Industrial-grade Smart Touch HMI Display Computer

by ProtoDevs® GmbH

Datasheet for V.1.9 Rev.C

PD.Screen Family description

"**PD.Screen**" 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 an HMI touch module giving the system a full power of graphical user interface (GUI). PD.Screen modules family consist of 4" , 7", 10.1" industrial-grade TFT matrixes + driver board with a firmware that gives a user a simple interface to interact with. All the complex hardware-type interfaces and commands between PD.Screen 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 PD.Screen 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.

"**PD.Screen_PRO**" family has an additional MCU (ESP32 for v.1.9) and a rich set of onboard drivers so the module can be programmed as a standalone active device or peripheral device depending on the task. ESP32 examples are written for Arduino IDE and Python. The module's API-accessible core is based on the newest STM32 microcontroller utilizing the power and cost-effectivines of the latest industry standarts, STM32 Firmware could be updated via I2C / UART or SWDIO.

There are several PD.Screen PRO variants available :

PD.Screen_7inch_PRO_m30_NT	: 7-inch TFT, no touch feature
PD.Screen_7inch_PRO_m30	: 7-inch capacitive-touch TFT
PD.Screen_7inch_PRO_CSTM	: Custom edition, individual design for Customers

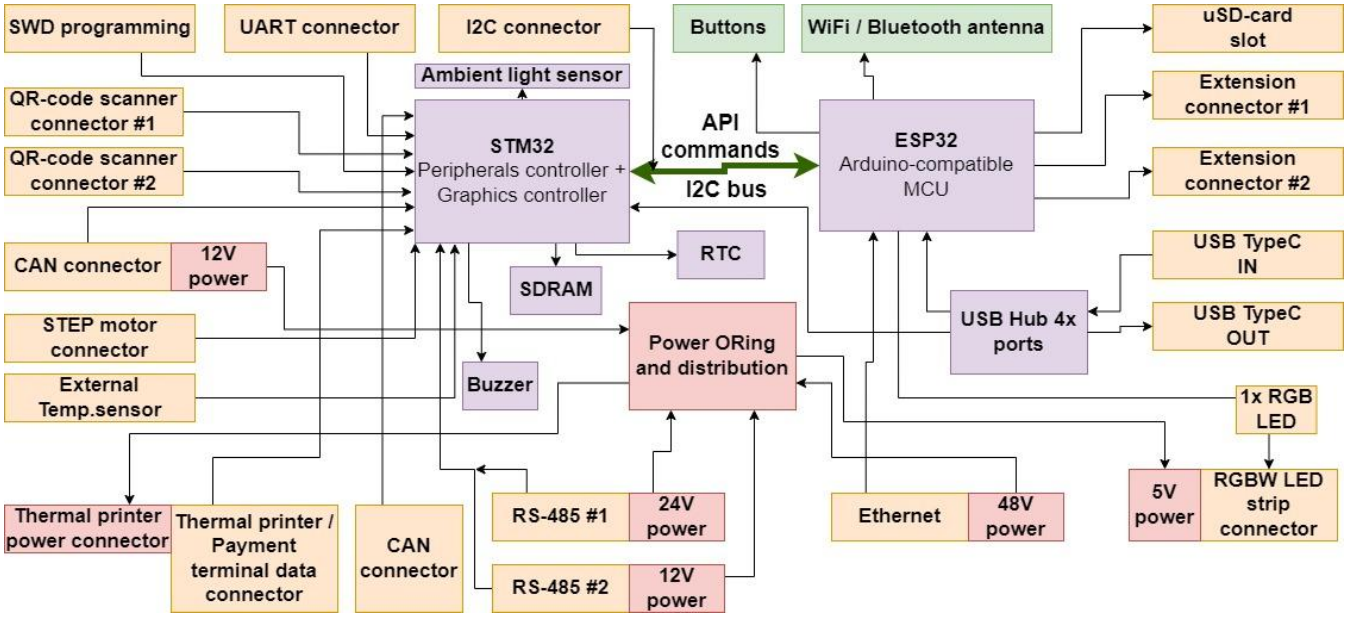
All the PD.Screen_7inch_PRO modules has a full set of powering and peripherals. See the description below.

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

Table of Contents

Simplified architectural block diagram

(PDScreen_7inch_PRO V.1.9 Rev.C)



Operational conditions

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

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

Input powering variants: **5V** on USB Type C, **12V** on J4, **24V** on X9 and X10, **48V PoE** on J12

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**)

RS485 speed: up to 10 megabit per second (**10 Mbps**), up to 256 devices on the bus

Ethernet speed: up to Fast Ethernet (**100 Mbps**)

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

General specifications

PD.Screen_7inch_PRO (v.1.9):

Screen parameters:

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

Powering parameters (all the inputs are OR-ed, single or any combination could be used):

Input connector	Voltage Min, V	Voltage Avg, V	Voltage Max, V	Remark
J7, USB Type C	4.9	5.0	5.1	USB ™ Specified
J4 (12V nominal)	3.5	12	28	
X9 (24V nominal)	8	24	72	
X10 (12V nominal)	8	12	28	
J12 (48V nominal)	12	48	72	PoE (wires **+** 4,5 / wires **-** 7,8)

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 12V.

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

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

	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

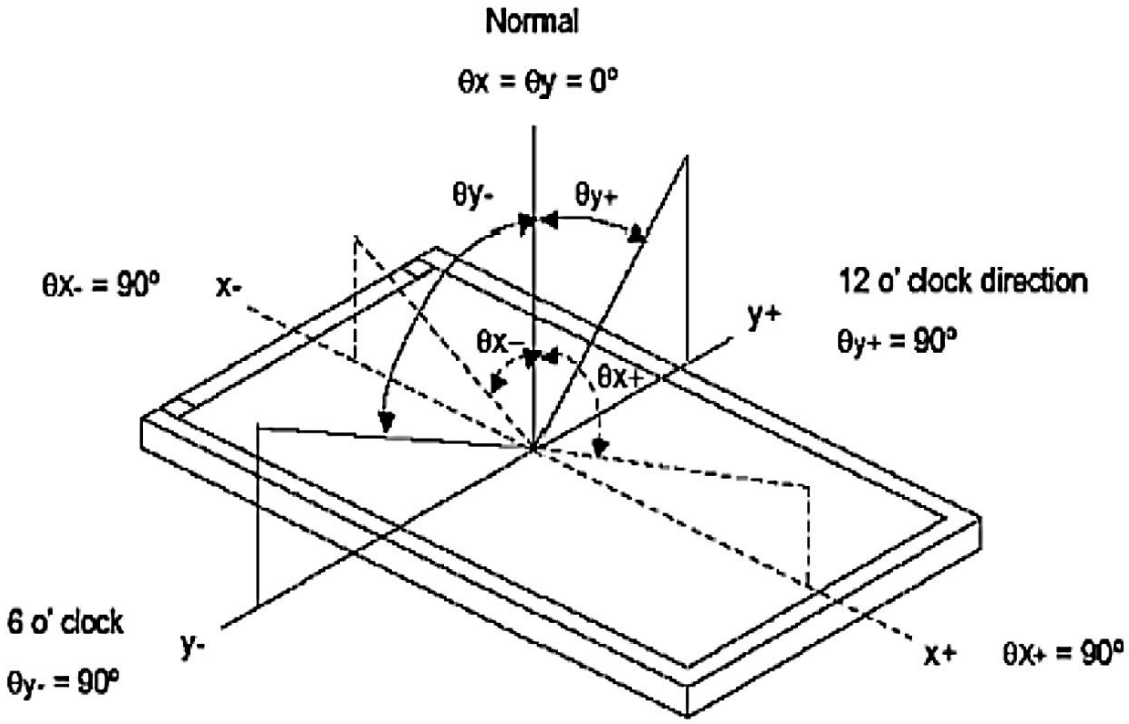
LCD TFT screen optical characteristics

PD.Screen_7inch_PRO (v.1.9):

ITEM		SYMBOL	CONDITIONS	SPECIFICATIONS			UNIT	NOTE
				MIN	TYP.	MAX		
Luminance		L		320	400	-	Cd/㎡	
Contrast ratio		CR	θ =0°	400	500			
Response time	Rising	TR	25℃		10	20	ms	
	Falling	TF			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≥10	60	70		Degree	
		θ x−		60	70			
	Ver.	θ y+		40	50			
		θ y−		60	70			

Viewing angles definition

PD.Screen_7inch_PRO (v.1.9):



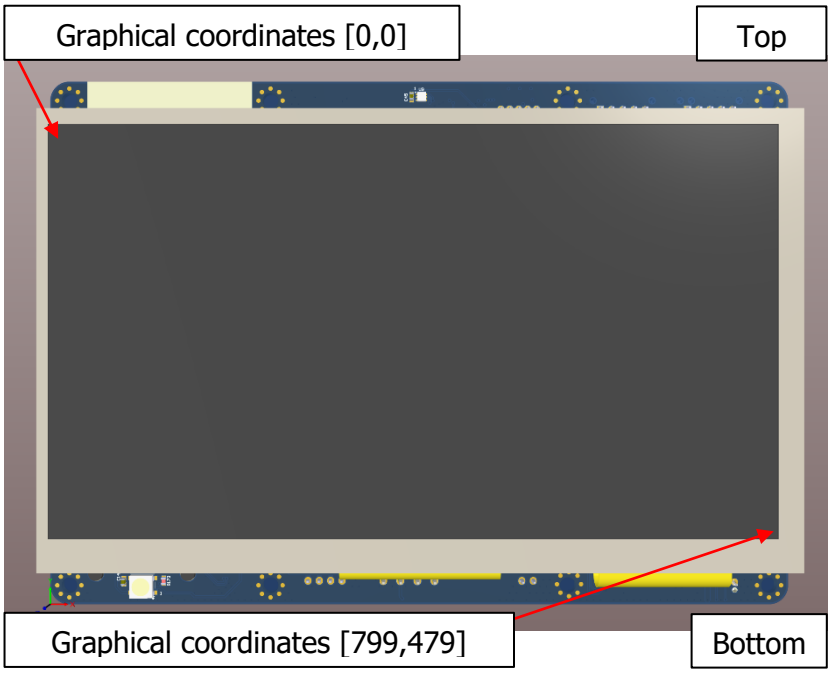
Graphical coordinates definition

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

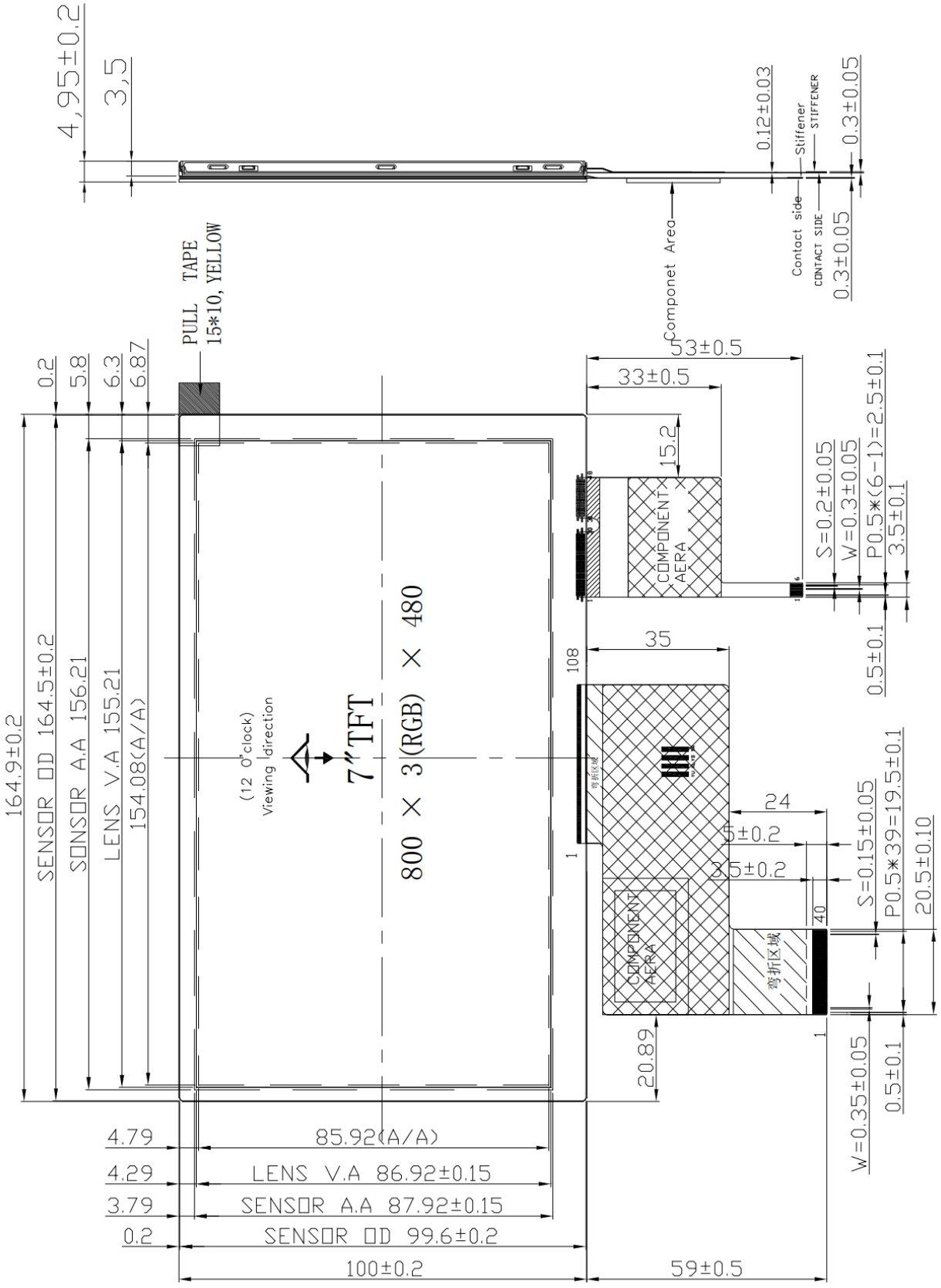
Top side has light sensor and 2x connectors, bottom side has 1x connector and flat cables from the TFT screen.

