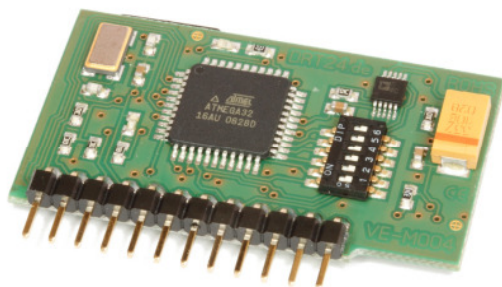


REFERENZHANDBUCH

VoiceEmitter VE-M004

Version: 01.02



DRT DOHRENBUSCH REGEL-TECHNIK GmbH
Hauptstraße 47, 53902 Bad Münstereifel

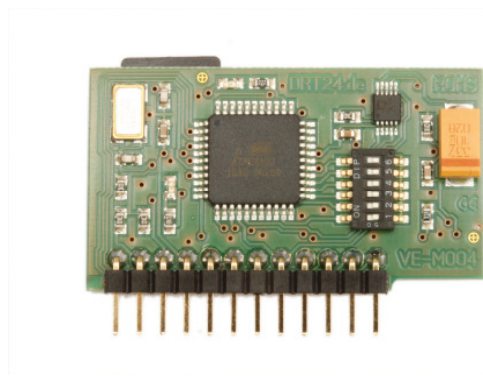
Inhalt

1	Überblick.....	3
1.1	Kurzbeschreibung.....	3
1.2	Funktionen des VoiceEmitters.....	4
1.3	Anschlussbelegung.....	5
1.4	Einstellungen des DIL-Schalters.....	6
1.4.1	STARTUP.WAV.....	6
1.4.2	TWI-Adresse.....	6
1.5	LED's.....	7
1.6	Beispielbeschaltungen.....	8
1.6.1	Testbeschaltung.....	8
1.6.2	Standardbeschaltung.....	9
1.6.3	Anschluss an externen Verstärker.....	10
1.7	Sicherheitshinweise.....	10
2	Ansteuerung des VoiceEmitters über den TWI-Bus.....	11
2.1	Starten des VoiceEmitters.....	11
2.2	Schritte beim Senden und Empfangen.....	11
2.3	Kommunikationsprotokoll.....	11
2.4	ASCII- und Binär-Modus.....	12
2.5	Statusbyte.....	12
2.6	Datei- und Verzeichnisnamen.....	13
2.7	Formate von Klangdateien.....	13
2.8	Kompatible SD-Karten.....	14
2.9	Fehlermeldungen.....	14
3	Befehlsreferenz.....	15
3.1	Kurzreferenz.....	15
3.2	Befehl #.....	16
3.3	Befehl %.....	16
3.4	Befehl @.....	16
3.5	Befehl A.....	16
3.6	Befehl B.....	17
3.7	Befehl C.....	17
3.8	Befehl D.....	17
3.9	Befehl I.....	17
3.10	Befehl L.....	18
3.11	Befehl O.....	18
3.12	Befehl P.....	18
3.13	Befehl p.....	18
3.14	Befehl S.....	19
3.15	Befehl V.....	20
3.16	Befehl X.....	21
4	Beispiele.....	22
4.1	Beispiele in C.....	22
4.2	Beispiele in BASCOM.....	30
5	Technische Parameter.....	31

1 Überblick

1.1 Kurzbeschreibung

Der VoiceEmitter ist ein Modul, das Klanginformationen von einer Mikro-SD-Karte ausliest und diese als Audiosignal ausgibt. Er arbeitet mit Klangdateien im Format der Windows-WAV-Dateien.

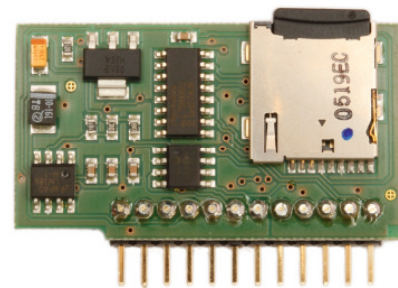


Das Audiosignal wird durch den integrierten Verstärker aufbereitet und kann direkt zu einem externen Lautsprecher geleitet werden.

