DoubleTalk RC8660

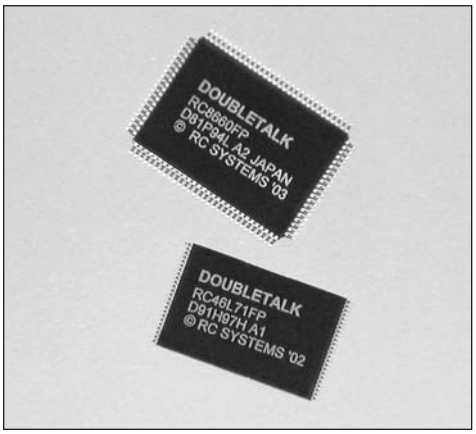
CMOS, 3.3 Volt / 5 Volt
Voice Synthesizer Chipset

GENERAL DESCRIPTION

The RC8660 is a versatile voice and sound synthesizer, integrating a text-to-speech (TTS) processor, audio recording and playback, musical and sinusoidal tone generators, telephone dialer and A/D converter, into an easy to use chipset. Using a standard serial or eight bit bus interface, virtually any ASCII text can be streamed to the RC8660 for automatic conversion into speech by the TTS processor. The audio record and playback modes augment the TTS processor for applications requiring very high voice quality and a relatively small, fixed vocabulary, applications requiring special sounds or sound effects, and/or the recording of voice memos. 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.

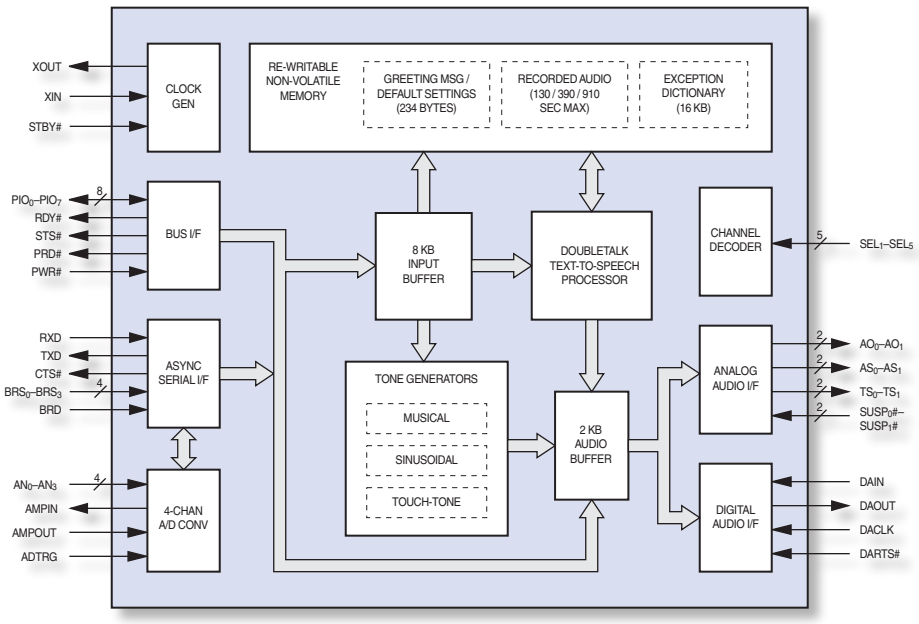
The RC8660's integrated TTS processor incorporates RC Systems' DoubleTalk™ TTS technology, which is based on a unique voice concatenation technique using real human voice samples. The DoubleTalk TTS processor also gives the user unprecedented real-time control of the speech signal, including pitch, volume, tone, speed, expression, articulation, and so on.

Up to 3.5 MB of nonvolatile memory is included in the RC8660 for the storage of up to 15 minutes of recorded messages and sound effects. A programmable "greeting" message can be stored that is automatically played whenever the RC8660 is powered up, allowing a custom message to be played or the RC8660's default settings to be reconfigured. A user-programmable dictionary allows the pronunciation of virtually any character string to be redefined, or even trigger the playback of tones, pre-recorded 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 RC8660 has been integrated into the end-product.



The RC8660 is comprised of two surface-mounted devices. Both operate from 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.

RC8660 FUNCTIONAL BLOCK DIAGRAM



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.
- On-chip recording, storage and playback of sound files:
  - Record to chip via microphone
  - Upload, download, and erase recordings and sound files, even in the field
  - Data logging mode allows analog quantities to be sampled and stored for later retrieval
  - Recording times from 130 sec to 15 min available
- 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
- Serial and 8 bit 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
- 8 KB input buffer for virtually no-overhead operation
- Available in 3.3 V and 5 V versions
- Low power (typ @ 3.3 V):
  - 6 mA active
  - 600 µA idle
  - 100 nA standby

APPLICATIONS

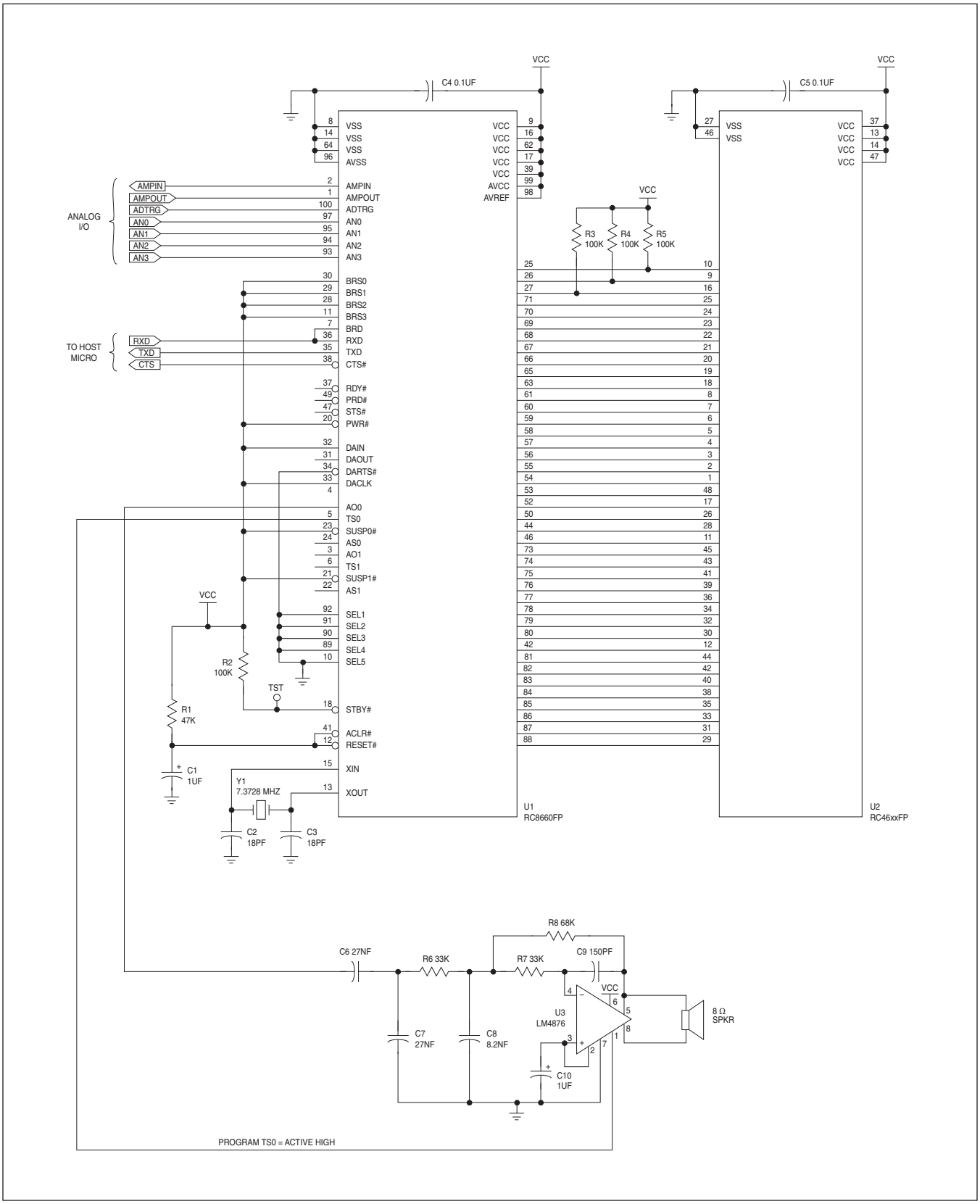
- Robotics
- Talking OCR systems
- ATM machines
- Talking pagers and PDAs
- GPS navigation systems
- Vending and ticketing machines
- Remote diagnostic reporting
- Dial-up information systems
- Handheld barcode readers
- Electronic test and measurement
- Security systems
- Aids for the orally or visually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

RC8660 Product Summary

Part Number	Recording Capacity *	Operating Voltage
RC8660-1	130 sec	5 V
RC86L60-1	130 sec	3.3 V
RC86L60-2	390 sec	3.3 V
RC86L60-3	910 sec	3.3 V

* Based on 8 kHz sampling rate with ADPCM encoding

TYPICAL APPLICATION CIRCUIT



SECTION 1: SPECIFICATIONS

PINOUTS

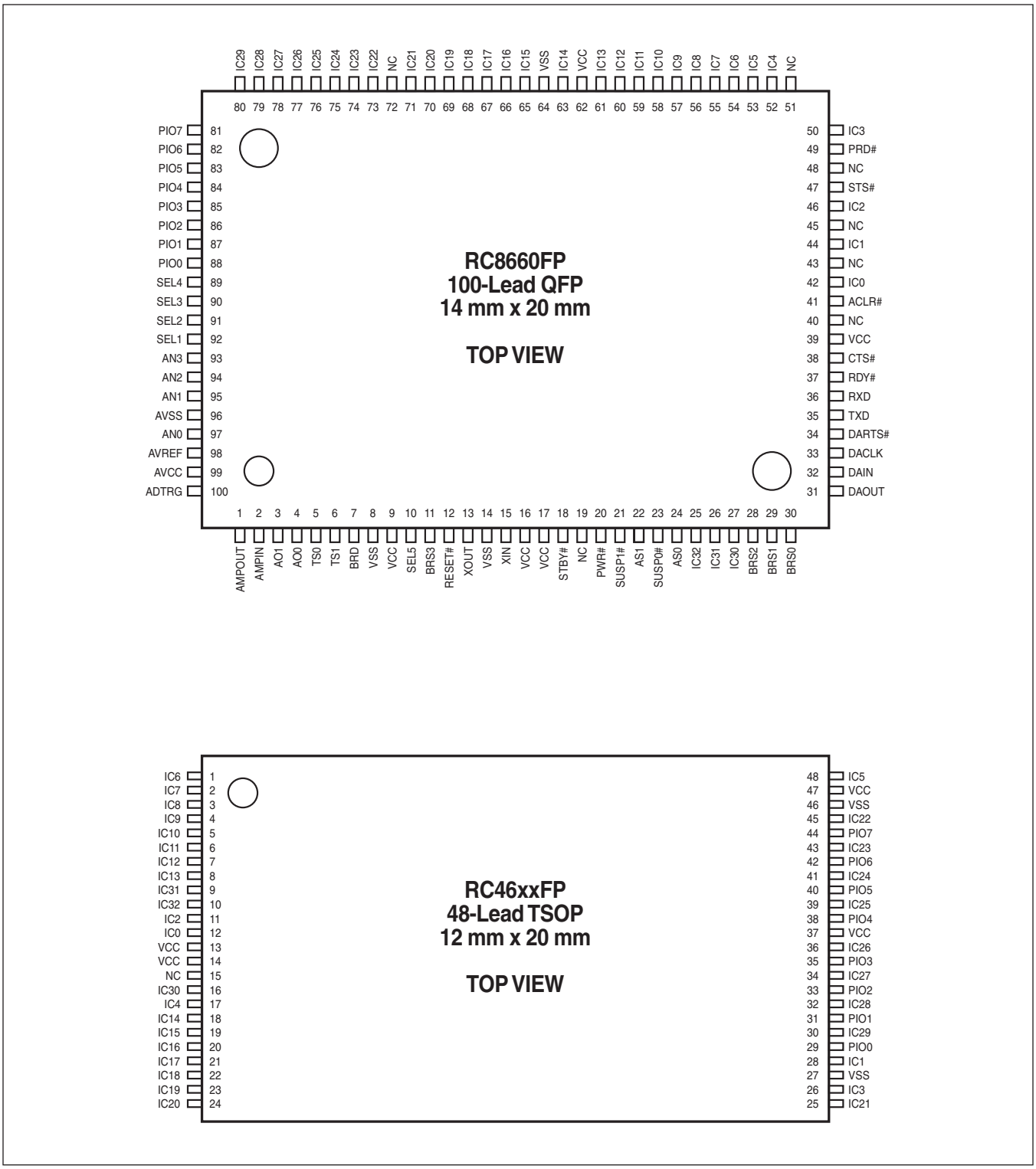


Figure 1.1. Pin Assignments

PIN DESCRIPTIONS

Pin Name	Type	Name and Function
IC_0–IC_{32}	INTPUT/ OUTPUT	**CHIPSET INTERCONNECTS:** Interconnections between the RC8660 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 from 0 V to AV_{REF}. AO_1 is reserved for future use.
TS_0 TS_1	OUTPUT	**TALK STATUS:** Indicates whether a voice channel is active. TS_n 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). During recording operations, $SUSP_0$# will suspend recording when Low. $SUSP_1$# 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. During recording, AS_0 changes state each time the A/D converter input is sampled. 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 multichannel 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 RC8660 over this bus.
STS#	OUTPUT	**STATUS:** Controls the transfer of status information from the RC8660 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 RC8660. 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 RC8660. Data on the PIO_0–PIO_7 pins is latched in the RC8660 on the rising edge of PWR#. Sufficient time must be given for the RC8660 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 RC8660 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 RC8660'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, or stored in recording memory. 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 1.1. Pin Descriptions

Pin Name	Type	Name and Function
AMPIN AMPOUT	OUTPUT INPUT	**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.
RXD	INPUT	**RECEIVE DATA:** Asynchronous serial data input used to send text, data and commands to the RC8660. 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 RC8660. This pin changes from a CMOS output to an N-channel open-drain output if any of the SEL pins are High, allowing multiple TXD pins to be wire-OR'd together in a multi-channel system.
CTS#	OUTPUT	**CLEAR TO SEND:** The CTS# pin is Low when the RC8660 is able to accept data. If the input buffer becomes full as a result of the last byte received, CTS# will go High and remain High until room becomes available.
BRD	INPUT	**BAUD RATE DETECT:** BRD is used by the RC8660 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_3 pins affect the operation of BRD. Connect this pin to a High level if not used.
BRS_0– BRS_3	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_3 to a High level will allow the RC8660 to automatically detect the baud rate with the BRD pin. Connect these pins to a High level if not used.
STBY#	INPUT	**STANDBY/INIT:** Dual function pin which either puts the RC8660 in Standby mode or initializes the RC8660's internal parameter memory. *STBY# must be High on the rising edge of RESET#.* **Driving STBY# Low for 250 ms or longer causes the RC8660 to enter Standby mode.** All peripheral and serial port handshake lines are driven to their false ("not ready") states, and the input buffer is cleared. During standby, the RC8660 draws the minimum possible current (100 nA typ @ 3.3 V), but it is not able to respond to any input pin except STBY# and RESET#. Returning STBY# High causes the RC8660 to enter Idle mode (600 µA typ); the handshake lines are re-asserted and the RC8660 will be able to accept input again. If the RC8660 entered standby due to a Sleep Timer event, driving STBY# Low for 250 *ns* or longer then High will return the RC8660 to Idle mode. **Driving STBY# Low for less than 250 ms initializes the RC8660's non-volatile parameter memory.** The greeting message and user dictionary are erased, and all voice parameters and register settings are restored to their factory default settings. The audio recording memory is not affected. The RC8660 then announces its version number via the AO_0 pin. Connect this pin to a High level if not used.
SEL_1– SEL_5	INPUT	**SELECT:** Programs the channel pair that the RC8660 is to respond to in a multi-channel system. Connect these pins to a Low level in single-channel systems.
RESET#	INPUT	**RESET:** A Low immediately terminates all activity and sets all pins to a predetermined state. 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 RC8660's D/A and A/D converters. Connect ACLR# to RESET#.
XIN XOUT	INPUT OUTPUT	**CLOCK INPUT/OUTPUT:** These pins connect to the internal clock generating circuit. All timing for the RC8660 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.
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 1.1. Pin Descriptions (Continued)

FUNCTIONAL DESCRIPTION

The RC8660 chipset includes a number of features that make it ideally suited for any design requiring voice output. The RC8660's major features are described below.

