ACORN COMPUTER

Speech System USER GUIDE

THE BBC MICROCOMPUTER

SPEECH SYSTEM

USER GUIDE

Part No. 403950 / Issue 2 / 4.5.83

CONTENTS

SECTION ONE
FITTING THE SPEECH SYSTEM

The BBC Microcomputer Speech System is an on-board extension to your computer. This extension must be carried out by an authorised dealer, and once installed, you need no external hardware or software to start using the system.

When this extension has been fitted, you will notice that two sockets have appeared to the left of the keyboard. These sockets are provided to let you insert special plug-in cartridges which will be available in the future. Not only will you be able to expand your Speech System in this way, but you will also be able to use other cartridge-based software with a wide variety of applications.

Note that the Speech System must be fitted to your BBC Microcomputer before you can use any plug-in cartridges.

SECTION TWO

INTRODUCTION TO THE SPEECH SYSTEM

The Speech System uses your BBC Microcomputer's sound generator, audio amplifier and speaker to "speak" words.

The system contains a Speech Processor and a special chip called a Phrase Read Only Memory (PHROM). In this device are stored 165 ready-made words and word-parts. These "word-parts" are for putting on the beginning or end of the words contained in the PHROM, for example, -ING IN- -ED and so on.

The Speech Processor contains a digital filter which is rather like a computer version of the human vocal tract. It relies on being fed a large number of parameters to produce the necessary sounds for making speech. Although you have direct access to the Speech Processor, it is a very highly complex task to synthesise speech, and this is why the PHROM is provided.

The actual words and word-parts stored in the PHROM depend on the particular PHROM fitted. The first Speech System expansions contain a " WORD PHROM A", and this is the one referred to in this Guide. Other PHROM's will become available in the future, and also in the form of plug-in cartridges as mentioned earlier.

Take a look at Appendix A on page 31 which gives a list of all the available words and word-parts stored in Word PHROM A.

Unless a distinction needs to be made, "words" implies both words AND word-parts from now on..

ACCESS TO THE SPEECH SYSTEM

1. In BASIC

This is done by using a special form of the BASIC command
SOUND
All the existing SOUND command facilities remain intact, but the Speech System provides a range of its own commands, which usually takes the form

```
        SOUND -1,<variable>,0,0
```

2. In Assembly Language

This is done by using the OSWORD and OSBYTE calls in your Assembly language programs. See page 15 for more details.

USING THE SPEECH SYSTEM

There are three ways in which you can use the Speech System.

1 You can access and manipulate the 165 words already stored in the PHROM. This can be done quickly and easily in BASIC, and is described fully in Section 3.

2 You can build your own words, again by using BASIC. This requires a thorough understanding of the speech data parameters used in the Speech System.
Section 7 explains how the speech system works, and Section 5 contains a program to input the speech data to the Speech System.

3 You can also do 1 and 2 above by writing Assembly language programs. This is explained in Section 6, but again it is stressed that Section 7 should be understood before attempting to build your own words.

SECTION THREE

USING THE SPEECH SYSTEM FROM BASIC

NOTE: From now on, any text <enclosed in brackets> in the Speech SOUND command is to tell you what sort of data is needed there. For example SOUND -1,<word number>, 0,0 means type in a number in place of <word number>. Using the word number 160, the SOUND command would be SOUND -1,160,0,0

The existing BASIC command SOUND is used to operate the Speech System, but takes a special form :

```
SOUND -1,<word number>,0,0
```

The value of this "word number" determines which of the 165 words and word -parts is selected from Word PHROM A.

Turn to Appendix A. This contains a complete list of all the words and word-parts available in the PHROM in alphabetical order. You will see that there are three columns containing : the word number (starting at 127), its absolute address in the PHROM (ignore this for the time being) , and the corresponding word itself.

To produce the word "Acorn", type the following :

```
SOUND -1,160,0,0
```

Word numbers in the range 32 to 126, which corresponds to the ASCII code range, will produce words which have an association with the ASCII character of the same number.

Appendix B lists these words in ASCII code number order. As you may have noticed, they are also available elsewhere in the PHROM in the complete list (see Appendix A).

So as an alternative to inserting the word number in the SOUND command, you can use the following format for those words listed in Appendix B :

```
SOUND -1,ASC"<character>",0,0
```

— Using upper case letters will produce letters of the alphabet

— Numeric characters will produce numbers

— Lower case letters and remaining characters will produce words with some kind of association where possible.

For example, to produce the letter "I",

```
SOUND -1,ASC"I",0,0
```

To produce the number "8" :

```
SOUND -1,ASC"8",0,0
```

To produce the letter "A" :

```
SOUND -1,ASC"A",0,0
```

To produce the word "large" :

```
SOUND -1,ASC">",0,0
```

To produce the word "button" :

```
SOUND -1,174,0,0
```

In this case, there is no ASCII character which corresponds to the word "button", but the word does reside in the PHROM, and has the number 174.

Try putting line numbers in front of each of the lines above to make a program, then run it.

Alternatively, make one line of BASIC by separating each of the five commands by a colon.

Although "I ate a large button" isn't a particularly profound statement, it does show how you can manipulate the words stored in the PHROM.

```
10 SOUND -1,ASC"I",0,0
20 SOUND -1,ASC"8",0,0
30 SOUND -1,ASC"A",0,0
40 SOUND -1,ASC">",0,0
50 SOUND -1,174,0,0
```

The following line of BASIC will allow you to listen to the words associated with all the ASCII codes generated from the keyboard.

```
REPEAT : SOUND -1,GET,0,0 : UNTIL 0
```

After you have typed in this line and pressed RETURN, every time you press a key, you will hear the corresponding word. Remember to use the SHIFT key to get the full range.

COMPOUNDING WORDS AND WORD-PARTS

You will find that the word-parts, which are all listed in Appendix A, either start or end with a hyphen to show that they are word-parts rather than words.

You can see that the "**I** ate a large button" program on page 6 produces a sequence of words. A similar program can be written which contains words and word-parts.

This enables you to compose your own words.

Try producing the word "Incomplete" :

INCOMPLETE   = IN- COMPLETE

Look up the word/word-part numbers for IN- and COMPLETE.

```
    IN-       is no. 207
    COMPLETE is no. 178
```

Type the following line :

```
        SOUND -1,207,0,0 : SOUND -1,178,0,0
```

When you press RETURN at the end of this line, you will hear the word "INCOMPLETE"

Here are some more :

ALIGN   = UH LINE

```
        SOUND -1,276,0,0 : SOUND -1,216,0,0
```

ALIGNS   = UH LINE -Z

```
        SOUND -1,276,0,0 : SOUND -1,216,0,0 : SOUND -1,138,0,0
```

BEES   = B -Z

```
        SOUND -1,170,0,0 : SOUND -1,138,0,0
```

DEEDS   = D -D -Z

```
        SOUND -1,181,0,0 : SOUND -1,131,0,0 : SOUND -1,138,0,0
```

INTEND   = IN- TEN -D

```
         SOUND -1,207,0,0 : SOUND -1,264,0,0 : SOUND -1,131,0,0
```

With experimentation and a little imagination you will be able to create a surprising number of meaningful words and phrases.

