RocDisplay
Technical Description

Doku Version 1

January 2017

RocDisplay contents

The project

What's it all about?

The starting point of all considerations was a software-driven platform displays in scale H0

Recently available and affordable OLED display with 96x16 pixels and 26,3x8mm size to meet the demand for sense of scale very well.

Fig. The red frame is about the display size at a scale of 1:87 compared to a ÖBB platform represents.

We have, of course, envisaged as users of the open source model railway control software RocRail the development of a control of the display by RocRail. The interface electronics / controller was designed as an assembly for RocNetNode.

Disclaimer

This is a purely private hobby project. The RocDisplay is not a toy. Everyone is responsible for the operation and use. It is expressly disclaim any liability for hardware or software, or for damages and consequential damages in every conceivable way. As noted in the source code of each project for his private purposes is allowed to use and modify. Commercial use of the software, hardware or parts thereof is prohibited without the permission of the authors.

In short: as soon as anyone for anything any from this project in return in some form and some way somehow wants or demands, he must have it if he has some time to get the approval for the authors.

Why bother? The basic idea for this project was Peter Ploiner. He is the intellectual father of this project. In the source code parts were used by people who have released the code under GPL on the Internet (see copyright).

Since this project was born as a hobby it is to others that something will also have to give much pleasure and not wishing to capitalize on someone.

Source code and hardware has been created by Walter Sax. The copyright, except for the parts of the code in the section above copyright lies with the author.

If you like the RocDisplay you may invite us for a coffee or a beer, we should run into each other ;)

Translated from German by www.onlinedoctranslator.com, with some modifications by Dagnall. Should information in this version differ from the Original, it is my fault (Dagnall) and please use the German Version as “Master“. Cheers.

components

The RocDisplay consists of three components.

1. display controller
2. Display backplane
3. display

The controller can be connected directly to the RocNetNode. , When the host program, the firmware of the controller integrated to be updated on the bootloader.

The display backplane is a small circuit board to which the display, the necessary SMD components for the voltage converter in the display and the connections to the controller are provided.

The display is soldered directly to the backplane. At a backplane 2 displays can be soldered.

the controller

The controller is connected directly to the RocNetNode via RJ12 or RJ45 plug. The power supply is only possible via the RocNetNode.

The controller works with 5V. A higher input voltage can destroy the controller and displays!

Up to four displays can be connected to a controller.

It is recommended at this more controllers when possible split over several RocNetNodes, or connecting it to a RocNetNode which has no feedback tasks for trains. (Division into process critical and –non critical- tasks).

controller data

designation	min	Type	Max
VDD	4.5V	5.0V	5.5V
I2C speed	-	100kHz	400kHz
I2C Address	0x50		0x5F
VDisplay Logic OUT	1.65V	2.8V	3.5
VDisplay VDB OUT		3.3V	4.2V
Command to Display			300ms
Command Length	0		128
Char Encoding		ISO 8859-1 / UTF-8 *	
Fonts		4	
font size			6x8
Char Range	32	-	255
bitmaps		20	
bitmap size			20x8
text templates		20	
template Length	0		30
display Columns	1		2
display Lines		2	
analog Clock		R / L / none	

* UTF-8 is not 100% special character of the German language work.

controller structure

The controller operates with an AT MEGA 328 PU microcontroller. This is due to security reasons RocNetNode before to protect high voltage via an I2C level shifter to the I2C bus driver (P82B714) via the hardware connected I2C bus. The maximum I2C bus frequency is 400kHz. The pullup resistors for the I2C bus from the RocNetNode are not provided on the display controller. a 3.3V linear voltage regulator with the associated capacitors is available for the bus voltage between bus driver and level shifter. Thus, the RocNetNode is protected from excessive bus voltage even when accidental HIGH output from the microcontroller. The I2C bus address can be set via jumpers. With a plugged jumper means a logical 0th

Table of possible jumper settings:

J1	J2	J3	J4	I2C address Hex	I2C Address Dec
X	X	X	X	0x50	80
X	X	X	-	0x51	81
X	X	-	X	0x52	82
X	X	-	-	0x53	83
X	-	X	X	0x54	84
X	-	X	-	0x55	85
X	-	-	X	0x56	86
X	-	-	-	0x57	87
-	X	X	X	0x58	88
-	X	X	-	0x59	89
-	X	-	X	0x5A	90
-	X	-	-	0x5B	91
-	-	X	X	0x5C	92
-	-	X	-	0x5D	93
-	-	-	X	0x5E	94
-	-	-	-	0x5F	95

set ... Offenx ... -

This address is also used by the boot loader. To release the RocNetNode as quickly as possible the I2C bus is cached every command in a buffer and not processed until after the stop bit. When receiving and sending data, the red LED for the duration of the transmission light up. If the Red LED light up constantly, the last message has not been received in full. (Missing stop bit).

For storage of fonts, bitmaps, text templates and display settings 8KB NV-RAM is available. This is a ferro-electric non-volatile RAM memory. This type of Ram opposite EEprom's and Flash memory that the access times in a range of about 100 ns are the advantage. This corresponds to the access time of D-Ram's. The number of read and write cycles by a factor of 104 higher than EEPROM and 106 higher than in flash memory. The only disadvantage is the reading of one bit has the same physical effect as a franchise of one bit.

For the four displays an I2C switch is available which also serves as a Level Shifter of 5V to 3.3V logic voltage of the displays. For the I2C switch a soldering jumper is available to possibly change the bus address. By default, the jumper with a thin conductor GND is connected. Before the jumper must be soldered to the printed conductor 3V be broken to GND with a scalpel. (But without firmware modification it makes no sense) - (Is intended for a possible extension for additional support (of larger displays 128x64))

each is a voltage divider available to limit the 5V microcontroller to 3V for the reset lines of the display and the I2C switch.

Two status LEDs are available for optical control of the controller status. The red LED lights up when I2C bus are transmitted to the hardware data (in bootloader mode).

In boot loader mode, the yellow LED blinks. In program mode, the yellow LED lights up when everything is OK. A flicker means data activity on software I2C bus. If the yellow light is off then there is a communications failure between controllers and displays. This occurs for. As to when three displays are connected to the controller, but four connected displays are set in the settings.

four connector with the following pin assignments are available for connecting the display:

Pin code	colour	function	Comment / Color
1	None	+ 5V	**NOT CONNECTED**
2	RED	+ 3.3V	
3	BLACK	Ground (GND)	
4	BLUE	SDA	
5	YELLOW	SCL	
6	WHITE	Reset (RES)	

The colors are marked on the display connector with short colored shrink sleeves.
The cables to the displays must be kept as short as possible. A shielded cable would be the optimum, but is very difficult to be soldered to the display backplanes.

The display plugs are cut off to one side, as they can be removed from the controller easier. The trimmed side Pin 1

The pins for Rx, Tx UART interface and the neighboring GPIO pins are designed to solder points 2,54mm, if someone wants to rewrite the firmware of I2C UART communication. For the development they were as Serial Debug Output quite enjoyable.

The pullup resistors for the displays and the software I2C bus are rather high dimensioned with 2k2. At long cable to the display, the pull-up's can be possibly reduced to approximately 1k. In the test it (individual veins) at 40 cm display connection cable been no problems.

