

# Moppel Hardware

## Interface Multi-IO-Karte

(Stand 19.12.2017)

### Inhalt:

Seite 2	Übersicht Hardware
Seite 3	ECB-Teil
Seite 4	UART-teil
Seite 5	PIO-Teil
Seite 6	Steuer- und Statuswort UART
Seite 7	Timer und UART Adressen
Seite 8	Belegung und Adressen der PIO-Bausteine
Seite 9	RTC Signalanalyse und Besonderheiten
Seite 10	Signalanalys V24
Seite 12	Softwaredetails

Anlagen:

Quellcode Testprogramme

Zusammengestellt:  
Werner Römer  
eMail: [werner.roemer@t-online.de](mailto:werner.roemer@t-online.de)  
WEB: [werners-seiten.de](http://werners-seiten.de)

# Beschreibung:

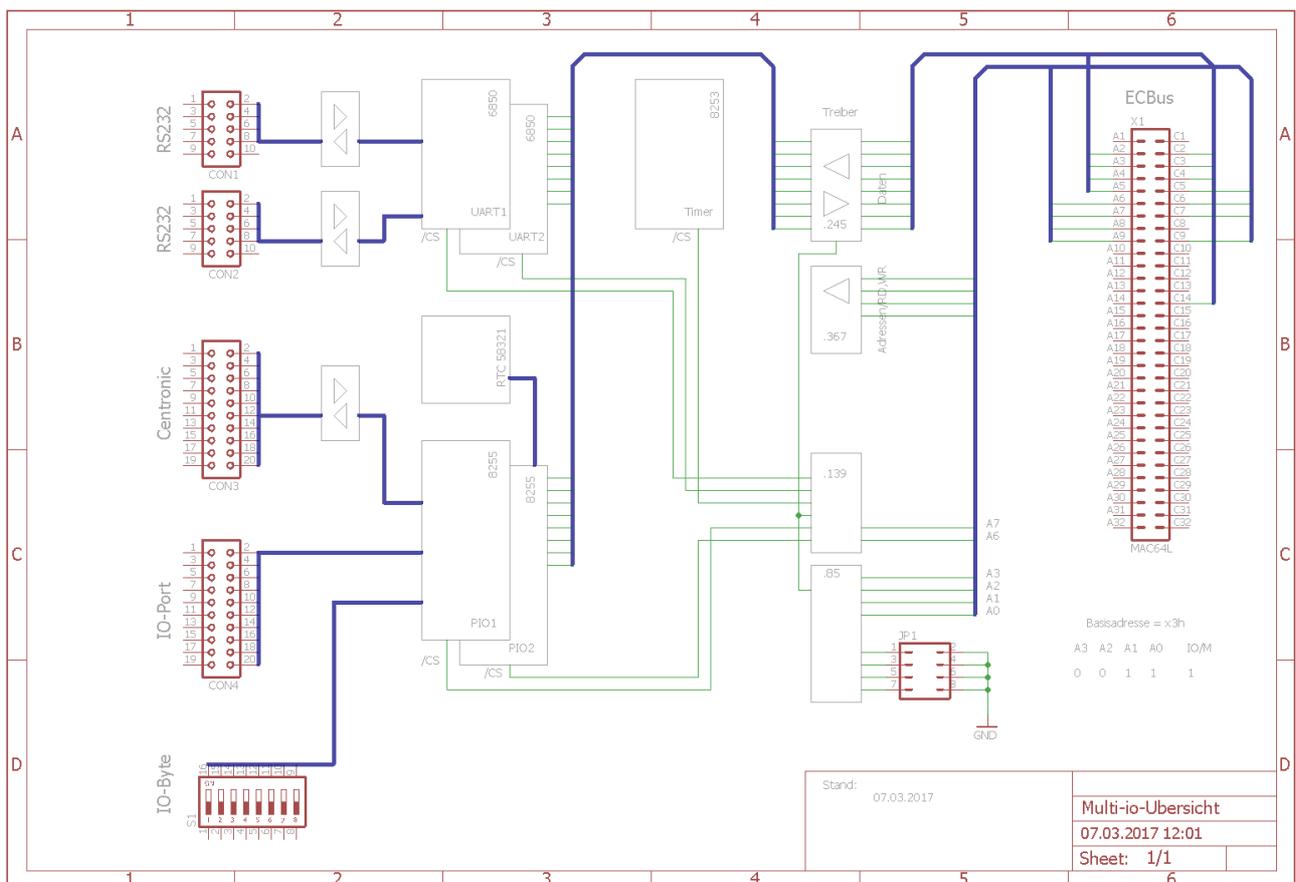
Diese Multi-IO-Karte beinhaltet:

- 2 Serielle Schnittstellen (V24) mit den Baudraten 300Bd bis 9600Bd
- 1 Parallele Schnittstelle (Centronic's)
- 1 Systemuhr mit dem Epson Baustein RTC-58321 und für Systemeinstellungen
- 8 DIP-Schalter die per Softwareabfrage die Systemkonfiguration einstellt.
- Die restlichen Ports sind für zukünftige Anwendungen auf Pfostenleisten geführt

# Hardware:

Die Schaltung gliedert sich in:

- 1 - ECB-Teil, BUS-Interface Adresssdecodierung
- 2 - PIO-Teil, Parallele Schnittstelle, Systemuhr, Konfigurationsschalter
- 3 - UART-Teil, 2x V24 als COM1 und COM2, wobei COM1 Interrupt fähig ist (RST6.5)