Text-to-Speech Synthesizer

The RC8660 provides text-to-speech conversion with its integrated DoubleTalk™ text-to-speech synthesizer. Any English text written to the RC8660 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 RC8660 that is automatically spoken immediately after the RC8660 is reset. Most any of the commands recognized by the RC8660 may be included as part of the greeting message, which can be used to set up custom default settings and/or play a pre-recorded 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.

Audio Recording and Playback

Up to 15 minutes of recorded messages and sound effects can be stored in the RC8660 for on-demand playback. Recordings are stored in on-chip nonvolatile memory, providing zero-power message storage. Additionally, the RC8660 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 RC8660 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. Data logging and audio recording to the RC8660's recording memory is also possible through the ADC.

Versatile I/O

All data is sent to the RC8660 through its built in serial and/or parallel ports. *For maximum flexibility, including infield product update capability, use of the serial port is recommended whenever possible.*

The RC8660'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 RC8660 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 RC8660 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 1.1). **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 RC8660 and RC46xx chips. IC_{30}, IC_{31}, and IC_{32} must have 47 kΩ–100 kΩ pullup resistors to V_{CC}.

Clock Generator

The RC8660 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 1.2 for recommended clock connections.

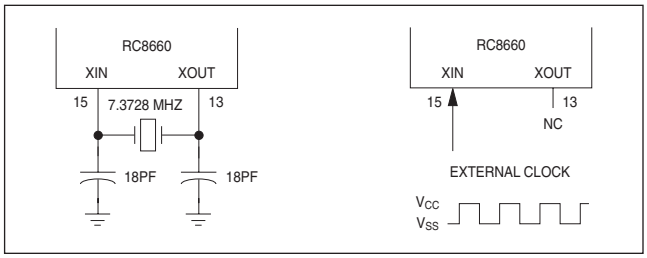


Figure 1.2. Clock Connections

INTERFACING THE RC8660

The RC8660 contains both asynchronous serial and 8 bit bus interfaces. All text, commands, tone generator data, real time audio data, etc., are transmitted to the RC8660 via one of these ports. For maximum flexibility, use of the serial port is recommended whenever possible. *Not all RC8660 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 serial port operates with 8 data bits (LSB first), 1 or more stop bits, no parity, and any standard baud rate between 300 and 115200 bps.

A typical RS-232C interface is shown in Figure 1.3. Note that the MAX232A transceiver is not required if the host system's serial port operates at logic levels compatible with the RC8660 (0/+5 V or 0/+3.3 V). The RC8660's serial port may be connected directly to the host system in this case.

The CTS# pin should be used to control the flow of serial data to the RC8660. 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 RC8660 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_3 pins, as shown in Table 1.2. **Auto-detect** enables the serial port to automatically detect the baud rate of the incoming data. The baud rate **command** (described in Section 2) allows the baud rate to be changed at any time, effectively overriding the first two methods.

The automatic baud rate detection mechanism is enabled when the BRS_0–BRS_3 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 RC8660; it is discarded.

BRS_3	BRS_2	BRS_1	BRS_0	Baud Rate
L	L	L	L	300
L	L	L	H	600
L	L	H	L	1200
L	L	H	H	2400
L	H	L	L	4800
L	H	L	H	9600
L	H	H	L	19200
L	H	H	H	Auto-detect
H	L	L	L	38400
H	L	L	H	57600
H	L	H	L	115200
All other settings				Auto-detect

Table 1.2. Baud Rate Options

If the measured bit period is determined to be a valid baud rate, the RC8660 acknowledges lock acquisition by transmitting the ASCII character "l" (6Ch) on the TXD pin. The baud rate will remain locked unless changed with the baud rate command, or the RC8660 is reset.

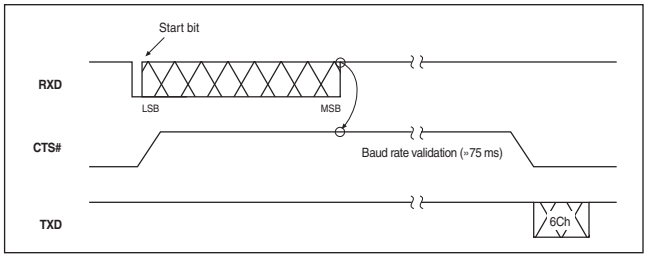


Figure 1.4. Baud Rate Detection Timing

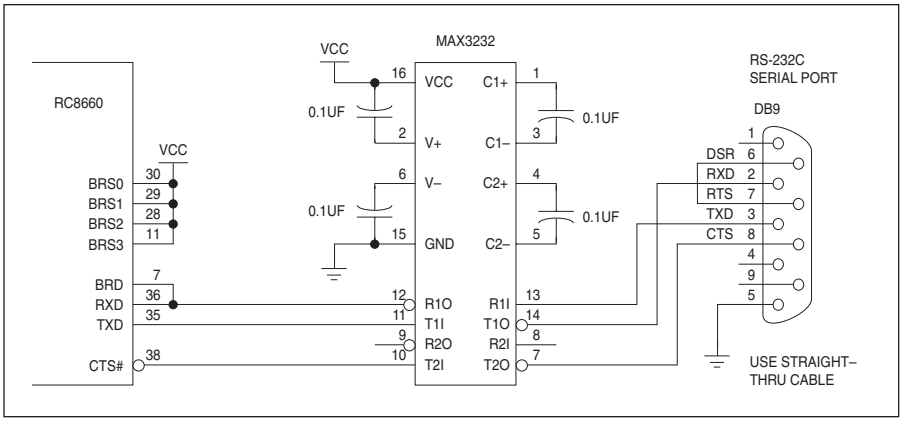


Figure 1.3. RS-232C Interface

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 RC8660 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 RC8660 is actually ready to accept data. See Figure 1.4.

Status messages

Real-time status information is provided via the TXD pin. Status are transmitted as one-byte messages, shown in Table 1.3. Each message correlates to a status flag in the Status Register, shown in Table 1.5. The specific character used, and whether it will be transmitted, are functions of the V86 and STM bits in the Protocol Options Register. (The Protocol Options Register is described in Section 2.) 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 1.3. Status Messages

Multi-channel system

Multiple RC8660s can be connected in a multi-channel configuration by wiring the RXD pins together and TXD pins together, as shown in Figure 1.5. All communication is performed over the resulting RXD/TXD bus. Individual RC8660s are addressed through a simple addressing scheme, shown in Table 1.4. *Single-channel systems should have all of the RC8660's SEL pins connected to a Low logic level. This permanently activates the RC8660 so that no activation code is required, and configures the TXD pin as a CMOS output.* Note that if any of the SEL pins are connected to a High logic level, the TXD output pin will be automatically configured to be an open-drain, N-channel output.

NOTE Table 1.4 refers to *channel pairs*, because each RC8660 can potentially support two channels (AO_0 and AO_1). This feature may be implemented in a future version of the RC8660.

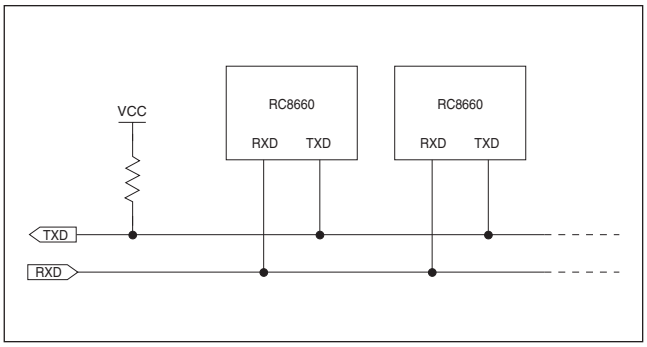


Figure 1.5. Multi-Channel System

SEL_5	SEL_4	SEL_3	SEL_2	SEL_1	Channel Pair	Activation Code
L	L	L	L	L	All	–
L	L	L	L	H	2-3	C2h
L	L	L	H	L	4-5	C4h
L	L	L	H	H	6-7	C6h
.	.	.	.	.	.	.
.	.	.	.	.	.	.
H	H	H	H	L	60-61	FCh
H	H	H	H	H	Disabled	–

Table 1.4. RC8660 Channel Addressing

Each RC8660's address is programmed through the strapping of its SEL pins; the logic levels present on these pins determine which activation code each RC8660 will respond to. When an RC8660 sees its activation code on the RXD bus, it becomes enabled and functions normally. All other RC8660s on the bus disable themselves at the same time, although the contents of their input buffers will not be affected. The 8 KB buffer within each RC8660 allows a channel to be opened, accept messages (text and commands), then be immediately closed—thereby allowing another channel to be opened—while the first RC8660 processes the information it just received.

To address a specific RC8660, issue the appropriate activation code. For example, to address channel 4, transmit C4h. All subsequent output will be accepted by the RC8660 configured for channel 4, and ignored by all the others.

Handshaking with each RC8660 in a multi-channel system is simplified through a special interrogation mechanism. Instead of monitoring each channel's CTS# pin, any channel can instead be queried by transmitting code C0h. In response, the active channel will return an eight bit status code on the TXD bus, as defined in Table 1.5. The Ready status bit should be checked at least every 8 data bytes in order to avoid data loss.

Bus/Printer Interface

The RC8660's bus interface allows the RC8660 to be connected to a microprocessor or microcontroller in the same manner as a static RAM or I/O device, as shown in Figure 1.7. The microprocessor controls all transactions with the RC8660 over the system data bus using the RD and WR# signals. RD controls the reading of the RC8660's Status Register; WR# controls the transfer of data into the RC8660. The Status Register bits and their definitions are shown in Table 1.5.

A registered bus transceiver is required for communication between the RC8660 and microprocessor; two 74HC374s placed back to back may be substituted for the 74HC652 shown in the figure. Prior to each write operation to the RC8660, the host processor should verify that the RC8660 is ready by testing the RDY status flag.

The RC8660 can also be interfaced to a PC's printer port as shown in Figure 1.7. 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 RC8660 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 overwriting 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 RC8660 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 the loop is 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 1.6 illustrates the recommended method of writing data to the RC8660's bus interface. This method should be used for writing all types of data, including text, commands, tone generator and real time audio data.

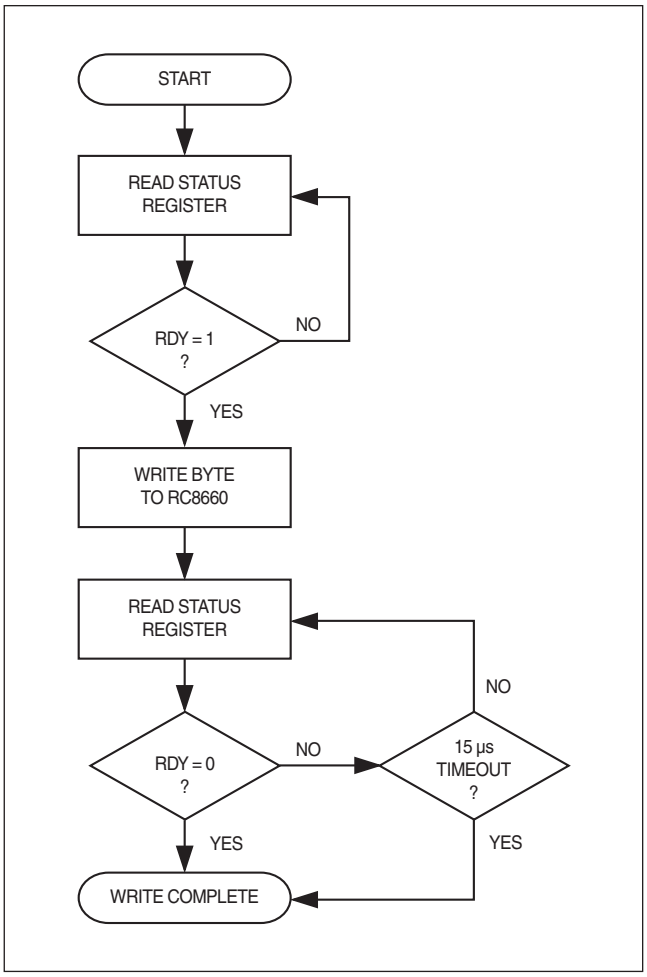


Figure 1.6. Recommended Method of Writing Data Via the Bus Interface

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	The TS bit has the same meaning as the TS_0 pin. "1" means that the RC8660 is producing output; "0" means 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 RC8660 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 = RC8660 is in Standby mode 0 = RC8660 not in Standby mode	This bit is "1" when the RC8660 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 1.5. Bus Interface Status Register Bit Definitions

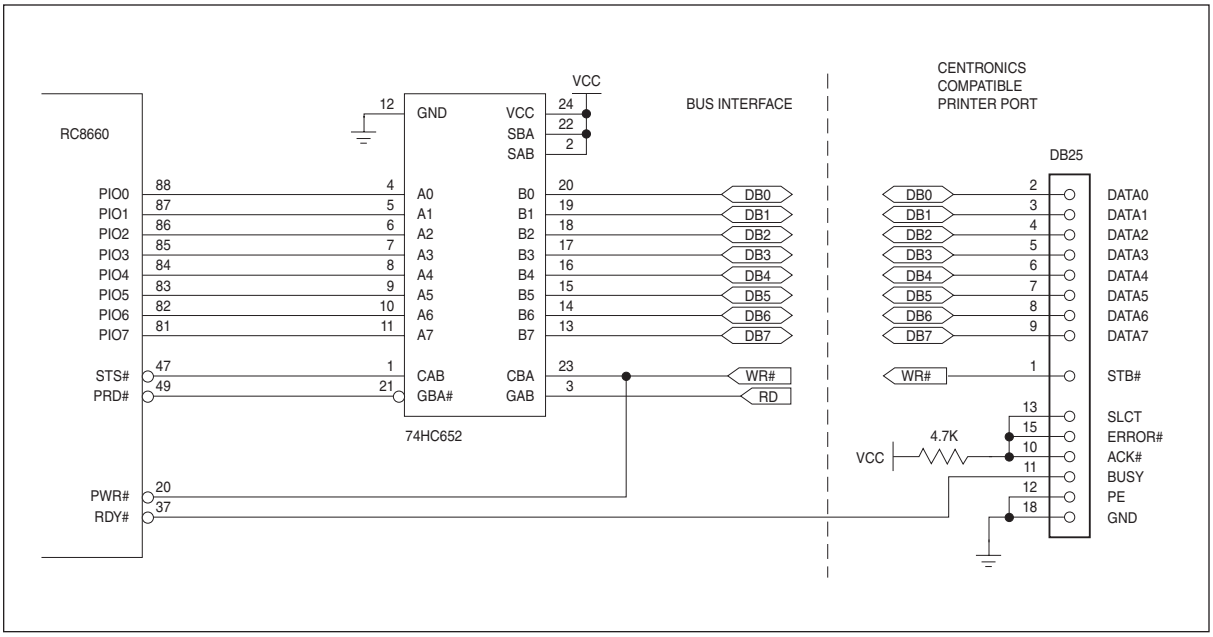


Figure 1.7. Bus/Printer Interface

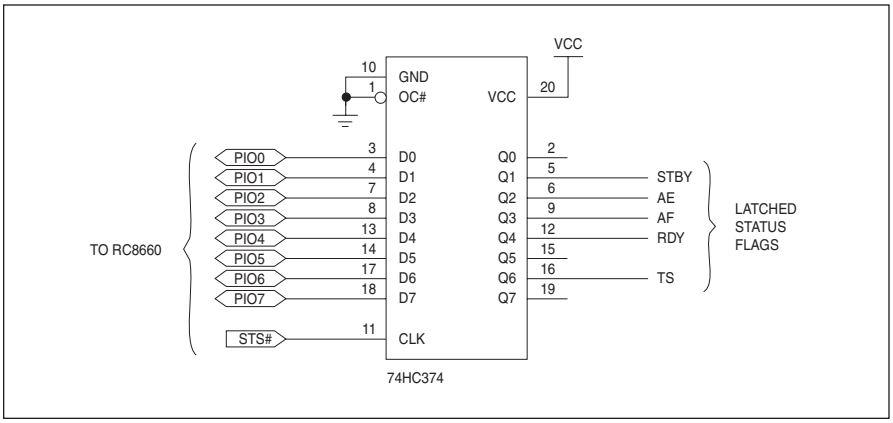


Figure 1.8. Method of Capturing Status Information for Driving External Circuitry

Analog Audio Output

The analog output pins AO_0 and AO_1 are high impedance (10 kΩ typical) outputs from the RC8660'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 1.9 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 RC8660's TS_0 pin to minimize current drain when the RC8660 is inactive.