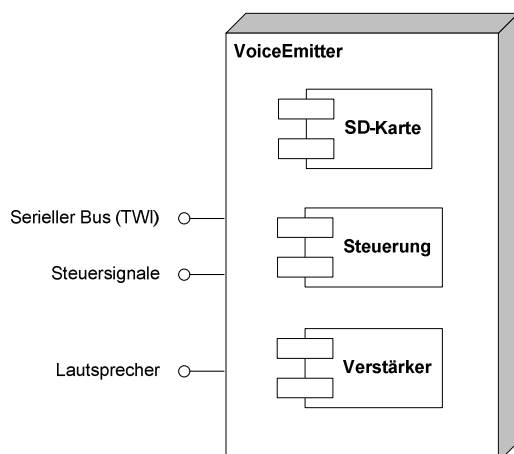
Zur Vermeidung von Knack- und Störgeräuschen schaltet der VoiceEmitters den angeschlossenen Lautsprecher bei Bedarf stumm.

Die Steuerung des VoiceEmitters erfolgt über eine serielle Schnittstelle vom Typ TWI. Über einen DIL-Schalter kann die Geräte-Adresse aus 32 unterschiedlichen Werten eingestellt werden.

Der VoiceEmitter schaltet nach einer konfigurierbaren Zeit in den Ruhemodus. Im Ruhemodus werden Verstärker und SD-Karte abgeschaltet und die Steuereinheit in den Stromsparmodes versetzt. Der VoiceEmitter aktiviert sich selbstständig, sobald er über den seriellen Bus angesprochen wird. Der Stromsparmodes ist für das steuernde Gerät transparent.



Die Signale für das Stummschalten des Lautsprechers und den Stromsparmodes sind an den Anschlüssen des VoiceEmitters verfügbar.



1.2 Funktionen des VoiceEmitters

Der VoiceEmitter 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. Die konkreten Software-Befehle sind im Kapitel 3, Befehlsreferenz, beschrieben.

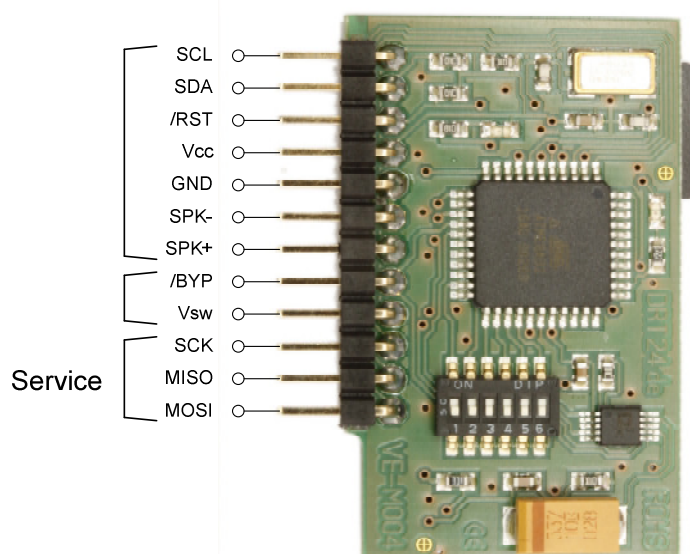
- Dateiauswahl
 - Auswahl einer WAV-Datei
 - Wechsel des aktuellen Verzeichnisses auf der SD-Karte
- Ablaufkontrolle
 - Abspielen starten
 - Abspielen stoppen
 - Abspielen fortsetzen
 - Abspielposition innerhalb der Klangdatei verändern
- Lautstärkenwahl
 - Lautstärke in 64 Schritten einstellen
- System-Informationen
 - 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 einer Datei und Ausgabe über den seriellen Bus

1.3 Anschlussbelegung

Der VoiceEmitter ist für den Einsatz mit integriertem Verstärker und externem Lautsprecher ausgelegt. Hierzu ist die erste Gruppe mit sieben Anschlüssen ausreichend. Die anderen Anschlüsse müssen nicht beschaltet werden.