PLURALS, TONES and PAUSES

Here is some more information to help you compound your words and word-parts.

-S -Z : These are two word-parts used to make single words plural. The "-S" word-part is used when a "hard S" is needed, and the "-Z" is used when a "soft S" is needed.

Here are some examples :

Hard S	Soft S
POINTS = POINT -S	POUNDS = POUN -Z
DATES = DATE -S	DOLLARS = DOLLAR -Z

So to produce the word "DATES", you need to look up the word/word-part numbers for DATE and -S in Appendix A. These are 182 and 134 respectively, then type :

SOUND -1,182,0,0 : SOUND **-1,134,0,0**

NUMBERS : The following word-parts are used for numbers more than ten.

2- produces "TWEN-"
3- produces "THIR-"
4- produces "FOR-"
5- produces "FIF-"
6- produces "SIX-"
7- produces "SEVEN-"
8- produces "EIGHT-"
9- produces "NINE-"

Use these word-parts with the word-parts "-TY" and "-TEEN"

For example, to produce the word "FOURTEEN", use "4-" and "-TEEN". Look up the word/word-part numbers in Appendix A, and then you will be able to type the following line :

SOUND -1,148,0,0 : SOUND **-1,135,0,0**

TONES and PAUSES : There are four special numbers in the PHROM. These are numbered 127 to 130, and produce two fixed duration pauses and two tones of individual pitch. In more detail :

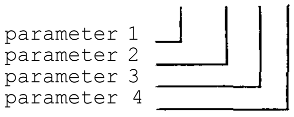
number		
127	(0.125)	produces a period of silence 0.125 seconds long
128	(0.25)	produces a period of silence 0.25 seconds long
129	(TONE 1)	produces a high-pitched tone
130	(TONE 2)	produces a low-pitched tone

SECTION FOUR
THE SOUND COMMAND IN DETAIL

The BASIC SOUND command requires four parameters. If the first parameter is a positive value, the computer treats the command as the normal SOUND command (see section 30 in the BBC Microcomputer User Guide). If the first parameter has a negative value, the computer treats the command as a Speech System command.
Although both types of SOUND command use four parameters, do not confuse them, they are completely different!

The Speech System SOUND command parameters are expressed as in the example below, which produces the word "First"

SOUND -1,198,0,0



Parameters 3 and 4 are always zero, but must always be included in the Speech System SOUND command: the reason for their inclusion is to maintain the syntax of the SOUND command. However, Parameters 1 and 2 can be expressed in a number of ways, and these are described below.

PARAMETER 1 : The computer expects a 4-digit hexadecimal number, and the value of this parameter tells the computer what to do with the other three parameters. There are four values you can use for parameter 1, and these are as follows :

Parameter 1 Value	Function
&FFFx	Speak using word number in PHROM number "x"
&FFBx	Speak using absolute address in PHROM number "x"
&FF60	Speak from RAM - see page 12
&FF00	Speak from RAM - see page 12

You are using WORD PHROM A which has been allocated the number 15. This is equivalent to &F, so in the first two cases above, x = &F. In the future you will be able to expand your Speech System to accomodate up to 16 WORD PHROMS, and these will be identified by numbers 0 to &F.

So far we have only used the value "-1" for parameter 1. This is equivalent to &FFFF ("Speak using word number from WORD PHROM A"). Try replacing the "-1" with &FFFF in any of the previous examples and you will see that they are interchangeable.

REMEMBER! Always enter the "&" sign before a hexadecimal number, or the computer will expect a decimal value.

The other three values of Parameter 1 are explained later.

PARAMETER 2 : Depending on Parameter 1, the computer expects Parameter 2 to be in one of three forms :

word number
ASC"<character>"
absolute address of the word

USING ABSOLUTE ADDRESSES

If you look at Appendix A again, you will see that beside each word number there is an "absolute address" value. This represents the store location in the PHROM where the associated word can be accessed, and is in hexadecimal.

For example, to produce the word "Correct" the command would be

```
        SOUND &FFBF,&1483,0,0
```

Parameter 1 = &FFBF : Speak using absolute addresses in PHROM number &F (decimal 15)

Parameter 2 = &1483 : The absolute address of the word "Correct" in the PHROM

Parameters 3 and 4 : zero as always

Note that there are four other ways to produce the word "Correct" :

```
        SOUND -1,180,0,0
        SOUND &FFFF,180,0,0

        SOUND -1,ASC"c",0,0
        SOUND &FFFF,ASC"c",0,0
```

When you incorporate speech in your programs, you will find that the flexibility of the speech SOUND command is very useful.

SECTION FIVE

MAKING YOUR OWN WORDS

INTRODUCTION

Up to now *we* have produced speech by using the ready-made words which are stored in the PHROM. Each word stored in the PHROM is in the form of a block of data, and this data contains all the parameters necessary for the Speech Processor to produce a word.

To make your own words, you need to do the following :

1 Work out the speech parameters needed to produce your new word. To do this, you will need to understand Section 7 very thoroughly.

2 Store the above parameters, in the form of a data block, **in** RAM.

3 Write a program to send the data block to the Speech Processor **in** a format it understands. The program on the next page will do this for you.

Here is a brief explanation of what the program is doing:

PROCINIT reserves memory space called ARRAY% and assembles the assembler mnemonics. The machine code routine reverses the order of the byte held in the accumulator. Thus it would change 00110010, for example, into 01001100.

Lines 1020 to 1050 read in the speech data byte by byte, reverse the order of the bits in each byte using the machine code routine REVERSE, and store the reverse order bytes in memory starting at ARRAY%. This is essential as the operating system expects the speech data to be stored in this back-to-front way.

The rest of the procedure sends the speech data to the Speech Processor, two bytes at a time:

Line 1070 sends the first two bytes of speech data to the speech processor using the SOUND &FF60 command. The first two bytes MUST be sent using this command as this also initialises the speech processor.

Lines 1080 to 1100 send the rest of the speech data using the SOUND &FF00 command. The SOUND &FF60 command should not be used again **or** the speech processor will be reset.

