

Industrial-grade Smart Touch HMI Display Module

by ProtoDevs® GmbH

Datasheet

PDScreen Family description

"PDScreen" device family is an Industrial grade (-30...85 deg C) Smart Touch HMI TFT Displays which support different powering options and communication interfaces (like I2C, UART, CAN, USB, Ethernet) to operate as a HMI touch module giving the system a full power of graphical user interface (GUI). PDScreen modules family consist of 4" , 7" , 10.1" industrial-grade TFT matrixes + driver board with a firmware which gives a user a simple interface to interact with. All the complex hardware-type interfaces and commands between PDScreen MCU / RAM / Flash and peripherals are translated into the easy-to-implement interface commands which could be read and write to the several communication interfaces of the PDScreen modules. There are examples written in Python included which shows several examples, also the API included into this document to evaluate the principles and command list.

PDScreen module based on the newest STM32 microcontroller utilizing the power and cost-effectiveness of the latest industry standards.

There are several PDScreen variants available :

PDScreen_4inch_m30	: 4-inch capacitive touch TFT module
PDScreen_4inch_m30_nt	: 4-inch no-touch TFT module
PDScreen_7inch_m30	: 7-inch capacitive touch TFT
PDScreen_7inch_m30_nt	: 7-inch no-touch TFT module
PDScreen_7inch_m30_Ethernet	: 7-inch capacitive touch TFT with Ethernet + 48V PoE
PDScreen_10inch_m30	: 10.1-inch capacitive touch TFT
PDScreen_10inch_m30_nt	: 10.1-inch no-touch TFT module
PDScreen_10inch_m30_Ethernet	: 10.1-inch capacitive touch TFT with Ethernet + 48V PoE

ProtoDevs GmbH supports the helpdesk line in case of additional questions or PDScreen hardware / firmware / software customization requests - <https://www.protodevs.de/forum/>

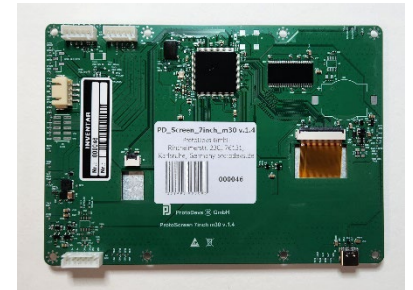
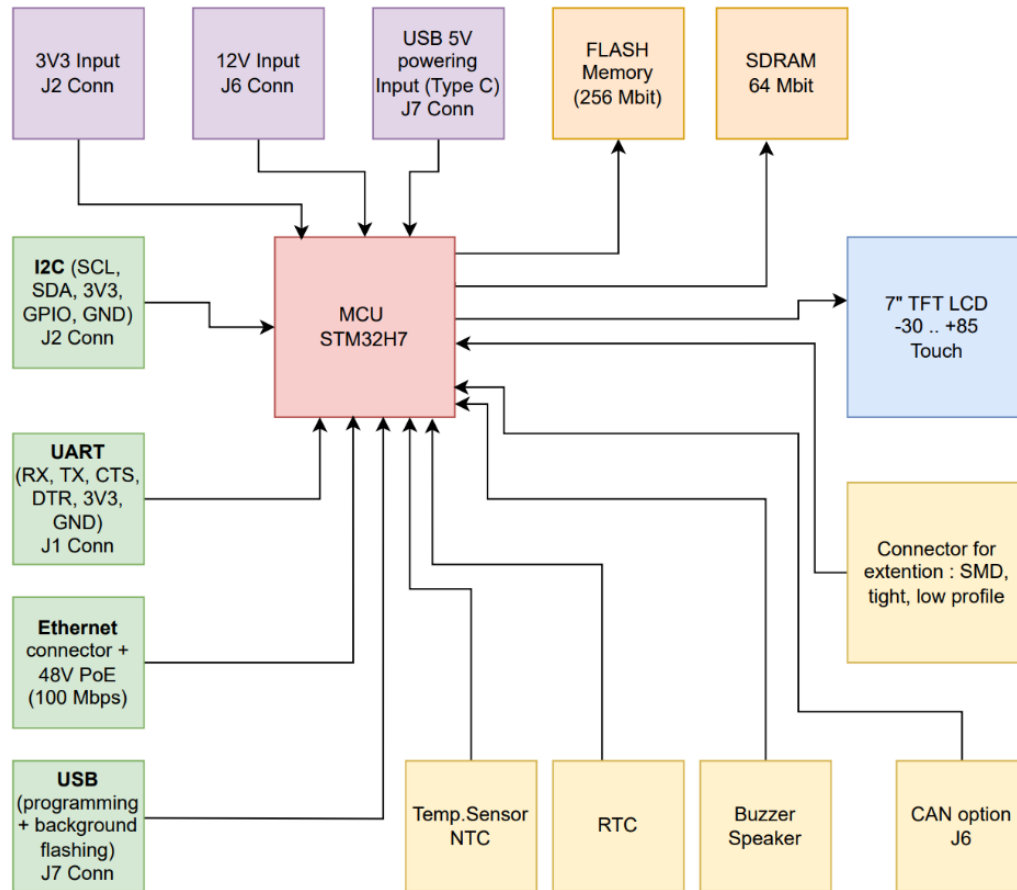


Table of Contents

PDScreen Family description	1
Simplified block diagram	3
Operational conditions	3
General specifications	4
Electrical specifications	5
LCD screen optical characteristics	6
Viewing angles definition	7
Graphical coordinates definition	7
Mechanical dimensions (TFT Screen part)	8
Mechanical dimensions (PDScreen module)	9
PDScreen module installation and operating requirements	10
PDScreen module mounting recommendations	11
Sun reflectance reduction tips while installation	12
Communication connectors placement	15
Communication interfaces pinouts	16
Preparations for PDScreen module programming	17
Content upload example	18
Running the script commands example	20
List of script commands supported (UART)	21
List of commands supported (I2C)	24

Simplified block diagram

(For PDScreen_7inch_m30)



Operational conditions

Operational temperature: **-30 ... 85 C**

Storage temperature: **-55 to 150 C**

Input voltages variants: **3.3V** on J2, **5V** on USB Type C, **12V** on J6, **48V PoE** for ETH version

ESD rating: Charged device model (CDM), per JEDEC specification JESD22-C101

Mass: 200 grams (Industrial touch TFT + PCB with components + Conformal coating)

UART speed: up to 1000000 (1 Mbps), firmware locked to 921000 (921 kbps) by default

I2C speed: up to 1 megabit per second (1 Mbps)

CAN speed: up to 1 megabit per second (1 Mbps)

Ethernet speed: up to Fast Ethernet (100 Mbps)

USB speed: up to USB 2.0 Full Speed (480 Mbps)

General specifications

PDScreen_7inch_m30 (v.1.4):

ITEM	STANDART VALUE	UNIT
LCD TYPE	TFT/TN/ NORMALLY WHITE/TRANSMISSIVE	
LCD MODULE SIZE	164.90*100.00*4.95	mm
ACTIVE AREA	154.08*85.92	mm
PIXEL PITCH (W*H)	0.0642*0.1790	
NUMBER OF PIXELS	800*480	pixels
RECOMMEND VIEWING DIRECTION	12	O'clock
GRAY SCALE INVERSION DIRECTION	6	O'clock
COLORS	16.7 M	
BACKLIGHT TYPE	27-DIES WHITE LED	
TOUCH PANEL TYPE (For touch panel version)	CTP	

Electrical specifications

The next tests were executed with the most power-efficient firmware configuration for STM32H7:
MCU frequency = 120 MHz, SRAM frequency = 60 MHz;

If the full spec is needed – please contact ProtoDevs GmbH by writing to info@protodevs.de.

PDScreen powering specified for 3,3V and 12V – both variants of powering are possible.

Test picture: white background + small blue rectangle in the center with backlight value ([LINK](#)).

Current consumption tests for **3.3V, 25 deg C** ambient:

ProtoScreen_7inch_1v2_m30 LuxMeter tests				
	Date	30 Aug 2024	Karlsruhe, ProtoDevs GmbH office	
	LuxMeter	Dr.Meter LX1010BS	Range 1 - 100 000 Lux	
	Multimeter	PeakTech 3442		
	3V3 Power supply	RIGOL DP932E	3.3V out on CH2 , 1A limit	
	ProtoScreen	7inch m30 v.1.2		
	Firmware	Protoscreen7m30-30.08.24-bkl.test.elf		
device 1				
Nn	BackLight value	LuxMeter Value	Current @3V3 , A	Power, Watt
1	0	0	0.15	0.495
2	1	52	0.18	0.594
3	2	169	0.25	0.825
4	3	285	0.32	1.056
5	4	400	0.38	1.254
6	5	514	0.46	1.518
7	6	625	0.53	1.749
8	7	734	0.62	2.046
9	8	839	0.7	2.31
10	9	942	0.8	2.64
11	10	1040	0.91	2.9029

Current consumption tests for **12V, 25 deg C** ambient:

ProtoScreen_7inch_1v4_m30 LuxMeter tests				
	Date	1 Oct 2024	Karlsruhe, ProtoDevs GmbH office	
	LuxMeter	Dr.Meter LX1010BS	Range 1 - 100 000 Lux	
	Multimeter	PeakTech 3442		
	12V Power supply	RIGOL DP932E	12V out on CH1 , 1A limit	
	ProtoScreen	7inch m30 v.1.4		
	Firmware	Protoscreen7m30-30.08.24-bkl.test.elf		
device 1				
Nn	BackLight value	LuxMeter Value	Current @12V , A	Power, Watt
1	0	0	0.05	0.6
2	1	36	0.06	0.72
3	2	147	0.08	0.96
4	3	258	0.1	1.2
5	4	369	0.12	1.44
6	5	481	0.14	1.68
7	6	590	0.16	1.92
8	7	698	0.19	2.28
9	8	803	0.21	2.52
10	9	906	0.24	2.88
11	10	1008	0.26	3.12

