1 Funktionsüberblick

Die hier vorgestellte Schaltung bietet eine sehr einfache und preisgünstige Möglichkeit, die vielerorts eingesetzten LED-Matrix RGB Displaymodule anzusteuern. Verwendet werden sie hauptsächlich als Videowände, Werbedisplays oder Präsentationsdisplay. Diese meist großen Displays werden aus einzelnen Modulen mit je 32x32, 64x64 oder auch anderen Maßen eingesetzt. Zahlreiche Anbieter (insb. aus China) bieten neben den Großdisplays auch Einzelmodule für kleines Geld zum Kauf an. Hier geht es um die Ansteuerung eines einzelnen oder weniger Einzelmodule zum Eigenbau eines sehr dekorativen und technisch sehr universellem Matrix Displays mit 64x64 einzeln steuerbaren RGB LEDs. Solche Einzelmodule gibt es in verschiedenen mechanischen Abmessungen, wobei die Anzahl der LEDs gleichbleibt, jedoch der Abstand variiert. Gängige Abmessungen sind 2mm (sog. P2 Modul), 2,5mm und 3mm. Dies führt dann zu Abmessungen zwischen 128mmx128mm bis zu 192mmx192mm. Die mechanische Größe ist für die hier beschriebene Schaltung nicht relevant und kann beliebig gewählt werden.

Der elektrische Aufbau eines solchen Moduls besteht i.W. aus mehreren langen Schieberegistern, welche jeweils eine LED-Reihe betreibt. Für ein typisches 64x64 LED-Modul werden 6 Schieberegister mit je 64 Bit verwendet. Jeweils 3 für die oberen 32 Reihen und 3 für die unteren 32 Reihen. Jede 3er Gruppe treibt dabei jeweils die roten, grünen und blauen RGB-LEDs an. Die Eingänge der genannten 6 Schieberegister sind an 6 Pins als jeweils 2 RGB Gruppen herausgeführt.

Von den 32 oberen bzw. den 32 unteren LED-Reihen kann immer nur eine gleichzeitig leuchten. Die Selektion dieser Reihe erfolgt über 5 herausgeführte Adressleitungen. Dabei sind obere Hälfte (32 Reihen) und untere Hälfte (32 Reihen) gleichgeschaltet, d.h. wenn oben die 3. Reihe angesteuert wird, ist auch unten die 3. Reihe aktiviert.

Neben den o.g. 11 herausgeführten Leitungen sind noch 3 weitere zur Ansteuerung nötig. Dies sind einmal das CLK Signal, welches den Schiebetakt der 6 Schieberegister vorgibt. Des Weiteren gibt es das LATCH Signal, mit dem die eingeschobenen Daten der Schieberegister in die sog. Output-Register übernimmt. Die Output-Register - jedes Schieberegister verfügt darüber wobei jedes Bit des Schieberegisters ein entsprechendes Bit im Output-Register hat - steuern dann die LEDs an. Das dritte Steuersignal ist das OE (= Output Enable) Signal. Hiermit lassen sich alle Output-Register gleichzeitig aktivieren oder abschalten. Sind die Register abgeschaltet, so sind alle LED aus, sind sie aktiviert, so zeigen die LED die zuletzt per LATCH übernommenen Daten. Die OE Leitung wird z.B. zur Steuerung der Gesamthelligkeit des Panels genutzt.

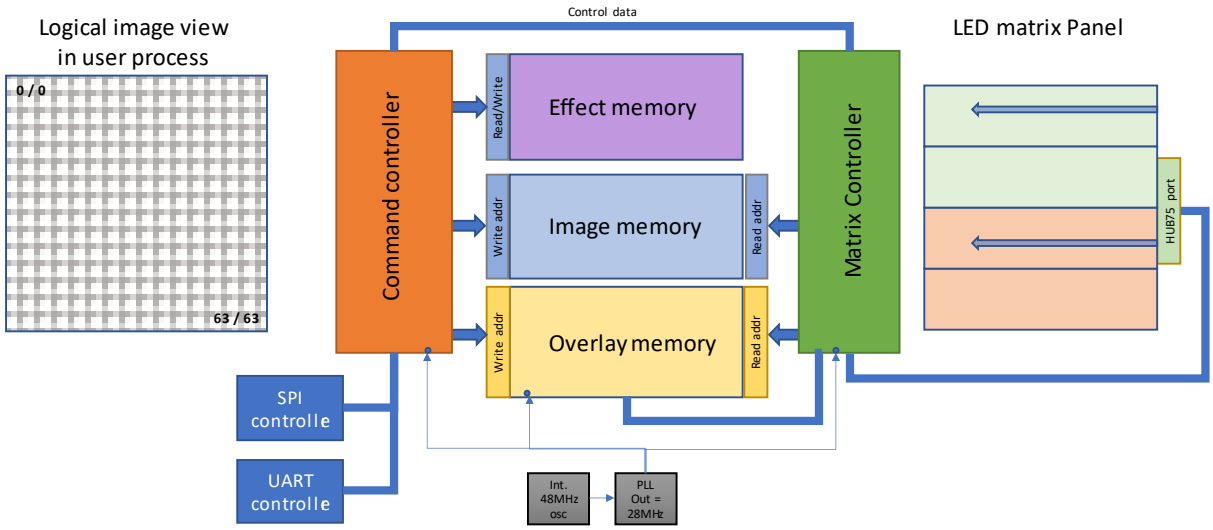
Der gesamte Anzeigevorgang ist also simpel (hier ohne OE Steuerung):

