SKRIPSI

PERANCANGAN DAN PEMBUATAN SISTEM PENGKONVERSI TULISAN KE SUARA SEBAGAI SARANA PEMBELAJARAN PEMBACAAN KATA DALAM BAHASA INGGRIS

Disusun Oleh:

EKO VERIONO

NIM : 04.12.253

JURUSAN TEKNIK ELEKTRO S-1

KONSENTRASI TEKNIK ELEKTRONIKA

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI NASIONAL MALANG

AGUSTUS 2010

ADUSTUS SCYLLA
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
KONSENTRASI TEKNIK ELEKTRONIKA
JURUSAN TEKNIK ELEKTRO S-1

NIM : 04.17.009
ERIC VERNON

PENGENDALIAN KECEPATAN MOTOR INDUKSI 3 FASA
MENGGUNAKAN METODA PENGATURAN FREKUENSI
PENGENDALIAN KECEPATAN MOTOR INDUKSI 3 FASA

LEMBAR PERSETUJUAN

PERANCANGAN DAN PEMBUATAN SISTEM PENGKONVERSI TULISAN KE SUARA SEBAGAI SARANA PEMBELAJARAN PEMBACAAN KATA DALAM BAHASA INGGRIS

SKRIPSI

Disusun dan Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Teknik Elektronika Strata Satu (S-1)

Disusun Oleh :
EKO VERIONO
NIM : 04.12.253

Diperiksa dan Disetujui,

Dosen Pembimbing

Sotyohadi, ST. MSc
NIP.Y. 1039700309

Mengetahui,
Ketua Jurusan Teknik Elektro S-1

Ir. Yusuf Ismail Nakhoda, MT.
NIP Y. 1018800189

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2010

PERKUMPULAN PENGELOLAH PENDIDIKAN DAN TEKNOLOGI NASIONAL
MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

NI (PERSERO) MALANG Kampus I: Jl. Bendungan Sigura-gura No.2 Telp (0341)551431 (Hunting) Fax.(0341)553015 Malang
5
ANK NIAGA MALANG Kampus II:Jl.Raya Karanglo, Km 2 Telp.(0341)417634 Malang

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

Nama Mahasiswa : Eko Veriono
N.I.M : 04.12.253
Jurusan : Teknik Elektro S-1
Konsentrasi : Teknik Elektronika
Judul Skripsi : *"Perancangan Dan Pembuatan Sistem Pengkonversi Tulisan Ke Suara Sebagai Sarana Untuk Pembelajaran Pembacaan Kata Dalam Bahasa Inggris"*

Dipertahankan dihadapan majelis penguji skripsi Jenjang Strata Satu (S-1) pada :

Hari : Sabtu
Tanggal : 21 Agustus 2010
Dengan Nilai : 84,1 (A)

PANITIA UJIAN SKRIPSI

KETUA,

Ir. Yusuf Ismail Nakhoda, MT.
NIP.Y. 1018800189

ANGGOTA PENGUJI

PENGUJI I

Joseph Dedy Irawan, ST, MT
NIP. 197404162005011002

PENGUJI II

Ir. Eko Nurcahyo
NIP.Y 1028700172

ABSTRAK

PERANCANGAN DAN PEMBUATAN SISTEM PENGKONVERSI TULISAN KE SUARA SEBAGAI SARANA UNTUK PEMBELAJARAN PEMBACAAN KATA DALAM BAHASA INGGRIS

Eko Veriono
04.12.253
Jurusan Teknik Elektronika – Institut Teknologi Nasional Malang
Jln. Raya Karanglo Km 2 Malang
X_tro2253@yahoo.com
Dosen Pembimbing : Sotyohadi, ST, MSc

Kata Kunci : Konversi Teks Ke Suara, Modul SP03, Atmega16, I2C

Sebagaian besar orang menganggap bahwa belajar Bahasa Inggris itu menyenangkan. Namun, tidak jarang juga orang merasa bahwa belajar Bahasa Inggris itu sangat susah. Salah satu kendalanya adalah teknik pengucapan yang berbeda dengan Bahasa Indonesia. Kesalahan pengucapan seringkali dapat menyebabkan arti yang berbeda. Berdasarkan pada permasalahan tersebut, pada penyusunan skripsi ini dibuat sebuah alat yang dapat mengkonversikan tulisan ke suara dalam logat Bahasa Inggris agar dapat membantu seseorang untuk belajar mengucapkan kata dalam Bahasa Inggris.

Alat ini memanfaatkan modul SP03 *Text To Speech*. Modul ini dapat menerima karakater berstandar ASCII (*American Standart Code For Information Interchange*). Sebagai pengendali sistem digunakan mikrokontroler ATmega16. Mikrokontroler ini akan menerima kode scan dari *keyboard* dan mengkonversikannya menjadi karakter berstandart kode ASCII. Setelah dikonversikan, maka mikrokontroler akan mengirimkan kode ASCII tersebut ke modul SP03 untuk diubah menjadi ucapan. Komunikasi yang digunakan antara mikrokontroler dengan keyboard adalah secara serial sinkron melalui jalur clock dan jalur data. Sedangkan komunikasi yang digunakan antara mikrokontroler dengan modul SP03 adalah secara serial sinkron berstandar I2C.

KATA PENGANTAR

Dengan memanjatkan puji syukur kepada Tuhan Yang Maha Esa, atas berkat, rahmat dan karunia-Nya sehingga dapat menyelesaikan skripsi yang berjudul "Perancangan Dan Pembuatan Sistem Pengkonversi Tulisan ke Suara Sebagai Sarana Untuk Pembelajaran Pembacaan Kata Dalam Bahasa Inggris" dengan lancar. Skrispsi ini merupakan persyaratan kelulusan studi di jurusan Teknik Elektro S-1 konsentrasi Teknik Elektronika ITN dan untuk mencapai gelar sarjana teknik.

Keberhasilan peyelesaian laporan skripsi ini tidak lepas dari dukungan dan bantuan berbagai pihak. Untuk itu penyusun menyampaikan terima kasih kepada :

1. Bapak Prof. Dr. Eng. Ir. Abraham Lomi, MSEE selaku Rektor ITN Malang
2. Bapak Ir. Sidik Noertjahjono, MT selaku Dekan Fakultas Teknologi Industri.
3. Bapak Ir. Yusuf Ismail Nakhoda, MT selaku Ketua Jurusan Teknik Elektro S-1.
4. Bapak Sotyohadi, ST, MSc selaku Dosen Pembimbing.
5. Orangtua serta saudara-saudara kami yang telah memberikan doa restu, motivasi, semangat dan biaya.
6. Rekan-rekan Instruktur di Laboratorium Kendali Industri.
7. Semua yang telah membantu dalam penyelesaian penyusunan skripsi ini.

Penulis telah berusaha semaksimal mungkin dan menyadari sepenuhnya akan keterbatasan pengetahuan dalam menyelasaikan laporan ini. Untuk itu penyusun mengharapkan saran dan kritik yang membangun dari pembaca demi kesempurnaan laporan ini.

Harapan penyusun semoga laporan skripsi ini memberikan manfaat bagi perkembangan ilmu pengetahuan dan pembaca.

Malang, Agustus 2010

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1. Latar Belakang

Beberapa orang menganggap bahwa belajar Bahasa Inggris itu menyenangkan. Namun tidak jarang juga orang menganggap bahwa belajar Bahasa Inggris itu sangat susah. Salah satu kendalanya adalah cara pembacaan kata yang berbeda dengan bahasa Indonesia. Bagi kita yang tidak terbiasa berbicara dalam Bahasa Inggris, tentu akan sangat susah melafalkan kata-kata dalam Bahasa Inggris. Padahal kesalahan pengucapan kata dalam Bahasa Inggris bisa mengakibatkan makna yang berbeda.

Berdasarkan pada permasalah tersebut, dalam penyusunan skripsi ini akan dirancang sebuah alat yang dapat memberikan contoh pelafalan kata Bahasa Inggris . Alat ini memanfaatkan modul *text to speech* SP03 sebagai pengkonversi tulisan ke suara. Tentunya hasil konversi tulisan tersebut akan menggunakan logat dalam Bahasa Inggris. Jadi akan tidak cocok apabila kita memasukkan kata dalam Bahasa Indonesia untuk dikonversikan menjadi suara. Untuk penulisan kata yang akan dikonversikan, *user* dapat mengetikkan kata tersebut pada *keyboard* seperti halnya yang digunakan pada PC. Sedangkan untuk tampilan digunakann LCD 2x16. Kata yang diketikkan pada *keyboard* akan dikirimkan ke mikrokontroler untuk kemudian diteruskan ke modul SP03. Bagian yang berfungsi untuk mengkonversikan teks ke suara adalah modul SP03. Modul ini dapat menerima karakter alfabet dan numerik dengan standar ASCII (*American Standard Code of*

Information Interchange). Komunikasi yang digunakan antara mikrokontroler dengan modul SP03 adalah secara serial I2C (Inter-Integrated Circuit).

1.2. Rumusan Masalah

Mengacu pada permasalahan yang ada, maka rumusan masalah dalam skripsi ini adalah:

1. Bagaimana merancang dan membuat alat yang dapat mengkonversikan teks ke suara dalam bahasa Inggris
2. Bagaimana merancang komunikasi antara mikrokontroler dengan modul *text to speech* SP03 secara serial I2C.
3. Bagaimana merancang komunikasi antara *keyboard* dengan mikrokontroller.
4. Bagaimana merancang rangkaian LCD agar dapat berfungsi sebagai tampilan teks.
5. Bagaimana membuat perangkat lunak untuk mengendalikan sistem secara keseluruhan.

1.3. Batasan Masalah

Untuk menjaga agar tidak melebarnya masalah yang dibahas dan sesuai dengan tujuan yang diharapkan, maka penulis akan membatasi pembahasan permasalahan yang ada yaitu:

1. Modul *text to speech* hanya dapat membaca teks dengan logat bahasa Inggris.
2. Modul SP03 hanya dapat menerima karakter alfabet dan numerik berstandar ASCII.

3. Alat dirancang untuk membaca kata atau frasa singkat dalam bahasa Inggris
4. Banyaknya karakter yang diketikkan oleh *user* maksimal adalah 24 karakter.

1.4. Tujuan

Tujuan skripsi ini adalah merancang dan membuat alat yang dapat mengkonversikan teks ke suara sebagai sarana pembelajaran untuk membaca kata dalam bahasa Inggris.

1.5. Metodologi Perancangan

Metodologi dalam perancangan dan pembuatan sistem pengkonversi tulisan ke suara sebagai sarana pembelajaran pembacaan kata dalam Bahasa Inggris adalah sebagai berikut:

1. Studi literatur untuk memahami cara kerja dari modul *text to speech*, pemrogrmanan mikrokontroller ATmega16, serta komunikasi serial sinkron antara *keyboard* dengan mikrokontroller.
2. Pada tahap realisasi alat yang dibuat, dilakukan perancangan alat yang meliputi merancang rangkaian untuk tiap-tiap blok dan rancangan rangkaian keseluruhan sistem, pembuatan PCB, dan perakitan hasil rancangan.
3. Untuk mengetahui cara kerja alat, maka dilakukan pengujian tiap blok dan pengujian secara keseluruhan. Pengujian tiap blok meliputi pengujian : (1) Komunikasi antara *keyboard* dengan mikrokontroller menggunakan LCD sebagai tampilannya, (2) Pengujian komunikasi antara mikrokontroller dengan modul *text to speech* menggunakan I2C *bus*, (3) Pengujian modul

text to speech sendiri. Untuk pengujian keseluruhan alat dilakukan dengan memasukkan kata melalui *keyboard* dengan LCD sebagai tampilannya kemudian mengkonversikan kata tersebut ke suara.

1.6. Sistematika Pembahasan

Sistematika pembahasan dari skripsi ini terdiri dari pokok pembahasan yang saling berkaitan antara satu dengan lainnya, yaitu :

BAB I Pendahuluan

Pada bab ini dibahas tentang latar belakang permasalahan, rumusan masalah, batasan masalah, sistematika pembahasan dari alat yang direncanakan.

BAB II Landasan Teori

Pada bab ini dibahas tentang teori-teori yang mendukung dalam perencanaan dan pembuatan alat ini

BAB III Perencanaan Dan Pembuatan Alat

Pada bab ini dibahas tentang perencanaan dan pembuatan keseluruhan sistem perangkat keras (hardware) dan perangkat lunak (software).

BAB IV Pengujian Alat

Pada bab ini dibahas tentang proses serta hasil dari pengujian alat, yang didasarkan oleh pengukuran-pengukuran.

BAB V Penutup

Pada bab ini akan disampaikan kesimpulan dari perencanaan dan pembuatan sistem ini.

BAB II

DASAR TEORI

2.1 Pendahuluan

Pada bab ini akan dibahas mengenai teori penunjang dari peralatan yang direncanakan. Pokok pembahasan pada bab ini adalah :

1. Konversi Teks Ke Suara
2. Modul SP03 *Teks To Speech*
3. AT *keyboard* (PS/2 connector)
4. *Inter Integreted Circuit* (I2C)
5. IC LA4440
6. LCD M1632
7. Mikrokontroller ATmega16

2.2 Konversi Teks Ke Suara

Pada modul *SP03 Text To Speech*, bagian utama yang berfungsi untuk mengkonversikan tulisan ke suara adalah sebuah IC WTS701. IC ini dapat menerima inputan berupa teks dan memprosesnya sehingga menghasilkan keluaran berupa ucapan.

Pada umunya suatu sistem pensintesa ucapan atau *Text To Speech* terdiri dari dua sub sistem yaitu:

1) Bagian Konverter Teks ke Fonem (*Text to Phoneme*)
2) Bagian Konverter Fonem Ke Ucapan (*Phoneme to Speech*)

2.2.1 Konverter Teks Ke Fonem

Bagian Konverter Teks ke Fonem berfungsi untuk mengolah teks masukan menjadi urutan kode-kode bunyi yang kemudian direpresentasikan dengan kode-kode fonem. Kode fonem ini merupakan suatu segmen bunyi terkecil yang menyusun suatu kata.

Setiap fonem harus dilengkapi dengan informasi durasi dan *pitch*. Informasi durasi diperlukan untuk menentukan berapa lama suatu fonem diucapkan, sedangkan informasi *pitch* diperlukan untuk menentukan tinggi rendahnya nada pengucapan suatu fonem. Durasi dan *pitch* bersama-sama akan membentuk intonasi suatu ucapan. Kedua informasi ini dalam suatu sistem TTS biasanya dibangkitkan oleh modul pembangkit intonasi. Berikut ini adalah tabel yang menampilkan kode-kode fonem.

Tabel 2.1. Kode Fonem

Vowels			Consonants		
Phoneme	Hex Value	Example	Phoneme	Hex Value	Example
I	0x69	Beat	P	0x70	pet
I	0x49	B*i*t	T	0x74	ten
E	0x65	Bait	K	0x6b	*k*it
E	0x45	Bet	B	0x62	*b*et
@	0x40	Bat	D	0x64	*d*ebt
U	0x75	B*oo*t	G	0x67	*g*et
U	0x55	B*oo*k	H	0x68	*h*at
0	0x6f	B*oa*t	F	0x66	*f*at
C	0x63	B*ough*t	T	0x54	*th*ing
A	0x61	B*o*b	D	0x44	*th*at
A	0x41	B*u*t	S	0x73	*s*at
R	0x52	B*urr*	S	0x53	*sh*ut
O	0x4f	B*oy*	V	0x76	*V*at
Y	0x59	B*uy*	Z	0x7a	*Z*oo
W	0x57	D*ow*n	Z	0x5a	A*z*ure
X	0x78	*A*bout	Y	0x79	*y*ou

X	0x58	Roses	W	0x77	*W*it
			R	0x72	*R*ent
			L	0x6c	*L*et
			M	0x6d	*M*et
			N	0x6e	*N*et
			G	0x47	Si*ng*
			C	0x43	*Ch*urch
			J	0x4a	*J*udge
			P	0x50	But*t*er
			Q	0x51	Wri*tt*en

Sumber. Data Sheet IC WTS701

2.2.2 Konvert Fonem Ke Ucapan

Bagian Konverter Fonem ke Ucapan akan menerima masukan berupa kode-kode fonem serta *pitch* dan durasi yang dihasilkan oleh bagian sebelumnya. Pada bagian ini akan dihasilkan bunyi atau sinyal ucapan sesuai dengan teks yang telah dimasukkan.

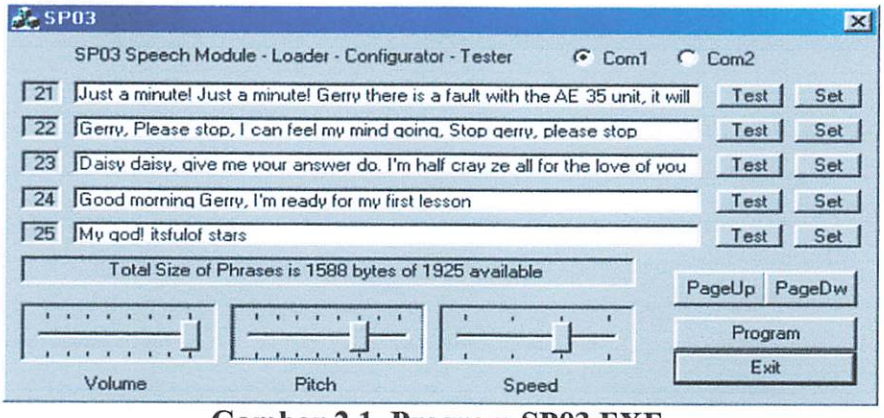
Teknik yang seringkali digunakan pada bagian ini adalah *diphone concatenation.* Teknik ini bekerja dengan cara menggabungkan segmen-segmen bunyi yang telah direkam sebelumnya. Setiap segmen berupa *diphone* atau gabungan dari dua buah fonem. Namun demikian, pada setiap sub-sistem ini terdapat sifat-sifat serta proses-proses yang sangat spesifik dan sangat tergantung dari bahasanya.

Sub sistem ini harus memiliki pustaka setiap unit ucapan dari suatu bahasa. Pada sistem yang menggunakan teknik *diphone concatenation,* sistem harus didukung oleh suatu *diphone database* yang berisi rekaman segmen-segmen ucapan yang berupa diphone. Ucapan dalam suatu bahasa dibentuk dari satu set bunyi yang mungkin berbeda untuk setiap bahasa, oleh karena itu setiap bahasa harus dilengkapi dengan diphone database yang berbeda.

2.3 Modul Text To Speech (TTS) SP03

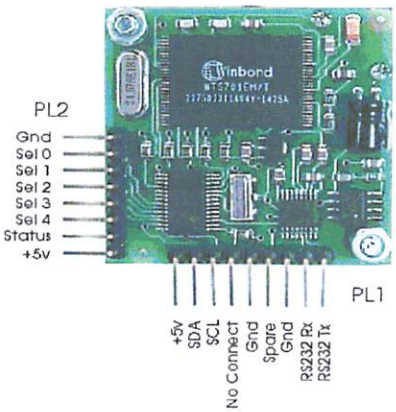
Modul *Text to speech* (TTS) *SP03* merupakan suatu modul yang berfungsi untuk mengubah teks ke suara/ucapan. Modul ini dapat menerima karakter berstandar ASCII untuk dikonversikan menjadi ucapan. Bagian utama penyusunnya adalah sebuah IC WTS701 yang dapat mengubah teks menjadi sinyal suara. IC tersebut dikontrol oleh sebuah mikroprosesor yaitu PIC.

Teks yang ingin dikonversikan dapat didownloadkan ke modul SP03 baik dengan PC ataupun dengan mikrokontroller. Untuk mendownloadkan teks dengan PC dibutuhkan sebuah *software* yang bernama SP03.EXE. Dengan *software* ini kita dapat *mendownloadkan* 30 buah frasa kedalam 30 buah *buffer* yang dimiliki oleh modul. Tiap-tiap frasa tersebut dapat dikonversikan ke suara dengan cara mengakses *buffer* yang memiliki alamat dari 0x01 sampai dengan 0x1E. Sebagai contoh apabila kita menginginkan konversi pada frasa pertama maka kita akan memerintahkan modul untuk mengakses *buffer* dengan alamat 0x00 dan memberikan *command* serta instruksi-instruksi untuk mengkonversikan teks pada *buffer* yang telah ditunjuk.



Gambar 2.1. Program SP03.EXE
Sumber. *Manual Book SP03 Speech Shynthesizer*

Terdapat tiga macam cara yang dapat digunakan untuk mengakses *buffer* dan memberikan instruksi pada modul SP03, yaitu komunikasi secara serial dengan RS232, komunikasi dengan *I2C bus*, dan komunikasi secara paralel. Untuk komunikasi secara serial RS232 menggunakan pin Rx dan TX, untuk komunikasi secara I2C bus menggunakan pin SDA dan SCL, sedangkan untuk komunikasi secara paralel menggunakan pin-pin pada PL2. Gambar berikut ini menunjukkan pin-pin yang dimiliki oleh modul SP03.



Gambar 2.2. Modul *Text To Speech SP03*
Sumber. *Manual Book SP03 Speech Shynthesizer*

Selain menggunakan PC, untuk mendownloadkan teks inputan ke modul juga dapat dilakukan dengan menggunakan mikrokontroller. Untuk fasilitas ini hanya disediakan sebuah *buffer* untuk penyimpanan teks dengan kapasitas sebesar 81 *byte*. Komunikasi antara modul dengan mikrokontroller hanya dapat dilakukan dengan dua cara yaitu secara serial dengan RS232 dan serial menggunakan I2C *bus*. Pada skripsi ini komunikasi dilakukan dengan menggunakan I2C *bus*.

Pin yang digunakan untuk komunikasi menggunakan I2C *bus* adalah pin SCL dan SDA. Untuk frekuensi kerja100kHz disarankan agar menggunakan resistor *pull-up* sebesar 4k7, sedangkan untuk frekuensi kerja lebih dari 400kHz disarankan untuk menggunakan resistor *pull-up* sebesar 1k8. Beberapa hal yang

perlu diperhatikan dalam komunikasi menggunakan I2C bus adalah sebagai berikut:

- **SP03 I2C address**

  SP03 memiliki alamat I2C bus yaitu 0xC4 untuk mode tulis (*write*) dan 0xC5 untuk mode baca (*read*).

- **Command Register**

  Register ini terletak pada alamat 0x00. Isi dari register ini menentukan apa yang akan dilakukan oleh modul.

Tabel 2.1. *Command Register*

Command	Action
NOP (0x00)	*Load Text to Buffer*
SPKPRE 1 to 30, or 0x01 to 0x1E)	*Speak pre-defined phrase*
SPKBUF 64 or 0x40	*Speak text previously stored in buffer*

Untuk melakukan pengisian teks ke *buffer*, register *command* diisi dengan data *byte* 0x00. Sedangkan untuk melakukan proses konversi dari teks ke suara kita dapat mengisikan alamat *buffer* yang kita inginkan kedalam register *command*. Alamat *buffer* dari 0x01 sampai dengan 0x1E merupakan alamat *buffer* yang pengisian teksnya dilakukan dengan PC. Sedangkan untuk teks yang didowloadkan melalui mikrokontroller hanya memiliki satu alamat *buffer* yaitu 0x40.

- **Control Bytes**

  Pengiriman teks ke *buffer* harus didahului dengan tiga *byte* pengontrol yaitu *volume*, *pitch*, dan *speed*. *Pitch* berfungsi untuk mengatur frekuansi pengucapan (tinggi rendahnya nada), *speed* berfungsi untuk menentukan

kecepatan pengucapan antar kata, dan *volume* berfungsi untuk mengatur volume suara hasil dari proses konversi. Berikut ini adalah contoh urut-urutan untuk proses *load* kata "Hello" ke dalam *buffer* melalui mikrokontroller.

Tabel 2.2. Contoh *Download Text Ke Buffer*

Start Bit	I2C Start Sequence
0xC4	SP03 I2C Address
0x00	SP03 Command Register
0x00	SP03 NOP Command
0x00	Volume (Max.)
0x05	Speech Speed
0x03	Speech Pitch
'H' (0x48)	Text
'e' (0x65)	Text
T (0x6C)	Text
T (0x6C)	Text
'o' (0x6F)	Text
0x00	NULL
Stop Bit	I2C Stop Sequence

Teks yang telah didownloadkan akan tersimpan dalam *buffer* dengan alamat 0x40. NULL berfungsi untuk menandakan bahwa pengiriman data hexa untuk tiap-tiap karakter sudah diakhiri. Berikut ini adalah cara untuk melakukan konversi teks yang telah didownload ke *buffer*.

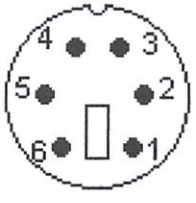
Tabel 2.3. Proses Konversi Teks

Start Bit	I2C Start Sequence
0xC4	SP03 I2C Address
0x00	SP03 Command Register
0x40	Speak Buffer
Stop Bit	I2C Stop Sequence

2.4 *AT keyboard* (PS/2 *connector*)

Dalam penyusunan skripsi ini, *keyboard* digunakan untuk mengetikkan kata yang akan dikonversikan menjadi suara. *Keyboard* yang dipakai adalah AT *keyboard* yang umum digunakan pada komputer. Tiap karakter yang diketikkan pada *keyboard* akan disimpan pada memori mikrokontroller dan kemudian akan dikirimkan ke modul TTS.

Komunikasi yang digunakan antara *keyboard* dengan mikrokontroller adalah komunikasi secara serial sinkron. Ada empat jalur yang menghubungkan *keyboard* dengan mikrokontroller. Dua diantaranya adalah *power supply* dan *ground* yang digunakan untuk memberikan daya ke rangkaian *keyboard.* Sedangkan dua jalur yang lain adalah jalur *data keyboard* dan jalur *clock.* Jalur data merupakan jalur yang berfungsi untuk mengirimkan informasi berupa kode scan ke mikrokontroller. Sedangkan jalur clock berfungsi untuk mengirimkan sinyal clock, dengan suatu nilai yang berosilasi dari logika "1" ke "0" dengan pola yang teratur. Tujuan dari sinyal clock ini adalah untuk mensinkronisasi keyboard dan mikrokontroller, sehingga mereka selalu bekerja secara bersamaan. Gambar dibawah ini menunjukkan pin-pin konektor PS/2 yang digunakan pada *keyboard.*



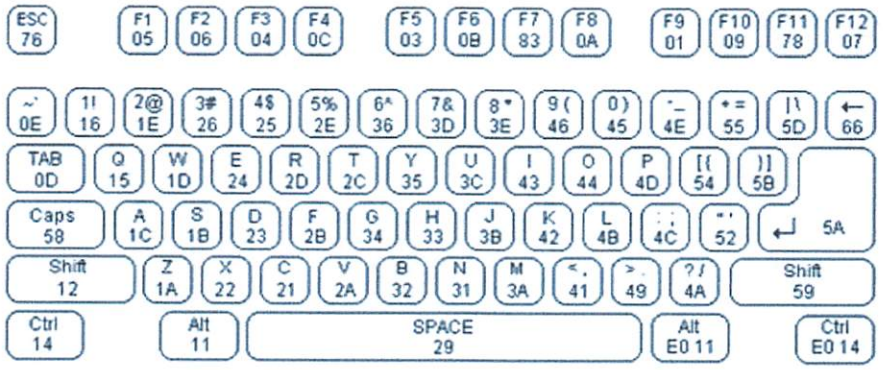
Gambar 2.3. Konektor *keyboard* PS/2
Sumber. www.mytoutorial.com

Keterangan: - (1) *Keyboard Clock*

- (2) Ground
- (3) Data *keyboard*

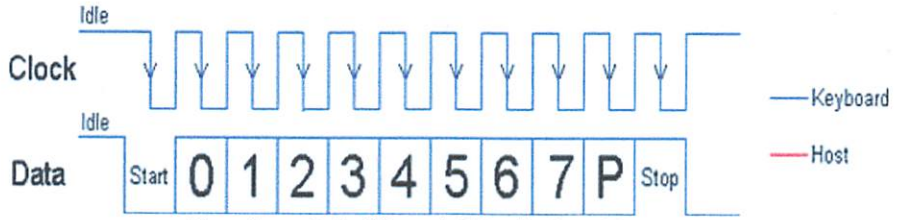
- (4) N/C
- (5) +5V (Vcc)
- (6) N/C

Masing-masing tombol yang terdapat pada *keyboard* memiliki kode scan tertentu dalam bentuk hex sehingga pada saat terjadi penekanan tombol kode scan tersebut akan dikirimkan sebagai informasi data. Gambar dibawah ini akan menunjukkan kode scan pada masing-masing tombol.



Gambar 2.4. Kode Scan Pada keyboard PC
Sumber. www.mytoutorial.com

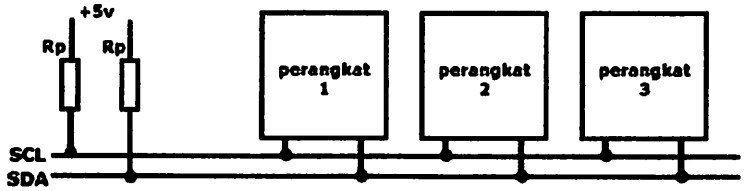
Semua kode scan diatas ditunjukkan dalam hex. Sebagai contoh, kode scan untuk ESC adalah 76 hex. Transmisi data, dari keyboard ke sistem, dilakukan dengan frame 11 bit. Bit pertama adalah bit start (logika 0), diikuti dengan 8 bit data (LSB first), satu bit paritas (paritas ganjil) dan bit stop (logika 1). Setiap bit harus dibaca pada sisi turun dari clock.



Gambar 2.5. Sinyal Clock dan Data
Sumber. www.mytoutorial.com

2.5 *Inter Integreted Circuit* (I2C)

Inter Integrated Circuit atau sering disebut I2C adalah standar komunikasi serial dua arah menggunakan dua saluran yang didisain khusus untuk pengontrolan IC. Sistem I2C terdiri dari saluran SCL (*Serial Clock*) dan SDA (*Serial Data*) yang membawa informasi data antara I2C dengan pengontrolnya. Piranti yang dihubungkan dengan sistem I2C Bus dapat dioperasikan sebagai *Master* dan *Slave*. *Master* adalah piranti yang memulai *transfer* data pada I2C Bus dengan membentuk sinyal *Start*, mengakhiri *transfer* data dengan membentuk sinyal *Stop*, dan membangkitkan sinyal *clock*. Sedangkan *slave* adalah piranti yang dialamati *master*. Dalam aplikasinya, I2C dapat menghubungkan sebuah *master* dengan beberapa *slave*. Agar jalur *data* dan *clock* dapat bekerja dengan baik, maka pada keduanya dibutuhkan pull up resistor yang besarnya antara 1k sampai dengan 47k.



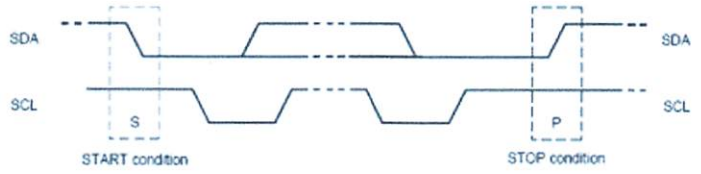
Gambar 2.6. Contoh implementasi jalur I2C

Dengan adanya resistor pull-up, jalur SCL dan SDA menjadi *open drain*, sehingga perangkat hanya perlu memberikan output "0" untuk membuat jalur menjadi "low". Sedangkan untuk membuat jalur menjadi "high" dapat dilakukan dengan memberikan output berlogika "1" atau dengan membiarkan output tersebut dalam keadaan mengambang (bukan "0" dan bukan "1").

Dalam komunikasi menggunakan I2C, hanya perangkat *master* yang dapat mengontrol jalur SCL. Sedangkan tugas perangkat *slave* hanya merespon apa yang diminta *master.* Transfer data harus diinisialisasi terlebih dahulu oleh perangkat *master* melalui serangkaian pulsa clock. *Slave* dapat memberi data ke *master* dan menerima data dari *master* setelah *server* melakukan inisialisasi.

2.5.1 Sinyal *Start* dan *Stop*

Inisialisasi awal yang dilakukan oleh *master* adalah dengan membangkitkan sinyal *Start.* Sinyal *Start* merupakan sinyal untuk memulai semua perintah, didefinisikan sebagai perubahan tegangan SDA dari "1" menjadi "0" pada saat SCL "1". Sedangkan sinyal *Stop* merupakan sinyal untuk mengakhiri semua perintah, didefinisikan sebagai perubahan tegangan SDA dari "0" menjadi "1" pada saat SCL "1". Kondisi sinyal *Start* dan sinyal *Stop* seperti tampak pada gambar berikut.

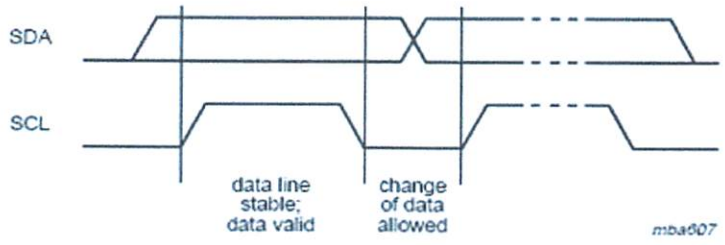


Gambar 2.7. Kondisi Sinyal *Start* dan sinyal *Stop*
Sumber: UM10204 I2C-bus specification and user manual

2.5.2 Pengiriman Data

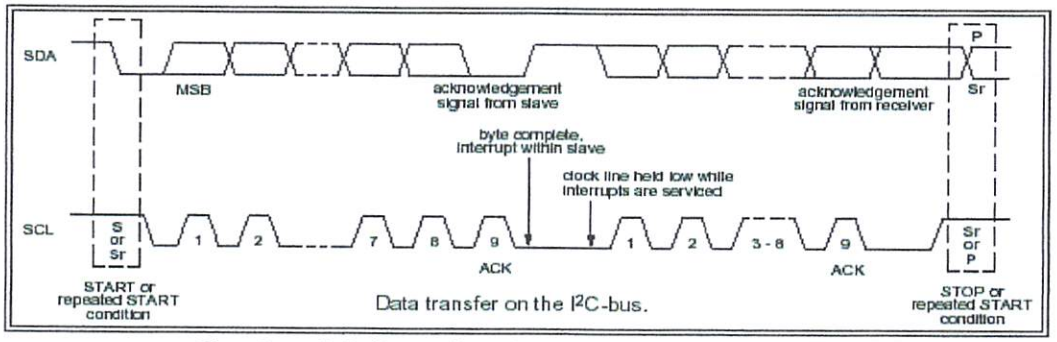
Untuk pengiriman data, banyaknya byte data yang dikirim tidak ada ketentuannya. Jika transfer data yang ingin dilakukan sebesar 2 byte, maka pengiriman pertama adalah 1 byte dan setelah itu 1 byte berikutnya . Setiap byte yang di transfer harus diikuti dengan bit Acknowledge (ACK) dari perangkat *slave* yang menandakan bahwa data berhasil diterima. Byte yang dikirim dari pengirim diawali dari bit MSB. Saat bit dikirim, pulsa clock (SCL) di set ke

HIGH lalu ke LOW. Bit yang dikirim pada jalur SDA tersebut harus stabil saat periode clock (SCL) HIGH. Kondisi HIGH atau LOW dari jalur data (SDA) hanya dapat berubah saat kondisi sinyal SCL itu LOW.



Gambar 2.8. Transfer bit pada jalur I2C
Sumber: UM10204 I2C-bus specification and user manual

Setiap pulsa *clock* pada jalur SCL digunakan untuk mentransfer satu bit data pada jalur SDA. Sehingga untuk pengiriman 8 bit akan ada 9 pulsa *clock* yang harus dihasilkan, satu *clock* terakhir digunakan untuk bit ACK. Proses sebelum perangkat penerima memberikan sinyal ACK adalah sebagai berikut: saat pengirim selesai mengirimkan bit data ke-8, pengirim melepaskan jalur SDA ke *pull-up*. Sesuai dengan penjelasan sebelumnya kondisi ini akan membuat jalur SDA menjadi *open drain* sehingga kondisi menjadi HIGH. Nah saat kondisi tersebut terjadi, penerima memberikan kondisi LOW ke SDA saat pulsa clock kesembilan berada dalam kondisi HIGH.



Gambar 2.9. Data (byte) transfer pada jalur I2C
Sumber: UM10204 I2C-bus specification and user manual

Jika SDA tetap dalam kondisi HIGH saat pulsa clock ke-9, maka ini didefinisikan sebagai sinyal Not Acknowledge (NACK). Master dapat menghasilkan sinyal STOP untuk menyudahi transfer, atau mengulang sinyal START untuk memulai transfer data yang baru. Ada 5 kondisi yang menyebabkan NACK:

1. Tidak adanya penerima dengan alamat yang diminta pada jalur, sehingga tidak ada perangkat yang merespon ACK.
2. Penerima tidak dapat menerima atau mengirim karena sedang mengeksekusi fungsi lain dan tidak siap untuk memulai komunikasi dengan master.
3. Pada saat transfer data, penerima mendapatkan data atau perintah yang tidak dimengerti oleh penerima.
4. Pada saat transfer data, penerima tidak dapat menerima lagi byte data yang dikirimkan.
5. Penerima-master perlu memberi sinyal pengakhiran transfer data ke penerima-slave.

2.5.3 Pengalamatan I2C

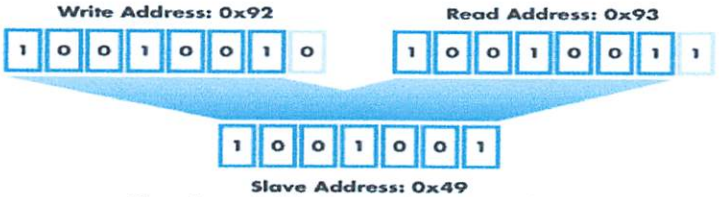
Dalam satu jalur I2C memungkinkan adanya beberapa *slave*. *Master* dapat mengirimkan ataupun menerima data dari semua *slave* yang terhubung dengan *master*. Oleh karena hal tersebut, agar komunikasi antara *master* dengan *slave* tidak rancu maka tiap-tiap perangkat *slave* memiliki alamat yang berbeda satu sama lain sesuai dengan ketentuan yang sudah ditetapkan oleh produsen perangkat *slave* tersebut.

Pengalamatan pada I2C bisa 7 bit atau 10 bit. Pengalamatan 7-bit berarti semua perangkat yang terhubung ke dalam jalur I2C yang sama dapat dialamati

sebanyak 7 bit. Ini berarti sebuah jalur I2C dengan pengalamatan 7 bit dapat menampung 128 perangkat. Saat mengirimkan data alamat (7-bit), kita tetap mengirim data 1 byte (8 bit). 1 bit yang lain yaitu bit LSB digunakan untuk menginformasikan perangkat *slave* apakah *master* menulis (*write*) data ke *slave* atau membaca (*read*) data dari *slave*. Jika bit tersebut 0, maka master menulis data ke *slave*. Jika bit tersebut 1, maka *master* membaca data dari *slave*. Bit ini (untuk infomasi tulis/baca) merupakan LSB, sedangkan sisanya adalah data alamat 7 bit. Sebagai contoh sebuah modul dengan alamat 0x14 (dengan perhitungan 7-bit), maka untuk menulis ke alamat 0×14 kita harus memberikan byte 0×28 dengan menggesernya 1 bit (bit 0 pada LSB berarti menulis).

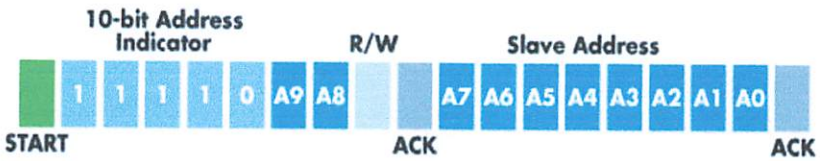
Gambar 2.10. Pengalamatan 7-bit
Sumber. www.totalphase.com

Beberapa *vendor* menggunakan pengalamatan 8-bit pada perangkat yang diproduksi. Ini berarti bahwa dalam pengalamatan tersebut sudah berisi informasi tentang status R/W (baca atau tulis) pada bit LSB. Sebagai contoh, sebuah *device* dengan alamat 0x92. Ini berarti bahwa R/W bit pada alamat tersebut adalah "0" (tulis) sedangkan alamat sebenarnya dari *device* tersebut adalah 0x49. Apabila kita ingin mengoperasikan pada mode (baca) maka alamat yang harus kita kirimkan ke *device* adalah 0x93.



Gambar 2.11. Pengalamatan 8-bit
Sumber. www.totalphase.com

Untuk pengalamatan 10-bit dilakuakan dengan dua kali pengiriman byte. A9,A8 dan R/W pada byte pertama sedangkan A7 sampai A0 pada byte berikutnya.



Gambar 2.12. Pengalamatan 10-bit
Sumber. www.totalphase.com

Hal pertama kali yang dilakukan dalam proses komunikasi adalah *server* akan mengirimkan sinyal *start*. Ini akan menginformasikan perangkat-perangkat *slave* yang terhubung dalam jalur I2C bahwa akan ada transfer data yang ingin dilakukan oleh *master* dan semua *slave* harus siap memantau siapa yang akan dipanggil alamatnya. Selanjutnya *master* akan mengirimkan data berupa alamat *slave* yang ingin diakses. Perangkat *slave* yang sesuai dengan alamat yang diberikan *master* akan meneruskan transaksi data, *slave* lainnya dapat mengacuhkan transaksi tersebut dan menunggu sampai sinyal berikutnya. Setelah mendapatkan *slave* yang sesuai dengan alamat tersebut, kemudian *master* memberitahukan register internal yang ingin ditulis atau dibaca dari *slave* tersebut. Kemudian master dapat melanjutkan mengirim byte data ke *slave* dan byte-byte data akan ditampung di register setelahnya karena slave secara otomatis akan menaikkan alamat internal register setelah setiap pengiriman satu byte data. Ketika *master* selesai menulis semua data ke *slave*, master akan mengirim sinyal *stop* untuk mengakhiri transaksi data. Jadi untuk menulis ke *slave* langkahnya adalah sebagai berikut:

1. Mengirim sinyal START
2. Mengirim alamat *slave* serta operasi yang akan dilakukan (Write)
3. Mengirim nomor internal register yang ditulis

4. Mengirim byte data ke register yang telah ditentukan
5. Mengirim sinyal STOP

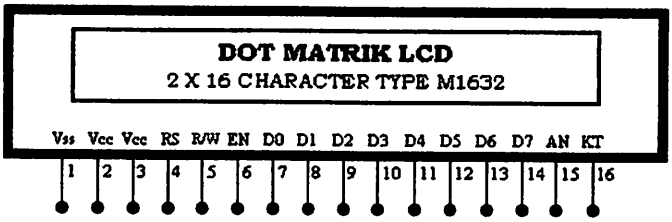
Sedangkan untuk membaca data dari *slave* langkah-langkahnya adalah seperti dibawah ini.

1. Mengirim sinyal START
2. Mengirim alamat *slave* serta operasi yang akan dilakukan (Write)
3. Mengirim nomor internal register yang ditulis
4. Mengirim sinyal START lagi
5. Mengirim alamat *slave* serta operasi yang akan dilakukan (read)
6. Membaca byte data dari register yang telah ditentukan
7. Mengirim sinyal STOP

2.6 LCD (*Liquid Crystal Display*)

Liquid Crystal Display adalah modul tampilan berkonsumsi daya yang relatif rendah dan terdapat sebuah kontroller CMOS didalamnya. Kontroler tersebut sebagai pembangkit karakter dari ROM/RAM dan *display* data RAM. Semua fungsi tampilan dikontrol oleh suatu intruksi dan modul LCD dapat dengan mudah untuk diinterfacekan dengan mikroprossor/mikrokontroller. Input yang diperlukan untuk mengendalikan modul ini berupa bus data yang termultipleks dengan bus alamat dan 3 bit sinyal kontrol. Pengendali *dot matrik* LCD dilakukan secara internal pada modul LCD sendiri.

LCD merupakan suatu bentuk kristal cair yang akan beremulsi apabila dikenakan tegangan padanya. Tampilannya ini berupa *dot matrik* 5 x LCD sehingga jenis huruf yang dapat ditampilkan akan lebih banyak dan lebih baik resolusinya jika dibandingkan dengan 7 segment.