Digital Audio Output

The digital audio pin DAOUT outputs the RC8660's audio signal as a digital audio stream consisting of 8 data bits per sample. The normalized sampling rate for all text to speech modes, sinusoidal generator, and DTMF generator is 84 kbs (10,500 bytes/sec). The prerecorded and real time audio playback mode rates are user programmable, so their normalized rates will vary. See the Pin Descriptions and Audio Control Register command description for further details.

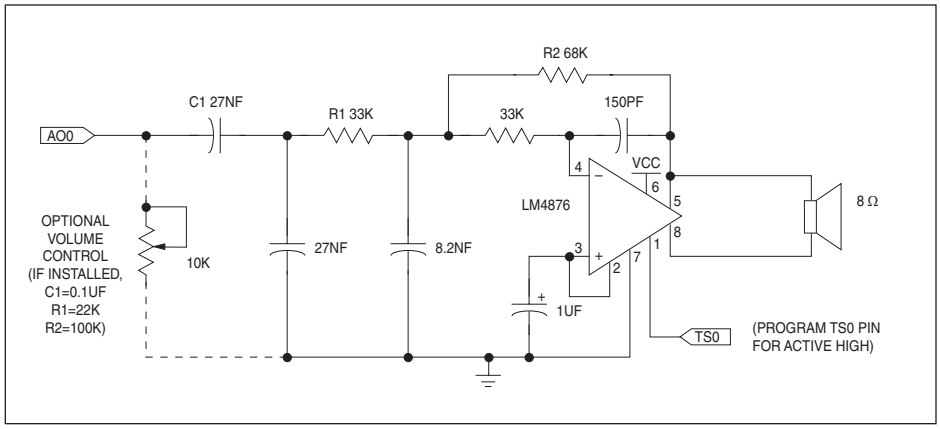


Figure 1.9. 3 kHz Low-Pass Filter/Power Amplifier

ELECTRICAL SPECIFICATIONS

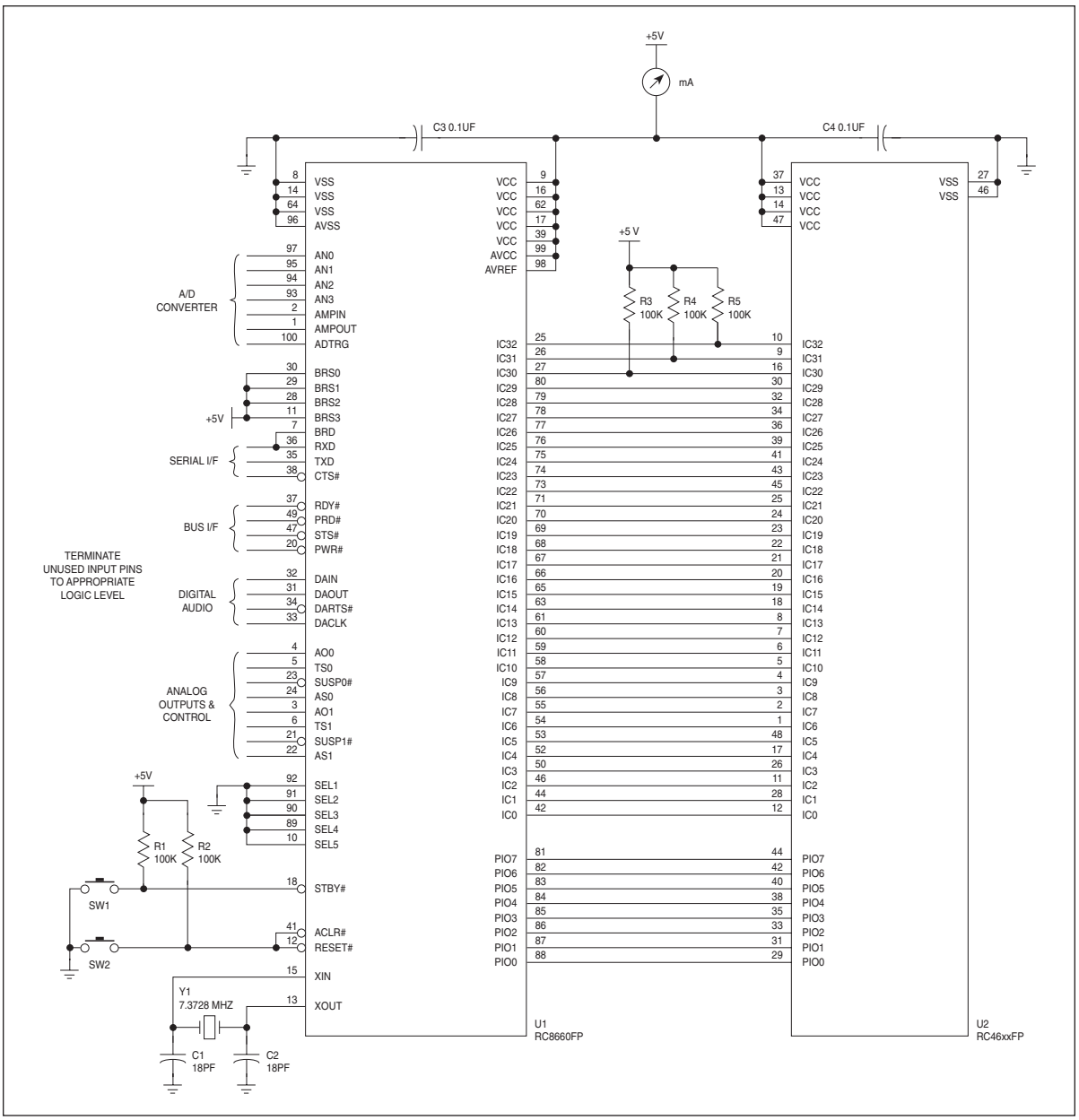


Figure 1.10. Test Circuit

ABSOLUTE MAXIMUM RATINGS*

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

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.*

DC CHARACTERISTICS

T_A = 0 °C to +70 °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 ± 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		6	18		13	33	mA	
	Idle		0.6	1.5		1.2	2.5	mA	
	Standby		0.1	15		0.2	20	µA	
	Program (Note 1)			35			65	mA	

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

AC CHARACTERISTICS

T_A = 0 °C to +70 °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 ± 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

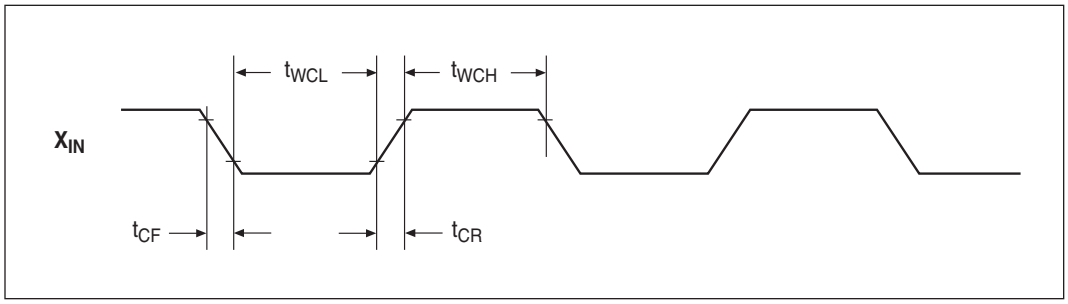


Figure 1.11. External Clock Waveform

Bus Interface Timing

Symbol	Parameter	3.3 ± 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.

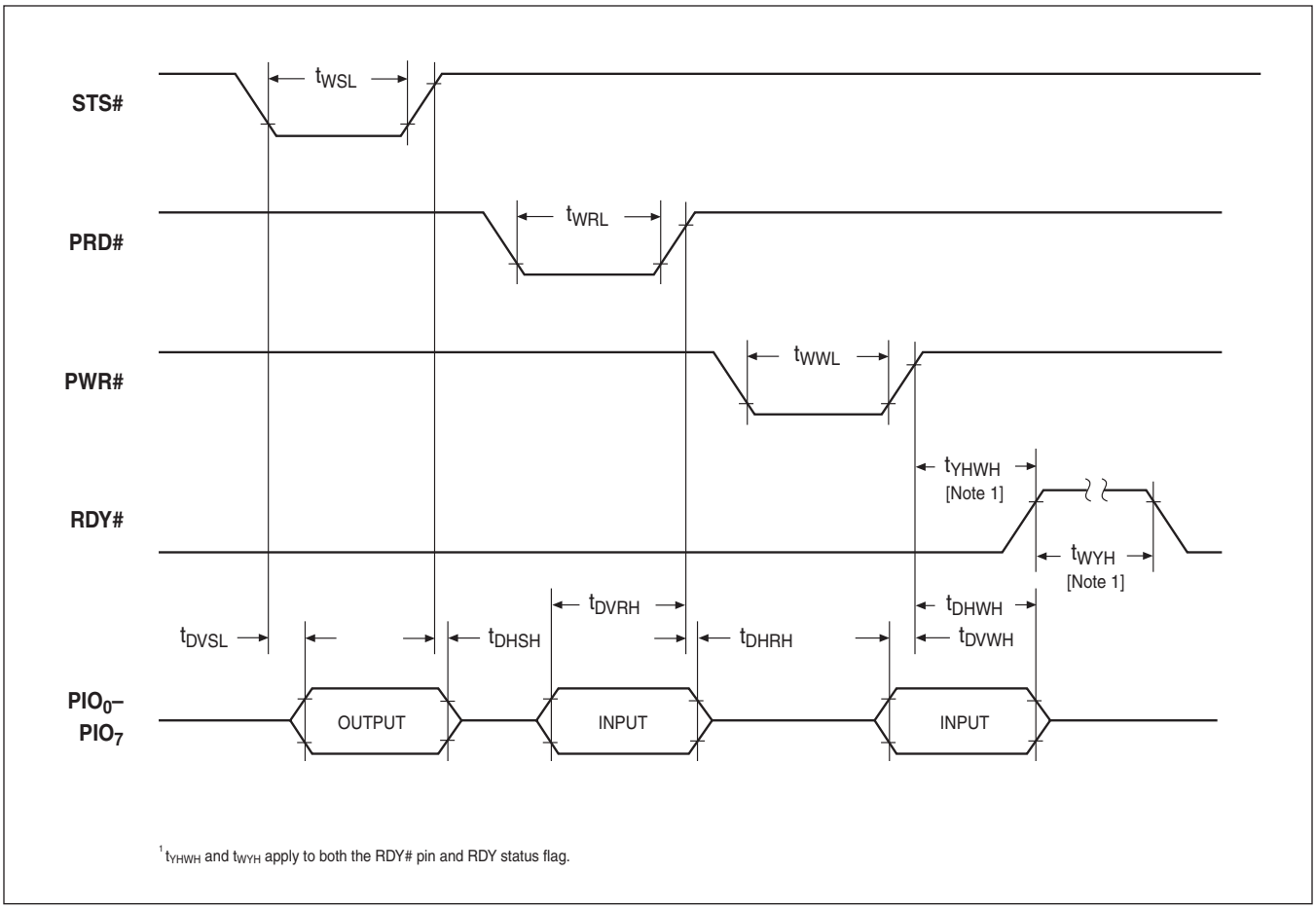


Figure 1.12. Bus Interface Waveforms

Analog Audio Timing

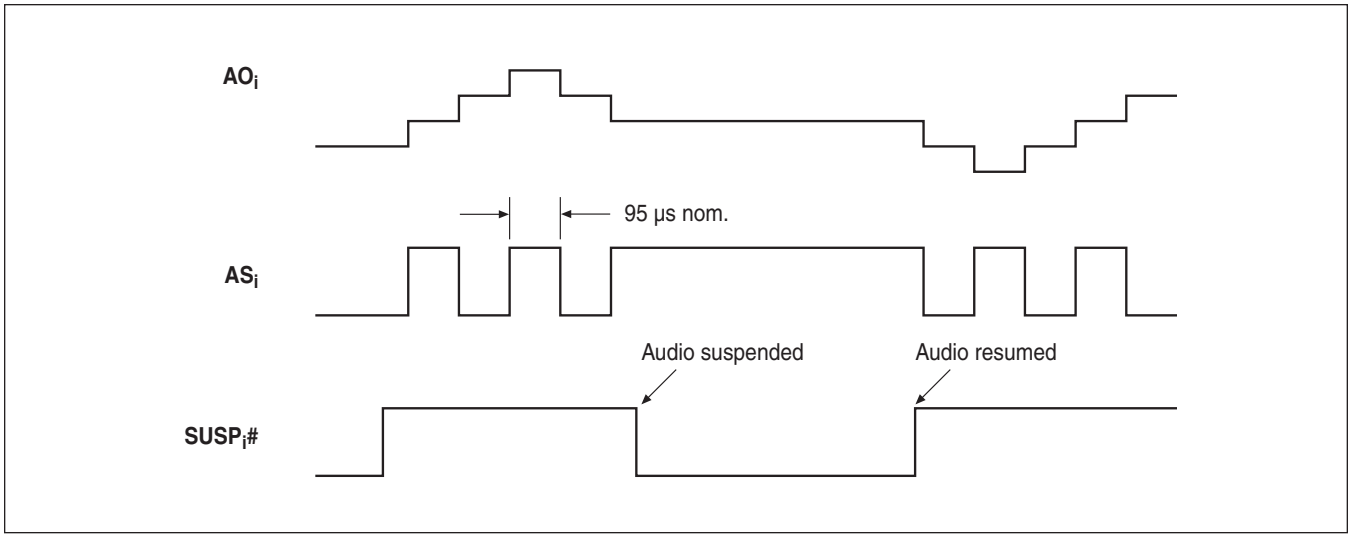


Figure 1.13. Analog Audio Waveforms

Digital Audio Timing

Symbol	Parameter	Min	Max	Unit	Notes
t_{CYC}	DACLK cycle time	200		ns	
t_{WCL}	DACLK pulse width Low	100		ns	
t_{WCH}	DACLK pulse width High	100		ns	
t_{DVCL}	DACLK Low to data valid		80	ns	
t_{DHCL}	Data hold from DACLK going Low	0		ns	
f_S	TTS and DTMF generator internal sampling rate	10.5	10.5	kHz	Nominal

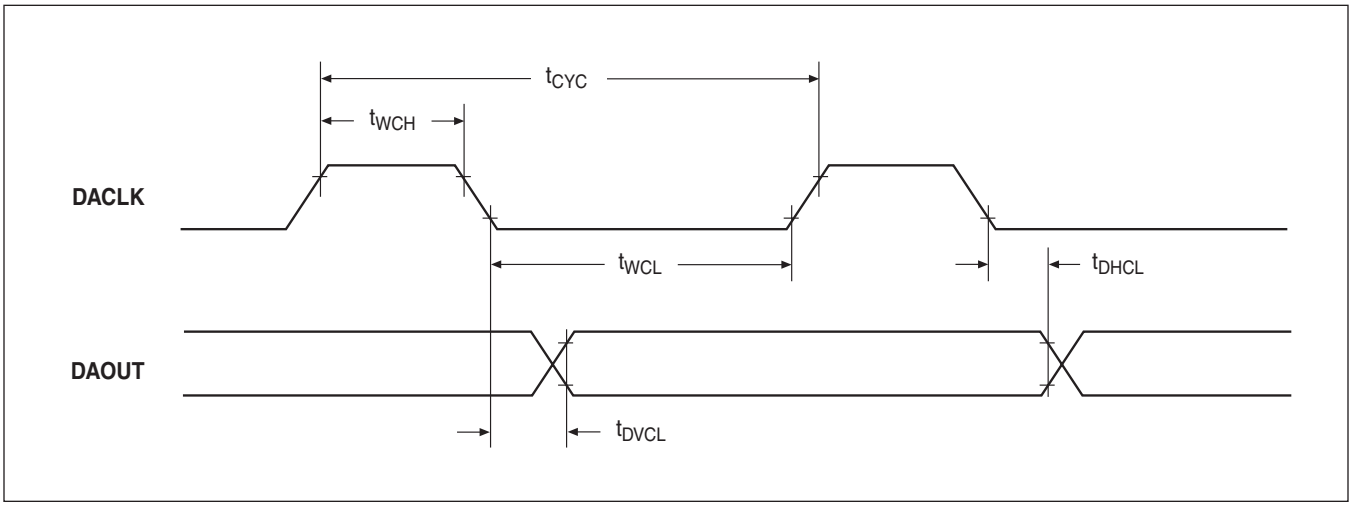


Figure 1.14. Digital Audio Waveforms

Standby Timing

Symbol	Parameter	3.3 ± 0.3 V		5 V ± 0.5 V		Unit
		Min	Max	Min	Max	
t_{WSBL}	STBY# pulse width Low					
	To enter Standby mode	250		250		ms
	To reinitialize parameter memory	8	250	8	250	ms
	To exit Standby mode (Sleep Timer invoked; Note 1)	380		250		ns

