TSI S14001A Speech Synthesizer
LSI Integrated Circuit Guide
Version 0.89b

By Jonathan Gevaryahu AKA Lord Nightmare,
with a HUGE amount of help from Kevin "kevtris" Horton.
E-mail: jgevaryahu @t hotmail d0t com
Alternately: lord_nightmare @t users d0t sf d0t net

Overview:

The TSI S14001A... Researching this chip was like trying to batter down a brick wall with your head, at first. No datasheet, no known patent (initially), but demand for emulation (and possibly FPGA or PIC re-implementation to repair arcade and pinball boards). The chip is mentioned in "Talking calculator [for the blind] incorporates 1-chip mu C plus custom microcontroller" -- Source: EDN, v 21, n 11, 5 June 1976, p 35-7

I found a copy of the EDN article at my university library, and as it turns out, the most important piece of information was in that article: The name of the person who designed the compression and synthesis scheme used in the chip. That person was Forrest S. Mozer, a physics professor at Berkeley who, as it turns out, also designed the compression scheme used in the Digitalker chip from National Semiconductor, the compression used on several 'SSI' C64 games such as Ghostbusters, and the compression schemes used by Sensory Inc. on their chips. (Sensory Inc. was created by Mozer's Sons, basically to market his technology!) Once I knew this, I compared the ROMs from each, and indeed they all do seem to use a very similar format. (I haven't checked Ghostbusters yet though.) But MOST importantly, it gave me a patent to look at to learn about the device: US Patent 4,214,125 Also, patents 4,433,434 and 4,435,831 contain additional useful data which helped with figuring out the mirroring method used in the delta demodulator.

History:

The TSI S14001A was developed by TeleSensory, Inc. in 1975 as a single-IC speech chip for a portable talking calculator for the blind, called the Speech+. The speech technology was licensed (I believe with a 3 year exclusive deal) from Forrest S. Mozer, a professor of atomic physics (speech was a spare time thing for him) at Berkeley. Forrest Mozer would encode the speech in his basement laboratory using his novel form of speech encoding (the encoding process apparently involved several minicomputers running FFTs and a spectrum analyzer), and then General Instruments would make the resulting speech data into a mask ROM to be used with the TSI chip. In 1978, Dr. Mozer made another exclusive 3-year license: to National Semiconductor, who used his design in their Digitalker, Digitalker II, and Microtalker ICs, the latter two of which apparently never saw the light of day. Later, in 1983 or 1984, after his sons first company, SSI, went bankrupt, they started a second company called "Sensory Circuits Inc." , later shortened to "Sensory Inc.", which used a slightly modified form of the Digitalker compression in their products, and called it "MX". The TSI S14001A was used in SIX products, as far as I'm aware so far: (Please feel free to point out any more you may discover)

- [1976] The TeleSensory Inc. "Speech+" Talking calculator (which it was originally designed for)
- [1978 or 1979] Atari's unreleased prototype 'Wolf Pack' arcade machine (Thanks to Stefan Jokisch for pointing this one out)
- [1979] The Fidelity Electronics TALKING Chess Challenger (NOT the plain Chess Challenger, the logo being separate on the Talking one and not on the normal one) (Thanks to Kevin Horton for pointing this one out)
- [1980] The Canon "Canola SP1260" adding machine (both US and GERMAN versions are known to exist, with different speech ROMs)
- [1979-1982] The Stern VSU-100 speech board, which was used in six stern pinball machines: Flight 2000, Catacomb, Freefall, Lightning, Orbitor-1, and Split Second, with different ROMs in each case. (It also may have been used in the Prototype 'Cue' machine which never made it to full release)
- [1980-1981] The Stern VSU-1000 speech board, which was used in the arcade games 'Berzerk' and 'Frenzy', and the unreleased prototype of 'Moon War' (but NOT the final version of 'Moon War', which was on Konami 'Scramble' hardware).