PROGRAM FOR SENDING SPEECH DATA TO THE SPEECH SYSTEM

PROCSPEAK(LL%)

```
1005 REM This program assumes you have already loaded the
speech data into memory, starting at location SPEECH%.
1006 REM LL% is the length of the data
1010 DEF PROCSPEAK(LL%)
1015 PROCINIT
1020 FOR I=0 TO LL%-1
1030 A%=I?SPEECH%
1040 ARRAY%?I=USR(REVERSE) AND &FF
1050 NEXT
1060 ARRAY%?(LL%+1)=0
1070 SOUND &FF60,!ARRAY% AND &FFFF,0,0
1080 FOR 1=2 TO LL% STEP 2
1090 SOUND &FF00,ARRAY%!I AND &FFFF,0,0
1100 NEXT
1110 ENDPROC
2000 DEFPROCINIT
2010 DIM ARRAY% 200
2020 DIM MC% 50
2030 FOR I=0 TO 2 STEP 2
2040 P%=MC%
2050 [OPTI
2060 .REVERSE STA &80
2070 ROL &80
2080 ROR A
2090 ROL &80
2100 ROR A
2110 ROL &80
2120 ROR A
2130 ROL &80
2140 ROR A
2150 ROL &80
2160 ROR A
2170 ROL &80
2180 ROR A
2190 ROL &80
2200 ROR A
2210 ROL &80
2220 ROR A
2230 RTS
2240 ]
2250 NEXT
2260 ENDPROC
```

LOADING SPEECH DATA FOR THE WORD "ZERO" INTO MEMORY

EXAMPLE PROGRAM

The following example program reads the speech data for the word "zero" into an array called SPEECH%, and uses the procedure PROCSPEAK(LL%) to send the speech data to the Speech System. PROCSPEAK(LL%) is listed on the previous page.

```
05 REM To read the data for the word "ZERO"
10 DIM SPEECH% 200
15 I=0
20 REPEAT
30 READ Y
40 SPEECH%?I=Y
50 I=I+1
60 UNTIL Y=0
70 REM data now in memory
80 REM now speak the word
90 LL%=I
100 PROCSPEAK(LL%)
110 END

3000 DATA 69,212,4,180,85,88,85,109
3010 DATA 129,43,17,78,53,1,241,35
3020 DATA 44,9,85,241,34,171,194,178
3030 DATA 211,104,204,49,109,34,196,89
3040 DATA 59,132,229,214,181,6,20,193
3050 DATA 57,121,174,65,150,72,46,77
3060 DATA 139,143,225,46,85,145,98,228
3070 DATA 24,108,156,58,232,185,6,21
3080 DATA 73,44,218,45,69,101,83,75
3090 DATA 190,139,17,206,109,79,21,105
3100 DATA 96,179,193,178,196,186,237,52
3110 DATA 240,46,207,19,59,81,58,149
3120 DATA 37,220,170,213,206,167,71,92
3130 DATA 35,182,52,146,209,188,246,169
3140 DATA 141,30,178,235,30,41,101,38
3150 DATA 195,204,71,74,89,73,177,16
3160 DATA 207,210,134,82,140,59,29,100
3170 DATA 221,164,160,238,169,94,47,0
```

SECTION SIX

USING THE SPEECH SYSTEM *IN* ASSEMBLY LANGUAGE

INTRODUCTION

There are two Operating System calls which allow you to access the Speech System in Assembly Language : OSWORD and OSBYTE. For more information on the Operating System calls, refer to Section 43 in the BBC Microcomputer Users Guide.

OSWORD call with A = &07

Using the OSWORD call with A=&07 gives you the same facilities as the BASIC command SOUND. As we have seen already, the SOUND command requires four parameters, and to use Speech, parameter no.1 must be one of four values special to the Speech System.
Registers X and Y are used as a register pair for sending the store locations containing the values of parameters 1 to 4 to the Operating System routine, and this is done as follows :

Set up a data block which contains the least significant byte of parameter 1, the MSB of parameter 1, the LSB of parameter 2... and so on up to the MSB of parameter 4.

Use registers X and Y to point to each successive store location in the data block. In the table below, the eight store locations in the data block have been called XY to XY+7.

ADDRESS	CONTENTS
XY	PARAMETER 1 LSB = COMMAND TYPE, see Note below
XY+1	PARAMETER 1 MSB = &FF
XY+2	PARAMETER 2 LSB = WORD NUMBER or ADDRESS; LOW BYTE
XY+3	PARAMETER 2 MSB = WORD NUMBER or ADDRESS; HIGH BYTE
XY+4	PARAMETER 3 LSB = &00
XY+5	PARAMETER 3 MSB = &00
XY+6	PARAMETER 4 LSB = &00
XY+7	PARAMETER 4 MSB = &00

Note - see page 10

Location XY contains the command type as for the SOUND command, i.e.

&Fx	to speak using pointers in PHROM number x
&Bx	to speak using absolute addresses in PHROM number x
&60 and &00	to speak from RAM

The following example program causes the computer to say the alphabet.

```
10 DIM Q% 100
20 DIM H% 8
30 FOR Z%=0 TO 2 STEP 2
40 P%=Q%
50 [OPTZ%
60 .START LDA #&FF
70 STA H%
80 STA H%+1
90 LDA #&00
100 LDX #&00
110 .LOOP STA H%+3,X
120 INX
130 CPX #&05
140 BNE LOOP
150 LDA #&41
160 .LOOP1 LDX #(H% MOD 256)
170 LDY #(5% DIV 256)
180 STA H%+2
190 LDA #&07
200 JSR &FFF1
210 LDA H%+2
220 CLC
230 ADC #&01
240 CMP #91
250 BNE LOOP1
260 RTS
270 ]  : NEXT
280 CALL START
290 END
```

OSBYTE CALLS &9E and &9F

The above section dealt with an OSWORD call which has the same effect as the BASIC SOUND command for the Speech System. It is possible to use the Speech System at an even lower level by directly writing to and reading from the Speech Processor. Before doing this you are advised to read Section 7 very carefully. It is quite a complicated matter to control the Speech Processor, and most of the time it is best to let the Operating System do so by using the OSWORD call or the SOUND command. However, two OSBYTE calls (call address &FFF4) have been provided to enable you to read and write to the Speech Processor.

OSBYTE call with A=&9E — Read the Speech Processor.

This call allows you to read from the Speech Processor. The byte read is either from the status register or the data register of the Speech Processor, depending on the previous command sent to the Processor. To read the Speech Processor's data register you should send a "Read Byte" command before making this call. Please refer to page 21 for a description on the "Read Byte" command. If the "Read Byte" command has not been sent, the status register will be read. The result is returned in Y.

OSBYTE call with A=&9F - Write to the Speech Processor.

This call allows you to send commands/data to the Speech Processor. The commands are discussed in Section 7. The command to be sent should be put in Y when the call is made.