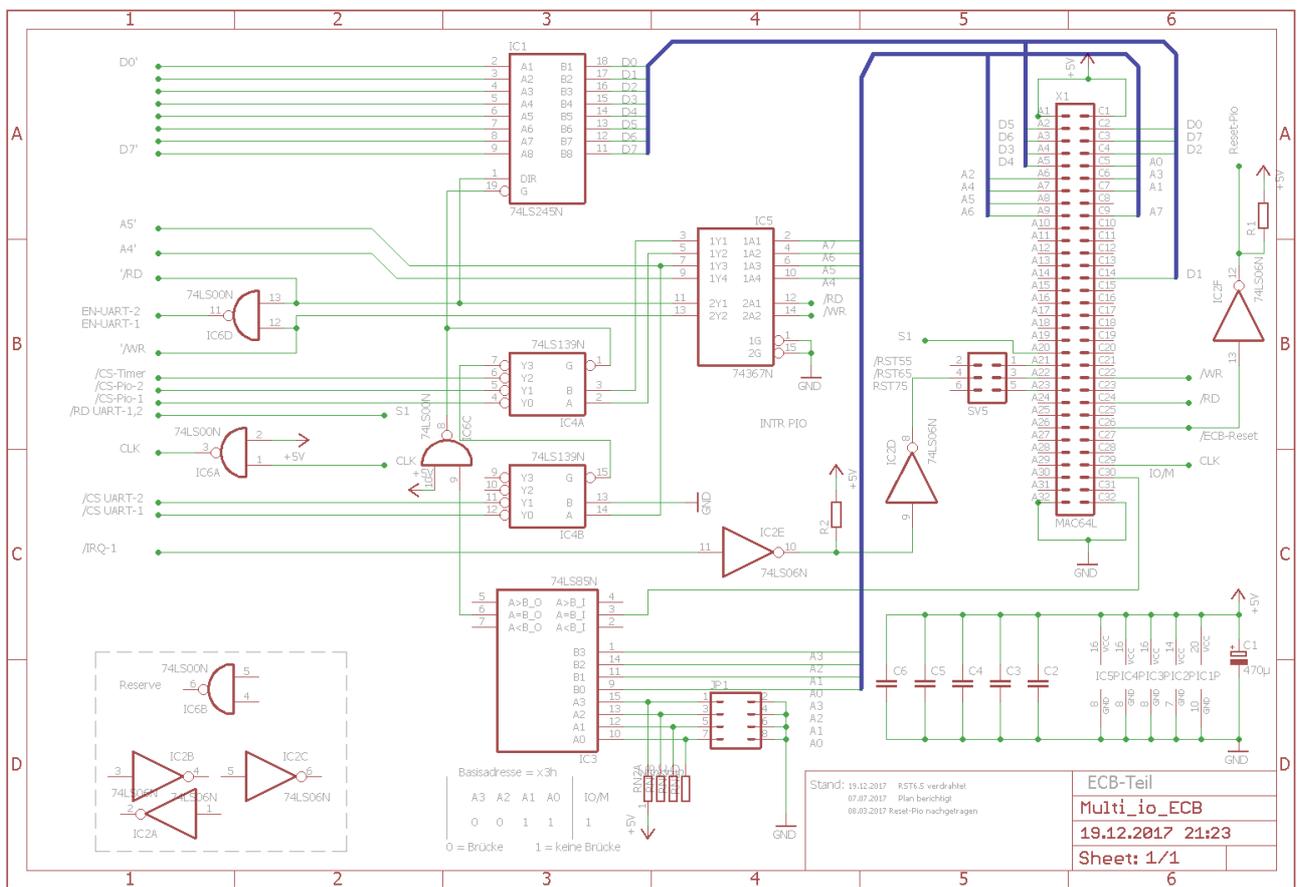


Übersichtsplan

# 1. ECB-Teil

neben den Datenbustreiber ermöglicht der 74LS85 die frei Adresswahl im IO-Bereich  
 Die Karte belegt 16 Adressen im IO-Bereich (03h bis F3h)

- PIO1 Centronic's und UNI-IO Kanal      Adr.: 03h bis 33h
- PIO2 IO-Byte und Systemuhr            Adr.: 43h bis 73h
- Timer Baudratenerzeugung             Adr.: 83h bis B3h
- COM1 V24                                    Adr.: C3h bis D3h
- COM2 V24                                    Adr.: E3h bis F3h



