logical Key Numbering Scheme**

**Functionality:**

Function 2Fh is a replacement for Function 2Ah and should be used in preference to it.

Function 2Fh turns off the specified attribute(s) of the specified key. The attributes that can be controlled are Beep, Data Returned, ScreenKey Display, and Repeat Key.

This Command is related to Commands 07h and 08h (Disable/Enable Beep per keypress) and also Command 2Ch which sets the Typematic rate for the repeating keys.

For example, use Command 07h to disable beeps per keypress for all keys and then use Command 2Bh to enable beeps for selected keys.

The *Flags* are...

| | |
|---|---|
| 80h | 1 = Disable Beep attribute (0 = no change) |
| 40h | 1 = Disable Return Data Attribute (0 = no change) |
| 20h | 1 = Disable ScreenKey Display Attribute (0 = no change) |
| 10h | 1 = Disable Repeat Key Attribute (0 = no change) |

The default values are: Beeps on, do Return Data, do display ScreenKeys, don't repeat keys.

# Command 30h - Enable specified key / KeyLock attribute(s)

**Format:**

| | |
|---|---|
| Command No. | 30h |
| Command XOR | CFh |
| Key No. | xxh |
| Flags | xxh |
| Checksum | xxh |

**Logical Key Numbering Scheme**

**Functionality:**

Function 30h is a replacement for Function 2Bh and should be used in preference to it.

Function 30h turns on the specified attribute(s) of the specified key or KeyLock. The attributes that can be controlled are Beep, Data Returned, ScreenKey Display and Repeat Key.

This Command is related to Commands 07h and 08 (Disable/Enable Beep per keypress) and also Command 2Ch which sets the Typematic rate for the repeating keys.

For example, use Command 07h to disable beeps per keypress for all keys and then use Command 2Bh to enable beeps for selected keys.

The *Flags* are...

| | |
|---|---|
| 80h | 1 = Enable Beep attribute (0 = no change) |
| 40h | 1 = Enable Return Data Attribute (0 = no change) |
| 20h | 1 = Enable ScreenKey Display Attribute (0 = no change) |
| 10h | 1 = Enable Repeat Key Attribute (0 = no change) |

The default values are: Beeps on, do Return Data, do display ScreenKeys, don’t repeat keys.

# Command 31h – Specify Card Reader Type

**Format:**

| | |
|---|---|
| Command No. | 31h |
| Command XOR | CEh |
| Type Code | xxh |
| Checksum | xxh |

**Functionality:**

This command allows the user to tell the ROM what type of Card Reader is installed. If this command is not used the ROM defaults to inferring which Track the data came from by looking at the data itself. This can result in misdiagnosing the track if the track data doesn't conform to the rules below.

The *Type Codes* are...

| | |
|---|---|
| 1 | Track 1 only Reader |
| 2 | Track 2 only Reader |
| 3 | Track 3 only Reader |
| 4 | Track 1 & 2 Reader |
| 5 | Track 2 & 3 Reader |

**Track characteristics**

The sort of information available to us from which to infer the type of data include…

| | Track 1 | Track 2 | Track 3 |
|---|---|---|---|
| Maximum no. of bytes | 79 | 40 | 107 |
| Bits per byte (excluding Parity) | 6 | 4 | 4 |
| STX | 45h | 0Bh | 0Bh |
| Number of Separators | n/a | at most 1 | at least 3 |

# Command 32h – Specify Event Notification Mode

**Format:**

| | |
|---|---|
| Command No. | 32h |
| Command XOR | CDh |
| Type Code | xxh |
| Checksum | Xxh |

**Functionality:**

This command allows the user to tell the ROM what type of handling is required for event notification. Events are notified by Beeps and /or Event (Error) Codes returned to the host.

If this command is not used the ROM defaults to Default Mode.

The *Type Codes* are...

1 Default Mode – keep compatibility with previous ROM releases

2 Application Mode – return more event info to the application

In Default Mode the console is responsible for handling events and does not return many codes to the application. In Application Mode the console sends a wider range of event codes to the Application.

# Command 33h – Write Text to ODA

**Format:**

| | |
|---|---|
| Command No. | 33h |
| Command XOR | CCh |
| Line number | xxh |
| Text | xx xx xx xx xx xx xx xx xx xx xx xx xx xx … xx xx |
| Checksum | xxh |

**Functionality:**

This command displays the text provided on the specified line of the *Operator Display Area* (ODA). This command is only valid with consoles that have an integrated large-panel LCD, e.g. SK-7000. The ODA is a dedicated 2x20 section of this display.

The *Line numbers* are...

0 Write specified text to BOTH lines
1 Write specified text to TOP line only
2 Write specified text to BOTTOM line only

*Text* is a fixed 20 bytes and is the ASCII data to be displayed on the ODA. It should be padded with 0's to the end if the text is less than 20 bytes.

The specified line(s) of the ODA is cleared if *Text* contains all 0's, i.e. no data.

# Command 34h – Write Text to CDA

**Format:**

| | |
|---|---|
| Command No. | 34h |
| Command XOR | CBh |
| Text | xx xx xx xx xx xx xx xx xx xx xx xx xx xx … xx xx |
| Checksum | xxh |

**Functionality:**

This command automatically activates the pop-up single-line *Control Display Area* (CDA) and displays the text provided. This command is only valid with consoles that have an integrated large-panel LCD, e.g. SK-7000. The CDA is a pop-up 1x20 section of this display.

*Text* is a fixed 20 bytes and is the ASCII data to be displayed on the CDA. It should be padded with 0's to the end if the text is less than 20 bytes.

The CDA is cleared if *Text* contains all 0's, i.e. no data, but will remain visible. Use Command 35h – Select CDA to hide the CDA when no longer required.

The CDA is off by default. When displayed, it reduces the operating size of the TDA to 8 lines (instead of its normal 12). The TDA returns to normal when the CDA is turned off.

Typically, the CDA is used to provide "alert" or "warning" messages to the operator, e.g. in a retail POS environment, the CDA may be used to instruct the operator to check the bottom of the trolley at the end of a transaction.

# Command 35h – Select CDA

**Format:**

| | |
|---|---|
| Command No. | 35h |
| Command XOR | CAh |
| OnOff | xxh |
| Checksum | xxh |

**Functionality:**

This command is used to turn the display of the CDA on (visible) or off (hidden). The command is only valid with consoles that have an integrated large-panel LCD, e.g. SK-7000. The CDA is a pop-up 1x20 section of this display.

The CDA is turned off (hidden) by default. When turned on it displays the text last written to the CDA using Command 34h – Write Text to CDA.

# Command 36h – Write Text to TDA

**Format:**

| | |
|---|---|
| Command No. | 36h |
| Command XOR | C9h |
| PageNumber | xxh |
| Text | xx xx xx xx xx xx xx xx xx xx xx xx xx xx … xx xx |
| Checksum | xxh |

**Functionality:**

This command writes the supplied text to the specified page of the *Transaction Display Area* (TDA). This command is only valid with consoles that have an integrated large-panel LCD, e.g. SK-7000. The TDA is a dedicated 12x40 section of this display.

The TDA is effectively a "window" for displaying information contained in the console memory. The console maintains 1 or more buffers (pages) that can be written to using this command (a maximum of 4 buffers is supported). The console operator scrolls up and down through the text on a selected page using the scroll buttons below the TDA. The currently visible page is determined by Command 37h – Select TDA.

*PageNumber* refers to a specified page or "0" can be used to refer to the currently displayed page.

*Text* is a fixed 40 bytes and is the ASCII data to be displayed on the TDA. It should be padded with 0's to the end if the text is less than 40 bytes.

Each time this command is sent, the supplied text is appended as a new line to the specified page.

The default page size is 200 lines of text.

The specified TDA page is cleared if *Text* contains all 0's, i.e. no data.

Typically, the TDA is used to provide "transactional" data to the operator, e.g. in a retail POS environment, the TDA would be used to show receipt information.

# Command 37h – Select TDA

**Format:**

| | |
|---|---|
| Command No. | 37h |
| Command XOR | C8h |
| PageNumber | xxh |
| Checksum | xxh |

**Functionality:**

This command is used to select which page buffer to display on the TDA. The command is only valid with consoles that have an integrated large-panel LCD, e.g. SK-7000. The TDA is a dedicated 12x40 section of this display.

Page 1 is visible by default.

This command can be useful for compiling separate data pages and flipping between them quickly on the large-LCD display.

