RocDisplay
Technische Beschreibung

Doku Version 0

September 2015

RocDisplay
Inhaltsverzeichniss

Das Projekt

Worum geht es hier eigentlich?

Ausgangspunkt aller Überlegungen war ein Software-gesteuerter Bahnsteiganzeiger im Maßstab H0

Neuerdings verfügbare und preiswerte OLED-Displays mit 96x16 Pixel und 26,3x8mm Baugröße erfüllen den Anspruch an Maßstäblichkeit schon sehr gut.

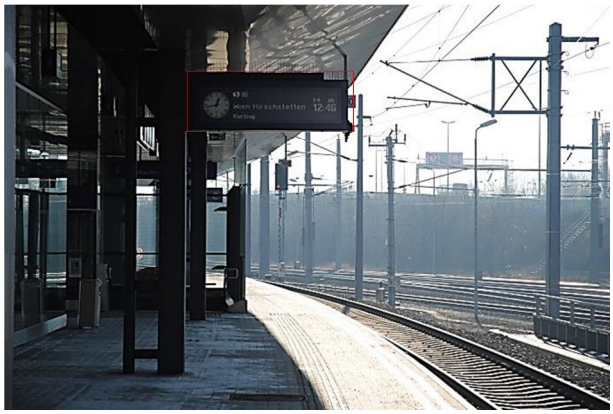


Abb. Der rote Rahmen stellt ca. die Displaygröße im Maßstab 1:87 im Vergleich zu einem ÖBB Bahnsteig dar.

Wir haben als Nutzer der Open-Source-Modellbahnsteuerungs-Software RocRail natürlich die Entwicklung einer Steuerung des Displays durch RocRail ins Auge gefasst. Die Schnittstellenelektronik/der Controller wurde als Baugruppe zum RocNetNode entwickelt.

Disclaimer

Wie im Source Code vermerkt darf jeder das Projekt für seine Privaten Zwecke verwenden und ändern. Es wird jegliche Haftung für Schäden oder Folgeschäden in jeder erdenkbaren und nicht erdenkbaren Art und Weise abgelehnt. Eine kommerzielle Nutzung der Software, Hardware oder Teile davon ist ohne Zustimmung der Autoren nicht gestattet.

Kurz gesagt: Sobald irgendwer für irgendwas aus diesem Projekt irgendeine Gegenleistung in irgendeiner Form und irgendeiner Weise irgendwie will oder verlangt, so darf er es nur wenn er irgendwann die Zustimmung dafür von den Autoren bekommen hat.

Warum das Ganze? Die Grund Idee für dieses Projekt hatte Peter Ploiner. Er ist der gedankliche Vater dieses Projektes. Im Source Code wurden Teile von Personen verwendet welche den Code unter GPL im Internet veröffentlicht haben (siehe Abschnitt Copyright).

Da dieses Projekt als Hobby entstanden ist soll es anderen welche so was auch haben wollen Freude bereiten und nicht jemanden der daraus Kapital schlagen will.

Source Code und Hardware ist von Walter Sax erstellt worden. Das Copyright, ausgenommen der Codeteile im Abschnitt Copyright erwähnt, liegt beim jeweiligen Autor.

Eine Einbindung der Funktionalitäten für eine Ansteuerung des RocDisplay's in eine Software ist gerne gesehen. ;)
Also jedem dem das RocDisplay gefallt, darf uns auf einen Kaffee oder ein Bier einladen, sollten wir uns mal über den Weg laufen ;)

Komponenten

Das RocDisplay besteht aus 3 Bauteilen.

1. Display Controller
2. Display Backplane
3. Display

Der Controller kann direkt an die RocNetNode angeschlossen werden. Über den Bootloader kann wenn im Host integriert die Firmware des Controllers upgedatet werden.

Die Display Backplane ist eine kleine Platine an der das Display, die notwendigen SMD Bauteile für den Spannungskonverter im Display und die Anschlüsse zum Controller vorhanden sind.

Das Display wird direkt auf die Backplane gelötet. An einer Backplane können 2 Displays angelötet werden.

Der Controller

Der Controller wird direkt mit der RocNetNode über RJ12 oder RJ45 Stecker verbunden. Die Spannungsversorgung ist nur über die RocNetNode möglich.

Der Controller arbeitet mit 5V. Eine höhere Eingangsspannung kann den Controller und die Displays zerstören!

An einen Controller können bis zu vier Displays angeschlossen werden.

Es wird empfohlen bei mehreren Kontrollern diese wenn möglich auf mehrere RocNetNodes aufzuteilen, oder an eine RocNetNode anschließen welche keine Rückmeldeaufgaben für Züge besitzt. (Teilung in Prozess- kritische und -unkritische Aufgaben).

Controller Daten

Bezeichnung	**Min**	**Typ**	**Max**
VDD	4,5V	5,0V	5,5V*
I2C Speed	-	-	400kHz
I2C Address	0x50		0x5F
$V_{Display\ Logic\ OUT}$	2,9	3	3,1
$V_{Display\ VDB\ OUT}$		VDD – ~0,7V	VDD
Command to Display			300ms
Command Length	0		128
Char Encoding		ISO 8859-1	
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	

* Nur bei Verwendung einer oder zwei Diode(n) in der +5V Leitung zu den Displays (Pin1)

Controller Aufbau

Der Controller arbeitet mit einen AT MEGA 328 PU Mikrocontroller. Dieser ist aus Sicherheitsgründen um die RocNetNode vor zu hoher Spannung zu schützen über einen I2C Level Shifter an den I2C Bustreiber (P82B714) über den Hardware I2C Bus angeschlossen. Die maximale I2C Busfrequenz liegt bei 400kHz. Die Pullup Wiederstände für den I2C Bus von der RocNetNode sind nicht am Display Controller vorgesehen. Für die Busspannung zwischen Bustreiber und Level Shifter ist ein 3,3V Linear Spannungsregler mit den dazugehörigen Kondensatoren vorhanden. Dadurch ist auch bei versehentlichen HIGH output vom Microcontroller die RocNetNode vor zu hoher Busspannung geschützt.
Die I2C Bus Adresse kann über Jumper eingestellt werden. Wobei ein gesteckter Jumper eine Logische 0 bedeutet. Die Adresse ist von HEX 0x50 bis 0x5F einstellbar (7 Bit Adresse – 8 Bit Adresse = 7Bit Adresse x 2).

Tabelle der möglichen Jumper settings:

J1	J2	J3	J4	I2C Adresse Hex	I2C Adresse 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

- … Offen
x … gesetzt

Diese Adresse wird ebenfalls vom Bootloader verwendet.
Um den I2C Bus zur RocNetNode möglichst schnell wieder freizugeben wird jedes Command in einen Buffer zwischengespeichert und erst nach dem Stopp Bit verarbeitet. Beim Empfangen und Senden von Daten leuchtet die rote LED für die Dauer der Übertragung auf. Sollte die Rote LED konstant leuchten, so wurde die letzte Nachricht nicht vollständig empfangen. (fehlendes Stop bit).

Zur Speicherung der Fonts, Bitmaps, Text Templates und Displayeinstellungen ist ein 8KB NV-Ram vorhanden. Dies ist ein Ferro elektrischer nicht flüchtiger Ram Speicher. Diese Art von Ram hat gegenüber von EEprom‘s und Flash Speicher den Vorteil dass die Zugriffszeiten in einen Bereich von ca. 100 ns liegen. Dies entspricht der Zugriffszeit von D-Ram’s. Die Zahl der Schreib und Lesezyklen ist um den Faktor 10^4 höher als bei EEprom und 10^6 höher als bei Flash Speicher. Einziger Nachteil ist das ein Lesen von einem Bit dieselbe physikalische Wirkung hat wie ein schreiben von einen Bit.