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

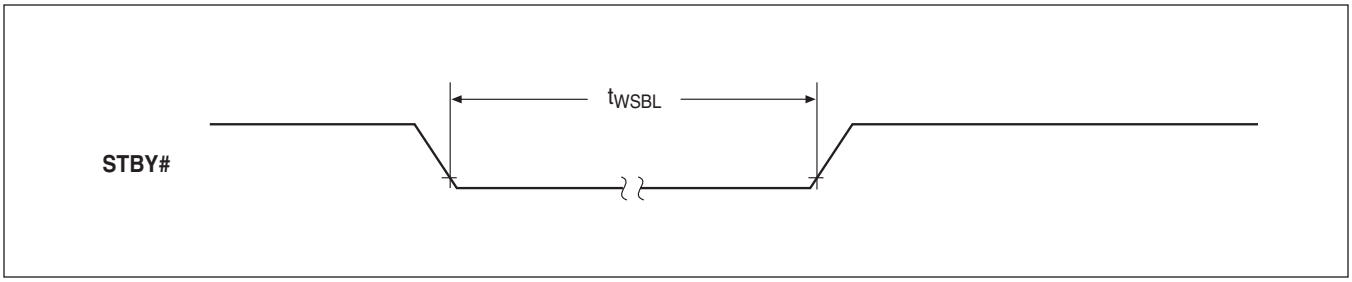


Figure 1.15. Standby Waveform

Reset Timing

Symbol	Parameter	Min	Max	Unit	Notes
t_{WRS}	RESET# pulse width Low				
	After power on / V_{CC} stable	1		ms	Hold RESET# Low during power-up
	During operation	3		µs	
t_{DRR}	RESET# recovery delay		2	ms	

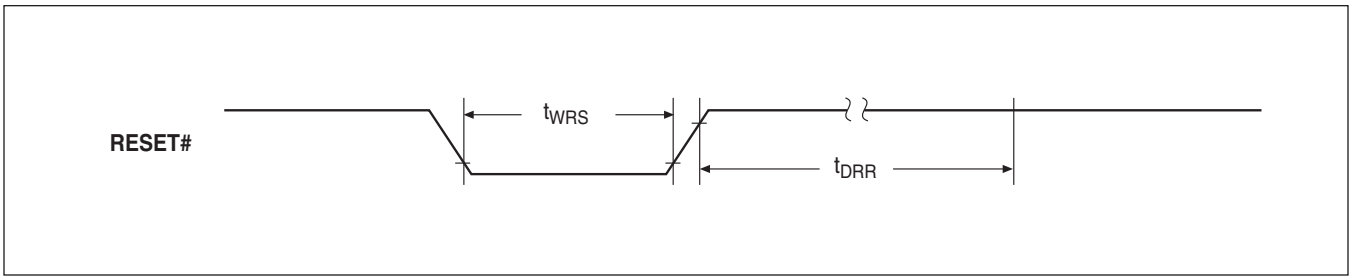
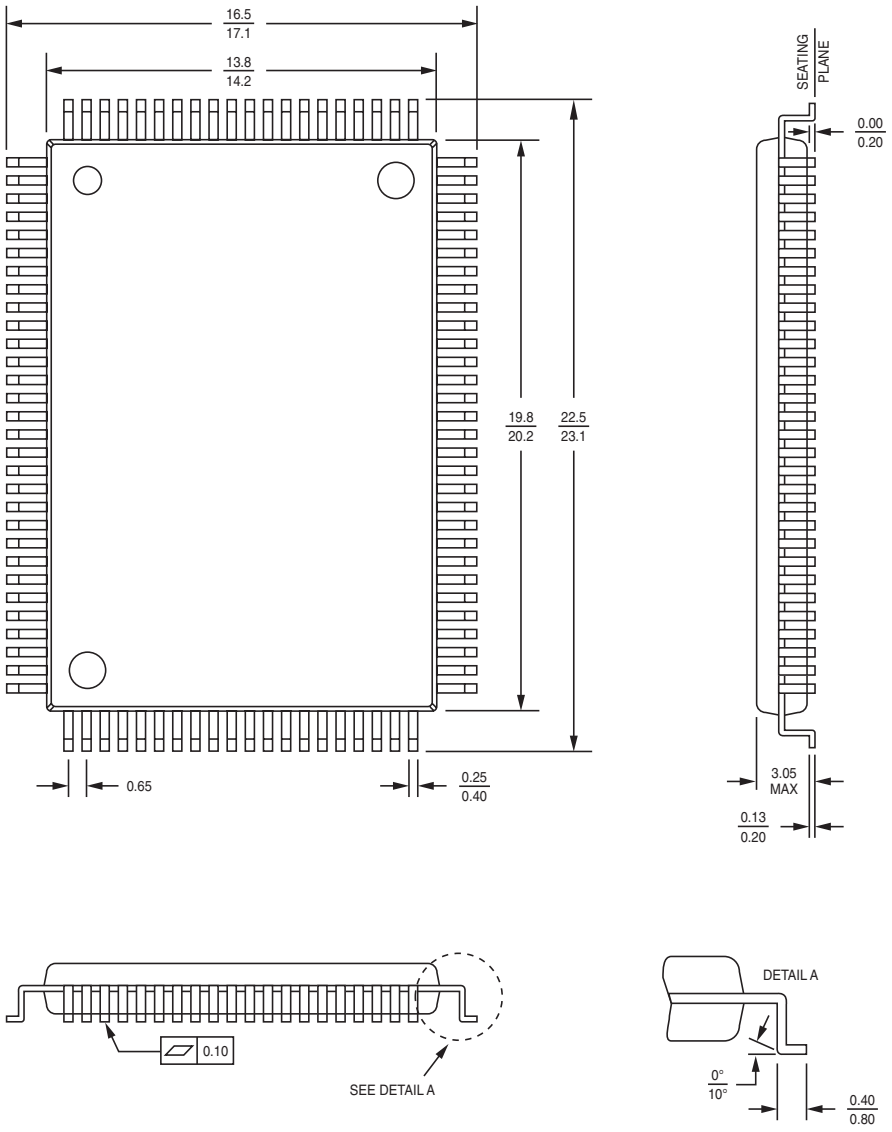


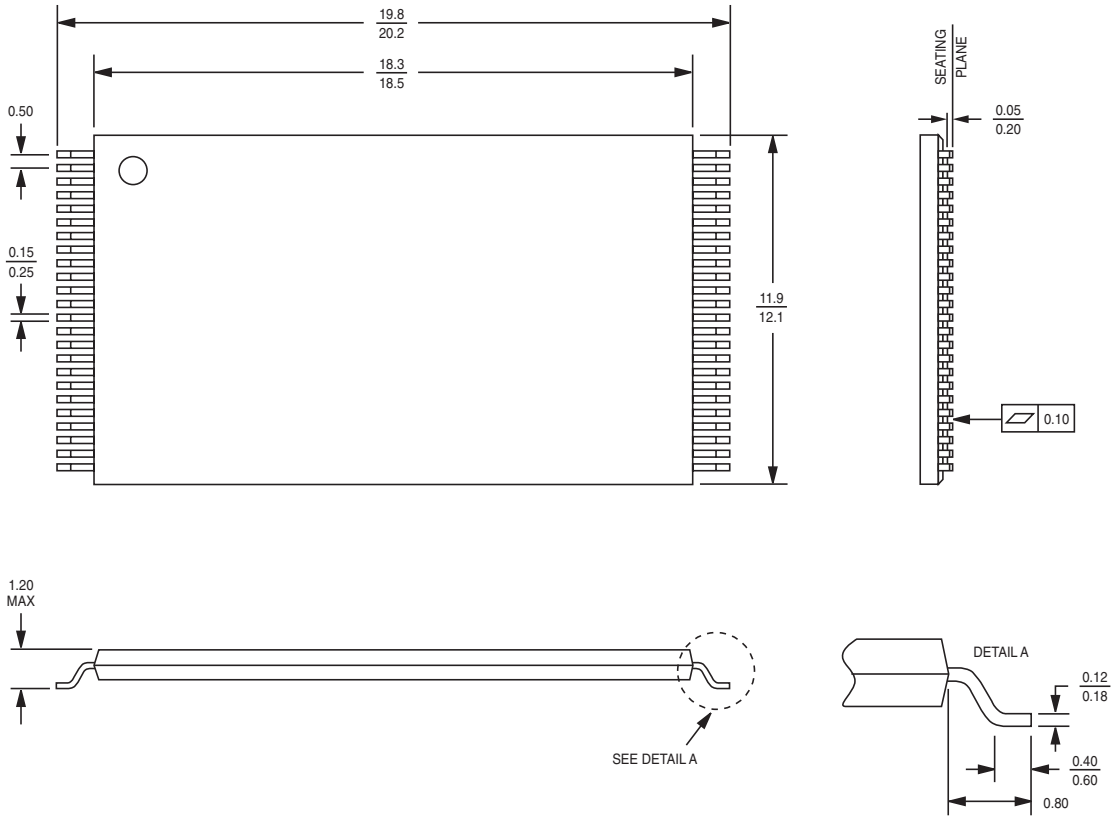
Figure 1.16. Reset Waveform

PACKAGE INFORMATION

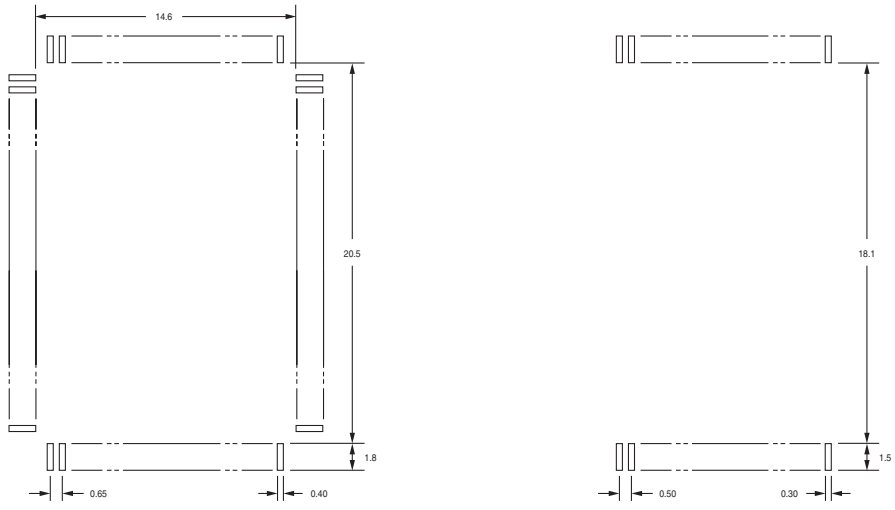
100 Pin Plastic 14 x 20 mm QFP (measured in millimeters)



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

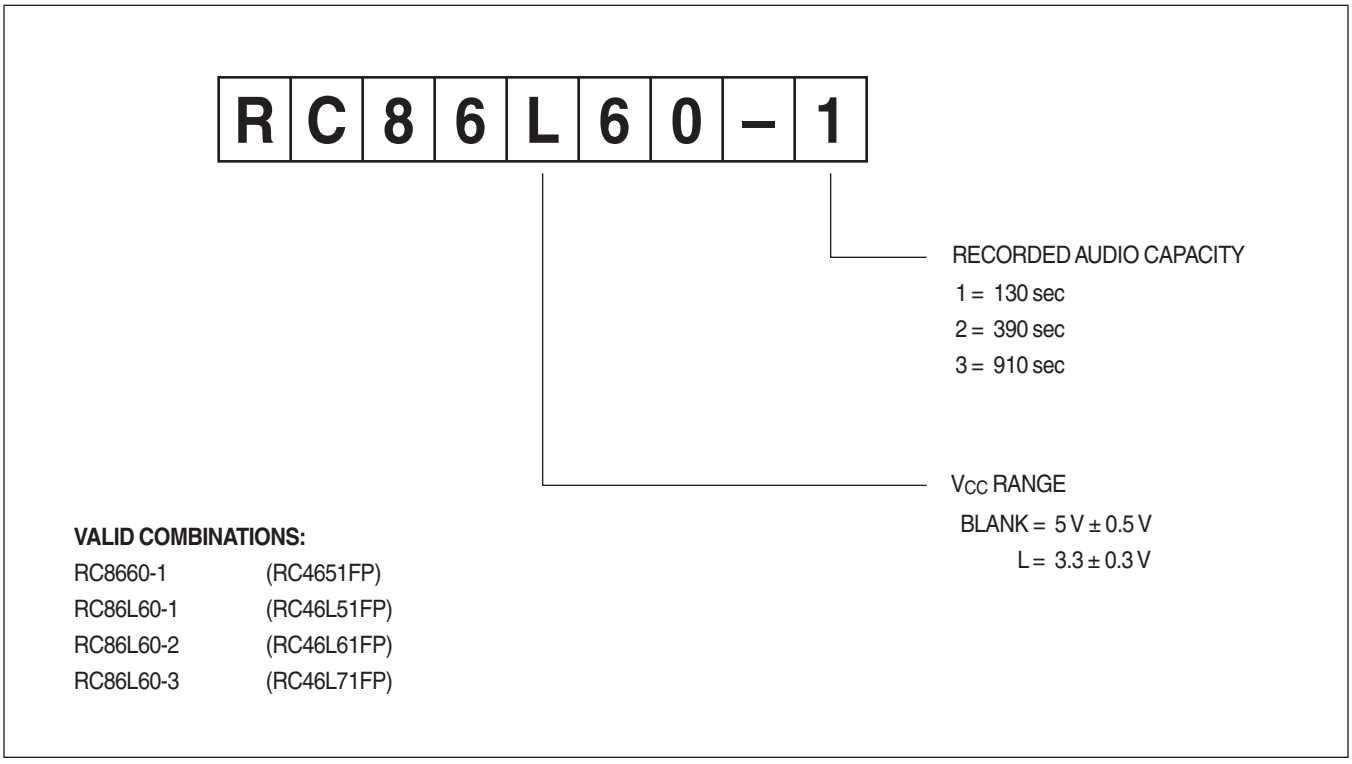


Recommended PCB Layouts (measured in millimeters)



ORDERING INFORMATION

The RC8660 is available in several recording capacities and voltage ranges. 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. All configurations include the RC8660FP chip; the companion chip is shown in parentheses. For example, the RC86L60-1, a 3.3 V part with 130 seconds of recording memory, is composed of the RC8660FP and RC46L51FP.



VALID COMBINATIONS:

RC8660-1	(RC4651FP)
RC86L60-1	(RC46L51FP)
RC86L60-2	(RC46L61FP)
RC86L60-3	(RC46L71FP)

SECTION 2: PRINCIPLES OF OPERATION

OPERATING MODES

The RC8660 has five primary operating modes and two inactive modes designed to achieve maximum functionality and flexibility while consuming the least possible amount of power. The operating mode can be changed at any time by issuing the appropriate command.

NOTE The RC8660 will not begin speaking nor execute commands issued it until it receives a CR (ASCII 13) or NUL (ASCII 00) 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, use the Timeout Delay command.

The RC8660 does not make any distinction between uppercase and lowercase characters. All data sent to the RC8660 is buffered in an internal 8 KB input buffer, allowing additional text and commands to be queued even while the RC8660 is producing output.

Text-To-Speech mode. By default, all text sent to the RC8660 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.

Recorder mode. Any of the RC8660's four ADC inputs can be used to make audio recordings, such as voice memos. Analog voltages, such as from a temperature transducer or battery, can be sampled and recorded using the ADC's one-shot mode. All recordings can be retrieved via the serial port or played back on demand.