The controller board (Size: 50x80 mm):

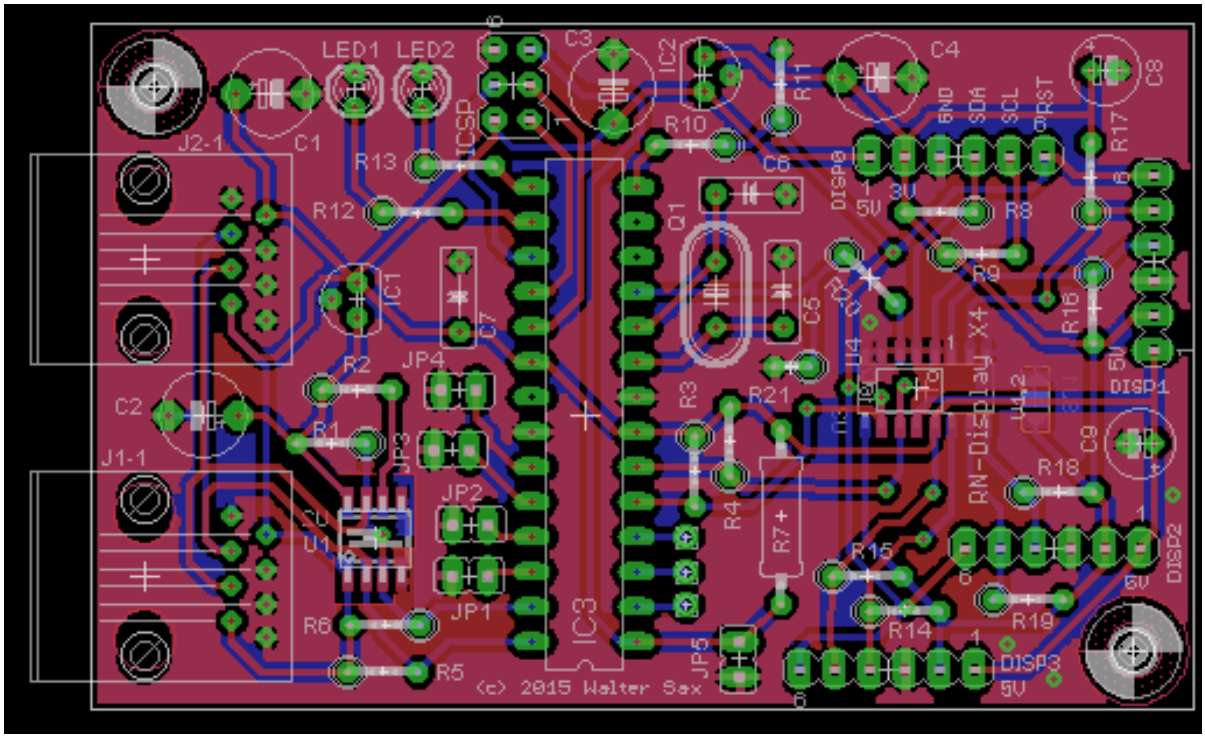

Illustration 1 display controller

BOM for the controller:

component	value	comment
C1, C3, C8	1ïF / 16V	
C2, C4, C9	2.2uF / 16V	
C5, C6	22pF ceramic	
C7	100nF ceramic	
Disp0 - 3	6-pin connector 2,54	Or pin layout
IC1, IC2	No. LP2950ACZ-third 3	
IC3	AT Mega 328P - PU	with base
ICSP	6 Pol 2,54 Pinheader	
J1, J2	RJ45 or RJ12	
JP1, JP2, JP3, JP4, JP5	jumper	JP5 optional
led1	Led 3mm Yellow	
Led2	Led 5mm Red	
Q1	16Mhz Quartz HC49U-V	
R1, R2, R11, R21	3k3	
R3, R4, R5, R6	4k7	
R7	47k	
R8, R9, R10, R14, R15, R16, R17, R18, R19, R20	2k2	
R12, R13	1k	
U1	P82B715TD	SO8
U2	PCA9517	SO8
U3	PCA9546	SO16
U4	FM24-64SG	SO8

Display backplane and display

On the display backplane contains the necessary capacitors for each display.

The display is soldered directly to the backplane. A backplane can accommodate two displays. The connections are separate for each display. The size of the backplane is 10,2x25mm so that it fits between two displays.

When soldering the backplane, the displays should be soldered as all last, otherwise it is almost impossible to solder the leads.

Be careful when soldering display, the lead member of the displays is at V1. 2 bent sharply by an offset of the display to prevent at bilateral mounting.

The display backplane board (Dimensions 25 mm x10,2):

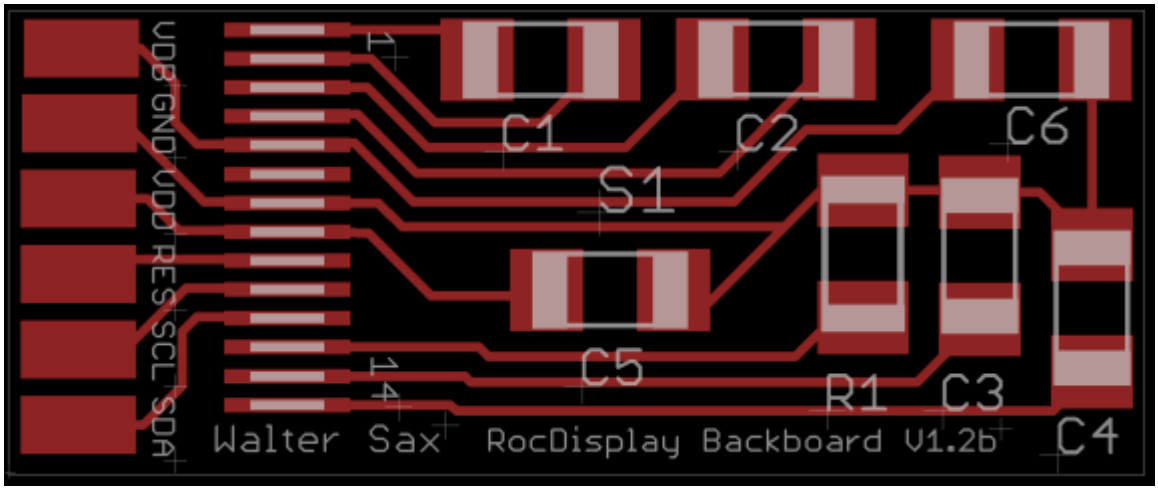

Illustration 2 display board

The board is double-sided, so it can be placed between 2 displays in the middle. The connections are as follows:

Display backplane	controller	Pin the plug and controller
VDD	+ 3.3V logic	2
VDB	+ 3.3V voltage display	2
GND	Ground (GND)	3
RES	RESET	6
SCL	SCL	5
SDA	SDA	4

The bill of the display backplane:

component	value	comment
C1, C2	1īF / 16V X5R	SMD 1206 or 0803
C3	4,7µF / 16V X7R	SMD 1206 X7R
C4	2.2uF / 16V	SMD 1206
C5, C6	1īF / 16V	SMD 1206
R1	430k	SMD 1206
display	OLED 96x16 I2C 14Pin	SSD1306 chip

When equipping it is easier if the display is soldered as a last resort after the connecting cables.

Assembly of the controller and the display

Please this chapter before assembly accurately reading.
The controller 4 has SMD components on the board, on each side two. In the SMD components, a circuit is printed on the PCB layout at pin. 1 The SMD components, this can also be a circle deepened in the housing or flattened on one side the cabinet with a facet. When loading is to ensure that either the circles are at the same position, or the side where the housing is flattened with a facet on the side where is printed on the circuit board of the circuit.

Since SMD components are included, the assembly is recommended only if very good soldering skills and a basic knowledge of electronics is available. After the soldering of the components, the board must be cleaned in any case of the flux residues. When the display board, it is the terminal lug of the display and the circuit board be cleaned with a glass fiber pin of advantage if, in addition, before and after brazing.