Gambar 2.13. Deskripsi pin pada LCD Tipe M1632
Sumber : LCD Manual Book

LCD tipe M1632 memiliki ciri-ciri sebagai berikut :

- LCD ini terdiri dari 32 karakter dengan 2 baris masing-masing 16 karakter dengan *display dot matrik* 5 x 7
- Karakter generator ROM dengan 192 tipe karakter
- Karakter generator RAM dengan 8 tipe karakter
- 80 x 8 display data RAM
- Dapat diinterfacekan ke MPU 8 atau 4
- Dilengkapi fungsi tambahan : *display clear, cursor hpme, display ON/OFF, cursor ON/OFF, display character blink, cursor shift, dan display shift.*
- Internal Data
- Internal Otomatis, reset pada saat power ON
- +5 volt PS Tunggal

Tabel 2.5. Fungsi Pin – Pin LCD

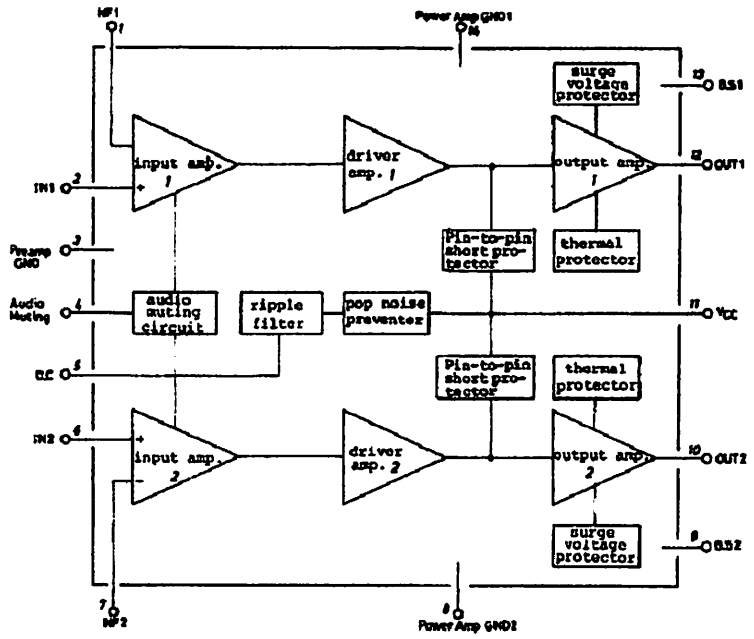
Nama Pin	Jumlah	I/O	Tujuan	Fungsi
DB0-DB3	4	I/O	MPU	Tri state bidirectional lower data bus: data dibaca dari modul ke MPU atau dari MPU ditulis ke modul melalui bus

DB4-DB7	4	I/O	MPU	Tri state bidirectional upper fourdata bus: data dibaca dari modul ke MPU atau dari MPU ditulis ke modul melalui bus
E	1	Input	MPU	Sinyal operasi dimulai: sinyal aktif baca/tulis
R/W	1	Input	MPU	Sinyalpilih data dan tulis (0:tulis,1:baca)
RS	1	-	Power supply	Sinyal pilih register : 0 : Instruction register (write) Busy flag dan addess counter (read) 1 : Data register (write dan read)
Con.	1	-	Power supply	Penyetelan kontras pada tampilan LCD
Vdd	1	-	Power supply	+ 5V
Vss	1	-	Power supply	Ground 0V

Sumber : LCD Module User Manual

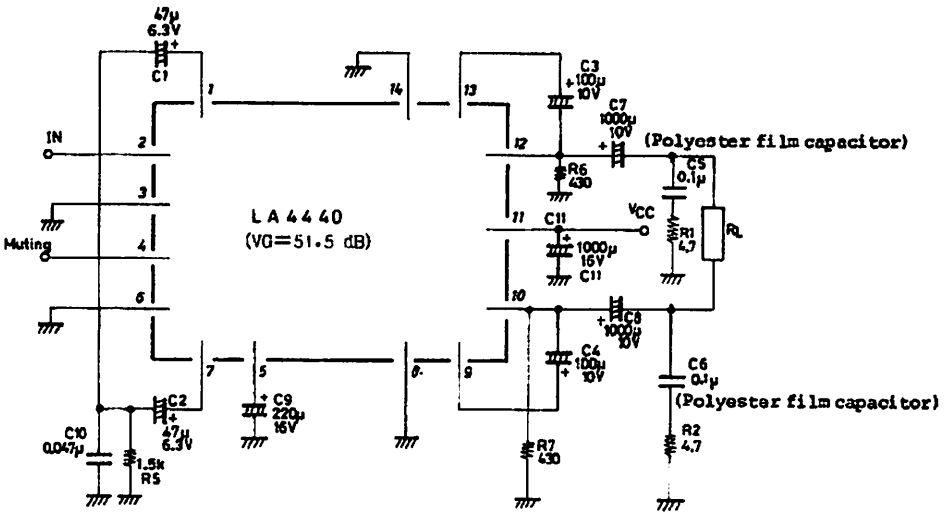
2.7 IC LA4440

LA4440 merupakan sebuah IC yang berfungsi sebagai *power amplifier*. IC ini dapat difungsikan baik secara stereo ataupun mono. Jika dioperasikan secara stereo IC ini dapat menghasilkan daya sebesar 6watt pada masing-masing sisi (kiri dan kanan). Jika dioperasikan secara mono, maka LA4440 dapat menghasilkan daya sebesar 19watt. Tegangan kerja dari IC ini adalah 13,2 volt dengan resistansi beban speaker antara 4 ohm sampai dengan 8 ohm. Berikut ini adalah blok diagram dari LA4440. Berikut ini adalah blok diagram dari IC LA4440.

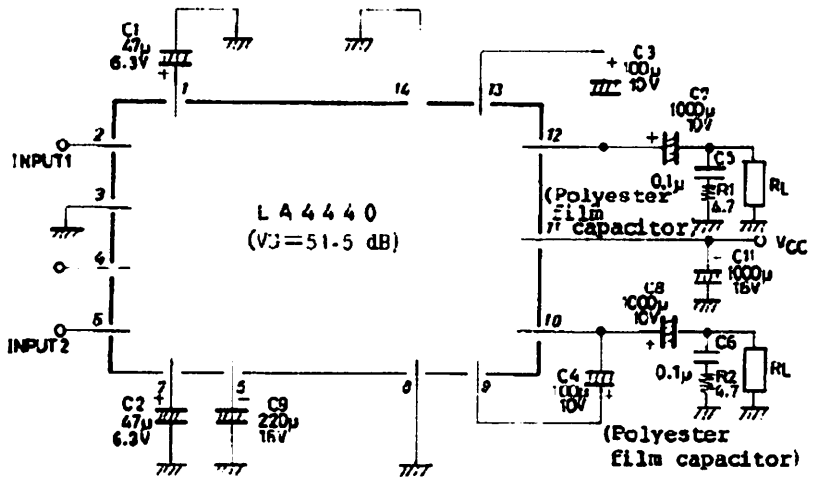


Gambar 2.14 Blok Diagram IC LA4440
Sumber Data Sheet LA4440

IC LA4440 memiliki dua buah input pada pin dua dan tiga serta memiliki dua buah output pada pin 10 dan 12. LA4440 juga memiliki fasilitas *audio muting* pada pin 4. Berikut ini adalah gambar aplikasi IC untuk aplikasi secara mono dan stereo



Gambar 2.15.Aplikasi Mono Pada IC LA4440
Sumber Data Sheet LA4440



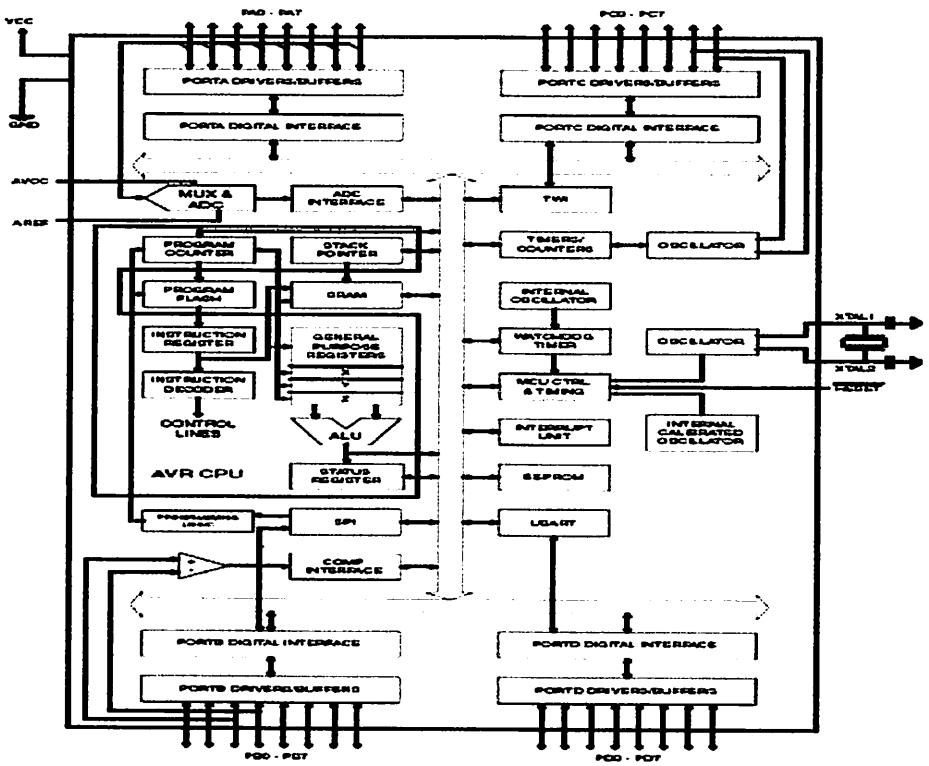
Gambar2.16. Aplikasi Mono Pada IC LA4440
Sumber Data Sheet LA4440

2.8 Mikrokontroler ATMEGA 16

Mikrokontroler AVR memiliki arsitektur RISC 8 bit, dimana semua instruksi dikemas dalam kode 16-bit (16-bit word) dan sebagian besar instruksi dieksekusi dalam 1 (satu) siklus clock. Secara umum, AVR dapat dikelompokkan menjadi 4 kelas, yaitu ATtiny, keluarga AT90Sxx, keluarga ATMega, dan AT86RFxx. Pada dasarnya yang membedakan masing-masing kelas adalah memori, peripheral, dan

fungsinya. Dari segi arsitektur dan instruksi yang digunakan, mereka bisa dikatakan hampir sama.

2.8.1 Arsitektur ATMega16



Gambar 2.17. Blok diagram fungsional ATMega16
Sumber : www.atmel.com, datasheet ATMega16

Pada gambar diatas dapat dilihat bahwa mikrokontroler ATMega16 memiliki bagian-bagian sebagai berikut :

1. Saluran I/O sebanyak 32 buah, yaitu Port A, Port B, Port C, dan Port D.
2. ADC 10 bit sebanyak 8 saluran.
3. Tiga buah *Timer/Counter* dengan kemampuan pembandingan.
4. CPU yang terdiri atas 32 buah register.
5. *Watchdog Timer* dengan osilator internal.
6. SRAM sebesar 1 Kbyte .
7. Memori Flash sebesar 16 kb dengan kemampuan *Read While Write.*

8. Unit interupsi internal dan eksternal.
9. Port antarmuka SPI.
10. EEPROM sebesar 512 byte yang dapat diprogram saat operasi.
11. Antarmuka komparator analog.
12. Port USART untuk komunikasi serial.

2.8.2 Fitur ATMega16

Kapabilitas detail dari ATMega16 adalah sebagai berikut :

1. Sistem mikroprosesor 8 bit berbasis *RISC* dengan kecepatan maksimal 16 MHz.
2. Kapabilitas memori *flash* 16 KB, *SRAM* sebesar 1024 byte, dan *EEPROM* (*Electrically Erasable Programmable Read Only Memory*) sebesar 512 byte.
3. *ADC* internal dengan fidelitas 10 bit sebanyak 8 *channel.*
4. Portal komunikasi serial (*USART*) dengan kecepatan maksimal 2,5 Mbps.
5. Enam pilihan mode *sleep* menghemat penggunaan daya listrik.

2.8.3 Konfigurasi Pin ATMega16

Konfigurasi pin ATMega16 bisa dilihat pada gambar 1.2. Dari gambar tersebut dapat dijelaskan secara fungsional konfigurasi pin ATMega16 sebagai berikut :

A. VCC merupakan pin yang berfungsi sebagai pin masukan catu daya.
B. GND merupakan pin *ground.*
C. Port A (PA0..PA7) merupakan pin I/O dua arah dan pin masukan ADC.
D. Port B (PB0..PB7) merupakan pin I/O dua arah dan pin fungsi khusus, yaitu Timer/Counter, komparator analog, dan SPI.

E. Port C (PC0..PC7) merupakan pin I/O dua arah dan pin fungsi khusus, yaitu TWI, komparator analog, dan *Timer Oscilator*.

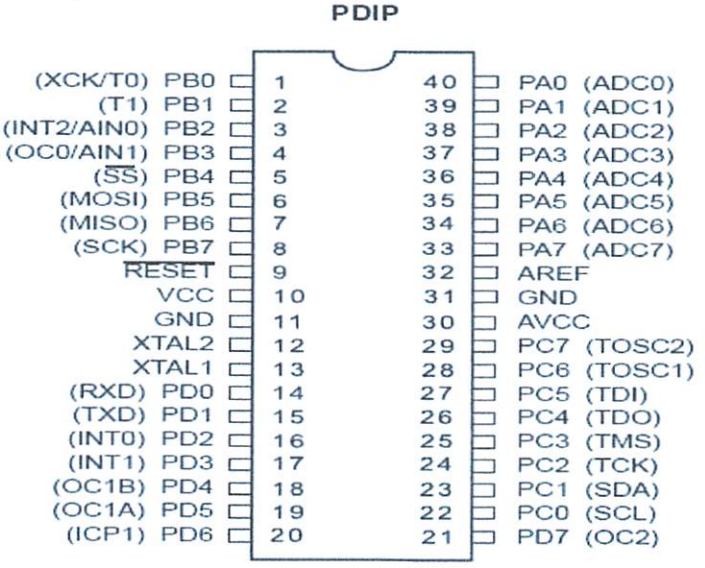
F. Port D (PD0..PD7) merupakan pin I/O dua arah dan pin fungsi khusus, yaitu komparator analog, interupsi eksternal, dan komunikasi serial.

G. RESET merupakan pin yang digunakan utuk me-reset mikrokontroler.

H. XTAL1 dan XTAL2 merupakan pin masukan *clock* eksternal.

I. AVCC merupakan pin masukan tegangan untuk ADC.

J. AREF merupakan pin masukan tegangan referensi ADC.



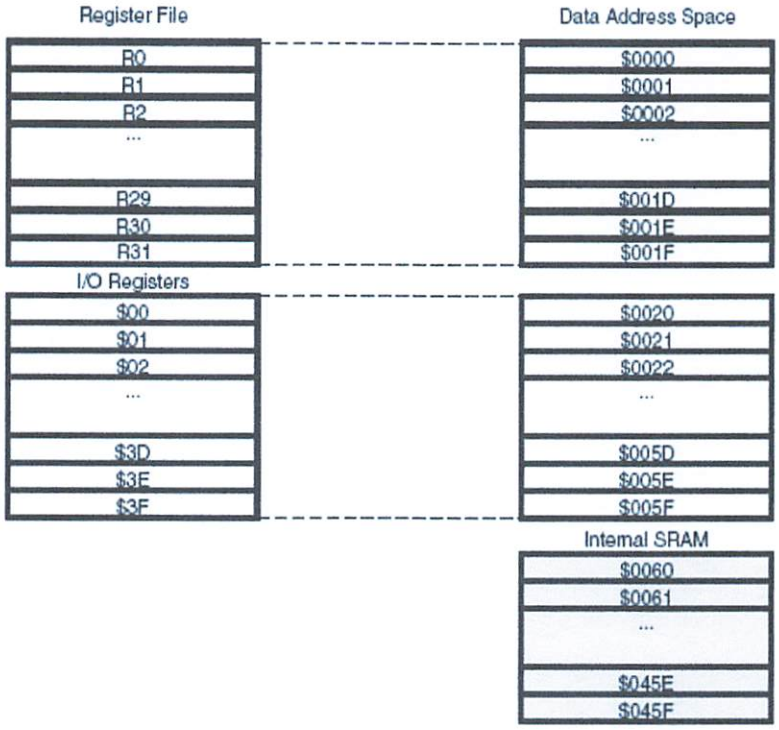
Gambar 2.18. Pin ATMega16
Sumber : www.atmel.com, datasheet ATMega1

2.8.4 Peta Memori

AVR ATMega16 memiliki ruang pengamatan memori data dan memori program yang terpisah. Memori data terbagi menjadi 3 bagian, yaitu 32 buah register umum, 64 buah register I/O, dan 512 *byte* SRAM *Internal*.

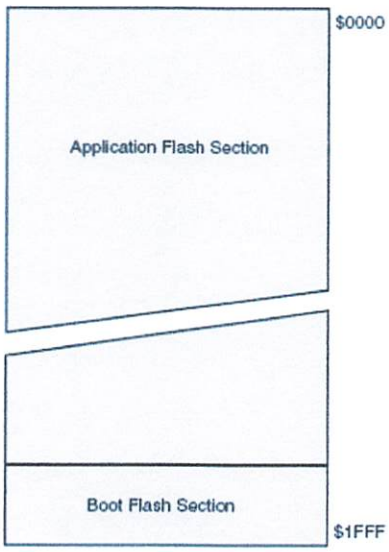
Register keperluan umum menempati space data pada alamat terbawah, yaitu $00 sampai $1F. Sementara itu, register khusus untuk menangani I/O dan control terhadap mikrokontroler menempati 64 alamat berikutnya, yaitu mulai dari

$20 hingga $5F. Register tersebut merupakan register yang khusus digunakan untuk mengatur fungsi terhadap berbagai *peripheral* mikrokontroler, seperti control register, *timer/counter*, fungsi-fungsi I/O, dan sebagainya. Alamat memori berikutnya digunakan untuk SRAM 512 byte, yaitu pada lokasi $60 sampai dengan $25F. Konfigurasi memori data ditunjukkan pada gambar dibawah ini.



Gambar 2.19. Konfigurasi Memori Data AVR ATMega16
Sumber : www.atmel.com, *datasheet* ATMega16

Memori program yang terletak dalam *Flash PEROM* tersusun dalam word atau 2 byte karena setiap instruksi memiliki lebar 16-bit atau 32-bit. AVR ATMega16 memiliki 16K*Byte*X16-bit *Flash PEROM* dengan alamat mulai dari $000 sampai $FFF. AVR tersebut memiliki 12-bit *Program Counter* (PC) sehingga mampu mengalamati isi *Flash*.



Gambar 2.20. Peta Memory Program AVR ATMega16
Sumber : www.atmel.com, *datasheet* ATMega16

Selain itu, AVR ATMega16 juga memiliki memori data berupa EEPROM 8-bit sebanyak 512 *byte*. Alamat EEPROM dimulai dari $000 sampai $1FF.

2.8.5 Status Register (SREG)

Status register adalah register berisi status yang dihasilkan pada setiap operasi yang dilakukan ketika suatu instruksi dieksekusi. SREG merupakan bagian dari inti CPU mikrokontroler.

a. Bit 7 – I : *Global Interrupt Enable*

Bit harus diset untuk meng-enable interupsi. Setelah itu, anda dapat mengaktifkan interupsi mana yang akan anda gunakan dengan cara meng-enable bit control register yang bersangkutan secara individu. Bit akan di-clear apabila terjadi suatu interupsi yang dipicu oleh hardware, dan bit tidak akan mengizinkan terjadinya interupsi, serta akan diset kembali oleh instruksi RETI.

b. Bit 6 – T : *Bit Copy Storage*

Instruksi BLD dan BST menggunakan bit-T sebagai sumber atau tujuan dalam operasi bit. Suatu bit dalam sebuah register GPR dapat disalin ke bit menggunakan instruksi BST, dan sebaliknya bit-T dapat disalin kembali ke suatu bit dalam register GPR menggunakan instruksi BLD.

c. Bit 5 – H : *Half Carry Flag*

d. Bit 4 – S : *Sign Bit*

Bit-S merupakan hasil operasi EOR antara flag-N (negatif) dan flag V (komplemen dua overflow).

e. Bit 3 – V : *Two's Complement Overflow Flag*

Bit berguna untuk mendukung operasi aritmatika.

f. Bit 2 – N : *Negative Flag*

Apabila suatu operasi menghasilkan bilangan negatif, maka *flag-N* akan diset.

g. Bit 1 – Z : *Zero Flag*

Bit akan diset bila hasil operasi yang diperoleh adalah nol.

h. Bit 0 – C : *Carry Flag*

Apabila suatu operasi menghasilkan *carry*, maka bit akan diset.

BAB III

PERANCANGAN DAN PEMBUATAN ALAT

3.1 Pendahuluan

Pada bab ini akan dibahas perancangan dan pembuatan alat. Pembahasan akan dilakukan pada setiap blok rangkaian, cara kerja masing-masing blok rangkaian serta fungsi masing-masing blok rangkaian tersebut. Secara garis besar terdapat dua bagian perangkat yang ada yaitu :

- Perancangan perangkat keras *(Hardware).*
- Perancangan perangkat lunak *(Software).*

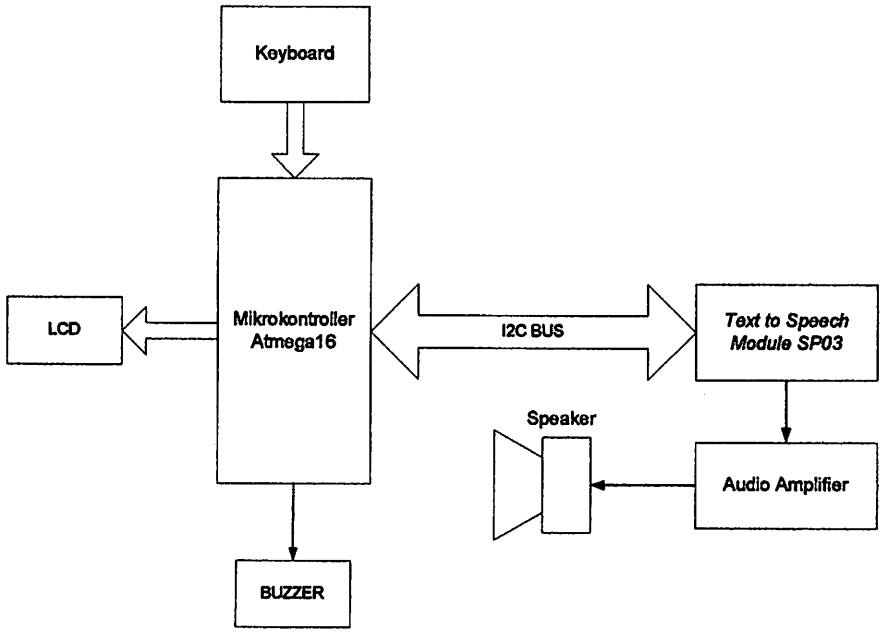
Perancangan perangkat keras meliputi seluruh *peripheral* yang digunakan pada sistem ini. Sedangkan perancangan perangkat lunak meliputi diagram alir dan *software* secara umum. Akan tetapi kedua perangkat ini akan saling menunjang satu sama lain.

Pada perangkat keras sendiri terdiri dari beberapa rangkaian antara lain rangkain LCD, rangkaian konektor *keyboard* dengan mikrokontroller, rangkaian buzzer, dan rangkian audio amplifier.

3.2 Perancangan Perangkat Keras

Perancangan perangkat keras yang direncanakan meliputi, blok diagram keseluruhan dan prinsip kerja alat, pembuatan skema seluruh rangkaian yang direncanakan, dan perakitan seluruh komponen.

3.2.1 Blok Diagram Alat



Gambar 3.1. Blok Diagram Sistem

Fungsi masing – masing diagram blok :

- ***Keyboard***

  Berfungsi untuk memasukkan kata yang akan dokonversikan menjadi suara. Selain itu keyboard juga digunakan untuk pemilihan menu pada tampilan LCD

- ***LCD***

  Berfungsi untuk menampilkan karakter pada saat pengetikkan sehingga pengguna dapat meyakinkan apakah penulisan kata yang akan dikonversikan benar atau tidak. Selain itu LCD juga digunakan untuk menampilkan menu.

- ***Text To Speech Module SP03***

  Merupakan modul yang berfungsi untuk mengkonversikan tulisan kedalam bentuk suara.

- ***Buzzer***

  Buzzer digunakan apabila saat pengetikan pada *keyboard* pengguna menekan tombol-tombol yang tidak termanfaatkan dalam sistem

- **I2C *Bus***

  Berfungsi sebagai *interface* antara mikrokontroller dengan modul TTS SP03

- ***Audio Amplifier***

  Rangkain audio amplifier digunakan sebagai penguat sinyal audio yang dihasilkan oleh modul TTS SP03.

- ***Speaker***

  Output suara hasil konversi teks

- **Mikrokontroller ATmega 16**

  Berfungsi sebagai pemroses data yang masuk dari *keyboard*, menampilkkannya ke LCD dan mengirimkan data tersebut ke modul TTS. Selain itu mikrokontroller juga berfungsi untuk mengirimkan *command* ke modul TTS serta mengendalikan buzzer

3.2.2 Prinsip Kerja Alat

Proses awal yang dilakukan oleh sistem adalah menampilkan empat macam menu pada LCD yaitu *volume*, *pitch*, *speed* serta *text*. Volume berfungsi untuk mengatur kekerasan suara yang dihasilkan, *pitch* berfungsi untuk mengatur frekuensi pengucapan, *speed* berfungsi untuk mengatur kecepatan pengucapan. Menu-menu diatas dapat dipilih dengan menekan tombol pada *keyboard* sesuai dengan nomor urut yang ditampilkan pada LCD. Untuk menu *volume* dan *pitch* memiliki level mulai dari "0" sampai dengan "7". Semakin tinggi level volume

yang dipilih maka semakin tinggi pula volume yang dihasilkan. Demikian juga semakin tinggi level *pitch* maka pengucapan kata yang dihasilkan semakin cepat. Sedangkan untuk *speed* memiliki level mulai dari "0" sampai dengan "3". Semakin tinggi levelnya maka jeda pengucapan kata akan semakin lama. Tombol *keyboard* yang digunakan untuk menaikkan atau menurunkan level *speed*, *pitch*, dan *volume* adalah tombol "+" dan " - ". Ketika pengaturan level selesai, *user* dapat mengakirinya dengan menekan tombol *enter*. Dengan demikian mikrokontroller akan menyimpan nilai level tersebut pada memori yang dimilikinya. Setelah itu mikrokontroller akan memerintahkan LCD untuk menampilkan kembali menu utama.

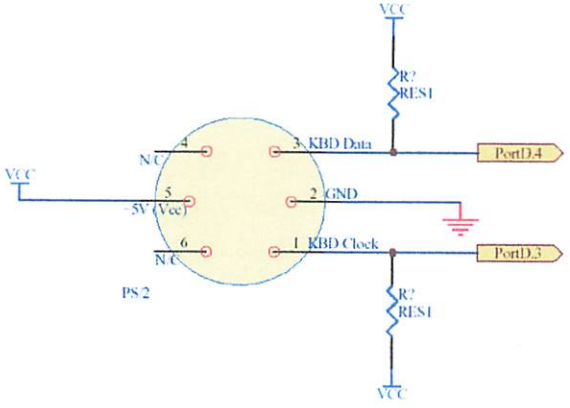
Menu *Text* berfungsi untuk mengetikkan kata yang akan dikonversikan. Ketika menu ini dipilih, mikrokontroller akan menunggu adanya penekanan tombol alfabet dan numerik pada *keyboard*. Satu kali penekanan tombol pada *keyboard* menunjukkan adanya satu karakter yang diketikkan. Tiap-tiap karakter yang telah diketikkan akan disimpan terlebih dahulu dalam memori mikrokonroller sampai terusun satu kata yang utuh sesuai dengan yang diinginkan oleh *user*. Jika proses pengetikkan kata sudah selesai, *user* dapat menekan tombol *enter*.

Saat tombol *enter* ditekan, mikrokontroller akan mengirimkan data serta tiga buah *byte* pengontrol yang dibutuhkan untuk proses konversi kepada modul SP03. Data yang dikirimkan merupakan rangkaian karakter yang telah diketikkan pada menu *Text*, sedangkan tiga buah *byte* pengontrol merupakan hasil pengaturan level *volume*, *pitch*, dan *speed*. Proses pengiriman data serta instruksi dari mikrokontroller ke modul TTS dilakuakan secara serial dengan menggunakan I2C

bus. Setelah modul SP03 menerima data beserta tiga buah *byte* pengontrol, modul tersebut akan memprosesnya untuk dikonversikan menjadi sinyal suara. Karena sinyal suara yang dihasilkan oleh mikrokontroller masih sangat lemah, maka dibutuhkan rangkaian *audio amplifier* agar suara yang dihasilkan lebih keras. Setelah melalui *audio amplifier*, sinyal *audio* akan diubah menjadi suara oleh *speaker*.

Dalam perancangan ini tidak semua tombol yang terdapat pada *keyboard* difungsikan. Tombol-tombol dengan fungsi khusus seperti F1, tab, *insert*, pg Up dan beberapa diantaranya tidak difungsikan. Apabila tombol-tombol tersebut ditekan, maka mikrokontroller akan menghidupkan buzzer.

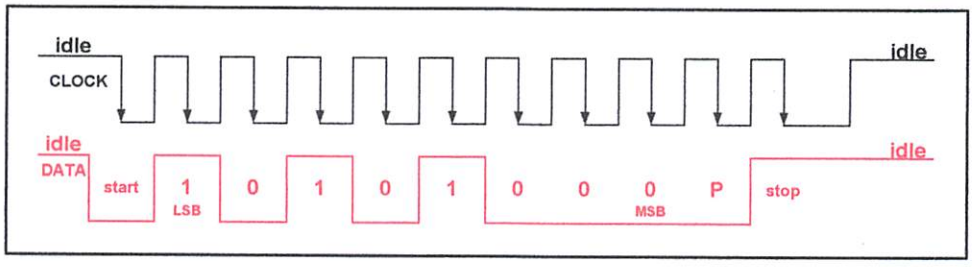
3.2.3 Rangkaian Antarmuka Mikrokontroller Dengan *AT Keyboard*



Gambar 3.2. Rangkaian Konektor PS/2 *Keyboard*

Rangkaian antarmuka antara *keyboard* dengan mikrokontroller membutuhkan konektor PS/2. Pada konektor ini terdapat enam pin, namun hanya empat pin yang terkoneksi. Pin pertama terhubung dengan PD2 (INT0), pin kedua terhubung dengan ground, pin ketiga terhubung dengan PD4, dan pin kelima terhubung dengan Vcc sebagai sumber tegangan bagi *keyboard* untuk membangkitkan sinyal *clock* dan sinyal data. Untuk pin ke empat dan keenam tidak terkoneksi (N/C)

Jalur *keyboard clock* harus terhubung dengan pin *interrupt* (PD3) mikrokontroler, sehingga setiap kali terjadi perubahan kondisi pada sinyal *clock* (dari "1" ke "0") maka mikrokontroller akan menjalankan sub program interupsi. Isi dari sub-program interupsi ini adalah membaca kondisi pada jalur *keyboard data* apakah dalam kondisi "high" atau "low". Dengan demikian setiap satu siklus sinyal *clock* kita dapat memperoleh informasi satu bit data. Proses pengiriman data didahului dengan *start* bit kemudian diikuti 8 bit data, satu bit paritas dan diakhiri dengan satu *stop* bit. Start bit diberikan dengan cara memberikan logika "0" pada saat proses transisi *keyboard clock* dari "high" ke "low", sedangkan *stop bit* merupakn kebalikan dari *start bit*. Bit paritas berfungsi untuk melakukan pengecekan apakah dalam proses transmisi data terjadi kesalahan atau tidak. Berikut ini adalah contoh proses pengiriman data karakter "Q".



Gambar3.3. Contoh Pengiriman Data Dari Keyboard ke Mikrokontroller

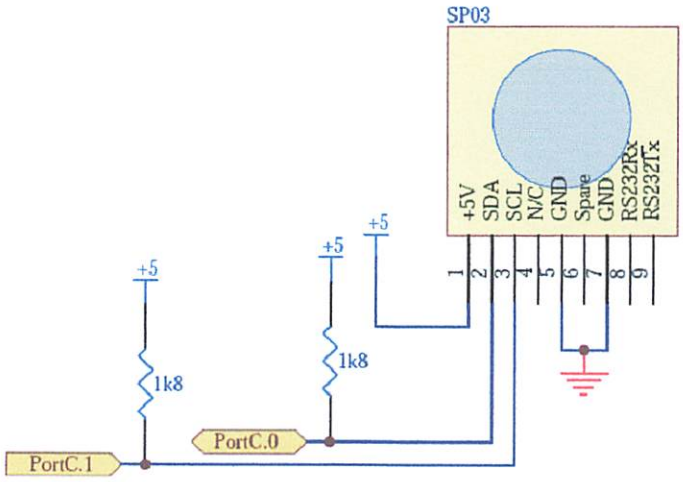
Gambar diatas merupakan contoh pengiriman data karakter "Q" dari *keyboard* ke mikrokontroller. Pada awalnya jalur *clock* dan jalur *data* dalam kondisi "high", pada gambar diatas dinyatakaan dengan "idle". Kondisi ini akan berlangsung terus sampai dengan adanya sinyal start. Sinyal *start* diberikan dengan membuat logika "0" pada jalur data. Setelah sinyal *start* diberikan, maka *keyboard* akan membangkitkan sinyal *clock*. Seperti yang telah dijelaskan sebelumnya, bahwa jalur *keyboard clock* terhubung dengan pin *interrupt* pada

mikrokontroller. Dengan demikian, ketika terjadi perubahan sinyal clock dari "1" ke "0" mikrokontroller akan menjalankan program interupsi yaitu membaca data pada jalur data, pada gambar diatas dinyatakan dengan tanda panah. Setiap satu sinyal clock akan berisi 1 bit data, sehingga untuk mendapatkan 1 byte data dibutuhkan delapan siklus sinyal clock. Pengiriman data dimulai dari bit terendah (LSB) menuju ke bit teringgi (MSB).

Data yang dikirim pada contoh diatas adalah 00010101 (15hex). Jika kita melihat pada gambar 2.6, *scan code* untuk karakter "Q" adalah 15hex. Data inilah yang dikirimkan oleh *keyboard* pada saat tombol "Q" ditekan. Setelah 8 bit data dikirim, selanjutnya *keyboard* akan mengirimkan bit paritas. Bit paritas ini berfungsi untuk melakukan koreksi apakah dalam proses pengiriman data terjadi kesalahan atau tidak. Bit terakhir yang dikirimkan oleh *keyboard* adalah stop bit. Setelah kode scan *keyboard* diterima, mikrokontroller akan mengkonversikannya menjadi kode ASCII. Kode ASCII untuk karakter "Q" adalah 81dec atau 51hex.

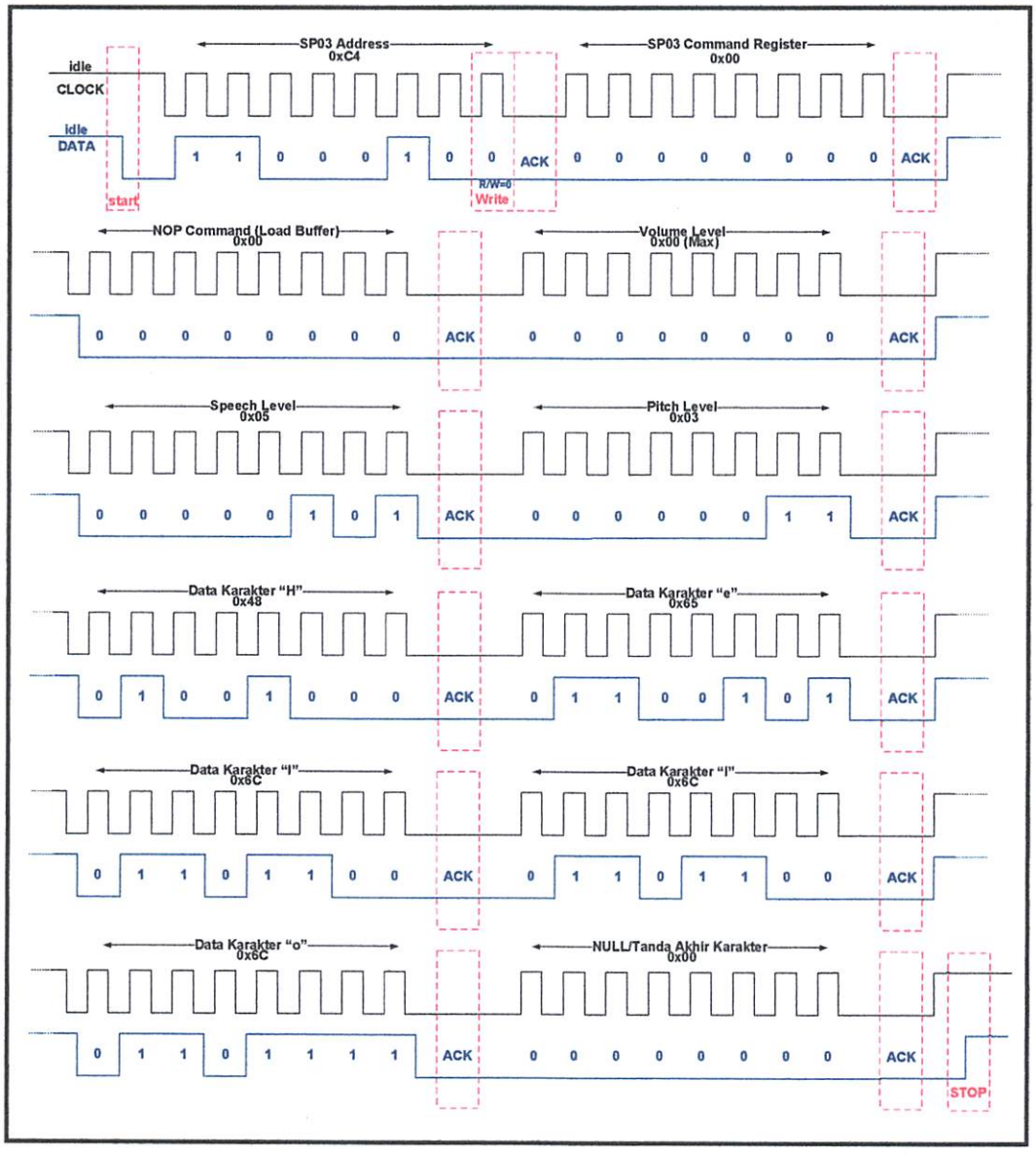
3.2.4 Antarmuka Mikrokontroler Dengan Modul SP03

Komunikasi yang digunakan antara mikrokontroller dengan modul SP03 adalah secara serial menggunakan I2C bus. Pin yang digunakan pada modul tersebut adalah pin SDA dan SCL. Pin SDA berfungsi untuk mengirimkan data dari mikrokontroller ke modul SP03, sedangkan pin SCL berfungsi untuk pengiriman sinyal *clock* sebagai sinkronisasi antara modul dengan mikrokontrloller. Berikut ini adalah gambar rangkaian yang menghubungkan antara modul dengan mikrokontroller.



Gambar 3.4. Rangkaian Modul SP03

Dalam proses pengiriman data dari mikrokontroller ke modul SP03, mikrokontroller bertindak sebagai *master* dan modul SP03 bertindak sebagai *slave*. Mikrokontroller akan membangkitkan sinyal *clock* melalui PortC.1 dan mengirimkan data melalui PortC.0. Proses pengiriman data didahului dengan sinyal START setelah itu diikuti dengan pengiriman byte data. Setelah satu byte data diterima oleh modul, maka modul tersebut akan mengirimkan sinyal ACK *(acknowlage)* sebagai tanda bahwa satu byte data telah diterima. Untuk pengiriman byte data berikutnya, mikrokontroller dapat melakukannya setelah sinyal ACK diterima. Byte data pertama merupakan alamat yang dimiliki oleh modul, alamat tersebut adalah C4hex. Untuk melakukan komunikasi dengan modul, alamat ini harus diakses terlebih dahulu karena jika tidak maka semua data yang dikirimkan melalui jalur data akan diabaikan. Untuk mengakhiri pengiriman data mikrokontroller akan mengirimkan sinyal STOP. Berikut ini adalah contoh pengiriman data dari mikrokontroller ke modul SP03.



Gambar 3.5. Contoh Komunikasi Antara Mikrokontroller Dengan Modul Sp03

Gambar diatas merupakan contoh pengiriman teks "Hallo" yang dilakukan oleh mikrokontroller ke modul SP03. Pada gambar diatas tampak bahwa sebelum sinyal start, pada awalnya jalur data dan jalur *clock* dalam keadaan high. Ini merupakan keadaan dimana pada saat itu tidak terjadi pengiriman data antara mikrokontroller dengan modul SP03. Sinyal *start* merupakan keadaan dimana jalur clock dalam kondisi *high* sedangkan pada jalur data terjadi transisi dari *high* ke *low* . Setelah sinyal *start* dikirimkan, selanjutnya mikrokontroller akan

melakukan proses pengiriman data. Pembacaan tiap bit data dilakukan pada saat jalur clock dalam kondisi *high*. Sedangkan perubahan kondisi pada jalur data (dari "1" ke "0" atau sebaliknya) dilakukan pada saat jalur clock dalam kondisi "0".

Gambar diatas menunjukkan bahwa ada dua belas data byte yang dikirimkan. Berikut ini adalah penjelasan untuk masing byte data yang dikirimkan.

- Data pertama (C4hex)

  C4hex merupakan alamat (*adrress*) yang dimiliki oleh modul SP03. Modul ini menggunakan pengalamatan delapan bit dengan R/W status pada bit terakhir. Jika kita konversikan menjadi bilangan biner maka C4hex menjadi 11000100bin. Bit terakhir dari *address* tersebut adalah "0", ini berarti bahwa *address* tersebut digunakan untuk perintah *write* (tulis).

- Data kedua (00hex)

  Merupakan register *command* yang dimiliki modul.

- Data ketiga (00hex)

  Merupakan *load buffer command* (LDBUF *command*) yang berfungsi untuk menginstruksikan penyimpanan karakter pada *buffer*,

- Data keempat (00hex)

  Berfungsi untuk pengaturan level volume. Volume terdiri dari 8 level mulai dari 00hex sampai dengan 07hex. Volume tertinggi pada saat 00hex, sedangkan volume terendah 07hex.

- Data kelima (05hex)

  Berfungsi untuk pengaturan *speed* (kecepatan pengucapan). Pengaturan kecepatan dimulai dari 00hex sampai dengan 07 hex.

- Data keenam (03hex)

  Berfungsi untuk pengaturan pitch, dimulai dari 00hex sampai dengan 03hex.

- Data ketujuh sampai dengan sebelas (48hex, 65hex, 6Chex, 6Chex, 6Fhex)

  Merupakan kode ASCII untuk masing-masing karakter yang akan didownloadkan ke *buffer*. Pada contoh diatas karakter yang akan dikonversikan adalah H,a,l,l,o.

- Data kesebelas (00hex)

  Data kesebelas ini merupakan kode untuk mengakhiri pengiriman karakter.

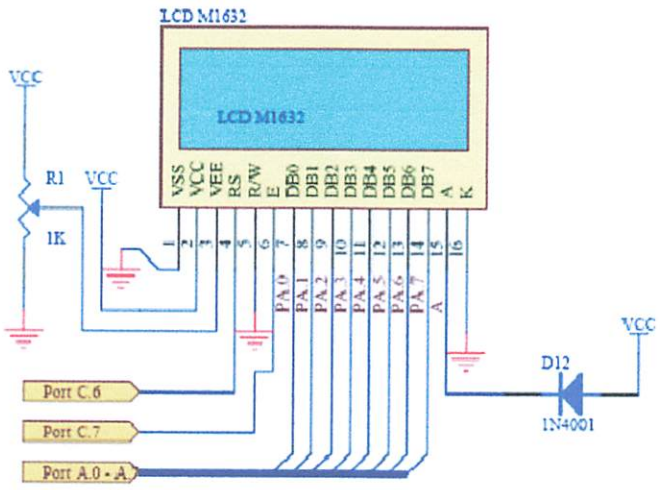
Pada tiap pengiriman satu byte data, mikrokontroller akan menerima sinyal ACK (acknowlage) dari modul sebagai tanda bahwa modul tersebut sudah menerima data yang dikirim. Bit ACK ini dibangkitkan oleh modul SP03 dengan cara memberikan logika "0" setelah menerima byte data. Setelah menerima bit ACK, maka mikrokontroller akan megirimkan data berikutnya. Untuk mengakhiri pengiriman data, mikrokontroller akan memberikan sinyal stop. Sinyal ini berkebalikan dengan sinyal *start,* yaitu pada saat jalur *clock* high pada jalur data terjadi transisi dari *low* ke *high.*

3.2.5 Perancangan Rangkaian LCD

LCD digunakan sebagai tampilan agar *user* dapat memastikan bahwa setiap karakter yang diketikkan sudah benar. Selain itu LCD juga digunakan untuk menampilkan menu-menu yang dimiliki alat.

Pada perancangan ini digunakan LCD Dot Matrik 2 x 16 karakter yaitu M1632. Sinyal-sinyal kontrol yang diperlukan oleh LCD adalah RS dan Enable, sinyal RS dan Enable dipergunakan sebagai pengontrol kerja dari LCD. LCD akan

aktif apabila mikrokontroler memberikan instruksi tulis pada LCD. Saat kondisi RS *don't care* dan Enable 0 maka LCD tetap pada kondisi semula, pengiriman data ke LCD dilakukan saat RS berlogika 0 dan enable berlogika 1. Instruksi dikirim pada LCD bila keadaan RS 1 dan Enable 1. Pin untuk data terkoneksi pada *Port A* mikrokontoler ATMEGA16. Kemudian untuk RS dihubungkan pada *Port* C.6, tulis/baca *(Read/Write)* diberikan logika *low* karena disini LCD bersifat menulis data, dan yang terakhir *Enable* (E) dikendalikan dengan *Port C.7.* Gambar rangkaian LCD ditunjukkan pada gambar 3.6.



Gambar 3.6. Rangkaian *Liquid Crystal Display* (LCD)

3.2.6 Perancangan Rangkaian *Buzzer*

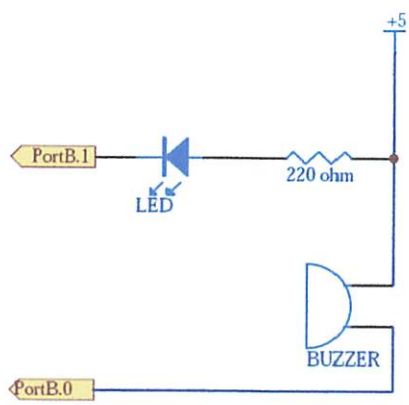
Buzzer digunakan apabila *user* menekan tombol-tombol tertentu yang tidak difungsikan pada *keyboard.* Nilai resistansi pada *buzzer* yang digunakan dalam perancangan alat ini adalah 650 ohm. *Buzzer* ini dapat bekerja pada tegangan 5 volt. Dari nilai tegangan dan resistansi tersebut dapat diperoleh arus yang diperlukan oleh *buzzer.* Berikut ini adalah perhitungan untuk arus yang dibutuhkan oleh *buzzer.*

$$I_{Buzzer} = \frac{V}{R_{Buzzer}}$$

$$= \frac{5}{650\Omega} = 0{,}00769 \text{ A}$$

$$I_{Buzzer} = 7{,}69 \text{ mA}$$

Dari hasil perhitungan dapat dilihat bahwa besarnya arus yang dibutuhkan oleh buzzer dengan sumber tegangan 5 Volt adalah 7,69 mA. Tiap-tiap pin pada mikrokontroler dapat menghasilkan tegangan 5 volt dan arus maksimum sebesar 20 mA. Dengan demikian *buzzer* dapat kita hubungkan langsung ke mikrokontroler.



