RS232-DALI - Gateway

DALI (Digitally Addressable Lighting Interface) ist ein System zur Ansteuerung von Leuchten und wird im Bereich Hausautomation eingesetzt.

Es gibt inzwischen erschwingliche DALI-Leuchten und auch Schaltnetzteile mit Strombegrenzung zur Ansteuerung von LED's und integrierter DALI-Schnittstelle. Damit habe ich gearbeitet.

Das Projekt besteht aus 2 Teilen:

Der Hardware (ATTiny2313), programmiert in C und einem User-Interface, realisiert mit Python / TKinter.

1.) Hardware + C-Programm

Ein ATTiny2313 fungiert als Gateway zwischen serieller Schnittstelle und dem DALI-Bus.
Der Controller ist C programmiert, der Flash des Controllers ist zu 50% genutzt.

Der Controller wird über die serielle Schnittstelle mit Datensätzen gefüttert. Das kann manuell erfolgen über ein Terminalprogramm oder über eine beliebige Software, die in der Lage ist, die serielle Schnittstelle zu bedienen.

Die Datensätze bestehen aus einer Folge von genau 4 Bytes:

1. Byte = Startbyte
2. Byte = DALI-Byte 1
3. Byte = DALI-Byte 2
4. Byte = Cmd (1x senden, 2x senden, das Timing der DALI-Bit-Flanken zurückliefern)

Die beiden DALI-Bytes müssen entsprechend der DALI-Vorgaben fertig formatiert sein, sie werden unverändert auf den Dali-Bus geschickt.

Nach Ablauf der Sende- und Wartezeit liefert der Controller eine Antwort zurück - auch wenn die Antwort häufig nur lautet, dass keine Antwort empfangen wurde.

Die Rückgabe besteht aus genau 3 Byte:

1. Byte = Startbyte (0xA5)
2. Byte = Statusbyte (zeigt an, ob gültige Daten eingegangen sind oder ob ein Fehler erkannt wurde)
3. Byte = Datenbyte (ist nur gültig, sofern das Statusbyte dies signalisiert)

2.) User Interface in Python

Die manuellen Eingabe über ein Terminalprogramm ist möglich, aber etwas mühsam und fehlerträchtig. Außerdem ist das DALI-Command-Set recht umfangreich, ein Datenblatt neben der Tastatur ist Pflicht.

So ist ein ein Tool mit grafischer Oberfläche entstanden, das die manuelle Eingabe komfortabler gestaltet.

Aufwändigere Aktionen wie etwa das Durchsuchen des DALI-Busses nach antwortenden Geräten, das Durchsuchen nach RandomAdressen sowie die Vergabe von ShortAdressen sind damit per Knopfdruck möglich.

Alle Aktionen, die häufiger benötigt werden, können direkt per Button-Click ausführt werden.

Trotz alledem ist dieses Programm nur zum Testen und zur Konfiguration gedacht, nicht zur praktischen Steuerung eines DALI-Systems.

Weitergehende Beschreibungen und einen Schaltplan gibt's in der DALI_readme_151001.pdf,
den C-Code für einen ATtiny2313 in "DALI_c_151001.zip",
die Python-Dateien in "DALI_python_151001.zip".

ATtiny2313 als RS232-DALI Gateway

Hardware

Als Controller wird ein ATtiny2313 eingesetzt, alternativ kann auch der AT90S2313 verwendet werden (es müssen aber einige Register umbenannt werden). Das Programm belegt ca. 50% des Flash.
Ein 3.686.400 MHz Quarz ist erforderlich.

Der Schaltplan des während der Tests aufgebaute DALI-Netzteil, des Sende- und Empfangsteils mit den Optokopplern sowie des Controllers ist als Anhang beigefügt.

Verwendet habe ich durchweg Bauteile, die zufälligerweise in der Bastelkiste auf ihren Einsatz warteten.

Funktionsweise

Der Controller verharrt im Sleep-Modus solange, bis ein Byte via Serielle Schnittstelle empfangen wird.
Ein RX-Complete Interrupt beendet daraufhin den Sleep-Modus.

Die RX-ISR prüft, ob das erste Byte ein Startbyte (0xA5) ist. Wenn nicht, dann wird das Byte verworfen.

Wenn doch, dann ist ein erstes Byte empfangen worden, drei weitere werden noch erwartet.
Bei einer Baud-Rate von 9600 wird das ca. 3ms dauern.

Nun wird der Timer0 gestartet, ein Overflow wird nach 4.44ms eintreten.

In einer Schleife wird geprüft, ob 4 Byte eingegangen sind (dann ist das Flag 'rx_flag' gesetzt) oder ob ein Timer0_OVL eingetreten ist (dann ist das Flag 'timer_ovl' gesetzt).

Ist ein Overflow eingetreten, dann wird eine Fehlermeldung gesendet, der Empfangspuffer gelöscht und wieder in den Sleep-Modus zurückgekehrt (denn es sind nicht die erwarteten 3 weiteren Bytes eingegangen).

