Powered by AquesTalk pico

Data Sheet

音声合成 LSI 「AquesTalk pico LSI」

ATP3011

AQUEST
www.a-quest.com

1. 概要

ATP3011 は、1チップでローマ字表記のテキストを音声に変換する音声合成 LSI です。
シリアルインターフェースを介して、簡単に任意の音声メッセージを出力できます。また、あらかじめプリセットしたメッセージを端子のレベル変化で読み上げることもできます。

ATP3011 は、Atmel 社の 8bit マイクロコントローラ ATmega328(または ATmega328P)に、音声合成ミドルウェア AquesTalk pico をファームウェアとして搭載した製品です。電気的特性、外形寸法などに関しては、ATmega328(P)のデータシートを併せて参照ください(付録 13.5 に参照先有り)。

【注意】ATP3011 に対して、チップイレーズ(Erase Device)を実行とファームウェアが消え、以降は音声合成 LSI として機能しなくなります。特にプログラマ(AVRISP mkII など)を接続する場合は十分にご注意ください。

1.1. 特長

- 文字列を送るだけで音声出力するシンプルインターフェース
- 実装に便利な 28pinDIP、小サイズの 32pinTQFP の2種類のパッケージ
- 外付け部品不要(アナログオーディオアンプは別途)
- Arduino uno などの基板に直接装着して使用可能
- 3種のシリアルインターフェース UART/I2C/SPI
- 2種類のカスタマイズ可能なチャイム音
- 数値を適切な読みとアクセントで読み上げ(タグ指定)
- カスタマイズ可能な 15 種類のプリセットメッセージ
- 端子の変化をトリガにプリセットメッセージを発声
- 話速やプリセットメッセージなどを実装後に設定可能
- ユーザ設定専用アプリ PicoRomWriter を別途配布

1.2. 製品型番による違い

型番は ATP3011XX-PP の形式で、XX 部分が声種を、PP 部分がパッケージを示します。
それ以外の機能・動作は同じです。 -PU: 28pin DIP -AU: TQFP32pin

型番	概要
ATP3011F4-PU	女声(かわいい系の女声) 28pinDIP
ATP3011F1-PU	女声("ゆっくり") 28pinDIP
ATP3011M6-PU	男声 28pinDIP
ATP3011R4-PU	ロボット声 28pinDIP
ATP3011F4-AU	女声(かわいい系の女声) TQFP32pin
ATP3011M6-AU	男声 TQFP32pin

2. 仕様

パッケージ	ATP3011XX-PU: 28pin DIP ATP3011XX-AU: 32pin TQFP (0.8 mm Lead Pitch)	
動作電圧	2.5V - 5.5V	
消費電流(typ)	3V 時	5V 時
発声時	3.5mA	6.6mA
スリープ時	1μA	1μA
待機時		
スタンドアロンモード	1μA	1μA
コマンドモード(セーフモード)		
UART	0.5mA	1mA
	(ボーレート自動設定中は 3mA)	(ボーレート自動設定中は 6mA)
I2C	0.07mA	0.1mA
SPI	0.5mA	1mA
ベースチップ	Atmel ATmega328(P)	
音声合成コア	AquesTalk pico	
入力データ	ローマ字表記音声記号列(ASCII)	
声種	1 種（型番により声種が異なる）	
出力	8KHz サンプリング PCM 音声波形を 8bitPWM 出力	
インターフェース	UART/I2C/SPI	
プリセットメッセージ	15 種類	

3. ピン配置

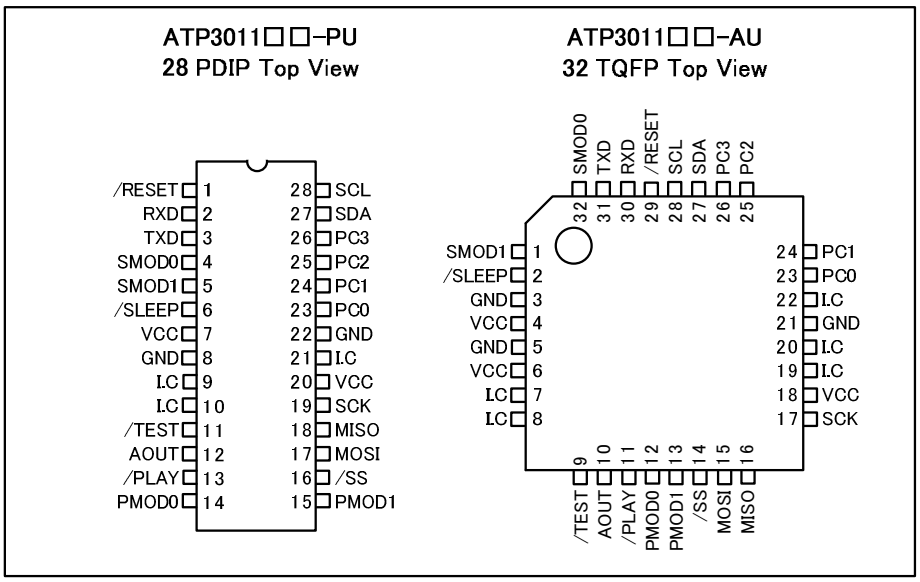


図 1 ピン配置(Top View)

4. ピン機能

表 4.1 ピン機能

端子名	I/O	説明
/RESET	I*	Lowでリセット リセットタイミング等はデバイスのデータシート参照
VCC	-	電源(2.5-5.5V)
GND	-	GND
RXD	I	USART RXD UART通信プロトコル参照
TXD	O	USART TXD UART通信プロトコル参照
SMOD0	I*	通信モード選択 表9.1
SMOD1	I*	通信モード選択
/SLEEP	I	スリープ状態にして消費電流を下げる
I.C	-	未使用 OPENにする
/TEST	I*	TEST用端子 常にOPENまたはプルアップ
AOUT	O	音声出力端子 (0-VDD[V] PWM出力)
/PLAY	O	発声ステータス 発声中はLOWレベルになる
PMOD0	I*	動作モード選択 表7.1
PMOD1	I*	動作モード選択
/SS	I*	SPI Slave Select SPI通信プロトコル参照
MOSI	I	SPI Master Out Slave In
MISO	O	SPI Master In Slave Out
SCK	I	SPI Serial Clock
PC0	I*	プリセットメッセージ選択 表10.1
PC1	I*	プリセットメッセージ選択
PC2	I*	プリセットメッセージ選択
PC3	I*	プリセットメッセージ選択
SDA	I/O	I2C Serial Data
SCL	I	I2C Serial Clock

* 内部でプルアップ

RESET

LOWレベルでリセットされます。リセット後のスタートアップタイムを65msecに設定しています。このため、リセット後のコマンド入力等の操作は、リセット端子の立ち上がりから 80msec 以上あけてから行なってください。リセットのパルス幅等の詳細は、ベースチップのデータシートに準じます。

VCC/GND

2.5V-5.5V の電源を接続します。VCC,GND はそれぞれすべての端子を接続します。

RXD/TXD

UART 通信モード時に使用します。

SMOD0/SMOD1

UART/I2C/SPI のいずれかの通信モードを選択します(9.通信モード 参照)。

SLEEP

消費電流の削減のためのスリープ状態にするときに使用します(10.スリープ 参照)。

AOUT

音声出力端子です。アナログアンプを内蔵していないので直接スピーカに接続することはできませんが、20mA のドライブ能力がありますので、簡単なアナログ回路でスピーカを鳴らすことができます。付録 12.1「参考アナログ回路」も参照ください。

音声信号の周波数帯域は 4KHz までとなっていますが、PWM 出力のため高域のノイズ成分も含まれています。そのため、カットオフ 4KHz のローパスフィルタを追加することで音質を向上させることができます。

PLAY

音声出力中はこの端子が LOW になります。スタンバイ機能付きのアンプに接続して、待機時のアンプの消費電流を抑える、発声状態を LED で示すなどに利用できます。

