RC Systems
now you're talking!

RC8650
voice synthesizer

DoubleTalk RC8650

CMOS, 3.3 Volt / 5 Volt
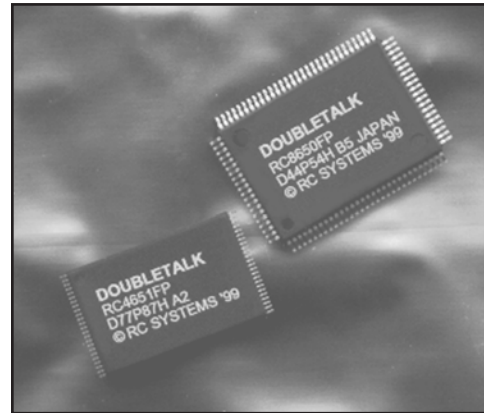## Voice Synthesizer Chipset

RoHS COMPLIANT

GENERAL DESCRIPTION

The RC8650 is a versatile voice and sound synthesizer, integrating a text-to-speech (TTS) processor, real time and prerecorded audio playback, musical and sinusoidal tone generators, telephone dialer and A/D converter, into an easy to use chipset. Using a standard serial or 8-bit bus interface, virtually any ASCII text can be streamed to the RC8650 for automatic conversion into speech by the TTS processor. The audio playback modes augment the TTS processor for applications requiring very high voice quality and a relatively small, fixed vocabulary, or applications requiring special sounds or sound effects. The audio output is delivered in both analog and digital PCM audio formats, which can be used to drive a speaker or digital audio stream.

The RC8650's integrated TTS processor incorporates RC Systems' DoubleTalk™ TTS technology, which is based on a unique voice concatenation technique using real human voice samples. The DoubleTalk TTS processor also gives the user unprecedented real-time control of the speech signal, including pitch, volume, tone, speed, expression, articulation, and so on.

Up to 3.5 MB of nonvolatile memory is included in the RC8650 for the storage and on-demand playback of up to 15 minutes of prerecorded messages and sound effects. A programmable "greeting" message can be stored that is automatically played whenever the RC8650 is powered up, allowing a custom message to be played or the RC8650's default settings to be reconfigured. A user-program-

mable dictionary allows the pronunciation of virtually any character string to be redefined, or even trigger the playback of tones, prerecorded messages and sounds based on specific input patterns. All of these features can be programmed and updated via a standard serial port, even in the field after the RC8650 has been integrated into the end-product.

The RC8650 is comprised of two surface-mounted devices. Both operate from either a +3.3 V or +5 V supply and consume very little power. Most applications require only the addition of a lowpass filter/audio power amplifier to implement a fully functional system.

FUNCTIONAL BLOCK DIAGRAM

RC Systems
now you're talking!
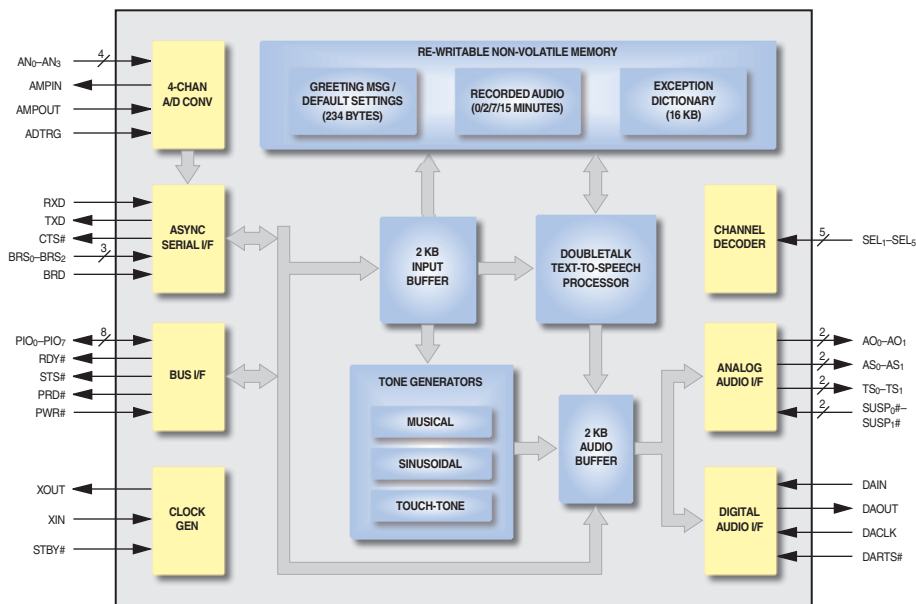
FEATURES

- Integrated text-to-speech processor:
  - High voice quality, unlimited vocabulary
  - Converts any ASCII text into speech automatically
  - Capable of very high reading rates
  - Add/modify messages by simply editing a text file
  - On-the-fly control of speed, pitch, volume, etc.
- Playback of sound files:
  - Real-time PCM and ADPCM
  - Prerecorded on chip, up to 15 minutes
- Tone generation:
  - Three voice musical
  - Dual sinusoidal
  - DTMF (Touch-Tone) dialer
- On-chip A/D converter:
  - Four channels, 8-bit resolution
  - One-shot, continuous, single sweep, and continuous sweep modes of operation
  - Software and hardware triggering
  - Support for external op amp
- Analog and digital audio outputs
- Stop, pause, and resume controls
- Standard serial (UART) and bus interfaces
- User programmable greeting and default settings
- Flexible user exception dictionary:
  - Change the pronunciation of any input string based on spelling and context
  - Convert encrypted data into meaningful messages
  - Trigger tone generation, recorded message playback, voice parameter changes
- In-circuit, field programmable
- 2 KB input buffer for virtually no-overhead operation
- Available in 3.3 V and 5 V versions
- Low power (typ @ 3.3 V):
  - 4 mA active
  - 700 µA idle
  - 0.7 µA standby

APPLICATIONS

- Robotics
- Talking OCR systems
- ATM machines
- Talking pagers and PDAs
- GPS navigation systems
- Vending and ticketing machines
- Remote diagnostic reporting
- Dial-up information systems
- Handheld barcode readers
- Electronic test and measurement
- Security systems
- Aids for the orally or visually disabled
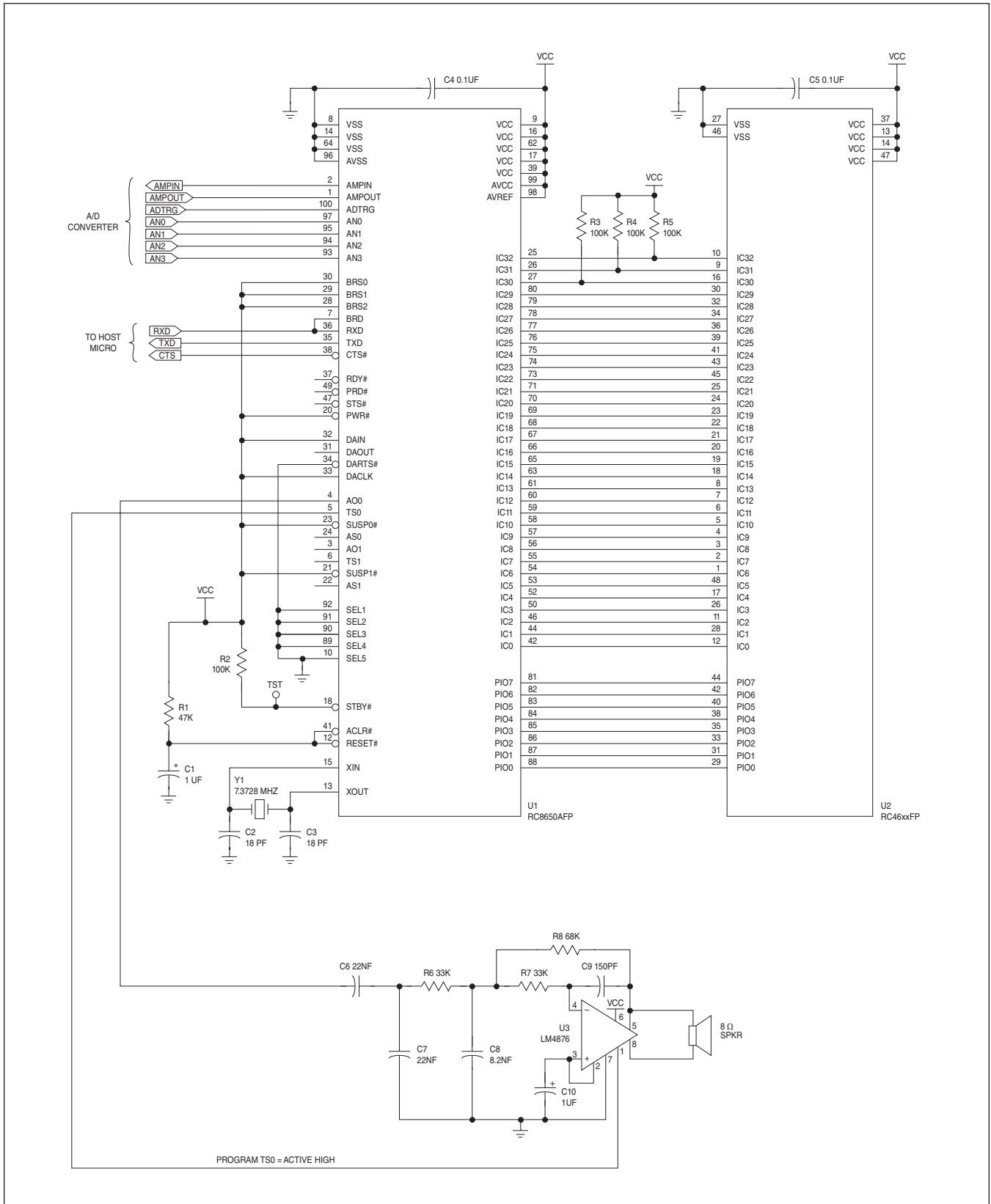- Meeting federal ADA requirements

RC8650 Product Summary

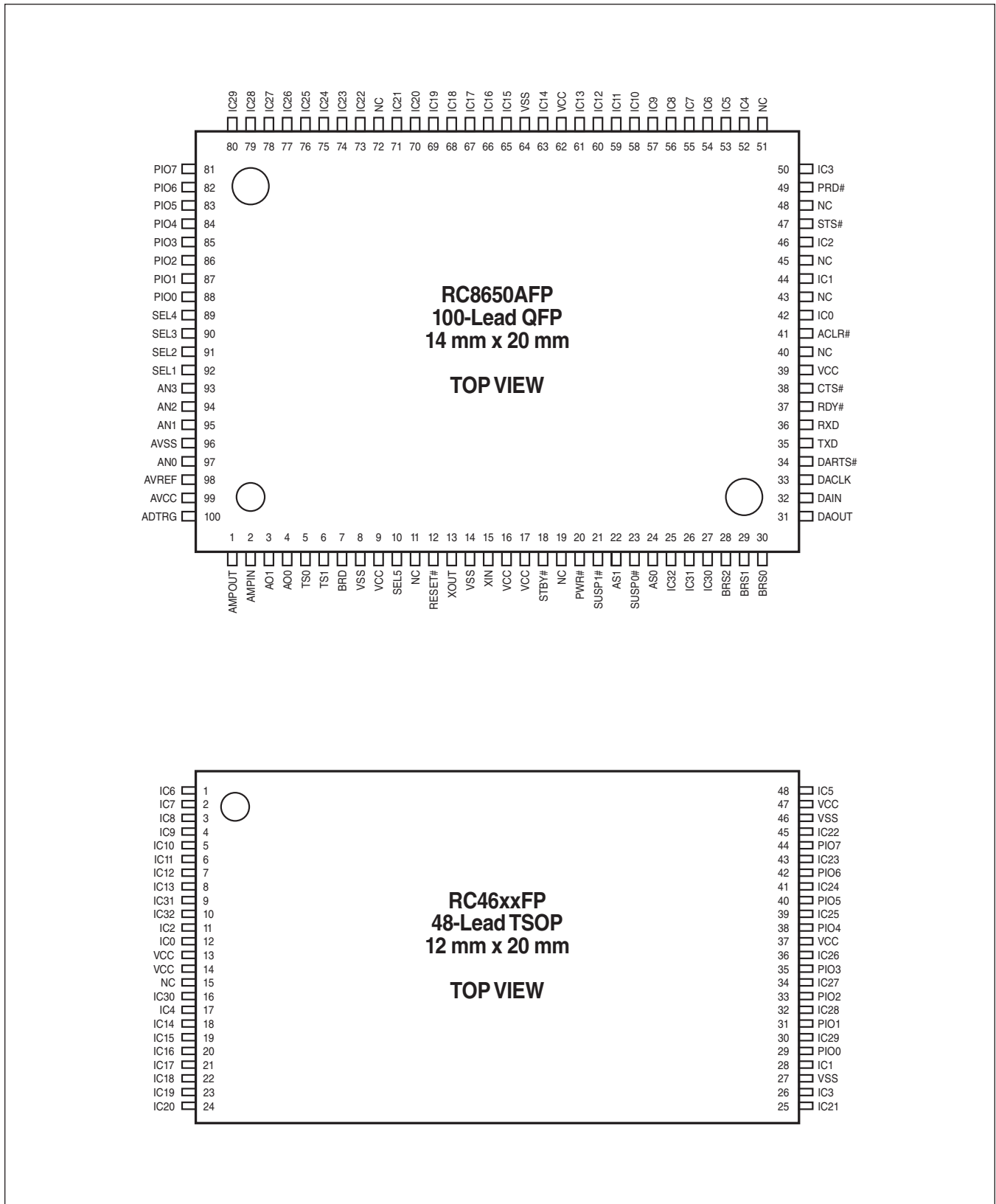| Order Number	Recording Capacity *	Operating Voltage
RC8650F1C	2 min	5 V
RC86L50F1I	2 min	3.3 V
RC86L50F3I	15 min	3.3 V

* Based on 8 kHz sampling rate with ADPCM encoding

See Ordering Information for a complete list of package and temperature options.

TYPICAL APPLICATION CIRCUIT

C4 0.1UF   VCC
C5 0.1UF   VCC

U1 — RC8650AFP

Pin	Signal
8	VSS
14	VSS
64	VSS
96	AVSS
9	VCC
16	VCC
62	VCC
17	VCC
39	VCC
99	AVCC
98	AVREF

A/D CONVERTER:
Pin	Signal
2	AMPIN
1	AMPOUT
100	ADTRG
97	AN0
95	AN1
94	AN2
93	AN3

Pin	Signal
30	BRS0
29	BRS1
28	BRS2
7	BRD

TO HOST MICRO:
Pin	Signal
36	RXD
35	TXD
38	CTS#
37	RDY#
49	PRD#
47	STS#
20	PWR#
32	DAIN
31	DAOUT
34	DARTS#
33	DACLK
4	AO0
5	TS0
23	SUSP0#
24	AS0
3	AO1
6	TS1
21	SUSP1#
22	AS1
92	SEL1
91	SEL2
90	SEL3
89	SEL4
10	SEL5
18	STBY#
41	ACLR#
12	RESET#
15	XIN
13	XOUT

R3 100K   R4 100K   R5 100K   VCC

IC bus (U1 pin — IC# — U2 pin):
U1	IC	U2
25	IC32	10
26	IC31	9
27	IC30	16
80	IC29	30
79	IC28	32
78	IC27	34
77	IC26	36
76	IC25	39
75	IC24	41
74	IC23	43
73	IC22	45
71	IC21	25
70	IC20	24
69	IC19	23
68	IC18	22
67	IC17	21
66	IC16	20
65	IC15	19
63	IC14	18
61	IC13	8
60	IC12	7
59	IC11	6
58	IC10	5
57	IC9	4
56	IC8	3
55	IC7	2
54	IC6	1
53	IC5	48
52	IC4	17
50	IC3	26
46	IC2	11
44	IC1	28
42	IC0	12

PIO bus:
U1	PIO	U2
81	PIO7	44
82	PIO6	42
83	PIO5	40
84	PIO4	38
85	PIO3	35
86	PIO2	33
87	PIO1	31
88	PIO0	29

U2 — RC46xxFP

Pin	Signal
27	VSS
46	VSS
37	VCC
13	VCC
14	VCC
47	VCC

VCC   R2 100K   R1 47K   TST   C1 1UF

Y1 7.3728 MHZ   C2 18 PF   C3 18 PF

C6 22NF   R6 33K   R7 33K   R8 68K   C9 150PF
C7 22NF   C8 8.2NF   C10 1UF

U3 LM4876   VCC   8 Ω SPKR

PROGRAM TS0 = ACTIVE HIGH

CONNECTION DIAGRAMS

RC8650AFP
100-Lead QFP
14 mm x 20 mm

TOP VIEW

Top pins (80–51): IC29, IC28, IC27, IC26, IC25, IC24, IC23, IC22, NC, IC21, IC20, IC19, IC18, IC17, IC16, IC15, VSS, IC14, VCC, IC13, IC12, IC11, IC10, IC9, IC8, IC7, IC6, IC5, IC4, NC

Left pins (81–100): PIO7, PIO6, PIO5, PIO4, PIO3, PIO2, PIO1, PIO0, SEL4, SEL3, SEL2, SEL1, AN3, AN2, AN1, AVSS, AN0, AVREF, AVCC, ADTRG

Right pins (50–31): IC3, PRD#, NC, STS#, IC2, NC, IC1, NC, IC0, ACLR#, NC, VCC, CTS#, RDY#, RXD, TXD, DARTS#, DACLK, DAIN, DAOUT

Bottom pins (1–30): AMPOUT, AMPIN, AO1, AO0, TS0, TS1, BRD, VSS, VCC, SEL5, NC, RESET#, XOUT, VSS, XIN, VCC, VCC, STBY#, NC, PWR#, SUSP1#, AS1, SUSP0#, AS0, IC32, IC31, IC30, BRS2, BRS1, BRS0

RC46xxFP
48-Lead TSOP
12 mm x 20 mm

TOP VIEW

Left pins (1–24): IC6, IC7, IC8, IC9, IC10, IC11, IC12, IC13, IC31, IC32, IC2, IC0, VCC, VCC, NC, IC30, IC4, IC14, IC15, IC16, IC17, IC18, IC19, IC20

Right pins (48–25): IC5, VCC, VSS, IC22, PIO7, IC23, PIO6, IC24, PIO5, IC25, PIO4, VCC, IC26, PIO3, IC27, PIO2, IC28, PIO1, IC29, PIO0, IC1, VSS, IC3, IC21

PIN DESCRIPTIONS

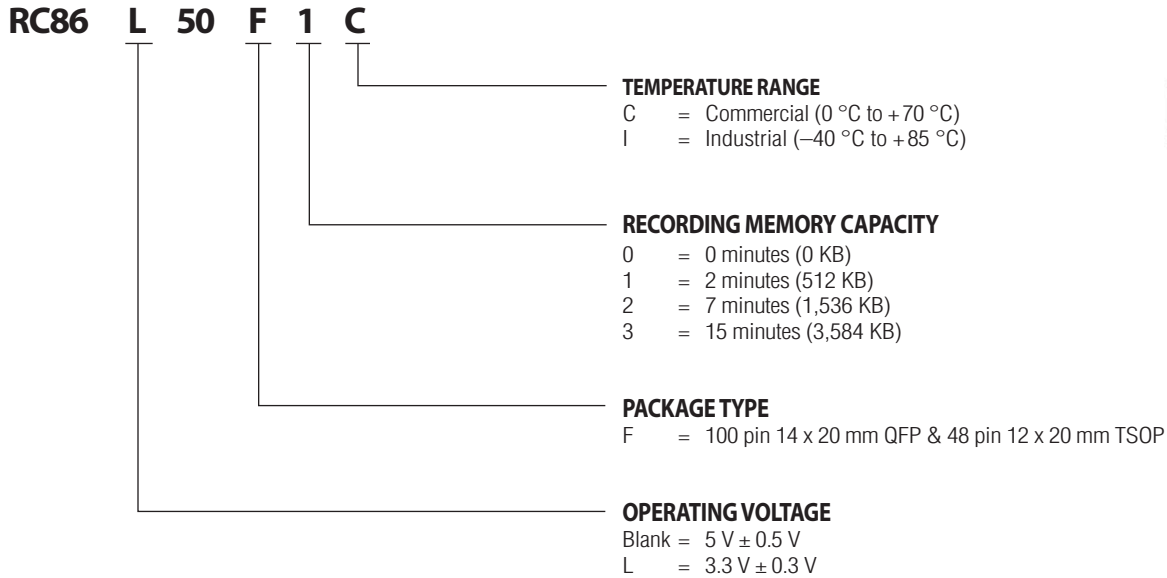| Pin Name	Type	Name and Function
$IC_0 - IC_{32}$	INTPUT/ OUTPUT	**CHIPSET INTERCONNECTS:** Interconnections between the RC8650 and RC46xx chips. IC_0 connects to IC_0, IC_1 to IC_1, etc. $IC_{30} - IC_{32}$ must have a 47 kΩ − 100 kΩ pullup resistor to V_{CC}. No other connections should be made to these pins.
AO_0 AO_1	OUTPUT	**ANALOG OUTPUT:** Channels 0 and 1 digital to analog (D/A) converter outputs. The output voltage range is 0 V to AV_{REF} and is normally biased at $AV_{REF}/2$. Output impedance is 10 kΩ typical. AO_1 is reserved for future use.
TS_0 TS_1	OUTPUT	**TALK STATUS:** Indicates whether a voice channel is active. These pins can be used to enable external devices such as a transmitter, telephone, or audio amplifier. The pins' polarity are programmable, and can be activated automatically or under program control. TS_1 is reserved for future use.
$SUSP_0$# $SUSP_1$#	INPUT	**SUSPEND:** Suspends audio output when Low, allowing playback to be paused. When High, playback resumes at the point output was suspended. These pins affect only the corresponding AO pin; they do not affect the digital audio DAOUT pin (use DARTS# to control DAOUT). $SUSP_1$# is reserved for future use. Connect these pins to a High level if not used.
AS_0 AS_1	OUTPUT	**AUDIO SYNC:** Outputs a clock signal in synchronization with the updating of analog outputs AO_0 and AO_1. The pin changes state whenever the corresponding D/A converter is updated. AS_1 is reserved for future use.
DAOUT	OUTPUT	**DIGITAL AUDIO OUTPUT:** Provides the same 8-bit digital audio stream that is fed to the internal D/A converters. This pin can be programmed to be a CMOS or open-drain output. The communication protocol is progammable, and can operate in synchronous or asynchronous mode.
DACLK	INPUT	**DIGITAL AUDIO CLOCK:** This pin is used to clock data out of the DAOUT pin and data into the DAIN pin in the synchronous digital audio output mode. DACLK can be programmed to transfer data on either the rising edge or falling edge of the clock. Connect this pin to a High level if not used.
DAIN	INPUT	**DIGITAL AUDIO CONTROL INPUT:** This pin is used to control the operation of the DAOUT pin in a multi-channel system. Reserved for a future product; connect this pin to a High level.
DARTS#	INPUT	**DIGITAL AUDIO REQUEST TO SEND:** A Low on this pin enables transmission from the DAOUT pin; a High suspends transmission. DARTS# may be used in both the synchronous and asynchronous transfer modes. Connect this pin to a Low level if not used.
$PIO_0 - PIO_7$	INPUT/ OUTPUT	**PERIPHERAL INPUT/OUTPUT BUS:** Eight-bit bidirectional peripheral bus. Data is input from a peripheral when PRD# is active. Status information is output when STS# is active. $PIO_0 - PIO_7$ also connect to the RC46xx chip. Text, data and commands can be sent to the RC8650 over this bus.
STS#	OUTPUT	**STATUS:** Controls the transfer of status information from the RC8650 to a peripheral. Status information is driven on the $PIO_0 - PIO_7$ pins when STS# is Low. STS# is active only when there is new status information.
PRD#	OUTPUT	**PERIPHERAL READ:** Controls the transfer of data from a peripheral to the RC8650. Data is read from the $PIO_0 - PIO_7$ pins when PRD# is Low.
PWR#	INPUT	**PERIPHERAL WRITE:** Controls the writing of peripheral data to the RC8650. Data on the $PIO_0 - PIO_7$ pins is latched in the RC8650 on the rising edge of PWR#. Sufficient time must be given for the RC8650 to process the data before writing additional data—RDY# or Status Register bit SR.4 should be used for this purpose. Connect this pin to a High level if not used.
RDY#	OUTPUT	**READY:** RDY# High indicates that the RC8650 is busy processing the last byte that was written over the Peripheral I/O Bus. Wait for RDY# to be Low before attempting to write more data. RDY# goes High briefly after each write operation over the $PIO_0 - PIO_7$ bus, acknowledging receipt of each byte. If the RC8650's input buffer becomes full as a result of the last write operation, RDY# will remain High until room becomes available. Note that RDY# can also be read from Status Register bit SR.4.
$AN_0 - AN_3$	INPUT	**A/D CONVERTER INPUTS:** Analog to digital converter input pins. Analog signals sampled on these pins can be read through the serial interface. Leave any unused pins unconnected.
ADTRG	INPUT	**A/D CONVERTER TRIGGER:** Starts A/D conversion when hardware triggering is selected. Minimum Low pulse width is 200 ns. Leave this pin unconnected if not used.