Wenn insgesamt 4 Byte empfangen sind, dann wird geprüft, ob das 4. Byte eines der 3 vereinbarten Kommandos enthält.

Wenn nicht, dann wird eine Fehlermeldung zurückgeliefert und wieder in den Sleep-Modus übergegangen.

Entspricht das Format des Datensatz den Erwartungen, dann werden die beiden DALI-Bytes inklusive eines Start- und der beiden Stopbits auf dem DALI-Bus gesendet.

Anschließend wird geprüft, ob der DALI-Slave eine Antwort zurückliefert.

Dazu wird der Timer1 gestartet. Sein Preload ist so gewählt, dass nach 9.17ms ein Overflow eintritt.
Ein Overflow setzt das Flag 'timer_ovl'.

Nun wird das Flag 'timer_ovl' und der Pegel des DALI-Signals gepollt.

Wird ein timer_ovl erkannt, dann ist keine Antwort eingegangen, der Controller sendet eine entsprechende Meldung (0xA5, 0x01, 0x00) zurück.

Wird dagegen ein Flankenwechsel erkannt, dann stammt der von Startbit der eingehenden Antwort.

In diesem Fall wird der Timer0 zur Kontrolle der Bitzeiten und der Timer1 zur Kontrolle der Bytedauer gestartet.

Bei jedem Flankenwechsel wird der Zählerstand des Timer0 ausgelesen und es wird entschieden, ob das Timing einer halben Bitzeit oder zu einer ganzen Bitzeit zugeordnet werden kann.

Liegt ein Flankenwechsel bei einer ganzen Bitzeit vor, dann wird das anliegende Bit gespeichert und der Counter0 zurückgesetzt.

Der Empfang wird beendet, wenn ein timer_ovl auftritt (nach 9.17ms).
Das kann ein Overflow des Timer0 sein, dann liegt ein Fehler im Bit-Timing vor (0xA5, 0x02, 0x00).

Ein overflow des Timer1 ereignet sich 9.17ms nach der fallenden Flanke des Startbit.

In dieser Zeit sollten 1 Startbit, 8 Datenbit und 2 Stopbit eingegangen sein. (Naja, die Stopbits gibt es im eigentlichen Sinne ja gar nicht.)

Die Datenbits werden gezählt. Sind nach dem Timer1_OVL genau 8 Bit gezählt worden, dann war die Übertragung fehlerfrei. Weniger oder mehr Bits deuten auf einen Fehler hin (0xA5, 0x03, 0x00).

Der Controller liefert auf jeden empfangenen Datensatz eine Rückmeldung.

Jetzt sollte noch der zeitliche Abstand von 9.17ms eingehalten werden, bevor eine nächste DALI-Übertragung gestartet wird.

Wurde von der Seriellen Schnittstelle im Hintergrund ein neuer Datensatz empfangen, dann wird der nun abgearbeitet.

Ist Empfangspuffer leer, dann geht der Controller wieder in den Sleep-Modus über, aus dem er nur durch einen RX-Complete Interrupt geweckt wird.

Die Kommunikation mit dem Gateway läuft nach folgendem Muster ab:

Das Gateway erwartet genau 4 Byte, wobei das 1. Byte und das 4. Byte geprüft werden:

```
1.Byte = Startbyte (0xA5)
2.Byte = DALI-Byte 1 (DALI-Adresse)
3.Byte = DALI-Byte 2 (Arc / DALI-Command)
4.Byte = cmd
```

Die beiden DALI-Bytes müssen entsprechend des DALI-Protokolls fertig "konfiguriert" sein, sie werden unverändert auf dem DALI-Bus ausgegeben.

Für das 4. Byte gibt drei Optionen:

```
0xA1 = die beiden DALI-Bytes 1x senden
0xA2 = die beiden DALI-Bytes 2x nacheinander senden
0x81 = zur Fehlersuche nur das Timing der Antwort auswerten
```

Das Gateway liefert zu jedem empfangenen Datensatz eine Antwort zurück.
Sie besteht immer aus 3 Byte:

```
1. Byte = Startbyte (0xA5)
2. Byte = Status der Übertragung/ Fehlercode
3. Byte = Datenbyte oder Platzhalter

Antwort auf die Reaktion des DALI-Slaves

0xA5, 0x00, Datenbyte    es wurde ein Datenbyte vom Slave empfangen
0xA5, 0x01, 0x00         der Slave hat nicht geantwortet
0xA5, 0x02, 0x00         Fehler, Bitzeit überschritten
0xA5, 0x03, 0x00         Fehler, Framedauer überschritten
0xA5, 0x04, 0x00         Fehler, Bittakt außerhalb der definierten Grenzen

Antwort bei einem Fehler im Datensatz an das Gateway:

0xA5, 0x10, 0x00         es wurden nicht genau 4 Byte erkannt.
0xA5, 0x11, 0x00         das Byte 4 konnte nicht zugeordnet werden.
```