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)

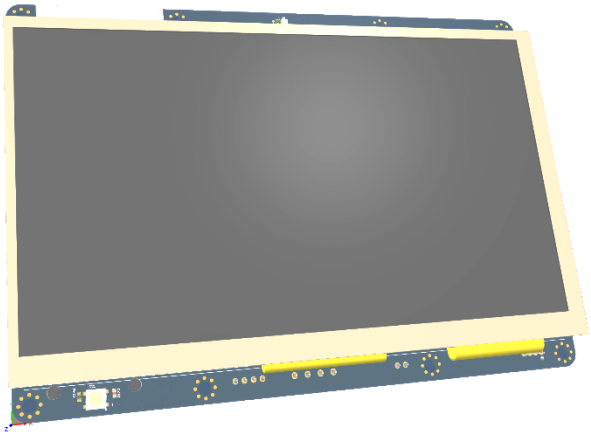


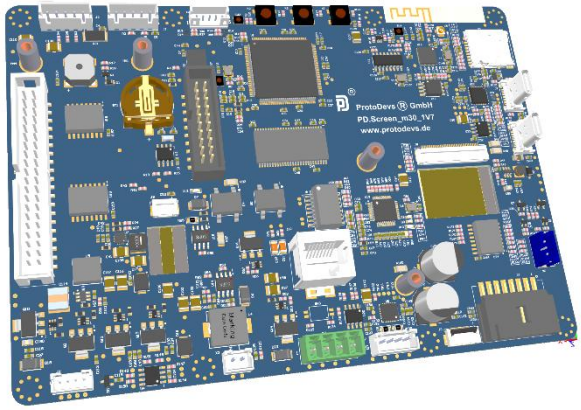
Mechanical dimensions (PD.Screen PRO module)

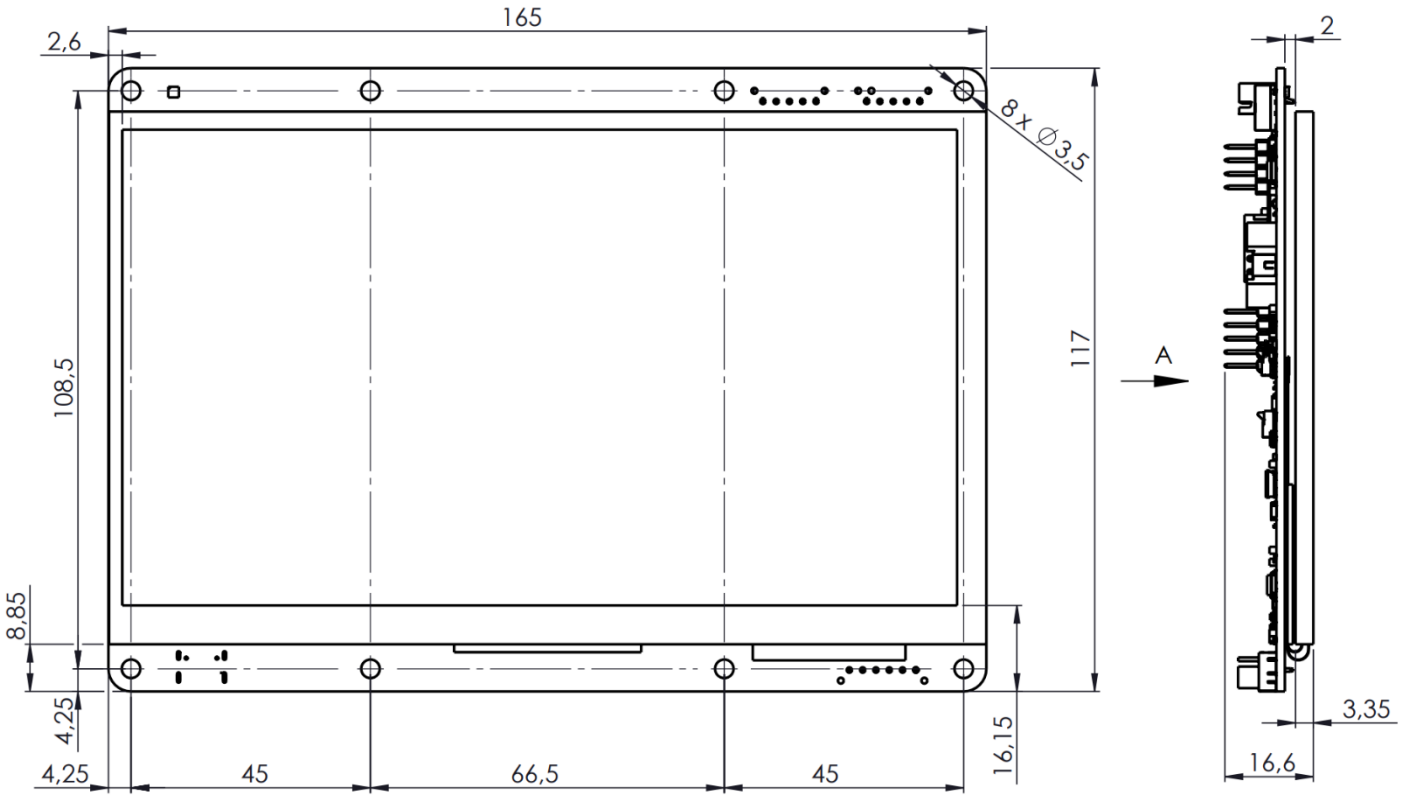
(For PD.Screen_7inch_PRO)

Dimensions (assembled module v.1.9):

117mm X 165mm X 19,2mm







PD.Screen 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.

PD.Screen 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.

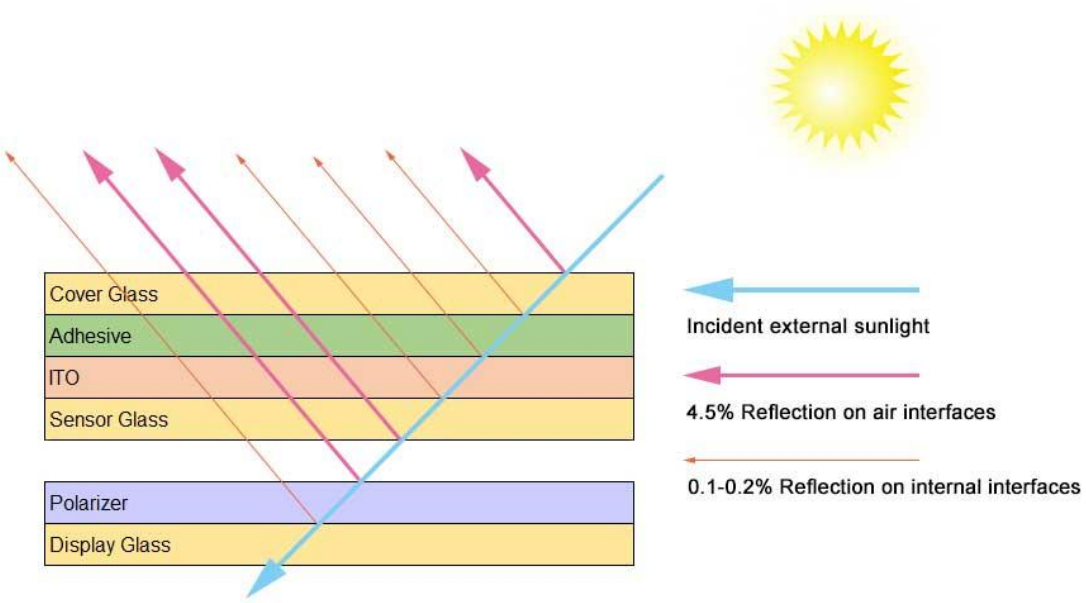
Sun reflectance reduction tips while installation

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=[(n2-n1)/(n2+n1)]^2 (n1 & n2 are indexes of refraction for 1st and 2nd material)

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

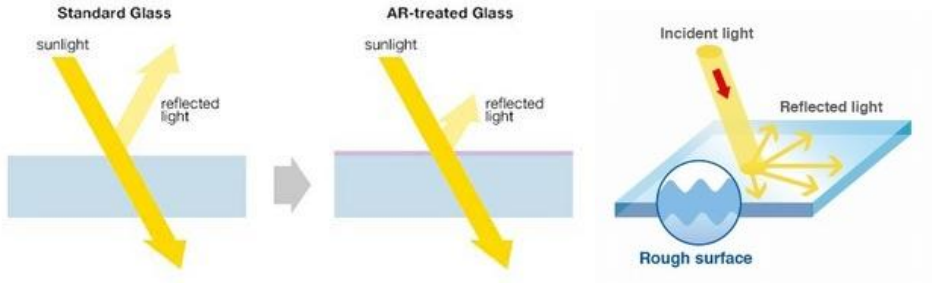
The regular TFT LCD module with touch panel has the multilayer structure. There are several places light reflection happens.



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.



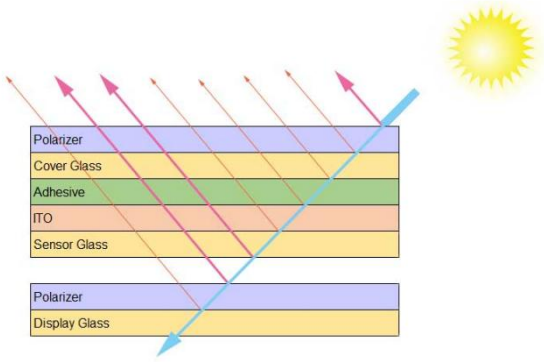
An LCD screen with external film solves this issue nicely. As for automotive applications, in an accident, broken LCD with top AR film won't produce sharp edge glass that could harms auto occupant. Nevertheless, a top film always reduces TFT LCD's surface hardness. And it is susceptible to scratches. 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

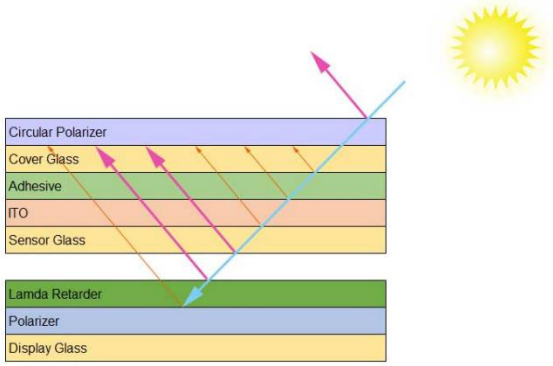
Another quick and easy way to tackle reflectance is to affix a linear polarizer on the top of TFT screen. When ambient light gets to the top polarizer, only half of the light passes through. Which results in reflection light cutting to half. This is a very low cost way to increase TFT LCD's contrast,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

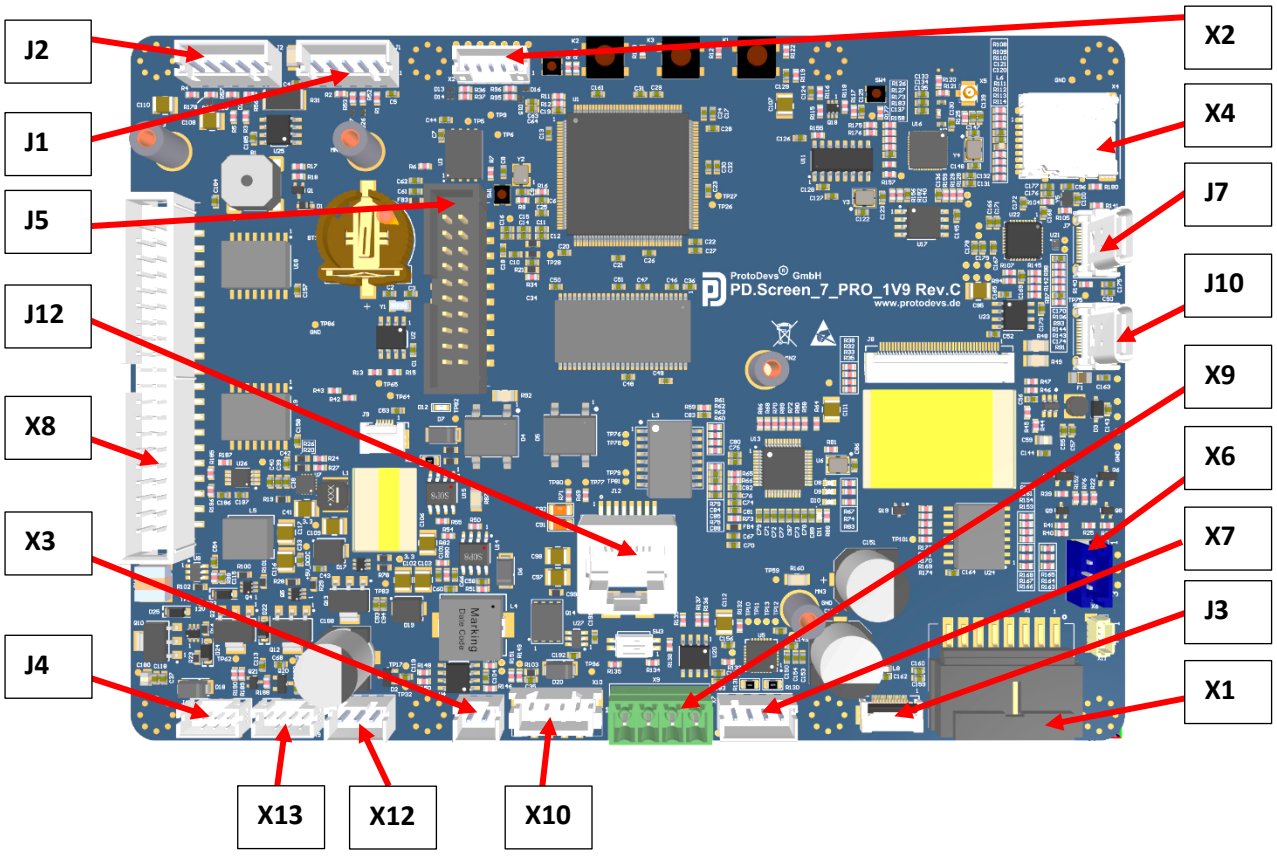
Add AR films on both interfaces of internal air gap. The add-ons can reduce this area's reflection from 8.5% to 2%. And since the AR films are not outside facing, they are much cheaper than the one used outside. Keeping the air gap also retains the ease of service, in case either touch panel or LCD display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

External connector's function and placement table

PD.Screen_7inch_PRO (v.1.9 Rev.C):