The following example procedure illustrates the use of these calls to speak from RAM (you will need to refer to page 12 to understand how it works). The speech data should be in memory starting at location SPEECH%. This procedure could replace the one given on page 13. If you have typed in the program given on page 13, you can try this procedure by deleting lines 1010 to 1110 and typing in lines 1000 to 1240 below.

```
1000 DEF PROCSPEAK(LL%)
1010 PROCINIT
1020 FOR I=0 TO LL%-1
1030 A%=SPEECH%?I
1040 ARRAY%?I=USR(REVERSE) AND &FF
1050 NEXT
1060 ARRAY%?(LL%+1)=0
1070 DIM CODE% 100
1080 FOR N=0 TO 2 STEP 2
1090 P%=CODE%
1100 [OPTN
1110 .START LDY #&60
1120 LDA #&9F
1130 JSR &FFF4
1140 LDX #&00
1150 .LOOP LDY ARRAY%,X
1160 JSR &FFF4
1170 INX
1180 CPY #&00
1190 BNE LOOP
1200 RTS
1210 ]
1220 NEXT
1230 CALL START
1240 ENDPROC
```

PROCINIT should be the same as on page 13.

Lines 1020 to 1050 reverse the order of the bits in the speech data bytes and store the reversed bytes at ARRAY%.

Lines 1110 to 1130 send the byte &60 to the Speech Processor which is the "Speak External" command (see page 23). OSBYTE call &9F is used to do this.

The loop, (lines 1150 to 1190), sends the reversed speech data bytes to the Speech Processor using OSBYTE call &9F.

SECTION SEVEN

TECHNICALINFORMATION

INTRODUCTION

The Speech System consists of a Speech Processor and speech data held in the associated Word Phrase Read Only Memory (Word PHROM). The words which are available depend upon the contents of the Word PHROM fitted. The first Word PHROM fitted (Word PHROM A) is provided with the initial speech expansion which contains 165 words and word-parts.
These "word-parts" are for putting on the beginning or end of the words contained in the PHROM. For example, -ING IN- -ED and so on. Please refer to Section 3.
The complete contents of Word PHROM A is given in Appendix A.

The data held in the Word PHROM A is based upon recordings made by Kenneth Kendall, the BBC newsreader. The BBC Microcomputer Speech system is now one of the few available with an English accent!

THE SPEECH PROCESSOR

The Speech Processor used in the BBC Microcomputer Speech System is the Texas Instruments TMS 5220 Voice Synthesis Processor and you are referred to the data manual available from the manufacturers for full details of its use (see Appendix E). An introduction to the Speech Processor is given here.

To create speech, the Speech Processor must be fed with suitable speech data either by the computer's CPU (Central Processor Unit) or by direct access of the speech memory (Word PHROM's). The Speech Processor decodes the data to construct a time-varying digital filter model of the vocal tract. This model is excited with a digital representation of either glottal air impulses (voiced sounds) or the rush of air (unvoiced sounds) . The output of this model is passed to an eight-bit digital-to-analogue converter to produce a synthetic speech waveform.

The TMS 5220 Speech Processor may be likened to a standard Microprocessor since it has a number of registers and responds to a set of commands (similar to microprocessor op-codes).

THE SPEECH PROCESSOR REGISTERS

The TMS 5220 Speech Processor has two input registers :

a command register
a 128-bit FIFO (first in, first out) buffer

and two output registers :

a data register
a status register

Command register

The command register receives a command from the CPU and holds it for the Speech Processor to interpret and execute. The Speech Processor behaves as an attached processor to the host CPU and only acts when commanded to do so. The commands are discussed on pages 21 and 22.

FIFO buffer

The 128-bit FIFO buffer is organized as an 8-bit parallel-in, serial-out buffer which can hold up to 16 bytes. This buffer is used to hold the speech data sent by the CPU when speaking under CPU control (usually from RAM).

The speech synthesizer in the Speech Processor requires the data to be in serial form starting with the least significant bit of the first byte sent. This buffer provides the required action : the bytes of speech data are sent to the Speech Processor, first byte first, and are stored in this buffer until removed from the bottom by the speech synthesizer.

A stack pointer keeps track of the last free location so that data from the CPU is always loaded in the correct place. When the stack becomes less than half full the buffer-low status bit is set (see status register). This tells the CPU that more data should be sent as the buffer will be completely empty within 25 milliseconds. This is the worst case condition corresponding to only one byte left in the buffer.

If the buffer does become empty, the buffer-empty status bit is set (see status register) and the talk status latch is reset causing speech to stop immediately.

To carry on speaking under CPU control, another "Speak External" command will have to be sent (see pages 21 and 22 for command details).

Data register

This is an eight-bit serial-in parallel-out register. It is used by the Speech Processor to form a byte of data from the serial data read from the Speech data ROM (WORD PHROM) during a "Read byte" command (see page 21). Data is loaded so that the last bit loaded is placed in the least significant bit of the byte.

Status register

This is a three-bit register which provides information on the state of the Speech Processor. This register may be read at any time except immediately after a "Read byte" command. The register contents are transferred to the three most significant bits of the data bus when read (we will call these D7 to D5 inclusive). The three bits are as follows:

D7 Talk Status (TS). This is high (1) if the Speak command is sent, OR if the FIFO has been loaded more than half full following a Speak External command. It goes low (0) when the stop code (Energy = 1111 - see page 24) is processed, or if the FIFO buffer empties or if the Speech Processor is reset.

D6 Buffer Low (BL). This is high (1) when the FIFO buffer is more than half empty. Buffer low is set when the last-in byte passes the half way mark, and is cleared when more data is loaded in so that the buffer is more than half full.

D5 Buffer Empty (BE). This is high (1) when the FIFO buffer has run out of data while executing a "Speak External" command. When this bit is set, the talk status bit is cleared and speech is terminated.

THE SPEECH PROCESSOR COMMANDS

The Speech Processor responds to a number of commands sent by the CPU. Once the command is sent, the Speech Processor carries on and executes the command without involving the CPU. The commands available are:

Command Code	Operation
x000xxxx	Nop
x001xxxx	Read byte
x010xxxx	Nop
x110xxxx	Speak External
x011xxxx	Read and Branch
x100aaaa	Load address
x101xxxx	Speak
x111xxxx	Reset

x = don't care
a = address

These commands can be sent to the Speech Processor using the OSBYTE call with A=&9F as discussed on page 16.

Read Byte
This command allows the CPU to access the serial data held in the Speech Data ROM (e.g. Word PHROM A). This command causes the next eight bits to be read from the Speech Data ROM. The bits are placed in the eight-bit data register, the last bit read being placed in the least significant bit position. This data byte is then available to
be read by the CPU (using OSBYTE &9E). If another command is sent before the Speech Processor is read, the data byte will be lost and reading the processor will give the status register.

Read and Branch

This command acts as an indirect Load Address instruction, and is issued after setting up an address (by using the Load Address command) whose contents are known to contain the start address of speech data. It is strongly recommended that this command should not be used as it can only be guaranteed to work when only one Speech Data ROM is fitted.