# Command 38h – Download Graphic Image to EEPROM

**Format:**

| | |
|---|---|
| Command No. | 38h |
| Command XOR | C7h |
| ImageRef | xxh |
| Length | xxh |
| ImageData | xx xx xx xx xx xx xx xx xx xx xx xx xx xx … xx xx |
| Checksum | xxh |

**Functionality:**

This command is used to store graphic images in the non-volatile EEPROM memory. This command is only valid with consoles that support EEPROM memory.

*ImageRef* is the specified number that a host will use to recall the image for display on a ScreenKey using Command 39h – Display a Downloaded Graphic Image.

*Length* is the number of bytes in *ImageData*. This **must** be 64 for LC16 images and 108 for LC24 images.

*Note:*
*All images must be sent in "compiled" or "ScreenKey Ready" format.*

# Command 39h – Display a Downloaded Graphic Image

**Format:**

| | |
|---|---|
| Command No. | 39h |
| Command XOR | C6h |
| KeyNo. | xxh |
| ImageRef | xxh |
| Checksum | xxh |

**ScreenKey Display Numbering Scheme**

**Functionality:**

This command is used to display an image previously stored in the onboard EEPROM on a specified ScreenKey. If *KeyNo*. is zero then the image is displayed on all ScreenKeys.

This command is only valid with consoles that support EEPROM memory.

*ImageRef* is the specified number that was allocated to that image when it was first sent to the console.

# Command 3Ah – Display LC24 Data on ScreenKeys

**Format:**

| | |
|---|---|
| Command No. | 3Ah |
| Command XOR | C5h |
| Key No. | xxh **ScreenKey Display Numbering Scheme** |
| Graphic Data | xx xx xx xx xx xx xx xx xx xx xx xx ..... xx xx xx |
| Checksum | xxh |

**Functionality:**

Display LC24 data on the ScreenKey(s). If *Key No*. is zero then display on all available ScreenKeys--see the Key Numbering Diagram on page 86 for valid *Key No*. values.

LC24 *Graphic Data* is always 108 Bytes and is sent using "Compiled" format (see Command 25h - Display Data on ScreenKey and page **Error! Bookmark not defined.** for details).

This Command is a limited version of Command 25h. It is similar to Command 21h - Display LC16 Data on ScreenKeys but is specifically for LC24 ScreenKeys.

# Command 3Bh – Set ScreenKey Initialisation Registers

**Format:**

| | |
|---|---|
| Command No. | 3Bh |
| Command XOR | C4h |
| Frequency | xxh |
| MuxReg1 | xxh |
| MuxReg2 | xxh |
| Checksum | xxh |

**Functionality:**

This command is for **debug purposes only**.

It can be used to temporarily adjust the internal initialisation register sused by the ScreenKeys. Refer to the ScreenKey specification documents for a dewscription of these registers.

# Command 40h – Download Reset

**Format:**

| | |
|---|---|
| Command No. | 40h |
| Command XOR | C0h |
| DwnlCmd | 00h |
| Checksum | A6h |

**Functionality:**

This command resets the 16-bit download procedure. This should always be used before initiating a new 16-bit formware download and may be used in the event that a download in progress stops responding. This command is only supported by 16-bit consoles.

# Command 41h – Download Information Request

**Format:**

| | |
|---|---|
| Command No. | 41h |
| Command XOR | BFh |
| DwnlCmd | 01h |
| DataType | xxh |
| DwnlFlag | xxh |
| Checksum | xxh |

**Functionality:**

This command is used to retrieve version, size and status information for a particular download data type. This command is only supported by 16-bit consoles.

*DataType* refers to the type of file that may be downloaded into the console memory:

- 01 = Firmware (code update) file
- 02 = Configuration (options) file
- 03 = SAC (ScreenKey Active Control) file

When *DwnlFlag* is '1' the returned information pertains to data present in the download buffer, e.g. the number of bytes already downloaded if called after a partial download. If "Dwnl Flag" is zero the returned information pertains to the data version presently in the memory of the console.

# Command 42h – Download Update Init

**Format:**

| | |
|---|---|
| Command No. | 42h |
| Command XOR | BEh |
| DwnlCmd | 02h |
| DataType | xxh |
| Version | xx xx xx xxh |
| DwnlSize | xx xx xx xxh |
| Checksum | xxh |

**Functionality:**

To initiate a new download to a 16-bit console, this command must be issued to the console prior to issuing **Download Data Update** commands. Upon receipt of this command the console will halt its normal keyboard operation, flush any partial downloads if present for that data type and move into download mode.

*DataType* refers to the type of file that may be downloaded into the console memory (see Command 41h – Download Information Request).

The *Version* parameter is a temporary identifier that remains valid only for the duration of the download. It is returned for **Info Request** while the download is in progress. However after the download is completed and activated the version info returned is that embedded in the data itself.

The download size parameter is the total number of bytes that will be downloaded. The console will use this figure to ascertain if enough ram space is available to receive the update.

# Command 43h – Download Data Update

**Format:**

| | |
|---|---|
| Command No. | 43h |
| Command XOR | BDh |
| DwnlCmd | 03h |
| DataType | xxh |
| SyncCount | xxh |
| DwnlData | xx xx xx xx xx xx xx xx xx xx xx xx ..... xx xx xx |
| Checksum | xxh |

**Functionality:**

The data file is downloaded in a series of data 'chunks' with this command. After each chunk is downloaded the *SyncCount* is incremented. The console monitors it's received SyncCount which it returns in response to a **Download Information Request** command. If the *SyncCounts* differ then the host can resend the 'chunks' of data since the last successfully sync.

*DataType* refers to the type of file that may be downloaded into the console memory (see Command 41h – Download Information Request).

*DwnlData* is a variable of bytes of data up to a maximum of 245.

# Command 44h – Download Update Complete

**Format:**

| | |
|---|---|
| Command No. | 44h |
| Command XOR | BCh |
| DwnlCmd | 04h |
| DataType | xxh |
| Checksum | xxh |

**Functionality:**

This command must be issued to inform the console that the download for the specified data type has completed. The console will perform some final checks and processing on the new data and then revert to normal operation if no errors were detected.

*DataType* refers to the type of file that may be downloaded into the console memory (see Command 41h – Download Information Request).

# Command 45h – Download Dump Data

**Format:**

| | |
|---|---|
| Command No. | 45h |
| Command XOR | BBh |
| DwnlCmd | 05h |
| DataType | xxh |
| DwnlFlag | xxh |
| Checksum | xxh |

**Functionality:**

This command is used to instruct the console that is should delete/invalidate previously downloaded data.

If *DwnlFlag* is set to '1':

If the console is in download mode then any data already received is removed and the console exits download mode. The console will reset.

If the console is not in download mode then no action is taken.

If *DwnlFlag* is set to '0':

If a previous active data download is present in the console memory then it is deleted. The console will reset.

*DataType* refers to the type of file that may be downloaded into the console memory (see Command 41h – Download Information Request).

M E S S A G E S

# ScreenKey Console Messages

The ScreenKey Console sends Messages to the host. Messages consists of...

- a Header Byte
- zero, one or more data bytes
- a Trailer byte

There are three exceptions to this general rule...

- The *Keystroke Message* bundles the data bytes in with the header and trailer so that the message consists of a Header-plus-data byte and a Trailer-plus-data byte
- The trailer byte is absent altogether from the *Error Code Message*.
- A number of 00h bytes are sent before the *ROM ID Message*.

## Header Byte

The Header is used to identify the message.

## Trailer byte

The trailer is used to signal the end of the message.

# List of Messages

This is a list of the messages sent by the ScreenKey Console to the host.

| Header | Data | Trailer | Description |
|---|---|---|---|
| 9xh | <none> | Exh | Key Close |
| 8xh | <none> | Exh | Key Open |
| A1h | 4 | C1h | MSR Track-1 buffer (Two Tracks) |
| A2h | 4 | C2h | MSR Track-2 buffer (Two Tracks) |
| A3h | 4 | C3h | MSR Track-3 buffer (Two Tracks) |
| B1h | 4 | C1h | MSR Track-1 buffer (One Track) |
| B2h | 4 | C2h | MSR Track-2 buffer (One Track) |
| B3h | 4 | C3h | MSR Track-3 buffer (One Track) |
| B5h | 4 | <none> | Error Code |
| (00h 00h) B6h | 4 | 00h | ROM ID |
| B9h | 4 | C6h | Configuration Data |
| D3h | <none> | D2h | ACK |

## Unknown Message

The Host Computer may receive invalid data caused by initialisation of Ports, UARTs or Interrupts. This invalid data should be handled appropriately.