PMOD0/PMOD1

コマンド入力モード・スタンドアロンモードなどの動作モードを選択します（7.動作モード 参照）

SS/MOSI/MISO/SCK

SPI 通信モード時に使用します。

PC0/PC1/PC2/PC3

スタンドアロンモード時に使用します（11.プリセットメッセージ 参照）。

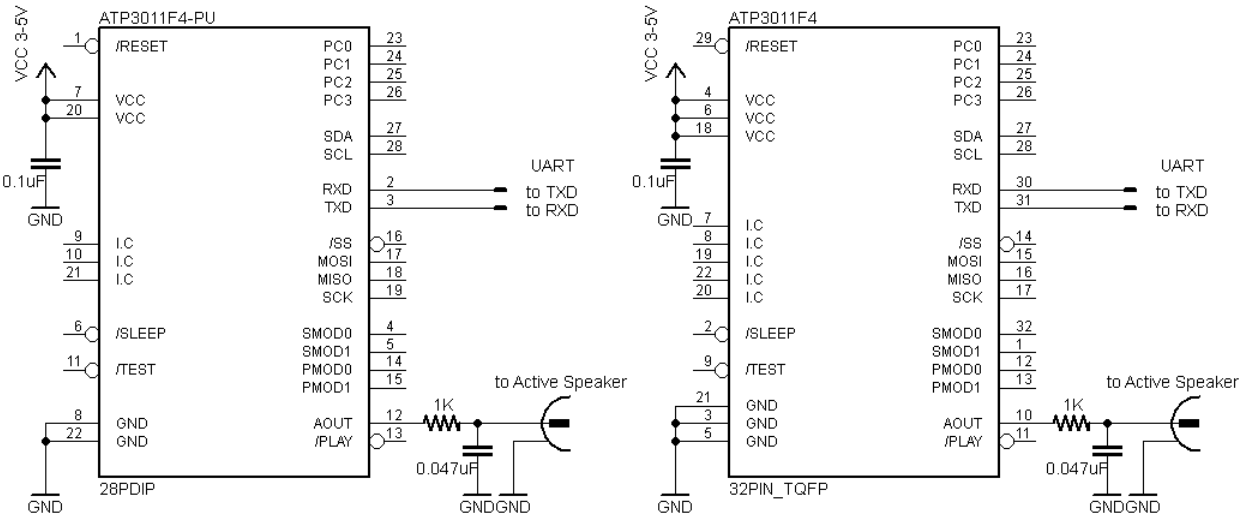
SDA/SCL

I2C 通信モード時に使用します。

I.C

未使用端子です。OPEN にします。

5. 基本回路



6. 内蔵 RC クロックによる注意事項

本製品は、内蔵している RC オシレータで内部の CPU や周辺ユニットを動かしています。この RC オシレータは VCC:3V、動作温度：25℃で 8MHz±10%以内になるように調整されています。また、電源電圧の変動や温度変化によりクロック周波数が変化します（詳細は、ベースチップのデータシート Figure 30-319, Figure 30-320 参照）。

クリスタルオシレータと比べると個体差や環境変動が大きいため、これに伴い音声出力も声の高さなどの個体差が有ります。また、UART 通信のボーレートはこのクロック周波数に依存しますので、動作中に大幅な温度変化や電源電圧変動があるとフレームエラーなどで正常に通信ができなくなる恐れがあります。このような環境で長時間使用する場合は、次の方法を検討ください。

1) I2C や SPI のような内蔵クロック周波数に依存しない通信を利用する。
2) UART 通信では、定期的にボーレートの再調整（9.1 UART 通信 参照）を行う。
3) 外部クリスタルタイプの ATP3010/ATP3012 を使用する。

7. 動作モード

リセット直後に、PMOD1、PMOD0 端子状態に応じて動作モード（表 7.1）が決まります。動作中に変更はできません。

表 7.1 動作モード

PMOD1	PMOD0	動作	説明
1	1	コマンド入力モード	基本モード。シリアル通信で任意のメッセージを音声出力する。
1	0	セーフモード	EEPROMの設定に関わらず、USARTのボーレートが9600bps、I2Cのスレーブアドレスが2EHとなる。それ以外はコマンド入力モードと同じ。
0	1	スタンドアロンモード	プリセットメッセージをPC0-PC3の端子の操作で発声させる
0	0	デモモード	全てのプリセットメッセージを順に繰り返して発声

コマンド入力モード

UART/I2C/SPI のいずれかのシリアル通信でホストからコマンドを送ることで、任意の音声メッセージを出力するモードです。

セーフモード

EEPROM の設定に関わらず、USART のボーレートが 9600bps*、I2C のスレーブアドレスが 2EH になるので、EEPROM を誤って不正な値に設定した場合に使用します。

* 内蔵 RC クロックの誤差（±10%）により、実際には 8640bps から 10560bps の範囲で固体差があるため、USART 通信で文字化けが発声する場合は、ホスト側でボーレートを上下に変更してお試しください。なお、この理由により、通常の USART 通信では、セーフモードを使わずにコマンド入力モードで使う必要があります。

スタンドアロンモード

予め EEPROM に設定した 15 種類のプリセットメッセージを、PC0-PC3 端子のレベル変化で出力することができます。

デモモード

15 種類のプリセットメッセージを順次繰り返して発声出力します。システムの動作確認に使用できます。

8. コマンド

コマンド入力モード時に、ホストと ATP3011 間で通信するコマンドを以下に示します。

具体的な通信手順は、通信モード毎に異なるので次章の通信モードを参照してください。

8.1. コマンドフォーマット

□□□[CR]

1つのコマンドは 1byte～127byte の ASCII 文字列の最後にデリミタとして CR(0x0D)を付与します。

AT30110F4 は、この CR の受信後にコマンドを実行します。なお、コマンドの大文字、小文字は区別されます。

8.2. コマンド応答フォーマット

□□□> or *

コマンドの実行後に ATP3011 が返す応答は、0byte 以上の ASCII 文字列の最後にデリミタとして'>'が追加されたものです。ただし、コマンド実行中は Busy 状態になり、この間に他のコマンドを受信すると、コマンド応答として '* '(0x2A)が返ります。なお、発声中は Busy 状態となります。

8.3. メッセージコマンド：

ローマ字記号列[CR]

指定の音声記号列を音声に変換して出力します。音声記号列はローマ字表記の読みにアクセント情報を付与したものです。詳細は、12 章の「音声記号列仕様」を参照してください。

コマンド応答:発声の完了後に'>'が返ります。音声記号列が正しくないなどエラーの場合は"EXXX>"が返ります(XXX は 3 桁の数値のエラーコード)。ローマ字記号列(半角英数小文字)の長さは最大 127byte です。

8.4. バージョンコマンド

#V[CR] バージョンが返ります。

コマンド応答: 4byte のバージョンコードと'>'が返ります。バージョンコードは "VF4a"等、品番やリビジョンにより異なります。

8.5. チャイムコマンド:

#J[CR] チャイム音 J を出力します。

#K[CR] チャイム音 K を出力します。

コマンド応答: 出力の完了後'>'が返ります。EEPROM のチャイム音データが壊れているなどエラーの場合は"EXXX>"が返ります(XXX は 3 桁の数値のエラーコード)。

8.6. ブレークコマンド:

$ メッセージの発声またはチャイム音の出力を中断します。

'$'(0x24)を送出(CR は付けない)します。ブレークコマンドは busy 状態中も有効です。

コマンド応答: 中断処理の完了後に'E255>'が返ります。

8.7. 内部 EEPROM 読み込みコマンド:

#Rnnn[CR] 内部 EEPROM の指定アドレスの1byte の値が返ります。

nnn:アドレス(000 - 3FF) 3 桁の 16 進数で指定。アルファベットは半角大文字のみ。

mm:値 2桁の 16 進数で指定。アルファベットは半角大文字のみ。

コマンド応答: 'mm>'が返ります(2桁の 16 進数の値)。正しくないコマンドを指定すると"EXXX>"が返ります(XXX は 3 桁の数値のエラーコード)。

8.8. 内部 EEPROM 書き込みコマンド:

#Wnnnmm[CR] 内部 EEPROM の指定アドレスに1byte の値を書きこみます。

nnn:アドレス(000 - 3FF) 3 桁の 16 進数で指定。アルファベットは半角大文字のみ。

mm:書きこむ値を2桁の 16 進数で指定。アルファベットは半角大文字のみ。

コマンド応答: 書き込みの完了後'>'が返ります。正しくないコマンドを指定すると"EXXX>"が返ります(XXX は 3 桁の数値のエラーコード)。

9. 通信モード

リセット直後に、SMOD1、SMOD0 端子状態に応じて次の通信モードが決まります。動作中に変更はできません。

表 9.1 通信モード

SMOD1	SMOD0	通信モード
1	1	UART
1	0	I2C
0	1	SPI(MODE3)
0	0	SPI(MODE0)