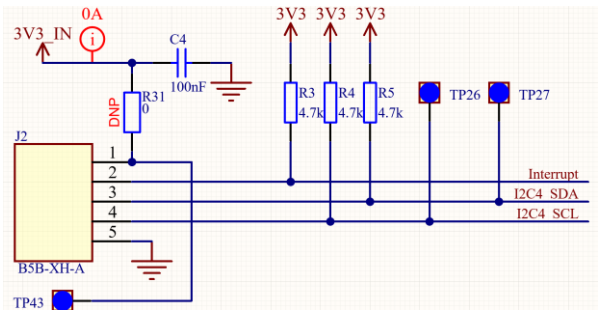
J2	**I2C** connector
J1	**UART** connector
J5	**GPIO** port, has **SPI** from ESP32, other GPIOs from STM32 (not accesible for users, for custom designs only)
J12	**Ethernet** with **PoE** 48V DC**, up to 100 Mbit**
X8	**GPIO** port, ESP32 : UART and I2C, **16x GPIOs** from 2x **PCF8574 (port extenders)** connected to **ESP32 I2C bus**
X3	External thermo-sensor channel (NTC), accesible from custom STM32 firmware (by request)
J4	**CAN** connector (from SN65HVD231), also has **VIN_12V** and **GND (12V DC Input)**
X1	**GPIO** port, **8x GPIOs** from **PCF8574 (port extender)** connected to **ESP32 I2C bus**
J3	**QR barcode module** connector for scanner (1D and 2D) in UART mode
X7	**STEP Motor driver** connector, TMC2209-LA-T chip controlled by ESP32
X6	**RGB LED Strip** connector, has connection to the one RGB front Led which receives command from ESP32
X9	**RS-485** connector, daisy-chain connections supported
X10	**RS-485** connector, for daisy-chain connections
X12	**Thermal printer power** connector, switchable (5V out, customisable)
X13	**UART** for Thermal printers or payment Terminals (TxD, RxD, DTR)
J10	**USB Type-C** downlink, is used only while **J7** connected to the host due to onboard USB hub operation
J7	**USB Type-C** uplink to host, connect to PC to be able to flash ESP32, to see UART from STM32 and to use **J10**
X4	**SD Card connector**, direct connection to ESP32 SPI
X2	**STM32 SWDIO connector** to reprogramm the STM32 in case of need



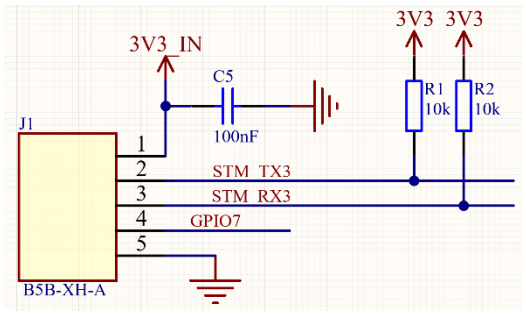
Communication interfaces pinouts

(For PD.Screen_7inch_PRO)

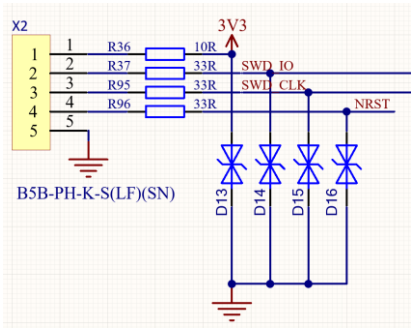
I2C / 3V3 (**J2**): ***R31 is not soldered by default**



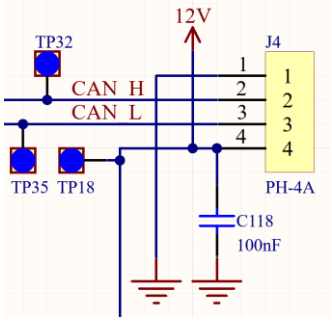
UART (**J1**):



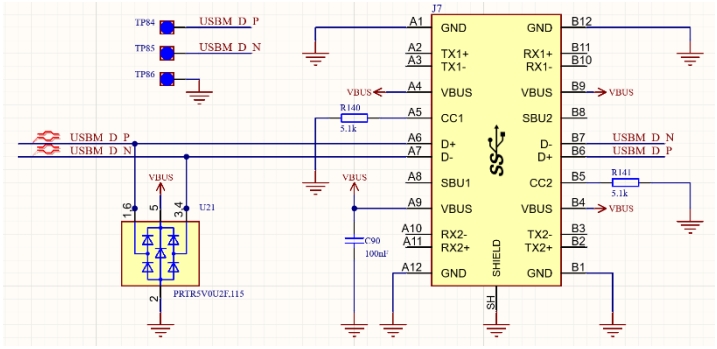
STM32 **SWDIO** (**X2**)



CAN /12V (**J4**)

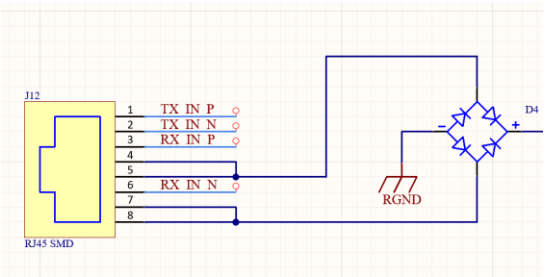


USB Type-C Upsteram /**5V in** (**J7**)



Ethernet / **48V** (**J12**)

IEEE 802.3af (**Mode B**)

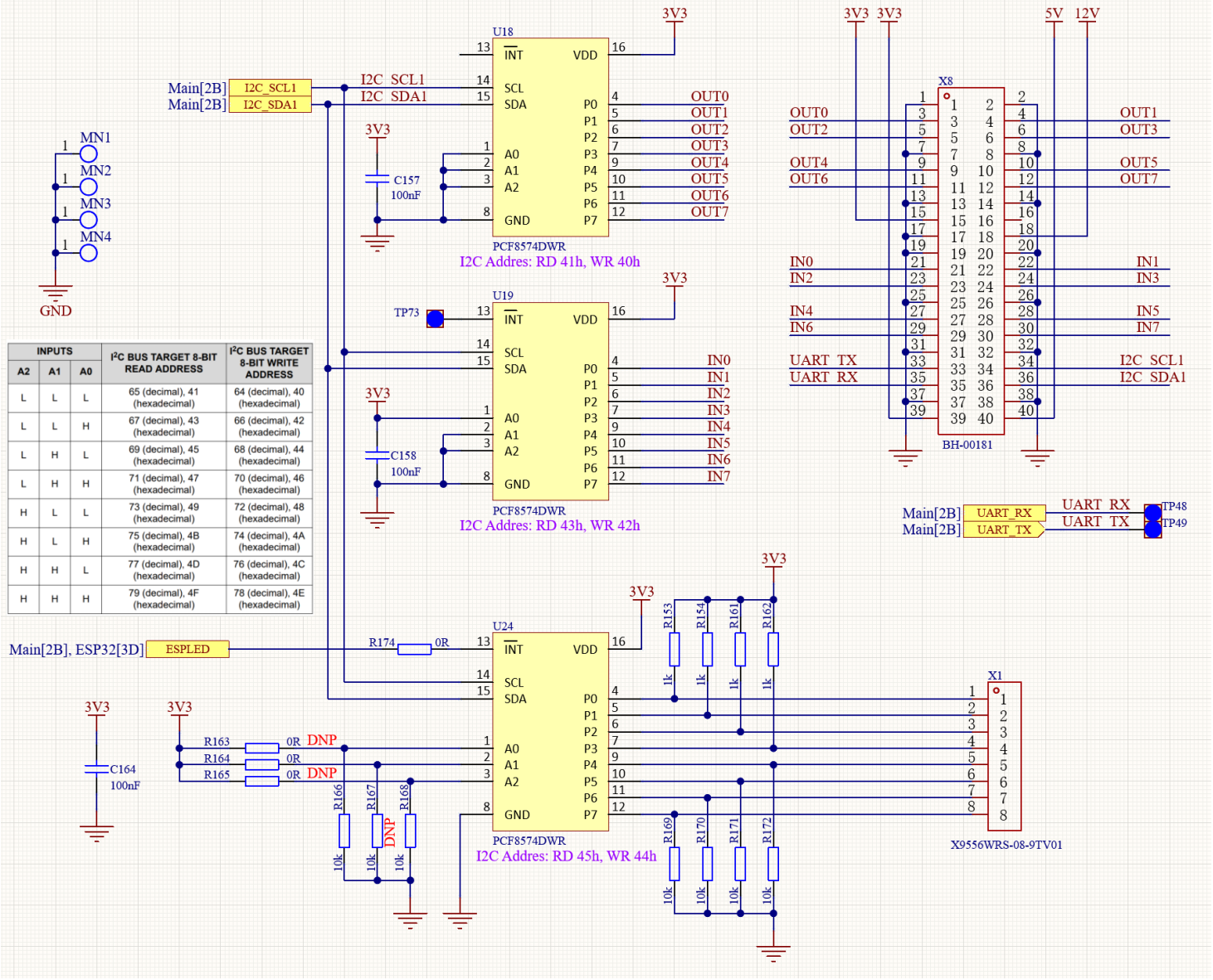


Wires 4,5 = DC **VCC** (Alt. GND)

Wires 7,8 = DC **GND** (Alt. Vin)

GPIO extension connectors (**X1** and **X8**):

Common I2C bus (I2C_SDA1, I2C_SCL1) from all 3x PCF8574 is connected to **ESP32**:
SDA1 = pin 25 = GPIO16 ;
SCL1 = pin 27 = GPIO17 ;



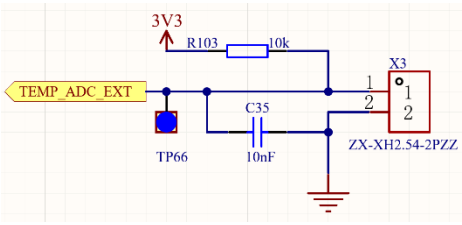
INPUTS			I²C BUS TARGET 8-BIT READ ADDRESS	I²C BUS TARGET 8-BIT WRITE ADDRESS
A2	A1	A0		
L	L	L	65 (decimal), 41 (hexadecimal)	64 (decimal), 40 (hexadecimal)
L	L	H	67 (decimal), 43 (hexadecimal)	66 (decimal), 42 (hexadecimal)
L	H	L	69 (decimal), 45 (hexadecimal)	68 (decimal), 44 (hexadecimal)
L	H	H	71 (decimal), 47 (hexadecimal)	70 (decimal), 46 (hexadecimal)
H	L	L	73 (decimal), 49 (hexadecimal)	72 (decimal), 48 (hexadecimal)
H	L	H	75 (decimal), 4B (hexadecimal)	74 (decimal), 4A (hexadecimal)
H	H	L	77 (decimal), 4D (hexadecimal)	76 (decimal), 4C (hexadecimal)
H	H	H	79 (decimal), 4F (hexadecimal)	78 (decimal), 4E (hexadecimal)

X1 has 8-pin angled connector on the PCB + interrupt pin with purpose: this connector should be used for the external keyboards (tact or membrane type, antivandal option, custom shape design).

X8 together with **J5** could be used to attach the dedicated prototyping breadboard or those connectors could be used for custom extension modules design (by ProtoDevs or by user),

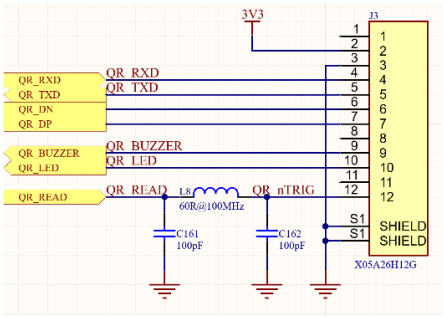
contact info@protodevs.de for details.

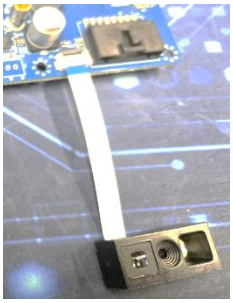
Ext. NTC (**X3**)



Sensor types to connect (10k):

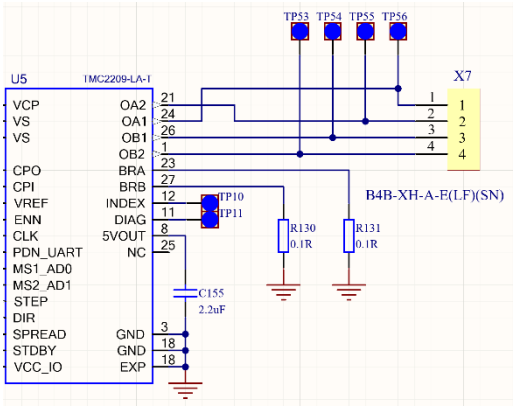
QR barcode scanner (**J3**)





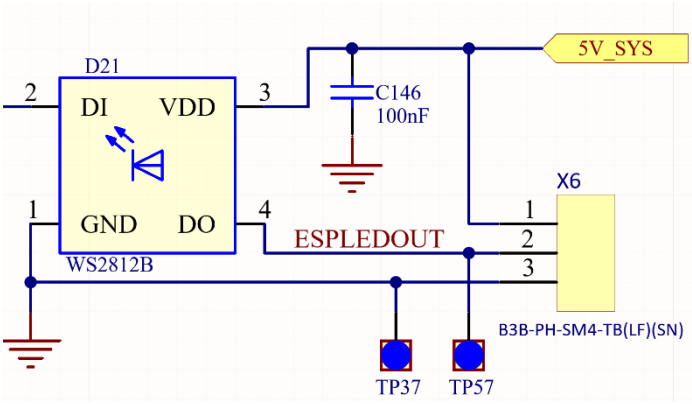
QR module Front **must be** oriented as PCB component side

Motor driver connector (**X7**)



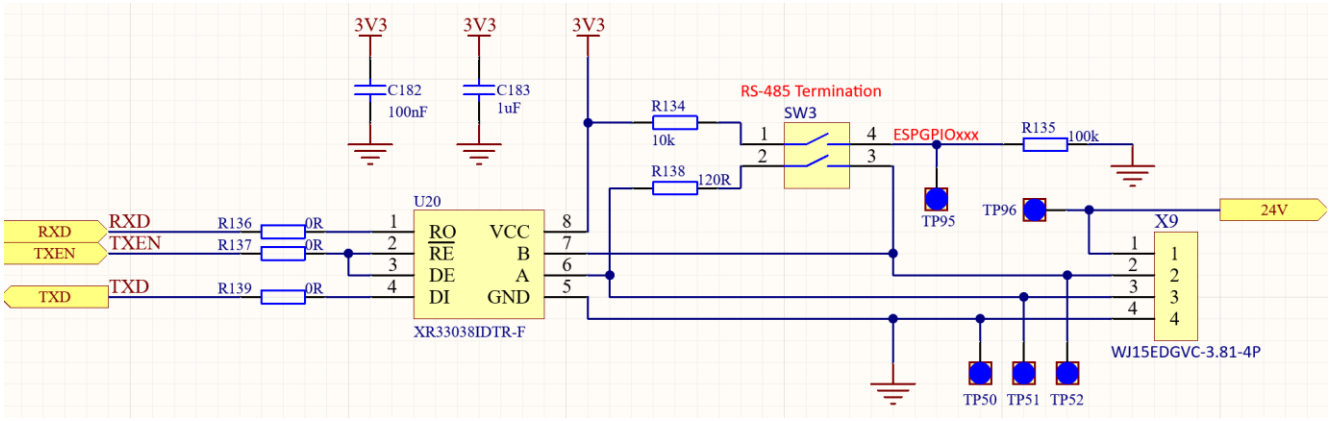
Motors to connect: NEMA 17/23

RGB LED Strip connector (**X6**)