Gambar 3.7. Rangkaian Buzzer

Selain *buzzer*, LED juga digunakan sebagai penanda adanya kesalahan penekanan tombol. Tegangan kerja LED yang digunakan adalah 2,5 volt dan arus yang dibutuhkan adalah 15 mA. Karena sumber tegangan yang dimliki adalah 5 volt, maka dibutuhkan resistor agar LED bekerja pada tegangan yang diinginkan. Berikut ini adalah perhitungan untuk nilai resitor yang digunakan.

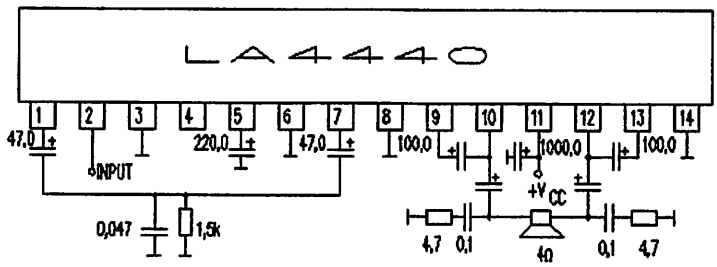
$$R_a = \frac{V - V_{LED}}{I_{LED}}$$

$$= \frac{5 - 2{,}5}{0{,}015}$$

$$= \frac{2,5}{0,015} = 166,66\ \Omega$$

Karena dipasaran tidak terdapat resistor dengan nilai tersebut, maka sebagai penggantinya digunakan resistor denagan nilai resistansi sebesar 220 Ω

3.2.7 Perancangan *Audio Amplifier*

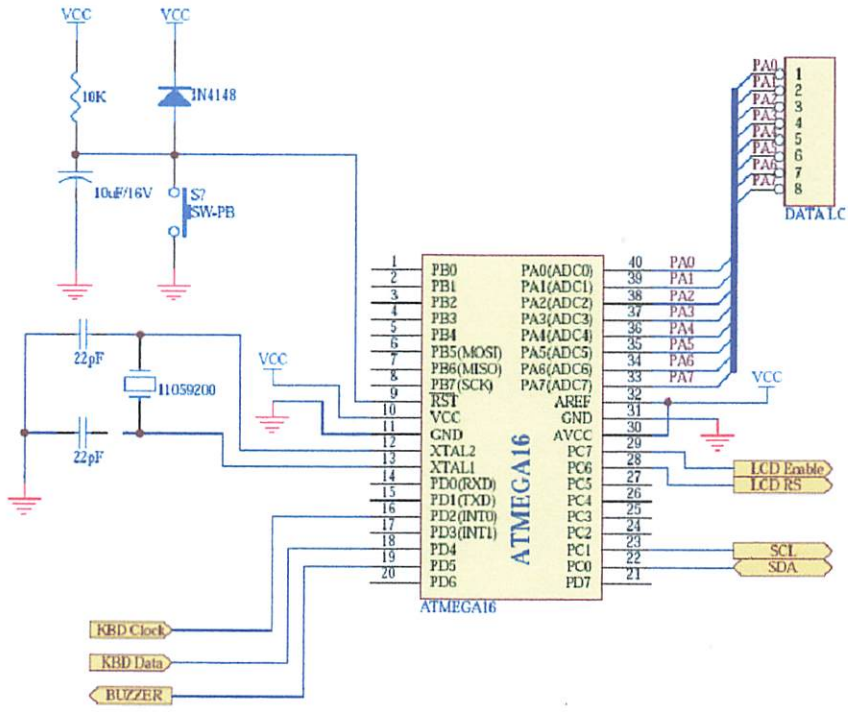


Gambar 3.8. Rangkaian *Audio Amplifier* dengan IC LA4440

Pada perancangan *audio amplifier*, penguat yang digunakan adalah IC LA4440. IC ini memiliki output sebesar 19watt. Agar dapat bekerja, IC ini membutuhkan beberapa komponen tambahan yaitu kapasitor dan resistor yang dirangkai seperti pada gambar 3.8. IC LA4440 dapat bekerja dapat dirangkai secara mono ataupun stereo. Tetapi karena sinyal audio yang dihasilkan oleh modul SP03 adalah sinyal audio mono, maka pada perancangan ini IC LA4440 durangkai secara mono.

Sinyal audio yang dihasilkan oleh modul TTS dimasukkan ke dalam pin ke dua IC LA4440. Sinyal audio ini kemudian akan dikuatkan dan hasilnya akan dukeluarkan pada pin ke 10 dan ke 11. Sinyal audio yang telah dikuatkan tersebut kemudian akan dihubungkan ke *speaker* untuk menghasilkan suara.

3.2.8 Perancangan Minimum Sistem ATmega16



Gambar 3.9. Rangkaian Minimum Sistem dan Alokasi Pin

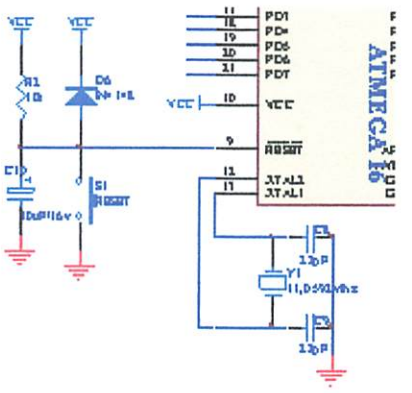
Mikrokontroler yang digunakan ialah mikrokontroler keluarga AVR ATMega 16 yang mempunyai arsitektur *RISC(Reduce Instruction Set)*. Data yang diperoleh dari *keyboard* akan diolah oleh MCU ATMEGA16 untuk kemudian ditampilkan pada LCD. Selain berfungsi untuk menampilkan data ke LCD, mikrokontroller juga berfungsi untuk mengendalikan kerja dari modul *text to speech*. Untuk dapat berkomunikasi dengan modul SP03 TTS, digunakan port serial pada ATmega16.

Berikut ini adalah alokasi untuk pin-pin yang terdapat pada mikrokontroler ATmega 16:

- PA.0-PA.5 digunakan sebagai port output yang akan mengeluarkan data ke LCD M1632.

- PC.6 digunakan sebagai port output operasi RS (Register Select) pada LCD.
- PC.7 digunakan sebagai port output untuk operasi *enable* LCD (aktif atau tidaknya LCD).
- PC.0 terhubung dengan pin SDA modul untuk pengiriman data
- PC.1 terhubung dengan pin SCL modul untuk pengiriman sinyal *clock*
- PD.2 difungsikan sebagai INT0 dan terhubung dengan pin pertama konektor PS/2 untuk menerima sinyal *clock* dari *keyboard.*
- PD.4 terhubung dengan pin ketiga konektor PS/2 untuk menerima data dari *keyboard.*
- PD5 berfungsi untuk mengaktifkan *buzzer* apabila terjadi kesalahan penekanan tombol *keyboard* oleh *user. Buzzer* akan aktif apabila diberikan logika "0" pada pin PD5.
- Pin no.9 ialah pin reset mikrokontroler ATMega 16, reset terjadi bila pin ini diberi logika *low* dengan level tegangan 0 volt selama 1,5 µs atau lebih.
- X_1 dan X_2 sebagai masukan dari rangkaian osilator kristal. Rangkaian osilator kristal terdiri atas osilator 11,0592 MHz, kapasitor C_1 dan C_2 yang masing-masing bernilai 22 pF yang akan membangkitkan pulsa *clock* yang digunakan sebagai penggerak bagi sejumlah operasi internal CPU.

Perancangan rangkaian reset pada mikrokontroler ATMega 16 ialah dengan memberikan logika low pada pin reset mikrokontroler ATMega 16. Rangkaian reset ini diperoleh dari *application note AVR Design Consideration* dari ATMEL. Berikut ialah gambar rancangan rangkaian reset pada ATMega 16 :



Gambar 3.10. Rangkaian reset pada ATMega 16

Osilator pada rangkaian minimum sistem ATMega 16 menggunakan kristal 11,0592 MHz dan kapasitor 22 pF. Nilai kapasitor ini diperoleh dari tabel datasheet tentang penggunaan kapasitor untuk rangkaian osilator / sistem clock pada ATMega 16. Penggunaan kristal 11,0592 MHz ini bertujuan agar perhitungan baudrate tidak mengalami error yang disebabkan karena selisih perhitungan. Perhitungan bautrate pada ATMega 16 dengan menggunakan kristal 11,0592 MHz :

Bautrate yang diinginkan ialah 19200 bps, maka nilai pada *UBRR(USART Baut Rate Register)* dapat ditentukan dengan perhitungan :

$$UBRR = \frac{fosc}{16.Baud} - 1$$

$$UBRR = \frac{11059200}{16.19200} - 1$$

$$UBRR = \frac{11059200}{307200} - 1$$

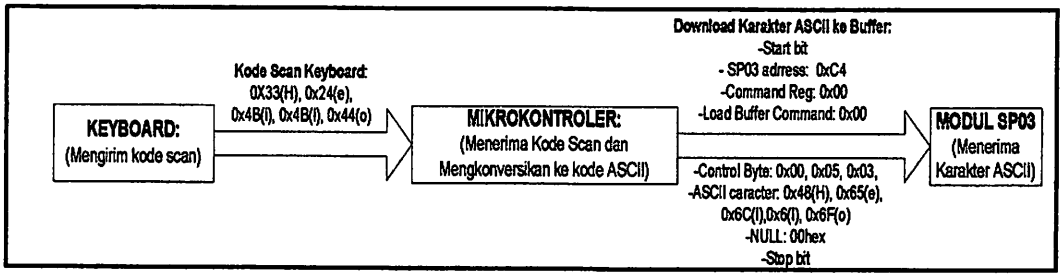
$$UBRR = 36 - 1 = 35 = 23H$$

Penggunaan kristal 11,0592 MHz memungkinkan hasil perhitungan *baudrate* tidak sisa dan *error* dari selisih perhitungan tidak ada.

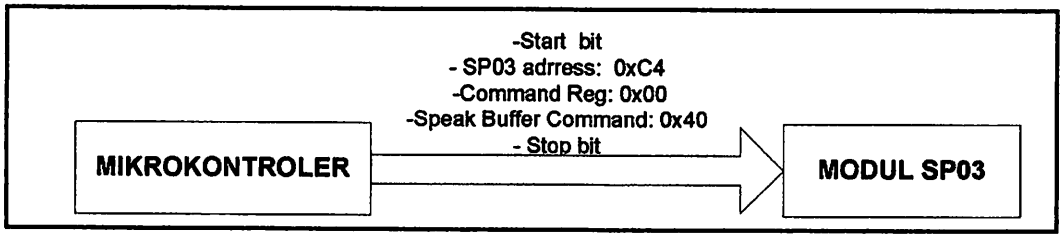
3.3 Perancangan Perangkat Lunak

Dalam perancangan alat ini, mikrokontroler bertindak sebagai jembatan atau perantara untuk mengirimkan data karakter dari *keyboard* ke modul SP03. Data yang dikeluarkan oleh *keyboard* adalah kode scan untuk penekanan tombol. Sedangkan modul SP03 hanya dapat menerima karakter berupa ASCII (*American Standard Code for Information Interchange*). Oleh karena itu mikrokontroler bertugas untuk mengkonversikan kode scan yang diterima dari *keyboard* dan mengkonversikannya menjadi karakter ASCII . Setelah itu barulah mikrokontroler mengirimkan karakter ASCII tersebut ke modul SP03 beserta dengan *command byte* dan *control byte.*

Ada dua hal yang harus dilakukan untuk mengkonversikan teks menjadi suara. Yang pertama adalah *download* teks ke *buffer* SP03 setelah itu mengirimkan *speak buffer command* (instruksi untuk proses konversi).



Gambar 3.11. *Download* Teks ke *Buffer*



Gambar 3.12. Konversi Teks Dalam *Buffer*

Gambar3.11 menunjukkan pengiriman karakter "H,e,l,l,o" ke dalam *buffer* modul SP03. Proses awalnya adalah mirkokontroler akan mengambil data pada

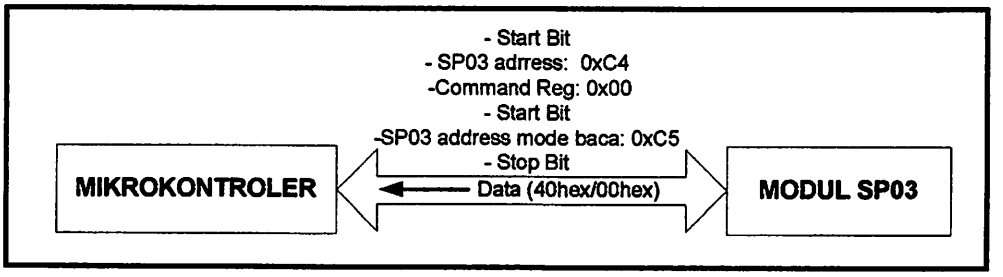
keyboard. Saat keyboard ditekan, maka *keyboard* akan mengirimkan kode scan yang terdapat pada masing-masing tombol. Berikut ini adalah kode scan untuk masing-masing karakter, H=0x33, e=0x24, l=0x4B, l=0x4B, o=0x44. Kode scan yang diterima oleh mikrokontroler kemudian akan dikonversikan ke dalam kode ASCII dan kemudian disimpan dalam memori yang dimiliki oleh mikrokontroler. Masing-masing kode ASCII untuk karakter pada contoh diatas adalah sebagai berikut, H=0x48, e=0x65, l=0x6C, l=6Chex, o=0x6F

Setelah selesai mengetikkan kata yang ingin dikonversikan, *user* dapat menekan tombol enter. Pada saat penekanan tombol enter inilah mikrokontroler akan mengirimkan semua karakter ASCII yang tersimpan pada memori. Seperti yang sudah dijelaskan pada bagian sebelumnya, dalam pengiriman karakter ke modul SP03 harus didahului dengan alamat modul (0xC4), *command register* (0x00), *load buffer command* (0x00), 3 byte pengontrol yang terdiri dari volume level (0x00), pitch level (0x05), dan speed level (0x03). Setelah semuanya itu barulah karakter-karakter ASCII dikirimkan. Pada akhir pengiriman karakter harus ditambahkan dengan NULL *byte* yaitu 00hex.

Teks yang didownloadkan ke modul SP03 akan tersimpan pada *buffer* dengan alamat 0x40. Teks ini tidak akan terkonversikan selama tidak ada instruksi untuk melakukan proses konversi. Instruksi untuk melakukan proses konversi adalah *speak buffer command* (SPKBUF *command*) seperti yang ditunjukkan pada gambar3.7. Pengiriman SPKBUF *command* didahului dengan pengiriman alamat modul (0xC4) dan register *command* (0x00). SPKBUF *command* menunjuk langsung pada alamat dari *buffer* yang akan dikonversikan yaitu 0x40. Setelah

command ini diterima, maka modul SP03 akan mengkonversikan teks yang tersimpan pada *buffer* tersebut.

Untuk melakukan proses *download* kata berikutnya, mikrokontroler harus menunggu proses konversi selesai. Pada saat melakukan proses konversi modul SP03 dalam keadaan *busy* (sibuk). Dalam keadaan ini, modul akan mengabaikan instruksi untuk *download* kata pada *buffer*. Untuk mengetahui status dari modul dapat dilakukan dengan membaca kembali register *command*. Berikut ini adalah langkah-langkah untuk membaca register *command*.

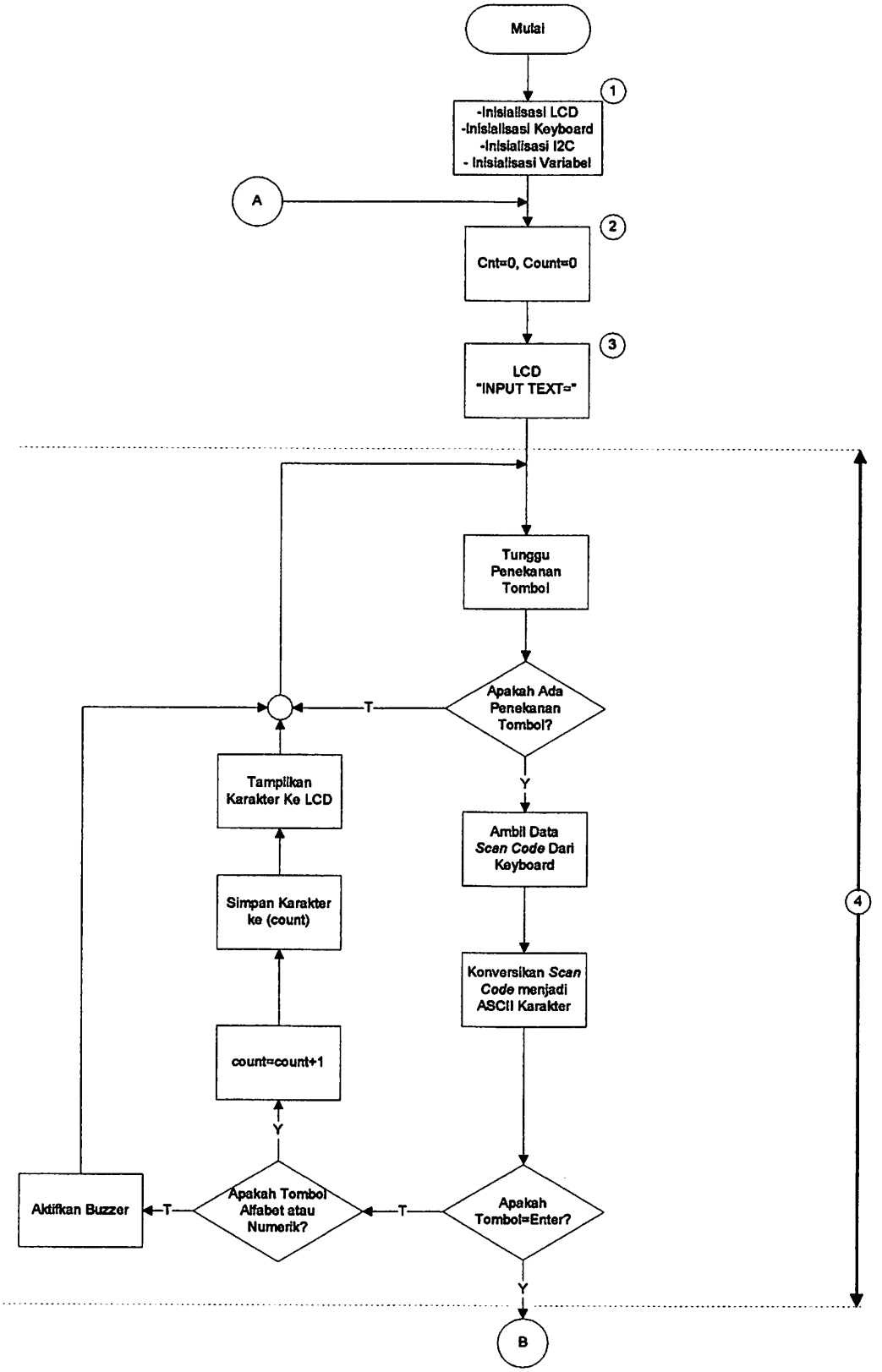


Gambar 3.13. Membaca Status Pada Pada Modul SP03

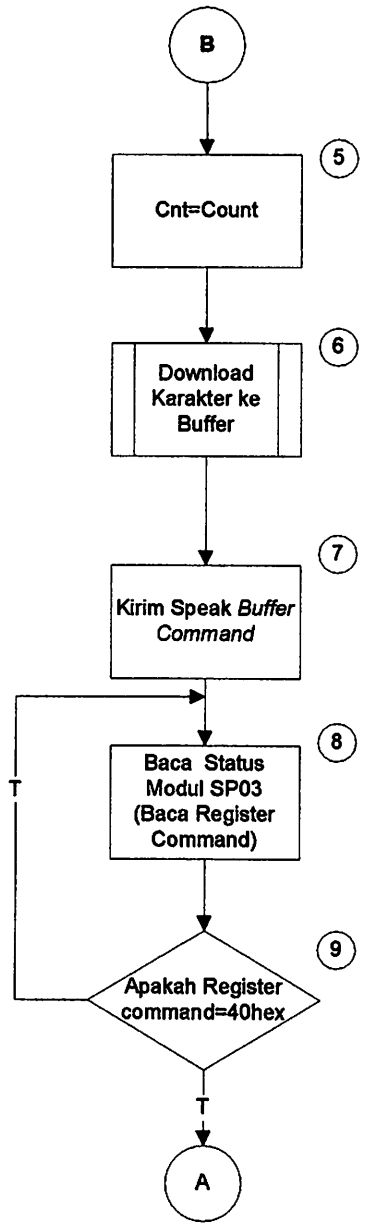
Proses untuk membaca register *command* diawali dengan start bit kemudian mengirimkan alamat modul setelah itu mengirimkan alamat register yang ingin dibaca (*register command*). Setelah menentukan register yang akan dibaca, mikrokontroler akan mengirimkan kembali *start bit* kemudaian alamat dari modul dengan mode baca (*read*) setelah itu mengirimkan *stop bit*.

Setelah itu semua dilakukan, maka modul SP03 akan mengirimkan isi dari register tersebut ke mikrokontroler. Selama modul SP03 masih melakukan proses konversi, modul tersebut akan mengirimkan data 40hex. Apabila proses konversi selesai SP03 akan mengirimkan data 00hex.

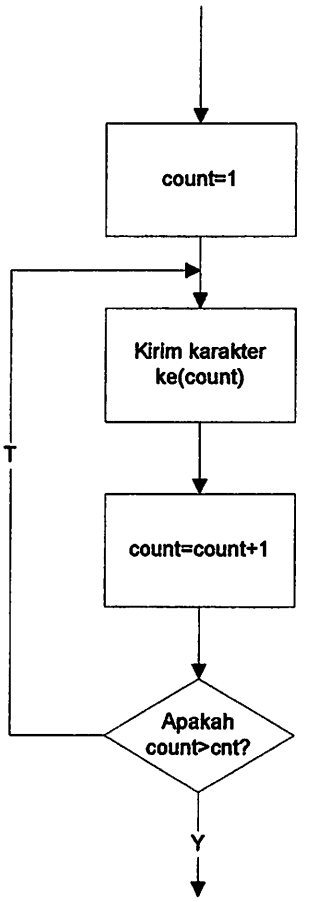
3.3.1 *Flowchart* Program



Gambar3.14. Flowchart Program Bagian ke-1



Gambar 3.15. Flowchart Program Bagian ke-2



Gambar 3.16. Flowchart Sub Program Download Karakter ke Buffer

Gambar flowchart diatas menunjukkan alur dari program dalam mikrokontroller. Dalam penyusunan skripsi ini perancangan program menggunakan bahasa basic dengan bantuan *compiler* bascom AVR. Berikut ini adalah keterangan untuk masing-masing tahap pada flowchart.

1. Inisialisasi LCD, Inisialiasi keyboard, inisialisasi I2C, dan inisialisasi variabel.
   - Inisialisasi LCD merupakan penentuan port pada mikrokontroller yang digunakan untuk mengendalikan kerja dari LCD. Berikut ini adalah kutipan program untuk inisialisasi LCD.

```
'-------------------- Inisialisasi LCD ------------------
Config Lcdpin = Pin , Db4 = Porta.4 , Db5 = Porta.5
Config Lcdpin = Pin , Db6 = Porta.6 , Db7 = Porta.7
Config Lcdpin = Pin , E = Portc.7   , Rs = Portc.6
Config Lcd = 16 * 2
Cursor Off
Display On
'--------------------------------------------------
```

   Komfigurasi untuk pin LCD adalah menentukan *bus* data yaitu pada PortA.4 sampai dengan PortA.7. Selain itu konfigurasi pin juga digunakan untuk menentukan sinyal E pada PortC.7 dan Rs pada PortC.6. *Cursor off* merupakan instruksi untuk tidak menampakkan *cursor* pada LCD. *Display On* merupakan instruksi mengaktifkan tampilan pada LCD

   - Program untuk inisialisasi keyboard adalah seperti dibawah ini.

```
'---------------- Inisialisasi Keyboard -----------------
Config Keyboard = Pind.7 , Data = Pind.6 , Keydata =
Keydata
'--------------------------------------------------
```

Konfigurasi untuk keyboard adalah PinD.7 untuk jalur *clock* sedangkan PinD.6 untuk jalur data. Keydata merupakan nama label untuk tabel konversi *scan code keyboard* menjadi kode ASCII.

- Program untuk inisialisasi I2C adalah seperti dibawah ini

```
'------------------- Inisialisasi I2C -------------------
Config Scl = Portc.1
Config Sda = Portc.0
'--------------------------------------------------------
```

Konfigurasi untuk I2C adalah PinC.1 untuk jalur *clock* sedangkan PinC.0 untuk jalur data.

- Program untuk inisialisasi variabel adalah seperti dibawah ini

```
'-------------------Inisialisasi Variabel---------------

Sp03 = &HC4 'SP03 I2c Address
Cmd_reg = &H00 'Command Register
Ldbuf = &H00 'Load Buffer
Spkbuf = &H40 'Speak Buffer Command
Null = &H00 'Null
Vol = &H03 'variabel Volume
Pitch = &H07 'variabel Pitch
Speed = &H00 'variabel speed

'--------------------------------------------------------
```

2. Variabel "cnt" merupakan jumlah karakter yang ingin didownloadkan ke buffer. Nilai awal yang digunakan untuk variabel ini adalah nol. Ini menandakan bahwa sebelum terjadi penekanan tombol, maka jumlah karakter yang tersimpan pada memori adalah nol. Variabel "count" menunjukkan urutan dari variabel. Misalnya nilai "count=3", ini berarti menunjukkan karakter ke tiga. Sebelum terjadi penekanan nilai ini ditentukan sebagai nol.
3. Berikut ini adalah intruksi untuk menampilkan tulisan "Input Text:" pada LCD.

```
Cls
Lcd "Input Text" ; ":"
```

4. Flowchart pada bagian nomor empat menunjukkan sub-program untuk edit teks. Berikut ini adalah bagian sub-program edit teks.

```
'-------------------- Sub Program Edit Text ----------------
Do
  Kbrd = Getatkbd()
   If Kbrd <> 0 Then
     If Kbrd = Chr(13) Then
       Count = Cnt
       Goto Konversi
     End If

     Incr Cnt
     Kbrd_str = Chr(kbrd)
     Txt(cnt) = Kbrd_str
     Lcd Txt(cnt)
   End If
If Kbrd = 0 Then
   Gosub Buzzer
End If
Loop

'------------------------------------------------------------
```

Untuk melakukan scan data pada *keyboard*, perintah yang diberikan adalah "Getatkbd". Pada saat terjadi penekanan tombol, *keyboard* akan mengirimkan kode scan. Kode scan ini akan diterima oleh keyboard dan akan dikonversikan menjadi karakter ASCII. Untuk mengkonversikannya dibutuhkan tabel konversi. Berikut ini adalah tabel konversi yang ditanamkan pada memori mikrokontroler.

```
'----------Tabel Konversi Kode Scan Keyboard Ke ASCII-------

'Normal Keys Lower Case

Data 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
Data 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
Data 0 , 81 , 33 , 0 , 0 , 0 , 90 , 83 , 65 , 87
Data 34 , 0 , 0 , 67 , 88 , 68 , 69 , 0 , 35 , 0
Data 0 , 32 , 86 , 70 , 84 , 82 , 37 , 0 , 0 , 78
Data 66 , 72 , 71 , 89 , 38 , 0 , 0 , 76 , 77 , 74
Data 85 , 47 , 40 , 0 , 0 , 59 , 75 , 73 , 79 , 61
Data 41 , 0 , 0 , 58 , 95 , 76 , 48 , 80 , 63 , 0
Data 0 , 0 , 0 , 0 , 0 , 96 , 0 , 0 , 0 , 0
Data 13 , 94 , 0 , 42 , 0 , 0 , 0 , 62 , 0 , 0
Data 0 , 8 , 0 , 0 , 49 , 0 , 52 , 55 , 0 , 0
```

```
Data 0 , 0 , 48 , 44 , 50 , 53 , 54 , 56 , 0 , 0
Data 0 , 43 , 51 , 45 , 42 , 57 , 0 , 0

'Shifted Keys Upper Case

Data 0 , 0 , 0 , 0 , 0 , 200 , 0 , 0 , 0 , 0
Data 0 , 0 , 0 , 0 , &H5E , 0 , 0 , 0 , 0 , 0
Data 0 , 113 , 49 , 0 , 0 , 0 , 122 , 115 , 97 , 119
Data 50 , 0 , 0 , 99 , 120 , 100 , 101 , 52 , 51 , 0
Data 0 , 32 , 118 , 102 , 116 , 114 , 53 , 0 , 0 , 110
Data 98 , 104 , 103 , 121 , 54 , 7 , 8 , 44 , 109 , 106
Data 117 , 55 , 56 , 0 , 0 , 44 , 107 , 105 , 111 , 48
Data 57 , 0 , 0 , 46 , 45 , 108 , 48 , 112 , 43 , 0
Data 0 , 0 , 0 , 0 , 0 , 92 , 0 , 0 , 0 , 0
Data 13 , 0 , 0 , 92 , 0 , 0 , 0 , 60 , 0 , 0
Data 0 , 0 , 8 , 0 , 0 , 49 , 0 , 52 , 55 , 0
Data 0 , 0 , 48 , 44 , 50 , 53 , 54 , 56 , 0 , 0
Data 0 , 43 , 51 , 45 , 42 , 57 , 0 , 0
```

Kode scan yang dikirim oleh *keyboard* akan menunjuk kepada data ASCII yang terdapat pada tabel konversi. Kode scan tombol tertentu jika kita tambah satu akan menunjukkan urutan dari data kode ASCII. Contohnya adalah tombol Q yang mempunyai kode scan 15hex(21 dec) jika kita tambahkan satu akan menjadi 22. Data dengan urutan ke-22 inilah yang merupakan data ASCII dari karakter Q. Untuk tombol-tombol yang tidak difungsikan pada perancangan alat, akan dikonversikan menjadi kode ASCII 00hex. Jika kode ASCII yang didapat oleh mikrokontroler adalah 00hex, maka program akan meloncat kepada sub-program untuk mengkatifkan buzzer.

```
'-----------------Sub Program Mengaktifkan Buzzer----------
Buzzer:
Reset Portb.0
Reset Portb.1
Waitms 500
Set Portb.0
Set Portb.1
Return
```

Apabila kode ASCII yang didapatkan tidak sama dengan nol maka mikrokontroler akan menyimpan karakter ASCII tersebut. Variabel "cnt" merupakan nomor urutan karakter, sedangkan variabel "count" merupakan jumlah dari karakter yang tersimpan. Untuk variabel "cnt" akan bertambah ketika ada karakter ASCII baru yang tersimpan.

Sub program ini akan melakukan looping secara terus menerus sampai dideteksi adanya penekanan tombol enter.

```
------------------------------------------------------------
If Kbrd = Chr(13) Then
       Count = Cnt
       Goto Konversi
     End If
------------------------------------------------------------
```

"Chr(13)" merupakan data karakter untuk enter. Saat terjaadi penekanan tombol enter, program akan melompat ke sub-program dengan label konversi.

5. Variabel "count" merupakan jumlah dari karakter yang tersimpan. Variabel ini harus dikondisikan sebelum menuju ke label konversi. Nilai dari variabel ini adalah sama dengan variabel "cnt" terakhir sebelum melompat ke label konversi.

```
-------------
Count = Cnt
Goto Konversi

-------------
```

6. Berikut ini adalah sub-program untuk download teks.

```
'---------------------Load Data Ke Buffer-----------------
Konversi:
Cnt = 1
I2cstart
Delay
I2cwbyte Sp03
Delay
I2cwbyte Cmd_reg
Delay
I2cwbyte Ldbuf
```

```
Delay
I2cwbyte Vol
Delay
I2cwbyte Pitch
Delay
I2cwbyte Speed
Delay
Do
   I2cwbyte Txt(cnt) 'ASCII Char
   Delay
   Incr Cnt
   If Cnt > Count Then
     Goto Lanjut
   End If
Loop
Delay


Lanjut:
I2cwbyte Null
Delay
I2cstop
Delay
'-------------------------------------------------------
```

Proses pengiriman data karakter didahului dengan alamat SP03, *command register*, *load buffer command*, volume level, *pitch level*, dan *speed level.* Data dikirimkan mulai dari karakter pertama (cnt=1) sampai pada data terakhir (Cnt=Count)

7. Berikut ini adalah sub-program untuk *Speak Buffer Command.*

```
'---------------------Speak Buffer Command-----------------
I2cstart
Delay
I2cwbyte Sp03
Delay
I2cwbyte Cmd_reg
Delay
I2cwbyte Spkbuf
Delay
I2cstop
Gosub Tunggu
'-------------------------------------------------------
```

Setelah mengirimkan *speak buffer command*, program akan menuju ke label tunggu untuk membaca satus apakah proses konversi sudah selesai atau belum.

8. Berikut ini adalah sub-program untuk baca status

```
'--------------------Speak Buffer Command-----------------
Tunggu:
do
  I2cstart
  Delay
  I2cwbyte Sp03              'alamat modul dengan mode tulis
  Delay
  I2cwbyte Cmd_reg           'register yang ingin dibaca
  I2cstart
  I2creceive &HC5 , Value    'alamat modul dengan mode baca
  Delay
  I2cstop
  Loop Until Value = 0
  Delay
Return
```

Dalam penggalan program tersebut register yang dibaca adalah command register. Isi dari register ini akan tersimpan pada variabel "value".

9. Pada saat melakukan konversi teks, isi dari register command adalah 40hex. Sebaliknya setelah selesai melakukan konversi, isi dari register command adalah 00hex. Sub program untuk membaca status akan diulang terus menerus sampai didapati register command bernilai 00hex.

BAB IV

PENGUJIAN DAN HASIL ANALISA

Pengujian dilakukan untuk mengetahui sejauh mana alat dapat bekerja sesuai dengan perencanaan. Langkah pengujian dilakukan melalui 2 tahap, yakni pengujian pada setiap blok dan pengujian pada sistem keseluruhan. Tahap pertama dimaksudkan untuk mengetahui sejauh mana blok-blok rangkaian dapat berjalan, sedangkan tahap kedua dilakukan setelah diperoleh kepastian bahwa tiap blok rangkaian telah berjalan sesuai rencana.

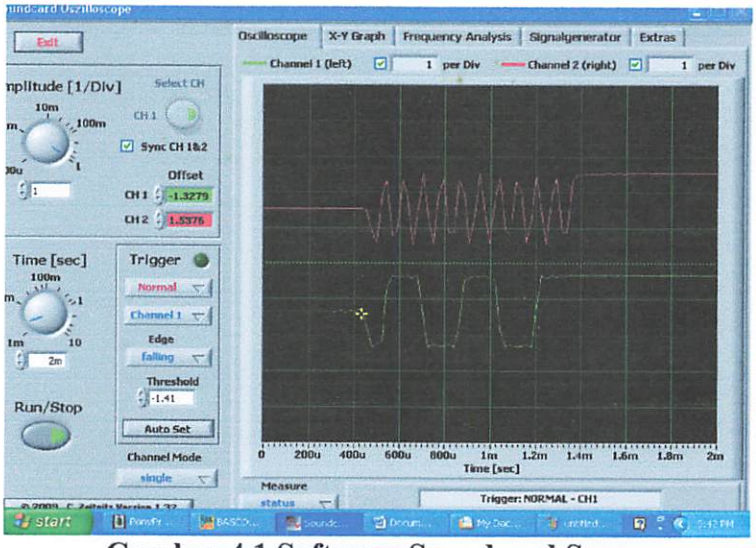
4.1 Pengujian Komunikasi Antara Mikrokontroler Dengan *Keyboard*

4.1.1 Alat yang digunakan :

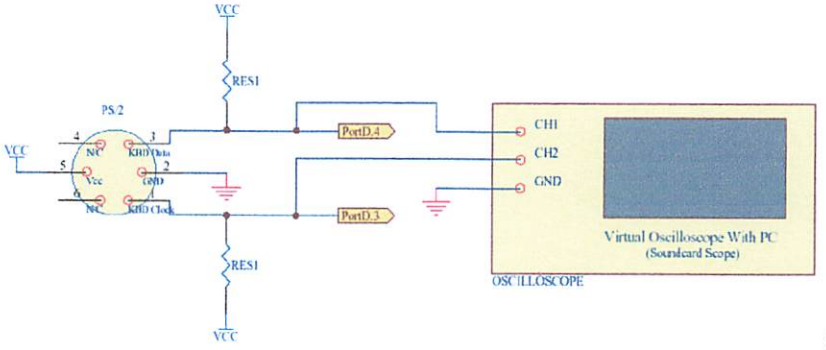
1. *Keyboard*
2. Mikrokontroler ATmega16
3. Konektor PS/2
4. PC dengan software Soundcard Scope (*virtual oscilloscope*)

4.1.2 Langkah Pengujian :

1. Menyiapkan program mikrokontroler yang berfungsi untuk mengambil data *scan code* dari *keyboard.*
2. Menghubungkan *keyboard* ke mikrokontroler dengan konektor PS/2 .
3. Menghubungkan jalur *keyboard clock* pada CH1 *oscilloscop* dan *keyboard data* pada CH2 *oscilloscope.*
4. Memberikan catu daya pada mikrokontroler dan *keyboard.*
5. Menekan tombol pada *keyboard.*
6. Mengamati sinyal yang ditunjukkan pada *oscilloscope.*



Gambar 4.1 Software Soundcard Scope



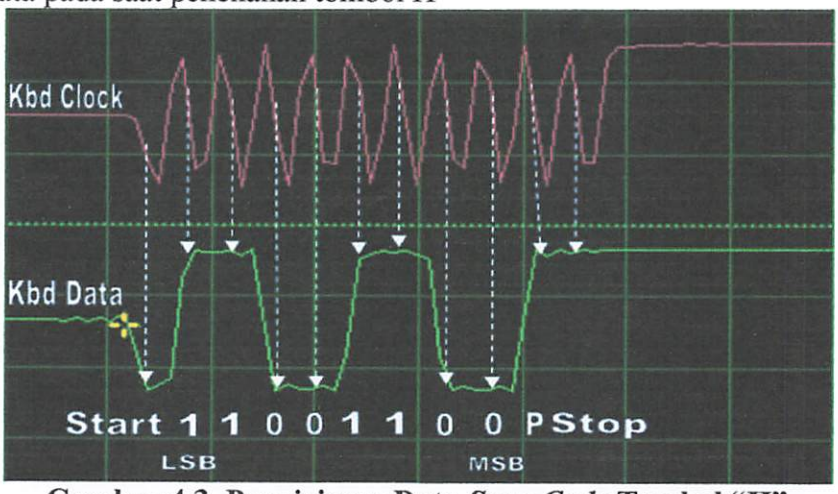
c

Gambar 4.2. Pengujian Komunikasi Antara *Keyboard* Dengan Mikrokontroler

4.1.3 Hasil Pengujian Komunikasi Antara Mikrokontroler Dengan *Keyboard*

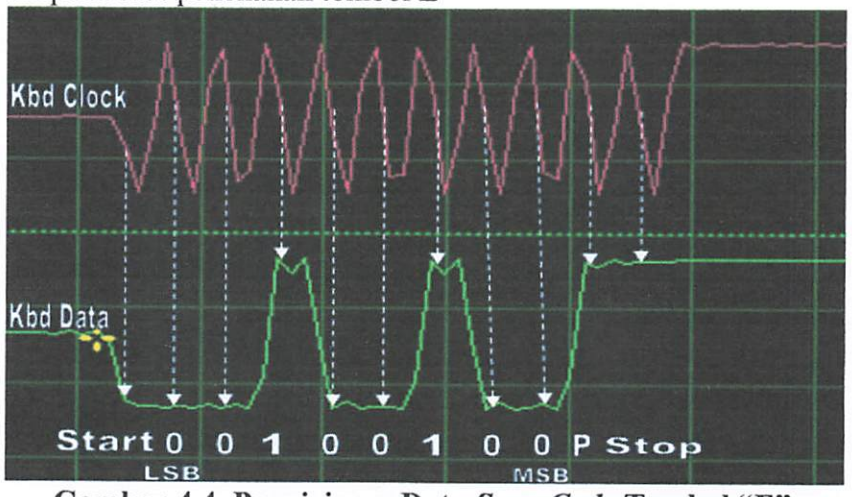
Pengujian pada komunikasi *keyboard* dengan mikrokontroler dilakukan dengan menekan tombol pada *keyboard* dan mengamati sinyal data dan sinyal *clock* pada *oscilloscope*. Sinyal data merupakan *scan code* (kode scan) yang dikirimkan *keyboard* ke mikrokontroler. Berikut ini data *scan code* yang didapatkan pada pengujian *keyboard* dengan penekanan tombol H,E,L,L,O.