The component names are made up of letters and numbers. Where the numbers is a sequential numbering of the component type. The letters have the following meanings:

- U - SMD IC
- IC - *bedrahtetes* IC or voltage regulator
- C - capacitor
- R - resistance
- Q - quartz
- Led - led
- JP - jumper
- ICSP - Pin header for programming the microcontroller
- J - RJ12 or RJ45 socket
- Disp - Connectors for displays

needed tools

For the smooth and successful assembly of the following tools should be available.

- Electronic soldering iron with a very fine tip.
- Elektroniklötzinn max. diameter 0.5
- Braid
- flux
- glass fiber post
- SMD tweezers
- Magnifier with Light
- Scalpel or enamelled copper wire puller wire 0.2mm CuL
- Small side cutter
- Double-sided tape

Assembling the controller

The controller board before the solder on the components cleaned thoroughly so that it is grease and dust free. It is recommended with the SMD components to start, otherwise no place for soldering is provided by the other components. Thereafter, clean the upper side of the board from the solder residue.

The components of the series of the following recommendations to solder:

episode	component	value
1	U1	P82B715TD
2	U4	FM24-64SG
3	U2	PCA9517
4	U3	PCA9546
5	Remove soldering residues above	
6	R7	47k
7	IC3	base
8th	Q1	16Mhz quartz HC49U-V
9	LED1, LED2	3mm LED red and yellow
10	C7	100nF ceramic
11	C5, C6	22pF ceramic
12	ICSP	6 pin pin-header
13	JP1, JP4, JP2, JP3, (JP5)	Jumper 2 pol
14	IC1, IC2	No. LP2950ACZ-third 3
15	R2, R1, R11, R21	3k3
16	R6, R5, R3, R4	4k7
17	R13, R12	1k
18	R8, R9, R10, R16, R17, R20, R18, R19, R15, R14	2k2
19	Disp0, Disp1, disp2, Disp3	6-pin display connector
20	C8	1īF / 16V
21	C9	2.2uF / 16V
22	C1, C3	1īF / 16V
23	C2, C4	2.2uF / 16V
24	J1, J2	RJ12 / RJ45 socket
25	IC3	ATMEGA328P plug into the socket (notch to the PCB edge addressed without components)
26	jumper bridges	Set jumpers according to the desired address after addressing Table

The order of this proposal is such that it goes from the lowest and components to the higher parts, the ranking of the components with the same values is chosen so that the loading is facilitated by the placement.
The capacitors C1, C2, C3, C4 and C8 and C9 is the correct polarity to Be! LED1 and LED2 are to be fitted with the longer connection direction boards middle.
The plug for the display must be positioned so that the cut side with Pin1. This is also the site where the component name is printed on the board.

A note for soldering SMD components:

- A tin solder pad on the circuit board easily.
- place the component with a SMD tweezers carefully and pay attention to the correct position.
- Solder pin with the previously tinned solder pad and possibly then correct the orientation of the component.
- The pins on the opposite side soldering with little solder
- The pins of the first page with little solder solder and resolder the first pin again.

On the Internet you can also find some instructions for SMD soldering. With version to smear the solder pads with flux and then take the soldering iron from one side to the other, I have personally made any such good experiences in this SMD size. This procedure is z. As for the tinning of the display terminal lug suitable.

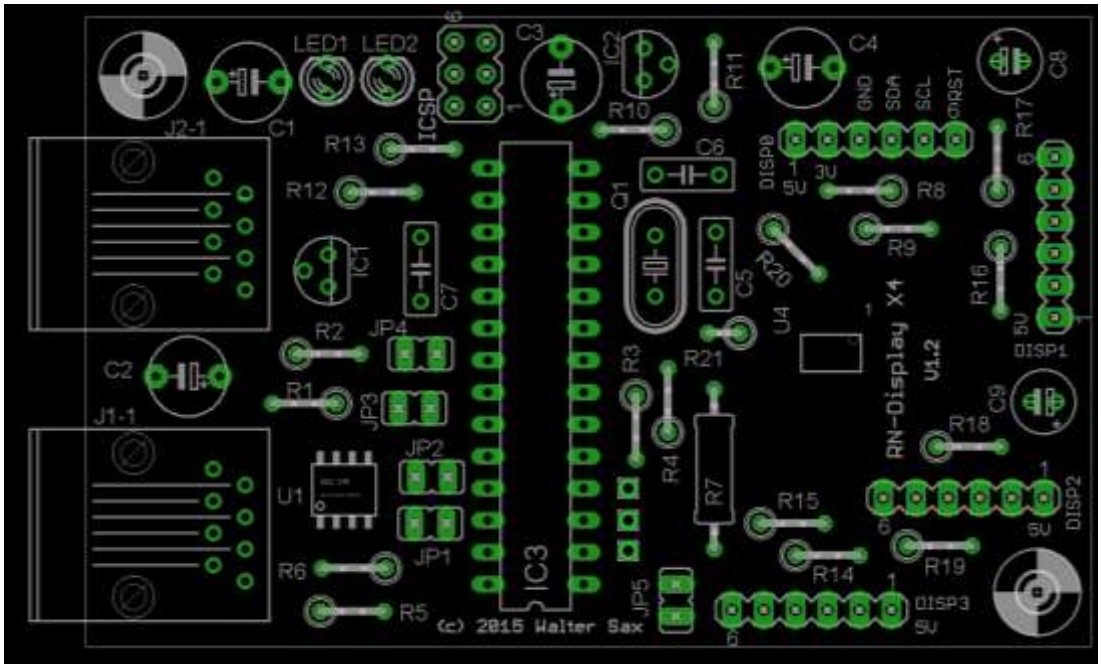

Illustration 3 Silkscreen controller top

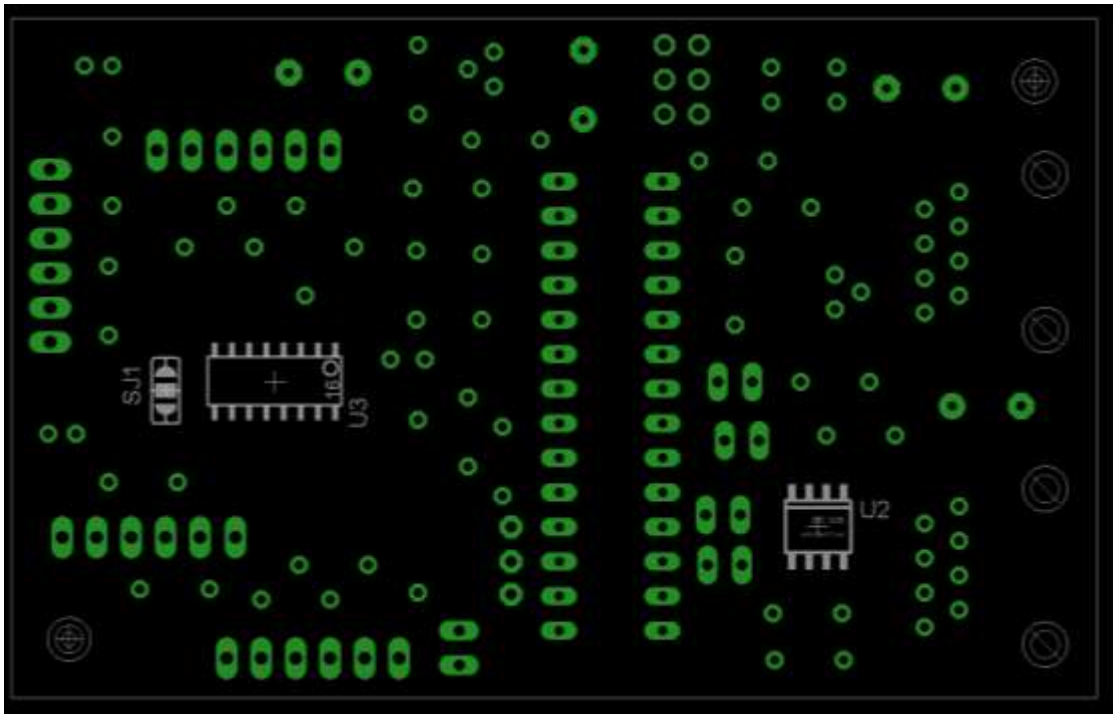

Illustration 4 Silkscreen controller underside

For U2 the silkscreen of the board is unfortunately incomplete !! The side with the facet needs to boards look middle.