Für die vier Displays ist ein I2C Switch vorhanden welcher auch als Level Shifter von 5V auf die Logikspannung von 3V der Displays dient.
Für den I2C Switch ist ein Löt-Jumper vorhanden um ggf. die Bus Adresse ändern zu können. Standardmäßig ist der Jumper mit einer dünnen Leiterbahn auf GND verbunden. Bevor der Jumper auf 3V gelötet wird muss mit einen Skalpell die Leiterbahn zu GND durchbrochen werden. (Doch ohne

Firmware Änderung macht es keinen Sinn) – (Ist gedacht für eine eventuelle Erweiterung zur zusätzlichen Unterstützung von größeren Displays (128x64))

Für die Reset-Leitungen der Displays und des I2C Switch ist jeweils ein Spannungsteiler vorhanden um die 5V des Mikrokontrollers auf 3V zu begrenzen.

Zwei Status LED sind zur Optischen Kontrolle des Controller Status vorhanden. Die rote Led leuchtet auf wenn am Hardware I2C Bus Daten übertragen werden (auch im Bootloader Modus).

Im Bootloader Modus blinkt die Gelbe LED. Im Programm Modus leuchtet die Gelbe LED wenn alles OK ist. Ein Flackern bedeutet das Datenaktivität am Software I2C Bus stattfindet. Wenn die gelbe LED aus ist so liegt ein Übertragungsfehler am Software I2C Bus vor.

Für den Anschluss der Displays sind vier Stecker mit folgender Kontaktbelegung vorhanden:

Pin 1	+ 5 V
Pin 2	+ 3 V
Pin3	GND
Pin4	SDA
Pin5	SCL
Pin6	RESET (Active LOW)

Die Leitungen zu den Displays müssen so kurz wie möglich gehalten werden. Ein geschirmtes Kabel wäre das Optimum, aber ist sehr schwer auf die DisplayBackplanes lötbar.

Um das Display vor zu hoher Spannung am Pin1 (+5V = VDDB) zu schützen muss in die Leitung zum Display eine oder zwei Dioden zusätzlich eingefügt werden. Ohne diese Dioden ist die Maximale Versorgungsspannung des Kontrollers statt 5,5V nur max 5V.

Die Pins für Rx, Tx der UART Schnittstelle und der benachbarte GPIO Pin sind auf Lötpunkte im Raster 2,54mm ausgeführt, falls jemand die Firmware von I2C auf UART Kommunikation umschreiben möchte. Für die Entwicklung waren sie als Serielle Debug Ausgabe recht angenehm.

Die Pullup Wiederstände für die Displays und den Software I2C Bus sind mit 2k2 eher hoch dimensioniert. Bei längerem Kabel zu den Displays können die Pullup‘s ggf. bis ca. 1k herabgesetzt werden.
Im Test hat es bei 40 cm Display Anschlusskabel (einzeladern) keine Probleme gegeben.

Die Controller Platine (Maße: 50x80 mm):

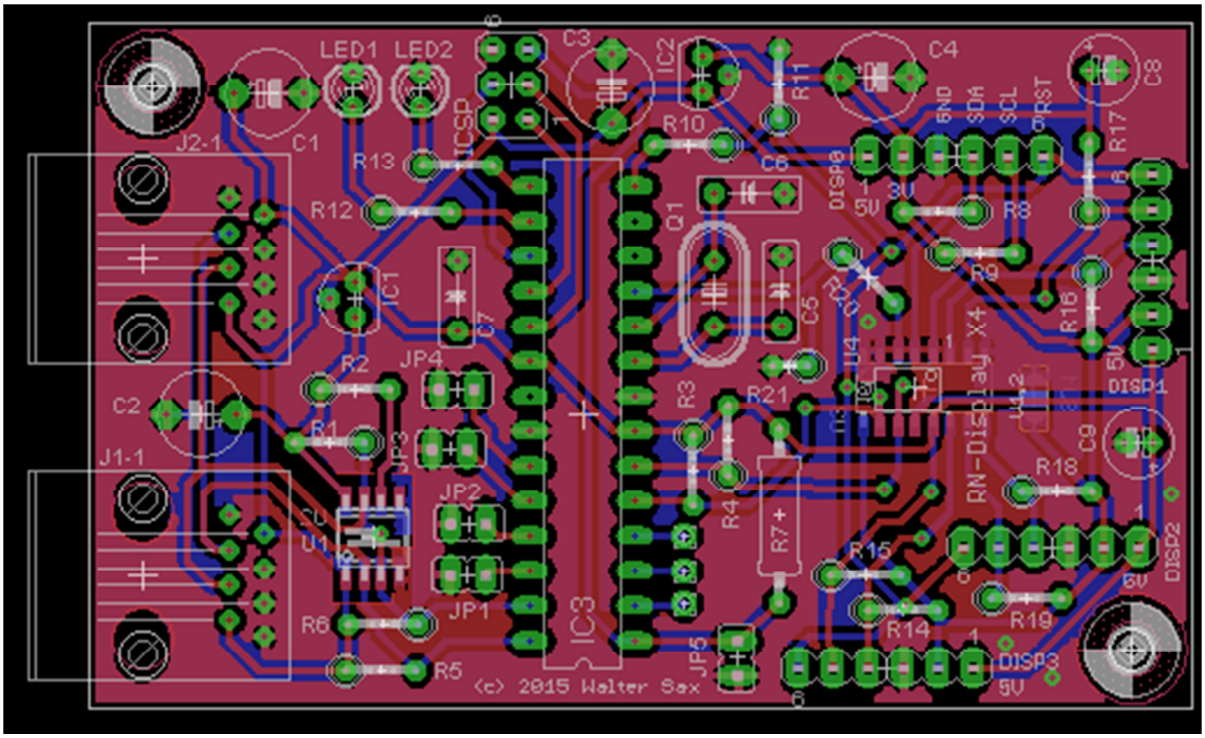


Abbildung 1 Display Kontroller

Stückliste für den Controller:

Bauteil	Wert	Bemerkung
C1, C3, C8	1µF / 16V	
C2, C4, C9	2,2µF / 16V	
C5, C6	22pF Keramik	
C7	100nF Keramik	
Disp0 – 3	6Pol 2,54mm Stecker	Oder auch Pinleiste
IC1	LP2950ACZ-3.3	
IC2	LP2950ACZ-3.0	
IC3	AT Mega 328P – PU	
ICSP	6 Pol 2,54mm Pinheader	
J1, J2	RJ45 oder RJ12	
JP1,JP2,JP3,JP4,JP5	Jumper	
Led1	Led 3mm Gelb	
Led2	Led 5mm Rot	
Q1	16Mhz Quarz HC49u-V	
R1,R2,R11,R21	3k3	
R3,R4,R5,R6	4k7	
R7	47k	
R8,R9,R10,R14,R15,R16,R17,R18,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 und Display

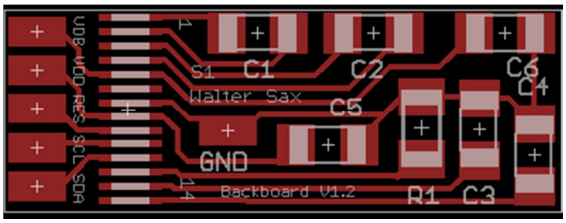
Auf der Display Backplane befinden sich die notwendigen Kondensatoren für jedes Display.

Das Display wird direkt auf die Backplane aufgelötet. Eine Backplane kann 2 Displays aufnehmen. Die Anschlüsse sind jedoch für jedes Display getrennt.
Die Größe der Backplane ist 10x25mm so dass sie zwischen zwei Displays passt.

Beim Löten der Backplane sollten die Displays als aller letztes aufgelötet werden, sonst ist es fast nicht mehr möglich die Anschlussdrähte anzulöten.

Achtung beim Display löten, die Anschlussfahne der Displays ist bei V1.2 scharf geknickt um einen Versatz der Displays bei Beidseitiger Montage zu verhindern.