Load Address

This command allows the CPU to alter the address register of the Speech data ROM (see page 22). The four bits labelled "a" above are loaded into a nibble of the Speech data ROM's address register. This command has to be called five times in succession to fully load the Speech Data ROMs address register. The least significant bit of the command (the rightmost "a") is loaded into the least significant bit position of the nibble in the Speech Data ROM address register. If five of these commands are called to fully load the address register, the least significant nibble of the register is loaded first, the most significant nibble last.

Speak

This command allows speech to be generated from the data stored in the Speech Data ROM. The Talk Status bit is set and speech starts using the next available data from the ROM. The Speech Processor continues to get data and produce speech until a stop code (Energy =1111) is received. Execution of the speak command can also be stopped by the execution of a reset command.

Speak External

This command allows the CPU to supply speech data to the Speech Processor, usually from RAM. When this command is received, the FIFO buffer is flushed and data subsequently sent to the Speech Processor is put into this buffer. When the buffer is more than half full, speech begins and continues until a stop code is encountered or the buffer becomes empty. While this command is executing all data sent to the Speech Processor is routed to the FIFO buffer and the reset command is not recognised as a command.

Reset

This command allows the CPU to halt a Speak command and put the Speech Processor in a known state. Speech is halted immediately the command is executed. TS is cleared, BL and BE are set (high) and the FIFO buffer is purged. A load address command is sent to the Speech ROM using dummy address data.

NOTE : The Reset command cannot halt the Speak External command.

SPEECH DATA ROMS

The Speech data ROM used in the BBC Microcomputer is the Texas Instruments 16K TMS6100, and you are referred to the data manual available from the manufacturers for full details (see Appendix E). As used in the BBC Microcomputer, this has a one bit serial output. Once the 14 bit address (of any of the 16K eight-bit bytes) is written into the device, data is read out one bit at a time by toggling a control pin. After eight bits have been read, the address is internally incremented. This device is thus ideal for use with the TMS 5220 Speech Processor which uses data in serial form.

The first thing to do before reading from the Speech ROM is to load it with the address within the ROM where you want to start reading. To do this you need to send the ROM five nibbles (a nibble is 4 bits long).   This is done by using the Speech Processors "Load address" command five times in succession. The least significant nibble is sent first, the most significant last. The five nibbles are used as follows:

```
+------------------------------------------------------------+
|     A     |     B     |     C     |     D     |     E     |
+------------------------------------------------------------+

Least significant                              Most significant


+-------------------------------------------+
|     4     |     4     |     4     |   2   |
+-------------------------------------------+-----------+
                                            |     4     |
     14 address bits                        +-----------+-----+
                                               4 chip   |  2  |
                                             select bits +-----+

                                                        ignored
```

The 4 chip select bits correspond to the ROM number referred to on page 10. Word PHROM A has been given ROM number 15.

If you wanted to work at a very basic level, the following steps would have to be carried out to speak the word at, for example, address &BB6 in ROM number 15 ("ACORN" in Word PHROM A):

(1) Use OSBYTE call &9F five times. Each time the call is made, Y should contain 01000110 (&46), 01001011 (&4B), 01001011 (&4B), 01001100 (&43), 01000011 (&4C) in succession. This sends the following 5 nibbles to the Speech ROM (0100aaaa being the "Load address" command): 6BBC3 or 0110 1011 1011 00 1111 00 , which as you can see sets up address &BB6 and ROM number 1111 (15).

(2) Use OSBYTE call &9F to send Y=01010000 (&50) which is the "Speak" command.

(3) To terminate a word before its conclusion so as to produce sounds from parts of words, you can use OSBYTE call &9F to send Y=01110000 (&70) after a suitable delay statement to reset the Processor.

SPEECH DATA

The speech is divided up into a number of frames, each 1/40th second long. Each frame consists of a number of parameters, the maximum being one energy parameter, one pitch parameter and ten reflection coefficients (K1 to K10). However, the actual values of the parameters are not stored as this would need an uneconomically large amount of memory. The parameters are firstly coded into a standard number of bits for each parameter. When the coded parameters are received by the speech synthesizer, the actual parameter value is looked up in a "Parameter Look-up ROM" within the Speech Processor.
The properties of the coded parameters are given below:

Parameter	Number of different values possible	number of bits
Energy	15	4
Pitch	64	6
K1	32	5
K2	32	5
K3	16	4
K4	16	4
K5	16	4
K6	16	4
K7	16	4
K8	8	3
K9	8	3
K10	8	3

Appendix D gives a table for converting actual parameter values into the coded parameters.

A number of special cases allow frames to have less than the maximum of 12 parameters. These are:

(a) The repeat facility. A repeat bit follows the energy parameter in each frame. If the bit is set (1) then only the energy and pitch parameters are required, the reflection parameters of the previous frame are used.

(b) Unvoiced speech (with Pitch = 000000) requires only four reflection parameters (K1 to K4). The other reflection parameters are automatically zeroed.

(c) When Energy = 0000 no other data is required. This will be the case during interword or intersyllable pauses.

With all the previous points taken into consideration we arrive at the following five different types of frame:

Frame	Energy	Repeat	Pitch	K1-K4	K5-K10	Total number of bits
Voiced	eeee	0	pppppp	needed	needed	50
Unvoiced	eeee	0	000000	needed	not used	29
Repeat	eeee	1	pppppp	not used	not used	11
Zero energy	0000	not used		not used	not used	4
Stop code	1111	not used		not used	not used	4

For example, the following would be the code for the word "zero":

Energy	Repeat	Pitch	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10
0100	0	101110	10100	00000	1001	0110	1000	1010	1010	101	100	001
0101	0	101101	10110	00000	1001	0101	1000	1000	1010	011	100	011
0101	0	000000	11111	00010	0100	0110						
0101	1	000000										
1001	0	101010	11111	00010	0100	0101	0101	0111	1000	010	101	100
1011	0	100110	11010	00110	0110	0001	1000	1011	0110	100	100	010
1100	0	100010	11001	00111	0111	0000	1001	1100	1011	101	011	010
1101	0	100000	11000	01010	0110	0000	1001	1100	1011	110	011	010
1110	0	100000	11001	01100	1001	0000	0101	1100	1001	101	100	010
1110	0	011111	11000	01001	0111	0010	1010	1100	1000	101	100	010
1110	0	100000	11000	01101	1001	0011	1000	0111	0101	110	100	010
1110	0	100000	11000	01010	1010	0100	1001	0110	0110	110	100	010
1101	0	100010	10110	01010	1010	0110	1001	0111	0111	110	100	010
1100	0	100011	10011	10011	0110	1010	0111	1000	1010	101	101	001
1100	0	100101	10011	11000	0011	0110	0101	1000	1001	011	101	011
1011	0	100110	10011	11000	0001	0111	0110	0111	1000	100	110	011
1011	0	101000	10011	10101	0010	1010	0100	1011	1011	100	101	010
1011	0	101011	10011	10101	0011	1010	0011	1010	1110	011	101	011
1011	0	110001	10100	10010	0101	1010	0011	0111	1001	111	011	010
1010	0	110001	10100	01111	0101	1001	0111	0101	1000	111	100	010
1001	0	110010	10010	01101	1000	0111	1001	1000	1000	111	010	010
1001	0	110010	10010	01101	1000	1000	1000	0110	0111	111	010	010
1000	0	110010	10010	10001	1000	0111	0110	0011	1010	110	010	011
0111	0	110100	10010	10000	0111	0111	0101	0100	1010	111	100	010
1111												