Die zweite Gruppe stellt zwei Steuerleitungen zur Verfügung. Sie geben an, ob der Lautsprecher zu- oder abgeschaltet ist und ob der VoiceEmitter sich im Stromsparmodes befindet. Die Steuerleitungen sind für mögliche Erweiterungen gedacht und werden in der Standardbeschaltung nicht benötigt.

Die dritte Gruppe ist nur für Servicezwecke relevant und darf nicht beschaltet werden. Beachten Sie bei diesen drei Anschlüssen unbedingt Kapitel 1.7, Sicherheitshinweise.



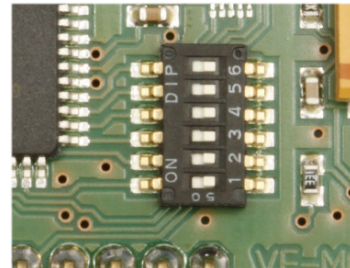
Anschluss	Bedeutung	Bemerkung
SCL	Clock (TWI)	SCL und SDA machen den TWI-Bus aus und müssen beide beschaltet werden.
SDA	Datenleitung (TWI)	SCL und SDA machen den TWI-Bus aus und müssen beide beschaltet werden.
/RST	Reset	Wird die Leitung für kurze Zeit auf Low gezogen, dann führt der VoiceEmitter einen Reset durch. Die minimale Dauer für den Low-Impuls beträgt 1,5µs. Reset muss nicht beschaltet werden.
Vcc	Versorgungsspannung +5V	Für einen fehlerfreien Betrieb ohne Brummgeräusche muss die Versorgungsspannung stabil und geglättet sein.
GND	Masse	
SPK-	Lautsprecher, negativ	Ist auf dem VoiceEmitter mit Masse verbunden.
SPK+	Lautsprecher, positiv	verstärktes Audiosignal
/BYP	Bypass speaker	Dieses Signal zeigt an, ob der Verstärkerausgang auf den Ausgang SPK+ durchgeschaltet ist (Lautsprecher ist zugeschaltet) oder ob der Ausgang SPK+ offen ist

Anschluss	Bedeutung	Bemerkung
		(Lautsprecher abgeschaltet). Low: Lautsprecher abgeschaltet High: Lautsprecher zugeschaltet. Das Signal ist für mögliche Erweiterungen gedacht und wird in der Standardbeschaltung nicht verwendet.
Vsw	Signal für geschaltete Versorgungsspannung	Vsw ist die interne Versorgungsspannung für die SD-Karte und den Verstärker. Sie wird Null, wenn der VoiceEmitter den Stromsparmodus aktiviert. Vsw darf mit maximal 15mA belastet werden. Das Signal ist für mögliche Erweiterungen gedacht und wird in der Standardbeschaltung nicht verwendet.
SCK	Service-Schnittstelle	Nicht beschalten!
MISO	Service-Schnittstelle	Nicht beschalten! Achtung: MISO ist direkt mit der SD-Karte verbunden. Eine Spannung > 3,3V wirkt direkt auf die SD-Karte und kann sie zerstören!
MOSI	Service-Schnittstelle	MOSI Service-Schnittstelle Nicht beschalten!

1.4 Einstellungen des DIL-Schalters

Auf der Vorderseite des VoiceEmitters befindet sich ein DIL-Schalter mit 6 Schiebeschaltern.

Mit ihnen kann eingestellt werden, ob beim Einschalten des VoiceEmitters eine Datei automatisch abgespielt wird und auf welche Adresse der VoiceEmitter am TWI-Bus reagieren soll. Der VoiceEmitter liest die Schalterstellungen des DIL-Schalters einmalig beim Systemstart aus. Eine Änderung der Schalter hat im laufenden Betrieb keine Auswirkungen.