# Keystroke Message

**Key Close Format:**

| | | |
|---|---|---|
| Header Byte | 10010xxxb | xxx = Row, 0-7 |
| Trailer Byte | 1110yyyyb | yyyy = Column, 0 to 15 |

**Key Open Format:**

| | | |
|---|---|---|
| Header Byte | 10001xxxb | xxx = Row, 0-7 |
| Trailer Byte | 1110yyyyb | yyyy = Column, 0 to 15 |

**Functionality:**

Data is sent to the host when a key is pressed (Key Close) and when it is released (Key Open). Sending Key Opens to the host is optional (See Commands 05h and 06h).

Turning the key in the KeyLock is equivalent to pressing a key.

This message differs from the normal message format in that the data is bundled in with the Header and Trailer bytes.

The first byte identifies itself as the first byte of a pair of keystroke bytes and specifies whether it is a key open or a key close. It also contains the row number corresponding to the key pressed.

The second byte identifies itself as the second byte of the pair and contains the column number corresponding to the key pressed.

See the *Key Press Numbering Scheme* in the Key Numbering Diagram on page 86 for expected values.

# MSR Data Message

**Format:**

| | | | | | | |
|---|---|---|---|---|---|---|
| Header Byte | A1h | A2h | A3h | B1h | B2h | B3h |
| Data Length | xxh | | | | | |
| Data | xx xx xx xx xx xx xx xx xx xx ..... xx xx xx | | | | | |
| Trailer Byte | C1h | C2h | C3h | C1h | C2h | C3h |

**Functionality:**

Data is sent to the host when a Magnetic Stripe is successfully read by the MSR. The data returned may hold information about one track only or two tracks. The Header byte contains the following information...

- header information--identifies this message as a MSR Data Message
- track information--identifies the track (1, 2 or 3)
- magnetic stripe information--tells us if this is the first of two tracks read from a single card

Some magnetic cards contain one track of information, others contain two, a few contain three. The MSR is capable of reading, at most, two tracks from a single magnetic card. Each MSR Data Message contains information from one track. Swiping a single magnetic card with two tracks of data on it will result in two separate MSR data messages being sent to the host. It is necessary to distinguish between two MSR data messages from two separate card swipes and two MSR data messages from a single card swipe. The information contained in the header byte allows us to do this.

For example, A1h says that the information in this MSR data message is from Track 1 and that there is a second MSR data message on its way from the same card swipe. B1h says that the information in this MSR data message is from track 1 and that this is the only MSR data message generated by this card swipe.

The possible combinations are...

| | |
|---|---|
| B1h <*Length*><*card data*> C1h | Track 1 |
| B2h <*Length*><*card data*> C2h | Track 2 |
| B3h <*Length*><*card data*> C3h | Track 3 |
| A1h <*Length*><*card data*> C1h B2h <*Length*><*card data*> C2h | Track 1 & 2 |
| A3h <*Length*><*card data*> C3h B2h <*Length*><*card data*> C2h | Track 3 & 2 |

When sending two tracks the ScreenKey Console will send one immediately after the other--it won't send any other message in between.

**Incomplete / corrupt MSR data**

Normally, the ScreenKey Console should only send back good data and discard bad data. It gets a bit complicated when dealing with a two track reader and a card with two tracks of data on it.

Depending on the way the MSR is configured using Commands 0Dh to 13h, if one track is bad but the other is good then this is a good read of one track of data - forget the bad track and don't return an error.

On the other hand, if the user specified that they require both tracks then this is a bad read - discard the data (good and bad) and return an error message.

# Error Code Message

**Format:**

| | |
|---|---|
| Header Byte | B5h |
| Error Code | xxh |

**Functionality:**

The console returns various error codes to the host. See the section on Error Codes for a detailed description of the errors and the recommended responses.

This message differs from the normal message format in that there is no Trailer byte.

It may be difficult to synchronise the error contained in the *Error Code Message* with the event that caused the error.

# ROM-ID Message

**Format:**

| | |
|---|---|
| Leading Zeros | 00h 00h |
| Header Byte | B6h |
| ROM-ID | xx xx xx xx xx xx xx xx xx xx ..... xx xx xx |
| Trailer Byte | 00h |

**Functionality:**

The console automatically sends a ROM ID message to the host when the ScreenKey Console is turned on or reset. The ROM ID is also returned in response to Command 17h from the host.

A number of leading zeros may be sent before the ROM ID Header Byte. This is because the ROM ID is always the first message sent from the ScreenKey Console to the host after the system is powered on. The leading zeros are intended to "clear out" the ports, UART etc. There are usually 2 but there may be only 1 or even none.

The ROM ID itself is made up as follows...

| | |
|---|---|
| Version | The ROM version in the form "9.9" |
| Message | Variable length message describing, in English, the version followed by "***". |
| Configuration Text | The Configuration of Panels in text format followed by a dash ("-"). |
| Configuration Byte | The Configuration of Panels in binary format--as per the Configuration Byte--see the *Configuration Data Message* |
| End Marker | A dot (".") to mark the end of the ROM ID |

The total length of the ROM ID will not exceed 128 characters.

Example:

```
5.0 for Type-3 PCB (c) SKI Limited 20/Aug/04 *** RGB24-x.
```

where the x at the end represents the Configuration Byte--in this case 3Dh.

# Configuration Data Message

**Format:**

| | |
|---|---|
| Header Byte | B9h |
| H/w Version | 32h |
| Configuration | xxh |
| RAM Size Lo/Hi | xxh xxh |
| Prom Version | xxh xxh |
| Trailer Byte | C6h |

**Functionality:**

In response to **Command 1Ch** from the host, the console sends console configuration and version information to the host.

The Configuration Data Message provides the host with information about the console.

H/W Version: '2' = Current structure (as described here)

Please contact SKI for the configuration information if the console returns any other value.

Configuration Byte: This byte specifies the type of keyboard and the type of ScreenKey panel attached:

**16-bit (e.g. SK-7000)**

Bits 7 – 2 Specify type of console

Bits 1 – 0 Specify the type of ScreenKey panel

00 "LC24" RG
01 "LC16" RG
10 "LC24" RGB
11 "LC16" RGB

**8-bit (e.g. OEM-5400)**

Bits 7 – 6 Unused

Bits 5 – 4 Specify the type of ScreenKey panel

00 "LC24" RG
01 "LC16" RG
10 "LC24" RGB
11 "LC16" RGB

Bits 3 – 0 Specify type of console

| | |
|---|---|
| RAM Size: | The Amount of External RAM is in the console after the ROM download, if any, has been taken into account i.e. amount of External RAM physically installed less RAM used by the ROM download. |
| PROM Version: | These 2 bytes contain the major and minor version number. For example, Version “3.4” will be returned as 33h, 34h. |

# ACK Message

**Format:**

| | |
|---|---|
| Header Byte | D3h |
| Trailer Byte | D2h |

**Functionality:**

In response to Command 03h from the host, the ScreenKey console sends an (ACK) Acknowledgement message to the host.

The ACK message allows the host to check that communications between itself and the ScreenKey Console are still working.

E V E N T  H A N D L I N G

# Event Notification

An event can be notified to the user by:

- a Beep (Simple or Pattern)
- an Error Code Message - see page 68

## Event Notification Modes

There are a number of ways to specify how errors and other events should be notified to the user.

### Default Mode Event Reporting

When an error occurs the beep should be sufficient to tell the operator that it's a normal error like a bad swipe so try again or it's a fatal error like stack overflow so call a supervisor.

In this mode the console handles the errors and reports them in such a way that the operator can easily decide what to do but also can feedback some info if the error is serious.

### Application Mode Event Reporting

Application mode relies on returning error codes to the host. The application programmer can decide how best to deal with it. The Application knows the context of the error so it can do something sensible with the error code – like error recovery, logging for statistical purposes, reporting…

# Comms Error Recovery in the ScreenKey Console

The host sends Commands as a stream of bytes. If the stream is interrupted, due to errors, then the ScreenKey Console must have some way of deciding to discard invalid bytes and start processing again from the start of the next valid Command.

The ScreenKey Console uses the timer, the Command XOR byte and its list of valid commands to recover from an error condition.

