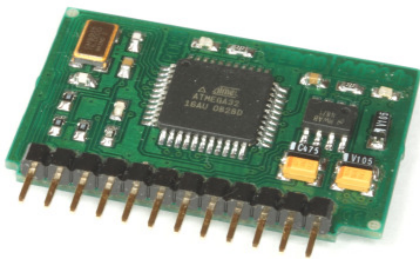


REFERENZHANDBUCH

Voice-Emitter VE II

Dokumentation für Anwender und Systementwickler

Version: 01.05



DRT DOHRENBUSCH REGEL-TECHNIK GmbH
Zur Alten Schule 12, D-46569 Hünxe, www.drt24.de

Inhalt

1	Allgemeines.....	4
1.1	Symbolerklärung	4
1.2	Informationen zum Referenzhandbuch	4
1.3	Urheberschutz	4
1.4	Entsorgung	4
2	Sicherheitshinweise	5
2.1	Allgemeines	5
2.2	Bestimmungsgemäße Verwendung.....	5
2.3	Arbeitssicherheit.....	5
3	Überblick.....	6
3.1	Kurzbeschreibung	6
3.2	Funktionalität	7
3.3	Anschlussbelegung	8
3.4	Konfigurationsdatei.....	9
3.4.1	TWI-Adresse	9
3.4.2	Sleep-Modus.....	10
3.4.3	TWI-Adresse	10
3.4.4	STARTUP.WAV	10
3.5	LED's.....	11
3.5.1	LED „Betrieb“	11
3.5.2	LED „Zugriff MicroSD-Karte“	11
3.5.3	LED „Verstärker“	12
3.6	Beispielbeschaltungen.....	12
3.6.1	Testbeschaltung	13
3.6.2	Standardbeschaltung.....	14
3.6.3	Anschluss an einen externen Verstärker	15
3.7	Sicherheitshinweise.....	16
4	Ansteuerung des Voice-Emitter II über den TWI-Bus.....	17
4.1	Starten des Voice-Emitter II	17
4.2	Schritte beim Senden und Empfangen	17
4.3	Kommunikationsprotokoll	17
4.4	ASCII- und Binär-Modus.....	18
4.5	Statusbyte	19
4.6	Datei- und Verzeichnisnamen	19
4.7	Formate von Klangdateien.....	20
4.8	Kompatible MicroSD-Karten	20
4.9	Fehlermeldungen.....	20
5	Befehlsreferenz.....	21
5.1	Kurzreferenz	21
5.2	Einzelbefehle.....	22
5.2.1	Befehl #.....	22
5.2.2	Befehl %.....	23
5.2.3	Befehl &.....	23
5.2.4	Befehl @.....	23
5.2.5	Befehl A.....	23
5.2.6	Befehl B.....	24
5.2.7	Befehl C.....	24

5.2.8	Befehl D.....	24
5.2.9	Befehl I.....	25
5.2.10	Befehl L.....	25
5.2.11	Befehl O.....	25
5.2.12	Befehl P.....	25
5.2.13	Befehl p.....	26
5.2.14	Befehl R.....	26
5.2.15	Befehl r.....	26
5.2.16	Befehl S.....	26
5.2.17	Befehl T.....	28
5.2.18	Befehl V.....	29
5.2.19	Befehl X.....	30
5.2.20	Befehl Y.....	30
6	WAV-Dateien mit VoiceEmitter-Loop-Table.....	31
6.1	Klangphasen.....	31
6.2	Spuren.....	32
6.3	Definition von Spurwechseln.....	32
6.4	Beispiel einer WAV-Datei mit Klangphasen, Spuren und Spurwechseldefinitionen	33
7	Beispiele.....	35
7.1	Beispiele in C.....	35
8	Technische Daten.....	43

1 Allgemeines

1.1 Symbolerklärung

Wichtige sicherheits- und gerätetechnische Hinweise in dieser Betriebsanleitung sind durch Symbole gekennzeichnet. Die Hinweise sind unbedingt zu befolgen, um Unfälle, Personen- und Sachschäden zu vermeiden.



ACHTUNG!

Dieses Symbol kennzeichnet Hinweise, deren Nichtbeachtung Beschädigungen, Fehlfunktionen und/oder Ausfall des Gerätes zur Folge haben kann.



HINWEIS!

Dieses Symbol hebt Tipps und Informationen hervor, die für eine effiziente und störungsfreie Bedienung des Gerätes zu beachten sind.

1.2 Informationen zum Referenzhandbuch

Das Referenzhandbuch dient Systementwickler für die Installation und softwaretechnische Ansteuerung des Gerätes und dem Anwender als wichtige Informationsquelle und Nachschlagewerk. Sie soll die qualitativ hochwertige und betriebssichere Funktion des Gerätes durch eine sachgemäße Bedienung unterstützen.

Voraussetzung hierfür ist die Kenntnis der bei Installation und Betrieb einzustellenden Parameter sowie deren Auswirkung auf Sprachausgabesystem Voice-Emitter II.



HINWEIS!

Die grafischen Darstellungen in dieser Bedienungsanleitung können unter Umständen leicht von der tatsächlichen Ausführung des Gerätes abweichen.

1.3 Urheberschutz

Alle unsere Produkte und Unterlagen sind im Sinne des Urheberrechtsgesetzes geschützt.

Weitergabe sowie Vervielfältigung von Unterlagen, auch auszugsweise, Verwertung und Mitteilung ihres Inhaltes sind nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen sind strafbar und verpflichten zu Schadenersatz.

Alle Rechte der Ausübung von gewerblichen Schutzrechten behalten wir uns vor.

1.4 Entsorgung



Die getrennte Sammlung der Elektroaltgeräte ist ein wichtiger Schritt zur dauerhaft umweltgerechten Kreislaufwirtschaft. Geben Sie Ihr Altgerät bitte zur fachgerechten Entsorgung bei Ihrer kommunalen Sammelstelle für Elektronikschrott ab.

2 Sicherheitshinweise

Dieses Kapitel bietet einen Überblick über alle wichtigen Sicherheitsaspekte.

Zusätzlich sind in den einzelnen Kapiteln konkrete Sicherheitshinweise zur Abwendung von Gefahren gegeben und mit Symbolen gekennzeichnet.

Die Beachtung aller Sicherheitshinweise ermöglicht den optimalen Schutz der Anwender und des SERVICE-Personals vor Gefährdungen und gewährleistet sicheren und störungsfreien Betrieb des Gerätes.

2.1 Allgemeines

Das Gerät ist nach den derzeit gültigen Regeln der Technik gebaut und betriebssicher.

2.2 Bestimmungsgemäße Verwendung

Die Betriebssicherheit des Gerätes ist nur bei bestimmungsgemäßer Verwendung entsprechend der Angaben in der Betriebsanleitung gewährleistet.

2.3 Arbeitssicherheit

Durch das Befolgen der Sicherheitshinweise kann eine Gefährdung von Personen und/oder des Gerätes verhindert werden.

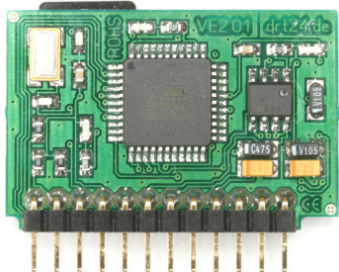
Das Nichtbeachten dieser Hinweise kann eine Gefährdung von Personen und Gegenständen durch elektrische Einwirkungen oder den Ausfall des Gerätes bewirken.

Nichtbeachten der Sicherheitsbestimmungen führt zum Verlust jeglicher Garantieansprüche.

3 Überblick

3.1 Kurzbeschreibung

Der Voice-Emitter II ist ein Modul, das Klanginformationen von einer MikroSD-Karte ausliest und diese als Audiosignal ausgibt. Er arbeitet mit Klangdateien im Format der **Windows-WAV**-Dateien.

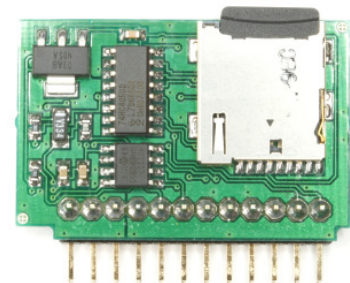


Das Audiosignal wird durch den integrierten **1.5 W-Verstärker** aufbereitet und kann direkt auf einen angeschlossenen **8 Ohm-Lautsprecher** ausgegeben werden.

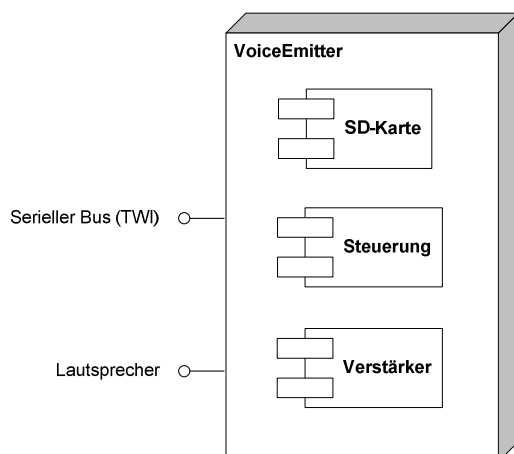
Um Knack- und Störgeräusche zu vermeiden, schaltet der Voice-Emitter II den angeschlossenen Lautsprecher bei Bedarf stumm.

Die Steuerung des Voice-Emitter II erfolgt über eine serielle Schnittstelle vom **Typ TWI**. Über eine Konfigurationsdatei auf der MikroSD-Karte kann die Geräte-Adresse eingestellt werden.

Der Voice-Emitter II schaltet nach einer konfigurierbaren Zeit in den **Ruhemodus**. Im Ruhemodus werden Verstärker und MikroSD-Karte abgeschaltet und die Steuereinheit in den Stromsparmodes versetzt.



Der Voice-Emitter II aktiviert sich selbstständig, sobald er über den seriellen Bus angesprochen wird. Der Stromsparmodes ist für das steuernde Gerät transparent.



3.2 Funktionalität

Der Voice-Emitter II ist ein autonomes System. Er empfängt seine Befehle über den seriellen Bus, führt entsprechende Aktionen aus und gibt Status- und System-Informationen über den Bus zurück.

Die folgende Auflistung enthält die Steuerungsmöglichkeiten über den seriellen Bus.

- **Dateiauswahl**
 - Auswahl einer WAV-Datei
 - Wechsel des aktuellen Verzeichnisses auf der MicroSD-Karte

- **Ablaufkontrolle**
 - Abspielen starten
 - Abspielen stoppen
 - Abspielen fortsetzen
 - Abspielposition innerhalb der Klangdatei verändern
 - Klang als Loop abspielen
 - Abspielgeschwindigkeit einstellen

- **Lautstärkewahl**
 - Lautstärke in 64 Schritten einstellen

- **System-Informationen**
 - Abfrage der Abspielposition (in Prozent, Sekunden oder Sample)
 - Abfrage der aktuellen Lautstärke (in Hardware-Schritten oder Prozent)
 - Dateilänge der geöffneten Datei (in Sekunden oder Sample)
 - Modellangabe und Versionsnummer
 - Status und Fehlernummer

- **Datenausgabe**
 - Auslesen und Ausgabe einer Datei über den seriellen Bus

3.3 Anschlussbelegung

Der Voice-Emitter II ist für zwei Betriebsarten vorbereitet:

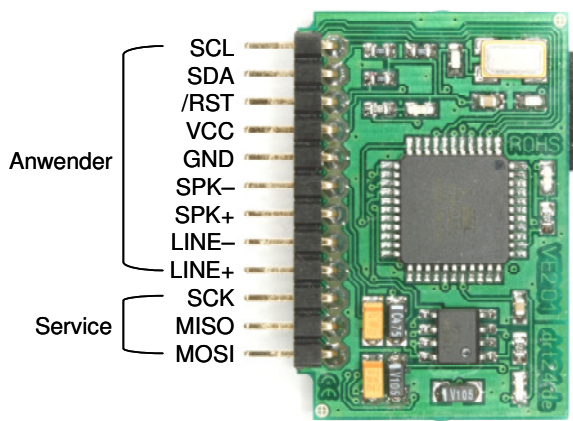
- Einsatz mit integriertem Verstärker und direkt angeschlossenem Lautsprecher (SPK)
- Betrieb mit angeschlossenem Verstärker (LINE)

In beiden Fällen werden ausschließlich Pins der ersten Anschlussgruppe „Anwender“ benötigt. Die zweite Anschlussgruppe „Service“ darf nicht beschaltet werden (nur für DRT-Servicezwecke).