Die Display Backplane Platine (Maße 25 x10 mm):



Die Platine ist Doppelseitig, somit kann sie zwischen 2 Displays in der Mitte platziert werden.
Die Anschlüsse sind folgende:

Display Backplane	Controller	Pin
SDA	SDA	4
SCL	SCL	5
RES	RST (RESET)	6
VDD	+3V	2
VDB	+5V	1

Die Stückliste der Display Backplane:

Bauteil	Wert	Bemerkung
C1, C2	1µF / 16V X5R	SMD 1206 oder 0803
C3	4,7µF / 16V X7R	SMD 1206 X7R
C4	2,2µF / 16V	SMD 1206
C5, C6	1µF / 16V	SMD 1206
R1	430k	SMD 1206
Display	OLED 96x16 I2C 14Pin	SSD1306 Chip

Bei der Bestückung ist es einfacher wenn das Display als letztes nach den Anschlussleitungen aufgelötet wird.

Kontroller Software

Die Software wurde im AVR Studio 6 in C geschrieben.
Der AT Mega wird mit folgenden Fuses geflashed:

Fuse Low: 0xDF (16Mhz – ext. OSC)
Fuse High: 0xD1 (2048 Bootloader, enter Bootloader after reset)
Fuse Ext.: 0xFD (2.7V Brown out dedection)

Wenn es gewünscht ist, das der Kontroller nach dem Einschalten nicht 5 Sek. Im Bootloader wartet kann die Fuse High alternativ auch auf 0xD0 gesetzt werden. Somit wird sofort das Programm gestartet.

Bootloader

Um ein updaten der Firmware zu erleichtern ist ein Bootloader vorhanden, welcher über den I2C Bus die Intel Hex Files verarbeiten kann. Das Kommunikationsprotokoll für den Bootloader ist folgendes:

Command	**Read / Write**	**Antwort Len**	**Code**	**Len**
Get Boot Loader Version	R	<=40	0x01	
Get µC Info	R	18	0x02 0x00 0x00 0x00 0x00	
Flashfile Line	W	ACK/NACK	0x02 0x01{Line of HEX File}	Max 64
Exit Bootloader	W	-	0x01 0x80	

ROT=Master send / Slave Receive
BLAU=Slave Send / Master Receive
Get Bootloader Version:

I2C Protokoll: SLA+W, 0x01, SLA+R, {<= 40 Bytes Ascii chars}, STO

Dieses Kommando bricht das timeout und das daraus folgende automatische Starten des Hauptprogrammes ab.

Die Rückgabe sieht folgend aus: TWI_RN_Display Bootloader m328p v1.11

Get µC Info:

I2C Protokoll: SLA+W, 0x02, 0x00, 0x00, 0x00, 0x00, SLA+R, {14 Bytes}, STO

Dieses Kommando bricht das timeout und das daraus folgende automatische Starten des Hauptprogrammes ab.
Die Rückgabe ist folgende:
[Signatur0]; [Signatur1];[Signatur2];[FlashPagesize];[BootloaderStart HighByte][BootloaderStart LowByte];[EEprom EndAdress HighByte][EEprom EndAdress LowByte]

FlashFile Line:

Mit diesem Kommando wird eine Zeile des HEX Files übertragen.
Maximale Zeilen Länge ist 64 Bytes.
Ein Pharesen des HEX Files ist nicht notwendig, dies ist im Bootloader integriert.
Folgende Intel Hex LineCodes sind integriert:

- 0 Flashdaten
- 1 Last Line
- 2 Extended Segment Address
- 4 Extended Linear Address

Nach dem erfolgreichen Flashen wird das Programm automatisch gestartet.
Sollte ein Checksum Fehler auftraten so leuchten die Gelbe und die Rote LED gleichzeitig und der Kontroller wartet auf einen manuellen Reset (Unterbrechen der Spannungsversorgung oder Jumper JP5 für einen kurzen Moment kurzschließen).

Nach übertragen einer FlashLine muss der ACK + 5ms abgewartet werden. Das Schreiben der FlashPage dauert im Schnitt ca. 3 – 4 ms.

Um die Übertragung zu beschleunigen ist es auch möglich die 5ms nur dann ab zu warten wenn eine Page in das Flash geschrieben wird. Das ACK Bit muss in jedem Fall abgewartet werden. Dazu kann mit dem µC-Info Befehl die Flash Page größe vom Bootloader ausgelesen werden.

Exit Bootloader:

Mit diesem Kommando kann der Bootloader ohne ein HEX File zu flashen verlassen werden.

Der Bootloader wartet ca. 5 sek nach dem Anlegen der Betriebsspannung oder des Senden des Bootloader Befehles auf Daten. Danach wird das Display Programm gestartet.

DisplaySoftware

Für die Displayansteuerung wurden Teile des Adafruit GFX Libraries verwendet.
Der Software I2C Bus wurde mit dem Arduino Soft I2C Library welches für die Verwendung ohne Adruino Framework angepasst wurde realisiert.
Das Copyright beider Libraries (excl. der Änderungen) liegt bei den ursprünglichen Autoren. Das Programm darf für Private Zwecke verwendet und geändert werden. Ich lehne jegliche Haftung für Schäden und jegliche Garantien ab. Eine Verwendung oder Veröffentlichung im Kommerziellen Bereich wo für die Software, für Komponenten mit - oder für diese Software oder auch Teile daraus, entgeltlich zu Profitzwecken vertrieben werden, ist ohne die vorherige Zustimmung untersagt.

Der Hardware I2C Bus ist so Programmiert, dass die Kommandos erst nach dem letzten ACK des Kontrollers ausgeführt werden um den I2C Bus schnellst möglich wieder für andere Übertragungen frei zu geben. Daraus resultiert eine Zeitspanne wo der Kontroller keine neuen Kommandos entgegennehmen kann. In dieser Zeit werden Startsequenzen welche an den Kontroller gerichtet sind mit NACK beantwortet. Das heißt sollte das Host Programm ein NACK nach einer Startsequenz bekommen dann ist bis zum nächsten Versuch eine Wartezeit von ca. 300ms einzuhalten. Nach dreimaligen NACK kann man annehmen das der Kontroller nicht am Bus angeschlossen ist. Als Beispiel: Das aktualisieren der Uhrzeit benötigt pro Display ca. 16ms bei 4 angeschlossenen Displays kann der Controller nach ca. 64ms nach dem Stoppbit neue Befehle verarbeiten. Bei Synchronisierten Displays braucht die Ausgabe entsprechend länger, doch die 3x 300ms sollten der worst case sein.

Die Datenlänge an den Kontroller ist auf 129 Bytes beschränkt.
Wobei das erste Byte nach der I2C Adresse die Displaynummer ist.
Die restlichen 128 Bytes bleiben für Text oder Setup über.

Die Befehle für die Display Steuerung sind alles ASCII Befehle. Ein Befehl für eine Aktion beginnt mit einer geschwungenen Klammer auf und wird mit geschwungener Klammer zu beendet. Die meisten Befehle können innerhalb der Klammern kombiniert werden. Es gibt ein paar Befehle welche nur alleine zwischen den Klammern stehen dürfen, dies ist bei der Auflistung der Befehle vermerkt. Bei Abfrage der Werte darf nur ein Befehl gesendet werden und vor dem nächsten Senden ist die Antwort abzuwarten.
Alle Zeichen außerhalb der Klammern werden als Text am Display dargestellt.
Die Nummerierung der Displays, Zeilen und Spalten beginnt mit 0. Außer bei der Displaynummer als erstes Byte in der Datenübertragung, da ist die Nummerierung in Rücksicht auf die bestehende Implementierung von RocRail mit 1 beginnend. Bei dem Befehl SetActive Display ist das erste Display mit 0 anzusprechen.
Die Befehle gelten immer für das im Moment aktive Display. Wird innerhalb einer Befehlskette das aktive Display gewechselt, so gelten alle folgenden Anweisungen für das Display auf welches in der Befehlskette gewechselt wurde.
Alle Befehle welche eine Änderung des Displays zur Folge haben schreiben in einen internen Displaybuffer. Um die Änderungen am Display anzuzeigen muss dieser mit einem Befehl auf das Display geschrieben werden. Ein löschen des Displaybuffers setzt die Spalte auf 0 und die Zeile auch auf 0 zurück. Somit kann nach dem Löschen für eine Ausgabe links oben beginnend die Befehlsfolge für das setzen der Zeile und Spalte entfallen.