1.4.1 STARTUP.WAV

Ist der Schalter 1 auf „on“ eingestellt, dann prüft der VoiceEmitter beim Start (Anlegen einer Versorgungsspannung oder Reset), ob auf der SD-Karte eine Datei mit dem Namen STARTUP.WAV vorhanden ist. Ist die Datei vorhanden, dann wird sie abgespielt. Ist die Datei nicht vorhanden und der Schalter 1 ist auf „on“ eingestellt, dann wird der interne Fehlerstatus entsprechend gesetzt (siehe auch Kapitel xxx ToDo, Fehlermeldungen). Im Falle eines Fehlerstatus ist der VoiceEmitter uneingeschränkt funktionsbereit.

1.4.2 TWI-Adresse

Mit den Schaltern 2-6 wird die TWI-Adresse des VoiceEmitters eingestellt. Die Adresse besteht aus 7 Bit und ist wie folgt einzustellen:

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

Bit 0 einer TWI-Adresse bestimmt, ob ein lesender oder schreibender Zugriff erfolgen soll. Der VoiceEmitter erkennt an diesem Bit, ob ein Befehl an den VoiceEmitter gesendet werden soll oder ob eine Abfrage von Daten erfolgt.

Beispiel:

Der VoiceEmitter soll auf die Adresse 50h eingestellt werden.
50h entspricht 01010000b;

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

Entsprechend muss der DIL-Schalter 5 auf „on“, die Schalter 6, 4, 3 und 2 auf „off“ eingestellt werden.

1.5 LED's

Auf der Vorderseite des VoiceEmitters befinden sich zwei LED's. Mit ihnen werden der Status oder bestimmte Aktionen des VoiceEmitters angezeigt.

LED-Anzeige	Bedeutung
Grüne LED leuchtet dauerhaft	VoiceEmitter ist betriebsbereit
Grüne LED blinkt mit kurzer Dauer (ca. 2 mal je Sekunde)	Der VoiceEmitter hat einen invaliden Befehl erhalten oder konnte eine Datei nicht finden
Grüne LED blinkt mit langer Dauer (ca. 1 mal je 2 Sekunden)	Der VoiceEmitter kann auf die SD-Karte nicht zugreifen
Grüne LED funkelt	Zugriff auf den VoiceEmitter über den seriellen Bus
Rot LED leuchtet	Zugriffe auf die SD-Karte

1.6 Beispielbeschaltungen

Die folgenden Beschaltungsbeispiele zeigen, wie der VoiceEmitter anzuschließen ist. Achten sie auf eine konstante und brumfreie Spannungsversorgung, da der VoiceEmitter selber keine eigene Spannungsregelung besitzt. In den Beispielen ist ein Stützkondensator eingezeichnet, der die Betriebssicherheit verbessert. Bei ausreichend stabiler Spannungsversorgung kann er weggelassen werden.

1.6.1 Testbeschaltung

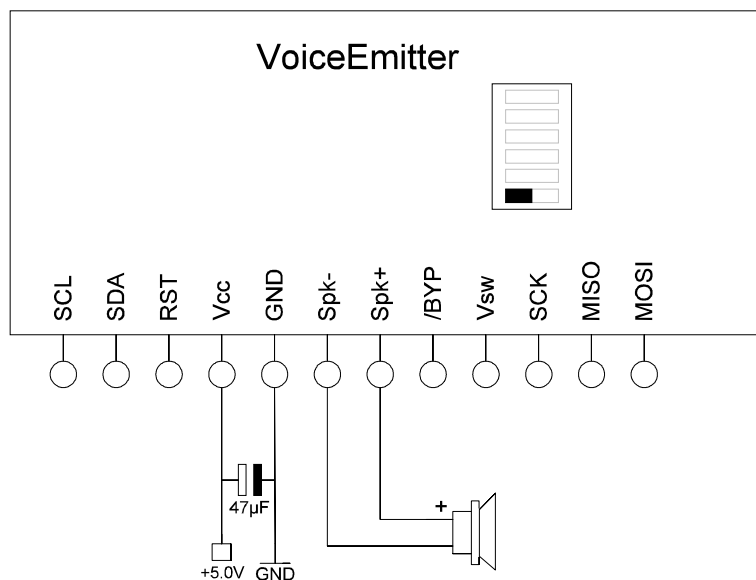
Ziel: Einfacher Funktionstest des VoiceEmitters

