Speech Handler,
External Reference Specification,
1983-07-22 Version, Revision 2

The document itself begins on the next page.

Document source:

Original backup tapes owned by Dutchman2000, obtained by Atarimania.
Documentary research and PDF layout by Laurent Delsarte.
Note that these backup tapes contain A LOT of information spread out in many folders, meaning it will take time to process the important bits.

Document identification:

Original file name:	EMAIL extracted from CEO.01JUN84
Title of document:	Speech Handler, External Reference Specification, 1983-07-22 Version, Revision 2
Author(s):	Harry B. Stewart & Brad Fuller
Original file date:	1983-07-22
Type of document:	Memo
Target audience:	Internal
Status:	Work in progress
Reference (Atari):	Product Number T1808
Reference (Laurent Delsarte):	For any discussion, this PDF has been given the reference **BKUP-1983-07-22-MEMO-0017A-1** which should be quoted in any communication.
Tags:	#Atari #8bit #6502 #1400XL #1450XLD #T1808 #SpeechHandler #Votrax #SC01 #SC02

Comments:

This document seems very comprehensive, except that we know it isn't the final version: others will follow.

It describes the specifications of a speech handler: a new feature enabling the forthcoming Atari 1400XL and 1450XLD to produce synthetic human speech.

ERS stands for External Reference Specification.

Obvious spelling mistakes and typos have been corrected.
Additions by Laurent Delsarte are indicated by [and].

This page intentionally left blank

Speech Handler,
External Reference Specification,
1983-07-22 Version, Revision 2

Revision Two
July 22, 1983
Product Number T1808
Harry Stewart and Brad Fuller

APPROVED BY:

Software Development	DATE
Marketing Product Mgr	DATE
Product Test Mgr	DATE
1400 CPU Development Mgr	DATE
Software QA Mgr	DATE
Supervisor Technical Pub.	DATE

Table of Contents

1. Purpose

1.1 Introduction

The Speech Handler shall be an ATARI compatible I/O handler that shall be designed under the criteria described in all applicable documents (see section 2.0). It shall be designed to interface with the VOTRAX SC01 speech chip for the ATARI 1400XL and 1450XLD home computers.

There are several levels of speech representation which shall be supported to drive the SC01 speech device:

Representation	Explanation
`'heloe'`	World English Spelling of "hello".
`'H EH1 EH2 L O1 PA0'`	Votrax symbolic phoneme representation of "hello".
`1B 02 01 18 35 03`	Votrax numeric phoneme representation of "hello", in hexadecimal.

In order to allow for upward mobility of application and user software to other speech chips which might be utilized in Atari's future products, all speech data will be identified as being of a specific (device dependent) type and form. Future products will be able to accept data of their native type as well as translate data of older types to the closest equivalent sound in their new hardware.

The three forms supported for the SC01 are called

- '1P' (type #1, phonetic form),
- '1S' (type #1, symbolic form)
- '1N' (type #1, numeric form)

Future type/forms for the SC02 (or any other device) might be called '2P', '2P', '2S' and '2N'. Perhaps even a '2T' (type #2, text form) might be supported.

The handler shall also support both upper-case and lower-case letters in Phonetic and Symbolic translation.

1.2 Consumer Profile

The Speech Handler shall be a user transparent handler similar to existing I/O handlers.

1.3 Interface to Other Products

The Speech Handler is not compatible with any existing ATARI software except to interface with the SURELY OS

1.4 Family of Products

The Speech Handler shall be in the SYSTEMS category of ATARI home computer products.

2. Applicable documents

- SURELY External Reference Specification,
  Revision 2, 8-Apr-1983

- The Software Implementation of Parallel Handlers and Drivers,
  Draft, 30-Mar-1983

3. Requirements

3.1 Interfaces

3.1.1 Initialization

The Speech Handler initialization routine will:

1. Set device mask (PDVMSK) and clear IRQ mask (PDIMSK)
2. Set handler table (HATABS)
3. Send a STOP phoneme to the SC01
4. Check for Self-Test mode

   The self-test shall be invoked when:
   1. COLDSTART is active
   2. OPTION key is pressed
   3. NO disk is to be booted

3.1.2 Handler/CIO interface

The device name is 'V:'

The BUS I.D. = $60

3.1.2.1 Open

The IOCB buffer pointer shall point to a sequence of ATASCII characters of the form shown.