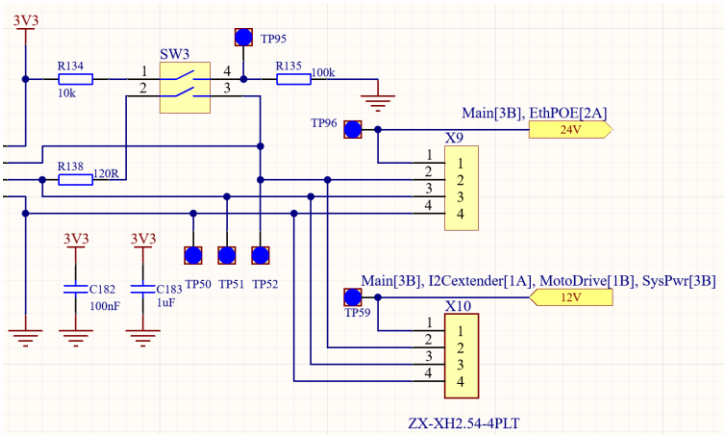


Single (1st) RGB LED D21 is installed on the module PCB

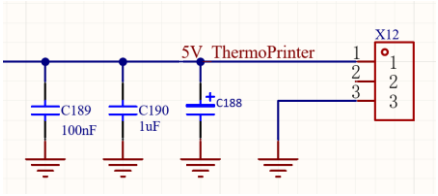
RS-485 connector (**X9**)



X10 connector: **RS-485** for daisy-chain connections

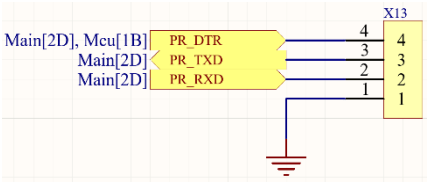


X12 connector: **Thermal printer power** connector switchable (5V out, customisable)

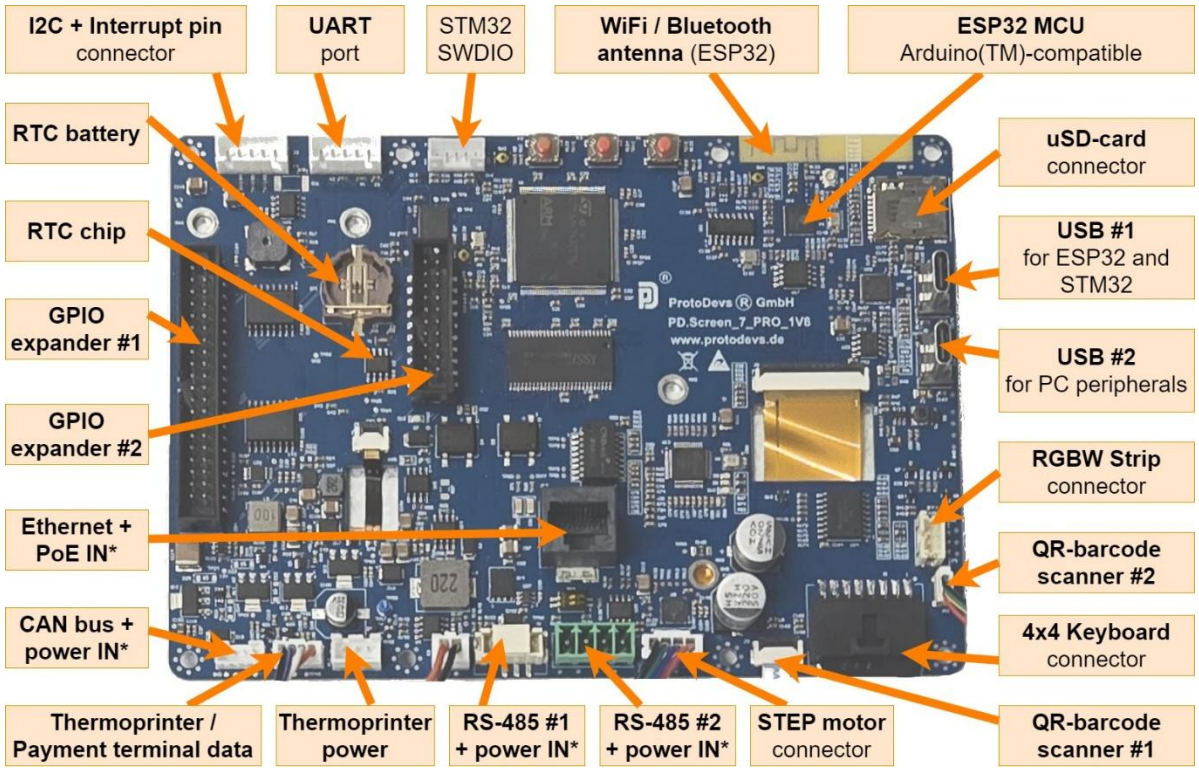


Connector type: ZX-XH2.54-3PZZ

X13 connector: **UART** for Thermal printers or payment Terminals (TxD, RxD, DTR)



Connector type: PH-4A



Preparations for PD.Screen_PRO 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 reqiures **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 (see '**Fonts FNT format description**' section)

[download] section:
dir parameter contains name of directory where to store images downloaded from the board
[commands] section:
Contains commands executed by exec run option. Allowed commands:
***name [xx]** - executes commands from section [name].
xx - optional parameter, number of times to call the section, * - infinite loop. ex: *tst1 - calls tst1 section 1 time, *tst1 12 - calls tst1 section 12 times, *tst1 * - calls tst1 sections in infinite loop

3. Run

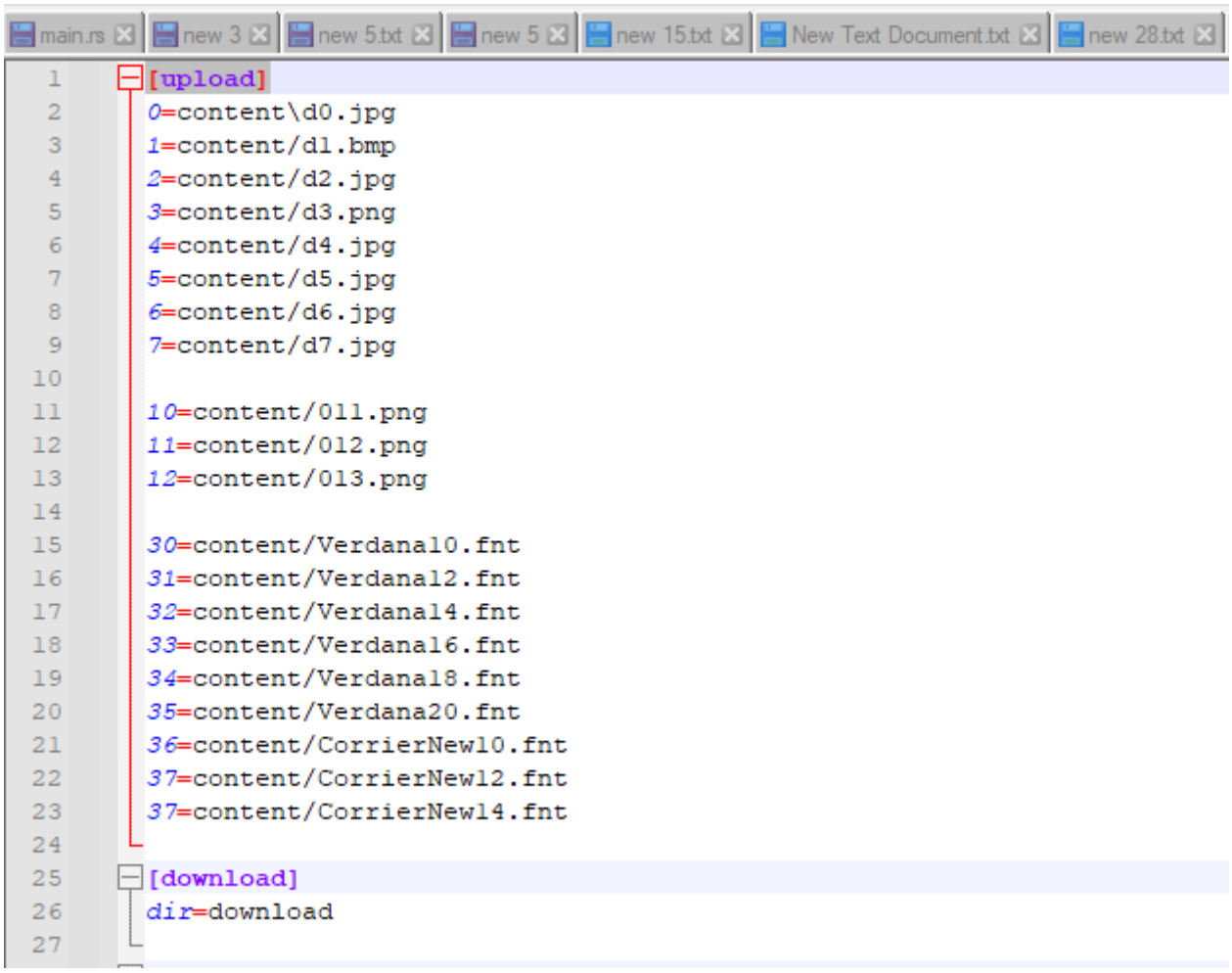
Run script with **'upload'** parameter to upload pictures to the board:
sudo python3 ProtoScreen.py upload

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.



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

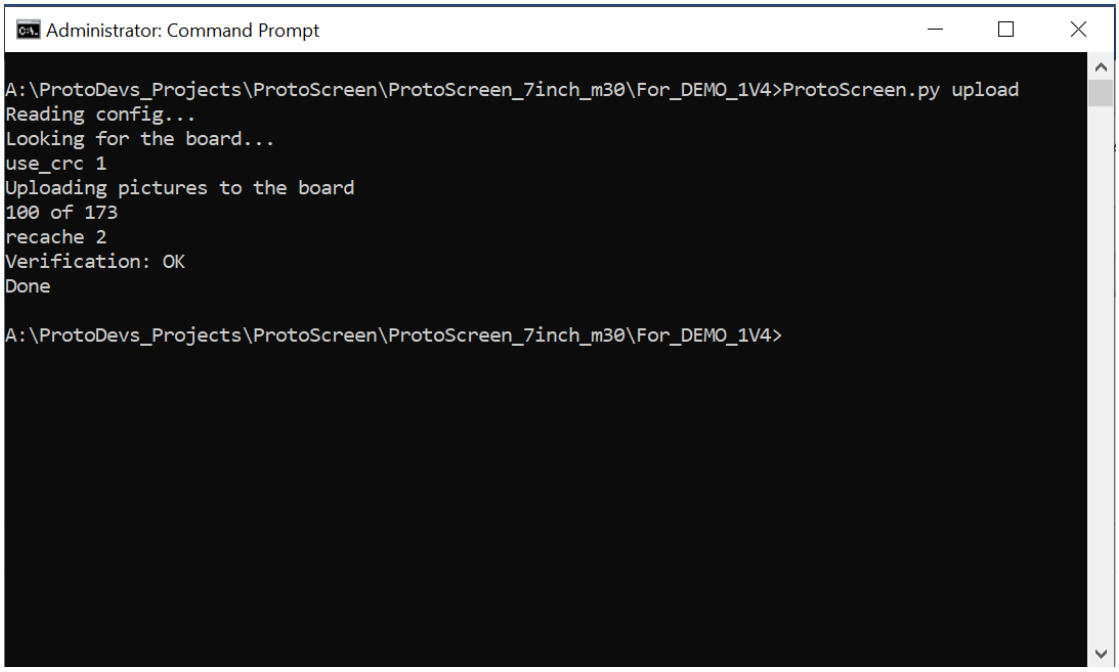
ProtoScreen.py upload COM18

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.

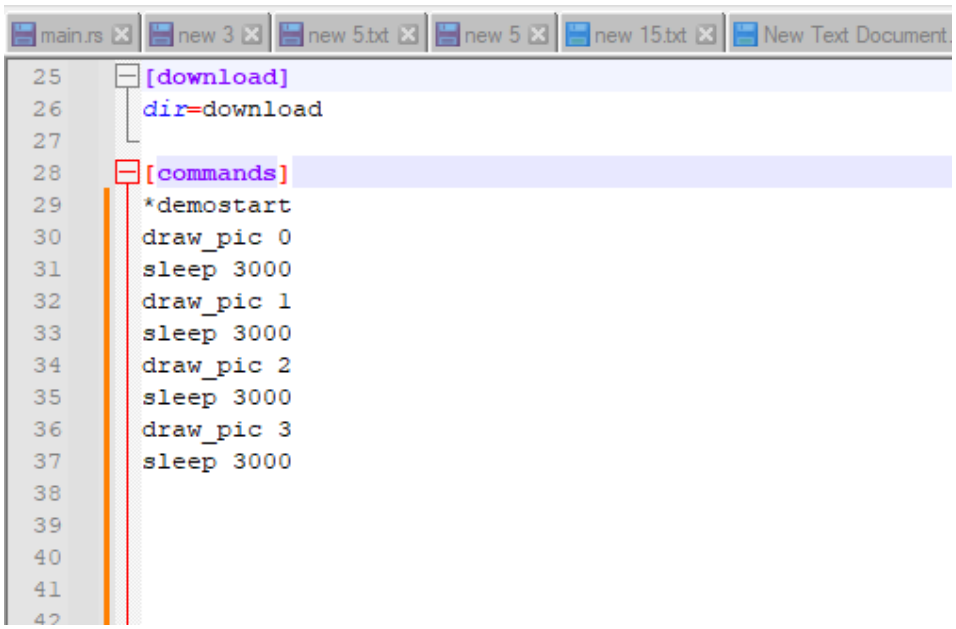
Example of the **Upload** command execution:



Running the script commands example

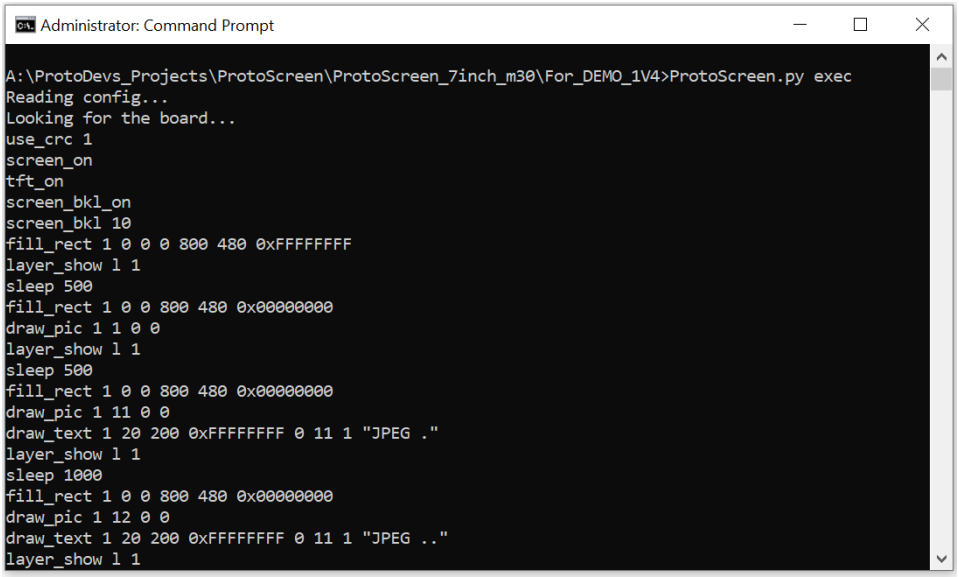
Script can be used to issue commands to the board. You can use it to check the uploaded images, for example. Commands must be listed in **[commands]** section of **ProtoScreen.ini** file.