Note

If you wanted to put this data into RAM before sending it to the Speech Processor, the bits should be formed into eight-bit bytes. The first byte would be 01000101 or &45, the second 11010100 or &D4 etc. If the data does not exactly fit, the last byte should be padded out with zeros. The data is actually fed to the synthesizer in serial form via the Speech Processor's FIFO buffer and it is able to split up the bytes into the required length groups.

WORD PHROM A

Besides containing speech data in the form discussed in the previous section, Word PHROM A contains some other information related to the words. This includes a table of pointers, which allow the use of word numbers rather than addresses, and text strings providing yet another way to access the data. The format of the Word PHROM has been standardised as shown in the diagram on the next page.

```
Offset   Contents                                 Size
         +----------------------------------------+
   0     | PHROM format type                      |  1
         +----------------------------------------+
   1     | File data flag (&00 or &FF)            |  1
         +----------------------------------------+
   2     | ASCII "(" (decimal 40)                 |  1
         +----------------------------------------+
   3     | ASCII "C" (decimal 67)                 |  1
         +----------------------------------------+
   4     | ASCII ")" (decimal 41)                 |  1
         +----------------------------------------+
  5,53   | Text - 3 strings separated by &00      | 49
         +----------------------------------------+
 54,55   | PHROM serial number                    |  2
         +----------------------------------------+
 56,57   | Number of ASCII associated words       |  2
         +----------------------------------------+
 58,59   | Number of pointers                     |  2
         +----------------------------------------+
 60,61   | Pointer to end of *ROM data            |  2
         +----------------------------------------+
 62,63   | Pointer to end of speech data          |  2
         +----------------------------------------+
 64,65   | Pointer to first word                  |  2 ---+
         +----------------------------------------+        |
 66,67   | Pointer to second word                 |  2     |
         +----------------------------------------+        |
         :          :        :         :          :        |
         :          :        :         :          :        |
         +----------------------------------------+        |
         | Pointer to last word                   |  2     |
         +----------------------------------------+        |
         | Byte &FF                               |  1     |
         +----------------------------------------+        |
         | Word 1 name(s)                         |        |
         +----------------------------------------+        |
         | Data for 1st word:  bottom byte first  | ------+
         +----------------------------------------+
         | Word 2 name(s)                         |
         +----------------------------------------+
         | Data for 2nd word                      |
         +----------------------------------------+
         :          :        :         :          :
         :          :        :         :          :
         +----------------------------------------+
         | Last word name(s)                      |
         +----------------------------------------+
         | Data for last word                     |
         +----------------------------------------+
         | *ROM file data (if present)            |
         +----------------------------------------+
         | Unused space                           |
         +----------------------------------------+
```

PHROM format type

The first two bytes identify the PHROM format type as follows:

&00, &00 - Acorn format with *ROM data
&00, &FF - Acorn format with no *ROM data.

Copyright prefix and string

Bytes at offsets 2 to **4** are used by the Operating System to verify the existence of a speech ROM at a given address. The bytes at offsets 2, 3 and 4 must have the ASCII values for "(", "C" and ")".

The next 49 bytes are used for a text string which is usually a copyright message. The conventional format is:

    Year number **in** ASCII (4 bytes)
    ASCII space (1 byte)
    Company name
    ASCII NUL (1 byte)
    PHROM title
    ASCII NUL (1 byte)
    Version number (4 bytes: format n.mm, e.g. 1.42)
    ASCII NUL (1 byte)
    Padding to 49 bytes with ASCII NUL

Note that the total length of the text strings cannot exceed 48 bytes. For example:

```
"1982    Acorn@Word     PHROM A@1.00@@@@@@@@@@@@@"
```

(where @ denotes ASCII NULL).

PHROM serial number

A 16-bit number allocated by Acorn uniquely identifying each type of PHROM. Serial number 0 indicates that the PHROM is a WORD PHROM.

Number of ASCII associated words

A 16-bit number giving the number of words in the PHROM with an ASCII association.

Number of word pointers

A 16-bit number indicating the number of word pointers present.

Pointer to end of *ROM file data

A 2-byte pointer (lo-byte first) to the end of any *ROM data that is present. This is also the start address of the unused area.

Pointer to *ROM file data

A 2-byte (10-byte first) pointer to the last word. This is also the starting address of any *ROM file data that may be present.

Word pointers

A series of 2-byte pointers (10-byte first), stored in order, to the word data. Note, there may be more pointers than words. This is because some words may be referred to by more than one pointer.

Pointers to words with ASCII association should be located at offset 2*N, where N is the corresponding ASCII code. For example, if the set of words includes the alphabet (as in Word PHROM A), then the pointers to words for "A" to "Z" should be located at offsets 130 (ASCII "A" is 65) to 180 (ASCII "Z" is 90).

Word data

Speech data used by the Speech Processor to create speech. The bytes are reversed so that they are in a suitable format to be directly fed to the speech synthesizer. (Remember that data from RAM had to be reversed before being sent to the Speech Processor - the data in the ROM is already reversed so that this step is unnecessary when using the Speech ROM).

Word name(s)

Each word in the ROM has an associated name or names. The name of the word M is stored in reverse order in decreasing addresses starting below the word data for phrase M. The ASCII code for each letter is first rotated before being stored. Thus ASCII "R", for example, which is &52 (01010010) is stored as &A4 (10100100).

For example, a word called ENTER would appear as:

```
+-----------------------+
|    10100100 = &A4     |  rotates to give 01010010 = "R"
+-----------------------+
|    10001010 = &8A     |  rotates to give 01000101 = "E"
+-----------------------+
|    10101000 = &A8     |  rotates to give 01010100 = "T"
+-----------------------+
|    10011100 = &9C     |  rotates to give 01001110 = "N"
+-----------------------+
|    10001010 = &8A     |  rotates to give 01000101 = "E"
+-----------------------+
|      Word data        |  (reversed order bytes)
|                       |
:                       :
:                       :
```

The existence of the word names allows for another way to access the speech data by reading the ROM (using the Speech processors "Read Byte" command) to find the required word name. This might be done as follows (using the Speech Processor commands):

(1) Use the "Read byte" command to keep reading the ROM until the correct string has been found and identified.
(2) Stop reading the ROM once the correct string has been found. The address register of the ROM will now point to the required speech data.
(3) Use the "Speak" command to speak the required word.