9.1. UART 通信

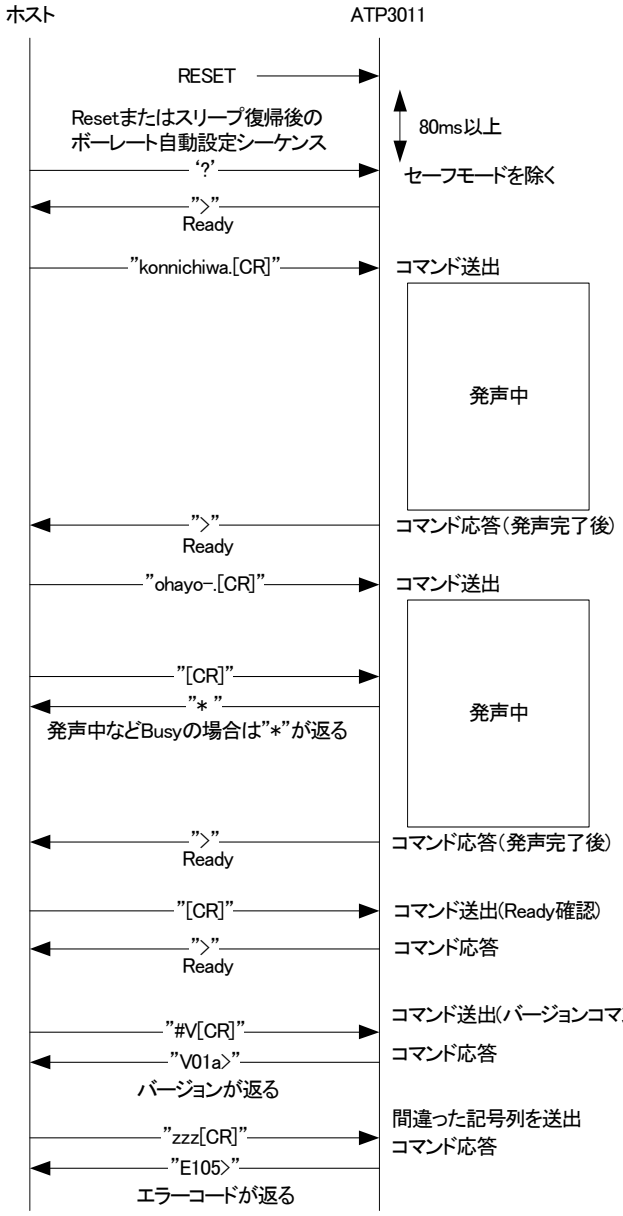
TXD、RXD の2線でホストと通信します。

- ボーレート: 2400～76800 自動設定
- 8ビット、1ストップビット、ノンパリティ

【注意】

・UART 通信では、ボーレート自動設定のため、最初に'?'をホストから送信する必要があります(セーフモードを除く)。

・UARTの電圧レベルはこのLSIの電源レベルに合わせる必要があるので、異なる電源電圧のデバイスと通信する場合はレベルコンバータが必要となります。異なる電圧レベルを与えるとチップが破損します。



ボーレートの自動設定(コマンド入力モード)

リセット後、またはスリープ(10 章)からの復帰後にボーレート自動設定シーケンスになります。ただし、動作モードがセーフモードのときはボーレート自動設定シーケンスにはなりません。

ボーレート自動設定シーケンスに入ると、ATP3011 は受信状態になり、ホスト側から'?'(0x3F)を送信します。ATP3011 はこの信号を元にボーレートを自動設定し、設定したボーレートにて'>'(0x3E)を返し、自動設定シーケンスを完了します。

ボーレート自動設定シーケンスにおいてホストから'?'(0x3F)以外のデータを送ると、ボーレートが適切に設定されず、ホストは正しく'>'(0x3E)を受信できません。この場合は、再リセットや SLEEP 端子を利用してボーレートを再設定します。

【注意】内蔵 RC クロックは電源電圧や動作温度により変動するため(6 章参照)、これに伴い UART 通信のボーレートも変化します。リセット後に動作温度が±40℃以上変化する場合は、そのままではフレームエラー等の通信エラーを生じますので、スリープ機能(10 章)を利用して、適時ボーレートを再設定する必要があります。

コマンド/コマンド応答

コマンドは CR 後に実行されます。

コマンドの実行後は応答メッセージを返します。メッセージコマンドやチャイムコマンドは発声(再生)の終了後に応答を返しますので、ホスト側はこの応答で発声の終了タイミングを知ることができます。

発声中にブレークコマンド "$'(0x24)で発声を中止できます。このときの応答は'E255>'となります。

9.2. I2C 通信

SDA と SCL の2線でホストと通信します。

I2C を使うときは、SDA、SCL 共に 5KΩ程度のプルアップ抵抗が必要です。

- 最大クロックスピード 400KHz
- I2C スレーブとして動作
- ジェネラルコールは未対応
- 7bit アドレス

スレーブアドレスは EEPROM レジスタの EP_I2CADR の設定で変更可能です(出荷時設定は 7bit の 0x2E)。この変更はリセット後に反映されます。

コマンドは分割して(複数の Start/Stop コンデション)送信することもできます。

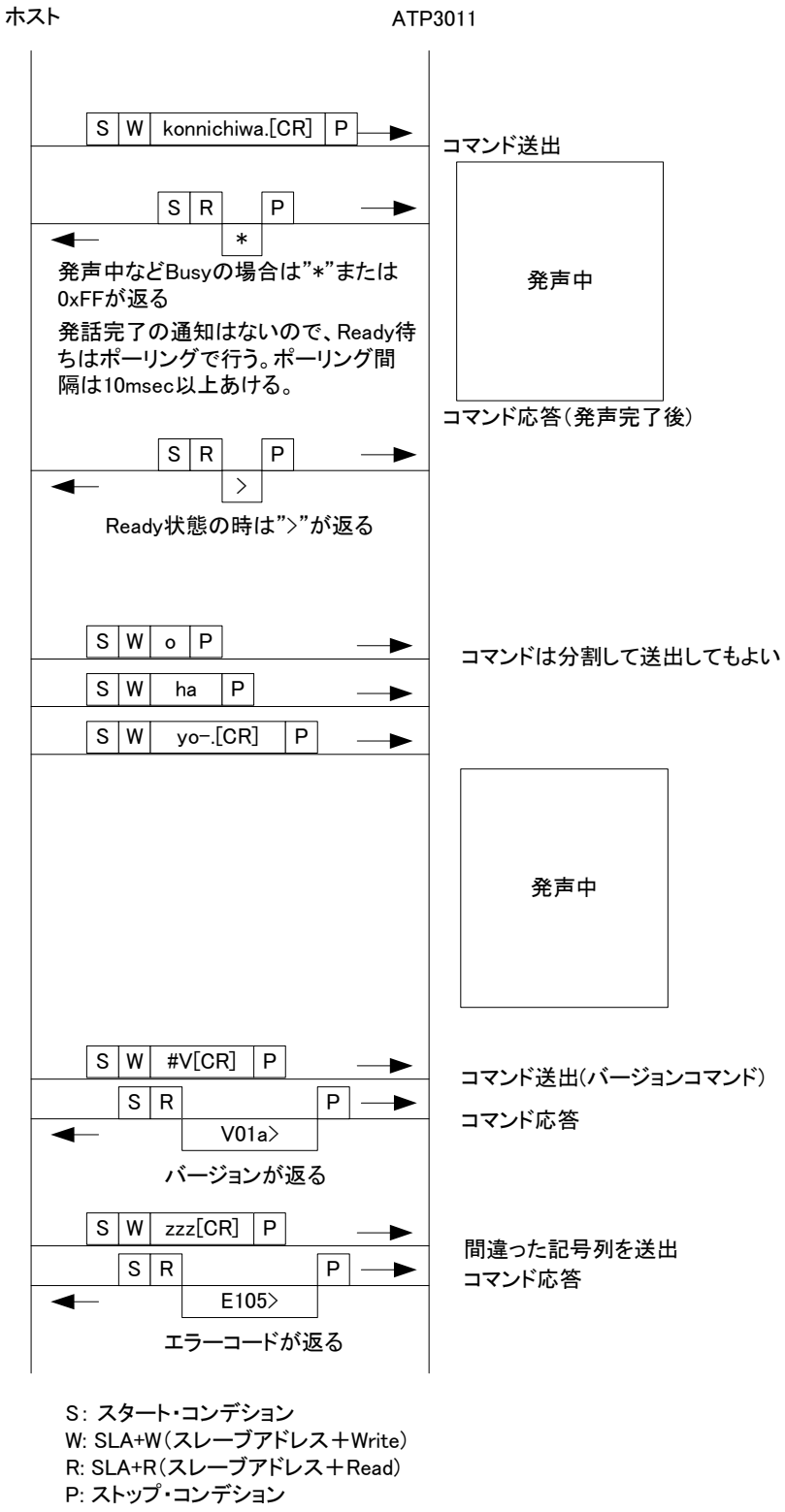
コマンドは CR の送信直後に実行されます。

コマンド応答は基本的に USART と同じです。ただし、USART では非同期に応答が返されますが、I2C では、マスタ側からアクセスしない限り応答は得られません。したがって、発話の完了通知はないので、ポーリングにより Ready 状態を確認することになります。このポーリングの間隔は **10msec** 以上あけてください。短い間隔でポーリングを行うと出力音声が途切れます。