When a byte is received it is assumed to be a valid Command byte. The ScreenKey Console starts a 2-second timer. If the next byte arrives before this times out then it is checked to see if it is the XOR of the first byte recieved. If it is the ScreenKey Console checks if this is on its list of valid Commands. If not, it discards both bytes – see Event Code 65h. If it is a valid command the ScreenKey Console will wait to receive the rest of the command (including the checksum byte) from the host. If the Checksum is valid the ScreenKey Console process the command. If not it discards all bytes received – see Event Code A8h.

The ScreenKey Console will discard the command byte without notification if the timer times out before the Command XOR byte arrives and will go back to waiting for a command byte.

If the 2$^{nd}$ byte is not the XOR of the 1$^{st}$ byte then the ScreenKey Console discards both bytes without notification and goes back to waiting for a command byte.

# Error Codes

| 41h | CPU POST Failed |
|---|---|
| Explanation | The ScreenKey Console has detected that the CPU is not working properly. |
| ScreenKey Console response | This is a fatal error. The Console can only leave the beeper on to indicate this error condition. It is not possible to do a pattern beep or return anything to the host |
| Recommended API response | The host doesn't know about it. |
| How to Avoid | Due to faulty Hardware--fix Hardware |

| 42h | Bad RAM |
|---|---|
| Explanation | The ScreenKey Console runs a RAM test on its internal and external RAM. This has failed for some reason. |
| ScreenKey Console response | This is a fatal error. The Console can only leave the beeper on to indicate this error condition. It is not possible to do a pattern beep or return anything to the host |
| Recommended API response | The host doesn't know about it. |
| How to Avoid | Due to faulty Hardware--fix Hardware. |

| 62h | Stack Overflowed |
|---|---|
| Explanation | Timer-0 interrupt runs a test to see if the stack is OK. This test has detected a Stack Overflow |
| ScreenKey Console response | Although the ROM continues to run there is no way to fix the Stack and the event is triggered every time Timer-0 is run to the detriment of the other jobs the console has to do. It's as good as hung. Hopwever, it is still possible to do a RAM Dump. This may retrun useful info about why the Stack overflowed.<br>Note: In general it would be best not to keep going if POST or FailSafe checks fail. Once a fail safe error condition is detected it is unwise to trust anything that happens after that. |
| Recommended API response | If the Application receives an error code then it should inform the user and suggest that they contact product support. |
| How to Avoid | Due to faulty Hardware or bugs in the ScreenKey Console code. Fix bug or Hardware. |

| 63h | MSR - Incomplete Byte at Card-End |
|---|---|
| Explanation | MSR data is read from the Magnetic Stripe Reader a bit at a time. Depending on which Track is being read, a "Byte" will be made up of 5 or 7 bits. In this case, there aren't enough bits to make up a byte. |
| ScreenKey Console response | Assumes the card data was bad and carries on |
| Recommended API response | Reject Card Data |
| How to Avoid | No Recommendation |

| 64h | PROM POST Failed |
|---|---|
| Explanation | The PROM checksum was invalid. |
| ScreenKey Console response | Beep on/off and don't do anything else.<br>The on/off is based on loop counters because Timer-0 is not running. Loop counters are affected by the speed of the CPU – so the beep will be depend on the speed of the CPU |
| Recommended API response | The host doesn't know about it. |
| How to Avoid | Due to faulty Hardware or Firmware . |

| 65h | Invalid Command Code from host |
|---|---|
| Explanation | The ScreenKey Console is expecting the first byte (Command No.) of a new Command from the host. However, the byte received is not a valid Command No. |
| ScreenKey Console response | Tries to get back in sync with the dat stream from the host – see discussion on Comms Error Recovery on page 74. |
| Recommended API response | Inform the user. Possibly resend the last communications |
| How to Avoid | Probably a Comms problem. It could be a bug in code in the ScreenKey Console or host or Cable problems. |

| 66h | Internal Buffers Corrupt/Overflowed |
|---|---|
| Explanation | The ScreenKey Console uses a number of RAM work areas. There is a marker at the end of each area. If this marker is missing it means that the buffer has overflowed. Markers are checked repeatedly. |
| ScreenKey Console response | Although the ROM continues to run there is no way to un-corrupt the buffers. The beep/error code is reported continuously to the detriment of the other jobs the console has to do. It's as good as hung. However, a RAM dump is still possible which may help debug the problem. |
| Recommended API response | Inform the user |
| How to Avoid | There is, most likely, a bug in the ScreenKey Console ROM. |

| 82h | MSR - Check-Digit Error |
|---|---|
| Explanation | The ScreenKey Console has calculated the Luhn check digit value for the A/C number read from the Magnetic card and found that it does not match the last digit of the card. See Commands 0Bh and 0Ch for more information on Luhn Checks. |
| ScreenKey Console response | Report the error and carry on. |
| Recommended API response | Inform the user that the card has been rejected and why. |
| How to Avoid | Use Command 0Ch to disable the Luhn check in the ScreenKey Console. Not all A/c numbers read from cards have a Luhn check digit. This is the default setting. |

| 83h | Invalid Console Combination |
|---|---|
| Explanation | The ScreenKey Console has detected that old and new ScreenKey panels are mixed on the same ScreenKey Console. This should never happen. |
| ScreenKey Console response | Hangs. |
| Recommended API response | Inform the user that the ScreenKey Console contains incompatible ScreenKey panels. Recommend that they return it to their supplier for replacement. |
| How to Avoid | Checking that the ScreenKey panels are all the same before delivery of the ScreenKey Console to the customer. See Diagnostics Mode (page 11) |

| 85h | MSR – No recognisable data or errors on the card |
|---|---|
| Explanation | Tsoem data was read from the card but it did not conform to expected data format not did it cause the analysis to report a error. |
| ScreenKey Console response | Report the error and carry on. |
| Recommended API response | No Recommendation |
| How to Avoid | Card may have been swiped incorrectly--retry, or may be faulty-try another card. |

| 86h | MSR - Parity error |
|---|---|
| Explanation | Parity error on the data read from the Card Reader. |
| ScreenKey Console response | Report the error and carry on. |
| Recommended API response | No Recommendation |
| How to Avoid | No Recommendation |

| 87h | host - the host is not taking Bytes |
|---|---|
| Explanation | The ScreenKey Console cannot send data to the host. Specifically, the ScreenKey Console is waiting for the host to go "Ready". |
| ScreenKey Console response | Continues to wait in 5-sec blocks until the host is ready |
| Recommended API response | The API won't know about it because, by definition, the ScreenKey Console can't send the error code to the host. |
| How to Avoid | May be due to cables coming loose at the back of the host or ScreenKey Console. Ensure all cables are secure. |

| 88h | MSR - End-of-Card, but no ETX or LRC |
|---|---|
| Explanation | The card did not have an ETX or LRC byte. |
| ScreenKey Console response | Report the error and carry on. |
| Recommended API response | No Recommendation |
| How to Avoid | No Recommendation |

| 8Ah | MSR - Card Swiped. but Buffer Full |
|---|---|
| Explanation | The ScreenKey Console's buffer that holds input from the MSR is full. |
| ScreenKey Console response | Report the error and carry on. |
| Recommended API response | No Recommendation |
| How to Avoid | No Recommendation |

| 8Ch | MSR - LRC Error |
|---|---|
| Explanation | The Longitudinal Redundancy Check (LRC) byte is one of the pieces of data read from a Card. In this case this checksum failed. |
| ScreenKey Console response | Report the error and carry on. |
| Recommended API response | Reject the card |
| How to Avoid | Avoid faulty cards |

| 8Dh | MSR - Mandatory Track missing |
|---|---|
| Explanation | Commands 0Dh to 13h are used to tell the ScreenKey Console which if any tracks are mandatory. If the data contained on a card doesn't include the specified mandatory track(s) then the ScreenKey Console rejects the card. |
| ScreenKey Console response | Report the error and carry on. |
| Recommended API response | No Recommendation |
| How to Avoid | Relax restrictions on what tracks are mandatory. |

| 8Eh | MSR - Processed Buffer Overflow |
|---|---|
| Explanation | There are two buffers handling MSR input--the input buffer itself that takes raw data from the Card Reader and the Processed Buffer which holds the result of analysing the raw data. The buffer holding the Processed Data has overflowed. |
| ScreenKey Console response | |
| Recommended API response | No Recommendation |
| How to Avoid | There is, most likely, a bug in the ScreenKey Console ROM. |