APPENDIX A

FULL LIST OF WORDS AND WORD-PARTS IN WORD PHROM A

This Appendix lists all the words and word-parts which are contained in WORD PHROM A in word/word-part number order.
The first column contains the word and word part numbers
The second column contains the corresponding absolute addresses
The third column contains the corresponding words and word-parts with examples of some other uses for the words

Word or word-part number	**Absolute Address (hex)**	**Word**	**Word or word-part number**	**Absolute Address (hex)**	**Word**
127	250	(0.125)	164	DBD	AN
128	25F	(0.25)	165	E12	AND
129	272	(TONE 1)	166	E8E	ANOTHER
130	2B0	(TONE2)	167	EFD	ANSWER
131	2E9	-D	168	F72	ANY
132	300	-ED	169	FCF	AVAILABLE
133	32B	-ING	170	065	B,BEE,BE
134	361	-S	171	10BD	BAD
135	384	-TEE	172	13B	BETWEEN
136	3DC	-TH	173	1B1	BOTH
137	3F3	-T	174	1F8	BUTTON
138	415	-	175	124D	C,SEE,SEA
139	437	ZERO (0)	176	12B1	CASSETTE
140	4C8	HUNDRED (00)	177	1326	CHARACTER
141	530	THOUSAND (000)	178	139F	COMPLETE
142	5CA	ONE,WON	179	140B	COMPUTER
143	633	TWO,TO,TOO	180	1483	CORRECT
144	692	2- (TWEN-)	181	14F0	D
145	6D2	THREE	182	153B	DATA
146	73F	3- (THIR-)	183	159F	DATE
147	788	FOUR,FOR,FORE	184	15E6	DO
148	7F7	4- (FOR-)	185	162A	DOLLAR
149	840	FIVE	186	167E	DONT
150	8B6	5- (FIF-)	187	16D7	DOWN
151	8E9	SIX	188	175A	E
152	936	6- (SIX-)	189	178B	EACH
153	96E	SEVEN	190	17D6	ELEVEN
154	9CB	7- (SEVEN-)	191	1852	ENGAGED
155	A12	EIGHT,ATE	192	18FA	ENTER
156	A46	8- (EIGHT-)	193	195C	ERROR
157	A71	NINE	194	19AA	ESCAPE
158	AFD	9- (NIN-)	195	1A1C	F
159	B5D	A	196	1A5D	FEW
160	BB6	ACORN	197	1AAB	FILE
161	C4B	AFTER	198	1B1E	FIRST
162	CB3	AGAIN	199	1896	FOUND
163	D46	AMOUNT	200	1C11	FROM

APPENDIX A continued

Word or word-part number	Absolute Address (hex)	Word
201	1C6B	G
202	1CC9	GOOD
203	1D09	H
204	1D60	HAVE
205	1DCC	I,EYE
206	1E3E	ILLEGAL
207	1EBD	IN-
208	1EFC	INPUT
209	1F57	IS
210	1FA6	J,JAY
211	2008	K
212	2061	KEY
213	209F	L
214	20FC	LARGE
215	2195	LAST
216	21F5	LINE
217	226F	M
218	22CD	MANY
219	2321	MINUS
220	239D	MORE
221	2409	MUST
222	246F	N
223	24C9	NAME
224	2544	NEGATIVE
225	25DE	NEW
226	263D	NO,KNOW
227	269D	NOT,KNOT
228	26F7	NOW
229	276A	NUMBER
230	27E5	O
231	282D	O'CLOCK
232	2892	OF
233	28D9	OFF
234	2923	OLD
235	2980	ON
236	29DE	ONLY
237	2A48	OR
238	2A90	P,PEA
239	2AC7	PARAMETER
240	2B58	PENCE
241	2BBA	PLEASE
242	2C45	PLUS
243	2C8B	POINT
244	2CE6	POSITIVE
245	2D64	POUN-

Wordor word-part number	Absolute Address (hex)	Word
246	2DCD	PRESS
247	2E3F	PROGRAM
248	2EC8	Q,QUEUE
249	2F1E	R,ARE
250	2F59	RED
251	2FD1	RESET
252	3051	RETURN
253	30E7	RUN
254	3153	RUNNING
255	31E1	S
256	3231	SAME
257	329A	SCORE
258	3320	SECOND
259	339A	SMALL
260	341F	START
261	349F	STOP
262	34FB	SWITCH
263	3573	T,TEA,TEE
264	35CA	TEN
265	362B	THANK
266	3684	THAT
267	36DF	THE
268	3724	THEN
269	377E	THIRD
270	3808	THIS
271	3874	TIME
272	38E2	TRY
273	3953	TWELVE
274	39D2	TYPE
275	3A21	U,YOU
276	3A91	UH
277	3AC0	UP
278	3AF7	V
279	3B5C	VERY
280	3BB8	W
281	3C1D	WANT
282	3C6C	WAS
283	3CC2	WERE
284	3D1A	WHAT
285	3D6E	WHICH
286	3DB5	X
287	3E11	Y,WHY
288	3E86	YEAR
289	3EFC	YES
290	3F3D	YOUR
291	3F9B	Z

APPENDIX B

LIST OF WORDS AND WORD-PARTS IN WORD PHROM A
WITH CORRESPONDING ASCII CHARACTER

Word or word-part number	ASCII character	Word	Word or word-part number	ASCII character	Word
32	space	(0.125)	80	P	P,PEA
33	!	TEN	81	Q	Q,QUEUE
34	"	2-	82	R	R,ARE
35	#	3-	83	S	S
36	$	4-	84	T	T,TEA,TEE
37	%	5-	85	U	U,YOU
38	&	6-	86	V	V
39	'	7-	87	W	W
40	(	8-	88	X	X
41	)	9-	89	Y	Y,WHY
42	*	TIME	90	Z	Z
43	+	PLUS	91	[	START
44	,	THOUSAND (000)	92	\	OFF
45	-	MINUS	93	]	STOP
46	.	POINT	94	^	THANK
47	/	ON	95	_	LINE
48	0	ZERO (0)	96	`	POUN-
49	1	ONE,WON	97	a	AND
50	2	TWO,TO,TOO	98	b	BAD
51	3	THREE	99	c	CORRECT
52	4	FOUR,FOR,FORE	100	d	-D
53	5	FIVE	101	e	-ED
54	6	SIX	102	f	FILE
55	7	SEVEN	103	g	GOOD
56	8	EIGHT,ATE	104	h	HUNDRED (00)
57	9	NINE	105	i	IN-
58	:	(TONE1)	106	j	-ING
59	;	(TONE2)	107	k	KEY
60	<	SMALL	108	1	ILLEGAL
61	=	IS	109	m	MUST
62	>	LARGE	110	n	NO,KNOW
63	?	WHAT	111	o	ONLY
64	@	AMOUNT	112	p	PRESS
65	A	A	113	q	OF
66	B	B,BEE,BE	114	r	RETURN
67	C	C,SEE,SEA	115	s	-S
68	D	D	116	t	THE
69	E	E	117	u	YOUR
70	F	F	118	v	VERY
71	G	G	119	w	WHICH
72	H	H	120	x	NUMBER
73	I	I,EYE	121	y	YES
74	J	J,JAY	122	z	-Z
75	K	K	123	{	FIRST
76	L	L	124	|	OR
77	M	M	125	}	LAST
78	N	N	126	~	NOT
79	0	0			

APPENDIX C

SOME EXAMPLES OF COMPOUND WORDS FROM WORD PHROM A

This Appendix provides a list of some of the words which can be made up from words and word-parts available in Word PHROM A. No doubt you will think of many more!