応答データのバイト数以上を要求した場合は、最後の文字(通常'>')が連続して返されます。

NOACK または応答がない場合は、ATP3011 が正常に動作していない可能性が高いのでスレーブアドレス、配線などを確認してください。

【注意】ATP3011 の音声記号列の長さは 127byte まで指定可能ですが、Arduino の Wire.h ライブラリで一度に送れるのは 32byte までという制約があります。



S: スタート・コンデション
W: SLA+W(スレーブアドレス＋Write)
R: SLA+R(スレーブアドレス＋Read)
P: ストップ・コンデション

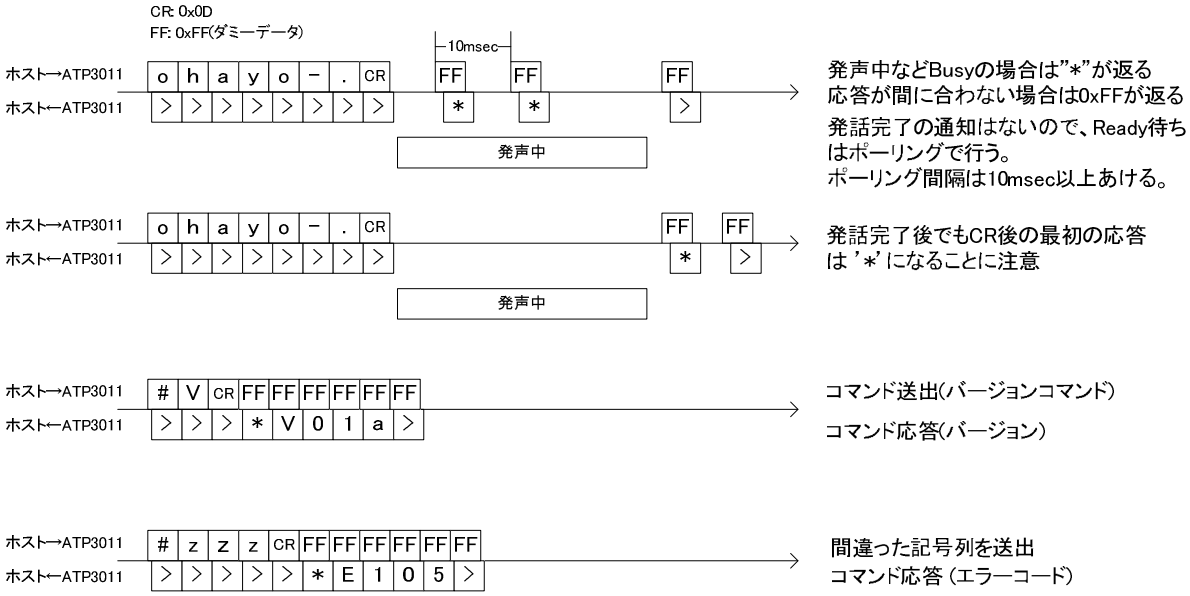
9.3. SPI 通信

SPI スレーブとして、SCK、MISO、MOSI の3線+(/SS)でホストと通信します。

- スレーブセレクト/SS 端子も有効(アクティブ Low)
- SPI モードは Mode 0 または Mode 3 を選択可能(通信モード SMOD1、SMOD0 端子で選択 表 9.1)
- データは8ビットで MSB ファースト
- 最大クロックは 1MHz

SPI コマンド間(byte 間)は **20μsec** 以上の間隔をあけてください。間隔が短いと正しい応答が得られなくなります。
CR 後の最初の応答は必ず '*' となります。すでに Ready 状態になった後でも最初の応答は '*' が返ります。
ポーリングによる応答確認は前項「I2C 通信」を参照ください。

SPI 通信手順



10. スリープ機能

SLEEP 端子を LOW にするとスリープ状態になります。この状態では内部クロックが停止し消費電流が数μA まで低下します。
【注意】スリープ機能を使わないとき、SLEEP 端子は OPEN で構いませんが、スリープ端子を使用するときは HI、LOW は適切な電圧を外部から与えてください。オープンコレクタの出力に接続する場合はプルアップが必要です。

スリープ移行

待機状態(コマンドの入力を受け付けている状態)では、SLEEP 端子の立下りエッジでスリープ状態に移行します。発声中などコマンド実行中はスリープ状態になりません。コマンド実行処理が終了して待機状態になるタイミングで SLEEP 端子が LOW であればスリープ状態になります。なお、通信途中(UART/I2C/SPI)に SLEEP 端子を LOW にしてはいけません。

スリープ状態中

スリープ状態では、すべての通信や PC0、PC1、PC2、PC3 端子の操作は機能しません。出力端子はスリープ移行前の状態が保存されます(AOUT は Low、/PLAY は Hi レベル)。

スリープ復帰

SLEEP 端子の立ち上がりエッジで、スリープ状態から復帰します。

10.1. スリープ復帰によるボーレートの再設定

動作モードがコマンド入力モードで、かつ、通信モードが UART の場合には、スリープ復帰時にボーレート自動設定シーケンス(9.1 参照)に入ります。そこで、まずホスト側から'?'(0x3F)を送信してボーレートを再設定する必要があります。

本機能により、長期間のスリープ状態での温度等の環境変化によってホストとのボーレートがずれることを防ぐことができます。

11. プリセットメッセージ

ATP3011 は、15 種類のプリセットメッセージを内部 EEPROM に設定することができます。

設定は、基本的に EEPROM 設定プログラム PicoRomWriter.exe(別途ダウンロード)を用いて行います。

プリセットメッセージは、ローマ字表記の音声記号列(null 終端)で表現し、1 メッセージは 128byte(null 終端含む)以下で、15 メッセージ全体の長さ(null 終端を含む)は 960byte 以下に制限されています。

プリセットメッセージは、動作モードがスタンドアロンモードの時に、PC0、PC1、PC2、PC3 いずれかの端子の変化をトリガとしてメッセージを発声します。なお、チャタリング防止のため変化から約 10msec 後の PC0、PC1、PC2、PC3 端子状態で発声するメッセージを確定します(表 11.1)。なお、発声中のトリガは予約されずに無視されます。

15 種類のメッセージを 3 つの端子の論理値で表現するため、プッシュスイッチ等で操作するときは図 7 の回路を参考にしてください。

表 11.1 端子状態とメッセージ

PC3	PC2	PC1	PC0	メッセージ
0	0	0	0	MSG0
0	0	0	1	MSG1
0	0	1	0	MSG2
0	0	1	1	MSG3
0	1	0	0	MSG4
0	1	0	1	MSG5
0	1	1	0	MSG6
0	1	1	1	MSG7
1	0	0	0	MSG8
1	0	0	1	MSG9
1	0	1	0	MSG10
1	0	1	1	MSG11
1	1	0	0	MSG12
1	1	0	1	MSG13
1	1	1	0	MSG14
1	1	1	1	発声なし