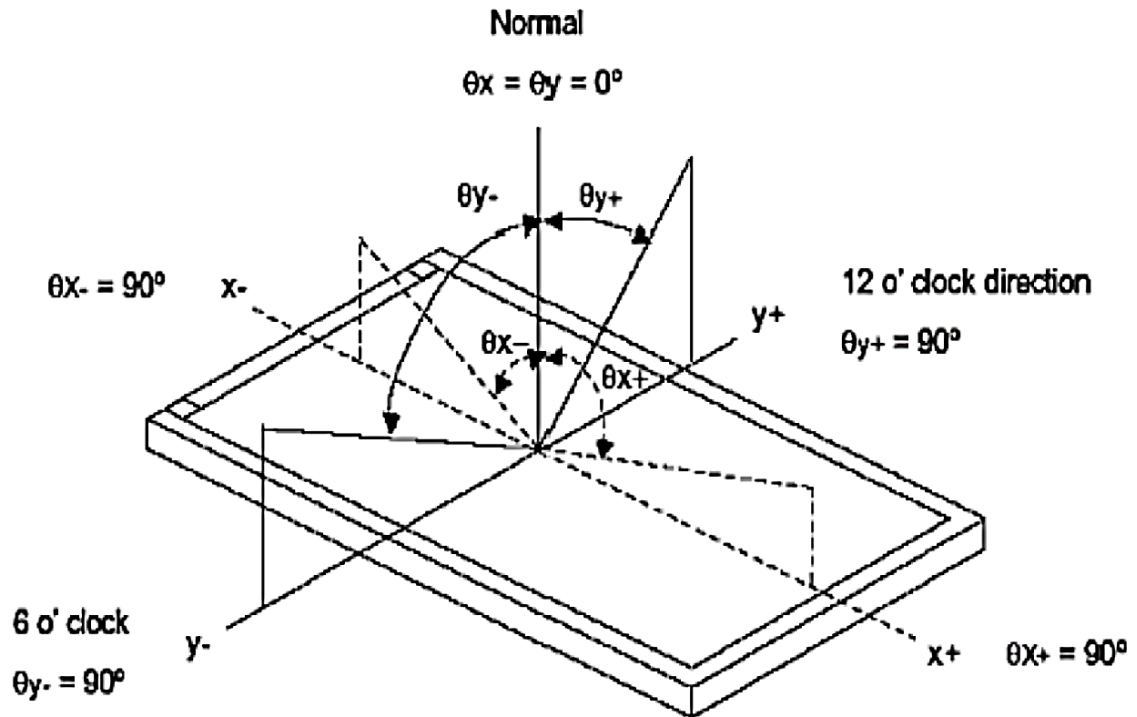
LCD screen optical characteristics

PDScreen_7inch_m30 (v.1.4):

ITEM		SYMBOL	CONDITIONS	SPECIFICATIONS			UNIT	NOTE
				MIN	TYP.	MAX		
Luminance		L		320	400	-	Cd/m ²	
Contrast ratio		CR	$\theta = 0^\circ$	400	500			
Response time	Rising	T _R	25°C		10	20	ms	
	Falling	T _F			15	30		
CIE COLOUR COORDINATE	RED	XR	VIEWING NORMAL ANGLE					
		YR						
	GREEN	XG						
		YG						
	BLUE	XB						
		YB						
	WHITE	XW		0.278	0.308	0.338		
		YW		0.297	0.327	0.357		
VIEWING ANGLE	Hor.	θ_{x+}	CR \geq 10	60	70		Degree	
		θ_{x-}		60	70			
	Ver.	θ_{y+}		40	50			
		θ_{y-}		60	70			

Viewing angles definition

PDScreen_7inch_m30 (v.1.4):



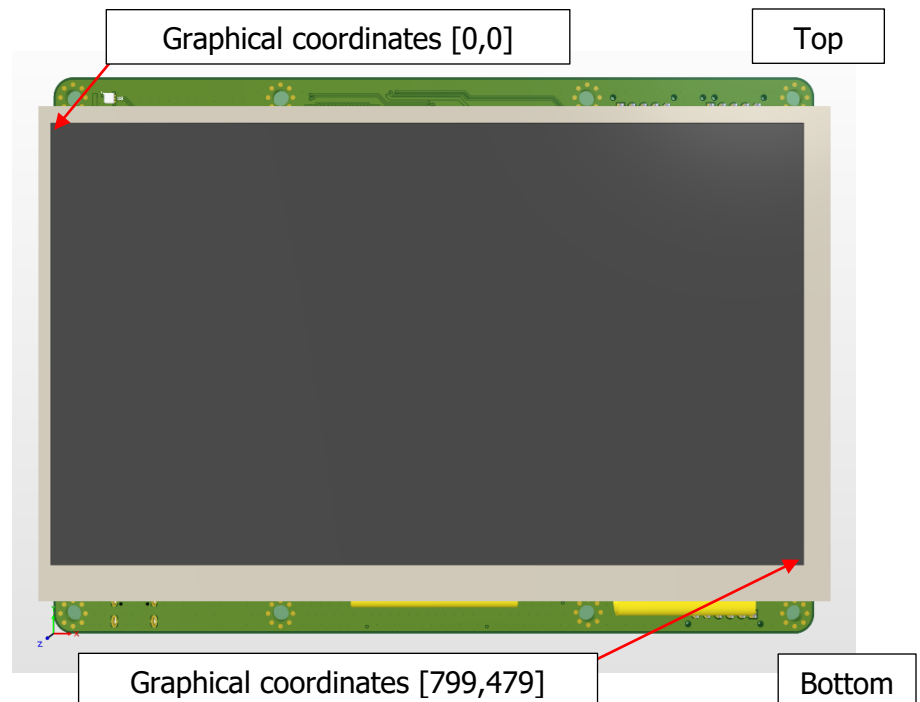
Graphical coordinates definition

The module orientation should be as mentioned on the picture from the right in order to use text commands in a readable way.

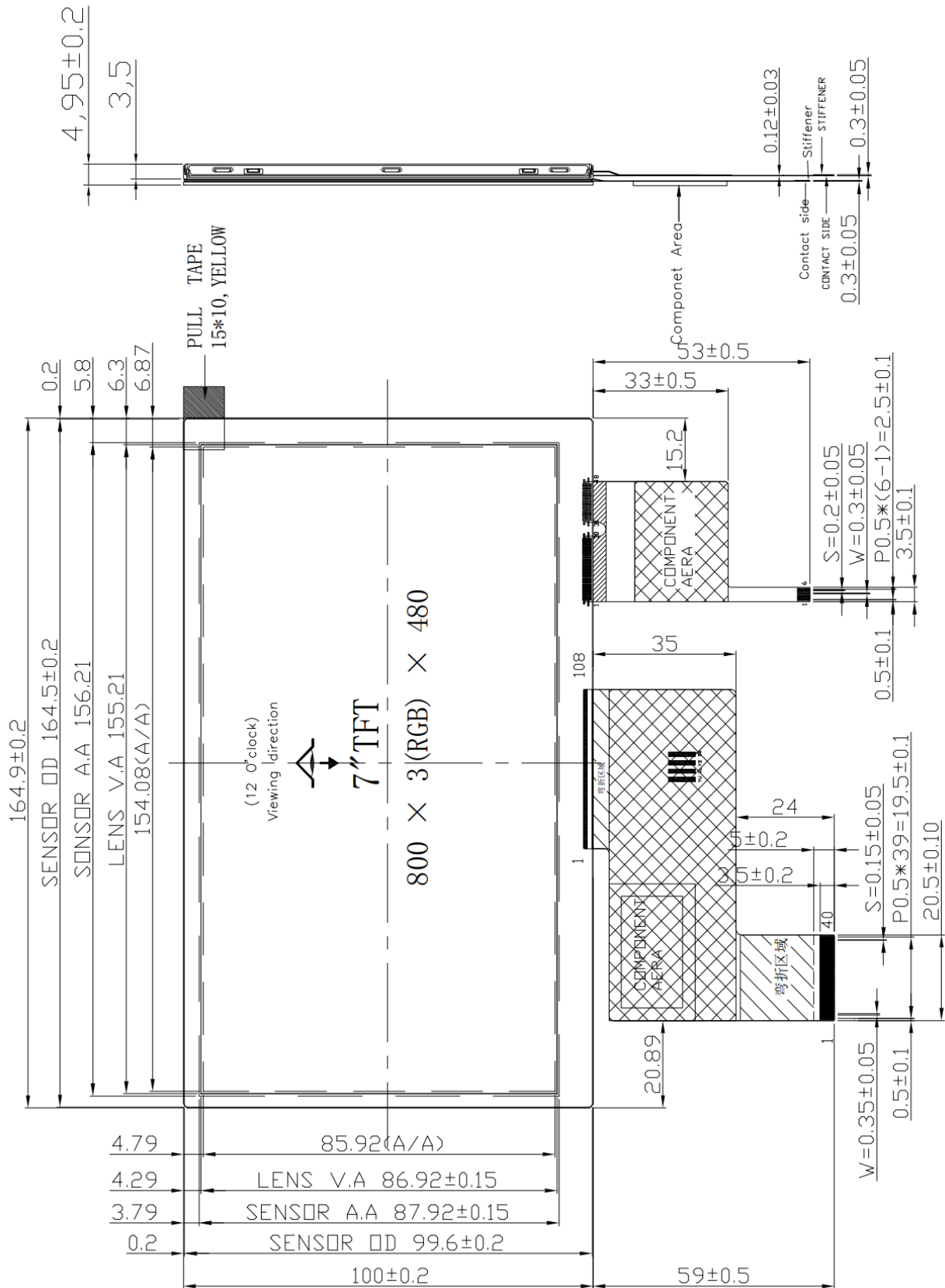
Top side has light sensor and 2x connectors, bottom side has 1x connector and flat cables from the TFT screen.

The screen rotation commands will be supported in the next versions of the API and Firmware, please visit our support forum for the latest updates (don't forget to register in order to get the full access):

<https://www.protodevs.de/forum/>



Mechanical dimensions (TFT Screen part)