ACHTUNG!

Beachten Sie bei den Anschlüssen bitte Kapitel 3.7, „Sicherheitshinweise“.



Anschluss	Bedeutung	Bemerkung
SCL	Clock (TWI)	SCL und SDA bilden den TWI-Bus und müssen beide beschaltet werden.
SDA	Datenleitung (TWI)	SCL und SDA bilden den TWI-Bus und müssen beide beschaltet werden.
/RST	Reset	Wird die Leitung für kurze Zeit auf Low gezogen, führt der Voice-Emitter II einen Reset durch. Die minimale Dauer für den Low-Impuls beträgt 1,5µs. Reset muss nicht beschaltet werden.
VCC	Versorgungsspannung +5V	Für den Betrieb ohne Brummgeräusche muss die Versorgungsspannung stabil und geglättet sein.
GND	Masse	
SPK-	Lautsprecher, negativ	Verstärktes Audiosignal (negativ), Lautsprecher 8Ω
SPK+	Lautsprecher, positiv	Verstärktes Audiosignal (positiv), Lautsprecher 8Ω
LINE-	Verstärker, negativ	Signalleitung (negativ) zum Verstärker
LINE+	Verstärker, positiv	Signalleitung (positiv) zum Verstärker
SCK	Service-Schnittstelle	Nicht beschalten!
MISO	Service-Schnittstelle	Nicht beschalten! Achtung: direkt mit der MicroSD-Karte verbunden. Eine Spannung >3,3V kann die MicroSD-Karte zerstören!
MOSI	Service-Schnittstelle	Nicht beschalten!

3.4 Konfigurationsdatei

Im Hauptverzeichnis der MicroSD-Karte erwartet der Voice-Emitter II eine Datei mit dem Namen „**VECONFIG.INI**“. Der Voice-Emitter II liest die Einstellungen dieser Datei einmalig beim Systemstart aus.

Bei **Werksauslieferung** lautet der Inhalt der VECONFIG.INI:

```
ADR=50
SLEEP=100
MUTE=50
STARTUP=STARTUP.WAV
```

3.4.1 TWI-Adresse

ADR=<Wert> gibt die Adresse des Voice-Emitter II auf dem TWI-Bus an, das heißt, hier wird die Adresse definiert, auf die der Voice-Emitter II „hört“. Der Wert wird hexadezimal angegeben.

Die Adresse besteht aus 8 Bit und kann binär wie folgt eingestellt werden:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Immer 0	immer 1	0 / 1	0 / 1	0 / 1	0 / 1	0 / 1	R / W

Bit 0 wird später bei der Ansteuerung des Voice-Emitter II verwendet, um festzulegen, ob ein lesender oder schreibender Zugriff erfolgen soll (s.u.). Bei der Definition der TWI-Adresse in der VECONFIG.INI wird dieses Bit erst einmal immer mit „0“ angegeben.

Bit 1 ... Bit 5 können frei gewählt werden also entweder logisch „0“ oder logisch „1“.

Bit 6 muß immer 1 sein

Bit 7 muß immer 0 sein

Beispiele:

Die kleinste, mögliche TWI-Adresse lautet:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Immer 0	immer 1	0	0	0	0	0	R / W

Binäre Darstellung: 01000000, Dezimal: 64, Hexadezimal: 40

Werkseinstellung ADR=50:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Immer 0	immer 1	0	1	0	0	0	R / W

Binäre Darstellung: 01010000, Dezimal: 80, Hexadezimal: 50

Die größte, mögliche TWI-Adresse lautet:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Immer 0	immer 1	1	1	1	1	1	R / W

Binäre Darstellung: 01111110, Dezimal: 126, Hexadezimal: 7E

Insgesamt ergeben sich so 32 verschiedene Möglichkeiten, die TWI-Adresse zu definieren:
Hexadezimal: 40 ... 7E



HINWEIS!

Weiter oben wurde erwähnt, daß **Bit 0** bei der Ansteuerung des Voice-Emitter II verwendet wird, um festzulegen, ob ein lesender oder schreibender Zugriff erfolgen soll. Beim lesenden Zugriff wird Bit 0 = „**1**“ gesetzt, beim schreibenden Zugriff wird Bit 0 = „**0**“ gesetzt.

Die werkseingestellte TWI-Adresse ADR=50 (hexadezimal, binär = 01010000) wird also folgend verwendet, wenn die Voice-Emitter II auf dem TWI-Bus adressiert wird: zum Lesen mit „0101000**1**“, der schreibende Zugriff erfolgt mit „0101000**0**“

3.4.2 Sleep-Modus

SLEEP=<Wert> gibt die Zeit in Sekunden an, bis die CPU und Peripherie des Voice-Emitter II in den Sleep-Modus wechselt.

3.4.3 TWI-Adresse

MUTE=<Wert> gibt die Zeit in Sekunden an, bis der integrierte Verstärker des Voice-Emitter II in den Sleep-Modus wechselt.

3.4.4 STARTUP.WAV

STARTUP=<Alpha,8>.WAV gibt eine Datei im Hauptverzeichnis der MicroSD-Karte an, die automatisch abgespielt wird, wenn Spannung an den Voice-Emitter II angelegt wird.



HINWEIS!

Bitte beachten: Der Name der dieser Datei muß immer aus 8 Zeichen (Buchstaben oder Ziffern), gefolgt von einem Punkt und der 3-stelligen Endung „WAV“. Alle Buchstaben müssen groß geschrieben werden. Das genaue Dateiformat entnehmen Sie bitte den Informationen in Kapitel 4.7, „Formate von Klangdateien“.



HINWEIS!

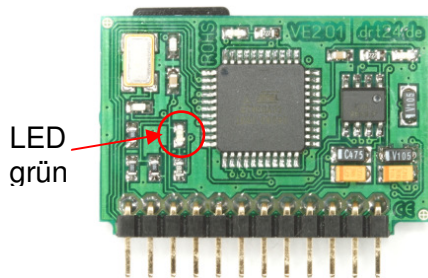
Soll keine STARTUP.WAV abgespielt werden, dann bitte diese Zeile in der VECONFIG.INI löschen.

3.5 LED's

Auf der Vorderseite des Voice-Emitter II befinden sich drei LED's. Mit ihnen werden der Status oder bestimmte Aktionen des Voice-Emitter II angezeigt.

3.5.1 LED „Betrieb“

Diese LED befindet sich direkt links neben dem Mikroprozessor:

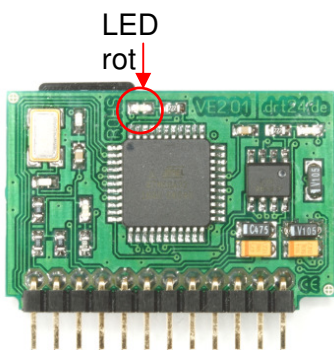


Folgende Betriebszustände kann die LED „Betrieb“ darstellen:

LED Betrieb	Bedeutung
LED leuchtet dauerhaft	Voice-Emitter II ist betriebsbereit
LED blinkt mit kurzer Dauer (ca. 2 mal je Sekunde)	Der Voice-Emitter II hat einen invaliden Befehl erhalten oder konnte eine Datei nicht finden
LED blinkt mit langer Dauer (ca. 1 mal je 2 Sekunden)	Der Voice-Emitter II kann auf die MicroSD-Karte nicht zugreifen
LED blinkt sehr schnell	Zugriff auf den Voice-Emitter II über den seriellen Bus
LED leuchtet nicht	Modul befindet sich im Sleep-Modus

3.5.2 LED „Zugriff MicroSD-Karte“

Diese LED befindet sich direkt über dem Mikroprozessor:

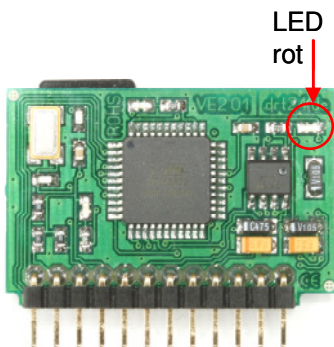


Folgende Betriebszustände kann die LED „Zugriff MicroSD-Karte“ darstellen:

LED-Anzeige	Bedeutung
LED leuchtet	Zugriffe auf die MicroSD-Karte
LED leuchtet nicht	Keine Zugriffe auf die MicroSD-Karte

3.5.3 LED „Verstärker“

Diese LED befindet sich rechts oberhalb des 8-poligen Verstärkers:



Folgende Betriebszustände kann die LED „Verstärker“ darstellen:

LED-Anzeige	Bedeutung
LED leuchtet	Verstärker ist eingeschaltet
LED leuchtet nicht	Verstärker befindet sich im Sleep-Modus

3.6 Beispielbeschaltungen

Die folgenden Beschaltungsbeispiele zeigen, wie der Voice-Emitter II anzuschließen ist.

HINWEIS!

Achten sie auf eine konstante und brummfreie Spannungsversorgung, da der Voice-Emitter II selbst keine eigene Spannungsregelung besitzt. In den Beispielen ist ein Stützkondensator eingezeichnet, der die Betriebssicherheit verbessert. Bei ausreichend stabiler Spannungsversorgung kann er entfallen.

3.6.1 Testbeschaltung

Ziel: Einfacher Funktionstest des Voice-Emitter II, Ausgabe der STARTUP-Datei über einen direkt angeschlossenen Lautsprecher.

Die mitgelieferte MicroSD-Karte muss sich in der Kartenhalterung des Voice-Emitter II befinden. Auf ihr befindet sich neben der VECONFIG.INI auch bereits eine Datei STARTUP.WAV. Alternativ können Sie auch eigene WAV-Dateien auf der MicroSD-Karte (im Hauptverzeichnis) ablegen (bitte die Datei VECONFIG.INI in der Zeile STARTUP=<Alpha,8>.WAV entsprechend definieren).

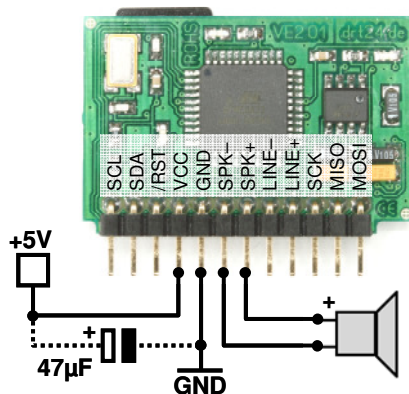


Bitte beachten: Der Name dieser Datei muß immer aus 8 Zeichen (Buchstaben oder Ziffern) bestehen, gefolgt von einem Punkt und der 3-stelligen Endung „WAV“. Alle Buchstaben müssen groß geschrieben werden. Das genaue Dateiformat entnehmen Sie bitte den Informationen in Kapitel 4.7, „Formate von Klangdateien“.