Assembling the display board

This is only recommended to use a magnifying glass with good lighting. The distances between the solder pads on the lead member of the displays have each other only 0.3mm distance!

Before the components to solder the solder points on the board clean with a fiberglass pin, this facilitates the soldering.

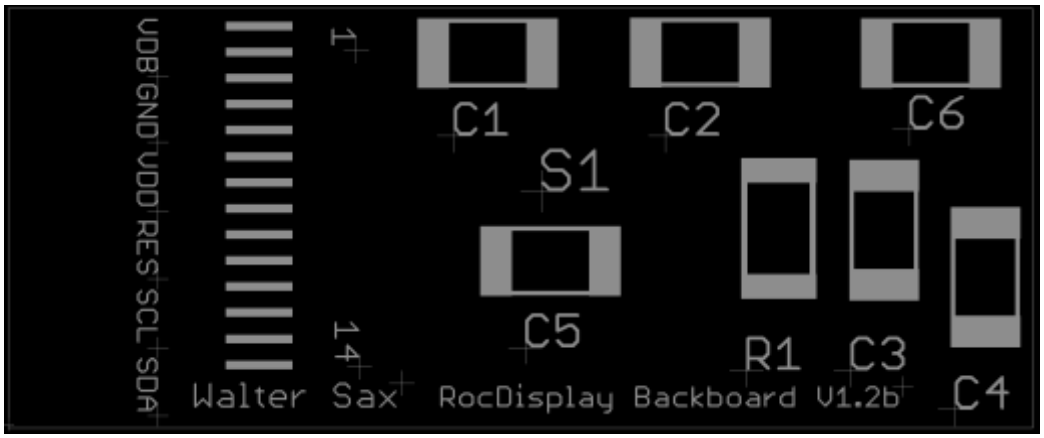

Illustration 5 Silkscreen the display board Page 1

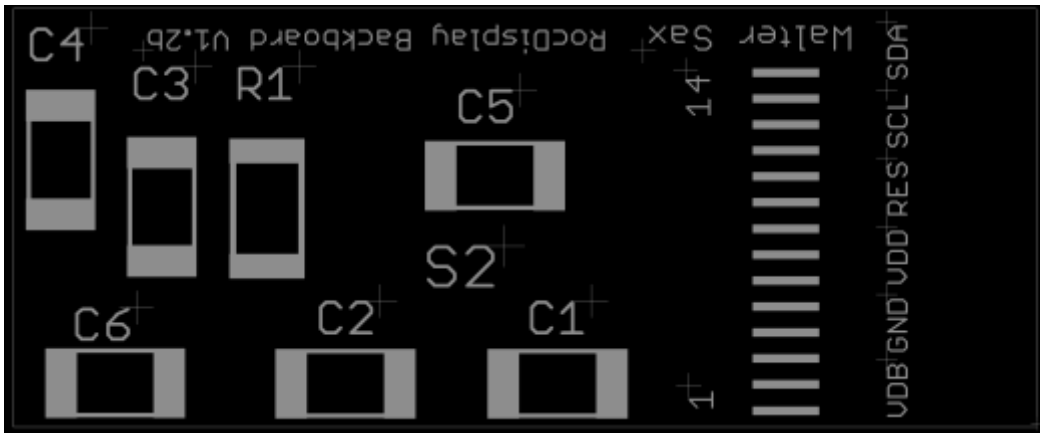

Illustration 6 Silkscreen the display board Page 2

Recommendation to solder the board according to the following steps:

episode	component	value
1	C1, C2	1ïF 16V X5R SMD 1206 or 0803
2	C6	1ïF 16V SMD 1206
3	R1	430k SMD1206
4	C3	4,7µF 16V SMD1206
5	C4	2.2uF 16V SMD 1206
6	C5	1ïF 16V SMD 1206

After that, the solder pads of the lead terminals well tinning.

Wires 2mm down to scratch the paint of copper paint on one side with a scalpel or Lackabzieher and solder the wires to the solder pads.

Then carefully remove the soldering residues. The solder pads of the display not tin on the board, but only once cleaned with a glass fiber post.

After that tin solder tail of the display as following:

1. Clean the bare contacts with a glass fiber post.
2. Coat the contacts with flux.
3. Take up with the soldering tip solder and drive through the contacts so that they are tinned.
4. Visual check under the microscope whether there is a thin layer of tin. This can be seen that the solder is evenly distributed. Remove the given case with a solder wick the excess solder.
5. as well as just now tin the second page.
6. Clean both sides with a glass fiber post. Important because this can cause contact problems

The most challenging part is the soldering of the display on the board.

Place the display face down and place the flexible connecting tab on the already soldered connecting wires, so that the contacts of the board and the display superimposed accurately and in alignment. With one finger the terminal lug press on one side, keep up with the remaining fingers, the display into position and secure it with the soldering iron on one side 2-3 contacts. hold down the remaining contacts per contact with the SMD tweezers and solder with a soldering iron without additional solder. Afterwards, wet the Braid with flux and thus free the just soldered contacts from excess solder. Thereby the contact tab will be better soldered flat on the board. In most cases it is necessary to solder contact 1 with very little solder to. remove with a glass fiber post the solder residue on the contacts. Thereafter, on the one hand to check with a continuity tester and very fine probe tips to the passage of the contacts to the display board and on the other hand against each other to short-circuit.

Now the connection lugs of the plug contacts are tin generous, since the copper magnet wire can not be crimped. The wire ends about 4mm wide clean up the paint and push the colored heat shrink tubing to the corresponding wires. The bare wire ends to the plug contacts soldered, pinch off from the connecting strip along bend the small crimp connector (Crimpfahnen), slide the heat shrink tubing on the check contact and shrinking with hot air. The individual contacts in the connector housing until the contact insert engages.

gluing a small piece of double sided tape on the SMD components of the display board and the display fold it is attached with the adhesive tape on the components.

If a two-sided display is desired, each listed here step is also carried out similarly on the other side boards.

The display on the controller plug and the controller can be put into operation.