## 2. UART-Teil

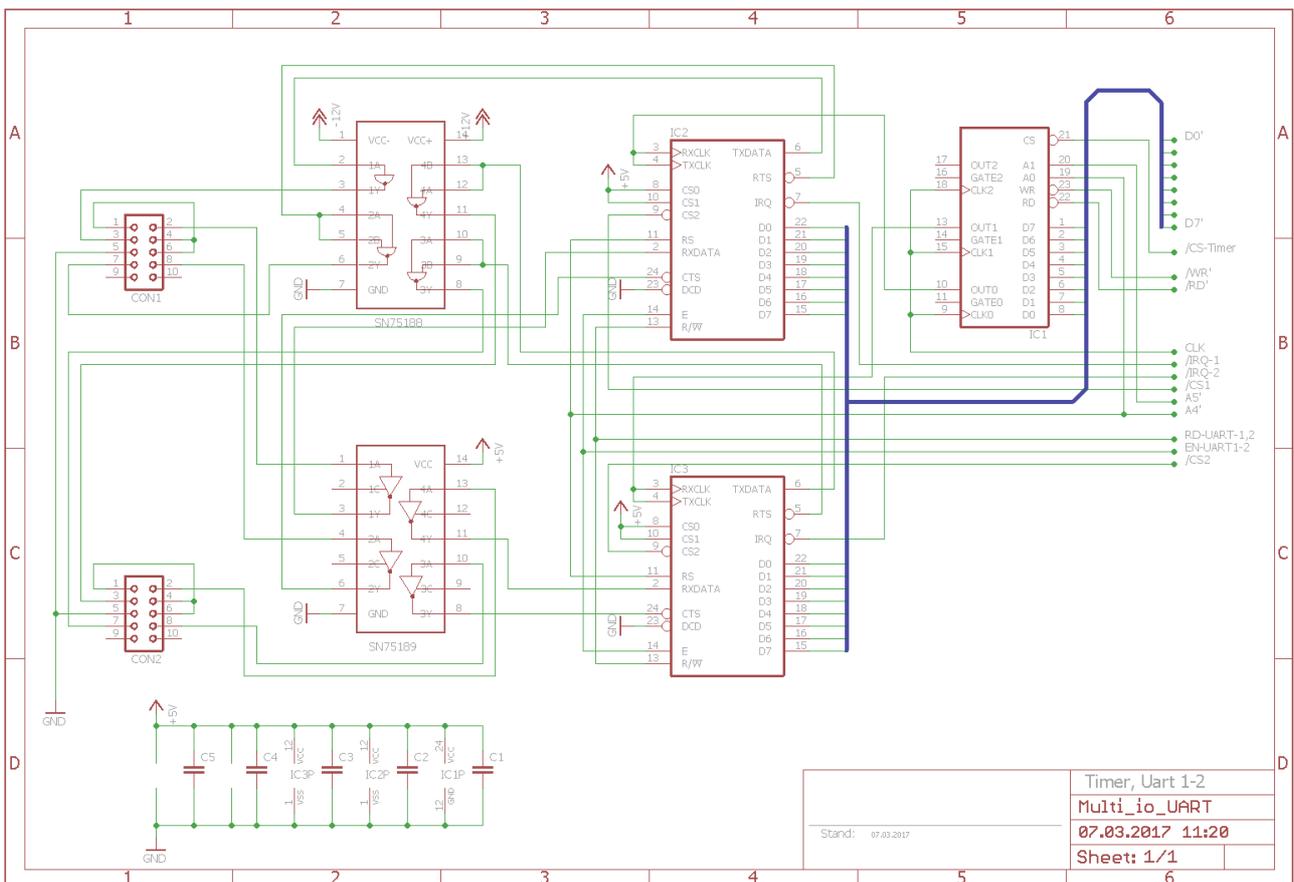
Der 8253 erzeugt aus dem Systemtakt die Baudraten für die beiden V24 Schnittstellen  
 Der 1.Timer für die COM1 und entsprechend der 2.Timer für die COM2, der dritte ist nicht belegt.

Als UART verwende ich hier die im Moppel üblichen 6850er Bausteine.

Über die V24Treiber SN75188 und SN75189 werden die Pegel auf  $\pm 12V$  angepasst.

Die stehen dann an zwei Pfostenleisten zur Verfügung.

Der IRQ-Ausgang vom UART1 (COM1) kann bei entsprechender Programmierung den RST6.5 ansteuern zB. Datenempfang von einem modernen PC mit FIFO.

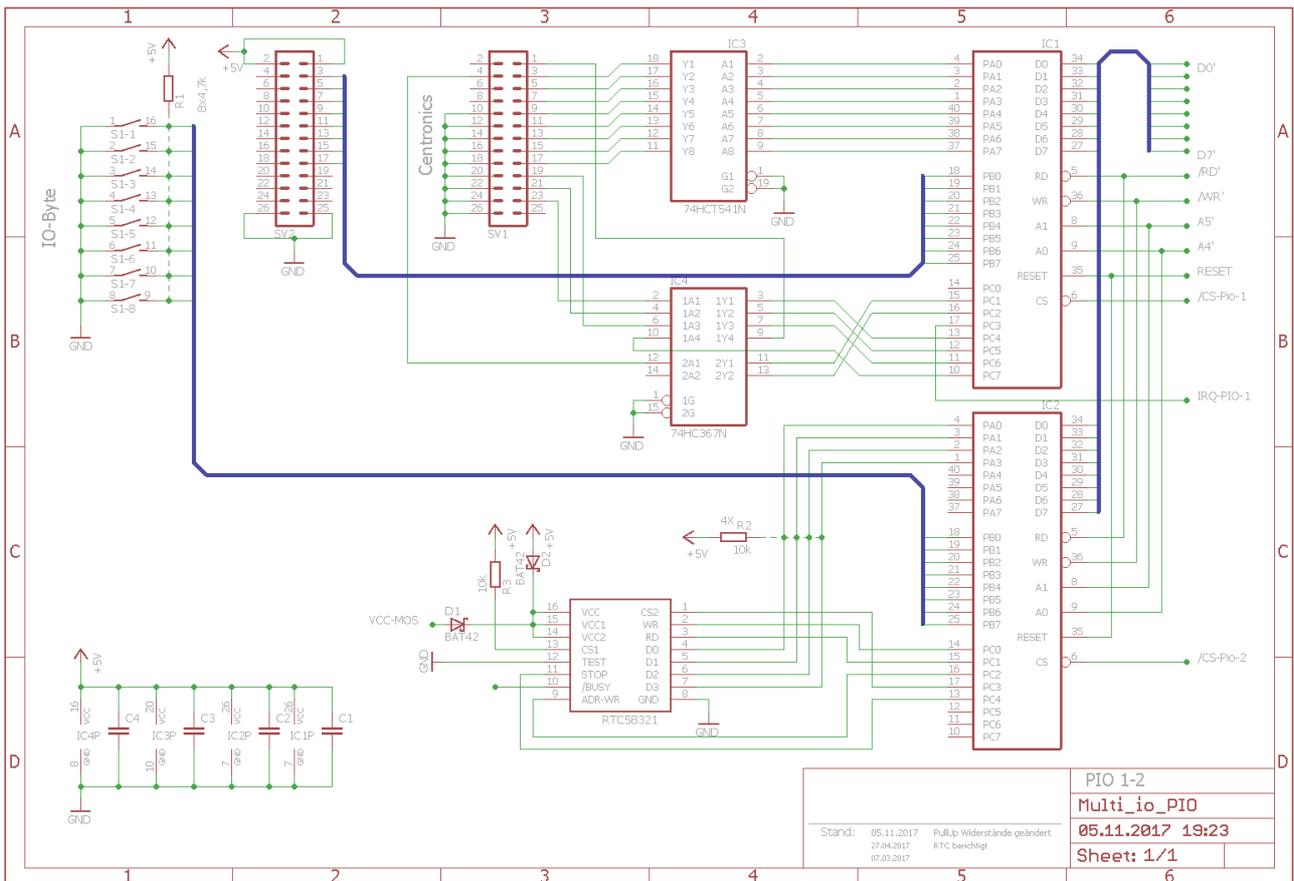


### 3. PIO-Teil

Hier sorgen die beiden 8255 mit den entsprechenden Treiberbausteinen 74HCT541 und 74HC367 für Centronics Schnittstelle.