Wird als 4. Byte ein '0x81' an den Controller gesendet, dann wird zum Debuggen das Timing der vom DALI-Slave empfangenen Bits zurückgeliefert. Die Werte sind in in ein Paket von 23 Byte verpackt.
Zunächst eine Transaktion mit 0xA1 (= 1x senden) als 4. Byte

```
gesendet: A5, FE, 00, A1
          empfangen: A5, 01, 00
```

gesendet: Shortadresse 63, direct arc 0 = Licht aus, Status = 1, keine Antwort

Nun das gleiche Kommando, jedoch mit 0x81 als 4. Byte, das Timing wird geliefert

```
gesendet: A5, FE, 00, 81
          empfangen: 00,04,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
```

Overflow von Timer1 (TCNT1 = 0x0004), keine Daten empfangen
Der Wert von ca. 0x0004 für TCNT1 entsteht, wenn der Timer1 überläuft, also keine Antwort empfangen wird. Bis zum Abschalten/Auslesen des Timer1 sind noch einige Takte abzuarbeiten, darum steht der Zähler nicht auf 0x0000.

Im nächsten Beispiel liefert der DALI-Slave eine Antwort zurück

```
gesendet: A5, FF, 99, A1
          empfangen: A5, 00, 06
```

gesendet: ShortAdresse: 63, Query Device Type, Antwort: Status = 0, Antwort = 6

Das Timing dazu:

```
gesendet: A5, FF, 99, 81
          empfangen: 07,3D,18,30,17,2F,17,2F,17,2F,31,18,31,30,00,00,00,00,00,00,00
```

(TCNT1 - Preload) = 0x073D
Byte 0 und 1 (highbyte / lowbyte) liefern den Zählerstand des Timer1 zurück

(Anmerkung: der Controller hat den Timer1-Preload dabei bereits berücksichtigt / subtrahiert) .

Mit dem Faktor 8/3.686.400 multipliziert erhält man die Zeit, die zwischen dem Versenden des letzten Stopbits bis zur Erkennung der Startflanke des Startbits vergangen ist.

Das bedeutet: vom Stopbit bis zum Startbit des Slaves sind 4.02ms vergangen, danach jeweils 833us / 416us pro Takt / Halbtakt für die Datenbits.

Die "Sollwerte" für die Bits sind 24 (0x18) für einen halben Bittakt (416µs), 48 (0x30) für einen ganzen Bittakt (833µs).

Die Bitzeit in Sekunden kann berechnet werden durch Multiplikation mit dem Faktor 64 / 3.686.400

Die Kommunikation mit dem Gateway erfolgt immer sequenziell, zuerst wird ein Datensatz an das Gateway gesendet und anschließend auf die Antwort des Gateways gewartet.

Danach darf die nächste Aktion gestartet werden.

Auf diese Weise wird sichergestellt, dass das Gateway nicht "überfahren" wird und die im DALI-Protokoll definierte Wartezeit (9.17ms) eingehalten wird.

Python-Tool als GUI

Das Python-Tool bietet die Option, die an den DALI-Bus zu versendenden beiden Bytes in einfacher Form manuell zu versenden oder gar vorbereitete Scripte (genaugenommen Methoden) per ButtonClick abarbeiten zu lassen.
Die korrekte Formatierung der beiden DALI-Bytes übernimmt die Software.