Kurzzug
Tulln Stadt
23:15

List of commands

All commands are sent as ASCII characters. The fonts are stored in accordance with ISO 8859-1. Characters that are not available between {and} as text on the display. An order for an action to begin with a curly brace and will be completed with a curved clip. There are a few commands that may stand between the brackets only alone, it is mentioned in the list of commands. The numbering of the display rows and columns starts with 0. The commands always apply to the currently active display. Switches the active display within a chain of command, all the following instructions for the display apply to which was changed in the chain of command. All commands or writing texts have a change to the display result in an internal display buffer. To see the changes on the display it must be written with a command to the display. A Clear the display buffer sets the column to 0 and the line also to 0. Thus, the sequence of instructions after deleting for output left starting at the top for the set of row and column omitted. A reading of display settings is possible by the controller ago, but currently outside RocRail. The description is used here only for completeness of documentation. A reading of display settings is possible by the controller ago, but currently outside RocRail. The description is used here only for completeness of documentation. A reading of display settings is possible by the controller ago, but currently outside RocRail. The description is used here only for completeness of documentation. Read commands are generally to be sent as byte values. The read register are given in the explanation of the commands. In search of the values only one command can be sent and before the next time you send the answer is to be seen. The answers to the read commands are sent as byte values, unless explicitly stated that the response is sent as ASCII. All commands are sensitive case. The maximum text length including commands that may be sent from RocRail from the display controller 110 characters.

command table

command name	code	parameter	Read / Write	Single Command	example	comment
Connected displays	Z	0th , 15	R / W	No	{Z3}	Parameters as bit value of the display Display0 = Bit0 Display1 = bit1 Display2 = Bit2 Display3 = Bit3
display Sync	V	0th , 15	R / W	No	{V2}	Parameters as bit value of the display Display0 = Bit0 Display1 = bit1 Display2 = Bit2 Display3 = Bit3
Invert Display / Clock	I	0,1,128,129	R / W	No	{I129}	Inverts the entire display or just the clock. 0th , normal 1 . , Disp. Invert 128 pm Invert 129 Disp. Invert Invert + AM
rotation	R	0,1,2,3	R / W	No	{R2}	Alignment of the text display: 0th , 0 °. 1 , 90 °. 2 , 180 °. 3 , 270 ° H = 2
brightnes	H	0th , 255	R / W	No	{H255}	Sets the brightness of the display. Changes <30 are almost imperceptible.
Show Clock / Clock Side	S	0,1, L, R	R / W	No	{S1SR} {S0} {S1} {SL}	The clock display: 0 -Clock Hidden 1 -Clock Display R -Clock Right L -Clock Left
Show Departure Column	D	0.1	R / W	No	{D1}	Display of departure times column: 0 , Hides 1 , Displayed
Departure ColumnWidth	W	0th , 95	R / W	No	{W20}	Width of the departure times column in pixels
Active display	A	0th , 3	R / W	No	{A0}	Sets the specified display number

						as active. All following instructions are for the specified display.
Write bitmap	M	No; Data	R / W	Yes	{M0; [21 bytes 2char HEX]}	Overrides the bitmap specified under number with the data from Data. No = 0th , 19Data = 21 bytes with hex 2digit; as seperator. Byte 0 = width Byte first , 20 bitmap array
Write font	N	No; CharCode; Data	R / W	Yes	{N0;4D;[7 bytes 2char Hex]}	Overrides the character of the font indicated in point with the data from data specified with CharCode. No = 0th , 3 CharCode = 20th , FF Data = 7 bytes with hex 2digit; as a separator. Byte 0 = width Byte first , 6 Char array
display ON	U	0,1,2,3, A	W	No	{UA}	the specified display comes on. A parameter switches all Connected displays a sequence with a previous reset of the display = Startup.
display OFF	O	0,1,2,3, A	W	No	{O1}	Turns the specified parameter from the display. A parameter switches all connected from the display. This command with parameter A should before each of the Cut supply voltage is sent as the last command.
Setup instructions for Synced displays	K	0.1	W	No	{K1}	After this command setup changes are applied to the synchronized under DisplaySync displays. This setting is not saved and automatically reset even after emptying the display buffer.
ASCII encoding	? EA		W	Yes	{?} EA	Setting the ASCII character set ISO *
UTF-8 encoding	? EU		W	Yes	{?} EU	Setting the UTF-8 encoding *
Display firmware version	? V		W	Yes	{?} V	Displays the firmware version on the displays *
Set Clock	C	Hour: Minute	W	Yes	{00:25}	Sets the time. Updates the clock display on all connected displays with displayed PM.
Clear display	E		W	No	{E}	Clears the internal display buffer. No update of the display. For deletion of update is {EP} put (Erase and Punto display)
Put on Display	P		W	No	{P}	Updates the display with the data from the display buffer.
Set Line	L	0.1	W	No	{L0}	Sets the X cursor on the start position of the specified parameter in the line. Deleted but not a pre-existing content.
Set Column	T	0.1	W	No	{T1}	Changes in the specified column: 0 = train target column 1 = Departure times column If the departure times column is hidden and is still described, this text is not displayed.
Set font	F	0,1,2,3	W	No	{F2}	Sets the font for the following text. Will be saved for the row and column permanently until a new value is set.
Draw bitmap	B	0th , 19	W	No	{B10}	Set in the current row in the current column in the current X position of the specified under the number in the parameter bit map

						in the display buffer.
Blink	J	0,1,2, S	W	NO	{J1}xxx{J2} yyy {J0} {JS2}	J1 of the start of the flashing area is set. If J2 is specified, the following text alternative text. J0 sets the flash range the end. In the flashing area fonts and bitmaps can be set. A new line automatically exits the flashing area. JS stores the flashing time.
Save Txt Template	GS	NR 0.. 19 \ text \. , max 30char	W	Yes	{GS0 \ test text \}	Saves the text between the backslash character from a template under no.
Load Template Txt	GL	0th , 19	R / W	No	{GL0}	Load the text of the template into the display buffer. In search for answer on I2C Bus: {QG # [n]}
Set Csr X	X	0th , 95	W	No	{X20}	the x position set for the next edition.
Set Csr Y	Y	0th , 15	W	No	{Y4}	Sets the y position for the next edition.
Character {	{		W	No	{{}	To Darzu put the sign on the display {{another sign must be sent in command mode.
Start Bootloader	2	BOOT	W	Yes	{2BOOT}	Switches to running the boot loader
firmware reset	! *!	1 . , 7	W	Yes	{!*!7}	Bit 0th , 1 = reset Display Settings Bit. 1 , 1 = reset Font & BMP Bit. 2 , 1 = reset Txt Templates After the firmware reset the program will automatically restart.

* From firmware version 1. 1. 0

All commands with single Command = no can be combined within a command sequence.
E.G. {EB0F3} 2 2 {F2} Line 0{L1F0}Line 1{L0T1F2}Col 1 L0{L1F2}Col1 L1{P}

This example is the following sequence of commands (commands in []):
[Erase Display Buffer] [Set bitmap 0 in the display buffer] [Use Font 3] [write **2 2**] [Use Font 2] [write **Line 0**] [To line 1] [Use Font 0] [Write **Line 1**] [To line 0] [Use column 1] [Use font 2] [write **Col 1 L0**] [Use line 1] [Use font 2] [write **Col1 L1**] [Show me now on display]

Column 0 1
Line 0 bitmap0 22 Line 0 Col 1 L0
Line 1 Line 1 Col 1 L1

detailed explanation of the commands

The commands are sent as ASCII Chars. (The figures -., For example, 10 = 0x31, 0x30)

Displays Connected {} Znn

This command tells the controller with the display occupied ports. The number is in the range 0 (no port assigned) to 15 (all ports assigned) valid.

The ports correspond to the bit's number.

port	value
0	1
1	2
2	4
3	8th

The number is derived from the sum of the connected port values.

The settings are stored in non-volatile memory and must be set only once.

Read with a set of register 0x01 to display statement.

The response length is 1 byte with a value from 0 to 15 (non-ASCII)

Display Sync {} Vnn

This command sets the with the output to be synchronized with the display of the currently active display. Because this setting can be set individually for each screen to watch before sending out is that the desired display is selected.

This setting is similar to the Connected displays. The number represents the value of the set bits 0 to 3 for the synchronization. The setting is stored in the nonvolatile memory.