Die mitgelieferte SD-Karte muss sich in der Kartenhalterung des VoiceEmitters befinden. Auf ihr befindet sich bereits eine Datei STARTUP.WAV.

Alternativ können Sie eine WAV-Datei auf einer Mikro-SD-Karte im Hauptverzeichnis ablegen. Beachten Sie, dass der Name der WAV-Datei STARTUP.WAV sein muss. Die WAV-Datei und die SD-Karte müssen kompatibel sein (siehe hierzu Kapitel xxx ToDo, Formate von Klangdateien, und Kapitel xxx ToDo, Arten und Formatierung der Mikro-SD-Karte). Sie können auch eine STARTUP.WAV aus unserem Service-Bereich im Internet downloaden. Setzen Sie die SD-Karte in die Kartenhalterung des VoiceEmitters ein. Achten Sie darauf, dass sich der VoiceEmitter im spannungslosen Zustand befindet.

Stellen Sie nun den DIL-Schalter 1 auf „on“ (siehe Kapitel xxx ToDo, Einstellungen des DIL-Schalters).

Beschalten Sie den VoiceEmitter nach dem nachfolgenden Schaltbild.

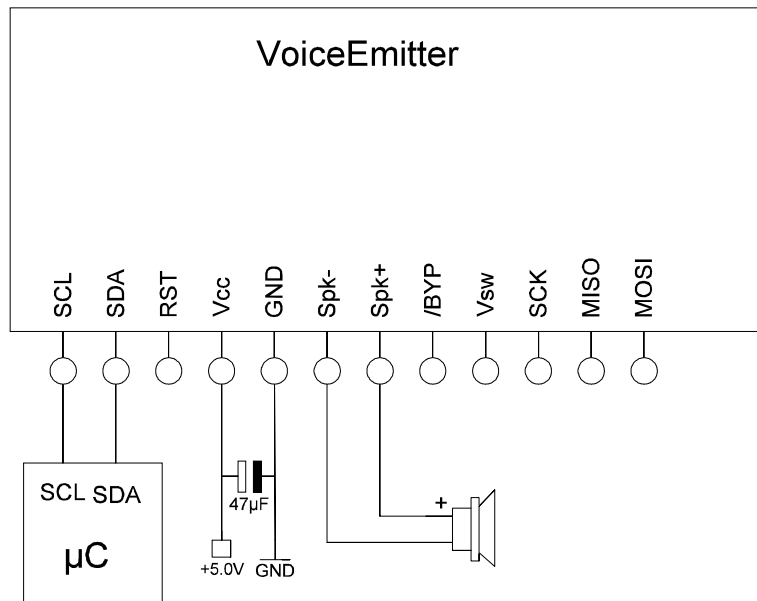


Nach dem Anlegen der Versorgungsspannung gibt der VoiceEmitter die Klanginformationen der Datei STARTUP.WAV auf dem Lautsprecher aus.

1.6.2 Standardbeschaltung

Ziel: Ansteuern des VoiceEmitters über TWI, Ausgabe der Klänge über einen Lautsprecher.