Der Uhrenbaustein RTC58321 stellt die Systemuhr bereit und ist direkt am 2. PIO-Baustein angeschlossen. PortA Bit 0-3 für den Datenaustausch und PortC Bit 0-4 für die Steuerung.

Die DIP-Schalter sind wieder direkt am PortB angeschlossen. Diese können dann über Softwareabfrage als Konfigurationseinstellung dienen.



#### 4. COM1 und COM2 Steuer- und Statuswort

```

-----
; Multi-IO-Schnittstellenkarte
; Portbeschreibung: COM1/COM2
;
; Controlregister UART (aus Datenblatt Motorola M6850)
;
;   7 6 5 4 3 2 1 0
;   : : : : : : : :
;   : : : : : : 0 0 = clk/1
;   : : : : : : 0 1 = clk/16
;   : : : : : : 1 0 = clk/64
;   : : : : : : 1 1 = Reset
;   : : : 0 0 0 = ev 2Stop 7Datenbits
;   : : : 0 0 1 = od
;   : : : 0 1 0 = ev 1Stop (gerade Paritaet)
;   : : : 0 1 1 = od (ungerade Paritaet)
;   : : : 1 0 0 = -- 2Stop 8Datenbits
;   : : : 1 0 1 = -- 1Stop
;   : : : 1 1 0 = ev 1 Stop
;   : : : 1 1 1 = od
;   : 0 0 = RTS low TX-IRQ disable
;   : 0 1 = RTS low TX-IRQ enable
;   : 1 0 = RTS high TX-IRQ disable
;   : 1 1 = RTS low TX-IRQ disable
;   1 = RX-IRQ enable
;
;   0 0 0 0 0 0 1 1 = 03h Baustein Reset
;   0 0 0 1 0 1 0 1 = 15h clk/16, 8,n,1
;   0 1 0 1 0 1 0 1 = 55h ... mit RTS=1
;
;   1 0 0 1 0 1 0 1 = 95h clk/16, 8,n,1 ; RTS=0 RX-IRQ enable
;   1 1 0 1 0 1 0 1 = D5h "" ; RTS=1 ""
;
; Statusregister UART nur lesen
;
;   7 6 5 4 3 2 1 0
;   : : : : : : : :
;   : : : : : : : 1 = Empfangsregister voll
;   : : : : : : 1 = Senderegister leer
;   : : : : : 1 = DCD Eingang
;   : : : : x = CTS Eingang
;   : : : 1 = Fehlerbits -Frame
;   : : 1 = -Overrun
;   : 1 = -Paritaet
;   1 = IRQ-Ausgang (ungerade Paritaet)
-----

```

#### 4.1 Timer, COM1 und COM2 Adressen und Betriebsarten

```
-----  
tim0    equ    83h    ; BD-Rate fuer com1  
tim1    equ    93h    ; BD-Rate fuer com2  
tim2    equ    0a3h   ; frei  
tims    equ    0b3h   ; Steuerregister  
          .....  
          ; Timer0-2 als BCD-Zaehler, Rechteckgenerator,  
          ; erst Timewert-Low, dann Timewert-High laden  
bati0   equ    37h    ; 0011 0111  
bati1   equ    77h    ; 0111 0111  
bati2   equ    0b7h   ; 1011 0111  
;  
com1s   equ    0c3h   ; Steuerregister  
com1d   equ    0d3h   ; Daten com1  
bacom1  equ    15h    ; Betriebsart (8bit,1Stop,ohne Paritaet, Takt/16)  
;  
com2s   equ    0e3h   ; Steuerregister  
com2d   equ    0f3h   ; Daten com2  
bacom2  equ    15h    ; Betriebsart (8bit,1Stop,ohne Paritaet, Takt/16)  
;  
;  
-----
```