Die MicroSD-Karte bitte nur im spannungslosen Zustand in die Kartenhalterung einsetzen oder daraus entfernen.

Beschalten Sie den Voice-Emitter II nach dem nachfolgenden Schaltbild. Nach dem Anlegen der Versorgungsspannung gibt der Voice-Emitter II die Klanginformationen der Startup-Datei auf dem Lautsprecher aus:



3.6.2 Standardbeschaltung

Ziel: Ansteuern des Voice-Emitter II über TWI, Ausgabe der Klänge über einen direkt angeschlossenen Lautsprecher.

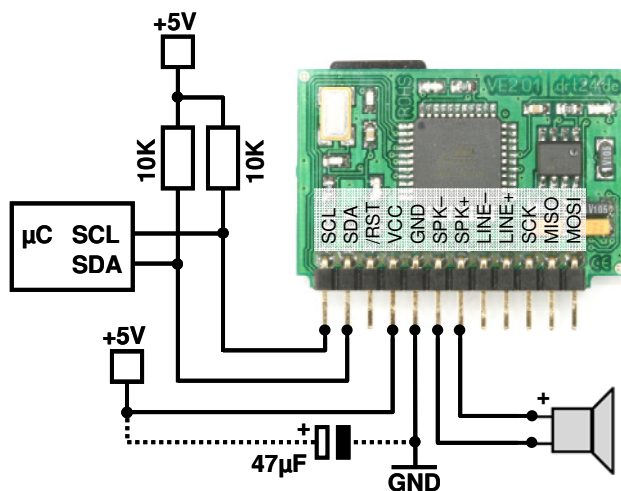
- Stellen Sie die TWI-Adresse über die Anweisung `ADR=<Wert>` in der Initialisierungsdatei `VECONFIG.INI` ein (siehe Kapitel 3.4.1, TWI-Adresse).
- Verbinden Sie die Anschlüsse SCL und SDA mit Ihrer μ C-Steuereinheit.
- Schließen Sie einen Lautsprecher am Voice-Emitter II an.
- Schließen Sie Masse und Versorgungsspannung (+5V) an.

 **HINWEIS!**

Falls noch nicht auf Ihrer μ C-Seite geschehen, terminieren Sie die beiden TWI-Leitungen SCL und SDA bitte jeweils mit einem 10 Kohm-Widerstand gegen +5V.

 **HINWEIS!**

Bitte beachten: Der Name der abzuspielenden Datei muß immer aus 8 Zeichen (Buchstaben oder Ziffern) bestehen, gefolgt von einem Punkt und der 3-stelligen Endung „WAV“. Alle Buchstaben müssen groß geschrieben werden. Das genaue Dateiformat entnehmen Sie bitte den Informationen in Kapitel 4.7, „Formate von Klangdateien“.



3.6.3 Anschluss an einen externen Verstärker

Ziel: Ansteuern des Voice-Emitter II über TWI, Ausgabe der Klänge über einen angeschlossenen, externen Verstärker.



ACHTUNG!

Spk+ oder Spk- niemals direkt an eine Verstärkeranlage anschließen. Beide Signale beziehen sich nicht auf Masse. Dafür immer Line+ und Line- verwenden.

- Stellen Sie die TWI-Adresse über die Anweisung `ADR=<Wert>` in der Initialisierungsdatei `VECONFIG.INI` ein (siehe Kapitel 3.4.1, TWI-Adresse).
- Verbinden Sie die Anschlüsse SCL und SDA mit Ihrer μC -Stuereinheit.
- Schließen Sie LINE+ und LINE- an den Eingang eines Verstärkers an.
- Schließen Sie einen Lautsprecher an den Verstärker an.
- Schließen Sie Masse und Versorgungsspannung (+5V) an.



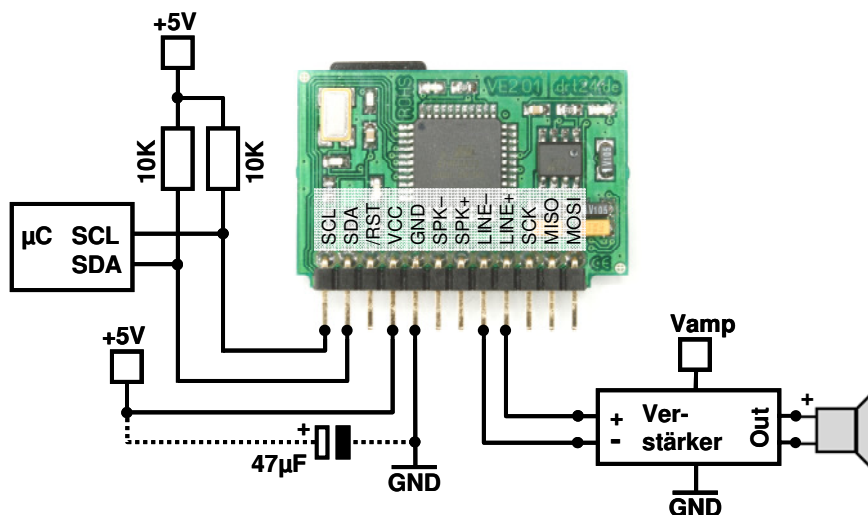
HINWEIS!

Falls noch nicht auf Ihrer μC -Seite geschehen, terminieren Sie die beiden TWI-Leitungen SCL und SDA bitte jeweils mit einem 10 Kohm-Widerstand gegen +5V.



HINWEIS!

Bitte beachten: Der Name der abzuspielenden Datei muß immer aus 8 Zeichen (Buchstaben oder Ziffern) bestehen, gefolgt von einem Punkt und der 3-stelligen Endung „WAV“. Alle Buchstaben müssen groß geschrieben werden. Das genaue Dateiformat entnehmen Sie bitte den Informationen in Kapitel 4.7, „Formate von Klangdateien“.



3.7 Sicherheitshinweise



ACHTUNG!

Der Voice-Emitter II kann bei elektrostatischen Entladungen oder bei Überspannungen zerstört werden. Treffen Sie daher Vorkehrungen, um dieses zu vermeiden.



ACHTUNG!

Um Schäden an der MicroSD-Karte oder am Voice-Emitter II zu vermeiden, setzen Sie die MicroSD-Karte ausschließlich im spannungslosen Zustand ein. Gleiches gilt für das Entnehmen der MicroSD-Karte.



ACHTUNG!

Das Signal MISO an der Anschlussleiste hat eine direkte Verbindung zur MicroSD-Karte. Beschalten Sie dieses Signal daher nicht. SD-Karten sind im Allgemeinen nicht für Spannungen von 5V ausgelegt und können Schaden nehmen.



ACHTUNG!

Schließen Sie am Lautsprecherausgang (LINE+ und LINE-) nur Lautsprecher mit einer ausreichend hohen Impedanz an (mindestens 8Ω). Schließen Sie den Lautsprecherausgang nicht kurz. Eine zu geringe Impedanz führt zu einer Überlastung des Voice-Emitter II und kann zu seiner Zerstörung führen.



ACHTUNG!

Spk+ oder Spk- niemals direkt an eine Verstärkeranlage anschließen. Beide Signale beziehen sich nicht auf Masse. Dafür immer Line+ und Line- verwenden.

4 Ansteuerung des Voice-Emitter II über den TWI-Bus

4.1 Starten des Voice-Emitter II

Der Voice-Emitter II startet beim Anlegen der Betriebsspannung automatisch.



Wenn Sie den Voice-Emitter II einschalten, dann sollten sie im ca. 100ms Zeit geben, bevor Sie ihn über TWI ansprechen. Fragen Sie dann immer zuerst ab, ob er bereit ist (BUSY-Flag im Statusbyte), da er je nach verwendeter MicroSD-Karte noch mit der Initialisierung beschäftigt ist oder je nach Konfiguration noch eine Datei (Startup.wav) abspielt.

4.2 Schritte beim Senden und Empfangen

Der Voice-Emitter II agiert am TWI-Buss immer als Slave, d.h. er reagiert immer nur auf Anfragen eines TWI-Masters.

Um Daten an den Voice-Emitter II zu senden, müssen Sie die folgenden Schritte durchführen:

- Senden der Startkondition
- Senden der Adresse mit Kennzeichen Schreiben (Bit 0 der TWI-Adresse)
- Senden eines Bytes
- Empfang von ACK oder NACK vom Voice-Emitter II
- Wiederholung der letzten beiden Schritte in Abhängigkeit von ACK oder NACK und der Anzahl der zu sendenden Daten
- Senden der Stoppkondition

Um Daten vom Voice-Emitter II auszulesen, müssen Sie die folgenden Schritte durchführen:

- Senden der Startkondition
- Senden der Adresse mit Kennzeichen Lesen (Bit 0 der TWI-Adresse)
- Empfang eines Bytes
- Senden von ACK, wenn weitere Daten gelesen werden sollen (anschließend erneuter Empfang eines Bytes)
- Senden von NACK, wenn keine weiteren Daten gelesen werden sollen
- Senden der Stoppkondition

4.3 Kommunikationsprotokoll

Daten, die zum Voice-Emitter II gesendet werden, bestehen aus Befehlen und Parametern. Das genaue Format ist im Kapitel 5, Befehlsreferenz, beschrieben.

Der Voice-Emitter II wartet die vollständige Übertragung der Daten ab, analysiert sie anschließend und führt die entsprechenden Befehle aus. Das Kennzeichen für eine vollständige Übertragung ist die Stoppkondition auf dem TWI-Bus. Erst mit diesem Signal beginnt der Voice-Emitter II mit der Befehlsausführung. Jeder Befehl muss zwischen einer Start- und einer Stoppkondition liegen. Das Senden von mehreren Befehlen hintereinander (ohne Stoppkondition) ist nicht möglich.

Werden Daten vom Voice-Emitter II abgerufen, dann sendet er immer mindestens drei Bytes:

Byte	Bezeichnung	Bedeutung
1	Anzahl Bytes	Enthält die Anzahl der Bytes, die noch folgen (dieses Byte nicht mitgerechnet)
2	Status	Byte, das den Status des Voice-Emitter II angibt
3	Fehlernummer	Gibt die Fehlernummer an, die bei der letzten Befehlsausführung aufgetreten ist.
4...n	Weitere Datenbytes	optional, in Abhängigkeit von „Anzahl Bytes“

Das erste Byte „Anzahl Bytes“ gibt die Anzahl der noch folgenden Bytes an und ist immer ≥ 2 . Ist es größer als 2, dann folgen weitere Datenbytes. Weitere Bytes werden nur gesendet, wenn vorher ein Befehl zur Datenausgabe gesendet wurde (z.B. Auflisten eines Verzeichnisses oder Auslesen einer Datei).

Der Voice-Emitter II verwaltet intern einen kleinen Puffer von 16 Bytes, so dass maximal 19 Bytes in einem Block gesendet werden (Anzahl Byte + Status + Fehlernummer + maximal 16 weitere Datenbytes).