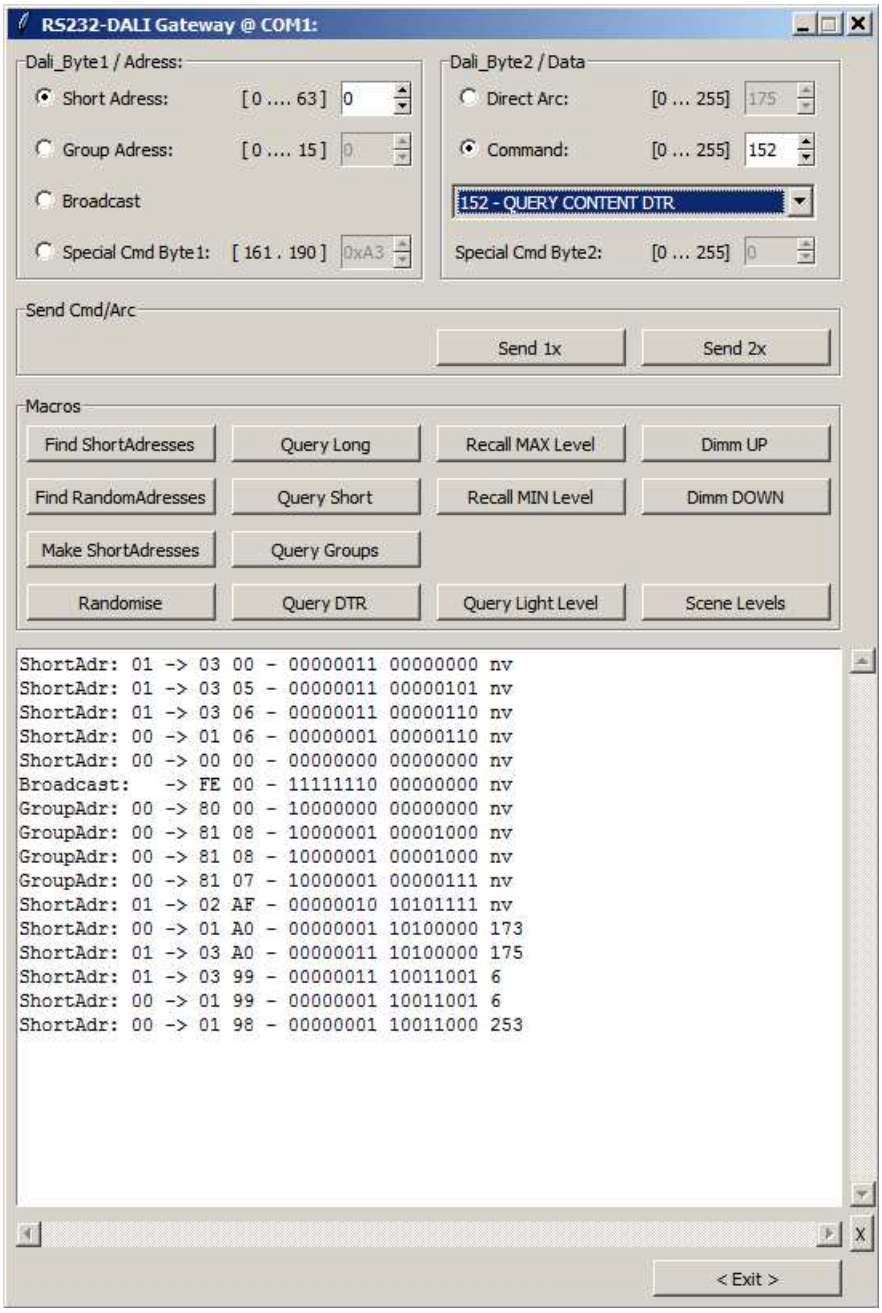
Dazu werden die Einstellungen der Radiobuttons und der Entry-Felder in den beiden Frames Dali_Byte1 / Adress und Dali_Byte 2 / Cmd interpretiert.

Das DALI_Byte 1 dient zur Adressierung mittels ShortAdressen, Gruppenadressen, Broadcasts oder Special Commands.

Das Dali-Byte2 definiert den einzustellenden Helligkeitswert bzw. überträgt das codierte Command.

Da die Liste der Commands umfangreich ist, kann anstelle der manuellen, numerischen Eingabe eine Auswahl in der Combo-Box getroffen werden, der zugehörige Code wird in das Entry-Feld übertragen.

Alle Entry-Felder nehmen nur positive numerische Eingaben dezimal oder hexadezimal entgegen.
Eine hexadezimale Eingabe muss mit '0x' beginnen. Der Wertebereich der Eingaben wird ebenfalls geprüft.



Hinweis: bei einer hexadezimalen Eingaben funktionieren die Spin-Buttons und Cursor_Up/ Cursor_Down nicht.

Werden durch Eingaben in den beiden Frames Daten manuell an das Gateway übertragen, dann erfolgt zur Kontrolle eine Ausgabe der gesendeten Bytes (und Bits) im Editorfenster.

Ebenso wird die empfangene Antwort ausgegeben.

'nv' bedeutet, dass keine Antwort empfangen wurde.

Hier einige Beispiele:

```
ShortAdr: 63 -> 7E FE - 01111110 11111110 nv
ShortAdr: 62 -> 7D 05 - 01111101 00000101 nv
ShortAdr: 62 -> 7D 99 - 01111101 10011001 6
ShortAdr: 62 -> 7D A0 - 01111101 10100000 254
GroupAdr: 00 -> 81 06 - 10000001 00000110 nv
Broadcast:   -> FF 05 - 11111111 00000101 nv
```

Die Zeilen sagen uns:

An Adresse 63 wird "Direct Arc" 254 gesendet, es folgt keine Antwort
An Adresse 62 wird "Recall Max Level" gesendet, es folgt keine Antwort
An Adresse 62 wird "Query Device Type" gesendet, die Antwort lautet '6'
An Adresse 62 wird "Query Actual Level" gesendet , die Antwort lautet '254' (= Max Level)
An die Group 00 wird "Recall Min Level" gesendet, es folgt keine Antwort
Ein Broadcast mit "Recall Max Level" wird gesendet, es folgt keine Anwort

Das Editor-Fenster kann gelöscht werden, indem man den Button 'X' rechts unten im Kreuzungspunkt der Scrollbars anklickt. Copy und Paste aus dem Editorfenster ist ebenfalls möglich.

Eine Reihe von häufiger benötigte Commands sind per "Script" auf Buttons gelegt.

Zusätzlich gibt es einige Scripte, die umfangreichere Arbeiten ausführen, z.B.:
- Alle Querys zu einer ShortAdresse auslesen
- den Adressraum 0 ... 63 nach antwortenden Shortadressen durchsuchen
- den DALI-Bus nach allen RandomAdressen durchsuchen
- DALI-Geräten eine ShortAdresse zuweisen (Commissioning)
- Leuchten 5x Blinken lassen

Zum Commissioning (der Vergabe von ShortAdressen) sind einige Erläuterungen notwendig.