I2C Protokoll

Die Übertragung am I2C Bus ist mit einen Start Bit zu beginnen und mit einen Stopp Bit abzuschließen.
Bei einen schreiben von „werten“ zum nachfolgenden auslesen darf zwischen dem Schreiben und dem weiteren Startzeichen für das Lesen kein Stoppzeichen gesendet werden. Das letzte angeforderte Byte ist mit einem NACK als Kennzeichen für keine weiteren Daten gefordert vor dem Stoppbit zu bestätigen.

Generelles I2C Protokoll Daten Schreiben:

M M S M S M S M S M
<STA><Adresse+W><ACK><DisplayNr [byte]><ACK><Byte1><ACK><Byte n><ACK><STO>

Generelles I2C Protokoll Daten Lesen:

M M S M S M S M S
<STA><Adresse+W><ACK><DisplayNr[byte]><ACK><Byte1><ACK><Byte n><ACK>

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

M… gesendet von Master
S… gesendet von Slave

Die Adresse des Kontrollers ist als 7Bit Adresse angegeben und muss beim Senden um ein Bit nach rechts geschoben werden und das Bit 0 ist das Schreib- oder Lesebit.

Die DisplayNr ist im Protokoll als Bytewert anzugeben.

RasPi HW Bug: Durch einen Hardware Bug ist ein Auslesen der Einstellungen mit einer I2C Baudrate von 100kHz nicht möglich. Für ein Auslesen der Daten ist eine Umstellung der Baudrate auf 400kHz erforderlich.

Auflistung der Befehle:

Alle Befehle werden als ASCII Zeichen gesendet. Die Schriftarten sind nach ISO 8859-1 hinterlegt.
Zeichen welche nicht zwischen { und } stehen werden als Text am Display ausgegeben.
Lesekommandos sind generell als Bytewerte zu senden. Die Leseregister sind in der Erklärung der Befehle angegeben.
Die Antworten bei den Lesekommandos sind als Bytewerte gesendet, außer es ist explizit angegeben, dass die Antwort als ASCII gesendet wird. Alle Befehle sind Case sensitive.

Befehl Name	Code	Parameter	Read / Write	Single Command	Beispiel	Bemerkung
Displays Connected	Z	0 .. 15	R/W	No	{Z3}	Parameter als Bit Wert der Displays Display0 = Bit0 Display1 = Bit1 Display2 = Bit2 Display3 = Bit3
Display Sync	V	0 .. 15	R/W	No	{V2}	Parameter als Bit Wert der Displays Display0 = Bit0 Display1 = Bit1 Display2 = Bit2 Display3 = Bit3
Invert Display / Clock	I	0,1,128,129	R/W	No	{I129}	Invertiert das gesamte Display oder nur die Uhr. 0 .. Normal 1 .. Disp.Invert 128 Uhr Invert 129 Disp.Invert+Uhr Invert
Rotation	R	0,1,2,3	R/W	No	{R2}	Ausrichtung der Textanzeige: 0 .. 0° 1 .. 90° 2 .. 180° 3 .. 270° Std = 2
Brightnes	H	0 .. 255	R/W	No	{H255}	Setzt die Helligkeit des Displays. Änderungen <30 sind fast nicht wahrnehmbar.
Show Clock / Clock Side	S	0,1,L,R	R/W	No	{S1SR} {S0} {S1}{SL}	Anzeige der Uhr: 0 .. Uhr ausgeblendet 1 .. Uhr eingeblendet R .. Uhr Rechts L .. Uhr Links
Show Departure Column	D	0,1	R/W	No	{D1}	Anzeige der Abfahrtszeiten Spalte: 0 .. Ausgeblendet 1 .. Eingeblendet
Departure Column Width	W	0 .. 95	R/W	No	{W20}	Breite der Abfahrtszeiten Spalte in Pixel
Active Display	A	0 .. 3	R/W	No	{A0}	Setzt die angegebene Displaynummer als aktiv. Alle folgenden Anweisungen gelten für das Angegebene Display.
Write Bitmap	M	Nr;Data	R/W	Yes	{M0;[21 Bytes 2char HEX]}	Überschreibt das unter der Nr angegebene Bitmap mit den Daten aus Data. Nr = 0 .. 19 Data = 21 Bytes 2Digit Hex mit ; als seperator. Byte 0 = Breite Byte 1 .. 20 Bitmap array
Write Font	N	Nr;CharCode;Data	R/W	Yes	{N0;4D;[7 Bytes 2char Hex]}	Überschreibt das mit CharCode angegebene Zeichen der unter Nr angegebenen Font mit den Daten aus Data. Nr = 0 .. 3 CharCode = 20 .. FF Data = 7 Bytes 2Digit Hex mit ; als Seperator. Byte 0 = Breite Byte 1 .. 6 Char Array

Display ON	U	0,1,2,3,A	W	No	{UA}	Schaltet das angegebene Display ein. Parameter A schaltet alle Connected Displays ein mit einen vorherigen Reset der Displays = Startup sequence.
Display OFF	O	0,1,2,3,A	W	No	{O1}	Schaltet das im Parameter angegebene Display aus. Parameter A schaltet alle connected Displays aus. Dieser Befehl mit Parameter A sollte vor jeden abschalten der Versorgungsspannung als letzter Befehl gesendet werden.
Setup Befehle auch für Synced Displays	K	0,1	W	No	{K1}	Nach diesem Befehl werden Setupänderungen auch auf den unter DisplaySync synchronisierten Displays angewendet. Diese Einstellung wird nicht gespeichert und auch nach dem leeren des Display Buffer wieder automatisch zurückgesetzt.
Set Clock	C	Hour:Minute	W	Yes	{0:25}	Setzt die Uhrzeit. Aktualisiert die Uhranzeige auf allen angeschlossenen Displays mit eingeblendeter Uhr.
Clear Display	E		W	No	{E}	Löscht den internen DisplayBuffer. Keine Aktualisierung des Displays. Für ein Löschen mit Aktualisierung ist {EP} zu setzen (Erase und PuntOnDisplay)
Put on Display	P		W	No	{P}	Aktualisiert das Display mit den Daten aus dem Displaybuffer.
Set Line	L	0,1	W	No	{L0}	Setzt den X cursor auf die Startposition der im Parameter angegebenen Zeile. Lösch jedoch nicht einen bereits vorhandenen Inhalt.
Set Column	T	0,1	W	No	{T1}	Wechselt in die angegebene Spalte: 0 = Zug Ziel Spalte 1 = Abfahrtszeiten Spalte Wenn die Abfahrtszeiten Spalte ausgeblendet ist und trotzdem beschrieben wird, so wird dieser Text nicht angezeigt.
Set Font	F	0,1,2,3	W	No	{F2}	Setzt die Font für den folgenden Text. Wird jeweils für die Zeile und Spalte dauerhaft gespeichert bis ein neuer Wert gesetzt wird.
Draw Bitmap	B	0 .. 19	W	No	{B10}	Setzt in der aktuellen Zeile in der aktuellen Spalte bei der aktuellen X Position das unter der Nr im Parameter angegebene Bitmap in den Display buffer.
Blink	J	0,1,2,S	W	NO	{J1}xxx{J2}yyy{J0} {JS2}	Mit J1 wird der Beginn des Blinkenden Bereiches gesetzt. Wenn J2 angegeben wurde ist der Folgende Text der Alternative Text. J0 setzt das Ende das Blinkbereiches. Im Blinkbereich können Fonts und Bitmaps gesetzt werden. Ein Zeilenwechsel beendet automatisch den Blinkbereich. JS speichert die Blinkzeit.
Save Txt Template	GS	NR 0..19 \Text\ .. max 30char	W	Yes	{GS0\Test Text\}	Speichert den Text zwischen den Backslash Zeichen als Template unter der Nr ab.
Load Txt Template	GL	0 .. 19	R/W	No	{GL0}	Ladet den Text des Template in den Display Buffer. Bei Abfrage für antwort auf I2C Bus: {QG[Nr]}
Set Csr X	X	0 .. 95	W	No	{X20}	Setzt die x position für die nächste Ausgabe.