I'm looking for copies of the speech ROMs from the two SP1260 versions and the 'Cue' machine (if it used an S14001A), contact me if you have one of these! I did learn some fascinating and somewhat useless footnotes to history about the interaction of TSI and MITalk, Kurzweil, and how TSI eventually split into TSI and Speech Plus, then seemed to re-merge with itself later. Maybe I can write a doc on the other TSI speech 'chip', the mutilated-MITalk-based Prose 2000 series, which was used in a number of other TSI 'unlimited text to speech' speech synthesizers. Based on the fact that both the Canon instruction manual, the Stern schematics, the silkscreen on the Stern VSU-1000 PCB, and a few patents on devices which probably never saw the light of day, I conclude that the original datasheet probably called it a 'Custom ROM Controller' or 'CRC chip'. Which is pretty much what it was. Go figure.

I STILL don't have a copy of the original datasheet, so if anyone finds a copy, please, PLEASE scan it and send it to me!

Pinouts:

Based on some very good scans of the Canon adding machine board, and the Stern VSU-1000 schematics, I was able to work out a mostly complete pinout of the chip. The TSI S14001A is a 40Pin DIP IC. Its an ASIC, and has a 4-bit DAC built in. The bottom of the chip on the VSU-1000 PCB I own has "Philippines" and "S170X" on it.

Here's the pinout, as far as I can tell. It looks rather haphazard compared to most 'modern' chips: IDx are the word input lines Ax are the word ROM address lines Dx are the word ROM data lines /BUSY is low while a word is being played/said START is pulled high when a word is to be said and the word number is on the input lines The Canon 'Canola' uses a separate 'ROM strobe' signal independent of the chip to either enable or clock the speech ROM. Its likely that they did this to force the speech chip to stop talking, which is normally impossible.

1	+5VDC	**40**	/BUSY
2	*Something to do with RC clocking?*	**39**	D7
3	Clock Input *(for direct AND RC clocking)*	**38**	A11
4	Clock Output *(for clocked ROMs? according to the patent this is likely a signal at half the speed of the clock input. Alternatively it may have to do with RC clocking)*	**37**	D6
5	A test pin, gives an analog signal. *(attached to one of the 4 DAC INPUTS?)*	**36**	A10
6	A test pin, gives an analog signal. *(attached to one of the 4 DAC INPUTS?)*	**35**	A9
7	A test pin, gives an analog signal. *(attached to one of the 4 DAC INPUTS?)*	**34**	D5
8	A test pin, gives an analog signal. (*attached to one of the 4 DAC INPUTS?)*	**33**	A8
9	/ROMEN or 'Address read' *(speech ROM is being read when this goes low)*	**32**	D4
10	START	**31**	A7
11	Analog Out	**30**	D3
12	A0	**29**	A6
13	ID0	**28**	ID5
14	A1	**27**	D2
15	ID1	**26**	ID4
16	A2	**25**	D1
17	ID2	**24**	ID3
18	A3	**23**	D0
19	A4	**22**	A5
20	GND	**21**	-10VDC

Note from Kevin Horton when testing the hookup of the S14001A: the /BUSY line is not a standard voltage line: when it is in its HIGH state (i.e. not busy) it puts out a voltage of -10 volts, so it needs to be dropped back to a sane voltage level before it can be passed to any sort of modern IC. The address lines for the speech ROM (A0-A11) do not have this problem, they output at a TTL/CMOS compatible voltage.

Operation:

Put the 6-bit address of the word to be said onto the ID0-ID5 lines. Then clock the START line. As long as the START line is held high, the first address byte of the first word will be read repeatedly every clock, with the enable line . The signal is just passed through the chip. Once START has gone low-high-low, the /BUSY line will go low until 3 clocks after the chip is done speaking.