| A3h | Keypress buffer full – keypress discarded |
|---|---|
| Explanation | The console can buffer up to 10 keypresses. Keypresses are transmitted to the host immediately so the buffer will usually never have more than one keypress awaiting transmission. However, if the host is busy and cannot accept transmissions then the buffer will fill up if keys are pressed. This error is an indication that the buffer is full and that the key just pressed has been discarded |
| ScreenKey Console response | Report the error and carry on. |
| Recommended API response | No Recommendation |
| How to Avoid | Make sure that the software in the host responsible for communications with the console is always ready to take data from the console. |

| A4h | Serial In-Buffer has Overflowed |
|---|---|
| Explanation | The ScreenKey Console's Serial data receive buffer is full. The buffer is 250 bytes. At 170 bytes the ScreenKey Console tells the host that it is busy so the host should stop sending. Only if the host continues to send will the buffer ever fill. |
| ScreenKey Console response | The beeping will be triggered for each byte received while the buffer is full – this could be a lot of beeps. |
| Recommended API response | The host should stop sending when the ScreenKey Console tells it to. |
| How to Avoid | No recommendation. |

| A8h | Checksum error-data received from host |
|---|---|
| Explanation | The host has sent a Command to the ScreenKey Console but the ScreenKey Console has decided that the checksum is invalid. |
| ScreenKey Console response | Tries to get back in sync with the data stream from the host – see discussion on Comms Error Recovery on page 74. |
| Recommended API response | No Recommendation |
| How to Avoid | May be due to cables coming loose at the back of the host or ScreenKey Console. Ensure all cables are secure. |

| B0h | Ints-Bad, T0 dead ! |
|---|---|
| Explanation | The code in the ScreenKey Console runs various checks to ensure the interrupts in the ScreenKey Console are operating correctly. In this case, Timer-0 should have been active but was not. |
| ScreenKey Console response | Console halts operation. |
| Recommended API response | No Recommendation |
| How to Avoid | Fix faulty Hardware--may be the CPU |

APPENDIX A

# Documentation Control

## A.1 Change Control

This document is the responsibility of the author and is subject to formal change control after the initial approved release (i.e. issue 1.0).

## A.2 Abbreviations Used/Terms of Reference

| | |
|---|---|
| Fixed Keys | Standard POS Console keys--contrast with ScreenKeys. |
| LED | Light Emitting Diode. There are 4 LEDs on the Model 6000 |
| LRC | Longitudinal Redundancy Check--a byte used to verify that the data read from a Magnetic Card is valid. |
| MSR | Magnetic Stripe Reader |
| O/S | Operating Systems -- MS DOS, UNIX, OS/2, Windows etc... |
| host | Personal Computer. |

## A.3 Historical Change Reference

| Issue | Date | Author | Changes Made |
|---|---|---|---|
| 0.1 | 10/05/95 | Ultan Carroll | Initial Draft. |
| 0.2 | 15/05/95 | Ultan Carroll | After review. |
| 1.0 | 16/05/95 | Ultan Carroll | First release ROM 2.9. |
| 1.1 | 21/02/96 | Ultan Carroll | Corrections made. |
| 2.0 | 25/06/96 | Ultan Carroll | Update to ROM 3.4 |
| 2.1 | 18/10/96 | Ultan Carroll | LK2-A and ROM 3.5 |
| 2.2 | 11/11/96 | Ultan Carroll | Updated to ROM 3.6 |
| 3.7 | 27/01/97 | Leo Ferris | ROM 3.7 |
| 4.1 | 23/02/00 | Ultan Carroll | ROM 4.1 |
| 4.2 | 27/07/00 | Ultan Carroll | ROM 4.2 |
| 4.3 | 16/07/01 | Ultan Carroll | ROM 4.3 |
| 4.4 | 14/01/02 | Ultan Carroll | ROM 4.4 |

| | | | |
|---|---|---|---|
| 5.0 | 22/06/05 | M McDonnell | ROM 5.0 |
| 5.1 | 30/11/06 | M McDonnell | ROM 5.1 |

## A.4 Change Summary

1.1 Corrections made.

2.0 This issue of the document corresponds to ROM 3.4 of the ScreenKey Console.

Renamed “Diagnostics Mode” to “Debug mode” to avoid confusion with built-in diagnostics.

Revised the format of the document.

Major changes made in order to fully document all Commands and Messages as well as improve the presentation of the Interface/protocols and error codes explained.
Added Appendices for ROM History, Character Set and Pixel Addressing

2.1 This issue of the document corresponds to ROM 3.5 of the ScreenKey Console.

Updated the copyright notice.

Added a note to Command 29h about how to compare version numbers.

Updated to cover the LK2-A version of the ScreenKey Console. This is the CE approved version of the hostB. It uses DTR to hold the Serial ScreenKey Console reset.

2.2 This issue of the document corresponds to ROM 3.6. No changes to the document are necessary except to note that ROM 3.6 is functionally exactly the same as ROM 3.5 but now has an alternative “no blip” version.

3.7 Changed the Issue numbering of this document to match the version of the ROM that it is describing.

This issue of the document corresponds to ROM 3.7 of the ScreenKey Console.

4.1 This issue of the document corresponds to ROM 4.1 of the ScreenKey Console.

(CENSUS 15) Clarified the description of “RAM Size”

(CENSUS131) Dropped the “If Length is zero…” warning for

commands 1Ah and 1Bh.

Replaced "PASKey" with "ScreenKey".

Dropped Commands 18h and 19h because they do nothing. Dropped Command 20h because it isn't needed.

Added sections on the Model 6000 and 5400 consoles. Up till now this document had been dealing with the Model 2000 only.

Dropped references in ROM history to ROM versions before 3.0 since these ROMs were never issues.

Emphasised the fact that only one byte of the Repeat Key delay is used even though the command calls for 2 bytes.

Added more detail about the differences between theLK2 and LK2-A model 2000 hardware and the way the Reset works.

References to "1/18$^{th}$ second" were corrected to read "1/20$^{th}$ second". The Console works off a 5 millisec timer.

References to "ModeLock" were replaced with "KeyLock" and references to "MCR" with "MSR".

4.2 This issue of the document corresponds to ROM 4.2 of the ScreenKey Console.

Added Command 31h – Specify Card Reader Type

Added Command 32h – Specify Event Notification

Major revision of the Error Handling section to take account of the new Event Notification command. Added error message A3h description. Dropped errors B8h and 61h. Reassigned errors 41h and 64h.

Also, various updates to reflect the changes in ROM 4.2.

4.3 This issue of the document corresponds to ROM 4.3 of the ScreenKey Console. Changed the meaning of error codes 85h and 88h returned by the Card Reader. Dropped error codes 81h, 88h and 89h.

4.4 This issue of the document corresponds to ROM 4.4 of the ScreenKey Console.

Updated the ROM History to include changes in ROM 4.4.

Corrected a reference to Command 21h to say that Comamnd 21h required the "Compiled" version of a Graphic. Removed a reference to Command 20h. Updated the Copyright message and contact details.

| | |
|---|---|
| 5.0 | Major rewrite of firmware and supporting documentation. Includes changes to standard RTS/CTS flow-contorl, change to new ROMID and Config Structures. Addtion of RGB and LC24 support. |
| 5.1 | Fixes inability to download new firmware using command 29h, also fixes comms bugs that could cause data loss |

A P P E N D I X  B

# Key Numbering Schemes

There are 3 key numbering schemes.

## Key Press Numbering Scheme

The Key Press Numbering scheme is based on an 8x4 array. The Key press number is the value returned in the KeyStroke Message (see page 65) when that key is pressed. The value is shown on the top line of the key in the diagrams below.

Examples:


Fixed Key


ScreenKey

## ScreenKey Display Numbering Scheme

The ScreenKey Display Numbering Scheme is the value used in Commands 1Dh, 1Eh, 1Fh, 21h and 25h when sending data to a Screen Key to be displayed. ScreenKey Display Numbering does not apply to Fixed Keys. The value is shown on the bottom left corner of the key in the diagrams below.

Example:


ScreenKey

## Logical Key Numbering Scheme

The Logical Key Numbering scheme is based on an 8x4 array. The logical Key Number is used in Commands 2Ah, 2Bh, 2Fh and 30h, It is also the number displayed when the key is pressed in Diagnostics. The value is shown on the bottom right corner of the key in the diagrams below.