1. Adresszähler <AZ> zur Ansteuerung der Zeilen auf 0 stellen und Ansteuern der Reihe über die 5 Adressleitungen
2. Einschieben von 64 Bit für Zeile <AZ+1> MOD 32 in die Schieberegister über die 6 RGB-Leitungen per CLK. Während des gesamten Schiebevorgangs zeigt die Matrix die Zeile <AZ> und damit die Daten des vorherigen Schiebevorgangs an.
3. Erzeugen eine LATCH Impulses zur Übernahme der Daten in die Output-Register
4. <AZ> um 1 erhöhen und weiter bei (2). Wenn <AZ> den Wert 32 erreicht, bei (1) fortfahren.

Um sog. Glitching zu vermeiden, kann man während des Umschaltens des Zeilenzählers das OE-Signal einsetzen.

Zum Timing: Um einmal alle 4096 LED zu bedienen braucht man mindestens 64 x 32 = 2048 Takte. Soll eine Farbtiefe von 8 Bit erreicht werden, so muss man den Vorgang 256-mal wiederholen und dabei die RGB Helligkeiten durch PWM oder BCM Modulation steuern. Dafür braucht man dann schon über 500.000 Takte für ein Bild. Um ein Bildflimmern zu vermeiden, sollte man mindesten 50 Bilder (bei BCM reicht das, bei PWM bleibt ein Flimmern sichtbar) pro Sekunde erzeugen, so dass man mindestens ca. 26 MHz Schiebetakt benötigt. Die hier eingesetzten LED-Panels haben einen maximalen Schiebetakt von ca. 30MHz.

Die hier beschriebene Schaltung generiert das notwendige Timing wie oben grob beschrieben mit Hilfe eines sehr preisgünstigen FPGA von Lattice Semiconductor. Der Baustein ICE40UP5K verfügt über ausreichend Ressourcen, um ein 64x64 RGB-Modul in 24bit Farbtiefe anzusteuern und über ein Command-Interface per SPI oder UART zu kontrollieren. Das folgende Schema zeigt die Funktionsblöcke der FPGA-Konfiguration:



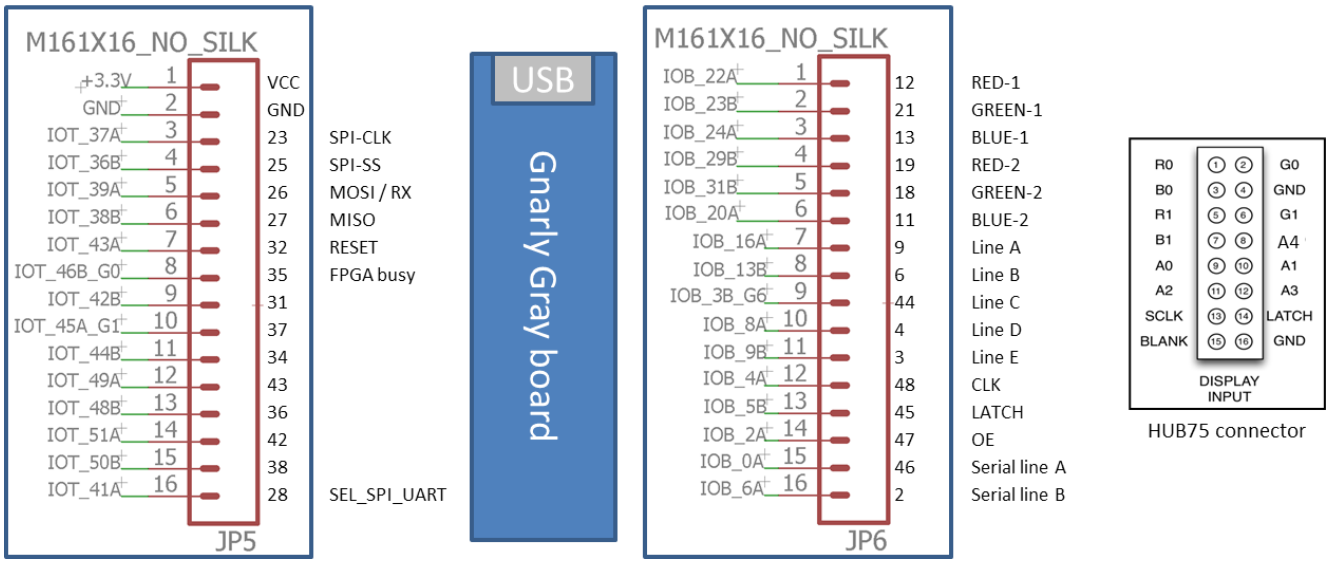
Funktionsüberblick: Mit Hilfe des integrierten 48MHz Oszillators sowie der PLL wird ein zentraler Takt von 28MHz erzeugt. Dieser Takt versorgt alle anderen Komponenten mit einem synchronen Takt und dient auch als CLK Signal für das LED-Panel. Der ‚Matrix Controller' erzeugt die notwendigen Steuer- und Datensignale für den Anschluss Port (HUB75E Format) des LED-Panels. Die Daten der beiden RGB-Ports werden synchron zum 28MHz Takt aus zwei BRAMs – dem Image Memory und dem Overlay Memory – geladen und seriell an die RGB-Ports geleitet. Der ‚Matrix Controller' kann zwischen verschiedenen Panel-Layouts unterscheiden. Neben dem 64x64, 1/32 Scan Layout wie oben beschrieben, können auch andere Layouts wie z.B. 128x32, 1/16 Scan oder auch ein Daisy Chain aus 2 Panels à 64x32, 1/16 Scan betrieben werden. Des Weiteren werden 2 verschiedene Typen von Schieberegistern unterschieden: Typen mit einfachem, ungetaktetem LATCH Signal (wie z.B. ICN2037), sowie Typen mit getaktetem LATCH Signal, wie z.B. FM6126A. Bei diesen Typen muss das LATCH-Signal genau 3 CLK Zyklen auf HIGH gesetzt sein, um die Daten zu übernehmen.

Über einen SPI Slave-Controller oder umschaltbar über einen UART Controller gelangen die Daten vom Host-Processor wie z.B. ESP32, AVR, ARM etc. in den ‚Command Controller'. Eine sehr umfangreiche FSM modifiziert basierend auf den übertragenen Kommandos (siehe Kap. 3 - SPI and UART command protocol) die drei Speicherbereiche ‚Effect Memory', ‚Image Memory' und ‚Overlay Memory'. Dabei sind ‚Image Memory' und ‚Overlay Memory' als Dual-Port RAMs implementiert, so dass gleichzeitig der ‚Command Controller' schreiben und der ‚Matrix Controller' lesen kann.

2 Umsetzung der Hardware

Eine einfache Grundlage für den Aufbau der benötigten Hardware kann z.B. ein UPduino (Gnarly Grey) Board sein. Das folgende Bild zeigt die Anschlussbelegung für dieses Board:

Gnarly Grey DevBoard with Lattice ICE40UP5K
Interface to an 64 x 64 LED matrix display with HUB75 connector

