SmartDisplay Command Panel
LCD 36x24 version

Revision B
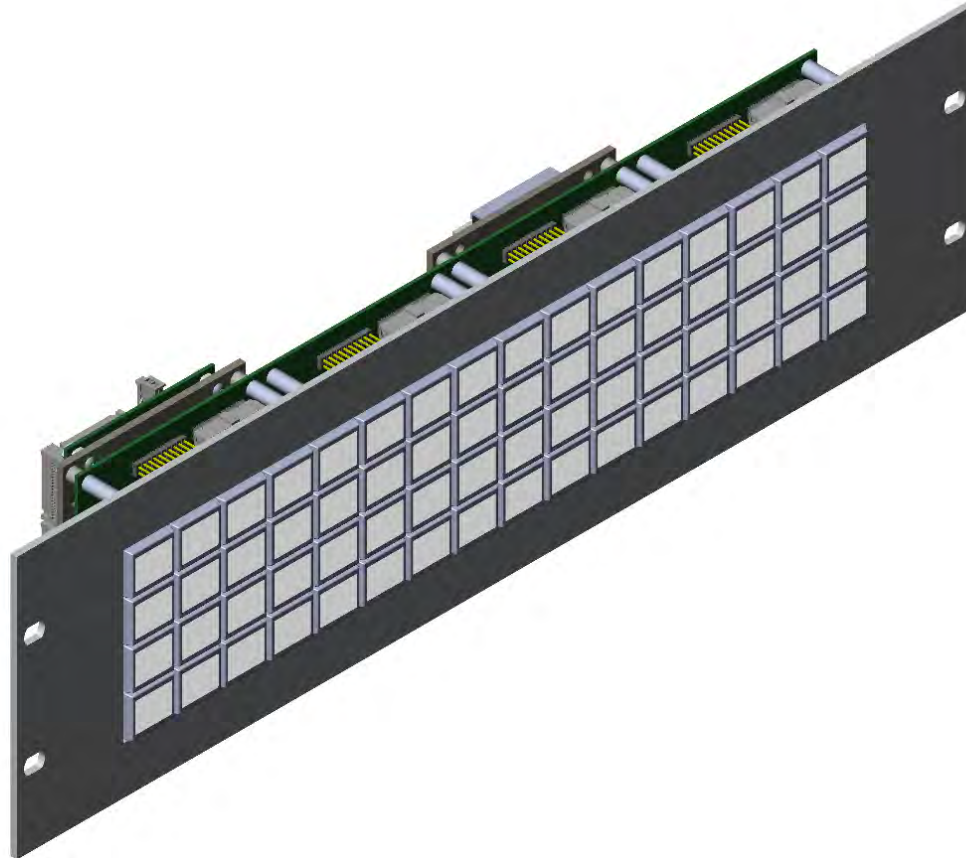
All Rights Reserved Worldwide

Table of Contents

1. LCD 36x24 SmartDisplay Switch

The LCD 36x24 SmartDisplay switch is a graphic 36x24 LCD display mounted in the key cap of a momentary pushbutton. It has an RGD backlight with discrete control.

Please contact engineering@nkkswitches.com with your requirements for custom solutions.

NKK can supply subsystems with any configuration and number of LCD 36x24 switches with USB and Ethernet communication. For instance, the SmartDisplay Command Panel is a powerful human-to-machine-interface tool that uses 64 of the LCD SmartDisplays to allow designers to dynamically change switch legends and images based on desired application functions. The system is ready to interface with a customer's application through Ethernet and USB. It can receive commands, send information, and update the SmartDisplay images.

An example system is available as outlined below.
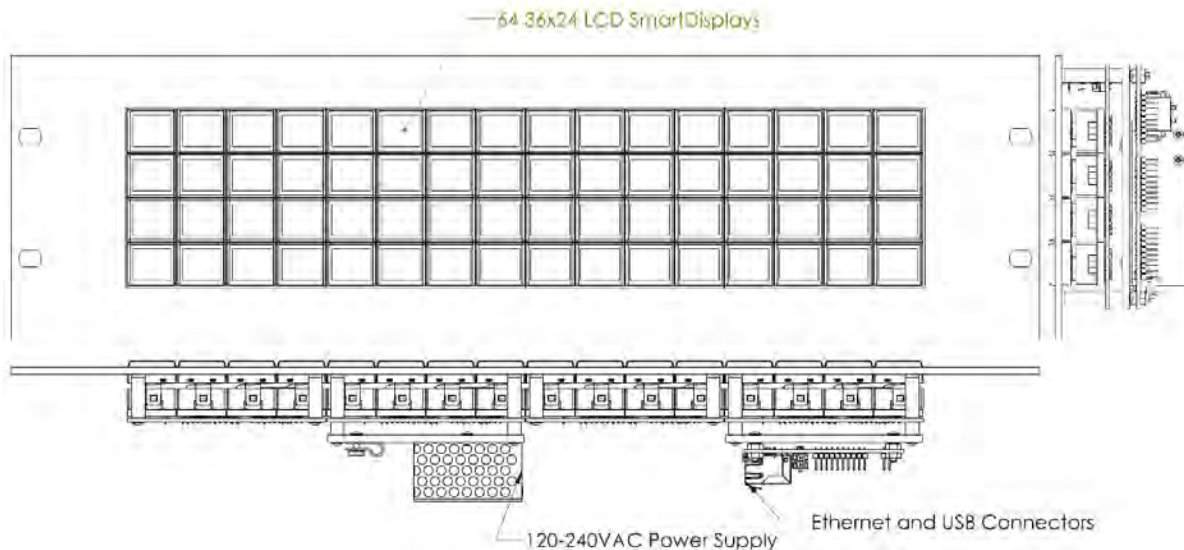
The system is ideal for use in applications with multiple, complex functions which would ordinarily require many dedicated switches and complex training. The dynamic nature of the system allows for instantaneous transitions from generalized lists of categories down to function specific actions. This reduces the need for complicated controls and shortens the time for training by only displaying relevant options and commands.