Set Csr Y	Y	0 .. 15	W	No	{Y4}	Setzt die y position für die nächste Ausgabe.
Zeichen {	{		W	No	{{}	Um das Zeichen { am Display darzu stellen muss im Befehlsmodus ein weiteres Zeichen { gesendet werden.
Start Bootloader	2	BOOT	W	Yes	{2BOOT}	Wechselt zur Ausführung des Bootloaders
Firmware Reset	!*!	1 .. 7	W	Yes	{!*!7}	Bit 0 .. 1 = reset Display Settings Bit 1 .. 1 = reset Font & BMP Bit 2 .. 1 = reset Txt Templates Nach dem Firmware reset wird das Programm automatisch neu gestartet.

Alle Kommandos mit Single Command = no können innerhalb einer Befehlsfolge kombiniert werden.
z.B. {EB0F3} 2 2 {F2} Line 0{L1F0}Line 1{L0T1F2}Col 1 L0{L1F2}Col1 L1{P}
Dieses Beispiel ist folgende Befehlsabfolge(Befehle in [ ]) :
[Lösche Display Buffer] [Setze Bitmap 0 in Display Buffer] [Verwende Font 3] [schreibe 2 2]
[Verwende Font 2] [schreibe Linie 0] [Verwende Linie 1] [Verwende Font 0] [schreibe Line 1]
[Verwende Linie 0] [Verwende Spalte 1] [Verwende Font 2] [schreibe Col1 L0] [Verwende Linie 1]
[Verwende Font 2] [schreibe Col1 L1] [Zeige mir das jetzt am Display]

Erklärung der Befehle detailiert

Die Befehle werden als ASCII Chars gesendet. (Auch die Zahlen – z.B. 10 = 0x31, 0x30)

Displays Connected {Znn}

Dieser Befehl teilt dem Controller die belegten Display Ports mit.
Die Ziffer ist im Bereich 0 (kein Port belegt) bis 15 (alle Ports belegt) gültig.

Die Ports entsprechen den Bit's der Zahl.

Port	Wert
0	1
1	2
2	4
3	8

Die Zahl ergibt sich aus der Summe der angeschlossenen Portwerte.

Die Einstellungen werden im nicht flüchtigen Speicher hinterlegt und müssen nur einmal gesetzt werden.

Auslesbar mit Setzen von Register **0x01** nach Display Angabe.

Die Antwortlänge ist 1 Byte im Wert von 0 bis 15 (kein ASCII)

Display Sync {Vnn}

Dieser Befehl setzt die mit der Ausgabe mit zu synchronisierenden Displays des gerade Aktiven Displays.
Da diese Einstellung für jedes Display einzeln gesetzt werden kann ist vor dem Senden darauf zu achten, dass das gewünschte Display ausgewählt wurde.

Diese Einstellung ist analog zu den ConnectedDisplays. Die Zahl repräsentiert den Wert der gesetzten Bits 0 bis 3 für die Synchronisation.
Die Einstellung wird im nicht flüchtigen Speicher gespeichert.

Auslesbar mit Setzen von Register **0x02** nach Display Angabe.

Die Antwortlänge ist 1 Byte im Wert von 0 bis 15 (kein ASCII).

Invert Display / Clock {Innn}

Dieser Befehl bestimmt ob die Displayanzeige Invertiert ist oder nur die Analoge Uhr invertiert ist.

Befehl	Ausgabe
{I0}	Uhr und Display nicht invertiert
{I1}	Uhr und Display invertiert
{I128}	Uhr invertiert, Display nicht invertiert
{I129}	Uhr invertiert, Display invertiert

Die Einstellung wird im nicht flüchtigen Speicher für jedes Display gespeichert.
Es ist darauf zu achten, dass das entsprechende Display als aktiv ausgewählt wurde.

Mit dem Befehl [Setup Befehle auch für Synced Displays] kann der Wert auch für alle Synchronisierten Displays mitgesetzt werden.

Auslesbar mit Setzen von Register **0x03** nach Display Angabe.

Die Antwortlänge ist 1 Byte (nicht ASCII).

Rotation {Rn}

Dieser Befehl setzt die Rotation der Anzeige des Displays.

Es sind Werte von 0 bis 3 zulässig.

Wert	Rotation
0	0° (für Doppeldisplay notw.)
1	90° (macht keinen Sinn)
2	180° (Standard Einstellung)
3	270° (macht keinen Sinn)

Die Einstellung wird im nicht flüchtigen Speicher für jedes Display gespeichert.
Es ist darauf zu Achten, dass das entsprechende Display als aktiv ausgewählt wurde.

Auslesbar mit Setzen von Register **0x04** nach Display Angabe.

Die Antwortlänge ist 1 Byte (nicht ASCII).

Brightnes {Hnnn}

Dieser Befehl setzt die Helligkeit des Displays.

Gültige Werte sind von 0 bis 255

Die Einstellung wird im nicht flüchtigen Speicher für jedes Display gespeichert.
Es ist darauf zu achten, dass das entsprechende Display als aktiv ausgewählt wurde.

Auslesbar mit Setzen von Register **0x05** nach Display Angabe.

Die Antwortlänge ist 1 Byte (nicht ASCII).

Show Clock / Clock Side {Sn}

Dieser Befehl setzt die Anzeige der Uhr, ob ein oder ausgeblendet und ob links oder rechts.

Gültige Werte sind 0 oder 1 und L und R

Befehl	Anzeige
{S0}	Blendet die Uhr aus
{S1}	Blendet die Uhr ein
{SR}	Uhr Rechts
{SL}	Uhr Links

Die Einstellung wird im nicht flüchtigen Speicher für jedes Display gespeichert.
Es ist darauf zu achten, dass das entsprechende Display als aktiv ausgewählt wurde.

Mit dem Befehl [Setup Befehle auch für Synced Displays] kann der Wert auch für alle Synchronisierten Displays mitgesetzt werden. Dies kann sinnvoll sein wenn für eine Darstellung mehr Platz benötigt wird.

Show Clock: Auslesbar mit Setzen von Register **0x06** nach Display Angabe.
Clock Side: Auslesbar mit Setzen von Register **0x07** nach Display Angabe.

Die Antwortlänge ist 1 Byte.

Show Departure Column {Dn}

Dieser Befehl setzt die Anzeige einer sogenannten Abfahrtszeiten Spalte.

Beim Schreiben von Displaytext muss die jeweilige Spalte vorher durch den Befehl {Tn} ausgewählt werden. Sollte der Text länger als die Spalte sein so wird dieser an der letzten darstellbaren Position abgeschnitten. Ein Blinken über beide Spalten ist nicht möglich. Jede Spalte und jede Zeile in einer Spalte kann eine eigene Font Einstellung haben.

Die Einstellung wird im nicht flüchtigen Speicher für jedes Display gespeichert.
Es ist darauf zu achten, dass das entsprechende Display als aktiv gesetzt wurde.