#### 4.2 Timer Steuer- und Statuswort

## 5. Belegung der PIO-Bausteine 8255

```

;-----
;
;
; PIO1 Centronics und UNI-IO Kanal
;
pio1a equ 03h ; Centronic's
pio1b equ 13h ; frei
pio1c equ 23h ; Steuerung Centronics
;
; Bit 7 6 5 4 3 2 1 0
; I I I I I I + + frei
; I I I I I I
; I I I I I + - - /Error <--
; I I I I + - - - INTRa -->
; I I I + - - - - Papierende <--
; I I + - - - - - BUSY <--
; I + - - - - - - /ACK <--
; + - - - - - - - /STROBE -->

pio1s equ 33h ; Steuerregister
bap1 equ 0abh ; 1010 1011
; Kanal A Modus 1 Ausgabe, Kanal B Modus 0 Eingabe
; Kanal C Eingabe

;
; PIO2 IO-Byte und Systemuhr
;
pio2a equ 43h ;
; RTC 58321 Daten
; Bit 7 6 5 4 3 2 1 0
; x x x x + + + + Daten 0-3 <->
; Bit 4-7 frei

pio2b equ 53h ; IO-byte DIP-Schalter <--

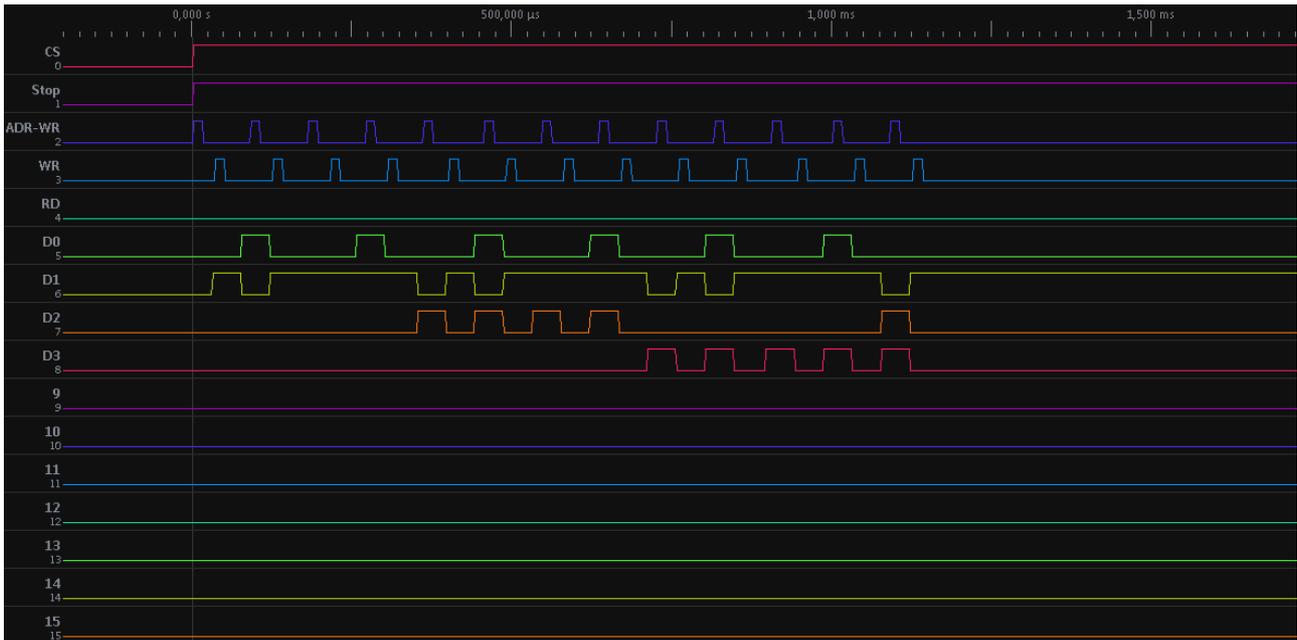
pio2c equ 63h ; RTC 58321 Control
; Bit 7 6 5 4 3 2 1 0
; x x x I I I I I
; x x x I I I I + WR -->
; x x x I I I + - RD -->
; x x x I I + - - AD/WR -->
; x x x I + - - - CS -->
; x x x + - - - - Stop -->
; Bit 5-7 frei -->

pio2s equ 73h ; Steuerregister
; Betriebsarten
bap2 equ 82h ; 1000 0010
; Kanal A Modus 0 Ausgabe
; Kanal B Modus 0 Eingabe
; Kanal C lo Ausgabe
; Kanal C hi Ausgabe

bap2rdu equ 92h ; 1001 0010
; Kanal A Modus 0 Eingabe
; Kanal B Modus 0 Eingabe
; Kanal C lo Ausgabe
; Kanal C hi Ausgabe
;
;-----