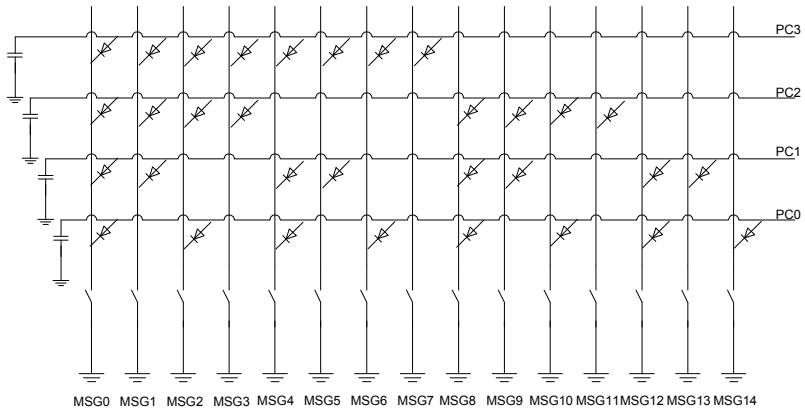


図7 スタンドアロンモード(プリセットメッセージ)回路例

12. 内部 EEPROM

ATP3011 は、内部 EEPROM データを変更することで発話スピードやプリセットメッセージなどの各種のユーザ設定を行うことができます。

内部 EEPROM は、アドレス 0x000 から 0x3FF までの 1Kbyte ですべての領域が予約されています。各アドレスの定義を表 10.1 に示します。EP_MSG を除いてバイナリで指定します。なお、2byte のデータはリトルエンディアンです。例えば、EP_SPEED はアドレス 0x002 が LSB 側 8bit を示し、0x003 が MSB 側 8bit のデータとなります。

内部 EEPROM は、内部 EEPROM 書き込みコマンド #Wnnnmm[CR] で変更することができます。また、EEPROM 設定プログラム PicoRomWriter を使用するのも簡単です。PicoRomWriter は当社ホームページのダウンロードページからダウンロードできます。その使い方は付属のマニュアルを参照してください。

表 12.1 EEPROM レジスタ

アドレス	名称	出荷時	内容
0x000-0x001	RESERVED		予約済
0x002-0x003	EP_SPEED	100	発話速度（50-300 default:100）値が大きいほど速くなる。変更は次のメッセージコマンドから有効。
0x004-0x005	EP_PAUSE	65535	文末のポーズの長さ[125usec単位] 65535の指定で自動設定。連続して発声させないときは256を指定する。
0x006	EP_I2CADR	0x2E	I2Cのスレーブアドレス(0x08-0x77) 変更はリセット後に反映。
0x008-0x01D	EP_CHIME0		チャイム音データ コマンド#Jのチャイム音。変更は次のチャイムコマンドから有効。
0x020-0x035	EP_CHIME1		チャイム音データ コマンド#Kのチャイム音。変更は次のチャイムコマンドから有効。
0x038-0x03A	RESERVED		予約済 変更不可
0x040-0x3FF	EP_MSG		プリセットメッセージｘ15 詳細は「図8 プリセットメッセージ配置」参照。変更は次のスタンドアロンモード時のトリガから有効。

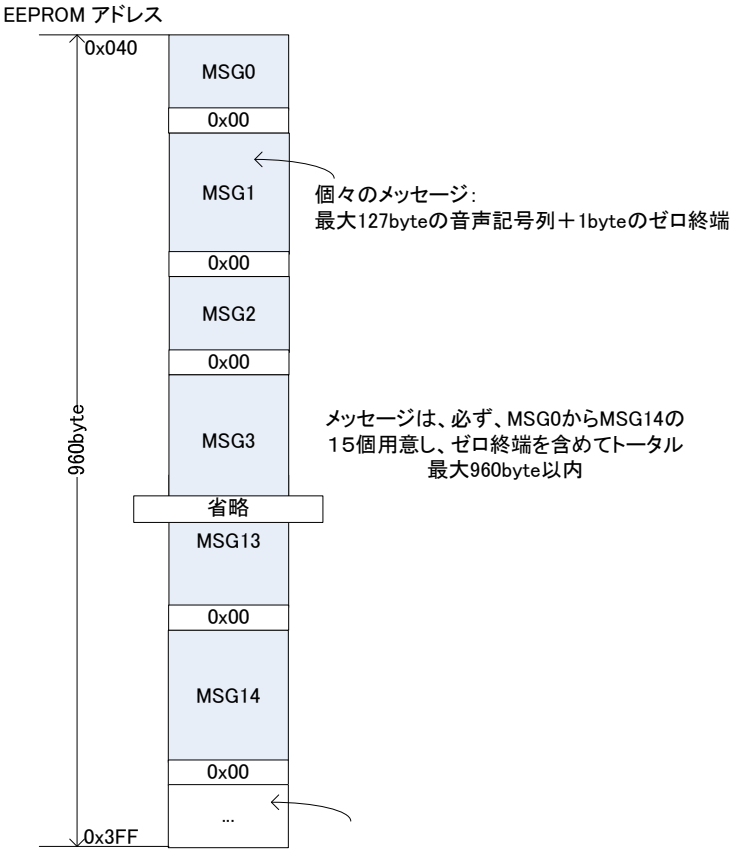
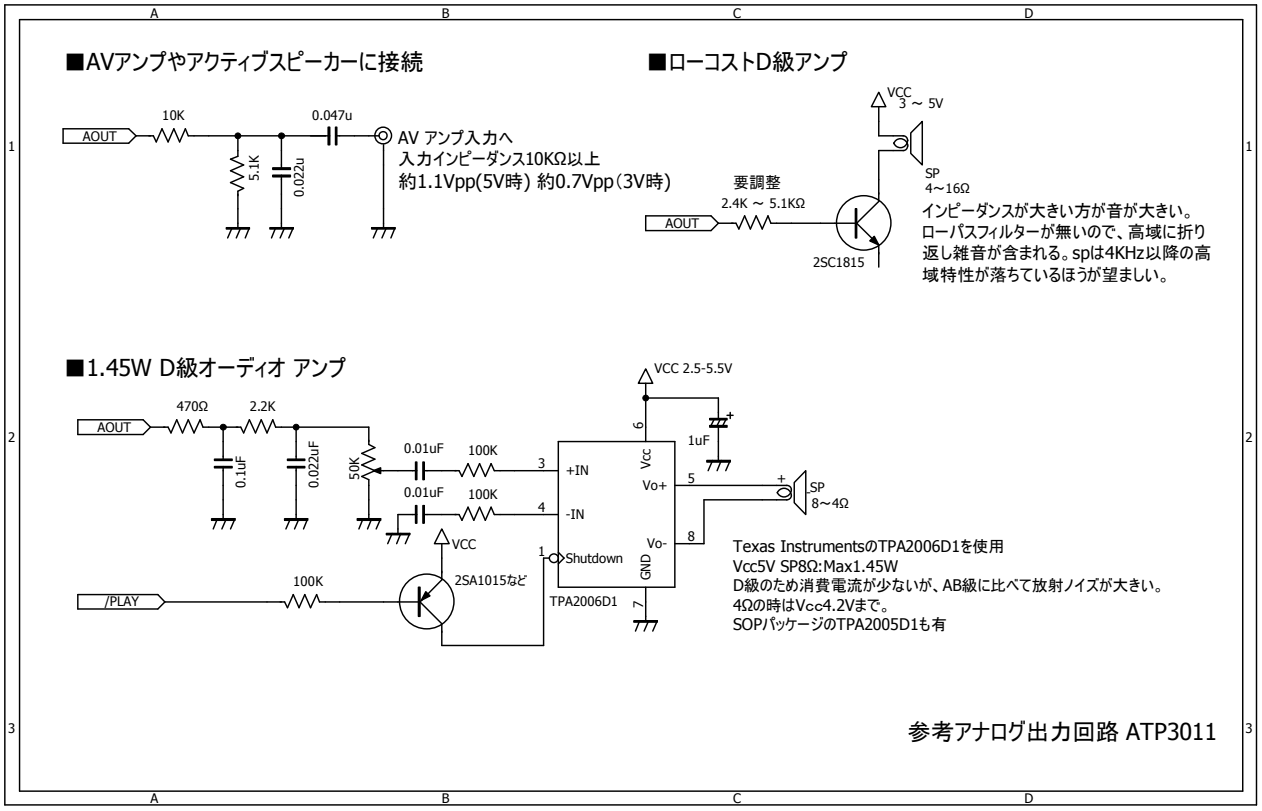


図 8 プリセットメッセージ配置

13. 付録

13.1. アナログ・オーディオ回路（参考回路）



他の IC を使用した参考回路を http://blog-yama.a-quest.com/?eid=970162 で紹介しています。

13.2. エラーコード表

値	内容
100	その他のエラー
101	メモリ不足
105	音声記号列に未定義の読み記号が指定された
106	音声記号列のタグの指定が正しくない
107	タグの長さが制限を越えている（または[>]がみつからない）
200	音声記号列が長すぎる
251	コマンドの指定が正しくない
254	チャイム音データが壊れている
203	ヒープメモリ不足
255	（Warning）発声中にブレークされた