Um jungfräulichen DALI-Geräten, die bereits verbaut, aber dadurch nicht individuell ansprechbar sind, ShortAdressen zu vergeben, ist folgendes Verfahren vorgesehen.

1.) Randomise

Hier werden allen angeschlossenen Geräten (hoffentlich) zufällige 3 Byte Adressen zugeteilt.
Neue Geräte scheinen die Adresse 0xFF, 0xFF, 0xFF zu besitzen, ohne Randomising sind die also nicht zu unterscheiden.

2.) Read RandomAdr

Hier wird in numerisch aufsteigender Folge nach allen RandomAdressen gesucht, die Adressen werden angezeigt (zusammen mit ggf. bereits vorhandenen Shortadressen)

Das Ergebnis der Suche kann per Option in eine Datei geschrieben werden ("random_export.cfg").
Diese Datei enthält zeilenweise die gefundenen Randomadressen und die Shortadresse.

Beispiel für "random_export.cfg":

```
084,064,211,063
118,170,147,062
```

Die im obigen Beispiel gefunden 2 Devices hatten bereits zu einem früheren Zeitpunkt die ShortAdressen 62 und 63 zugewiesen bekommen.

3.) Import-Datei erzeugen

Die Datei "random_export.cfg" wird kopiert auf den Name "short_import.cfg".
Nun wird, getrennt durch ein Komma, hinter jeden Datensatz als 5. Feld die gewünschte Adresse in dezimaler Schreibweise angehängt.

Beispiel für "short_import.cfg":

```
084,064,211,063,001
118,170,147,062,000
```

Die Adressen sollen geändert werden: 63 -> 1, 62 -> 0.

Soll eine ShortAdresse unverändert bleiben, dann kann eine negative Adresse angegeben werden.

4.) Commissioning

Wird die Datei "short_import.cfg" gefunden, dann wird die Option angeboten, die neuen ShortAdressen aus dieser Datei zu übernehmen.

Beim nachfolgenden Suchlauf werden nun automatisch die Shortadressen entsprechend der Angaben im 5. Feld geändert.
Zwingende Voraussetzung ist allerdings, dass alle Geräte und in der gleichen Reihenfolge gefunden und anhand ihrer RandomAdr identifiziert werden können.
Ein neues Randomising, das Abschalten von einzelnen Devices oder das Ändern der Zeilenfolge in der Konfigurationsdatei führt dazu, dass ab dem ersten aufgetretenen Fehler keine Änderungen mehr durchgeführt werden.

Wird die Datei "short_import.cfg" nicht gefunden oder der Import aus dieser Datei abgewählt, dann gibt es die noch die beiden anderen Möglichkeiten, die Devices

- in aufsteigender Folge ihrer Randomadressen mit Adressen aufsteigend ab Adresse 0 oder
- in aufsteigender Folge ihrer Randomadressen mit Adressen absteigend ab Adresse 63

automatisch umzunummerieren.

Auf diese Weise erhält jedes DALI-Device eine eindeutige ShortAdresse.

Theoretisch könnte man danach die Shortadressen individuell korrigieren:

- Es wird die gewünschte ShortAdresse ins DTR geschrieben
- Anschließend führt man das Kommando "STORE DTR AS SHORTADR" 2x aus.

Das hat bei mir allerdings nicht funktioniert.

Michael S.

Quellen zu DALI-Protokoll und DALI Command Set:

NXP Semiconductors - AN10760 - USB-DALI master using the LCP2141
Microchip - AN 1465 - Digitally Addressable Lighting Interface (DALI) Communication
Microchip - AN 1487 - DALI Control Gear

Nachtrag

Bevor ich meinen DALI-Master in ein bestehendes DALI-System einbinde, möchte ich gerne sehen, was am Bus 'geplaudert' wird - und natürlich, ob meine Implementierung die dort ablaufende Kommunikation verstehen kann.

Dazu musste ein Logger her, der am Bus lauscht und die Informationen mitschneidet.

Der erste Gedanke war, einen eigenständigen Logger zu bauen.

Auf der anderen Seite war der verwendete Controller nur zu 50% ausgelastet, so dass der Gedanke nahe lag, Logger und Master auf ein und demselben Controller zu installieren.

Ein Jumper legt fest, in welchem Modus der Controller startet, ob als Master oder als Logger.
Bei nicht gesetztem Jumper wird der Master-Modus aktiviert.

Zwei LED's zeigen an, in welchem Modus gearbeitet wird.
Um den Modus zu ändern muss der Jumper gesetzt/gezogen und ein Reset ausgeführt werden.

Das C-Programm geht vom Einsatz eines AT90S2313 aus.
Um einen ATtiny2313 zu verwenden sind einige Registernamen zu ändern.

Am Programmcode für den DALI-Master wurden keine relevanten Veränderungen vorgenommen, einige veraltete Kommentare wurden korrigiert.

