HCS-VOICE

Text to Speech Synthesizer

Rev. 1.0

***** Notice *****

Resistor R3 has been changed from 1M ohm (brn-blk-grn) to 100k ohm (brn-blk-yel). Please make note of this when assembling the board.

RTC-VOICE-KIT

***** CAUTION *****

PLEASE READ THE ENTIRE MANUAL ** BEFORE ** ATTEMPTING ASSEMBLY !!
Understand the significance of each component and jumper setting
* BEFORE * assembly and applying power. CHECK AND RECHECK !
FAILURE TO DO SO WILL PUT YOUR WARRANTY AT RISK !!

CIRCUIT CELLAR INC.

4 Park Street ● Vernon, CT 06066

TECHNICAL MANUAL

HCS-VOICE

Text to Speech
Synthesizer

Technical Manual

Release 1.0
10/14/93

Circuit Cellar Inc.
4 Park St.
Vernon, CT 06066

COPYRIGHT

DISCLAIMER

While we have attempted to provide accurate and up-to-date information in this manual, Micromint Inc. makes no representations or warranties respecting its contents. We reserve the right to make periodic changes to the text and to issue new editions of this manual without notification.

Occasionally in this manual we refer to other manufacturers' products. Such references do not constitute an endorsement of these products, but are included for the purpose of illustration or clarification. We do not intend such technical information and interface data to supersede information provided by individual manufacturers.

Conditions of Sale and Product Warranty

Circuit Cellar, Inc. (CCI) and the Buyer agree to the following terms and conditions of Sale and Purchase:

1. CCI extends the following warranty: a factory manufactured circuit board or assembly carries with it a one-year warranty covering both parts and labor. Any unit which is found to have a defect in materials or workmanship within this period will, at the discretion of CCI Inc., be repaired or replaced.

2. Products distributed, but not manufactured by CCI carry the full original manufacturer's warranty, usually 90 days. Such products include, but are not limited to, power supplies, sensors, I/O modules, LCD displays, and battery-backed RAM modules.

3. NO WARRANTY is extended on USER ASSEMBLED systems or kits. However, assembled kits will be inspected and repaired with charges based on the current minimum one hour charge. CCI retains the right to refuse to repair any USER ASSEMBLED item. This right is at the sole discretion of CCI. However, in the event that the repair charge would exceed a reasonable amount, the user may be consulted for determination. Repairs on user assembled items must be prepaid. Return authorization must be obtained prior to any return.

4. A minimum inspection fee must be prepaid for the repair of units no longer under warranty. Contact CCI for information on current minimum charges.

5. CCI will not be responsible for the repair or replacement of any unit damaged by user modification, negligence, abuse and/or mishandling, or improper installation.

6. CCI is not responsible to the Buyer for any loss or claim of special or consequential damages.

7. All units returned for repair must first receive prior authorization from CCI. A return authorization number may be obtained by phone or letter. Please retain a record of this number, because most subsequent correspondence will refer to this authorization. Under no circumstances should any product be returned to CCI without prior authorization. CCI will assume no responsibility for unauthorized returns. Returns must be shipped prepaid. Insurance is recommended as losses by a shipping carrier are not the responsibility of CCI. Repaired units will be returned postage and insurance paid.

8. CCI reserves the right to alter any feature or specification at any time. This right extends to fees, charges, and any other conditions or warranties contained herein.

REV. 5/93

Table of Contents

HCS-VOICE Board Overview

The HCS-VOICE board is designed as an I/O expansion board for the HCS stacking bus. The HCS-VOICE board is actually a fully functional embedded controller which can perform text to speech conversions without any external control. The HCS processor only has to hand off text to the HCS-VOICE board to make it speak. The "hand off" is handled through three I/O registers.

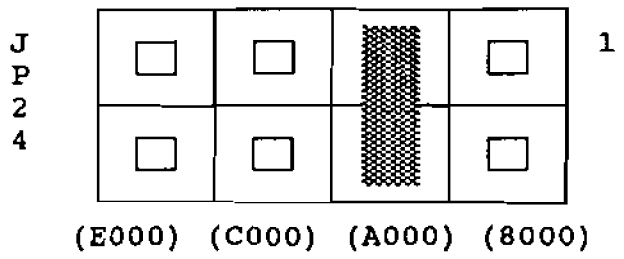
The first register is used as the destination in transfers of text streams, character by character. The second register reflects the current status of HCS-VOICE, while the third register can be used to provide a software reset on the VOICE microcontroller.

Addressing the HCS-VOICE Board

All HCS-bus I/O devices must be mapped into a non-conflicting address space. The HCS supervisory controller expects to find the HCS-VOICE board at a particular address. Since all HCS expansion boards have multiple address possibities to avoid potential address conflicts, make sure yours is set correctly.

Base Address

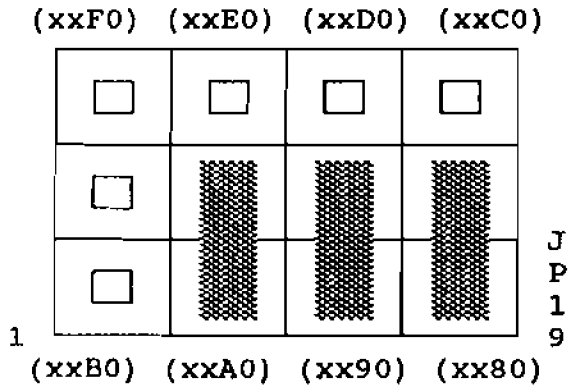
The Base Address choice for the HCS-VOICE board is Axxx (A000H-BFFFH).



A jumper on JP24 is shown selecting Base Address Axxx (A000H-BFFFH).

Offset Address

The Offset Addresses for the HCS-VOICE board are x80 for outputting text, x90 for reading the input status, and xA0 for reset. (xB0 is not used).



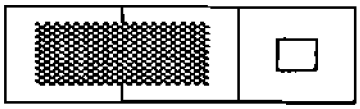
Jumpers on JP19 are shown selecting Offset Addresses x80, x90, and xA0.

The Base address plus the Offset address equals the actual address. Therefore when using the examples above for JP19 and JP24, the actual addresses for this board are as follows:

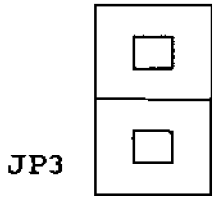
JP24	JP19	Actual Address	Function
Axxx +	x80 =	Ax80 **(A080H)**	Outputing text
Axxx +	x90 =	Ax90 **(A090H)**	Input status
Axxx +	xA0 =	AxA0 **(A0A0H)**	Reset

Reset

JP2 enables the system reset from the HCS expansion headers to force a reset of the HCS-VOICE on powerup.

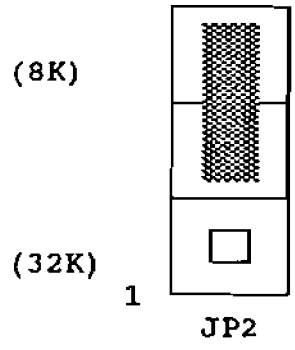


Additionally, shorting the pins on the JP3 header will cause a manual reset of the VOICE card.