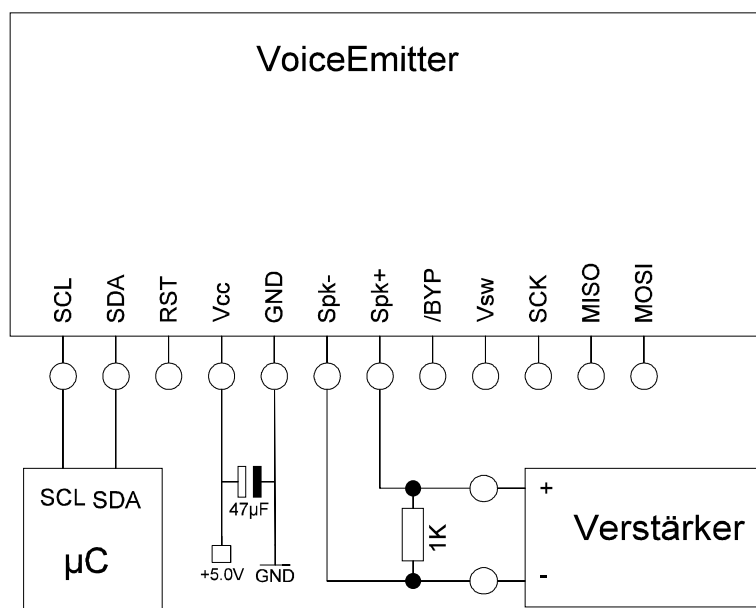
- Stellen Sie die TWI-Adresse, auf die der VoiceEmitter reagieren soll, mit den DIL-Schaltern ein (siehe Kapitel xxx ToDo).
- Verbinden Sie die Anschlüsse SCL und SDA mit Ihrer Steuereinheit.
- Schließen Sie einen Lautsprecher am VoiceEmitter an.
- Schließen Sie Masse und Versorgungsspannung (+5V) an.



1.6.3 Anschluss an externen Verstärker

Ziel: Ansteuern des VoiceEmitters über TWI, Ausgabe der Klänge über einen externen Verstärker

- Stellen Sie die TWI-Adresse, auf die der VoiceEmitter reagieren soll, mit den DIL-Schaltern ein (siehe Kapitel xxx ToDo).
- Verbinden Sie die Anschlüsse SCL und SDA mit Ihrer Steuereinheit.
- Schließen Sie Spk+ und Spk- an einem Verstärker an.
- Spk+ wird durch den VoiceEmitter vom internen Verstärker getrennt, wenn kein Signal ausgegeben wird. Um Knackgeräusche bei offenem Spk+ zu vermeiden, verbinden Sie Spk+ und Spk- über einen 1K-Widerstand.
- Schließen Sie Masse und Versorgungsspannung (+5V) an.



1.7 Sicherheitshinweise

Der VoiceEmitter kann bei elektrostatischen Entladungen oder bei Überspannung zerstört werden. Treffen Sie daher Vorkehrungen, um diese zu vermeiden.

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

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

Verursachen Sie den Lautsprecherausgang (Spk+ und Spk-) niemals kurz.

2 Ansteuerung des VoiceEmitters über den TWI-Bus

2.1 Starten des VoiceEmitters

Der VoiceEmitter startet beim Anlegen der Betriebsspannung automatisch. Wenn Sie den VoiceEmitter 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 SD-Karte noch mit der Initialisierung beschäftigt ist oder je nach Konfiguration noch eine Datei (Startup.wav) abspielt.

2.2 Schritte beim Senden und Empfangen

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

Um Daten an den VoiceEmitter 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 VoiceEmitter
- In Abhängigkeit von ACK oder NACK und der Anzahl der zu sendenden Daten Wiederholung der letzten beiden Schritte
- Senden der Stoppkondition

Um Daten vom VoiceEmitter 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

2.3 Kommunikationsprotokoll

Daten, die zum VoiceEmitter gesendet werden, bestehen aus Befehlen und Parametern. Das genaue Format ist in der Befehlsreferenz, Kapitel xxx ToDo beschrieben.

Der VoiceEmitter 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 VoiceEmitter 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 VoiceEmitter abgerufen, dann sendet der VoiceEmitter 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 VoiceEmitters 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 VoiceEmitter 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 VoiceEmitters 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 VoiceEmitter keine weiteren Befehle annimmt.