Read with a set of register 0x02 to display statement.

The response length is 1 byte with a value from 0 to 15 (non-ASCII).

Invert Display / Clock {} Innn

This command determines whether the display is inverted or just the analog clock is inverted.

command	Output
{I0}	Clock and display not inverted
{I1}	Clock and display inverted
{I128}	AM inverted display not inverted
{I129}	AM inverted, inverted display

The setting is stored in non-volatile memory for each display. It is important to ensure that the corresponding display has been selected as active.

The command [Setup commands for Synced Displays], the value can be posited for all Synchronized displays.

Read with a set of register 0x03 to display statement.

The response length is 1 byte (not ASCII).

Rotation {Rn}

This command sets the rotation of the display of the display.

There are permissible values from 0 to the third

value	rotation
0	0 ° (not for Double Display).
1	90 ° (makes no sense)
2	180 ° (standard setting)
3	270 ° (makes no sense)

The setting is stored in non-volatile memory for each display. It is sure to Ensure that the corresponding display has been selected as active.

Read with a set of register 0x04 to display statement.

The response length is 1 byte (not ASCII).

Brightnes {} Hnnn

This command sets the brightness of the display.

Valid values are from 0 to 255

The setting is stored in non-volatile memory for each display.
It is important to ensure that the corresponding display has been selected as active.

Read with a set of register 0x05 to display statement.

The response length is 1 byte (not ASCII).

Show Clock / Clock Side {Sn}

This command sets the clock display, whether displayed or hidden and whether left or right.

Valid values are 0 or 1, and L and R

command	Display
{S0}	Hides the clock
{S1}	Displays the clock
{SR}	am right
{SL}	am left

The setting is stored in non-volatile memory for each display. It is important to ensure that the corresponding display has been selected as active.

The command [Setup commands for Synced Displays], the value can be posited for all Synchronized displays. This can be useful if more space is needed for a presentation.

Show Clock: Readable with setting Register 0x06 to display statement. Clock Side: Can be read out with setting Register 0x07 to display statement.

The response length is 1 byte.

Show Departure Column {Dn}

This command sets the display of a so-called departure times column.

When writing display text the respective column must first be selected by the instruction {Tn}. If the text is longer than the column be as it is cut off at the last displayable position. A flashing across both columns is not possible. Each column and each row in a column can have its own font setting.

The setting is stored in non-volatile memory for each display. It is important to ensure that the corresponding display has been set as active.

The command [Setup commands for Synced Displays], the value can be posited for all Synchronized displays. This can be useful if more space is needed for a presentation.

Read with a set of register 0x08 to display statement.

The response length is 1 byte (not ASCII).

Departure ColumnWidth Wnn {}

This command sets the width in pixels of the departure times column.

The command [Setup commands for Synced Displays], the value can be posited for all Synchronized displays. This can be useful if more space is needed for a presentation.

The departure times column is always aligned to the right. If the clock right appears as the departure times will be moved column to the width of the clock to the left.

The setting is stored in non-volatile memory for each display. It is important to ensure that the corresponding display has been set as active.

Read with a set of register 0x09 to display statement.

The response length is 1 byte (not ASCII).

Active Display {An}

With this command the active display can be set. By protocol definition between RocNetNode and display controller that is automatically included in every message. However, to be held at a transmission an update on several screens with different data, so the display can be changed with this command.

This setting is not saved. The display is 0 start after the active display.

Read with setting register 0x0A to display indication, but by the protocol definition does not really make sense.

The response length is 1 byte (not ASCII).

Write bitmap {Mn0; n1; nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd;}

With this command, a bitmap can be overwritten with new data in the nonvolatile memory.

The bitmap size is limited to 20x8 pixels.

The first parameter n0 is the number of bitmaps 0-19.

The second parameter is the bit width in bytes as 2 digit hexadecimal number in ASCII.

The following 20 parameters nd are the bit map data as a 2 digit hexadecimal number in ASCII.

The parameters must be separated by a semicolon. (Also after the last parameter)

How to determine the bitmap data is explained later specifically.

By the hardware bug of no Clockstreching the I2C bus does not allow read.

Write Font {NN0; n1; n2; nd nd nd nd nd nd;}

With this command a font characters with neunen data in the non-volatile memory can be overwritten.

The character size is limited to 6x8 pixels.

The first parameter n0 is the font number from 0 to 3

The second parameter n1 is the code for the character as a 2 digit hexadecimal number in ASCII 00 to FF

The third parameter n2 is the character width in bytes as 2 digit hexadecimal number in ASCII 00-06

The following 6 parameters nd are the font data as a 2 digit hexadecimal number in ASCII.

The parameters must be separated by a semicolon. (Also after the last parameter)

How to determine the font data is explained later specifically.

By the hardware bug of no Clockstreching the I2C bus does not allow read.

Display ON {Un}

This command switches the display of the specified displays again. When an A is given instead of the display number, this command is executed for all connected displays.

This command is executed after starting for all connected displays.

Display OFF {On}

This command switches the display of the specified displays. An A specified instead of the display number, the command is performed on all connected displays.

This command with parameter A should be performed before removing the power supply.

Setup instructions for Synced displays a {K1} / or {K0}

This command sets the following setup instructions are executed for the Synchronized displays the setting. An erase the display buffer, this command is reset, but not the altered setup settings.

{K1} activates the setup for Synchronized displays. {K0} disables the setup for Synchronized display (default setting).

Set Clock {Chh: mm}

The command the clock display is updated.

hh is the hour of 0 to 24mm the minute of 0 to 59ES are values with leading zeros or without permitted. For example, 5 or 05

The time is not automatically updated. The time displayed remains until the next Set Clock command. After starting the clock on 14:00 is set.

This command may not be combined with other commands between the braces. After processing the command all connected displays are updated with the contents of the display buffer.

Clear display {E}

This command deletes the contents of the display buffer.

This command and the command is blinking, setup commands for Synced displays reset. A Active flashing display is reset by this command.

Furthermore, the line 0 and column 0 is set as active.

Put on display {P}

This command displays the contents of the display buffer of the currently active displays on the display.

Without sending this command, the display is not updated. This command ends a not completed by {J0} Blink definition.

Set Line {Ln}

This command indicates the current line to be described. The following text or bitmap instructions are written to the specified line. The line numbering starts. 0

The 96x16 display has 2 lines - Valid values 0 or 1

This command also an unfinished Blink instruction is terminated, as a flashing assignment over multiple lines is not possible.

The Current column is not changed by this command.

Set Column {Tn}

With this command, you can switch between the two columns if the departures column is active.

Column 0 is the destination column, column 1 if the departure times activated column.

Is with deactivated departure times column this chosen and described in the data is discarded and not displayed.

Set font {footnote}

This command sets the font to be used for output.
Values from 0 to 3 are allowed.

Font 0 is a 5x7 pixel font with a pixel spacing between the characters (large letters). Font 1 is a 5x6 pixel font with a pixel spacing between the characters (mean writing). Font 2 is a 4x6 pixel font with a pixel spacing between the characters (small letters). Font 3 is a 4x5 pixel inverse font without spacing between characters, they must be explicitly set with Space.

The selected font is stored per display for each row and column in the nonvolatile memory. It uses the selected font to last in describing the row and column.

Draw Bitmap {Bnn}

This command writes the bitmap under the nn number specified in the currently active row and column at current position in the display buffer.

Valid values are from 0 to 19

Blink - start {J1} / alternate {J2} / end {J0}

With this command, an area can be defined as flashing. There is a flashing area per display in a row and a column possible. In the flashing area, the font can also be specifically defined with {Fn}.