Table 1. Pin Descriptions

Pin Name	Type	Name and Function
AMPIN	OUTPUT	**A/D CONVERTER AMPLIFIER:** Connecting an operational amplifier between these pins allows the input voltage to all four A/D converter input pins to be amplified with one operational amplifier. Leave these pins unconnected if not used.
AMPOUT	INPUT	
RXD	INPUT	**RECEIVE DATA:** Asynchronous serial data input used to send text, data and commands to the RC8650. Connect this pin to a High level if not used.
TXD	OUTPUT	**TRANSMIT DATA:** Asynchronous serial data output used to read information out of the RC8650.
CTS#	OUTPUT	**CLEAR TO SEND:** The CTS# pin is Low when the RC8650 is able to accept data. CTS# acknowledges each byte received on the RXD pin by going High briefly. If the RC8650's input buffer becomes full as a result of the last byte received, CTS# will remain High until room becomes available.
BRD	INPUT	**BAUD RATE DETECT:** BRD is used by the RC8650 to sample the host's serial data stream in order to determine its baud rate. BRD is normally connected to the RXD pin. The BRS_0-BRS_2 pins affect the operation of BRD. Connect this pin to a High level if not used.
BRS_0- BRS_2	INPUT	**BAUD RATE SELECT:** Programs the asynchronous serial port's baud rate. Both the RXD and TXD pins are programmed to the baud rate set by these pins. Connecting BRS_0-BRS_2 to a High level will allow the RC8650 to automatically detect the baud rate with the BRD pin. Connect these pins to a High level if not used.
STBY#	INPUT	**STANDBY:** A Low immediately terminates all activity and places the RC8650 in Standby mode. The RDY# and CTS# pins are driven High, and the input buffer is cleared. During standby, the RC8650 draws the minimum possible current (0.7 µA typ @ 3.3 V), but it is not able to respond to any input pin except STBY# and RESET#. Returning STBY# High causes the RC8650 to enter Idle mode (700 µA typ); the handshake lines are re-asserted and the RC8650 will be able to accept input again. If the RC8650 entered standby due to a Sleep Timer event, driving STBY# Low for t_{WSBL} or longer then High will return the RC8650 to Idle mode. STBY# is also used to restore the RC8650 to its factory default settings. To prevent this from happening unintentionally, make sure that STBY# is High no later than 0.5 sec after RESET# goes High. See Appendix A for additional information. Connect this pin to a High level if not used.
SEL_1- SEL_5	INPUT	**SELECT:** Programs the channel pair that the RC8650 is to respond to in a multi-channel system. These pins are reserved for a future product; connect them to a Low level to ensure upward compatibility.
RESET#	INPUT	**RESET:** A Low immediately terminates all activity and sets all pins, internal voice parameters and register settings to their default states. During power-up, RESET# must be held Low a minimum of 1 ms after V_{CC} has stabilized in the proper voltage range. All pins will be valid within 2 ms after reset.
ACLR#	INPUT	**ANALOG CLEAR:** A Low initializes the RC8650's D/A and A/D converters. Connect ACLR# to RESET#.
XIN	INPUT	**CLOCK INPUT/OUTPUT:** These pins connect to the internal clock generating circuit. All timing for the RC8650 and RC46xx chips are derived from this circuit. Connect a 7.3728 MHz crystal between XIN and XOUT. Alternatively, an external 7.3728 MHz square wave may be applied to XIN.
XOUT	OUTPUT	
V_{CC}		**POWER:** +5 V ± 0.5 V, + 3.3 V ± 0.3 V power supply connection.
V_{SS}		**GROUND:** Connect these pins to system ground.
AV_{CC}		**ANALOG POWER:** Power supply input for the D/A and A/D converters. Connect this pin to V_{CC}.
AV_{SS}		**ANALOG GROUND:** Ground input for the D/A and A/D converters. Connect this pin to V_{SS}.
AV_{REF}		**ANALOG REFERENCE VOLTAGE:** Reference voltage for the D/A and A/D converters. Connect this pin to V_{CC}. *Caution: any noise present on this pin will appear on the AO pins and affect A/D converter accuracy.*
NC		**NO CONNECT:** NC pins must remain unconnected. Connection of NC pins may result in component failure or incompatibility with future product enhancements.

Table 1. Pin Descriptions (Continued)

ORDERING INFORMATION

The RC8650 is available in several voltage and temperature ranges and recording capacities. The ordering part number is formed by combining several fields, as indicated below. Refer to the "Valid Combinations" table, which lists the configurations that are planned to be supported in volume. Other combinations may be available on a request basis.

All configurations include two surface-mount devices: the 100 pin RC8650AFP and 48 pin RC46xxFP. Only the RC46xxFP is affected by the voltage and recording capacity option chosen, as can be seen in the "Chipset" column in the table below.

RC86 L 50 F 1 C

TEMPERATURE RANGE
C   =   Commercial (0 °C to +70 °C)
I   =   Industrial (−40 °C to +85 °C)

RECORDING MEMORY CAPACITY
0   =   0 minutes (0 KB)
1   =   2 minutes (512 KB)
2   =   7 minutes (1,536 KB)
3   =   15 minutes (3,584 KB)

PACKAGE TYPE
F   =   100 pin 14 x 20 mm QFP & 48 pin 12 x 20 mm TSOP

OPERATING VOLTAGE
Blank = 5 V ± 0.5 V
L   =   3.3 V ± 0.3 V

RoHS
COMPLIANT

VALID COMBINATIONS

Order Number	Old Order No.	Voltage	Package	Rec Capacity	Temp Range	Chipset
RC8650F1C	RC8650-1	5 V	QFP + TSOP	2 min	0 °C to +70 °C	RC8650AFP + RC4651FP
RC86L50F1I	RC86L50-1	3.3 V	QFP + TSOP	2 min	−40 °C to +85 °C	RC8650AFP + RC46L51FP
RC86L50F3I	RC86L50-3	3.3 V	QFP + TSOP	15 min	−40 °C to +85 °C	RC8650AFP + RC46L71FP

FUNCTIONAL DESCRIPTION

The RC8650 chipset includes a number of features that make it ideally suited for any design requiring voice output. The RC8650's major features are described below.

Text-to-Speech Synthesizer

The RC8650 provides text-to-speech conversion with its integrated DoubleTalk™ text-to-speech synthesizer. Any English text written to the RC8650 is automatically converted into speech. Commands can be embedded in the input stream to dynamically control the voice, even at the phoneme level (phonemes are the basic sound units of speech).

A greeting message can be stored in the RC8650 that is automatically spoken immediately after the RC8650 is reset. Most any of the commands recognized by the RC8650 may be included as part of the greeting message, which can be used to set up custom default settings and/or play a prerecorded message or tone sequence. An integrated nonvolatile memory area is also provided for storing a custom pronunciation dictionary, allowing the pronunciation of any character string to be redefined.

Recorded Audio Playback

Up to 15 minutes of recorded messages and sound effects can be stored in the RC8650 for on-demand playback. Recordings are stored in on-chip nonvolatile memory, providing zero-power message storage. Additionally, the RC8650 can play eight-bit PCM and ADPCM audio in real time, such as speech and/or sound effects stored in an external memory or file system.

Musical Tone Generator

An integrated, three-voice musical tone generator is capable of generating up to three tones simultaneously over a four-octave range. Simple tones to attention-getting sounds can be easily created.

Touch-Tone Generator

The RC8650 includes an integrated DTMF (Touch-Tone) generator. This is useful in telephony applications where standard DTMF tones are used to signal a remote receiver, modem, or access the public switched telephone network.

Sinusoidal Tone Generator

A precision, dual sinusoidal tone generator can synthesize the tones often used in signaling applications. The tone frequencies can be independently set, allowing signals such as call-progress tones to be generated.

Analog-to-Digital Converter

The four channel, 8-bit A/D converter can be used to monitor battery cell voltages, temperature, and other analog quantities. The ADC can be programmed on the fly to convert any single channel, or scan up to four channels repetitively.

Versatile I/O

All data is sent to the RC8650 through its built in serial and/or parallel ports. *For maximum flexibility, including infield update capability, use of the serial port is recommended whenever possible.*

The RC8650's audio output is available in both analog and digital formats. The analog output should be used in applications where no further processing of the audio signal is required, such as driving a speaker or headphones (the output still needs to be filtered and amplified, however). The digital output is for applications that require further processing of the audio signal, such as digital mixing or creating sound files for later playback.

RECOMMENDED CONNECTIONS

Power/Ground

Power and ground connections are made to multiple pins of the RC8650 and RC46xx chips. Every V_{CC} pin must be connected to power, and every V_{SS} pin must be connected to ground. To minimize noise, the analog and digital circuits in the RC8650 use separate power busses. These busses are brought out to separate pins and should be tied to the supply as close as possible.

Make sure adequate decoupling is placed on the AV_{REF} pin, as noise present on this pin will also appear on the AO output pins and affect A/D converter accuracy. In systems where the power supply is very quiet, AV_{REF} can be connected directly to V_{CC}. Designs incorporating a switching power supply, or supplies carrying heavy loads, may require filtering at the AV_{REF} pin; a 150 Ω series V_{CC} resistor in combination with a 100 μF capacitor to ground should suffice.

Connect any unused input pins to an appropriate signal level (see Table 1). **Leave any unused output pins and all NC pins unconnected.**

Chip Interconnects

Pins IC_0 through IC_{32} and PIO_0 through PIO_7 must be connected between the RC8650 and RC46xx chips. IC_{30}, IC_{31}, and IC_{32} must have 47 kΩ−100 kΩ pullup resistors to V_{CC}.

Clock Generator

The RC8650 has an internal oscillator and clock generator that can be controlled by an external 7.3728 MHz crystal, ceramic resonator, or external 7.3728 MHz clock source. If an external clock is used, connect it to the XIN pin and leave XOUT unconnected. See Figure 1 for recommended clock connections.
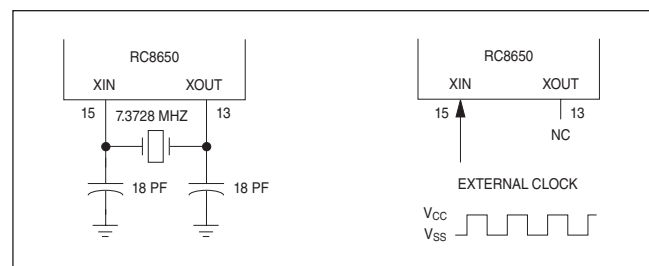
Figure 1. Clock Connections

INTERFACING THE RC8650

The RC8650 contains both asynchronous serial and 8-bit bus interfaces. All text, commands, tone generator and real-time audio data, etc., are transmitted to the RC8650 via one of these ports. For maximum flexibility, use of the serial port is recommended, because *not all RC8650 functions are supported through the bus interface*. In particular, index markers, firmware updates, certain status information and A/D conversion are only supported through the serial interface.

Serial Interface

The RC8650's serial port is asynchronous and operates with 8 data bits (LSB first), 1 or more stop bits, no parity, and any standard baud rate between 300 and 115200 bps. Most modern microcontrollers have at least one UART that can be used to connect to the RC8650.

A typical microcontroller interface circuit is shown in Figure 2. The circuit includes an external programming port, which allows the RC8650 to be programmed in-circuit. Data files and firmware updates can be downloaded from a PC to the RC8650 through this port. Note that an external RS-232 transceiver chip will generally be required, in order to convert the PC's RS-232 voltage levels to the RC8650's logic levels.

The CTS# pin should be used to control the flow of serial data to the RC8650. It is not necessary to check CTS# before transmitting every byte, however. All data is routed through a high speed 16-byte buffer within the RC8650 before being stored in the primary buffer. *CTS# may be checked every eight bytes with no risk of data loss.*

Baud rate selection

The serial port's baud rate can be programmed using any of three methods: pin strapping, auto-detect, and by command. **Pin strapping** sets the baud rate according to the logic levels present on the BRS_0–BRS_2 pins, as shown in Table 2. **Auto-detect** enables the serial port to automatically detect the baud rate of the incoming data.
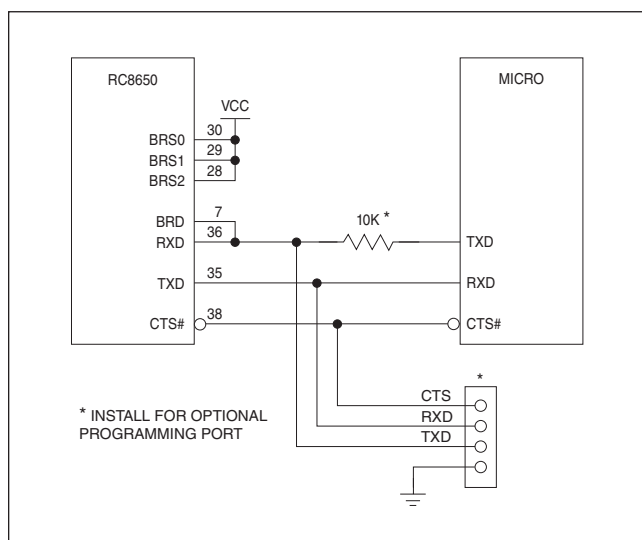
BRS_2	BRS_1	BRS_0	Baud Rate
L	L	L	300
L	L	H	600
L	H	L	1200
L	H	H	2400
H	L	L	4800
H	L	H	9600
H	H	L	19200
H	H	H	Auto-detect

Table 2. Baud Rate Options

The baud rate **command** allows the baud rate to be changed at any time, effectively overriding the first two methods. Pin strapping cannot be used to program baud rates higher than 19200; to do this, auto-detection or the baud rate command must be used.

The automatic baud rate detection mechanism is enabled when the BRS_0–BRS_2 pins are all at a High logic level and the BRD pin is connected to the RXD pin. The baud rate is determined by the shortest High or Low period detected in the input stream. This period is assumed to be the bit rate of the incoming data; therefore, it is important that there be at least one isolated "1" or "0" in the input character. The CR character, 0Dh, is recommended for locking the baud rate. The character is not otherwise processed by the RC8650; it is discarded. In addition to the baud rates listed in Table 2, auto-detect mode also supports 38400, 57600, and 115200 baud rates.

If the measured bit period is determined to be a valid baud rate, the RC8650 will acknowledge lock acquisition by transmitting the ASCII character "l" (6Ch) on the TXD pin. (Note that nothing will be transmitted if the baud rate has been programmed for a fixed rate.) The baud rate will remain locked unless changed with the baud rate command, or the RC8650 is reset.

Note The measurement cycle ends when there have been no High-to-Low nor Low-to-High transitions on the BRD pin for 75 ms or longer. Consequently, the RC8650 will ignore any data sent to it for a period of 75 ms after the "lock-on" character has been received. The CTS# pin is driven High during this time, and the acknowledgment character is not transmitted until the RC8650 is actually ready to accept data. See Figure 3.
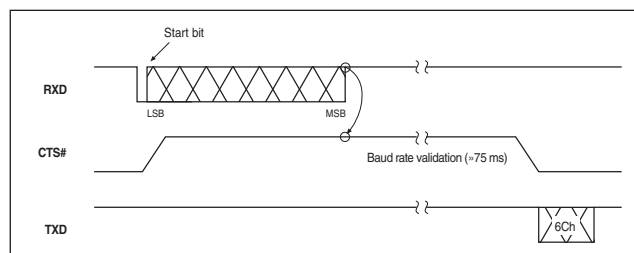

Figure 2. Serial Interface

Figure 3. Baud Rate Detection Timing

Status messages

Real-time status information is provided via the TXD pin. Status are transmitted as one-byte messages, shown in Table 3. Each message correlates to a status flag in the Status Register, shown in Table 4. The specific character used, and whether it will be transmitted, are functions of the V86 and STM bits of the Protocol Options Register. For information about how to obtain reading-progress status, see the Index Marker command description.

Event	V86 = 0 / V86 = 1	Requires STM = 1
Output has started	"B" / "s"	Yes
Output has stopped	"E" / "t"	Yes
Buffer almost empty (<100 bytes remaining)	— / "e"	Yes
Buffer almost full (<100 bytes available)	— / "f"	Yes
Standby mode confirmation	"S" / "p"	No
Baud rate lock confirmation	"L" / "l"	No

Table 3. Status Messages

Bus/Printer Interface

The RC8650's bus interface allows the RC8650 to be connected to a microprocessor or microcontroller in the same manner as a static RAM or I/O device, as shown in Figure 5. The microcontroller controls all transactions with the RC8650 over the system data bus using the RD and WR# signals. RD controls the reading of the RC8650's Status Register; WR# controls the transfer of data into the RC8650. The Status Register bits and their definitions are shown in Table 4.

A registered bus transceiver is required for communication between the RC8650 and microcontroller; two 74HC374s placed back to back may be substituted for the 74HC652 shown in the figure. Prior to each write operation to the RC8650, the host processor should verify that the RC8650 is ready by testing the RDY status flag.

The RC8650 can also be interfaced to a PC's printer port as shown in Figure 5. A 74HC374 can be used in place of the 74HC652, since bidirectional communication is not necessary. Handshaking is performed automatically via the BUSY pin.

Because the RC8650 can take up to 15 µs to accept data written to it (AC Characteristics, t_{YHWH} parameter), software drivers should wait for RDY to drop to 0 after a byte is written in order to avoid overwrit-

ing it with the next data byte. Not doing so could result in the loss of data. Waiting for RDY to drop to 0 ensures that RDY will not falsely show that the RC8650 is ready the next time the driver is called.

If a system interrupt can occur while waiting for RDY to become 0, or if RDY cannot otherwise be checked at least once every 8 µs, a software timeout should be enforced to avoid hanging up in the wait loop. The time RDY stays 0 is relatively short (8 µs min.) and can be missed if interrupted. The timeout should be at least 15 µs, which is the maximum time for RDY to drop to 0 after writing a byte of data. In non time-critical applications, the output routine could simply delay 15 µs or longer before exiting, without checking for RDY = 0 at all.