- Data pada saat penekanan tombol H



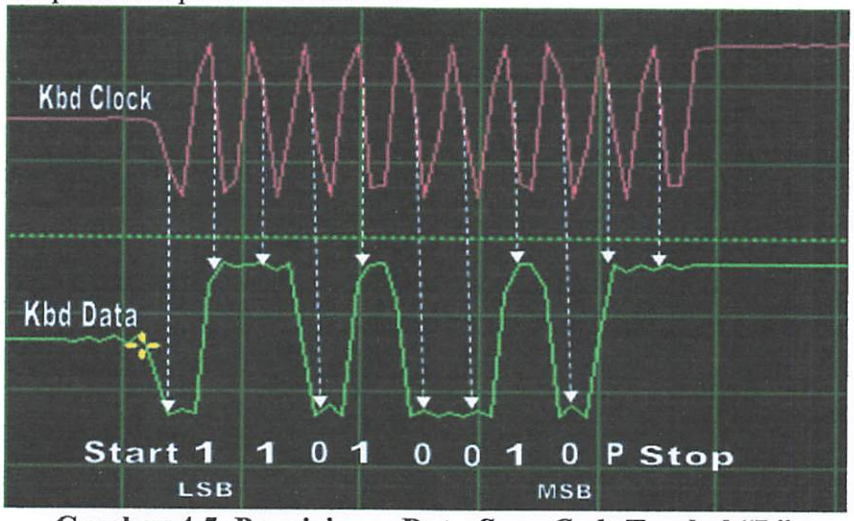
Gambar 4.3. Pengiriman Data *Scan Code* Tombol "H"

- Data pada saat penekanan tombol E



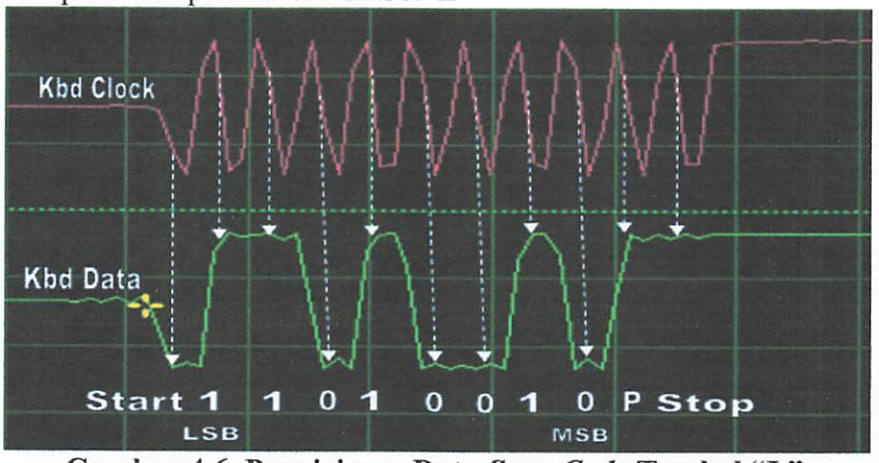
Gambar 4.4. Pengiriman Data *Scan Code* Tombol "E"

- Data pada saat penekanan tombol L



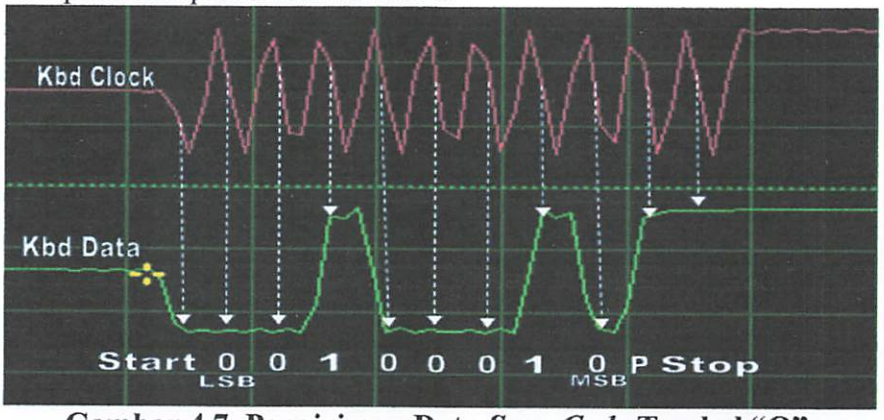
Gambar 4.5. Pengiriman Data *Scan Code* Tombol "L"

- Data pada saat penekanan tombol L



Gambar 4.6. Pengiriman Data *Scan Code* Tombol "L"

- Data pada saat penekanan tombol O



Gambar 4.7. Pengiriman Data *Scan Code* Tombol "O"

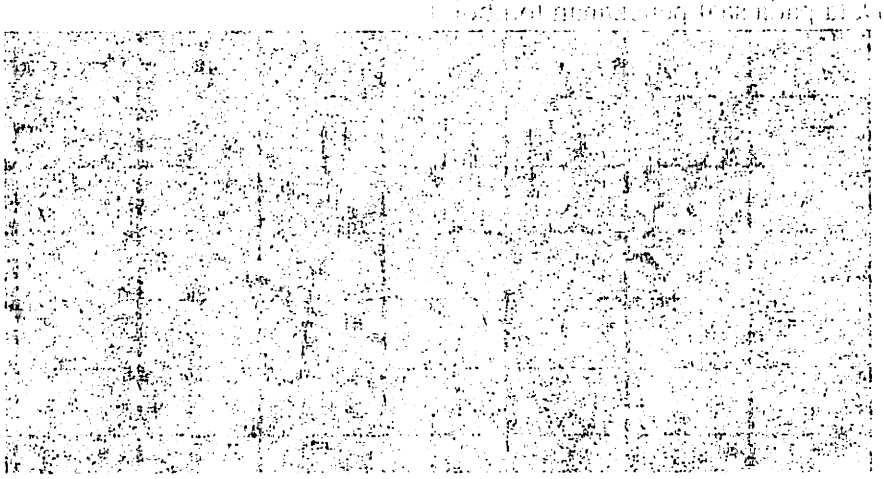
Tabel 4.1. Hasil Pengujian Komunikasi Antara *Keyboard* Dengan Mikrokontroler

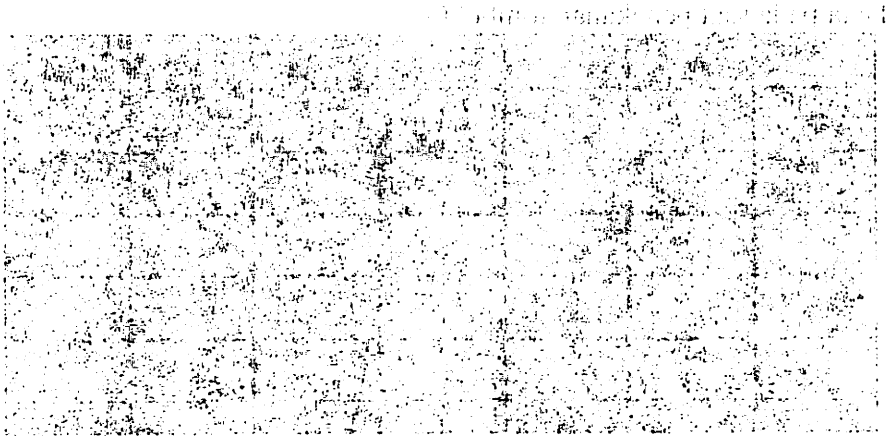
No	Tombol	Kode Scan (Biner)	Kode Scan (Hexa)
1	H	00110011b	33 Hex
2	E	00100100b	24 Hex
3	L	01001011b	4B Hex
4	L	01001011b	4B Hex
5	O	01000100b	44 Hex

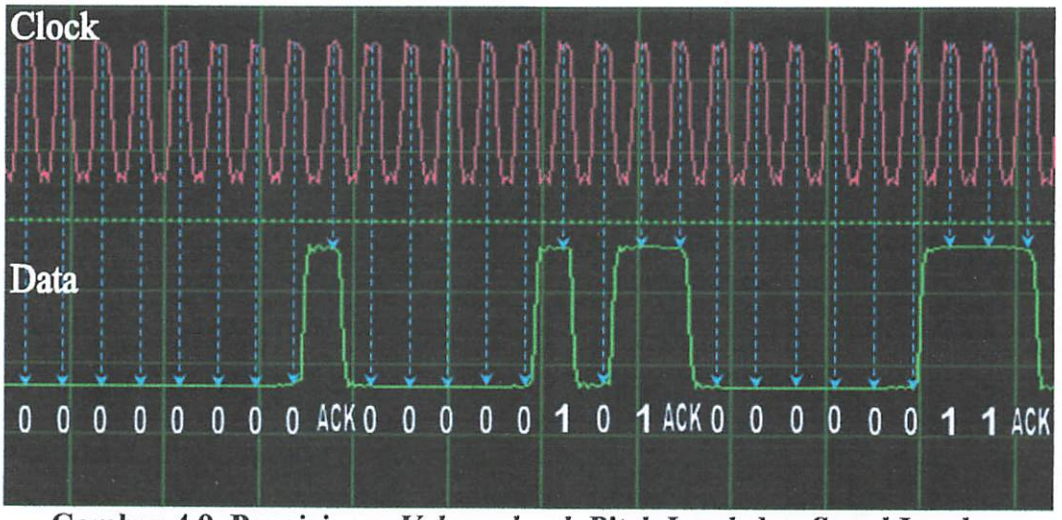
4.2 Pengujian Komunikasi Antara Mikrokontroler Dengan Modul SP03

4.2.1 Alat yang digunakan :

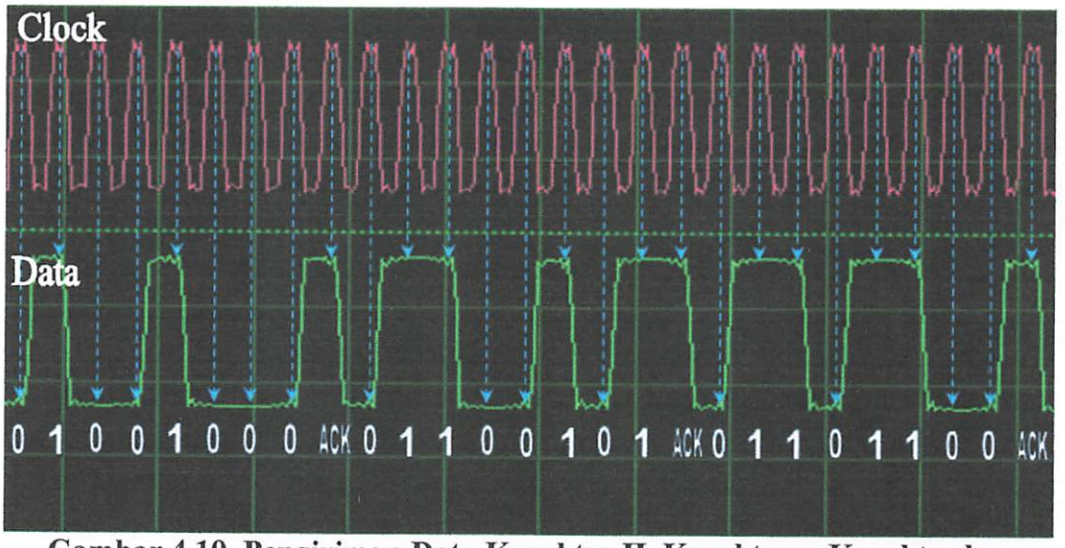
1. Modul *Text to Speech* SP03



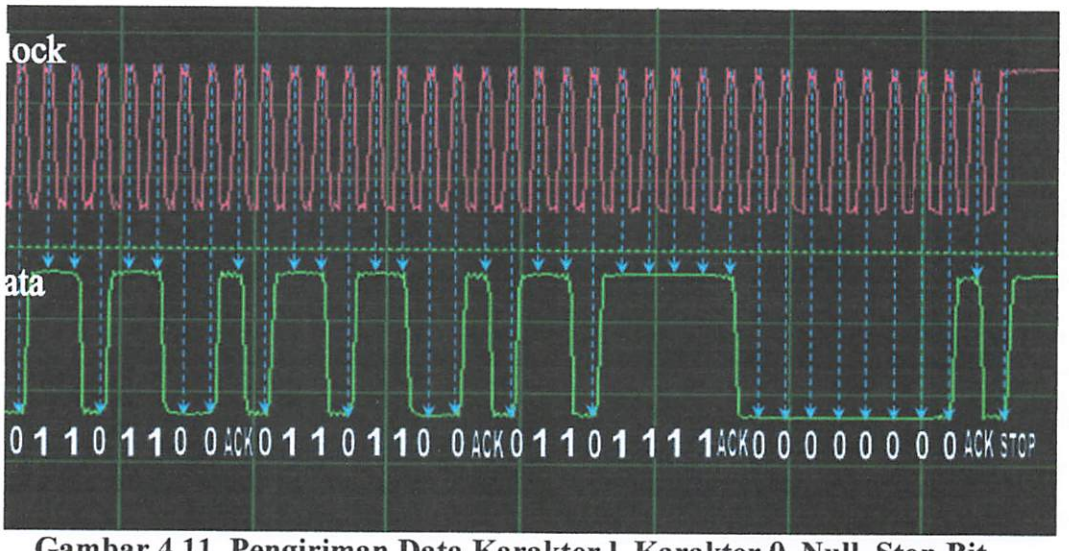




Gambar 4.9. Pengiriman *Volume level*, *Pitch Level*, dan *Speed Level*



Gambar 4.10. Pengiriman Data Karakter H, Karakter e, Karakter l

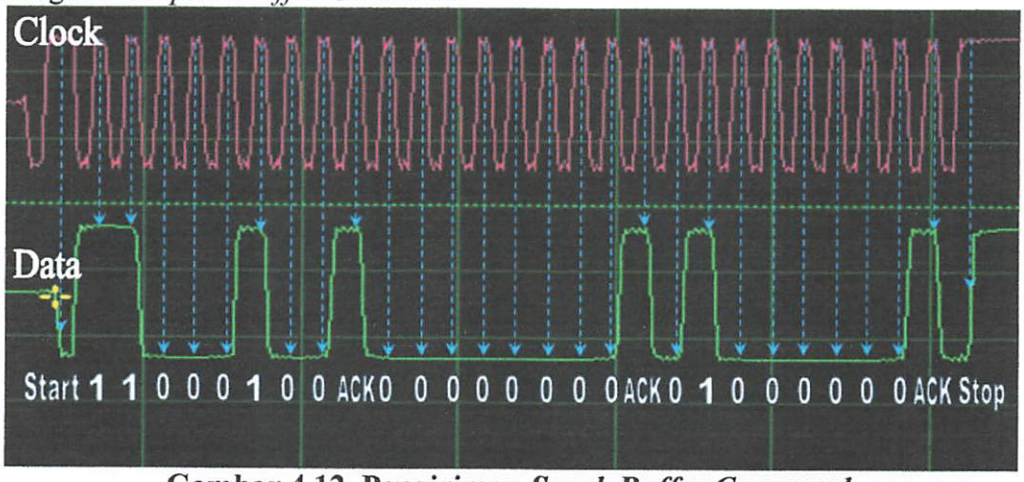


Gambar 4.11. Pengiriman Data Karakter l, Karakter 0, Null, Stop Bit

Tabel 4.2. Hasil Pengujian *Download* Teks ke *Buffer*

No.	Pengiriman	Data Biner	Data Hexa
1.	*Start Bit*		
2.	*SP03 Address*	11000100b	C4hex
3.	*Command Register*	00000000b	00hex
4.	*Load Buffer Command*	00000000b	00hex
5.	*Volume Level*	00000000b	00hex
6.	*Pitch Level*	00000101b	05hex
7.	Speed Level	00000011b	03hex
8.	Karakter H	01001000b	48hex
9.	Karakter e	01100101b	65hex
10.	Karakter l	01101100b	6CHex
11.	Karakter l	01101100b	6CHex
12.	Karakter o	01101111b	6Fhex
13.	*Null*	00000000b	00hex
14.	*Stop Bit*		

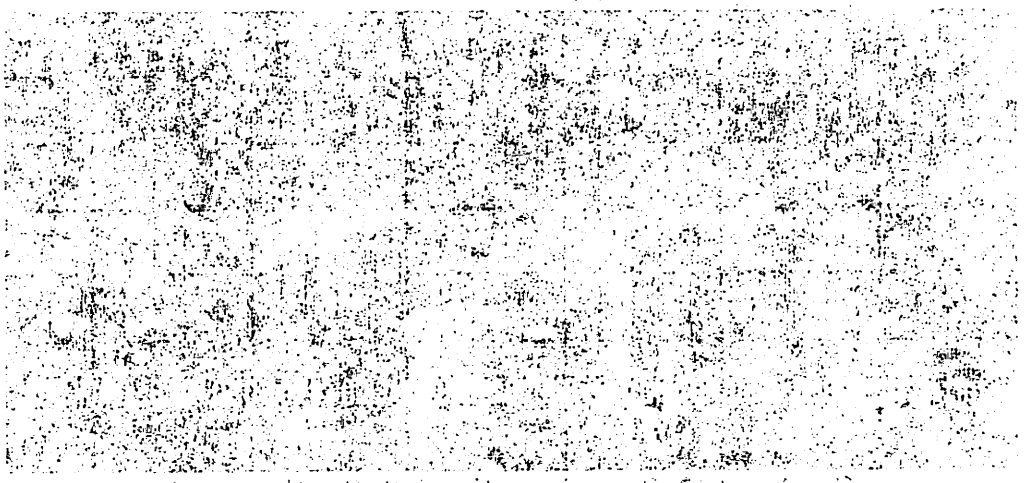
- Pengiriman *Speak Buffer Command*



Gambar 4.12. Pengiriman *Speak Buffer Command*

Tabel 4.3. Hasil Pengujian Pengiriman *Speak Buffer Command*

No.	Pengiriman	Data Biner	Data Hexa
1.	*Start Bit*		
2.	*SP03 Address*	11000100b	C4hex
3.	*Command Register*	00000000b	00hex
4.	*Speak Buffer Command*	01000000b	40hex
5.	*Stop Bit*		



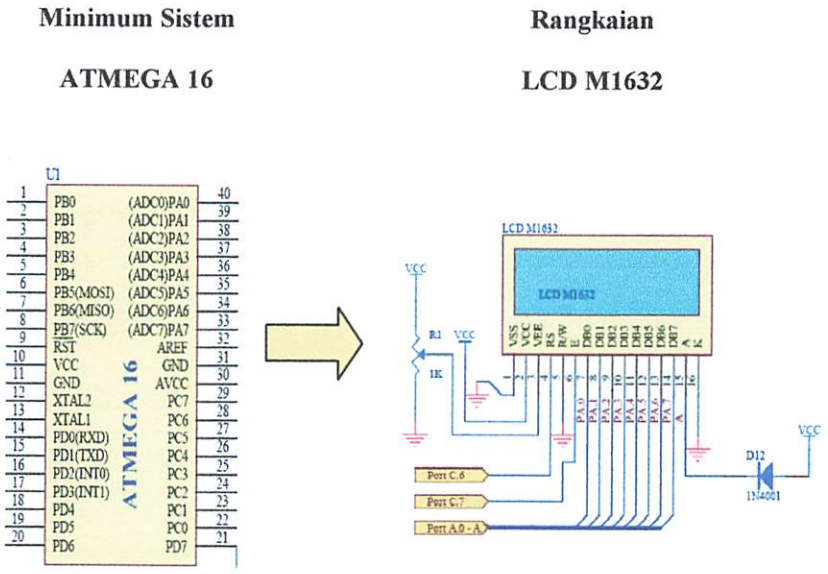
4.3 Pengujian Rangkaian LCD

4.3.1 Alat dan Bahan

1. Rangkaian LCD
2. Rangkaian Mikrokontroler ATMEGA16
3. Catu daya 5 Volt

4.3.2 Langkah Pengujian

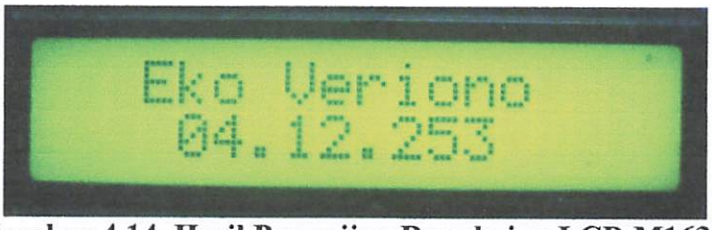
1. Mempersiapkan *software* mikrokontroler yang berfungsi untuk inisialisasi LCD dan menampilkan suatu kata-kata pada baris pertama bertuliskan "Eko Veriono", baris kedua bertuliskan "0412253".
2. Mempersiapkan rangkaian pengujian seperti Rangkaian pengujian LCD pada gambar 4.13.
3. Mengamati hasil pengujian.



Gambar 4.13. Blok Diagram Pengujian Rangkaian LCD M1632

4.3.3 Hasil Pengujian

Setelah melakukan pengujian LCD diatas didapatkan tulisan "Eko Veriono' pada baris pertama dan "0412253" pada baris kedua . Dengan demikian setelah mengetahui hasil pengujian rangkaian LCD dapat disimpulkan bahwa rangkaian LCD tersebut telah berfungsi sesuai yang direncanakan.



Gambar 4.14. Hasil Pengujian Rangkaian LCD M1632

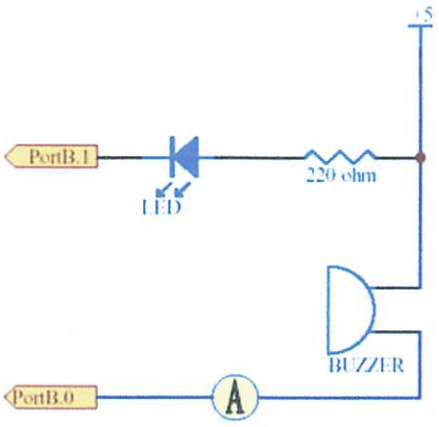
4.4 Pengujian Buzzer

4.4.1 Alat dan Bahan

1. Rangkaian Buzzer
2. Mikrokontroler
3. Multimeter

4.4.2 Langkah Pengujian

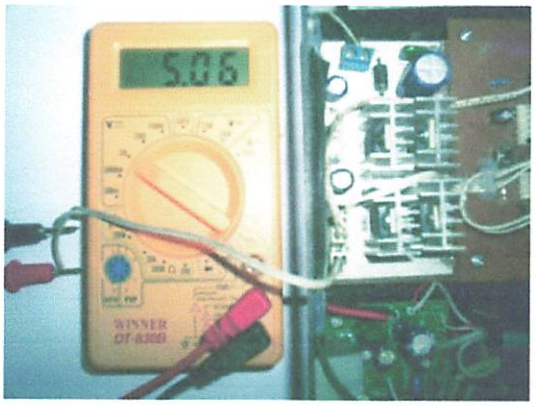
1. Menghubungkan buzzer pada PortB.0 dan LED pada PortB.1
2. Menghubungkan mutimeter secara seri dengan buzzer
3. Mengamati arus yang ditunjukkan pada multimeter
4. Mengukur tegangan sumber



Gambar 4.15. Pengujian Rangkaian Buzzer

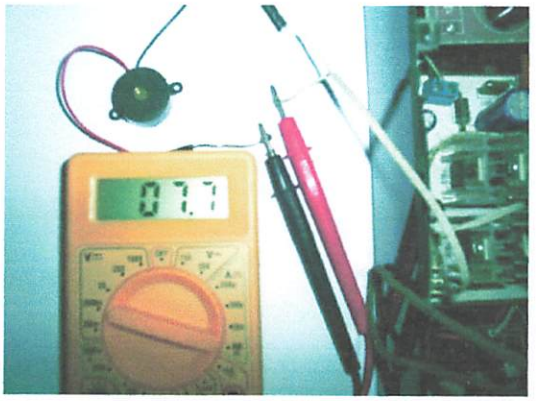
4.4.3 Hasil Pengujian

- Pengukuran Tegangan Sumber (Vcc)

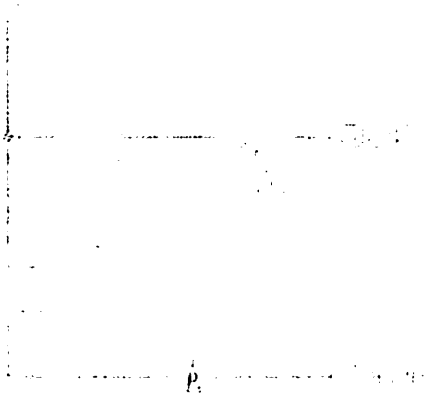


Gambar 4.16. Pengukuran Tegangan Sumber

- Pengukuran Arus Yang Mengalir Pada Buzzer



Gambar 4.17. Pengukuran Arus Yang Mengalir Pada Buzzer



- Perhitungan Arus Yang Mengalir Pada Buzzer

R_{buzzer}= 650 Ω

$$I_{buzzer} = \frac{V}{R_{Buzzer}}$$

$$= \frac{5{,}06}{650}$$

$$= 0.0078 = 7.8 \text{ mA}$$

$I_1 = Hasil\_perhitungan$

$I_2 = Hasil\_pengujian$

$$\%Error = \frac{|I_1 - I_2|}{I_1} x100\%$$

$$\%Error = \frac{|7.8 - 7.7|}{7.8} x100\% = 1.3\%$$

4.5 Pengujian Mikrokontroller ATMega16

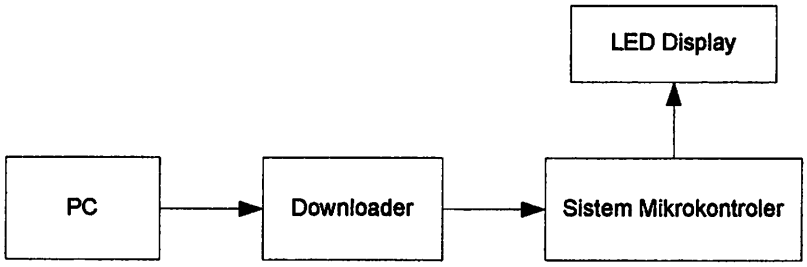
4.5.1 Alat dan Bahan

1. *Personal Computer* (PC) yang dilengkapi LPT1
2. *Downloader* mikrokontroller ATMega 16
3. ISP *programmer* dengan *connector* LPT1
4. Minimum sistem Mikrokontroler
5. Catu daya 5 volt

4.5.2 Langkah Pengujian

Tahapan Pengujian mikrokontroller ATMega 16 sebagai berikut:

1. Rangkaian dibuat seperti gambar 4.18.



Gambar 4.18. Diagram Blok Pengujian Mikrokontroler

2. Memberikan catu daya 5 volt.
3. Membuat program yang digunakan untuk menguji mikrokontroler. Program yang digunakan dalam pengujian mikrokontroler ini merupakan program yang sederhana yang meletakkan $F0_H$ dan $0F_H$ secara bergantian pada *PortC* ATMega 16.
4. Mengamati keluaran pada LED Display.

4.5.3 Hasil dan Analisa Pengujian

Tabel 4.4. Hasil Pengujian Sistem Mikrokontroler

Kondisi	Keluaran pada LED *Display*							
	Bit 0	*Bit 1*	*Bit 2*	*Bit 3*	*Bit 4*	*Bit 5*	*Bit 6*	*Bit 7*
Satu	0	0	0	0	1	1	1	1
Dua	1	1	1	1	0	0	0	0

Keterangan :

- Kondisi bit *low* (0) = LED menyala
- Kondisi bit *height* (1) = LED mati

Dari hasil pengujian pada tabel 4–4 dapat dilihat bahwa *port C* memberikan logika $0F_H$ dan $F0_H$ secara bergantian sesuai dengan isi program.

4.6 Pengujian Keseluruhan Sistem

Ada dua hal yang dilakukan untuk pengujian keseluruhan sistem yaitu, pengujian tampilan menu dan pengujian sinyal audio yang dikeluarkan oleh modul SP03.

4.6.1 Alat dan Bahan

1. *Keyboard*
2. Modul *Text to Speech* SP03
3. Rangkaian Minimum Sistem Mikrokontroler ATmega16
4. Rangkaian LCD
5. Rangkian *Buzzer*
6. Penguat Audio
7. Catu Daya

4.6.2 Langkah Pengujian

1. Menghubungkan SDA modul ke PortC.0 dan pin SCL modul ke PortC.1.
2. Menghubungkan Kbd Clock ke PortD.2 dan Kdb Data ke PortD.4.
3. Menghubungkan rangkaian *buzzer* ke PortB.0 dan Port B.1
4. Menghubungkan data LCD ke port A, RS ke port C.6, E ke port C.7.
5. Menghubungkan *audio outputt* dari modul SP03 ke oscilloscope
6. Mengatur beberapa menu
7. Memasukkan beberapa kata untuk dikonversikan
8. Mengamati hasil pengujian.

4.6.3 Pengujian Tampilan Menu

Setelah semua Rangkaian dihubungkan dan diberikan catu daya maka sistem akan menampilkan daftar menu sebagai berikut:

Tabel 4.5. Daftar Menu Pada Sistem

No	Menu
1	*Volume*
2	*Pitch*
3	*Speed*
4	*Text*

Gambar 4.20. Menu Awal

Pemilihan menu dilakukan dengan menekan tombol pada *keyboard* sesuai dengan nomor menu yang ditampilkan. Penekanan tombol selain nomor menu akan mengakibatkan *buzzer* berbunyi sebagai indikator kesalahan penekanan tombol.

4.6.3.1 Pengaturan Level Volume

Program akan masuk ke menu "Volume" apabila terjadi penekanan tombol 1 (satu). Menu ini digunakan untuk mengatur volume yang dihasilkan pada saat proses konversi ke suara. Terdapat 8 level volume pada menu ini yaitu mulai dari level 0 sampai dengan level 7. Volume awal ditentukan pada level 4.

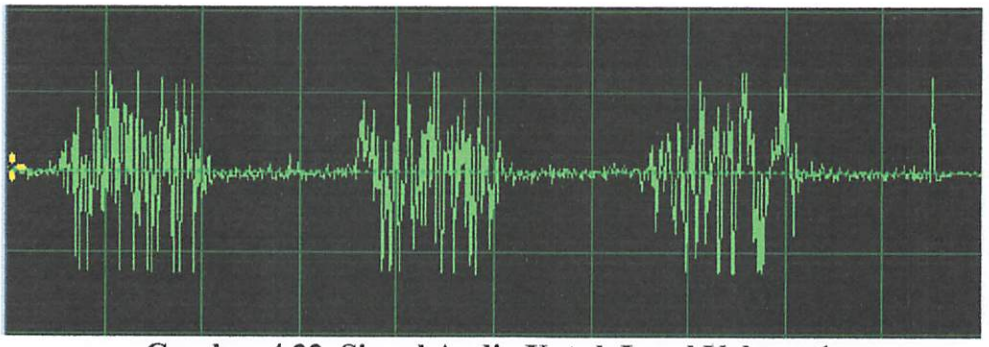
Gambar 4.21. Tampilan Menu Volume

Untuk melakukan pengaturan level volume, tombol yang digunakan pada keyboard adalah tombol "+" dam "-".

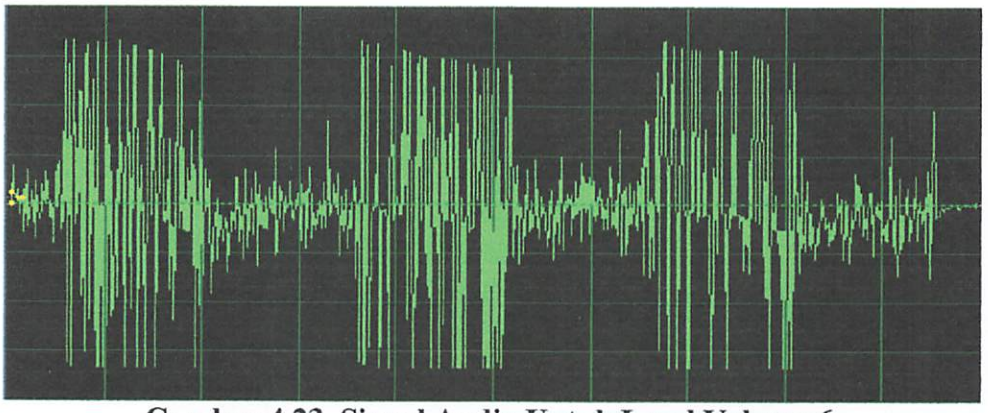
Tabel 4.6. Pengaturan Level Volume

No	Tombol	Keterangan
1	(+)	Level volume naik
2	(-)	Level volume turun
3	Enter	Kembali Ke Menu Utama
4	Selain (-), (+), dan Enter	*Buzzer* Aktif

Perubahan level volume dapat diamati dengan melihat sinyal audio yang dihasilkan oleh modul SP03. Pada pengujian level volume ini dilakukan dengan memasukkan teks "How are you" dan mengamati perubahan sinyal audio pada oscilloscope apabila level volume di ubah.



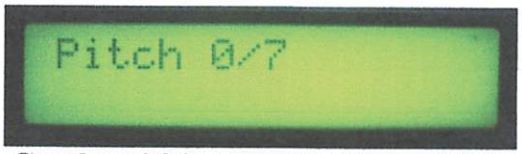
Gambar 4.22. Sinyal Audio Untuk Level Volume 1



Gambar 4.23. Sinyal Audio Untuk Level Volume 6

4.6.3.2. Pengaturan Level *Pitch*

Program akan masuk ke menu "Pitch" apabila terjadi penekanan tombol 2 (dua). Menu ini digunakan untuk mengatur frekeunsi sinyal audio yang dihasilkan pada saat proses konversi ke suara. Terdapat 8 level *pitch* pada menu ini yaitu mulai dari level 0 sampai dengan level 7. *Pitch* awal ditentukan pada level 0.



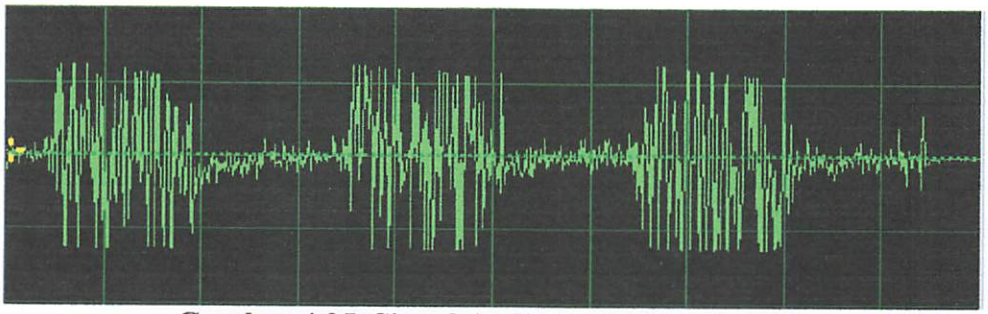
Gambar 4.24. Tampilan Menu *Pitch*

Untuk melakukan pengaturan level *speed*, tombol yang digunakan pada *keyboard* adalah tombol "+" dam "-".

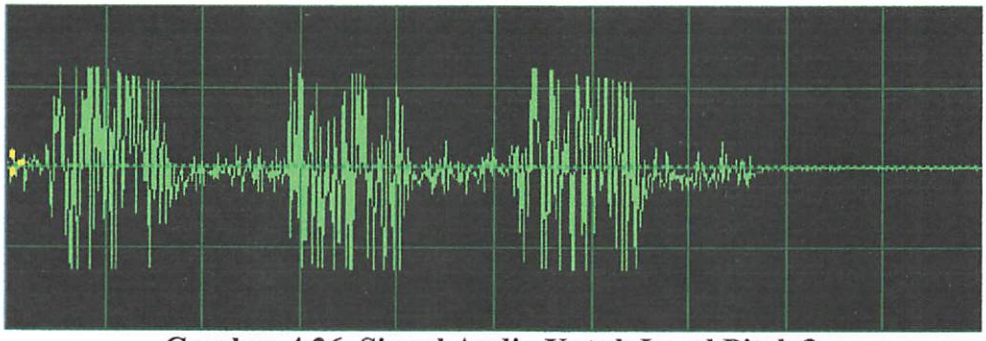
Tabel 4.7. Pengaturan Level *Pitch*

No	Tombol	Keterangan
1	(+)	Level *pitch* naik
2	(-)	Level *pitch* turun
3	Enter	Kembali Ke Menu Utama
4	Selain (-), (+), dan Enter	*Buzzer* Aktif

Perubahan level *pitch* dapat diamati dengan melihat sinyal audio yang dihasilkan oleh modul SP03. Pada pengujian level *pitch* ini dilakukan dengan memasukkan teks "How are you" dan mengamati perubahan sinyal audio pada oscilloscope apabila level *pitch* di ubah.



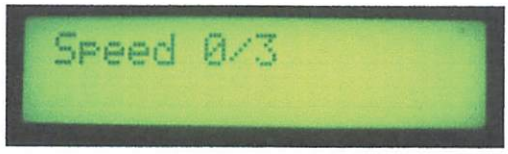
Gambar 4.25. Sinyal Audio Untuk Level *Pitch*



Gambar 4.26. Sinyal Audio Untuk Level Pitch 3

4.6.3.3 Pengaturan Level *Speed*

Program akan masuk ke menu "Speed" apabila terjadi penekanan tombol 3(tiga). Menu ini digunakan untuk mengatur kecepatan pengucapan kata hasil dari proses konversi. Terdapat 3 level *speed* pada menu ini yaitu mulai dari level 0 sampai dengan level 2. *Speed* awal ditentukan pada level 0.



Gambar 4.27. Tampilan Menu *Speed*

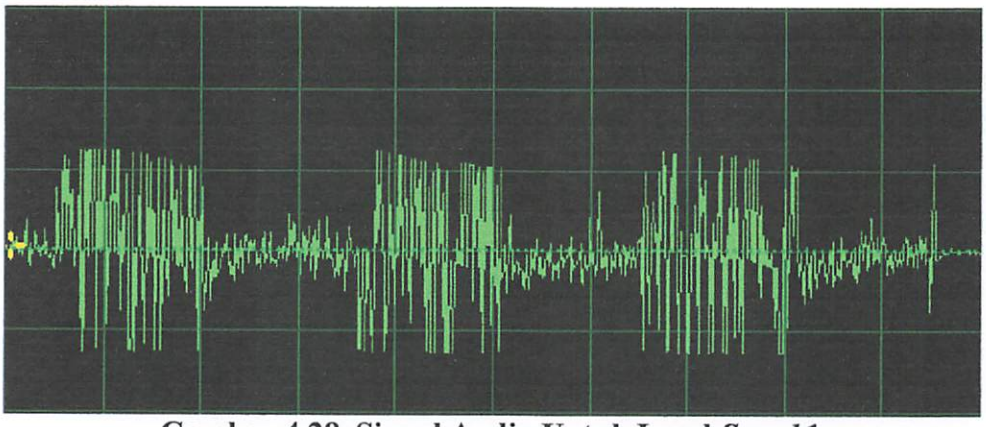
Untuk melakukan pengaturan level *speed*, tombol yang digunakan pada *keyboard* adalah tombol "+" dam "-".

Tabel 4.8. Pengaturan Level *Pitch*

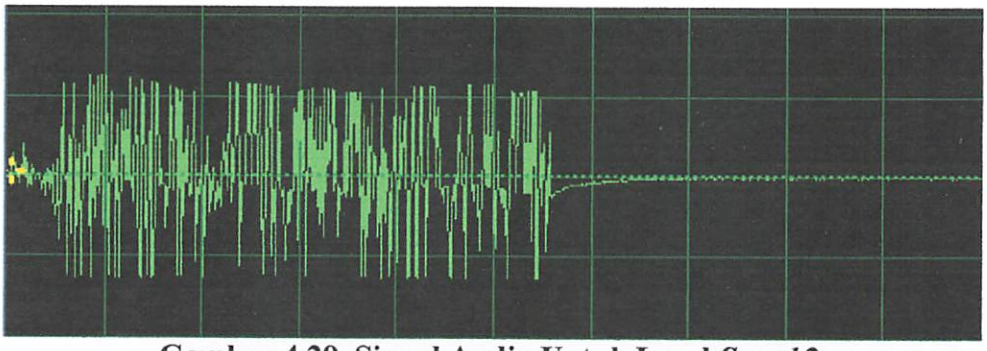
No	Tombol	Keterangan
1	(+)	Level *speed* naik
2	(-)	Level *speed* turun
3	Enter	Kembali Ke Menu Utama
4	Selain (-), (+), dan Enter	*Buzzer* Aktif

Perubahan level *speed* dapat diamati dengan melihat sinyal audio yang dihasilkan oleh modul SP03. Pada pengujian level *speed* ini dilakukan dengan

memasukkan teks "How are you" dan mengamati perubahan sinyal audio pada oscilloscope apabila level *speed* di ubah.



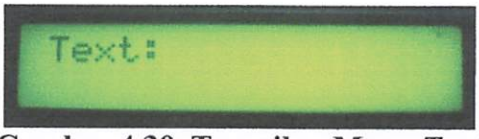
Gambar 4.28. Sinyal Audio Untuk Level *Speed* 1



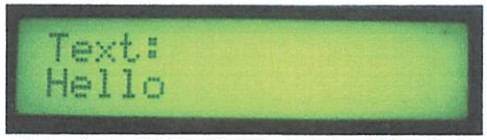
Gambar 4.29. Sinyal Audio Untuk Level *Speed* 2

4.6.3.4 Pengaturan Teks Yang Akan Dikonversikan

Program akan masuk ke menu "Text" apabila terjadi penekanan tombol 4(empat). Menu ini berfungsi untuk menentukan teks yang akan dikonversikan menjadi ucapan. Tombol-tombol yang digunakan pada menu ini adalah tombol alfabet dan numerik. Jika terjadi penekanan tombol selain tombol-tombol tersebut, maka buzzer akan berbunyi. Maksimum teks yang diijinkan pada menu ini adalah sebanyak 24 karakter. Apabila melebihi, maka *buzzer* akan berbunyi.



Gambar 4.30. Tampilan Menu *Text*



Gambar 4.31. Input Teks "Hello"

Program akan melakukan proses konversi ke ucapan apabila terjadi penekanan tombol "enter". Setelah melakukan proses konversi, maka program akan kembali ke menu utama.

Gambar 4.32. Proses Konversi Teks

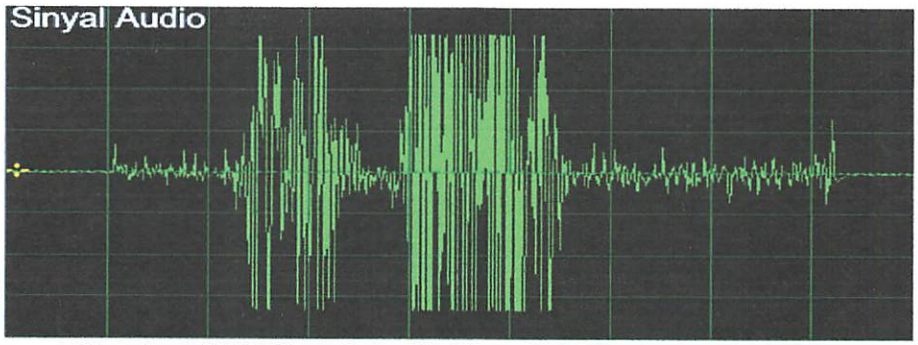
Tabel 4.9. Pengaturan Menu *Text*

No	Tombol	Keterangan
1	Alfabet dan Numerik	Input Karakter
2	Enter	Konversi teks ke ucapan
3	Selain alfabet, numerik , dan Enter	*Buzzer* Aktif

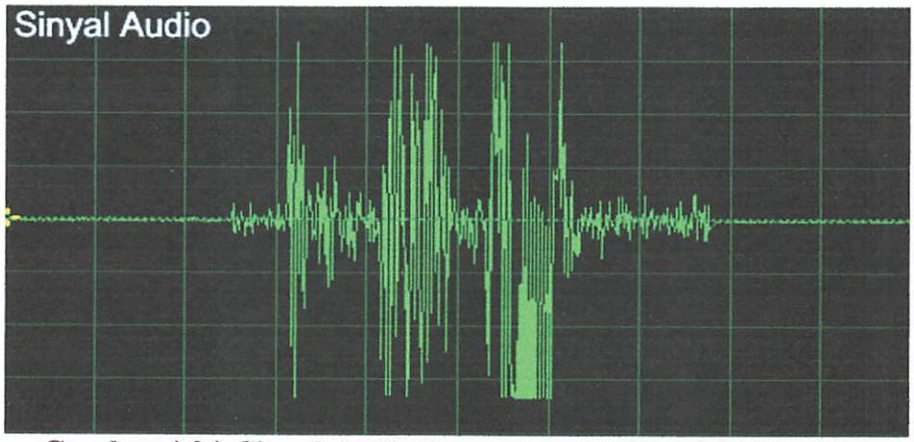
4.6.4 Pengujian *Output* Sinyal Audio Dari Modul Modul SP03

Pengujian untuk output sinyal audio dilakukan dengan memasukkan satu kata, dua kata, tiga kata, dan beberapa kata yang hampir mirip kemudian mengamati hasil konversi yang berupa sinyal audio pada *osscilloscope.*

4.6.4.1 Pengujian Untuk Inputan Satu Kata



Gambar 4.33. Sinyal Audio Untuk Pengucapan "Apple"

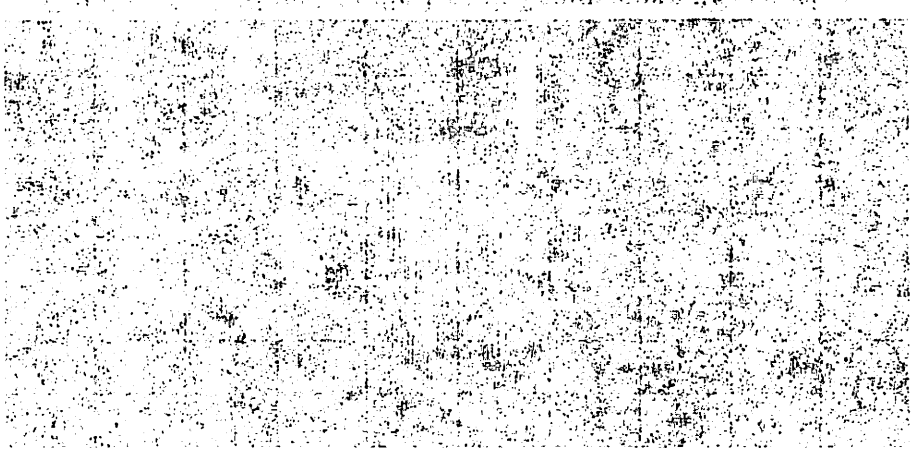


Gambar 4.34. Sinyal Audio Untuk Pengucapan "Expansive"

Berikut ini adalah data hasil pengujian dengan memberikan inputan satu buah kata:

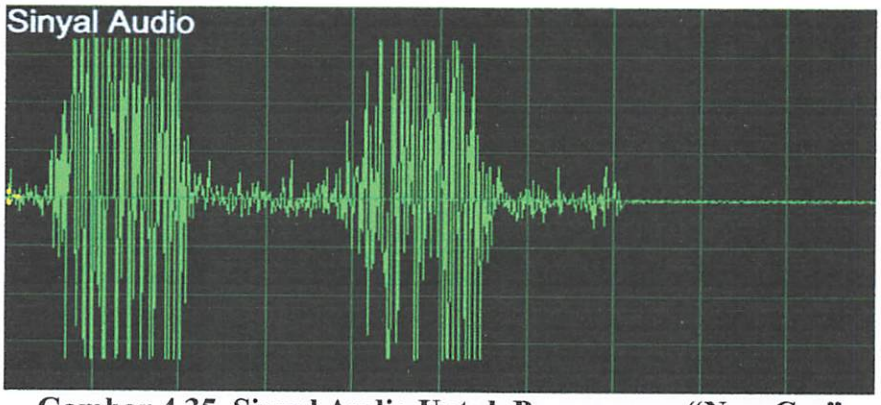
Tabel 4.10. Hasil Pengujian Untuk Inputan Satu Kata

No.	Kata	Pelafalan	
		Sesuai	Tidak Sesuai
1	One	√	
2	Two	√	
3	Apple	√	
4	Architect	√	
5	Area	√	
6	Babe	√	
7	Baby	√	
8	Backside	√	
9	Benefit		√
10	Bless	√	
11	expansive	√	
12	Fine	√	
13	Magnetic		√

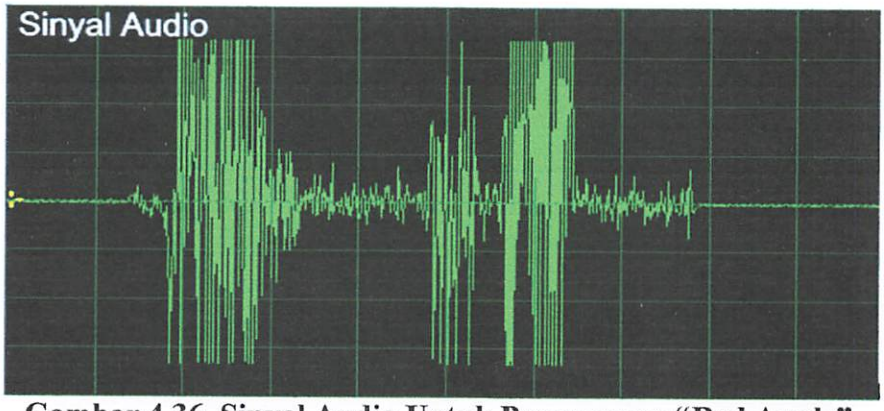


14	Magnesium		√
15	Push	√	
16	Table	√	
17	unpredictable	√	
18	value	√	
19	Economic	√	
20	Wild	√	

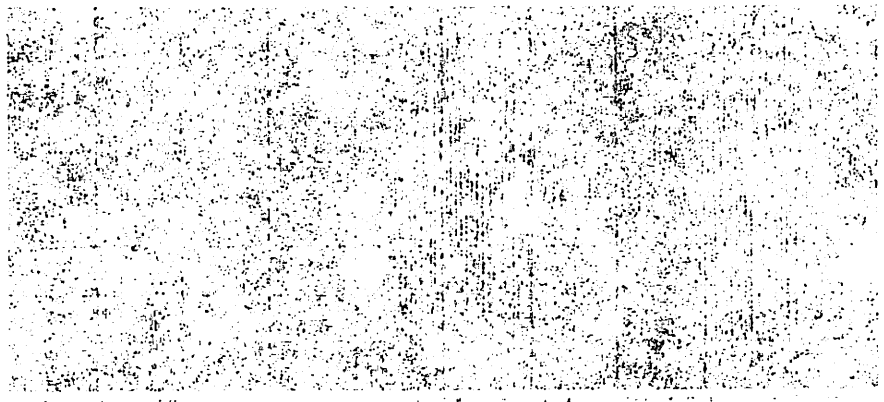
4.6.4.2 Pengujian Untuk Inputan Dua Kata