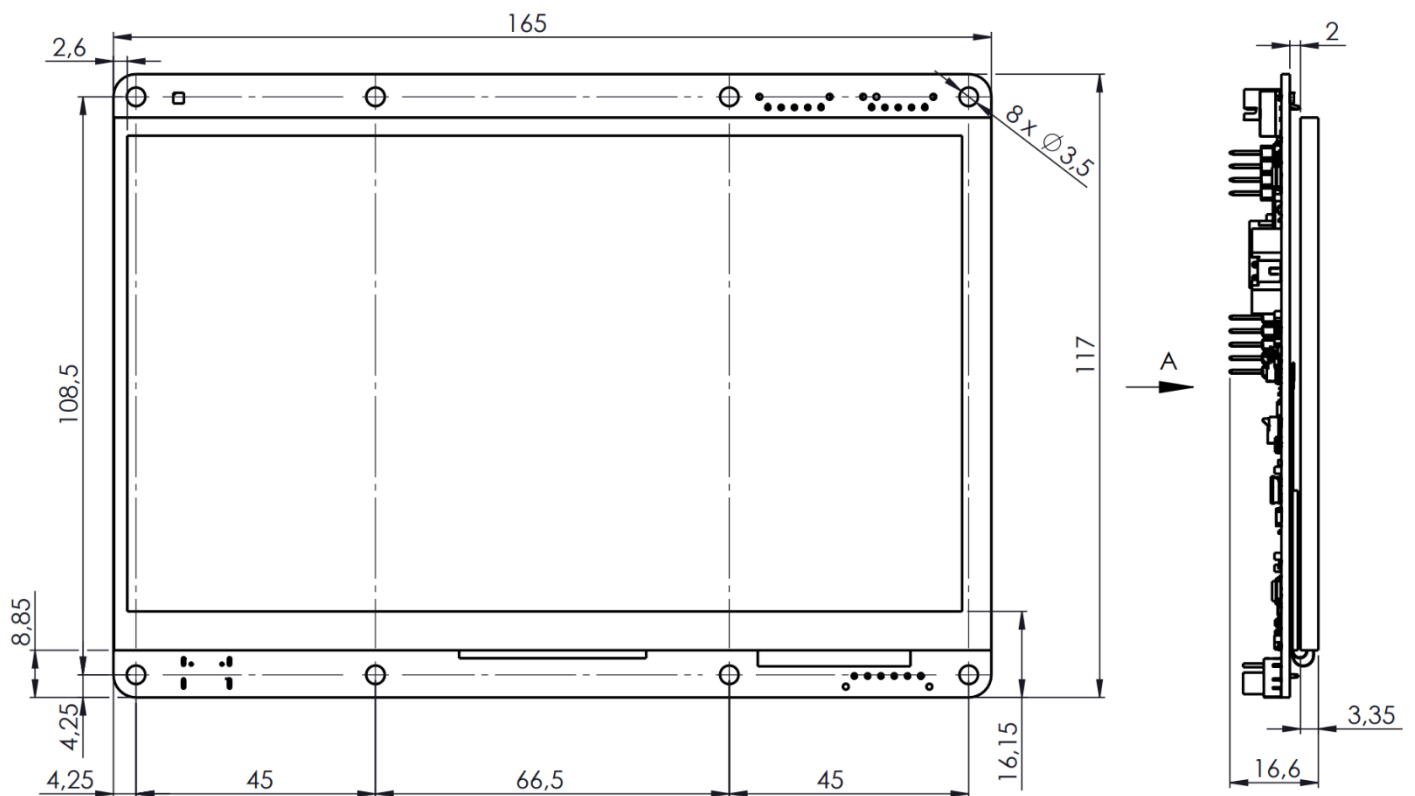
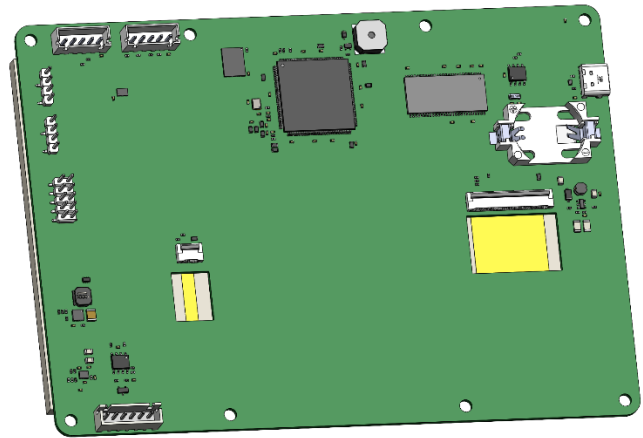
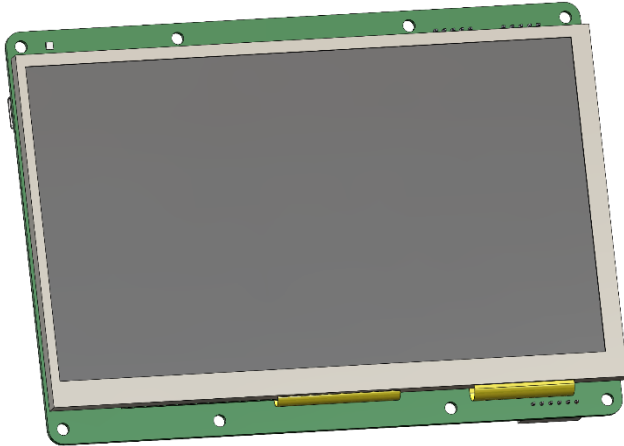


Mechanical dimensions (PDScreen module)

(For PDScreen_7inch_m30)

Dimensions (assembled module v.1.4):

117mm X 165mm X 19,2mm



PDScreen module installation and operating requirements

Precautions:

The display panel is made of glass. Do not subject it to a mechanical shock by dropping it from a high place, etc.

If the display panel is damaged and the liquid crystal substance inside it leaks out, do not get any in your mouth. If the substance come into contact with your skin or clothes promptly wash it off using soap and water.

When storing the PDScreen modules, avoid exposure to direct sunlight or to the light of fluorescent lamps. Keep the modules in bags designed to prevent static electricity charging under low temperature / normal humidity conditions (avoid high temperature / high humidity and low temperatures below 0). Whenever possible, the LCD modules °C should be stored in the same conditions in which they were shipped from our company.

Mounting and operating requirements:

- 1) Do not apply excessive force to the display surface or the adjoining areas since this may cause the color tone to vary.
- 2) The polarizer covering the display surface of the LCD module is soft and easily scratched. Handle this polarize carefully.
- 3) To prevent destruction of the elements by static electricity, be careful to maintain an optimum work environment.
- 4) Be sure to ground the body when handling the PDScreen module and while installing the PDScreen module.
- 5) Tools required for assembly and installation, such as cables, gloves, packaging, must be properly grounded.
- 6) To reduce the amount of static electricity generated, do not conduct assembly and other work under dry conditions.
- 7) The LCD module is coated with a film to protect the display surface. Exercise care when peeling off this protective film since static electricity may be generated.
- 8) There is a need to change the picture on the screen. If the LCD modules have been operating for a long time showing the same display patterns may remain on the screen as ghost images and a slight contrast irregularity may also appear. Abnormal operating status can be resumed to be normal condition by suspending use for some time. It should be noted that this phenomenon does not adversely affect performance reliability.
- 9) To minimize the performance degradation of the PDScreen modules resulting from caused by static electricity, etc. exercise care to avoid holding the following sections when handling the modules: - Exposed area of the printed circuit board, - Terminal electrode sections, - communication interface connectors, -programming and expansion connectors, -electronic components pins and pads, - sensors and battery holders.

Others:

Liquid crystals solidify at low temperature (below the storage temperature range which is defined as -40 deg C) leading to defective orientation of liquid crystal or the generation of air bubbles (black or white). Air bubbles may also be generated if the module is subjected to a strong shock at a low temperature.

PDScreen module mounting recommendations

Outdoor installation of the PDScreen modules require certain type of protection and climatic considerations to follow.

The module should be mounted into enclosure which is a solid metal, outdoor plastic or similar material which gives fully weatherproof barrier but also through climatic controls ensures the interior of the enclosure is the optimum environment for running a screen – no matter what the ambient conditions or temperatures are like.

Always avoid direct sunlight. Direct sunlight can cause the liquid crystal to boil and result in black blotching on the display, a phenomenon called Solar Clearing. The LCD needs to remain below this boiling point, which requires some mechanism for cooling the LCD. This mechanism must be implemented, in other case the PDScreen modules will not work properly and the warranty will not be longer active.

1. Install lightning protection devices

In outdoor environments, LED display screens are susceptible to lightning. Therefore, lightning protection devices must be installed on the display screen and the building where it is located. The main body and casing of the display screen should be well grounded, and the grounding resistance should be less than **3 Ohms** to ensure that the large current caused by lightning can be discharged in time. Effective lightning protection measures can prevent strong electric and magnetic attacks on the display screen caused by lightning, thereby protecting the safety of the equipment.

2. Waterproof measures

Outdoor LED display modules must have excellent waterproof performance. The display screen body and its joints with the building must be strictly waterproof and leakproof to ensure that water can be discharged smoothly in case of accumulation. Since the outdoor environment is often exposed to the sun, rain, wind and dust, the display screen may face serious challenges in waterproofing and moisture resistance. Once electronic equipment is wet or damp, it may cause a short circuit or even a fire, causing serious losses. Therefore, waterproof design is the key to ensure the long-term stable operation of the display screen

3. Ventilation and cooling

In order to prevent the display screen module from malfunctioning due to overheating, ventilation equipment must be installed for cooling. Usually, an axial fan is installed above the back of the screen or rack where the module is mounted to discharge heat and ensure that the internal temperature of the screen is maintained between -30°C and 60°C. Big LED display screens generate a lot of heat when working. If the heat dissipation is poor, the integrated circuit may be abnormal or even burn out, causing the display system to fail to work properly.

The installation of the outdoor PDScreen modules must consider the requirements of lightning protection, waterproofing, moisture-proofing, heat dissipation, and display effects. Only by making full preparations for these details can ensure that the display module can operate stably and efficiently in various harsh environments and provide users with high-quality display services.

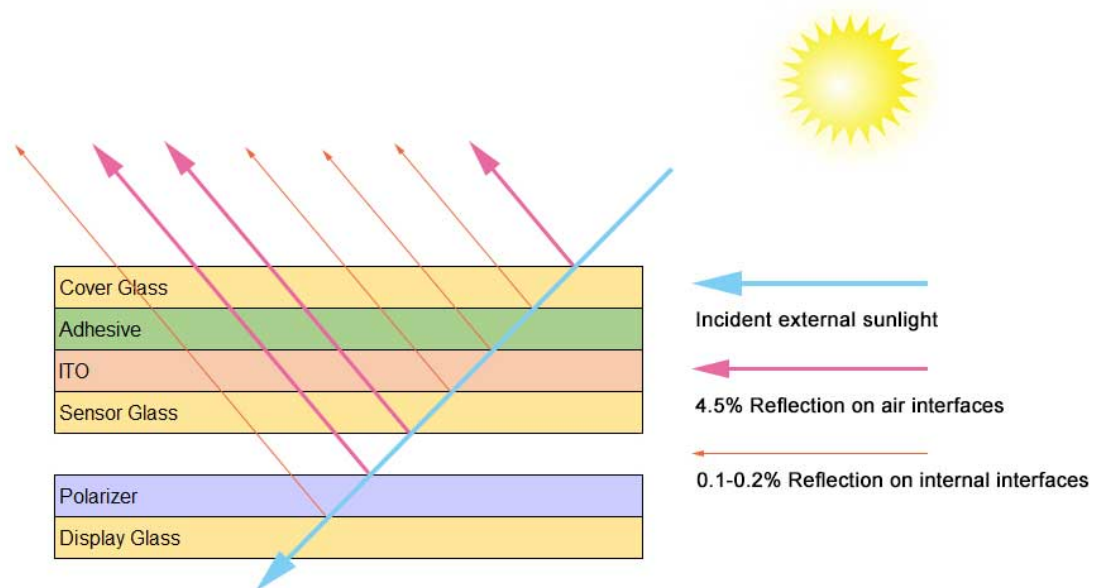
Sun reflectance reduction tips while installation

When light traveling in one transparent medium encounters a boundary with another transparent medium, a portion of the light bounces off the border. Through the simplest version of Fresnel's equation, we can calculate the amount of reflected light.

$$R = \left[\frac{n_2 - n_1}{n_2 + n_1} \right]^2 \quad (n_1 \text{ \& } n_2 \text{ are indexes of refraction for 1}^{\text{st}} \text{ and 2}^{\text{nd}} \text{ material})$$

It is clear from the equation that the more difference between two materials, the more light will be reflected.