V<type>:<form><mode><EOL>

Where:

Non terminal symbol	Possible values
<type> =	'1'
<form> =	'P' for World English Spelling phonetic form 'S' for Votrax symbolic form, 'N' for Votrax numeric form, 'U' for user specified symbolic form.
<mode> =	'D' for direct mode, 'S' for semi-buffered mode, 'F' for fully buffered mode.

In response to an OPEN command, the handler initializes the output FIFO, resets all handler database variables and flags, initializes the translate table pointer, and sends a STOP phoneme to the SC01. If the SC01 does not respond with a READY status within 100 milliseconds, an error status is returned.

If the form is 'U', the user is responsible for providing the address of his translate table by issuing a SPECIAL command after the OPEN. By using a special sequence using PUT-BYTE.

If not specified, the type, form and mode default shall be as shown below.

Non terminal symbol	Default values
<type> =	'1'
<form> =	'N'
<mode> =	'D'

[[So, **V1:ND** ?]]

Error conditions:

Value[1]	Description
$8A	Votrax not responding.
$84	Invalid form (not equal to 'P', 'N', 'S', or 'U').
$84	Invalid mode (not equal to 'D', 'S', or 'F').

3.1.2.2 Close

In response to a CLOSE command, the handler sends a STOP phoneme to the SC01.

Error conditions: none.

[1] Is this $84 value used for both invalid form or mode? Is this a typo? The same table appears in a later version – 14 October 1983 – with the same values.

The handler accepts a single byte (character) of phonetic form, symbolic form, user form or numeric form data. The data is in the A register; the handler returns status in the Y register. The table below shows the processing that shall occur for each byte, depending upon the form and mode.

Form	Mode	Process
N	**D**	The phoneme is output to the SC01, and the handler waits for the next SC01 request before returning.
	S	The handler waits for the SC01 to request a byte, outputs the phoneme to the SC01 and then returns.
	F	The byte goes to the output FIFO. If the FIFO is full, the handler loops waiting for an empty slot.
P,S,U	**D**	A token is assembled, translated to the proper numeric code, and then processed as in N-D above.
	S	A token is assembled, translated to the proper numeric code, and then processed as in N-S above.
	F	A token is assembled, translated to the proper numeric code, and then that phoneme is inserted to the output FIFO.

When operating in the fully buffered mode, the handler shall enable the SC01 interrupt, so that the handler's interrupt service routine can empty the FIFO. Put-Byte shall be responsible for incrementing the phoneme counter and the marker counter in direct and semi-buffered mode.

Error conditions: $84 Unrecognizable text token.

3.1.2.3b Put-Byte (Special function)

SPECIAL FUNCTION is implemented thru PUT-BYTE.

A "Special Sequence" can set:
1. User defined FIFO buffer location and size.
2. User defined translation routine location and translation table location.
3. User code appended to handler IRQ service routine.
4. End a special sequence.
5. Indicate to handler to stop processing phonemes in fully-buffered mode.

To set-up a Special Sequence the user should do the following:
1. Store a non-zero value in CMCMD ($07).
2. Send escape character ($1B) with Put-Byte.
3. Send one of the following command directives with Put-Byte.
   'A' To indicate that a new buffer size follows. (1-byte FIFO size)
   'B' To indicate that a new buffer location follows. (2-byte address)
   'C' To indicate that a translation routine location follows. (2-byte address)
   'D' To indicate that a translation table location follows. (2-byte address).
   'E' To indicate END OF SPECIAL SEQUENCE.
   'F' To indicate that the location of users sync code follows. (2-byte address).
   'G' To indicate to the handler to ABORT emptying the buffer in fully-buffered mode.

Example:
```
CMCMD <> 0
PUT-BYTE-> $1B, 'B', 00, 06
```

Indicates: will set buffer location to hexadecimal 600
2-Byte addresses are LOW followed by HIGH.

3.1.2.4 Get-Byte

Get-Byte shall return a "Function not implemented" error code in the Y register and return to the OS.

