ebusd

(ein Daemon für den eBus)

v0.1

von

Roland Jax

Inhaltsverzeichnis

ebus - ebusd

ebus steht für Energie Bus und wird in Heizungssystemen eingesetzt. Ein serieller Bus dient dabei zum Austausch von Nachrichten zwischen den einzelnen Bus Teilnehmern. Es gibt sowohl aktive (Master) als auch passive (Slave) Teilnehmer am Bus. Ein Teilnehmer kann gleichzeitig Master und Slave sein.

Über einen ebus-Adapter (usb, lan) tritt der **ebusd** (ebus Daemon) als zusätzlicher, aktiver Teilnehmer am ebus auf. Er liest dabei die laufende Kommunikation mit und kann den Inhalt definierter, zyklischer Nachrichten für die weiter Verarbeitung zwischenspeichern. Über eine TCP Socket Verbindung (telnet) können dem Daemon Befehle (ebus Befehle in lesbarer Form) zur aktiven Ausführung gesendet werden. Nach Abarbeitung werden dem Absender die gewünschten Daten rückgemeldet. Zur Zeit nimmt der Daemon selbst keine Befehle vom ebus entgegen.

In Sachen ebus Befehle kocht jeder Heizungshersteller sein eigenes Süppchen. Der Daemon benötigt deshalb für jeden Heizungshersteller abgestimmte Konfigurationsdateien in einem bestimmten Format.

Features

daemon

- Logfile mit Logging Klassen
- Aufzeichnung der ebus Nachrichten im RAW Format
- Konfiguration über Datei bzw. Kommandozeilen Optionen
- Einlesen ein oder mehrerer ebus Kommandodateien im CSV Format

ebus – serielle Kommunikation auf dem ebus

- Senden von Broadcast (BC), Master-Master (MM) und Master-Slave (MS) Nachrichten
- CRC Berechnung für gesendete und empfangen Daten
- Entfernen nicht erlaubter Byte Sequenzen im ebus Datenstrom
- Erkennen von Buskollisionen und Sendungswiederholung bei missglückten Sendeversuchen
- Zwischenspeichern von Daten zyklischer ebus Nachrichten

telnet – Netzwerk Schnittstelle zur Interaktion

- Multiuser fähig
- frei definierbarer TCP Port (default 8888)
- Scriptfähig

Build

Für das Bauen sind die GNU Autotools notwendig. Bei fehlen einer notwendigen Komponente wird eine dementsprechende Fehlermeldung ausgegeben

Der Sourcecode kann direkt aus dem SVN Repository bezogen werden.

```
$ svn checkout https://openautomation.svn.sourceforge.net/svnroot/openautomation/tools/ebusd
```

Im Sourcecode Verzeichnis befindet sich ein Script zum automatischen kompilieren.

```
$ cd ebusd
$ ./autogen.sh
```

Diese Skript versucht die folgenden Befehle in der angegebenen Reihenfolge abzuarbeiten.

```
$ aclocal
$ autoconf
$ autoheader
$ automake
$ ./configure
$ make
```

Daemon Konfiguration

Der Daemon ist mit einer Default Konfiguration ausgestattet. In der Konfigurationsdatei ebusd.conf können alle Optionen überschrieben werden. Mit den Kommandozeilen Optionen können nur die wichtigsten Einstellungen verändert werden.

Priorität der möglichen Konfiguration:

eingebaute Konfiguration < ebusd.conf < Kommandozeilen Option

Beim Starten des Daemons versucht dieser zuerst die Konfigurationsdatei im Verzeichnis /etc/ebusd/ zu öffnen. Ist diese nicht vorhanden wird im Verzeichnis des Daemons nach der Konfigurationsdatei gesucht.

Über die möglichen Daemon Konfigurationen gibt die Konfigurationsdatei ebusd.conf selbst bzw. der Daemon mit dem Aufruf ./ebusd -h aus. Kommentare sind mit einem # (Hash) zu kennzeichnen.