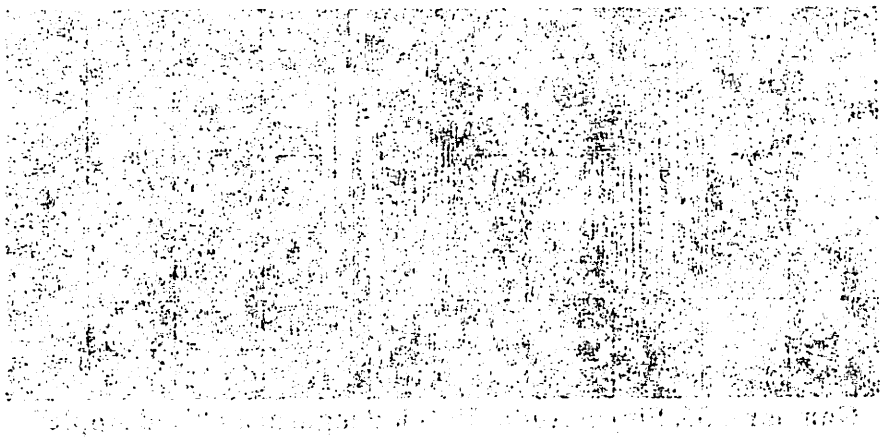


Gambar 4.35. Sinyal Audio Untuk Pengucapan "New Car"



Gambar 4.36. Sinyal Audio Untuk Pengucapan "Red Apple"



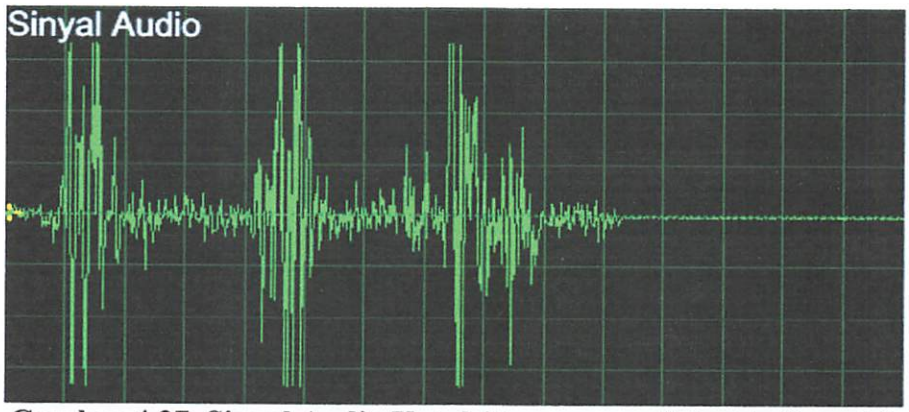


Berikut ini adalah data hasil pengujian dengan memberikan inputan dua buah kata:

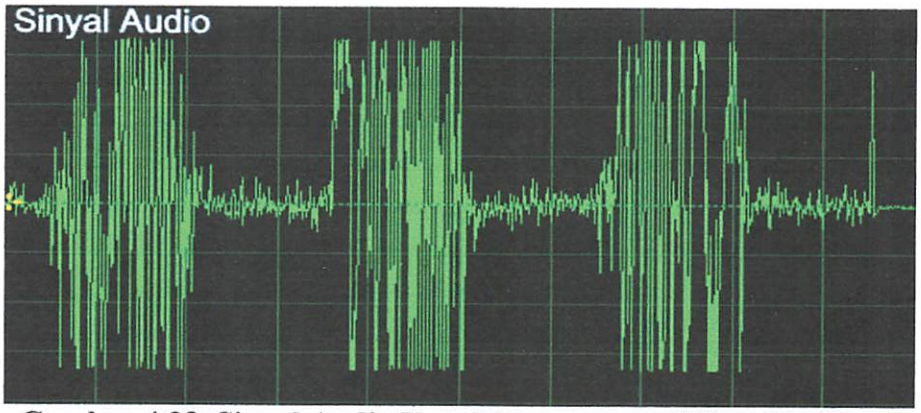
Tabel 4.11. Hasil Pengujian Untuk Inputan Dua Kata

No.	Kata	Pelafalan	
		Sesuai	Tidak Sesuai
1	Red Apple	√	
2	New Car	√	
3	Hand Phone	√	
4	Handphone		√
5	Electrical Engineering	√	
6	Text Convertion	√	
7	class room	√	
8	classroom	√	

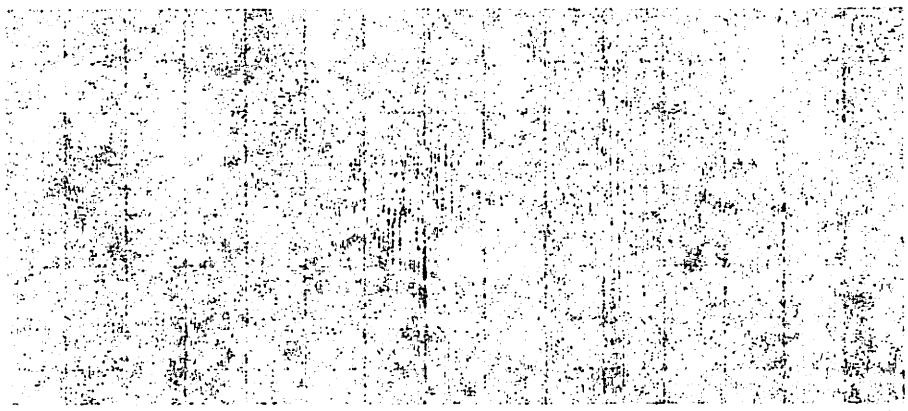
4.6.4.3 Pengujian Untuk Inputan Tiga Kata

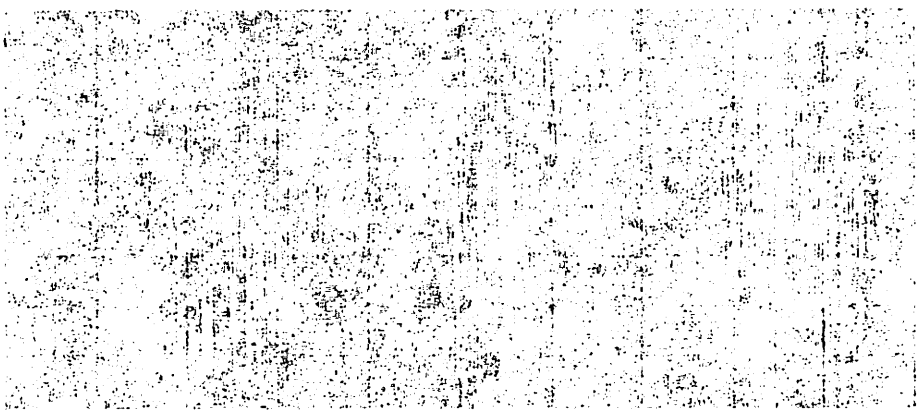


Gambar 4.37. Sinyal Audio Untuk Pengucapan "Text to Speech"



Gambar 4.38. Sinyal Audio Untuk Pengucapan "How Are You"



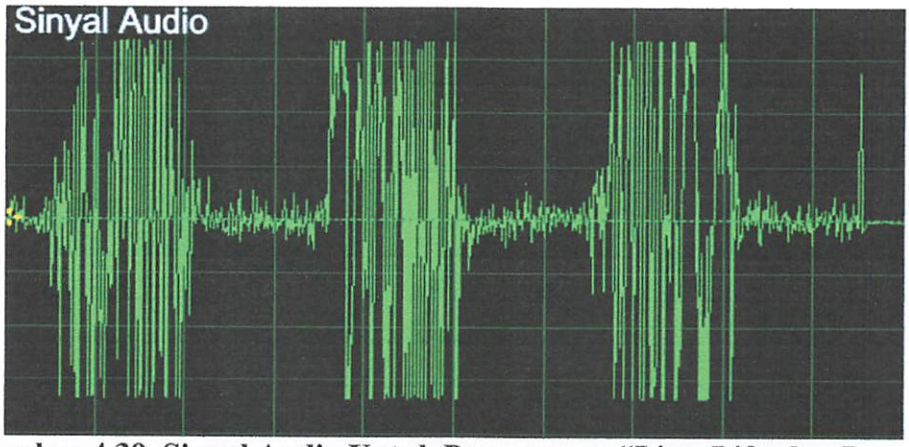


Berikut ini adalah data hasil pengujian dengan memberikan inputan tiga buah kata.

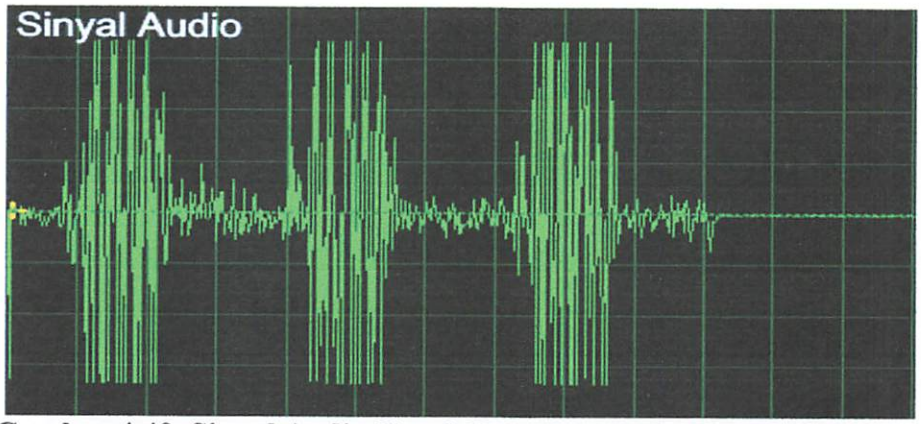
Tabel 4.12. Hasil Pengujian Untuk Inputan Tiga Kata

No.	Kata	Pelafalan	
		Sesuai	Tidak Sesuai
1	Fresh red apple	√	
2	New expensive car	√	
3	Junior high school	√	
4	Text to speech	√	
5	How are you	√	
6	I am Fine	√	

4.6.4.4 Pengujian Untuk Inputan Kata-Kata Yang Hampir Sama



Gambar 4.39. Sinyal Audio Untuk Pengucapan "Live, Life, dan Leave"



Gambar 4.40. Sinyal Audio Untuk Pengucapan "Tu, Two, dan Too"

Berikut ini adalah data hasil pengujian dengan memberikan inputan beberapa kata yang hampir sama.

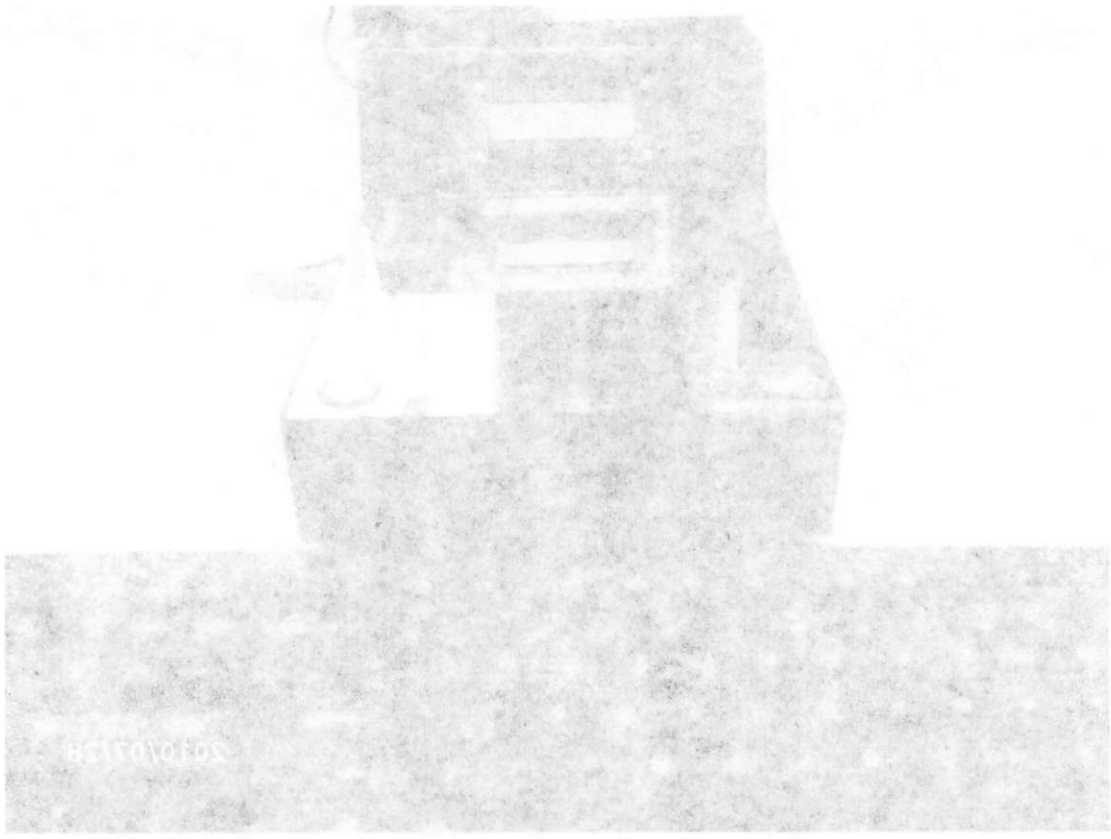
Tabel 4.13. Hasil Pengujian Untuk Inputan Kata-Kata Yang Hampir Sama

No.	Kata	Pelafalan	
		Sama	Tidak Sama
1	With, Width, Wide		√
2	Live, Life,Leave		√
3	Three, tree	√	
4	To, Two		√
5	For, Four	√	
6	Know, known		√
7	Too, Two, Tu	√	

4.7. Spesifikasi Alat

- Dimensi : 21 x 28 x 12 cm
- Power Supply : AC 220V
- Pengendali Mikrokontroler : ATMEGA 16
- Modul *Text to Speech* : SP03 *Module*
- LCD : M1632 (2 Baris, 16 Kolom)
- Keyboard : AT Keyboard (dengan konektor PS/2)
- Audio Power Ampifier : IC LA4440

Gambar 4.41. Foto Alat



BAB V

PENUTUP

5.1 Kesimpulan

Kesimpulan yang dapat diambil dari perancangan dan pembuatan alat ini adalah sebagai berikut:

1. Tidak semua kata dapat dikonversikan kedalam ucapan dengan pelafalan yang tepat. Kata-kata tersebut antara lain adalah: kata yang berakhiran dengan "...*efit*" (contohnya *benefit*), kata yang diawali dengan "*mag...* " (contohnya *magnetic*), dan kata *finance*.
2. Tidak semua penggabungan dua buah kata menjadi satu kata dapat terkonversikan dengan baik. Contohnya adalah kata "*hand*" dan "*phone*" yang digabungkan menjadi "*handphone*" tidak dapat dilafalkan dengan benar.
3. Terdapat beberapa kata yang penulisannya berbeda namun memiliki pelafalan yang sama. Kata-kata tersebut antara lain adalah *too*, *two*, dan *tu*. Contoh lain adalah "*for*" dan "*four*".
4. Dari hasil pengujian didapatkan nilai arus yang mengalir pada *buzzer* adalah sebesar 7.7mA, ini berarti bahwa *buzzer* dapat dihubungkan langsung ke PortB.0 karena tiap Port pada mikrokontroler dapat dilalului arus sebesar 20mA. Besarnya *error* antara perhitungan dan pengukuran pada rangkaian *buzzer* adalah 1.3%.

5.2 Saran

Pada perancangan dan pembuatan alat ini masih mempunyai kekurangan-kekurangan, untuk itu ada beberapa hal yang dapat dilakukan untuk melakukan pengembangan antara lain adalah:

1. Agar alat ini dapat mengkonversikan teks dengan jumlah karakter yang lebih banyak lagi.
2. Agar alat ini dapat mengkonversikan dengan baik berbagai macam inputan teks termasuk teks dengan awalan "*mag*...", akhiran "...efit", penggabungan dua kata menjadi satu kata, dan dapat mengatasi permasalahan pada pelafalan yang sama pada beberapa kata.
3. Untuk pengembangan lebih lanjut, penulis menyarankan agar selain hanya dapat digunakan untuk mengubah teks ke suara alat juga dirancang untuk menerjemahkan teks Bahasa Inggris yang ditulis ke dalam Bahasa Indonesia.

DAFTAR PUSTAKA

[1]. Konversi Teks Ke Ucapan, Arman Arry Akhmad," Bandung, www.ITB.ac.id

[2]. Interface Keyboard PC PS/2 Dengan Mikrokontroler, www.mytutorial.com

[3]. Wardana Lingga, Belajar Sendiri Mikrokontroler AVR Seri ATMega8535 Simulasi, Hardware, dan Aplikasi, Andi, Yogyakarta, 2006.

[4]. Agfianto Eko Putra, "Belajar Mikrokontroler AT89C51/52/55", Gava media, 2002.

[5]. Parsons. Thomas W. (1986). "***Voice and Speech Processing***", McGraw-Hill, New York.

[6]. Dutoit. Thierry. (1997). "***An Introduction to Text-to-Speech Synthesis***", Kluwer Academic Publisher, Dordrecht.

[7]. Widodo, Thomas. 2002. **Elektronika Dasar**. Salemba Teknika. Jakarta

[8]. www.mcselectronic.com

[9]. www.atmel.com

LAMPIRAN

INSTITUT TEKNOLOGI NASIONAL
Jl. Bendungan Sigura-gura No. 2
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : EKO VERIONO
Nim : 0412253
Masa Bimbingan : 19 Februari 2010 s/d 19 Agustus 2010
Judul Skripsi : Perancangan Dan Pembuatan Sistem Pengkonversi Tulisan Ke Suara Sebagai Sarana Untuk Pembelajaran Pembacaan Kata Dalam Bahasa Inggris

NO	Tanggal	Uraian	Paraf Pembimbing
1	31/5 2010	Revisi Bab II dan III	
2	22/7 2010	Acc Bab I, Bab II, dan Bab III	
3	4/8 2010	Revisi Bab IV dan V	

Malang,
Dosen Pembimbing

Sotyohadi, ST, Msc
NIP.Y. 1039700309

Form S-4b 5/0

PERKUMPULAN PENGELOLAH PENDIDIKAN DAN TEKNOLOGI NASIONAL
MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

PERSERO) MALANG Kampus I: Jl. Bendungan Sigura-gura No.2 Telp (0341)551431 (Hunting) Fax.(0341)553015 Malang

NIAGA MALANG Kampus II:Jl.Raya Karanglo, Km 2 Telp.(0341)417634 Malang

FORMULIR PERBAIKAN SKRIPSI

Dari hasil Komprehensip Jenjang Strata Satu (S-1) Jurusan Teknik Elektro Konsentrasi Teknik Elektronika yang diselenggarakan pada :
Hari : Sabtu
Tanggal : 21 Agustus 2010

Telah dilaksanakan Perbaikan Skripsi oleh Saudara
Nama : Eko Veriono
N.I.M : 04.12.253
Masa Bimbingan : 6 Bulan (19 Pebruari 2010 – 19 Agustus 2010)
Judul Skripsi : "*Perancangan Dan Pembuatan Sistem Pengkonversi Tulisan Ke Suara Sebagai Sarana Untuk Pembelajaran Pembacaan Kata Dalam Bahasa Inggris*"

Tanggal	Dosen Penguji	Uraian	Paraf
21-8-2010	Penguji I	- Pengujian Untuk Volume, Pitch dan Speed -Lampiran: Rangkaian Lengkap dan Listing Program	
21-8-2010	Penguji II		

Mengetahui,

Dosen Pembimbing

Sotyohadi, ST. MSc
NIP.Y. 1039700309

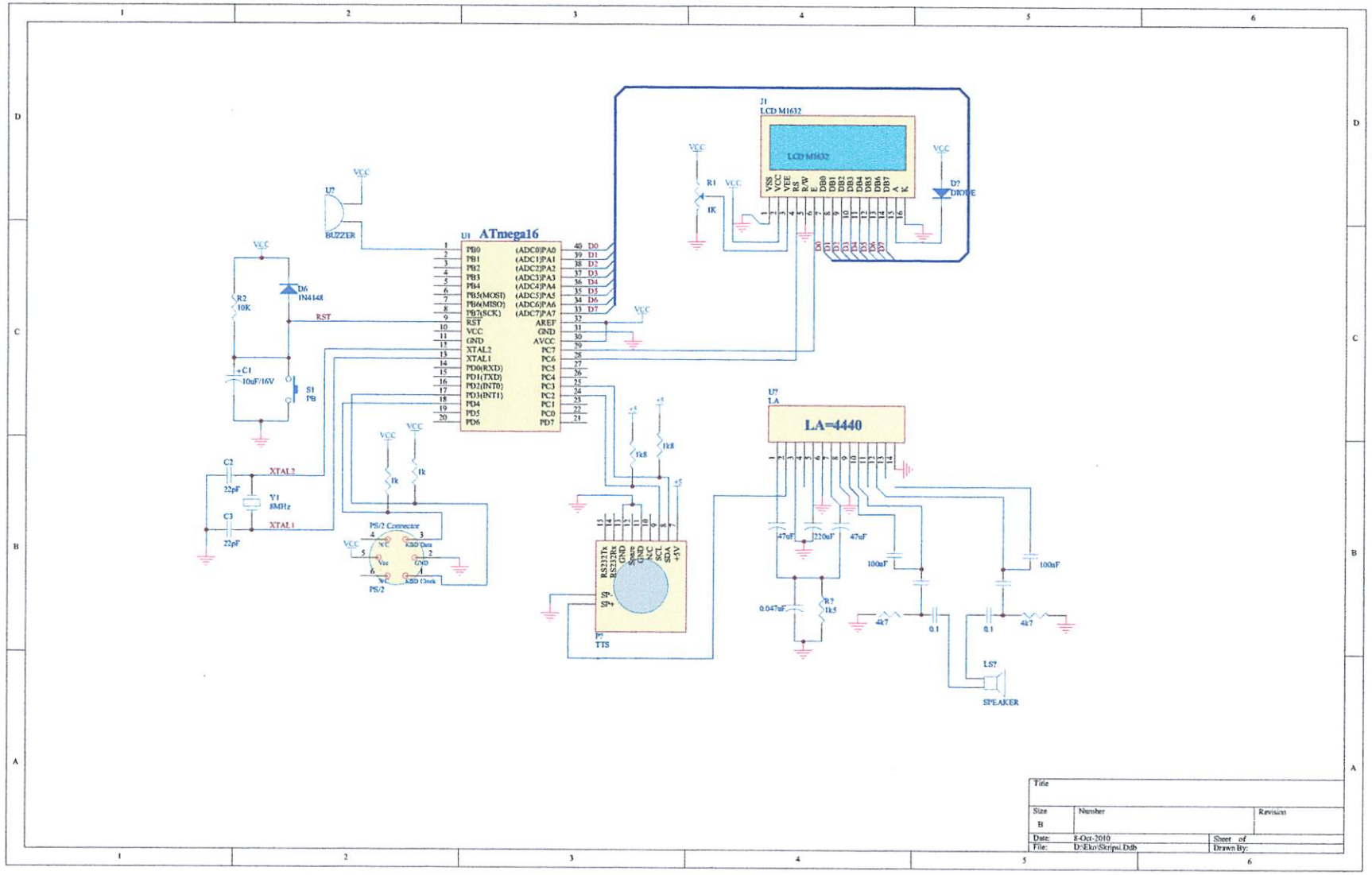
PENGUJI I

Joseph Dedy Irawan, ST, MT
NIP. 197404162005011002

PENGUJI II

Ir. Eko Nurcahyo
NIP.Y 1028700172

1. Gambar Skematik Rangkaian Keseluruhan


J1
LCD M1632
LCD M1632
VSS
VCC
VEE
RS
R/W
E
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
A
K
R1
1K
VCC
D?
DIODE
U?
BUZZER
U1
ATmega16
PB0 (ADC0)PA0
PB1 (ADC1)PA1
PB2 (ADC2)PA2
PB3 (ADC3)PA3
PB4 (ADC4)PA4
PB5(MOSI) (ADC5)PA5
PB6(MISO) (ADC6)PA6
PB7(SCK) (ADC7)PA7
RST AREF
VCC GND
GND AVCC
XTAL2 PC7
XTAL1 PC6
PD0(RXD) PC5
PD1(TXD) PC4
PD2(INT0) PC3
PD3(INT1) PC2
PD4 PC1
PD5 PC0
PD6 PD7
D0
D1
D2
D3
D4
D5
D6
D7
R2
10K
D6
IN4148
RST
+C1
10uF/16V
S1
PB
C2
22pF
XTAL2
Y1
8MHz
C3
22pF
XTAL1
1k
1k8
PS/2 Connector
KBD Data
GND
KBD Clock
PS/2
P?
TTS
RS232Tx
RS232Rx
GND
SCL
SDA
+5V
U?
LA
LA=4440
47nF
220nF
47uF
100nF
0.047uF
R?
1k5
4k7
0.1
LS?
SPEAKER
Title
Size
B
Number
Revision
Date:
8-Oct-2010
Sheet of
File:
D:\Eko\Skripsi.Ddb
Drawn By:

2. Listing Program

```
------------------------------------------------------------------------------

                          Skripsi
                Program Text To Speech Converter
                      Dirancang Oleh:
                    Eko Veriono 0412223
                  Teknik Elektronika ITN Malang

------------------------------------------------------------------------------

----------------------------- Inisialisasi Hardware & Software ---------------

egfile = "m16def.dat"                                          ' Definisi Prosessor
rystal = 11059200                                              ' Crystal frequency
aud = 38400                                                    ' Inisialisasi baud rate
wstack = 32
wstack = 8
ramesize = 24
arge

------------------------------------------------------------------------------




----------------------------- Inisialisasi LCD -------------------------------

nfig Lcdpin = Pin , Db4 = Porta.4 , Db5 = Porta.5 , Db6 = Porta.6 , Db7 = Porta.7
E = Portc.7 , Rs = Portc.6
nfig Lcd = 16 * 2
rsor Off
splay On
s

------------------------------------------------------------------------------


--------------------------- Inisialisasi Keyboard ----------------------------

nfig Keyboard = Pind.7 , Data = Pind.6 , Keydata = Keydata
------------------------------------------------------------------------------

----------------------------- Inisialisasi I2C -------------------------------

nfig Scl = Portc.1
nfig Sda = Portc.0
------------------------------------------------------------------------------'
nfig Portb = Output
nfig Portb.0 = Output
nfig Portb.1 = Output

------------------------------------------------------------------------------

----------------------------- Deklarasi Variable -----------------------------
m Kbrd As Byte , Kbrd_str As String * 2
m Txt(16) As String * 1
m Hasil_text As String * 16 , Cnt As Byte , Count As Byte
m Sp03 As Byte , Cmd_reg As Byte , Ver_reg As Byte
m Ldbuf As Byte , Spkbuf As Byte
m Vol As Byte , Pitch As Byte , Speed As Byte , Null As Byte
m Lcd_vol As Byte , Lcd_pitch As Byte
```

```
m Value As Byte , Status As Byte
m N As Byte , M As Byte
m Kbrd_test As Byte



------------------------------------------------------------------------------

---------------------------- Deklarasi Sub Program ---------------------------

eclare Sub Volume
eclare Sub Pitch
eclare Sub Speed
eclare Sub Text

------------------------------------------------------------------------------

---------------------------- Inisialisasi Variable ---------------------------

03 = &HC4                                                   'SP03 I2c Addres
d_reg = &H00                                                'SP03 Command Register
r_reg = &H01                                                'SP03 Revision Register
buf = &H00                                                  'NOP Command is Load
ffer
kbuf = &H40                                                 'Speak Buffer Command
ll = &H00                                                   'Null
l = &H03                                                    'variabel Volume
tch = &H07                                                  'variabel Pitch
eed = &H00
t = 0
unt = 0
t(16) = ""
d_vol = &H04
d_pitch = &H00
= 0
= 0



------------------------------------------------------------------------------
t Portb.0
t Portb.1

---------------------------- Program Utama -----------------------------------

s
d "  Eko Veriono"
werline
d "   04.12.253"

it 2

s
d "  Daftar Menu"
it 1


al:


s
```

```
d "1.Volume 2.Pitch"
werline
d "3.Speed  4.Text"


rd = Getatkbd()
 Kbrd <> 1 Then
 Kbrd_str = Chr(kbrd)

    If Kbrd_str = "1" Then
    Goto Volume


    Elseif Kbrd_str = "2" Then
    Goto Pitch


    Elseif Kbrd_str = "3" Then
    Goto Speed


    Elseif Kbrd_str = "4" Then
    Goto Text

    Else
    Gosub Buzzer
    End If

d If

 Kbrd = 1 Then
 Gosub Buzzer
d If

op


--------------------------------------------------------------------------------


---------------------------- Sub Program Volume ----------------------------

lume:
itms 200
s
d "Volume" ; " " ; Lcd_vol ; "/7"


brd = Getatkbd()

 If Kbrd <> 1 Then
 Kbrd_str = Chr(kbrd)


    If Kbrd_str = "-" Then
      Incr Vol
      Decr Lcd_vol
         If Vol > 6 Then
         Vol = 7
         Lcd_vol = 0
```

```
      End If
    Cls
    Lcd "Volume" ; " " ; Lcd_vol ; "/7"


  Elseif Kbrd_str = "+" Then
    Decr Vol
    Incr Lcd_vol
       If Vol > 254 Then
       Vol = 0
       Lcd_vol = 7
       End If
    Cls
    Lcd "Volume" ; " " ; Lcd_vol ; "/7"


  Elseif Kbrd = Chr(13) Then
     Waitms 200
     Goto Awal


  Elseif Kbrd = 0 Then
     Goto Terus


  Else
     Gosub Buzzer
  End If

End If

If Kbrd = 1 Then
   Gosub Buzzer
End If

rus:
op

------------------------------------------------------------------------------



------------------------------------------------------------------------------

---------------- Sub Program Pitch Speech ---------------------------------

tch:

s
d "Pitch" ; " " ; Lcd_pitch ; "/7"


rd = Getatkbd()

If Kbrd <> 1 Then
Kbrd_str = Chr(kbrd)
```

```
    If Kbrd_str = "-" Then

      Incr Pitch
      Decr Lcd_pitch

         If Pitch > 6 Then
          Pitch = 7
          Lcd_pitch = 0
         End If

      Cls
      Lcd "Pitch" ; " " ; Lcd_pitch ; "/7"



    Elseif Kbrd_str = "+" Then

      Decr Pitch
      Incr Lcd_pitch

         If Pitch > 254 Then
         Pitch = 0
         Lcd_pitch = 7
         End If

      Cls
      Lcd "Pitch" ; " " ; Lcd_pitch ; "/7"



    Elseif Kbrd = Chr(13) Then
       Waitms 200
       Goto Awal

    Elseif Kbrd = 0 Then
       Goto Terus2


    Else
       Gosub Buzzer
    End If

 End If

 If Kbrd = 1 Then
    Gosub Buzzer
 End If

rus2:
op



------------------------------------------------------------------------------


------------------------- Sub Program Speed Speed ----------------------------

eed:

s
```

```
:d "Speed" ; " " ; Speed ; "/3"

)
)rd = Getatkbd()

 If Kbrd <> 0 Then
 Kbrd_str = Chr(kbrd)


    If Kbrd_str = "+" Then
    Incr Speed
       If Speed > 2 Then
       Speed = 3
       End If
    Cls
    Lcd "Speed" ; " " ; Speed ; "/3"


    Elseif Kbrd_str = "-" Then
    Decr Speed

       If Speed > 254 Then
       Speed = 0
       End If
    Cls
    Lcd "Speed" ; " " ; Speed ; "/3"


    Elseif Kbrd = Chr(13) Then
       Waitms 200
       Goto Awal

    Elseif Kbrd = 0 Then
       Goto Terus3

    Else
       Gosub Buzzer
    End If

 End If

 If Kbrd = 1 Then
    Gosub Buzzer
 End If

rus3:
op



------------------------------------------------------------------------------


------------------------ Sub Program Edit Text -----------------------------

xt:

sil_text = ""
t = 0
unt = 0
```

```
ls
:d "Text" ; ":"
)cate 2 , 1
ırsor Blink
)


)rd = Getatkbd()


 If Kbrd <> 1 Then

    If Kbrd = 0 Then
       Goto Terus4
    End If


    If Kbrd = Chr(13) Or Kbrd = Chr(&H7f) Then

       Count = Cnt

       If Cnt > 15 Then
          Do
             Decr N
             Shiftlcd Right
             Waitms 20
          Loop Until N = 0
       End If
       Cursor Noblink
       Goto Konversi
    End If

   If Kbrd = 8 Then
      Cursor Blink
      Locate 2 , Cnt
      Lcd " "
      Locate 2 , Cnt
      Decr Cnt
      If Cnt = 255 Then
         Cnt = 0
      End If
      If Cnt > 14 Then
         Shiftlcd Right
         Decr N
      End If
      Goto Terus4
   End If

   If Cnt = 23 Then
       Cursor Noblink
   End If

   If Cnt = 24 Then
       Gosub Buzzer
       Goto Terus4
   End If


   Incr Cnt

   Kbrd_str = Chr(kbrd)
```

```
    Txt(cnt) = Kbrd_str

    Locate 2 , Cnt
    Lcd Txt(cnt)
    If Cnt > 15 Then
        Shiftlcd Left
        Incr N
    End If

  End If

  If Kbrd = 1 Then
     Gosub Buzzer
  End If
:rus4:
)op


--------------------------------------------------------------------------------
                              'KONVERSI TEXT KE SUARA"
)nversi:

)t = 1


------------------------------Load Data Ke Buffer-------------------------------
   Reset Portb.0
   I2cstart
   Delay
   I2cwbyte Sp03
   Delay
   I2cwbyte Cmd_reg
   Delay
   I2cwbyte Ldbuf
   Delay
   I2cwbyte Vol
   Delay
   I2cwbyte Pitch
   Delay
   I2cwbyte Speed
   Delay
     Do

     I2cwbyte Txt(cnt)                                          'ASCII Char
     Delay
     Incr Cnt

     If Cnt > Count Then
     Goto Lanjut
     End If

     Loop
   Delay
njut:
   I2cwbyte &H20
   Delay
   I2cwbyte &H20
   Delay
   I2cwbyte &H00
   Delay
   I2cstop
   Delay
```

```
-------------------------------------------------------------------------------
-------------------------Baca Data Dari Buffer---------------------------------
  I2cstart
  Delay
  I2cwbyte Sp03
  Delay
  I2cwbyte Cmd_reg
  Delay
  I2cwbyte Spkbuf
  Delay
  I2cstop
  Gosub Tunggu
  Delay
  Reset Portb.0
  Gosub Tunggu
  Set Portb.0
-------------------------------------------------------------------------------
to Awal
d

nggu:


  I2cstart
  Delay
  I2cwbyte Sp03
  Delay
  I2cwbyte &H00
  I2cstart
  Delay
  I2creceive &HC5 , Value
  Locate 1 , 1
  Lcd "Speaking..."
  Delay
  I2cstop
  Shiftcursor Right
op Until Value = 0
lay
turn

-------------------------------------------------------------------------------
zzer:
set Portb.0
set Portb.1
itms 500
t Portb.0
t Portb.1
turn


----------------------- Tabel Data Keyboard ---------------------------------

ydata:


ɔrmal Keys Lower Case

ta 1 , 1 , 1 , 1 , 1 , 200 , 1 , 1 , 1 , 1

ta 1 , 1 , 1 , 1 , &H5E , 1 , 1 , 1 , 1 , 1
```

ıta 1 , 113 , 49 , 1 , 1 , 1 , 122 , 115 , 97 , 119

ıta 50 , 1 , 1 , 99 , 120 , 100 , 101 , 52 , 51 , 1

ıta 1 , 32 , 118 , 102 , 116 , 114 , 53 , 1 , 1 , 110

ıta 98 , 104 , 103 , 121 , 54 , 7 , 8 , 44 , 109 , 106

ıta 117 , 55 , 56 , 1 , 1 , 44 , 107 , 105 , 111 , 48

ıta 57 , 0 , 0 , 46 , 45 , 108 , 48 , 112 , 43 , 0

ıta 1 , 1 , 1 , 1 , 1 , 92 , 1 , 1 , 1 , 1

ıta 13 , 1 , 1 , 92 , 1 , 1 , 1 , 60 , 1 , 1

ıta 1 , 1 , 8 , 1 , 1 , 49 , 1 , 52 , 55 , 1

.ta 0 , 0 , 48 , 44 , 50 , 53 , 54 , 56 , 0 , 0

.ta 0 , 43 , 51 , 45 , 42 , 57 , 0 , 0

hifted Keys Upper Case

.ta 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1

.ta 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1

ta 1 , 81 , 33 , 1 , 1 , 1 , 90 , 83 , 65 , 87

ta 34 , 1 , 1 , 67 , 88 , 68 , 69 , 1 , 35 , 1

ta 1 , 32 , 86 , 70 , 84 , 82 , 37 , 1 , 1 , 78

ta 66 , 72 , 71 , 89 , 38 , 1 , 1 , 76 , 77 , 74

ta 85 , 47 , 40 , 1 , 1 , 59 , 75 , 73 , 79 , 61

ta 41 , 1 , 1 , 58 , 95 , 76 , 48 , 80 , 63 , 1

ta 1 , 1 , 1 , 1 , 1 , 96 , 1 , 1 , 1 , 1

ta 13 , 94 , 1 , 42 , 1 , 1 , 1 , 62 , 1 , 1

ta 1 , 1 , 8 , 1 , 49 , 1 , 52 , 55 , 1 , 1

ta 1 , 1 , 48 , 44 , 50 , 53 , 54 , 56 , 1 , 1

ta 1 , 43 , 51 , 45 , 42 , 57 , 1 , 1

---

3. Manual Book Modul SP03

SP03 Text to Speech Synthesizer

The robotics community has been without a low cost speech synthesizer chip for a long time. The ever popular SP0256-AL2 has long gone out of production, though there are a few still available. There are other multi-chip products around but none have captured the popular imagination like the Winbond WTS701. And with good reason, the WTS701 is not only a complete single chip synthesizer but also includes a text to speech processor. Given the near impossibility of producing a definitive set of rules for text to speech processing, the WTS701 performs impressively. The only downer for the hobbyist is that the WTS701 comes in a 56 lead TSOP package with pins on a 0.5mm pitch.

The SP03 module includes an audio amplifier, a 3volt regulator and level conversion to 5volts, a PIC processor to provide easy communication with your host processor and even a small 40mm speaker, along with the WTS701. Interfaces include an RS232 serial interface, I2C bus interface and a parallel interface to speak up to 30 predefined phrases. A PC program SP03.EXE is available to load the 30 predefined phrases into the SP03.

Connections to the Speech Module

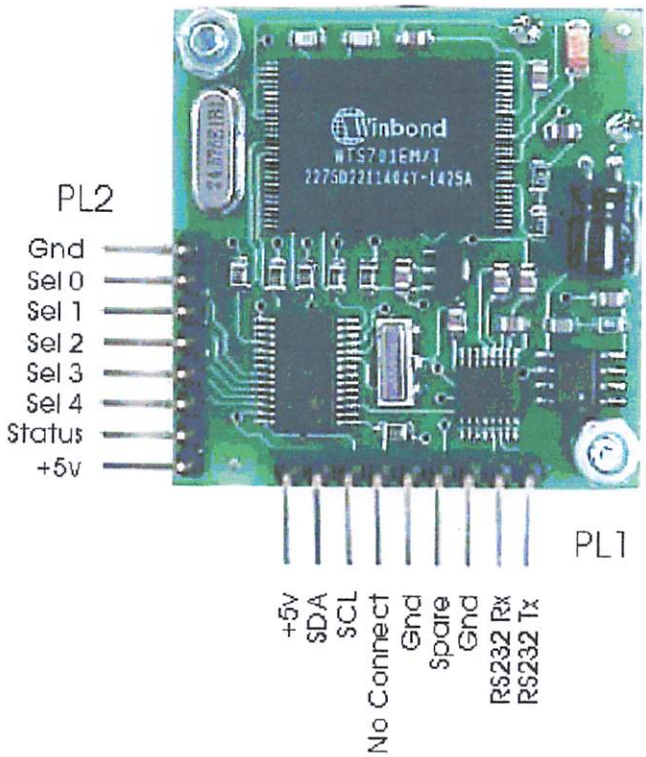
There are two connectors on the SP03 module. The 5v power supply may be applied to either of them - the other may be left unconnected.

PL1

+5V - 5V Power Supply - up to 100mA
SDA - IC2 bus SDA connection
SCL - I2C bus SCL connection
No Connect - Do not connect this pin
GND - The 0 volt Ground line
Spare - Undefined pin - do not connect
GND - The 0 volt Ground line
RS232 Rx - Connect to Tx on the PC
RS232 Tx - Connect to Rx on the PC

PL2

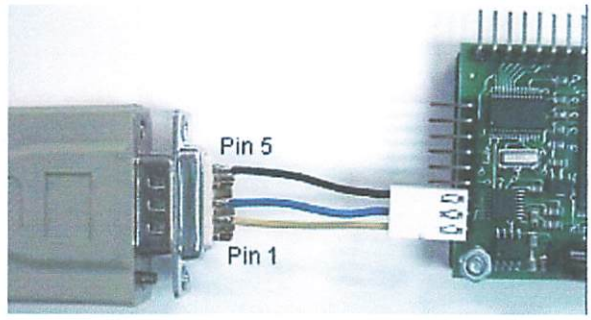
+5V - 5V Power Supply - up to 100mA
Status - High when speaking, Low when done
Sel 4 -
Sel 3 - These are the binary select
Sel 2 - inputs. They select one of the
Sel 1 - 30 predefined phrases
Sel 0 -
GND - The 0 volt Ground line



RS232 Communications

To use the RS232 serial port to control the SP03, just three connections to the PC (or other host) and a 5v power supply is required - Ground, Rx, Tx and 5Volts. The modules RS232 Rx pin should be connected to the PC's Tx line (pin 3 on the DB9 socket). The SP03 RS232 Tx pin should be connect to the PC's Rx line (pin 2 on the DB9 socket). The SP03 Ground line is connected to pin 5 on the DB9

ocket. The 5volt supply is not shown in the photo below.



'he serial data format is 8bits, No Parity, 2 stop bits, 38400 baud.

RS232 Commands

'here are 32 serial commands that can be sent to the SP03. Thirty of these (commands 1 to 30, or 0x01 o 0x1E) are used to speak one of the thirty predefined phrases. To speak any of these phrases, just send single command byte to the SP03. When the SP03 has finished speaking it will send the command ack to the PC as an acknowledgement. Don't send any commands to the module whilst it is speaking, s they will be ignored. When the command is acknowledged, the module will be ready to receive nother command.

'ommand 128 (0x80)

'his command is used to speak a line of text. It is followed by 3 control bytes, then the ASCII text, then n 0x1a character and finally a zero (0x00 or NULL). The three control bytes are the volume, pitch and peed values for the text. For every byte sent, here will be an acknowledge byte sent back from the nodule. It is essential that you wait for the acknowledge byte before sending the next character. Here is ne sequence to speak the word "Hello" at full volume and moderate pitch and speed. Note that the olume value 0-7 is from loudest to quietest. The PIC's text buffer is 85 bytes in size, so that is the limit or a single phrase. The volume, pitch, speed and trailing NULL character take 4 bytes leaving 81 for the ext.

Command byte Transmitted to SP03 Module	Acknowledge byte returned from SP03
Command 0x80	0x01
Full Volume 0x00	0x00
Speech Pitch 0x04	0x04
Speech Speed 0x02	0x02
Text 'H'	'H'
Text 'e'	'e'
Text 'l'	'l'
Text 'l'	'l'
Text 'o'	'o'
NULL 0x00	0x00
SP03 will now speak the text	0x00 indicates text loading is complete

ommand 129 (0x81)

his command is used to read back the WTS701's status register. After sending 0x81 to the module it ill respond by sending back first the low byte and then the high byte of the WTS701 status register. his may be used to determine when the WTS701 has actually finished speaking the text. You should

onsult the WTS701 data sheet for full details of the status bits, however a simple way to do this in C is;

```
.o {
  SerOut(0x81);              // send Get Status command
  sts = SerIn();             // get low byte
  sts += SerIn()<<8;         // get high byte
  sts &= 0x8003;             // select bits to test
while(sts != 0x8001);        // and loop until finished speaking
```

Command 130 (0x82)
This command is used by SP03.EXE to download the 30 predefined phrases into the PIC's Flash memory. Do not try to use it as the protocol requires that the text is sent in a compressed format. Use the onfiguration utility SP03.EXE instead.

All other values sent as commands will be ignored and no acknowledgement will be returned.

2C Bus Communications
Along with the 5Volt power supply, the I2C bus just requires the SDA and SCL lines. The I2C interface oes not have any pull-up resistors on the board, these should be provided elsewhere, most probably /ith the bus master. They are required on both the SCL and SDA lines, but only once for the whole bus, ot on each module. I suggest a values of 4k7 for 100KHz and 1k8 if you are going to be working up to 00KHz or higher. If your going for higher speeds than 400KHz then you must separate ***all*** bytes ansmitted over the I2C bus to the SP03 with a 40uS delay. This is to give the processor time to transfer ie incoming data to the buffer. By doing this we have tested the SP03 with SCL up to 1MHz.

2C communication protocol with the speech module is the same as popular eeprom's such as the 24C04. he SP03 only has two registers, the command register and the software revision number. To read the ftware revision number, first send a start bit, the module address (0XC4) with the read/write bit low, ien the register number you wish to read (0x01). This is followed by a repeated start and the module ldress again with the read/write bit high (0XC5). You now read one byte which is the PIC software vision number and follow this with the stop bit.