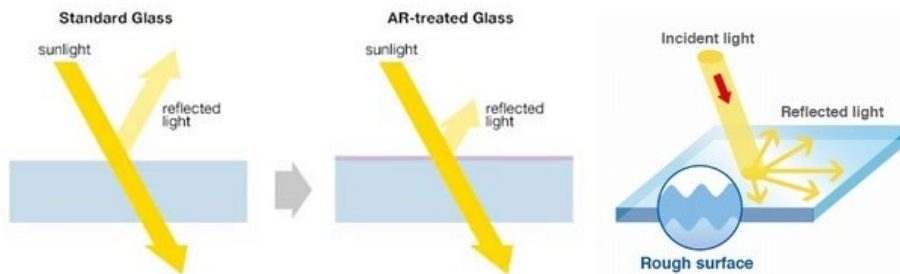
The regular TFT LCD module with touch panel has the multilayer structure. There are several places light reflection happens.



The total reflectance on a TFT LCD with touch panel is the sum of reflected light on any interface where two materials meet. As an example, between polarizer and display glass, the difference in index of refractions for the two materials is very small, around 0.1. So the reflected light on this interface is only 0.1%. As Fresnel's equation points out, we should focus reflection reduction on air interfaces. For air, its index of refraction is 1; for glass, it is 1.5. And that results in a reflectance of 4.5%. Therefore, the three air interfaces contribute majority of TFT LCD's reflectance, at about 13%.

Reduce Top Surface Reflection

The quick and easiest thing we can do to reduce air-glass interface reflectance is to use an Anti-Reflection and Anti-Glare film or apply AR coating. An external film with AR properties not only reduces reflected light, but also brings other benefits.



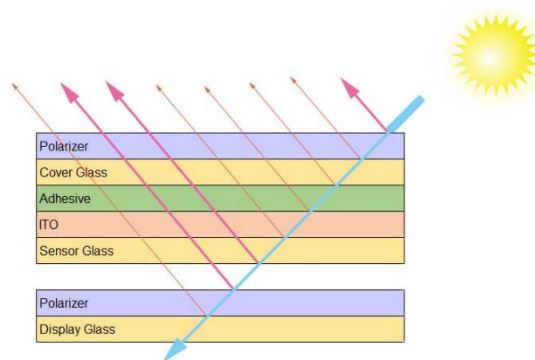
An LCD screen with external film solves this issue nicely. As for automotive applications, in an accident, broken LCD with top AR film won't produce sharp edge glass that could harm an auto occupant. Nevertheless, a top film always reduces TFT LCD's surface hardness. And it is susceptible to scratches. On the other hand, AR coating retains LCD's hardness and touch performance. But it comes with a bigger price tag.

PDScreen modules screens can be covered with any existing additional layers in case of design customization.

With above measurements, TFT LCD's top layer reflectance can be reduced by 2~3%.

Reduce Ambient Light Getting Into LCD

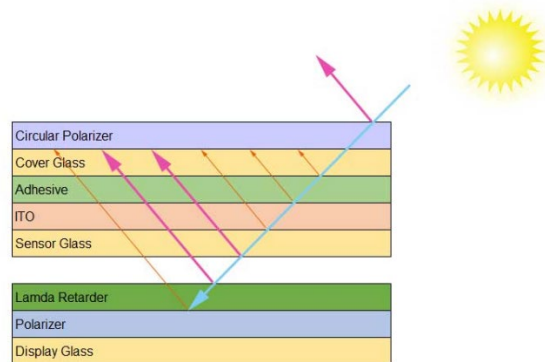
Another quick and easy way to tackle reflectance is to affix a linear polarizer on the top of TFT screen. When ambient light gets to the top polarizer, only half of the light passes through. Which results in reflection light cutting to half. This is a very low cost way to increase TFT LCD's contrast, such that making it more sunlight readable.



Putting any film on the top will reduce LCD's surface hardness, and polarizer will cause a loss in transmission. Those are not very ideal.

Block Reflected Light

Laminating a circular polarizer in TFT LCD will get rid of a lot of reflectance. That is because when ambient light passes through circular polarizer it gets circularly polarized. And when it is reflected, the polarization direction flips by 180 degrees. So, when reflected light comes back to the circular polarizer, nothing goes through to viewer's eyes.



This method is very effective for an LCD display with resistive touch panel. We know resistive touch LCD has two air gaps: air gap between two ITO (Indium tin oxide) layers and air gap between touch panel and LCD display. Reflectance caused by the two air gaps is very high. Applying circular polarizer blocks off most of the reflected light, and makes the LCD display sunlight readable.

The disadvantage of such solution is its cost. Since we need not only a circular polarizer, but also a retarder film on the top of LCD display, making sure light originates from within LCD is not blocked by external circular polarizer.

Lessen Reflectance on Air Gap

Air gap reflection is the main culprit causing bad sunlight readability. We can improve this from two directions.

Add AR films on both interfaces of internal air gap. The add-ons can reduce this area's reflection from 8.5% to 2%. And since the AR films are not outside facing, they are much cheaper than the one used outside. Keeping the air gap also retains the ease of service, in case either touch panel or LCD display needs to be repaired.

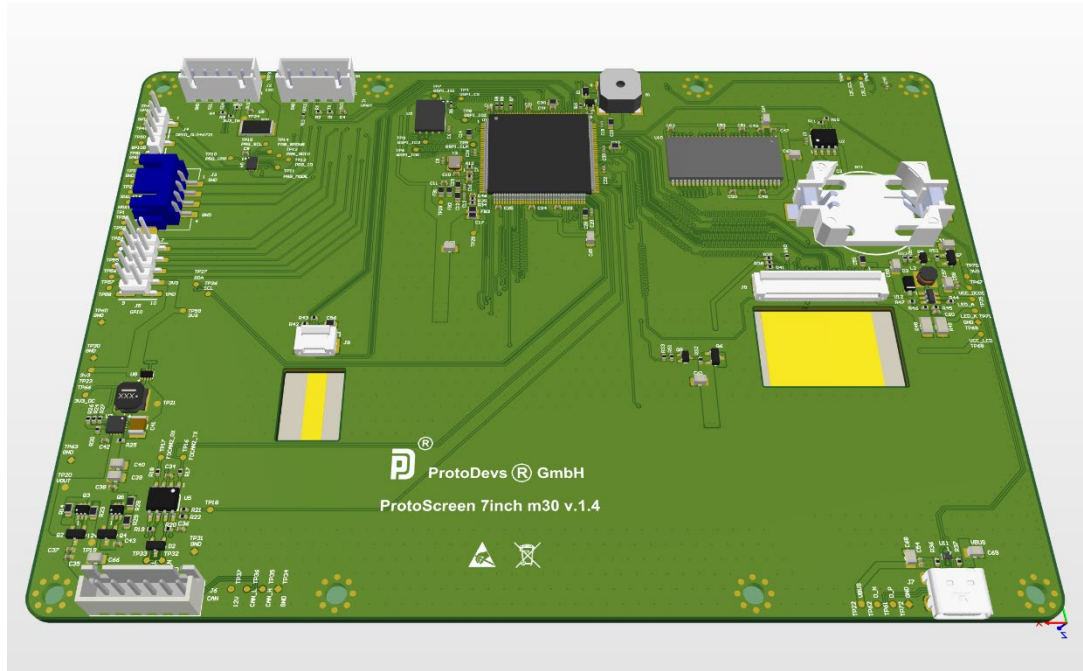
The most effective way is to eliminate air gap totally, by using optical bonding. In plain language, we fill air gap with special optical adhesive, to smooth out the area's refraction index differences. Such that reflectance caused by internal air gap drops from 8.5% to 0.5%. Optical bonding is expensive but effective way to improve TFT LCD sunlight readability. It enhances durability and resistance to impact. Moreover, no air gap means no moisture condensation and fogging.

Summary: ProtoDevs GmbH can customize the PDScreen modules by customer request in case of special defined sunlight protection requirements.

Communication connectors placement

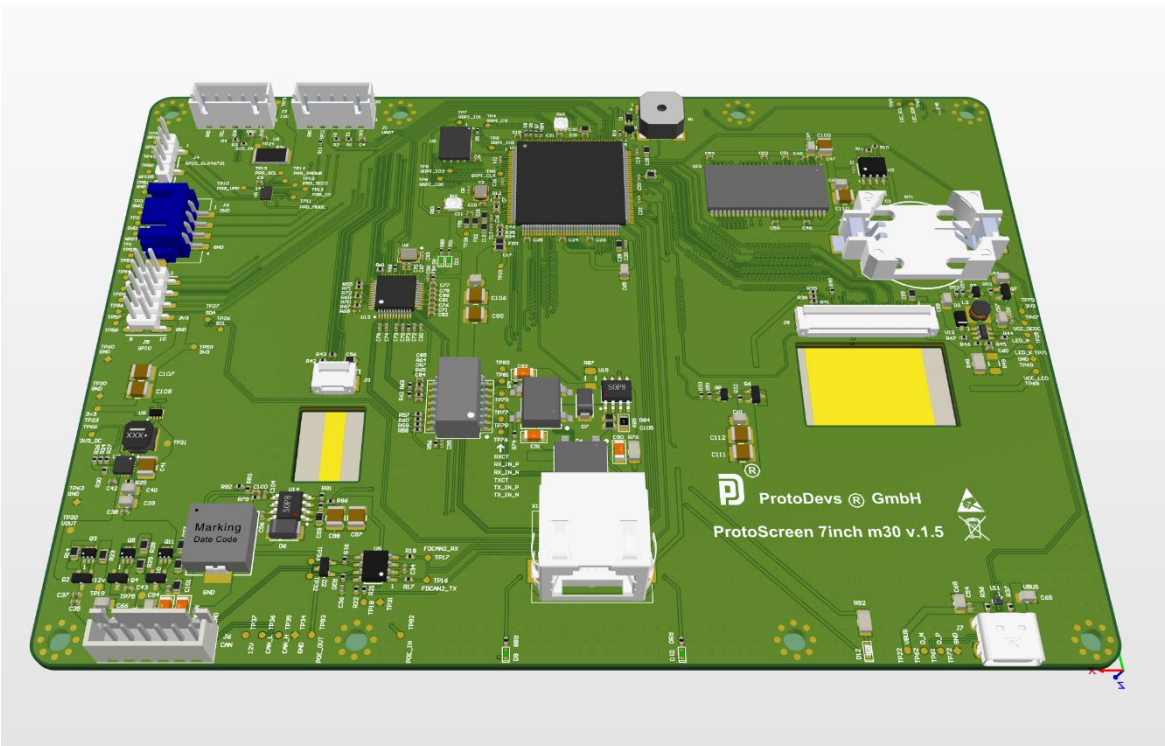
PDScreen_7inch_m30 (v.1.4):