Mit dem Befehl [Setup Befehle auch für Synced Displays] kann der Wert auch für alle Synchronisierten Displays mitgesetzt werden. Dies kann sinnvoll sein wenn für eine Darstellung mehr Platz benötigt wird.

Auslesbar mit Setzen von Register **0x08** nach Display Angabe.

Die Antwortlänge ist 1 Byte (nicht ASCII).

Departure Column Width {Wnn}

Dieser Befehl setzt die Breite in Pixel der Abfahrtszeiten Spalte.

Mit dem Befehl [Setup Befehle auch für Synced Displays] kann der Wert auch für alle Synchronisierten Displays mitgesetzt werden. Dies kann sinnvoll sein wenn für eine Darstellung mehr Platz benötigt wird.

Die Abfahrtszeiten Spalte ist immer am rechten Rand ausgerichtet. Wenn die Uhr Rechts angezeigt wird so wird die Abfahrtszeiten Spalte um die Breite der Uhr nach links verschoben.

Die Einstellung wird im nicht flüchtigen Speicher für jedes Display gespeichert.
Es ist darauf zu achten, dass das entsprechende Display als aktiv gesetzt wurde.

Auslesbar mit Setzen von Register **0x09** nach Display Angabe.

Die Antwortlänge ist 1 Byte (nicht ASCII).

Active Display {An}

Mit diesem Befehl kann das aktive Display gesetzt werden.
Durch die Protokolldefinition zwischen RocNetNode und Display Controller ist dies in jeder Nachricht automatisch enthalten.
Soll jedoch mit einer Übertragung ein Update auf mehreren Displays mit verschiedenen Daten stattfinden, so kann mit diesem Befehl das Display gewechselt werden.

Diese Einstellung wird nicht gespeichert. Das Display 0 ist nach dem starten das aktive Display.

Auslesbar mit Setzen von Register **0x0A** nach Display Angabe, doch durch die Protokolldefinition nicht wirklich sinnvoll.

Die Antwortlänge ist 1 Byte (nicht 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;nd;nd;}

Mit diesem Befehl kann ein Bitmap mit neuen Daten im nicht flüchtigen Speicher überschrieben werden.

Die Bitmap Größe ist auf 20x8 Pixel begrenzt.

Der erste Parameter n0 ist die Nummer des Bitmaps zwischen 0 und 19.

Der zweite Parameter ist die Bitmap Breite in Bytes als 2 digit Hexadezimal Zahl in ASCII.

Die folgenden 20 Parameter nd sind die Bitmap Daten als 2 digit Hexadezimal Zahl in ASCII.

Die Parameter müssen jeweils durch ein Semikolon getrennt werden. (auch nach dem letzten Parameter)

Wie die Bitmap Daten ermittelt werden ist später eigens erklärt.

Durch den Hardware Bug der kein Clockstreching am I2C Bus ermöglicht nicht auslesbar.

Write Font {Nn0;n1;n2;nd;nd;nd;nd;nd;nd;}

Mit diesem Befehl kann ein Font Zeichen mit neunen Daten im nicht flüchtigen Speicher überschrieben werden.

Die Zeichengröße ist auf 6x8 Pixel begrenzt.

Der erste Parameter n0 ist die Font Nummer 0 bis 3

Der zweite Parameter n1 ist der Code für das Zeichen als 2 digit Hexadezimal Zahl in ASCII 00 bis FF

Der dritte Parameter n2 ist die Zeichen Breite in Bytes als 2 digit Hexadezimal Zahl in ASCII 00 bis 06

Die folgenden 6 Parameter nd sind die Font Daten als 2 digit hexadezimale Zahl in ASCII.

Die Parameter müssen jeweils durch ein Semikolon getrennt werden. (auch nach dem letzten Parameter)

Wie die Font Daten ermittelt werden ist später eigens erklärt.

Durch den Hardware Bug der kein Clockstreching am I2C Bus ermöglicht nicht auslesbar.

Display ON {Un}

Dieser Befehl schaltet die Anzeige des angegebenen Displays wieder ein.
Wenn statt der Display Nummer ein A angegeben wird, wird dieser Befehl für alle angeschlossenen Displays ausgeführt.

Dieser Befehl wird nach dem Starten für alle angeschlossenen Displays ausgeführt.

Display OFF {On}

Dieser Befehl schaltet die Anzeige des angegeben Displays aus.
Wird statt der Display Nummer ein A angegeben, so wird dieser Befehl für alle angeschlossenen Displays ausgeführt.

Dieser Befehl mit Parameter A sollte vor dem Entfernen der Versorgungsspannung ausgeführt werden.

Setup Befehle auch für Synced Displays ein {K1} / aus {K0}

Dieser Befehl setzt die Einstellung das folgende Setup Befehle auch für die Synchronisierten Displays ausgeführt werden. Durch ein löschen des Displaybuffers wird dieser Befehl wieder zurückgesetzt, jedoch nicht die veränderten Setup Einstellungen.

{K1} aktiviert das Setup für Synchronisierte Displays.
{K0} deaktiviert das Setup für Synchronisierte Displays (Standard Einstellung).

Set Clock {Chh:mm}

Mit dem Befehl wird die Anzeige der Uhr aktualisiert.

hh ist die Stunde von 0 bis 24
mm ist die Minute von 0 bis 59
Es sind Werte mit führenden Nullen oder ohne zulässig. Z.B. 5 oder 05

Die Uhrzeit wird nicht automatisch aktualisiert. Die angezeigte Uhrzeit bleibt bis zum nächsten Set Clock Befehl stehen.
Nach dem Start ist die Uhrzeit auf 14:00 Uhr eingestellt.

Dieser Befehl darf zwischen den geschwungenen Klammern nicht mit anderen Befehlen kombiniert werden.
Nach dem verarbeiten des Befehles werden alle angeschlossenen Displays mit dem Inhalt des Display Buffers aktualisiert.

Clear Display {E}

Dieser Befehl löscht den Inhalt des Display Buffers.

Durch diesen Befehl wird auch der Befehl Blinken, Setup Befehle auch für Synced Displays zurückgesetzt. Auch eine Aktive Blinkanzeige wird durch diesen Befehl zurückgesetzt.

Weiter wird die Line 0 und Column 0 als aktiv gesetzt.

Put on Display {P}

Dieser Befehl gibt den Inhalt des Display Buffers des gerade Aktiven Displays auf der Anzeige aus.

Ohne das Senden von diesem Befehl wird die Anzeige nicht aktualisiert.
Dieser Befehl beendet auch eine nicht mit {J0} abgeschlossene Blinkdefinition.

Set Line {Ln}

Dieser Befehl gibt die aktuell zu beschreibende Zeile an. Folgende Text oder Bitmap Anweisungen werden in der angegebenen Zeile geschrieben. Die Zeilennummerierung beginnt mit 0.

Das 96x16 Display hat 2 Zeilen – Gültige Werte 0 oder 1

Durch diesen Befehl wird auch eine nicht abgeschlossene Blinkanweisung beendet, da eine Blinkzuordnung über mehrere Zeilen nicht möglich ist.

Die Aktuelle Spalte wird durch diesen Befehl nicht gewechselt.

Set Column {Tn}

Mit deisem Befehl kann zwischen den beiden Spalten gewechselt werden, falls die Abfahrtszeiten Spalte aktiv ist.

Spalte 0 ist die Zielort Spalte, Spalte 1 ist wenn aktiviert die Abfahrtszeiten Spalte.

Wird bei deaktivierter Abfahrtszeiten Spalte diese ausgewählt und beschrieben, so werden die Daten verworfen und nicht angezeigt.

Set Font {Fn}

Dieser Befehl setzt die für die Ausgabe zu verwendende Schriftart.
Werte von 0 bis 3 sind zulässig.