Um den vollständigen Abruf aller Daten sicherzustellen, sollte solange lesend auf den VoiceEmitter 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).

2.4 ASCII- und Binär-Modus

Der VoiceEmitter 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 VoiceEmitter 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.

2.5 Statusbyte

Das vom VoiceEmitter 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 SD-Karte erkannt wurde und das Dateisystem kompatibel ist. =0 wenn keine SD-Karte angesprochen werden kann oder nicht kompatibel ist
0	VE_BUSY	=0, wenn der VoiceEmitter keinen Befehl ausführt = 1, wenn der VoiceEmitter beschäftigt ist

2.6 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 VoiceEmitter 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 VoiceEmitter 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

2.7 Formate von Klangdateien

Klangdateien, die der VoiceEmitter 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 VoiceEmitter 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.

2.8 Kompatible SD-Karten

Mit dem VoiceEmitter können verwendet werden

- Mikro-SD-Karten, FAT16-Formatierung
- Mikro-HDSD-Karten (hohe Kapazität), FAT32-Formatierung

2.9 Fehlermeldungen

Die vom VoiceEmitter übermittelte Fehlernummer hat die folgende Bedeutung:

Name	Wert	Bedeutung
VE_ERR_OK	0	kein Fehler
VE_ERR_NO_MEDIUM	1	Keine (kompatible) SD-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 SD-Karte)
FATERR_INVALID_FAT	17	Nicht unterstützte FAT-Version
FATERR_READ	18	Fehler beim Lesen von der SD-Karte
FATERR_BYTES_PER_SEC	19	Ungültige Anzahl Bytes pro Sektor (falsch formatierte oder nicht kompatible SD-Karte)
FATERR_INVALID_VID	20	Ungültige Volumen-ID (falsch formatierte oder nicht kompatible SD-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

3 Befehlsreferenz

3.1 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
- <b4> für einen binären Wert bestehend aus mindestens einem und maximal vier Bytes. Das niederwertigste Byte wird immer zuerst gesendet.

Befehl	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
Ablaufkontrolle				
p	[<z>] <z> := { 0 1 }	[<b1>] <b1> := { 00h 01h }	Spielt eine zuvor durch P oder O geöffnete Datei ab.	Nein
S	[+ -] <d> [% s z]	<b1><b4>	Stellt die aktuelle Abspielposition einer geöffneten Datei ein.	Ja
X	[<z>] <z> := { 0 1 }	[<b1>] <b1> := { 00h 01h }	Stoppt das Abspielen einer Datei	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>, bzw. <b1> spezifiziert wird.	Ja
%			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 VoiceEmitter in den Sleep-Modus wechselt	Nein
A			Stellt den ASCII-Modus ein	Nein
B			Stellt den Binär-Modus ein	Nein

3.2 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

3.3 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.

3.4 Befehl @

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

Der Befehl @ stellt die Anzahl Sekunden ein, nach dem der VoiceEmitter bei Inaktivität in den Sleep-Modus wechselt. Im Sleep-Modus werden der Verstärker und die SD-Karte vollständig abgeschaltet.

Durch Adressierung des VoiceEmitters über den TWI-Bus wird der Stromsparmodus beendet. Die Reaktivierung der SD-Karte und des Verstärkers benötigen etwas Zeit, so dass durch den Sleepmodus die Reaktionszeit des VoiceEmitters verlängert wird (im Bereich von Millisekunden).

3.5 Befehl A

Befe hl	Parameter ASCII-Modus	Parameter Binär-Modus	Busy- Modus

A		Stellt den ASCII-Modus ein	Nein
---	--	----------------------------	------

Mit dem Befehl A stellt der VoiceEmitter 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).

3.6 Befehl B

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

Mit dem Befehl B stellt der VoiceEmitter 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 (Beispiel). 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).

3.7 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.