I2C (on J2), **UART** (on J1), **CAN** (on J6), **USB** (on J7)



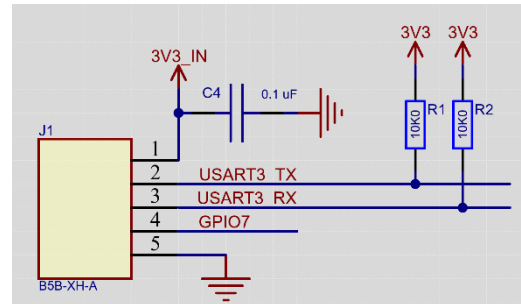
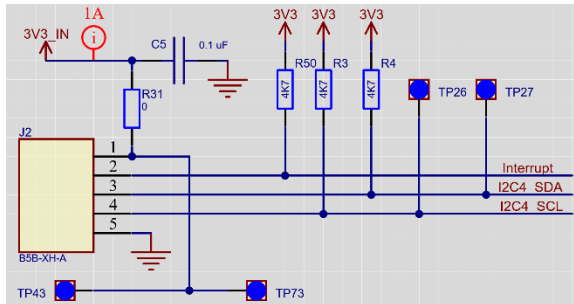
PDScreen_7inch_m30_ETH (v.1.5):

I2C (on J2), **UART** (on J1), **CAN** (on J6), **USB** (on J7), **Ethernet** with **PoE** (on X1)

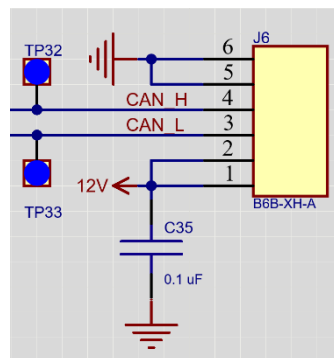
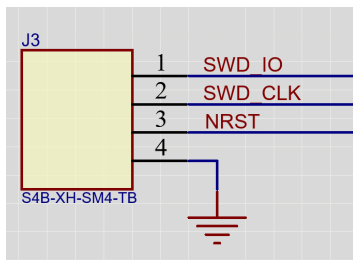


(For PDScreen_7inch_m30)

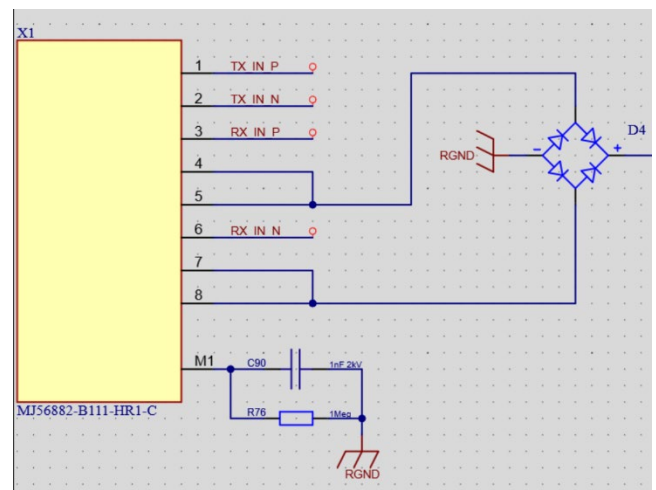
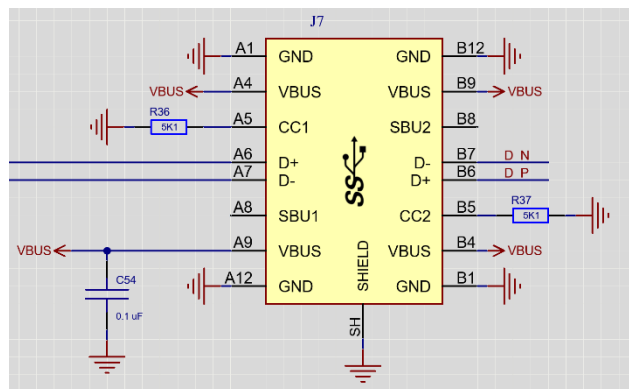
UART (J1):



CAN /12V (J3)



Ethernet / 48V (X1)



Preparations for PDScreen module programming

1. Installation

ProtoScreen.py is the Python script to show the main PDScreen board functionality. It requires Python 3.3 or upper. PDScreen commands were tested with **Python 3.9.6**. It also requires **pyserial** library.

Following commands may help to install it:

MacOS:

```
sudo python3 -m ensurepip  
sudo pip3 install pyserial
```

Windows:

```
pip install pyserial
```

Linux:

```
sudo pip install pyserial
```

2. Configuration

Script is configured via the **ProtoScreen.ini** file. File contains the following sections:

[upload] section:

Contains path to images or fonts to upload to the board. Path may be relative to the script directory or absolute.

Supported formats are:

- BMP images with 24 bits per pixel color
- JPG images, colored with 24 bits per pixel color
- PNG images, with 24 bits per pixel color or 32 bits per pixel color (with A channel)
- fonts in a special FNT format

[download] section:

dir parameter contains name of directory where to store images downloaded from the board

3. Run

Run script with 'upload' parameter to upload pictures to the board:

```
sudo python3 ProtoScreen.py upload
```

Run script with 'download' parameter to download pictures from the board:

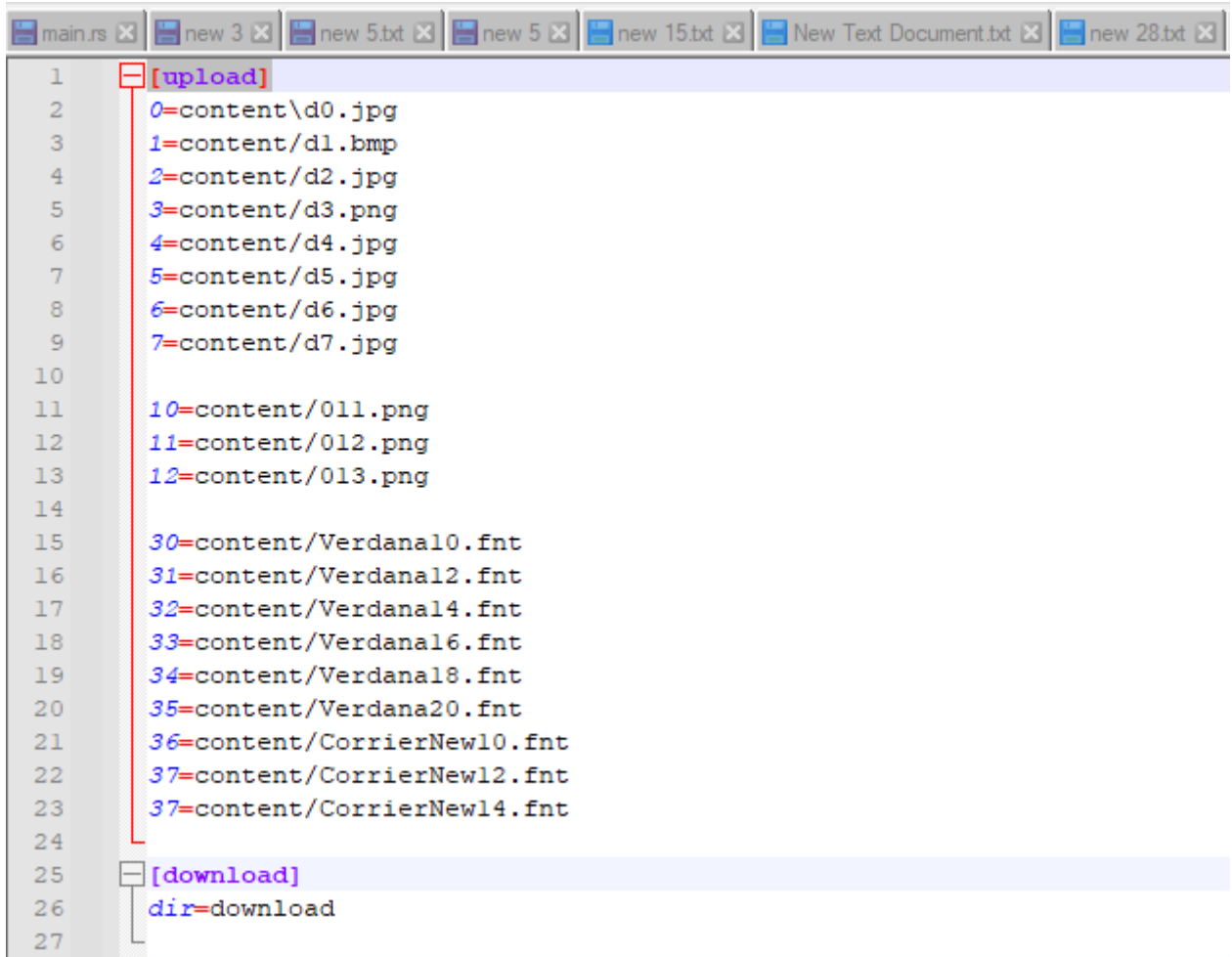
```
sudo python3 ProtoScreen.py download
```

Run script with 'exec' parameter to execute commands:

```
sudo python3 ProtoScreen.py exec
```

Content upload example

Content to be uploaded is configured in the **[upload]** section of the **LogView.ini** file.



```
1 [upload]
2 0=content\d0.jpg
3 1=content/d1.bmp
4 2=content/d2.jpg
5 3=content/d3.png
6 4=content/d4.jpg
7 5=content/d5.jpg
8 6=content/d6.jpg
9 7=content/d7.jpg
10
11 10=content/011.png
12 11=content/012.png
13 12=content/013.png
14
15 30=content/Verdana10.fnt
16 31=content/Verdana12.fnt
17 32=content/Verdana14.fnt
18 33=content/Verdana16.fnt
19 34=content/Verdana18.fnt
20 35=content/Verdana20.fnt
21 36=content/CorrierNew10.fnt
22 37=content/CorrierNew12.fnt
23 37=content/CorrierNew14.fnt
24
25 [download]
26 dir=download
27
```

Numbers should start from 0 and be consecutive. Actually, it is not mandatory for numbers to start from 0 and gaps in numbering are allowed, but be aware that board reserves memory for numbers from 0 to maximum number and memory overflow is possible if maximum number is too big.