For example, lets have the chip play word 03 from the Berzerk ROMs. For two clocks, the chip will read the word address high nibble from byte 6 (one with /ROMEN low, one with it high) Then, for another two clocks, read the low address nibble from byte 7 (again one clock with /ROMEN low, one with it high). For simplicity's sake assume all reads have this 2-clock behavior, and that all reads hence take two clocks. The next two clocks, the chip reads the syllable address pointed to by bytes 6 and 7. Since locations 06 and 07 have the bytes 05 E0, we read the next byte from 0x5E. The S14001A will read a lookup index location in the ROM (called the 'word memory' in the patent) and then based on that index data, will read data from an address based on the data read in the ROM (this is called the 'syllable memory'), and start speaking.

ROM Format:

```
03 A0 (epiphany at 16:50 on 12/04/2005) means word data starting at 0x003A
17 1d (epiphany at 17:05 on 12/04/2005) means phoneme? (syllable) data starting
at 0x017xx ?

 / 00000000   03 A0 04 40  04 E0 05 E0  06 E0 07 80  08 40 08 C0  ...@.........@..
 | 00000010   09 20 09 A0  0A E0 0B 40  0B 80 0B E0  0C 60 0C A0  . .....@.....`..
 1 00000020   0D 80 0E 60  0F 80 10 80  10 C0 11 E0  12 C0 13 60  ...`...........`
 = 00000030   13 C0 14 C0  15 60 15 E0  16 00 <-1| |2-> 17 1D  17 1D 1D 49  .....
`.........I
 | 00000040   1D 6E 20 9F  22 1F 22 1F  17 1E 24 41  28 D9 29 59  .n ."."...$A(.)Y
 | 00000050   29 78 2D 1F  2D 1F 2D 7F  2A 49 2A 78  22 9F 2D 1F  )x-.-.-.*I*x".-.
 | 00000060   2D 1F 2F 1C  37 41 37 7C  2D 1F 3B 5B  2F 9C 22 1F  -./.7A7|-.;[/.".
 | 00000070   3C 51 3E 51  3E 6E 2D 9F  2F 1D 2F 1E  2F 1D 40 50  <Q>Q>n-./././.@P
 | 00000080   40 6E 2D 9F  22 1F 42 49  42 78 2D 9F  45 41 49 1C  @n-.".BIBx-.EAI.
 | 00000090   49 9C 51 59  52 41 52 78  2D 9F 2D 1F  56 52 56 79  I.QYRARx-.-.VRVy
 | 000000A0   2D 1F 58 51  58 78 2D 1F  5A 51 5A 78  2D 9F 5C 1E  -.XQXx-.ZQZx-.\.
 | 000000B0   5C 1E 60 C1  64 41 68 D9  69 50 69 6E  2D 9F 5C 1E  \.`.dAh.iPin-.\.
 | 000000C0   5C 1E 6B 5A  6C C1 70 41  74 D1 17 1D  17 1D 76 41  \.kZl.pAt.....vA
 2 000000D0   7A 41 7E 59  7E 7C 2D 9F  22 1F 22 1F  22 7E 7F 41  zA~Y~|-."."."~.A
 | 000000E0   83 58 49 1D  49 9D 20 1F  17 1E 84 51  84 78 22 1F  .XI.I. ....Q.x".
 | 000000F0   22 7F 86 58  86 78 2D 9F  87 49 87 7C  2D 1F 2D 1F  "..X.x-..I.|-.-.
 | 00000100   2D 7E 8F 5B  8A 5A 8B C1  96 51 98 D0  9A 58 49 1D  -~.[.Z...Q...XI.
 | 00000110   49 1D 49 7C  22 1F 17 1E  9B 49 9B 79  20 9F 2D 1F  I.I|"....I.y .-.
 | 00000120   9E 52 49 1D  49 1D 49 79  A0 41 A4 D9  A5 41 49 1D  .RI.I.Iy.A...AI.
 | 00000130   49 1D 49 78  2D 9F A9 49  A9 78 2D 9F  2D 1F 2D 1F  I.Ix-..I.x-.-.-.
 | 00000140   2F 1C AC 58  AC 79 22 1F  22 7F AD C9  B0 1D B0 1D  /..X.y"."......
 | 00000150   B6 49 B6 78  2D 9F B9 41  B9 78 22 1F  17 9F BD C1  .I.x-..A.x".....
 \ 00000160   8F 41 8F 7C  20 1F 93 49  93 78 2D 9F  FF FF FF FF  .A.| ..I.x-.....
 / 00000170   75 A3 28 C9  D7 59 D9 67  5D 69 75 D8  C7 97 5A 29  u.(..Y.g]iu...Z)
 | 00000180   89 A5 A5 A6  26 97 5A 89  67 65 9D 96  69 89 A5 A2  ....&.Z.ge..i...
 3 00000190   67 5D 76 28  9E 22 98 CA  63 27 28 9A  65 A5 9A 66  g]v(."..c'(.e..f
```