Recorded Audio Playback mode. This mode allows messages and sound effects that have been recorded or downloaded into the RC8660 to be played back on demand. PCM and ADPCM data types are supported.

Real Time Audio Playback mode. Data sent to the RC8660 is written directly to the RC8660'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.

Tone Generator modes. These modes activate the RC8660'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 RC8660 automatically enters a reduced-power state whenever it is inactive. Data can still be read and written to the RC8660 while in this mode. Current draw is typically 600 µA @ 3.3 V.

Standby mode. This mode powers down the RC8660, where current draw is typically only 100 nA. Standby mode can be invoked from either the STBY# pin or with the Sleep command. Data cannot be read from nor written to the RC8660 in this mode.

COMMAND SYNTAX

The RC8660 commands provide a simple yet flexible means of controlling the RC8660 under software control. They can be used to vary voice attributes, such as the 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. Commands affect only the data that follows them in the data stream.

All RC8660 commands are composed of the command character (CTRL+A by default), a parameter *n* comprised of an ASCII number string, and a single string literal that uniquely identifies the command. Some commands simply enable or disable a feature of the RC8660 and do not require a parameter. The general command format is:

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

If two or more commands are to be used together, each must be prefaced with the command character. This is the only way the RC8660 knows to treat the following characters as a command, rather than text that is to be spoken. For example, the following commands program pitch level 40 and volume level 7:

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

The Command Character

The default RC8660 command character is CTRL+A (ASCII code 01). The command character itself can be spoken by the RC8660 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 RC8660, 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 (ASCII 01-26) by sending the current command character, followed by the new character. To change the command character to CTRL+D, for example, issue the command `CTRL+A CTRL+D`. To change it back, issue the command `CTRL+D CTRL+A`. It's generally a good idea 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+^` (ASCII 30) to the RC8660.

Command Parameters

Command parameters are composed of ASCII number strings. The RC8660 supports two types of parameters: absolute and relative. **Absolute** parameters explicitly specify a 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 allow you to specify a desired *change* in a parameter's 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 RC8660 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 RC8660, 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 RC8660; the RC8660 does the rest. The RC8660 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 RC8660 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 RC8660'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. Exception dictionaries are covered in detail in Section 3.
3. Use the RC8660's Phoneme mode.

The first technique is the easiest way to fine tune word pronunciations—by tricking the RC8660 into the desired pronunciation. Among the more commonly mispronounced words are compound words (*baseball*), proper names (*Sean*), and foreign loan words (*chauffeur*). Compound words can usually be corrected by separating the two words with a space, so that "baseball" becomes "base ball." 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 RC8660 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 RC8660 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 RC8660's phonemes to be accessed directly. Table 2.1 lists the phonemes that can be produced by the RC8660.

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 tokens

The RC8660 supports seven phoneme attribute tokens that can be used in addition to the standard commands. These tokens do not require the command character or any parameters, but they can only be used in Phoneme mode and exception dictionaries. In addition, changes made to parameters with attribute tokens are only temporary. Table 2.2 lists these tokens and their equivalent "standard" commands.

Applications of Phoneme mode

Phoneme mode is useful for creating customized speech, 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.

Phoneme Symbol	Example Word	Phoneme Symbol	Example Word
A	das (Spanish)	M	me
AA	cot	N	new
AE	cat	NG	rung
AH	cut	NY	niño (Spanish)
AW	cow	O	no (Spanish)
AX	bottom	OW	boat
AY	bite	OY	boy
B	bib	P	pop
CH	church	PX	spot
D	did	R	ring
DH	either	RR	tres (Spanish)
DX	city	S	sell
E	ser (Spanish)	SH	shell
EH	bet	T	tin
EI	mesa (Spanish)	TH	thin
ER	bird	TX	stick
EW	acteur (French)	U	uno (Spanish)
EY	bake	UH	book
F	fee	UW	boot
G	gag	V	valve
H	he	W	we
I	libro (Spanish)	WH	when
IH	bit	Y	mayo (Spanish)
IX	rabbit	YY	you
IY	beet	Z	zoo
J	age	ZH	vision
K	cute	space	variable pause *
KX	ski	,	medium pause
L	long	.	long pause

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

Table 2.1. DoubleTalk Phoneme Symbols

Symbol	Function	Equiv Cmd
nn	Set pitch to 'nn' (0-99)	*n*P
/	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 ≅ 2n

Table 2.2. Phoneme Attribute Tokens

Speed (nS)

The synthesizer's speech rate can be adjusted with this command, from 0S (slowest) through 13S (fastest). The default rate is 5S.

Voice (n0)

The text-to-speech synthesizer has 11 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 10O, shown in Table 2.3. 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
8	Goliath
9	Alvin
10	Gretchen

Table 2.3. 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 RC8660 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 RC8660 to speak in a monotonic voice. Intonation should be disabled whenever manual intonation is applied using the Pitch command or phoneme attribute tokens. 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 99F. By varying the frequency, voice quality can be fine-tuned or voice type changed. The default frequency is 50F.

Pitch (nP)

This command varies the synthesizer's 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), which work much like the bass and treble controls on a stereo. 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 RC8660 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 2.4.

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 the NM bit 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." 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, leading zero suppression enabled).

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

Punctuation Filter Register Bits
PFR.7–4 = RESERVED (R) Write "0" to ensure future compatibility.
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 2.4. Punctuation Filter Register Definitions

A/D CONVERTER

ADC Control Register (n$)

This register controls the operation of the integrated analog-to-digital converter. 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

Figure 2.1 is a functional block diagram of the ADC input stage; Figure 2.2 illustrates the ADC in operation. Table 2.5 lists the definitions of each bit of the ADC Control Register. All ADC results are transferred via the TXD pin except in sound recorder mode—when the results are written to the RC8660's recording memory. The default register setting is 128$.

Operation of the ADC is not mutually exclusive of other RC8660 functions. The ADC can operate concurrently with text-to-speech, tone generation, audio playback, etc. The effective sampling rate in continuous mode is one-tenth the serial port baud rate (e.g., 115200 baud = 11.52 ksps).

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: 00 = AN_0 01 = AN_1 10 = AN_2 11 = AN_3 When ADR.3 = 1: 00 = undefined 01 = AN_0–AN_1 sweep 10 = undefined 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 2.5. ADC Control Register Definitions

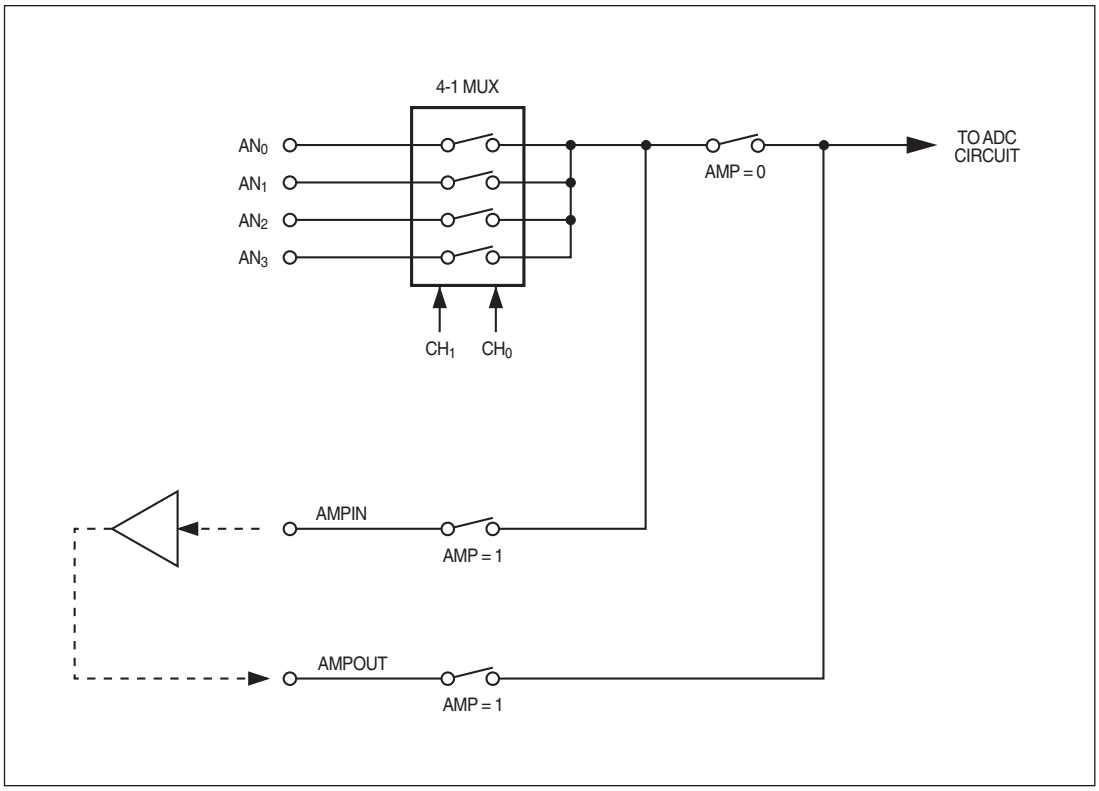


Figure 2.1. ADC Input Block Diagram

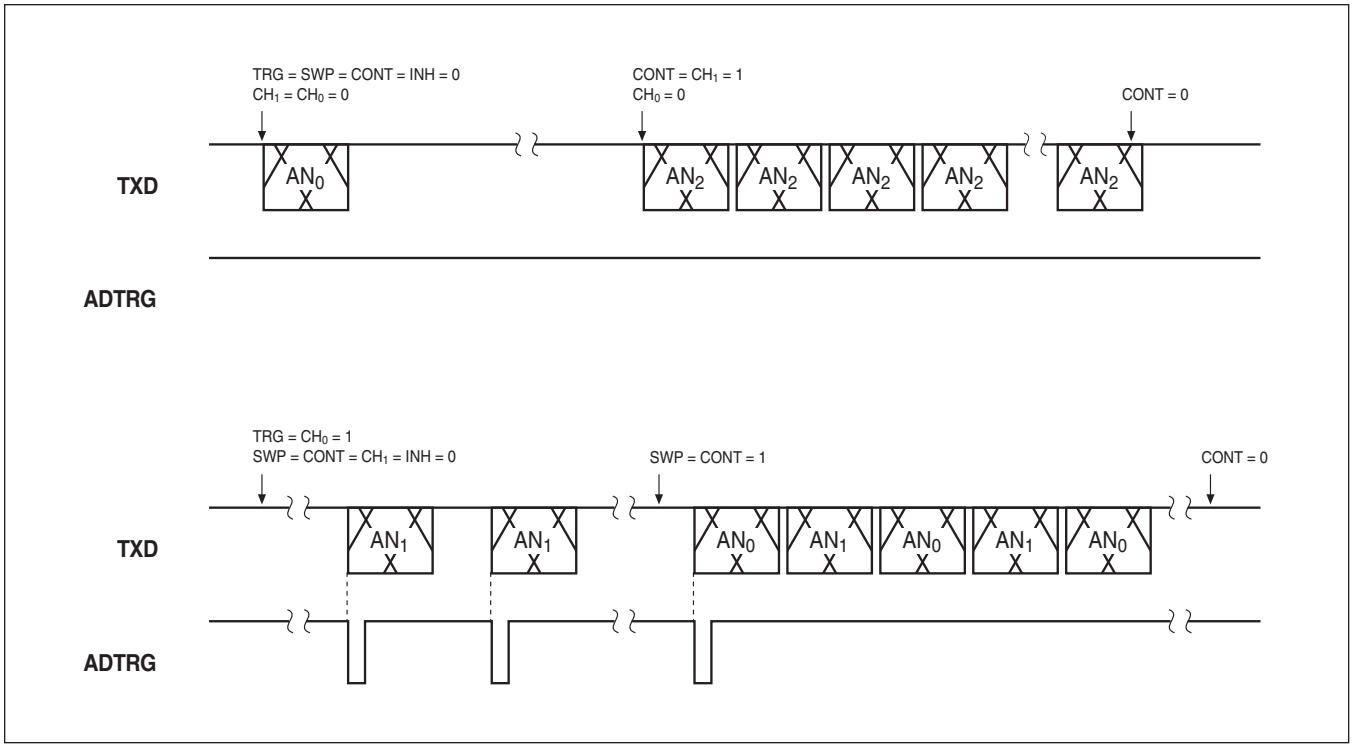


Figure 2.2. ADC Transfer Timing

RECORDING & PLAYBACK

Libraries of sounds and recorded messages can be stored in the RC8660's integrated recording memory for on-demand playback. New libraries can be downloaded at any time, even in the field. With the addition of a microphone and preamplifier, recordings can also be made directly to the recording memory through the RC8660's A/D converter inputs. The RC8660 includes built in support for downloading, uploading, and erasing sound files through the serial interface. The number of sound files is limited only by the amount of available on-chip recording memory.

Additionally, the RC8660 can play back eight bit audio in real time, such as speech and/or sound effects stored in an external memory or file system.

Recording Memory File System

All file functions—play, download, upload and erase—require a means of specifying the file to be operated on. The RC8660's built-in file system allows sound files, whether downloaded or recorded through the ADC, to be easily accessed.

Each sound file in memory is automatically assigned a unique record number, or index, beginning with zero. The first file is record 0, the second is record 1, and so on. Referencing a sound file by its record number is one way to select the file to be operated on. However, if files are added and/or deleted from memory frequently, their record numbers can become difficult to keep track of.

All RC8660 sound files contain a unique 16 bit "tag" that can also be used to reference the file. The tag is assigned to the sound file when it is created with the *RCStudio* software, or, in the case of recording through the ADC, specified in the recording command. Tags can range in value from 1 to 65534. A value of 0 is defined as a null tag, and will be ignored. The setting of the TAG bit in the Protocol Options Register determines whether the tag value or record number will be used when addressing a file in the recording memory.

The sound file functions described here each return one or more result codes via the TXD pin. These codes are summarized in Table 2.6.

Play Sound File (n&)

This command plays the sound file specified by the parameter *n*. The setting of the TAG bit in the Protocol Options Register determines whether *n* represents the record number or tag value. For example,

```
CTRL+A "52&"
```

plays record 52 (actually the 53rd file) if TAG = 0, or the record whose tag value equals 52 if TAG = 1.