ALIGN	UH LINE	LASTED	LAST -D
ALIGNS	UH LINE -Z	LASTING	LAST -ING
ANSWERING	ANSWER -ING	LINED	LINE -D
ANSWERS	ANSWER -Z	MINUSES	MINUS -Z
ARE	R	MOOR	MORE
BEES	B -Z	MOORS	MORE -Z
BUTTONING	BUTTON -ING	NAMES	NAME -Z
BUTTONS	BUTTON -Z	NAMING	NAME -ING
CASSETTES	CASSETTE -S	NEWS	NEW -Z
CHARACTERS	CHARACTER -Z	NOSE	NO -Z
COMPLETES	COMPLETE -Z	NUMBERED	NUMBER -D
DATES	DATE -S	NUMBERS	NUMBER -Z
DEED	D -D	ONCE	ONE -S
DEEDS	D -D -Z	PEA	P
DOING	DO -ING	PEAS	P -Z
DOLLARS	DOLLAR -Z	PLEASES	PLEASE -Z
DOWNS	DOWN -Z	POUND	POUN- -D
EASE	E -Z	POUNDS	POUN- -Z
END	N -D	PROGRAMMING	PROGRAM -ING
ENTERED	ENTER -ED	PROGRAMS	PROGRAM -Z
ENTERING	ENTER -ING	QUEUE	Q
ENTERS	ENTER -Z	SCORES	SCORE -S
ERRORS	ERROR -Z	SECONDS	SECOND -Z
ESCAPES	ESCAPE -Z	SEETHE	C -TH
EYE	I	SIR	-S UH
EYES	I -Z	SKI	-S KEY
FEUD	FEW -D	SWITCHES	SWITCH -S
FILES	FILE -Z	SWITCHING	SWITCH -ING
FILED	FILE -D	TEA	T
FOR,FORE	FOUR (4)	TEASE	T -Z
FUSE	FEW -Z	TEETH	T -TH
GOODS	GOOD -Z	TEND	TEN -D
HAVING	HAVE -ING	THANKING	THANK -ING
INBETWEEN	IN- BETWEEN	THANKS	THANK -Z
INCOMPLETE	IN -COMPLETE	TIMED	TIME -D
INCORRECT	IN- CORRECT	USE	U -Z
INDEED	IN- D -D	WANTING	WANT -ING
INNER	IN- UH	WANTS	WANT -Z
INTEND	IN- TEN -D	WISE	Y -Z
JAY	J	WITCHES	WHICH -Z
JAYS	J -Z	WORSE	WERE -Z
KEYS	KEY -Z	YAW	YOUR
KNEW	NEW	YOURS	YOUR -Z

APPENDIX D

SPEECH PARAMETER DATA CODING

The entries in the bulk of the table below are the actual parameter values, the entries under "Coded value" are the values recognised by the Speech Processor and stored in speech ROMs.

Each entry for K1 to K10 is the real value multiplied by 512. For example, the first value under K1 is -501. This represents -501/512 = -0.9785.

The pitch is expressed as a number of samples. For example, coded pitch 44 decodes to a value of 78. This means that a pitch excitation pulse is provided every 78 sample periods. At 8KHz, the sample period is 1/(8KHz) or 125 microseconds. The pitch period is thus 78*125 = 9.75 milliseconds, which corresponds to a frequency of 102Hz.

Energy	Pitch	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	Coded Value
0	0	-501	-328	-441	-328	-328	-256	-308	-256	-256	-205	0
52	15	-498	-303	-387	-273	-282	-212	-260	-161	-176	-132	1
87	16	-497	-274	-333	-217	-235	-168	-212	-66	-96	-59	2
123	17	-495	-244	-279	-161	-189	-123	-164	29	-15	14	3
174	18	-493	-211	-225	-106	-142	-79	-117	124	65	87	4
246	19	-491	-175	-171	-50	-96	-35	-69	219	146	160	5
348	20	-488	-138	-117	5	-50	10	-21	314	226	234	6
491	21	-482	-99	-63	61	-3	54	27	409	307	307	7
694	22	-478	-59	-9	116	43	98	75				8
981	23	-474	-18	45	172	90	143	122				9
1385	24	-469	24	98	228	136	187	170				10
1957	25	-464	64	152	283	182	232	218				11
2764	26	-459	105	206	339	229	276	266				12
3904	27	-452	143	260	394	275	320	314				13
5514	28	-445	180	314	450	322	365	361				14
7789	29	-437	215	368	506	368	409	409				15
	30	-412	248									16
	31	-380	278									17
	32	-339	306									18
	33	-288	331									19
	34	-227	354									20
	35	-158	374									21
	36	-81	392									22
	37	-1	408									23
	38	80	422									24
	39	157	435									25
	40	226	445									26
	41	287	455									27
	42	337	463									28
	44	379	470									29
	46	411	476									30
	48	436	506									31

Energy	Pitch	K1	K2	K3	K4	KS	K6	K7	K8	K9	K10	Coded Value
	50											32
	52											33
	53											34
	56											35
	58											36
	60											37
	62											38
	65											39
	68											40
	70											41
	72											42
	76											43
	78											44
	80											45
	84											46
	86											47
	91											48
	94											49
	98											50
	101											51
	105											52
	109											53
	114											54
	118											55
	122											56
	127											57
	132											58
	137											59
	142											60
	148											61
	153											62
	159											63

APPENDIX E

REFERENCES

TMS 6100 VOICE SYNTHESIS MEMORY DATA MANUAL TMS 5220 VOICE SYNTHESIS PROCESSOR DATA MANUAL

both these manuals are available from

Texas Instruments Limited
Manton Lane
BEDFORD MK41 7PA

ACORN
COMPUTER

Acorn Computers Limited, Fulbourn Road, Cherry Hinton, Cambridge CB1 4JN, England