Optionen der Kommandozeile

```
$ ./ebusd -h

Usage: ebusd [OPTIONS]
  -a --address          bus address (0xFF)
  -c --cfgfdir          configuration directory of command files (/etc/ebusd)
  -C --cfgfile          daemon configuration file (/etc/ebusd/ebusd.conf)
  -d --device           serial device (/dev/ttyUSB0)
  -e --extension        extension of command files (csv)
  -f --foreground       run in foreground
  -l --loglevel         log level (INF | INF, NOT, WAR, ERR, DBG, EBH, EBS, NET, ALL)
  -L --logfile          log file (/var/log/ebusd.log)
  -P --pidfile          pid file (/var/run/ebusd.pid)
  -p --port             port (8888)
  -r --rawdump          dump raw ebus data to file
  -R --rawfile          raw file (/tmp/ebusd.bin)
  -s --showraw          print raw data
  -S --settings         print daemon settings
  -v --version          print version information
  -h --help             print this message
```

Optionen in der Konfigurationsdatei

```
$ cat ebusd.conf

# configuration file for ebusd

# bus address (FF)
address=FF

# configuration directory of command files (/etc/ebusd)
cfgdir=/etc/ebusd

# serial device (/dev/ttyUSB0)
device=/dev/ttyUSB0

# extension of command files (csv)
#extension=csv

# run in foreground (NO | YES/NO)
foreground=NO

# log level (INF | INF, NOT, WAR, ERR, DBG, EBH, EBS, NET, ALL)
loglevel=INF

# log file (/var/log/ebusd.log)
logfile=/var/log/ebusd.log

# pid file (/var/run/ebusd.pid)
pidfile=/var/run/ebusd.pid

# port (8888)
port=8888

# dump raw ebus data to file (NO | YES/NO)
rawdump=NO

# raw file (/tmp/ebusd.bin)
awfile=/tmp/ebusd.bin

# print raw data (NO | YES/NO)
showraw=NO

# print daemon settings (NO | YES/NO)
settings=NO

# retry getting bus (3 | max: 10)
get_retry=3

# skipped ACK bytes after get-bus error (1)
skip_ack=1

# wait time for QQ compare (4000 uesc)
max_wait=4000

# retry sending msg (2 | max: 10)
send_retry=2

# number of printed hex bytes (30)
print_size=30
```

Die Konfigurationsdatei ebusd.conf befindet sich im Unterverzeichnis contrib.

Kommando Konfiguration

Der Aufbau einer csv Datei folgt der im Daemon verwendeten Strukturen.

Als Feldtrennzeichen wird das Semikolon ; verwendet. Der Daemon verlangt in jeder Spalte einen Wert. Falls kein Wert vorhanden dient der Bindestrich – als Füllzeichen.

Jeder Befehl besteht aus einem Nachrichten Kopf und einem variablen Teil (Wiederholgruppe). Im Kopf sind dabei alle Daten, welche für die Verarbeitung der Nachricht notwendig sind. In der anschließenden Wiederholgruppe werden die Nutzdaten definiert.

Nachrichten Kopf

Typ	Name	Beschreibung	Beispiele
char [3]	type	Typ des Befehls	get, set, cyc
char [20]	class	Klasse	ci, amu, cir2, mv
char [30]	cmd	Befehl	password, holiday_period, at_temp
char [256]	com	Beschreibung des Befehls	-
int	s_type	ebus Nachrichten Typ	1 (Broadcast) 2 (Master-Master) 3 (Master-Slave)
char [2]	s_zz	Adresse des Slave	(hex) 08, 15
char [4]	s_cmd	Primär und Sekundär ebus Befehl	(hex) B509
int	s_len	Anzahl Datenbytes des Befehls	3
char [32]	s_msg	Datenbytes des Befehls	(hex) 0D2C00
int	d_len	Anzahl der Elemente der Wiederholgruppe	4

Wiederholgruppe

Für jedes Element der Wiederholgruppe gilt folgender Aufbau.