```

## 6. Echtzeituhr Signalanalyse



6.2

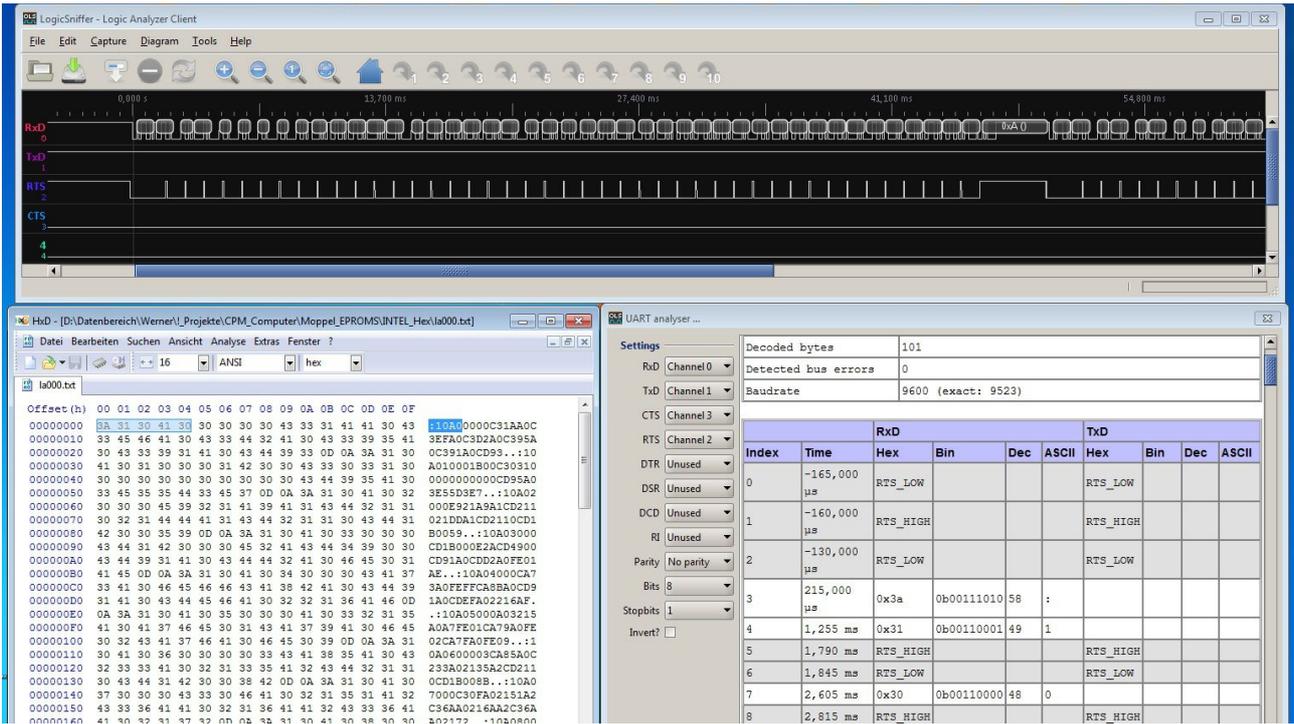
Signalverlauf RTC-58321 Uhr setzen



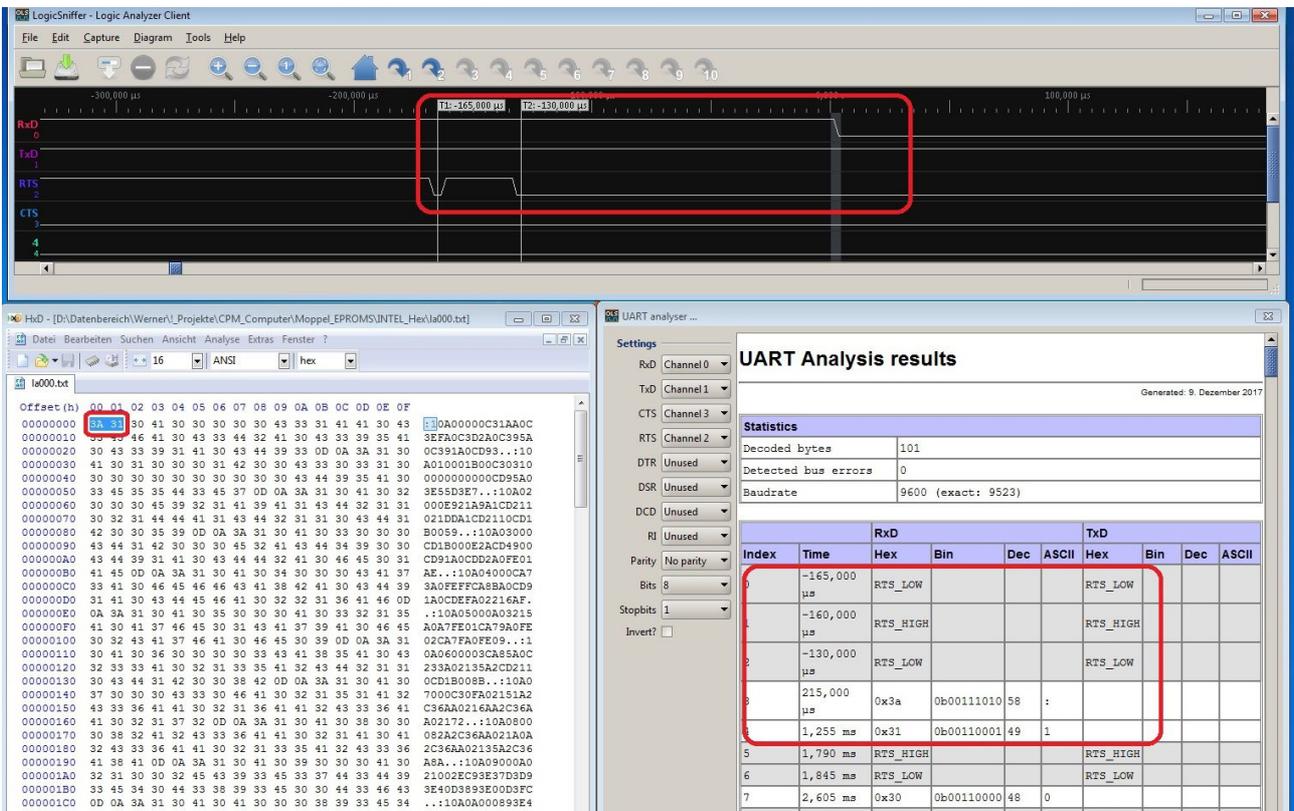
Signalverlauf RTC-58321 Uhr lesen

Hier sieht man deutlich am Signalverlauf für CS und Stop, dass bei der Modus-Umschaltung des Datenkanals von Aus- auf Eingabe die Steuerleitungen auf Low-Pegel wechseln, sowie die Datenleitungen über die PullUpWiderstände auf High-Pegel gezogen werden. Dies ist eine Eigenschaft der Portbausteine 8255, etwas unschön hat hier aber keinen Einfluss auf die Funktion.