Examples:


Fixed Key


ScreenKey

# Model 3000/7000 Layout

Models 3000 and 7000 support one 12-ScreenKey panel and one fixed key panels. Both models also support a KeyLock.

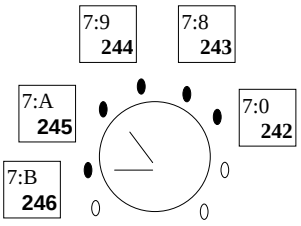
## Fixed Keys Panel

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1:6 **33** | 1:7 **41** | 1:5 **49** | 1:4 **57** | 1:2 **65** | | | |
| 0:6 **34** | 0:7 **42** | 0:5 **50** | 0:4 **58** | 0:2 **66** | 0:3 **74** | 0:1 **82** | 0:0 **90** |
| 4:6 **35** | 4:7 **43** | 4:5 **51** | 4:4 **59** | 4:2 **67** | 4:3 **75** | 4:1 **83** | 4:0 **91** |
| 5:6 **36** | 5:7 **44** | 5:5 **52** | 5:4 **60** | 5:2 **68** | 5:3 **76** | 5:1 **84** | 5:0 **92** |
| 6:6 **37** | 6:7 **45** | 6:5 **53** | 6:4 **61** | 6:2 **69** | 6:3 **77** | 6:1 **85** | 6:0 **93** |

## ScreenKey Panel

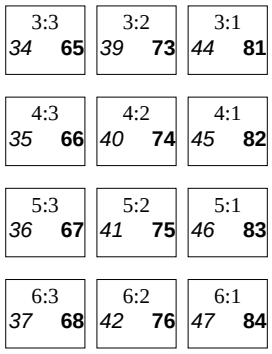
| | | |
|---|---|---|
| 3:F *2* **1** | 3:E *7* **9** | 3:D *12* **17** |
| 4:F *3* **2** | 4:E *8* **10** | 4:D *13* **18** |
| 5:F *4* **3** | 5:E *9* **11** | 5:D *14* **19** |
| 6:F *5* **4** | 6:E *10* **12** | 6:D *15* **20** |

## KeyLock



# Model 5400 Layout

The Model 5400 supports one panel of 12-ScreenKeys. These may be either LC16 or LC24 and RG or RGB keys.



APPENDIX C

# ScreenKey Console ROM History

| Ver | Date | Description |
|---|---|---|
| 3.0 | 09/May/95 | Increased "Morse" pattern from 4 to 5 blips/beeps. Many Internal Error-codes / Beep-patterns changed. Minor fine-tuning on Timer-0 frequency (12MHz -> 11.0592). PAS-Key" changed to "ScreenKey Console". Additional colour patterns in Diagnostics mode. Some very minor mods |
| 3.1 | 11/May/95 | Changed Colour patterns, in Diagnostics mode |
| 3.2 | 03/Nov/95 | Implemented **Commands 22h/23h** (DownLoad Char-Set). Default to "No LUHN" checks on Card-Readers. |
| 3.3 | 24/Nov/95 | Fixed bug whereby some ScreenKeys on LHS Panel were not displayed properly. Problem was introduced in 3.2. |
| 3.4 | 23/May/96 | Fixed a missing "POP" in Serial-In code. Fixed "Missing-Int" bug with Serial Handling, when chars were lost while requesting a KeyLock status because all interrupts were disabled. |
| 3.5 | 28/July/96 | Fixed ExtRAM-Dump and PROM-Dump, so that, if a length of 0 is submitted, the checksum is now read (& discarded). Added support for new **Commands 2Ah** and **2Bh** (from host). Changed the previous key beep On and Off commands (**07h** and **08h**) to Clear and Set the new arrays, when received. |
| 3.6 | 11/Nov/96 | Separate ASM option, to build a version which does not require a CTS "Toggle" from the host, when the host is Ready |
| 3.7 | 27/Jan/97 | Added a Repeat Key function which uses the commands **2Ah** and **2Bh** to flag a key as either repeatable or not repeatable. A new command **2Ch** was also added to enable |

| Ver | Date | Description |
|---|---|---|
| | | the user to change the Typematic rate of a repeating key.<br><br>Made some internal changes to MSR error handling to avoid multiple errors returned from a single bad swipe. |
| 3.8 | 06/Aug/97 | No Longer returns MSR error codes from ROM because they are of no benefit and only confuse the issue.<br><br>Fixed a bug introduced in 3.7 where once the keypress buffer was full it returned "ERR05--buffers overflow/corrupt". Returned to the old method of not accepting any more keystrokes and no beeps. |
| 3.9 | 16/Apr/98 | Disable Sleep Mode in Diagnostics. Sleep Mode was interfering with the factory burn in test.<br><br>Special Note: Windows Emulator allows only digits in the version number. It won't download 3.A, for example. Therefore we increased the major number. There are no adverse affects<br><br>See reference to a second version 3.9 below. |
| 4.0 | 22/Jun/99 | Reinitialise the VKeys (resend the frequency) every couple of seconds to auto repair any ScreenKey display corruption caused by ESD. |
| 4.1 | 23/Feb/00 | (CENSUS 124) Add Support for LED Indicators test in diagnostics. Added two new commands (**2Dh** and **2Eh**) to turn On/Off the LED Indicators on Model 6000.<br><br>(CENSUS 97) Added two new commands (**2Fh** and **30h**) to set key attributes. These are replacements for the existing commands **2Ah** and **2Bh** which should now be considered obsolete although they are still available. Commands 2Ah and 2Bh did not correctly process KeyLock key numbers. The new commands do.<br><br>(CENSUS 135) Dropped the RTI beep at power on / software reset and the download beep. Replaced the RTI logo power on / reset display with blank green display and showed all red display during download<br><br>(CENSUS 19 & 24) An unexpected checksum error was triggered when running on the Model 6000, after power on reset – sometimes. Fixed the loophole in S/w even though it is not necessary because the h/w fix will mean there won't be any more false byte reports. |

| Ver | Date | Description |
|---|---|---|
| | | (CENSUS 141) Increased the gap between refreshes of the ScreenKeys frequency (CENSUS 134) from 2 seconds to 60 seconds because there was a noticeable blink every 2 seconds on some Model 5400 boards. |
| 3.9 | 27/Jun/00 | Released another ROM with version 3.9. It's just a temporary measure to fix a specific problem. This ROM 3.9 will be withdrawn once ROM 4.2 is ready.<br><br>See CENSUS 157 |
| 4.2 | 24/Jul/00 | CENSUS 130 - Automatically return the Mode lock after a ROM download. There was a problem due to the fact that the Key Tables were not initialised before Mode Lock handling so Mode Lock was thought to be an inactive key and therefore not reported.<br><br>CENSUS 138 - MSR Buffer overflow problem Internal Buffers corrupt/overflowed error triggered by swiping a card from the FMI Show. It has 3 tracks - all 7-bit data! The Model 2000 gets into a fatal error loop returning error 66h. Improved the buffer overflow protection.<br><br>CENSUS 144 - Added Command 31h - specify MSR Reader Type. This is to overcome the problem of misidentifying the source of track data when using only the number of separators and whether the data is 5-bit or 7-bit as the means of identifying what track it came from.<br><br>CENSUS 149 - don't allow more than 1 beep per card swipe. CENSUS --- - Added Command 32h - allow SKI to filter errors. The Beep handling was re-written to centralise all beeps and error code to the host in one place. Then added support for Command 32h to allow the application to choose between standard error reporting and the new "Application Mode" error reporting.<br><br>CENSUS 151 - Errors which report repeatedly have had their frequency of reporting to the host tied to the frequency at which beeps are issued i.e. they have been slowed down. This may not b a sufficient solution.<br><br>CENSUS 157 - Activated the buffering of data to the host. This is a workaround that will need to be looked at again in the future. See also version 3.9 above. |

| Ver | Date | Description |
|---|---|---|
| | | CENSUS --- - Debug Mode no longer returns the card data associated with bad swipes. This was done in an attempt to tidyup/clarify the MSR handling code. Use Internal RAM dumps instead - they show the data associated with the bad swipe and more. |
| 4.3 | 16/Jul/01 | The card handling code was completely revised to correct the following problems. Also, changed the meaning of some error codes returned by the Card Reader.<br>• CENSUS 205 – wrong credit card details returned<br>• CENSUS 178 – CLS Handling needs work<br>• CENSUS 186 - Debug code in Card reader ISR<br>• CENSUS 192 - Card Swipe Data returned twice<br>• CENSUS 193 - Bad card swipe but no beeps<br>• CENSUS 210 - A good swipe is reported as bad<br>Because the card handling has changed the reaction to swiping particular individual cards may have changed.<br><br>The following issues have been addresses:<br>CENSUS 194 - Flashing keys left lit (but not flashing) in Sleep Mode |
| 4.4 | 14/Jan/02 | CENSUS 245 - Made changes to the way the ScreenKey Registers are written to avoid "Bleeding" effect seen on some ScreenKeys. |
| 5.0 | 19/Sept/04 | Full rewrite of the firmware. Added support for LC24 and RGB keys. Added support for non-volatile EEPROM to store images, downloadable character set and downloaded firmware. |