13.3. Fuse/Lock Bit の設定値

出荷時のヒューズビット、ロックビットの設定は下記の通りです。

値の示す意味については、ベースチップのデータシートを参照してください。

名前	値
FUSE BIT	EXTENDED:0xFF HIGH:0xD7 LOW:0xE2
LOCK BIT	0xFC

13.4. Arduino ボードを利用した簡易動作

ATP3011 のベースチップは Arduino uno と互換のチップであるので、Arduino uno に実装されている ATmega328(P)を、DIP 版の ATP3011 に差し替えることで、電源やシリアルインターフェースを用意することなく、簡単に ATP3011 を評価、動作確認することができます。また、当社では動作確認していませんが、Arduino Diecimila/Duemilanove など ATmega328(P)や ATmega168 を使用している他の Arduino ボードでも同様に利用できると思います。

使用例：

1. Arduino uno ボードの ATmega328(P)と DIP 版の ATP3011 を差換える（向きに注意）。
2. Arduino uno ボードの DIGITAL 6 番ピンと、GND ピンをアクティブスピーカーに接続（出力レベルが高いので必要に応じて抵抗で分圧してください）する。
3. USB で PC と Arduino uno ボードを接続する。
4. PC で Arduino IDE を起動し、MENU>Tools>Serial Port で、com ポートを設定します。
   (com ポート番号を設定は Arduino uno の使用方法に従ってください)
5. MENU>Tools>Serial Monitor を開く。
6. Serial Monitor 右下のコンボボックスで、Carriage return と 9600bps を選択。
7. Send ボックスに"?” 1 文字を入力して、send ボタンを押下してボーレートを設定。
8. Send ボックスに"konnnichiwa.”を入力して、send ボタンを押下すると、「こんにちは」とアクティブスピーカーから音声が聞こえます。

ArduinoIDE の Serial Monitor の代わりに TeraTerm などの他のターミナルソフトを使用する場合、「改行コード 受信:CR+LF 送信:CR」「ローカルエコー:する」の設定を推薦します。

参考 :「AquesTalk pico LSI を Arduino 基板に挿して音声合成」

You Tube http://www.youtube.com/watch?v=p9rTu4J5sjc

【注意】ATP3011 は Arduino のブートローダが入っていませんので、スケッチの実行など Arduino 用として使用することはできません。

13.5. ベースチップのデータシート

ATmega328(P)チップのデータシートは下記を参照ください。

「8-bit Atmel Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash/ATmega328(P)」

http://www.atmel.com/ja/jp/devices/ATMEGA328.aspx?tab=documents

13.6. ATP3011 と ATP3010F4 の主な違い

	ATP3011	ATP3010F4
動作クロック	内蔵 RC オシレータ(8MHz)	外部クリスタルまたはセラロック(16MHz)
動作電圧	2.5V - 5.5V	3.8V - 5.5V
UART 通信ボーレート	設定シーケンスによる自動設定	EEPROM に設定

14. 音声記号列仕様

14.1. 音声記号列の記述方法

音声記号列とは、生成する音声を文字コードで表現したものです。音声記号列は ATP3011 の入力データとなります。ATP3011 は単語辞書を持っていませんので、漢字を読むことができません。したがって、基本的にはローマ字の文字列で表現することになります。

まず、簡単な音声記号列の例を示します。

```
korewa onse-ki'go-de_su.
```

助詞の「は」と長音

上記の例を普通にひらがなで記述したら「これは、おんせいきごうです。」となりますが、音声記号列では、これとは若干異なる表記を行います。

まず、助詞の「は」が、「wa になります。また、「おんせい」が「onse-」、「きごう」が「kigo-」になっています。音声記号列では、通常の表記方法にとらわれずに**音**として記述する必要があります。

助詞の「は」は、音声記号列では「wa」を、長音や二重母音は、必要に応じて長音記号の「-」(半角ハイフン 0x2D)を指定します。助詞の「を」は、「o」と記述します。

他には、「言う」を「yuu」などにするなど、もあります。いずれも、発音を基準に読みを変更してください。

アクセント記号

次に、「'」(0x27)の部分ですが、これはアクセント記号と呼び、アクセントを指定するための記号です。音声記号列を指定するときは、読み記号だけでなくアクセント記号を正しい位置に付与する必要があります。アクセントは正確な音声を生成するためにはかかせません。

次の例のように、アクセントが異なると意味も異なってしまいます。

```
ka'resi.
karesi.
```

```
ku'rabu.
kurabu.
```

アクセント記号は、アクセント核の読み記号の後ろに付与すると定義していますが、音の高さが「高→低」に変化する部分にアクセント記号をつけると考えるとよいと思います。
最初の例の「これわ」の部分に見られるように、アクセントが平板で「高→低」に変化しない場合にはアクセント記号はつけません。

なお、アクセント記号をつけるときの注意点として、必ず読み記号の後に指定してください。読み記号は、「jyu」のように複数の文字から成るので、「j'yu」と指定するとエラーになりますので、必ず「jyu'」と指定します。

```
j'yunbi dekitayo.  ×
```

```
jyu'nbi dekitayo.  ○
```

句切記号

次に、もう少し複雑な音声記号列を示します。

```
ko'ndowa mo-suko'si/fukuzatuna/onse-ki'go-de_su.
```

この例では、新たな記号「/」が出てきました。この記号は区切り記号と呼ばれるもののの一つで、アクセント句の境界を示すものです。

ここで、日本語の単語を一つの単位としてみると、単語の中には音の高さが「高→低」へと変化する箇所が１つあるか、または１つもないかのいずれかという法則があります。
発音記号列では、この単語という単位を拡張して、単語に助詞なども含め、この高さがシフトする部分をひとつだけ（あるいは無し）含むような区分単位を「アクセント句」と定義しています。ちょっと難しいので、まずは、助詞も含めた単語の単位と考えると良いでしょう。
句切記号は、このアクセント句の境界を指定するための記号です。

上の例では、「/」が無いと、前のアクセントの継続となりますので、音の高低を示すと次のようになり、ちょっとおかしくなりますね。

```
こんどは、もーすこしふくざつなおんせーきごーです。
```

正しいアクセントを表現するためには、アクセント記号と共に区切り記号の指定も重要です。

```
こんどは、もーすこしふくざつなおんせーきごーです。
```

また、一つのアクセント句に、２つ以上のアクセント記号を指定することはできません。次の例は間違いです。

```
hito'tunoakusento'kudesu.    ×
```

句切記号としては、「/」のほかに「;」「+」(共に半角)があります。また、読点「 」(半角スペース）句点「.」（ピリオド）も区切り記号に含まれます。すべての句切記号はアクセント句の句切りを表現していますが、それぞれ異なった機能を持っています。次にこれを説明します。

句切記号の違い

句切記号は、以下の種類、定義となっています。

. (0x2E)	この位置にポーズ(無音区間)が入ります。文の終わりを示します。
? (0x3F)	この位置にポーズ(無音区間)が入ります。文の終わりを示します。 文末の声が高めになります。
(0x20) 半角空白	この位置にポーズ(無音区間)が入ります。 文中の息継ぎの部分に指定します。一般に次の音が高くなります。
, (0x2C)	この位置に短いポーズ(無音区間)が入ります。 半角空白と機能は同じですが、無音区間の長さが短くなります。
; (0x3B)	次のアクセント句が比較的高い音で始まります。ポーズは入りません。
/ (0x2F)	通常のアクセント句の句切に指定します。ポーズは入りません。
+ (0x2B)	前後のアクセント句の句切があいまいな場合に指定します。ポーズは入りません。

それでは、句切記号の違いが合成音声にどのように影響するかを考えてみます。
次の例は、句切記号の種類を変えただけのものです。

```
a'kusento;na'dono/kanametona'ru,
```

```
a'kusento/na'dono/kanametona'ru,

a'kusento+na'dono/kanametona'ru,

a'kusentonadono/kanametona'ru,
```

実際に合成音を聴取しないと、なかなかわからないのですが、アクセント句「nadono」の部分が、上の例ほど強調されるようになります。

これは、下の例ほど、句切記号前後のアクセント句の分離度が弱くなる、逆に言うとより密に結合していることを示しています。基本的に２つのアクセント句の結合が密であるほど、後ろ側のアクセントは弱くなります。