## 7. V24 Signalanalyse

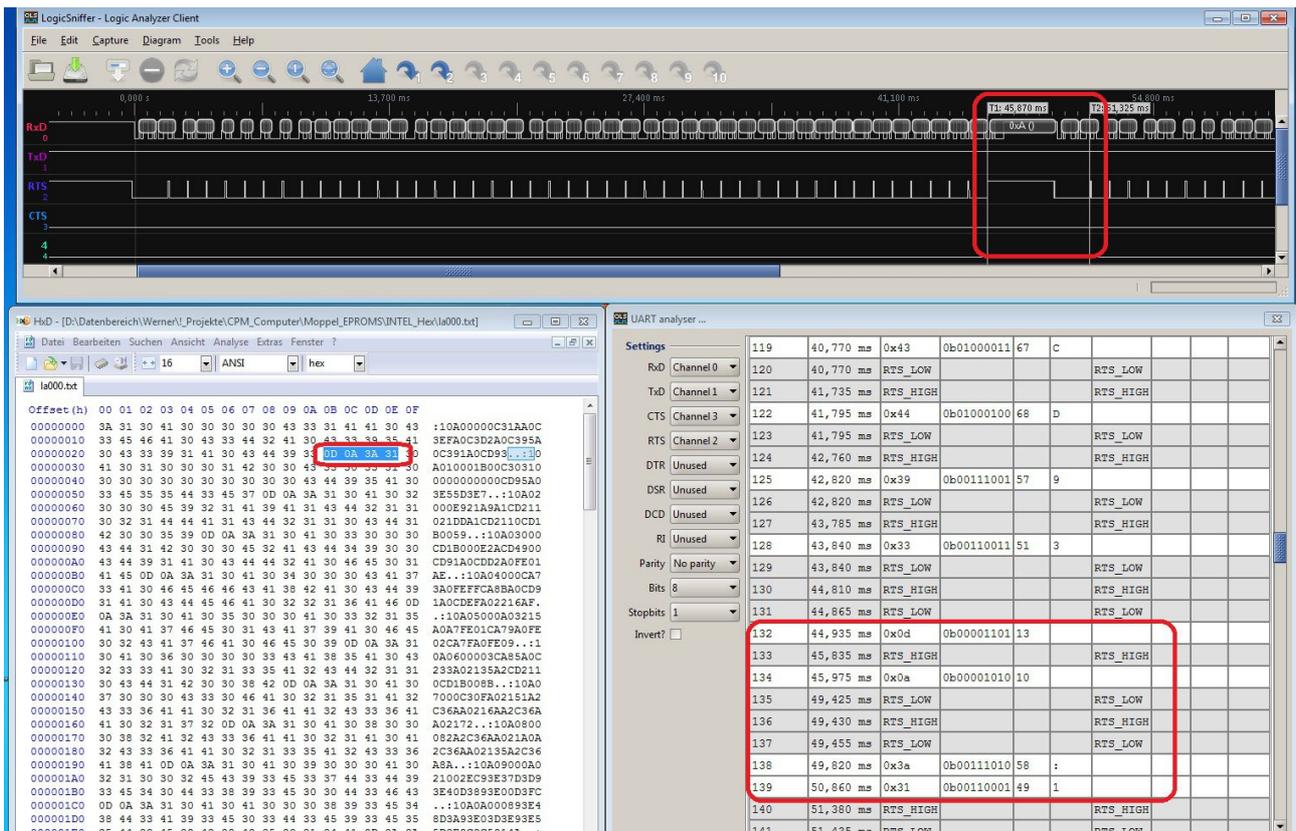


Signalverlauf, einlesen einer Intel-Hex Zeile

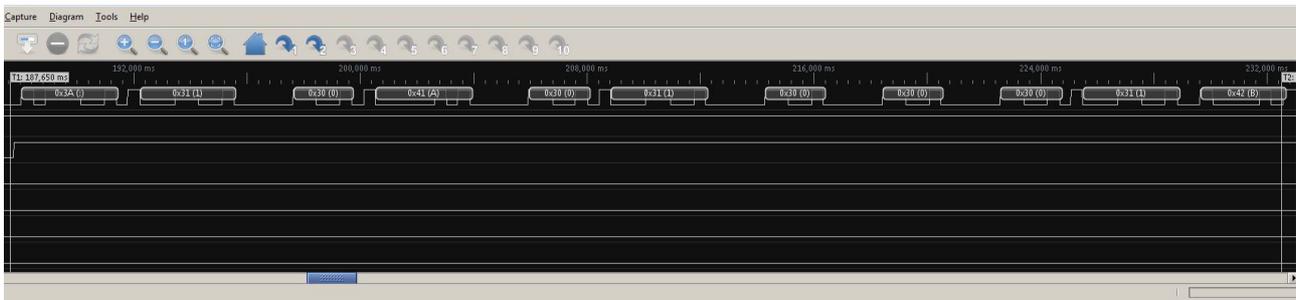


Startbedingung, RTS löschen und Startbit empfangen

## 7.2 V24 Signalanalyse Stopbedingung



RTS gesetzt, Daten aus dem Sende-FIFO werden noch abgesetzt



hier leert der Sender noch seinen kompletten FIFO obwohl RTS lange vom Moppel gesetzt ist.

Dieses Verhalten der modernen PC's halte ich für nicht Normgerecht, Halt heißt Halt – wenigstens nach meinen Vorstellungen. Da der „Bremsweg“ so lang ist, kann der Datenempfang nicht über das übliche Pollen der V24-Ports funktionieren. Hier muss im Moppel ein FIFO per Software und einer Interrupt-Service Routine nachgebildet werden.

## Software:

### 1. Initialisierung

Hier muss sichergestellt sein, dass die Bausteine mit den entsprechenden Startparameter wie Betriebsarten, Baudraten und Teilfaktoren versorgt werden.

### 2.1 RTC-Besonderheiten

Die Uhrzeit wird in den Registern des Bausteins als BCD-Zahl abgelegt

```
;
; RTC Registeradressen
; Adressen      Bedeutung                Gueltigkeit Datenbit 0-3
;
;      3 2 1 0
; 0    0 0 0 0 = Sekunden-Einer          0 - 9
; 1    0 0 0 1 = Sekunden-Zehner         0 - 5
; 2    0 0 1 0 = Minuten-Einer           0 - 9
; 3    0 0 1 1 = Minuten-Zehner          0 - 5
; 4    0 1 0 0 = Stunden-Einer            0 - 9
; 5    0 1 0 1 = Stunden-Zehner          0 - 1 / 0 - 2 im 24Std-Modus
; 6    0 1 1 0 = Wochentag                0 - 6
; 7    0 1 1 1 = Tag-Einer                0 - 9
; 8    1 0 0 0 = Tag-Zehner                0 - 3 (D2 = 1 fuer Februar mit 29Tagen/sonst 28Tage)
; 9    1 0 0 1 = Monat-Einer              0 - 9
; A    1 0 1 0 = Monat-Zehner             0 - 1
; B    1 0 1 1 = Jahr-Einer                0 - 9
; C    1 1 0 0 = Jahr-Zehner              0 - 9
;
```