A P P E N D I X D

# ScreenKey Console ROM Character Set

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 000 | 00h 00h 00h 00h 00h | | 044 | 00h 05h 06h 00h 00h | , | 088 | 63h 14h 08h 14h 63h | X |
| 001 | 20h 40h 45h 48h 30h | ? | 045 | 08h 08h 08h 08h 08h | - | 089 | 70h 08h 07h 08h 70h | Y |
| 002 | 20h 40h 45h 48h 30h | ? | 046 | 00h 03h 03h 00h 00h | . | 090 | 43h 45h 49h 51h 61h | Z |
| 003 | 20h 40h 45h 48h 30h | ? | 047 | 02h 04h 08h 10h 20h | / | 091 | 00h 7Fh 41h 41h 00h | [ |
| 004 | 20h 40h 45h 48h 30h | ? | 048 | 3Eh 45h 49h 51h 3Eh | 0 | 092 | 20h 10h 08h 04h 02h | \ |
| 005 | 20h 40h 45h 48h 30h | ? | 049 | 00h 21h 7Fh 01h 00h | 1 | 093 | 00h 41h 41h 7Fh 00h | ] |
| 006 | 20h 40h 45h 48h 30h | ? | 050 | 21h 43h 45h 49h 31h | 2 | 094 | 10h 20h 40h 20h 10h | ^ |
| 007 | 20h 40h 45h 48h 30h | ? | 051 | 42h 41h 51h 69h 46h | 3 | 095 | 01h 01h 01h 01h 01h | _ |
| 008 | 20h 40h 45h 48h 30h | ? | 052 | 0Ch 14h 24h 7Fh 04h | 4 | 096 | 00h 00h 60h 50h 00h | ` |
| 009 | 20h 40h 45h 48h 30h | ? | 053 | 72h 51h 51h 51h 4Eh | 5 | 097 | 02h 15h 15h 15h 0Fh | a |
| 010 | 20h 40h 45h 48h 30h | ? | 054 | 1Eh 29h 49h 49h 06h | 6 | 098 | 7Fh 11h 11h 11h 0Eh | b |
| 011 | 20h 40h 45h 48h 30h | ? | 055 | 40h 47h 48h 50h 60h | 7 | 099 | 0Eh 11h 11h 11h 11h | c |
| 012 | 20h 40h 45h 48h 30h | ? | 056 | 36h 49h 49h 49h 36h | 8 | 100 | 0Eh 11h 11h 11h 7Fh | d |
| 013 | 20h 40h 45h 48h 30h | ? | 057 | 30h 49h 49h 4Ah 3Ch | 9 | 101 | 0Eh 15h 15h 15h 0Ch | e |
| 014 | 20h 40h 45h 48h 30h | ? | 058 | 00h 36h 36h 00h 00h | : | 102 | 10h 10h 3Fh 50h 50h | f |
| 015 | 20h 40h 45h 48h 30h | ? | 059 | 00h 35h 36h 00h 00h | ; | 103 | 08h 15h 15h 15h 1Eh | g |
| 016 | 20h 40h 45h 48h 30h | ? | 060 | 08h 14h 22h 41h 00h | < | 104 | 7Fh 08h 10h 10h 0Fh | h |
| 017 | 20h 40h 45h 48h 30h | ? | 061 | 14h 14h 14h 14h 14h | = | 105 | 00h 11h 5Fh 01h 00h | i |
| 018 | 20h 40h 45h 48h 30h | ? | 062 | 00h 41h 22h 14h 08h | > | 106 | 02h 01h 11h 5Eh 00h | j |
| 019 | 20h 40h 45h 48h 30h | ? | 063 | 20h 40h 45h 48h 30h | ? | 107 | 7Fh 04h 0Ah 11h 00h | k |
| 020 | 20h 40h 45h 48h 30h | ? | 064 | 3Eh 41h 5Dh 55h 3Ch | @ | 108 | 00h 41h 7Fh 01h 00h | l |
| 021 | 20h 40h 45h 48h 30h | ? | 065 | 3Fh 48h 48h 48h 3Fh | A | 109 | 1Fh 10h 1Fh 10h 0Fh | m |
| 022 | 20h 40h 45h 48h 30h | ? | 066 | 7Fh 49h 49h 49h 36h | B | 110 | 1Fh 08h 10h 10h 0Fh | n |
| 023 | 20h 40h 45h 48h 30h | ? | 067 | 3Eh 41h 41h 41h 22h | C | 111 | 0Eh 11h 11h 11h 0Eh | o |
| 024 | 20h 40h 45h 48h 30h | ? | 068 | 7Fh 41h 41h 22h 1Ch | D | 112 | 1Fh 14h 14h 14h 08h | p |
| 025 | 20h 40h 45h 48h 30h | ? | 069 | 7Fh 49h 49h 49h 41h | E | 113 | 08h 14h 14h 14h 1Fh | q |
| 026 | 20h 40h 45h 48h 30h | ? | 070 | 7Fh 48h 48h 48h 40h | F | 114 | 1Fh 08h 10h 10h 00h | r |
| 027 | 20h 40h 45h 48h 30h | ? | 071 | 3Eh 41h 49h 49h 2Fh | G | 115 | 09h 15h 15h 15h 12h | s |
| 028 | 20h 40h 45h 48h 30h | ? | 072 | 7Fh 08h 08h 08h 7Fh | H | 116 | 10h 10h 7Eh 11h 11h | t |
| 029 | 20h 40h 45h 48h 30h | ? | 073 | 00h 41h 7Fh 41h 00h | I | 117 | 1Eh 01h 01h 02h 1Fh | u |
| 030 | 20h 40h 45h 48h 30h | ? | 074 | 02h 01h 41h 7Eh 40h | J | 118 | 1Ch 02h 01h 02h 1Ch | v |
| 031 | 20h 40h 45h 48h 30h | ? | 075 | 7Fh 08h 14h 22h 41h | K | 119 | 1Eh 01h 06h 01h 1Eh | w |
| 032 | 00h 00h 00h 00h 00h | | 076 | 7Fh 01h 01h 01h 01h | L | 120 | 11h 0Ah 04h 0Ah 11h | x |
| 033 | 00h 00h 79h 00h 00h | ! | 077 | 7Fh 20h 18h 20h 7Fh | M | 121 | 18h 05h 05h 05h 1Eh | y |
| 034 | 00h 70h 00h 70h 00h | " | 078 | 7Fh 10h 08h 04h 7Fh | N | 122 | 11h 13h 15h 19h 11h | z |
| 035 | 14h 7Fh 14h 7Fh 14h | # | 079 | 3Eh 41h 41h 41h 3Eh | O | 123 | 08h 36h 41h 41h 00h | { |
| 036 | 12h 2Ah 7Fh 2Ah24h | $ | 080 | 7Fh 48h 48h 48h 30h | P | 124 | 00h 00h 77h 00h 00h | \| |
| 037 | 62h 64h 08h 13h 23h | % | 081 | 3Eh 41h 45h 42h 3Dh | Q | 125 | 00h 41h 41h 36h 08h | } |
| 038 | 36h 49h 55h 22h 05h | & | 082 | 7Fh 48h 4Ch 4Ah 31h | R | 126 | 08h 08h 2Ah 1Ch 08h | ~ |
| 039 | 00h 50h 60h 00h 00h | ' | 083 | 32h 49h 49h 49h 26h | S | 127 | 08h 1Ch 2Ah 08h 08h | |
| 040 | 00h 1Ch 22h 41h 00h | ( | 084 | 40h 40h 7Fh 40h 40h | T | | | |
| 041 | 00h 41h 22h 1Ch 00h | ) | 085 | 7Eh 01h 01h 01h 7Eh | U | | | |
| 042 | 14h 08h 3Eh 08h 14h | * | 086 | 7Ch 02h 01h 02h 7Ch | V | | | |
| 043 | 08h 08h 3Eh 08h 08h | + | 087 | 7Eh 01h 0Eh 01h 7Eh | W | | | |

This is the built-in character set. The remainder of the character set in the range 128 to 255 are set to 20h 40h 45h 48h 30h; the question mark.