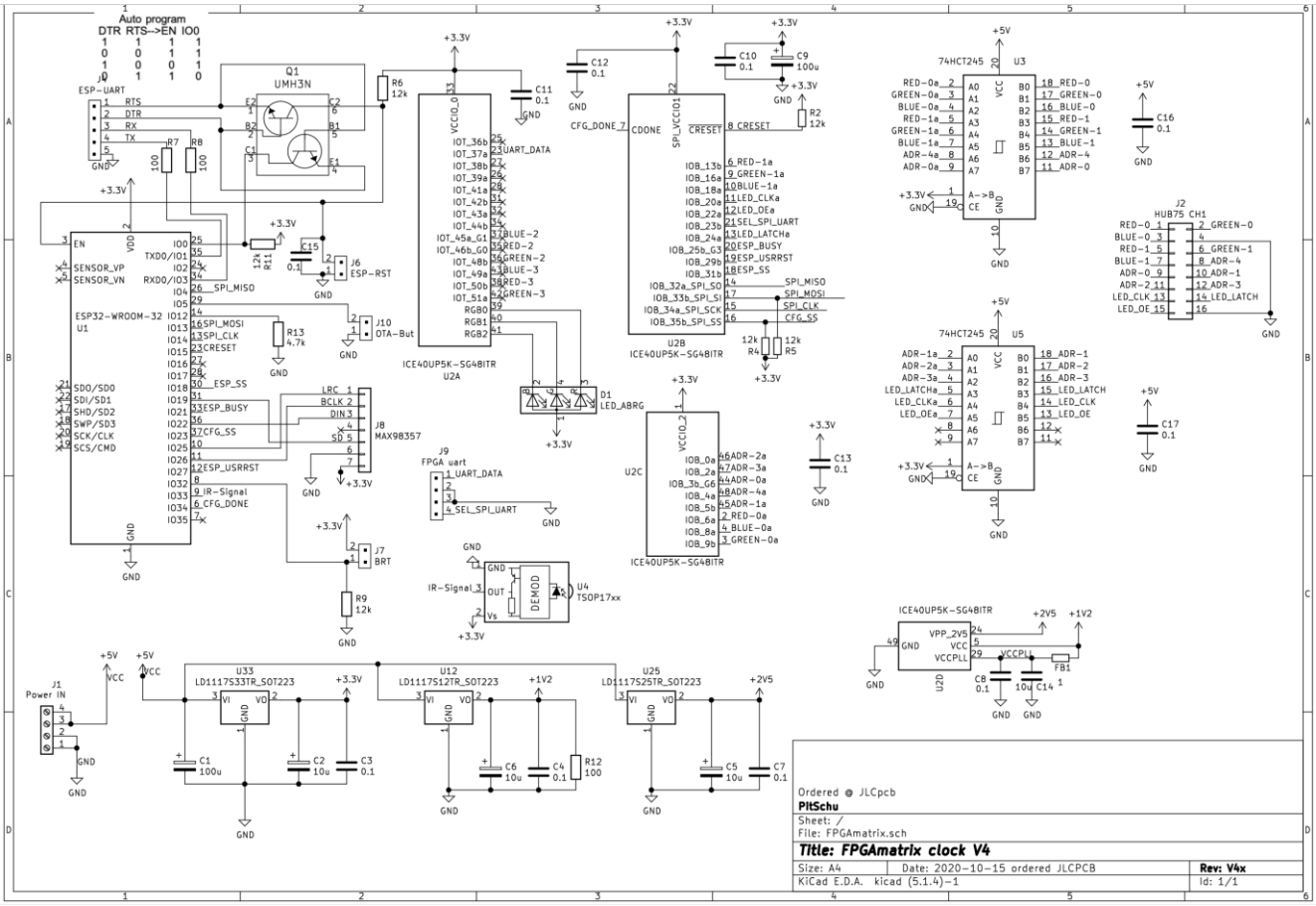


Dieses Board ist sehr kostengünstig zu beziehen (ca. 16 – 18€) und bietet den Vorteil, dass die Programmier-Hardware in Form eines FTDI-Chips direkt on-Board ist. Eine einfache Konfiguration ist daher über den Lattice Diamond Programmer oder auch über Tools wie ‚iceprog' möglich. Nachteilig ist, dass das UPduino Board nur 3,3V Ausgangspegel hat. Für LED-Panels, welche üblicherweise mit 5V betrieben werden, kann dieser Pegel etwas zu gering sein. Eine einfache Abhilfe ist, das Panel statt mit 5V einfach mit ca. 4.5V zu betreiben. Dann reichen 3.3V Pegel locker aus.

Neben den bereits beschriebenen Leitungen des HUB75E Ports (2xRGB, 5 Adressleitungen und 3 Steuerleitungen) sind folgende Verbindungen zum UPduino Board nötig:

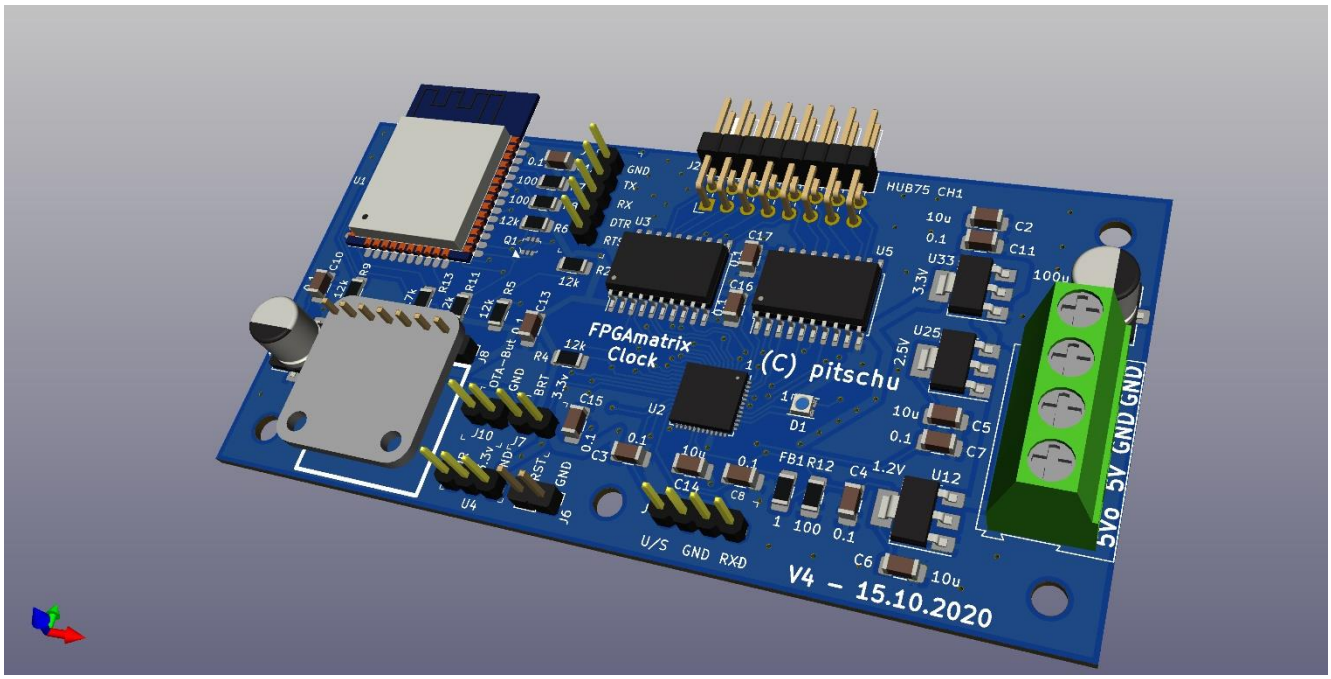
- ***SPI-CLK***: SPI-Takt vom SPI-Master (= Prozessor) zum FPGA Board (SPI-Slave). Der maximale Takt beträgt ca. 10MHz. Daten der MOSI-Leitung werden mit steigender Flanke übernommen.
- ***SPI-SS***: Slave-Select Leitung des SPI. Das erste übertragene Byte nach Aktivieren (LOW) der SPI-SS Leitung ist das Command-Byte. Alle weiteren Bytes sind Command-Parameter. Um einen neuen Befehl (command) zu starten, muss die SPI-SS Leitung kurz (min 1 CLK) auf HIGH gezogen werden.
- ***MOSI / RX***: Die SPI-Datenleitung vom Prozessor zum FPGA-Board bzw. die UART-RX Leitung falls statt SPI die UART Verbindung genutzt wird. Die Umschaltung erfolgt über die SEL_SPI_UART (s.u.).
- ***MISO:*** Wird hier nicht genutzt. Das FPGA Board überträgt zurzeit noch keinerlei Daten zum Host Prozessor.
- ***RESET:*** Über die RESET-Leitung kann das FPGA zurückgesetzt werden. Dies ist ein sog. User-Reset, durch ihn wird nicht die FPGA Konfiguration gelöscht oder neu geladen, sondern lediglich einige Parameter auf einen Defaultwert zurückgestellt.
- ***FPGA-busy***: Diese Leitung zeigt über einen ‚1'-Pegel an, dass das FPGA z.Z. einen Befehl abarbeitet und keine neuen Befehle annehmen kann.

Eine ähnlich aufgebaute Hardware, jedoch mit TTL-Ausgangstreibern sowie integriertem ESP32 Hostprozessor sowie verschiedenen Sensoren und Audio-Ausgabe zeigt folgendes Schaltbild:



Bei diesem Design wurde auf ein FPGA-Configuration Flash-ROM verzichtet. Das FPGA wird beim Start über die SPI-Schnittstelle mit einer entsprechenden Konfiguration versorgt und dann gestartet.

Eine Prototypenserie dieses Layouts wurde bereits realisiert:



2.1 Hinweise zur UART Schnittstelle

Gegenüber der SPI-Schnittstelle, die rein binäre Daten gemäß der in Kap. 3 beschriebenen Befehle erwartet, ist die UART-Schnittstelle eine Text basierende ASCII-Schnittstelle. Das bedeutet, dass diese Schnittstelle z.B. über eine beliebiges Terminalprogramm bedient werden kann. Um z.B. das Kommando ‚Clear screen with green color' auszuführen, erwartet der ‚Command Controller' die Byte-Sequenz ‚40 00 ff 00'. Über das SPI-Interface würden genau diese 4 Bytes übertragen. Über die UART-Schnittstelle sind jedoch die folgenden 8+1 Zeichen zu übertragen:

4 0 0 0 f f 0 0<cr>

Die hier eingeschobenen Spaces sind nicht zu übertragen. Das abschließende <cr> (Cariage Return) beendet einen Befehl und muss immer zwischen 2 Befehlen gesendet werden.