Um weitere Datenbytes auslesen zu können, muss die Übertragung erst mit einer Stoppkondition auf dem TWI-Bus abgeschlossen werden und anschließend erneut ein Lesezugriff durchgeführt werden. Das Lesen über die angegebene Anzahl von Bytes hinaus, führt nicht zum Erfolg und liefert nur konstante Dummy-Werte.

Ist der interne Puffer des Voice-Emitter II voll, dann unterbricht er seine Ausführung und wartet auf die Abholung der Daten. Das Anfordern von Daten (z.B. Auflisten eines Verzeichnisses) ohne die Abholung der Daten führt dazu, daß der Voice-Emitter II keine weiteren Befehle annimmt.

Um den vollständigen Abruf aller Daten sicherzustellen, sollte solange lesend auf den Voice-Emitter II zugegriffen werden, bis er keine Folgedaten mehr liefert (Anzahl Bytes = 2) und im Statusbyte angibt, dass er den Befehl abgeschlossen hat (Flag BUSY = 0) (siehe Kapitel xyz ToDo).

4.4 ASCII- und Binär-Modus

Der Voice-Emitter II verfügt über die Arbeitsmodi BINÄR und ASCII. Ist der Modus BINÄR eingestellt, dann müssen alle numerischen Parameter als binäre Werte übertragen werden. Im ASCII-Modus werden die ASCII-Zeichen 0 – 9 als Dezimalzahl erwartet.

Bei Rückgabewerten unterscheidet der Voice-Emitter II ebenfalls die beiden Modi:

ASCII-Modus:

Numerische Antworten werden als ASCII-Ziffern gesendet, Dateien werden als Hex-Dump dargestellt.

Binär-Modus:

Numerische Antworten werden als binäre Zahlen gesendet, Dateien werden binär übertragen.

In der Befehlsreferenz wird bei jedem Befehl der Unterschied der beiden Modi aufgezeigt.

4.5 Statusbyte

Das vom Voice-Emitter II gelieferte Statusbyte besteht aus mehreren Bitwerten. Die einzelnen Bits haben die folgende Bedeutung:

Bit	Bezeichnung	Bedeutung
7		ist immer 1
6		reserviert, kann 0 oder 1 sein
5		reserviert, kann 0 oder 1 sein
4		reserviert, kann 0 oder 1 sein
3	VE_BINARY	= 1 wenn der Binär-Modus eingestellt ist = 0 wenn der ASCII-Modus eingestellt ist
2	VE_PLAYING	= 1, wenn ein Klang ausgegeben wird = 0, wenn kein Klang ausgegeben wird
1	VE_MEDIUM_ONLINE	= 1 wenn die MicroSD-Karte erkannt wurde und das Dateisystem kompatibel ist. = 0 wenn keine MicroSD-Karte angesprochen werden kann oder nicht kompatibel ist
0	VE_BUSY	= 0, wenn der Voice-Emitter II keinen Befehl ausführt = 1, wenn der Voice-Emitter II beschäftigt ist

4.6 Datei- und Verzeichnisnamen

In verschiedenen Befehlen werden Datei- oder Verzeichnisnamen als Parameter verwendet.

Die Angabe dieser Namen unterliegt den folgenden Regeln:

- Angabe immer nur im 8.3-Format.
Die Angabe von langen Dateinamen (> 8 Zeichen) wird nicht unterstützt. Zu jedem langen Dateinamen wird immer auch ein kurzer im 8.3-Format abgelegt (z.B. „NAME~1.WAV“). Kurze Namen kann man unter Windows im DOS-Fenster ansehen.
- Es müssen immer Großbuchstaben verwendet werden. Der Voice-Emitter II führt keine Konvertierung von Klein- in Großbuchstaben durch. Soll eine Datei „STARTUP.WAV“ abgespielt werden und der Name der Datei wird mit „Startup.wav“ angegeben, dann findet der Voice-Emitter II diese Datei nicht.
- Umlaute dürfen nicht verwendet werden.
- Es dürfen keine Pfadangaben verwendet werden. Die Angabe von z.B. „\AKTIV\NEU\EREIGNIS.WAV“ ist nicht möglich. Hier müssen Sie schrittweise vorgehen:
Einstellen von AKTIV
Einstellen von NEU
Abspielen von EREIGNIS.WAV

4.7 Formate von Klangdateien

Klangdateien, die der Voice-Emitter II abspielen kann, müssen folgende Bedingungen erfüllen:

- Dateien liegen im Windows-WAV-Format (RIFF-Format) vor
- sie enthalten nicht komprimierte Daten im PCM-Format
- sie enthalten nur einen Kanal (mono)
- sie haben eine Auflösung von 8 Bit
- die Samplerate ist maximal 44,1kHz
- der DATA-Chunk der Datei beginnt innerhalb der ersten 500 Bytes:
Eine WAV-Datei besteht immer aus einer Kennung am Anfang der Datei, einem Format-Bereich (mit Angaben zur Auflösung Sample-Rate, etc.) und einem Daten-Bereich (mit den Klanginformationen). Optional kann man weitere Bereiche (Chunks) für z.B. Autorenangaben vor dem Datenbereich einfügen. Der Voice-Emitter II unterstützt dies jedoch nur, wenn der Datenbereich in den ersten 500 Bytes beginnt. Sollen weitere Daten in der WAV-Datei abgespeichert werden, so kann dies alternativ hinter dem DATA-CHUNK erfolgen.

4.8 Kompatible MicroSD-Karten

Mit dem Voice-Emitter II können verwendet werden

- MikroSD-Karten, FAT16-Formatierung
- MikroHDSK-Karten (hohe Kapazität), FAT32-Formatierung

4.9 Fehlermeldungen

Die vom Voice-Emitter II übermittelte Fehlernummer hat die folgende Bedeutung:

Name	Wert	Bedeutung
VE_ERR_OK	0	kein Fehler
VE_ERR_NO_MEDIUM	1	Keine (kompatible) MicroSD-Karte
VE_ERR_UNKNOWN_CMD	2	Unbekannter Befehl
VE_ERR_INV_PARAMETER	3	Ungültiger Parameter
TWI_ERR_CMD_SIZE	10	Befehlslänge zu groß
FATERR_INVALID_MBR	16	Ungültiger MBR (falsch formatierte oder nicht kompatible MicroSD-Karte)
FATERR_INVALID_FAT	17	Nicht unterstützte FAT-Version
FATERR_READ	18	Fehler beim Lesen von der MicroSD-Karte
FATERR_BYTES_PER_SEC	19	Ungültige Anzahl Bytes pro Sektor (falsch formatierte oder nicht kompatible MicroSD-Karte)
FATERR_INVALID_VID	20	Ungültige Volumen-ID (falsch formatierte oder nicht kompatible MicroSD-Karte)
FATERR_FILE_NOT_FOUND	21	Datei oder Verzeichnis nicht gefunden
WAVERR_INV_FORMAT	64	Kein gültiges WAV-Format
WAVERR_UNSUP_FORMAT	65	Nicht unterstütztes WAV-Format
WAVERR_NO_FILE	66	Keine geöffnete WAV-Datei

5 Befehlsreferenz

5.1 Kurzreferenz

In der folgenden Kurzreferenz werden folgende Buchstaben verwendet:

- <z> für eine einzelne ASCII-Ziffer
- <d> für eine Folge von ASCII-Ziffern, die eine Dezimalzahl bezeichnen.
- <b1> für einen binären Wert bestehend aus einem Byte
- <b2> für einen binären Wert bestehend aus genau zwei Byte. Das niederwertigste Byte wird immer zuerst gesendet
- <b4> für einen binären Wert bestehend aus mindestens einem und maximal vier Bytes. Das niederwertigste Byte wird immer zuerst gesendet.

Be- fehl	Parameter ASCII-Modus	Parameter Binär-Modus	Auswirkung	Busy- Modus
Datei- und Verzeichnisauswahl				
C	<Verzeichnis> \ ..	<Verzeichnis> \ ..	Wechselt in das angegebene Verzeichnis	Nein
D	<Datei>	<Datei>	Öffnet die angegebene Datei und sendet den Inhalt über TWI	Nein
O	<WAV-Datei>	<WAV-Datei>	Öffnet die angegebene Datei	Nein
P	<WAV-Datei>	<WAV-Datei>	Öffnet die angegebene Datei und spielt sie ab	Nein
R	<WAV-Datei>	<WAV-Datei>	Öffnet die angegebene Datei und spielt sie als Loop ab	Nein
Ablaufkontrolle				
p	[<z>] <z> := { 0 1 }	[<b1>] <b1> := { 00h 01h }	Spielt eine zuvor durch P, R oder O geöffnete Datei ab.	Nein
r	[<z>][-] <z> := { 0...3 }	[<b1>][-] <b1> := { 00h...03h }	Spielt eine zuvor durch P, R oder O geöffnete Datei als Loop ab.	Nein
S	[+ -] <d> [% s z]	<b1><b4>	Stellt die aktuelle Abspielposition einer geöffneten Datei ein.	Ja
T	<d>[,<d>]	<b2><b4>	Stellt die Abspielgeschwindigkeit ein	Ja
X	[<z>] <z> := { 0 1 }	[<b1>] <b1> := { 00h 01h }	Stoppt das Abspielen einer Datei	Ja
Y	[<z>] <z> := { 0...3 }	[<b1>] <b1> := { 00h...03h }	Beendet die aktuelle Loop und wechselt entweder zu der angegebenen Spur oder beendet die Wiedergabe	Ja
Lautstärkenkontrolle				
V	[+ -] <d> [% s z]	<b1><b4>	Stellt die Lautstärke des abzuspielenden Klangs ein.	Ja
Informationsabfrage				
#	<z>	<b1>	Liefert den Wert einer Systemvariablen, die durch <n>,	Ja

			bzw. <b1> spezifiziert wird.	
%			Abspielposition in Prozent, entspricht dem Befehl #4, bzw #04h	Ja
I			Gibt Hersteller, Modellname und Softwareversion aus	Nein
L	[D]	[D]	Listet das aktuell eingestellte Verzeichnis auf: mit Parameter = ‚D‘ alle Unterverzeichnisse, ohne Parameter alle Dateien.	Nein
Grundlegendes Verhalten				
@	<n>	<b4>	Stellt die Anzahl Sekunden ein, nach der bei Inaktivität der Voice-Emitter II in den Sleep-Modus wechselt	Nein
&	<n>	<b4>	Stellt die Anzahl Sekunden ein, nach der bei Inaktivität der Verstärker des VoiceEmitters in den Sleep-Modus wechselt	Nein
A			Stellt den ASCII-Modus ein	Nein
B			Stellt den Binär-Modus ein	Nein

5.2 Einzelbefehle

5.2.1 Befehl

Befehl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy-Modus
#	<z>	<b1>	Ja

Der Befehl # liefert den Wert der Systemvariablen, die durch den Parameter <z>, bzw. <b1> spezifiziert wird. <z> ist eine ASCII-Ziffer von ‚0‘ bis ‚8‘. <b1> ist entsprechend ein Byte-Wert von 00h bis 08h. Der Parameter <z>, bzw. <b1> bezeichnen die folgenden Systemvariablen:

<z>	<b1>	Systemvariable
0	00h	Dateilänge in Sample
1	01h	Dateilänge in 1/10 Sekunden
2	02h	Abspielposition in Sample
3	04h	Abspielposition in 1/10 Sekunden
4	04h	Abspielposition in Prozent
5	05h	Maximale Lautstärke in Hardware-Schritten
6	06h	Aktuelle Lautstärke in Hardware-Schritten
7	07h	Aktuelle Lautstärke in Prozent
8	08h	Anzahl Sekunden bis zum Sleep-Modus

5.2.2 Befehl %

Befehl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy-Modus
%			Ja

Der Befehl % ist eine alternative Schreibweise für den Befehl #4 (ASCII-Modus), bzw #04h (Binär-Modus) und liefert die aktuelle Abspielposition in Prozent zurück.