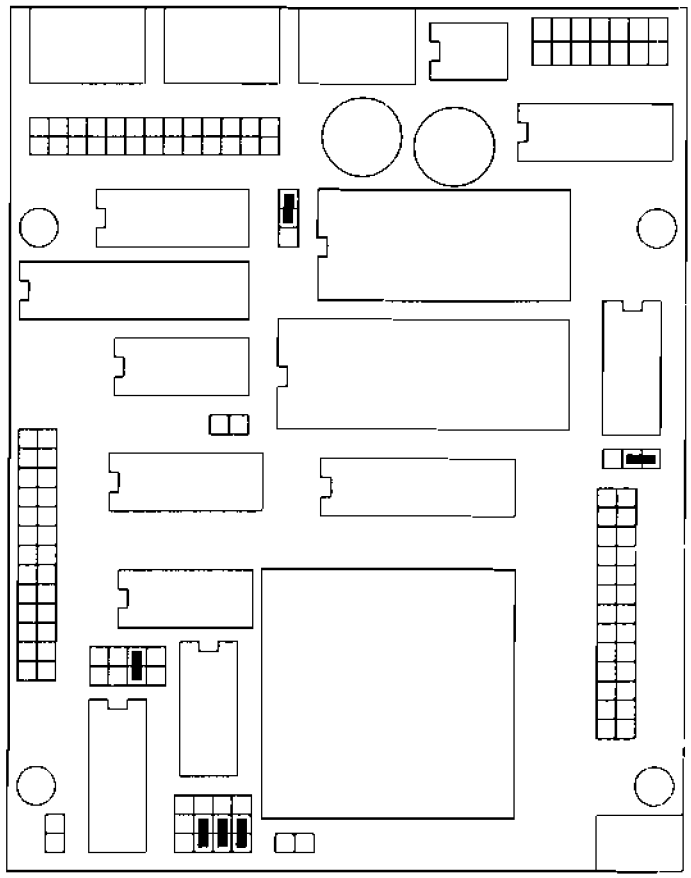


RAM Use

When the HCS-VOICE board uses 8K of static RAM, the text buffer and exception table size is appromimately 2K characters each. This leaves about 4K for the embedded controller's use. Replacing the 8K RAM with 32K RAM does NOT increase the input buffer, but does increase the exception table space to about 26K. Using a non-volatile RAM will protect the exception table while the system is off. For most applications, 8K static RAM is more than sufficient.



JP2 is shown selecting an 8K RAM for IC socket U4



Default Jumper Configuration

Audio Outputs

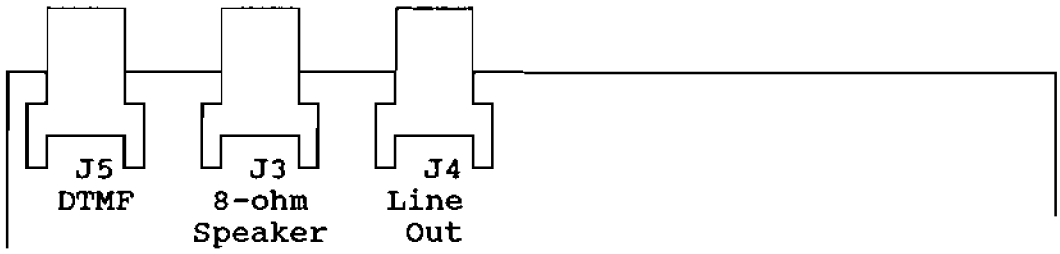
Three audio outputs are simultaneous available on the HCS-VOICE board. All output levels vary according to the setting of POT1.

To attach the HCS-VOICE board directly to an 8-ohm speaker, use J3.

To attach the HCS-VOICE board to external audio equipment, use the line output J4.

To attach the HCS-VOICE to the HCS-DTMF board use J5 on the HCS-VOICE to J2 (line in) on the HCS-DTMF.

Since the audio into the HCS-DTMF board is most critical adjust POT1 for the highest volume that does not trigger the squelch circuitry on the HCS-DTMF board, then adjust the gain on your external audio equipment appropriately.

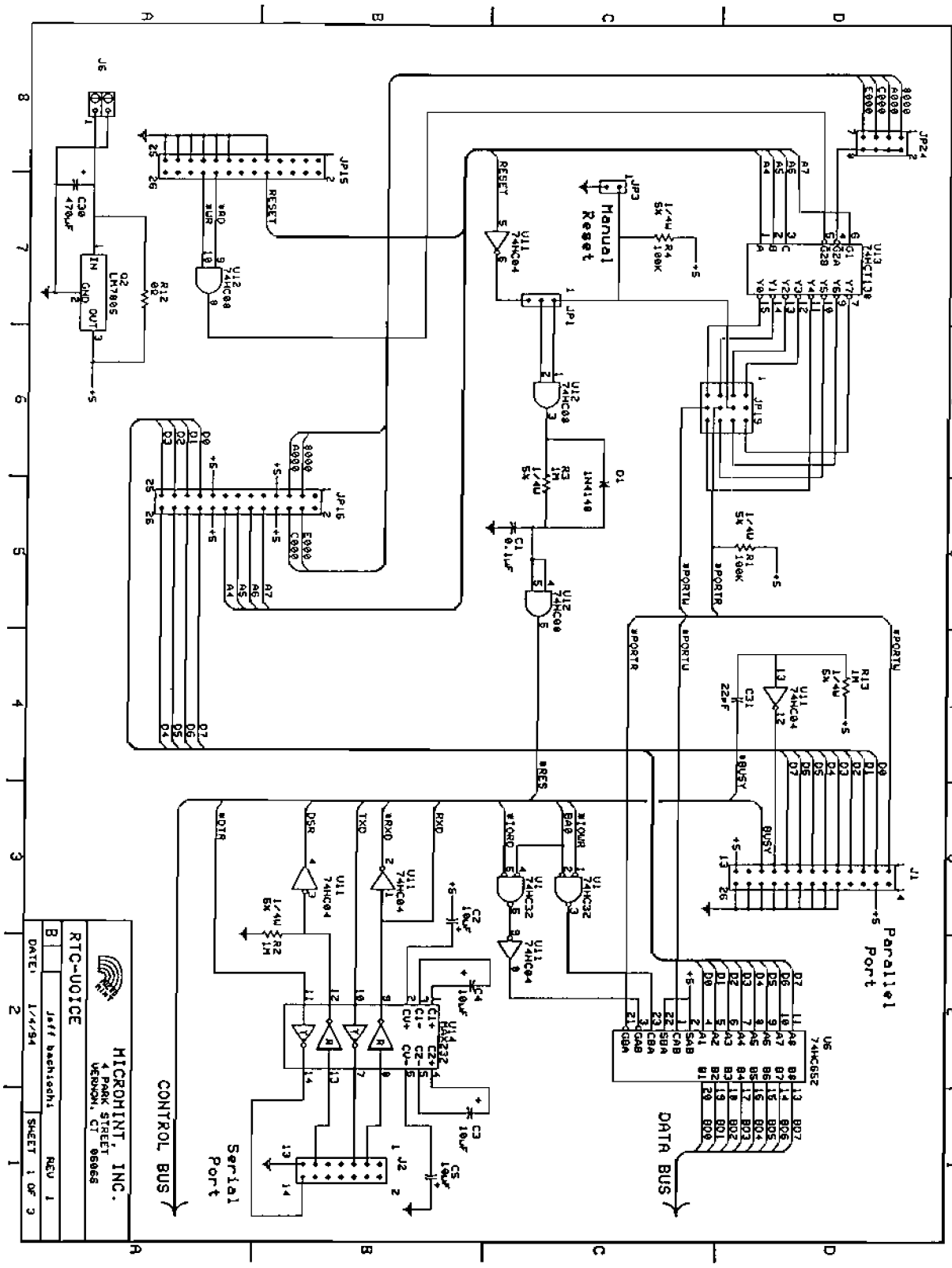


Adding the HCS-VOICE board to your supervisory controller.