Die UART-Schnittstelle bietet damit eine sehr einfache Möglichkeit, sowohl manuell für Debugging Zwecke oder auch automatisch bzw. Script gesteuert die LED-Matrix zu bespielen.

3 SPI and UART command protocol

CMD	Description	Parameters
01	Draw pixel at	X Y R G B
02	Set cursor position	INC X Y
03	Draw many pixels	R G B {R G B}……
04	Draw <len> of pixels	LEN R G B
05	Draw <len> of pixels at	INC X Y LEN R G B……
06	Draw many pixels in line Y starting at X	X Y R G B {R G B}……
10	Switch on/off 'hidden write'	ONOFF + SCRBUF#
22	Set window and cursor mode	INC LEFT TOP WID HIG
23	Set alpha channel (invisible color)	ONOFF R G B
30	Perform scroll effect step number STEP	STEP
31	Perform scroll steps with delay	DELAY FROM TO
32	Set scroll effect background color	R G B
33	Set scroll effect type	TYPE
40	Clear screen with color	R G B
41	Fill window with color	R G B
42	Flood window starting at current cursor	{R G B} {R G B}……
45	Blank Overlay buffer	---
46	Set Overlay colors 1 - 3	{R G B} {R G B} {R G B}
47	Set Overlay mode ON/OFF	MODE
51	Set CLUT mode (color lookup table)	MODE
52	Fill CLUT	R G B {R G B}…….
70	Set brightness level	BRT
81	Set FPGA defaults	-
82	Set number of pixel planes; ori is ignored	PIXMASK ORIENTATION
83	Map board RGB lines to panel RGB lines	RGB-2-RGB
84	Set FPGA options	FPGAOPTS
85	LED panel config (only for special panels)	LEN MSB LSB
86	Set UART baudrate	BR-CODE (0..7)
Fx	Set ICE40UP5K PWM module parameters	CMD DATA

Draw pixel at **01** **X Y R G B**

Draw pixel at <X/Y> in window with color <RGB>
X and Y are relative to window position (cmd 22)

Set cursor position **02** **INC X Y**

Set cursor to position X/Y relative to window and set auto increment values to:
INC(0) = 1: the X position is auto incremented after any drawing command
INC(1) = 1: the Y position is auto incremented ...

Draw many pixels **03** **R G B {R G B} {R G B} {R G B} ....**

Draws adjacent pixels with RGB color starting at the current cursor position. The cursor position is incremented after each pixel in the direction defined by the last command which sets the INC value.
X and Y does NOT wrap around when window border is reached,

Draw <len> of pixels **04** **LEN R G B**

Draws LEN pixels with color RGB in the direction defined by last command 02.
Command ends when X or Y screen limit is reached.

Draw <len> of pixels at **05** **INC X Y LEN R G B …**

Combines cmd 02 and 04 in one transaction.

Draw many pixels **06** **X Y R G B {R G B} {R G B} {R G B} ....**

Draw many pixels in line Y starting at pos X.
Sets INC value to 01.

Hidden Write mode and set screen buffer **10** **SCR# | ONOFF**

When ONOFF (bit [0]) is set to 1, all subsequent writes to the screen are only written to the effect screen buffer and not to the screen. This is used to fill the effect buffer with new content without showing it immediately. Perform a ‘scroll effect’ step command ’30 00’ to flush the effect buffer to the screen. Setting ONOFF (bit 0) to 0 switches the mode OFF but does not flush the shadow buffer to the screen.
Bits [2:1] sets one of 4 different effect screen buffers which are used during scroll effects.
Any DRAW, CLEAR, BLANK or SET PIXELS operation always changes the selected effect screen buffer. Only if the HIDDEN flag is FALSE, the same operation takes place in the main screen buffer.

Set window **22** **INC LEFT TOP WID HIG**

Set painting window LEFT, TOP, WIDTH and HEIGHT of virtual window.
Cursor position is set to <TOP, LEFT>

Set Alpha channel **23** **ONOFF R G B**

Set alpha channel color to RGB. If ONOFF is 1 then alpha channel is used else it's inactive.

Perform scroll effect step **30** **STEP**

Performs the scroll effect step number <STEP> of the scroll type defined by command 33. Only the defined window (cmd 22) is affected. <STEP> ranges from 0 (= original content) to 2 * <window size> in X or Y direction – whichever is larger. The effect

uses a effect memory buffer which always has the same content as the main screen buffer before the scroll effect is started (but see command 10 – Hidden mode).