3.1.2.5 Special

Special shall return a "Function not implemented" error code in the Y register and return to the OS.

3.1.2.6 Interrupt Service Routine

This routine shall empty the FIFO buffer one character at any time. Upon an IRQ the service routine shall:

1. Obtain a translated phoneme from FIFO buffer.
2. Output phoneme to SC01.

If buffer is empty the service routine shall disable SC01 IRQ's. See 3.7.3

In fully buffered mode the IRQ service routine shall be responsible for incrementing the marker counter and the phoneme counter.

3.1.2.7 Status

The phoneme counter and the marker counter shall be passed as the first (DVSTAT) and second (DVSTAT+1) variables upon a status call respectively. See 3.7.2

3.1.2.8 Low Level I/O

The Low-Level routine shall clear the carry bit and return to the OS.

This is so future Vx: devices can answer the Low-Level call.

3.2 Functional description

3.2.1 Handler functionality

The handler shall be able to output speech data (phonemes) when presented with data in any of three user formats: phonetic, symbolic or numeric. In the phonetic and symbolic formats, a phoneme is represented by one to three ATASCII characters followed by a delimiter character; the handler converts each symbolic phoneme to one or more six-bit numeric phoneme codes. In the numeric format, a phoneme is comprised of the lower six bits of an eight-bit byte, which is assumed to contain the numeric phoneme code.

In addition, the handler shall have three output modes: direct, semi-buffered and fully-buffered. The output characteristics of the three modes are described below.

1. **Direct output phoneme**:
   Wait for SC01 READY, then return to caller.
2. **Semi-buffered**:
   Wait for prior phoneme done, output new phoneme, then return to caller.
3. **Fully-buffered**:
   Output all phonemes at interrupt level.

The caller may determine the completion of a phoneme string or synchronize himself to specific phonemes by any of several techniques, as listed below.

- **Phoneme counter in database**. The handler shall maintain a one-byte counter in the PBI database area and in location VPCTR($3ED) which shall be incremented as each phoneme is strobed, into the SC01. This variable may be examined and/or altered by the caller at will. The phoneme counter variable shall also be passed as the first variable on a status call (DVSTAT).
- **FIFO control in database**. There shall be a FIFO associated with the handler which shall be used when phoneme output is to be fully buffered. This FIFO and its control elements are resident in the PBI database area (starting at location $D6E0) and may be examined (but not altered) by the caller at will.
- **Markers**. There shall be one or more codes reserved for "markers". When a marker is about to be processed as phoneme data, the handler shall increment a marker variable in the PBI database area, in location VMCTR($3EE) and produce a null phoneme (zero duration). The marker counter variable shall also be passed as the second variable in a status call (DVSTAT+1).
  The user may append synchronization code to the handler's IRQ service routine through one of the special sequences. It is the responsibility of the user to control interrupt timing.

Which technique to use is a function of:
1. The output mode selected,
2. How well the caller's records map to the synchronization points, and
3. Whether the handler is called directly by the user or is invoked through CIO.

3.2.2.1 World English Spelling

In the World English Spelling format, "hello" is shown encoded in WES format, which requires 6 characters (see 4.1 for more information).

```
+-----------+
|H E L O E  |
+-----------+
```

[[Note that the 6th character is a space character]]

For the P form, the following characters shall be treated as word delimiters and produce pauses:

This character	Produces
`' '`	space produces a short pause.
`','`	comma produces a short pause (intermediate when followed by a space).
`'.'`	period produces a long pause.
`'?'`	question mark produces a long pause.
ATASCII EOL	produces a short pause.

For the P form, the hyphen is a token delimiter which allows the handler to unambiguously process the multi-character tokens. For example, the word 'mishap' would be spelled 'mis-hap'.

The asterisk character shall be treated as a marker by the handler.

3.2.2.2 Symbolic

In the Symbolic format, "hello" is shown encoded in Votrax symbolic format, which requires 18 characters (see 4.2 for more information).

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|H   E H 1   E H 2   L   0 1   P A 0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

For the S form, the following character shall be treated as token delimiters and shall not produce or alter phonemes:

This character	
`' '`	Space
`','`	Comma
`'.'`	Period
`'?'`	Question mark
`'-'`	Hyphen

The asterisk character is treated as a marker by the handler.