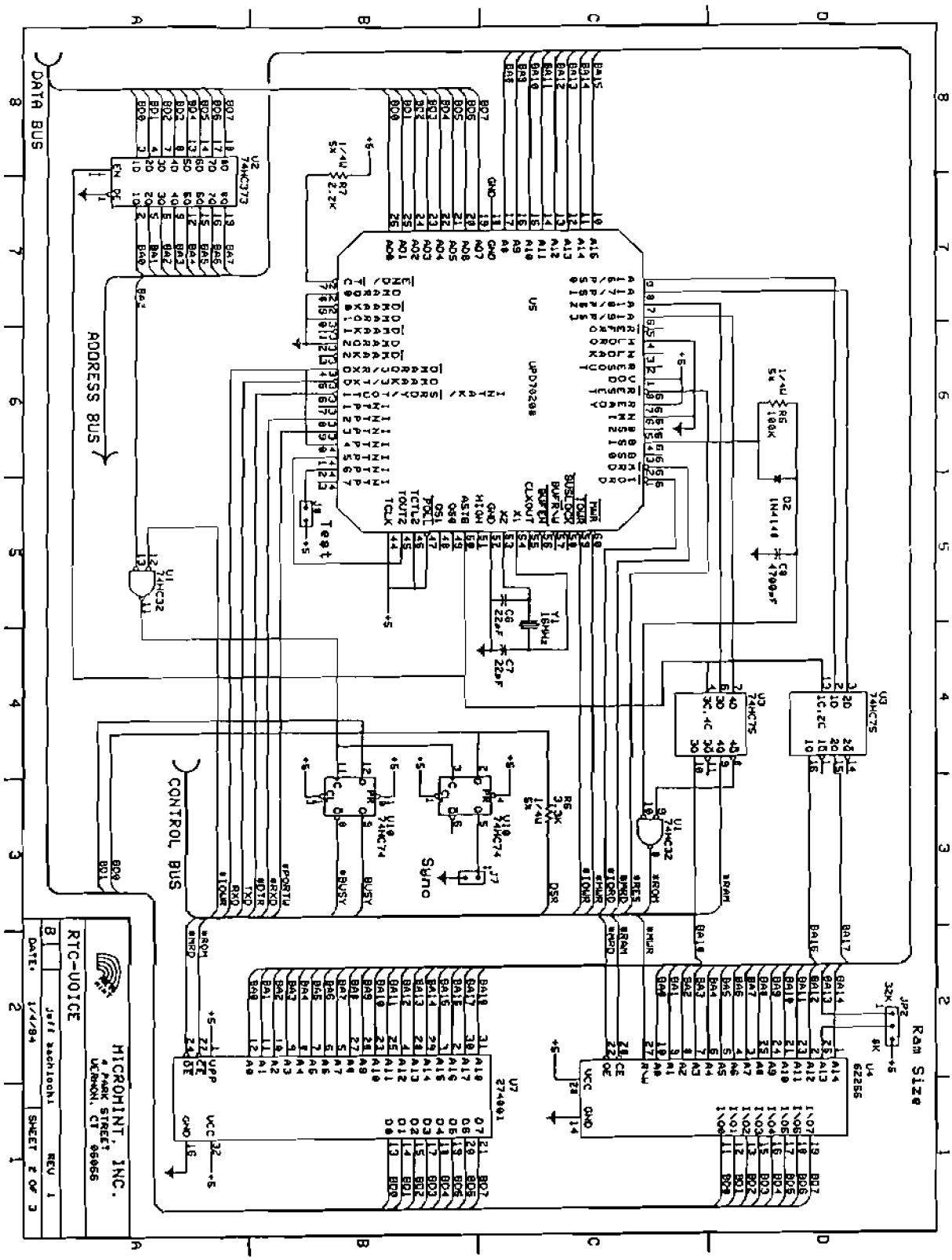
In an effort to prevent misalignment of the HCS-VOICE board on the supervisory controller's expansion headers, a universal keying system has been implemented. The expansion headers, JP15 and JP16, each have pin missing, pin 18 on JP16 and pin 23 on JP15. Corresponding holes are blocked on the expansion sockets on the solder side of the HCS-VOICE board. Carefully match these keyed positions between the HCS and HCS-VOICE board. When properly mated, the four 1/8 inch mounting holes in the HCS-VOICE board will exactly match four mounting holes on the board below.

Boards permanently mounted (and not subjected to vibration) will not need standoffs between boards, the expansion connectors will act as electrical and mechanical connections. If standoffs are used, 5/8 inch #4 nylon are recomended (older boards may use 1/2 inch #4 spacers, measure your boards to be sure).

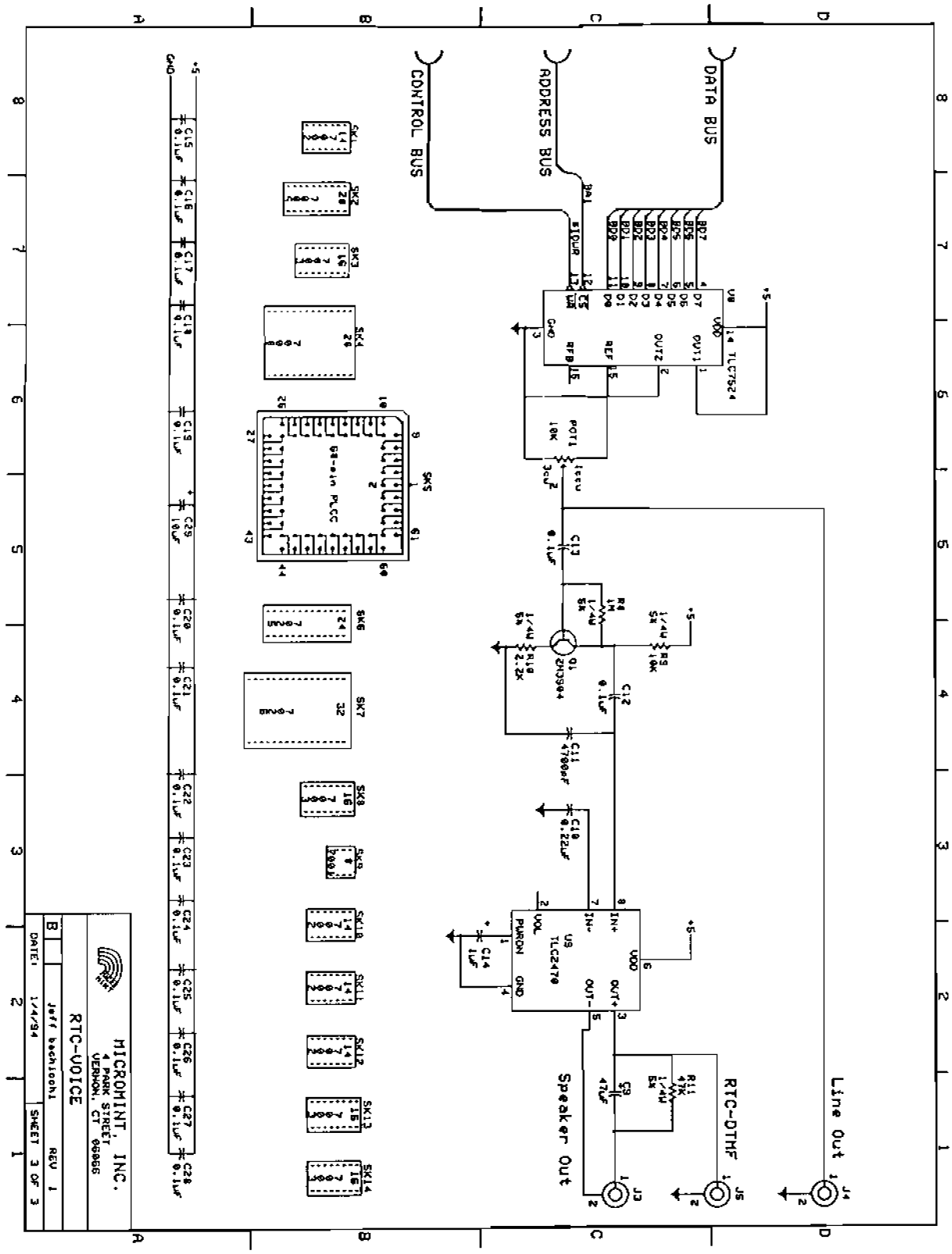
If you are stacking the HCS-VOICE and HCS-DTMF, we suggest placing the HCS-DTMF at the top of the stack. This will allow easy access to the modular phone jack on the HCS-DTMF board.