最後の例は、「aakusento」と「ndono」２つのアクセント句を１つのアクセントに結合したものです。このようにすると、後ろ側のアクセントは消失します。

また、上の例の「かなめとなる」の部分も、「kanameto+na'ru」「kanameto/na'ru」のように記述することも可能です。

上記の例で、どれが正解なのかは決まっていません。実際に合成した音声を聞いて、適切な句切記号を選択していくことが必要でしょう。

文末の句切記号

音声記号列の最後の句切記号として、「.」のほかにも、「,」「?」を指定できます。
「.」の代わりに「,」を指定すると、最後の音の高さが比較的高く終了します。体言止めなど、文の終了感をあまり出したくない場合には「,」で文を終了すると良いと思います。

```
fairuo/hozonn,
```

また、疑問文などで、文末の音を高くしたい場合には、「?」を指定します。ただし、疑問文でも文末を高くしない場合もありますので、そのときは、「.」を指定します。

```
rokuon+sima'suka?
```

無声化

無声化とは、無声子音に挟まれた、母音の[i][u]が声帯の振動を伴わずに発声される現象を言います。例えば、喉に手を当てて「ありますか？」と発声してみると、「す」の母音部分では手に振動が感じられないと思います。

無声化を行う場合には、次のように"_"(半角アンダーバー)のついた読み記号を使用します。

```
e'rume_suno/a'_kusesari-.
```

アンダーバーを付けられる読み記号は限られていますので、読み記号表に定義されているものだけを使う必要があります。

数値読みタグ（棒読みタグ）

次に、AquesTalk の大きな特徴である、数値読み機能を使った例を示します。

```
denwaba'ngo-wa <NUM VAL=01-2345-6789>de_su.
```

「<」「>」記号で囲まれた部分が数値読みのタグ部分です。タグを使って数値を直接指定して読み上げることがで

きます。

もしタグを使わなければ、次のように指定しなくてはなりません。この場合、連濁や促音化による読みの変化、アクセント結合によるアクセント位置の変化などを考慮して記述する必要があり、数値が固定でない限り現実的とはいえません。

```
zeroi'chi ni-sa'nn/yongo- rokuna'na/hachikyu'-de_su.
```

上記の数値読みタグの例は、棒読みの場合の指定方法で、他に桁読みの指定方法がありますが、こちらは後述します。
棒読みタグの書式は次の通りで、半角の英数記号で記述します。

```
<NUM VAL=(数値)>
```

(数値)の部分には、「0」～「9」までの数値と「-」「.」の並びで構成します。
「-」は強制的に読みの句切りをつける場合に使用し、この指定が無ければ、内部のルールにより適当に区切られます。
したがって、最初の例を次のように指定することも可能です。

```
denwaba'ngo-wa <NUM VAL=0123456789>de_su.
```

「.」は小数点で、「てん」と発声します。

```
pa'iwa <NUM VAL=3.1415926535897932>.
```

数値読みタグ（桁読みタグ）

次は、桁読みの指定方法です。

```
kino'-wa <NUMK VAL=321162567>+de_sita.
```

この指定で、「三億、二千百十六万、二千五百六十七」と読みます。
桁読みタグの書式は次の通りで、半角の英数記号で記述します。

```
<NUMK VAL=(数値)>
```

(数値)の部分には、「0」～「9」までの数値と「.」の並びで構成します。桁読みでは数値部分に「-」や「,」は指定できません。
「.」は小数点で、「てん」と発声し、小数点以降は棒読みします。

ところで、桁読みのときには、助数詞の影響をうけて数詞・助数詞の読みやアクセントが変化することが知られています。
有名なところでは、

```
一本　　いっぽん （促音化）
二本　　にほん
三本　　さんぼん （濁音化）
 ・
```

このような変化は、タグの中に助数詞を指定することで自動的に処理することができます。
書式は次の通りです。

```
<NUMK VAL=(数値) COUNTER=(助数詞)>
```

(助数詞)の部分には、助数詞の読み記号(アクセント記号も可）の並びを指定します。
例えば、次のように指定します。

```
nokori+ji'kanwa a'to/<NUMK VAL=10 COUNTER=funn>de_su.
```

この場合の読み及びアクセントの変化は、ライブラリ内部のルールに従って行われます。
AquesTalk サンプルアプリの[助数詞つきの数値の桁読み]で、様々な数値と助数詞を指定して確認してみてください。

英数読みタグ

アルファベット、数値、記号を、適切な読みとアクセントで読み上げます。
数値は、棒読みになります。
また、内部ルールで自動的に区切って読み上げられ、また、区切り位置を指定することも可能です。

```
<ALPHA VAL=(英数)> または <ALPHA VAL="(英数)">
```

(英数）の部分は、「0」-「9」(半角数値）、「a」-「z」、「A」-「Z」(半角英数小文字/大文字)、半角記号(巻末の「記号読み表」を参照）の並びで構成します。

```
ko-doba'ngo-wa <ALPHA VAL=AT-3568P>de_su.
```

上記は、次のように指定したのと同じようになります。

```
ko-doba'ngo-wa e-thi'- ha'ifun san/go'-/roku/ha'chi pi'-de_su.
```

(英数）の部分に、区切り指定が無い場合でも、内部のルールにより適宜区切られて発声されます。

```
<ALPHA VAL=abcdef>.
```

上記は、次のように指定したのと同じようになります。

```
e-/bi-/si-/dhi'- i-/e'fu.
```

その他

読み記号は基本的な音韻をすべて用意していますが、一部の外来音韻に定義されていないものがあります。例えば「gwi」は未定義であり、指定するとエラーになります。この場合は、他の読み記号を連結して、それらしいものを指定してください。この例では、「gui」のように指定します。定義されている読み記号は、本文書後部の「読み記号表」を参照してください。

```
bi'-ruo/gui'tto/nomita'ina.
```

読点記号はいわゆる書き言葉の読点よりも、多めに指定したほうが良いようです。

次の例に示すように、音声記号列は複数の文を連続して指定することも可能です。しかしながら、1 度に指定できる文字数が 127byte と制限があり、また、タグを多用した場合などでは内部のメモリ不足で生成エラーになる場合があります。なるべく 1 文毎に指定するほうが良いでしょう。

```
sandaruo tukkaketoyuu.cyo'tto/ma'tteo ta'nnmatoyuu.
```

14.2. 音声記号列仕様

文字コード

音声記号列の文字コードは、ASCII 半角記号で指定します。全角や漢字コードを指定することはできません。また、規定されている文字コード以外、例えば、Tab・LF 等を含めることはできません。

記号の種類

音声記号列は、次の４種類の記号を並べて構成します。

1. 読み記号
2. アクセント記号
3. 句切記号
4. タグ記号

読み記号

読みをローマ字で指定します。一つ一つが日本語の音節に、ほぼ該当します。使用可能な読み記号を「読み記号表」に示します。

アクセント記号

アクセントの位置を「'」(0x27)で指定します。アクセント記号はアクセント核の読み記号の直後に指定します。アクセント記号の数は、１つのアクセント句（書式規定参照）に１つ、あるいは指定なしとします。

句切記号

アクセント句の境界を示し、次の種類があります。

. (0x2E)	文の終わりに指定
? (0x3F)	文の終わりに指定(疑問形)
空白(0x20)	呼気段落の境界に指定　ポーズ有
, (0x2C)	呼気段落の境界に指定　ポーズ有（ 空白より短いポーズ）
; (0x3B)	声立てを行う場合に指定　ポーズ無
/ (0x2F)	通常のアクセント句の句切り　ポーズ無
+ (0x2B)	後ろのアクセント句が副次アクセントの場合に指定　ポーズ無

タグ記号

現在、棒読み、桁読み、英数読みの３つのタグが定義されています。
これらは、数値や英数字を直接指定する場合に利用します。それぞれのタグ記号の書式は、次の通りです。なお、助数詞の長さは、31 文字以下に制限されています。

棒読み	<NUM VAL=(数値)>
桁読み	<NUMK VAL=(数値)[COUNTER=(助数詞)]>
英数読み	<ALPHA VAL=(英数)> または、<ALPHA VAL="(英数)">

（[]は任意の指定を示す）

棒読みの(数値)は、「0」～「9」までの数値と「-」「.」の並びで構成します。
「-」は、読みを句切る位置に指定します。
「.」は、小数点で「てん」と発声します。

桁読みの(数値)は、「0」～「9」までの数値と「.」の並びで構成します。
数値の上限は兆の位までで、最大、9999999999999999 となります。
「.」は小数点で「てん」と発声し、小数点以降は、棒読みになります。