The ATASCII EOL produces a short pause.

3.2.2.3 Numeric

In the Numeric format, "hello" is shown encoded in Votrax numeric format, which requires 6 bytes.

```
+--+--+--+--+--+--+
|1B 02 01 18 35 03|
+--+--+--+--+--+--+
```

All bytes received by the handler shall be truncated to 6-bit phoneme values and sent to the SC01, with the following two exceptions:

- $9B is an EOL and shall be ignored by the handler (null operation).
- $7F is the code for a marker.

Note that any byte, including those above $3F, shall be treated as phonemes and shall be truncated to 6 bits (except for the 2 exceptions noted above).

3.2.3 Implementation Details

FIFO related items:

- If the FIFO fills in mid-record, the handler shall wait until there is room in the FIFO for the next phoneme.
- FIFO size shall be limited to 255 items (phonemes and markers). The default FIFO shall contain up to 32 items.

Translate table related items (see 3.2.4 for more information):

Each translate table shall be limited to 256 bytes.

RAM utilization:

Phoneme counter [1].
Marker flag [1].

FIFO input index [1].
FIFO output index [1].
FIFO size [1].
FIFO base pointer [2].
FIFO phoneme counter [1].

Default FIFO buffer [32].

Current data type [1].
Current data form [1].
Current output mode [1].

Translate table base pointer [2].
Translate table offset [1].
Translate routine base pointer [2].

Two 2-byte temporaries [4].
Two 1-byte temporary [2].
One 1-byte flag for special [1].
One 2-byte user sync vector [2].

3.2.4 Translate Table Format

The speech translate table shall be a state table that allows the translator to be implemented as a finite state machine (FSM). The advantages of this approach are twofold: 1) the table is very compact, and 2) the FMS requires character storage for only one character at a time, rather than the characters for one complete token at a time.

The translate table shall consist of a collection of multiple byte entries. Each entry shall consist of a match character followed by a match directive which shall be executed if the input character matches the match character. In general, the match directive shall advance the FSM to a new state, and may in addition produce one or more output phonemes.

The formats for the match byte and directive byte shall be as shown below.

Match byte

```
+-+-+-+-+-+-+-+-+
|0|  match char |   match character
+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+
|1|   xxx       |   NIL
+-+-+-+-+-+-+-+-+
```

Directive byte:

```
+-+-+-+-+-+-+-+-+
|0 0|  phoneme  |   SC01 phoneme code
+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+
|0 1 0|    n    |   n phonemes follow
+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+
|0 1 1|  code   |   special action code
+-+-+-+-+-+-+-+-+   (see Note 1, below)

+-+-+-+-+-+-+-+-+
|1|    offset   |   offset to next state
+-+-+-+-+-+-+-+-+   (see Note 2, below)
```

Note 1 – Special actions shall include the following:

delimiter (ignore). = $60+2
generate a marker. = $60+1
error (invalid token). = $60+0

Note 2 – The offset is used to update the translate table index as follows:

index = (index + offset) mod 256

This implies the following:

The maximum table size shall be 256 bytes.

All state transitions shall be in the forward direction, except for the transition to the top-level state which shall be implicit in the specification of the FSM.

A state transition destination must be within 127 bytes of the source directive.

The FSM scanner operates with the input data as described below.

1. The translate table index is set to the beginning of the translate table (=0).
2. A new character is obtained.
3. The scanner scans the table linearly trying to find a match between the token character and one of the table entry match characters before a NIL entry is seen.
4. If a match is found, the scanner processes the match directive and does one of the following actions, based upon the type of directive.
   a. If the directive is a phoneme or multiple phonemes, the phoneme(s) are output and scanning proceeds at step 1.
   b. If the directive is a special action, the specified action is taken and scanning proceeds at step 1.
   c. If the directive is a table offset (new state), the table index is updated as specified and scanning proceeds at step 2.
5. If a match is not found (NIL found first), the scanner processes the match directive associated with the NIL and does one of the following actions, based upon the type of directive.
   a. If the directive is a phoneme or multiple phonemes, the phoneme(s) are output and scanning proceeds at step 6.
   b. If the directive is a special action, the specified action is taken and scanning proceeds at step 6.
   c. If the directive is a table offset (new state), the table index is updated as specified and scanning proceeds at step 2.
6. The table pointer is set to the beginning of the translate table (this is the same as step 1).
7. Scanning proceeds at step 3, using the character that produced the NIL match.