To help with development, NKK Switches provides free software, Engineering Kits Communicator, to save and erase images on the controller. Also, NKK Switches provides all the documentation necessary to get up and running quickly on our website:
https://www.nkkswitches.com/SmartDisplay-resources/

2. General Features of the SmartDisplay Command Panel

The system is a 64 programable display-on-pushbutton system on a 3U rack-mount panel. It comes with the following features:

Features:
- 4,032 functions available with a maximum of 3 switch presses.
- 64 36x24 LCD SmartDisplays with momentary pushbutton functionality.
- USB or Ethernet controlled.
- Power Specs: 120-240VAC, Max 24 Watts. (with included power cord).
- The unit comes with a 6-foot USB 2.0 A to Mini-B cable (IS-USB1).
- On-board memory for 1000 images minimum (up to 3000 upon request).
- 16 levels of brightness.
- Real-time control by host.
  - Save images to memory.
  - Show any saved image on any switch.
  - Reports switch activity to host.
  - Ability to send images directly to switches without saving to memory.
- Controller board firmware can be customized based on customer requirements.
- Firmware field upgradable via USB.
- Windows based software is available for communication.
  - Accepts bitmap files, extracts the images and download them to the controller.
  - Allows typing of commands and downloading to the controller.
  - Extracts HEX or ASCII data from Excel files and downloads to the controller.
  - Messages to and from the controller are displayed in different colors.
- The communication protocol can be modified to meet customer's requirements.
- Please contact the factory about custom builds and firmware modifications.

3. Application Level Considerations

The information displayed by the system can be set up in a variety of ways. Two arrangements are common; menus of items and function specific actions.
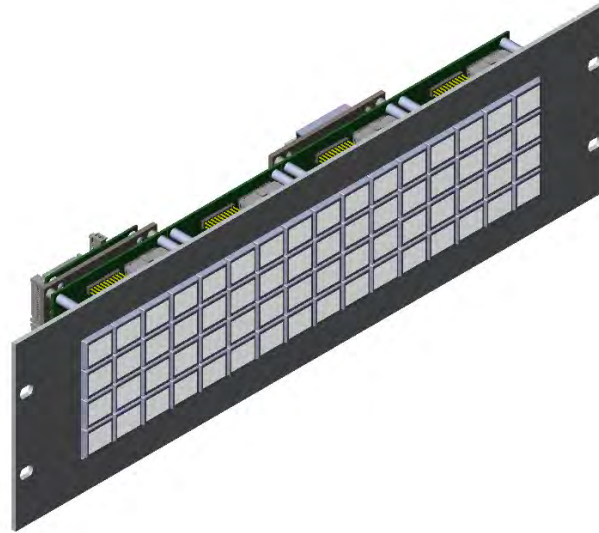
Menus of items is just that; a list of different items from which the user may choose from. Once an item is chosen then all the displayed images could be changed to show only actions or other menus associated with just that item. One way to set on the displays is to have the menu on the left and the function actions associated with a particular item on the right.
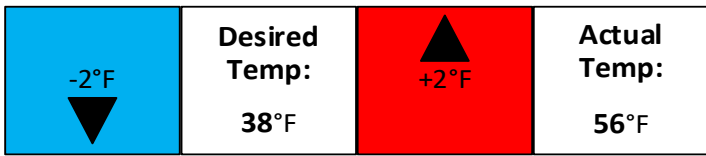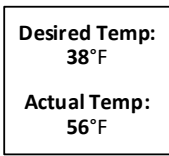
Example: List of menu items on the left. Actions associated with Menu Item 2 on the right.					
Menu Item 1	Menu Item 2	Menu Item 3		Menu Item 2 Action 1	Menu Item 2 Action 2
Menu Item 4	Menu Item 5			Menu Item 2 Action 3	Menu Item 2 Action 4 …

Buttons can also just show information without actually having any actions associated with them. Buttons that are not needed for a particular menu or function can simply have nothing displayed.