The text or the bitmap between {J1} and {J2} is the flashing area. The text or the bitmap between {J2} and {J0} and is the alternative display. the display of the flash range and the alternatives display alternate depending on the flashing time setting. The actual width of the area is taken from the larger area.

{J2} Is not specified or contains no representing characters so the flashing area and a hidden alternately.

A Clear the display buffer is flashing back. The flashing areas are written to the external RAM and read from there. This allows the display Conrtoller depending flashing area is charged a little more.

Set Time Blink {JSn}

This command displays the flashing interval for the currently active display is set.

Valid values are 1 to 9

Table of flashing times to the values:

value	Time
1	0.5s
2	1s
3	1.5s
4	2s
5	2.5s
6	3s
7	3.5s
8th	4s
9	4.5s

This setting is stored in nonvolatile memory per display.

Read with setting register 0x0B for display indication.

Response length is 1 byte (not ASCII).

Save Txt template {GSnn\TEXT\}

This command saves a text template in nonvolatile memory for later retrieval.

Nn the memory number is specified. The text between the backslash as text boundaries will be saved. It can be specified between the text boundaries other commands. These are then stored as text and then processed in demand.

This command may not be combined with other commands in the same command instruction.

Load txt template {GLnn}

This command creates a text template is retrieved from memory and written into the display buffer.

Nn the location of the template is given.

By the hardware bug of no Clockstreching the I2C bus does not allow read.

With a shorter text anyway all 31 bytes are transferred. With repeated overwrite {GS} still text length of a previous longer text after the actual thing can be over in memory.

Set Csr X {Xnn}

This command sets the cursor for the next issue to the specified X position regardless of whether this is in the time column or elsewhere.

Valid values are from 0 to 95

Set Csr Y {Ynn}

This command sets the cursor for the next issue to the specified X position regardless of whether this is in the time column or elsewhere.

Valid values are from 0 to 15

Character { {{}

has a {character on display represent first be changed with {to command mode and then again one {be specified. After that, the command mode} required to leave.

Start bootloader {2BOOT}

With this sequence, the boot loader. The boot loader then waits for commands until the time-out expires (approx 5 sec). Then the display program is started again. The displays are turned off with display off before the start of the boot loader.

Firmware reset {! *! N}

With this command, the controller can be partially or entirely to "factory settings" reset.

N the corresponding range bit is as indicated:

Bit 0 = Reset Display EinstellungenBit 1 set = ResetFont and BMPs in non-volatile memory. Bit 2 = Reset text templates.

A combination of each bit's is possible in any form. z. B. {! *! 7} resets all areas. (Bit 0 to 2 = 1).

How are the byte values of fonts and bitmaps determined

The fonts and bitmaps are next to Brechnen.

A byte is 8 bits's - OK - each bit represents a pixel - OK -> Pro byte = 8 pixels.

The bytes are starting the x axis. 0 The bit's in a byte are starting at bit 0 above, the y axis

Example of an "A":

		value	X axis	X axis	X axis	X axis	X axis	X axis
			byte 0	byte 1	byte 2	byte 3	byte 4	byte n
y-axis	pixel 0	1		1	1			
y-axis	pixel 1	2	1			1		
y-axis	pixel 2	4	1			1		
y-axis	pixel 3	8th	1	1	1	1		
y-axis	pixels 4	16	1			1		
y-axis	pixel 5	32	1			1		
y-axis	pixels 6	64						
y-axis	pixel 7	128						
TOTAL			62	9	9	62	0	0

All values of Y-axis where a 1 in the X-axis are summed and is registered under the sum. So you have the byte value of 8 pixels.

The length is one of the bytes representing the width.

The example is: length = 5, = 62 byte0, byte1 = 9, byte 2 = 9, Byte3 = 62, Byte4 = 0in hexadecimal 05, 3E, 09, 09, 3E, 00

When writing the font's and BMP's are not used bytes must be specified with 0 anyway. That would be for a new font length + 6 bytes: 05 09 09 3E 3E 00 00 for bitmaps, the bytes have to be filled on the 20th

Copyright, disclosure, etc. ...

This project is made with the help of examples and code from people who have made public this code and have granted the right to private use. I adapted for my use or completely rewritten some of these parts.

Here is a list of pieces of code with author:

Bootloader:

Author: Copyright (C) 08/2010 by Olaf Rempel
License: GNU V2
Note: Adapted and partly rewritten HexFile Pharsing by:
http: // www. microcontroller. net / articles / AVR_Bootloader_in_C_-_eine_einfache_Anleitung

Software I2C:

Author: no appearances
was derived from Peter Fleury's I2C software library,
License: GNU header lt v3.
Note: Adapted, so it works without the Arduino Framework.
https: // github. com / felias-fogg / SoftI2CMaster

Display Control:

Adafruit SSD1306 Library: Written by Limor Fried / Adafruit Industries for Ladyada.
Adafruit GFX Library: Written by Limor Fried / Adafruit Industries for Ladyada.
License: Copyright (c) 2013 Adafruit Industries. All rights reserved.
Redistribution and use in source and binary forms, with or without modification, are permitted provided did the Following conditions are met:
- redistribution of source code must retain the above copyright notice, this list of conditions and the Following disclaimer.
- redistribution in binary form must reproduce the above copyright notice, this list of conditions and the Following disclaimer in the documentation and / or other materials provided with the distribution.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANT ABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE Liable FOR ANY DIRECT, INDIRECT, incidental, SPECIAL, EXEMPLARY, OR Consequential DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER Caused AND ON ANY THEORY OF LIABILITY, IN Whether CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) Arising IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SEARCH DAMAGE.

Note: used from the two libraries Adafruit only the code and partially adapted, which was needed. (Too much overhead and ported from C ++ to C what I needed) This copyright applies only to the respective pieces of code originally from the SSD1306 and GFX Library

Entire project:

As noted in the source code of each project for his private purposes is allowed to use and modify. I reject all liability for damages or consequential damages in every conceivable and not conceivable way. Commercial use of the software, hardware or parts thereof is not permitted without my consent.

In short: as soon as anyone wants any remuneration in any form or way from this project, they may only do so after first having requested, and obtained, permission from me.

Why bother? I was taken by a model train colleagues on the idea and asked if I could achieve something. So the basic idea is certainly not mine. In implementing the idea I also have some mental products used on the other (see above). Since this project was born as a hobby it is to others that something will also have to give much pleasure and not wishing to capitalize on someone.

Who am I? (Yes, I often wonder ....).

I'm Walter Sax and claim the copyright to the software and hardware of this project, except for the parts where the copyright is owned by the original authors.

The idea came from who others: Peter Ploiner - he has made the search for the displays.

for Developers

controller software

The software was written in AVR Studio 6 in C.
AT Mega is flashed with the following fuses:

Fuse Low: 0xDF (16Mhz - ext. OSC)
Fuse High: 0xD1 (2048 bootloader enter boot loader after reset)
Fuse Ext.: 0xFD (2, 7V Brown out Detection)

If it is desired that the controller after power not 5 seconds. In the boot loader waits, the Fuse High alternatively also set to 0xD0. Thus, the program will start immediately.

bootloader

In order to update the firmware to facilitate there is a boot loader, which can be processed via the I2C bus, the Intel hex files. The communication protocol for the boot loader is:

Command	Read / Write	Answer Len	code	Len
Get bootloader version	R	<= 40	0x01	
Get Info .mu.C	R	18	0x02 0x00 0x00 0x00 0x00	
Flash File Line	W	ACK / NACK	0x02 0x01 {Line of HEX file}	Max 64
Exit Bootloader	W	-	0x01 0x80	

RED send = Master / Slave Receive
BLUE = Slave Send / Receive Master
Get bootloader version:

I2C protocol: SLA + W, 0x01, SLA + R, {<= 40 bytes ascii chars} STO

This command cancels the timeout and the consequent automatic start of the main program.