The translate table for the SC01 Symbolic Format is shown in 4.3 and the translate table for the World English Spelling Format is shown in 4.4.

3.3 Performance requirements

[[Empty]]

3.4 Design requirements

These items represent design requirements for the hardware interface.

1. A VOTRAX SC01 speech chip shall be utilized.
2. One phoneme (6 bits) of external buffering is provided. This latch can be accessed by writing to addresses $D104 to $D107. This causes the contents of the data bus to be latched as an output to the SC01.
3. The computer has direct control of the SC01 STB line. This strobe is accessed at addresses $D100 TO $D103. A write to any of these addresses causes a strobe to be sent to the SC01 that indicates that valid data is present in the phoneme selection latch.
4. A status bit and disable-able IRQ interrupt is associated with the SC01 A/R line. Bit 7 of the data latch acts as the IRQ interrupt enable/disable switch. Bit 7 cleared indicates IRQ's are disabled. Bit 7 set indicates IRQ's are enabled. Bit 7 of address $D1FF represents the A/R line. Bit 7 set indicates the SC01 is processing a phoneme and Bit 7 cleared indicates the SC01 is ready.

There shall be no pitch/inflection control.

There shall be no volume control.

3.5 Packaging requirements

The speech hardware shall be designed for internal construction within the 1400XL and the 1450XLD computers. See 3.4.

3.6 Special requirements

[[Empty]]

3.7 Helpful hints

3.7.1 User Accessible State Variables

`VMCTR, location - $3EE (marker counter):`
The user shall be able to track his position in a V: string (a series of phonemes or tokens to be processed by the V: handler) by inserting MARKERS within the V: string. Every time the handler processes a marker the handler shall increment VMCTR and shall produce a null operation (see 3.2.1).

`VPCTR, location - $3ED (phoneme counter):`
The user shall be able to track the number of phonemes that have been processed by accessing the variable VPCTR. This variable will reflect ALL phonemes that have been processed including EOL characters when in the Symbolic or Phonetic forms (see 3.2.1)

3.7.2 Status calls

A CIO status call shall make available the phoneme counter and the marker counter in DVSTAT and DVSTAT+1 respectively (see 3.2.1).

3.7.3 User appended IRQ code

Please note that when a user appends his synchronization code to the handler's IRQ service routine that all processes will channel through the user's appended code. The handler will not shut down IRQ's until the buffer is empty, which is one more handler access to shut down IRQ's then phonemes to be processed.

This could be particularly puzzling if the user is relying completely on IRQ's to synchronize with since the last IRQ is only to shut down IRQ's and will not produce a phoneme. It is recommended that the user use VMCTR and VPCTR in addition to appending his synchronization code to the handlers IRQ service routine.

3.7.4 Errors

Please note that upon an 'invalid token error' the VOTRAX will continue to sound the last valid token. This could provide an easy debugging procedure for invalid tokens. For instance, when an error occurs, the user should establish what phoneme the VOTRAX is producing and find that phoneme in the V: string. The next phoneme in the V: string is invalid.

4. Tables

4.1 World English Spelling Form

Speech Token	Sound	Votrax Equivalent
0	ZERO	Z/12+I2/0A+R/2B+0/26
1	ONE	W/2D+UH1/32+N/0D
2	TWO	T/2A+U/28
3	THREE	TH/39+R/2B+E/2C
4	FOUR	F/1D+O2/34+R/2B
5	FIVE	F/1D+AH1/15+EH3/00+Y/29+V/0F
6	SIX	S/1F+I1/0B+K/19+S/1F
7	SEVEN	S/1F+EH1/02+V/0F+EH2/01+N/0D
8	EIGHT	A/20+Y1/22+T/2A
9	NINE	N/0D+AH1/15+EH3/00+Y/29+N/0D
a	fAt	AE /2E
aa	fAther	AH1/15
ae	pAy	A /20 + Y/29
ar	fAR	AW2/30 + AH2/08 + R/2B
au	tAUt	AW /3D
b	But	B /0E
ch	CHum	T /2A + CH /10
d	Dig	D /1E
e	sEt	EH3/00
ee	sEE	E/2C
er	gathER	ER /3A
f	Fat	F /1D
g	Gum	G /1C
h	Hat	H /1B
i	In	I /27
ie	tIE	AH2/08 + EH3/00 + Y/29
j	Jam	D /1E + J /1A
k	Kit	K /19
l	Let	L /18
m	Met	M /0C
n	Net	N /0D
ng	siNG	NG /14
nk	siNK	NG /14 + K/19
o	On	AW /30 + UH3/23
oe	tOE	O /26
oi	bOY	O1 /35 + UH3/23 + Y/29
oo	tOO	U /28