The first area of ROM is the word table. This table holds the addresses where each word's data starts. In the Berzerk speech ROM, it runs from 0x0000 to 0x0039, but theoretically it can run from 0x0000 to 0x007F if all 64 possible words are used. To decode addresses stored in this table, read them as big endian 16-bit values, and rightshift them by 4.

The second area of ROM is the syllable table. This table holds the addresses and parameters of the delta-modulation-encoded pcm data to be played back. In the Berzerk ROM, it runs from 0x003A to 0x016B, with four FFs after that to pad the ROM to exactly 0x170. To decode the data stored in this table, read them as pairs of bytes.

The first byte of data is the high 2 nibbles of the address, i.e. 17 xx means the data starts at 0x170
The second byte, or parameter byte, is formatted as such:

```
7 6 5 4 3 2 1 0
G B Y S S S R R
```

- G is set if this is the last syllable of a word.
- B is CLEAR if the syllable is played through straight only once instead of mirroring after the block end.
  (B being clear disables all repeats of phones, BUT then R acts as an additional multiplier for total number of phones played)
  If B is clear, double the number of total syllables (i.e 4x the number of audible since all are now audible unless the silence bit is set) are played in the same time period as otherwise.
- Y is set if the syllable is silent. the syllable is read and probably rendered normally internal to the chip, but the output DAC is held at 0x07.
- S is the length of the word in syllables. number of syllables is 8 - (this_number).
- R is the number of times a syllable repeats. number of repeats is 8 - (2 * this_number).