5.2.3 Befehl &

Befehl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy-Modus
&	<n>	<b4>	Nein

Der Befehl @ stellt die Anzahl Sekunden ein, nach dem der Verstärker VoiceEmitters bei Inaktivität in den Sleep-Modus wechselt. Durch Abspielen eines Klangs wird der Verstärker wieder aktiviert

5.2.4 Befehl @

Befehl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy-Modus
@	<n>	<b4>	Nein

Der Befehl @ stellt die Anzahl Sekunden ein, nach denen der Voice-Emitter II bei Inaktivität in den Sleep-Modus wechselt. Im Sleep-Modus wird die MicroSD-Karte vollständig abgeschaltet. Durch erneute Adressierung des Voice-Emitter II über den TWI-Bus wird der Stromsparmodus beendet.



HINWEIS!

Die Reaktivierung der MicroSD-Karte und des Verstärkers benötigen etwas Zeit, so dass durch den Sleepmodus die Reaktionszeit des Voice-Emitter II etwas verlängert wird (im Bereich von wenigen Millisekunden).

5.2.5 Befehl A

Befehl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy-Modus
A		Stellt den ASCII-Modus ein	Nein

Mit dem Befehl A stellt der Voice-Emitter II den ASCII-Modus ein. In diesem Modus werden alle numerischen Parameter als Dezimalzahl in Form von ASCII-Ziffern erwartet. Numerische Rückgabewerte werden ebenfalls als Folge von ASCII-Ziffern (Dezimalzahl) gesendet. Der Befehl D (Dump einer Datei) liefert dann einen Hexdump (siehe Befehl D).

5.2.6 Befehl B

Befehl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy-Modus
B		Stellt den Binär-Modus ein	Nein

Mit dem Befehl B stellt der Voice-Emitter II den Binär-Modus ein. In diesem Modus werden alle numerischen Parameter als binäre Werte (Byte oder Folge von Bytes) erwartet.



Wird ein 32-Bit-Wert erwartet (4 Bytes), dann dürfen 1-4 Bytes angegeben werden. Das niederwertigste Byte muss immer zuerst gesendet werden.

Numerische Rückgabewerte werden im Binärmodus ebenfalls als Folge von BYTE-Werten gesendet (niederwertiges Byte zuerst). Der Befehl D (Dump einer Datei) liefert dann einen Bytestrom, der dem originalen Inhalt der Datei entspricht (siehe Befehl D).

5.2.7 Befehl C

Befehl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy-Modus
C	<Verzeichnis> \ ..	<Verzeichnis> \ ..	Nein

Der Befehl C wechselt in das als Parameter angegebene Verzeichnis. Angegeben werden kann ein Unterverzeichnis, das Hauptverzeichnis (, \) oder das übergeordnete Verzeichnis (, ..).



Eine Pfadangabe mit mehreren Verzeichnisangaben in Folge ist nicht möglich.

5.2.8 Befehl D

Befehl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy-Modus
D	<Datei>	<Datei>	Nein

Der Befehl D sendet den Inhalt der angegebenen Datei über TWI zurück. Ist der Binärmodus eingestellt, dann wird der exakte Dateinhalt gesendet. Ist der Binärmodus aktiviert, wird die Datei als Hexdump gesendet. Der Hexdump hat die folgende Form:

```
0000:08C0 7C A5 A0 79 78 97 AA 9D 87 94 B3 B6 9C A2 C3 B3 |..yx.....
0000:08D0 99 A8 B5 A2 81 81 8A 67 4F 55 49 38 3B 38 31 36 .....gOUI8;816
0000:08E0 36 4C 4E 41 43 57 8E 8A 6A 7C A0 AE 7C 6E 9F AA 6LNACW..j|..|n..
```

5.2.9 Befehl I

Befehl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy-Modus
I			Nein

Der Befehl I gibt den Hersteller, Modellinformation und Softwareversion des Voice-Emitter II aus.

Im ASCII-Modus hat die Rückgabe das Format

<Hersteller> <Modellname> <Softwareversion>#<Build>

Beispiel: DRT GmbH, VoiceEmitter, V1.0#231

Im Binär-Modus hat die Rückgabe das Format

<Hersteller><Modellnummer><SW-Version Major><SW-Version Minor><Build>

<Hersteller> 8 Byte, immer ‚DRT GmbH‘

<Modellnummer> 2 Byte, immer 001fh

<SW-Version Major> 1 Byte

<SW-Version Minor> 1 Byte

<Build> 2 Byte

5.2.10 Befehl L

Befehl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy-Modus
L	[D]	[D]	Nein

Der Befehl L listet den Inhalt des aktuell eingestellten Verzeichnisses auf. Ist der Parameter 'D' angegeben, dann werden nur die Namen der Unterverzeichnisse aufgelistet. Ist der Parameter nicht angegeben, dann werden alle Dateien (ohne Unterverzeichnisse) aufgelistet.

5.2.11 Befehl O

Befehl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy-Modus
O	<WAV-Datei>	<WAV-Datei>	Nein

Der Befehl O öffnet die angegebene WAV-Datei und bereitet sie zum Abspielen vor. Sie kann anschließend mit dem Befehl p abgespielt werden.

5.2.12 Befehl P

Befehl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy-Modus
P	<WAV-Datei>	<WAV-Datei>	Nein

Der Befehl P öffnet die angegebene WAV-Datei und spielt sie ab.

5.2.13 Befehl p

Befehl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy-Modus
p	[<z>] <z> := { 0 1 }	[<b1>] <b1> := { 00h 01h }	Nein

Der Befehl p spielt eine bereits geöffnete Datei ab. Wurde das Abspielen einer Datei mit dem Befehl X unterbrochen, dann setzt p an der Stelle, an der abgebrochen wurde, das Abspielen fort. Der Parameter z (,0' oder ,1' im ASCII-Modus) bzw. b1 (00h oder 01h im Binärmodus) gibt an, ob das Abspielen sofort mit kurzzeitigem Einblenden (Parameter = 1) oder von Beginn an mit voller Lautstärke (Parameter = 0) erfolgen soll. Die Dauer der Einblendung ist sehr kurz und soll Knackgeräusche vermeiden.

5.2.14 Befehl R

Befehl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy-Modus
R	<WAV-Datei>	<WAV-Datei>	Nein

Der Befehl R öffnet die angegebene WAV-Datei und spielt sie wiederholt als Loop ab.

5.2.15 Befehl r

Befehl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy-Modus
r	[<z>][<->] <z> := { 0...3 }	[<b1>][<->] <b1> := { 00h...03h }	Nein

Der Befehl spielt eine bereits geöffnete Datei als Loop ab. Die Parameter haben nur Auswirkungen, wenn in der WAV-Datei eine VoiceEmitter-Loop-Table definiert ist (siehe Kapitel 6, WAV-Dateien mit VoiceEmitter-Loop-Table). Der Parameter z (,0', ,1', ,2' oder ,3' im ASCII-Modus) bzw. b1 (00h, 01h, 02h oder 03h im Binärmodus) gibt an, welche Spur abgespielt werden soll (Default = 0). Ist das Zeichen ,-' angegeben, dann wird die Spur ohne Intro wiedergegeben.

5.2.16 Befehl S

Befehl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy-Modus
S	[+ -] <d> [% s z]	<b1><b4>	Ja

Mit dem Befehl S kann die aktuelle Abspielposition beeinflusst werden. Der Parameter besteht aus drei Komponenten:
<Modus><Betrag><Einheit>

<Modus> gibt an, ob die Abspielposition um <Betrag> verschoben werden soll oder ob <Betrag> die absolute, neu einzustellende Abspielposition ist.

<Betrag> ist ein numerischer Wert.

<Einheit> gibt an, welche Bedeutung <Betrag> hat: Sample, Sekunden oder 1/10 Sekunden.

ASCII-Modus:

<Modus> wird durch die Zeichen ‚+‘ (addieren, bzw. verschiebe zum Ende), ‚-‘ (subtrahiere, bzw. verschiebe zum Anfang) oder keinem Zeichen (absolute Positionsangabe) angegeben.

<Betrag> wird durch eine Folge von ASCII-Ziffern angegeben und stellt einen dezimalen Wert dar.

<Einheit> wird durch die Zeichen ‚s‘ (Sekunde), ‚z‘ (Zehntel-Sekunde), ‚%‘ (Prozentwert) oder keinem Zeichen (Sample) angegeben.

Binär-Modus

Im Binärmodus wird <Modus> und <Einheit> zu einem Bytewert b1 zusammengefasst, indem der Code für <Modus> und der Code für <Einheit> addiert werden:

Modus absolut 00h

Modus + 01h

Modus - 02h

Format Sample 00h

Format Prozent 10h

Format Sekunde 20h

Format 1/10 Sekunde 30h

Der Betrag ist danach als Folge von 1 bis 4 Bytes anzugeben (niederwertiges Byte zuerst).

Beispiele:

ASCII-Modus S[+ -]<d>[% s z]	Umrechnung	Binär-Modus S<b1><b4>	Bedeutung
S+10%	+ -> 01h % -> 10h b1 = 01h+10h = 11h b4 = 10 = 0Ah	S 11h 0Ah	Verschiebt die Abspielposition um 10% der Dateilänge zum Ende.
S-10s	- -> 02h s -> 20h b1 = 02h+20h = 22h b4 = 10 = 0Ah	S 22h 0Ah	Verschiebt die Abspielposition um 10 Sekunden zum Anfang
S44100	absolut -> 00h Sample -> 00h b1 = 00h b4 = 44100 = AC44h	S 00h 44h ACh	Stellt die Abspielposition auf das Sample 44100 ein

S20z	absolut -> 00h 1/10 s -> 30h b1 = 30h b4 = 20 = 14h	S 30h 14h	Stellt die Abspielposition auf 20 Zehntel-Sekunden ein (entspricht S2s).
------	--	-----------	--



HINWEIS!

<b4> wurde hier auf die notwendigen Bytes gekürzt. Es können jedoch auch immer vier Bytes angegeben werden: Anstelle von S 11h 0Ah kann auch S 11h 00h 00h 0Ah gesendet werden.

5.2.17 Befehl T

Befehl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy-Modus
T	<d>[,<d>]	<b2>[<b4>]	Ja

Mit dem Befehl T kann die Abspielgeschwindigkeit des VoiceEmitters beeinflusst werden. Der erste Parameter d (im Binärmodus b2) gibt die gewünschte Abspielgeschwindigkeit in Prozent der Normalgeschwindigkeit an. Der Parameter muss mindestens 50 und maximal 200 sein. Das Prozentzeichen darf nicht angegeben werden.

Der zweite Parameter d (im Binärmodus b4) ist optional und gibt die Zeitdauer an, in der sich der Wechsel der Abspielgeschwindigkeit vollziehen soll. Die Zeitdauer wird in 1/100 Sekunden angegeben.

Hinweis: Wird im Binärmodus der Parameter b4 angegeben, dann muss der Parameter b2 mit zwei Bytes angegeben werden und darf nicht verkürzt werden. Ist nur der Parameter b2 angegeben, dann darf er in der verkürzten Form mit einem Byte angegeben werden.