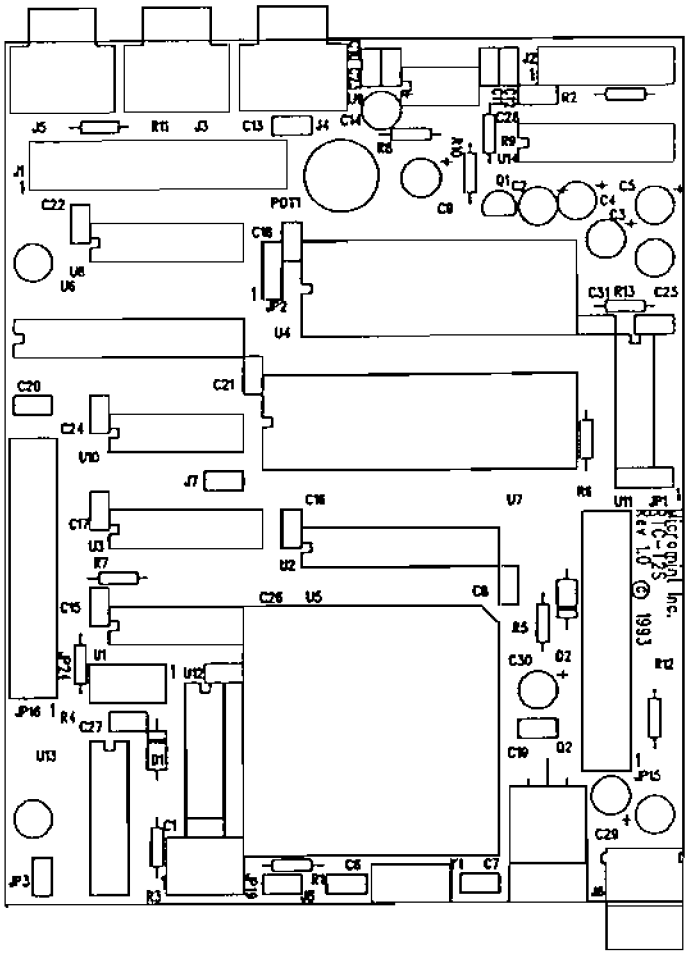
HCS-VOICE Schematic (1 of 3)



HCS-VOICE Schematic (2 of 3)



HCS-VOICE Schematic (3 of 3)



Silkscreen for the HCS-VOICE

PARTS LIST for the HCS-VOICE Board

DESIGNATION	PART#	DESCRIPTION
Printed Circuit Board		
PB1	HCS-VOICE	RTC style Printed Circuit Board
Integrated Circuits		
U1	74HC32	Quad OR Gate
U2	74HC373	Octal Tri-state Latch
U3	74HC75	4-bit Bistable Latch
U4	62256	32Kx8 RAM
U5	UPD70208(V40)	CMOS Microprocessor
U6	74HC652	Octal Registered Bus Transceiver
U7	27C4001	512Kx8 EPROM (programmed)
U8	TLC7524	8-bit Multiplying D/A
U9	TLC2470	Differential Audio Filtered Amplifier
U10	74HC74	Quad D-Flip Flop
U11	74HC04	Hex Inverter
U12	74HC08	Quad AND Gate
U13	74HC138	3-8 Decoder
U14	**MAX232**	**Level Shifter (not used)**
Resistors		
R1, R4-R5, R8	100K	1/4W, 5%, (brn-blk-yel)
R2-R3, R13	1M	1/4W, 5%, (brn-blk-grn)
R6	3.3K	1/4W, 5%, (org-org-red)
R7, R10	2.2K	1/4W, 5%. (red-red-red)
R9	10K	1/4W, 5%, (brn-blk-org)
R11	47K	1/4W, 5%, (yel-vio-org)
R12	0Ω	not used
POT1	10K	Potentiometer
Capacitors		
C1, C12-C13, C15-C28	0.1μF	Monolithic
C2-C5	**10μF**	**Tantalum (not used)**
C6-C7, C31	22pF	Monolithic
C8, C11	4700pF	Monolithic
C9	47μF	Radial Electrolytic
C10	0.22μF	Monolithic
C14	1μF	Tantalum
C29	10μF	Tantalum
C30	**470μF**	**Radial Electrolytic (not used)**

PARTS LIST for the HCS-VOICE Board (continued)

DESIGNATION	PART#	DESCRIPTION
Semiconductors		
D1-D2	1N4148	Small Signal Diode
Q1	2N3904	NPN Transistor
Q2	**7805**	**5-volt Regulator (not used)**
Connectors		
J1	**2x13**	**Square Pin Header (not used)**
J2	**2x7**	**Square Pin Header (not used)**
J3-J5		Phone Jack
J6	**1x2**	**Screw Terminal Block (not used)**
J7-J8, JP3	1x2	Square Pin Header
JP1-JP2	1x3	Square Pin Header
JP15-JP16	2x13	.630 Vertical Stacking Header
JP19	3x4	Square Pin Header
JP24	2x4	Square Pin Header
Sockets		
SK1, SK10-SK12	14-Pin	IC Socket
SK2	20-Pin	IC Socket
SK3, SK8, SK13	16-Pin	IC Socket
SK14	**16-Pin**	**IC Socket (not used)**
SK4	28-Pin	IC Socket
SK5	68-Pin	PLCC Socket
SK6	24-Pin	IC Socket
SK7	32-Pin	IC Socket
SK9	8-Pin	IC Socket
Miscellaneous		
Y1	16MHz	Crystal
SJ1-SJ6		Shorting Jumper

Generic Instructions for Kit Assembly

Tools Required for Assembly:

Low Wattage or Temperature Controlled Soldering Iron
Rosin Core Solder
Lead Cutters

Additional Tools to Ease the Assembly:

Needle-Nose Pliers
Lead Bender
Solvent for removing Rosin (flux)
Screwdriver

Tools for trouble-shooting (and their use):

Oscilloscope	Viewing AC/DC signals
Logic Probe	Indicating Logic Levels (activity)
Continuity Checker	Determining Shorted or Open Traces
Volt/Ohm Meter	Checking Power Supply, Logic Levels, or continuity

Familiarize yourself with all of the parts included in the kit. Pay particular attention to proper orientation of parts. Markings might include a bump, hole, number, arrow, or notch indicating pin 1 (or the pin 1 end) of ICs, sockets, and other devices. A stripe may indicate a plus or minus potential lead of a capacitor or the cathode of a diode. An LED might indicate the cathode by a notch or flat side on the girth of the component.

Inspect the PC Board prior to installing any parts. If held up to a lamp, you can usually see the signal traces fairly clearly. Eye each trace for defects, a copper short between adjacent traces or pads, or a break in the copper trace. Verify any traces that look shorted by first looking at the schematic to verify they should not be connected and then checking for continuity between the traces. You should check traces which seem to be broken for continuity as well. Circuit Cellar Inc. inspects each and every board for manufacturing defects; we feel confident that the components packaged for you are free from defects. However, discovering a PCB defect is much easier before any parts are inserted which would obscure a defect from view.