Perform scroll cycle **31 DELAY STEP_FROM STEP_TO**

Performs the scroll effect from step STEP_FROM to step STEP_TO. Between two steps a delay of <DELAY> milliseconds is inserted.
If STEP_FROM is 0xFF then step from is set to half of the full cycle (= <window size> / 2>.
If STEP_TO = 0 then step to is set to the end of a full cycle (= <win size>). If STEP_TO = 0xFF then step to is set to half of the full effect cycle (= <window size> / 2>.
Using these special values you can perform the following effect:

1. Draw any picture
2. set a partial window using CMD-22
3. Set Effect background with CMD-32
4. Run Effect cycle CMD-31 with STEP_FROM=0 and STEP_T>O=0xff. This scrambles the window.
5. Set the ‘hidden flag’ to TRUE using CMD 10.
6. Draw new picture into the window
7. Run Effect cycle CMD-32 with STEP_FROM=0xff and STEP_TO=0
8. Reset the ‘hidden flag’ to FALSE

Keep in mind that the BUSY pin is held high until the last effect step is finished.

Scroll Effect background color **32 R G B**

Set the background color of Scroll Effect. Pixels having the same color are ignored while stepping.

Set Scroll Effect type **33 TYPE**

There are several scroll effects possible:
00 : The Penny Arcade Effect moves pixels diagonally
01 : Scroll pixels left and mirror them on left edge
02 : Scroll and mirror to the right
03 : Scroll up
04 : Scroll down
05 : Implode (all pixels travel to center of screen and back)
06 : Flip vertical (= old flip effect cmd 50 in previous version)
07 : Flip horizontal
08 : Fader effect; cross fade from old to new image

Blank screen **40 R G B**

Blanks screen and fill with color RGB. Reset window to full screen. Sets INC to 01 and sets the Effect background color to RGB.

Fill window **41 R G B**

Fill window (cmd 22) with color RGB. INC is set to 01 and sets the Effect background color to RGB.

Flood window **42 {R G B} {R G B} {R G B} ....**

Flood window starting at actual pos X/Y using INC.; cursor is INCed after each pixel. Only INC values 01 or 10 are allowed (01 = line wise fill is used by default).

Clear overlay buffer **45**

The overlay buffer is a 2-bit 64x64 pixel layer which works in parallel to the regular screen buffer. When an overlay pixel is 00 then the regular screen pixel is shown. When the overlay pixel is 01, 10 or 11 then the corresponding overlay color (see cmd 46) is shown instead of the screen pixel. The overlay display is independent of any scroll effects etc..
The command 45 sets all overlay pixels to 00.

Set overlay colors — 46 {R G B} {R G B} {R G B}

Up to three overlay colors can be defined. These colors are used whenever an overlay pixel is 01, 10 or 11.

Set overlay mode ON/OFF — 47 MODE

MODE can be 0 or 1. If mode is 1, any further pixel operations are only written into the overlay buffer. All RGB byte sequences in all other commands (except cmd 46 – set overlay colors) are reduced to only 1 byte, and only bits 1:0 are written to the overlay.

Set CLUT / RAW mode — 51 MODE

Switch CLUT mode (Color Lookup Table) on or off.
MODE = 0 => full 24 bit is used
MODE = 1 => CLUT mode is switched on, all RGB byte sequences in all other commands (besides cmd 52 - fill clut) are reduced to only 1 byte for CLUT index.
MODE = 2 => 565 mode is used. All RGB byte sequences are shortened to 2 bytes. Top 5 bits are RED, middle 6 bits are GREEN and lowest 5 bits are BLUE.

Independent of the mode set, the LED screen is operated always in 24bit RGB mode. Switching the CLUT mode only affects the pixels painted after the change.

Fill CLUT table — 52 R G B {R G B} {R G B} {R G B} ....

Fill the CLUT with RGB values starting index 0 and up to index 255. The new CLUT is not applied to pixels already shown.

Set brightness — 70 BRT

Set master brightness to value BRT (0..PANEL WIDTH [64]) Values higher than PANEL WIDTH are equal to PANEL WIDTH

Set defaults — 81 -

Set some default values: BRT=32, FPGA options = 1,
CLUT mode is set to 0 (= 24bit RGB), PIXMASK is set to 8 planes, overlay mode OFF.