Register	Function
0	Command Register
1	Software Revision Number

ll commands and text to be spoken are sent to the command register. There are 32 valid commands, sted below:

Command	Action
NOP (0x00)	No Action
SPKPRE 1 to 30, or 0x01 to 0x1E)	Speak pre-defined phrase
SPKBUF 64 or 0x40	Speak text previously stored in buffer

he NOP command is followed by the text that you want spoken. You may send as little or as much (up the 85 byte limit) as you wish. A number of NOP sequences may be used to build up the buffer before e SPKBUF command is issued. The buffer is flushed after a SPKPRE or SPKBUF command is used. text sequence is the same as for the RS232 protocol, that is, 3 control bytes, then the ASCII text, and

nally a zero (0x00 or NULL). The text in the buffer may then be spoken by sending a SPKBUF ommand. The PIC's text buffer is 85 bytes in size, so that is the limit for a single phrase. The volume, itch, speed and trailing NULL characters take 4 bytes leaving 81 for the text.
'o say "Hello" send the following sequence over the I2C bus:

Start Bit	I2C Start Sequence
0xC4	SP03 I2C Address
0x00	SP03 Command Register
0x00	SP03 NOP Command
0x00	Volume (Max.)
0x05	Speech Speed
0x03	Speech Pitch
'H' (0x48)	Text
'e' (0x65)	Text
'l' (0x6C)	Text
'l' (0x6C)	Text
'o' (0x6F)	Text
0x00	NULL
Stop Bit	I2C Stop Sequence

'ote - there is no command to "speak the following text", as the RS232 interface does. If you wish to)eak a single phrase then you should send a NOP command followed by the text sequence. Then, in a :parate I2C bus transaction, the SPKBUF command on its own.

o check to see when the SP03 has finished speaking, you can read back the command register. Whilst)eaking, it will be the command that initiated the speaking, either 1-30 (0x01-0x1E) or 64 (0x40). It ill be cleared to zero (0x00) when speaking is complete and the module is ready for the next phrase.

---

arallel Port Communications

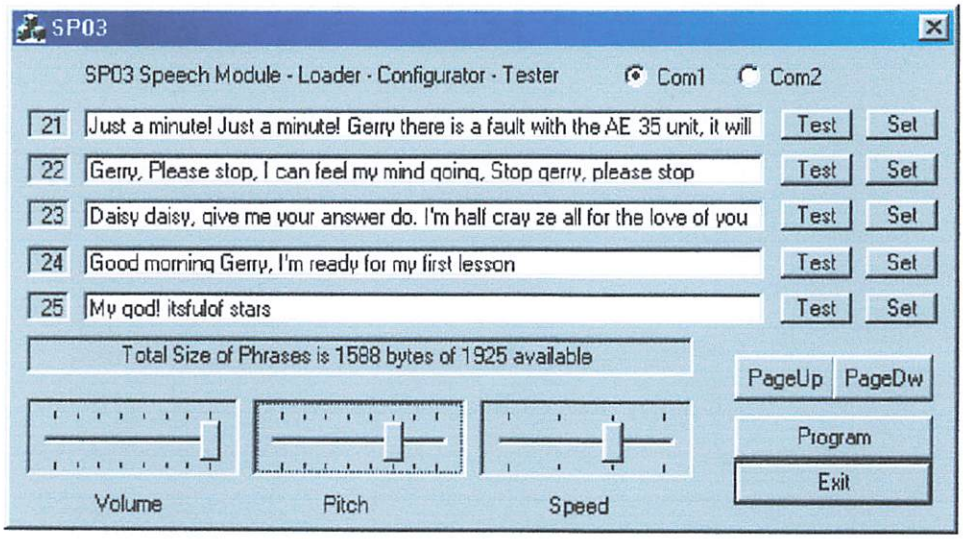
L2 is used to speak one of the 30 pre-defined phrases. Text cannot be sent to the parallel port for)eaking. To speak any of the 30 (1-30) pre-defined phrases, apply the phrase number to the 5 bit input)rt SEL4 - SEL0. The numbers 0 and 31 (0x00 and 0x1F) are just parking values and to not cause any ırase to be spoken. The SP03 has pull-up resistors on the inputs so they can be left unconnected if not :ed. As soon as the CPU has recognised and confirmed the new input (a few uS at most) it will raise e STATUS bit to a logic 1 (high) and speak the phrase. As soon as the STATUS bit goes high, the put code may be removed by returning the input to 0x00 or 0x1F. This must happen before the unit ıs finished speaking or the phrase will be repeated. The STATUS bit will go to a logic 0 (low) when e SP03 has finished speaking, and this can be monitored by the host.

)wer Up

n power up, the module will speak phrase #1, if one has been stored by the configuration program ?03.EXE

onfiguration program SP03.EXE

The SP03 configuration program is shown below .



When you run the program for the first time, you should select the communications port you will be sing, either COM1 or COM2. This will be remembered for the next time you run the program.
he SP03 configuration program can store 30 phrases in 6 pages of 5 phrases each. Press the PageUp nd PageDw buttons to change pages. Below the 5 edit boxes for the phrases is a message/status bar and sliders for volume, pitch and speed. The volume and Speed sliders work as expected but the pitch is a ttle strange, try the seventh position! The pitch seems out of sequence with the rest of the positions, but at's the WTS701. Below the PageUp and PageDw buttons is the program button. This will program all 0 phrases into the Flash memory of the PIC16F872 processor on the SP03.

P03.EXE Operation

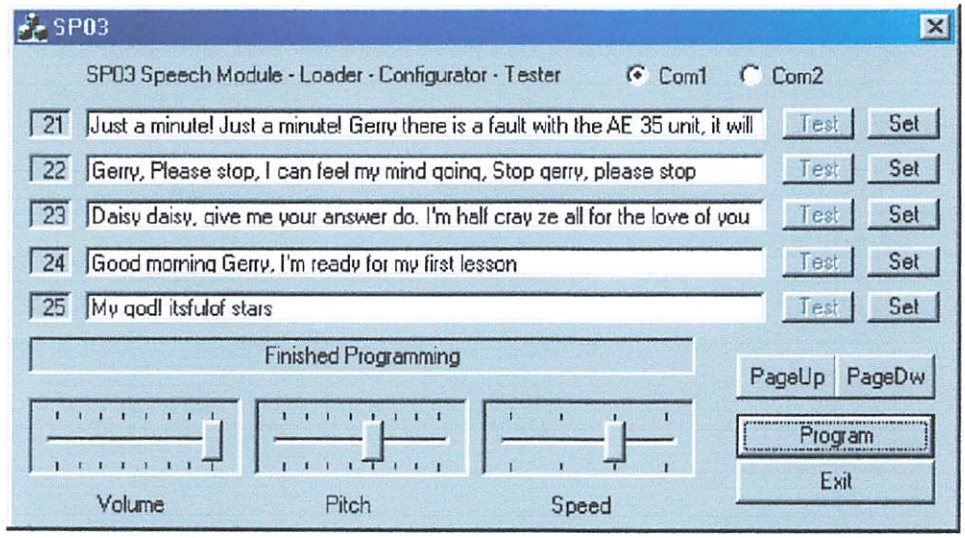
he operation of the program is fairly easy. Start by selecting the Com port and setting the Volume, itch & Speed sliders as shown above. Now type something into one of the edit boxes and press the Test" button to the right of that edit box. The words you typed will be spoken. Notice that the message ar keeps track of the number of characters used so far. This is the total for all 30 phrases.

he "Test" and "Set" buttons

oth of these buttons cause the phrase to be spoken. The "Test" button uses and also stores the Volume, itch & Speed values set on the sliders. The "Set" button uses the Volume, Pitch & Speed values stored om the previous use of the "Test" button. Therefore you can have different Volume, Pitch & Speed ttings for each of the 30 phrases. When you've all your phrases set-up and tested using the "Set" uttons you're ready to program them into the PIC16F872.

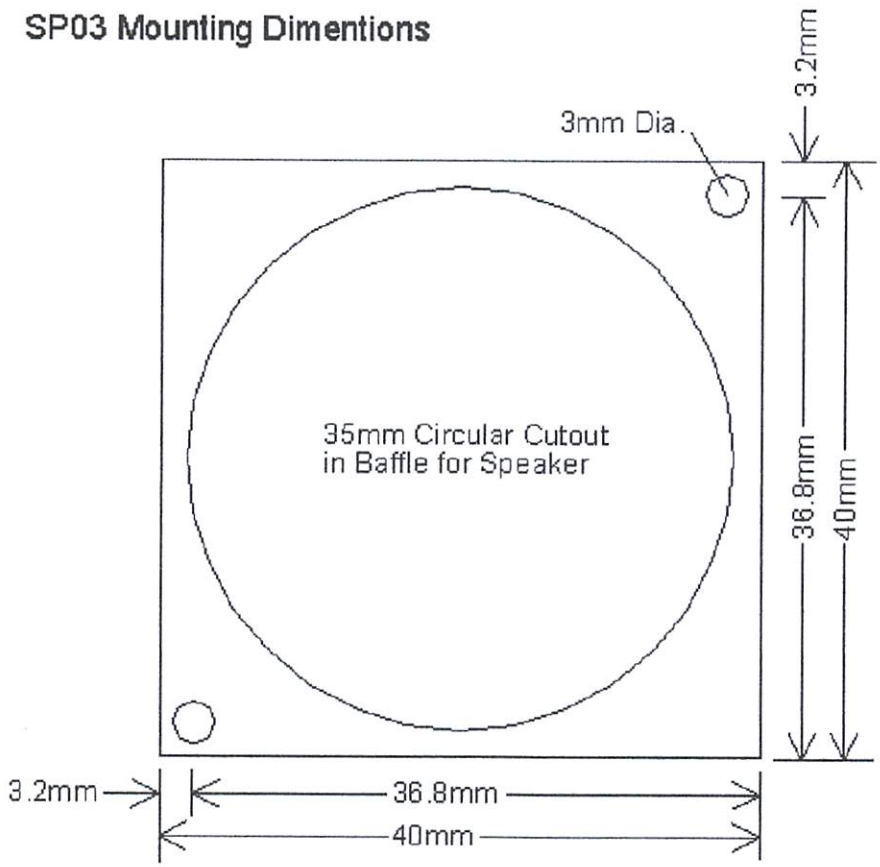
rogramming the Phrases

asy! Just press the "Program" button. Your phrases will be compressed and stored in the Flash memory the PIC16F872 processor. The message bar will report the progress of programming. When rogramming is complete you will see the following screen:



Notice that the "Test" buttons are de-selected. The remaining "Set" buttons have now changed mode and when pressed will cause the phrase to be spoken directly from the pre-defined phrases that you just programmed. Click in any of the edit boxes to restore the "Test" and "Set" buttons to normal.

Baffle Cutout and Dimensions



4. Data Sheet IC WTS701

PRELIMINARY

Winbond Electronics Corp.

WTS701

WINBOND SINGLE-CHIP TEXT-TO-SPEECH PROCESSOR

Publication Release Date May 2003
Revision 3.09

1. GENERAL DESCRIPTION

The WTS701 is a high quality, fully integrated, single-chip Text-to-Speech solution that is ideal for use in applications such as automotive appliances, GPS/navigation systems, cellular phones and other portable products or accessories. The WTS701 product accepts ASCII (Unicode and Big5 for Mandarin) input via a SPI port and converts it to spoken audio via an analog output or digital CODEC output.

The WTS701 integrates a text processor, smoothing filter and multi-level memory storage array on a single-chip. Text-to-speech conversion is achieved by processing the incoming text into a phonetic representation that is then mapped to a corpus of naturally spoken word parts. The synthesis algorithm attempts to use the largest possible word unit in the appropriate context to maximize natural sounding speech quality. The speech units are stored uncompressed in a multi-level, non-volatile analog storage array to provide the highest sound quality to density trade-off. This unique, single-chip solution is made possible through Winbond's patented multilevel storage technology. Voice and audio signals are stored directly into solid-state memory in their natural, uncompressed form, providing superior quality voice reproduction.

The chip can be programmed through the SPI port, allowing downloading of different languages and speaker databases when made available by Winbond.

2. FEATURES

- **Fully Integrated Solution**
  - Single-chip compact text-to-speech translation
  - No algorithm development required
  - Selectable digital and analog audio output
  - Simple SPI interface
  - Reprogrammable solution enables loading different voice or language

- **Text-To-Speech Algorithm Characteristics**
  - High quality speech synthesis using speech element concatenation
  - Winbond's standard 100-year speech retention
  - Audio stored as uncompressed analog waveform – industry's highest quality and most natural sounding

- **Easy to Use and Control**
  - Real time conversion for streaming text
  - General text preprocessing and normalization
  - User customization for special characters such as SMS icons and chat emoticons
  - User customization for application specific abbreviations

- **Language Support**
  - Support U.S. English and Mandarin (Beijing dialect)
  - Other languages in development or in planning

- **Device Management**
  - Accepts ASCII, Unicode or Big5 streaming text
  - 256-byte text buffer
  - Playback of Phonetic Alphabet
  - Variable speed playback
  - Control of pitch change
  - Supports Power Down mode.
  - Supports Pause and Resume, Stop and Finish text conversion commands
  - Embedded characters support to control speed, volume, case sensitivity, and silent behavior

- **Peripheral Control**
  - 16-bit linear PCM slave interface output support
  - SPI serial port for control commands and status report to system's host controller
  - Hardware handshake control signals
  - Analog audio output with 8Ω speaker driver, digital volume control and line level o/p
  - Analog audio input (AUXIN) for driving external audio to the speaker

- **Low Power Consumption**
  - +2.7 to +3.3V (V_{CC}) Supply Voltage
  - Operating Current:
    $I_{CC\ Convert}$ = 35 mA (typical)
  - Standby Current:
    I_{SB} < 1μA (typical)

- **Device Characteristics**
  - Available in 56-lead TSOP package
  - Industrial temperature range (-40C to +85C)
  - 3V/5V logic tolerance

3. BLOCK DIAGRAM

3.1. WTS701 BLOCK DIAGRAM

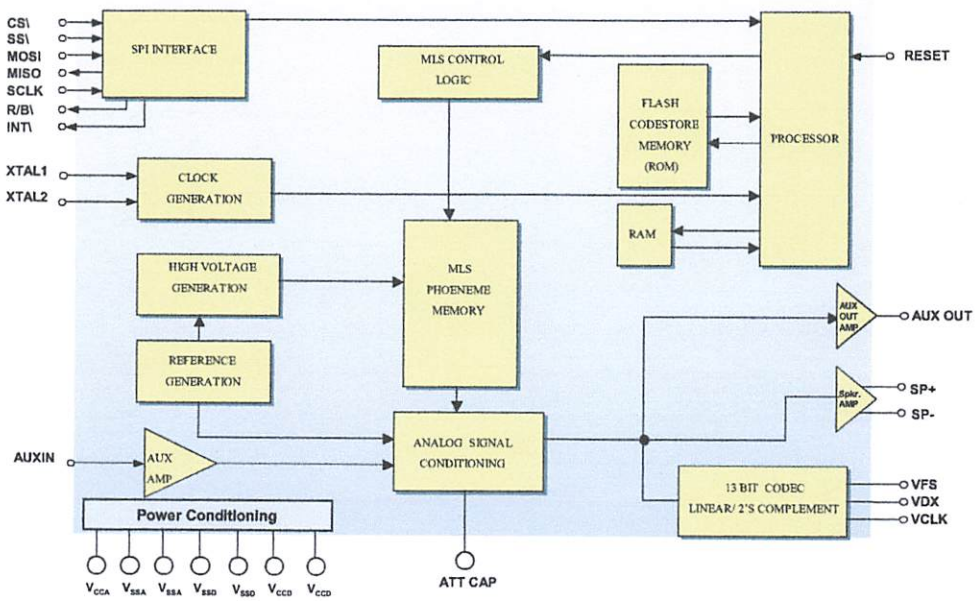


Figure 1. WTS701 Block Diagram.

3.2. WTS701 TYPICAL APPLICATIONS

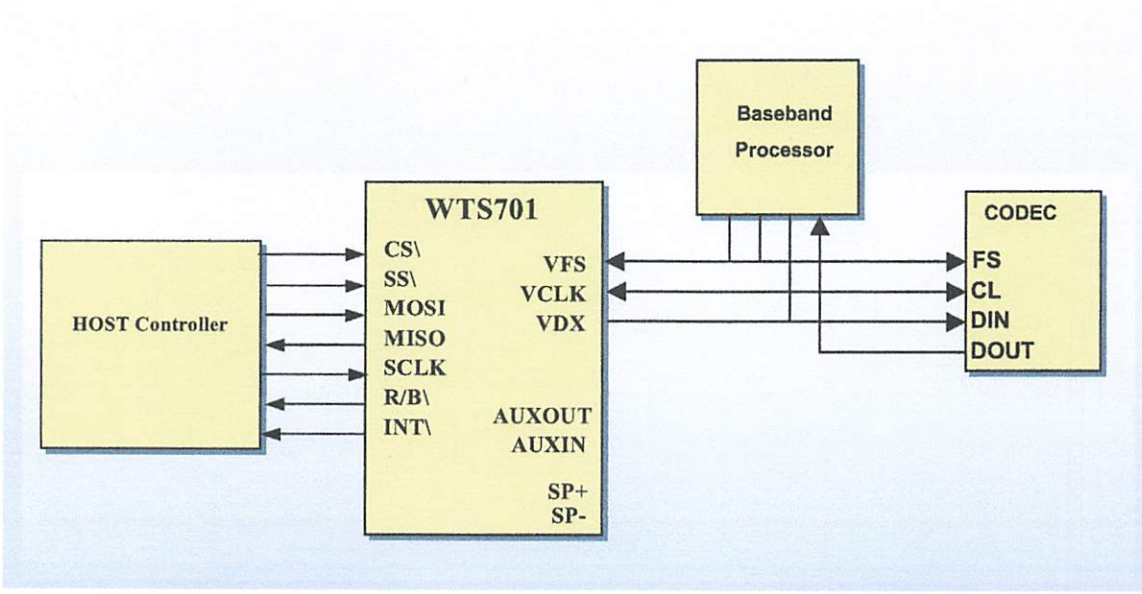


Figure 2. WTS701 Configuration for Digital (CODEC) Environment.

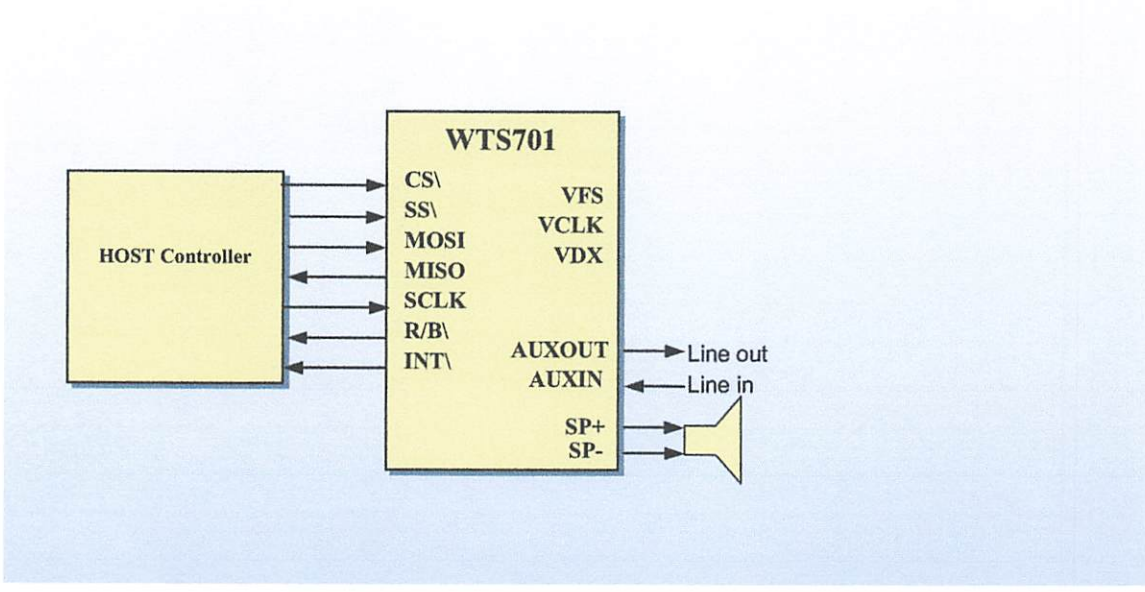


Figure 3. WTS701 Configuration for Analog Environment

4. TABLE OF CONTENTS

5. PIN CONFIGURATION

The following sections detail the pins of the WTS701 processor.

Table 1 shows all the pins and the signals that use them in different configurations. It also shows the type and direction of each signal. Figure 4 shows the physical pin out of the 56-pin TSOP package.

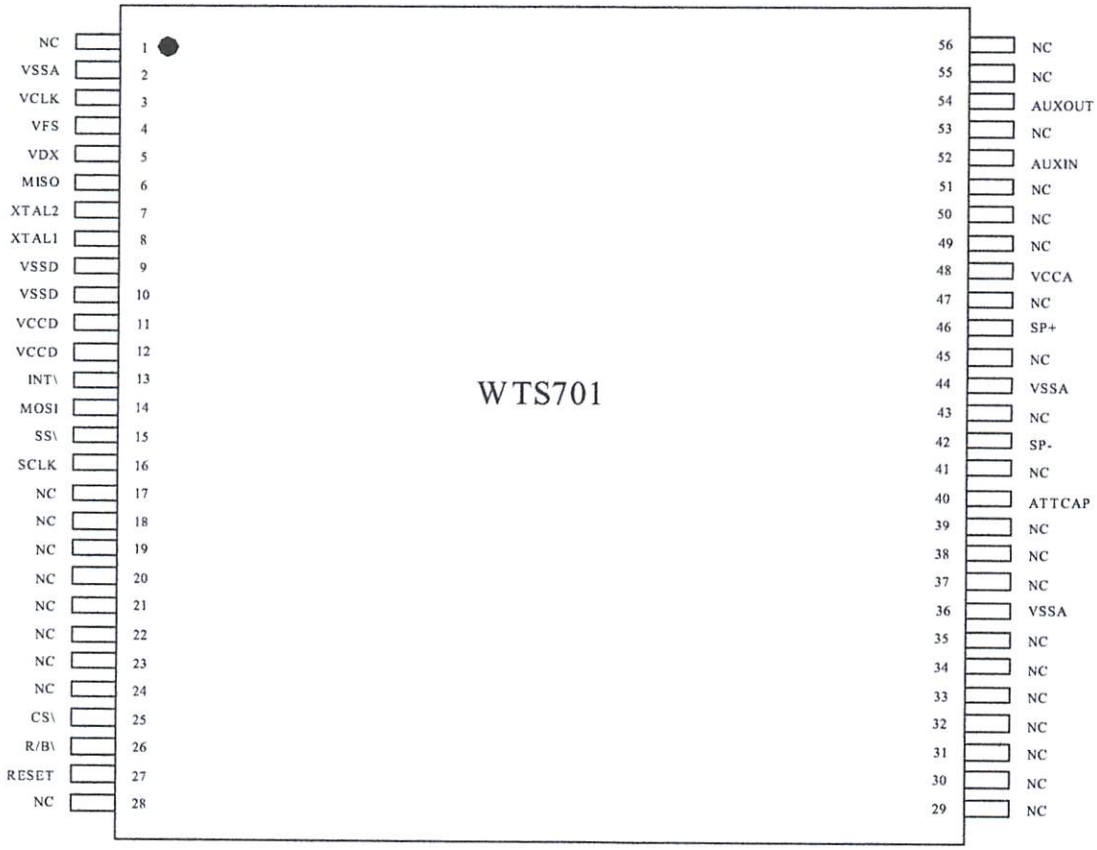


Figure 4. 56-pin TSOP Package Connection Diagram.

6. PIN DESCRIPTION

Table 1. WTS701 Pin Signal Assignment.

PIN NO.	SYMBOL	I/O	FUNCTION
2,36,44	VSSA	G	Analog Ground pins.
3	VCLK	I	CODEC master clock
4	VFS	I	CODEC frame synchronization signal
5	VDX	O	CODEC data output. This pin puts data out in the linear PCM unsigned or 2's complement format. It is tri-stated until the user requests a CONVERT operation.
6	MISO	O	SPI **M**aster **I**n, **S**lave **O**ut pin. Serial data line used to communicate with SPI master. Pin is tri-state when $\overline{SS}$ =1.
7	XTAL2	O	CRYSTAL 2: This is the crystal oscillator output. It is the inversion of XTAL1.
8	XTAL1	I	CRYSTAL 1: This is the crystal oscillator input. This pin may be driven by an external clock. The clock to the WTS701 processor is configured by a clock configuration register, which is set by the host processor during the initialization phase.
9,10	VSSD	G	Digital Ground pin.
11,12	VCCD	P	Positive Digital Voltage Supply pin. These pins carry noise generated by internal clocks in the chip. They must be **independently** bypassed to Digital Ground to ensure correct device operation and not connected together.
13	$\overline{INT}$	O	Interrupt Output; an open drain output that indicates that the device wishes an interrupt service. The device can request an interrupt when it finishes an operation or needs more data to process. Under what conditions the device generates an interrupt can be configured through the user configuration registers. This pin remains LOW until a Read Interrupt command is executed.
14	MOSI	I	SPI **M**aster **O**ut, **S**lave **I**n. Serial data input from Master and Open Drain
15	$\overline{SS}$	I	SPI Slave Select input. This is an active LOW input used to select the device to respond to an SPI transaction.
16	SCLK	I	SPI Serial clock input.
25	$\overline{CS}$	I	Chip Select (active LOW) Pin must be LOW to access WTS701 device.
26	R/$\overline{B}$	O	Ready/busy signal; This pin defaults HIGH indicating the device is ready for data transfer. The pin is driven LOW to handshake a pause in SPI data transfer and Open Drain.

PIN NO.	SYMBOL	I/O	FUNCTION
27	RESET	I	Global reset signal.
40	ATTCAP	I/O	AutoMute Capacitor Pin. Should have a 4.7uF capacitor to V_{SSA}.
42	SP-	O	Differential Negative Speaker Driver Output.
46	SP+	O	Differential Positive Speaker Driver Output.
48	VCCA	P	Positive Analog Voltage Supply pin. This pin supplies the LOW level audio sections of the device. It should be carefully bypassed to Analog Ground to ensure correct device operation.
52	AUXIN	I	Analog input pin. This pin should be capacitively coupled. See page 73 for example.
54	AUXOUT	O	Analog Output for single ended output from the device.
1,17-24,28-35,37-39,41,43,45,47,49-51,53,55-56	NC		Not Connected – must be floating.

Note: TYPE I:Input, O:Output, I/O bi-directional, P:Power, G:Ground

7. FUNCTIONAL DESCRIPTION

As a real System-On-Chip solution, the WTS701 performs the overall control functions for host controller and text-to-speech processing.

The WTS701 system architecture consists of the following functions:

- Serial interface to monitor the SPI port and interpret commands and data
- Text normalization module to pre-process incoming text into pronounceable words
- Words to phoneme translator, which converts incoming text to phoneme codes
- Phoneme mapping module that maps incoming phonemes to words, sub-words, syllables or phonemes present in the MLS memory
- Volume and speed adjustments
- Digital and analog output blocks for off-chip usage

The WTS701 system performs text-to-speech synthesis based on concatenative samples. The units for concatenation can vary from whole words down to phoneme units. The convention is that the larger the sub-word unit used for synthesis the higher the quality of the speech output. A corpus of pre-recorded words is stored in Winbond's patented multilevel storage (MLS) memory and a mapping of the various sub-word parts is held in a lookup table. The speech creation is achieved by concatenation of these speech elements to produce words. The system process flow is shown in Figure 5.

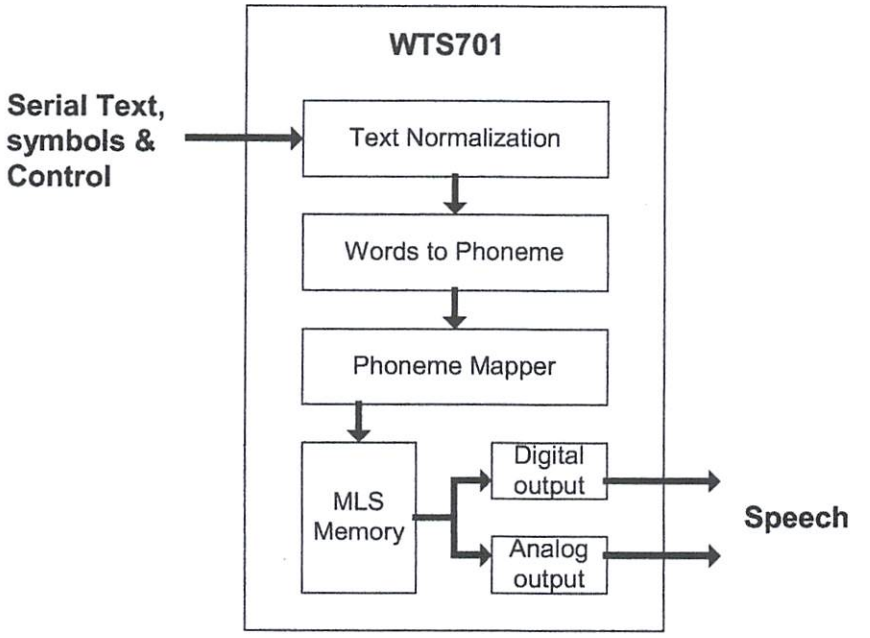


Figure 5. WTS701 System Process Flow.

7.1 TEXT-TO-SPEECH MECHANISM

The text to speech component of the system consists of three principal blocks:

- Text normalization
- Word to phoneme conversion
- Phoneme mapping

7.1.1 Text Normalization

Text normalization involves the translation of incoming text into pronounceable words. It includes such functions as expanding abbreviations and translating numeric strings to spoken words. It involves a certain amount of context processing to determine correct spoken form.

In addition, the WTS701 looks into the abbreviation list stored in the device's internal memory and converts acronyms, abbreviations or special characters (such as Instant Messaging icons or emoticons) into the appropriate text representation.

The default abbreviation list supported by the WTS701 is a general one that cannot be modified by the user to match the domain that the text is being loaded from. But the default list can be overridden by the user abbreviation list. This enables a flexibility of adding abbreviation specifically for the text either by the developer or even the end user to best customize the product for its preferences. Instant Messaging or Short Messages Service (SMS) unique characters are supported through this functionality as well, defining the icon, ASCII/Unicode/Big5 text, and its replacement. The default abbreviation list supported is described in the specific language release letter.

7.1.2 Words-to-Phoneme conversion

Once the data stream has been translated to pronounceable words, the system next determines how to pronounce them. This function is obviously highly language dependent. For a language such as English it is impossible to break this task down to a set of definitive rules. The task is achieved by a combination of rule based processing together with exception processing.

7.1.3 Phoneme Mapping

This algorithm maps phoneme strings into the MLS phonetic inventory. This task falls into two portions. First, the word must be split into sub-word portions. This splitting must be done at appropriate phonetic boundaries to achieve high quality concatenation. Once a sub-word unit is determined, the inventory is searched to determine if a match is present. A matching weight is assigned to each match depending on how closely the phonetic context matches. Each sub-word has a left and right side context to match as well as the phoneme string itself. If no suitable match is found in the inventory, then the sub-word is further split in a tree like manner until a match is found. The splitting tree is processed from left to right and each time a successful match occurs the address and duration of the match in the corpus is placed in a queue of phonetic parts to be played out the audio interface.

7.2 PHYSICAL INTERFACE

The following sections describe the physical pin properties and the timing associated with the physical interface to the device. Note that all input pins are 3V and 5V tolerant, except for the $\overline{CS}$ signal which is only 3V tolerant.

7.2.1 Clocking Requirements

The WTS701 processor can receive its clock from either an external clock source or a crystal oscillator. The XTAL1 and XTAL2 pins provide the crystal interface to the device. The clock to the WTS701 processor is configured by a clock configuration register, which must be set by the host processor during the initialization phase. Figure 6 below shows how to connect the WTS701 to a crystal oscillator. An external clock can be connected to the WTS701 providing the clock source for the system, as shown in Figure 6.

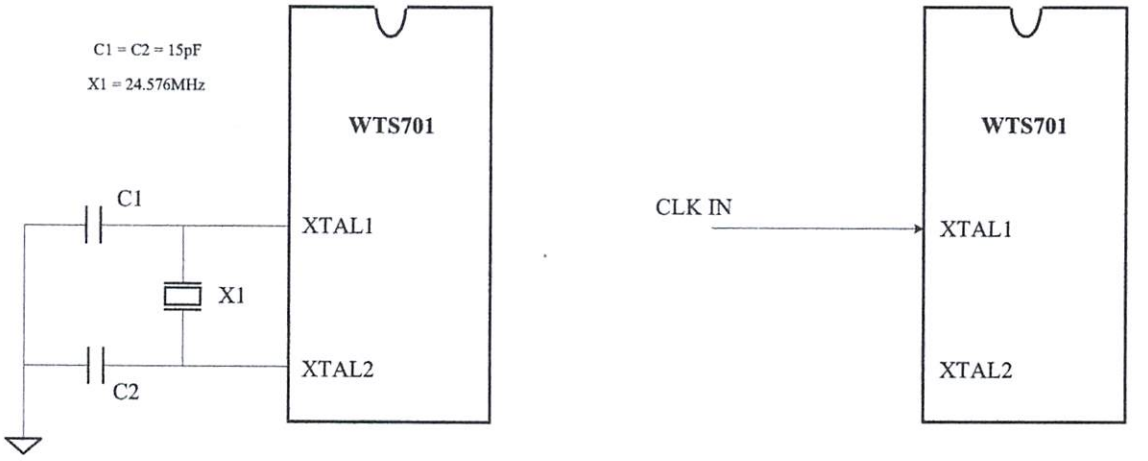


Figure 6. Clock Generation.

Suggested Crystal Specification:

F = 24.576 MHz Fundamental Mode Operation

C_L = 16 pF

ESR = 60 Ω maximum

7.2.2 Power Down Mode

Upon application of power, the WTS701 will enter the RESET state and then be in a POWER DOWN state. In the POWER DOWN mode, only Class0 SPI commands are valid. (See subsection 7.3.1). The Power Down status of the device can be determined with a RDST (Read Status) command, specified by the RDY bit in STATUS BYTE 0.

Issuing the PWDN (Power Down) command to the WTS701 processor will return the processor to the POWER DOWN mode. In POWER DOWN mode the external crystal oscillator is shut off and the processor is deactivated. POWER DOWN mode is exited by issuing a PWUP (Power Up) command to the WTS701. The PWUP command should be preceded by a SCLC (Set Clock) command to ensure correct clock configuration.

7.2.3 Power and Grounding

The WTS701 can operate over 2.7V to 3.3V supply voltage range. The power supply and ground pins (V_{CCA}, V_{CCD}, V_{SSA}, V_{SSD}) should be carefully bypassed as close to the chip as possible to ensure high quality audio. In addition, ATTCAP pin should have a 4.7 µF capacitor connected to ground. This pin must NOT be left floating. The pins that are marked as NC (Not Connected), MUST be left floating.

V_{CCA}, V_{CCD} (Voltage Inputs)

To minimize noise, the analog and digital circuits in the WTS701 device use separate power busses. These +3.0 V busses lead to separate pins. For optimal noise immunity, tie the V_{CCD} pins together as close as possible and decouple both supplies as near to the package as possible.

V_{SSA}, V_{SSD} (Ground Inputs)

The WTS701 series utilizes separate analog and digital ground busses. The analog ground (V_{SSA}) pins should be tied together as close to the package as possible and connected through a low-impedance path to power supply ground. The digital ground (V_{SSD}) pin should be connected through a separate low-impedance path to power supply ground. These ground paths should be large enough to ensure that the impedance between the V_{SSA} pins and the V_{SSD} pin is less than 3Ω. The backside of the die is connected to V_{SSD} through the substrate resistance.

NC (Not Connect)

These pins MUST not be connected to the board at any time. Connection of these pins to any signal, ground or V_{CC} may result in incorrect device behavior or cause damage to the device.

7.2.4 SPI Interface

Communications with the WTS701 is conducted over the SPI serial communications port. The device responds to a command when the Chip Select signal ($\overline{CS}$) is LOW and addressed by an active LOW signal on the $\overline{SS}$ (Slave Select) pin. Under this condition, it accepts data on the MOSI input, which is clocked in on rising edges of the serial clock (SCLK) signal. Concurrently, valid data from the WTS701 device to the bus master is available on MISO for the HIGH period of SCLK. The protocol implemented on the WTS701 defines that the first two bytes of data sent in an SPI transaction is a command word. A transaction is defined as the SPI transfers conducted while $\overline{SS}$ is LOW, the transaction ends when $\overline{SS}$ returns HIGH. A list of available commands can be found in subsection 7.10 (Text-To-Speech Processor Commands Quick Reference Table). The data flow over the SPI interface is MSB first, both in and out of the WTS701.

All Input pins are 3V and 5V tolerant, except for the $\overline{CS}$ signal which is only 3V tolerant.

The following is a description of the WTS701 SPI interface signals:

SCLK (Serial Clock)

The Serial Clock line is a digital input. It is driven by the SPI master and controls the timing of the data exchanged over the SPI data lines, MOSI and MISO. The maximum frequency for this pin is 5 MHz.

$\overline{SS}$ (Slave Select)

The Slave Select line is an active LOW digital input. It is driven by the SPI master and acts as a chip select line. The device only responds to SPI transactions when this line is selected (LOW) and then raised HIGH after SPI communication ends.

$\overline{CS}$ (Chip Select)

The Chip Select line is an active LOW digital input. It can be driven by the host controller to enable SPI transactions to the device. Normally this pin is tied LOW unless more than one device is to share the same $\overline{SS}$ signal.

MOSI (Master Out, Slave In)

The MOSI line is a digital input. MOSI is driven by the SPI master. It provides data transfer, MSB first, from the master to the slave. (See page 64)

MISO (Master In, Slave Out)

The MISO line is an open drain digital output. When $\overline{SS}$ is HIGH, this pin is tri-state. When $\overline{SS}$ is LOW, MISO is driven by the device. It provides serial data transfer, MSB first, from the slave to the master.

7.2.5 Flow Control Interface

In addition to the SPI interface, the WTS701 has two control lines to facilitate data transfer and host communications. The $\overline{INT}$ (interrupt) pin is used by the WTS701 to request an interrupt service from the host controller. The interrupt types that the device generates are controlled by the communications control register command (SCOM). The R/$\overline{B}$ (ready/busy) pin is used to control the flow of data across the SPI bus. When this signal is HIGH, the device can accept more data. When it is LOW, SPI transactions must be paused or terminated.

$\overline{INT}$ (Interrupt)

$\overline{INT}$ is an open drain output pin. The WTS701 interrupt pin goes LOW and stays LOW when an interrupt event has occurred, as defined by the SCOM command. The interrupt is cleared when a RINT (read interrupt) command is executed. The status register defines what type of interrupt has occurred.

R/$\overline{B}$ (Ready/Busy Signal)

The R/$\overline{B}$ line is an output open drain pin used to control data transfer rate across the SPI port. The line is used as a handshake signal to the SPI Master to indicate when the device is ready for more data. When HIGH, the master is free to send more data. When LOW, the device is busy and cannot accept more data.

7.2.6 The CODEC Interface

The WTS701 provides an on chip interface for digital environment systems, supporting slave CODEC interface mode. The WTS701 CODEC interface is controlled by an external source hence the WTS701 only transmits data. Thus, it is effectively an analog-to-digital converter. Each analog sample is converted to 10 bit digital word. This digital word is transmitted with the MSB first. Since the host expects either 13 or 16 bit data in the short frame format, either three or six zeros are appended as the LSB. It interfaces to the baseband CODEC via the VCLK, VFS and VDX lines. Refer to Figure 2, for more information about the connection between the WTS701 and a CODEC.

All Input pins are 3V and 5V tolerant.

The following is a description of the WTS701 CODEC interface signals:

VCLK (CODEC Clock Line)

The CODEC clock line supplies the sampling clock to the internal CODEC. This is a digital input and expects a 512kHz—2.048MHz clock.

VFS (CODEC Synchronization Line)

The CODEC synchronization line supplies a frame synchronization signal to the internal CODEC. This is a digital input. After receipt of a synchronization pulse, the CODEC will output data on the VDX line. The VFS line expects an 8kHz sample rate and supports both short frame and long frame synchronization signal.

VDX (CODEC Data Transmit Line)

The CODEC data transmit line is a digital output that places digital audio data onto the CODEC bus. The line is in a tri-state condition until the device is due to transmit data. The data output from the VDX line is selected by the SCOD Command. When WTS701 places data on the VDX line, it is required that the VFS line should be in tri-state condition when another device is connected to the CODEC as well.

7.2.7 The Analog Interface

The WTS701 provides an on-chip analog interface for audio output via an 8Ω speaker driver or an output buffer capable of driving a 5kΩ load. Additionally, an analog input (AUXIN) allows an audio signal to be fed through the WTS701 chip to either output device. The command SAUD configures the analog path. A digitally controlled attenuator provides volume control via the SVOL command.

The following is a description of the analog pins:

AUXIN (Analog Input)

The AUXIN is an additional audio input to the WTS701. This input has a nominal 694 mV p-p level at its minimum gain setting (0 dB) (See Table 2). Additional gain is available in 3 dB steps (controlled by the SAUD Command) up to 9 dB. The use and equivalent circuit of the input amplifier is shown in Figure 7. (Must be AC coupled)

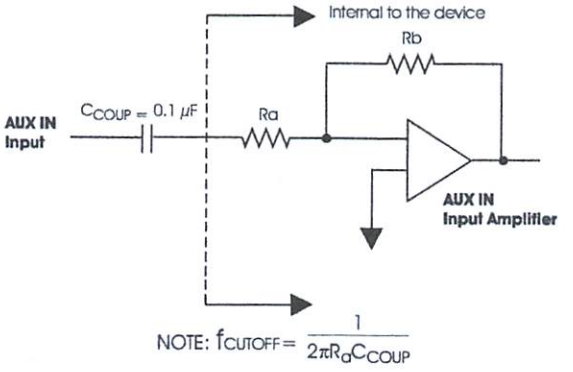


Figure 7. AUXIN Input Amplifier.

Table 2. AUXIN Gain Settings.

0TLP Input V_{P-P}[1]	AUD Register		Gain[2]	Gain[2] (dB)	Resistor Ratio (R_b/R_a)	Speaker Out V_{P-P}[3]
	AIG1	AIG0				
0.694	0	0	1.00	0	40.1 / 40.1	1.388
0.491	0	1	1.41	3	47.0 / 33.2	1.388
0.347	1	0	2.00	6	53.5 / 26.7	1.388
0.245	1	1	2.82	9	59.2 / 21	1.388

[1]OTLP Input is the reference Transmission Level Point that is used for testing. This level is typically 3 dB below clipping.

[2]From AUXIN to AUXOUT. [3]Measured differentially at SP+/SP-.

AUXOUT (Analog Output)

The AUXOUT is an audio output pin used to provide an analog output of the synthesized speech from the WTS701. It drives a minimum load of 5 kΩ up to a maximum of 1 V p-p. The AC signal is superimposed on approximately 1.2 VDC bias and must be capacitively coupled to the load. This output stage may be powered down by clearing the AOPU bit via the SAUD command.

SP +, SP- (Speaker +/-)

This is the speaker differential output circuit. It is designed to drive an 8Ω speaker connected across the speaker pins up to a maximum of 23.5 mW power. This stage has selectable gains of 1.32 and 1.6, which can be chosen through the SPG bit via the SAUD command. These pins are biased to approximately 1.2 VDC and, if used single-ended, must be capacitively coupled to their load. Do **NOT** ground the unused pin. This output stage may be powered down by clearing the SPPU bit via the SAUD command.

ATTCAP (AutoMute Attenuator Capacitor)

This pin provides a capacitor connection for setting the AutoMute. It should have a 4.7 µF capacitor connected to ground and it cannot be left floating. The AutoMute circuit reduces the amount of noise present in the output during quiet pauses.

7.2.8 Resetting

The chip has an internal power-on reset circuit that ensures correct initialization upon application of power. The reset pin signal must be held HIGH for 0.5µs to achieve a reset (see Figure 8) and to put the WTS701 in the RESET state. Once the WTS701 completes the reset, it will enter the POWER DOWN mode. Before issuing active commands, a clock configuration and device power up command must be issue in the POWER DOWN mode.

Issuing a Reset command (RST) resets the WTS701 processor to the initial POWER DOWN state. Applying the reset pin, while the chip is active, allows the host processor to reset the WTS701 to its default values and the IDLE state.

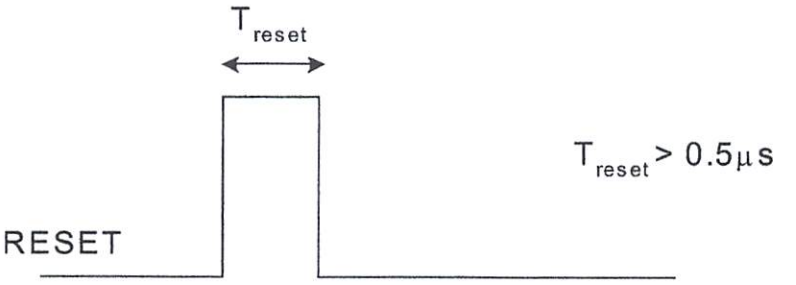


Figure 8. Reset Condition Timing.

7.3 COMMUNICATION PROTOCOL

The WTS701 is controlled by a series of SPI transactions to send commands to the device. The general format of an SPI transaction is shown in Figure 9. A transaction is always started by sending a command word. The command word consists of a command byte followed by a command data byte. At the same time, the status register is shifted out on the MISO line. What follows depends on what command is sent. The general case is that following the command word, up to *n*-bytes of data can be sent to the device and *n*-bytes can be read from the device. An SPI transaction is finished when $\overline{SS}$ is returned to the HIGH condition.

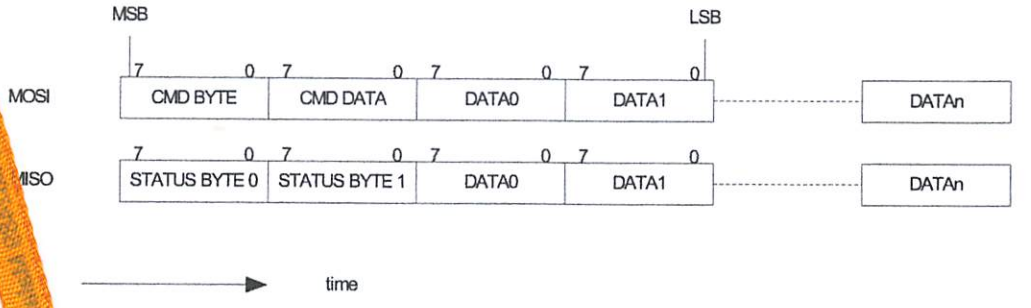


Figure 9. SPI Transaction Format.