Another consideration is how many switches to use to display information and/or actions. For a quick glance having as much information as possible displayed might be desired so using one switch per item would be a good set up. On the other hand, the density of overall information might be large enough that buttons might need to display both information and also be an action. This will also impact the number of buttons that need to be pressed in sequence to get to an item or action.

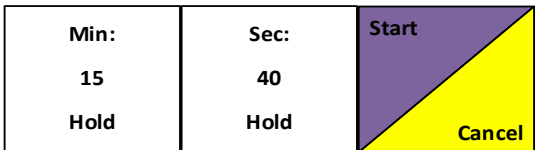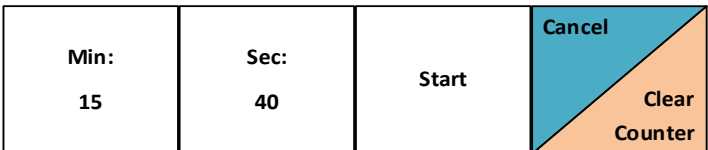| Example: Four buttons; each dedicated to one action or one source of information.	One button showing two sources of information. Also, it is an action button that loops through a predefined number of preset items.
-2°F ▼	**Desired Temp:** 38°F	▲ +2°F	**Actual Temp:** 56°F	**Desired Temp:** 38°F **Actual Temp:** 56°F

Example: The Minutes and Seconds buttons are used to set a timer and then display the countdown. The Start and Cancel buttons could be combined (showing only Start or Cancel depending on the state) or the Cancel could be used to stop the countdown and, with a change in display, a Clear Counter. (Single color at a time.)

| **Min:** 15 **Hold** | **Sec:** 40 **Hold** | **Start** / **Cancel** |

| **Min:** 15 | **Sec:** 40 | **Start** | **Cancel** / **Clear Counter** |

4. Electrical Specifications

Power Specs: 120-240VAC, Max 30 Watts.

5. Communication

The systems can communicate over USB and ethernet. If using ethernet, the system must be configured over USB first. The protocol is the same regardless of the communication method. All commands and responses are detailed in the associated Command List. A non-inclusive list of commands is as follows:

- Acknowledge.
- Erase flash memory.
- Get/Set ethernet settings.
- Reset system.
- Query version.
- Save image to flash memory.
- Send image directly to switch.
- Set image from flash memory on specific switch.

The system shows up as a generic USB COM port. This allows quick testing, loading of images, and integration with customer software. For testing, the NKK Engineering Kits Communicator or a standard terminal program such as Putty can be used.
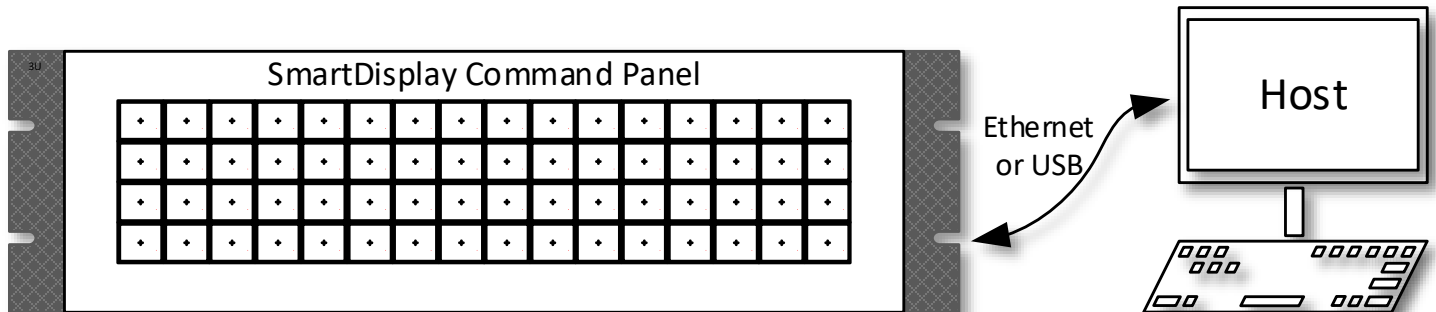
6.Images

Images can be created in any graphics software such as Paint, Photoshop, etc, or even user-created software. All images can be saved onto the system by using the free Engineering Kits Communicator, located on the NKK Website:
https://www.nkkswitches.com/

(Images can also be loaded onto the system with user-created software as long as the rules for the images and communications are followed.)

To use this software, images must be saved in the proper format:

LCD 36x24	Monochrome bitmap (.bmp) 36x24 pixels

Please note that the **flash memory must be erased before new images are loaded**, or images will not display properly. Erasing can take up to 2 minutes depending on the size of the flash memory.