3.8 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, werden der exakte Dateiinhalt 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..
```

3.9 Befehl I

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

Der Befehl I gibt den Hersteller, Modellinformation und Softwareversion des VoiceEmitters 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

3.10 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.

3.11 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.

3.12 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.

3.13 Befehl p

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

Der Befehl 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.

3.14 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	S 11h 0Ah	Verschiebt die Abspielposition um 10% der Dateilänge zum Ende.

	b4 = 10 = 0Ah		
S-10s	- -> 02h % -> 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).

Anmerkung:

<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 00h 0Ah gesendet werden.

3.15 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 VoiceEmitters 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%)

Anmerkung:

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

3.16 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.

4 Beispiele

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

Der nachfolgende Programmcode beschreibt, wie der VoiceEmitter von einem Mikrocontroller angesprochen werden kann.

Die Beispiele werden in C und in BASCOM dargestellt.

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).

4.1 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 VoiceEmitter kann mit den TWI-Basis-Funktionen wie folgt durchgeführt werden.

```

/* -----
Befehl an den VoiceEmitter senden
bAdr:      TWI-Adresse des VoiceEmitters
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 VoiceEmitter mit den TWI-Basis-Funktionen durchgeführt. Die Besonderheit beim VoiceEmitter ist, das er immer zuerst die Anzahl noch folgender Bytes mitteilt.

```

/* -----
Daten vom VoiceEmitter abholen
bAdr:      TWI-Adresse des VoiceEmitters
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 VoiceEmitter sendet, beginnen immer mit der Anzahl Folgebytes, einem Statusbyte und einer Fehlernummer. Danach folgen ggf. die eigentlichen Nutzdaten wie z.B. Dateiinhalte oder abgefragte Systemparameter.

Wenn der VoiceEmitter 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 VoiceEmitter nicht mehr BUSY ist und keine Daten mehr vorhanden sind, sind alle Daten übertragen worden.

Die folgende Funktion fragt den VoiceEmitter nach Daten ab und blende Status- und Fehlerbyte aus. Sind keine Daten vorhanden, dann wartet sie ab, bis der VoiceEmitter 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 VoiceEmitters
pBuf:      Zeiger auf den zu sendenden Datenbereich
           Achtung: Der Puffer muss 17 Bytes groß sein

Rückgabe: Anzahl gelesener Bytes
           0: es folgen keine weitem 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
        // 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 können nicht mehr generiert werden

return 0;
}

```

Die Funktion `ve_getPayloadData` kann dazu verwendet werden, um zu warten, bis der VoiceEmitter nicht mehr BUSY ist.

```

/* -----
Warten, bis VoiceEmitter nicht mehr BUSY ist
bAdr:      TWI-Adresse des VoiceEmitters
----- */
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 VoiceEmitters abgefragt werden soll, ohne Nutzdaten zu empfangen, kann die folgende Funktion verwendet werden. Vorsicht: wenn der interne Puffer des VoiceEmitters voll ist ab, dann wartet er auf die Abholung der Daten.

```

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

Rückgabe: Fehlernummer des VoiceEmitters
           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 VoiceEmitter gesteuert werden. Das Beispiel ist mit Kommentaren versehen, so dass es ohne weiteren Bemerkungen verständlich sein sollte. Die gezeigten Beispiele verwenden die Dateien auf der mitgelieferten SD-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*)"READY.WAV", strlen("READY.WAV"));

    /* -----
    VoiceEmitter bis 4 zählen lassen
    ----- */
    for (BYTE idx = 1; idx <= 4; idx++)
    {
        // Befehl aufbereiten
        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 abberechnen
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;
}
```

4.2 Beispiele in BASCOM

5 Technische Parameter

Parameter	typisch	minimal	maximal	Einheit
Versorgungsspannung	5V	4,2	5,4	V
Stromaufnahme bei Klangausgabe	80	60	100	mA
Stromaufnahme Standby	58			mA
Stromaufnahme Sleepmodus	18			μ A