7.3.1 Command Classes

The SPI transactions to the WTS701 fall into four classes. The four classes represent variations in how the command, and any associated data, is handled. The class of a command is defined by the two most significant bits of the command byte. A summary of the command classes is given below

Class 0 Commands

These are commands that are executed irrespective of the state of the WTS701. That is, the command will execute even if the device is busy or powered down. These commands are executed internally by a hardware command interpreter. All commands not of class 0 require that the WTS701 be in a powered up state. Example of class 0 command is the Read Status (RDST) command.

Class 1 Commands

Class 1 commands require interpretation by the internal firmware of the WTS701. Class 1 commands consist only of a command byte and command data byte. Any further data sent in a transaction is ignored. Class 1 commands are most often used for setting a configuration register in the device or sending commands that have no data such as the conversion pause (PAUS) command.

Class 2 Commands

Class 2 commands have associated data. After the command word, any data bytes following are loaded into an internal FIFO buffer for processing. If this FIFO becomes full, the $R/\overline{B}$ signal is asserted (LOW) indicating that the host must pause data transfer. An alternative to monitoring the $R/\overline{B}$ line, the $R/\overline{B}$ bit of the status register can be monitored instead (see subsection 7.3.2) or via the RDST command. The $R/\overline{B}$ bit cannot be used for intra-byte flow control, e.g. if a string of characters is sent, only every other byte is checked.

Class 3 Commands

Class 3 commands have data to return to the host. The $R/\overline{B}$ line will go to busy immediately following the command word indicating that the WTS701 is fetching the requested data. Data is put into the BCNT0 and BCNT1 (see subsection 7.3.4) registers and is read out in the two subsequent bytes after $R/\overline{B}$ is released. If more than two bytes are returned from the command, $R/\overline{B}$ will again be asserted until data is ready to read. The primary Class 3 commands are to read the contents of internal configuration registers such as RREG command.

7.3.2 Status Register

The WTS701 has a sixteen-bit status register whose value is returned to the host controller during the command word. For class 2 commands, the status register is repeatedly returned every two bytes. This status register provides the host with information regarding the current status of the chip. The host can decide on required actions with this information. The Status Register is echoed back by all commands.

Table 3. Status Bytes.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
atus Byte 0	*ICNT*	*IBUF*	*ICNV*	*COD*	*BFUL*	*BEMP*	*CNVT*	*RDY*
atus Byte 1	*R/$\overline{B}$*	*Reserved*	*Reserved*	*Reserved*	*Reserved*	*IABB*	*Reserved*	*ICMD*

The contents of the status bytes are described in Table 4.

Table 4. Status Bit Description.

Byte	Bit Name	Bit #	
Status Byte 0	RDY	0	Ready to accept commands. After the device has been powered up, this bit is set after the Power Up latency delay.
	CNVT	1	Converting. This bit is set anytime while the conversion process is running. If this bit is clear when a convert command is sent, the count in the Count register is set to 0.
	BEMP	2	The text input buffer is empty. This bit is set anytime the input buffer is empty.
	BFUL	3	The text input buffer is full. This bit is cleared after 128 bytes become available in the input buffer.
	COD	4	CODEC is enabled. This is set when the CODEC has been enabled by the SCOD command.
	ICNV	5	Conversion finished interrupt has occurred. To stop CODEC transmission or Power Down the analog outputs an IDLE command should be sent. This bit is cleared by RINT command.
	IBUF	6	The input text buffer been filled above the defined threshold and then gone below the defined threshold. The buffer threshold level is set by the SCOM command. If set by the SCOM command, the $\overline{\text{INT}}$ pin will also go LOW. This bit is cleared by RINT command.
	ICNT	7	Count interrupt has occurred. This interrupt is generated every time a word has been spoken if activated by the SCOM command. This bit is cleared by RINT command.
Status Byte 1*	ICMD	0	Command was ignored. Anytime ICMD is set, the transaction must revert to a single word command and the command must be resent. Any data sent will be ignored.
	IABB	2	Abbreviation interrupt has occurred, abbreviation add or abbreviation delete has been completed. Now the ENTER_RRSM command can be sent.
	$\overline{\text{R/B}}$	7	Current state of the $\overline{\text{R/B}}$ pin. If this bit is 0, any data sent will be ignored.

*5 bits are reserved.

7.3.3 Interrupt Handler

If an interrupt has occurred, no further interrupts will be registered until the first interrupt has been cleared. Only one interrupt can be active at any time.

The RINT command will read and clear pending interrupts while the RDST command will read interrupts without clearing them.

Make sure that all interrupts that are not being used are masked by clearing the corresponding bits in the COM register.

7.3.4 BCNT -- Byte Count Register

The byte count register (BCNT) is a tool for the host to keep track of where in a conversion the WTS701 is. When a new conversion is started, the byte count register is reset to zero. As each word (as defined by white-space separated characters) is spoken, the byte count register is updated to point to the first character of the next word to be spoken. In this way, the host can position a new conversion if the user wishes to repeat or skip text. The BCNT register is sent with BCNT1 (MSB) first and BCNT0 (LSB) second.

7.3.5 Command Acceptance

The WTS701 processes commands and data as they are sent to the device. Under certain conditions the device will not be ready to accept a new command or data. If the device has not finished processing the previous command, the ICMD bit of the status register will be set. If this bit is set, it implies that device is not in a position to accept the command being sent and that it will be ignored. The host should monitor this bit when a command is sent and, if it is detected, the SPI transaction should be terminated at the end of the command word. The host can then resend the command until the command is accepted.

7.3.6 Data Acceptance

The WTS701 has an eight byte FIFO to buffer data from the SPI port to the internal processor. During a conversion, data is read from this FIFO into an internal RAM data buffer. If SPI transmission is too fast for the WTS701 to keep up with, the $R/\overline{B}$ line will be asserted (LOW) to pause data transfer. Alternatively, the STATUS register can be monitored for the state of the $R/\overline{B}$ signal.

7.4 COMMANDS OVERVIEW

Control of the WTS701 is implemented through a 16-bit command word. The command word is always the first word to follow the falling edge on the $\overline{SS}$ signal. The command word consists of the command byte followed by the command data byte. Many commands do not require a command data byte, although one must be sent. For commands that have no data, the command data byte is a 'don't care'.

Commands fall into five categories. Commands that control an operational synthesis function of the text-to-speech processing, commands that modify internal configuration registers, commands that change system state, commands that read internal status registers, and customization commands.

Status Commands

Table 5. Status Opcodes.

Opcode	Mnemonic	Function
0x04	RDST	Read Status Register
0x06	RINT	Read Interrupt Register
0x12	RVER	Device Version

The WTS701 has three read-only registers accessed by the opcodes, which are shown to the right.

- The Read Status Register returns the device's operational status and the numbers of bytes that have been converted.
- The Read Interrupt Register returns the same status data and clears any of the interrupt status bits that are set.
- The Version Register returns the hardware and firmware version of the chip.

System Commands

Table 6. System Opcodes.

Opcode	Mnemonic	Function
0x02	PWUP	Power Up
0x40	PWDN	Power Down
0x10	RST	Reset
0x57	IDLE	Go Idle

The WTS701 responds to various system commands that change the state of the system, namely:

- The Power Up command wakes up the device from POWER DOWN mode.
- The Power Down command requests that the device enter the POWER DOWN mode.
- The Reset command resets the device (see subsection 7.4.4).
- The Idle command puts WTS701 processor in IDLE mode

Synthesis Commands

The synthesis commands affect the text-to-speech synthesis. They are detailed in the table to the right. The basic commands are:

- Start a conversion
- Pause the conversion
- Resume the conversion
- Stop the conversion
- Finish conversion at the end of the current word.
- Finish the conversion at the end of the buffer
- Volume up/down
- Speed up/down the text-to-speech conversion

Table 7. Synthesis Opcodes.

Opcode	Mnemonic	Function
0x81	CONV	Start Converting
0x49	PAUS	Pause Conversion
0x4A	RES	Resume Conversion
0x4B	ST	Stop Conversion
0x4D	FINW	Finish Word
0x4C	FIN	Finish Buffer
0x53	VLUP	Volume Up
0x54	VLDN	Volume Down
0x55	SPUP	Speed Up Conversion
0x56	SPDN	Slow Down Conversion

Configuration Commands

The WTS701 has several configuration registers. The commands are:

- The COM configuration register governs the behavior of how the chip uses the $\overline{INT}$ and R/$\overline{B}$ hardware lines to communicate with the host
- The CODEC register configures the mode of the digital audio output
- The AUDIO register sets parameters of the analog audio path
- The VOLUME register sets the volume level of output
- The SPEED register sets the speed level of output speech
- The CLC register sets the master clock frequency of the device
- The SPTC command sets speech pitch

Table 8. Configuration Opcodes.

Opcode	Mnemonic	Function
0xC0	RREG	Read Configuration register
0x4E	SCOM	COM Configuration register
0x4F	SCOD	CODEC Configuration register
0x50	SAUD	AUDIO Configuration register
0x51	SVOL	VOL Configuration register
0x52	SSPD	SPEED Configuration register
0x14	SCLC	CLC (Clock) Configuration register
0x77	SPTC	Set Speech Pitch

Customization Commands

The WTS701 has the ability for the user to customize the way in which it responds to certain text strings. This is done by way of an abbreviation table. The customization opcodes allow the user to interrogate and modify the abbreviation table.

Table 9. Customization Opcodes.

Opcode	Mnemonic	Function
0xC8	ABBR_NUM	Get number of abbrev. entries
0xC9	ABBR_RD	Read abbreviation table
0xC7	ABBR_MEM	Get number of free bytes.
0xAF	ABBR_ADD	Add abbrev. entry
0x83	ABBR_DEL	Delete abbrev. entry
0x0C	ENTER_RRSM	Swap memory

7.4.1 Command Description

The following section list all the standard commands that can be executed on the WTS701.

PWDN Go to POWER DOWN Mode

This command puts the WTS701 processor in power-down mode. This is a single word command therefore no data is required for this command. The Power Down command places the WTS701 device into its lowest power consumption mode. In POWER DOWN mode, the device will only respond to a Power Up command (PWUP) and Read Status (RDST) command. As soon as Power Down sequence has ended, the RDY flag in the status word is cleared.

PWDN	Class	1	Type	I	
Byte Sequence:	Host controller		0x40		0x00
	WTS701		Status Byte 0		Status Byte 1
Description:	Put the WTS701 processor in power-down mode.				

PWUP Power Up

This command wakes up the WTS701 processor to IDLE state. The result of this command is that the WTS701 starts the power up sequence, which leads to bringing up internal supplies, resetting the processor, all configuration registers are initialized to their default values and entering IDLE state. As soon as power up sequence has ended, the RDY flag in the status word is asserted. The SCLC command must be sent BEFORE PWUP.

PWUP	Class	0	Type	I	
Byte Sequence:	Host controller		0x02		0x00
	WTS701		Status Byte 0		Status Byte 1
Description:	Wake up the WTS701 processor to IDLE state.				

CONV Convert

The convert command starts the text to speech conversion process. The convert command is followed by ASCII text data. The device has a buffer of 256 bytes. When this buffer is full, the chip pulls the R/$\overline{B}$ line LOW and sets the BFUL bit in the status word indicating that the WTS701 buffer manager is in the buffer full condition. The WTS701 remains in the buffer full condition until the input buffer has been emptied of half the buffer space (128 bytes). When the buffer is full, the Host may do one of three things:

1. The Host may end the command at that point, then poll the BFUL bit of the SPI status register until it is clear, and then send new CONV commands with the additional ASCII text data.

2. The Host may also continue the command (keep $\overline{SS}$ LOW) and wait for the R/$\overline{B}$ pin to go HIGH. As each word is processed by the WTS701, space will become free in the buffer and the R/$\overline{B}$ pin will go HIGH until it is full again.

3. The device may also be configured such that it will generate an interrupt to the host when the buffer threshold (set by RCOM command) has been crossed. (See Tables 3 and 4) This allows the host to fill the buffer then wait for the Interrupt to send the additional data.

During conversion, the Convert Count Register is updated as each word has been spoken. This register is cleared to zero at power up, and at the beginning of a new conversion process after one has been terminated.

A convert command is terminated in several ways:

- Send a finish command (FIN) indicating that the host has finished sending data. In this case, the device finishes converting the text buffer, then stops and enters a wait state.
- The conversion process will also stop when the EOT (^D, ASCII 0x1A, UNICODE/Big5 0x00 0x1A) character is part of the input text. When the device detects the EOT character, it will continue the conversion process until the buffer is emptied and the final word spoken. Then it will stop and enter the wait state.
- The finish word command (FINW) will cause the WTS701 device to finish the word currently being spoken, then flush the buffers and enter the wait state.

- The stop command (ST) will cause the WTS701 to immediately stop converting, flush the buffer and enter the wait state.

Once the wait state has been entered the device will clear the convert (CONV) bit from the status register and, if enabled, generate an ICVT interrupt. At this stage the CODEC and analog path are still active. To release the CODEC bus, or Power Down the analog path, an IDLE command should be sent to the device. If a convert command is terminated using any of the methods described in this section, another convert command cannot be sent until the previous conversion is completed. The CNVT bit must be polled to determine that conversion is completed before a new conversion can be started.

CONV	Class	2	Type	III			
Byte Sequence:	Host controller		0x81	0x00	DATA0	...	DATAn
	WTS701		Stadus Byte 0	Status Byte 1	Status Byte 0	...	Status Byte n%2
Description:	Start or continue a conversion process. Data sent is text data for conversion.						

PAUS **PAUSE**

This command causes a pause of the conversion process. There is no data associated with this command. The pause condition is terminated by the RES (Resume) command

PAUS	Class	1	Type	I	
Byte Sequence:	Host controller		0x49		0x00
	WTS701		Status Byte 0		Status Byte 1
Description:	This command pauses the conversion process.				

RES **RESUME**

This command causes the conversion to resume if it was paused. There is no data associated with this command

RES	Class	1	Type	I	
Byte Sequence:	Host controller		0x4A		0x00
	WTS701		Status Byte 0		Status Byte 1
Description:	This command resumes conversion after pause.				

ST **STOP**

This command immediately stops conversion without finishing buffer, and clears the buffer.

ST	Class	1	Type	I	
Byte Sequence:	Host controller		0x4B		0x00
	WTS701		Status Byte 0		Status Byte 1
Description:	Stop conversion.				

FINW **FINISH WORD**

This command directs the WTS701 to finish text conversion at the end of the current word.

FINW	Class	1	Type	I	
Byte Sequence:	Host controller		0x4D		0x00
	WTS701		Status Byte 0		Status Byte 1
Description:	This indicates that conversion is to end with the processing of the current word.				

FIN **FINISH**

This command indicates that no further conversion data is to follow and to stop conversion after processing the current buffer contents.

FIN	Class	1	Type	I	
Byte Sequence:	Host controller		0x4C		0x00
	WTS701		Status Byte 0		Status Byte 1
Description:	Finish conversion after processing the current buffer.				

IDLE **IDLE**

This command is executed after the receipt of an end-of-conversion interrupt (ICNV) has occurred. The IDLE command will deactivate all audio outputs and bring the device to the IDLE state.

IDLE	Class	1	Type	I	
Byte Sequence:	Host controller		0x57		0x00
	WTS701		Status Byte 0		Status Byte 1
Description:	Put WTS701 in IDLE state.				

RDST **READ STATUS**

The Read Status command reads the status word of the device. If two dummy data bytes are also sent, the contents of the byte count register are also returned. Refer to subsections 7.3.2 and 7.3.4 for more information regarding the STATUS register and BCNT register.

RDST	Class	0	Type	II			
Byte Sequence:	Host controller		0x04		0x00	0x00	0x00
	WTS701		Status Byte 0		Status Byte 1	BCNT 1	BCNT0
Description:	Read Status word of the device.						

RVER **READ VERSION**

The Read version command reads the WTS701 version information. The software version information is only valid when the device is powered up.

RVER	Class	0	Type	II			
Byte Sequence:	Host controller		0x12		0x00	0x00	0x00
	WTS701		Status Byte 0		Status Byte 1	HW VER	SW VER
Description:	Read WTS701 Software and Hardware versions.						

RINT READ INTERRUPT

The Read Interrupt command reads the status word of the device, it also clears the status interrupt request flags at the end of the transaction. As a result of this command, all interrupt bits are cleared and $\overline{INT}$ pin is released. Refer to subsections 7.3.2 and 7.3.4 for more information regarding the STATUS register and BCNT register.

RINT	Class	0	Type	II		
Byte Sequence:	Host controller		0x06	0x00	0x00	0x00
	WTS701		Status Byte 0	Status Byte 1	BCNT1	BCNT0
Description:	Read status word and clear the status interrupt bits.					

RREG READ CONFIGURATION REGISTER

The read configuration register command reads the configuration register specified in the command data byte. The code 0xNN is the register number and it is described in Table 10 – Configuration Registers, subsection 7.4.2.

RREG	Class	3	Type	IV		
Byte Sequence:	Host controller		0xC0	0xNN	0x00	0x00
	WTS701		Status Byte 0	Status Byte 1	XX	REG
Description:	Read configuration register 0xNN.					

Note: XX = don't care.

SCOM SET COM REGISTER

Set the COM (interrupt communication) configuration register to value 0xNN. Refer to subsection 7.4.3 describing all configuration registers and the COM register in particular.

The Default value of this register after Power-Up or Reset is 0x00. Refer to subsection 7.4.3 - Configuration Registers, which describes all register bits.

SCOM	Class	1	Type	I	
Byte Sequence:	Host controller		0x4E		0xNN
	WTS701		Status Byte 0		Status Byte 1
Description:	Set the COM (interrupt communication) configuration register to value 0xNN.				

SCOD SET COD REGISTER

Set the COD (CODEC control) configuration register to value 0xNN.

The Default value of this register after Power-Up or Reset is 0x01. Refer to subsection 7.4.3 - Configuration Registers, which describes all register bits.

SCOD	Class	1	Type	I	
Byte Sequence:	Host controller		0x4F		0xNN
	WTS701		Status Byte 0		Status Byte 1
Description:	Set the COD (CODEC control) configuration register to value 0xNN.				

SAUD SET AUD REGISTER

Set the AUD (analog audio) configuration register to value 0xNN.

The Default value of this register after Power-Up or Reset is 0x43. Refer to subsection 7.4.3 - Configuration Registers, which describes all register bits.

SAUD	Class	1	Type	I	
Byte Sequence:	Host controller		0x50		0xNN
	WTS701		Status Byte 0		Status Byte 1
Description:	Set the AUD (analog audio) configuration register to value 0xNN.				

SVOL SET VOL REGISTER

Set the VOL (volume) configuration register to value 0xNN.

The Default value of this register after Power-Up or Reset is 0x07. Refer to subsection 7.4.3 - Configuration Registers, which describes all register bits.

SVOL	Class	1	Type	I	
Byte Sequence:	Host controller		0x51		0xNN
	WTS701		Status Byte 0		Status Byte 1
Description:	Set the VOL (volume) configuration register to value 0xNN.				

SSPD **SET SPD REGISTER**

Set the SPD (speech rate/speed) configuration register to value 0xNN.

The Default value of this register after Power-Up or Reset is 0x02. Refer to subsection 7.4.3 - Configuration Registers, which describe all register bits.

SSPD	Class	1	Type	I	
Byte Sequence:	Host controller		0x52		0xNN
	WTS701		Status Byte 0		Status Byte 1
Description:	Set the SPD (speech rate/speed) configuration register to value 0xNN.				

SCLC **SET CLC REGISTER**

Set the Clock configuration register (CLC) to value 0xNN.

The value of this register **must** be set before Power-Up or Reset command to 0x00. Refer to subsection 7.4.3 - Configuration Registers, which describes all register bits.

SCLC	Class	0	Type	I	
Byte Sequence:	Host controller		0x14		0xNN
	WTS701		Status Byte 0		Status Byte 1
Description:	Set the Clock configuration register (CLC) to value 0xNN.				

SPTC **SET SPEECH PITCH**

Set the speech pitch to value 0xNN. The valid pitch values are between 0x00 and 0x06 while the default pitch value is 0x05, and these values can be used to control the speech output pitch. The command can be executed only when the WTS701 is in IDLE state.

SPTC	Class	1	Type	I	
Byte Sequence:	Host controller		0x77		0xNN
	WTS701		Status Byte 0		Status Byte 1
Description:	Set the speech pitch parameter to value 0xNN.				

VLUP VOLUME-UP COMMAND

Increment the volume (VOL) register. Has no effect if already at maximum volume.

The Default value of this register after Power-Up or Reset is 0x07. Refer to subsection 7.4.3 - Configuration Registers, which describes all register bits.

VLUP	Class	1	Type	I	
Byte Sequence:	Host controller		0x53		0x00
	WTS701		Status Byte 0		Status Byte 1
Description:	Increment the volume (VOL) register.				

VLDN VOLUME DOWN COMMAND

Decrement the volume (VOL) register. This has no effect if already at minimum volume. The Default value of this register after Power-Up or Reset is 0x07. Refer to subsection 7.4.3 - Configuration Registers, which describes all register bits.

VLDN	Class	1	Type	I	
Byte Sequence:	Host controller		0x54		0x00
	WTS701		Status Byte 0		Status Byte 1
Description:	Decrement the volume (VOL) register.				

SPUP SPEED UP COMMAND

Increase speaking rate (SPD register). This has no effect if already at maximum speaking rate. The Default value of this register after Power-Up or Reset is 0x02. Refer to subsection 7.4.3 - Configuration Registers, which describes all register bits.

SPUP	Class	1	Type	I	
Byte Sequence:	Host controller		0x55		0x00
	WTS701		Status Byte 0		Status Byte 1
Description:	Increase speaking rate (SPD register).				

SPDN **SPEED DOWN COMMAND**

Decrease speaking rate (SPD register). Has no effect if already at minimum speaking rate. The Default value of this register after Power-Up or Reset is 0x02. Refer to subsection 7.4.3 - Configuration Registers, which describes all register bits.

SPDN	Class	1	Type	I	
Byte Sequence:	Host controller		0x56		0x00
	WTS701		Status Byte 0		Status Byte 1
Description:	Decrease speaking rate (SPD register).				

RST **RESET COMMAND**

Sending this command has the same affect as a Power-On reset, the WTS701 enters the POWER DOWN state.

RST	Class	0	Type	I	
Byte Sequence:	Host controller		0x10		0x00
	WTS701		Status Byte 0		Status Byte 1
Description:	Reset the WTS701 device.				

ABBR_ADD **ADD ABBREVIATION**

Add an entry to the abbreviation table.

ABBR_ADD	Class	2	Type	III			
Byte Sequence:	Host controller		0xAF	0x00	DATA0	...	DATAn
	WTS701		Status Byte 0	Status Byte 1	Status Byte 0	...	Status Byte n%2
Description:	Add an entry to the abbreviation table..						

ABBR_MEM RETURN ABBREVIATION MEMORY

The ABBR_MEM command will return the number of bytes available in the abbreviation table in MEM_HI and MEM_LOW.

ABBR_MEM	Class	3	Type	IV		
Byte Sequence:	Host controller		0xC7	0x00	0x00	0x00
	WTS701		Status Byte 0	Status Byte 1	MEM_HI	MEM_LOW
Description:	Return the number of bytes available in the abbreviation table.					

ABBR_NUM RETURN NUMBER OF ABBREVIATION ENTRIES

The ABBR_NUM command will return the number of abbreviation entries in the abbreviation table in NUM_HI and NUM_LOW.

ABBR_NUM	Class	3	Type	IV		
Byte Sequence:	Host controller		0xC8	0x00	0x00	0x00
	WTS701		Status Byte 0	Status Byte 1	NUM_HI	NUM_LOW
Description:	Return the number of abbreviation entries in the abbreviation table.					

ABBR_RD READ ABBREVIATION TABLE

The ABBR_RD command will return the abbreviation table. This command must read 2048 bytes after receiving the Status register.

ABBR_RD	Class	3	Type	IV			
Byte Sequence:	Host controller		0xC9	0x00	0x00	.....	0x00
	WTS701		Status Byte 0	Status Byte 1	ABBR0	.....	ABRRn
Description:	Return the 2048 bytes of abbreviations.						

ABBR_DEL DELETE ABBREVIATION ENTRY

This command deletes abbreviation entry from abbreviation table.

ABBR_DEL	Class	2	Type	I			
Byte Sequence:	Host controller		0x83	0x00	DATA0	...	DATAn
	WTS701		Status Byte 0	Status Byte 1	Status Byte 0	...	Status Byte n%2
Description:	Delete an entry from the abbreviation table.						

ENTER_RRSM SWAP MEMORY

This command is used in programming mode, and causes the xdata and code store memory to swap spaces. Please refer to subsection 7.8 for more information about customizing abbreviations.

ENTER_RRSM	Class	0	Type	I	
Byte Sequence:	Host controller		0x0C		0x00
	WTS701		Status Byte 0		Status Byte 1
Description:	Swap memory between xdata and code store.				

7.4.2 Illegal Commands

All commands described in section 7.4.1 are the only legal commands that should be sent to the WTS701 device, unless stated otherwise. Other commands should not be sent to the device, as the device behavior cannot be predicted. Specific illegal commands are those in which the 2 Most Significant Bits of the Command Byte are zeros and are not defined in this document as commands allowed to be sent to the WTS701.

7.4.3 Configuration Registers

The configuration registers are accessed by sending the appropriate configuration command followed by a single byte of data to load the register. The definition of the contents of the various registers is given below. The default value for each of these registers after Power Up or Reset is also described in Table 10.

Table 10. Configuration Registers.

			MSB							LSB
Register	Reg. #	Default	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
COM	0x4E	0x00	ICNT	IBUF	ICNV	X	X	X	BUF1	BUF0
COD	0x4F	0x01	X	X	X	X	X	MD2	MD1	MD0
AUD	0x50	0x43	AOPU	SPPU	SPG	FDTH	X	X	AIG1	AIG0
VOL	0x51	0x07	X	X	X	X	X	VL2	VL1	VL0
CLC	0x14	None	X	X	X	CLC4	CLC3	CLC2	CLC1	CLC0
SPD	0x52	0x02	X	X	X	X	X	SPD2	SPD1	SPD0

X = Reserved.

The bits of each register are described below:

COM Register

ICNT If set to a '1', the device will generate an interrupt when the Count register has been updated. This occurs after each word has been spoken.

IBUF If set to '1', the device will generate an interrupt when the buffer level crosses the threshold set by the BUF bits. (see Table 3, Status bytes).

ICNV If set to '1', the device will generate an interrupt when the end of a conversion is reached.

BUF1..0 If IBUF is set, BUF1..0 determines the buffer level at which the interrupt will be generated.

00b – Input buffer empty.

01b – Input buffer <10% full.

10b – Input buffer <50% full.

11b – Input buffer <75% full.

COD Register

MD2 CODEC enable, possible modes are:
0b: CODEC disabled.
1b: CODEC enabled during conversion.

MD1 CODEC precision, possible modes are:
0b: 13 bit linear PCM output
1b: 16 bit linear PCM output.

MD0 CODEC output format, possible modes are:
0b: unsigned PCM output
1b: 2's complement PCM output.

AUD Register

AOPU 1b: Power up the analog output buffer.

SPPU 1b: Power up the analog speaker driver.

SPG Speaker Driver gain selection.
0b: 8Ω Speaker. A_v = 1.32
1b: 100Ω Speaker. A_v = 1.6

FDTH 1b: Enable feed-through path from AUXIN to AUXOUT.

AIG1..0
AUXIN gain setting
00b – 0dB
01b – 3dB
10b – 6dB
11b – 9dB

VOL Register

VL2..0

Volume level of output.

000 – 0dB

001 – -4dB

010 – -8dB

011 – -12dB

100 – -16dB

101 – -20dB

110 – -24dB

111 – -28dB

Each step gives a 4dB attenuation of output.

CLC Register

CLC4..0

Configure the device for different master clock frequencies.

0x00	24.576 MHz
0x10	16.384 MHz
0x08	32.768 MHz

(The only clock frequency currently recommended for operation is 24.576MHz.)

SPD Register

SPD2..0

Configure the speech speed register. 0x04 is the fastest speed and 0x00 is the slowest.

7.4.4 System Operation

The WTS701 is a single chip solution for text-to-speech synthesis. The Text-to-Speech operation is accomplished by a process of screening the incoming text to normalize common abbreviations and numbers into a spoken form. The normalized text is then analyzed for phonetic interpretation and this phonetic translation is mapped into samples to be played out of the analog storage array. This output signal is then smoothed by a low-pass filter and is available as an analog signal, or can be passed through the CODEC for digital audio output.

The WTS701 processor state machine

The WTS701 functions as a state machine and changes states either in response to a command sent by the host controller, after execution of command is completed, or as a result of an internal event.

The WTS701 states are described below in reference to Figure 10.

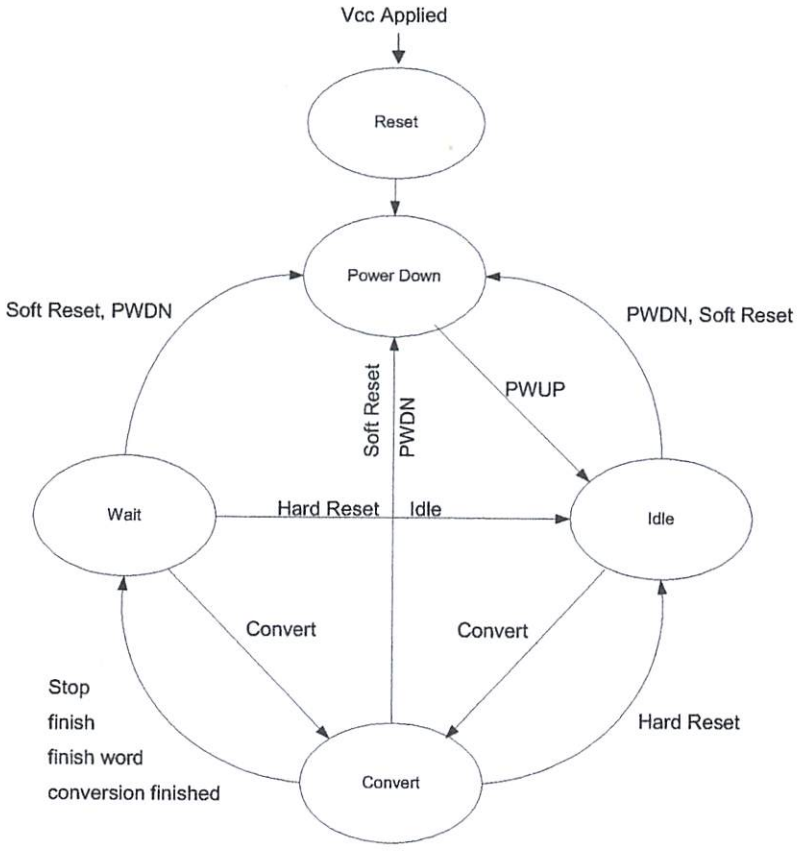


Figure 10. WTS701 Processor States

RESET

The WTS701 processor is initialized to the RESET state when Vcc is first applied to the part.

After a reset condition the device enters the POWER DOWN state. All configuration registers are initialized to their default values after issuing the PWUP command.

Once the WTS701 is active and a hardware reset is applied on the RESET pin, the WTS701 will be in IDLE state, and all configuration registers will return to their default values.

POWER DOWN

In this state, the power consumption of the WTS701 is minimal. All analog outputs are tristate, the crystal interface is deactivated and the microcontroller is stopped. The only commands valid in the Power Down mode are PWUP, SCLC and RDST. All configuration registers will return to their default values after issuing the PWUP command.

IDLE

The idle state is first entered with the PWUP command. In this state, the micro-controller is running and the device is ready to respond to further commands. From the IDLE state, the device can go to the active CONVERT state or the POWER DOWN state.

CONVERT

This state is initiated by the CONV command. The text located in the internal buffer is converted into speech and played back to the analog or digital interface according to the state of the configuration registers. Once the active conversion has finished, the device enters the WAIT state.

WAIT

Once a conversion has finished, the device enters the WAIT state. In this state, audio outputs are still active. To deactivate, audio outputs and return to the IDLE state an IDLE command is issued.

7.4.5 Initialization and Configuration

Configuration

After power-on or a Reset command (RST) the WTS701 processor can be configured for operation. This involves initializing the internal configuration registers for the users requirements.

Table 11. Initialization Commnad Sequence

State	Command	Description
POWER DOWN	------	State after power-on or RST command.
	SCLC	Set clock configuration.
	PWUP	Power up device.
IDLE	SCOM	Set up communication register to enable interrupts.
	SCOD	Set up CODEC configuration (if used).
	SAUD	Set up audio control register.
	SVOL	Set the initial volume level.
	SSPD	Set the initial speech output speed level.
	SPTC	Set the initial speech pitch level.

7.4.6 Converting Text

After configuration, the WTS701 is ready for text-to-speech conversion. Because of the real-time nature of speech, some form of flow control is necessary to inform the host system:

1. When the device is ready for more text data
2. When the device has finished converting text
3. When the device can release the audio interface

The CONVERT state is entered by sending a CONV command along with some textual data. The WTS701 has an internal 256-byte buffer to accept text data. The $R/\overline{B}$ signal (both the hardware line and the status bit) will become active (LOW):

1. When the internal buffer is full
2. If the host sends data at a rate too fast for the WTS701 to process it to the internal buffer

When $R/\overline{B}$ becomes active the user may:

1. Wait for the $R/\overline{B}$ pin to return to the (HIGH) ready state
2. Terminate the SPI transaction until a later time and resend the data

The user has the choice of enabling interrupts to signal the host when there is free space in the internal buffer. When all text data has been sent, the user must indicate this by:

1. Sending a FIN (Finish) command
2. Sending an EOT (ASCII 0x1A) character as the last byte of a CONV command (MANDARIN UNICODE/Big5 0x00 0x1A)

When conversion has been terminated using either of these commands, another CONV command cannot be sent until conversion is completed and the chip enters the WAIT state.

Notes

1. Buffer length limit:
   The max. character length of a white-space-bounded string is 53. The exceeding characters will be truncated.
2. Undefined characters:
   All the undefined characters will be deleted (prior to the word pronounciation process). The defined characters range from '0x00' to 0x7A' excluding '0x22', '0x3C', '0x3E' and '0x60'.

Once the WTS701 has synthesized the contents of the text buffer, it will enter the WAIT state. In this state the audio interface is still active. To disable the audio interface and return to the IDLE state an IDLE command is sent. The WAIT state can be detected either through polling of the CNVT bit or enabling of the ICNV interrupt. An example of the flow for a conversion is shown in Figure 11. The flow here assumes that the COM register is set such that the WTS701 generates an IBUF interrupt at the 75 percent buffer level (64 free bytes) and that the ICNV interrupt is enabled.

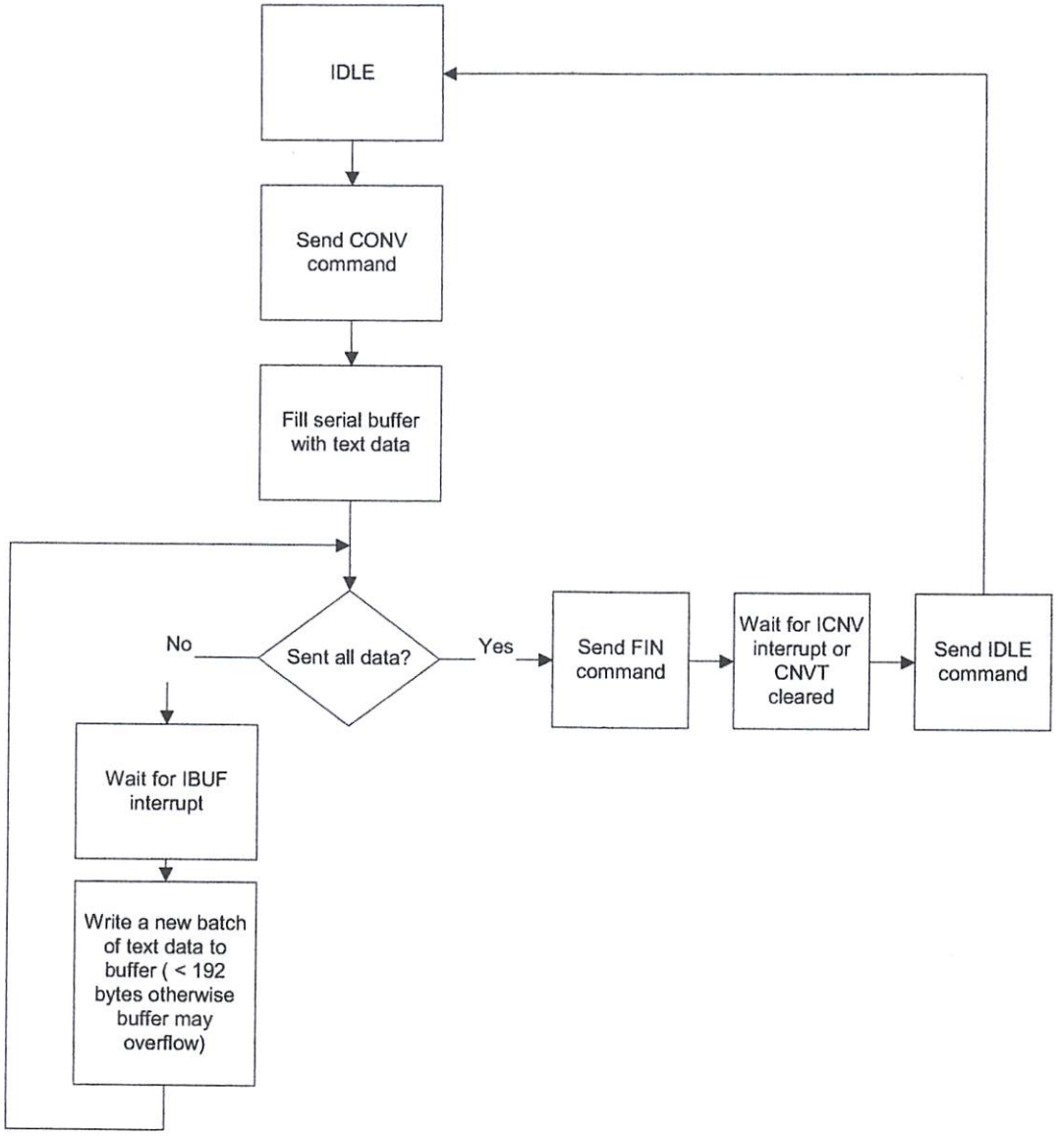


Figure 11. Flow Diagram for Convert Operation.

Controlling Text Conversion

The WTS701 offers several features to control text conversion. The PAUS (Pause) and RES (Resume) commands allow the host to pause and then continue speech output. The FINW command allows the host to end a conversion after the next whole word is spoken. The ST (Stop) command will terminate a conversion immediately – even mid-word. To allow more advanced control, the WTS701 allows the host to interrogate the byte count register, which keeps track of the position in the input stream that is currently being spoken. If the host wishes to repeat a spoken word, it should:

1. Read the byte count register
2. Send a FINW or ST command
3. Wait for ICNV
4. Send a new CONV command resending the data starting at the desired number of bytes, according to the repeated spoken words, prior to the count returned in the byte count

In a similar way a skip function could be implemented to skip ahead words or sentences.

7.5 SPI INTERFACE

The SPI interface consists of the 4-wire bus $\overline{SS}$, SCLK, MOSI and MISO. In addition, flow control protocols are implemented via the R/$\overline{B}$ signal and/or the WTS701 Status register transmitted via MOSI. The WTS701 processor also has the option to communicate with the host via interrupt services requested by the interrupt request line. The timing and behavior of these signals is dependent upon the command class being executed, i.e., commands with or without associated data. Additionally the use of the R/$\overline{B}$ hardware control line is not compulsory; rather the host can monitor the R/$\overline{B}$ bit of the status register to determine when data has been accepted. The status register also contains the ICMD bit. This bit is set to indicate an SPI transaction has been ignored, indicating that the WTS701 processor is unable to service a new command. Asynchronous (Class 0) commands are always accepted. For more information, refer to subsection 7.3.1 which describes the command classes.

7.5.1 SPI Transactions

SPI Transactions with the WTS701 are broken down into four classes and four basic types:

Type I - Single Word Transactions

Single word transactions are Class 0 or Class 1 commands that have no data to transmit or transmit all required data in the command data byte. R/$\overline{B}$ will never become active for these commands. ICMD could be active for a Class 1 command if the WTS701 is still interpreting the previous command.

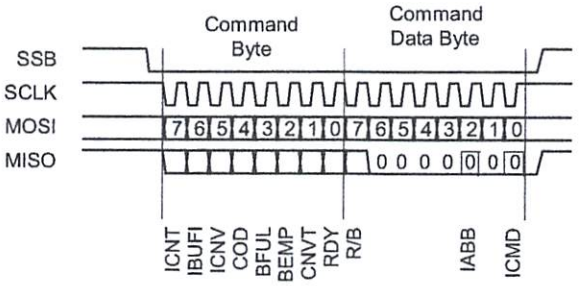


Figure 12. Type I SPI Transaction.

Type II – Two Word Transactions that Receive Data

Type II transactions are four byte transactions that read out the byte count register. As these commands are all Class0, ICMD will never be active and R/$\overline{B}$ will never occur.

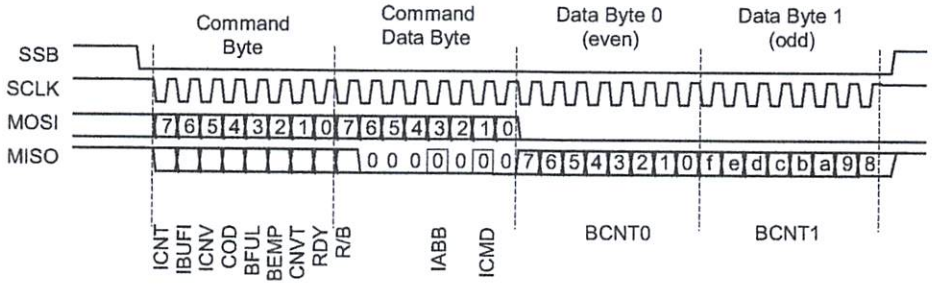


Figure 13. Type II SPI Transaction.

Type III – Transactions that send data

Type III transactions send data to the WTS701. If the data rate exceeds the ability of the WTS701 to read data from the input FIFO or if the internal data queue becomes full then the R/$\overline{B}$ line will handshake a pause in the SPI transaction. The host can either:

- Wait for R/$\overline{B}$ to return HIGH then continue sending data
- Terminate the transaction and try sending data later

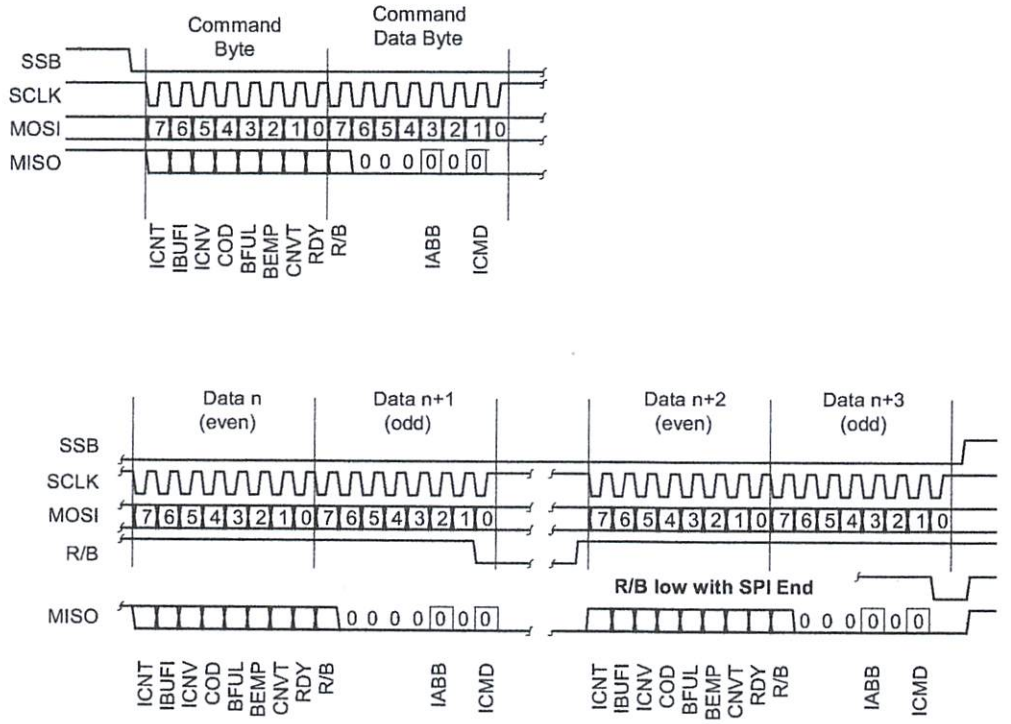


Figure 14. Type III SPI Transaction.

Type IV – Transactions reading data

Type IV transactions read data from the WTS701. Because of the latency required for the WTS701 to place data in the output register, R/$\overline{B}$ must be monitored.