The Engineering Kits Communicator will auto-convert the monochrome .bmp file to the switch format and send the data. If writing custom software, be aware bitmap format specifies the bottom-left corner as the "top". Therefore, to send images properly to the switches the data needs to be sent last row first, followed by next to last, etc.

The system expects LCD image pixels to be monochromatic. Pixels are on or off. Only the backlighting has color. Each bit corresponds to a pixel in the image.

When saving images to flash, the data needs to be converted to ASCII hex for 240 bytes of data. If streaming live images, data should be sent as-is for 120 bytes of data.

Monochrome bitmap (.bmp) 36x24 pixels	1 byte per 8 pixels	120 bytes per image

*the last 4 bits of every byte are dummy bits and not used

7. Operational Overview

Upon power-up the system configures and turns on the switches. The system then waits for a command from the host. The only action the system performs automatically is reporting switch presses. All other actions must be commanded from the host.

The flash memory stores both ethernet settings and images. They are stored in separate sections of memory, so erasing the images will not remove the ethernet settings and vice versa. Each image is assigned a sequential address when saved into flash memory; A list of which image is stored at which address must be kept on the host software to know which image to display where.

When a switch is pressed, the system reports that back to the host software. The system only reports switch state changes (a press or release). More than one switch can be in the pressed position at the same time. Bits are set when the switch is pressed and cleared when the switch is released. An example of a switch response is:

<div align="center">

0x50 0x74 0x31 0x34 0x36 0x41

</div>

The 0x50 signifies a switch press response. The 0x74 represents the bank number. Each bank has 0 to 15 for the switch index. The next 4 numbers are ASCII hex of a 16-bit number, and represent what switches are pressed. Each bit represents the corresponding switch, and a high bit is pressed while a low bit is not pressed. In the example, converting from ASCII hex gives the 16-bit number of 0x146A. That corresponds with bits 12, 10, 6, 5, 3, and 1 being high. The switch numbers are 0-indexed in this response, so that corresponds with switches 13, 11 ,7, 6 ,4, and 2 being pressed.

Example 0x146A:

Identifier	Bank Number	Switch ->	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
		Index ->	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x71	1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x72	2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x73	3		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74	4		0	0	0	1	0	1	0	0	0	1	1	0	1	0	1	0

8. Saving Images Using Engineering Kits Communicator

The Engineering Kits Communicator loads the images in alphanumeric order according to the image files names. It auto-assigns a sequential address to each image. Be sure to keep this in mind when naming images so that video images or animations are listed in the desired order. Avoid using symbols in the names as some symbols interfere with alphanumeric ordering. All images to be loaded should be saved in a single folder. The default starting address is 0001. This can be changed if needed.

To save images to the system:
1. Open the Engineering Kits Communicator.
2. From the drop-down menu at the top, select the COM port of the system (usually the last one).
3. Click the 'Open Port' button.
4. Press the call button and verify the system responds with '61' in blue text in the left text box.
5. Select the image type from the drop-down in the 'Loading Images' section.
6. Click the 'Import Images' button.
7. Navigate to the directory with all the images and select one and click 'Open'.
8. Note that the images are loaded alphanumerically and automatically assigned addresses.
   a. If some/all images do not show up in the image list after selecting the directory, it is because the image is not in the proper resolution or file type (.bmp). Double-check the image size is correct *before* downloading. If an image was skipped the images will load one address off and will have to be erased before reloading them.
9. If images were previously saved, click the 'Erase Flash' button.
   a. Note that this operation can take up to **2 minutes**.
10. Click the 'All selected images' button at the bottom.
11. Wait for the 'Success' message. If the process fails, click the 'All selected images' button again.

If writing custom software to save images, all data after the command must be sent in ASCII hex (See Sections Images and ASCII Hex).

9. Switch Bank Setup

The switches are setup up in 4 "banks" of 16 switches each, as shown below:

Bank 1 Switch 1	Bank 1 Switch 2	Bank 1 Switch 3	Bank 1 Switch 4	Bank 2 Switch 1	Bank 2 Switch 2	Bank 2 Switch 3	Bank 2 Switch 4	Bank 3 Switch 1	Bank 3 Switch 2	Bank 3 Switch 3	Bank 3 Switch 4	Bank 4 Switch 1	Bank 4 Switch 2	Bank 4 Switch 3	Bank 4 Switch 4
Bank 1 Switch 5	Bank 1 Switch 6	Bank 1 Switch 7	Bank 1 Switch 8	Bank 2 Switch 5	Bank 2 Switch 6	Bank 2 Switch 7	Bank 2 Switch 8	Bank 3 Switch 5	Bank 3 Switch 6	Bank 3 Switch 7	Bank 3 Switch 8	Bank 4 Switch 5	Bank 4 Switch 6	Bank 4 Switch 7	Bank 4 Switch 8
Bank 1 Switch 9	Bank 1 Switch 10	Bank 1 Switch 11	Bank 1 Switch 12	Bank 2 Switch 9	Bank 2 Switch 10	Bank 2 Switch 11	Bank 2 Switch 12	Bank 3 Switch 9	Bank 3 Switch 10	Bank 3 Switch 11	Bank 3 Switch 12	Bank 4 Switch 9	Bank 4 Switch 10	Bank 4 Switch 11	Bank 4 Switch 12
Bank 1 Switch 13	Bank 1 Switch 14	Bank 1 Switch 15	Bank 1 Switch 16	Bank 2 Switch 13	Bank 2 Switch 14	Bank 2 Switch 15	Bank 2 Switch 16	Bank 3 Switch 13	Bank 3 Switch 14	Bank 3 Switch 15	Bank 3 Switch 16	Bank 4 Switch 13	Bank 4 Switch 14	Bank 4 Switch 15	Bank 4 Switch 16