The return looks following: TWI_RN_Display bootloader m328p v1. 11

Get .mu.C info:

I2C protocol: SLA + W, 0x02, 0x00, 0x00, 0x00, 0x00, SLA + R{14} bytes, STO

This command cancels the timeout and the consequent automatic start of the main program.
The return is the following:
[Signatur0]; [Signature1]; [Signatur2], [Flash Page Size]; [boot loader start HighByte] [boot loader start LowByte]; [EEprom end address HighByte] [EEprom end address LowByte]

Flash File Line:

With this command one line of the HEX file is transmitted.
Maximum line length is 64 bytes.
A Pharesen the HEX file is not necessary, it is integrated in the boot loader.
The following Intel Hex Line codes are integrated:

- 0 Flash data
- 1 Last Line
- 2 Extended Segment Address
- 4 Extended Linear Address

After successfully flashing the program will start automatically.
If a checksum error occurred as a yellow lights and the red LED at the same time and the controller waits for a manual reset (interrupting the power supply or jumper JP5 for a brief moment short).

After transmitting a Flash Line of ACK + 5ms remains to be seen. Writing the Flash Page takes an average of 3 - 4 ms.

To speed up the transfer, it is also possible the 5ms only to wait and see if a page is written to the Flash. The ACK bit remains to be seen in any case. For this size can be read by the boot loader with .mu.C Info command the Flash Page.

Exit Bootloader:

With this command the boot loader to flash file without a HEX be left.

The boot loader waits about 5 sec after the application of the operating voltage or the transmission of the boot loader command data. Next, the display program is started.

display software

For the display control parts of Adafruit GFX libraries were used. The software I2C bus has been with the Arduino Soft I2C library which has been adapted for use without Adruino Framework realized. The copyright of both libraries (excl. Changes) lies with the original authors. The program may be used for private purposes and modified. I reject all liability for damages and any guarantees. Any use or publication in the Commercial area where the software for components with - or for that software or parts of it, are sold in return for payment for profit, is prohibited without prior approval.

The hardware I2C bus is so programmed that the commands are executed only after the last of the controller ACK to the I2C bus as quickly as possible for other transmissions to free. This results in a period where the controller can not accept any new commands. During this time start sequences which are directed to the controller answered with NACK. That is, should the host program a NACK after a start sequence is then get to the next test, a waiting time of about 300ms observed. After three times NACK can be assumed that the controller is not connected to the bus. As an example: The update of the time required per display for about 16ms connected at 4 displays the controller can process new commands after about 64ms after the stop bit.

The data length to the controller is limited to 129 bytes.
Wherein the first byte is the display number according to the I2C address.
The remaining 128 bytes remain for text or setup.

The commands for the display control are all ASCII commands. An order for an action to begin with a curly brace and will be completed with a curved clip. Most commands can be combined within the brackets. There are a few commands that may stand between the brackets only alone, it is mentioned in the list of commands. In search of the values only one command can be sent and before the next time you send the answer is to be seen.
All characters outside the brackets are displayed as text on the display.
The numbering of the display, rows and columns beginning with 0. Except for the display number as the first byte in the data transmission, as is the numbering beginning in consideration of the existing implementation of RocRail at 1. In the command SetActive display the first display is to address. 0
The commands always apply to the currently active display. Switches the active display within a chain of command, all the following instructions for the display apply to which was changed in the chain of command.
Write all the commands which result in a change of the display resulted in an internal display buffer. To see the changes on the display it must be written with a command to the display. A Clear the display buffer sets the column to 0 and the line also to 0. Thus, the sequence of instructions after deleting for output left starting at the top for the set of row and column omitted.

I2C protocol

The transfer to the I2C bus is to begin with a start bit and conclude with a stop bit. In a write of "cost" to the subsequent read may be sent between the writing and the other starting signal for reading no stop signs. The last requested byte is required with a NACK as an indicator for any further data to confirm before the stop bit.

General I2C protocol data writing:

M M S M S M S M S M
<STA> <Address + W><ACK><DisplayNr [byte]><ACK><Byte 1><ACK><Byte n><ACK><STO>

General I2C protocol data reading:

M M S M S M S M S
<STA> <Address + W><ACK><DisplayNr [byte]><ACK><Byte 1><ACK><Byte n><ACK>

M M S S M S M S M M
<STA> <Address + R><ACK> <Byte 1><ACK><Byte 2><ACK><Byte n><NACK> <STO>

M ... sent from MasterS ... sent from slave

The address of the controller is specified as 7-bit address and must be sending one bit are shifted to the right and Bit 0 is the write or read bit.

The DisplayNr must be specified in the protocol as a byte value.

Raspi HW Bug: Due to a hardware bug a readout of settings with an I2C baud rate of 100kHz is not possible. For reading out the data, a conversion of the baud rate to 400 kHz is required.

the Raspberry PI settings

A send the commands is no problem with the default setting.

Unfortunately, the I2C hardware has a bug which has such an impact when reading the data that there is a bit shift when defaults are set. Therefore adjust for reading the I2C baud rate. A reading of data is only necessary during the initial setup. After that it should no longer be needed.

A configuration of the Rocdisplays must not be performed during operation.

The following steps are necessary to enable a readout of settings:

Connect to the Raspberry Pi by means of the terminal program such. B. PUTTY

In the console the following steps to perform:

Stop the service rocnetnoded:

```
sudo service rocnetnoded stop
```

Setting the I2C baud rate (not permanent)

```
sudo modprobe -r i2c_bcm2708
sudo modprobe i2c_bcm2708 baudrate = 400000
```

```
Starting the tools to configure: cd / home / pi / RocDisplayConfig
sudo. / rocdisplayconf

lt performing the configuration. prompts.

After the configuration tools finish:

Setting the standard I2C Baud Rate: sudo modprobe -r i2c_bcm2708
sudo modprobe i2c_bcm2708 baudrate = 100000

Starting the services rocnetnoded: sudo service rocnetnoded start
```

WARNING: Rocnet mouse (PI-04) can only be operated with the default setting of the I2C baud rate on the Raspberry Pi.

This sequence is contained in a sh script in the configuration tool.

RocDisplay configuration tool

This tool is carried out directly on the Raspberry Pi in the console.

With this tool it is possible to play new firmware to the display controller and also change the settings of the display.

This tool must be called to access the I2C bus with the superuser sudo.

To change settings using the configuration tool, or to load a new firmware to the controller as previously mentioned be set to 400kHz I2C baud rate. Here, the RocNetNode Service must be stopped.

The tool guides you through simple menus through the configuration. the aufzuspielende file must to have the menu for firmware update visibly be specified when the program.

The following options are possible for calling the tools:

-f [Hex File Name]	Specifying the files for Firmware Update
-a [I2C bus address]	Specify the bus address on the I2C bus
-u	Updating the firmware without further user intervention

To upgrade it to the firmware without user intervention must be given every 3 parameters.

After configuring the service of RocNetNode must be started again. If the service is not to be terminated, it is possible that with simultaneous access to the I2C bus through both applications an undefined status is created and the messages can not be forwarded to RocRail. Unfortunately, the Raspberry PI does not check whether it is already in use by another communications front of the bus access.

The installation and use of the configuration tool is in the readme. txt described by the configuration tool.