Figure 4 illustrates the recommended method of writing data to the RC8650's bus interface. This method should be used for writing all types of data, including text, commands, tone generator and real-time audio data.
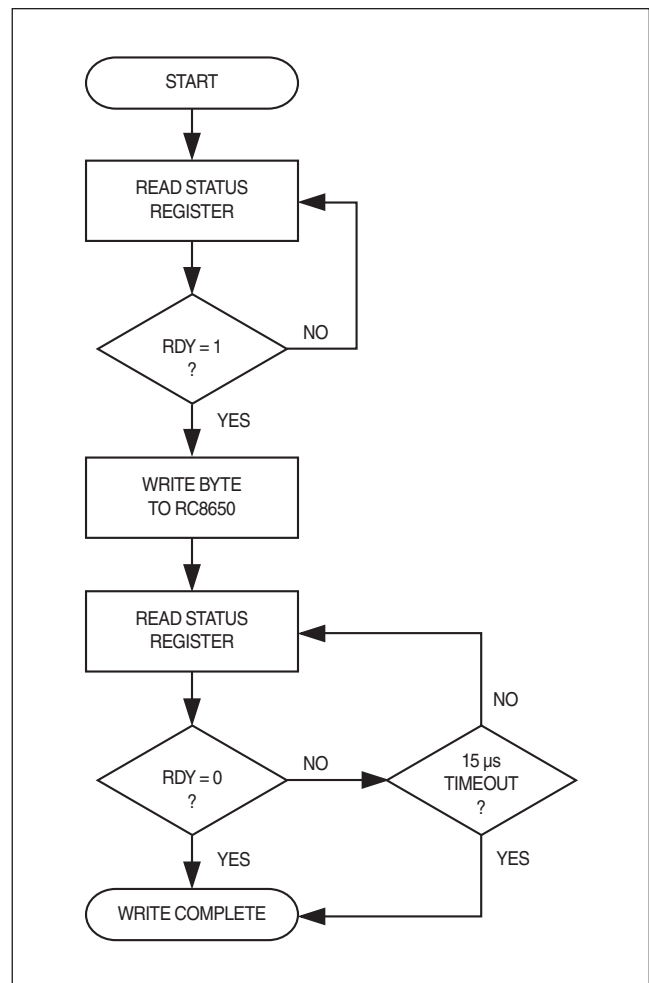

Figure 4. Recommended Method of Writing Data Via the Bus Interface

R	TS	R	RDY	AF	AE	STBY	R
7	6	5	4	3	2	1	0

Status Register Bit	Description
SR.7 = RESERVED (R)	Reserved for future use. Mask when polling the Status Register.
SR.6 = TALK STATUS (TS) 1 = Talking 0 = Idle	This bit is "1" when the RC8650 is producing output, "0" when output has ceased. The TS bit is not affected by the TS Pin Control command, which affects only the TS pins.
SR.5 = RESERVED (R)	Reserved for future use. Mask when polling the Status Register.
SR.4 = READY STATUS (RDY) 1 = Ready 0 = Busy	The RDY bit has the same meaning as the RDY# pin. The RC8650 sets RDY to "1" to indicate that it is ready to receive data. RDY drops to "0" momentarily after each write operation over the PIO bus, acknowledging receipt of each character.
SR.3 = ALMOST FULL (AF) 1 = Buffer almost full 0 = Buffer not almost full	This bit is "1" anytime there are less than 100 bytes available in the input buffer. AF is always "0" in the real-time audio playback mode and when using the musical tone generator.
SR.2 = ALMOST EMPTY (AE) 1 = Buffer almost empty 0 = Buffer not almost empty	This bit is "1" anytime there are less than 100 bytes remaining in the input buffer. AE is always "1" in the real-time audio playback mode and when using the musical tone generator.
SR.1 = STANDBY MODE (STBY) 1 = RC8650 is in Standby mode 0 = RC8650 not in Standby mode	This bit is "1" when the RC8650 has entered Standby mode. Standby mode is entered either by setting the STBY# pin Low or by allowing the Sleep Timer to expire.
SR.0 = RESERVED (R)	Reserved for future use. Mask when polling the Status Register.

Table 4. Bus Interface Status Register
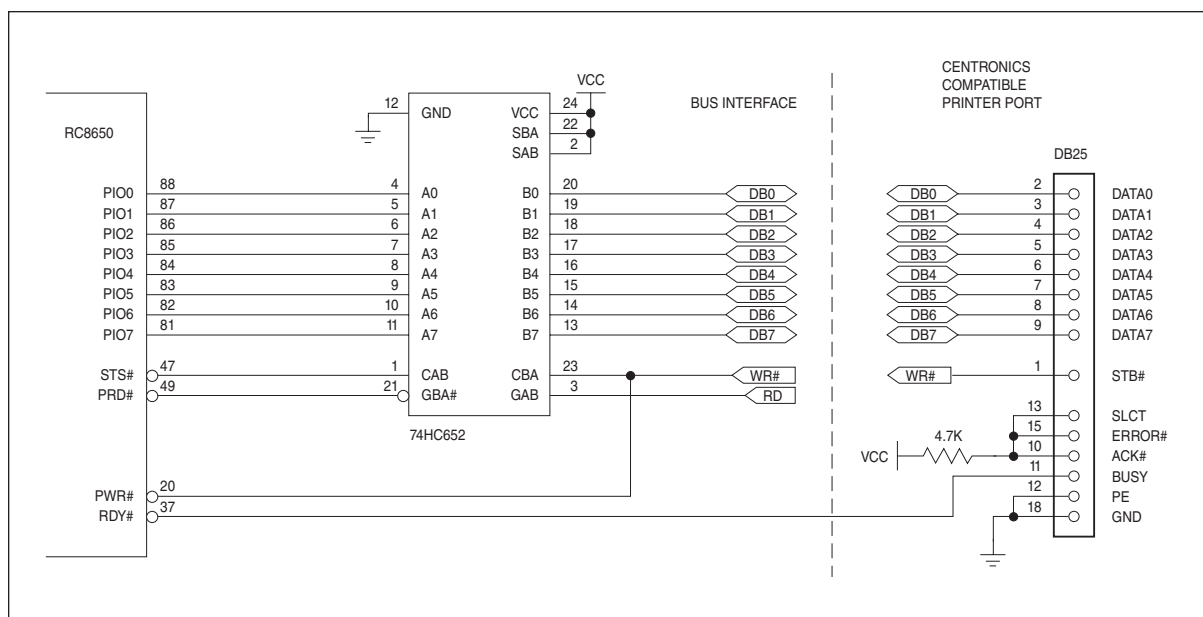
Figure 5. Bus/Printer Interface
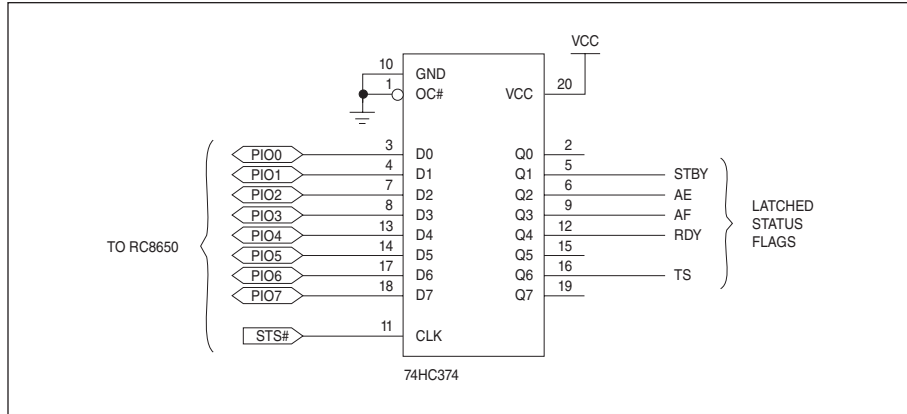
Figure 6. Method of Capturing Status Information for Driving External Circuitry

Analog Audio Output

The analog output pins AO_0 and AO_1 are unfiltered, high impedance outputs from the RC8650's internal D/A converters. When using these outputs, the addition of an external low-pass filter is highly recommended. When laying out the printed circuit board, avoid running digital lines near the AO lines in order to minimize induced noise in the audio path. If space permits, run a guard ground next to the AO traces.

The circuit shown in Figure 7 is a low-pass filter/power amplifier capable of delivering 1.1 W to an 8 Ω load when operating from a +5 V power supply (power output will be less when operating from +3.3 V). The amplifier's shutdown pin can be controlled by the TS_0 pin to minimize current drain when the RC8650 is inactive. The resistor in series with the headphone jack provides balance between relative speaker volume and headphone volume, as well as improving headphone S/N ratio and providing short-circuit protec-

tion. See Typical Application Circuit for a similar circuit without a volume control.

Digital Audio Output

The digital audio pin DAOUT delivers the RC8650's audio signal as a digital audio stream. The data format is 8-bit linear, offset binary, where 80h = midscale. The DAOUT pin can be programmed for synchronous or asynchronous operation. The transfer format and clock polarity are programmable in the synchronous mode.

The normalized sampling rate for all text to speech modes and the DTMF generator is 80 kbs (10 kbytes/sec). Because the sinusoidal generator, prerecorded and real-time audio playback mode rates are user programmable, their normalized rates will vary. See the Pin Descriptions and Audio Control Register command description for more information.
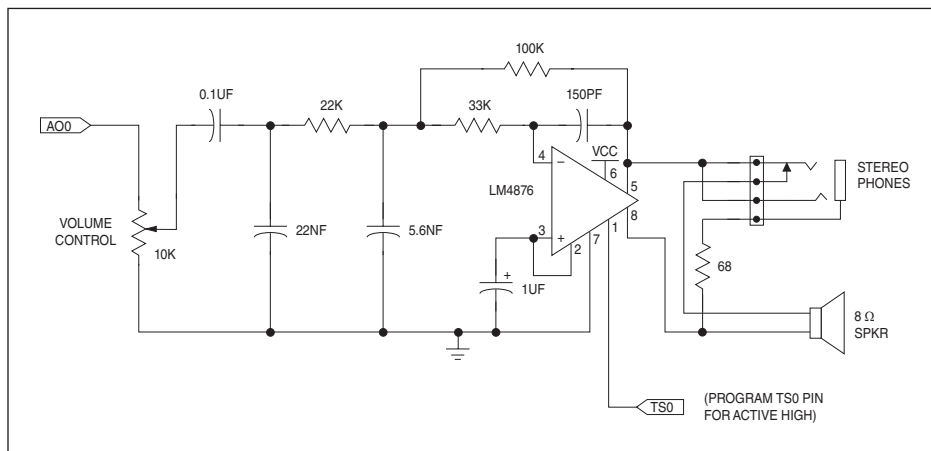
Figure 7. Low Cost 3 kHz Low-Pass Filter/Power Amplifier

ELECTRICAL SPECIFICATIONS
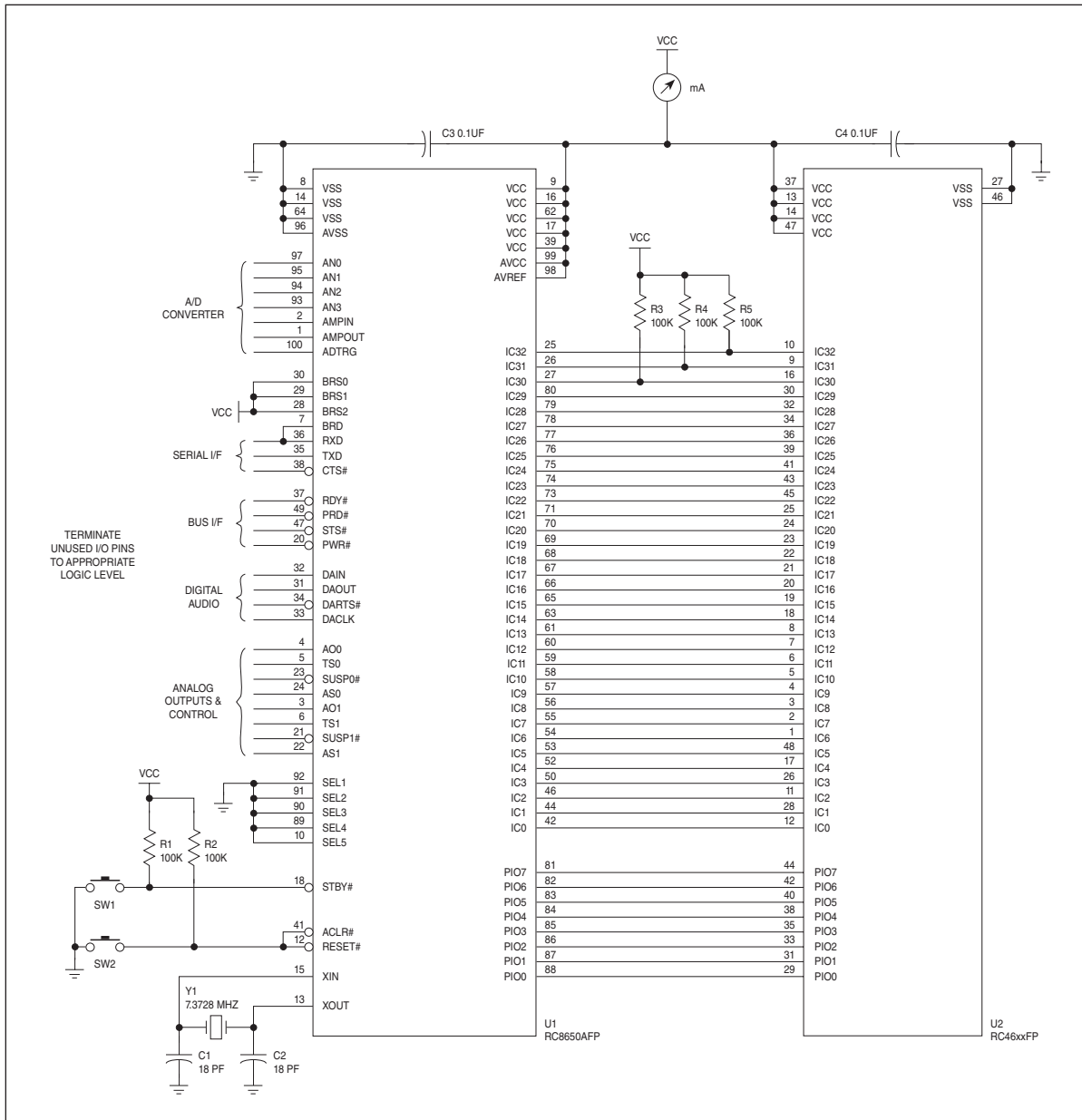
Figure 8. Test Circuit

ABSOLUTE MAXIMUM RATINGS*

Supply voltage, V_{CC} and AV_{CC}

3.3 V devices . . . . −0.3 V to +4.0 V

5 V devices . . . . −0.3 V to +6.5 V

DC input voltage, V_I . . . . −0.3 V to V_{CC} +0.3 V

Operating temperature, T_A . . . . 0 °C to +70 °C / −40 °C to +85 °C

Storage temperature, T_S . . . . −55 °C to +125 °C

WARNING: Stresses greater than those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

RC Systems
now you're talking!

DC CHARACTERISTICS

T_A = 0 °C to +70 °C / −40 °C to +85 °C, V_{CC} = AV_{CC} = AV_{REF} = 3.3 V / 5 V, V_{SS} = AV_{SS} = 0 V, X_{IN} = 7.3728 MHz

Symbol	Parameter	3.3 V ± 0.3 V			5 V ± 0.5 V			Unit	Test Conditions
		Min	Typ	Max	Min	Typ	Max		
V_{IL}	Input voltage, Low	−0.3		$0.2V_{CC}$	−0.3		$0.2V_{CC}$	V	
V_{IH}	Input voltage, High	$0.7V_{CC}$		V_{CC} + 0.3	$0.7V_{CC}$		V_{CC} + 0.3	V	
V_{IA}	Analog input voltage (AN$_{0-3}$)	−0.3		AV_{REF}	−0.3		AV_{REF}	V	
V_{HYR}	Input hysterisis, RESET#	0.2		1.8	0.2		1.8	V	
V_{OL}	Output voltage, Low			0.5			0.5	V	I_{OL} = 1 mA
V_{OH}	Output voltage, High	V_{CC}−0.5			V_{CC}−0.5			V	I_{OH} = −1 mA
I_{IL}	Input load current			±4			±5	µA	V_{IN} = V_{SS} to V_{CC}
R_O	Analog output resistance (AO$_0$, AO$_1$)	4	10	20	4	10	20	kΩ	
I_{CC}	Supply current								All outputs open; all inputs = V_{CC} or V_{SS}; AV_{CC} and AV_{REF} currents included
	Active		4	15		7.5	24	mA	
	Idle		0.7	1.5		1.2	2.5	mA	
	Standby		0.7	15		0.8	20	µA	
	Program (Note 1)			30			40	mA	

[1] Applies during internal programming operations: greeting message, dictionary, sound library and microcode updates.

AC CHARACTERISTICS

T_A = 0 °C to +70 °C / −40 °C to +85 °C, V_{CC} = AV_{CC} = AV_{REF} = 3.3 V / 5 V, V_{SS} = AV_{SS} = 0 V

External Clock Input Timing

Symbol	Parameter	3.3 V ± 0.3 V			5 V ± 0.5 V			Unit
		Min	Nom	Max	Min	Nom	Max	
f_C	External clock input frequency	7.2991	7.3728	7.4465	7.2991	7.3728	7.4465	MHz
t_{WCL}	External clock input Low pulse width	60	67.8		40	67.8		ns
t_{WCH}	External clock input High pulse width	60	67.8		40	67.8		ns
t_{CR}	External clock rise time			18			15	ns
t_{CF}	External clock fall time			18			15	ns
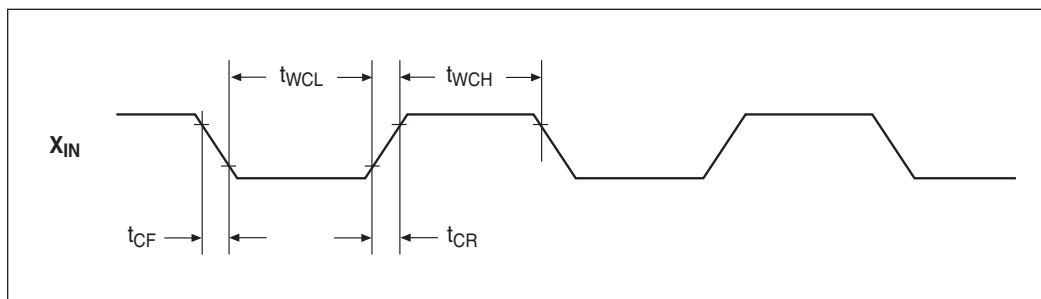
Figure 9. External Clock Waveform

Bus Interface Timing

Symbol	Parameter	3.3 V ± 0.3 V		5 V ± 0.5 V		Unit
		Min	Max	Min	Max	
t_{WSL}	STS# pulse width Low	215		250		ns
t_{DVSL}	STS# Low to data valid		155		150	ns
t_{DHSH}	Data hold from STS# going High	5		5		ns
t_{WRL}	PRD# pulse width Low	215		250		ns
t_{DVRH}	Data setup to PRD# going High	85		40		ns
t_{DHRH}	Data hold from PRD# going High	0		0		ns
t_{WWL}	PWR# pulse width Low	380		250		ns
t_{DVWH}	Data setup to PWR# going High	−2		−2		µs
t_{DHWH}	Data hold from PWR# going High	15		15		µs
t_{YHWH}	RDY# High from PWR# going High (Note 1)		15		15	µs
t_{WYH}	RDY# pulse width High (Note 1)	8		8		µs

[1] Applies to the RDY# pin and RDY status flag.
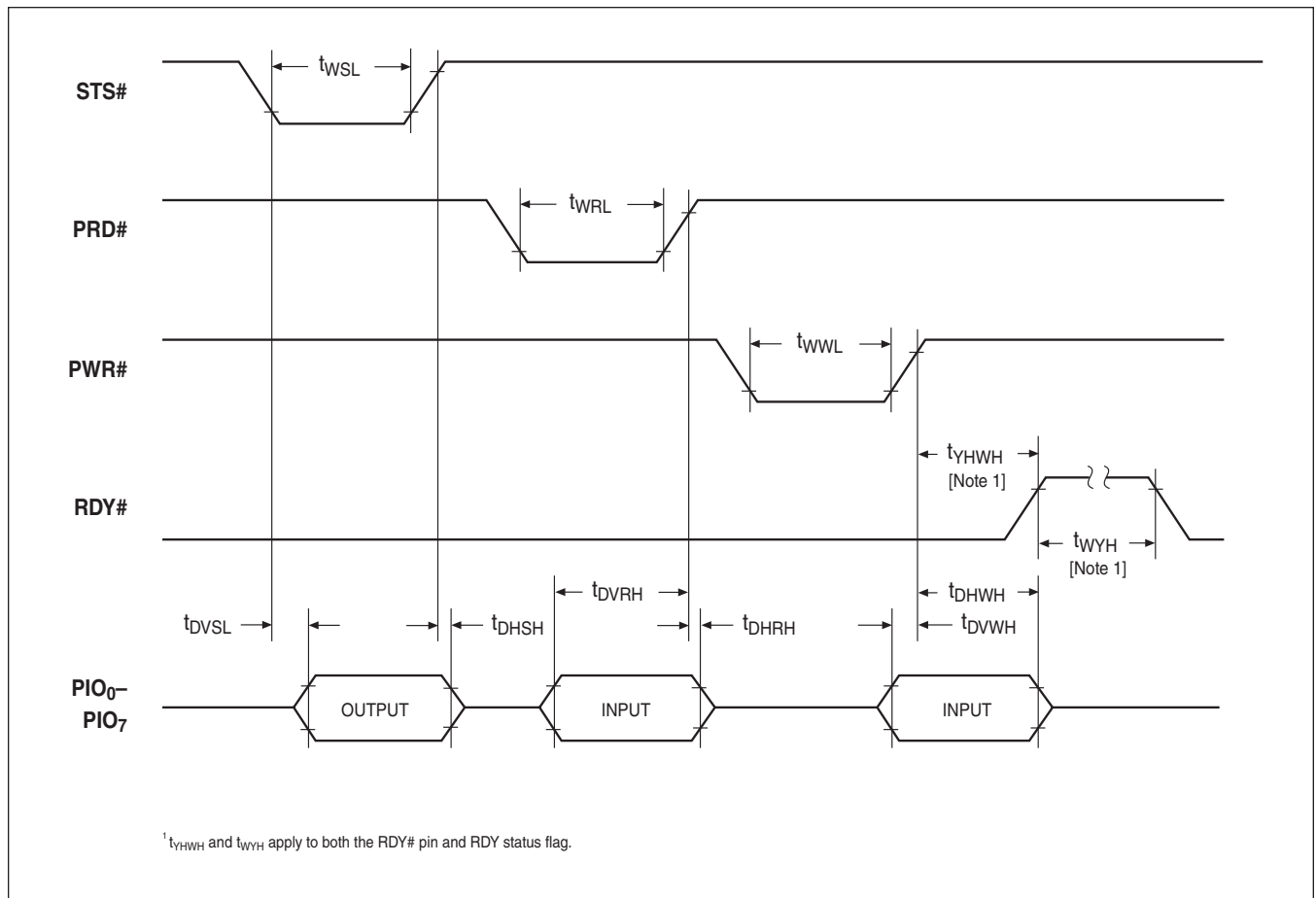
Figure 10. Bus Interface Waveforms