10. Controller Hardware

External connections to board. Ribbon cables omitted for clarity. Product may vary in appearance.
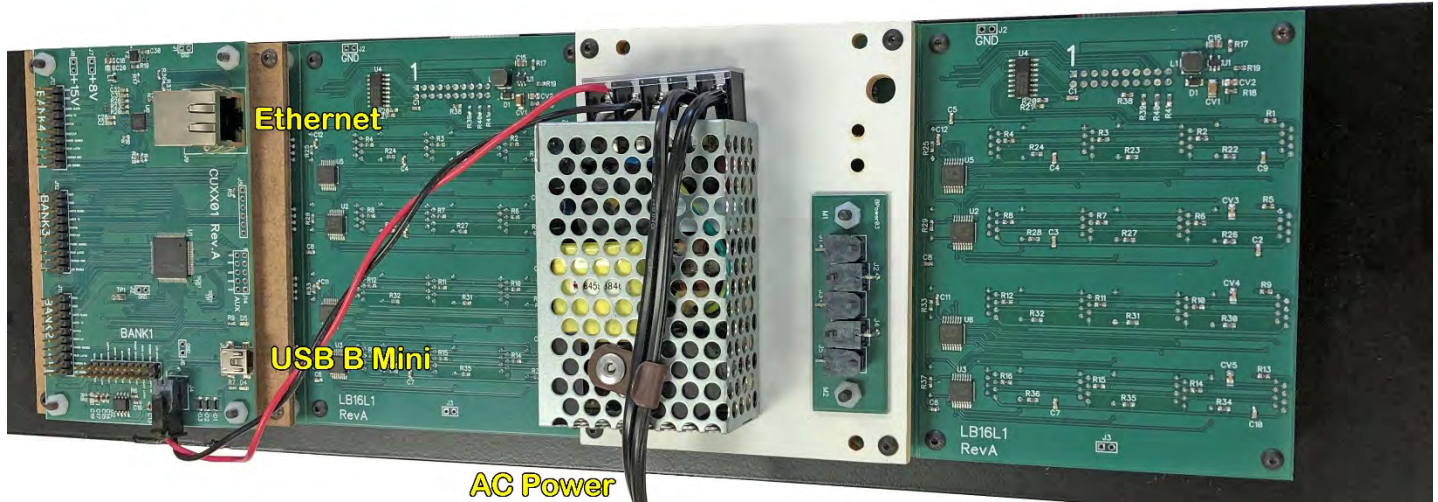
- The USB connection automatically shows up as a COM device in Windows 10. Earlier versions of windows may require a driver. All communication is done using serial over USB. The communication is baud-rate agnostic.
- Ethernet can be configured for TCP or UDP. Both DHCP and static IP addresses are supported.