B, Y, S and R can also be described by a table:

```
0x1a = (straight, 24 syllables, no repeats)                 0001 1010
0x1b = (straight, 20 syllables, no repeats)                 0001 1011
0x1C = (straight, 16 syllables, no repeats)                 0001 1100
0x1D = (straight, 12 syllables, no repeats)                 0001 1101
0x1E = (straight, 8 syllables, no repeats)                  0001 1110
0x1F = (straight, 4 syllables, no repeats)                  0001 1111
0x40 = (mirrored, 8 syllables, 8 full repeats, 4 audible)   0100 0000
0x41 = (mirrored, 8 syllables, 6 full repeats, 3 audible)   0100 0001
0x42 = (mirrored, 8 syllables, 4 full repeats, 2 audible)   0100 0010
0x43 = (mirrored, 8 syllables, 2 full repeats, 1 audible)   0100 0011
0x44 = (mirrored, 7 syllables, 8 full repeats, 4 audible)   0100 0100
0x45 = (mirrored, 7 syllables, 6 full repeats, 3 audible)   0100 0101
0x46 = (mirrored, 7 syllables, 4 full repeats, 2 audible)   0100 0110
0x47 = (mirrored, 7 syllables, 2 full repeats, 1 audible)   0100 0111
0x48 = (mirrored, 6 syllables, 8 full repeats, 4 audible)   0100 1000
0x49 = (mirrored, 6 syllables, 6 full repeats, 3 audible)   0100 1001
0x4A = (mirrored, 6 syllables, 4 full repeats, 2 audible)   0100 1010
0x4B = (mirrored, 6 syllables, 2 full repeats, 1 audible)   0100 1011
0x4C = (mirrored, 5 syllables, 8 full repeats, 4 audible)   0100 1100
0x4D = (mirrored, 5 syllables, 6 full repeats, 3 audible)   0100 1101
0x4E = (mirrored, 5 syllables, 4 full repeats, 2 audible)   0100 1110
0x4F = (mirrored, 5 syllables, 2 full repeats, 1 audible)   0100 1111
0x50 = (mirrored, 4 syllables, 8 full repeats, 4 audible)   0101 0000
0x51 = (mirrored, 4 syllables, 6 full repeats, 3 audible)   0101 0001
0x52 = (mirrored, 4 syllables, 4 full repeats, 2 audible)   0101 0010
0x53 = (mirrored, 4 syllables, 2 full repeats, 1 audible)   0101 0011
0x54 = (mirrored, 3 syllables, 8 full repeats, 4 audible)   0101 0100
0x55 = (mirrored, 3 syllables, 6 full repeats, 3 audible)   0101 0101
0x56 = (mirrored, 3 syllables, 4 full repeats, 2 audible)   0101 0110
0x57 = (mirrored, 3 syllables, 2 full repeats, 1 audible)   0101 0111
0x58 = (mirrored, 2 syllables, 8 full repeats, 4 audible)   0101 1000
0x59 = (mirrored, 2 syllables, 6 full repeats, 3 audible)   0101 1001
0x5A = (mirrored, 2 syllables, 4 full repeats, 2 audible)   0101 1010
0x5B = (mirrored, 2 syllables, 2 full repeats, 1 audible)   0101 1011
0x5C = (mirrored, 1 syllables, 8 full repeats, 4 audible)   0101 1100
0x5D = (mirrored, 1 syllables, 6 full repeats, 3 audible)   0101 1101
0x5E = (mirrored, 1 syllables, 4 full repeats, 2 audible)   0101 1110
0x5F = (mirrored, 1 syllables, 2 full repeats, 1 audible)   0101 1111
...
0x6a = (straight, 24 syllables, silent)                     0110 1010
0x6b = (straight, 20 syllables, silent)                     0110 1011
0x6C = (straight, 16 syllables, silent)                     0110 1100
0x6D = (straight, 12 syllables, silent)                     0110 1101
0x6E = (straight, 8 syllables, silent)                      0110 1110
0x6F = (straight, 4 syllables, silent)                      0110 1111
0x70 = (mirrored, 4 syllables, 8 full repeats, 0 audible)   0111 0000
0x71 = (mirrored, 4 syllables, 6 full repeats, 0 audible)   0111 0001
0x72 = (mirrored, 4 syllables, 4 full repeats, 0 audible)   0111 0010
0x73 = (mirrored, 4 syllables, 2 full repeats, 0 audible)   0111 0011
0x74 = (mirrored, 3 syllables, 8 full repeats, 0 audible)   0111 0100
0x75 = (mirrored, 3 syllables, 6 full repeats, 0 audible)   0111 0101
0x76 = (mirrored, 3 syllables, 4 full repeats, 0 audible)   0111 0110
0x77 = (mirrored, 3 syllables, 2 full repeats, 0 audible)   0111 0111
0x78 = (mirrored, 2 syllables, 8 full repeats, 0 audible)   0111 1000
0x79 = (mirrored, 2 syllables, 6 full repeats, 0 audible)   0111 1001
0x7A = (mirrored, 2 syllables, 4 full repeats, 0 audible)   0111 1010
0x7B = (mirrored, 2 syllables, 2 full repeats, 0 audible)   0111 1011
0x7C = (mirrored, 1 syllables, 8 full repeats, 0 audible)   0111 1100
0x7D = (mirrored, 1 syllables, 6 full repeats, 0 audible)   0111 1101
0x7E = (mirrored, 1 syllables, 4 full repeats, 0 audible)   0111 1110
0x7F = (mirrored, 1 syllables, 2 full repeats, 0 audible)   0111 1111
```

And all these numbers +8 for when they occur as the last syllable, to account for G.