Font 0 ist eine 5x7 Pixel Schriftart mit einen Pixel Abstand zwischen den Zeichen (große Schrift).
Font 1 ist eine 5x6 Pixel Schriftart mit einen Pixel Abstand zwischen den Zeichen (mittlere Schrift).
Font 2 ist eine 4x6 Pixel Schriftart mit einen Pixel Abstand zwischen den Zeichen (kleine Schrift).
Font 3 ist eine 4x5 Pixel Invers Schriftart ohne Abstand zwischen den Zeichen, diese müssen mit Space explizit gesetzt werden.

Die gewählte Schriftart wird je Display für jede Zeile und Spalte im nicht flüchtigen Speicher gespeichert. Es wird beim Beschreiben der Zeile und Spalte die zu Letzt ausgewählte Schriftart verwendet.

Draw Bitmap {Bnn}

Dieser Befehl schreibt das Bitmap unter der nn angegebenen Nummer in die aktuell aktive Zeile und Spalte an aktueller Position in den Display Buffer.

Gültige Werte von 0 bis 19

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

Mit diesem Befehl kann ein Bereich als blinkend definiert werden.
Es ist ein Blinkbereich pro Display in einer Zeile und einer Spalte möglich.
Im Blinkbereich kann die Schriftart auch eigens mit {Fn} definiert werden.

Der Text oder die Bitmaps zwischen {J1} und {J2} ist der zu Blinkende Bereich.
Der Text oder die Bitmaps zwischen {J2} und {J0} ist die Alternativ Anzeige.
Je nach Blinkzeit Einstellung wechseln sich die Anzeige des Blinkbereiches und der Alternativen Anzeige ab. Die Tatsächliche Breite des Bereiches wird vom größeren Bereich hergenommen.

Wird {J2} nicht angegeben oder enthält keine zu darstellenden Zeichen so wird der Blinkbereich abwechselnd ein und ausgeblendet.

Ein löschen des Display Buffers setzt das Blinken wieder zurück.
Die Blinkbereiche werden in das externe RAM geschrieben und von dort gelesen. Dadurch wird der Display Conrtoller je Blinkbereich etwas mehr belastet.

Set Blink Time {JSn}

Mit diesem Befehl wird die Blinkzeit für das gerade aktive Display eingestellt.

Gültige Werte sind zwischen 1 und 9.

Tabelle der Blinkzeiten zu den Werten:

Wert	Zeit
1	0,5s
2	1s
3	1,5s
4	2s
5	2,5s
6	3s
7	3,5s
8	4s
9	4,5s

Diese Einstellung wird im nicht flüchtigen Speicher je Display gespeichert.

Auslesbar mit Setzen von Register **0x0B** nach Display Angabe.

Antwortlänge ist 1 Byte (nicht in ASCII).

Save Txt Template {GSnn\TEXT\}

Dieser Befehl speichert eine Textvorlage im nicht flüchtigen Speicher für einen späteren Abruf ab.

Mit nn wird die Speichernummer angegeben.
Der Text zwischen den Backslash als Textbegrenzungen wird gespeichert. Es können zwischen den Textbegrenzungen auch andere Befehle angegeben werden. Diese werden dann so wie der Text gespeichert und bei Abruf dann verarbeitet.

Dieser Befehl darf nicht mit anderen Befehlen in derselben Befehlsanweisung kombiniert werden.

Load Txt Template {GLnn}

Mit diesem Befehl wird eine Textvorlage vom Speicher abgerufen und in den Display Buffer geschrieben.

Mit nn wird die Speicher Position der Vorlage angegeben.

Durch den Hardware Bug der kein Clockstreching am I2C Bus ermöglicht nicht auslesbar.

Bei einem kürzeren Text werden trotzdem alle 31 Bytes übertragen. Bei mehrmaligen überschreiben mit {GS} kann noch von einen vorherigen längeren Text nach der Aktuellen Textlänge etwas im Speicher über sein.

Set Csr X {Xnn}

Dieser Befehl setzt den Cursor für die nächste Ausgabe auf die angegeben X Position unabhängig ob dies in der Uhrzeitspalte oder sonst wo ist.

Gültige Werte sind von 0 bis 95

Set Csr Y {Ynn}

Dieser Befehl setzt den Cursor für die nächste Ausgabe auf die angegeben X Position unabhängig ob dies in der Uhrzeitspalte oder sonst wo ist.

Gültige Werte sind von 0 bis 15

Zeichen { {{}

Um ein { Zeichen am Display darzustellen muss zuerst mit { in den Befehlsmodus gewechselt werden und dann nochmals ein { angegeben werden. Danach muss der Befehlsmodus mit } wieder verlassen werden.

Start Bootloader {2BOOT}

Mit dieser Sequenz wird der Bootloader gestartet. Der Bootloader wartet dann auf Befehle bis das Timeout abläuft (ca. 5 sek). Dann wird wieder das Displayprogramm gestartet.
Die Displays werden vor start des Bootloaders mit Display Off ausgeschaltet.

Firmware Reset {!*!n}

Mit diesem Befehl kann der Kontroller teilweise oder ganz auf „Werkseinstellung“ zurückgesetzt werden.

Mit n wird der entsprechende Bereich Bit weise angegeben:

Bit 0 gesetzt = Reset Display Einstellungen
Bit 1 gesetzt = Reset Font und BMP’s im nicht flüchtigen Speicher.
Bit 2 gesetzt = Reset Text vorlagen.

Eine Kombination der einzelnen Bit’s ist in jeder Form Möglich.
z.B. {!*!7} setzt alle Bereiche zurück. (Bit 0 bis 2 = 1).

Wie werden die Bytewerte der Fonts und Bitmaps ermittelt

Die Fonts und Bitmaps sind folgend zu brechnen.

Ein Byte hat 8 Bit's – OK – jedes bit stellt ein Pixel dar – OK -> Pro Byte = 8 Pixel.

Die Bytes sind mit 0 beginnend die x Achse. Die Bit's in einem Byte sind mit Bit 0 oben beginnend die y Achse

Beispiel für ein „A“:

		Wert	x-Achse	x-Achse	x-Achse	x-Achse	x-Achse	x-Achse
			Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte n
y-Achse	Pixel 0	1		1	1			
y-Achse	Pixel 1	2	1			1		
y-Achse	Pixel 2	4	1			1		
y-Achse	Pixel 3	8	1	1	1	1		
y-Achse	Pixel 4	16	1			1		
y-Achse	Pixel 5	32	1			1		
y-Achse	Pixel 6	64						
y-Achse	Pixel 7	128						
SUMME			62	9	9	62	0	0

Alle Werte der Y-Achse wo eine 1 in der X-Achse steht werden Summiert und unter Summe eingetragen. Somit hat man den Bytewert dieser 8 Pixel.

Als Länge gibt man die Bytes an welche die Breite repräsentieren.

Das Beispiel ist also: Länge = 5, Byte0=62, Byte1=9, Byte2=9, Byte3=62, Byte4=0
in Hexadezimal 05, 3E, 09, 09, 3E, 00

Beim Schreiben der Font's und BMP's sind die nicht gebrauchten Bytes mit 0 trotzdem anzugeben.
Das wäre für eine Font dann Länge + 6 Bytes: 05 3E 09 09 3E 00 00
Für Bitmaps müssen die Bytes auf 20 gefüllt werden.

RocNet Protokoll Erweiterung Vorschlag

Im Rocnet Protokoll ist die Displaysteuerung in Gruppe 12.

Code 1 ist Display Text – mit dem ist es auch möglich die Einstellungen des Displays zu Ändern.

Weiter würde ich Code 2 Vorschlagen für ein Auslesen der Daten.
Als Daten werden die Auslesebefehle gesendet. Als Antwort zum RR Server auch Code 2 mit Reply signed. In den Daten ist dann der AusleseBefehl und mit einen Trennzeichen die Antwort.
Falls es leichter ist kann ich auch noch die Replys vom DisplayController für die beiden ASCII Replys auf Byte umstellen.