Analog Audio Timing
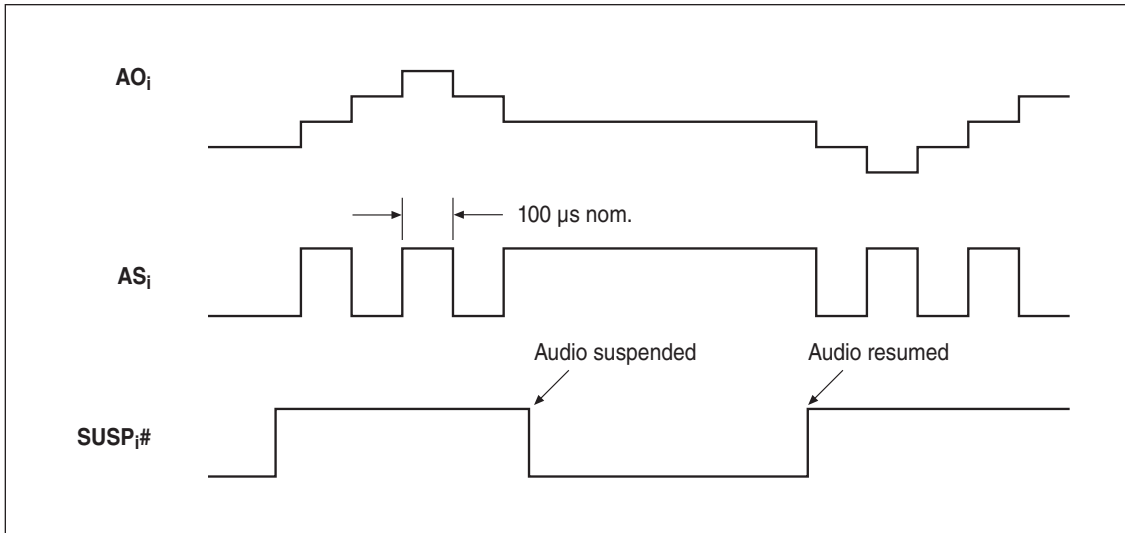
Figure 11. Analog Audio Waveforms

Digital Audio Timing

Symbol	Parameter	3.3 V ± 0.3 V		5 V ± 0.5 V		Unit
		Min	Max	Min	Max	
t_{CYC}	DACLK cycle time	300		200		ns
t_{WCL}	DACLK pulse width Low	150		100		ns
t_{WCH}	DACLK pulse width High	150		100		ns
t_{DVC}	DAOUT output delay time		160		80	ns
t_{DHC}	DAOUT hold time	0		0		ns
f_S	Nom. TTS, sinusoidal and DTMF generators internal sampling rate	10	10	10	10	kHz
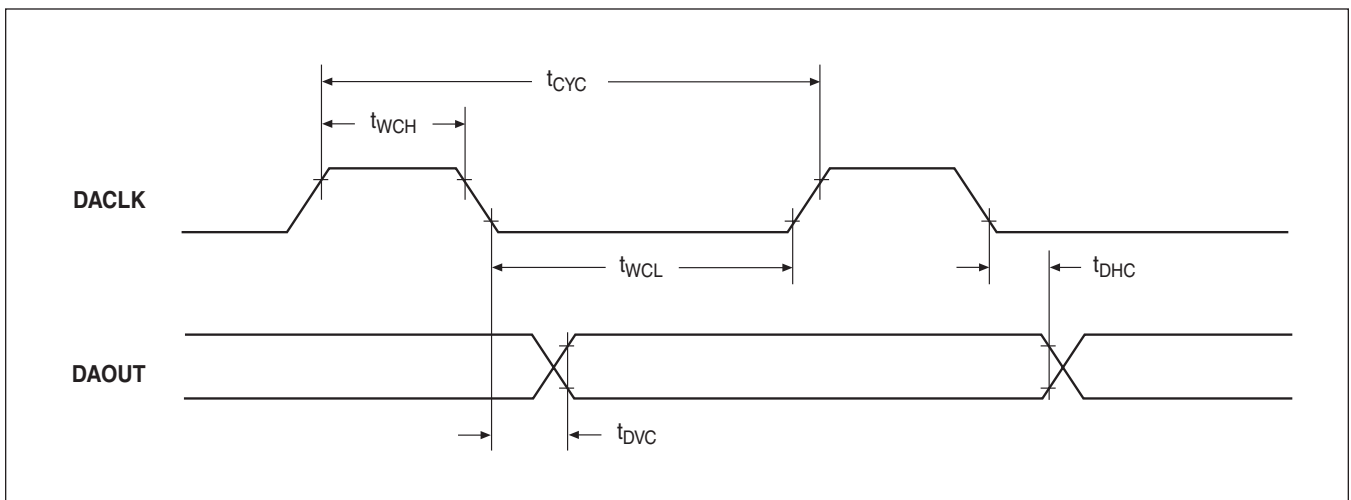
Figure 12. Digital Audio Waveforms

Standby Timing

Symbol	Parameter	3.3 V ± 0.3 V		5 V ± 0.5 V		Unit
		Min	Max	Min	Max	
t_{WSBL}	STBY# pulse width Low					
	To enter Standby mode	5		5		ms
	To exit Standby mode (Sleep Timer invoked; Note 1)	380		250		ns

[1] Monitor handshake lines to determine when Standby mode has terminated.
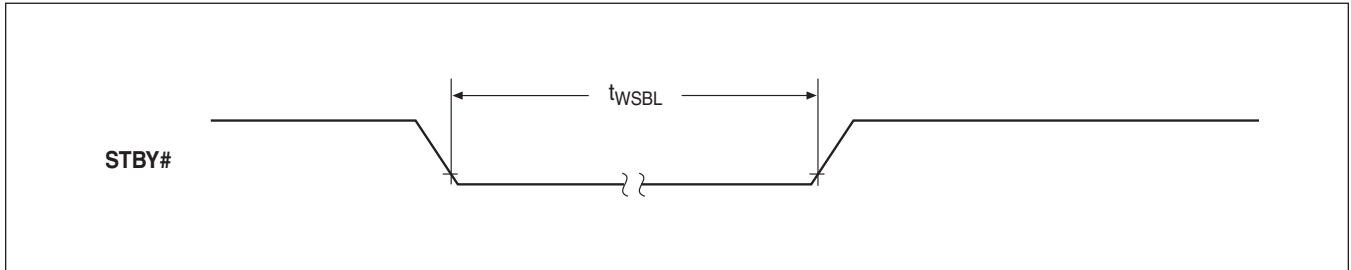
Figure 13. Standby Waveform

Reset Timing

Symbol	Parameter	Min	Max	Unit	Notes
t_{WRS}	RESET# pulse width Low				Hold RESET# Low during power-up. STBY# must be High no later than 0.5 sec after RESET# goes High.
	After power on / V_{CC} stable	1		ms	
	During operation	3		µs	
t_{DRR}	RESET# recovery delay		2	ms	RC8650 performs internal initialization; all outputs will be floating. Do not attempt to send data to the RC8650 during this period.
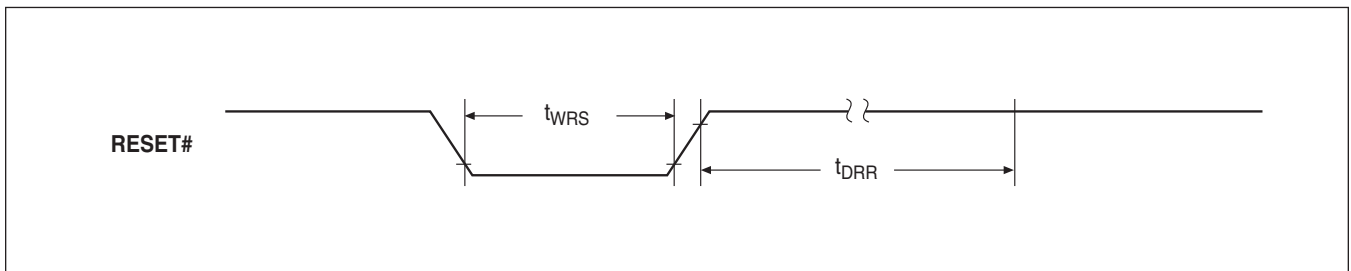
Figure 14. Reset Waveform

OPERATING MODES

The RC8650 has four primary operating modes and two inactive modes designed to achieve maximum functionality and flexibility. The operating mode can be changed anytime, even on the fly, by issuing the appropriate command to the RC8650.

Note: The RC8650 will not begin speaking or execute commands until it receives a CR (0Dh) or NUL (00h) character — this ensures that a complete contextual analysis can be performed on the input text. If it is not possible for the application to send a CR or NUL at the end of each text message, use the Timeout Delay command.

The RC8650 does not make any distinction between uppercase and lowercase characters. All data sent to the RC8650 is buffered in an internal 2 KB input buffer, allowing additional text and commands to be queued even while the RC8650 is producing output.

Text-to-speech mode. By default, all text sent to the RC8650 is automatically translated into speech by the integrated DoubleTalk TTS engine. TTS mode can be further subdivided into three translation modes: *Text*, which reads text normally; *Character*, which reads (spells) one character at a time; and *Phoneme*, which allows the TTS engine's phonemes to be directly accessed.

Real-Time Audio Playback mode. Data sent to the RC8650 is written directly to the RC8650's audio buffer. This results in a high data rate, but provides the capability of producing the highest quality speech, as well as sound effects. PCM and ADPCM data types are supported.

Prerecorded Audio Playback mode. This mode allows recorded messages and sound effects that have been stored in the RC8650 to be played back. PCM and ADPCM data types are supported.

Tone Generator modes. These modes activate the RC8650's musical tone generator, sinusoidal generator, or DTMF generator. They can be used to generate audible prompts, music, signaling tones, dial a telephone, etc.

Idle mode. To help conserve power in battery-powered systems, the RC8650 automatically enters a reduced-power state whenever it is inactive. Data can still be read and written to the RC8650 while in this mode. Current draw is typically 700 µA @ 3.3 V.

Standby mode. This mode powers down the RC8650, where current draw is typically only 0.7 µA. Standby mode can be invoked from either the STBY# pin or with the Sleep command. Data cannot be read from or written to the RC8650 in this mode.

RC8650 COMMAND SYNTAX

The RC8650 commands provide a simple yet flexible means of controlling the RC8650 under software control. They can be used to vary voice attributes, such as volume or pitch, to suit the requirements of a particular application or listener's preferences. Commands are also used to change operating modes.

Commands can be freely intermixed with text that is to be spoken — allowing the voice to be dynamically controlled — or to dynamically change operating modes, such as generating tones or playing recorded messages, in the middle of a passage of text.

All RC8650 commands are composed of the command character (CTRL+A by default), a parameter *n* comprised of an ASCII number string, and an ASCII character that uniquely identifies the command. Some commands simply enable or disable a feature of the RC8650 and do not require a parameter. The command format is:

<command character>[<number string>]<ASCII character>

Commands are not case sensitive. If two or more commands are to be used together, each must be prefaced with the command character. This is the only way the RC8650 knows to treat the subsequent characters as commands, rather than text that is to be spoken. For example, the following commands program pitch level 40 and volume level 7 (quotes and spaces shown for clarity purposes only):

```
CTRL+A "40p" CTRL+A "7v"
```

The Command Character

The default RC8650 command character is CTRL+A (01h). The command character itself can be spoken by the RC8650 by sending it twice in a row: CTRL+A CTRL+A. This special command allows the command character to be spoken without affecting the operation of the RC8650, and without having to change to another command character and then back again.

Changing the command character

The command character can be changed to another control character (01-1Ah) by sending the current command character, followed by the new character. To change the command character to CTRL+D, for example, send CTRL+A CTRL+D. To change it back, send CTRL+D CTRL+A. It's recommended to change the command character if the text to be read contains characters which may otherwise be interpreted as command characters (and hence commands). The command character can be unconditionally reset to CTRL+A by sending CTRL+^ (1Eh) to the RC8650.

Command Parameters

Command parameters are composed of ASCII number strings. The RC8650 supports two types of parameters: absolute and relative. **Absolute** parameters explicitly specify the parameter's new value, such as 9S or 3B. **Relative** parameters specify a *displacement* from a parameter's current value, not the actual new value itself.

Relative parameters specify positive or negative displacements from a parameter's current value. For example, the Volume command +2V increases the volume level by two (V+2→V). If the current volume is 4, the volume will increase to 6 after the command has executed. The command −2V will have a similar effect, except the volume will be *decreased* by two. When operating on an RC8650 register (Punctuation Filter, Protocol Options, Audio Control and ADC Control), relative parameters allow you to set ("+") and clear ("−") individual register bits. For example, +65G sets bits POR.0 and POR.6; −16$ clears ADR.4.

If the value of a parameter falls outside the command's range, the value will either wrap around or saturate, depending on the setting of the SAT bit of the Protocol Options Register. For example, if parameters are programmed to wrap, the current volume is 7 and the command $+4V$ is issued, the resultant volume will be $(7+4)-10 = 1$, since the volume range is 0-9. If parameters are programmed to saturate, the resultant volume would be 9 instead.

When writing application programs for the RC8650, it is recommended that relative parameters be used for temporarily changing voice attributes (such as raising the pitch of a word), using absolute-parameter commands only once in the program's initialization routine. This way, if the base value of an attribute needs to be changed, it only needs to be changed in the initialization routine.

TTS SYNTHESIZER

Using the TTS synthesizer couldn't be simpler: simply write the text to be read to the RC8650; the RC8650 does the rest. The RC8650 also includes a number of software commands that allow you to modify the behavior of the TTS synthesizer, as described in this section.

Translation Accuracy

Because the RC8650 must handle the highly irregular spelling system of English, as well as proper names, acronyms, technical terms, and borrowed foreign words, there inevitably will be words that it will mispronounce. If a word is mispronounced, there are three techniques for correcting it:

1) Spell the word phonetically for the desired pronunciation.

2) Redefine the way the word should be pronounced by creating an exception for it in the RC8650's exception dictionary. This method allows words to be corrected without having to modify the original text, and it automatically corrects all instances of the word.

3) Use the RC8650's Phoneme mode.

The first technique is the easiest way to fine tune word pronunciations — by tricking the RC8650 into the desired pronunciation. Among the more commonly mispronounced words are compound words, proper names (*Sean*), and foreign loan words (*chauffeur*). Compound words can usually be corrected by separating the two words with a space. Proper names and foreign words may require a bit more creativity, so that "Sean" becomes "Shon," and "chauffeur" becomes "show fur." Heteronyms (words with identical spelling but different meanings and pronunciations) can also be modified using this technique. For example, if the word *read* is to be pronounced "reed" instead of "red," it can simply be respelled as "reed."

Text Mode/Delay (T/nT)

This command places the RC8650 in the Text operating mode. The optional delay parameter *n* is used to create a variable pause between words. The shortest, and default delay of 0, is used for normal speech. For users not accustomed to synthetic speech, the synthesizer's intelligibility may be improved by introducing a delay. The longest delay that can be specified is 15. If the delay parameter

is omitted, the last set value will be used and *the exception dictionary will be disabled*. This feature is useful for returning from another operating mode or disabling the exception dictionary (see Enable Exception Dictionary command).

Character Mode/Delay (C/nC)

This command puts the RC8650 in the Character operating mode. The optional delay parameter *n* is used to create a variable pause between characters. Values between 0 (the default) and 15 provide pauses from shortest to longest, respectively. Values between 16 and 31 provide the same range of pauses, but control characters will not be spoken. If the delay parameter is omitted, the last set value will be used and *the exception dictionary will be disabled*.

Phoneme Mode (D)

This command disables the text-to-phonetics translator, allowing the RC8650's phonemes to be accessed directly. Table 5 lists the phonemes that can be produced by the RC8650.

Phoneme Symbol	Example Word	Phoneme Symbol	Example Word
A	das (Spanish)	N	new
AA	cot	NG	rung
AE	cat	NY	niño (Spanish)
AH	cut	O	no (Spanish)
AW	cow	OW	boat
AX	bottom	OY	boy
AY	bite	P	pop
B	bib	PX	spot
CH	church	R	ring
D	did	RR	tres (Spanish)
DH	either	S	sell
DX	city	SH	shell
E	ser (Spanish)	T	tin
EH	bet	TH	thin
EI	mesa (Spanish)	TX	stick
ER	bird	U	uno (Spanish)
EW	acteur (French)	UH	book
EY	bake	UW	boot
F	fee	V	valve
G	gag	W	we
H	he	WH	when
I	libro (Spanish)	Y	mayo (Spanish)
IH	bit	YY	you
IX	rabbit	Z	zoo
IY	beet	ZH	vision
J	age	space	variable pause *
K	cute	' (apostrophe)	short pause
KX	ski	, (comma)	medium pause
L	long	. (period)	long pause
M	me		

* Normally used between words; duration determined by nT command.

Table 5. DoubleTalk Phoneme Symbols

When concatenating two or more phonemes, each phoneme must be delimited by a space. For example, the word "computer" would be represented phonetically as

```
k ax m p yy uw dx er
```

Phoneme attribute modifiers

The RC8650 supports seven phoneme attribute modifiers that can be used in Phoneme mode and exception dictionaries. Table 6 lists these tokens and their equivalent RC8650 commands.

Symbol	Function	Equiv Cmd
nn	Set pitch to 'nn' (0-99)	nP
/	Increase pitch m steps *	$+m$P
\	Decrease pitch m steps *	$-m$P
+	Increase speed 1 step	+1S
−	Decrease speed 1 step	−1S
>	Increase volume 1 step	+1V
<	Decrease volume 1 step	−1V

* Step size determined by nE command; $m \cong 2n$

Table 6. Phoneme Attribute Modifiers

Applications of Phoneme mode

Phoneme mode is useful when the normal text-to-speech modes are inappropriate for producing the desired voice effect. For example, Phoneme mode should be used to change the stress or emphasis of specific words in a phrase. This is because Phoneme mode allows voice attributes to be modified on phoneme boundaries within each word, whereas Text mode allows changes only at word boundaries. This is illustrated in the following examples.

```
CTRL+A "d" CTRL+A "m" "//h aw  -/d>/eh r
+<\\yy uw  s p \iy k  t uw  \m iy  dh ae
t  -\w ey .+/"
```

Note that expression is disabled in this example, since the pitch variations due to the internal intonation algorithms would otherwise interfere with the pitch tokens. Compare this with the same phrase produced in Text mode with expression enabled:

```
CTRL+A "t" CTRL+A "e" "How dare you speak to
me that way!"
```

Phoneme mode is also useful in applications that provide their own text-to-phoneme translation, such as the front end of a custom text-to-speech system.

Speed (nS)

The synthesizer's speech rate can be adjusted with this command, from 0S (slowest) through 9S (fastest). The default rate is 1S (5S if the V86 bit of the Protocol Options Register is set to 0).

Voice (nO)

The text-to-speech synthesizer has eight standard voices and a number of individual voice parameter controls that can be used to independently vary the voice characteristics. Voices are selected with the commands 0O through 7O, shown in Table 7. Because the Voice command alters numerous internal voice parameters (articulation, pitch, expression, tone, etc.), it should precede any individual voice parameter control commands.

n	Voice Name
0	Perfect Paul (default)
1	Vader
2	Big Bob
3	Precise Pete
4	Ricochet Randy
5	Biff
6	Skip
7	Robo Robert

Table 7. Voice Presets

Articulation (nA)

This command adjusts the articulation level, from 0A through 9A. Excessively low articulation values tend to make the voice sound slurred; very high values, on the other hand, can make the voice sound choppy. The default articulation is 5A.

Expression (E/nE)

Expression, or intonation, is the variation of pitch within a sentence or phrase. When expression is enabled ($n > 0$), the RC8650 attempts to mimic the pitch patterns of human speech. For example, when a sentence ends with a period, the pitch drops at the end of the sentence; a question mark will cause the pitch to rise.

The optional parameter n determines the degree of intonation. 0E provides no intonation (monotone), whereas 9E is very animated sounding. 5E is the default setting. If the parameter is omitted, the current (last set) value will be used. This is useful for re-enabling intonation after a Monotone command.

Monotone (M)

This command disables all intonation (expression), causing the RC8650 to speak with a monotonic voice. Intonation should be disabled whenever manual intonation is applied using the Pitch command or phoneme attribute modifiers. This command is equivalent to the 0E command.

Formant Frequency (nF)

This command adjusts the synthesizer's overall frequency response (vocal tract formant frequencies), over the range 0F through 9F. By varying the frequency, voice quality can be fine-tuned or voice type changed. The default frequency is 5F.

Pitch (nP)

This command varies the pitch over a wide range, which can be used to change the average pitch during speech production, produce manual intonation, or create sound effects (including singing). Pitch values can range from 0P through 99P; the default is 50P.

Tone (nX)

The synthesizer supports three tone settings, bass (0X), normal (1X) and treble (2X). The best setting to use depends on the speaker being used and personal preference. Normal (1X) is the default setting.

Reverb (nR)

This command is used to add reverberation to the voice. 0R (the default) introduces no reverb; increasing values of n correspondingly increase the reverb delay and effect. 9R is the maximum setting.

Punctuation Filter Register (nB)

Depending on the application, it may be desirable to limit the reading of certain punctuation characters. For example, if the RC8650 is used to proofread documents, the application may call for only unusual punctuation to be read. On the other hand, an application that orally echoes keyboard entries for a blind user may require that all punctuation be spoken.

The Punctuation Filter Register determines which punctuation characters will be spoken and how number strings will be translated, as shown in Table 8.

Effect on number strings

When the NM bit is 0, number strings will be read one digit at a time, e.g., 0123 = "zero one two three." Setting NM to 1 forces number strings to be read as numbers (0123 = "one hundred twenty three"). Additionally, when NM = 1 and FM = 10 or 11, currency strings will be read as they are normally spoken. For example, $11.95 will be read as "eleven dollars and ninety five cents." Four-digit numbers will be read as years when YM = 0; e.g., 1492 = "fourteen ninety-two." Setting LZS = 1 disables leading zero suppression; number strings beginning with zero will always be read one digit at a time.

The default filter setting is 6B (Some punctuation, Numbers mode, Years mode, leading zero suppression enabled).

R	R	R	YM	LZS	NM	FM	FM
7	6	5	4	3	2	1	0

Punctuation Filter Register Bits
PFR.7−5 = RESERVED (R)   Write "0" to ensure future compatibility.

PFR.4 = YEARS MODE (YM)
    1 = Read four-digit numbers normally
    0 = Read four-digit numbers as years

PFR.3 = LEADING ZERO SUPPRESSION (LZS)
    1 = Do not suppress leading zeroes
    0 = Suppress leading zeroes

PFR.2 = NUMBERS MODE (NM)
    1 = Read number strings as numbers
    0 = Read number strings as digits

PFR.1−0 = FILTER MODE (FM)
    00 = All spoken
    01 = Most spoken (all but CR, LF, Space)
    10 = Some spoken ($%&#@=+*^|\<>)
    11 = None spoken

Table 8. Punctuation Filter Register

A/D CONVERTER

The integrated analog-to-digital converter can be used to monitor analog quantities, such as battery voltage or temperature. Figure 15 is a functional block diagram of the ADC input stage; Figure 16 illustrates the ADC in operation.