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.



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

ProtoScreen exec or **ProtoScreen exec COM18**



List of script commands supported (UART)

[commands] section:

Contains commands executed by exec run option. Allowed commands:

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

xx - optional parameter, number of times to call the section,

* - infinite loop. examples:

***tst1** - calls tst1 section 1 time,

***tst1 12** - calls tst1 section 12 times,

***tst1 *** - calls tst1 sections in infinite loop

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

fill_tria l x1 y1 x2 y2 x3 y3 clr - draws filled triangle (fill_tria 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

qr_print l symbol x y w h scale av ah opts clr_ink clr_paper clr_bg data data_prime - draws barcode symbol.
l - layer index, symbol - symbol index,
x, y - coordinates of start point of symbol rect,
w, h - symbol rect size, scale - symbol scale (if 0 - maximum scale to fit the rect is used),
av - vertical alignment (0 - top, 1 - bottom, 2 - center),
ah - horizontal alignment (0 - left, 1 - right, 2 - center),

opts - option flags (1 - draw guard whitespace, 2 - draw 2d code rectangles as circles, 4 - don't draw human readable text, 8 - don't draw the symbol, just calculate size),
clr_ink - symbol ink color in ARGB8888 format,
clr_paper - symbol paper color in ARGB8888 format (paper is not drawn if clr_paper=clr_ink),
clr_bg - background color in ARGB8888 format (background is not drawn if clr_bg=0),
data - data string to draw as barcode (in utf8),
data_prime - (optional) primary barcode data for composite codes

qr_start c - starts qr scanner to scan the barcode. c - continuous flag (if 1 - scanning is not stopped after code is scanned)

qr_check - reads code scanned by qr scanner

qr_check_wait t c - reads code scanned by qr scanner during the timeout or until successful code read. t - timeout in milliseconds, c - continuous flag (if 1 - successful code read does not break the operation)

qr_stop - stops qr scanner

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

List of commands supported (I2C)

PD.Screen family **i2c** address is **0x39** by default, speed up to **1000000 bps**

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.

Colors and layers

There are 2 layers and 1 buffer available for drawing.
Both layers are visible by default. Foregraund 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

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 preferrable as there is an imbedded support of JPG decoding.

Media uploading commands

use_crc;B - crc32 hash is used to check media uploading commands. disabled by default
B - 1 byte, 0 to disable crc, 1 to enable crc
response: ok\n

flashstate; - returns flash IC state
response: LSSIIABC
L - 1 byte, total number of bytes in the responce. must be 8
SS - 2 bytes, state read error code. 0 if no errors
II - 2 bytes, flash ID
A - 1 byte, status register 1
B - 1 byte, status register 2
C - 1 byte, status register 3

flashinfo; - requests information about available flash memory and crc state
response: LAAAASSSSB
L - 1 byte, total number of bytes in the responce. must be 10
AAAA - 4 bytes, flash start address. usually 0
SSSS - 4 bytes, flash size in bytes
B - 1 byte, 1 if crc32 check in media uploading commands is enabled

tstfl_start;AACC - starts flash IC write test. full test takes about 30 minutes
AA - 2 bytes, start sector address
CC - 2 bytes, number of sectors to test. there are 8192 sectors in total. 0 means test all sectors
response: ok\n - passed
er\n - failed because of invalid parameters

tstfl_status;B - returns flash IC test status
B - 1 byte. if 1 - test must be terminated
response: LSIIEE
L - 1 byte, total number of bytes in the responce. must be 6
S - 1 byte, test status. 1 - test is running
II - 2 bytes, sector index under test
EE - 2 bytes, number of errors

wrsect;(RRRR)AAAACCCCbbbbb... - erases flash sector, writes bytes to flash memory
RRRR - 4 bytes, crc32 hash of the folowing bytes (AAAA,CCCC,bbb...). Only if crc check is enabled
AAAA - 4 bytes, flash address. AAAA must be >= flash start address, AAAA + CCCC must be <= flash size in bytes returned by flashinfo.
If corresponding flash sector is erased if AAAA is divisible by 4096
CCCC - 4 bytes, number of bytes to write. CCCC must be <= 4096
bbb... - bytes to write
response: ok\n - passed
eX\n - failed. X - error code

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.

wrrsect;(RRRR)AAAACCCCbbbbb... - writes bytes to flash memory. Same as wrsect, but just writes, never erases
RRRR - 4 bytes, crc32 hash of the folowing bytes (AAAA,CCCC,bbb...). Only if crc check is enabled
AAAA - 4 bytes, flash address. AAAA must be >= flash start address, AAAA + CCCC must be <= flash size in bytes returned by flashinfo
CCCC - 4 bytes, number of bytes to write. CCCC must be <= 4096
bbb... - bytes to write
response: ok\n - passed
eX\n - failed. X - error code

wrsect/wrrsect can be called with chunks of any size.
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.

clrsects;(RRRR)AAAACCCC - erases flash memory
RRRR - 4 bytes, crc32 hash of the folowing bytes (AAAA,CCCC). Only if crc check is enabled
AAAA - 4 bytes, flash address. AAAA must be >= flash start address, AAAA + CCCC must be <= flash size in bytes returned by flashinfo
CCCC - 4 bytes, number of bytes to erase. Up to flash size in bytes
response: ok\n - passed
eX\n - failed. X - error code

There are 2 options to update flash:
- just use wrsect
- use clrsects to clear all required flash and then use wrrsect
Since wrsect erases flash by 4K bytes sectors and clrsect can erase up to 64K bytes sector and sector erase is the most time consuming operation,
clrsects/wrrsect results in a faster upload times

rdsect;AAAACCCC - reads bytes from flash memory
AAAA - 4 bytes, flash address
CCCC - 4 bytes, number of bytes to read
response: bbb...(RRRR)
bbb... - bytes from flash memory
RRRR - 4 bytes, crc32 hash of the preceding bytes (AAAA,CCCC,bbb...). Only if crc check is enabled

wrflusr;(RRRR)AACCbbbbb... - writes bytes to user space in flash memory.
Actually, same as wrsect, but 4 bytes shorter and executes an extra check to ensure that flash outside the user space is not affected.
User sector is erased when AA is 0, when AA is not 0 sector is not erased.
RRRR - 4 bytes, crc32 hash of the folowing bytes (AA,CC,bbb...). Only if crc check is enabled
AA - 4 bytes, flash address (from 0 to 4095)
CC - 4 bytes, number of bytes to write (from 1 to 4096)
bbb... - bytes to write
response: ok\n - passed
eX\n - failed. X - error code

rdflusr;AACC - reads bytes from user space in flash memory.
Actually, same as rdsect, but 4 bytes shorter and executes an extra check to ensure that flash outside the user space is not affected
AA - 4 bytes, flash address (from 0 to 4095)
CC - 4 bytes, number of bytes to read (from 1 to 4096)
response: bbb...(RRRR)
bbb... - bytes from flash memory
RRRR - 4 bytes, crc32 hash of the preceding bytes (AA,CC,bbb...). Only if crc check is enabled

read_ram;AAAACCCC - reads ram memory content. can be used to read screenshots
AAAA - 4 bytes, ram address
CCCC - 4 bytes, number of bytes to read
response: bbb...(RRRR)
bbb... - bytes from ram
RRRR - 4 bytes, crc32 hash of the preceding bytes (AAAA,CCCC,bbb...). Only if crc check is enabled

recache;B - re-reads media from flash. Must be used after flash update, can be used to ensure flash consistency.
May take a while, there is an option to run recache asynchronously
B - 1 byte. 0 - just re-read the header, do not check crc,
1 - check crc synchronously,
2 - check crc asynchronously,
3 - read result of previous asynchronous crc check
response: LRR
L - 1 byte, total number of bytes in the responce. must be 3
RR - 2 bytes. 0 - all ok,
0xFFF1 - asynchronous check is still running,
0xFFFF - header is empty of crc failed,
0x8000 - crc of media 0 failed,
0x8001 - crc of media 1 failed, etc.

show_pic_up;(RRRR)AAAACCbbbbb... - uploads fragment of image to RAM memory to show it with show_pic command
RRRR - 4 bytes, crc32 hash of the folowing bytes (AAAA,CC,bbb...). Only if crc check is enabled
AAAA - 4 bytes, picture fragment address. If you upload image by 4096 fragments, fragment 0 starts at address 0, fragment 1 starts at 4096, fragment 2 at 8192, etc.
CC - 2 bytes, number of bytes in the fragment
bbb... - bytes of the fragment
response: ok\n - passed
eX\n - failed. X - error code

show_pic;LXXYY - shows image previously uploaded to RAM with show_pic_up commands
L - 1 byte, layer index
XX - 2 bytes, x coordinate of image left-top corner
YY - 2 bytes, y coordinate of image left-top corner

Firmware updating commands (via I2C)

Firmware to update is provided as a **.bin file**.
Firmware update process consists of 3 steps:
- uploading firmware file to the board external flash,
- validating firmware uploaded to the board external flash
- actual firmware update

You must start with fwupd_prepare command to reserve the required space in the board external flash and provide an optional comment.
Then use fwupd_wrblock command to upload firmware file block by block. After you done with that, call fwupd_validate to validate
the uploaded firmware. If results of fwupd_validate satisty you, call fwupd_exec sequence to start the actual firmware update.
During the actual firmware update board will be irresponsible for several seconds.
fwupd_validate command is automatically called before the actual firmware update and error during the fwupd_validate process
terminates the actual firmware update. Therefore, you may consider to omit the fwupd_validate manual call.
fwupd_rdblock command is used to read firmware data blocks stored in the board external flash, in case you need it.
fwupd_rdinfo command can be used to get information about firmware currently stored in the board external flash and status of firware update.
tstfl_rdcurr command returns size and checksum of firmware currently used by the board. During the tstfl_rdcurr execution board
recalculates size and checksum of the firmware, it takes several seconds.

fwupd_prepare;(RRRR)SSSSLccc... - starts the firmware file upload to the board external flash.
Reserves the required space in the external flash, clears firmware update status,
stores firmware comment.
RRRR - 4 bytes, crc32 hash of the folowing bytes (SSSS,L,ccc...). Only if crc check is enabled
SSSS - 4 bytes, size of the firmware file, up to 2M
L - 1 byte, comment length, up to 230 characters
ccc... - comment itself. Comment does not affect the firware update in any way, it is up to user how to use it.
response: ok\n - passed
eX\n - failed. X - error code. Some error codes:
2 - could not reserve the required space in the external flash
3 - external flash write error
4 - not enough ram to complete the operation
A - command read timeout
B - command crc32 error

fwupd_wrblock;(RRRR)AAAASSSSbbb... - writes firmware file block to the board external flash.
RRRR - 4 bytes, crc32 hash of the folowing bytes (AAAA,SSSS,bbb...). Only if crc check is enabled
AAAA - 4 bytes, address of the firmware block. First block must have address 0000, adress of the following blocks must be increased sequentially.
SSSS - 4 bytes, size of the upcoming firmware block, up to 4096 bytes
bbb... - firmware block itself
response: ok\n - passed
eX\n - failed. X - error code. Some error codes:
2 - address is out of bounds
A - command read timeout
B - command crc32 error

fwupd_validate; - validates firmware uploaded to the board external flash.
response: LX(RRRR)
L - 1 byte, total number of bytes in the responce. must be 2 or 6
X - 1 byte, validation result:
0 - OK to update
2 - external flash is empty
3 - specified firmware size is out of the external flash bounds
4 - firmware crc check did not pass
5 - specified firmware size is too small
6 - not enough ram to complete the operation
7 - crc32 and size of the firmware is the same as crc32 and size of what is currently used by the board

fwupd_exec;(RRRR)CCCC - starts the actual firmware update. To start the actual firmware update you must sequentially call fwupd_exec 3 times
with the correct command.
Commands are 0x965ADB24, 0x5A96BD42, 0x965AD24B
I.e. you must call fwupd_exec;965ADB24, fwupd_exec;5A96BD42, fwupd_exec;965AD24B sequentially
RRRR - 4 bytes, crc32 hash of the folowing bytes (SSSS). Only if crc check is enabled
SSSS - 4 bytes, command. one of 0x965ADB24, 0x5A96BD42, 0x965AD24B
response: ok\n - passed
eX\n - failed. X - error code. Some error codes:
2 - external flash is empty
3 - specified firmware size is out of the external flash bounds
4 - firmware crc check did not pass
5 - specified firmware size is too small
6 - not enough ram to complete the operation
B - command crc32 error

fwupd_rdinfo; - reads info of the firmware stored in the board external flash.
response: LSSSSCCCCLccc...FF(RRRR)
L - 1 byte, total number of bytes in the responce
SSSS - 4 bytes, firmware size
CCCC - 4 bytes, firmware crc32 calculated on the validation stage
L - 1 byte, comment length
ccc... - L bytes, comment itself
FF - 2 bytes, firmware update status flags. Flag is set if corresponding bit is 0. Flags are:
bit 0: Firmware upload started
bit 1: Upload of firmware blocks started
bit 2: Firmware validation started
bit 3: Firmware validation passed
bit 4: Firmware validation vailed
bit 5: Uploaded firmware and current firmware are the same
bit 6: Flashing started
bit 7: Crc check during flashing failed
bit 8: Crc check during flashing passed
bit 9: Wait for MCU flash ready timed out
bit 10: MCU flash unlock failed
bit 11: Failed to allocate RAM
bit 12: One or more mcu flash re-read did not pass
bit 13: Firmware update terminated because of too many mcu flash re-read fails
bit 14: Check of mcu flash crc did not pass
bit 15: Firmware update done
RRRR - 4 bytes, crc32 hash of the preceding bytes. Only if crc check is enabled

fwupd_rdblock;AAAABBBB - reads the firmware file block from the board external flash.
AAAA - 4 bytes, data block address
BBBB - 4 bytes, data block size, up to 4096
response: bbb...(RRRR)
bbb... - data block bytes
RRRR - 4 bytes, crc32 hash of the preceding bytes (AAAA,SSSS,bbb...). Only if crc check is enabled

tstfl_rdcurr; - calculates current mcu firmware size and checksum.
response: LCCCCSSSS(RRRR)
L - 1 byte, total number of bytes in the responce
CCCC - 4 bytes, crc32 of the mcu firmware
SSSS - 4 bytes, size of the mcu firmware
RRRR - 4 bytes, crc32 hash of the preceding bytes. Only if crc check is enabled