(助数詞)の部分には、助数詞の読み記号(アクセント記号も可)の並びを指定します。句切記号は指定できません。

「対応数詞一覧表」にあるものが詳細に対応している助数詞です。それ以外の助数詞を指定した場合は、内部のデフォルトルールにしたがって変更されます。

英数読みタグは、アルファベット、数値、記号を、適切な読みとアクセントで読み上げます。
数値は、棒読みになります。内部ルールで自動的に区切って読みます。
(英数)の部分は、「0」-「9」(半角数値)、「a」-「z」、「A」-「Z」(半角英数小文字/大文字)、半角記号の並びで構成します。指定できる記号は「読み記号表」を参照してください。

書式規定

音声記号列の記号の並びには、制約があります。以下にその規定を示します。

```
音声記号列 ：   文 [文...]
文 ：           アクセント句 句切記号 [(アクセント句 句切記号)...]
アクセント句 ： 読み記号[読み記号...][アクセント記号][読み記号...]
タグ記号 :=     読み記号[読み記号...]
```

ひとつの音声記号列には、複数の文を含めることができます。もちろん、音声記号列が一つの文でも構いません。

文は、一つ以上の、アクセント句と区切り記号の組からなります。したがって、文の最後は必ず区切り記号になります。

アクセント句は、一つ以上の読み記号とアクセント記号からなります。1つのアクセント句内のアクセント記号は１つあるいは指定無しのいずれかとなります。

タグ記号は、内部的には、読み記号の列として処理されます。

読み記号並びの制限

読み記号は基本的には任意の読み記号を並べることができますが、以下に示すような通常用いない組み合わせは正しく読めない場合があります。

1. 語頭の長音
アクセント句の先頭に「-」。言い換えると文頭または区切り記号の次に「-」は指定できません。
例）　「-ka.」　「watasi;-wa,」

2. 無声化音＋（有声音）、 無声化音＋（長音）

通常、無声化は無声音に挟まれた場合の現象ですので、無声音の読み記号（アンダーバー指定）の後に有声音や長音を指定すると正しく発声しません。
例）　「nai_su-.」　「a_kiya.」

14.3. 音声記号列サンプル

```
dennwaba'nngo-wa <NUM VAL=01-2345-6789>desu.
sa-ba-;<NUM VAL=3512>no/ha-dodhi'_su_kuni e'ra-+haxtuse-.
getuyo'-no/<NUMK VAL=21 COUNTER=di>kara <NUMK VAL=8 COUNTER=cha'nnneru>de/yoyaku+sima'sita.
ryo'-kinnwa;<NUMK VAL=550 COUNTER=enn>desu.
sumimase'nn <NUMK VAL=10 COUNTER=funn>+okurema'su.
<NUMK VAL=20 COUNTER=funn>ni e'kide/ma'xtutemasu.
asunote'nnki to-kyo- hare'noti+kumori saiko-ki'onn <NUMK VAL=25 COUNTER=do>.
<NUMK VAL=100 COUNTER=me'-toru>saki ko-ennirigutino/ko-satenno+hidaride'su.
konosaki;<NUMK VAL=3 COUNTER=kiro>/ju-taichu-.tu-kadi'kann;<NUMK VAL=10 COUNTER=funn>
```

```
yorosi'idesuka?
koredei'i?
baxtuteri-no/ju-denn+kannryo-.
<NUMK VAL=2006 COUNTER=nenn> <NUMK VAL=1 COUNTER=gatu>;<NUMK VAL=15 COUNTER=niti>.
<NUMK VAL=16 COUNTER=di>;<NUMK VAL=5 COUNTER=funn>/<NUMK VAL=35 COUNTER=byo->desu.
bakuonnga ginnse'kaino/ko-gennni/hirogaru.
```

14.4. 読み記号表

あ	い	う	え	お	ん		ー(長音)	っ(促音)
a	i	u	e	o	nn/n		-	xtu/子音の重ね
か	き	く	け	こ	きゃ	きゅ	きぇ	きょ
ka	ki	ku	ke	ko	kya	kyu	kye	kyo
さ	し	す	せ	そ	しゃ	しゅ	しぇ	しょ
sa	si/shi	su	se	so	sya/sha	syu/shu	sye/she	syo/sho
た	ち	つ	て	と	ちゃ	ちゅ	ちぇ	ちょ
ta	ti/chi	tu/tsu	te	to	tya/cha/cya	tyu/chu/cyu	tye/che/cye	tyo/cho/cyo
な	に	ぬ	ね	の	にゃ	にゅ	にぇ	にょ
na	ni	nu	ne	no	nya	nyu	nye	nyo
は	ひ	ふ	へ	ほ	ひゃ	ひゅ	ひぇ	ひょ
ha	hi	hu/fu	he	ho	hya	hyu	hye	hyo
ま	み	む	め	も	みゃ	みゅ	みぇ	みょ
ma	mi	mu	me	mo	mya	myu	mye	myo
や		ゆ	いぇ	よ				
ya		yu	ye	yo				
ら	り	る	れ	ろ	りゃ	りゅ	りぇ	りょ
ra	ri	ru	re	ro	rya	ryu	rye	ryo
わ	うぃ		うぇ	を				
wa	wi		we	wo				
が	ぎ	ぐ	げ	ご	ぎゃ	ぎゅ	ぎぇ	ぎょ
ga	gi	gu	ge	go	gya	gyu	gye	gyo
ざ	じ	ず	ぜ	ぞ	じゃ	じゅ	じぇ	じょ
za	zi/ji	zu	ze	zo	ja/jya	ju/jyu	je/jye	jo/jyo
だ	ぢ	づ	で	ど	ぢゃ	ぢゅ	ぢぇ	ぢょ
da	di	du	de	do	zya/dya	zyu/dyu	zye/dye	zyo/dyo
ば	び	ぶ	べ	ぼ	びゃ	びゅ	びぇ	びょ
pa	pi	pu	pe	po	bya	byu	bye	byo
ぱ	ぴ	ぷ	ぺ	ぽ	ぴゃ	ぴゅ	ぴぇ	ぴょ
ba	bi	bu	be	bo	pya	pyu	pye	pyo
ヴァ	ヴィ	ヴ	ヴェ	ヴォ				
va	vi	vu	ve	vo				
つぁ	つぃ		つぇ	つぉ	すぃ	てぃ	とぅ	てゅ
tsa	tsi		tse	tso	swi	thi	twu	thu
ふぁ	ふぃ		ふぇ	ふぉ	ずぃ	でぃ	どぅ	でゅ
fa	fi		fe	fo	zwi	dhi	dwu	dhu

無声化読み記号

キ	ク	ヒ	フ	フィ	シ	シュ	ス	スィ
_ki	_ku	_hi	_fu/_hu	_fi	_si/_shi	_syu/_shu	_su	_swi
ティ	トゥ	ピ	プ		チ	チュ	ツ	ツィ
_thi	_twu	_pi	_pu		_ti/_chi	_tyu/_cyu/_chu	_tu/_tsu	_tsi

14.5. 対応助数詞一覧表

この表以外の助数詞を指定した場合は、デフォルトルールが適応されます

年	nenn
月	gatu
日	nichi
時	ji
分	funn
秒	byo-
円	enn
回	ka'i
階	kai
ヶ月	ka'getu
カロリー	ka'rori-
級	kyu-
行	gyo-
曲	kyoku

キロ	kiro
件	kenn
個	ko
人	ninn
才	sai
時間	ji'kann
台	dai
丁目	cyo-me
月	tuki
番	bann
本	honn
匹	hiki
%	pa-se'nto

14.6. 記号読み表

記号の読みは実装により異なる場合があります

!	びっくり
#	しゃーぷ
$	どる
%	ぱーせんと
&	あんど
*	あすた
+	ぷらす
,	かんま
-	はいふん
.	どっと
/	すらっしゅ

:	ころん
;	せみころん
<	しょーなり
=	いこーる
>	だいなり
?	はてな
@	あっと
¥	えん
^	はっと
_	あんだー
~~(スペース)~~	~~、(区切り)~~

15. 改変履歴

2012/1/19	新規作成
2012/02/21	「Arduino ボードを利用…」でボーレート設定手順を追加
2012/03/22	ベースチップに ATmega328P を追加。ボーレート自動設定の注意の記述を追加
2012/03/31	アナログ・オーディオ回路（参考回路）の1つを MC34119 から HT82V739 のものに変更
2012/06/04	各声種版の共通マニュアルに。HT82V739 のアナログアンプのフィルタ 2 次に変更
2014/10/21	誤字脱字修正
2015/01/27	アナログ・オーディオ回路（参考回路）の HT82V739 を LM4871 などに変更
2016/04/07	UART のシーケンスに RESET 追加