Beispiele:

ASCII-Modus	Binär-Modus	Bedeutung
T<d>[,<d>]	T<b2>[<b4>]	
T120,100	T 78h 00h 64h	Stellt die Abspielgeschwindigkeit auf 120% ein. Die Abspielgeschwindigkeit wird in 1 Sekunde erreicht.
T80,1100	T 50h 00h 4Ch 04h	Stellt die Abspielgeschwindigkeit auf 80% ein. Die Abspielgeschwindigkeit wird in 11 Sekunden erreicht.
T100	T 64h	Stellt die Abspielgeschwindigkeit sofort auf 100% ein. Der Parameter b2 im Binärmodus kann hier als verkürzter 1-Byte-Wert angegeben werden.



HINWEIS!

Wird im Binärmodus der Parameter b4 angegeben, dann muss der Parameter b2 mit zwei Bytes angegeben werden und darf nicht verkürzt werden. Ist nur der Parameter b2 angegeben, dann darf er in der verkürzten Form mit einem Byte angegeben werden.

5.2.18 Befehl V

Befehl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy-Modus
V	[+ -] <d> [%]	<b1><b4>	Ja

Mit dem Befehl S kann die aktuelle Lautstärke des Voice-Emitter II eingestellt werden.
Der Parameter besteht aus drei Komponenten:
<Modus><Betrag><Einheit>

- <Modus> gibt an, ob die Lautstärke um <Betrag> erhöht oder vermindert werden soll oder ob <Betrag> die absolute, neu einzustellende Lautstärke ist.
- <Betrag> ist ein numerischer Wert.
- <Einheit> gibt an, welche Bedeutung <Betrag> hat: Hardware-Schritte oder Prozent.

ASCII-Modus:

<Modus> wird durch die Zeichen ,+' (addieren, lauter), ',' (subtrahiere, leiser) oder keinem Zeichen (absolute Lautstärke) angegeben.
<Betrag> wird durch eine Folge von ASCII-Ziffern angegeben und stellt einen dezimalen Wert dar.
<Einheit> wird durch das Zeichen ,%' (Prozentwert) oder keinem Zeichen (Hardware-Schritte) angegeben.

Binär-Modus

Im Binärmodus wird <Modus> und <Einheit> zu einem Bytewert b1 zusammengefasst, indem der Code für <Modus> und der Code für <Einheit> addiert werden:

Modus absolut 00h
 Modus + 01h
 Modus - 02h

 Format HW-Schritt 00h
 Format Prozent 10h

Der Betrag ist danach als Folge von 1 bis 4 Bytes anzugeben (niederwertiges Byte zuerst).

Beispiele:

ASCII-Modus V[+ -]<d>[%]	Umrechnung	Binär-Modus V<b1><b4>	Bedeutung
V+10%	+ -> 01h % -> 10h b1 = 01h+10h = 11h b4 = 10 = 0Ah	V 11h 0Ah	Erhöht die Lautstärke um 10%
V-10	- -> 02h HW-Schritte -> 00h b1 = 02h b4 = 10 = 0Ah	V 02h 0Ah	Vermindert die Lautstärke um 10 Hardware-Schritte

V50	absolut -> 00h HW->Schritte -> 00h b1 = 00h b4 = 50 = 32h	V 00h 32h	Stellt die Lautstärke auf Stufe 50 ein
V100%	absolut -> 00h % -> 10h b1 = 10h b4 = 100 = 64h	V 10h 64h	Stellt die maximale Lautstärke ein (100%)



<b4> wurde hier auf die notwendigen Bytes gekürzt. Es können jedoch auch immer bis zu vier Bytes angegeben werden.

5.2.19 Befehl X

Befehl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy-Modus
X	[<z>] <z> := { 0 1 }	[<b1>] <b1> := { 00h 01h }	Ja

Der Befehl bricht das Abspielen einer WAV-Datei ab. Der Parameter <z> (,0' oder ,1' im ASCII-Modus) bzw. b1 (00h oder 01h im Binärmodus) gibt an, ob der Abbruch mit kurzzeitigem Ausblenden (Parameter = 1) oder sofort (Parameter = 0) erfolgen soll. Die Dauer der Ausblendung ist sehr kurz und soll Knackgeräusche vermeiden.

5.2.20 Befehl Y

Befehl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy-Modus
Y	[<z>] <z> := { 0...3 }	[<b1>] <b1> := { 00h...03h }	Ja

Der Befehl beendet das Abspielen der aktuellen Loop, die mit dem Befehl R oder r gestartet wurde. Der Parameter z (bzw. b1 im Binärmodus) gibt die Spur an, die nach Beendigung der aktuellen Loop abgespielt werden soll. Ist der Parameter nicht angegeben, dann wird das Abspielen des Klangs beendet. Der Parameter macht nur Sinn, wenn in der aktuell geöffneten WAV-Datei eine VoiceEmitter-Loop-Table definiert ist (siehe Kapitel 6, WAV-Dateien mit VoiceEmitter-Loop-Table).

Beispiele:

ASCII-Modus	Binär-Modus	Bedeutung
Y[<Z>]	Y[<b1>]	
Y1	Y 01h	Beendet die aktuelle Loop, wechselt in die Spur 1 und spielt diese dann als Loop ab.
Y	Y	Beendet zuerst die aktuelle Loop und beendet dann das Abspielen des aktuellen Klangs.

6 WAV-Dateien mit VoiceEmitter-Loop-Table

Der VoiceEmitter ist in der Lage, WAV-Dateien im PCM-Format, 8 Bit, mono zu öffnen und abzuspielen. Um das Abspielen von Klängen flexibler und für manche Anwendungsfälle realistischer zu gestalten, wurde eine Loop-Table definiert, die in die WAV-Datei integriert werden kann.

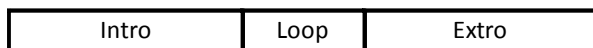
An dieser Stelle zeigen wir kurz die zusätzlichen Möglichkeiten auf, die durch die Loop-Table entstehen. Für detaillierte Informationen zu WAV-Dateien mit Voice-Emitter-Loop-Table besuchen sie unseren Download-Bereich auf www.drt24.de oder sprechen Sie uns an.

WAV-Dateien mit VoiceEmitter-Loop-Table bieten drei Möglichkeiten, das Abspielen von Klängen zu steuern, um sie dynamischer und realistischer zu gestalten:

- Klangphasen
- Spuren
- Definition von Spurwechseln

6.1 Klangphasen

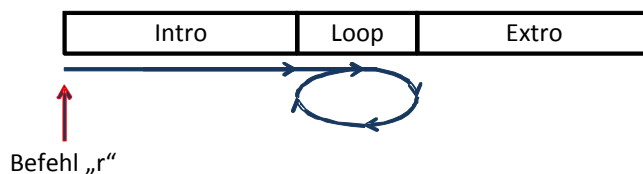
Jeder Klang in der VoiceEmitter-Loop-Table ist in drei unterschiedliche Phasen aufgeteilt:



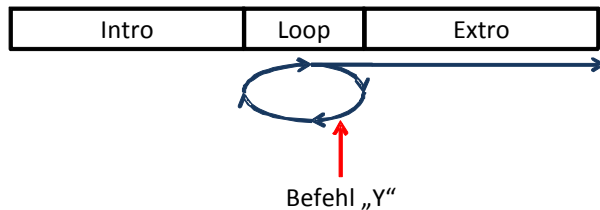
- **Intro**
Ist der Beginn eines Klangs.
Beispiel: das Einschwingen eines Blasinstruments oder das Anlassen eines Motors
- **Loop**
Ist der Hauptteil des Klangs, der endlos wiedergegeben werden kann.
Beispiel: das eingeschwungene Blasinstrument oder der Motor im Leerlauf
- **Extro**
Ist das Ende eines Klangs.
Beispiel: das Abklingen eines Blasinstruments oder der Motor, wenn er ausgeschaltet wird.

Beispiel:

- Intro enthält das Anlassen eines Motors
- Loop enthält das Leerlaufgeräusch
- Extro enthält das Ausschalten des Motors



Mit dem Befehl „O“ wird eine WAV-Datei mit Loop-Table geöffnet und anschließend mit dem Befehl „r“ als Loop gestartet. Zuerst wird das Anlassen des Motors abgespielt (Intro) und danach der Leerlauf des Motors (Loop).



Wenn der Befehl „Y“ übermittelt wird, dann beendet der VoiceEmitter zuerst die Loop und spielt danach das Ausschalten des Motors (Extro) ab.

6.2 Spuren

Eine WAV-Datei mit VoiceEmitter-Loop-Table kann vier verschiedene Spuren beinhalten. Spuren werden durch den VoiceEmitter nicht parallel, sondern hintereinander abgespielt. Jede einzelne Spur besteht wiederum aus Intro, Loop und Extro.

Die Spur einer bereits spielenden WAV-Datei kann mit dem Befehl „Y“, gefolgt von der Nummer der Spur, eingestellt werden.

Intro (3)	Loop (3)	Extro (3)
Intro (2)	Loop (2)	Extro (2)
Intro (1)	Loop (1)	Extro (1)
Intro (0)	Loop (0)	Extro (0)

6.3 Definition von Spurwechseln

Beim Betreten oder Verlassen einer Spur ist es nicht immer sinnvoll, das Intro oder das Extro abzuspielen. Ebenfalls kann es erwünscht sein, nicht aus jeder Spur heraus den Klang beenden zu können. Daher ist in der VoiceEmitter-Loop-Table definiert, ob

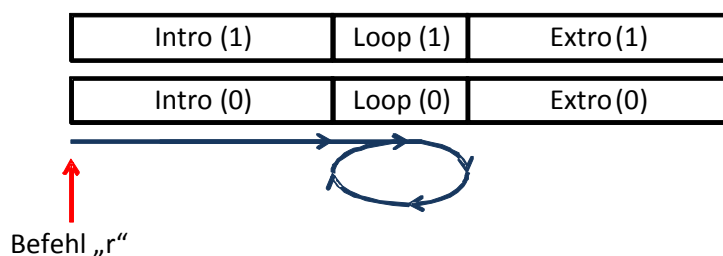
- beim Spurwechsel das Extro der aktuellen Spur abgespielt werden soll
- beim Spurwechsel das Intro der folgenden Spur abgespielt werden soll
- der Klang aus der aktuellen Spur heraus beendet werden kann oder nur aus einer anderen

Diese Logik bei den Spurwechseln wird durch den VoiceEmitter automatisch interpretiert und ausgeführt, ohne dass separate Befehle notwendig sind.