Damit das Uhrenprogramm einfacher zu lesen ist, habe ich für die Ports ein zweites Label zugewiesen und die Steuerworte für CS/WR/RD und ADRWR festgelegt.

```
;
; Portzuweisung (neues Label fuer die Systemuhrabfrage)
;
rtc_ctrl equ    pio2c
rtc_dat  equ    pio2a
;
;                                ; 7654 3210
;
rtccs   equ    18h                ; 0001 1000    = RTC Selektieren und Zaehler-Stop
rtcwr   equ    19h                ; 0001 1001    = RTC Datennibble schreiben
rtcrd   equ    1ah                ; 0001 1010    = RTC Datennibble lesen
rtcawr  equ    1ch                ; 0001 1100    = RTC Registeradresse schreiben
;
```

## 2.2 RTC-Registeradressen

```
;
; RTC Registeradressen
; Adressen      Bedeutung          Gueltigkeit Datenbit 0-3
;
;      3 2 1 0
; 0   0 0 0 0 = Sekunden-Einer     0 - 9
; 1   0 0 0 1 = Sekunden-Zehner    0 - 5
; 2   0 0 1 0 = Minuten-Einer      0 - 9
; 3   0 0 1 1 = Minuten-Zehner     0 - 5
; 4   0 1 0 0 = Stunden-Einer       0 - 9
; 5   0 1 0 1 = Stunden-Zehner     0 - 1 / 0 - 2 im 24Std-Modus
; 6   0 1 1 0 = Wochentag          0 - 6
; 7   0 1 1 1 = Tag-Einer          0 - 9
; 8   1 0 0 0 = Tag-Zehner         0 - 3 (D2 = 1 fuer Februar mit 29Tagen/sonst 28Tage)
; 9   1 0 0 1 = Monat-Einer        0 - 9
; A   1 0 1 0 = Monat-Zehner       0 - 1
; B   1 0 1 1 = Jahr-Einer         0 - 9
; C   1 1 0 0 = Jahr-Zehner        0 - 9
;
;
```

## 2.3 Umsetzung der Wochentage

```
;
; Wochentage ..oMiDiMoDrFaSoS
; 1 = Montag, 7 = Sonntag
; Tage sind in umgekehrter Reihenfolge abgelegt
; weil der ASCII-Buffer von hinten nach vorn
; aufgebaut wird
;
wt_tab: DB 52h          ; "R"
        DB 45h          ; "E" - Error
        DB 6Fh          ; "o" -
        DB 4Dh          ; "M" - Montag
        DB 69h          ; "i"
        DB 44h          ; "D"
        DB 69h          ; "i"
        DB 4Dh          ; "M"
        DB 6Fh          ; "o"
        DB 44h          ; "D"
        DB 72h          ; "r"
        DB 46h          ; "F"
        DB 61h          ; "a"
        DB 53h          ; "S"
        DB 6Fh          ; "o"
        DB 53h          ; "S"
;
;
```

## 2.4 RTC-Buffer

Hier gibt es einmal den „**uhrbuf**“ um die BCD-Daten aufzunehmen und ist genauso aufgebaut wie die Register im RTC. Zusätzlich gibt es einen ASCII-Buffer „**ascbuf**“ um den String für die Uhrzeit aufzunehmen „So,05.11.17;22:55:10h“ Dieser wird von hinten nach vorn aufgebaut, das vereinfacht die Programmierung.

### 3. V24 Details

Im reinen IRQ-Betrieb kann er Daten mit 9600Bd an einem Stück entgegennehmen. Da diese jedoch je nach Anwendung beim Eintreffen analysiert und entsprechend verarbeitet werden müssen benötigt man einen Eingangspuffer (FIFO).

```
;
; Parameter zur Übersicht
;
; buffer      equ 9000h      ; fuer BIN-Datei
; zbuff      equ 8400h      ; Software FIFO
;
;           v-Schreibzeiger (wrz)
;           I
; 8400h I-----+-----I 843fh (maxbuf)
;           ^
;           I- Lesezeiger (rdz)           842ah +----- RTS Setzen (rtsbuf)
;
; maxbuf      equ 3fh      ; Bufferlaenge
; rtsbuf      equ 2ah      ; Bufferposition fuer RTS
;
; buffz      dw 0000h      ; Schreibzeiger fuer Zielbuffer
; wrz        dw 0000h      ; Schreibzeiger fuer IRQ
; rdz        dw 0000h      ; Lesezeiger
; v24flag    db 00h        ; 1000 0001
;
;           ;
;           ;           +-- RTS punkt erreicht
;           ;           +----- Buffer voll
;
```

Er ist als einfacher Linearer Buffer konzipiert, Es werden solange Zeichen eingeschrieben bis RTS gesetzt wird und auch der sendende seinen FIFO hier abgeladen hat. Die Leseroutine, also die Schaufel die die Daten zum Anwendungsprogramm bringt läuft dem Schreibzeiger immer etwas hinterher. Bei Gleichstand sind dann einfach keine neuen Daten vorhanden und wartet bis der Timeout von ca. 2s abgelaufen ist. Wenn innerhalb dieser Zeit nichts mehr eintrifft, in der Regel wenn alle Daten empfangen wurden, wird das Programm beendet.

Der Buffer kann natürlich individuell angepasst werden max 256Byte, da die Zeiger nur auf das niedrige Byte geprüft werden und er muss daher sich auch innerhalb einer Speicherseite befinden.

Die Position RTSBUF sollte sich mindestens 16Bytes vor MAXBUF befinden, damit RTS rechtzeitig gesetzt wird. Bei einigen PC's kann der FIFO nicht abgeschaltet werden und wenn doch, sendet er trotz RTS ein bis zwei Bytes – siehe Signalanalyse.

Das ganze könnte natürlich auch als Ringbuffer programmiert werden, dazu müssen die beiden Zeiger aber aufwendiger überwacht und verglichen werden. Sobald der Schreibzeiger am Ende wieder nach vorn springt passt der Vergleich Lesezeiger < Schreibzeiger nicht mehr, das war mir im Moment zu komplex und diese einfache Lösung macht es ja auch recht gut.