The third, and final area of the ROM, is the phoneme data. This area holds the delta-modulation-compressed samples of speech. The encoding format is a bit strange, the patent calls it "floating-zero, two-bit delta modulation".

Decoding the Phoneme Data:

Here is how the phoneme data is decoded, according to the patent:

(initially and after a reset, the old 2-bit data (old_in) is 10b (2), the accumulator output is 7 (accumulator is connected to the DAC through an analog switch, and it runs from 0/low to F/high)), and the data shift count is 0)

- Step 1: Grab the (direction ? next : previous) data byte.
- Step 2: Mask the (direction ? high : low) two bits of the data byte and put into a register (cur_in). This is the 2-bit encoded delta data.
- Step 3: This part is a little tricky and is handled by a table in a PROM normally.
  If the direction is 1 (backwards), SWITCH the current and old values before feeding to the table.

  Here's the table. X means we don't care.
  Old 2 bits: Current 2 bits: 4-bit signed Output: Value of Output:

Old 2 bits:		**Current 2 bits:**		**4-bit signed Output:**				
MSB	LSB	MSB	LSB	MSB	---	---	LSB	Value
0	X	0	0	1	1	0	1	-3
0	X	0	1	1	1	1	1	-1
0	X	1	0	0	0	0	0	0
0	X	1	1	0	0	0	1	1
1	X	0	0	1	1	1	1	-1
1	X	0	1	0	0	0	0	0
1	X	1	0	0	0	0	1	1
1	X	1	1	0	0	1	1	3

- Step 4: Take the 4-bit signed output and add it to the signed old final output, the result is the unsigned 4-bit dac output. Yes, it's a little weird, but it should work.
- Step 5: Copy cur_in to old_in, shift data byte (direction ? left : right) by two, add one to shift count.
- Step 6: If the shift count is less than 4, shift the input byte right by two, and go to step 2.

If we're immediately after the mirror point in a mirrored sample, the last accumulator output is simply repeated and not recalculated using the delta. the old/new deltas update as usual though. **this is very important!** see the S14001A.c code at http://www.netaxs.com/~gevaryah/S14001A.c for an example delta demodulator with proper mirroring.

Reading the Data:

Once we've demodulated our data block Otherwise check if we're done our syllable:

- If we're not, zero the shift count and go to step 1.
- If we are, then on the S14001A we reset the decoder and output a block of silence exactly the same length as the syllable was. Then we check if we need to repeat the syllable, and we do so (including the silence) as many times as the parameter byte dictates. After we repeat the syllable the required number of times, we check if we're done our word:
- If we're not, retrieve the next syllable address and syllable parameters, reset the decoder, then start playing that new data.
- If we're done our word completely, I.E. the high parameter bit of the most recently played syllable was set (to indicate that it is the last one in a word), then output silence.

Berzerk Stuff:

Based on the above information, the Syllable table words for Berzerk are:

INDEX:
Word number (address in syllable table of word data) "word spelled out" (comments) : Syllable data of word first syllable byte pair (most similar "diphone code" to sound played) more syllable byte pairs