In der Python-GUI wurden separate Frames für das Versenden und für die "Makros" eingeführt.

Logger-Modus

Der Logger verharrt (genauso wie der Master) normalerweise im Sleep-Modus.

Eine fallende Flanke am DALI-Bus löst einen INT0-Interupt aus, der Sleep-Modus wird beendet.

Nun startet zunächst die Funktion, die die 16 Bit des Forward-Frame einliest.

Danach wird dieselbe Funktion aufgerufen, die auch beim Master auf die Antwort eines Slaves wartet.

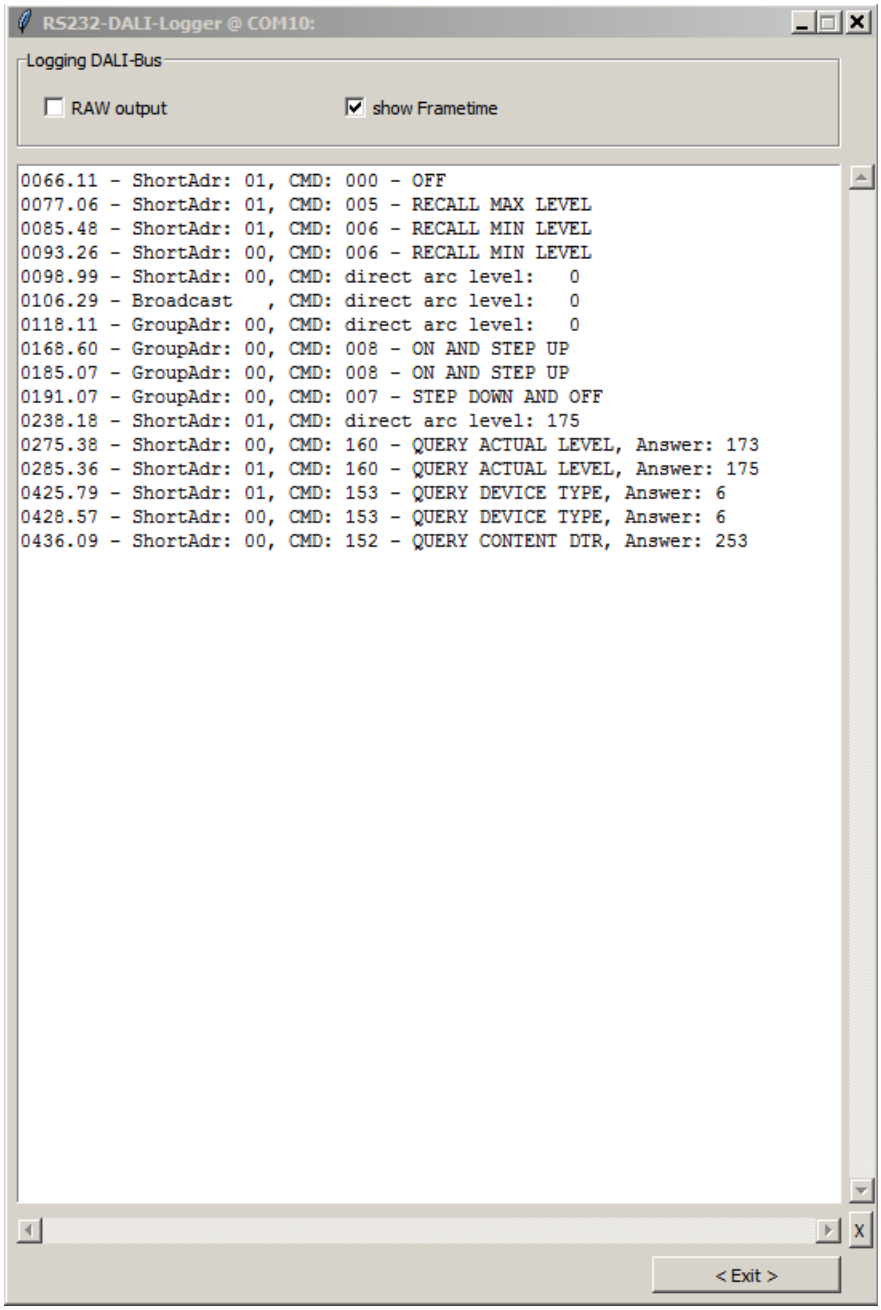
Am Ende werden 6 Byte über die Serielle Schnittstelle gesendet:

```
1. Byte = Startbyte (0xA5)
2. Byte = Anzahl der folgenden Bytes (hier immer = 4)
3. Byte = Forward Frame, high Byte (= DALI-Adr)
4. Byte = Forward Frame, low Byte (= DALI-Cmd)
5. Byte = Backward Frame (oder 0, falls kein Frame empfangen wurde)
6. Byte = StatusByte

          Das StatusByte sagt aus:
          0 = kein Fehler, Foreward Frame und Backward Frame sind o.k.
          1 = kein Fehler, es wurde kein Backward Frame empfangen
          2 = Fehler, Bitzeit überschritten
          3 = Fehler, Framedauer überschritten
          4 = Fehler, Bittakt außerhalb der definierten Grenzen
```

Die Interpretation des Foreward Frame über nimmt auf dem PC ein Python Script.
Hier werden entweder die empfangenen Daten im RAW-Format (so wie vom Controller gesendet) oder in einer decodierten Form im "Klartext" ausgegeben.
Bei Bedarf kann der Ausgabe ein Zeitstempel vorangestellt werden, der die Anzahl der Sekunden seit Programmstart angibt.