11. Board Dimensions

Typical dimensions. The system is designed to fit into a 3U rack mount region.
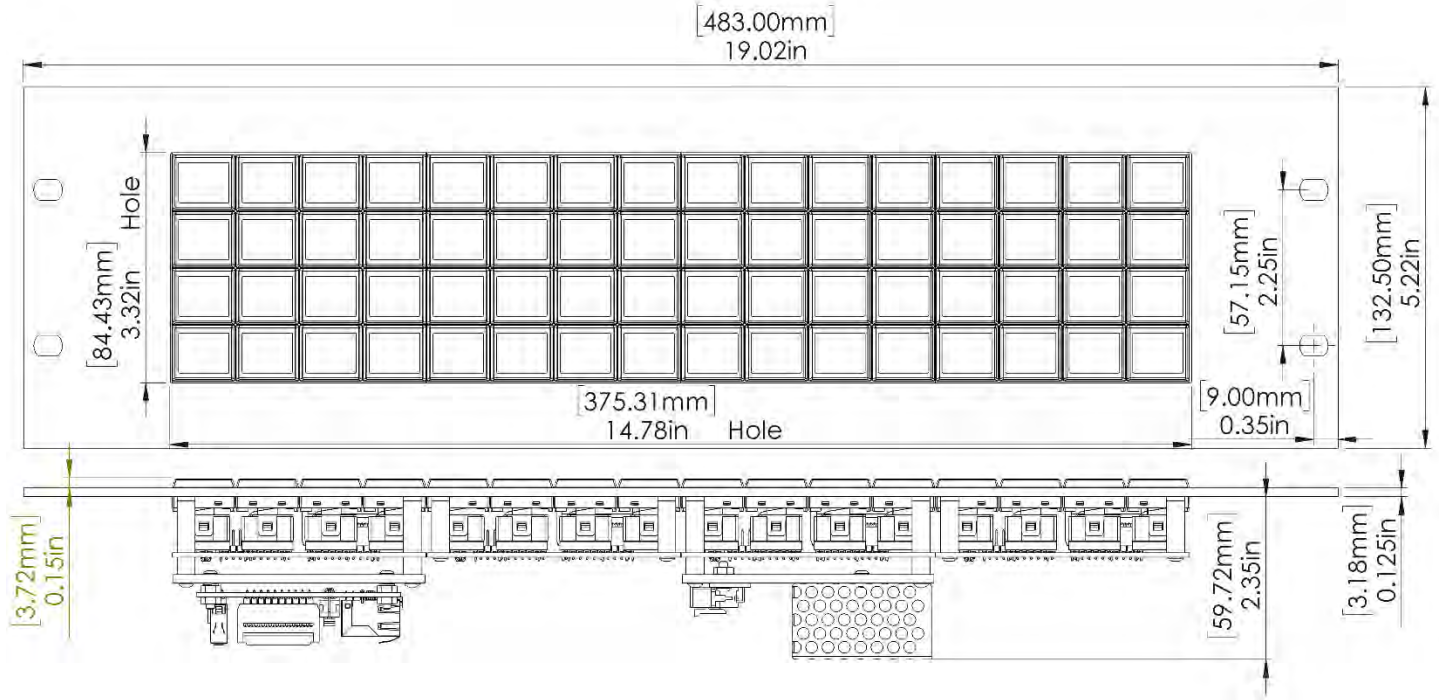
12. ASCII Hex

All data (except live data) is sent as ASCII hex as a safety measure to avoid being interpreted as a command. ASCII hex is a normal data byte split into two halves and converted to their ASCII equivalent (see www.asciitable.com). Therefore, all data received will be in the 0x30-0x46 number range, and anything received outside that is invalid. For example:

Data to be sent is 0x3D.
Each nibble is assigned its own byte: 0x3 and 0xD.
Each of those bytes is converted to ASCII equivalent: 0x33 0x44 (0x33 is the ASCII number '3'. 0x44 is the ASCII letter 'D').

Receiving is the same process reversed:

Data received is 0x41 0x37.
Each byte converted from the ASCII equivalent is 0xA 0x7 (0x41 is the ASCII letter 'A', 0x37 is the ASCII number '7').
Combine the two bytes to get the data byte 0xA7.

13. Key Terms & Definitions

Host   Any computer, terminal, or other device that can communicate over the USB line.

Byte   An eight-bit hex value ranging from 00H to FFH (Decimal 0 to 255). The bit format of a byte is: (B7 B6 B5 B4 B3 B2 B1 B0) where B7 is most significant and bit B0 is least significant bit.

Nibble/Hex Digit   A four-bit value ranging from 0H to FH. A byte consists of two nibbles.

ASCII   A byte value representing a symbol.

Communication Format   There are two formats to transmit a byte:

1. **Hex format** - A hex byte is transmitted without any change to it. [**xxH**] will be used to denote this.

   All commands and some data are sent by using this format.

2. **ASCII HEX format** - Each nibble of the byte is converted to ASCII code and sent as a byte. [**xxAH**] will be used to denote this.

   For example, the hex byte 5AH is transmitted in two bytes, **35H** and **41H**. The ASCII value for **5** is **35H** and the ASCII value for **A** is **41H**.

   All addresses and most data are sent using this format.

Address   A two-byte value ranging from 0001H to 03E8H representing the 1000 memory locations for images on the flash memory.

14. Warranty

NKK SWITCHES LIMITED WARRANTY AND LIMITATION OF LIABILITY

The following limits our liability. Please read.

NKK Switches hereby warrants this product against any and all manufacturing defects for a period of one year from the date of sale of this product to the original end user. NKK Switches' liability in the event of such defect is limited to repair or replacement of the defective products. NKK Switches disclaims any liability or warranty obligation with respect to any product that is misused, damaged by any user, or not used in conformity with all applicable product specifications.

NKK SWITCHES HEREBY DISCLAIMS ANY WARRANTY, EXPRESS OR IMPLIED, OTHER THAN THAT CONTAINED HEREIN. NKK SWITCHES EXPRESSLY DISCLAIMS THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND SHALL HAVE NO LIABILITY BASED ON OR ARISING FROM ANY CLAIM OF SUCH WARRANTY.