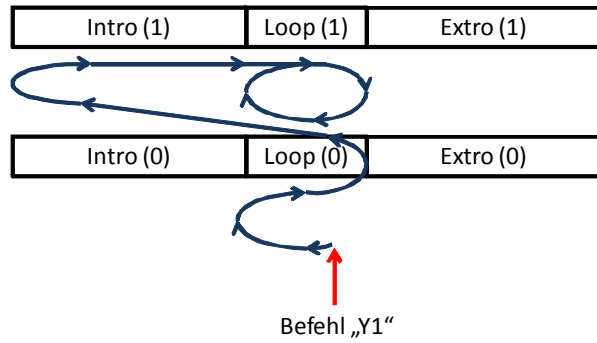
6.4 Beispiel einer WAV-Datei mit Klangphasen, Spuren und Spurwechseldefinitionen

Die WAV-Datei sei wie folgt definiert:

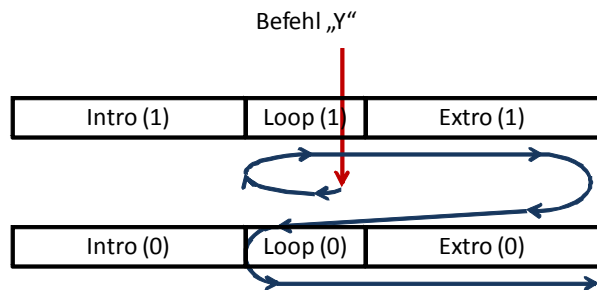
- Intro (0) enthält das Anlassen eines Motors
- Loop (0) enthält das Leerlaufgeräusch
- Extro (0) enthält das Ausschalten des Motors
- Intro (1) enthält das ansteigende Geräusch vom Leerlauf in den Arbeitsbereich
- Loop (1) enthält das Motorgeräusch bei Arbeitsdrehzahl
- Extro (1) enthält das abfallende Geräusch von der Arbeitsdrehzahl in den Leerlauf
- Beim Wechsel von Spur 0 nach 1 soll das Ausschalten des Motors nicht abgespielt werden (kein Extro bei 0->1)
- Beim Wechsel von Spur 0 nach 1 soll das Intro von Spur 1 abgespielt werden (ansteigend von Leerlauf zur Arbeitsdrehzahl)
- Beim Wechsel von Spur 1 nach 0 soll das Extro der aktuellen Spur abgespielt werden (abfallende Drehzahl bis in den Leerlauf)
- Beim Wechsel von Spur 1 nach 0 soll das Intro von Spur 0 nicht abgespielt werden (kein erneutes Starten des Motors)
- Der Klang soll nur aus Spur 0 heraus beendet werden können.



Mit dem Befehl „r“ wird das Abspielen gestartet. Zuerst ist das Starten des Motors zu hören (Intro 0), dann der Leerlauf (Loop 0).



Mit dem Befehl „Y1“ wird zuerst die Loop 0 (Motorleerlauf) bis zum Ende abgespielt, und danach in die Spur 1 gewechselt. Extro 0 (Motor ausschalten) wird nicht abgespielt, Intro 1 (Leerlaufdrehzahl anheben bis zur Arbeitsdrehzahl) wird jedoch abgespielt.



Mit dem Befehl „Y“ (ohne Parameter) soll der Klang nun beendet werden. Weil das Beenden des Klangs aus der Spur 1 heraus in diesem Beispiel nicht erlaubt ist, wechselt der VoiceEmitter zuerst zur Spur 0.

Dabei wird das Extro 1 (abfallende Drehzahl bis zum Leerlauf) abgespielt. Das Intro 0 (startender Motor) wird jedoch nicht wiedergegeben. Die Loop 0 (Leerlaufgeräusch) wird einmal abgespielt und dann der Klang über Extro 0 (Abschalten des Motors) beendet.

7 Beispiele

Der folgende Programmcode basiert auf einem ATmega8 der Firma Atmel. Das Beispiel setzt die Standardbeschaltung (siehe Kapitel 3.6.2, Standardbeschaltung) voraus, d.h. das am Voice-Emitter II eine Versorgungsspannung anliegt, ein Lautsprecher angeschlossen wurde und das der TWI-Bus mit dem steuernden Mikrocontroller verbunden ist.

Der nachfolgende Programmcode beschreibt, wie der Voice-Emitter II von einem Mikrocontroller angesprochen werden kann.

Zuerst werden jeweils Funktionen für die Ansteuerung des TWI-Busses gezeigt. Diese Funktionen sind als Beispiel zu verstehen. Sie können hinsichtlich Robustheit und Effizienz verbessert werden (z.B. durch Timeout bei Schleifen, Verwendung von Interrupts, etc).

7.1 Beispiele in C

Die Kommunikation über den TWI-Bus erfolgt immer nach dem Schema

- Startkondition senden
- Daten senden oder empfangen
- Stopkondition senden

Wenn gelesen werden soll, muss das Bit 0 der TWI-Adresse 1 sein, sonst 0.

Nach dem Lesen eines Bytes muss der Empfänger mit ACK antworten, sofern er ein weiteres Byte erwartet, sonst Antwortet er mit NACK.

Die folgenden vier Funktionen stellen die Basis für den Zugriff auf den TWI-Bus dar.

```
#define F_CPU 1000000L // CPU-Frequenz
#define F_SCL 10000L // TWI-Frequenz

#include <avr\io.h>
#include <util\twi.h>
#include <util\delay.h>
#include <string.h>

typedef uint8_t BYTE;

// initialisiert den Timer für den TWI-Bus
void twiInit()
{
    TWSR = 0; // kein prescaler
    TWBR = (BYTE)((F_CPU/F_SCL)-16)/2; // TWI-Speed
}

/* -----
Start condition senden
adr: Adresse des Slave

Rückgabe: 1 -> ok
          0 -> Fehler
----- */
BYTE twiStart(BYTE adr)
{
    // Start condition senden
    TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);

    // warten bis µC fertig
    while(!(TWCR & (1<<TWINT)));
}
```

```

// Abbruch bei Fehler
BYTE twiStatus = TW_STATUS & 0xF8;
if ( (twiStatus != TW_START) && (twiStatus != TW_REP_START) )
    return 0;

// Geräteadresse senden
TWDR = adr;
TWCR = (1<<TWINT) | (1<<TWEN);

// warten bis µC fertig und ACK/NACK empfangen wurde
while(!(TWCR & (1<<TWINT)));

// Abbruch bei Fehler
twiStatus = TW_STATUS & 0xF8;
if ( (twiStatus != TW_MT_SLA_ACK) && (twiStatus != TW_MR_SLA_ACK) )
    return 0;

return 1;
}

/* -----
   Stop condition senden
   ----- */
void twiStop()
{
    // Stop condition senden
    TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);

    // Warten bis Senden abgeschlossen ist
    while(TWCR & (1<<TWSTO));
}

/* -----
   Datenbyte senden
   data: zu sendendes Datenbyte

   Rückgabe: 1 -> ok
              0 -> Fehler
   ----- */
BYTE twiWrite(BYTE data)
{
    // Datenbyte versenden
    TWDR = data;
    TWCR = (1<<TWINT) | (1<<TWEN);

    // Warten, bis µC fertig
    while(!(TWCR & (1<<TWINT)));

    // Status abfragen, Fehler wenn Slave nicht mit ACK geantwortet hat
    if( (TW_STATUS & 0xF8) != TW_MT_DATA_ACK)
        return 0;
    return 1;
}

/* -----
   Datenbyte lesen und mit ACK beantworten

   Rückgabe: empfangenes Datenbyte
   ----- */
BYTE twiReadACK()
{
    // Daten empfangen, anschließend ACK senden
    TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWEA);
    // Warten bis Datenbyte empfangen wurde
    while(!(TWCR & (1<<TWINT)));

    // Datenbyte auslesen
    return TWDR;
}

```

```

/* -----
   Datenbyte lesen und mit NACK beantworten

   Rückgabe: empfangenes Datenbyte
   ----- */
BYTE twiReadNACK()
{
    // Daten empfangen, anschließend NACK senden
    TWCR = (1<<TWINT) | (1<<TWEN);
    // Warten bis Datenbyte empfangen wurde
    while(!(TWCR & (1<<TWINT)));

    // Datenbyte auslesen
    return TWDR;
}

```

Das Senden von Befehlen an den Voice-Emitter II kann mit den TWI-Basis-Funktionen wie folgt durchgeführt werden.

```

/* -----
   Befehl an den VoiceEmitter senden
   bAdr:      TWI-Adresse des VoiceEmitter
   pBuf:      Zeiger auf den zu sendenden Datenbereich
   bSize:     Anzahl zu sendender Bytes

   Rückgabe: 1 -> ok
              0 -> Fehler
   ----- */
BYTE ve_sendCmd(BYTE bAdr, BYTE* pBuf, BYTE bSize)
{
    // start condition senden, schreibender Zugriff,
    // daher Bit 0 der Adresse auf 0 setzen
    if (!twiStart(bAdr & 0xfe)) return 0;

    // Pufferinhalt übertragen
    while (bSize > 0)
    {
        // ein Byte senden
        if (!twiWrite(*pBuf) )
        {
            // Stop condition bei Fehler
            twiStop();
            return 0;
        }

        bSize--;
        pBuf++;
    }

    // Befehl abschließen
    twiStop();

    return 1;
}

```

Analog dazu werden Daten vom Voice-Emitter II mit den TWI-Basis-Funktionen durchgeführt. Die Besonderheit beim Voice-Emitter II ist, das er immer zuerst die Anzahl noch folgender Bytes mitteilt.

```

/* -----
   Daten vom VoiceEmitter abholen
   bAdr:      TWI-Adresse des VoiceEmitter
   pBuf:      Zeiger auf den zu sendenden Datenbereich
               Achtung: Der Puffer muss 17 Bytes groß sein

   Rückgabe: Anzahl gelesener Bytes

```

```

----- */
BYTE ve_getData(BYTE bAdr, BYTE* pBuf)
{
    // start condition senden, lesender Zugriff,
    // daher Bit 0 der Adresse auf 1 setzen
    if (!twiStart(bAdr | 0x01))
    {
        twiStop();
        return 0;
    }

    // 1. Byte = Anzahl noch folgender Bytes
    BYTE bBytes = twiReadACK();

    BYTE bCnt = bBytes;
    while (bCnt > 0)
    {
        if (bCnt > 1)
            // Byte lesen und mit ACK beantworten
            *pBuf = twiReadACK();
        else
            // letztes Byte lesen und mit NACK beantworten
            *pBuf = twiReadNACK();

        bCnt--;
        pBuf++;
    }

    twiStop();

    return bBytes;
}

```

Daten, die der Voice-Emitter II sendet, beginnen immer mit der Anzahl Folgebytes, einem Statusbyte und einer Fehlernummer. Danach folgen, sofern vorhanden, die eigentlichen Nutzdaten wie z.B. Dateiinhalte oder abgefragte Systemparameter.

Wenn der Voice-Emitter II keine Nutzdaten zur Verfügung stellt, bedeutet das nicht, das im Laufe der Befehlsabarbeitung keine weiteren Daten anfallen. Um die Daten vollständig abzuholen, muss das Flag BUSY im Status-Byte abgefragt werden. Erst wenn der Voice-Emitter II nicht mehr BUSY ist und keine Daten mehr vorhanden sind, sind alle Daten übertragen worden.

Die folgende Funktion fragt den Voice-Emitter II nach Daten ab und blendet Status- und Fehlerbyte aus. Sind keine Daten vorhanden, dann wartet sie ab, bis der Voice-Emitter II nicht mehr BUSY ist.