/ before a "diphone code" means that diphone is played silent due to the Y bit.

```
00 (0x03A) "help" : 17 1D 17 1D 1D 49 1D 6E 20 9F
17 1D (IH)
17 1D (IH)
```

```
1D 49 (EL)
1D 6E (/EL)
20 9F (P)

01 (0x044) "kill" : 22 1F 22 1F 17 1E 24 41 28 D9
22 1F (K)
22 1F (K)
17 1E (IH)
24 41 (UH)
28 D9 (L)

02 (0x04E) "attack" : 29 59 29 78 2D 1F 2D 1F 2D 7F 2A 49 2A 78 22 9F
29 59 (UH)
29 78 (/UH)
2D 1F (T)
2D 1F (T)
2D 7F (/T)
2A 49 (A)
2A 78 (/A)
22 9F (K)

03 (0x05E) "charge" : 2d 1f 2d 1f 2f 1c 37 41 37 7c 2d 1f 3b 5b 2f 9c
2D 1F (DT)
2D 1F (DT)
2F 1C (CH)
37 41 (AR)
37 7C (/AR)
2D 1F (DT)
3B 5B (J)
2F 9C (CH)

04 (0x06E) "got" : 22 1f 3c 51 3e 51 3e 6e 2d 9f
22 1f (K)
3c 51 (G)
3e 51 (AH)
3e 6e (/AH)
2d 9f (DT)

05 (0x078) "shoot" : 2f 1d 2f 1e 2f 1d 40 50 40 6e 2d 9f
2f 1d (CH)
2f 1e (CH)
2f 1d (CH)
40 50 (OO)
40 6e (/OO)
2d 9f (DT)

06 (0x084) "get" : 22 1f 42 49 42 78 2d 9f
22 1f (K)
42 49 (EH)
42 78 (/EH)
2d 9f (DT)

07 (0x08c) "is" : 45 41 49 1c 49 9c
45 41 (IH)
49 1c (SZ)
49 9c (SZ)

08 (0x092) "alert" : 51 59 52 41 52 78 2d 9f
51 59 (UHL)
52 41 (ER)
52 78 (/ER)
2d 9f (DT)

09 (0x09A) "detected" : 2d 1f 56 52 56 79 2d 1f 58 51 58 78 2d 1f 5a 51 5a 78 2d 9f
2d 1f (DT)
56 52 (EE)
56 79 (/EE)
```

2d 1f (DT)
58 51 (EH)
58 78 (/EH)
2d 1f (DT)
5a 51 (KTIH)
5a 78 (/KTIH)
2d 9f (DT)

0A (0x0AE) "the" : 5c 1e 5c 1e 60 c1
5c 1E (THV)
5c 1E (THV)
60 C1 (UEH <schwa>)

0B (0x0B4) "in" : 64 41 68 d9
64 41 (IH)
68 D9 (NN)

0C (0x0B8) "it" : 69 50 69 6e 2d 9f
69 50 (IH)
69 6E (/IH)
2d 9f (DT)

0D (0x0BE) "their/there" : 5c 1e 5c 1e 6b 5a 6c c1
5c 1E (THV)
5c 1E (THV)
6b 5A (EI)
6c C1 (R)

0E (0x0C6) "where" : 70 41 74 d1
70 41 (WHE)
74 D1 (ER)

0F (0x0CA) "humanoid" : 17 1d 17 1d 76 41 7a 41 7e 59 7e 7c 2d 9f
17 1D (IH)
17 1D (IH)
76 41 (YUU)
7A 41 (MAHN)
7E 59 (OY)
7E 7C (/OY)
2D 9F (DT)

10 (0x0D8) "coins" : 22 1f 22 1f 22 7e 7e 41 83 58 49 1d 49 9d
22 1F (K)
22 1F (K)
22 7E (/K)
7E 41 (OY)
83 58 (N)
49 1D (SZ)
49 9D (SZ)

11 (0x0E6) "pocket" : 20 1f 17 1e 84 51 84 78 22 1f 22 7f 86 58 86 78 2d 9f
20 1F (P)
17 1E (IH)
84 51 (IAH)
84 78 (/IAH)
22 1F (K)
22 7F (/K)
86 58 (ID)
86 78 (/ID)
2d 9F (DT)