The ADC has the following features:

- Four channels, 8-bit resolution (± 2 LSB precision)
- One-shot, continuous, single sweep, and continuous sweep modes of operation
- Selectable software or hardware triggering
- Support for external amplification/signal conditioning of all four ADC channels

The output of the ADC is governed by the following equation. The result n is transmitted via the TXD pin as an 8-bit value.

$$n = 256 \times V_{IN}/AV_{REF} \qquad (0 \le n \le 255)$$

ADC Control Register (n$)

The ADC Control Register controls the operation of the ADC. Table 9 lists the functions of each register bit. The default register setting is 128$.

Operation of the ADC is not mutually exclusive of other RC8650 functions. The ADC can operate concurrently with text-to-speech, tone generation, recording memory playback, etc. To start a conversion, simply write to the register with INH = 0. The effective sampling rate in continuous mode is one-tenth the serial port baud rate (e.g., 115200 baud = 11520 samples per second).

INH	AMP	TRG	CONT	SWP	R	CH	CH
7	6	5	4	3	2	1	0

ADC Control Register Bit	Description
ADR.7 = INHIBIT (INH) 1 = Stop A/D conversions 0 = Start A/D conversions	This bit must be "0" in order for the ADC to begin sampling the input channel(s); setting it to "1" while the ADC is operating will cause all conversions to stop. This bit allows the other register bits to be programmed without actually starting the ADC. INH automatically changes to "1" at the end of a conversion in one-shot mode. Default: "1."
ADR.6 = EXTERNAL AMPLIFIER (AMP) 1 = Amp connected 0 = Amp not connected	Set this bit to "1" to use an operational amplifier connected between the AMPIN and AMPOUT pins. Connecting an op amp and enabling this function allows the voltage input to each ADC input pin to be amplified with one op amp. Default: "0."
ADR.5 = TRIGGER SOURCE (TRG) 1 = Hardware trigger (ADTRG pin) 0 = Software trigger	Setting this bit to "1" enables hardware triggering of the ADC. The ADC will not begin operating until ADR.7 is set to "0" and the ADTRG pin changes from a High to a Low level. When TRG is "0" the ADC will begin operating as soon as ADR.7 is set to "0." Default: "0."
ADR.4 = CONTINUOUS MODE (CONT) 1 = Continuous mode 0 = One-shot mode	Setting this bit to "1" causes the ADC to operate continuously. If a single channel is selected for measurement (ADR.3 = 0), that channel will be read repeatedly. If sweep mode is selected (ADR.3 = 1), the active input channels will be continuously read in a cyclic fashion. Clearing this bit while the ADC is operating will stop the ADC. Default: "0."
ADR.3 = SWEEP MODE (SWP) 1 = Sweep mode 0 = Single-channel mode	This bit determines whether a single channel or multiple input channels will be read. When Sweep mode is selected, ADR.1 and ADR.0 determine which input channels will be scanned. Default: "0."
ADR.2 = RESERVED (R)	Reserved for future use. Write "0" to ensure future compatibility.

ADR.1−0 = CHANNEL SELECT (CH)
 	
When ADR.3 = 0: When ADR.3 = 1:
  00 = AN_0 00 = undefined
  01 = AN_1 01 = AN_0-AN_1 sweep
  10 = AN_2 10 = undefined
  11 = AN_3 11 = AN_0-AN_3 sweep
	These bits determine which input channel(s) will be read by the ADC. Default: "00."

NOTES:
1. The AMPOUT pin can be used as a fifth ADC input if an external op amp is not used. Set ADR.6 = 1 to select the AMPOUT pin for conversion.
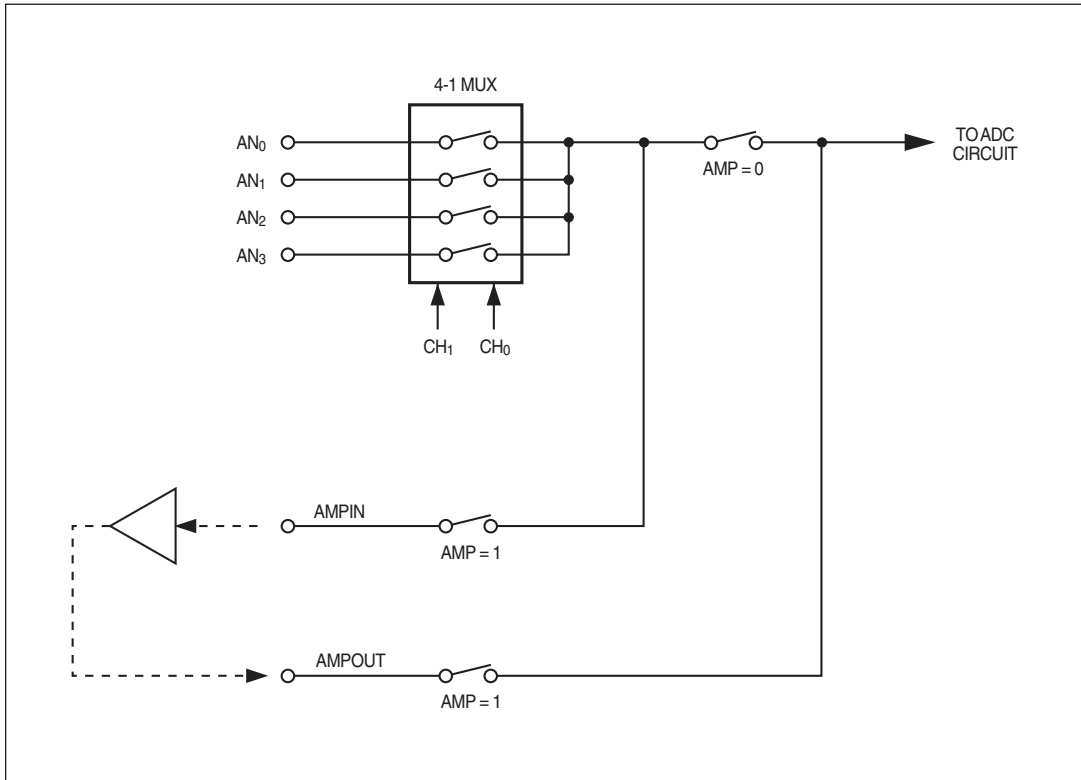
Table 9. ADC Control Register

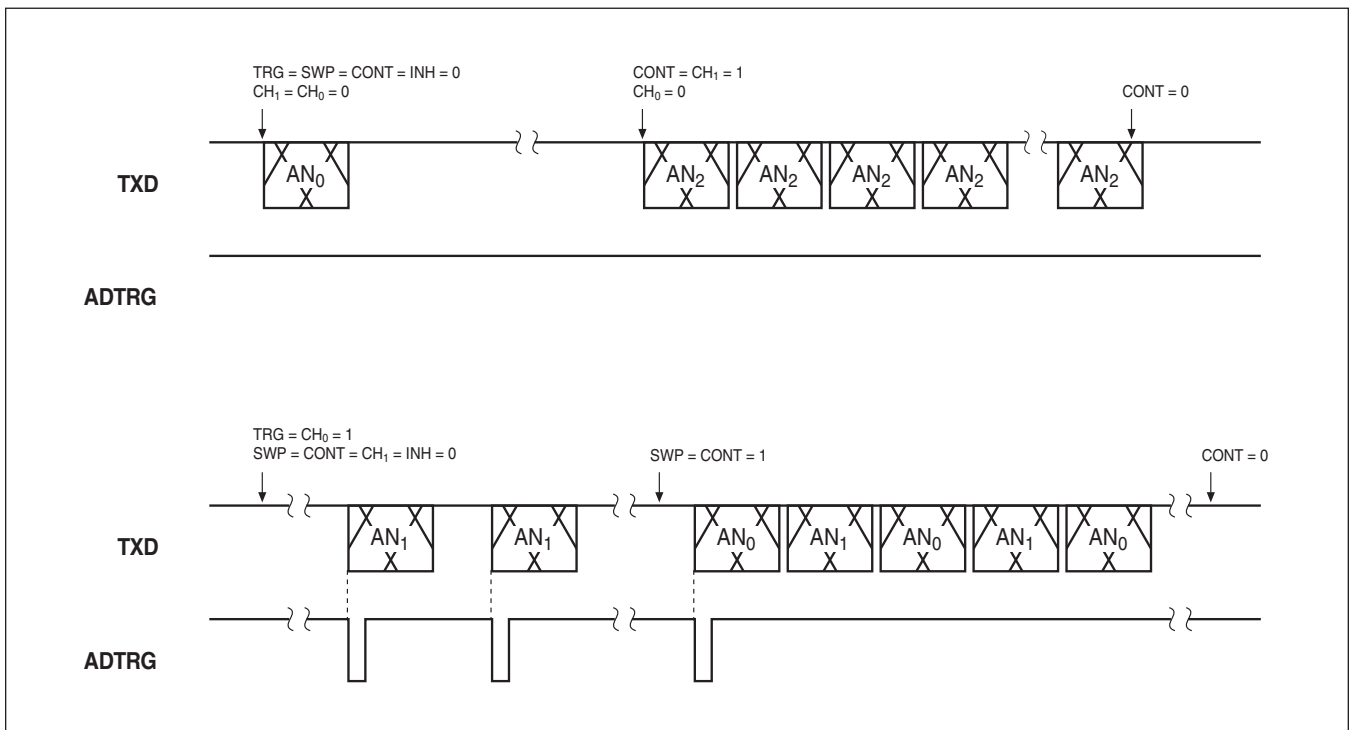**Figure 15. ADC Input Block Diagram**

Figure 16. ADC Transfer Timing

AUDIO PLAYBACK

Libraries of sounds and recorded messages can be stored in the RC8650's integrated flash-based memory for on-demand playback. New libraries can be downloaded at any time, even in the field, through the RC8650's serial interface. The number of sound files is limited only by the amount of available on-chip memory.

Additionally, the RC8650 can play back 8-bit audio in real time, such as speech and/or sound effects stored in an external memory or file system.

Download Sound Library (223W)

This command is generally only found in sound libraries created with *RCStudio*, a Windows-based application available from RC Systems. It initiates the download of the sound library (a collection of sound files) to the RC8650. *RCStudio* makes it easy to create, manage, and download sound libraries composed of standard Windows wave and MP3 files. See *RCStudio's* help for additional information regarding sound libraries.

Play Sound File (n&)

Each sound file (message or sound effect) in a sound library is automatically assigned a unique file number, or index, beginning with zero. The first file is file 0, the second is file 1, and so on. This command plays files in any random order, using n to specify the desired file.

The playback volume can be adjusted with the Volume (nV) command. A volume setting of 5 will cause sound files to be played back at their original volume level.

Text and/or commands may be freely intermixed with the playback command. For example,

```
CTRL+A "11*" "Hello" CTRL+A "-3v" CTRL+A "3&"
CTRL+A "+3v" CTRL+A "9&"
```

plays the Touch-Tone "#" key and says "hello" at the current volume setting, followed by the fourth sound file at a reduced volume level, and finally the tenth sound file at the original volume level.

Real-Time Audio Playback (n#/n%)

This mode allows audio samples to be written to the RC8650's digital-to-analog converter via the serial and parallel ports. All data sent to the RC8650 is routed directly to the RC8650's internal 2 KB audio buffer; the RC8650 then transfers samples from the buffer to the DAC at the rate programmed by n. Because the audio data is buffered within the RC8650, the output sampling rate is independent of the data rate into the RC8650, as long as the input rate is greater than the programmed sampling rate.

The RC8650 supports PCM and ADPCM audio data formats. The ADPCM format is a compressed format which yields data files that are half the size of PCM files, thereby reducing the required data bandwidth and storage requirements. RC Systems' *RCStudio* software can convert Windows wave and MP3 files to PCM and ADPCM formats for use with the RC8650.

The output sampling rate can be programmed to any rate between 4 and 11 kHz (32,000-88,000 bps) by choosing the appropriate parameter value. The relationship between the command parameter n and the sampling rate f_s is

$$n = 155 - 617/f_s$$
$$f_s = 617/(155 - n)$$

where f_s is measured in kHz. For example, to program an 8 kHz sampling rate, choose n=78. The range of n is 0–99, hence f_s can range from 4 to 11 kHz.

The following procedure should be used to transfer a PCM or ADPCM file to the RC8650:

1) Program the desired volume level with the Volume (nV) command. A volume setting of 5 will cause the data to be played back at its original volume level. This step is optional.

2) Issue the Real-Time Audio Playback Mode command n# if PCM data is being sent, or n% for ADPCM data. The RC8650 expects the audio data to immediately follow the command; therefore, *do not* terminate the command with a CR or NUL.

3) If the RC8650's serial port is being used to transfer the audio data, and the port isn't already running at 115200 baud, change the *host* system's baud rate to 115200 at this time.

4) Begin transferring the PCM file to the RC8650. The first four bytes of a PCM/ADPCM file form a header containing file length and encoding information, so transfer should begin with the fifth byte in the file. Note that the transfer of samples from the audio buffer to the DAC will not begin until at least 100 bytes have been sent or the value 80h is sent, whichever occurs first.

5) After the file has been completely transferred to the RC8650, send the value 80h (−128). This signals the RC8650 to terminate playback and return to the text-to-speech mode of operation. Additional files may be transferred without leaving playback mode by repeating step 4 for each file, sending the 80h character only after all the files have been transferred.

   Because up to 2 KB of data may be present in the audio buffer, the RC8650 can continue producing sound for as long as 0.5 second (at 4 kHz sampling rate). Note that from the time the 80h is sent, the handshake lines will remain in their "not ready" states until all of the audio data has been transferred from the buffer to the DAC.

   If the host's serial port baud rate was changed in step 3, it should now be changed back to its original rate.

TONE GENERATORS

The RC8650 contains three tone generators: musical, sinusoidal, and DTMF (Touch-Tone).

MUSICAL TONE GENERATOR

The musical tone generator is capable of producing three tones (voices) simultaneously, and works well in applications which require neither precise frequencies nor a pure (low distortion) output. The output is a pulse train rich in harmonic energy, which sounds more interesting than pure sinusoids in music applications. *RC-Studio* includes a music score editor and compiler which greatly simplifies the music-creation process.

Note: The musical tone generator output is available only from the AO pins. Digital audio output from the DAOUT pin is not possible.

The musical tone generator is activated with the J command (no parameter). Once activated, all data output to the RC8650 is directed to the musical tone generator.

Note: The RC8650 expects the tone generator data to immediately follow the J command; therefore, *do not* terminate the command with a CR or NUL.

The musical tone generator is controlled with four, four-byte data and command frames, called **Initialize**, **Voice**, **Play**, and **Quit** (Figure 17). With these, the volume, duration, and frequencies of the three voices can be controlled.

Initialize Command

The Initialize command sets up the tone generator's relative amplitude and tempo (speed). The host must issue this command to initialize the tone generator before sending any Voice frames. The Initialize command may, however, be issued anytime afterward to change the volume or tempo on the fly.

Initialize command format

The Initialize command consists of a byte of zero and three parameters. The parameters are defined as follows:

K_A	Voice amplitude (1-255)
K_{TL}	Tempo, low byte (0-255)
K_{TH}	Tempo, high byte (0-255)

The range of the tempo K_T (K_{TL} and K_{TH}) is 1-65,535 (1–FFFFh); the larger the value, the slower the overall speed of play. The amplitude and tempo affect all three voices, and stay in effect until another Initialize command is issued. If the command is issued between Voice frames to change the volume or tempo on the fly, only the Voice frames following the command will be affected.

Voice Frame

Voice frames contain the duration and frequency (pitch) information for each voice. All Voice frames are stored in a 2 KB buffer within the RC8650, but are not played until the Play command is issued. If the number of Voice frames exceeds 2 KB in length, the RC8650 will automatically begin playing the data.
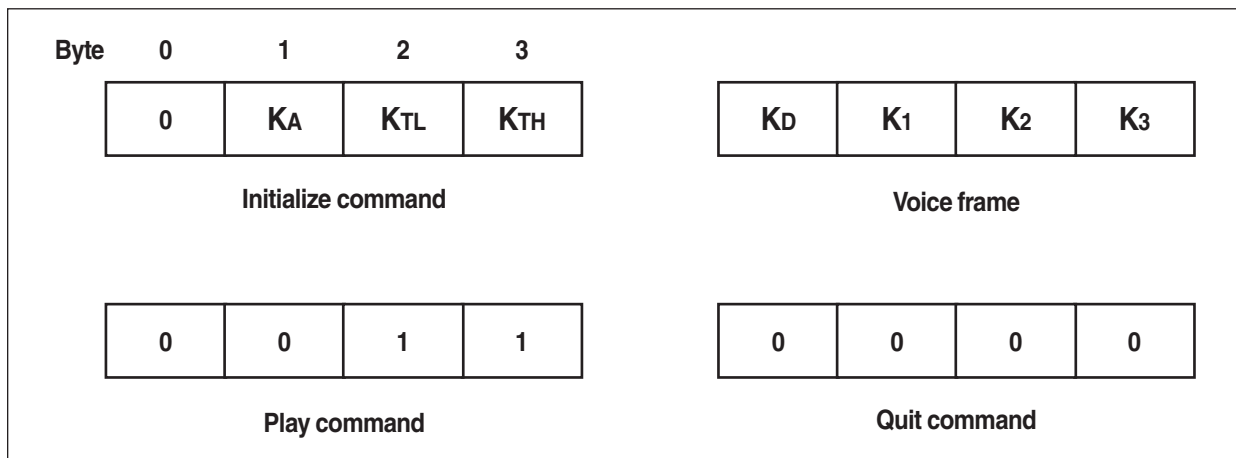
Voice frame format

Voice frames are composed of three frequency time constant bytes (K_1-K_3) and a duration byte (K_D), which specifies how long the three voices are to be played.

The relationship between the time constant K_i and the output frequency f_i is:

$$f_i = 16,768/K_i$$

where f_i is in Hertz and K_i = 4-255. Setting K_i to zero will silence voice *i* during the frame.

K_D may be programmed to any value between 1 and 255; the larger it is made, the longer the voices will play during the frame.

Byte	0	1	2	3
	0	K_A	K_{TL}	K_{TH}

Initialize command

K_D	K_1	K_2	K_3

Voice frame

0	0	1	1

Play command

0	0	0	0

Quit command

Figure 17. Musical Tone Generator Command Formats

The task of finding K_i for a particular musical note is greatly simplified by using Table 10. The tone generator can cover a four-octave range, from C two octaves below Middle C ($K_i = 255$), to D two octaves above Middle C ($K_i = 14$). K_i values less than 14 are not recommended.

For example, the Voice frame

```
DATA 24,64,0,0
```

plays Middle C using voice 1 ($K_1 = 64$). Since K_2 and K_3 are zero, voices 2 and 3 will be silent during the frame. The duration of the note is a function of both the tempo K_T and duration K_D (24).

As another example,

```
DATA 48,64,51,43
```

plays a C-E-G chord, for a duration twice as long as the previous example.

Note	K_i	Note	K_i
C	255 (FFh)	D	57 (39h)
C#	241 (F1h)	D#	54 (36h)
D	228 (E4h)	E	51 (33h)
D#	215 (D7h)	F	48 (30h)
E	203 (CBh)	F#	45 (2Dh)
F	192 (C0h)	G	43 (2Bh)
F#	181 (B5h)	G#	40 (28h)
G	171 (ABh)	A	38 (26h)
G#	161 (A1h)	A#	36 (24h)
A	152 (98h)	B	34 (22h)
A#	144 (90h)	C	32 (20h)
B	136 (88h)	C#	30 (1Eh)
C	128 (80h)	D	28 (1Ch)
C#	121 (79h)	D#	27 (1Bh)
D	114 (72h)	E	25 (19h)
D#	107 (6Bh)	F	24 (18h)
E	101 (65h)	F#	23 (17h)
F	96 (60h)	G	21 (15h)
F#	90 (5Ah)	G#	20 (14h)
G	85 (55h)	A	19 (13h)
G#	81 (51h)	A#	18 (12h)
A	76 (4Ch)	B	17 (11h)
A#	72 (48h)	C	16 (10h)
B	68 (44h)	C#	15 (0Fh)
C-Mid	64 (40h)	D	14 (0Eh)
C#	60 (3Ch)		

Table 10. Musical Note Pitch/K_i Values

Choosing note durations and tempo

Table 11 lists the recommended K_D values for each of the standard musical note durations. This convention permits shorter ($1/64$th note) and intermediate note values to be played accurately. This is important when, for example, a thirty-second note is to be played staccato, or a note is dotted (multiplying its length by 1.5).

Using the suggested values, it turns out that most musical scores sound best when played at a tempo of 255 or faster (i.e., $K_{TH} = 0$). Of course, the "right" tempo is the one that sounds the best.

Note Duration	K_D
Whole	192 (C0h)
Half	96 (60h)
Quarter	48 (30h)
Eighth	24 (18h)
Sixteenth	12 (0Ch)
Thirty-second	6 (06h)

Table 11. Musical Note Duration/K_D Values

Play Command

The Play command causes the voice data in the input buffer to begin playing. Additional Initialize commands and Voice frames may be sent to the RC8650 while the tone generator is operating. The TS_0 pin and TS flag are asserted at this time, enabling the host to synchronize to the playing of the tone data. TS becomes inactive after all of the data has been played.

Quit Command

The Quit command marks the end of the tone data in the input buffer. The RC8650 will play the contents of the buffer up to the Quit command, then return to the text-to-speech mode that was in effect when the tone generator was activated. Once the Quit command has been issued, the RC8650 will not accept any more data until the entire buffer has been played.

Example Tune

The Basic program shown in Figure 18 reads tone generator data from a list of DATA statements and PRINTs each value to the RC8650. The astute reader may have noticed some "non-standard" note durations in the DATA statements (cf. Table 11), such as the first two Voice frames in line 260. According to the original music, some voices were not to be played as long as the others during the beat. The F-C-F notes in the first frame are held for 46 counts, while the low F and C in the second frame are held for two additional counts. Adding the duration (first and fifth) bytes together, the low F and C do indeed add up to 48 counts (46 + 2), which is the standard duration of a quarter note.

```
100 OPEN "COM1:9600,N,8,1,BIN" FOR OUTPUT AS #1
110 PRINT #1,                  ' ensure serial port baud rate is locked
120 PRINT #1, CHR$(1);"J";  ' activate tone generator
130 READ B0,B1,B2,B3        ' read a frame (4 bytes)
140 PRINT #1, CHR$(B0); CHR$(B1); CHR$(B2); CHR$(B3);
150 IF B0 + B1 + B2 + B3 > 0 THEN 130  ' loop until Quit
160 CLOSE #1
170 END
180 '
190 '
200 ' Data Tables:
210 '
220 ' Init (volume = 255, tempo = 86)
230 DATA  0,255,86,0
240 '
250 ' Voice data
260 DATA  46,48,64,192, 2,0,64,192, 48,48,0,0, 48,40,0,0, 48,36,0,0
270 DATA  94,24,34,0, 2,24,0,0, 24,0,36,0, 24,0,40,0, 48,0,48,0
280 DATA  48,40,0,192, 46,36,0,0, 2,0,0,0, 48,36,0,0, 48,24,34,0
290 DATA  46,24,34,0, 2,0,34,0, 46,24,34,0, 2,24,0,0, 24,0,36,0
300 DATA  24,0,40,0, 48,0,48,0
310 '
320 ' Play, Quit
330 DATA  0,0,1,1, 0,0,0,0
```

Figure 18. Example Musical Tone Generator Program

SINUSOIDAL TONE GENERATOR

The sinusoidal tone generator generates two sinusoidal waveforms simultaneously. Applications range from creating simple tones to telephone call-progress tones, such as a dial tone or busy signal. The frequency range is 0 to 2746 Hz, with a resolution of 4 to 11 Hz.