This is a generic overview of the construction process. At this time, please read the Kit Specific Instructions if any are packaged with the kit. These explain any special requirements or mounting procedures necessary for individual components. Place these parts aside until the rest of the kit is completed.

The best approach to use in building a kit is to choose the smallest (or shortest) parts to install first. Start with any small signal diodes. Use the FOIBSAT method on each part or group of parts. Find, Orient, Insert, Bend, Solder, And Trim each part. Find the part, verify it using the parts list and silkcreen layout. Orient the part correctly using markings on the part and the silkscreen layout. Insert the part into its designated location, forming the component leads as necessary. Bend over the component leads to prevent the part from falling out, preferably in the direction of the connecting circuit board trace. Solder each lead filling in the area between the lead and its plated through hole. And finally, Trim the leads to prevent shorting between traces, components, or holes.

Continue with the 1/4 watt resistors, IC sockets, and smaller monolithic capacitors. Next, insert all the square pin headers and jumpers. These can't be easily bent and will simply fall out if not handled one at a time. Start by soldering only one or two pins, flipping over the board to check that the header isn't tilted. Fix, if necessary, by reheating the soldered pin and/or complete by soldering the rest of the header's pins.

Finally, the taller, odd-shaped components are added, transistors, LEDs, crystals, potentiometers, connectors, heatsinks, etc.

Now it's time to add those items listed in the Kit Specific Instructions (if attached). This is where you may be given special instructions on placement or a choice on how components are oriented according to your application.

Prior to installing the ICs, it is a good idea to give the PCB another inspection. Look for unsoldered or untrimmed leads. Cleaning the solder side of the PCB with a flux remover will make inspection easier, not to mention less sticky.

Another suggestion is to use a volt-ohm meter and measure the resistance between ground and any power supply input. A short circuit here could ruin your power supply and won't allow your board to operate properly. Power-up the board before inserting the ICs and check for voltages according to POWER TABLE listed in the schematics.

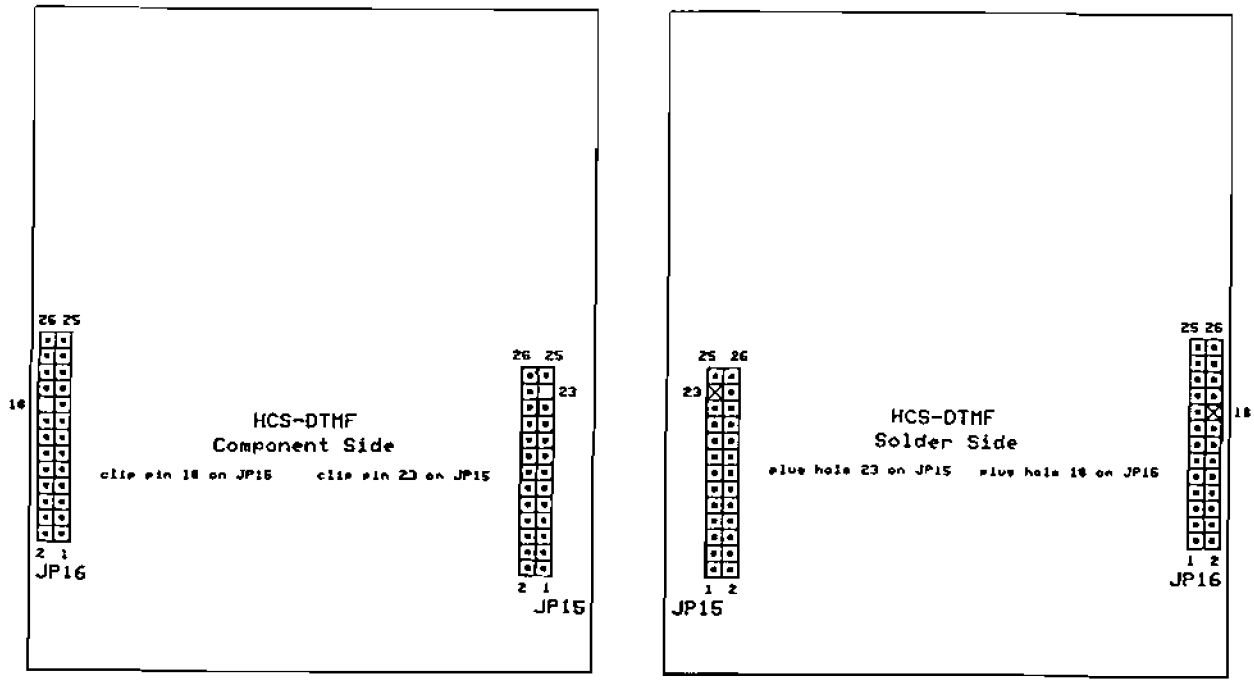
Refer to the parts list and silkscreen for proper IC orientation and insertion. This concludes the assembly instructions.

Kit Specific Instructions for the VOICE Board

JP15 and JP16 are vertical stacking connectors. These are inserted from the solder side up through the component side and are the only components that must be soldered on the component side of the PCB. Once inserted, solder only one or two pins and check to make sure the vertical stacking connector is NOT tilted and is snug with the solder side's surface. Once you are sure these are mounted correctly solder the rest of the pins.

Use extreme care in applying solder to the header's pins on the component side of the PCB, DO NOT let the solder cover the gold plating on the pins more than 1/4 inch from the component side's surface.

Once the expansion headers are soldered you need to key them correctly. This will help ensure proper mating between the HCS and the HCS-DTMF board. Carefully study the diagram below. Find pin 23 on JP15 and pin 18 on JP16. Verify you have the correct pins and clip them off on the component side of the HCS-DTMF board. Turn the board over and insert the tiny clear plastic plugs into the corresponding holes of the expansion headers.



U5, the V40 processor, will snap into the PLCC socket with a bit of finger pressure. It is designed to 'key' with the socket for correct insertion. The slashed corner on the silkscreen, the PLCC socket and the V40 should all match. Removing the V40 is difficult and should not be attempted without the proper tools. Damage to the V40, PLCC socket or both could result.

HCS-Voice
Speech Synthesizer
Command Manual

IMPORTANT NOTICE

The Speech Algorithm for the HCS-Voice board is licensed from RC Systems. Circuit Cellar Inc. and RC Systems reserve the right to make changes in the devices or the device specifications described in this publication without notice.

PROGRAMMING NOTICE

Refer to the HCS II System Reference Manual with XPRESS, for the programming syntax of HCS-Voice.

Commands

The commands described in the following pages provide the means of controlling the HCS-Voice under software control. They can be used to vary the synthesizer's attributes, such as the volume or pitch, to suit the requirements of a particular application or listener's preferences.

Commands can be freely intermixed with the text that is to be spoken, allowing the speech to be dynamically controlled. Unless otherwise noted, commands embedded in text affect only the text that follows the commands.

Command formats

All HCS-Voice commands are composed of a command character, a parameter *n*, and a letter which identifies the command. Some commands simply enable or disable a feature of the HCS-Voice; these commands do not require a parameter. The command formats are:

<command character><n><letter>
<command character><letter>

The command character

The default HCS-Voice command character is ~(7Eh). The command character itself can be spoken by the synthesizer by sending it twice in a row: `Control-A Control-A`. This special command allows the command character to be spoken without affecting the operation of the V860X, and without having to change to another command character and then back again.

Command parameters

Command parameters are composed of one or two-digit ASCII numbers. The HCS-Voice supports two types of parameters: *absolute* and *relative*. Absolute parameters explicitly specify the parameter's new value, such as 9V or 3B. Relative parameters specify a *displacement* in a parameter's value, not the actual new value itself.

Relative parameters may specify a positive or negative displacement from a parameter's existing value. For example, the command +2V increases the current value of V by two (V+2→V). If the current value of V is 4, it will increase to 6 after the command has executed. The command –2V will have a similar effect, except the V value will be *decreased* by two.