Later, numbers are used as indexes in board commands.

Content is uploaded with the ProtoScreen.py script with the following command:

ProtoScreen.py upload

Scripts finds the correct COM port automatically, or you can state the port explicitly

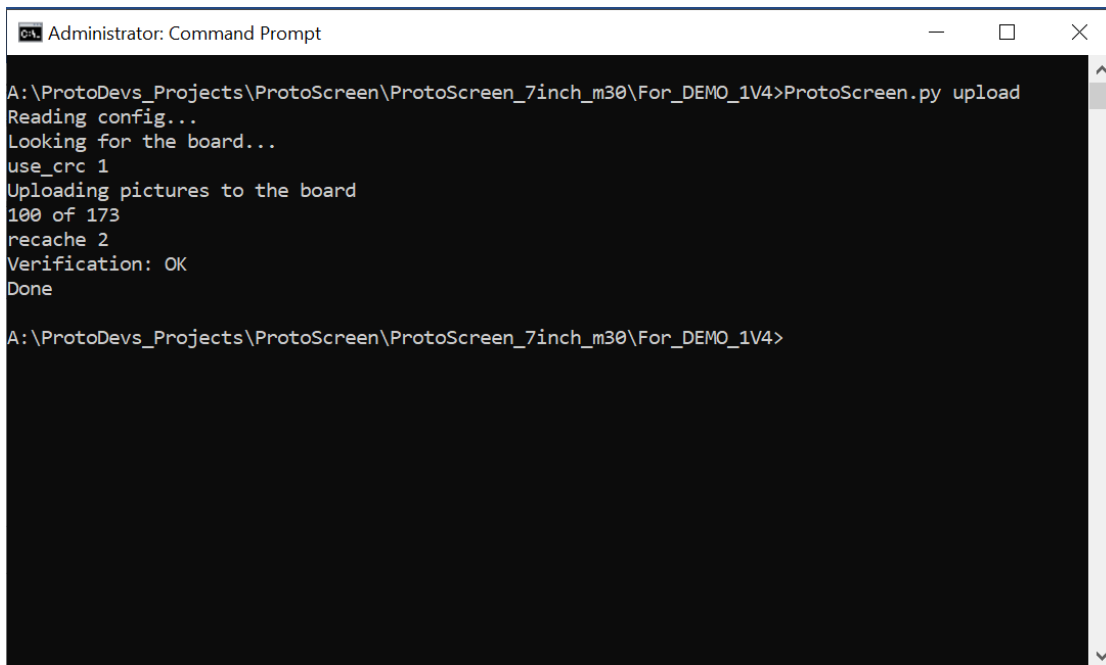
ProtoScreen.py upload COM18

Partial content update is not possible. If you need to add something to what is already on the board, you need to **download** content from the board first with

ProtoScreen.py download or **ProtoScreen.py download COM18**

and upload the correct content back.

Example of the **Upload** command execution:

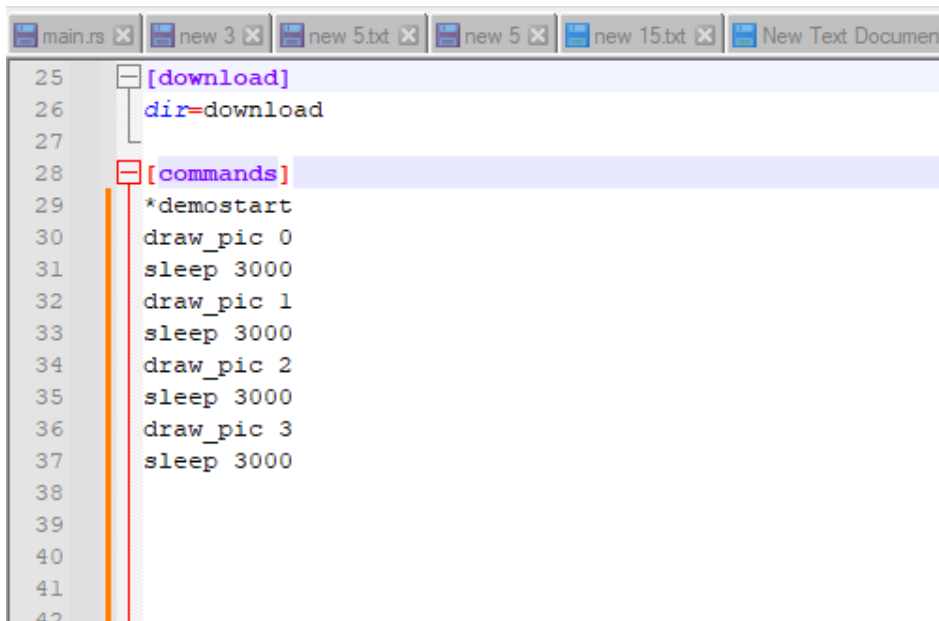


```
Administrator: Command Prompt
A:\ProtoDevs_Projects\ProtoScreen\ProtoScreen_7inch_m30\For_DEMO_1V4>ProtoScreen.py upload
Reading config...
Looking for the board...
use_crc 1
Uploading pictures to the board
100 of 173
recache 2
Verification: OK
Done
A:\ProtoDevs_Projects\ProtoScreen\ProtoScreen_7inch_m30\For_DEMO_1V4>
```

Running the script commands example

Script can be used to issue commands to the board. You can use it to check the uploaded images, for example. Commands must be listed in **[commands]** section of **ProtoScreen.ini** file.

This example runs **demostart** macros first (described in **[demostart]** section) to turn on the screen and backlight, and then draws pictures with indexes from 0 to 3 with 3 second delay.



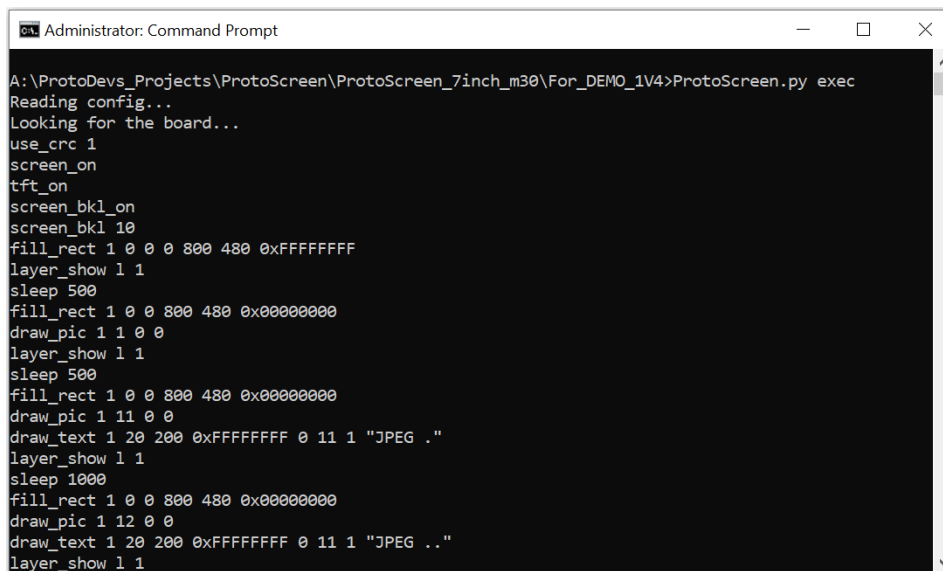
```

25 [download]
26   dir=download
27
28 [commands]
29   *demostart
30   draw_pic 0
31   sleep 3000
32   draw_pic 1
33   sleep 3000
34   draw_pic 2
35   sleep 3000
36   draw_pic 3
37   sleep 3000
38
39
40
41
42

```

To run commands there is a need to execute the next command:

ProtoScreen exec or ProtoScreen exec COM18



```

Administrator: Command Prompt
A:\ProtoDevs_Projects\ProtoScreen\ProtoScreen_7inch_m30\For_DEMO_1V4>ProtoScreen.py exec
Reading config...
Looking for the board...
use_crc 1
screen_on
tft_on
screen_bkl_on
screen_bkl 10
fill_rect 1 0 0 800 480 0xFFFFFFFF
layer_show 1 1
sleep 500
fill_rect 1 0 0 800 480 0x00000000
draw_pic 1 1 0 0
layer_show 1 1
sleep 500
fill_rect 1 0 0 800 480 0x00000000
draw_pic 1 11 0 0
draw_text 1 20 200 0xFFFFFFFF 0 11 1 "JPEG ."
layer_show 1 1
sleep 1000
fill_rect 1 0 0 800 480 0x00000000
draw_pic 1 12 0 0
draw_text 1 20 200 0xFFFFFFFF 0 11 1 "JPEG .."
layer_show 1 1

```

List of script commands supported (UART)

[**commands**] section:

Contains commands executed by exec run option. Allowed commands:

***name** - executes commands from section [name]

sleep xxxx - sleeps for xxxx milliseconds.

draw_pic l idx x y - draws image from board memory (draw_pic api command). l - layer index, idx - image index, same as in upload section, x, y - coordinates of top-left image corner

draw_pic2 l idx x y x_img y_img w h A a - draws part of image from board memory (draw_pic2 api command). l - layer index, idx - image index, same as in upload section, x, y - coordinates of top-left image corner on the screen, x_img y_img - coordinates of top-left point on the image, w h - width and height to draw, A - alpha value, a - alpha mode (0 - no modify alpha, 1 - replace alpha, 2 - combine alpha. for images without alpha 'replace alpha' mode always used)

draw_rect l x y w h clr - draws rectangle (draw_rect api command). l - layer index, x, y - coordinates of top-left rectangle corner, w - rect width, h - rect height, clr - rect color in ARGB8888 format

fill_rect l x y w h clr - draws filled rectangle (fill_rect api command). l - layer index, x, y - coordinates of top-left rectangle corner, w - rect width, h - rect height, clr - rect color in ARGB8888 format

draw_rrect l x y w h r clr - draws rounded rectangle (draw_rrect api command). l - layer index, x, y - coordinates of top-left rectangle corner, w - rect width, h - rect height, r - rounding radius, clr - rect color in ARGB8888 format

fill_rrect l x y w h r clr - draws filled rounded rectangle (fill_rrect api command). l - layer index, x, y - coordinates of top-left rectangle corner, w - rect width, h - rect height, r - rounding radius, clr - rect color in ARGB8888 format

draw_circ l x y r clr - draws circle (draw_circ api command). l - layer index, x, y - coordinates of circle center, r - radius, clr - rect color in ARGB8888 format

fill_circ l x y r clr - draws filled circle (fill_circ api command). l - layer index, x, y - coordinates of circle center, r - radius, clr - rect color in ARGB8888 format

draw_tri l x1 y1 x2 y2 x3 y3 clr - draws triangle (draw_tri api command). l - layer index, x1, y1, x2, y2, x3, y3 - coordinates of triangle corners, clr - rect color in ARGB8888 format