The playback volume can be adjusted with the Volume (*n*V) 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&"
```

Message	Code	Meaning
No_Error	0	No error
Invld_Cmd_Err	1	Invalid command error
Com_Err	2	Communications error
TmOut_Err	3	Communications timeout error
Erase_In_Prog	4	Memory erase in progress
Erase_Err	5	Memory erase error
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
Incompat_Err	12	Incompatible sound library in memory
Dup_Tag_Err	13	Duplicate tag in memory
Not_Found_Err	14	Record not found in memory
Rec_In_Prog	15	Recording in progress

Table 2.6. Recording Memory Return Codes

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 (TAG = 0 in this example).

Erase Sound File (234L)

The command 234L*mmmmm* deletes sound file *mmmmm* from the recording memory. The setting of the TAG bit in the Protocol Options Register determines whether *m* represents a record number or tag. *Note that all five* m *digits must be included in the command (use leading zeroes if necessary).*

During erasure, the CTS# pin will go High and one or more Erase_In_Prog messages will be transmitted on the TXD pin. Completion is indicated by CTS# going Low and the transmission of Erase_Ok on the TXD pin. Note that this process can take some time, as the recording memory is also automatically defragmented during erasure.

Download Sound File (236W)

This command initiates the download of a sound file to the RC8660. Files are always appended to the end of the sound library in memory. *RCStudio* may be used to create the RC8660 compatible sound files from standard Windows wave files. After issuing this command, simply transfer the sound file.

After the first dozen or so bytes of the sound file have been transferred, the RC8660 verifies that there is enough space in its recording memory and that there is not already a file in memory with the same tag (other than tag 0). No_Error will be transmitted if there are no errors, and the RC8660 will load the rest of the file. At the completion of the download, No_Error will be transmitted again (assuming no download errors).

Upload Sound File (245L)

This command is used to upload a sound file or recording from the RC8660's recording memory. The command 245L*mmmmm* retrieves sound file *mmmmm* from memory; immediately upon receipt of the command, the RC8660 will begin transmitting the file via the TXD pin. The setting of the TAG bit in the Protocol Options Register determines whether *m* represents a record number or tag. To stop the transfer of the file, issue the Stop (CTRL+X) command. *Note that all five* m *digits must be included in the command (use leading zeroes if necessary).*

The first six bytes transmitted contain information about the sound file, including its length, sampling rate/encoding type, and tag value:

$L_{24}\ R_8\ T_{16}$

where L_{24} is the length of the data (24 bits, least-significant byte first), R_8 is the sampling rate/encoding type, and T_{16} is the 16-bit tag value. The lower-seven bits of R_8 convey the sampling rate f_s:

$f_s = 617000/(155-R_7)$ Hz

while the high bit specifies the encoding type: 0 = PCM, 1 = ADPCM. For example,

```
45h 23h 01h CEh 07h 00h
```

indicates a sound file of 12345h bytes length, 8000 Hz sampling rate, ADPCM encoding, and a tag value of 7. The 12345h data bytes immediately follow the six-byte header.

Download Sound Library (223W)

This command is generally only found in sound libraries created with *RCStudio*. It initiates the download of the sound library (essentially a collection of sound files) to the RC8660.

Initialize Recording Memory (214W)

This command formats the RC8660's recording memory so that it can be used for storing sound files. This must be done before the *initial* use of the recording memory, unless a sound library has been previously downloaded, which also initializes the recording memory. The command can also be used as an "erase all files" function, if desired.

During the initialization process, the CTS# pin will go High and one or more Erase_In_Prog messages (Table 2.6) will be transmitted from the TXD pin. Completion is indicated by CTS# going Low and the transmission of Erase_Ok from the TXD pin. Note that this process can take some time, depending on the size of the recording memory.

Making a Recording

Recording to the RC8660 requires setting up the A/D converter for the desired recording mode and issuing the Record command.

To begin recording

1) Program the ADC for the desired input channel (CH), SWP = TRG = 0, and INH = 1. To record continuously, such as a voice memo, program CONT = 1. To record only 16 samples of the input channel (data logging mode), program CONT = 0.

2) Issue the Record command *n*L*mmmmm*. *N* programs the desired sampling rate f_s, where $n = 155-617/f_s$ and $4 \le f_s \le 11$ kHz ($0 \le n \le 99$). If ADPCM compression is desired (reduces memory storage requirements by a factor of 2), add 128 to the value of *n*. *M* defines the tag value that is to be associated with the recording. If you do not want to associate a tag with the recording, set *m* = 00000. *Note that all five* m *digits must be included in the command (use leading zeroes if necessary).* For example, the command `CTRL+A "227L01234"` begins recording at 11 kHz rate with ADPCM compression, assigning a tag value of 1234 to the recording.

   After the command has been transmitted to the RC8660, the RC8660 will verify that there is space in the recording memory and that there is not already a file with the same tag in memory (other than tag 0). Assuming there are no errors, the RC8660 will begin recording, transmitting Rec_In_Prog every time 1024 bytes has been written to the recording memory.

To stop or pause recording

1) To stop recording, issue the Stop (CTRL+X) command. The RC8660 will transmit No_Error to acknowledge it has stopped.

2) To pause recording, pull the $SUSP_0$# pin Low.

Real Time Audio Playback (n#/n%)

This mode allows audio samples to be written directly to the RC8660's digital-to-analog converter via the serial or parallel port. All data sent to the RC8660 is routed directly to the RC8660's internal audio buffer; the RC8660 then outputs samples from the buffer to the DAC at the rate programmed by *n*. Because the audio data is buffered within the RC8660, the output sampling rate is independent of the data rate into the RC8660, as long as the input rate is equal to or greater than the programmed sampling rate.

The RC8660 supports PCM and ADPCM audio data formats. RC Systems' *RCStudio* software can convert standard Windows wave files to PCM and ADPCM formats for use with the RC8660. ADPCM compression yields data files that are half the size of PCM files, thereby reducing the required data bandwidth and storage requirements.

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, therefore f_s can range from 4 to 11 kHz.

The following procedure should be used for sending PCM or ADPCM audio data to the RC8660 in real time:

NOTE If the sound file was created with *RCStudio*, simply transfer the file to the RC8660; the following steps should not be used. These sound files include the appropriate playback command in the header and 80h at the end of the file. However, the serial port's baud rate must be programmed for 115,200 baud in order for the data stream to be able to keep up with the RC8660.

1) Program the desired volume level with the Volume (*n*V) 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 command *n#* if PCM data is being sent, or *n%* for ADPCM data. The RC8660 expects the audio data to immediately follow the command; therefore, be sure not to terminate the command with a CR or NUL. The TS pin and TS flag will be asserted at this time.

3) If the RC8660's serial port is being used for transferring the audio data and the port is not already running at 115,200 baud, change the *host* system's baud rate to 115,200 baud at this time.

4) Begin transferring the audio data to the RC8660. The same methods employed for sending any other type of data to the RC8660 should be used. Note that the DAC will not begin taking samples from the audio buffer until at least 100 bytes have been sent or the value 80h is sent, whichever occurs first.

5) After the last byte of audio data has been sent to the RC8660, send the value 80h (–128). This signals the RC8660 to terminate Real Time Audio Playback mode and return to the text-to-speech mode of operation. Note that up to 2048 bytes of data may still be in the audio buffer, so the RC8660 may continue producing sound for as long as 0.5 second (at 4 kHz sampling rate) after the last byte of data has been sent. The TS pin/TS flag will not be cleared until all of the audio data has been output to the DAC, at which time the RC8660 will again be able to accept data from the host.

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

TONE GENERATORS

The RC8660 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.

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 RC8660 is directed to the musical tone generator.

NOTE The RC8660 expects the tone generator data to immediately follow the J command; therefore, be sure not to 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 2.3). 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.

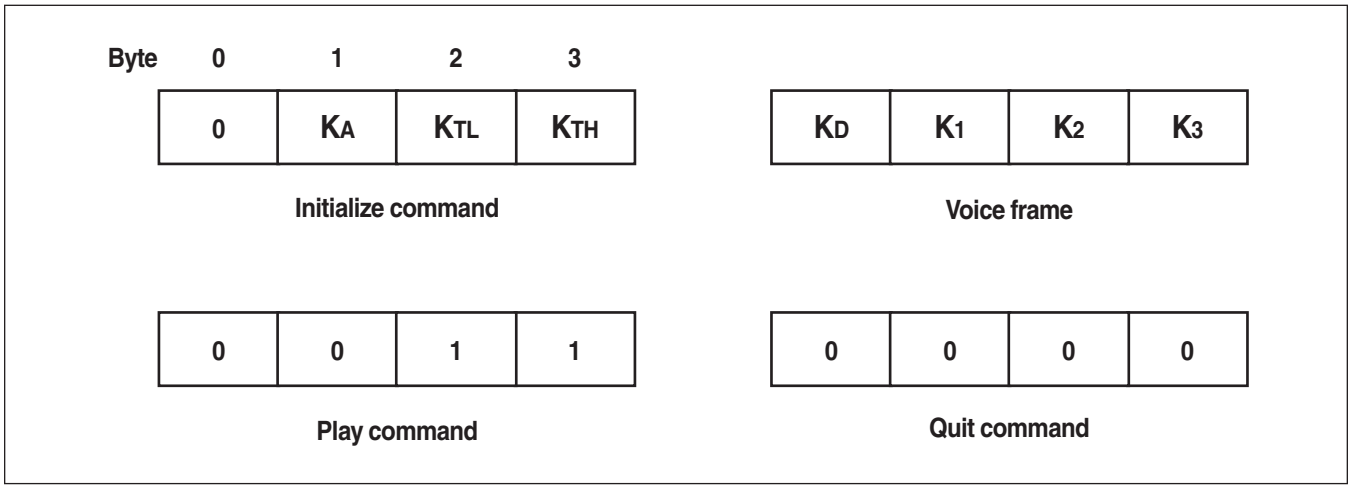


Figure 2.3. Musical Tone Generator Command Formats

Voice Frame

Voice frames contain the duration and frequency (pitch) information for each voice. All Voice frames are stored in an 8 KB buffer within the RC8660, but are not played until the Play command is issued. If the number of Voice frames exceeds 8 KB in length, the RC8660 will automatically begin playing the data.

Voice frame format

Voice frames are composed of three frequency time constants (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.

The task of finding K_i for a particular musical note is greatly simplified by using Table 2.7. 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.

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 2.7. Musical Note Pitch/K_i Values

For example, the Voice frame

```
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, which in this case is 24.

As another example,

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

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

Choosing note durations and tempo

Table 2.8 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 2.8. 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 RC8660 while the tone generator is operating. The TS 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 RC8660 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 RC8660 will not accept any more data until the entire buffer has been played.

Example Tune

The Basic program shown in Figure 2.4 reads tone generator data from a list of DATA statements and LPRINTs each value to the RC8660. The program assumes that the RC8660 is connected to a PC's printer port, although output could be redirected to a COM port with the DOS MODE command.

The astute reader may have noticed some "non-standard" note durations in the DATA statements, such as the first two Voice frames in line 240. 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 LPRINT               ' ensure serial port baud rate is locked
110 LPRINT CHR$(1);"J";  ' activate tone generator
120 READ B0,B1,B2,B3     ' read a frame (4 bytes)
130 LPRINT CHR$(B0); CHR$(B1); CHR$(B2); CHR$(B3);
140 IF B0 + B1 + B2 + B3 > 0 THEN 120  ' loop until Quit
150 END
160 '
170 '
180 ' Data Tables:
190 '
200 ' Init (volume = 255, tempo = 86)
210 DATA  0,255,86,0
220 '
230 ' Voice data
240 DATA  46,48,64,192, 2,0,64,192, 48,48,0,0, 48,40,0,0, 48,36,0,0
250 DATA  94,24,34,0, 2,24,0,0, 24,0,36,0, 24,0,40,0, 48,0,48,0
260 DATA  48,40,0,192, 46,36,0,0, 2,0,0,0, 48,36,0,0, 48,24,34,0
270 DATA  46,24,34,0, 2,0,34,0, 46,24,34,0, 2,24,0,0, 24,0,36,0
280 DATA  24,0,40,0, 48,0,48,0
290 '
300 ' Play, Quit
310 DATA  0,0,1,1, 0,0,0,0
```

Figure 2.4. Example Musical Tone Generator Program

SINUSOIDAL TONE GENERATOR

The sinusoidal tone generator generates two sinusoidal waveforms simultaneously. Applications for this generator range from generating simple, precise tones to telephone call-progress tones (such as a dial tone or busy signal). The frequency range is 0 to 4400 Hz with a resolution of 10 Hz.

The sinusoidal tone generator is activated with the command *nJaaaabbbb*. *N* specifies the tone duration in 10 ms increments, between 1 and 59999. *A* and *b* specify the frequencies of generator 1 and generator 2, respectively. *Note that all eight digits for* a *and* b *must be included in the command*. For example, the command