If the value of a parameter falls outside the command's specified range, the value will wrap around, instead of being truncated. For example, if the current volume is 7 and the command +4V is issued, the resultant volume will be (7+4)–10, or 1, since the volume range is 0-9.

When writing application programs for the V860X, 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.

Command Descriptions

The following is a description of each of the software commands supported by the V860X.

E (Enable Intonation)

Intonation is the variation of pitch throughout a sentence or phrase. When intonation is enabled, the synthesizer attempts to approximate the pitch patterns of human speech as closely as possible. For example, when a sentence ends with a period, the pitch drops at the end of the sentence. If a sentence ends with a question mark and the sentence does not begin with "wh" (who, what, where, etc.), the pitch rises—otherwise it falls, like a period.

When intonation is enabled, the pitch will also vary in the Character and Phoneme modes. The pitch follows the same rules for punctuation as it does for Text mode.

Intonation is normally enabled. To turn it off, use the Disable Intonation command, described next.

M (Disable Intonation)

If automatic intonation is not desired, this command will cause the synthesizer to speak in a monotone voice. Intonation should be disabled whenever manual intonation is applied using the Pitch command.

nF (Formant Frequency)

This command adjusts the synthesizer's overall frequency response (vocal tract formant frequencies), over the range 0F through 9F. By varying the frequency, speech quality can be fine-tuned or voice type changed. The default frequency is 5F. Note that this command takes effect immediately—any text sent prior to the command will be spoken at the new formant

frequency.

nS (Speed)

The synthesizer's overall rate (speed) of speech can be adjusted with this command, from 0S (slowest) through 9S (fastest). The default speed is 5S.

nP (Pitch)

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

nV (Volume)

This command controls the synthesizer's volume level. Volume commands can range from 0V through 9V; the default is 5V. The command 0V yields the lowest possible volume; maximum volume is attained at 9V. The command can be used to set a new listening level or create emphasis in speech.

nX (Tone)

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.

nB (Punctuation Level)

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

The V860X supports eight levels of punctuation, shown in Table 1. Besides determining which punctuation characters will be spoken and which will not, the Punctuation Level command also determines how number strings will be pronounced: as numbers (e.g., "one hundred twenty three") or digits ("one two three"). Levels 6 and 7 also cause currency strings to be read as they are normally spoken—for example, $11.95 is read as "eleven dollars and ninety five cents." The default punctuation level is 6B.

n	Punctuation Level
0	All/Digits
1	Most (all punctuation except CR, LF, Space)/Digits
2	Some ($%&#@=+*^<>|\)/ Digits
3	None/Digits
4	All/Numbers
5	Most/Numbers
6	Some/Numbers
7	None/Numbers

Table 1. Punctuation Levels

nY (Timeout Delay)

The Text and Phoneme modes of the synthesizer defer the translation and synthesis of the contents of the input buffer until a CR or Null is received. This ensures that text is spoken smoothly from word to word, and that the proper intonation is given to the beginning and ending of sentences. If text is sent to the synthesizer without a CR or Null, it will remain untranslated in the input buffer indefinitely. If it is expected that this condition may occur, use the Timeout command.

The V860X contains a software timer which forces the synthesizer to translate the buffer contents when the timer times out. The timer is enabled only if the Timeout parameter *n* is non-zero, the synthesizer is not active (not talking), and the input buffer contains no CR or Null characters. Any characters sent to the V860X before timeout will automatically restart the timer.

n	Delay
0	Indefinite (wait for CR/Null)
1	200 milliseconds
2	400 milliseconds
3	600 milliseconds
.	
.	
15	3000 milliseconds (3 sec.)

Table 2. 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 (Table 2). The default value is zero, which disables the timer.

L (Load Exceptions)

This command purges the V860X's exception dictionary and stores subsequent output from the host in the exception dictionary RAM. Because the memory used by the exception dictionary is the same physical RAM used by the input buffer, the space available for the input buffer is decreased proportionally by the size of the dictionary.

The dictionary can be purged from the V860X with the Reinitialize command, or by loading a "null" dictionary file into the V860X. Both methods reallocate the memory space occupied by the dictionary to the input buffer.

Although exceptions are composed of standard ASCII characters, they must be compiled into the internal format used by the V860X. Development software is available from RC Systems to aid in the development of exception dictionaries.

U (Enable Exceptions)

The exception dictionary is enabled with this command. If the synthesizer is not in the Text or Character modes, or if the exception dictionary is empty, the command will have no effect. The exception dictionary can be disabled by issuing one of the mode commands D, *n*T, or *n*C.

R (Clear)

The Clear command stops the synthesizer and clears the input buffer of all text and commands. None of the synthesizer settings are affected, but any untranslated commands will be ignored.

Alternate Clear method

The Clear command works well for all but the most critical applications where the synthesizer must react very quickly, such as text-readers for the visually impaired. Using the Clear command, the V860X must process two characters, as well as perform the handshaking when receiving them. This can become relatively sluggish if large amounts of text are being processed by the V860X. Worse yet, if the input buffer is full, the command may not be recognized for perhaps as long as 20 seconds.

If instead the host simply writes (|) to the V860X, the speech will stop faster. Functionally, there is no difference between this method and using the Clear command, except that the V860X reacts more quickly and works even when the input buffer is full. For this method to be most effective, the host should ignore the states of the V860X handshaking signals when writing the (|) character.

@ (Reinitialize)

This command reinitializes the V860X by clearing the input buffer and restoring the speech parameters to their default settings. The exception dictionary memory is also erased and reallocated to the input buffer.

Z (Zap Commands)

This command prevents the V860X from honoring subsequent commands, enabling it to read commands as they are issued (this can be useful for debugging some types of programs). 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 perform a hardware reset.

nT (Text Mode/Delay)

This command places the V860X in the Text translation mode (which is also the default mode). The optional delay parameter *n* can be 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 using a longer delay. The longest delay that can be specified is 15. If the delay parameter is omitted, the current value will be used. This feature is useful for returning from another translation mode or disabling the exception dictionary (see Enable Exceptions command).

D (Phoneme Mode)

This command disables the text-to-speech translator, allowing the synthesizer's phonemes to be accessed directly. Table 3 lists the phonemes that can be produced by the V8600.

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 T ER
```

Phoneme Symbol	Example Word	Phoneme Symbol	Example Word
A	das (Spanish)	M	mug
AA	father	N	new
AE	bat	NX	rung
AH	cut	NY	niño (Spanish)
AO	lawn	O	no (Spanish)
AW	cow	OW	tone
AX	about	OY	boy
AY	kite	P	past
B	bird	PX	spot
CH	cheese	R	ring
D	dare	RR	tres (Spanish)
DH	either	S	some
DX	computer	SH	dish
E	ser (Spanish)	T	tip
EH	set	TH	thick
EI	mesa (Spanish)	TX	mistake
ER	were	U	uno (Spanish)
EW	acteur (French)	UH	pull
EY	bake	UW	tool
F	fact	V	give
G	give	W	went
H	hire	WH	when
I	libro (Spanish)	Y	mayo (Spanish)
IH	sit	YY	you
IX	relative	Z	zero
IY	meet	ZH	leisure
JH	jet	space	variable pause*
K	cute	,	medium pause
KX	ski	.	long pause
L	long		

Table 3. Synthesizer Phonemes
** Normally used between words; duration determined by nT command*

Phoneme attribute tokens

Table 4 lists the voice attribute tokens that can be used in the Phoneme mode, in addition to the standard V860X commands. These tokens do not require the command character or any parameters.

The / token temporarily increases the pitch by ten steps, whereas the \ token temporarily decreases the pitch ten steps. Besides only *temporarily* changing the pitch, the difference between the pitch tokens and the +10P and −10P command equivalents is that the effective pitch range is extended beyond the normal 0-99 range by approximately ±20 steps, and if the pitch should fall out of range, it will simply bottom or top out, instead of wrap around.

All other attribute token commands remain in effect until explicitly changed.

Symbol	Function
nn	Set pitch to 'nn' (0-99)
/	Increase pitch 10 steps
\	Decrease pitch 10 steps
+	Increase speed 1 step
−	Decrease speed 1 step
>	Increase volume 1 step
<	Decrease volume 1 step

Table 4. Phoneme Attribute Tokens

Applications of Phoneme mode

Phoneme mode is useful for creating customized speech, when the normal text-to-speech (Text) mode is inappropriate for producing the desired voice effect. For example, Phoneme mode should be used when it is important that the correct stress or emphasis be placed on 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 program examples.