Flash structure:

--- Flash start, address 0x00001000 ---

CCCCCCCC - 4 bytes, maximum pictures number from ini file - N
SSSSSSSS - 4 bytes, number of used flash bytes = header size + sizes of all pictures
TSSSSSSS - 4 bytes, picture 0 index. bits 31-28 - picture type: 1 = BMP, 2 = JPG, 3 = PNG, 4 = FNT. bits 27-0 - picture size
TSSSSSSS - 4 bytes, picture 1 index. bits 31-28 - picture type: 1 = BMP, 2 = JPG, 3 = PNG, 4 = FNT. bits 27-0 - picture size
....
TSSSSSSS - 4 bytes, picture N index. bits 31-28 - picture type: 1 = BMP, 2 = JPG, 3 = PNG, 4 = FNT. bits 27-0 - picture size
RRRRRRRR - 4 bytes, header crc32 checksum, from Flash start address to picture N index address

bbbbb... - picture 0 content, picture size bytes, aligned to 4 bytes
RRRRRRRR - 4 bytes, picture 0 crc32 checksum
bbbbb... - picture 1 content, picture size bytes, aligned to 4 bytes
RRRRRRRR - 4 bytes, picture 1 crc32 checksum
....
bbbbb... - picture N content, picture size bytes, aligned to 4 bytes
RRRRRRRR - 4 bytes, picture N crc32 checksum

--- Flash end, address SSSSSSSS ---

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 - passed
eX\n - failed. X - error code

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 - passed
eX\n - failed. X - error code

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;LXXYYWWHHRRCCCC - 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;LXXYYWWHHRRCCCC - 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_tria;LX1Y1X2Y2X3Y3CC - 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;LFFXXYYWWHHCCCCBBBBSWrChCvAvAhCTTTTTTTT - 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;LFFXXYYWWHHPpPxPyAaCCCCSWrChCvAvAhCTTTTTTTT - 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 responce. must be 5
WW - 2 bytes, text rect width
HH - 2 bytes, text rect height

Sensors

get_temp; - returns board temperature
response: LTT
L - 1 byte, total number of bytes in the responce. 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 responce. 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 responce. 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 responce
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

Touch events and screenshot function

get_touch; - reads touch screen status
response: LSSXXYY
L - 1 byte, total number of bytes in the responce
SS - 2 bytes, touch state flags. Touch events occured since previous read.
bit0 - touch down,
bit1 - touch up,
bit2 - touch press (down for 500ms),
bit3 - touch click,
bit4 - touch double click,
bit5 - touch swipe up,
bit6 - touch swipe right,
bit7 - touch swipe down,
bit8 - touch swipe left,
bit9 - touch coordinates changed,
bit15 - touch state changed
XX - 2 bytes, last touch X position
YY - 2 bytes, last 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 responce
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 responce
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

Screen controls

tft_on; - turns tft on. just turns screen power on and starts ltdc on MCU, 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 turns screen power off and stops ltdc on MCU, 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, turns vcc_dcdc on
response: ok\n

screen_bkl_off; - turns backlight off, turns vcc_dcdc off
response: ok\n

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

fps_slow; - slows down screen refresh rate. Can be used to reduce screen update artifacts while ram is under high load
response: ok\n

fps_normal; - sets screen refresh rate to normal value
response: ok\n

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

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 responce. must be 2
S - 1 byte, status. 0 - not playing, 1 - playing

note_play;CNdNdNdNd... - plays notes
C - 1 byte, number of following bytes
Nd - 2 bytes, note itself. There must be C/2 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

QR barcodes scan and QR generation

qr_start;C - starts qr scanner to scan the barcode
C - 1 byte, continuous option to scan severalcodes in arow, if 1 - scanning is not stopped after code is scanned successfully
response: ok\n

qr_stop - stops qr scanner. qr scanner started in not continuous mode is stopped automatically after the successfull code scan
response: ok\n

qr_check - reads code scanned by qr scanner. Maximum code length is 1024 characters
response: LLd...
LL - 2 bytes, length of the upcoming code. if 0 - no valid code was skanned, yet
d... - LL bytes, code itself

qr_print;LSXXYYWWHHsAvAhfIIIIPPPPBBBBlld...
L - 1 byte, layer index
S - 1 byte, symbol type id
XX - 2 bytes, symbol rect left point
YY - 2 bytes, symbol rect top point
WW - 2 bytes, symbol rect width
HH - 2 bytes, symbol rect height
s - 1 byte, symbol scale. If 0 - maximum scale to fit the rect is used
Av - 1 byte, vertical alignment, 0 - top, 1 - bottom, 2 - center
Ah - 1 byte, horizontal alignment, 0 - left, 1 - right, 2 - center
f - 1 byte, drawing flags: 1 - draw guard whitespace, 2 - draw 2d code rectangles as circles, 4 - don't draw human readable text, 8 - don't draw the symbol, just calculate size
IIII - 4 bytes, symbol ink color in ARGB8888 format
PPPP - 4 bytes, symbol paper color in ARGB8888 format. Paper is not drawn if PPPP=IIII
BBBB - 4 bytes, background color in ARGB8888 format. Background is not drawn if BBBB=0
ll - 2 bytes, symbol data length
d... - ll bytes, symbol data itself. In utf8. For composite codes, where primary data is required, data must be in format 'data'+'\0'+'primary_data'. I.e. d... contains data and primary data separated with 0
response: EESWWHH
EE - 2 bytes, status code
S - 1 byte, actual symbol scale
WW - 2 bytes, actual symbol width
LL - 2 bytes, actual symbol height

Thermal Printer commands

Thermal printer commands

prnt_pwron; - turns printer power on (off by default)
response: ok\n

prnt_pwroff; - turns printer power off (default). Print commands are executed asynchronously, there should be a delay between last print command and prnt_pwroff
response: ok\n

prnt_txt;LLtxt... - prints text
LL - 2 bytes, text length
txt... - LL bytes, text itself
response: ok\n

prnt_lf; - prints the buffer content and set the paper feed as per line space, then adjusts print position to initial position at the next line
response: ok\n

prnt_cr; - adjusts print position to initial position of the same line
response: ok\n

prnt_feed_dots;N - prints the buffer content and paper feed N dots
N - 1 byte, number of dots
response: ok\n

prnt_feed_line;N - prints the contents in printing buffer and paper feed N lines
N - 1 byte, number of lines
response: ok\n

prnt_set_prntpos;NN - sets print position
NN - 2 bytes, number of dots from left side
response: ok\n

prnt_set_rspace;N - sets character right space
N - 1 byte, character right space
response: ok\n

prnt_set_lnspace;N - sets line space
N - 1 byte, line space dots
response: ok\n

prnt_set_xymvu;XY - sets horizontal and vertical movement units
X - 1 byte, horizontal movement unit = 25.4/X mm (1/X inch)
Y - 1 byte, vertical movement unit = 25.4/Y mm (1/Y inch)

response: ok\n

prnt_set_lnspdef; - sets line space to default (30 dots)
response: ok\n

prnt_set_chfnt;S - sets character font
S - 1 byte, 1 - small font, else - normal font
response: ok\n

prnt_set_chmeth;F - sets character method
F - 1 byte, flags. bit 0 - small font,
bit 1 - bold font,
bit 2 - double height,
bit 3 - double width,
bit 4 - underline
response: ok\n

prnt_set_chsz;F - sets character size
F - 1 byte, flags. bit 2 - double height,
bit 3 - double width
response: ok\n

prnt_set_whprnt;S - sets white printing
S - 1 byte, 1 - white printing mode is on
response: ok\n

prnt_set_uln;N - sets underline
N - 1 byte, 0 - no underline, 1 - 1 dot underline, 2 - 2 dots underline
response: ok\n

prnt_set_bold;S - sets bold
S - 1 byte, 1 - bold is on
response: ok\n

prnt_set_ovlp;S - sets overlapping
S - 1 byte, 1 - overlapping is on
response: ok\n

prnt_set_updn;S - sets character unside down printing
S - 1 byte, 1 - character upside down is on
response: ok\n

prnt_set_90rvlv;S - sets 90 degrees revolving
S - 1 byte, 1 - character 90 degrees revolving is on
response: ok\n

prnt_set_alksw;S - allows orbid key switch
S - 1 byte, 1 - forbid key switch
response: ok\n

prnt_set_lmrg;NN - sets left margin
NN - 2 bytes, left printing margin dots
response: ok\n

prnt_set_pos;NN - sets relative printing position
NN - 2 bytes, printing position relative to current position
response: ok\n

prnt_set_alg;A - sets print alignment
A - 1 byte, 0 - left, 1 - middle, 2 - right
response: ok\n

prnt_set_custch;S - allows user customized characters
S - 1 byte, 1 - allows customized characters
response: ok\n

prnt_def_custchs;SWABCbbb... - defines user customized characters
S - 1 byte, 2 - 6*12 characters, 3 - 12*24 characters
W - 1 byte, character width
A - 1 byte, character code from (from 32 to 126)
B - 1 byte, character code to (from 32 to 126)
C - 1 byte, number of following bytes
bbb... - C bytes, character bytes. Vertical lines, msb bit first
response: ok\n - passed

prnt_def_custch;SWACbbb... - defines user customized character
S - 1 byte, 2 - 6*12 characters, 3 - 12*24 characters
W - 1 byte, character width
A - 1 byte, character code (from 32 to 126)
C - 1 byte, number of following bytes
bbb... - C bytes, character bytes. Vertical lines, msb bit first
response: ok\n - passed

prnt_cncl_custch;C - cancels user customized character
C - 1 byte, character code (from 32 to 126)
response: ok\n - passed

prnt_set_chrset;n - selects international character set
n - 1 byte, charset code:
0 - U.S.A
1 - France
2 - Germany
3 - U.K
4 - Denmark I
5 - Sweden
6 - Italy
7 - Spain I
8 - Japan
9 - Norway
10 - Denmark II
11 - Spain II
12 - Latin America
13 - Korea
14 - Slovenia
15 - China
response: ok\n - passed

prnt_set_chrcode;n - selects character code
n - 1 byte, character code:
0 - CP437 [U.S.A., Standard Europe]
1 - KataKana
2 - CP850 [Multilingual]
3 - CP860 [Portuguese]
4 - CP863 [Canadian-French]
5 - CP865 [Nordic]
6 - WCP1251 [Cyrillic]
7 - CP866 Cyrilliec #2
8 - MIK [Cyrillic /Bulgarian]
9 - CP755 [East Europe, Latvian 2]
10 - Iran
15 - CP862 [Hebrew]
16 - WCP1252 Latin I
17 - WCP1253 [Greek]
18 - CP852 [Latina 2]
19 - CP858 Multilingual Latin I+Euro)
20 - Iran II
21 - Latvian
22 - CP864 [Arabic]
23 - ISO-8859-1 [West Europe]
24 - CP737 [Greek]
25 - WCP1257 [Baltic]
26 - Thai
27 - CP720[Arabic]
28 - CP855

29 - CP857[Turkish]
30 - WCP1250[Central Europe]
31 - CP775
32 - WCP1254[Turkish]
33 - WCP1255[Hebrew]
34 - WCP1256[Arabic]
35 - WCP1258[Vietnam]
36 - ISO-8859-2[Latin 2]
37 - ISO-8859-3[Latin 3]
38 - ISO-8859-4[Baltic]
39 - ISO-8859-5[Cyrillic]
40 - ISO-8859-6[Arabic]
41 - ISO-8859-7[Greek]
42 - ISO-8859-8[Hebrew]
43 - ISO-8859-9[Turkish]
44 - ISO-8859-15 [Latin 9]
45 - Thai2
46 - CP856
47 - Cp874
252 - CP932 SHIFT_JIS
253 - UNICODE UCS-2
254 - BIG5
255 - GBK

response: ok\n - passed

prnt_gr_vmodule;MWWCCbbb... - fills graphics vertical module data
M - 1 byte, mode: 0 - 8dots hor×2 ver×3, 1 - 8dots hor×1 ver×3, 32 - 24dots hor×2 ver×1, 33 - 24dots hor×1 ver×1
WW - 2 bytes, module width
CC - 2 bytes, number of following bytes, up to 1536
bbb... - CC bytes, module bytes. Vertical lines, msb bit first
response: ok\n - passed

prnt_gr_hmodule;MWHCCbbb... - prints graphics horizontal module data
M - 1 byte, mode: 0 - ver×1 hor×1, 1 - ver×2 hor×1, 2 - ver×1 hor×2, 3 - ver×2 hor×2
W - 1 byte, widths in bytes (8 dots)
H - 1 byte, height in dots
CC - 2 bytes, number of following bytes, up to 1536
bbb... - CC bytes, module bytes. Horizontal lines, msb bit first
response: ok\n - passed

prnt_gr_defbmp;WHCCbbb... - defines bitmap
W - 1 byte, width in dots
H - 1 byte, height in bytes (8 dots)
CC - 2 bytes, number of following bytes, up to 1536
bbb... - CC bytes, module bytes. Vertical lines, msb bit first
response: ok\n - passed

prnt_gr_prntbmp;S - prints defined bitmap
S - 1 byte, 1 - double width
response: ok\n - passed