Als Code 3 würde ich den Bootloader Modus vorschlagen.
Code 3 + Daten = 1 Zeile des Hexfiles.
Mit einer Message im Code 1 kann in den Bootloader Modus gewechselt werden und dann mit Code 3 jedes Mal eine Zeile des Hexfiles übertragen werden.

Da eigentlich bis auf den Bootloader und das Auslesen der Daten alles per normalen ASCII befehle gemacht werden kann ist meiner Meinung keine weitere Erweiterung des Protokolls notwendig.

Einstellungen des Raspberry PI

Ein senden der Kommandos ist mit der Standardeinstellung problemlos möglich.

Leider hat die I2C Hardware einen Bug welcher sich beim Auslesen der Daten so auswirkt, dass es zu einer Bitverschiebung kommt wenn die Standardwerte gesetzt sind. Deshalb ist für ein Auslesen die I2C Baudrate anzupassen. Ein Auslesen von Daten ist nur während der ersten Einrichtung notwendig. Danach sollte es nicht mehr von Nöten sein.

Eine Konfiguration des Rocdisplays darf nicht im laufenden Betrieb durchgeführt werden.

Folgende Schritte sind notwendig um ein Auslesen der Einstellungen zu ermöglichen:

Verbinden auf den Raspberry Pi mittels Terminal programm wie z.B. PUTTY

In der Konsole folgende Schritte durchführen:

Stoppen des Services rocnetnoded:

```
sudo service rocnetnoded stop
```

Setzen der I2C Baudrate (nicht dauerhaft)

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

```
Starten des Tools zur Konfiguration:
       cd /home/pi/RocDisplayConfig
       sudo ./rocdisplayconf

Durchführen der Konfiguration lt. Anweisungen am Bildschirm.

Nach beenden des Konfiguration Tools:

Setzen der standard I2C Baudrate:
       sudo modprobe -r i2c_bcm2708
       sudo modprobe i2c_bcm2708 baudrate=100000

Starten des Services rocnetnoded:
       sudo service rocnetnoded start
```

ACHTUNG: Eine Rocnet Maus (PI-04) kann nur mit der Standardeinstellung der I2C Baudrate am Raspberry Pi betrieben werden.

Dieser Ablauf ist in einen sh Skript bei dem Konfigurationstool enthalten.

RocDisplay Konfiguration Tool

Dieses Tool wird direkt am Raspberry Pi in der Konsole ausgeführt.

Mit diesem Tool ist es möglich neue Firmware auf den Displaykontroller zu spielen und auch die Einstellungen der Displays zu ändern.

Dieses Tool muss um auf den I2C Bus zugreifen zu können mit dem Superuser sudo aufgerufen werden.

Um Einstellungen ändern zu können ist die I2C Baudrate wie vorher angeführt auf 400kHz zu setzen. Dabei muss das RocNetNode Service gestoppt werden.

Das Tool leitet durch einfache Menüs durch die Konfiguration.
Um den Menüpunkt für Firmware update sichtbar zu haben muss beim Aufruf des Programmes das aufzuspielende File angegeben werden.

Folgende Optionen sind für den Aufruf des Tools möglich:

-f:[Hex File Name]	Angabe des Files für Firmwareupdate
-a:[I2C Busadresse]	Angabe der Busadresse am I2C Bus
-u	Update der Firmware ohne weiteren Benutzereingriff

Um die Firmware ohne Benutzereingriff aufzuspielen müssen alle 3 Parameter angegeben werden.

Nach der Konfiguration muss das Service der RocNetNode wieder gestartet werden.
Sollte das Service nicht beendet werden, so kann es vorkommen, dass bei gleichzeitigem Zugriff auf den I2C Bus durch beide Anwendungen ein undefinierter Status entsteht und die Meldungen nicht an RocRail weitergeleitet werden können. Der Raspberry PI prüft vor dem Bus Zugriff leider nicht ob dieser durch eine andere Kommunikation bereits belegt ist.

Copyright, Weitergabe, usw. …

Dieses Projekt ist mit Hilfe von Beispielen und Code von Leuten entstanden, welche diesen Code öffentlich zugänglich gemacht haben, und das Recht auf private Nutzung eingeräumt haben. Einige dieser Teile habe ich für meine Verwendung angepasst oder ganz neu geschrieben.

Hier eine Liste der Codeteile mit Autor:

Bootloader:

Autor: Copyright (C) 08/2010 by Olaf Rempel
Lizenz: GNU V2
Bemerkung: Angepasst und teilweise neu geschrieben HexFile Pharsing von:
http://www.mikrocontroller.net/articles/AVR_Bootloader_in_C_-_eine_einfache_Anleitung

Software I2C:

Autor: keiner Angegeben
wurde von Peter Fleury's I2C software library abgeleitet.
Lizenz: lt. Header GNU v3
Bemerkung: Angepasst, dass es ohne dem Arduino Framework auskommt.
https://github.com/felias-fogg/SoftI2CMaster

Displayansteuerung:

Adafruit SSD1306 Library: Written by Limor Fried/Ladyada for Adafruit Industries.
Adafruit GFX Library: Written by Limor Fried/Ladyada for Adafruit Industries.
Lizenz:

```
Copyright (c) 2013 Adafruit Industries.  All rights reserved.
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
- Redistributions of source code must retain the above copyright notice,
  this list of conditions and the following disclaimer.
- Redistributions 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 MERCHANTABILITY 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, WHETHER IN
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 SUCH DAMAGE.
```

Bemerkung: Aus den 2 Adafruit Libraries nur den Code verwendet und teilweise angepasst, welcher vonnöten war. (zu viel Overhead und von C++ auf C portiert was ich brauchte)
Dieses Copyright gilt nur für die jeweiligen Codeteile die ursprünglich aus dem SSD1306 und GFX Library kommen


S3 S40 Kurzzug
Tulln Stadt
ab
23:15

Gesammtes Projekt:

Wie im Source Code vermerkt darf jeder das Projekt für seine Privaten Zwecke verwenden und ändern. Ich lehne jegliche Haftung für Schäden oder Folgeschäden in jeder erdenkbaren und nicht erdenkbaren Art und Weise ab.
Eine kommerzielle Nutzung der Software, Hardware oder Teile davon ist ohne meine Zustimmung nicht gestattet.

Kurz gesagt: Sobald irgendwer für irgendwas aus diesem Projekt irgendeine Gegenleistung in irgendeiner Form und irgendeiner Weise irgendwie will oder verlangt, so darf er es nur wenn er irgendwann mich gefragt hat ob er es darf und ich dem auch zugestimmt habe.

Warum das Ganze? Ich wurde von einem Modellbahnkollegen auf die Idee gebracht und auch gefragt ob ich so was realisieren könnte. Also ist die Grundidee schon gar nicht meine. Bei der Umsetzung der Idee habe ich auch einige gedankliche Produkte anderer verwendet (siehe weiter oben). Da dieses Projekt als Hobby entstanden ist soll es anderen welche so was auch haben wollen Freude bereiten und nicht jemanden der daraus Kapital schlagen will.

Wer bin ich? (Ja das frage ich mich oft…..)

Ich bin Walter Sax und beanspruche das Copyright auf die Soft und Hardware von diesem Projekt ausgenommen der Teile wo das Copyright bei den ursprünglichen Autoren liegt.

Die Idee dazu hatte wer anderer: Peter Ploiner – er hat auch die Suche nach den Displays gemacht.

Eine Einbindung der Funktionalitäten für eine Ansteuerung des RocDisplay's in eine Software ist gerne gesehen. ;)

Also jedem dem das RocDisplay gefallt, darf uns auf einen Kaffee oder ein Bier einladen, sollten wir uns mal über den Weg laufen ;)