```
CTRL+A "100J03500440"
```

generates a 350/440 Hz tone pair (a dial tone) for 1 second.

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 2.9.

The "pause" tone can be used to generate longer inter-digit delays in phone number strings, or to create precise silent periods in the RC8660's output. The generator's output level can be adjusted with the Volume command (*n*V). 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 2.9. DTMF Dialer Button Map

RC8660 CONTROL

Volume (nV)

This is a global command that controls the RC8660's output volume level, from 0V through 9V. 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 RC8660 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 RC8660 without a CR or NUL, it will remain untranslated in the input buffer indefinitely.

The RC8660 contains a programmable timer that is able to force the RC8660 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 RC8660 is not active (not talking), and the input buffer contains no CR or NUL characters. Any characters sent to the RC8660 before timeout will automatically restart the timer.

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

Table 2.10. Timeout Delays

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.

Sleep Timer (nQ)

The sleep timer is used to force the RC8660 into Standby mode after a programmed time interval. For example, the RC8660 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 RC8660 is active, and begins running when the RC8660 enters Idle mode. In this way, the RC8660 will not shut itself down during normal use, as long as the programmed timer interval is longer than the maximum time the RC8660 is inactive.

The command parameter *n* determines when Standby mode will be entered. You can place the RC8660 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 2.11).

Note that the delay interval is simply *n* x 10 minutes for $0 < n < 16$. Adding 16 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 RC8660 to enter Standby mode as soon as all output has ceased.

If the sleep timer is allowed to expire, the RC8660 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 RC8660 has entered Standby mode.

Once the RC8660 has entered Standby mode, it can be re-awakened only by a hardware reset or by driving the STBY# pin low for 250 ns or longer, then High again. All of the RC8660 handshake signals (BUSY, CTS#, and RDY#) are forced to their "not ready" states while the RC8660 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 2.11. Sleep Timer

Download Exception Dictionary (247W)

This command initializes the RC8660's exception dictionary and stores subsequent output from the host in the RC8660's nonvolatile dictionary memory. The maximum dictionary size is 16 KB. The creation of exception dictionaries is covered in detail in Section 3.

Exception dictionaries must be compiled into the format required by the RC8660 before they can be used. The *RCStudio* software, 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.

Enable Exception Dictionary (U)

The exception dictionary is enabled with this command. If the RC8660 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.).

Write Greeting Message (255W)

Anytime the RC8660 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.

To create a new greeting message, perform the following steps:

1) Write the command `CTRL+A "255W"`.
2) Write the exact text/command sequence you want to store, up to 234 characters. For example, the string

   ```
   CTRL+A "3S" CTRL+A "2O" "ready"
   ```

   programs the RC8660 to use voice speed 3, Big Bob's voice, and say "ready" whenever it is reset.
3) Write a NUL (ASCII 00) to terminate the command and store the greeting in the RC8660's greeting memory.

The *RCStudio* software, available from RC Systems, can create and download greeting messages for you. The 255W command and terminating NUL are embedded within the greeting message file, simplifying the download process

Protocol Options Register (nG)

This command controls various internal RC8660 operating parameters. See Table 2.12 for the definition of each register bit. The default register setting is 144G.

Bit POR.7 (V86) programs the RC8660 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 1.3) are also affected by the setting of this bit.

V86	SAT	DDUR	R50	TAG	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 = RC8650 COMPATIBILITY (R50) 1 = Compatibility disabled 0 = Compatibility enabled	Emulates RC Systems' RC8650 chipset when set to "0." The RC8660's command set is adjusted to that of the RC8650, and RC8660-specific functions such as audio recording are disabled. Default: "1."
POR.3 = INDEX/TAG (TAG) 1 = Reference by tag 0 = Reference by index	Determines whether sound library files are referenced by their record number/index ("0") or by their tag value ("1"). Default: "0."
POR.2–1 = 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 2.12. Protocol Options Register Bit Definitions

Audio Control Register (nN)

The Audio Control Register determines whether the RC8660's audio stream will be output as an analog signal on the AO pins or as serial digital data on the DAOUT pin. See Table 2.13 for the definition of each register bit. The default register setting is 0N.

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

Figure 2.5 illustrates the synchronous data transfer mode. Note how either DARTS# or DACLK can be used to regulate the flow of data from the RC8660.

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 2.13. Audio Control Register Definitions

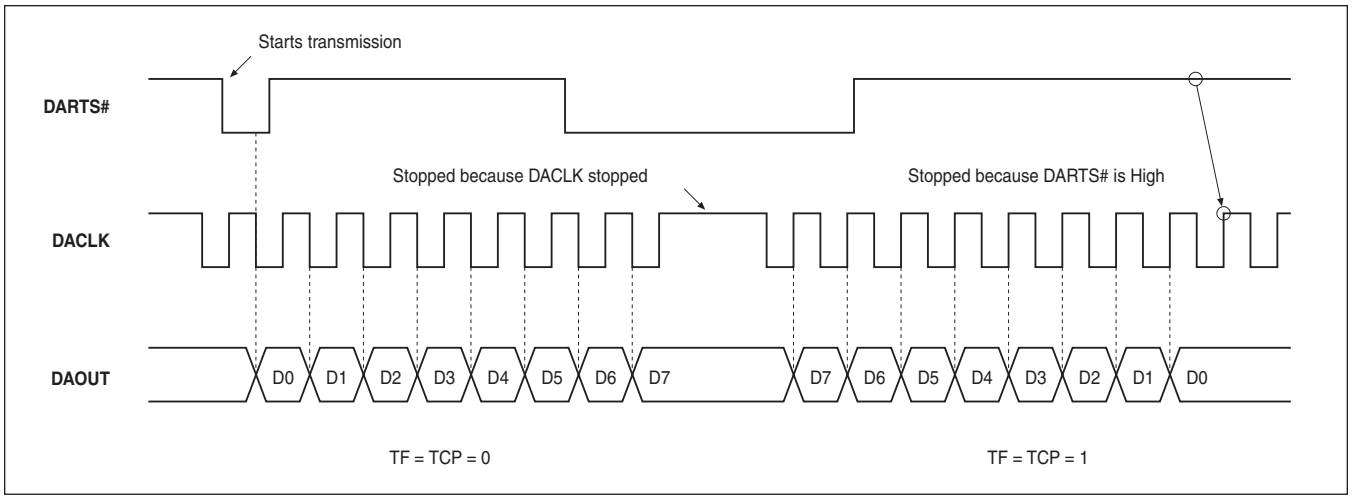


Figure 2.5. Synchronous Digital Audio Transfer Timing

Baud Rate (nH)

The serial port's baud rate can be programmed to the rates listed in Table 2.14. 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_3–BRS_0 pins

Table 2.14. Programmable Baud Rates

TS Pin Control (nK)

The TS pins provide talk status information for each audio channel, which can be used to activate a transmitter, take a telephone off hook, enable an audio power amplifier, etc., at the desired time. Each pin's state and polarity can be configured as shown in Table 2.15. The programming of the TS pins do not affect the Status Register TS flag in any way. The default setting is 1K.

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

Table 2.15. TS Pin Control

If a TS pin is programmed High or Low, it will remain so 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 pin is asserted as soon as output begins; it will return to its false state when all output has ceased. Note that because RC8660 commands work synchronously, the TS pin will not change state until all text and commands, up to the TS Pin Control command, have been spoken and/or executed.

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

The Stop command causes the RC8660 to stop whatever it is doing and flushes the input buffer of all text and commands. The Skip command skips to the next sentence in the buffer. Neither command affects any of the RC8660's settings.

NOTE The format of these commands is unique in that the command character (CTRL+A) is not used with them. The CTRL+X (ASCII 24) and CTRL+Y (ASCII 25) characters are *written directly* to the RC8660, which enables the RC8660 to react immediately, even if its input buffer is full. To be most effective, the states of the RC8660 handshaking signals should be ignored.

Zap Commands (Z)

This command prevents the RC8660 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+^` (ASCII 30) to the RC8660 or perform a hardware reset.

Reinitialize (@)

This command reinitializes the RC8660 by clearing the input buffer and restoring the voice parameters and control registers to their factory default settings. The exception dictionary, recording memory, greeting message, baud rate, nor TS pin control setting are affected.

MISCELLANEOUS FUNCTIONS

Index Marker (nI)

Index markers are nonspeaking "bookmarks" that can be used to keep track of where the RC8660 is reading within a passage of text. The parameter *n* is any number between 0 and 255; thus, up to 256 unique markers may be active at any given time.

When the RC8660 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 255, 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 RC8660 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, 41h 02h would be returned if the version number was 2.41.

Interrogate (12?)

This command retrieves the current operating settings of the RC8660. Table 2.16 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, with the exception of the last two parameters which are 16 bits each (low byte first).

Parameter	Cmd	Range
Mode	C/D/T	0=Char; 1=Phon; 2=Text
PFR register	*n*B	0-15
Formant freq	*n*F	0-99
Pitch	*n*P	0-99
Speed	*n*S	0-13
Volume	*n*V	0-9
Tone	*n*X	0-2
Expression	*n*E	0-9
Dict loaded	247W	0=not loaded; 1=loaded
Dict status	C/T/U	0=disabled; 1=enabled
Input buffer capacity	–	x256 bytes
Articulation	*n*A	0-9
Reverb	*n*R	0-9
TS pin control	*n*K	0-3
POR register	*n*G	0-255
ACR register	*n*N	0-255
Rec memory capacity	–	x16K bytes
Sleep delay	*n*Q	0-31
Timeout delay	*n*Y	0-15
Char mode delay	*n*C	0-31
Text mode delay	*n*T	0-15
Voice	*n*O	0-10
ADR register	*n*$	0-255
Free recording memory (16 bits)	–	x256 bytes
# of files in recording memory (16 bits)	–	0-999

Table 2.16. Parameters Returned by Interrogate Command

COMMAND SUMMARY

Command	Function	*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	0-99	50
*n*G	Protocol Options Register	0-255	144
*n*H	Baud rate	0-11	BRS pins
*n*I	Index marker	0-255	–
J	Musical tone generators	–	–
*n*J	Sinusoidal tone generators	1-59999	–
*n*K	TS pin control	0-3	1
*n*L	Record *	0-227	–
234L	Erase sound file *	234	–
245L	Upload sound file *	245	–
M	Monotone	–	–
*n*N	Audio Control Register *	0-255	0
*n*O	Voice	0-10	0
*n*P	Pitch	0-99	50
*n*Q	Sleep timer	0-31	0
*n*R	Reverb	0-9	0
*n*S	Speed	0-13	5
T/*n*T	Text mode/delay	0-15	0
U	Enable exception dictionary	–	–
*n*V	Volume	0-9	5
214W	Initialize recording memory *	214	–
223W	Download sound library *	223	–
236W	Download sound file *	236	–
247W	Download exception dictionary *	247	–
255W	Write 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 audio playback *	0-99	–
n&	Play sound file	0-65534	–
n$	ADC Control Register	0-255	128
n?	Chipset ID/Interrogate *	6/12	–
CTRL+X/Y	Stop/Skip	–	–

* Cannot be used in greeting messages.

Table 2.17. RC8660 Command Summary

SECTION 3: EXCEPTION DICTIONARIES

Exception dictionaries make it possible to alter the way the RC8660 interprets character strings it receives. This is useful for correcting mispronounced words, triggering the generation of tones and/or the playback of pre-recorded 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 RC8660.

The text-to-speech modes of the RC8660 utilize an English lexicon and letter-to-sound rules to convert text the RC8660 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. However, the dictionary must be compiled into the internal format used by the RC8660 before it can be used. 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 may 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 3.1).

Note that although context tokens are, by definition, valid only within 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.*

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
`	Wildcard (matches any character)

Table 3.1. Context Tokens

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 2.1), phoneme attribute tokens (Table 2.2), and commands (Table 2.17). Using the tone generator and pre-recorded audio playback commands, virtually limitless combinations 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 conference, economy, and icon 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 rhyme (the $ context token represents any non-alphabetic character, such as a space between words; see Table 3.1).

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: receive. (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: receive.

`(EI)=IY` is the first rule to match the second e, so *ei* receives the sound `IY`. Processing resumes at the character receive, 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 exceptions. 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 a 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 RC8660 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.

```
(????)=    (YOU fill in the blanks!)
(????)=CTRL+A 30j10000000
```

The first example simply says nothing when the defined word is encountered, whereas a 1 kHz tone is played in the second example.

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} and 17^{th}–18^{th} 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:

```
(CTRL+D CLOSE)=K L OW S
```

Note the CTRL+D character (ASCII code 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 "CTRL+D close" 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 RC8660 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 RC8660 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
```

Message Macros

Certain applications may not be able to send text strings to the RC8660. 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 1.7) 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, you could make the character "0" (corresponding to all four control bits = 0) 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 RC8660 will wait indefinitely for a CR/NUL character that will never come. The timer command could be included in the greeting message.

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.

When Things Don't Work as Expected

On rare occasions, an exception may not work as expected. This occurs 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 much to our horror, the RC8660 simply refuses to take on the new Canadian accent.

It so happens that the RC8660 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).

SECTION 4: RC8660 EVALUATION KIT

The RC8660 Evaluation Kit comes with everything required to evaluate and develop applications for the RC8660 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 RC8660 voice controls
- Manage collections of sound files and store them in the RC8660
- Exception dictionary editor/compiler, and much more...

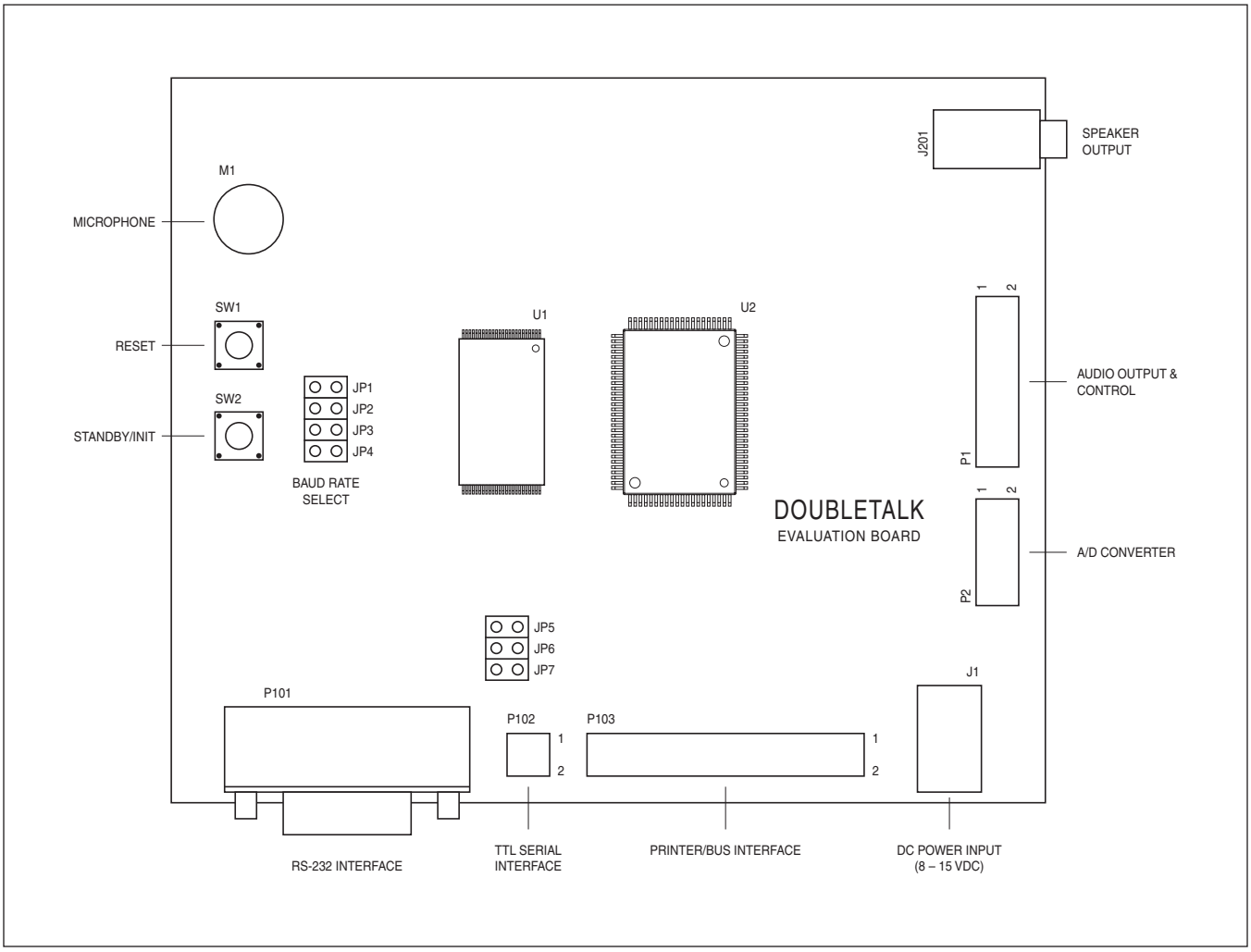
The evaluation board can also be used in stand-alone environments by simply printing the desired text and commands to it via the onboard RS-232 serial or parallel ports.

EVALUATION KIT CONTENTS

The following components are included in the DoubleTalk RC8660 Evaluation Kit:

- Printed circuit board containing the RC8660-1 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 4.1. 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 4.2. P2 Pin Assignments (A/D Converter)

JP4	JP3	JP2	JP1	Baud Rate
X	X	X	X	300
X	X	X		600
X	X		X	1200
X	X			2400
X		X	X	4800
X		X		9600
X			X	19200
X				Auto-detect
	X	X	X	38400
	X	X		57600
	X		X	115200
	X			Auto-detect
		X	X	Auto-detect
		X		Auto-detect
			X	Auto-detect
				Auto-detect (default)

Table 4.3. JP1-JP4 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 4.4. P101 Pin Assignments (RS-232 Serial Interface)

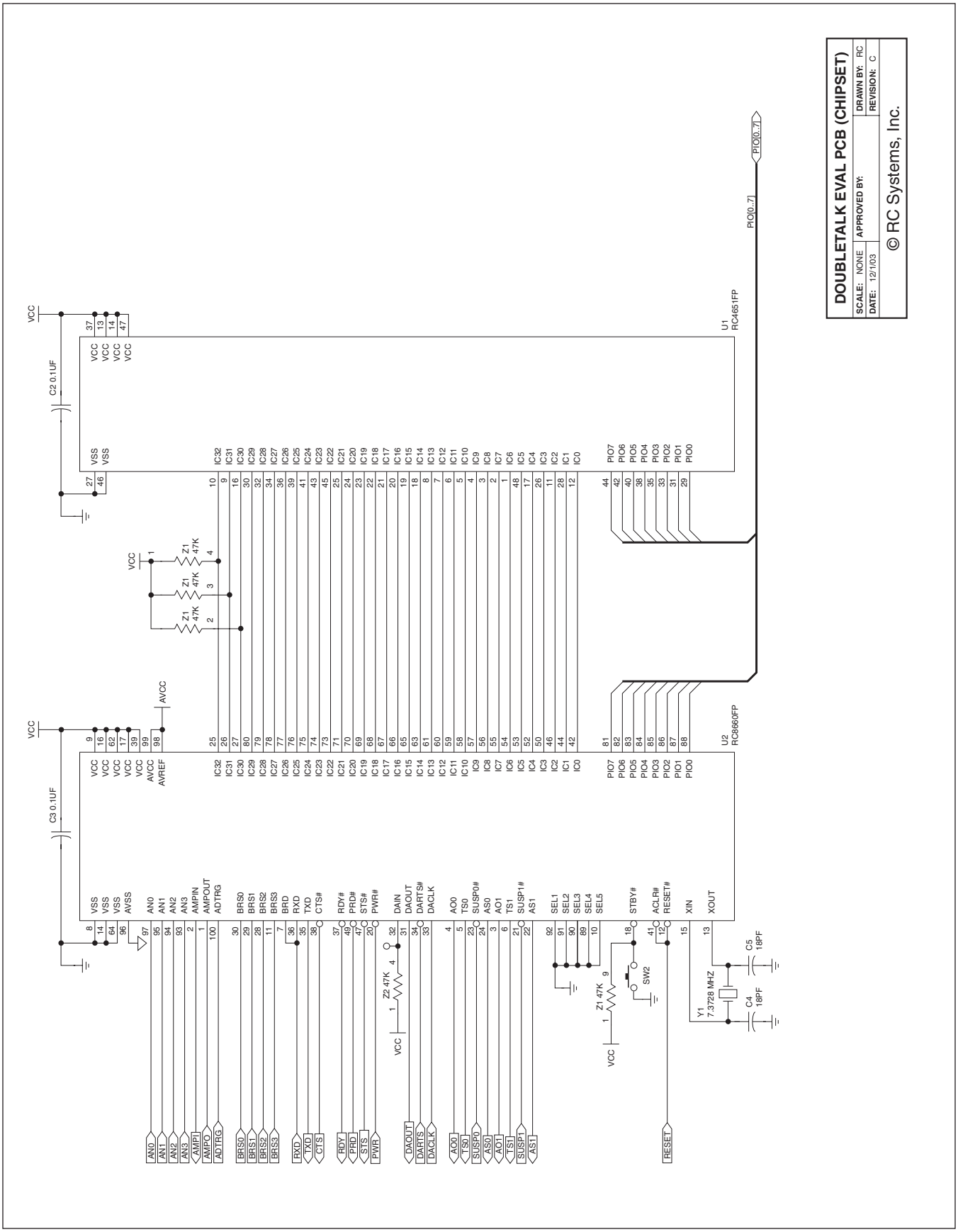
Pin No.	Pin Name	Pin No.	Pin Name
1	GND	3	TXD
2	CTS	4	RXD