Speech Token	Sound	Votrax Equivalent
or	fOR	O2 /34 + R/2B
ou	OUt	AH2/08 + UH3/23 + U1/37
p	Pet	P /25
r	Run	R /2B
s	Set	S /1F
sh	SHed	SH /11
t	Tin	T /2A
th	THis	THV/38
thh	THing	TH /39
u	Up	UH1/32
ue	hUE	Y /29 + U/28
ur	fUR	ER /3A + R/2B
uu	bOOk	00 /17
v	Van	V /0F
w	Win	W /2D
wh	WHen	W /2D + EH2/01
y	Yes	Y1 /22
z	Zoo	Z /12
zh	viSion	ZH /07

4.2 Votrax SC01 Symbolic Form

Speech Token	Sound	Votrax Numeric Equivalent
EH3	jackEt	0
EH2	Enlist	1
EH1	hEAvy	2
PA0	<pause>	3
DT	buTTer	4
A2	mAde	5
A1	mAde	6
ZH	aZure	7
AH2	hOnest	8
I3	inhibIt	9
I2	Inhibit	0A
I1	inhIbit	0B
M	Mat	0C
N	suN	0D
B	Bag	0E
V	Van	0F
CH	CHip	10
SH	SHop	11
Z	Zoo	12
AW1	1AWful	13
NG	thiNG	14
AH1	fAther	15
OO1	lOOking	16
OO	bOOk	17
L	Land	18
K	triCK	19
J	JuDGe	1A
H	Hello	1B
G	Get	1C
F	Fast	1D
D	paiD	1E
S	paSS	1F
A	dAy	20
AY	daY	21
Y1	Yard	22
UH3	missIOn	23
AH	mOp	24
P	Past	25
O	cOld	26
I	pIn	27
U	mOve	28

Speech Token	Sound	Votrax Numeric Equivalent
Y	anY	29
T	Tap	2A
R	Red	2B
E	mEEt	2C
W	Win	2D
AE	dAd	2E
AE1	After	2F
AW2	sAlty	30
UH2	About	31
UH1	Uncle	32
UH	cUp	33
O2	fOr	34
O1	abOArd	35
IU	yOU	36
U1	yOU	37
THV	THe	38
TH	THin	39
ER	bIRd	3A
EH	gEt	3B
E1	bE	3C
AW	cAll	3D
PA1	<pause>	3E
STOP	<stop>	3F

4.3 Translate Table for SC01 Symbolic Form

Label (State)	Match char	Next State	Directive: Special action	Phoneme Code	
Start	'A'	A1			
	'B'			B	0E
	'C'	Cx			
	'D'	Dx			
	'E'	Ex			
	'F'			F	1D
	'G'			G	1C
	'H'			H	1B
	'I'	Ix			
	'J'			J	1A
	'K'			K	19
	'L'			L	18
	'M'			M	0C
	'N'	Nx			
	'O'	Ox			
	'P'	Px			
	'R'			R	2B
	'S'	Sx			
	'T'	Tx			
	'U'	Ux			
	'V'			V	0F
	'W'			W	2D
	'Y'	Yx			
	'Z'	Zx			
	EOL				3
	' '		<delim>2		
	','		<delim>2		
	'.'		<delim>2		
	'?'		<delim>2		
	'-"		<delim>2		
	'*'		<marker>1		
	NIL		<error>0		
Ax	'E'	AEx			
	'H'	AHx			
	'W'	AWx			
	"Y'			AY	21
	'1'			A1	6
	'2'			A2	5
	NIL			A	20
Cx	'H'			CH	10