fill_tri l x1 y1 x2 y2 x3 y3 clr - draws filled triangle (fill_tri api command). l - layer index, x1, y1, x2, y2, x3, y3 - coordinates of triangle corners, clr - rect color in ARGB8888 format

draw_line l x1 y1 x2 y2 clr - draws line (draw_line api command). l - layer index, x1, y1, x2, y2 - coordinates of line corners, clr - rect color in ARGB8888 format

draw_pixel l x y clr - draws pixel (draw_pixel api command). l - layer index, x, y - coordinates of pixel, clr - rect color in ARGB8888 format

draw_text l x y clr clr_bg font sz "text" - draws text (draw_text api command). l - layer index, x, y - coordinates of start point, clr - text color in ARGB8888 format, clr_bg - background color in ARGB8888 format, font - font index, same as in upload section, sz - font size, "text" - text itself

draw_text_rect l font x y w h clr clr_bg sz wr ch cv av ah "text" - draws text rectangle with alignment options (draw_trct api command). l - layer index, font - font index, same as in upload section, x, y - coordinates of start point, w, h - text rect size, clr - text color in ARGB8888 format, clr_bg - background color in ARGB8888 format, sz - font size, wr - 1 if wrap is allowed, ch - 1 if horizontal clipping is allowed, cv - 1 if vertical clipping is allowed, av - vertical alignment (0 - top, 1 - bottom, 2 - center), ah - horizontal alignment (0 - left, 1 - right, 2 - center), "text" - text itself

draw_text_rect2 l font x y w h idx x_img y_img A a clr sz wr ch cv av ah "text" - draws text rectangle with alignment options and background picture (draw_trct2 api command). l - layer index, font - font index, same as in upload section, x, y - coordinates of start point, w, h - text rect size, idx - image index, same as in upload section, x_img y_img - coordinates of top-left point on the image, A - alpha value, a - alpha mode (0 - no modify alpha, 1 - replace alpha, 2 - combine alpha. for images without alpha 'replace alpha' mode always used), clr - text color in ARGB8888 format, sz - font size, wr - 1 if wrap is allowed, ch - 1 if horizontal clipping is allowed, cv - 1 if vertical clipping is allowed, av - vertical alignment (0 - top, 1 - bottom, 2 - center), ah - horizontal alignment (0 - left, 1 - right, 2 - center), "text" - text itself

clc_txt_rect font x y l r sz wr "text" - calculates size of rect covered by text (clc_txt_rect api command). font - font index, same as in upload section, x, y - coordinates of start point, l - left limit of text rect, r - right limit of text rect, sz - font size, wr - 1 if wrap is allowed, "text" - text itself

copy_layer lf lt xt yt xf yf w h A a - copies area from one layer to another layer (layer_copy api command). lf - source layer index, lt - destination layer index, xt, yt - coordinates of left-top point on destination layer, xf, yf - coordinates of left-top point on source layer, w, h - area size, A - alpha value, a - alpha mode (0 - no modify alpha, 1 - replace alpha, 2 - combine alpha)

layer_show l o - shows/hides layer - (layer_show api command). l - layer index, o - option (0 - hide layer, 1 - show layer)

bg_color r g b - sets background color (bg_color api command). r, g, b - background color in rgb format

screen_bkl x - sets screen backlight (screen_bkl api command). x - backlight value from 0 to 10

screen_bkl_on - turns backlight on (screen_bkl_on api command).

screen_bkl_off - turns backlight off (screen_bkl_off api command).

screen_on - turns screen on (screen_on api command) (tft + fill tft with black + turn backlight on).

screen_off - turns screen off (screen_off api command) (tft off and backlight off).

tft_on - turns tft on (tft_on api command).

tft_off - turns tft off (tft_off api command).

gettemp - prints board temperature to console (get_temp api command).

getlight - prints board light sensor measurements to console (get_light api command).

gettime - prints current board timestamp to console (get_time api command).

settime D M Y h m s - sets current board timestamp (set_time api command). D - day, M - month, Y - year (only 2 last digits matter, it is assumed that 2 upper digits are always 20), h - hour, m - minute, s - second

getram addr cnt - gets rtc ram content (get_rtcram api command). addr - ram address (0 to 63), cnt - number of bytes to read

setram addr cnt bts - sets rtc ram content (set_rtcram api command). addr - ram address (0 to 63), cnt - number of bytes to write, bts - bytes to write

gettouch - calls macros with waiting for the touch screen and drawing white dot at the touch point

touch_capton x y w h b - starts touch screen input capture in the specified rect. x,y,w,h - touch capture rect coordinates, b=1 - draw control buttons

touch_captoff wait - stops touch screen input capture and calls macros to download the captured input if capture is ended. wait=0 - exit immediately, wait=1 - wait for user to stop the capture with the control button

get_screenshot o - gets layer screenshot (get_screenshot api command). o - layer selection (0 - background layer, 1 - foreground layer)

note_bpm B - sets beats per minute speed (note_bpm api command). B - bpm value

note_status - prints to console information is sound is playing at the moment (note_status api command).

note_play NolNolNol... - plays notes (note_play api command). Nol - string with notes in format: N = Note (C, c, D, d, E, F, f, G, g, A, a, B), o = note octave, l = note length, ex: G52F52E52d52E52F52E52d52C52B42C52D52C52B42A42g42A42B42A49P02

List of commands supported (I2C)

ProtoScreen i2c address **0x39** by default, speed up to **1000000**

1. General information

Each command consists of command name + semicolon + parameters. In multibyte values smallest byte comes first.

Receive/transmidt buffer size is 4104 bytes, total request/response size cannot be more than that.

2. Colors and layers

There are 2 layers and 1 buffer available for drawing.

Both layers are visible by default. Foreground layer is always drawn above the background layer and supports transparency.

You can disable any layer, or even both. When both layers are disabled screen shows the default color, also configurable.

Buffer is not shown on the screen but can be used to prepare image before pushing it to one of layers.

Be aware that some operations spoil buffer content.

Layer indexes:

0 - background layer, color format RGB888

1 - foreground layer, color format ARGB1555

2 - buffer layer, color format ARGB8888

Colors in api commands are always in ARGB8888 format

3. Supported media types

- BMP images with 24 bits per pixel color
- JPG images, colored with 24 bits per pixel color
- PNG images, with 24 bits per pixel color or 32 bits per pixel color (with A channel)
- fonts in a special FNT format

Showing of JPG and PNG image spoils the buffer.

Using of JPG images is preferable as there is an imbedded support of JPG decoding.

4. Media uploading commands

flashinfo; - requests information about available flash memory

response: AAAASSSS

AAAA - 4 bytes, flash start address. usually 0

SSSS - 4 bytes, flash size in bytes

wrsect;AAAACCCCbbbb... - writes bytes to flash memory

AAAA - 4 bytes, flash address

CCCC - 4 bytes, number of bytes to write

bbb... - bytes to write

Number of bytes to write must be ≤ 4096 .

Flash is erased by 4096 sectors.

When AAAA is divisible by 4096, the corresponding sector is erased. Nothing is erased when AAAA is not divisible by 4096.

You MUST start write from address divisible by 4096 to erase the sector.

Then you can call wrsect with with chunks of any size to write new data.

It is better to have a chunk size = power of 2: 32, 64, 128, 256 etc. 256 is a flash memory page size. It's a good idea to use it.

rdsect;AAAACCCC - reads bytes from flash memory

AAAA - 4 bytes, flash address

CCCC - 4 bytes, number of bytes to read

response: bbb...

bbb... - bytes from flash memory

recache; - re-indexes media after flash memory was changed

response: ok\n

5. Drawing commands

draw_pic;LPPXXYY - draws picture from flash memory

L - 1 byte, layer index

PP - 2 bytes, picture index

XX - 2 bytes, picture left on the screen

YY - 2 bytes, picture top on the screen

response: ok\n

draw_pic2;LPPXXYYXIYIWWHHAa - draws picture from board memory

L - 1 byte, layer index

PP - 2 bytes, picture index

XX - 2 bytes, picture left on the screen

YY - 2 bytes, picture top on the screen

XI - 2 bytes, x coordinate on the picture

YI - 2 bytes, y coordinate on the picture

WW - 2 bytes, area width

HH - 2 bytes, area height

A - 1 byte, alpha value

a - 1 byte, alpha mode. 0 - no modify alpha, 1 - replace alpha, 2 - combine alpha

response: ok\n

draw_rect;LXXYYWWHHCCCC - draws rectangle

L - 1 byte, layer index

XX - 2 bytes, rectangle left

YY - 2 bytes, rectangle top

WW - 2 bytes, rectangle width

HH - 2 bytes, rectangle height

CCCC - 4 bytes, rectangle color in ARGB8888 format

response: ok\n

fill_rect2;LXXYYWWHHCCCC - draws filled rectangle

L - 1 byte, layer index

XX - 2 bytes, rectangle left

YY - 2 bytes, rectangle top

WW - 2 bytes, rectangle width

HH - 2 bytes, rectangle height

CCCC - 4 bytes, rectangle color in ARGB8888 format

response: ok\n

draw_rrect;LXXYYWWHRRCCCC - draws rounded rectangle

L - 1 byte, layer index
XX - 2 bytes, rectangle left
YY - 2 bytes, rectangle top
WW - 2 bytes, rectangle width
HH - 2 bytes, rectangle height
RR - 2 bytes, rectangle rounding radius
CCCC - 4 bytes, rectangle color in ARGB8888 format
response: ok\n

fill_rrect;LXXYYWWHRRCCCC - draws filled rounded rectangle

L - 1 byte, layer index
XX - 2 bytes, rectangle left
YY - 2 bytes, rectangle top
WW - 2 bytes, rectangle width
HH - 2 bytes, rectangle height
RR - 2 bytes, rectangle rounding radius
CCCC - 4 bytes, rectangle color in ARGB8888 format
response: ok\n

draw_circ;LXXYYRRCCCC - draws circle

L - 1 byte, layer index
XX - 2 bytes, circle center x
YY - 2 bytes, circle center y
RR - 2 bytes, circle radius
CCCC - 4 bytes, circle color in ARGB8888 format
response: ok\n

fill_circ;LXXYYRRCCCC - draws filled circle

L - 1 byte, layer index
XX - 2 bytes, circle center x
YY - 2 bytes, circle center y
RR - 2 bytes, circle radius
CCCC - 4 bytes, circle color in ARGB8888 format
response: ok\n

draw_tria;LX1Y1X2Y2X3Y3CCCC - draws triangle

L - 1 byte, layer index
X1 - 2 bytes, triangle point 1 x
Y1 - 2 bytes, triangle point 1 y
X2 - 2 bytes, triangle point 2 x
Y2 - 2 bytes, triangle point 2 y
X3 - 2 bytes, triangle point 3 x
Y3 - 2 bytes, triangle point 3 y
CCCC - 4 bytes, circle color in ARGB8888 format
response: ok\n

fill_tri;**LX1Y1X2Y2X3Y3CCC** - draws filled triangle.