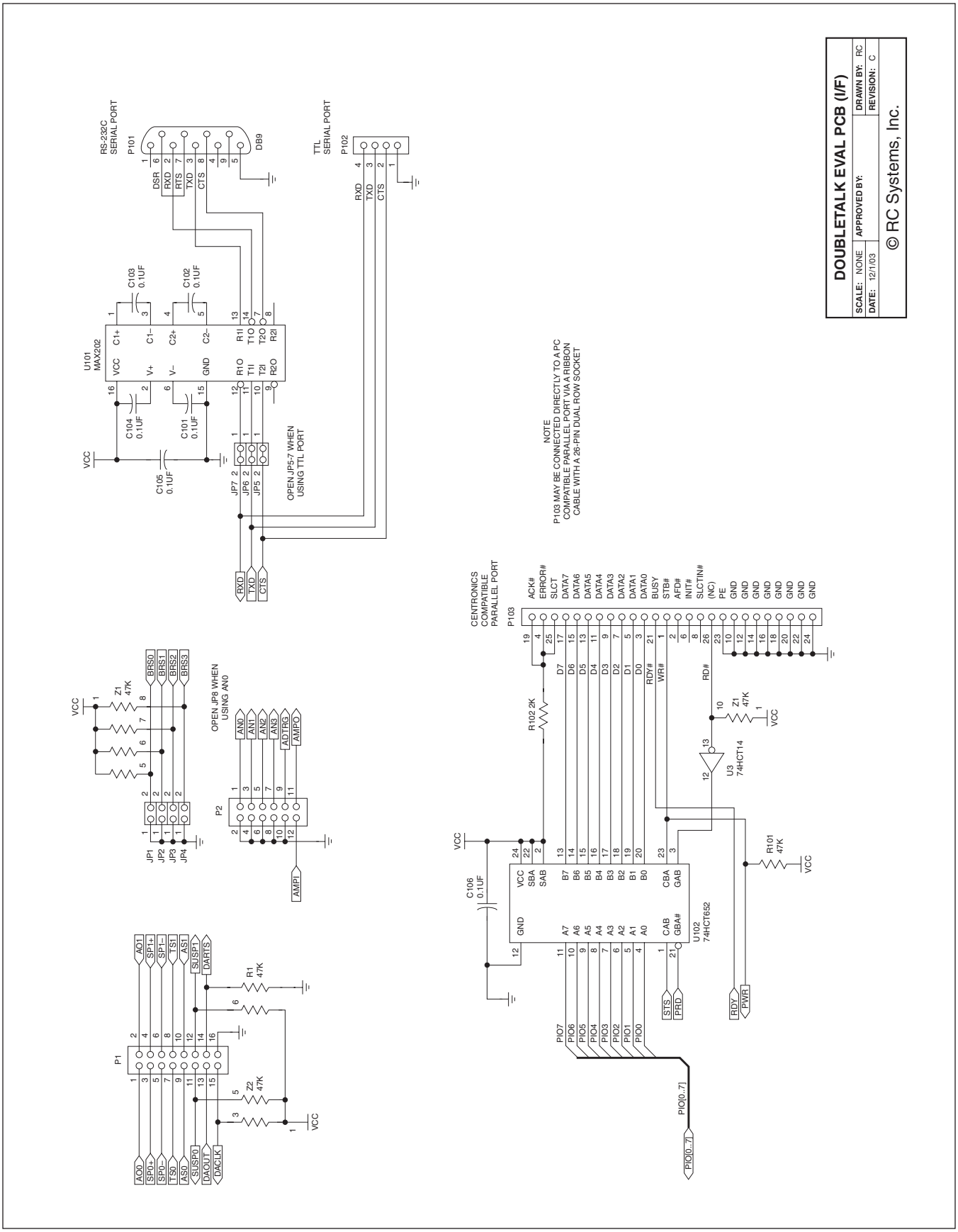
Note: JP5-JP7 must be open in order to use the TTL interface

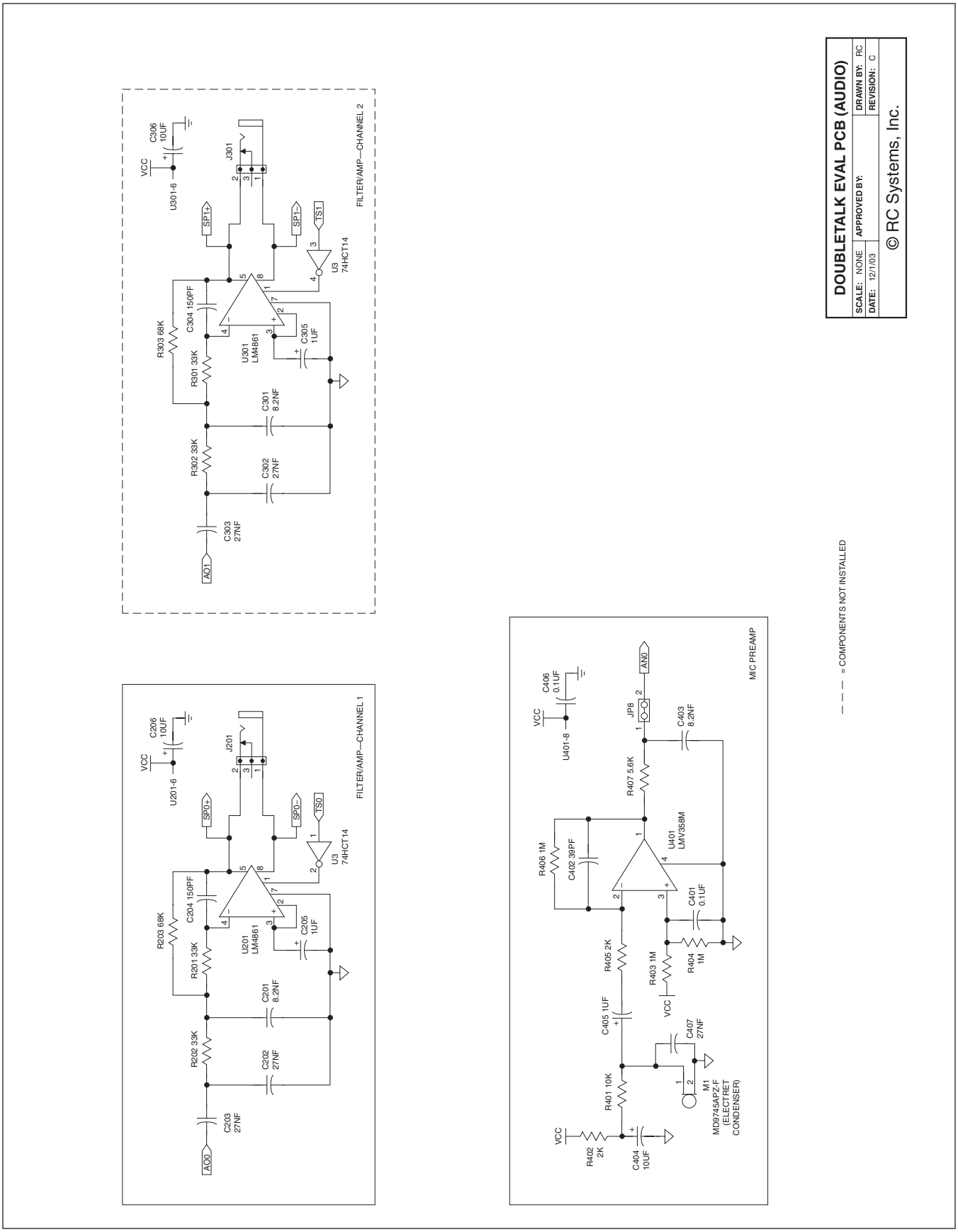
Table 4.5. P102 Pin Assignments (TTL Serial Interface)

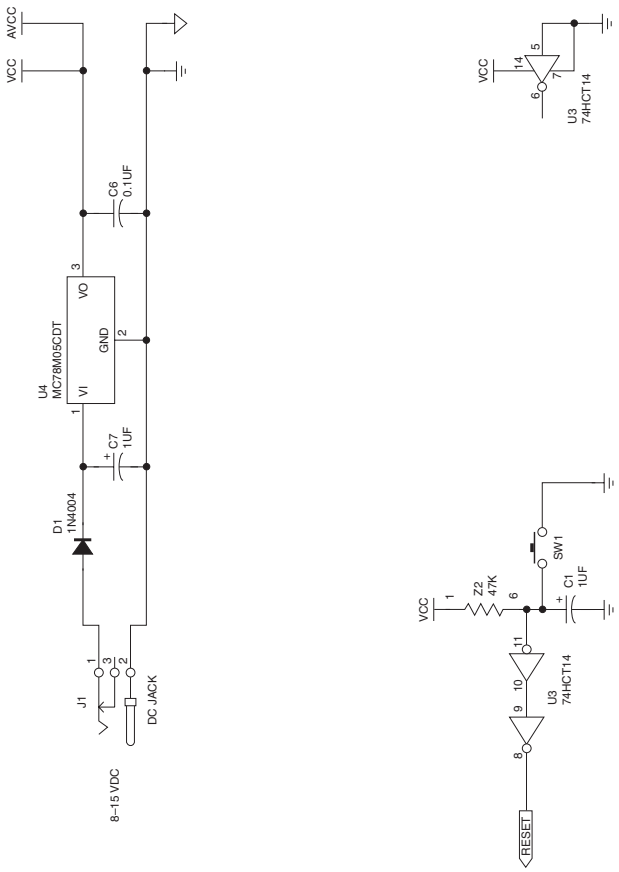
Pin No.	Pin Name	Pin No.	Pin Name
1	STB#	14	GND
2	AFD#	15	DATA6
3	DATA0	16	GND
4	ERROR#	17	DATA7
5	DATA1	18	GND
6	INIT#	19	ACK#
7	DATA2	20	GND
8	SLCTIN#	21	BUSY
9	DATA3	22	GND
10	GND	23	PE
11	DATA4	24	GND
12	GND	25	SLCT
13	DATA5	26	RD#

Table 4.6. P103 Pin Assignments (Printer/Bus Interface)


DOUBLETALK EVAL PCB (CHIPSET)
SCALE: NONE
APPROVED BY:
DRAWN BY: RC
DATE: 12/1/03
REVISION: C
© RC Systems, Inc.
U1 RC4651FP
U2 RC8660FP
PIO[0..7]
VCC
AVCC
C2 0.1UF
C3 0.1UF
C4 18PF
C5 18PF
Y1 7.3728 MHZ
Z1 47K
Z2 47K
SW2
AN0
AN1
AN2
AN3
AMPI
AMPO
ADTRG
BRS0
BRS1
BRS2
BRS3
RXD
TXD
CTS
RDY
PRD
STS
PWR
DAOUT
DARTS
DACLK
AO0
TS0
SUSPO
AS0
AO1
TS1
SUSP1
AS1
SEL1
SEL2
SEL3
SEL4
SEL5
STBY#
ACLR#
RESET
XIN
XOUT


DOUBLETALK EVAL PCB (I/F)
SCALE: NONE
APPROVED BY:
DRAWN BY: RC
DATE: 12/1/03
REVISION: C
© RC Systems, Inc.
RS-232C SERIAL PORT
P101
DB9
TTL SERIAL PORT
P102
U101 MAX202
C101 0.1UF
C102 0.1UF
C103 0.1UF
C104 0.1UF
C105 0.1UF
OPEN JP5-7 WHEN USING TTL PORT
NOTE
P103 MAY BE CONNECTED DIRECTLY TO A PC COMPATIBLE PARALLEL PORT VIA A RIBBON CABLE WITH A 26-PIN DUAL ROW SOCKET
CENTRONICS COMPATIBLE PARALLEL PORT
P103
R102 2K
U102 74HCT652
U3 74HCT14
C106 0.1UF
R101 47K
Z1 47K
OPEN JP8 WHEN USING AN0
P2
JP1
JP2
JP3
JP4
R1 47K
Z2 47K
P1
PIO[0..7]


FILTER/AMP—CHANNEL 2
VCC
C306 10UF
U301-6
J301
SP1+
SP1-
TS1
U3 74HCT14
R303 68K
C304 150PF
R301 33K
U301 LM4861
C305 1UF
C301 8.2NF
R302 33K
C302 27NF
C303 27NF
AO1
FILTER/AMP—CHANNEL 1
VCC
C206 10UF
U201-6
J201
SP0+
SP0-
TS0
U3 74HCT14
R203 68K
C204 150PF
R201 33K
U201 LM4861
C205 1UF
C201 8.2NF
R202 33K
C202 27NF
C203 27NF
AO0
MIC PREAMP
VCC
C406 0.1UF
U401-8
JP8
AN0
C403 8.2NF
R407 5.6K
R406 1M
C402 39PF
U401 LMV358M
C401 0.1UF
R405 2K
R403 1M
R404 1M
C405 1UF
C407 27NF
R401 10K
M1 MD9745APZ-F (ELECTRET CONDENSER)
R402 2K
C404 10UF
DOUBLETALK EVAL PCB (AUDIO)
SCALE: NONE
APPROVED BY:
DRAWN BY: RC
DATE: 12/1/03
REVISION: C
© RC Systems, Inc.
– – – = COMPONENTS NOT INSTALLED



DOUBLETALK EVAL PCB (P/S)		
SCALE: NONE	APPROVED BY:	DRAWN BY: RC
DATE: 12/1/03		REVISION: C

SECTION 5: COMPATIBILITY WITH THE RC8650

Item		RC8660	RC8650
I/O Pins	Pin 11	BRS_3: Fourth baud rate select pin	NC: No connection
	Pin 38	CTS#: Goes High only when there is data in the internal serial port overrun buffer	CTS#: Pulses High after each byte received
	Pins 10, 89–92	SEL pins fully support multi-channel applications	SEL pins not supported (connect to Low level)
Memory	Input buffer	8 KB	2 KB
	Greeting message	Downloading of Greeting message does not affect exception dictionary	Downloading of Greeting message erases exception dictionary (dictionary must be reloaded)
	Recording memory	130–910 seconds recording time	0–910 seconds recording time
		Internal file system allows individual sound files to be added and deleted in the field	No internal file system (entire sound library must be re-downloaded for changes to take effect)
		Record voice and analog data via ADC input pins; play back and/or upload recordings via serial port	Not supported
Commands†	Load Dictionary	247W	L
	Formant Freq	0–99F (50F)	0–9F (5F)
	Speed	0–13S (5S)	0–9S (1S)
	Voice	0–11O (0O)	0–7O (0O)
	Index Marker	0–255I	0–99I
	Interrogate (12?)	Returns same information as RC8650, plus free space available and # of files in recording memory	
	Sinusoidal tone generator	0–4400 Hz frequency range; 10 ms–600 sec duration range; simplified command syntax	0–2746 Hz frequency range; 23 ms–16.5 sec duration range
Data File Compatibility	Greeting messages	Compatible with RC8650 greeting files	
	Dictionaries	Compatible with RC8650 dictionary files	
	Sound libraries	Compatible with RC8650 and RC8660 library files, but internal RC8660 file system is compatible only with RC8660 libraries	Compatible with RC8650 library files only; individual sound file changes can be done only by downloading the new library file in its entirety
	Sound files	Can play both RC8650 and RC8660 sound files in real time playback mode, but POR.4 must be set to "0" to play RC8650 ADPCM-encoded files. Internal RC8660 file system is compatible only with RC8660 sound files when used in sound libraries	Compatible with RC8650 sound files only

† In RC8650 Compatibility mode (POR.4 = "0"), the RC8660 uses the RC8650 command ranges and defaults.

Specifications written in this publication are believed to be accurate, but are not guaranteed to be entirely free of error. RC Systems reserves the right to make changes in the devices or the device specifications described in this publication without notice. RC Systems advises its customers to obtain the latest version of device specifications to verify, before placing orders, that the information being relied upon by the customer is current.

In the absence of written agreement to the contrary, RC Systems assumes no liability relating to the sale and/or use of RC Systems products including fitness for a particular purpose, merchantability, for RC Systems applications assistance, customer's product design, or infringement of patents or copyrights of third parties by or arising from use of devices described herein. Nor does RC Systems warrant or represent that any license, either express or implied, is granted under any patent right, copyright, or other intellectual property right of RC Systems covering or relating to any combination, machine, or process in which such devices might be or are used. RC Systems products are not intended for use in medical, life saving, or life sustaining applications.

Applications described in this publication are for illustrative purposes only, and RC Systems makes no warranties or representations that the devices described herein will be suitable for such applications.

1609 England Avenue, Everett, WA 98203
Phone: (425) 355-3800 Fax: (425) 355-1098
http://www.rcsys.com