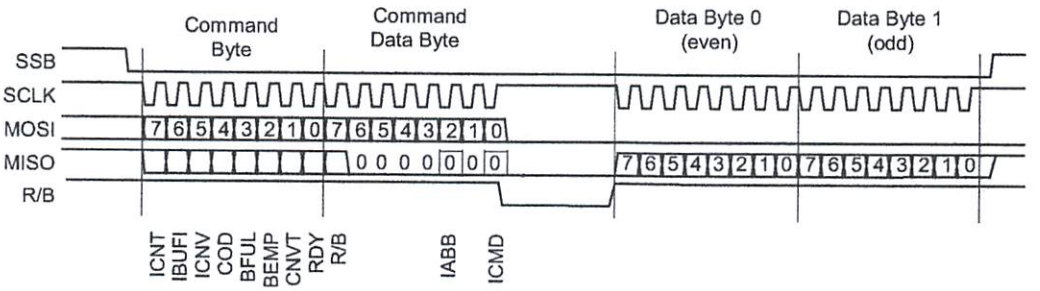


Figure 15. Type IV SPI Transaction.

7.6 CODEC INTERFACE

The WTS701 processor supports analog and digital telephony in various configurations. The WTS701 can be used in digital environments, along with a DSP that controls a CODEC. Therefore, the WTS701 is configured to operate in slave mode, where the control signals are provided by an external source, which is usually the DSP. It supports a variety of single channel CODECs, examples of which are listed in Table 12.

The CODEC interface is designed to send data in short frame format as well as long frame format. The channel width is 13 or 16 bits linear, the precision of the output is 10 bits. The operation mode of the CODEC is configured by the COD configuration register and SCOD command. See subsection 7.4.3 for details.

- The CODEC can be configured to transmit data in the unsigned or 2's Complement mode (see Table 13 for details).
- The CODEC responds to both the Long and Short sync format (see Figure 16 and Figure 17).
- The CODEC can be configured to tristate the VDX line after 13 or 16 bits.

Table 12. Supported CODEC Examples.

Manufacturer	CODEC Device Name	Characteristics	Operating Voltage	Conversion Type	Data Format
OKI	ML7041	Single codec	3 V	14-bit linear	2s Complement
OKI	MSM7716	Single codec	3 V	14-bit linear	2s Complement
OKI	MSM7732-011	Single codec	3 V	14-bit linear	2s Complement
Motorola	MC145483	Single codec	3 V	13-bit linear	2's Complement
Lucent	T8538B	Quad codec	3.3 V	16-bit linear	2's Complement

Table 13. CODEC Transmission Modes.

Level	Signed Mode (2's Complement)			Unsigned Mode	
	Sign Bit (MSB)	13 Bit Mode	16 Bit Mode	13 Bit Mode	16 Bit Mode
+ve full scale	0	1111 1111 1000	1111 1111 1000 0000	1 1111 1111 1000	1111 1111 1100 0000
+1 LSB	0	0000 0000 1000	0000 0000 1000 0000	1 0000 0000 1000	1000 0000 0100 0000
Zero (ground)	0	0000 0000 0000	0000 0000 0000 0000	1 0000 0000 0000	1000 0000 0000 0000
-1 LSB	1	1111 1111 1000	1111 1111 1000 0000	0 1111 1111 1000	0111 1111 1100 0000
-ve full scale	1	0000 0000 0000	0000 0000 0000 0000	0 0000 0000 0000	0000 0000 0000 0000

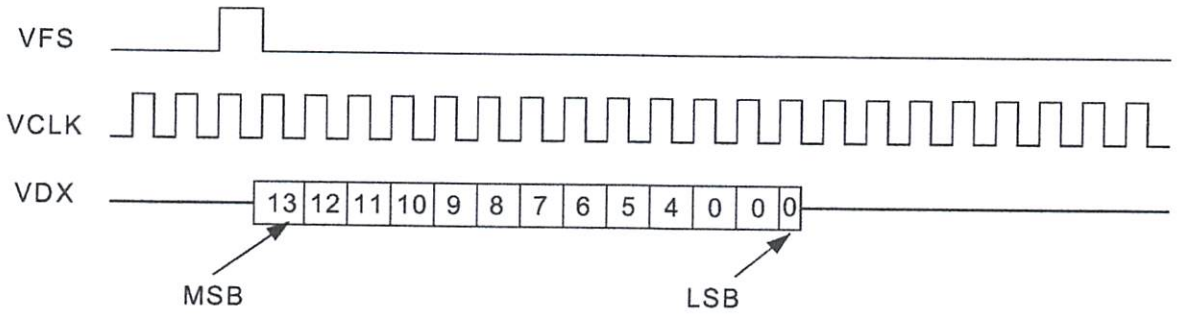


Figure 16. CODEC Protocol, 13 bit, Short Frame Sync.

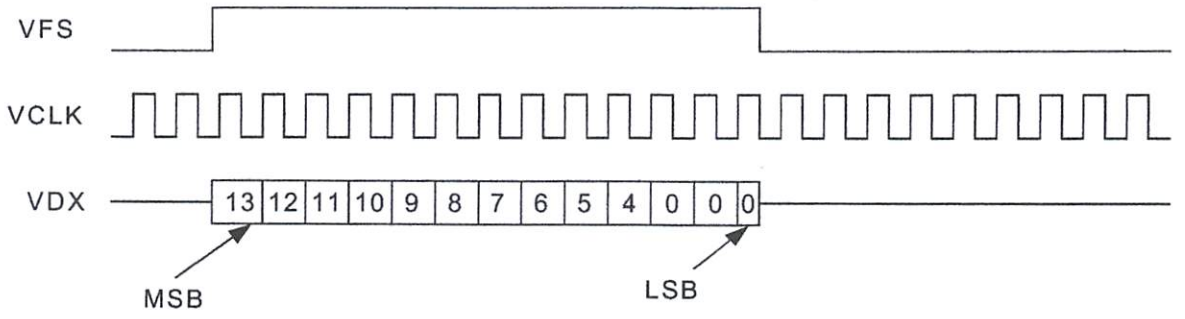


Figure 17. CODEC Protocol, 13 bit, Long Frame Sync.

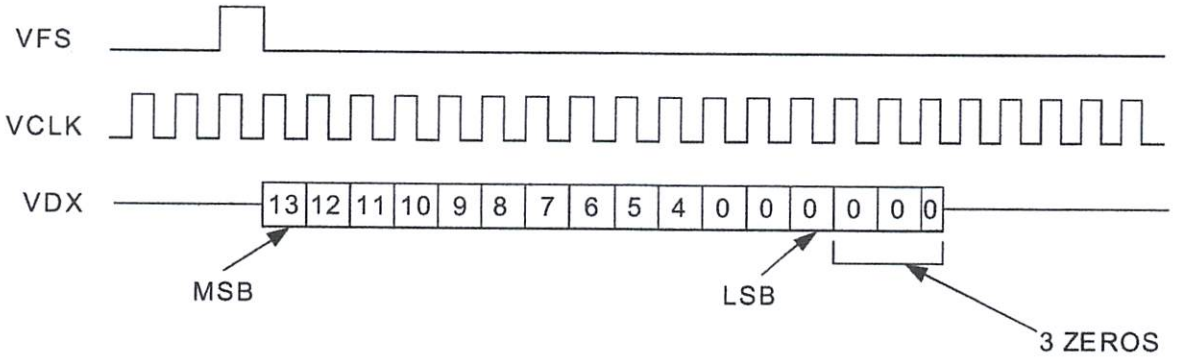


Figure 18. CODEC Protocol, 16 bit, Short Frame Sync.

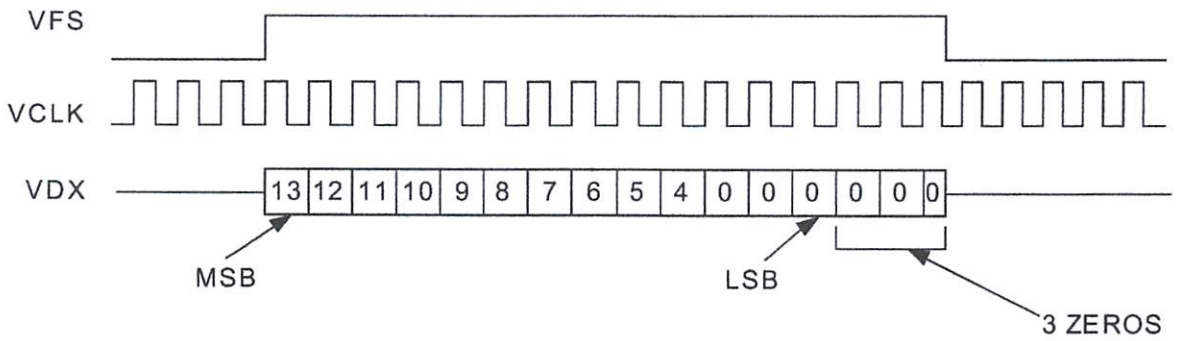


Figure 19. CODEC Protocol, 16 bit, Long Frame Sync.

7.7 CONTROL CHARACTERS

The WTS701 allows receiving control characters embedded in the text sent in the Convert command to better emphasize word or alter meaning of sentence. The control characters supported are for phonetic alphabet playback, speed, volume modification and case sensitivity behavior.

7.7.1 Phonetic Alphabet Playback

The WTS701 uses an intermediate phonetic translation represented as an alphabet that represents phonemes and stress for each input word. This feature allows the text sent to the WTS701 to consist of a combination of ASCII characters as well as phonetic alphabet. This capability offers the flexibility to send words already processed for phonetic representation, achieving the desired pronunciation.

Phonetic strings can be sent directly to the WTS701. This can be done by embedding phoneme strings in the text stream for conversion. To embed a phoneme string, the string must be preceded by a control-P (^P, ASCII 0x10) character and terminated by a space character.

For example:

"The quick ^Pbr1Wn fox."

The following table lists the phoneme symbols acceptable by the WTS701E (English software version). As the acceptable phoneme symbols are language dependent, please refer to the specific language User's Guide for details regarding characters accepted and other development considerations.

Table 14. Acceptable Phoneme Symbols.

Vowels			Consonants		
Phoneme	**Hex Value**	**Example**	**Phoneme**	**Hex Value**	**Example**
i	0x69	b*ea*t	p	0x70	*p*et
I	0x49	b*i*t	t	0x74	*t*en
e	0x65	b*a*it	k	0x6b	*k*it
E	0x45	b*e*t	b	0x62	*b*et
@	0x40	b*a*t	d	0x64	*d*ebt
u	0x75	b*oo*t	g	0x67	*g*et
U	0x55	b*oo*k	h	0x68	*h*at
o	0x6f	b*oa*t	f	0x66	*f*at
c	0x63	b*ough*t	T	0x54	*th*ing
a	0x61	B*o*b	D	0x44	*th*at
A	0x41	b*u*t	s	0x73	*s*at
R	0x52	b*ur*r	S	0x53	*sh*ut
O	0x4f	b*oy*	v	0x76	*v*at
Y	0x59	b*uy*	z	0x7a	*z*oo
W	0x57	d*ow*n	Z	0x5a	a*z*ure
x	0x78	*a*bout	y	0x79	*y*ou
X	0x58	ros*e*s	w	0x77	*w*it
			r	0x72	*r*ent
			l	0x6c	*l*et
			m	0x6d	*m*et
			n	0x6e	*n*et
			G	0x47	si*ng*
			C	0x43	*ch*u*rch*
			J	0x4a	*J*u*dge*
			P	0x50	B*tt*er*
			Q	0x51	Wri*tt*en*

Note that each phoneme is represented by exactly one character and each vowel is preceded by a *stress symbol.*

* Female English only.

Numbers 1 and 0 represent stress: each word has a single 1 stress, representing the main stress of the word; all other syllables have 0 stress.

Examples:

Input	**Phonetic translation**
hi.	h1Y (phoneme /h/, followed by a 1-stress vowel phoneme Y)
test	t1Est
testing	t1Est0IG

Special Characters in Text Input String

These characters inserted into the ASCII text are used to modify the behavior for special circumstances.

0x10 ^P This control flag indicates a phoneme string follows immediately (e.g. '^Ppr0Ez0Intles0In').

0x11 ^Q Pauses with variant length can be added within a sentence by using the '^QX' flag. 'X' is an integer which indicates the pause duration ('X' is 0.1 sec per unit). For instance, '^Q10' will add in a 1-second pause.

It is important to remember that all the text following these special characters will be subjected to the same special conditions. If a ^V+ command precedes a word then all the following words will be spoken louder. If a single word is to be emphasized then the ^V+ must be ahead of the word and a ^V- must follow.

7.7.2 Speed Change

The rate of speech can be changed by sending an SPI command to modify the speed or by adding a control character to control the speed in real-time.

0x13 ^S Speed Change Flag. A space character is required after the control characters before the input text string.

- ^S+: Increase speed by 1 (e.g. '^S+ Hello world').
- ^S-: Decrease speed by 1.
- ^SX: Set the speed to X. X starts from 0 to 4 (e.g. '^S1 Hello world').

  Any number that is greater than 4 will be set to 4.

It's the user's responsibility to verify the WTS701 speed setting before sending a control character and/or SPI command that modify speed.

7.7.3 Volume Change

The speech volume can be changed by sending SPI command to modify the speed or by adding control character to control the volume in real-time.

0x16 ^V Volume Change Flag. A space character is required after the control characters before the input text string.

- ^V+: Increase volume by 1 (e.g. '^V+ Hello world').
- ^V-: Decrease volume by 1.
- ^VX: Set the volume to X. X starts from 0 to 7 (e.g. '^V1 Hello world').

  Any number that is greater than 7 will be set to 0.

It's the user's responsibility to verify the WTS701 volume setting before sending a control character and/or SPI command that modify volume.

7.7.4 Case Sensitivity

The way upper/lower case is handled can be changed by adding a control character in the text sent to control the case sensitivity behavior in real-time.

0x15 ^U All-Uppercase Word (all CAPs) Control Flag. This flag controls interpretation of strings with all uppercase letters. A space character is required after the control characters before the acutual input text string.

- ^U0: This is the default setting. Some all-uppercase words (all CAPs) are spelled out, but others are treated as ordinary words, abbreviations, Roman numerals, ect. (e.g. '^U0 IBM equals 'i b m').
- ^U1: Avoid spelling out all-uppercase words. Any all-uppercase word in the input string will NOT be spelled out unless the system determines that the word is not pronounceable. In this mode, there is no abbreviation support (e.g. '^U1 NOKIA' equals 'nokia').
- ^U2: Spell out all words, regardless of case. All words in the input sting are spelled out in this mode. There is no abbreviation support in this mode (e.g. '^U2 NOKIA' equals 'n o k i a').

  ^U3: Force the all-capital words/strings to be spelled out regardless of string length. For instance, '^U3 HELLO' will be pronounced as 'H', 'E', 'L', 'L', 'O'.

7.7.5. Pause Control

Pause control flage: '^QX': Pauses with variant length can be added within a sentence by using the '^QX' flag. 'X' is an integer which indicats the pause duration ('X' is 0.1 sec per unit). For instance, '^Q10' will add in a 1-second pause.

7.8 CUSTOMIZING ABBREVIATIONS

The WTS701 has support for entering and using custom abbreviations in addition to the general abbreviation table supported internally by the WTS701. There are 2K bytes of flash memory reserved for this purpose. After the WTS701 internal software has been initially programmed, this entire area is free and available for custom abbreviations.

The commands associated with custom abbreviations are:

Command	Command Byte	Command Data Byte	
ABBR_ADD	0xaf	0x00 + abbreviation data.	Adds a new abbreviation to the abbreviation table in the WTS701 See next page for the format of the abbreviation data.
ABBR_DEL	0x83	0x00+ abbreviation data.	Deletes an existing abbreviation from the abbreviation table in the WTS701. See next page for the format of the abbreviation data.
ABBR_NUM	0xc8	0x00 + 0x00 + 0x00.	Returns the number of abbreviation currently active in the abbreviation table of the WTS701.
ABBR_MEM	0xc7	0x00 + 0x00 + 0x00.	Returns the number of free bytes in the abbreviation table of the WTS701.
ABBR_RD	0xc9	0x00 + 2048 0x00s.	Returns the abbreviation table contents from the WTS701. See next page for the format of the abbreviation table data.
ENTER_RRSM	0x0c	0x00	Causes the xdata and code store memory to swap spaces. The WTS701 begins to execute code previously stored into xdata after this command.

7.8.1 Abbreviation Data Format

The format of the abbreviation data that is sent with the ABBR_ADD and ABBR_DEL commands is:

XXX + "," + YYYY + ";".

XXX - the abbreviation characters.

"," – comma.

YYYY – abbreviation text.

";" – semi-colon.

Example: TTS,text to speech; After this is added using the ABBR_ADD command, when the text "TTS" is sent as part of the convert data, the WTS701 will speak "text to speech" instead of T T S.

Note: when deleting an abbreviation, the abbreviation text is optional.

To delete the TTS example, only "TTS,;" is necessary.

7.8.2 Abbreviation Table Format

The format of the abbreviation table returned with the ABBR_RD command is:

Abbreviation entry - Marker + Count + XXX + 0x00 + YYYY + 0x00.

Marker – The marker will be either 0xfe for active abbreviation or 0xfc for a deleted abbreviation.

Count – The byte count for this entry including the Marker, Count, XXX, YYYY, and zeros.

XXX – the abbreviation characters.

0x00 – Null terminator.

YYYY – abbreviation text.

0x00 – Null terminator.

The unused data are always 0xff.

7.8.3 Command Execution

ABBR_NUM & ABBR_MEM - These commands are executed by sending the command and command data, waiting for R/$\overline{B}$ to be ready, then receiving two bytes from MISO. The first byte received is the MSB, and the second is LSB.

ABBR_RD – This command is executed by sending the command and command data, waiting for R/$\overline{B}$ to be ready, then receiving 2048 characters (the entire abbreviation table).

ABBR_ADD & ABBR_DEL – These commands are executed by sending the command and command data, followed by the abbreviation data formatted as described in subsection 7.8.1. When the WTS701 is ready for the next step, it will generate an IABB interrupt. After the interrupt, send the ENTER_RSSM (0x0c + 0x00) command. After issuing the command wait for 100ms. After the timeout, the WTS701 will have programmed the new abbreviation entry and be ready to accept more commands. Adding or removing an abbreviation will reset the configuration registers to their default values.

After abbreviation entry deletion, the abbreviation entry is only deleted from the table and not used, however it still holds memory space. The only way to free all memory will be to reprogram the WTS701 firmware into the device.

7.9 DEVICE PROGRAMMING

The WTS701 is fully programmable to support different available languages or different voices that can be loaded to the device whenever the user wishes to do so. The language or the voice module should be stored externally and transmitted to the WTS701 processor with regards to a specific protocol defined in this section.

Programming the WTS701 consists of downloading a binary executable to the processor code memory and a digitized analog speech corpus to the non-volatile analog multi-level storage (MLS). Winbond will supply code as a set of ASCII readable data files. The information will be provided to qualified customers upon request.

7.10 TEXT-TO-SPEECH PROCESSOR COMMMANDS – QUICK REFERENCE TABLE.

Status Commands										
Command			Description	Opcode Hex		Previous State	Result State	Command Parameters		Return Value
Name	Class	Type		Command byte	Command data			Description	Bytes	Description
RDST	0	II	Read Status	04	00	Idle, Convert, Power Down	No change	None	-	Byte count
RINT	0	II	Read Interrupt	06	00	Idle, Convert	No change	None	-	Byte count
RVER	0	II	Read version	12	00	Idle, Convert	No change	None	-	Hw_ver, Sw_ver

System Commands											
Command			**Description**	**Opcode Hex**		**Previous State**	**Result State**	**Command Parameters**		**Return Value**	
Name	Class	Type		Command byte	Command data			Description	Bytes	Description	Bytes
PWUP	0	I	Exit Power Down mode	02	00	Power Down	Idle	None	-	None	-
PWDN	1	I	Go To Power Down mode	40	00	Idle, Convert Wait	Power Down	None	-	None	-
RST	0	I	Reset	10	00	Idle, Convert Wait, Power down	Power Down	None	-	None	-

Winbond Electronics Corp.

Synthesis Commands

Command			Description	Opcode Hex		Previous State	Result State	Command Parameters		Return Value	
Name	Class	Type		Command byte	Command data			Description	Bytes	Description	Bytes
CONV	2	III	Convert text	81	00	Idle, Wait	Convert	Text data	N	None	-
PAUS	1	I	Pause conversion	49	00	Convert	No change	None	-	None	-
RES	1	I	Resume conversion	4A	00	Convert	No change	None	-	None	-
ST	1	I	Stop conversion	4B	00	Convert	Wait	None	-	None	-
FINW	1	I	Finish word	4D	00	Convert	Wait	None	-	None	-
FIN	1	I	Finish	4C	00	Convert	Wait	None	-	None	-
VLUP	1	I	Volume up	53	00	Idle, Convert Wait	No change	None	-	None	-
VLDN	1	I	Volume down	54	00	Idle, Convert Wait	No change	None	-	None	-
SPUP	1	I	Speed up	55	00	Idle, Convert Wait	No change	None	-	None	-
SPDN	1	I	Speed down	56	00	Idle, Convert Wait	No change	None	-	None	-
IDLE	1	I	Switch to Idle state	57	00	Wait, Convert	Idle	None	-	None	-

Configuration Commands										
Command			**Description**	**Opcode Hex**		**Previous State**	**Result State**	**Command Parameters**		**Return Value**
Name	Class	Type		Command byte	Command data			Description	Bytes	Description
RREG	3	IV	Read configuration register	C0	Register Number	Idle, Convert	No change	None	-	Register value, dummy byte
SCOM	1	I	Set COM register	4E	Value	Idle, Wait	No Change	None	-	None
SCOD	1	I	Set COD register	4F	Value	Idle, Wait	No Change	None	-	None
SAUD	1	I	Set AUD register	50	Value	Idle, Wait	No Change	None	-	None
SVOL	1	I	Set VOL register	51	Value	Idle, Wait	No Change	None	-	None
SSPD	1	I	Set SPD register	52	Value	Idle, Wait	No Change	None	-	None
SCLC	1	I	Set CLC register	14	Value	Idle, Wait	No Change	None	-	None
SPTC	1	I	Set Speech Pitch	77	Value	Idle, Wait	No Change	None	-	None

Customization Commands										
Command			Description	Opcode Hex		Previous state	Result State	Command Parameters		Return Value
Name	Class	Type		Command byte	Command data			Description	Bytes	Description
ABBR_ NUM	3	IV	Return number of abbreviation entries	C8	00	Idle	No change	None	-	Num_of_entries
ABBR_ RD	3	IV	Read abbreviation table	C9	00	Idle	No change	None	-	Abbreviation table entries
ABBR_ MEM	3	IV	Return abbreviation memory	C7	00	Idle	No change	None	-	Available_mem
ABBR_ ADD	2	III	Add abbreviation	AF	00	Idle	No change	Abbreviation information	N	None
ABBR_ DEL	1	I	Delete abbreviation entry	83	00	Idle	No change	Abbreviation information	N	None
ENTER _RRS M	0	I	Swap memory	0C	00	Idle	No change	None	-	None

7.10.1 Text Input Format

The following table lists the ASCII characters acceptable by the WTS701E (English software version). Please refer to the specific language User's Guide for more details regarding characters accepted and other development considerations.

Note: Unexpected behavior may occur if the input text contains characters that are not defined in this ASCII table.

Table 15. Allowable ASCII Characters.

0x0		0x20	Space	0x40	@	0x60	
0x1		0x21	!	0x41	A	0x61	a
0x2		0x22		0x42	B	0x62	b
0x3		0x23	#	0x43	C	0x63	c
0x4		0x24	$	0x44	D	0x64	d
0x5		0x25	%	0x45	E	0x65	e
0x6		0x26	&	0x46	F	0x66	f
0x7		0x27	' (apostrophe)	0x47	G	0x67	g
0x8		0x28	(	0x48	H	0x68	h
0x9		0x29	)	0x49	I	0x69	i
0xa		0x2a	*	0x4a	J	0x6a	j
0xb		0x2b	+	0x4b	K	0x6b	k
0xc		0x2c	, (comma)	0x4c	L	0x6c	l
0xd		0x2d	- (dash)	0x4d	M	0x6d	m
0xe		0x2e	. (period)	0x4e	N	0x6e	n
0xf		0x2f	/ (slash)	0x4f	O	0x6f	o
0x10	^P	0x30	0	0x50	P	0x70	p
0x11	^Q	0x31	1	0x51	Q	0x71	q
0x12		0x32	2	0x52	R	0x72	r
0x13	^S	0x33	3	0x53	S	0x73	s
0x14		0x34	4	0x54	T	0x74	t
0x15	^U	0x35	5	0x55	U	0x75	u
0x16	^V	0x36	6	0x56	V	0x76	v
0x17		0x37	7	0x57	W	0x77	w
0x18		0x38	8	0x58	X	0x78	x
0x19		0x39	9	0x59	Y	0x79	y
0x1a	EOT	0x3a	: (colon)	0x5a	Z	0x7a	z
0x1b		0x3b		0x5b	] (right bracket)	0x7b	
0x1c		0x3c		0x5c	\ (back slash)	0x7c	
0x1d		0x3d	=	0x5d	[(left bracket)	0x7d	
0x1e		0x3e		0x5e		0x7e	
0x1f		0x3f	?	0x5f	_ (under score)	0x7f	

7.10.2. Buffer length limit

The max. character length of a white-space-bounded string is 53. The exceeding characters will be truncated.

7.10.3. Undefined characters

All the undefined characters will be deleted (prior to the word pornunciation process). The difined characters range from '0x00' to '0x7A' excluding '0x22', '0x3E', and '0x60'.

8. TIMING WAVEFORMS

8.1 SPI TIMING DIAGRAM

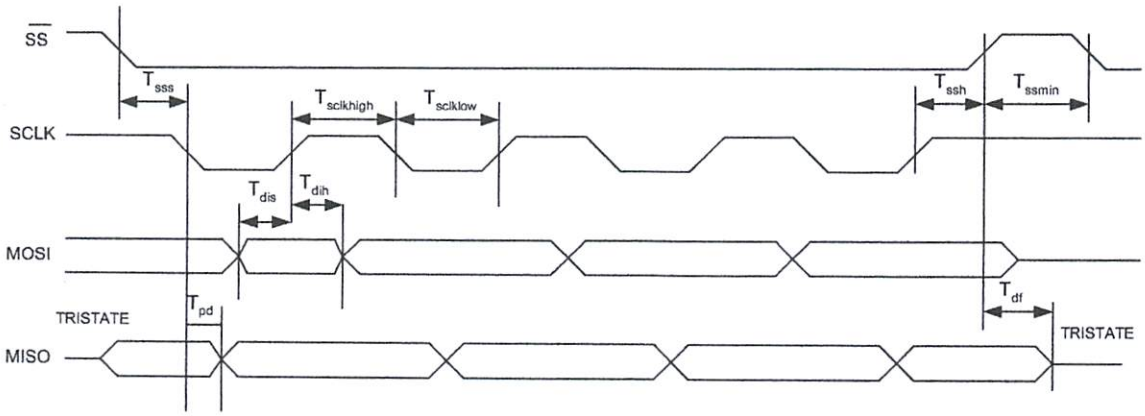


Figure 20. SPI Timing Specification.

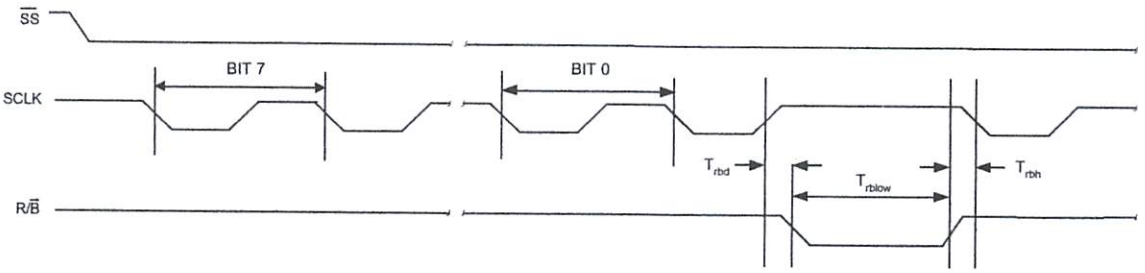


Figure 21. SPI R/$\overline{B}$ Timing.

Table 16. SPI Timing Parameters (see Figure 20 and Figure21)

Symbol	Parameters	Min	Typ(7)	Max	Units	Conditions
T_{SSS}	$\overline{SS}$ Setup Time	100			ns	
T_{SSH}	$\overline{SS}$ Hold Time	100			ns	
T_{DIS}	Data in Setup Time	50			ns	
T_{DIH}	Data in Hold Time	50			ns	
T_{PD}	Output Delay			100	ns	
T_{DF}	Output Delay to hiZ			100	ns	
T_{SSmin}	$\overline{SS}$ High	200			ns	
T_{SCKhi}	SCLK High Time	80			ns	
T_{SCKlow}	SCLK Low Time	80			ns	
T_{rbd}	Delay SCLK Hi to $R/\overline{B}$ low			80	ns	
T_{rblow}	$R/\overline{B}$ low time	0			ms	
T_{rbh}	SCLK hold time from $R/\overline{B}$ \ High	0				
F_O	CLK Frequency			5000	kHZ	

8.2 CODEC TIMING DIAGRAMS

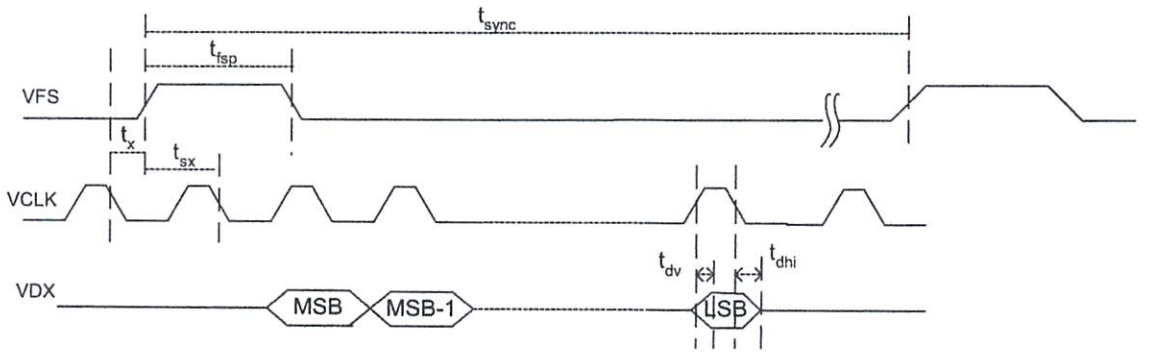


Figure 22. CODEC Timing — Short Frame Sync.

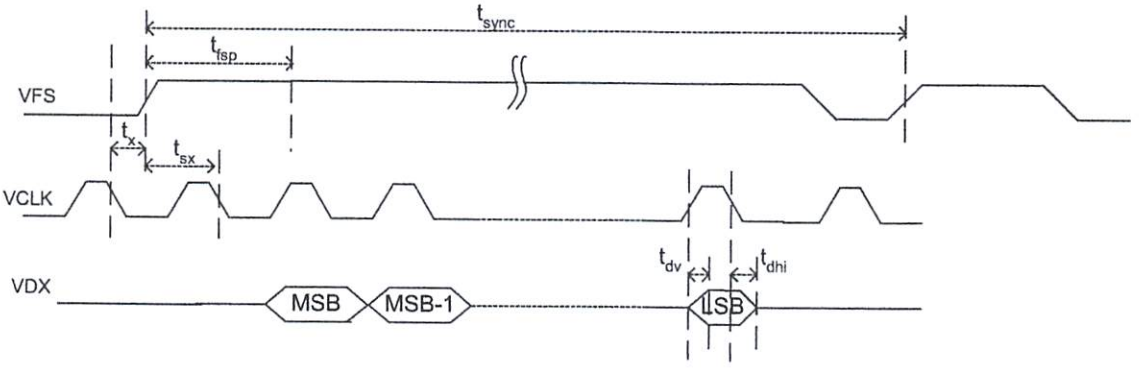


Figure 23. CODEC Timing -- Long Frame Sync.

Table 17. CODEC Timing Parameters (see Figure 22 and Figure 23)

Symbol	Parameters	Min	Typ[7]	Max	Units	Conditions
T_{clk}	Bit clock frequency	128		2048	kHz	
T_{sync}	Frame Sync. Frequency		8		kHz	
D_C	Clock Duty Cycle	45	50	55	%	
T_{ir}	Rise Time			50	ns	All digital inputs
T_{if}	Fall Time			50	ns	All digital inputs
T_{fsp}	Frame Sync. Pulse Width			100	ns	VFS
T_{rs}	Receive Sync. Timing	20			ns	VCLK to VFS
T_{sr}	Receive Sync. Timing	80			ns	VFS to VCLK
T_{dv}	Output Delay Time for VDX Valid	10		140	ns	VCLK to VDX
T_{dhi}	Output Delay Time for VDX High Impedance	10		140	ns	VCLK to VDX

9. ABSOLUTE MAXIMUM RATINGS

Table 18. Absolute Maximum Ratings (Packaged Parts) [1]

Condition	Value
Junction temperature	150°C
Storage temperature range	-65°C to +150°C
Voltage Applied to any pin	(V_{SS} - 0.3V) to (V_{CC} + 0.3V)
Voltage applied to any pin (Input current limited to +/-20 mA) [2]	(V_{SS} – 1.0V) to (V_{CC} + 2.2V)
Lead temperature (soldering – 10 seconds)	300°C
V_{CC} - V_{SS}	-0.3V to +7.0V

Table 19. Operating Conditions (Packaged Parts).

Condition	Value
Commercial operating temperature range [3]	0°C to +70°C
Extended operating temperature[2]	-20°C to +70°C
Industrial operating temperature[2]	-40°C to +85°C
Supply voltage (V_{CC})[4]	+2.7V to +3.3V
Ground voltage (V_{SS})[5]	0V

[1] Stresses above those listed may cause permanent damage to the device. Exposure to the absolute maximum ratings may affect device reliability. Functional operation is not implied at these conditions.

[2] All input pins except for $\overline{CS}$ signal, which is 3V tolerant ONLY.

[3] Case Temperature.

[4] $V_{CC} = V_{CCA} = V_{CCD}$.

[5] $V_{SS} = V_{SSA} = V_{SSD}$.

10. ELECTRICAL CHARACTERISTICS

(V_{CC} = 3.3V, V_{SS} = 0V, T_A = 0 to 70 °C)

Table 20. General Parameters.

PARAMETER	SYMBOL	TEST CONDITIONS	SPEC			UNIT
			MIN[6]	TYP[7]	MAX.	
Input LOW Voltage	V_{IL}				V_{CC} x 0.2	V
Input HIGH Voltage	V_{IH}		V_{CC} x 0.8			V
Output LOW Voltage	V_{OL}	I_{OL} = 10 µA			0.4	V
R/$\overline{B}$, $\overline{INT}$ Output LOW Voltage	V_{OL1}	I_{OL} = 1 mA			0.4	V
Output HIGH Voltage	V_{OH}	I_{OL} = -10 µA	V_{CC} – 0.4			V
V_{CC} Current (Operating)	I_{CC}					
- Convert		No Load[8]			50	mA
- Idle		No Load[8]			20	mA
- CODEC		No Load[8]			20	mA
- Speaker		No Load[8]			15	mA
V_{CC} Current (Standby)	I_{SB}	(8)		1	50	µA
Input Leakage Current	I_{IL}				+/-1	µA

[6] All Min/Max limits are guaranteed by Winbond via electrical testing or characterization. Not all specifications are 100% tested.

[7] Typical values are T = 25°C and V_{CC} = 3.0V, timing measured at 50% levels.

[8] V_{CCA} and V_{CCD} summed together.

Table 21. Speaker Driver Specifications.

PARAMETER	SYMBOL	TEST CONDITIONS	SPEC			UNIT
			MIN.	TYP.	MAX.	
SP+/- Output Voltage (HIGH Gain Setting)	V_{SPHG}	Peak-to-Peak, differential load = 150Ω			3.6	V
SP+/- Output Load Imp. (LOW Gain)	R_{SPLG}		8			Ω
SP+/- Output Load Imp. (HIGH Gain)	R_{SPHG}		70	150		Ω
SP+/- Output Load Cap.	$\overline{CS}_P$				100	pF
SP+/- Output Bias Voltage (Analog Ground)	V_{SPAG}			1.2		V_{DC}
Speaker Output DC Offset	V_{SPDCO}	With AUXIN to Speaker, AUXIN AC coupled to V_{SSA}			100	mV_{DC}
Power Supply Rejection Ratio	PSRR	Measured with a 1 kHz, 100 ma sine wave input at V_{CC} and V_{CC} pins		-55		dB
Frequency Response (300-3400 Hz)	F_R	With 0TLP input to AUX IN, 6 dB setting (12)	-0.25		+0.25	dB
Power Output (LOW Gain Setting)	P_{UTLOG}	Differential load at 8Ω	23.5			mW_{RMS}

Table 22. AUXOUT Parameters.

PARAMETER	SYMBOL	TEST CONDITIONS	SPEC			UNIT
			MIN.	TYP.	MAX.	
AUXOUT – Maximum Output Swing	V_{ANAIUT}	5kΩ Load (AC Coupled)			1.0	V
Minimum Load Impedance	R_L		5			kΩ
Maximum Load Capacitance	C_L				100	pF
AUXOUT	V_{BIAS}			1.2		VDC

Table 23. Volume Control Parameters.

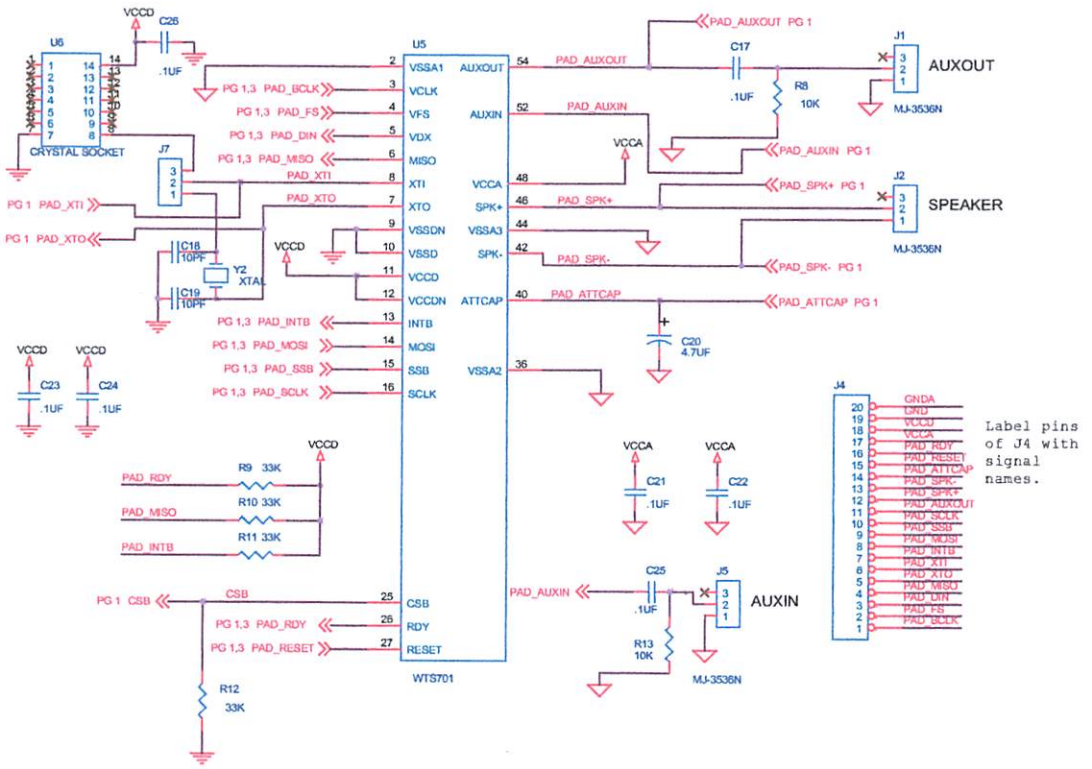
PARAMETER	SYMBOL	TEST CONDITIONS	SPEC			UNIT
			MIN.	TYP.	MAX.	
Output Gain	A_{OUT}	8 steps of 4 dB, referenced to output		-28 to 0		dB
Absolute Gain		AUXIN 1.0 kHz 0TLP, 6 dB gain setting measured differentially at SP+/-	-0.5		+0.5	dB

11. TYPICAL APPLICATION CIRCUIT

The following schematic diagrams are extracted from the WTS-ES701 evaluation board schematic.

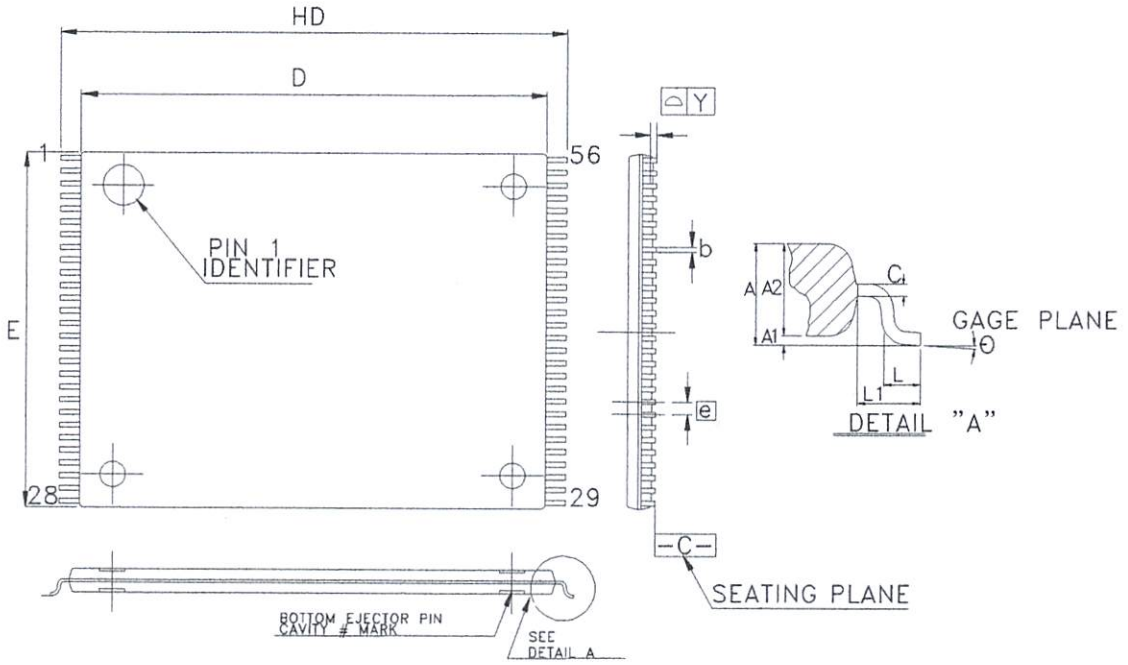
The evaluation system includes the following basic clusters:
WTS701 processor cluster working with 3.3V, including an 8-ohm speaker,
SPI connector to the host PC via the PC parallel port.

For more information about the evaluation system, please refer to the WTS-ES701 User's Guide.



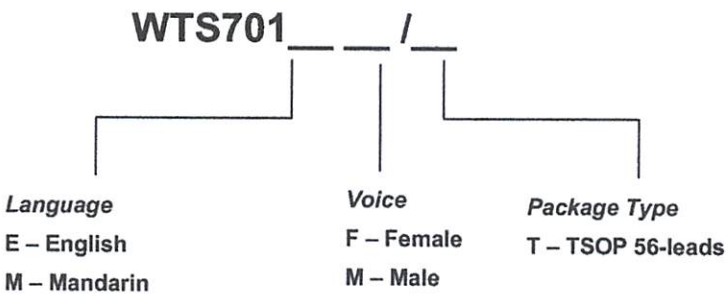
12. PACKAGE DRAWING AND DIMENSIONS

56 L TSOP(I) (14X20 MM)



Symbol	Dimension in mm			Dimension in inch		
	Min	Nom	Max	Min	Nom	Max
A	—	—	1.2	—	—	0.047
A1	0.05	—	0.15	0.002	—	0.006
A2	0.95	1.00	1.05	0.037	0.039	0.041
b	0.17	0.22	0.27	0.007	0.009	0.011
c	0.10	—	0.21	0.004	—	0.008
HD	20.00 BSC			0.787 BSC		
D	18.40 BSC			0.724 BSC		
E	14.00 BSC			0.551 BSC		
L	0.50	0.60	0.70	0.020	0.024	0.028
L1	0.80 REF			0.031 REF		
e	0.5 BSC			0.020 BSC		
θ	0°	3°	5°	0°	3°	5°
Y	—	—	0.10	—	—	0.004

13. ORDERING INFORMATION



For the latest product information, access Winbond's worldwide website at http://www.winbond-usa.com

14. VERSION HISTORY

VERSION	DATE	PAGE	DESCRIPTION
3.07	Apr. 2002	1-73	Initial issue
3.08	Jun. 2002	1-73	Improved package drawing, added illegal commands description, modified part number format
3.09	May 2003	all	Add crystal spec., description of buffer length limit & undefined characters, consonant phoneme characters (P & Q), ^U3 and ^Q control characters.

Headquarters
No. 4, Creation Rd. III
Science-Based Industrial Park,
Hsinchu, Taiwan
TEL: 886-3-5770066
FAX: 886-3-5665577
http://www.winbond.com.tw

Winbond Electronics Corporation America
2727 North First Street, San Jose,
CA 95134, U.S.A.,
TEL: 1-408-9436666
FAX: 1-408-5441798
http://www.winbond-usa.com

Winbond Electronics (Shanghai) Ltd.
27F, 299 Yan An W. Rd. Shanghai,
200336 China
TEL: 86-21-62365999
FAX: 86-21-62356998

Taipei Office
9F, No. 480, Pueiguang Rd
Neihu District,
Taipei, 114, Taiwan
TEL: 886-2-81777168
FAX: 886-287153579

Winbond Electronics Corporation Japan
7F Daini-ueno BLDG, 3-7-18
Shinyokohama Kohoky-ku,
Yokohama, 222-0033
TEL: 81-45-4781881
FAX: 81-45-4781800

Winbond Electronics (H.K.) Ltd.
Unit 9-15, 22F, Millennium City,
No. 378 Kwun Tong Rd.,
Kowloom, Hong Kong
TEL: 852-27513100
FAX: 852-27552064