Set pix planes — 82 PIXMASK ORIENTATION

Set the number of pixel planes used in LED controller. Value can be between 4 and 8
If PIXMASK=8 you have full 24bit RGB color but slightly low refresh rate.
Other values then 4..8 defaults to 8.
Sets the image orientation in 90° steps. 0 = 0°, 1=90° ... [not yet implemented]

Set RGB-2-RGB map — 83 RGB2RGB

RGB2RGB is a 6-bit map in the format ‘00RRGGBB’. Each 2-bit pair controls a pin of the two RGB ports. If the bits are ‘00’ then the red color from the image is sent to the red color pin of RGB ports. For ‘01’ it is green color, ‘10’ is blue color. The default value is ‘00000110’ giving a standard mapping R->R, G->G and B->B.

This map can be used to connect a LED panel which does not have the regular HUB75 layout.

Set FPGA options — 84 FPGAOPTS

This command only applies to FPGA configurations named '… universal …'. Other versions named '…N…' or '…S…' ignore this command.
Only bit 0 of FPGAOPTS is used. This bit controls the LATCH line of the panel port. If the bit is 1 the LATCH line is compatible with shift registers like ICN2038S which needs 3 clock pulses while LATCH is high (S-type).
If FPGAOPTS[0] is 0, the LATCH line is compatible to shift registers with an unclocked LATCH impulse (N-type). After a FPGA reset, this bit is '1'.

Set LED panel shifter options — 85 LEN MSB LSB

Some LED panels have special shifter ICs which can be programmed by setting so called 'configuration registers'. Up to now I could not find any documentation about how to set these configuration registers. Anyway, this command offers a way to set the registers. The addressing of config registers is done by different length of the LATCH line of the panel. The length varies between 1 and 16 clocks per shift register and is defined by the LEN byte. The bytes MSB and LSB are then shifted into the shift registers with <panel width> pulses on the panel CLK line and the LATCH line is pulled high when the desired length (16 – LEN) is reached.

Set UART baud rate — 86 BR-CODE

The UART interface can be run with following baud rates:

CODE	Baud rate
0	9600
1	19200
2	38400
3	57600
4	115200 (= default after FPGA user reset)
5	230400
6	460800
7	921600

The interface always starts at 115200 baud when the FPGA is reset.

Set ICE40UP5K PWM module parameters — Fx CMD DATA

Sends a control byte to the ICE40UP embedded LED-PWM IP core. The lower nibble of the command byte (x) is the control register address (LEDD_ADDR(0..3)). The command byte is followed by a DATA byte stored in the control register. The following registers are defined (see: Appendix D. RGB PWM IP - LED Control Bus Addressable Registers in the iCE40 LED Driver Usage Guide from Lattice)

LEDD_ADDR[3:0]	Name	Usage
1000	LEDDCR0	LED Driver Control Register 0
1001	LEDDBR	LED Driver Pre-scale Register
1010	LEDDONR	LED Driver ON Time Register
1011	LEDDOFR	LED Driver OFF Time Register
0101	LEDDBCRR	LED Driver Breathe On Control Register
0110	LEDDBCFR	LED Driver Breathe Off Control Register
0001	LEDDPWRR	LED Driver Pulse Width Register for RED
0010	LEDDPWRG	LED Driver Pulse Width Register for GREEN
0011	LEDDPWRB	LED Driver Pulse Width Register for BLUE
1111	reset	pulls RESET pin of PWM block high for 1 clock

Some RGB example settings:

```
LMX_CMD_PWM_LEDDCR0,0b10000111,  //   Enable; 125Hz; Mode=LFSR; prescaler high bits=11
LMX_CMD_PWM_LEDDBR, 0b11111111,  //   prescaler = 1023 (=48kHz at 48MHz sys clock)
LMX_CMD_PWM_LEDDONR,0b00010000,  //   Blink ON time = 0.5 sec (0=0 sec; FF=8.16 sec)
LMX_CMD_PWM_LEDDOFR,0b00010000,  //   Blink OFF time = 0.5 sec (0=0 sec; FF=8.16 sec)
LMX_CMD_PWM_LEDDBCRR,0b10100001,  //  Breath ON edge; 0.256 sec
          ('7=on/off; '6=edge 0=ON only, 1=ON+OFF; '5=should be 1; '3..0=time 0=0.128, F=2.048 sec)

LMX_CMD_PWM_LEDDBCFR,0b10100011, //   Breath OFF edge; 0.256 sec
          ('7=on/off; '6=PWM extend 0=ON, 1=ON+OFF; '5=should be 1; '3..0=time 0=0.128, F=2.048 sec)

LMX_CMD_PWM_LEDDPWRR,        0x02,
LMX_CMD_PWM_LEDDPWRG,        0x20,
LMX_CMD_PWM_LEDDPWRB,        0x20,
```