APPENDIX E

# Pixel Addressing

Graphic image data, as sent to the ScreenKeys, describes to the ScreenKey whether each pixel should be ‘on’ (lit) or ‘off’ (dark).

Within the ScreenKey, the mapping between the internal data registers and the pixels is defined according to the relevant LC16 or LC24 Datasheet. To reduce computing overheads, it is preferable to send graphic image data form the host to the ScreenKey console in this format. This data stream can then be written directly into the ScreenKey internal registers. This data format is called “Compiled” or “ScreenKey Ready”.

When constructing a graphic image, it is easier to visualise the graphic in a straight one-to-one mapping between data bits and the pixel locations. For example, bit 7 of the first byte refers to the first pixel of the top row (top left pixel), bit 6 refers to the second pixel of the top row, and so on. This format is called “uncompiled”.

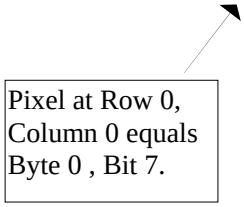
Both formats are described below for each type of ScreenKey, i.e. LC16 and LC24.

# LC16 ScreenKey Pixel Addressing -- Uncompiled

LC16 ScreenKeys have a resolution of 32 pixels across (columns) by 16 pixels down (rows).

The top row of pixels may be defined by 4 bytes exactly (8 bits per byte x 4 bytes = 32). The full LC16 ScreenKey pixel data may then be defined using only 64 bytes (4 bytes per row x 16 rows = 64 bytes).

The 64 bytes that make up an uncompiled LC16 Graphic is based on the following pixel Addressing scheme:




| 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 |
|---|---|---|---|
| Byte 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 31 |
| 32 | 33 | 34 | 35 |
| 36 | 37 | 38 | 39 |
| 40 | 41 | 42 | 43 |
| 44 | 45 | 46 | 47 |
| 48 | 49 | 50 | 51 |
| 52 | 53 | 54 | 55 |
| 56 | 57 | 58 | 59 |
| 60 | 61 | 62 | 63 |

# LC16 ScreenKey Pixel Addressing – ScreenKey Ready (Compiled)

LC16 ScreenKeys have a resolution of 32 pixels across (columns) by 16 pixels down (rows).

The datasheet for LC16.2 Trend ScreenKey defines the following Pixel Addressing scheme:

Pixel at Row 0, Column 0 equals Byte 0 , Bit 0.

| 0 1 2 3 4 5 6 7 | 0 1 2 3 4 5 6 7 | 0 1 2 3 4 5 6 7 | 0 1 2 3 4 5 6 7 |
|---|---|---|---|
| Byte 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 31 |
| 32 | 33 | 34 | 35 |
| 36 | 37 | 38 | 39 |
| 40 | 41 | 42 | 43 |
| 44 | 45 | 46 | 47 |
| 48 | 49 | 50 | 51 |
| 52 | 53 | 54 | 55 |
| 56 | 57 | 58 | 59 |
| 60 | 61 | 62 | 63 |

This "compiled" or "ScreenKey Ready" format is very similar to the uncompiled format described above. The only difference is that the bits are reversed – the top left hand pixel in the uncompiled scheme is represented by Byte 0, bit 7 whereas in the ScreenKey ready scheme it is represented by Byte 0, bit 0.

This format must be used when sending LC16 graphics data using command **Command 21h** or **Command 25h** (with the "Compiled" flag set to ON).

# LC24 ScreenKey Pixel Addressing -- Uncompiled

LC24 ScreenKeys have a resolution of 36 pixels across (columns) by 24 pixels down (rows).

The top row of pixels may be defined by 5 bytes (8 bits per byte x 4 bytes = 32 plus 4 bits from byte 5). The full LC24 ScreenKey pixel data then requires 120 bytes to define all 24 rows (5 bytes per row x 24 rows = 120 bytes).

The 120 bytes that make up an uncompiled LC24 Graphic is based on the following pixel Addressing scheme:

| 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 |
|---|---|---|---|---|
| Byte 0 | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 9 |
| 10 | 11 | 12 | 13 | 14 |
| 15 | 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 |
| 30 | 31 | 32 | 33 | 34 |
| 35 | 36 | 37 | 38 | 39 |
| 40 | 41 | 42 | 43 | 44 |
| 45 | 46 | 47 | 48 | 49 |
| 50 | 51 | 52 | 53 | 54 |
| 55 | 56 | 57 | 58 | 59 |
| 60 | 61 | 62 | 63 | 64 |
| 65 | 66 | 67 | 68 | 69 |
| 70 | 71 | 72 | 73 | 74 |
| 75 | 76 | 77 | 78 | 79 |
| 80 | 81 | 82 | 83 | 84 |
| 85 | 86 | 87 | 88 | 89 |
| 90 | 91 | 92 | 93 | 94 |
| 95 | 96 | 97 | 98 | 99 |
| 100 | 101 | 102 | 103 | 104 |
| 105 | 106 | 107 | 108 | 109 |
| 110 | 111 | 112 | 113 | 114 |
| 115 | 116 | 117 | 118 | 119 |

Pixel at Row 0, Column 0 equals Byte 0 , Bit 7.

Bits 3-0 of every 5th byte not used

***Note:***

***The ScreenKey console only accepts COMPILED format data for LC24 graphics. If UNCOMPILED data is sent it will be ignored.***

# LC24 ScreenKey Pixel Addressing – ScreenKey Ready (Compiled)

LC24 ScreenKeys have a resolution of 36 pixels across (columns) by 24 pixels down (rows).

The datasheet for LC24.2 Trend ScreenKey defines the following Pixel Addressing scheme (the upper line is the byte number and the lower line is the corresponding bit for that pixel):

| B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|
| b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 |
| B8 | B7 | B6 | B5 | B4 |
| b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 |
| B13 | B12 | B11 | B10 | B9 |
| b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 |
| B17 | B16 | B15 | B14 | B13 |
| b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 |
| B22 | B21 | B20 | B19 | B18 |
| b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 |
| B26 | B25 | B24 | B23 | B22 |
| b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 |
| B31 | B30 | B29 | B28 | B27 |
| b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 |
| B35 | B34 | B33 | B32 | B31 |
| b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 |
| B40 | B39 | B38 | B37 | B36 |
| b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 |
| B44 | B43 | B42 | B41 | B40 |
| b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 |
| B49 | B48 | B47 | B46 | B45 |
| b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 |
| B53 | B52 | B51 | B50 | B49 |
| b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 |
| B58 | B57 | B56 | B55 | B54 |
| b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 |
| B62 | B61 | B60 | B59 | B58 |
| b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 |
| B67 | B66 | B65 | B64 | B63 |
| b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 |
| B71 | B70 | B69 | B68 | B67 |
| b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 |
| B76 | B75 | B74 | B73 | B72 |
| b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 |
| B80 | B79 | B78 | B77 | B76 |
| b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 |
| B85 | B84 | B83 | B82 | B81 |
| b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 |
| B89 | B88 | B87 | B86 | B85 |
| b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 |
| B94 | B93 | B92 | B91 | B90 |
| b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 |
| B98 | B97 | B96 | B95 | B94 |
| b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 |
| B103 | B102 | B101 | B100 | B99 |
| b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3 \| b2 \| b1 \| b0 | b7 \| b6 \| b5 \| b4 b3\| b2 \| b1 \| b0 |
| B107 | B106 | B105 | B104 | B103 |
| b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7 \| b6 \| b5 \| b4 | b3 \| b2 \| b1 \| b0 b7\| b6 \| b5 \| b4 |

This "compiled" or "ScreenKey Ready" format is very different to the uncompiled format described above and uses 108 bytes instead of 120.

This format must be used when sending LC24 graphics data using command **Command 21h** or **Command 25h** (with the “Compiled” flag set to ON).

***Note:***

***All LC24 graphic MUST be sent in this COMPILED or ScreenKey Ready format.***