Typ	Name	Beschreibung	Beispiele
char [20]	d_sub	Unterbefehl	pin1, temp, stat,..
char [2]	d_part	Quelle des Datenbytes	MD (MasterDaten), SA (SlaveACK) SD (SlaveDaten), MA (MasterACK)
char [10]	d_pos	Position der Datenbytes	1,2 oder 9,7,6
char [3]	d_type	Datentyp	asc, bcd, d2c,..
float	d_fac	Skalierungsfaktor	1.0, 0.02
char [6]	d_unit	Einheit des Wertes	°C, bar,...
char [30]	d_valid	Mögliche Werte (noch nicht Implementiert)	(hex) 01,02,03,...
char [256]	d_com	Beschreibung des Unterbefehl	-

Datentypen

Für die Dekodierung bzw. Kodierung stehen folgende Datentypen zur Verfügung.

De	Ko	Typ [3]	Beschreibung
X	X	asc	ascii
X	X	bcd	bcd
X	X	d1b	data1b
X	X	d1c	data1c
X	X	d2b	data2b
X	X	d2c	data2c
X		bda	date (bcd Format)
X	X	hda	date (hex Format)
X		bti	time (bcd Format)
X	X	hti	time (hex Format)
X		bdy	day (bcd Format)
X	X	hdy	day (hex Format)
X		hex	2-nibble hex Format z.B: 2C

Eine Muster Kommandodatei (commands.csv) befindet sich im Unterverzeichnis contrib/csv.

telnet Schnittstelle

Die Verbindung wird mit einem telnet Client hergestellt.

daemon spezifisch

- loglevel → ändert den Logging Level zur Laufzeit
- quit → beendet die eigene Verbindung mit dem Daemon
- shutdown → stoppt den Daemon und trennt die Verbindung

ebus spezifisch

- get → sendet einen Befehl zum Lesen eines Heizungsparameter
- set → sendet einen Befehl zum Setzen eines Heizungsparameter
- cyc → fordert Werte für einen zyklisch aufgezeichneten Befehl an

Der Daemon erwartet für die Befehle get, set und cyc (type) noch die Klasse (class), den Befehl (cmd) und wenn notwendig einen Unterbefehl. Somit ergibt sich folgender Aufbau.

type class cmd sub

- Zwischen class, cmd und sub darf auch ein Punkt verwendet werden.
- Die Groß-/Kleinschreibung wird für class, cmd und sub ignoriert.

Beispiel: Befehl für Ermittlung der Außentemperatur: cyc broad date_time_temp temp

```
$ telnet localhost 8888
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
cyc broad date_time_temp temp
  5.00
quit
Connection closed by foreign host.
```

Sonstiges

Im Unterverzeichnis tools befinden sich 3 Hilfsprogramme.

- **check** nimmt interaktiv Datenbytes im hex Format entgegen und gibt diese in die ebus Standarddatentypen aus.

```
$ ./check
Input: c2 00
hex c2 → bcd: 255        d1b:  -62        d1c:  97.0
hex 00 → bcd:   0        d1b:    0        d1c:   0.0        d2b:    0.758      d2c:    12.1250    crc: 6e
Input: q
```

- **csv** liest im angegebenen Verzeichnis alle csv Dateien. Je nach Option wird die eingelesene Konfiguration am Bildschirm ausgegeben.

```
$ ./csv -h

Usage: csv [OPTION] cfgdir

  -a –all          print ALL
  -c –cyc          print CYC
  -d –detail       print DETAIL
  -g –get          print GET
  -s –set          print SET
  -h –help         print this message.
```

- **ebus_send** sendet einen angegebenen hex String auf den ebus und gibt die Antwort zurück.

```
$ ./ebus_send -h

Usage: ebus_send [OPTION] <ZZ PB SB NN DBx>

  <ZZ PB SB NN DBx>  spaces within message be removed.

  -a –address      set bus address. (ff)
  -d –device       use a specified serial device. (/dev/ttyUSB0)
  -p –prompt       stay on input prompt.
  -r –retry        max retry getting bus. (3)
  -s –skip         skipped ACK bytes after get-bus error. (1)
  -t –type         message type. (3)
                   1 = Broadcast, 2 = Master-Master, 3 = Master-Slave
  -w –wait         wait time for QQ compare. (~4000 usec)
  -h –help         print this message.
```