Die Interpretation des Backward Frame (Answer: xxx) bleibt dem Nutzer überlassen.

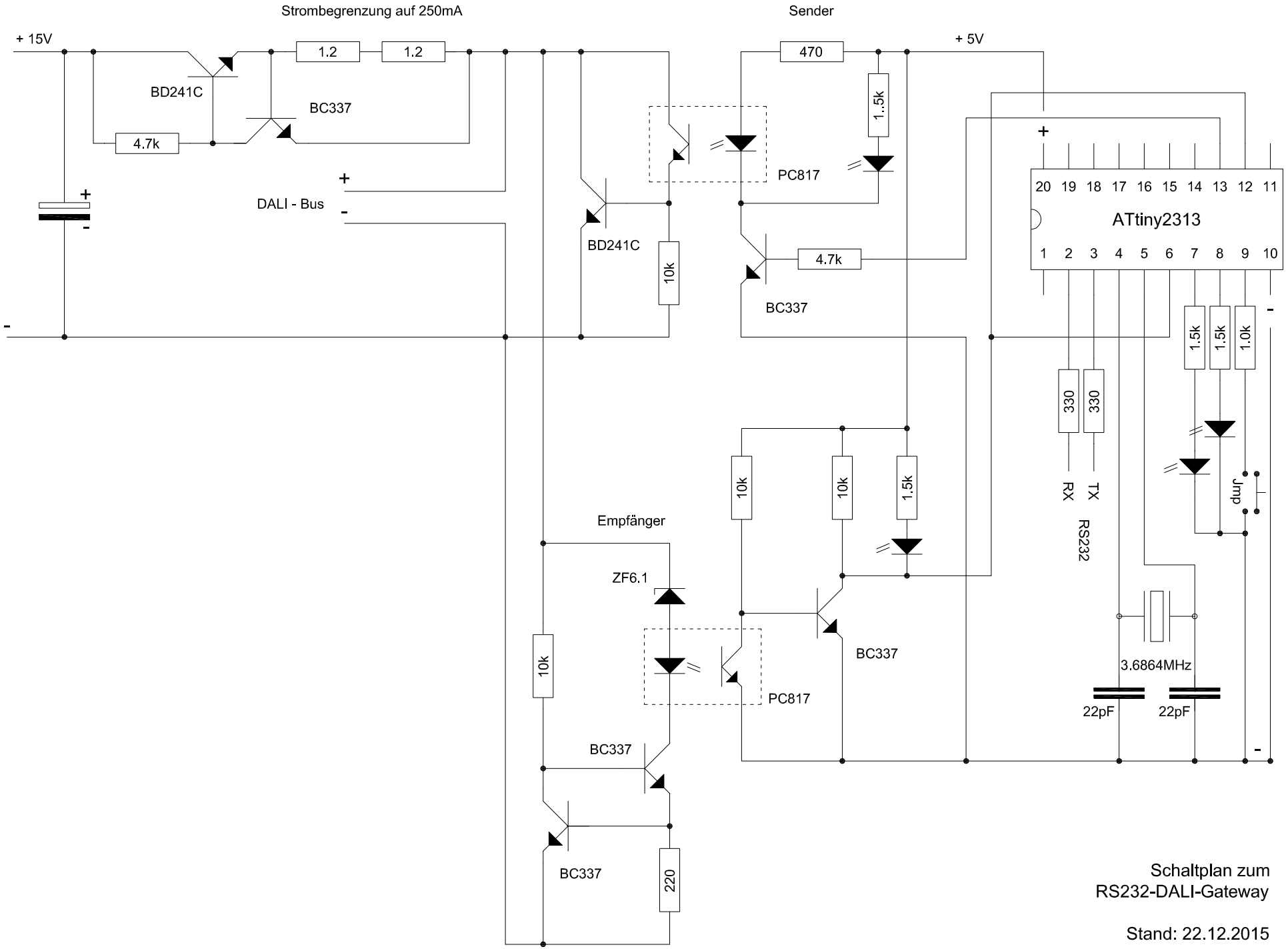


23.12.2015

Michael S.