L - 1 byte, layer index
X1 - 2 bytes, triangle point 1 x
Y1 - 2 bytes, triangle point 1 y
X2 - 2 bytes, triangle point 2 x
Y2 - 2 bytes, triangle point 2 y
X3 - 2 bytes, triangle point 3 x
Y3 - 2 bytes, triangle point 3 y
CCCC - 4 bytes, circle color in ARGB8888 format
response: ok\n

draw_pixel;**LXXYYCCCC** - draws pixel

L - 1 byte, layer index
XX - 2 bytes, pixel point x
YY - 2 bytes, pixel point y
CCCC - 4 bytes, circle color in ARGB8888 format
response: ok\n

draw_line;**LX1Y1X2Y2CCCC** - draws line

L - 1 byte, layer index
X1 - 2 bytes, line point 1 x
Y1 - 2 bytes, line point 1 y
X2 - 2 bytes, line point 2 x
Y2 - 2 bytes, line point 2 y
CCCC - 4 bytes, circle color in ARGB8888 format
response: ok\n

draw_text;**LXXYYCCCCBBBBFFSCTTTTTTTT** - draws text

L - 1 byte, layer index
XX - 2 bytes, text start point x
YY - 2 bytes, text start point y
CCCC - 4 bytes, text color in ARGB8888 format
BBBB - 4 bytes, text background color in ARGB8888. Must be equal to CCCC if not needed
FF - 2 bytes, font index
S - 1 byte, font scaling. 1 - normal scale. 2 - 2x scale, etc.
C - 1 byte, number of characters
T - C bytes, text characters
response: ok\n

draw_trct;LFFXXYYWWHHCCCCBBBBSWrChCvAvAhCTTTTTTTTT - draws text with alignment options

L - 1 byte, layer index

FF - 2 bytes, font index

XX - 2 bytes, text rect left point

YY - 2 bytes, text rect top point

WW - 2 bytes, text rect width

HH - 2 bytes, text rect height

CCCC - 4 bytes, text color in ARGB8888 format

BBBB - 4 bytes, text background color in ARGB8888. Must be equal to CCCC if not needed

S - 1 byte, font scaling. 1 - normal scale. 2 - 2x scale, etc.

Wr - 1 byte, 1 - wrap text

Ch - 1 byte, 1 - clip text horizontally

Cv - 1 byte, 1 - clip text vertically

Av - 1 byte, vertical alignment, 0 - top, 1 - bottom, 2 - center

Ah - 1 byte, horizontal alignment, 0 - left, 1 - right, 2 - center

C - 1 byte, number of characters

T - C bytes, text characters

response: ok\n

draw_trct2;LFFXXYYWWHHPpPxPyAaCCCCSWrChCvAvAhCTTTTTTTTT - draws text with alignment options and background picture. Spoils buffer content

L - 1 byte, layer index

FF - 2 bytes, font index

XX - 2 bytes, text rect left point

YY - 2 bytes, text rect top point

WW - 2 bytes, text rect width

HH - 2 bytes, text rect height

Pp - 2 bytes, picture indes

Px - 2 bytes, picture left coord

Py - 2 bytes, picture top coord

A - 1 byte, alpha value

a - 1 byte, alpha mode. 0 - no modify alpha, 1 - replace alpha, 2 - combine alpha

CCCC - 4 bytes, text color in ARGB8888 format

S - 1 byte, font scaling. 1 - normal scale. 2 - 2x scale, etc.

Wr - 1 byte, 1 - wrap text

Ch - 1 byte, 1 - clip text horizontally

Cv - 1 byte, 1 - clip text vertically

Av - 1 byte, vertical alignment, 0 - top, 1 - bottom, 2 - center

Ah - 1 byte, horizontal alignment, 0 - left, 1 - right, 2 - center

C - 1 byte, number of characters

T - C bytes, text characters

response: ok\n

clc_txt_rect;FFXXYYLLRRSWrCTTTTTTTT - calculates size of rect covered by text

FF - 2 bytes, font index

XX - 2 bytes, text rect left point

YY - 2 bytes, text rect top point

LL - 2 bytes, left limit of text rect

RR - 2 bytes, right limit of rect

S - 1 byte, font scaling. 1 - normal scale. 2 - 2x scale, etc.

Wr - 1 byte, 1 - wrap text

C - 1 byte, number of characters

T - C bytes, text characters

response: LWWHH

L - 1 byte, total number of bytes in the response. must be 5

WW - 2 bytes, text rect width

HH - 2 bytes, text rect height

6. Sensors

get_temp; - returns board temperature

response: LTT

L - 1 byte, total number of bytes in the response. must be 3

TT - 2 bytes, board temperature multiplied by 100. ie 3660 means 36.6 celsius

get_light; - returns lighting sensor measurements

response: LSMM

L - 1 byte, total number of bytes in the response. must be 4

S - 1 byte, sensor status. bit0 = 0 - sensor not ready, bit0 = 1 - sensor OK

MM - 2 bytes, sensor measurement. must be in lux

get_time; - returns current board time

response: LSYMDhms

L - 1 byte, total number of bytes in the response. must be 8

S - 1 byte, rtc status. bit0 = 0 - rtc error, bit0 = 1 - rtc OK

Y - 1 byte, year

M - 1 byte, month

D - 1 byte, day

h - 1 byte, hour

m - 1 byte, minute

s - 1 byte, second

set_time;YMDhms - sets current board time

Y - 1 byte, year

M - 1 byte, month

D - 1 byte, day

h - 1 byte, hour

m - 1 byte, minute

s - 1 byte, second

response: ok\n

get_rtcram;AC - returns board rtc ram content

A - 1 byte, start address (from 0 to 0x3F)

C - 1 byte, number of bytes to read

response: LSCB..

L - 1 byte, total number of bytes in the response

S - 1 byte, rtc read status. bit0 = 0 - rtc read error, bit0 = 1 - rtc read OK

C - 1 byte, number of following bytes

B..- C bytes, bytes from rtc ram

set_rtcram;ACB.. - writes to board rtc ram

A - 1 byte, start address (from 0 to 0x3F)

C - 1 byte, number of bytes to write

B..- C bytes, bytes to write to rtc ram

response: ok\n

7. Touch and screenshot

get_touch; - reads touch screen status

response: LISXXYY

L - 1 byte, total number of bytes in the response

I - 1 byte, touch state index. is incremented on every touch state change. can be used to find out if touch state was changed since previous read

S - 1 byte, touch status. bit0 - touch state changed (have the same meaning as I if only one system reads the touch state),

bit1 - touch is down, bit2 - touch is pressed, bit3 - touch is up,

bit7-bit4 - touch gesture: 1 - swipe up, 2 - swipe right, 3 - swipe down, 4 - swipe left, 5

- double click

XX - 2 bytes, touch X position

YY - 2 bytes, touch Y position

touch_capton;XXYYWWHHC - starts touch capture

XX - 2 bytes, left position of capture area, all positions must be divisible by 8

YY - 2 bytes, top position of capture area

WW - 2 bytes, width of capture area

HH - 2 bytes, height of capture area

C - 1 byte, control byte. bit0=1 - draw ok/cancel buttons

response: ok\n

touch_captoff;C - checks touch capture status or stops it

C - 1 byte, control byte. bit0=1 - check status, bit0=0 - stop touch capture

response: LS

L - 1 byte, total number of bytes in the response

S - 1 byte, touch capture status. bit0=1 - touch capture active

if touch capture not active there are more bytes in the response. Spoils buffer content:

AAAACCCC

AAAA - 4 bytes, ram address where jpg image produced from user input is located

CCCC - 4 bytes, size of jpg image
jpg image can be downloaded with read_ram command

get_screenshot;C - captures layer content in jpg image. Spoils buffer content
C - 1 byte, control byte. 1 - capture background layer, 2 - capture foreground layer
response: LAAAACCCC
L - 1 byte, total number of bytes in the response
AAAA - 4 bytes, ram address where jpg image is located
CCCC - 4 bytes, size of jpg image
jpg image can be downloaded with read_ram command

read_ram;AAAACCCC - captures layer content in jpg image. Spoils buffer content
AAAA - 4 bytes, ram address
CCCC - 4 bytes, number of bytes to read
response: bbb...
bbb... - bytes from ram

8. Screen controls

tft_on; - turns tft on. just tft, does not draw anything or affect the backlight. can be used to reduce flickering - turn tft on, draw what is needed, turn backlight on
response: ok\n

tft_off; - turns tft off. just tft, does not affect the backlight. but with the current screen, when tft is off backlight also turns off automatically
response: ok\n

screen_on; - turns screen on: turns tft on, fills screen with black, turns backlight on
response: ok\n

screen_off; - turns screen off. both tft and backlight
response: ok\n

screen_bkl_on; - turns backlight on
response: ok\n

screen_off; - turns backlight off
response: ok\n

screen_bkl;X - sets screen backlight.
X - 1 backlight value from 0 (no backlight) to 10 (maximum)
response: ok\n

9. Layers

layer_copy;LfLtXXYYxxyyWWHHAa - copies area from one layer to another layer

Lf - 1 byte, source layer index

Lt - 1 byte, destination layer index

XX - 2 bytes, left on the destination layer

YY - 2 bytes, top on the destination layer

xx - 2 bytes, left on the source layer

yy - 2 bytes, top on the source layer

WW - 2 bytes, area width

HH - 2 bytes, area height

A - 1 byte, alpha value

a - 1 byte, alpha mode. 0 - no modify alpha, 1 - replace alpha, 2 - combine alpha

response: ok\n

layer_show;LC - shows/hides layer

L - 1 byte, layer index

C - 1 byte, control. 0 - hide, 1 - show

response: ok\n

bg_color;RGB - sets screen background color. used when all layers are not shown

R - 1 byte, red part of background color

G - 1 byte, green part of background color

B - 1 byte, blue part of background color

response: ok\n

10. Sounds

note_bpm;B - sets beats per minute speed

B - 1 byte, bpm from 0 to 255

response: ok\n

note_status; - checks if sound is playing

response: LS

L - 1 byte, total number of bytes in the response. must be 2

S - 1 byte, status. 0 - not playing, 1 - playing

note_play;CNdNdNdNd... - plays notes

C - 1 byte, number of notes

Nd - 2 bytes, note itself. There must be C of Nd sequences. First byte of note is note name: C, c, D, d, E, F, f, G, g, A, a, B, where note name in lower case is note #

Second byte is: bits3-0 - note length, bits7-4 - note octave