```
100 A$ = CHR$(1)
105 LPRINT A$;"D";A$;"M"
110 LPRINT "70H AW  ~/D>/EH R  +<\\YY
    UW   S P\IY K   T UW  \M IY   DH
    AE T  -\W EY .+/"
```

Note in line 105 that the synthesizer's intonation is disabled, 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 intonation enabled:

```
100 A$ = CHR$(1)
105 LPRINT A$;"T";A$;"E"
110 LPRINT "How dare you speak to me
    that way!"
```

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

Command	Function	Range *n*	Default*
*n*B	Punctuation level	0-7	6
*n*C	Character mode/delay	0-31	0
D	Phoneme mode	-	-
E	Enable intonation	-	-
*n*F	Formant frequency	0-9	5
L	Load exception dictionary	-	-
M	Disable intonation	-	-
*n*P	Pitch	0-99	50
R	Clear	-	-
*n*S	Speed	0-9	5
*n*T	Text mode/delay	0-15	0
U	Enable exception dictionary	-	-
*n*V	Volume	0-9	5
*n*X	Tone	0-2	1
*n*Y	Timeout delay	0-15	0
Z	Zap commands	-	-
@	Reinitialize	-	-

Table 6. Command Summary
** Defaults may be programmed with Write Defaults command (V8601 only)*

Exception Dictionaries

Exception dictionaries make it possible to alter the way the synthesizer says any character or word. This is useful for correcting mispronounced words, or even speaking foreign languages. This section describes how to create exception dictionaries for the V860X.

The Text and Character modes of the V860X rely on a set of ROM-based English pronunciation rules for converting text sent from the host into speech. These rules determine which sounds, or phonemes, each character should receive. The position of each letter in a word, as well as its context, is considered by each rule. The V860X analyzes text in its input buffer by applying these rules to each word or character, depending on the translation mode in use. Exception dictionaries augment this process by defining exceptions (or even replacing) the ROM-based rules.

Exception dictionaries can be created and edited with word processors or editors that store documents as standard text (ASCII) files. However, the text file must be compiled into the internal format used by the V860X before it can be loaded. Development software is available from RC Systems to aid in the development of exception dictionaries, including a 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. Exceptions must always be terminated by a carriage return character.

The text fragment defines the characters that are to be translated by the exception, and may consist of any combination of letters, numbers, and symbols. The text fragment 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 can contain neither, either, or both contexts. There are also 13 special symbols, or *context tokens*, that can be used in an exception's context definitions. The tokens and their meanings are given in Table 9.

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: e, ed, er, ers, es, ely, ing, ings, ingly, ement, ements, eless, eness, able (must also be followed by a non-alphabetic character)
&	A sibilant: c, g, j, s, x, z, ch, sh
$	Any nonalphabetic character (includes numbers, spaces, etc.)
\	A digit (0-9)
|	One or more digits

Table 9. Context Tokens

The right side of an exception (P) specifies the pronunciation that the text fragment is to receive. The pronunciation consists of zero or more of the phonemes listed in Table 3. A space character must be used to delimit each phoneme when more than one is used in the pronunciation. If no pronunciation is given (no phonemes), no sound will be given to the text fragment, i.e., the text fragment will be silent.

One example of an exception is

```
C(O)N=AA
```

which states that o after c and before n gets the pronunciation AA, the a-sound in father. For example, the o in conference, economy, and icon would be pronounced according to this exception.

Another example is

```
$R(H)=
```

This exception states that h after initial r is silent, as occurs in the words rhinoceros and rhythm (the $ token represents any non-alphabetic character, such as a space between words; see Table 9).

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

```
(5)=S I NX K O                (Spanish five)
(+)=P L AH S                  (plus)
(CHR$)=K EH R EH K T ER       (Basic function)
```

Note that although the exceptions in the above examples do not contain any context definitions, parentheses are still used around the text fragments.

The Translation Algorithm

The synthesizer's translation algorithm works much like the human brain does when reading printed text. The algorithm scans input text from left to right and, for each character scanned, sequentially searches a list of pronunciation rules until it finds one that matches the text in the correct position and context. When a matching rule is found, the algorithm passes over the *input* characters bracketed in the rule (the text fragment), and assigns the pronunciation given by the right side of the rule to it. Scanning then resumes with the next character of text.

As an illustration of how the translation algorithm works, let's see how it would translate the word *receive*, using the ROM-based Text mode pronunciation rules.

The algorithm begins scanning at the letter *r* and searches the R pronunciation rules for a match. The first rule that matches is `$(RE)^#=R IH`, because the r in receive is an initial r and is followed by an e, a consonant, and a vowel. 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 fragment *re* as a whole gets the pronunciation `R IH`.)

The first match among the C rules is `(C)+=S`, because c is followed by e. The algorithm gives *c* the pronunciation `S`, and the scan continues with the second e: receive.

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

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

Rule precedence

Since the V860X uses its translation rules in a sequential manner, the relative position of each exception must be carefully considered. For example, consider the following 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 tone. The second exception does not place any restriction on what must come before or after o, so o in any context will match and receive the `UW` pronunciation. Note that if the exceptions were reversed, the `(O)+` exception would never be reached since 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 NX
(R)=R
```

This is an example of how *not* to organize exceptions. The exception for *rating* will never be used because *rat* will always match first. According to these exceptions, rating would be pronounced "rat-ing."

Generally, it is best 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. It doesn't matter how you organize the exceptions, so long as those with similar text fragments follow the sequential precedence outlined in the previous examples. However, grouping exceptions can aid in their development and gives an overall cleaner appearance.

Characters not matched by the exception dictionary

It is possible that some input characters will not match any of the exceptions, depending on the nature of the exception dictionary. For example, if a dictionary was written to handle unusual words, only those words would be defined by the exceptions. In this case, no other words would be translated by the exceptions. On the other hand, the dictionary might define the pronunciation for another language, in which case it would probably be comprehensive enough to encompass most all types of input. In any case, *if an exception is not found for a particular character, the English pro-*

nunciation will be given to that character according to the built in (ROM) English pronunciation rules.

Generally, the automatic switchover to the ROM rules is desirable if the dictionary is used to correct mispronounced words, since in essence it is defining exceptions to the ROM 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. To ensure the letter "a" will always be matched, for example, end the A exception group with the exception `(A)`=pronunciation. This technique works well to ensure matches only for specific characters, such as certain letters or numbers.

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

```
()=
```

This special exception causes unmatched characters to simply be ignored, rather than receive the pronunciation defined by the ROM rules.

Effect on punctuation

Punctuation defined in the exception dictionary has priority over the Punctuation Level command. Any punctuation defined in the dictionary will be used, regardless of the punctuation level selected. *Note:* If the dollar sign ($) character is defined in the text fragment of any exception, currency strings will not be read as dollars and cents.