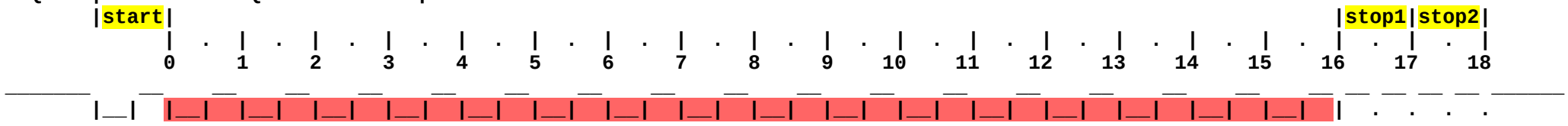
Als Anlage folgen noch ein aktualisierter Schaltplan (für den Logger muss der DALI-Input zusätzlich mit INT0 verbunden werden; der Modus Master/Logger muss über einen Jumper gewählt werden).
Ohne gesetzten Jumper arbeitet der Controller als Master.

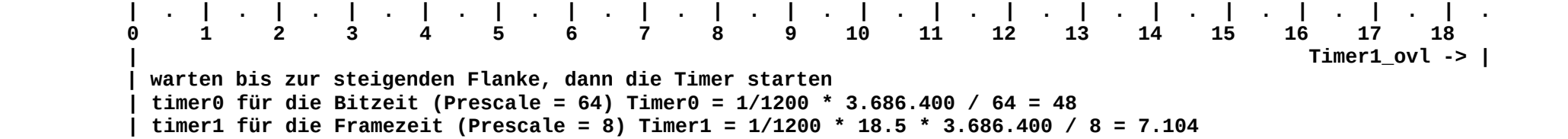
Außerdem ist eine zusätzliches Blatt beigefügt, auf dem das Timing der DALI-Kommunikation / der DALI-Bits sowie der Timer des Controllers visualisiert ist.


Strombegrenzung auf 250mA
Sender
+ 15V
+ 5V
BD241C
BC337
1.2
1.2
4.7k
DALI - Bus
+
-
BD241C
10k
PC817
470
1..5k
4.7k
BC337
ATtiny2313
20 19 18 17 16 15 14 13 12 11
1 2 3 4 5 6 7 8 9 10
1.5k
1.5k
1.0k
330
330
RX
TX
RS232
jmp
10k
10k
1.5k
Empfänger
ZF6.1
10k
PC817
BC337
3.6864MHz
22pF
22pF
BC337
BC337
220
Schaltplan zum
RS232-DALI-Gateway
Stand: 22.12.2015

DALI-LOGGER, DALI-Bit-Timing und Timer0 / Timer1

FORWARD FRAME





| warten bis zur steigenden Flanke, dann die Timer starten
| timer0 für die Bitzeit (Prescale = 64) Timer0 = 1/1200 * 3.686.400 / 64 = 48
| timer1 für die Framezeit (Prescale = 8) Timer1 = 1/1200 * 18.5 * 3.686.400 / 8 = 7.104

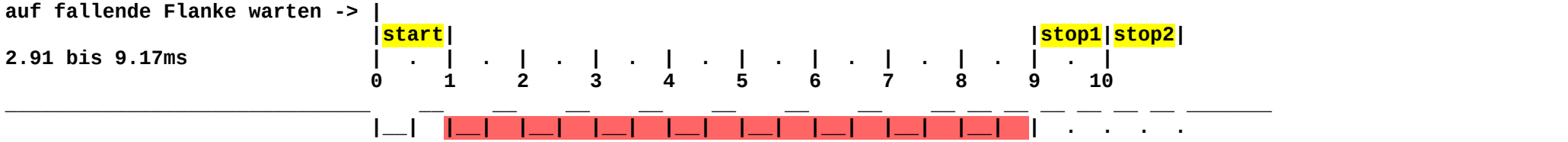
ABSTAND zum nächsten FORWARD FRAME >22TE = 1/2400 * 22 = 9.17ms

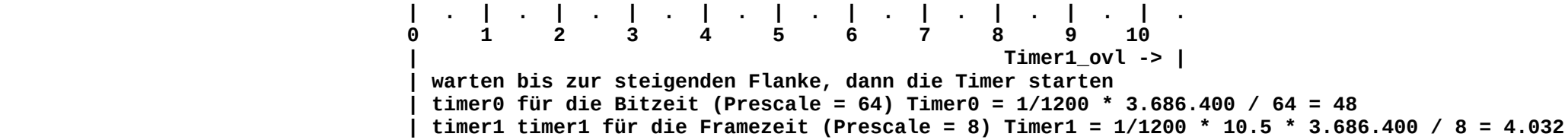
BACKWARD FRAME

Zeitlicher Abstand zum Forward Frame: >= 7TE < 22TE (>= 2.91ms < 9.17ms)

auf fallende Flanke warten ->

2.91 bis 9.17ms





| warten bis zur steigenden Flanke, dann die Timer starten
| timer0 für die Bitzeit (Prescale = 64) Timer0 = 1/1200 * 3.686.400 / 64 = 48
| timer1 timer1 für die Framezeit (Prescale = 8) Timer1 = 1/1200 * 10.5 * 3.686.400 / 8 = 4.032

ABSTAND zum nächsten FORWARD FRAME >22TE = 1/2400 * 22 = 9.17ms

Quelle: Microchip AN1465, Digitally Addressable Interface (DALI) Communication