NKK Switches shall have no liability to any person for any incidental, consequential, special, punitive, or other damages of any kind whatsoever relating to any use of this product.

USE OF THIS PRODUCT IN CONNECTION WITH ANY LIFE CRITICAL APPLICATION IS NOT RECOMMENDED.

15. Commands to the Controller

See the Command List on the next page.

SmartDisplay Command Panel LCD 36x24 Command List

ASCII Hex

An ASCII Hex byte is a normal hex byte split in two halves and converted to their ASCII equivalent (www.asciitable.com). This is a safety measure so that all data sent isn't accidentally interpreted as a command. Most data sent after commands and sub-commands are in ASCII Hex. The only exception is when sending live images (images not saved in memory and sent directly to the switches)

Example: To set bank 4, switch 4 to image 679 (0x02A7):
Switch 4 is 0x03 (0-based), change to 2 bytes: 0x30 (ASCII '0') and 0x33 (ASCII '3')
Image high byte 0x02 to 2 bytes: 0x30 (ASCII '0') and 0x32 (ASCII '2')
Image low byte 0xA7 to 2 bytes: 0x41 (ASCII 'A') and 0x37 (ASCII '7')
Using the table below, this would result in the command:
0x2D 0x51 0x74 0x33 0x30 0x32 0x41 0x37

Conversion **TO** ASCII Hex:
$x = ((data\ \&\ 0xF0) >> 4)$
$y = ((data\ \&\ 0x0F) >> 0)$
If ($0x0 <= x <= 0x9$) x+= 0x30
If ($0xA <= x <= 0xF$) x+= 0x37
If ($0x0 <= y <= 0x9$) x+= 0x30
If ($0xA <= y <= 0xF$) x+= 0x37

Conversion **FROM** ASCII Hex:
x = ASCII hex byte 1
y = ASCII hex byte 2 (conversion not shown)
z = converted byte
If ($0x30 <= x <= 0x39$) x -= 0x30
If ($0x41 <= x <= 0x46$) x-= 0x37
$z = (x << 4) + y$