The sinusoidal tone generator is activated with the command nJ, where n is an ASCII number between 0 and 99. Note the similarity to the musical tone generator command, J, which uses no parameter. The parameter n programs the internal sampling rate, much like the Real-Time Audio Playback command does; in fact, the sampling rate f_s has the same relationship to n as the Real-Time Audio Playback command:

$$f_s = 617/(155-n) \text{ (kHz)}$$

Immediately following the nJ command are three parameter bytes:

$$n\text{J } K_d \, K_1 \, K_2$$

where K_d determines the tone duration, and K_1 and K_2 set the output frequencies of generators 1 and 2, respectively.

The tone duration and frequencies are not only functions of these parameters, but of n as well. The output amplitude is a function of the Volume command (nV).

The tone duration T_d is calculated as follows:

$$T_d = K_d \times 256/f_s \text{ (sec)}$$

where $0 \le K_d \le 255$. Substituting the relationship $f_s = 617/(155-n)$ into the above equation,

$$T_d = K_d \times (155-n)/2410 \text{ (sec)}$$

Setting $K_d = 1$ yields the shortest duration; $K_d = 0$ (treated as 256) the longest. Depending on the value of n, T_d can range from 23 ms to 16.5 sec.

The tone frequencies F_1 and F_2 are computed as follows:

$$F_i = K_i \times f_s/1024 \text{ (Hz)}$$

where $0 \le K_i \le 255$. Substituting the relationship $f_s = 617/(155 - n)$ into this equation,

$$F_i = K_i \times 603/(155-n) \text{ (Hz)}$$

Depending on the value of n, F_i can range from 0 Hz to 2746 Hz. If only one tone is to be generated, the other tone frequency may be set to 0 ($K_i = 0$), or equal in frequency. Note, however, that due to the additive nature of the tone generators, the output amplitude from both generators running at the same frequency will be twice

that of just one generator running. Both K_1 and K_2 may be set to 0 to generate silence. Longer tones can be created by sending multiple commands, back-to-back.

Note that the frequency step size and frequency range are strictly functions of n. In general, the larger n is, the larger the step size and range will be. The parameter K_i can be thought of as a multiplier, which when multiplied by the step size, yields the output frequency. For example, setting $n = 95$ (corresponding to an internal sampling rate of 10.28 kHz) results in a frequency step size of $603/(155-95)$ Hz, or 10 Hz. Thus, the output frequency range spans 0 Hz to 255 x 10 Hz, or 2550 Hz, in 10 Hz steps.

As an example, suppose your application needed to generate the tone pair 440/350 Hz (a dial tone) for say, 2.5 seconds. We will choose $n = 95$, because it yields a convenient step size of 10 Hz. The tone duration parameter K_d is calculated as follows:

$$K_d = 2410 \times T_d/(155-n)$$

substituting $T_d = 2.5$ (sec) and $n = 95$,

$$K_d = 2410 \times 2.5/(155-95) = 100$$

K_1 (440 Hz) is computed as follows:

$$K_1 = F_1 \times (155-n)/603$$
$$= 440 \times (155-95)/603 = 44$$

In like manner, K_2 (350 Hz) is computed to be 35.

In order to embed the command in a text file, the computed values must be converted into their ASCII equivalents: 100 = "d", 44 = "," and 35 = "#". The complete command becomes

```
CTRL+A "95jd,#"
```

which can be embedded within normal text for the synthesizer.

RCStudio includes a graphically-based tone generator which creates tone generator commands automatically, greatly simplifying the command-creation process.

DTMF TONE GENERATOR

The DTMF (Touch-Tone) tone generator generates the 16 standard tone pairs commonly used in telephone systems. Each tone is 100 ms in duration, followed by a 100 ms inter-digit pause — more than satisfying telephone signaling requirements (both durations can be extended to 500 ms by setting the DDUR bit of the Protocol Options Register). The DTMF generator is activated with the command $n*$, where n is an ASCII number between 0 and 16. The mapping of the command parameter n to the buttons on a standard telephone is shown in Table 12.

The "pause" tone can be used to generate longer inter-digit delays in phone number strings, or to create precise silent periods in the RC8650's output. The generator's output level can be adjusted with the Volume command (nV). DTMF commands may be intermixed with text and other commands without restriction.

n	Button
0	0
.	.
.	.
9	9
10	*
11	#
12	A
13	B
14	C
15	D
16	pause

Table 12. DTMF Dialer Button Map

RC8650 CONTROL

Volume (nV)

This is a global command that controls the RC8650's output volume level. 0V yields the lowest possible volume; maximum volume is attained at 9V. The default volume is 5V. The Volume command can be used to set a new listening level, create emphasis in speech, or change the output level of the tone generators.

Timeout Delay (nY)

The RC8650 defers translating the contents of its input buffer until a CR or NUL is received. This ensures that text is spoken smoothly from word to word and that the proper intonation is given to the beginnings and endings of sentences. If text is sent to the RC8650 without a CR or NUL, it will remain untranslated in the input buffer indefinitely.

The RC8650 contains a programmable timer that is able to force the RC8650 to translate its buffer contents after a preset time interval. The timer is enabled only if the Timeout Delay parameter n is non-zero, the RC8650 is not active (not talking), and the input buffer contains no CR or NUL characters. Any characters sent to the RC8650 before timeout will automatically restart the timer.

The Timeout parameter n specifies the number of 200 millisecond periods in the delay time, which can range from 200 milliseconds to 3 seconds. The default value is 0Y, which disables the timer.

n	Delay
0	Indefinite (wait for CR/NUL)
1	200 milliseconds
2	400 milliseconds
.	.
.	.
15	3000 milliseconds (3 sec.)

Table 13. Timeout Delays

Sleep Timer (nQ)

The sleep timer is used to force the RC8650 into Standby mode after a programmed time interval. For example, the RC8650 can power down automatically if the user forgets to turn off the power at the end of the day. An audible "reminder" tone can even be programmed to sound every ten minutes to remind the user that the power was left on, before shutdown occurs.

The sleep timer is stopped and reset whenever the RC8650 is active, and begins running when the RC8650 enters Idle mode. In this way, the RC8650 will not shut itself down during normal use, as long as the programmed timer interval is longer than the maximum time the RC8650 is inactive.

The command parameter n determines when Standby mode will be entered. You can place the RC8650 in Standby mode immediately, program the sleep timer to any of 15 ten-minute intervals (10 to 150 minutes), or disable the sleep timer altogether (Table 14).

Note that the delay interval is simply n x 10 minutes for $0 < n < 16$. ORing 10h to these values ($16 < n < 32$) also enables the reminder tone, which sounds at the end of each ten minute interval. Programming $n = 0$ disables the sleep timer, which is the default setting. Setting $n = 16$ forces the RC8650 to enter Standby mode as soon as all output has ceased.

If the sleep timer is allowed to expire, the RC8650 will emit the ASCII character "p" from the TXD pin and the STBY status flag will be set to 1, just before entering Standby mode. This enables the host to detect that the RC8650 has entered Standby mode.

Once the RC8650 has entered Standby mode, it can be re-awakened only by a hardware reset or by driving the STBY# pin low for t_{WSBL} or longer, then High again. All of the RC8650 handshake signals (BUSY, CTS#, and RDY#) are forced to their "not ready" states while the RC8650 is in Standby.

n	Delay
0	Sleep timer disabled
1	10 min
.	.
.	.
15	150 min
16	0 (immediate)
17	10 min w/reminder
.	.
.	.
31	150 min w/reminder

Table 14. Sleep Timer

Download Exception Dictionary (247W)

This command initializes the RC8650's exception dictionary and stores subsequent output from the host in the RC8650's nonvolatile dictionary memory. The maximum dictionary size is 16 KB.

Exception dictionaries must be compiled into the format required by the RC8650 before they can be used. *RCStudio*, a Windows-based application available from RC Systems, includes a dictionary editor and compiler for performing this task. Dictionaries that have been compiled with *RCStudio* include the download command in the file header, simplifying the download process.

To download a dictionary, simply download the file to the RC8650. After the first 12 bytes of the file have been transferred, the RC8650 will drive the handshake lines to their false ("not ready") states. This is done to prevent any more of the file from being transferred while the RC8650 prepares the internal memory that will be used to store the dictionary. At this time the RC8650 will transmit two Erase_In_Prog messages (Table 20) followed by Erase_Ok via the TXD pin, each approximately 0.25 sec apart.

After the Erase_Ok message has been transmitted, the handshake lines will return to their true ("ready") states, allowing the remainder of the file to be transmitted. After the entire file has been transferred the RC8650 will transmit the result code, normally No_Error.

Note: It is not necessary to monitor the status messages sent from the RC8650, because the handshake lines ultimately control the timing of the download. The status messages are provided for informational purposes only.

Enable Exception Dictionary (U)

The exception dictionary is enabled with this command. If the RC8650 is in Phoneme mode, or if an exception dictionary has not been loaded, the command will have no effect. The exception dictionary can be disabled by issuing one of the mode commands D, T, or C. The dictionary is disabled by default.

Download Greeting Message (255W)

Anytime the RC8650 is reset, an optional user-defined greeting message is automatically played. The message may consist of any text/command sequence up to 234 characters in length. Modal commands can be included, such as tone generator and audio playback commands (refer to Table 21 for a list of the commands that can be used in greeting messages). For example,

```
CTRL+A "3s" CTRL+A "2o" "ready"
```

programs the RC8650 to use voice speed 3, Big Bob's voice, and say "ready" whenever the RC8650 is reset.

RC Systems' *RCStudio* software can create and download greeting messages for you. To download a greeting message created with *RCStudio*, simply download the file to the RC8650. All of the RC8650 commands required to initiate and terminate the download are embedded within the greeting file, greatly simplifying the downloading process.

Caution: The exception dictionary is erased from memory whenever a new greeting message is written to the RC8650.

Baud Rate (nH)

The serial port's baud rate can be programmed to the rates listed in Table 15. If included as part of the greeting message, the command will effectively override the baud rate set by the BRS pins.

n	Baud Rate
0	300
1	600
2	1200
3	2400
4	4800
5	9600
6	19200
7	Auto-detect
8	38400
9	57600
10	115200
11	Re-read BRS_2-BRS_0 pins

Table 15. Programmable Baud Rates

TS Pin Control (nK)

The TS0 pin provides talk status information that can be used to activate a transmitter, take a telephone off hook, enable an audio power amplifier, etc., at the desired time. The pin's mode and polarity can be configured as shown in Table 16. The programming of the TS_0 pin does not affect the Status Register TS flag in any way. The default setting is 1K.

If the TS_0 pin is programmed Forced High or Forced Low, it will remain in that state until changed otherwise. This feature can be used to activate a transmitter, for example, before speech output has begun. In the automatic mode, the TS_0 pin is asserted as soon as output begins; it will return to its false state when all output has ceased. Note that because RC8650 commands work synchronously, the TS_0 pin will not change state until all text and commands, up to the TS Pin Control command, have been spoken and/or executed.

n	TS Pin Mode/Polarity
0	Automatic/Active Low
1	Automatic/Active High
2	Forced Low
3	Forced High

Table 16. TS Pin Control

Protocol Options Register (nG)

This command controls various internal RC8650 operating parameters. See Table 17 for the definition of each register bit. The default register setting is 128G.

Bit POR.7 (V86) programs the RC8650 to emulate RC Systems' original V8600 voice synthesizer module. When this bit is set to 0, the TTS parameters and ranges are adjusted to match that of the V8600. The serial port status messages (see Table 3) are also affected by the setting of this bit.

V86	SAT	DDUR	R	R	R	R	STM
7	6	5	4	3	2	1	0

Protocol Options Register Bit	Description
POR.7 = V8600 COMPATIBILITY (V86) 1 = Compatibility disabled 0 = Compatibility enabled	Emulates RC Systems' V8600 voice synthesizer module when set to "0." Overall voice speed range and serial port status responses are adjusted to that of the V8600. Default: "1" (in the V8600A module, this bit defaults to "0").
POR.6 = SATURATE (SAT) 1 = Parameters saturate 0 = Parameters wrap	Determines whether command parameters wrap or saturate when their range has been exceeded. Default: "0."
POR.5 = DTMF DURATION (DDUR) 1 = 500 ms 0 = 100 ms	Determines DTMF (Touch-Tone) generator burst duration. When set to "1," tone bursts are 500 ms long; when "0," 100 ms. Default: "0."
POR.4−1 = RESERVED (R)	Reserved for future use. Write "0" to ensure future compatibility.
POR.0 = STATUS MESSAGES (STM) 1 = Enabled 0 = Disabled	Enables and disables the transmission of certain status messages from the TXD pin. Default: "0."

Table 17. Protocol Options Register

Audio Control Register (nN)

The Audio Control Register determines whether the RC8650's audio will be delivered as an analog signal on the AO pins or as a serial data stream on the DAOUT pin. See Table 18 for the definition of each register bit. The default register setting is 0N (analog output).

In the digital audio modes, data is transferred from the DAOUT pin in 8-bit linear, offset binary format (midscale = 80h). The DARTS# pin can be used to regulate the flow of data — it must be Low for transfers to begin. Once a transfer has begun, the current sample will

be transmitted to completion even if DARTS# goes High before the sample has completed. Figure 19 illustrates the synchronous data transfer mode; note how either DARTS# or DACLK can be used to regulate the data flow.

In the synchronous mode, do not attempt to read the data at a rate faster than 12 kbytes/sec (DACLK = 96 kHz). At clock rates above this, the host should pause between reading each sample in order to prevent the average transfer rate from exceeding 12 kbytes/sec.

AM	TM	DPC	TF	TCP	BR	BR	BR
7	6	5	4	3	2	1	0

Audio Control Register Bit	Description
ACR.7 = AUDIO MODE (AM) 1 = Digital 0 = Analog	Set this bit to "0" to direct the audio stream to the AO pin (analog). Set the bit to "1" to direct output to the DAOUT pin (digital). Default: "0."
ACR.6 = TRANSFER MODE (TM) 1 = Synchronous 0 = Asynchronous	In the asynchronous transfer mode the data rate and timing are controlled by the internal bit rate generator (ACR.2−0). Data is output on the DAOUT pin and formatted as 1 start bit, 8 data bits (LSB first), and 1 stop bit. In the synchronous transfer mode the data rate and timing are controlled by the host with the DACLK pin. Data is output from the DAOUT pin as 8-bit data frames. Default: "0."
ACR.5 = DAOUT PIN CONTROL (DPC) 1 = Open-drain 0 = CMOS	Set this bit to "1" to configure the DAOUT pin as an open-drain output, or to "0" for a CMOS output. The open-drain configuration should be used when wire-or'ing two or more DAOUT pins together. Default: "0."
ACR.4 = TRANSFER FORMAT (TF) 1 = MSB first 0 = LSB first	Set this bit to "1" to have the 8-bit data frames transmitted most-significant bit first, or to "0" for least-significant bit first. Valid only in the synchronous transfer mode. Default: "0."
ACR.3 = TRANSFER CLOCK POLARITY (TCP) 1 = Rising edge 0 = Falling edge	Set this bit to "1" to clock data out of the DAOUT pin on the rising edge of the DACLK pin, or to "0" to clock data on the falling edge. Valid only in the synchronous transfer mode. Default: "0."
ACR.2−0 = BIT RATE (BR) 000 = 2400 001 = 4800 010 = 9600 011 = 14400 100 = 19200 101 = 28800 110 = 57600 111 = 115200	These bits determine the bit rate used in the asynchronous transfer mode. Valid only in the asynchronous transfer mode. Default: "000."

NOTES:
1. ACR.6−ACR.0 are valid only when ACR.7 = 1.
2. ACR.4−ACR.3 are valid only when ACR.7 and ACR.6 = 1.
3. ACR.2−ACR.0 are valid only when ACR.7 =1 and ACR.6 = 0.

Table 18. Audio Control Register

DARTS# Starts transmission

Stopped because DACLK stopped   Stopped because DARTS# is High

DACLK

DAOUT   D0 D1 D2 D3 D4 D5 D6 D7   D7 D6 D5 D4 D3 D2 D1 D0

TF = TCP = 0   TF = TCP = 1
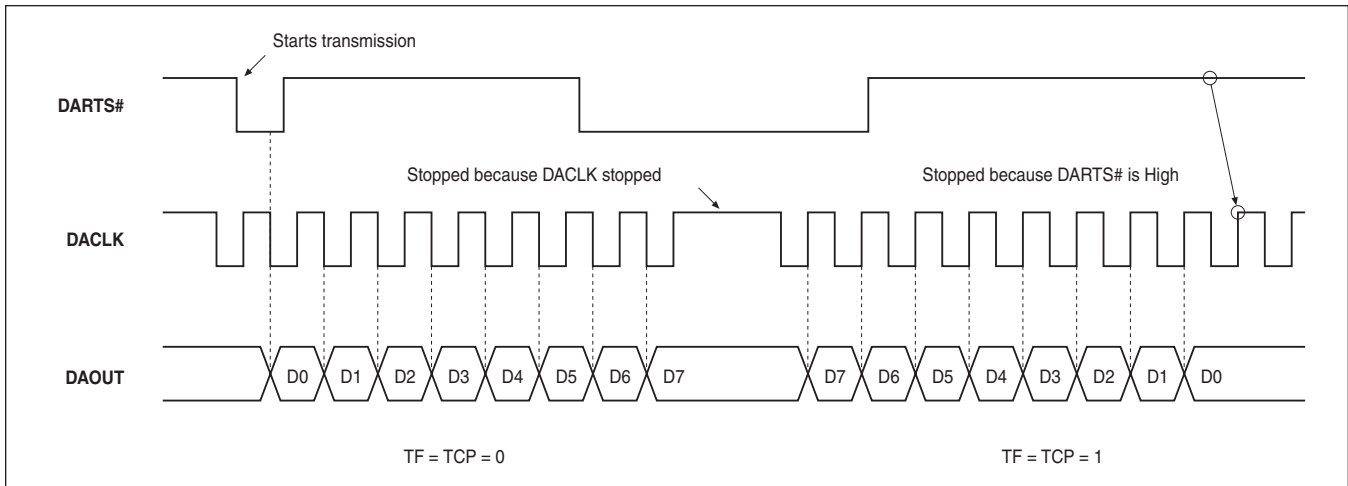
Figure 19. Synchronous Digital Audio Transfer Timing

Stop (CTRL+X), Skip (CTRL+Y)

The Stop command (18h) causes the RC8650 to stop whatever it is doing and flushes the input buffer of all text and commands. The Skip command (19h) skips to the next sentence in the buffer.

Note: The format of the Stop and Skip commands is unique in that the command character (CTRL+A) is not used with them. To be most effective, the states of the RC8650 handshaking signals should be ignored, which allows the RC8650 to react immediately even if its input buffer is full. These commands cannot be used during real-time audio playback nor with the musical tone generator.

Reinitialize (@)

This command reinitializes the RC8650 by clearing the input buffer and restoring the voice parameters and control registers to their default settings. The exception dictionary, prerecorded audio, greeting message, baud rate, nor TS pin control setting are affected.

Zap Commands (Z)

This command prevents the RC8650 from honoring subsequent commands, causing it to read commands as they are encountered (useful in debugging). Any pending commands in the input buffer will still be honored. The only way to restore command recognition after the Zap command has been issued is to write CTRL+^ (1Eh) to the RC8650 or perform a hardware reset.

Index Marker (nI)

Index markers are nonspeaking "bookmarks" that can be used to keep track of where the RC8650 is reading within a passage of text. The parameter *n* is any number between 0 and 99; thus, up to 100 unique markers may be active at any given time.

When the RC8650 has spoken the text up to a marker, it transmits the marker number to the host via the TXD pin. Note that this value is a *binary* number between 0 and 99, not a literal ASCII number string as was used in the command to place the marker. This allows the marker to be transmitted as a one-byte value.

There is no limitation to how many index markers can be used in a text string. The frequency depends on the resolution required by the application. In Text mode, for example, one marker per sentence or one marker per word would normally be used. In Phoneme mode, markers can be placed before each phoneme to monitor phoneme production, which is useful for synchronizing an animated mouth with the voice. Markers may also be placed with tone generation and recorded audio playback commands, if desired.

Chipset Identification (6?)

This command returns RC8650 system information that is used during factory testing. Eight bytes are transmitted via the TXD pin. The only information that may be of relevance to an application is the internal microcode revision number, which is conveyed in the last two bytes in packed-BCD format. For example, 13h 01h is returned if the version number is 1.13.

Interrogate (12?)

This command retrieves the current operating settings of the RC8650. Table 19 lists the parameters in the order they are transmitted from the TXD pin, the command(s) that control each parameter, and each parameter's range. The parameters are organized as a byte array of one byte per parameter.

Parameter	Cmd	Range
Mode	C/D/T	0=Char; 1=Phon; 2=Text
PFR register	nB	0-15
Formant freq	nF	0-9
Pitch	nP	0-99
Speed	nS	0-9
Volume	nV	0-9
Tone	nX	0-2
Expression	nE	0-9
Dict loaded	L	0=not loaded; 1=loaded
Dict status	C/T/U	0=disabled; 1=enabled
Input buffer capacity	–	x256 bytes
Articulation	nA	0-9
Reverb	nR	0-9
TS pin control	nK	0-3
POR register	nG	0-255
ACR register	nN	0-255
Rec memory capacity	–	x64K bytes
Sleep delay	nQ	0-31
Timeout delay	nY	0-15
Char mode delay	nC	0-31
Text mode delay	nT	0-15
Voice	nO	0-7
ADR register	n\$	0-255

Table 19. Interrogate Command

Programming Result Codes

Certain RC8650 programming (sound library, dictionary and greeting message downloads) and sound file playback operations return result codes via the TXD pin. These codes are summarized in Table 20.