Character mode exceptions

The discussion thus far applies to both the Text and Character modes of the synthesizer. Character mode exceptions, however, can be defined independently of the Text mode exceptions, or be included as a subset of them to avoid duplicating similar exceptions.

The beginning of the Character mode exceptions is defined by including the character `C` between the last Text exception and the first Character exception. No exceptions prior to this marker will be used when the synthesizer is in Character mode. For example:

```
.
.
.
()=
```

(Text mode exceptions)

(optional; used if following Character mode exceptions are not to be used in Text mode)

```
C
.
.
.
()=
```

(Character mode exceptions marker)

(Character mode exceptions)

(optional; used if ROM rules are not to be used in no-match situations)

Applications

The following examples were chosen to give you some ideas of how the exception dictionary can be used.

No cussing, please

The speaking of specific characters or words can be suppressed by writing "null" exceptions for them, that is, exceptions in which no pronunciation is given.

```
(????)=
```

(YOU fill in the blanks!)

When 0 is not zero

When we read addresses or lists of numbers, we often substitute the word "oh" 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
```

Arithmetic operators

Some characters may have more than one name; for example, / may be read as "slash" or "divided by," depending on the context. Such characters can be redefined if their default names don't fit the application. For example, the arithmetic operators (/, *, ^, etc.) can be defined for mathematical applications with the following exceptions:

```
(/)=D IH V AY D IH D   B AY
(*)=M AH L T IH P L AY D   B AY
(^)=R EY Z D   T UW
.
.
```

etc.

Correcting mispronounced words

Words that are mispronounced by the synthesizer can be corrected by writing exceptions for them.

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

The first exception corrects the pronunciation of all words containing *search* (search, searched, research,

etc.). As this exception exemplifies, it is only necessary to define the problem word in its root form. 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.

Acronyms and abbreviations

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

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

Heteronyms

Heteronyms are words that have similar spellings but are pronounced differently, depending on the context, such as read ("reed" and "red") and wind ("w-ih-nd" and "w-ah-ee-nd"). 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 "cloce," as in "close call." The synthesizer, however, will attempt to pronounce close as "cloze," as in "close the window." The following exception changes the way the *s* will sound:

```
(^DCLOSE)=K L OW S
```

Note the Control-D character (^D) in the text fragment. Although a non-printing character, the translation algorithms treat it as they would any printing character. Thus, the string "^D close" will be pronounced with its alternate pronunciation, "cloce," wherever it appears in the text stream. Plain "close" (without the Control-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.

Foreign languages

Dictionaries can be written which enable the synthesizer to speak in foreign languages. It's not as difficult as it may seem—all that is required is a high school-level foreign language textbook and a bit of patience. If you don't have a book for the language you're interested in, check your local library. Most libraries have foreign language books that include pronunciation guides, and perhaps even records or cassette tapes to go along with them. Such pronunciation guides make it easy to transcribe the pronunciation rules into dictionary form.

Language translation

Exception dictionaries will even allow the user to enter text in a foreign language and have it spoken in English. The following exceptions demonstrate how this can be done with three example Spanish/English words.

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

The sense of translation can also be reversed:

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

Message macros

Some applications may not be able to send text strings to a device such as the V860X. 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 could, however, assign an entire phrase or sentence to a single ASCII character with the exception dictionary. By driving four of the V860X's data bus pins and hardwiring the remaining four to Vcc or ground, virtually any set of 16 ASCII characters could be programmed.

For example, if you tied the four control bits from your device to data inputs D0 through D3, connected D4 and D5 to Vcc, D6 and D7 to ground, and your strobe to the V860X's WR pin, you could generate the ASCII digits 0 through 9 and the six ASCII characters following them (30h through 3Fh). You would then assign your message strings (in phonetic form) to each of these ASCII characters. For example, you could make ASCII "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 AA R T
ER
```

Due to the nature of this type of application, the nonvolatile memory feature of the V8601 would be required to permanently store the message strings. A PC could download the dictionary to the V8601 initially, through the parallel or serial ports.

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.

Exception anomalies

On rare occasions, an exception may not work as expected. This occurs when the ROM-based 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 V860X simply refuses to take on the new Canadian accent.

It so happens the V860X has a rule in its ROM 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 ROM 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)=P R OW would work).

CIRCUIT CELLAR, INC. MERCHANDISE RETURN FORM

IN THE EVENT THAT YOU NEED TO RETURN AN ITEM TO US, PLEASE TAKE THE FOLLOWING STEPS:

1. Call CCI and obtain an RMA number.

2. Complete the form below and return it with the merchandise. Include a copy of your original invoice.

3. Should you need a replacement item shipped immediately, contact a CCI customer service rep. at **(203) 875-2751.**

4. Ship the merchandise (pre-paid) to the address below. Using the original packaging, safely repack with all original accessories included.

PLEASE COMPLETE THIS FORM AND RETURN WITH THE MERCHANDISE AND A COPY OF THE ORIGINAL INVOICE:

RMA# _____

Date _____ Invoice # _____ P.O.# _____ Daytime Phone _____

Customer # _____ Account Name _____

Contact Name _____

METHOD OF PAYMENT:

☐ Check ☐ Charge ☐ Company Purchase Order (please attach copy)

Credit Card # _____ Expiration Date _____

Signature of Cardholder _____ Purchase Order # _____

ITEM(S) RETURNED	**REASON(S)**

(Use additional paper to be as explicit as possible on your reason(s) for return)

What action would you like us to take? (Repair or Replace item, Refund or Credit your account):

Is Shipping Data on Invoice correct? Yes _____ No _____

If not, please provide correct information: _____

To help us serve you better, use this space for your comments regarding Circuit Cellar Inc.'s service, support, product quality, etc. Thank you.

Ship to: CIRCUIT CELLAR, INC. • 4 PARK STREET • VERNON, CT 06066

CONDITIONS OF SALE AND PRODUCT WARRANTY

Circuit Cellar Inc. and the Buyer agree to the following terms and conditions of Sale and Purchase:

1. Circuit Cellar Inc. extends the following warranty: a factory manufactured circuit board or assembly carries with it a one-year warranty covering both parts and labor. Any unit which is found to have a defect in materials or workmanship will, at the discretion of Circuit Cellar Inc., be repaired or replaced.

2. A minimum inspection fee must be prepaid for the repair of units that are no longer under warranty. Call Circuit Cellar Inc. for a current list of fees.

3. NO WARRANTY is extended on USER ASSEMBLED systems or kits. However, assembled kits will be inspected and repaired with charges based on the current minimum one hour charge. Circuit Cellar Inc. retains the right to refuse to repair any USER ASSEMBLED item. This right is at the sole discretion of Circuit Cellar Inc. However, in the event that repair charges would exceed a reasonable amount, the user may be consulted for a determination. Repairs on user assembled items must be POSTPAID. Return authorzation must be obtained prior to any return.

4. Circuit Cellar Inc. will not be responsible for the repair or replacement of any unit damaged by user modification, negligence, abuse, mishandling, or improper installation.

5. Circuit Cellar Inc. is not responsible to the Buyer for any, or claim of, special or consequential damages.

6. All units returned for repair must first receive prior authorization from Circuit Cellar Inc. A return authorization number may be obtained by phone or letter. Please retain a record of this number, since most subsequent correspondence will refer to this authorization. Under no circumstances should any product be returned to Circuit Cellar Inc. without such authorization, and Circuit Cellar Inc. assumes no responsibility for returns unaccompanied by an authorization number. All returns must be shipped postpaid and ought to be insured. Circuit Cellar Inc. is not responsible for losses or damage during shipment. Repaired units will be returned postage and insurance paid.

7. Circuit Cellar Inc. reserves the right to alter any feature or specification at any time. This right extends to fees, charges, and any other conditions or warranties contained herein.