prnt_gr_wbmp;WHCCbbb... - prints bitmap
W - 1 byte, width in bytes (8 dots)
H - 1 byte, height in dots
CC - 2 bytes, number of following bytes, up to 1536
bbb... - CC bytes, bitmap bytes. Horizontal lines, msb bit first
response: ok\n - passed

prnt_gr_msbbmp;HCCbbb...; - prints msb full width bitmap. width is always 48 bytes (384 dots)
H - 1 byte, number of bitmap lines
CC - 2 bytes, number of following bytes, up to 1536
bbb... - CC bytes, bitmap bytes. Horizontal lines, msb bit first
response: ok\n - passed

prnt_gr_lsbbmp;HCCbbb...; - prints lsb full width bitmap. width is always 48 bytes (384 dots)
H - 1 byte, number of bitmap lines
CC - 2 bytes, number of following bytes, up to 1536
bbb... - CC bytes, bitmap bytes. Horizontal lines, lsb bit first
response: ok\n - passed

prnt_tab; - prints horizontal tab
response: ok\n - passed

prnt_tab_posset;Cttt... - defines tab positions
C - 1 byte, number of following bytes
ttt... - C bytes, tab positions in character widths, up to 16
response: ok\n - passed

prnt_q_pstat; - queries paper status
response: S
S - 1 byte, 1 - paper near end

prnt_q_stat; - queries status
response: F
F - 1 byte, status flags:
bit 0 - no paper,
bit 1 - printer failure,
bit 2 - printer overheat

prnt_q_ver; - queries printer version
response: SSS...
SSS... - printer version string

prnt_q_man; - queries printer manufacturer
response: SSS...
SSS... - printer manufacturer string

prnt_q_name; - queries printer name
response: SSS...
SSS... - printer name string

prnt_q_serial; - queries printer serial
response: SSS...
SSS... - printer serial string

prnt_q_chmode; - queries printer chinamode
response: SSS...
SSS... - printer chinamode string

prnt_q_rt_prnt; - queries real time printer status
response: B
B - 1 byte, real time printer status

prnt_q_rt_offl; - queries real time offline status
response: B
B - 1 byte, real time offline status

prnt_q_rt_err; - queries real time error status
response: B
B - 1 byte, real time error status

prnt_q_rt_psens; - queries real time paper sensor status
response: B
B - 1 byte, real time paper sensor status

prnt_uplstat;S - allow status uploading automatically
S - 1 byte, 1 - allow
response: ok\n - passed

prnt_selftest; - prints printer self test page
response: ok\n - passed

prnt_reset; - resents printer
response: ok\n - passed

prnt_qr;SWWHHsfxxAlldd... - prints qr code
S - 1 byte, symbol type id
WW - 2 bytes, symbol rect width
HH - 2 bytes, symbol rect height
s - 1 byte, symbol scale. If 0 - maximum scale to fit the rect is used

f - 1 byte, drawing flags: 1 - draw guard whitespace, 2 - draw 2d code rectangles as circles, 4 - don't draw human readable text
xx - 2 bytes, horizontal printer offset
A - 1 byte, horizontal alignment, 0 - left, 1 - right, 2 - center
ll - 2 bytes, symbol data length
ddd... - ll bytes, symbol data itself. In utf8. For composite codes, where primary data is required, data must be in format 'data'+'\0'+'primary_data'. I.e. d... contains data and primary data separated with 0
response: EE
EE - 2 bytes, status code

prnt_layer;lXXYYWWHHxxF - prints screen layer
l - 1 byte, layer. 0 - background, 1 - foreground, 2 - buffer
XX - 2 bytes, layer rect left
YY - 2 bytes, layer rect top
WW - 2 bytes, layer rect width
HH - 2 bytes, layer rect height
xx - 2 bytes, horizontal printer offset
F - 1 byte, flags. bit 0 - half scale (prints 2x2 screen layer rect as 1 dot), bits 1,2 - horizontal alignment, 0 - left, 1 - right, 2 - center
response: ok\n - passed

Other commands

fwrev; - firmware revision
response: LDDDDDDDDDDD.BB
L - 1 byte, total number of bytes in the responce. must be 15
DDDDDDDDDDD - 11 bytes, build date in format 'MMM DD YYYY', ex: Sep 19 2024
. - 1 byte = '.'
BB - 2 bytes, build number byte in hex format

apiver; - api version
response: LAA
L - 1 byte, total number of bytes in the responce. must be 3
AA - 2 bytes, api version number - 1, 2, etc.

screen_info; - display information
response: LWWHH
L - 1 byte, total number of bytes in the responce. must be 5
WW - 2 bytes, screen width
HH - 2 bytes, screen height

get_tick; - reads board millisecond ticks
response: LTTTT
L - 1 byte, total number of bytes in the responce. must be 5
TTTT - 4 bytes, millisecond ticks number

cpu_reset; - resets CPU (STM32). Reset is executed right after response is transferred.
response: ok\n - passed

Fonts FNT format description

FNT file format info
B - one byte.
BB - two bytes.
BBBB - four bytes.
bbb.... - many bytes.

FN - format marker. FN file - 256-character font
BBBB - header size
B - first character number
B - last character number
B - default character number
B - line height
B - y-offset. not used
..... characters. all characters in a row from first to last
(
B - bitmap character width
B - bitmap character height
B - character width
B - x bitmap character coordinate
B - y bitmap character coordinate
BBBB - starting index in bitmap array
)
..... // end of the header
BBBB - bitmap array size
bbb.... bitmap array
FU - format marker. FU file - 16 bit font
BBBB - header size
BB - default character index
B - line height
B - y-offset. not used
..... characters. in order, but not necessarily consecutive
(
BB - character code
B - character bitmap width
B - character bitmap height
B - character width
B - character bitmap x coordinate
B - character bitmap y coordinate
BBBB - starting index in bitmap array
)
..... // end of the header
BBBB - bitmap array size
bbb.... bitmaps

QRDraw description

Supported symbols:

1	BARCODE_CODE11	Code 11
2	BARCODE_C25STANDARD *	Standard Code 2 of 5
3	BARCODE_C25INTER	Interleaved 2 of 5
4	BARCODE_C25IATA	Code 2 of 5 IATA
6	BARCODE_C25LOGIC	Code 2 of 5 Data Logic
7	BARCODE_C25IND	Code 2 of 5 Industrial
8	BARCODE_CODE39	Code 3 of 9 (Code 39)
9	BARCODE_EXCODE39	Extended Code 3 of 9 (Code 39+)
10	BARCODE_EAN8 †	EAN-8 (European Article Number) GTIN-8
11	BARCODE_EAN_2ADDON †	EAN/UPC 2-digit add-on (standalone)
12	BARCODE_EAN_5ADDON †	EAN/UPC 5-digit add-on (standalone)
15	BARCODE_EAN13 †	EAN-13 (European Article Number) GTIN-13
16	BARCODE_GS1_128 *	GS1-128 (UCC.EAN-128)
18	BARCODE_CODABAR	Codabar
20	BARCODE_CODE128	Code 128 (automatic Code Set switching)
21	BARCODE_DPLEIT	Deutsche Post Leitcode
22	BARCODE_DPIDENT	Deutsche Post Identcode
23	BARCODE_CODE16K	Code 16K
24	BARCODE_CODE49	Code 49
25	BARCODE_CODE93	Code 93
28	BARCODE_FLAT	Flattermarken
29	BARCODE_DBAR_OMN *	GS1 DataBar Omnidirectional (including GS1 DataBar Truncated)
30	BARCODE_DBAR_LTD *	GS1 DataBar Limited
31	BARCODE_DBAR_EXP *	GS1 DataBar Expanded
32	BARCODE_TELEPEN	Telepen Alpha
34	BARCODE_UPCA	UPC-A
35	BARCODE_UPCA_CHK	UPC-A with check digit
37	BARCODE_UPCE	UPC-E
38	BARCODE_UPCE_CHK	UPC-E with check digit
40	BARCODE_POSTNET	POSTNET
47	BARCODE_MSI_PLESSEY	MSI Plessey
49	BARCODE_FIM	FIM
50	BARCODE_LOGMARS	LOGMARS
51	BARCODE_PHARMA	Pharmacode One-Track
52	BARCODE_PZN	PZN
53	BARCODE_PHARMA_TWO	Pharmacode Two-Track
54	BARCODE_CEPNET	Brazilian CEPNet
55	BARCODE_PDF417	PDF417
56	BARCODE_PDF417COMP *	Compact PDF417 (Truncated PDF417)
58	BARCODE_QRCODE	QR Code
60	BARCODE_CODE128AB	Code 128 (Suppress Code Set C)
63	BARCODE_AUSPOST	Australia Post Standard Customer

66	BARCODE_AUSREPLY	Australia Post Reply Paid
67	BARCODE_AUSROUTE	Australia Post Routing
68	BARCODE_AUSDIRECT	Australia Post Redirection
69	BARCODE_ISBNX	ISBN (EAN-13 with verification stage)
70	BARCODE_RM4SCC	Royal Mail 4-State Customer Code (RM4SCC)
71	BARCODE_DATAMATRIX	Data Matrix (ECC 200)
72	BARCODE_EAN14	EAN-14
73	BARCODE_VIN	Vehicle Identification Number
74	BARCODE_CODABLOCKF	Codablock-F
75	BARCODE_NVE18	NVE-18 (SSCC-18)
76	BARCODE_JAPANPOST	Japanese Postal Code
77	BARCODE_KOREAPOST	Korea Post
79	BARCODE_DBAR_STK *	GS1 DataBar Stacked
80	BARCODE_DBAR_OMNSTK *	GS1 DataBar Stacked Omnidirectional
81	BARCODE_DBAR_EXPSTK *	GS1 DataBar Expanded Stacked
82	BARCODE_PLANET	PLANET
84	BARCODE_MICROPDF417	MicroPDF417
85	BARCODE_USPS_IMAIL *	USPS Intelligent Mail (OneCode)
86	BARCODE_PLESSEY	UK Plessey
87	BARCODE_TELEPEN_NUM	Telepen Numeric
89	BARCODE_ITF14	ITF-14
90	BARCODE_KIX	Dutch Post KIX Code
92	BARCODE_AZTEC	Aztec Code
93	BARCODE_DAFT	DAFT Code
96	BARCODE_DPD	DPD Code
97	BARCODE_MICROQR	Micro QR Code
98	BARCODE_HIBC_128	HIBC Code 128
99	BARCODE_HIBC_39	HIBC Code 39
102	BARCODE_HIBC_DM	HIBC Data Matrix
104	BARCODE_HIBC_QR	HIBC QR Code
106	BARCODE_HIBC_PDF	HIBC PDF417
108	BARCODE_HIBC_MICPDF	HIBC MicroPDF417
110	BARCODE_HIBC_BLOCKF	HIBC Codablock-F
112	BARCODE_HIBC_AZTEC	HIBC Aztec Code
115	BARCODE_DOTCODE	DotCode
116	BARCODE_HANXIN	Han Xin (Chinese Sensible) Code
119	BARCODE_MAILMARK_2D	Royal Mail 2D Mailmark (CMDM) (Data Matrix)
121	BARCODE_MAILMARK_4S	Royal Mail 4-State Mailmark
128	BARCODE_AZRUNE	Aztec Runes
129	BARCODE_CODE32	Code 32
131	BARCODE_GS1_128_CC *	GS1 Composite Symbol with GS1-128 linear component
132	BARCODE_DBAR_OMN_CC *	GS1 Composite Symbol with GS1 DataBar Omnidirectional linear component
133	BARCODE_DBAR_LTD_CC *	GS1 Composite Symbol with GS1 DataBar Limited linear component

134	BARCODE_DBAR_EXP_CC *	GS1 Composite Symbol with GS1 DataBar Expanded linear component
135	BARCODE_UPCA_CC	GS1 Composite Symbol with UPC-A linear component
136	BARCODE_UPCE_CC	GS1 Composite Symbol with UPC-E linear component
137	BARCODE_DBAR_STK_CC *	GS1 Composite Symbol with GS1 DataBar Stacked component
138	BARCODE_DBAR_OMNSTK_CC *	GS1 Composite Symbol with GS1 DataBar Stacked Omnidirectional component
139	BARCODE_DBAR_EXPSTK_CC *	GS1 Composite Symbol with GS1 DataBar Expanded Stacked component
140	BARCODE_CHANNEL	Channel Code
141	BARCODE_CODEONE	Code One
142	BARCODE_GRIDMATRIX	Grid Matrix
143	BARCODE_UPNQR	UPNQR (Univerzalnega Plačilnega Naloga QR Code)
144	BARCODE_ULTRA	Ultracode
145	BARCODE_RMQR	Rectangular Micro QR Code (rMQR)
146	BARCODE_BC412	IBM BC412 (SEMI T1-95)
147	BARCODE_DXFILMEDGE	DX Film Edge Barcode
148	BARCODE_EAN8_CC †	GS1 Composite symbol with EAN-8 linear component
149	BARCODE_EAN13_CC †	GS1 Composite symbol with EAN-13 linear component

Error numbers:

warnings:
1 Human Readable Text was truncated (max 199 bytes)
2 Invalid option given but overridden by Zint
3 Automatic ECI inserted by Zint
4 Symbol created not compliant with standards
5 Generic warning
errors:
6 Generic error
7 Input data wrong length
8 Input data incorrect
9 Input check digit incorrect
10 Incorrect option given
11 Internal error (should not happen)
12 Error opening output file
13 Memory allocation (malloc) failure
14 Error writing to output file
15 Error counterpart of warning if WARN_FAIL_ALL set (see below)
16 Error counterpart of warning if WARN_FAIL_ALL set
17 Error counterpart of warning if WARN_FAIL_ALL set

Error codes:

0 Internal error
1 Input too long
2 Input too long, requires too many codewords
3 Input too long, requires too many codewords (maximum 144)
4 Input too long (maximum 3264 codewords)
5 Input too long for Version T (maximum 90)
6 Input too long for Version T (maximum 38 codewords)
7 Input too long, requires too many codewords (maximum 1313)
8 Input too long, requires too many codewords (maximum 1558)
9 Input too long for ECC level
10 Input too long for ECC
11 Input too long (maximum 35)
12 Input too long (maximum 415)
13 Input too long (maximum 26)
14 Input too long (maximum 90)
15 Input too long for the Version
16 Input too long for Version S (maximum 18)
17 Input too long (maximum 18)
18 Input too long (maximum 1558 codewords)
19 Input too long (maximum 3)
20 Input too long for Reader Initialisation (maximum 22 layers)
21 Input too long for number of columns
22 Input too long (maximum 3116)
23 Input too long, requires too many codewords (maximum 928)
24 Input too long, requires too many codewords (maximum 1558)
25 Input too long (maximum 6)
26 Input too long (maximum 8)
27 Input too long (maximum 125)
28 Input too long (maximum 67)
29 Input too long (maximum 92)
30 Input too long (maximum 14)
31 Input too long (maximum 21)
32 Input too long (maximum 69)
33 Input too long (maximum 136)
34 Input too long (maximum 7)
35 Input too long (maximum 13)
36 Input too long (maximum 11)
37 Input too long (maximum 30)
38 Input too long (maximum 68)
39 Input too long (maximum 86)
40 Input too long (maximum 86 symbol characters)
41 Input too long (maximum 123)
42 Input too long (maximum 123 symbol characters)
43 Input too long (maximum 140)
44 Input too long (maximum 103)