Message	Code	Meaning
No_Error	0	No error
Invld_Cmd_Err	1	Invalid command error
Com_Err	2	Communications error
TmOut_Err	3	Comm. timeout error (> 10 sec)
Erase_In_Prog	4	Memory erase in progress
(reserved)	5	Reserved
Erase_Ok	6	Memory erased, no errors
Write_Err	7	Memory write error
Out_Mem_Err	8	Out of memory
CkSum_Err	9	Data checksum error
(reserved)	10	Reserved
(reserved)	11	Reserved
(reserved)	12	Reserved
(reserved)	13	Reserved
Not_Found_Err	14	File not found in memory

Table 20. Programming Result Codes

COMMAND SUMMARY

Command	Function	Greet & Dict *	*n* Range	Default *n*
*n*A	Articulation	•	0-9	5
*n*B	Punctuation Filter Register	•	0-15	6
C/*n*C	Character mode	•	0-31	0
D	Phoneme mode	•	–	–
E/*n*E	Expression	•	0-9	5
*n*F	Formant frequency	•	0-9	5
*n*G	Protocol Options Register	•	0-255	128
*n*H	Baud rate	•	0-11	BRS pins
*n*I	Index marker	•	0-99	–
J	Musical tone generator		–	–
*n*J	Sinusoidal tone generator	•	0-99	–
*n*K	TS pin control	•	0-3	1
M	Monotone	•	–	–
*n*N	Audio Control Register		0-255	0
*n*O	Voice	•	0-7	0
*n*P	Pitch	•	0-99	50
*n*Q	Sleep timer	•	0-31	0
*n*R	Reverb	•	0-9	0
*n*S	Speed	•	0-9	1
T/*n*T	Text mode/delay	•	0-15	0
U	Enable exception dictionary	•	–	–
*n*V	Volume	•	0-9	5
223W	Download sound library		223	–
247W	Download exception dictionary		247	–
255W	Download greeting message		255	–
*n*X	Tone	•	0-2	1
*n*Y	Timeout delay	•	0-15	0
Z	Zap commands	•	–	–
@	Reinitialize	•	–	–
*n**	DTMF generator	•	0-16	–
n#/*n*%	Real-time PCM/ADPCM audio playback		0-99	–
n&	Play sound file	•	0-999	–
n$	ADC Control Register	•	0-255	128
6?	Chipset ID		6	–
12?	Interrogate		12	–
CTRL+X/Y	Stop/Skip		–	–

* These commands can be used in greeting messages and exception dictionaries.

Table 21. RC8650 Command Summary

EXCEPTION DICTIONARIES

Exception dictionaries make it possible to alter the way the RC8650 interprets character strings it receives. This is useful for correcting mispronounced words, triggering the generation of tones and/or the playback of prerecorded sounds, or even speaking in a foreign language. In some cases, an exception dictionary may even negate the need of a text pre-processor in applications that cannot provide standard text strings. This section describes how to create exception dictionaries for the RC8650.

The text-to-speech modes of the RC8650 utilize an English lexicon and letter-to-sound rules to convert text it receives into speech. The pronunciation rules determine which sounds, or phonemes, each character will receive based on its relative position within each word. The integrated DoubleTalk text-to-speech engine analyzes text by applying these rules to each word or character, depending on the operating mode in use. Exception dictionaries augment this process by defining exceptions for (or even replacing) these built in rules.

Exception dictionaries can be created and edited with a word processor or text editor that stores documents as standard text (ASCII) files. The dictionary must be compiled into the internal binary format used by the RC8650 before it can be used, however. The *RCStudio* software, available from RC Systems, includes a dictionary editor and compiler.

Exception Syntax

Exceptions have the general form

    L(F)R=P

which means "the **text fragment** F, occurring with **left context** L and **right context** R, gets the **pronunciation** P." All three parts of the exception to the left of the equality sign must be satisfied before the text fragment will receive the pronunciation given by the right side of the exception.

The text fragment defines the input characters that are to be translated by the exception, and may consist of any combination of letters, numbers, and symbols. Empty (null) text fragments can be used to generate sound based on a particular input pattern, without actually translating any of the input text. The text fragment (if any) must always be contained within parentheses.

Characters to the left of the text fragment specify the left context (what must come before the text fragment in the input string), and characters to the right define the right context. Both contexts are optional, so an exception may contain neither, either, or both contexts. There are also 15 special symbols, or **context tokens**, that can be used in an exception's context definitions (Table 22).

Note that although context tokens are, by definition, valid only for the left and right context definitions, the wildcard token may also be used within text fragments. *Any other context token appearing within a text fragment will be treated as a literal character.*

The right side of an exception (P) specifies the pronunciation that the text fragment is to receive, which may consist of any combination of phonemes (Table 5), phoneme attribute modifiers (Table 6), and commands (Table 21). Using the tone generator and prerecorded audio playback commands, virtually limitless combinations

Symbol	Definition
#	A vowel: a, e, i, o, u, y
+	A front vowel: e, i, y
^	A consonant: b, c, d, f, g, h, j, k, l, m, n, p, q, r, s, t, v, w, x, z
*	One or more consonants
:	Zero or more consonants
?	A voiced consonant: b, d, g, j, l, m, n, r, v, w, z
@	One of: d, j, l, n, r, s, t, z, ch, sh, th
!	One of: b, c, d, f, g, p, t
%	A suffix: able(s), ably, e(s), ed(ly), er(s), ely, eless, ement(s), eness, ing(s), ingly (must also be followed by a non-alphabetic character)
&	A sibilant: c, g, j, s, x, z, ch, sh
$	A nonalphabetic character (number, space, etc.)
~	One or more non-printing characters (spaces, controls, line breaks, etc.)
\	A digit: 0-9
|	One or more digits (commas are ignored)
`	Wildcard (matches any character)

Table 22. Context Tokens

of speech, tones, and sound effects can be triggered from any input text pattern. If no pronunciation is given, no sound will be given to the text fragment; the text fragment will be silent.

A dictionary file may also contain comments, but they must be on lines by themselves (i.e., they cannot be on the same line as an exception). Comment lines must begin with a semicolon character (;), so the compiler will know to skip over them.

An example of an exception is

    C(O)N=AA

which states that o after c and before n gets the pronunciation AA, the o-sound in cot. For example, the o in c<u>on</u>ference, ec<u>on</u>omy, and ic<u>on</u> would be pronounced according to this exception.

Another example is

    $R(H)=

which states that h after initial r is silent, as in the word <u>rh</u>yme (the $ context token represents any non-alphabetic character, such as a space between words; see Table 22).

Punctuation, numbers, and most other characters can be redefined with exceptions as well:

    (5)=S I NG K O   (Spanish five)
    (CHR$)=K EH R IX K T ER (Basic function)

The Translation Algorithm

In order to better understand how an exception dictionary works, it is helpful to understand how the DoubleTalk text-to-speech engine processes text.

Algorithms within the DoubleTalk engine analyze input text a character at a time, from left to right. A list of pronunciation rules is searched sequentially for each character until a rule is found that matches the character in the correct position and context. The algorithm then passes over the input character(s) bracketed in the rule (the text fragment) and assigns the pronunciation given by the right side of the rule to them. This process continues until all of the input text has been converted to phonetic sounds.

The following example illustrates how the algorithm works by translating the word *receive*.

The algorithm begins with the letter r and searches the R pronunciation rules for a match. The first rule that matches is `$(RE)^#=R IX`, because the r in receive is an initial r and is followed by an e, a consonant (c), and a vowel (e). Consequently, the text fragment *re* receives the pronunciation `R IH`, and the scan moves past *re* to the next character: re*c*eive. (E is not the next scan character because it occurred inside the parentheses with the r; the text fragment *re* as a whole receives the pronunciation `R IX`)

The first match among the C rules is `(C)+=S`, because c is followed by an e, i, or y. *C* thus receives the pronunciation `S`, and processing continues with the second e: rec*e*ive.

`(EI)=IY` is the first rule to match the second e, so *ei* receives the sound `IY`. Processing resumes at the character recei*v*e, which matches the default V rule, `(V)=V`.

The final e matches the rule `#:(E)$=`, which applies when e is final and follows zero or more consonants and a vowel. Consequently, *e* receives no sound and processing continues with the following word or punctuation, if any. Thus, the entire phoneme string for the word receive is `R IX S IY V`.

Rule Precedence

Since DoubleTalk uses its translation rules in a sequential manner, the position of each exception relative to the others must be carefully considered. For example, consider the following pair of exceptions:

```
(O)+=OW
(O)=UW
```

The first exception states that o followed by e, i, or y is to be pronounced `OW`, the o-sound in boat. The second exception does not place any restriction on what must come before or after o, so o in any context will receive the `UW` pronunciation. If the exceptions were reversed, the `(O)+` exception would never be reached because the `(O)` exception will always match o in any context. In general, tightly-defined exceptions (those containing many context restrictions) should precede loosely-defined exceptions (those with little or no context definitions).

```
(RAT)=R AE T
(RATING)=R EY T IH NG
(R)=R
```

This is an example of how *not* to organize a dictionary. The exception `(RATING)` will never be used because `(RAT)` will always match first. According to these exceptions, the word *rating* would be pronounced "rat-ing."

It can be beneficial to group exceptions by the first character of the text fragments, that is, all of the A exceptions in one group, all the B exceptions in a second group, and so on. This gives an overall cleaner appearance, and can prove to be helpful if the need arises to troubleshoot any problems in your dictionary.

Text Not Matched by the Dictionary

It is possible that some input text may not match anything in the dictionary, depending on the nature of the dictionary. For example, if a dictionary was written to handle unusual words, only those words would be included in the dictionary. On the other hand, if a dictionary defined the pronunciation for another language, it would be comprehensive enough to handle all types of input. In any case, *if an exception is not found for a particular character, the English pronunciation will be given to that character according to the built in pronunciation rules.*

Generally, the automatic switchover to the built in rules is desirable if the dictionary is used to correct mispronounced words, since by definition the dictionary is defining exceptions to the built in rules. If the automatic switchover is not desired, however, there are two ways to prevent it from occurring. One way is to end each group of exceptions with an unconditional exception that matches any context. For example, to ensure that the letter "a" will always be matched, end the A exception group with the exception `(A)=pronunciation`. This technique works well to ensure matches for specific characters, such as certain letters or numbers.

If the exception dictionary is to replace the built in rules entirely, end the dictionary with the following exception:

```
()=
```

This special exception causes unmatched characters to be ignored (receive no sound), rather than receive the pronunciation defined by the built in rules.

Effect on Punctuation

Punctuation defined in the exception dictionary has priority over the Punctuation Filter command. Any punctuation defined in the dictionary will be used, regardless of the Punctuation Filter setting.

Note If the dollar sign character ($) is defined within the text fragment of any exception, currency strings will not be read as dollars and cents.

Character Mode Exceptions

Exceptions are defined independently for the Character and Text modes of operation. The beginning of the Character mode exceptions is defined by inserting the letter C just before the first Character mode exception. No exceptions prior to this marker will be used when the RC8650 is in Character mode, nor will any exceptions past the marker be used in Text mode. For example:

```
.              (Text mode exceptions)
.
()=            (optional; used if built in rules are not to be used
               in no-match situations)

C              (Character mode exceptions marker)
.
.              (Character mode exceptions)
.
()=            (optional; used if built in rules are not to be used
               in no-match situations)
```

Applications

The following examples illustrate some ways in which the exception dictionary can be used.

Correcting mispronounced words

Correcting mispronounced words is the most common application for exception dictionaries.

```
S(EAR)CH=ER
$(OK)$=OW K EY
```

The first exception corrects the pronunciation of all words containing *search* (search, searched, research, etc.). As this exception illustrates, it is only necessary to define the problem word in its root form, and only the part of the word that is mispronounced (*ear*, in this case). The second exception corrects the word *ok*, but because of the left and right contexts, will not cause other words (joke, look, etc.) to be incorrectly translated.

No cussing, please

The reading of specific characters or words can be suppressed by writing exceptions in which an alternate pronunciation is given.

```
(????)=
(????)=\015i \0166j\0E\94\00
```

The first example simply says nothing when the defined word is encountered. A short 1 kHz tone is played in place of the word in the second example, and the application is notified by transmitting 05h on the TXD pin. Exceptions such as these may be duplicated for any number of words.

When zero isn't really zero

When reading addresses or lists of numbers, the word "oh" is often substituted for the digit 0. For example, we might say 1020 North Eastlake as "one oh two oh North Eastlake." The digit 0 can be redefined in this manner with the following exception:

```
(0)=OW
```

Acronyms and abbreviations

Acronyms and abbreviations can be defined so the words they represent will be spoken.

```
$(KW)$=K IH L AH W AA T
$(DR)$=D AA K T ER
$(TV)$=T EH L AX V IH ZH IX N
```

String parsing & decryption

Sometimes the data that we would like to have read is not available in a "ready-to-read" format. For example, the output of a GPS receiver may look something like this:

```
$GPGGA,123456,2015.2607,N,...
```

The first 14 characters of the string consists of a fixed header and variable time data, which we would like to discard. The following exception ensures that the header will not be read:

```
($GPGGA,``````,)=
```

Note how wildcard tokens are used for handling the time data ($8^{th}-13^{th}$ characters), since the content of this field is variable.

The $15^{th}-16^{th}$ (*20*) and $17^{th}-18^{th}$ (*15*) characters represent the latitudinal coordinate in degrees and minutes, respectively. The three exceptions shown below handle the latitudinal component of the GPS string. Note in the first exception how a null text fragment is used in the appropriate position to generate the word "degrees," without actually translating any of the input characters.

```
,\\()\\.=D IX G R IY Z , ,
(.)=M IH N IH T S , ,
(,N,)=N OW R TH   L AE T IH T UW D
```

The four exceptions together will translate the example string as "20 degrees, 15 minutes, north latitude." (Additional exceptions for handling the seconds component, and digits themselves, are not shown for clarity).

Heteronyms

Heteronyms are words that have similar spellings but are pronounced differently, depending on the context, such as *read* ("reed" and "red") and *wind* ("the wind blew" and "wind the clock"). Exceptions can be used to fix up these ambiguities, by including non-printing (Control) characters in the text fragment of the exception.

Suppose a line of text required the word "close" to be pronounced as it is in "a close call," instead of as in "close the window." The following exception changes the way the *s* will sound:

```
(\04CLOSE)=K L OW S
```

Note the CTRL+D character (\04) in the text fragment. CTRL+D is a non-printing character, but the translation algorithms treat it as they would any printing character. Thus, the string "\04close" will be pronounced with the *s* receiving the "s" sound. "Close" without the CTRL+D will be unaffected — the *s* will still receive the "z" sound. It does not matter where you place the Control character in the word, as long as you use it the same way in your application's text. You may use any non-printing character (except LF and CR) in this manner.

Non-English languages

Dictionaries can be created that enable the RC8650 to speak in languages other than English. It's not as difficult as it may seem — in most cases all that is required is a pronunciation guide and a bit of patience. If you don't have a pronunciation guide for the language you're interested in, check your local library. Most libraries have dictionaries for other languages that include pronunciation guides,

which make it easy to transcribe the pronunciation rules into exception form. A good example of an exception dictionary for the Spanish language is included with *RCStudio*.

Language translation
Exception dictionaries even allow text written in one language to be read by the RC8650 in yet another language, as long as the vocabulary is limited. The following exceptions demonstrate how this can be done with three example Spanish/English words.

```
(GRANDE)=L AA R J
(BIEN)=F AY N
(USTED)=YY UW
```

The sense of translation can also be reversed:

```
(LARGE)=G RR A N D EI
(FINE)=B I EI N
(YOU)=U S T EI DH
```

Play recordings from text
The playback of any recording stored in recording memory can be triggered by text strings defined in the dictionary. Say for example the words "good" and "morning" have been recorded and are accessed by indexes 3 and 7, respectively. The following exceptions will cause the RC8650 to play "good morning" from the recording memory when the text "good morning" is received.

```
(GOOD)=\013&
(MORNING)=\017&
```

Because the recordings can be created from virtually any source, it is easy to change the content, gender, language, etc. If the application's vocabulary is limited and pre-defined, a sort of "pseudo-TTS" system can be created using this method, concatenating words as required to make complete sentences.

Message macros
Certain applications may not be able to send text strings to the RC8650. An example of such an application is one that is only able to output a four bit control word and strobe. Sixteen unique output combinations are possible, but this is scarcely enough to represent the entire ASCII character set.

You can, however, assign an entire spoken phrase to a single ASCII character with the exception dictionary. By driving four of the data bus lines of the bus interface (see Figure 5) and hardwiring the remaining four to the appropriate logic levels, virtually any set of 16 ASCII characters can be generated, which in turn can be interpreted by the exception dictionary.

For example, by connecting the four control bits to DB_0 through DB_3, DB_4 and DB_5 to V_{CC}, DB_6 and DB_7 to ground and the strobe to WR#, ASCII codes 30h through 3Fh (corresponding to the digits "0" through "9" and the six ASCII characters following them) can

be generated by the four control bits. Message strings would then be assigned to each of these ASCII characters. For example, the character "0" (corresponding to all four control bits = 0) would say, "please insert quarter," with the following dictionary entry:

```
(0)=P L IY Z   IH N S ER T   K W OW R T ER
```

The Timeout timer should also be activated (1Y, for example) in order for the "message" to be executed. Otherwise, the RC8650 will wait indefinitely for a CR/NUL character that will never come. The timer command can be included in the greeting message so that it will be automatically executed after reset.

Tips
Make sure that your exceptions aren't so broad in nature that they do more harm than good. Exceptions intended to fix broad classes of words, such as word endings, are particularly notorious for ruining otherwise correctly pronounced words.

Take care in how your exceptions are organized. Remember, an exception's position relative to others is just as important as the content of the exception itself.

Additional information and examples are included with *RCStudio* and *RCStudio's* online help.

If things don't work as expected
On rare occasions, an exception may not work as expected. This can occur when the built in pronunciation rules get control before the exception does. The following example illustrates how this can happen.

Suppose an exception redefined the o in the word "process" to have the long "oh" sound, the way it is pronounced in many parts of Canada. Since the word is otherwise pronounced correctly, the exception redefines only the "o:"

```
PR(O)CESS=OW
```

But to our dismay, the RC8650 simply refuses to take on the new Canadian accent.

It so happens that the RC8650 has a built in rule which looks something like this:
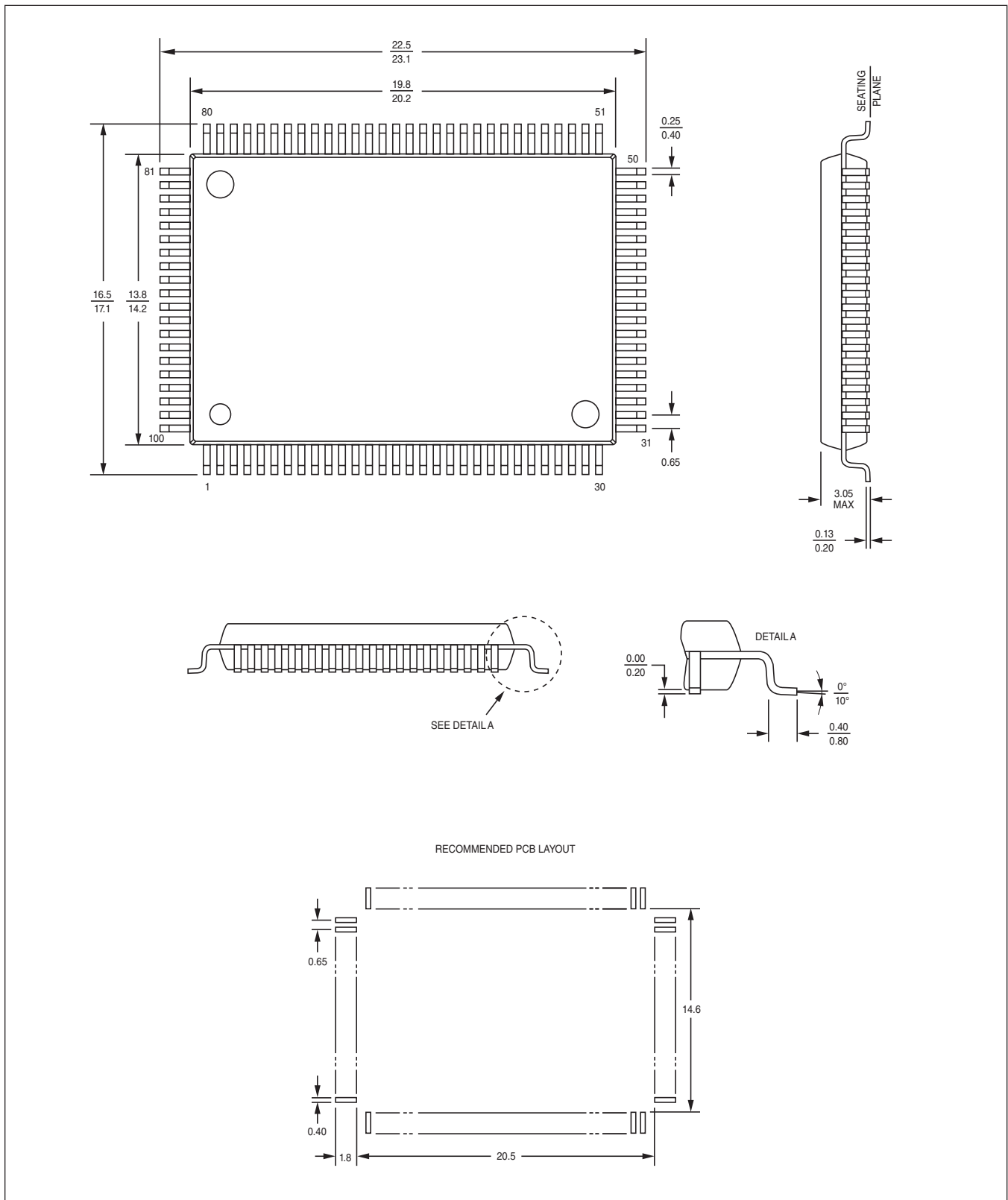
```
$(PRO)=P R AA
```