Die Funktion muss solange aufgerufen werden, bis sie keine Daten mehr liefert.

```

/* -----
Nutzdaten vom VoiceEmitter abholen, wartet
falls noch Daten gesendet werden könnten
bAdr:     TWI-Adresse des VoiceEmitter
pBuf:     Zeiger auf den zu sendenden Datenbereich
          Achtung: Der Puffer muss 17 Bytes groß sein

Rückgabe: Anzahl gelesener Bytes
          0: es folgen keine weiteren Daten
          >0: Anzahl Daten im Puffer, weitere Daten
              ggf. vorhanden
----- */
BYTE ve_getPayloadData(BYTE bAdr, BYTE* pBuf)
{
    // BUSY-Flag mit 1 initialisieren, damit while-Schleife
    // mindestens einmal durchlaufen wird
    BYTE bBusyFlag = 1;

    // Abfrage solange wiederholen, bis VE nicht mehr BUSY ist

```

```

while (bBusyFlag)
{
    // Daten vom VoiceEmitter abfragen
    BYTE bSize = ve_getData(bAdr, pBuf);

    // Der VoiceEmitter sendet immer drei Bytes:
    // Anzahl Folgebytes, Status und Fehlernummer
    // Status und Fehlernummer sind im Puffer abgelegt
    // Falls bSize < 2, dann Fehler bei der TWI-Kommunikation
    if (bSize < 2)
        return 0;

    // wenn Nutzdaten vorhanden, dann Puffer umkopieren,
    // d.h. die ersten beiden Bytes überschreiben
    if (bSize > 2)
    {
        for (BYTE bIdx = 0; bIdx < bSize-2; bIdx++)
        {
            *(pBuf+bIdx) = *(pBuf+bIdx+2);
        }

        // Länge der Nutzdaten korrigieren und zurückgeben
        return bSize-2;
    }

    // Status ist erstes Byte im Puffer
    BYTE bStatus = *pBuf;

    // BUSY-Flag ist Bit 0 des Statusbyte
    bBusyFlag = bStatus & 0x01;

    // Wenn BUSY, dann wird die Schleife wiederholt
    // Im Fall der Wiederholung etwas warten um keine unnötig große
    // Last beim VoiceEmitter durch ständige Abfragen zu erzeugen
    if (bBusyFlag)
        _delay_ms(100);
}

// VoiceEmitter ist nicht mehr BUSY, es sind keine Daten vorhanden
// und weitere Daten werden nicht mehr generiert

return 0;
}

```

Die Funktion `ve_getPayloadData` kann dazu verwendet werden, um zu warten, bis der Voice-Emitter II nicht mehr im Status BUSY ist.

```

/* -----
Warten, bis VoiceEmitter nicht mehr BUSY ist
bAdr:      TWI-Adresse des VoiceEmitter
----- */
void ve_waitForReady(BYTE bAdr)
{
    BYTE buffer[17];

    // in ve_getPayloadData werden Nutzdaten abgeholt und
    // so lange gewartet, bis das BUSY-Flag 0 ist
    while (ve_getPayloadData(bAdr, (BYTE*)&buffer) > 0);
}

```

Wenn nur die Fehlernummer des Voice-Emitter II abgefragt werden soll, ohne Nutzdaten zu empfangen, kann die folgende Funktion verwendet werden.

Vorsicht: wenn der interne Puffer des Voice-Emitter II voll ist, dann wartet er auf die Abholung der Daten und unterbricht seine Befehlsabarbeitung.

```

/* -----
Fehlernummer des VoiceEmitter abfragen
bAdr:      TWI-Adresse des VoiceEmitter
----- */

```

```

Rückgabe: Fehlernummer des VoiceEmitter
           0xff bei TWI-Kommunikationsfehler
----- */
BYTE ve_getError(BYTE bAdr)
{
    // start condition senden, lesender Zugriff,
    // daher Bit 0 der Adresse auf 1 setzen
    if (!twiStart(bAdr | 0x01))
    {
        twiStop();
        return 0xff;
    }

    // 1. Byte = Anzahl noch folgender Bytes
    BYTE bBytes = twiReadACK();
    // muss immer >= 2 sein, sonst TWI-Kommunikationsfehler
    if (bBytes < 2)
    {
        twiStop();
        return 0xff;
    }

    // 2. Byte = Status (wird hier nicht weiter verwendet)
    // BYTE bStatus =
    twiReadACK();

    // 3. Byte = Fehlernummer
    BYTE bError = twiReadNACK();

    // ggf. noch folgende Bytes werden hier nicht abgefragt
    // und müssen, sofern vorhanden, separat abgerufen werden

    twiStop();
    return bError;
}

```

Mit den zuvor gezeigten Funktionen kann der Voice-Emitter II gesteuert werden. Das Beispiel ist mit Kommentaren versehen, so dass es ohne weitere Bemerkungen verständlich sein sollte. Die gezeigten Beispiele verwenden die Dateien auf der mitgelieferten MicroSD-Karte.

```

#define VE_TWI_ADRESS 0x50

int main()
{
    /* -----
       TWI-Initialisieren und Start des VoiceEmitters abwarten
       ----- */
    // TWI-Timer initialisieren
    twiInit();

    // Start des VoiceEmitters abwarten
    _delay_ms(100);
    // Warten, bis VoiceEmitter ready
    ve_waitForReady(VE_TWI_ADRESS);

    /* -----
       Verzeichnis einstellen und einen Klang abspielen
       ----- */
    // Verzeichnis /demo einstellen (Befehl C)
    ve_sendCmd(VE_TWI_ADRESS, (BYTE*)"CDEMO", strlen("CDEMO"));

    // Klang READY.WAV abspielen
    ve_waitForReady(VE_TWI_ADRESS);
    ve_sendCmd(VE_TWI_ADRESS, (BYTE*)"PREADY.WAV", strlen("PREADY.WAV"));

    /* -----
       VoiceEmitter bis 4 zählen lassen
    */
}

```

```

----- */
for (BYTE idx = 1; idx <= 4; idx++)
{
    // Befehl P aufbereiten
    // Nutzung der Dateien ZD00?000.WAV
    char szCmd[20];
    strcpy((char*)&szCmd, "PZD001000.WAV");
    szCmd[5] = '0' + idx;

    ve_waitForReady(VE_TWI_ADRESS);
    // Befehl absenden
    ve_sendCmd(VE_TWI_ADRESS, (BYTE*)&szCmd, strlen((char*)&szCmd));
}

/* -----
Fehler provozieren und darauf reagieren
----- */
#define FATERR_FILE_NOT_FOUND 21

// Nicht vorhandene Datei abspielen lassen -> Fehler
ve_waitForReady(VE_TWI_ADRESS);
ve_sendCmd(VE_TWI_ADRESS, (BYTE*)"PXYZ.WAV", strlen("PXYZ.WAV"));

// Fehlermeldung abfragen
ve_waitForReady(VE_TWI_ADRESS);
BYTE bErr = ve_getError(VE_TWI_ADRESS);

// Im Fehlerfall einen Klang ausgeben
if (bErr == FATERR_FILE_NOT_FOUND)
    ve_sendCmd(VE_TWI_ADRESS, (BYTE*)"PERR_21.WAV", strlen("PERR_21.WAV"));

/* -----
Klang abspielen, Abspielposition und Lautstärke ändern und
Abspielen abbrechen
----- */
// ASCII-Modus einstellen, damit Parameter als ASCII-Ziffern angegeben
// werden können
ve_waitForReady(VE_TWI_ADRESS);
ve_sendCmd(VE_TWI_ADRESS, (BYTE*)"A", 1);

// Klang zum Abspielen öffnen, jedoch noch nicht abspielen
ve_waitForReady(VE_TWI_ADRESS);
ve_sendCmd(VE_TWI_ADRESS, (BYTE*)"OTAKE5.WAV", strlen("OTAKE5.WAV"));

// Abspielposition auf 27 Sekunden einstellen
ve_waitForReady(VE_TWI_ADRESS);
ve_sendCmd(VE_TWI_ADRESS, (BYTE*)"S27s", sizeof("S27s"));

// Abspielen starten
ve_waitForReady(VE_TWI_ADRESS);
ve_sendCmd(VE_TWI_ADRESS, (BYTE*)"p", 1);

// 5 sekunden warten
_delay_ms(5000);

// Lautstärke auf 50% einstellen
// Der Befehl 'V' funktioniert auch im BUSY-Modus, daher muss hier
// nicht auf den VoiceEmitter gewartet werden
ve_sendCmd(VE_TWI_ADRESS, (BYTE*)"V50%", sizeof("V50%"));

// 5 sekunden warten
_delay_ms(5000);

// Abspielposition auf 12 Sekunden einstellen
ve_sendCmd(VE_TWI_ADRESS, (BYTE*)"S12s", sizeof("S12s"));

// 5 sekunden warten
_delay_ms(5000);

// Lautstärke auf 90% einstellen,
ve_sendCmd(VE_TWI_ADRESS, (BYTE*)"V90%", sizeof("V90%"));

// 10 sekunden warten

```

```
_delay_ms(10000);

// Abspielen abbrechen
ve_sendCmd(VE_TWI_ADRESS, (BYTE*)"X1", sizeof("X1"));

/* -----
   Die ersten 50 Zeichen aus einer Datei abfragen und, sofern es
   Ziffern sind, vorlesen
   ----- */

// Binär-Modus einstellen
ve_waitForReady(VE_TWI_ADRESS);
ve_sendCmd(VE_TWI_ADRESS, (BYTE*)"B", 1);

// Die ersten 50 Zeichen der Datei DIGITS.TXT abfragen
ve_sendCmd(VE_TWI_ADRESS, (BYTE*)"DDIGITS.TXT", strlen("DDIGITS.TXT"));

BYTE buffer[50];
BYTE bBufferIdx = 0;
while (bBufferIdx < 50)
{
    BYTE tmpBuffer[17];
    // Daten abfragen
    BYTE bLen = ve_getPayloadData(VE_TWI_ADRESS, (BYTE*)&tmpBuffer);

    if (bLen == 0)
        // Keine weiteren Daten vorhanden
        break;

    // Überlauf des Puffers verhindern
    if (bLen + bBufferIdx > 50)
        bLen = 50-bBufferIdx;

    // tmpBuffer in buffer umkopieren
    for (BYTE n = 0; n < bLen; n++)
    {
        buffer[bBufferIdx+n] = tmpBuffer[n];
    }

    bBufferIdx += bLen;
}

// Dateiausgabe abbrechen
ve_sendCmd(VE_TWI_ADRESS, (BYTE*)"X", strlen("X"));
ve_waitForReady(VE_TWI_ADRESS);

// Zeichen vorlesen, wenn es Ziffern sind
for (BYTE idx = 0; idx < bBufferIdx; idx++)
{
    if ( (buffer[idx] >= '0') && (buffer[idx] <= '9') )
    {
        // Befehl aufbereiten
        char szCmd[20];
        strcpy((char*)&szCmd, "PZD001000.WAV");
        szCmd[5] = buffer[idx];

        ve_waitForReady(VE_TWI_ADRESS);
        // Befehl absenden
        ve_sendCmd(VE_TWI_ADRESS, (BYTE*)&szCmd, strlen((char*)&szCmd));
    }
}

ve_waitForReady(VE_TWI_ADRESS);

while (1);

return 0;
}
```

8 Technische Daten

Parameter	typisch	minimal	maximal	Einheit
Versorgungsspannung	5V	4,2	5,4	V
Stromaufnahme bei Klangausgabe	65	60	75	mA
Stromaufnahme Standby (Verstärker aus)	58	-	-	mA
Stromaufnahme Sleepmodus	28	-	-	mA
Lautsprecher Impedanz	8	8	-	Ω
Zulässiger Temperaturbereich Betrieb	-	0	55	$^{\circ}\text{C}$
Zulässiger Temperaturbereich Lagerung	-	0	55	$^{\circ}\text{C}$
Samplerate	-	6.000	41.100	Sample/s
Auflösung	8	-	-	Bit
Abmessungen	37,5 x 25,0 (Breite X Höhe)			mm
Gewicht	< 7			g