12 (0x0F8) "intruder" : 87 49 87 7c 2d 1f 2d 1f 2d 7e 8f 5b 8a 5a 8b c1
87 49 (IN)
87 7C (/IN)
2D 1F (DT)
2D 1F (DT)
2D 7E (/DT)

```
8F 5B (R)
8A 5A (OO)
8B C1 (DER)

13 (0x108) "no" : 96 51 98 d0
96 51 (N)
98 D0 (OWE)

14 (0x10C) "escape" : 9A 58 49 1D 49 1D 49 7C 22 1F 17 1E 9B 49 9B 79 20 9F
9A 58 (EH)
49 1D (SZ)
49 1D (SZ)
49 7C (/SZ)
22 1F (K)
17 1E (IH)
9B 49 (AY)
9B 79 (/AY)
20 9F (P)

15 (0x11E) "destroy" : 2D 1F 9E 52 49 1D 49 1D 49 79 A0 41 A4 D9
2D 1F (DT)
9E 52 (EE)
49 1D (SZ)
49 1D (SZ)
49 79 (/SZ)
A0 41 (TR)
A4 D9 (OY)

16 (0x12C) "must" : A5 41 49 1D 49 1D 49 78 2D 9F
A5 41 (MUH)
49 1D (SZ)
49 1D (SZ)
49 78 (/SZ)
2D 9F (DT)

17 (0x136) "not" : A9 49 A9 78 2D 9F
A9 49 (NOH)
A9 78 (/NOH)
2D 9F (DT)

18 (0x13C) "chicken" : 2D 1F 2D 1F 2F 1C AC 58 AC 79 22 1F 22 7F AD C9
2D 1F (DT)
2D 1F (DT)
2F 1C (CH)
AC 58 (IH)
AC 79 (/IH)
22 1F (K)
22 7F (/K)
AD C9 (N)

19 (0x14C) "fight" : B0 1D B0 1D B6 49 B6 78 2D 9F
B0 1D (F)
B0 1D (F)
B6 49 (IY)
B6 78 (/IY)
2D 9F (DT)

1A (0x156) "like" : B9 41 B9 78 22 1F 17 9F
B9 41 (LIY)
B9 78 (/LIY)
22 1F (K)
17 9F (IH)

1B (0x15E) "a" : BD C1
BD C1 (A)

1C (0x160) "robot" : 8F 41 8F 7C 20 1F 93 49 93 78 2D 9F
```

```
8F 41 (RO)
8F 7C (/RO)
20 1F (P)
93 49 (OH)
93 78 (/OH)
2D 9F (DT)
```

Encoded Phonemes:

The Berzerk encoded phonemes are:

```
0x170 = IH (used as a schwa+fricative 'H' for words like 'hustle' and 'humanoid')
0x1Dn = EL
0x20n = P (plosive)
0x22n = K
0x24n = UH ('front')
0x28n = L
0x29n = UH ('bucket')
0x2An = A ('rat' 'pack' 'sat')
0x2Dn = DT (plosive, not mirrored)
0x2Fn = CH (fricative, also used as SCH, not mirrored?)
0x37n = AR
0x3Bn = J
0x3Cn = G
0x3En = AH ('pocket')
0x40n = OO
0x42n = EH ('bet', descending?)
0x45n = IH
0x49n = SZ (voiced S sound like in 'zebra')
0x51n = UHL
0x52n = ER
0x56n = EE
0x58n = EH ('regret', ascending?)
0x5An = KTIH
0x5Cn = THV (voiced TH as in 'thee')
0x60n = UEH (schwa)
0x64n = IH
0x68n = N
0x69n = IH
0x6Bn = EI
0x6Cn = R
0x70n = WHE
0x74n = ER
0x76n = YUU
0x7An = MUHN
0x7En = OY
0x83n = N
0x84n = IAH
0x86n = ID (almost like it ends with an N)
0x87n = IN
0x8An = OO
0x8Bn = DER
0x8Fn = RO
0x93n = OH (AWH, like in 'rock')
0x96n = N
0x98n = OWE
0x9An = EH
0x9Bn = AY
0x9En = EE
0xA0n = TR
0xA4n = OY
0xA5n = MUH
0xA9n = NOH
0xACn = IH
0xADn = N
0xB0n = F (fricative)
```

```
0xB6n = IY
0xB9n = LIY
0xBDn = A (long A)
```