45 Input too long for HIBC LIC (maximum 110)
46 Input too long (maximum 19)
47 Input too long (maximum 12)
48 Input too long, requires too many symbol characters (maximum 2726)
49 Input too long (2D component)
50 Input too long (maximum 126 codewords)
51 Input too long, requires too many symbol characters (maximum 20)
52 Input too long (maximum 38)
53 Input too long (maximum 1)
54 Input too long (maximum 50)
55 Input too long (maximum 576)
56 Input too long (maximum 128)
57 Input too long (maximum 20)
58 Input too long (maximum 78 symbol characters)
59 Input too long (maximum 81)
60 Input too long (maximum 49 codewords)
61 Input too long (maximum 48 characters)
62 Input too long for Version 8 (maximum 51)
63 Input too long for Version 30 (maximum 70)
64 Input too long (maximum 32)
65 Input too long (maximum 10)
66 Input too short (minimum 14)
67 Input too short (minimum 7)
68 Input too short (minimum 3)
69 Input too short (minimum 28)
70 Input too short (minimum 32)
71 Invalid character in postcode in Primary Message
72 Invalid character in input for ECI 4
73 Invalid character in input (alphanumerics and space only)
74 Invalid Supply Chain ID (digits only)
75 Invalid Item ID (digits only)
76 Invalid postcode
77 Invalid character in input (Version S encodes digits only)
78 Invalid Structured Append ID (digits only)
79 Invalid character in input for Version M1 (digits only)
80 Invalid character in input for Version M2 (digits, A-Z, space and "$%*+-./:" only)
81 Invalid character in input (digits only)
82 Invalid PZN, check digit is "10"
83 Invalid check digit
84 Invalid primary, must be NUL-terminated
85 Invalid character in input (alphanumerics only, excluding "O")
86 Invalid character in input (digits and "ABCDEF" only)
87 Invalid character in Compressed Field data (digits only)
88 Invalid character in General Field data
89 Invalid character in input, extended ASCII not allowed
90 Invalid character in input (digits and "X" only)
91 Invalid position of "X" in Telepen data

92 Invalid character in input (alphanumerics, space and "-.$/+%%" only)
93 Invalid character in input (alphanumerics only)
94 Invalid DPD identification tag (first character), ASCII values 32 to 126 only
95 Invalid character in input (digits and "-" only)
96 Invalid check digit version (1 or 2 only)
97 Invalid character in input
98 Invalid character in input (cannot contain "A", "B", "C" or "D")
99 Invalid AI at in input (AI too long)
100 Invalid AI at in input (AI too short)
101 Invalid AI at in input (non-numeric characters in AI)
102 Invalid character in input (alphanumerics only, excluding "IOQ")
103 Invalid check digit (position 9)
104 Invalid EAN-8 check digit
105 Invalid ISBN (must begin with "978" or "979")
106 Invalid ISBN check digit
107 Invalid character in input, "X" not allowed in ISBN-13
108 Invalid character in input (digits and + or space only)
109 Invalid character in input (digits, "X" and "+" or space only)
110 Invalid add-on data (one + or space only)
111 Invalid add-on data (digits only)
112 Invalid character in input, "X" allowed in last position only
113 Invalid data length for AI
114 Invalid AI
115 Invalid character in input (ISO/IEC 8859-1 only)
116 Invalid input mode - reset to DATA_MODE
117 Invalid character in escape sequence in input
118 Invalid character for escape sequence in input (hexadecimal only)
119 Invalid UTF-8 in input
120 Invalid rotation angle
121 Invalid character in input (alphanumerics only after first)
122 Invalid character in input ("A", "B", "C", "D" or "E" only)
123 Invalid character in input ("D", "A", "F" and "T" only)
124 Invalid character in input (alphanumerics and "-" only)
125 Invalid character in input (2D component)
126 Invalid mode (CC-C only valid with GS1-128 linear component)
127 Invalid character in input (alphanumerics, space and "#" only)
128 Invalid character in input (digits only for FCC 59 length 16)
129 Invalid character in input (digits only for FCC 62 length 23)
130 Invalid character in DPID (first 8 characters) (digits only)
131 Invalid length for tracking code (20 characters required)
132 Invalid length for ZIP code (5, 9 or 11 characters required)
133 Invalid Version (8, 10 or 30 only)
134 Invalid character in input (alphanumerics and space only in first 45)
135 Invalid Information Type ID (cannot be space)
136 Invalid Version ID ("1" only)
137 Invalid Class (cannot be space)
138 Invalid Supply Chain ID (7 digits only)

139 Invalid Item ID (8 digits only)
140 Invalid Destination Post Code plus DPS
141 Invalid Service Type ("0" to "6" only)
142 Invalid Return to Sender Post Code
143 Invalid Reserved field (must be spaces only)
144 Invalid character in Service Indictor (first 2 characters) (alphabetic only)
145 Invalid character in Serial Number (middle characters) (digits only)
146 Invalid character in Country Code (last 2 characters) (alphabetic only)
147 Invalid Service Indicator (first character should not be any of "JKSTW")
148 Invalid first character, DX code should start with a number
149 Invalid character DX info (digits and "-" character only)
150 Insufficient memory for Reed-Solomon log tables
151 Insufficient memory for placement array
152 Insufficient memory for mode buffers
153 Insufficient memory for vector rectangle
154 Insufficient memory for vector hexagon
155 Insufficient memory for vector circle
156 Insufficient memory for vector string
157 Insufficient memory for vector string text
158 Insufficient memory for vector header
159 Insufficient memory for BMP row buffer
160 Insufficient memory for pixel buffer
161 Insufficient memory for raw segs buffer
162 Insufficient memory for raw text source buffer
163 Input value out of range (0 to 255)
164 Input length wrong (should be 8 digits)
165 Input length wrong (17 characters required)
166 DPD input length wrong (27 or 28 characters required)
167 Input length wrong (9, 10, or 13 characters required)
168 Input length wrong (2, 5, 7, 8, 12 or 13 characters required)
169 Input length wrong (7, 12 or 13 characters required)
170 Input length wrong (linear component)
171 Input length wrong (8, 13, 16, 18 or 23 characters required)
172 Input length wrong (12 or 13 characters required
173 Input value out of range (3 to 131070)
174 Input out of range (4 to 64570080)
175 Input value out of range (0 to 1999999999999)
176 Input value out of range
177 Input add-on length too long
178 Input EAN length too long
179 Input segments NULL
180 No input data (segment 0 empty)
181 No input data
182 Input length is not standard (should be 11 or 13 digits)
183 Input length is not standard (should be 5, 9 or 11 digits)
184 Too many input segments (maximum 256)
185 Input segment source NULL

186 No composite data (2D component)
187 Input segment empty
188 Cannot use Reader Initialisation in GS1 mode
189 Older Data Matrix standards are no longer supported
190 SCM prefix can not be used with ECI (ECI must be ASCII compatible)
191 Primary Message empty
192 Structured Append ID not available for MaxiCode
193 Mode out of range (2 to 6)
194 Primary Message length wrong (7 to 15 characters required)
195 Non-numeric country code or service class in Primary Message
196 Non-numeric postcode in Primary Message
197 SCM prefix version out of range (1 to 100)
198 Structured Append count out of range (2 to 8)
199 Structured Append index out of range
200 Error correction level Q requires Version M4
201 Error correction level H not available
202 UPNQR does not support GS1 data
203 Version M1 supports error correction level L only
204 Error correction level L not available in rMQR
205 Error correction level Q not available in rMQR
206 Version out of range (1 to 38)
207 Format (1st character) out of range (0 to 4)
208 Version ID (2nd character) out of range (1 to 4)
209 Class (3rd character) out of range (0 to 9 and A to E)
210 ECI code out of range (0 to 811799)
211 Revision out of range (1 or 2 only)
212 Structured Append ID too long (5 digit maximum)
213 Structured Append ID value out of range (1 to 80088)
214 Converted to GB 18030 but no ECI specified
215 Number of columns is out of range (5 to 200)
216 Resulting symbol size (HxW) is too large (maximum 200x200)
217 Resulting symbol is too large (maximum 200)
218 Resulting symbol is too small (minimum 5)
219 Structured Append count out of range (2 to 16)
220 Structured Append ID too long (3 digit maximum)
221 Structured Append ID value out of range (0 to 255)
222 Symbol ignored for Version S
223 ECI ignored for GS1 mode
224 Version out of range (1 to 10)
225 Error correction level out of range (1 to 4)
226 Version out of range (1 to 36)
227 Structured Append ID cannot contain spaces
228 Number of ECC codewords 3 at minimum
229 Structured Append count out of range (2 to 99999)
230 Structured Append ID too long (30 digit maximum)
231 Structured Append ID triplet value out of range (000 to 899)
232 Number of rows increased

233 Number of columns increased
234 Structured Append out of range (2 to 16)
235 Structured Append ID out of range (0 to 255)
236 Using ECI in GS1 mode not supported by GS1 standards
237 Using Structured Append in GS1 mode not supported by GS1 standards
238 Converted to Shift JIS but no ECI specified
239 Structured Append ID too long (6 digit maximum)
240 Structured Append ID1 and ID2 out of range (001 to 254)
241 Structured Append ID1 out of range (001 to 254)
242 Structured Append ID2 out of range (001 to 254)
243 Structured Append count out of range (2 to 35)
244 Structured Append ID not available for DotCode
245 Cannot have Structured Append and Reader Initialisation at the same time
246 Structured Append not available for Version S
247 Cannot have Structured Append and GS1 mode at the same time
248 Structured Append count out of range (2 to 128)
249 Structured Append ID not available for Code One
250 Multiple segments not supported for Version S
251 Escape '\^' only valid for Code 128 in extra escape mode
252 Symbol ECI must match segment zero ECI
253 Symbology does not support multiple segments
254 Height out of range (0 to 2000)
255 Guard bar descent out of range (0 to 50)
256 Whitespace width out of range (0 to 100)
257 Whitespace height out of range (0 to 100)
258 Border width out of range (0 to 100)
259 Too many stacked symbols
260 GS1 mode not supported for multiple segments
261 Number of ECC codewords less than 5% + 3 of data codewords
262 Version out of range for Reader Initialisation symbols (maximum 26)
263 Structured Append count out of range (2 to 26)
264 Output size too large for file size field of BMP header
265 Size not within the minimum/maximum ranges
266 No rows
267 Could not open BMP output file
268 Incomplete write of BMP output
269 Failure on closing BMP output file
270 Does not begin with A, B, C or D
271 Does not end with A, B, C or D
272 Processed input too long (maximum 77)
273 Height not compliant with standards
274 Corrupt Unicode data
275 Unicode sequences of more than 3 bytes not supported
276 Human Readable Text truncated
277 Extended ASCII characters are not supported by GS1
278 NUL characters not permitted in GS1 mode
279 Control characters are not supported by GS1

280 DEL characters are not supported by GS1
281 Data does not start with an AI
282 Malformed AI in input (brackets don't match)
283 Found nested brackets in input
284 For this UPC-E zero suppression, 3rd character cannot be "0", "1" or "2"
285 For this UPC-E zero suppression, 4th character cannot be "0"
286 For this UPC-E zero suppression, 5th character cannot be "0"
287 EAN-8 with add-on is non-standard
288 Empty data field in input
289 AI error
290 Incomplete escape sequence in input
291 Value of escape sequence in input out of range
292 Incomplete escape character in input
293 Unrecognised escape character in input
294 Codabar 18 not supported
295 UPCD1 not supported
296 Symbology out of range
297 Symbology does not support ECI switching
298 ECI code out of range (0 to 999999, excluding 1, 2, 14 and 19)
299 Scale out of range (0.01 to 200)
300 Dot size out of range (0.01 to 20)
301 Text gap out of range (-5 to 10)
302 Selected symbology does not support GS1 mode
303 Selected symbology cannot be rendered as dots
304 No primary (linear component)
305 2D component input too long, requires %d characters (maximum 2990)
306 Error correction level out of range (0 to 8)
307 Number of columns out of range (1 to 30)
308 Number of rows out of range (3 to 90)
309 Number of columns out of range (1 to 4), ignoring
310 Columns x rows value out of range (1 to 928)
311 Cannot specify rows for MicroPDF417
312 Number of rows out of range (0 to 44)
313 Number of columns out of range (9 to 67)
314 Minimum number of rows out of range (2 to 16)
315 Minimum number of rows out of range (2 to 8)
316 Composite type changed
317 Barcode Identifier (second character) out of range (0 to 4)
318 DPD relabel input length wrong (27 characters required)
319 Malformed RGB colour (6 or 8 characters only)
320 Malformed RGB colour (hexadecimal only)
321 Malformed CMYK colour (4 decimal numbers, comma-separated)
322 Malformed CMYK colour (3 digit maximum per number)
323 Malformed CMYK colour C (decimal 0 to 100 only)
324 Malformed CMYK colour M (decimal 0 to 100 only)
325 Malformed CMYK colour Y (decimal 0 to 100 only)
326 Malformed CMYK colour K (decimal 0 to 100 only)

327 Destination Country Code (last 3 characters) should be numeric
328 Service Code (characters 6-4 from end) should be numeric
329 Last 10 characters of Tracking Number (characters 16-7 from end) should be numeric
330 Non-standard Service Indicator (first 2 characters)
331 Country Code (last two characters) is not ISO 3166-1
332 Ignoring first digit which is not "0" or "1"
333 Escaped parentheses only valid in GS1 mode with GS1 parentheses flag
334 DX information too long
335 Frame number part too long
336 Frame number is invalid (expected digits, optionally followed by a single "A")
337 The "-" is used to separate DX parts 1 and 2, and should be used no more than once
338 Wrong format for DX parts 1 and 2 (expected format: NNN-NN, digits);
339 DX part 1 out of range (1 to 127)
340 DX part 2 out of range (0 to 15)
341 DX number is incorrect; expected 4 digits (DX extract) or 6 digits (DX full)
342 DX extract out of range (16 to 2047)
343 Frame number indicator "/", but frame number is empty
344 Frame number out of range (0 to 63)