This rule translates a group of three characters, instead of only one as most of the built in rules do. Because the text fragment PRO is translated as a group, the o is processed along with the initial "pr," and consequently the exception never gets a shot at the o.
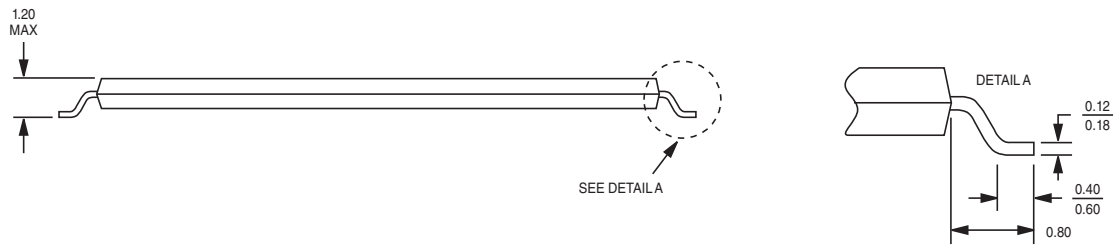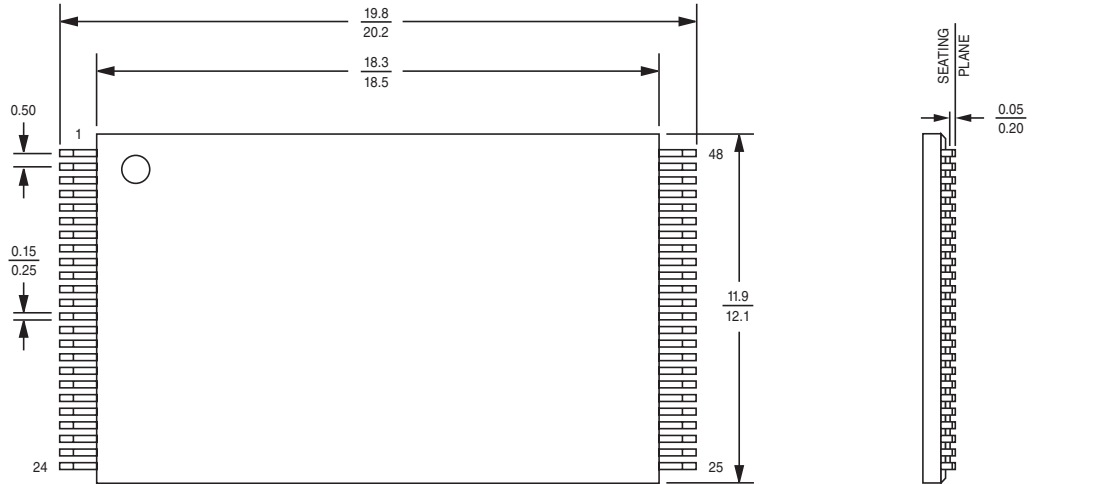
If you suspect this may be happening with one of your exceptions, include more of the left-hand side of the word in the text fragment (in the example above, (PRO)CESS=P R OW would work).
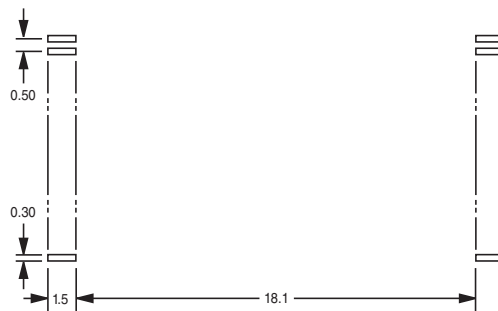
PACKAGE INFORMATION

RECOMMENDED PCB LAYOUT

48 Pin Plastic 12 x 20 mm TSOP (measured in millimeters)

19.8
20.2

18.3
18.5

0.50

1

48

0.15
0.25

11.9
12.1

24

25

SEATING
PLANE

0.05
0.20

1.20
MAX

SEE DETAIL A

DETAIL A

0.12
0.18

0.40
0.60

0.80

RECOMMENDED PCB LAYOUT

0.50

0.30

1.5

18.1

RC8650 EVALUATION KIT

NOTICE: THE RC8650 EVK HAS BEEN REPLACED WITH THE RC8660 EVK, WHICH CAN BE USED TO DEVELOP BOTH RC8650 AND RC8660-BASED APPLICATONS. SEE THE RC8660 USER'S MANUAL FOR FURTHER DETAILS.

The RC8650 Evaluation Kit comes with everything required to evaluate and develop applications for the RC8650 chipset using a Windows-based PC. The included *RCStudio™* software provides an integrated development environment with the following features:

- Read any text, either typed or from a file
- Easy access to the various RC8650 voice controls
- Manage collections of sound files and store them in the RC8650
- Exception dictionary editor/compiler, and much more...

The evaluation board can also be used in stand-alone environments by simply printing the desired text and commands to it via the on-board RS-232 serial or parallel ports.

Evaluation Kit Contents
The following components are included in the DoubleTalk RC8650 Evaluation Kit:

- Printed circuit board containing the RC8650F1C chipset
- AC power supply
- Speaker
- Serial cable
- *RCStudio™* development software CD

EVAL BOARD OUTLINE
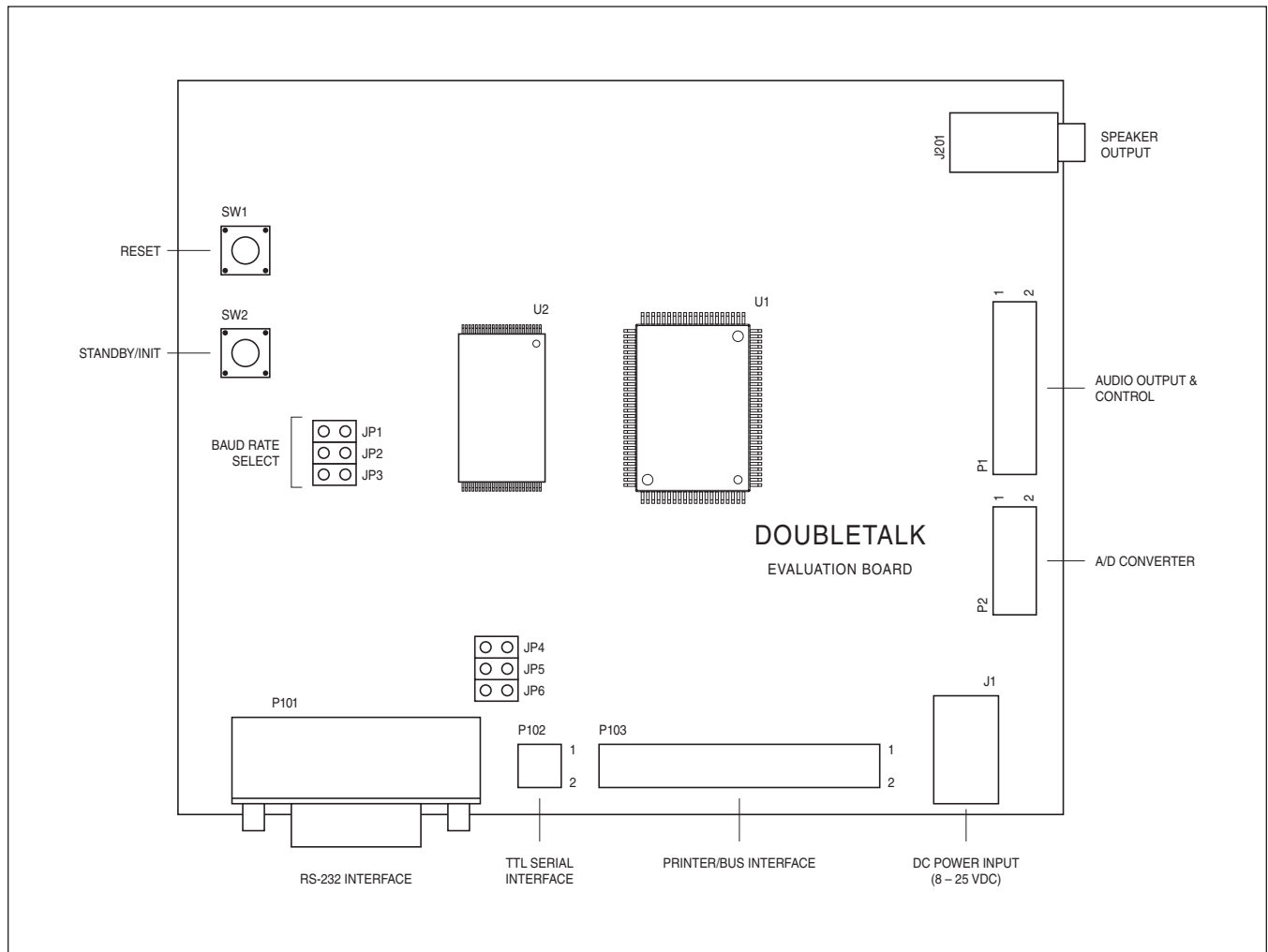
CONNECTOR PIN ASSIGNMENTS & SCHEMATICS

Pin No.	Pin Name	Pin No.	Pin Name
1	AO_0	9	AS_0
2	AO_1	10	AS_1
3	SP_{+0}	11	$SUSP_0\#$
4	SP_{+1}	12	$SUSP_1\#$
5	SP_{-0}	13	DAOUT
6	SP_{-1}	14	DARTS#
7	TS_0	15	DACLK
8	TS_1	16	GND

Table 23. P1 Pin Assignments (Audio Output & Control)

Pin No.	Pin Name	Pin No.	Pin Name
1	AN_0	6	GND
2	GND	7	AN_3
3	AN_1	8	GND
4	GND	9	ADTRG
5	AN_2	10	GND

Table 24. P2 Pin Assignments (A/D Converter)

JP3	JP2	JP1	Baud Rate
X	X	X	300
X	X		600
X		X	1200
X			2400
	X	X	4800
	X		9600
		X	19200
			Auto-detect (default)

Table 25. JP1-JP3 Jumper Assignments (Baud Rate)

Pin No.	Pin Name	Pin No.	Pin Name
1	NC	6	DSR
2	RXD	7	RTS
3	TXD	8	CTS
4	NC	9	NC
5	GND	–	–

Table 26. P101 Pin Assignments (RS-232 Serial Interface)

Pin No.	Pin Name	Pin No.	Pin Name
1	GND	3	TXD
2	CTS#	4	RXD

Note: JP5-JP7 must be open in order to use the TTL interface

Table 27. P102 Pin Assignments (TTL Serial Interface)

Pin No.	Pin Name	Pin No.	Pin Name
1	STB#	14	GND
2	AFD#	15	DATA6
3	DATA0	16	GND
4	ERROR#	17	DATA7
5	DATA1	18	GND
6	INIT#	19	ACK#
7	DATA2	20	GND
8	SLCTIN#	21	BUSY
9	DATA3	22	GND
10	GND	23	PE
11	DATA4	24	GND
12	GND	25	SLCT
13	DATA5	26	RD#

Table 28. P103 Pin Assignments (Printer/Bus Interface)
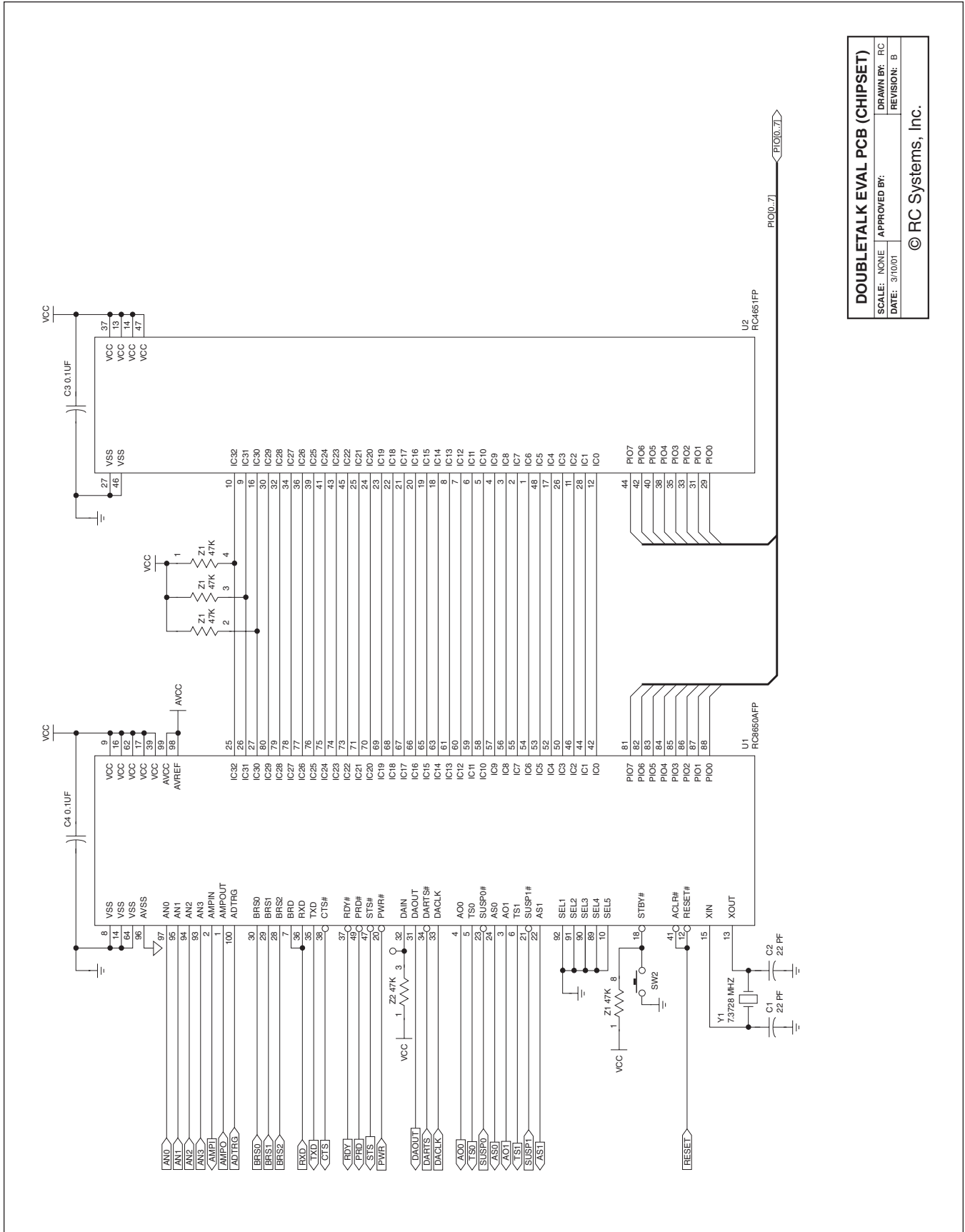
DOUBLETALK EVAL PCB (CHIPSET)

SCALE: NONE
DATE: 3/10/01

DRAWN BY: RC
APPROVED BY:
REVISION: B

© RC Systems, Inc.

DOUBLETALK EVAL PCB (I/F)

SCALE: NONE
DATE: 3/10/01
DRAWN BY: RC
APPROVED BY:
REVISION: B

© RC Systems, Inc.

NOTE
P103 MAY BE CONNECTED DIRECTLY TO A PC COMPATIBLE PARALLEL PORT VIA A RIBBON CABLE WITH A 26-PIN DUAL ROW SOCKET CONNECTOR TO A DB25 MALE CONNECTOR.

DOUBLETALK EVAL PCB (AUDIO)

SCALE:	NONE	APPROVED BY:	DRAWN BY:	RC
DATE:	3/10/01		REVISION:	B

© RC Systems, Inc.

FILTER/AMP—CHANNEL 2

FILTER/AMP—CHANNEL 1

A/D CONDITIONING
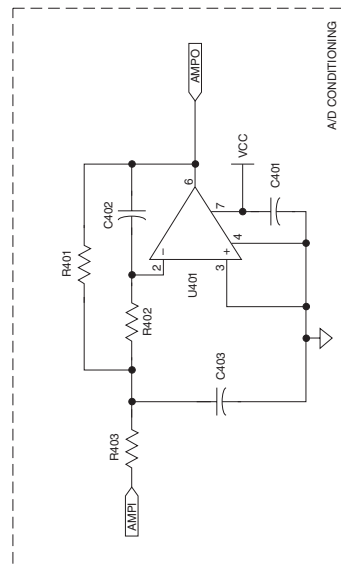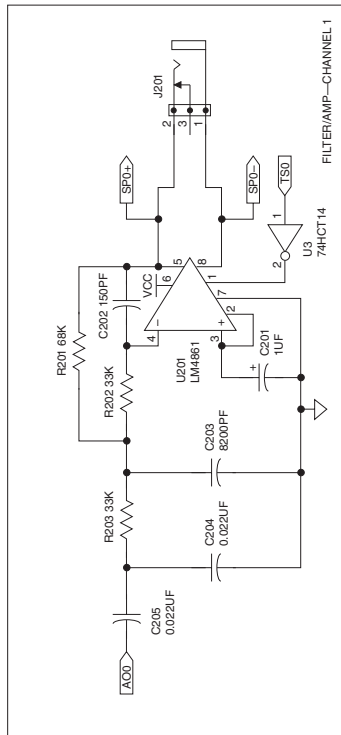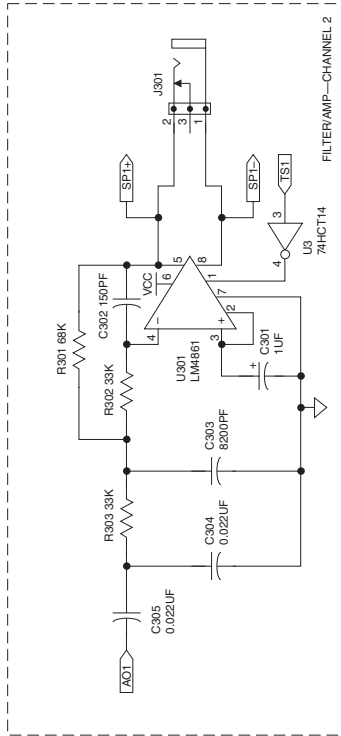
— — — = COMPONENTS NOT INSTALLED

45

APPENDIX A: RESTORE MODE

The RC8650 has a special "Restore" mode[1] which supports the following functions:

- Restoration of the internal system software (firmware)
- Clearing the exception dictionary and greeting message
- Quick verification of the RC8650's operation

How to enter Restore mode
To enter Restore mode, perform the following steps. The procedure is easiest to accomplish if pushbutton switches are connected to the RESET# and STBY# pins, similar to that found on the RC8650 evaluation board and V-Pod.

1) Hold the RESET# and STBY# pins Low. If power is not yet applied and reset is generated automatically at power-up, only STBY# needs to be held Low.

2) Set RESET# High (or simply apply power), while continuing to hold the STBY# pin Low. After a minimum of 0.5 sec has elapsed, set the STBY# pin High. One beep should be heard from the speaker or headphones, indicating that the RC8650 is in Restore mode.

Restoring the RC8650's firmware
RC8650 firmware updates can be performed with *RCLink,* an application included with the *RCStudio* software. If an update failed for one reason or another, the RC8650 may have been left in a non-functional state. In such a case, use *RCLink* to download the firmware while the RC8650 is in Restore mode. Make sure to set the baud rate in *RCLink* to 9600, because the RC8650 only operates at 9600 baud while in Restore mode, regardless of the state of the BRS pins. Please contact RC Systems support to obtain the latest firmware image file.

Clearing the dictionary and greeting message
Use this procedure in the unlikely event that a dictionary or greeting message you downloaded rendered the RC8650 inoperable.

1) While in Restore mode, set the STBY# pin Low and continue holding it Low until two beeps are heard from the speaker or headphones (approximately 3 seconds).

2) Set STBY# High; you should hear the firmware version read through the speaker or headphones. The dictionary and greeting message are now cleared from memory.

Verifying the RC8650's operation
This procedure does not alter the RC8650 in any way; it simply allows you to determine if the RC8650 and audio amplifier circuitry are working. If everything is functioning properly, you should hear the RC8650 firmware version read from the speaker or headphones.

1) While in Restore mode, set the STBY# pin Low momentarily, then High again. Do not hold it Low too long (less than 3 seconds), or the dictionary and greeting message will be cleared from memory.

[1] Prior to version 2.50 of the RC8650 firmware, Restore mode functioned quite differently. To clear the dictionary and greeting message with pre-v2.50 versions of the firmware, make sure that the RC8650 is in Idle mode, then drive the STBY# pin Low for less than 250 ms. The dictionary and greeting message will be immediately cleared, and the RC8650 will announce the firmware version. The firmware version can also be read programmatically with the Chipset ID (6?) command. The firmware restoration process is the same with all versions of the firmware.

APPENDIX B: REVISION HISTORY

REVISION 01 (MARCH 23, 2006)

Updated to reflect industrial operating temperature range option.

Specifications

Added −40 °C to +85 °C industrial operating temperature range.

Ordering Information

Revised part number system to include industrial temperature range option.

RC8650 Commands

Changed "Rec memory capacity" in Table 19 (Interrogate command), from 16K blocks to 64K blocks.

REVISION 02 (APRIL 18, 2007)

Updated for new RC8650 die.

Specifications

Changed I_{CC} supply current spec in DC Characteristics. Active current has been reduced by approximately 40%.

Ordering Information

Several part numbers previously available only in commercial grade have been replaced with industrial grade parts.

REVISION 03 (JUNE 27, 2008)

Updated to reflect new Restore mode. Reorganized certain sections and added Appendix A.

Specifications

Updated STBY# pin description in Table 1.

REVISION 04 (MARCH 18, 2009)

Updated Real-Time Audio Playback mode section.

REVISION 05 (MAY 22, 2009)

Rearranged several sections and updated TOC.

REVISION 06 (DECEMBER 19, 2011)

Removed RC86L50F2I from product offerings. Please use RC86L50F3I instead.

REVISION 07 (AUGUST 9, 2013)

Updated Table 21 (Command Summary).

Sinusoidal Tone Generator (Firmware v2.59)

The sampling rate of the sinusoidal tone generator has been fixed to 10 kHz, matching that of the DTMF generator and TTS synthesizer. Even though the output sampling rate no longer varies with the command's parameter n, the equations given for T_d, F_i, K_d and K_i are still valid and should continue to be used. The command parameter n is still valid for computing these variables. Applications that use the tone generator should continue to work as before without any changes.

REVISION 08 (FEBRUARY 7, 2014)

Updated Table 8 (Punctuation Filter Register). Removed RC8650F0C and RC86L50F0C from product offerings. Please use RC8650F1C or RC86L50F1I instead.

RC8650 Commands (Firmware v2.60)

The way four-digit numbers are read can now be controlled with the Punctuation Filter Register's new YM bit. See Punctuation Filter Register description for more information.

RC Systems
now you're talking!