Label (State)	Match char	Next State	Directive: Special action	Phoneme Code	
	NIL		<error>0		
Dx	'T'			DT	4
	NIL			D	1E
Ex	'H'	EHx			
	'R'			ER	3A
	'1'			E1	3C
	NIL			E	2C
Ix	'U'			IU	36
	'1'			I1	0B
	'2'			I2	0A
	'3'			I3	9
	NIL			I	27
Nx	'G'			NG	14
	NIL			B	0D
Ox	'O'	OOx			
	'1'			O1	35
	'2'			O2	34
	NIL			O	26
Px	'A'	PAx			
	NIL			P	25
Sx	'H'			SH	11
	'T'	STX			
	NIL			S	1F
Tx	'H'	THx			
	NIL			T	2A
Ux	'H'	UHx			
	'1'			U1	37
	NIL			U	28
Yx	'1'			Y1	22
	NIL			Y	29
Zx	'H'			ZH	7
	NIL			Z	12
AEx	'1'			AE1	2F
	NIL			AE	2E
AHx	'1'			AH1	15
	'2'			AH2	8
	NIL			AH	24
AWx	'1'			AW1	13
	'2'			AW2	30
	NIL			AW	3D
EHx	'1'			EH1	2

Label (State)	Match char	Next State	Directive: Special action	Phoneme Code	
	'2'			EH2	1
	'3'			EH3	0
	NIL			EH	3B
COx	'1'			1	16
	NIL			0	17
PAx	'0'			PA0	3
	'1'			PA1	3E
	NIL		<error>0		
STx	'O'	STOx			
	NIL		<error>0		
THx	'V'			THV	38
	NIL			TH	39
UHx	'1'			UH1	32
	'2'			UH2	31
	'3'			UH3	23
	NIL			UH	33
STOx	'P'			STOP	3F
	NIL		<error>0		

4.4 Translate Table for World English Spelling Form

Label (State)	Match char	Next state	Directive: Special action	Phoneme code
Start	'0'			Z+I2+R+O
	'1'			W+UH1+N
	'2'			T+U
	'3'			TH+R+E
	'4'			F+O2+R
	'5'			F+AH1+EH3+Y+V
	'6'			S+I1+K+S
	'7'			S+EH1+V+EH2+N
	'8'			A+Y1+T
	'9'			N+AH1+EH3+Y+N
	'A'	Ax		
	'B'			B
	'C'	Cx		
	'D'			D
	'E'	Ex		
	'F'			F
	'G'			G
	'H'			H
	'I'	Ix		
	'J'			J
	'K'			K
	'L'			L
	'M'			M
	'N'	Nx		
	'O'	Ox		
	'P'			P
	'R'			R
	'S'	Sx		
	'T'	Tx		
	'U'	Ux		
	'V'			V
	'W'	Wx		
	'Y'			Y1
	'Z'	Zx		
	EOL			PA0
	' '			PA0
	','			PA0
	'.'			PA1
	'?'			PA1
	'-'		<delim>	
	'*'		<marker>	

Label (State)	Match char	Next state	Directive: Special action	Phoneme code
	NIL		<error>	
Ax	'A'			AH1
	'E'			A+Y
	'R'			AW2+AH2+R
	'U'			AW
	NIL			AE
Cx	'H'			T+CH
	NIL		<error>	
Ex	'E'			E
	'R'			ER
	NIL			EH3
Ix	'E'			AH2+EH3+Y
	NIL			I
Nx	'G'			NG
	'K'			NG+K
	NIL			N
Ox	'E'			O
	'I'			O+I2
	'O'			U
	'R'			O2+R
	'U'			AH2+UH3+U1
	NIL			AW+UH3
Sx	'H'			SH
	NIL			S
Tx	'H'	THx		
	NIL	T		
Ux	'E'			Y+U
	'R'			ER
	'U'			OO
	NIL			UH1
Wx	'H'			W+EH2
	NIL			W
Zx	'H'			ZH
	NIL			Z
THx	'H'			THV
	NIL			TH