Command	Sub-Command	Description	Command Format	Sending example	Notes
0x01		Queries the controller if it is up and running (Controller responds with 0x61)	**0x01** (1 hex byte)	0x01	
0x21	0x55 0xAA 0x52 0x52	Erases all images on the flash memory	**0x21 0x55 0xAA 0x52 0x52** (5 hex bytes)	0x21 0x55 0xAA 0x52 0x52	Responds with 0x65 until the erase is complete, then sends 0x79
	0x55 0xAA 0x52 0x52 AA	Erases all images and settings on the flash memory	**0x21 0x55 0xAA 0x52 0x52 0xAA** (5 hex bytes)	0x21 0x55 0xAA 0x52 0x52 0xAA	Responds with 0x65 until the erase is complete, then sends 0x79
	0x55 0xAA 0x52 0x52 DD	Erases all images on the flash memory	**0x21 0x55 0xAA 0x52 0x52 0xDD** (5 hex bytes)	0x21 0x55 0xAA 0x52 0x52 0xDD	Responds with 0x65 until the erase is complete, then sends 0x79
	0x55 0xAA 0x52 0x52 EE	Erases all settings on the flash memory	**0x21 0x55 0xAA 0x52 0x52 0xEE** (5 hex bytes)	0x21 0x55 0xAA 0x52 0x52 0xEE	Responds with 0x65 until the erase is complete, then sends 0x79
0x22	0x51	Get IP settings	**0x22 0x51** (2 hex bytes)	0x22 0x51	Responds with 0x01 followd by the IP address in ascii hex separated by '.'s, followed by a ':', followed by the current port
	0x52	Set IP settings	0x22 0x52 (2 hex bytes) **0xAA** (1 ASCII hex bye of ethernet type) **0xBB** (1 ASCII hex byte of DCHP Enable) **0xCC 0xDD 0xEE 0xFF** (4 ASCII hex bytes of IP address) **0xGG 0xHH** (2 ASCII hex bytes of port number) **0xJJ 0xKK 0xLL 0xMM** (4 ASCII hex bytes of subnet mask) **0xNN 0xPP 0xQQ 0xRR** (4 ASCII hex bytes of Gateway) **0xSS 0xTT 0xUU 0xVV** (4 ASCII hex bytes of DNS)	0x22 0x52 0x30 0x31 0x30 0x30 0x43 0x30 0x41 0x38 0x30 0x31 0x44 0x32 0x32 0x36 0x32 0x30 0x46 0x46 0x46 0x46 0x46 0x46 0x30 0x30 0x43 0x30 0x41 0x38 0x30 0x31 0x30 0x31 0x43 0x30 0x41 0x38 0x30 0x31 0x30 0x35 (Sets to UDP, DHCP off, IP 192.168.1.210, Port 9760, Subnet 255.255.255.0, Gateway 192.168.1.1, DNS 192.168.1.5)	Mode: 0/1/2 = None/TCP/UDP DCHP: 0/1 = Off/On
0x24		Resets the controller	**0x24** (1 hex byte)	0x24	
0x25	0x51	Saves LED codes to flash memory	**0x25 0x51** (2 hex bytes) **0xAA** (1 ASCII hex byte of code offset) **0xAABBCC……** (256 ASCII hex codes of LED data)	0x25 0x51 0x30 0x32 0x34 0x32 0x31 0x34 0x31 0x31 0x34 0x31 (Save LED codes, offset of 512, image 1 on/off cycle = blue/green, image 2 = red/blue, image 3 = red/red, image 4 = blue/red)	offset = 256 * number LED codes sent as xBGRxBGR, where x is don't care, the first BGR is the off cycle, and the second BGR the on cycle
0x26		Retrieves the firmware version number	**0x26** (1 hex byte)	0x26	Returns the current version number in ASCII hex. It will be the letter 'v' followed by 4 ASCII hex bytes signifying the major version, followed by '.', followed by 4 ASCII hex bytes signifying the minor version number Example: 0x76 0x30 0x30 0x31 0x2E 0x30 x030 0x30 0x30 in ASCII is v0001.0000
0x28	0x51	Saves an image to the flash memory	0x28 0x51 (2 hex bytes) 0xAA 0xBB 0xCC 0xDD (4 ASCII hex bytes for the image address) After the controller responds with 0x61, send image data in ASCII hex. The controller responds with 0x79 when the save completes.	0x28 0x51 0x30 0x30 0x30 0x31 (Send an image, save at address 0x0001) (Followed by bytes of image data in ASCII hex after the controller responds with 0x61)	
0x2A	0x51 0x50	Sends a full live image (image not saved in memory, sent directly to switches)	**0x2A 0x56 0x50 0xAA** (1 hex byte bank #, 0x71-0x74) **0xBB** (1 hex byte switch number, 0 based) If the controller responds with **0x61**, send image data. If the controller responds with **0x65** (busy), re-send the command until the controller responds with **0x61**, then send the image data. The controller responds with 0x79 when the command completes.	0x2A 0x56 0x50 0x72 0x03 (Send an image, send to bank 2, switch 4) (Followed by image data after the controller responds with 0x61)	Switch number is 0 based. To optimize performance, stagger images between banks (IE send an image to bank 1, then bank 2, then bank 3, then bank 4, then send the next bank 1 image)
0x2D	0x51 0x60 0x71-74	Sets the image on a particular switch from an image saved in flash memory	**0x2D 0x51 0x60** (3 hex bytes) **0xAA** (1 hex byte bank #, 0x71-0x74) **0xBB 0xCC** (2 ASCII hex bytes switch #, 0 based) **0xDD 0xEE 0xFF 0xGG** (4 ASCII hex bytes for the image address)	0x2D 0x51 0x60 0x71 0x30 0x33 0x30 0x30 0x41 0x37 (Set image, bank 1, switch 4, image 167 (0x00A7))	Switch number is 0 based
	0x51 0x63 0x71-74	Sets the backlight color of a particular switch	**0x2D 0x51 0x63**(3 hex bytes) **0xAA** (1 hex byte bank #, 0x71-0x74) **0xBB 0xCC** (2 ASCII hex bytes switch #, 0 based) **0xDD** (1 ASCII hex byte for the backlight code)	0x2D 0x51 0x63 0x72 0x30 0x31 0x34 0x32 (Set backlight, bank 2, switch 1, on/off cycle = blue/green)	LED codes sent as xBGRxBGR, where x is don't care, the first BGR is the off cycle, and the second BGR the on cycle

Response	Description	Response Format	Response Example	Notes
0x50	Switch Pressed	**0x50 0xAA** (Bank #, 0x71-0x74) **0xBB 0xCC 0xDD 0xEE** (4 ASCII hex bytes of switch number	0x50 0x74 0x31 0x34 0x36 0x41 (Switch press/release, bank 4, switches 13,11,8,4, and 2 are pressed)	The highest bit of the switch number is switch 16 state, the next one down is switch 15 state, etc. so receiving 0001 0100 1000 1010 (0x14 0x8A after ASCII hex conversion) means switches 13,11,8,4, and 2 are all pressed
0x61	Ready/Command received	**0x61**	0x61	
0x65	Busy processing previous command	**0x65**	0x65	If there was data associated with this command, it is ignored. The system is busy with the previous command. If sending for a specific bank, either re-send the command until 0x61 is received, or send data for a bank that is not busy
0x6E	Error, followed by error code	**0x6E**	0x6E 0x02	Error Codes: 01 - Invalid Command 02 - Timeout 03 - Not enough data 04 - Invalid data
0x79	Command successful	**0x79**	0x79	