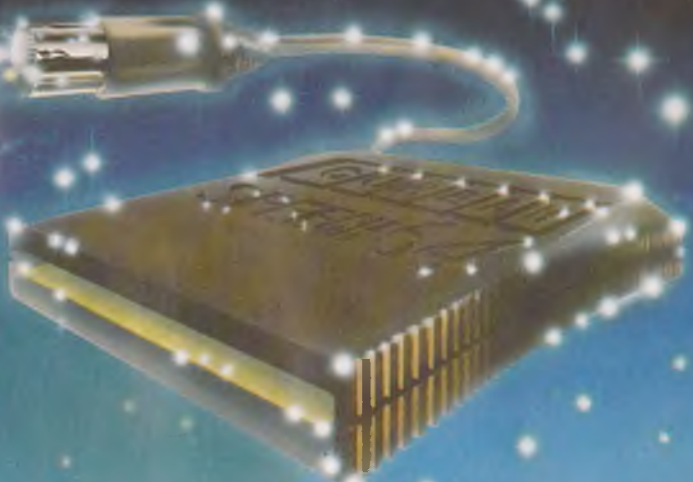
CURRAH
Computor Components Ltd



SPEECH 64
Programming Manual

REMEMBER !

1 Make sure all peripherals connected to the computer (e.g. printers, diskdrives, etc) are turned on before using SPEECH 64.

2 Use monitor adaptor if a monitor is hooked up to your C-64.

3 Your basic program will not execute any command after the say command if it is on the same line. Simply put your desired command on a new line.

CURRAH SPEECH 64
Programming Manual

By Mark Anson.
Machine code programming techniques in Chapter 6 by David Brown.

Edition 1, Revision 3, 1984

We would like to acknowledge the generous assistance furnished by
General Instrument Microeletronics in the UK and USA
in the development of this product.

WARNING

This unit must be used in accordance with these instructions. Do not attempt to use this unit with 'switchable' motherboards etc. which selectively cut off the power supplies to the slots. NEVER INSERT OR REMOVE THIS UNIT WHILE THE POWER TO THE COMPUTER IS ON. Failure to follow these instructions may result in permanent damage to the speech unit or the computer.

1. SETTING UP THE SPEECH 64

The CURRAH SPEECH64 is an allophone speech synthesiser, which means that it uses individual speech sounds strung together to make intelligible speech. Unlike some speech synthesisers which have a fixed vocabulary, the SPEECH64 has an unlimited vocabulary and can synthesise any word or sentence in the English language. SPEECH64 incorporates a Text-to-Speech conversion system which will convert most English text into speech automatically. Furthermore, the user can access the allophones directly for difficult words and the unit features both a high and a low voice, each with programmable intonation.

To set up the unit, remove it from its packing tray if you have not already done so, **SWITCH OFF THE CBM64**, and plug the unit into the **CARTRIDGE EXPANSION PORT** at the back of the computer, **CURRAH logo uppermost**. Now take the DIN plug which comes from the SPEECH64 unit and plug it into the **AUDIO / VIDEO SOCKET** at the back of the CBM64 — this is the DIN socket which is nearest to the expansion port — you can't get the DIN plug into the wrong socket as they are physically different. This lead allows the speech generated by the SPEECH64 to be fed into the computer's sound circuitry and come from the TV speaker.

Switch on the CBM64. The computer will initialise normally. Turn up the volume on your TV set (about half-way will do) and type in:

INIT (and **RETURN**)

A copyright message will now appear and SPEECH64 will now 'voice' any key pressed on the keyboard. For instance, if you press **RETURN** the unit will say 'return'. Try pressing some more keys. Note that all graphic characters are voiced as 'graphic', whatever their shape. If you type very fast then you will find that SPEECH64 always tries to speak the latest key depression. The unit will always voice the keys with the high voice after being INITialised, however if you wish to use the low voice then type:

KON 0 (and **RETURN**)

SPEECH64 will now voice any key pressed on the keyboard with the low voice. If you wish to use the high voice again then type:

KON 1 (and **RETURN**) or just **KON** (and **RETURN**)

Apart from being useful in themselves, the keyvoices serve to illustrate what can be achieved by allophone speech synthesis. Before going on to the next section, you may wish to turn off the keyvoices again. This time, try putting **KOFF** inside a simple program, like this:

```
10 KOFF
```

When you **RUN** this, the keyvoices will be disabled again. Both **KON** and **KOFF** can be used in a BASIC Program line.

2. SIMPLE WORDS AND PHRASES.

SPEECH64 contains a Text to Speech Interpreter — the unit's internal software scans inputted text and converts it automatically to the speech sounds required. Simply type in **SAY** followed by the words you want spoken, enclosed in quotation marks, like this (you can use upper or lower case letters):

SAY "HELLO"

When you press **ENTER,** the unit will say 'hello'. Now type in:

SAY "HELLO, MY NAME IS SPEECH SIXTY FOUR"

and the unit will say what you have enclosed in the quotation marks. You can use the SAY command just like the **KON** and **KOFF** commands — try this short program:

10 SAY "THIS IS A BASIC PROGRAM"

Note carefully what happens when you **RUN** it. Did you see how the program finished before the speech stopped? The speech allophones are 'queued' before output in an 'interrupt-driven buffer' — this is some jargon which basically means that your BASIC program is not slowed down when it is outputting speech, and can in fact be 'finished' whilst the voice continues to say what it has been told to say.

Try entering some sentences of your own now. Note that punctuation marks (commas, apostrophes, colons, periods etc.) are all interpreted as pauses of various length but where a period occurs before a digit (e.g. 3.4) it is spoken as 'point'. The mathematical symbols +, −, / and * are pronounced as 'plus', 'minus', 'divide by' and 'multiply' respectively.

The buffer can hold up to 256 allophones — or approximately 30 seconds worth of speech — at any moment, and if SPEECH64 finds that there is no room for a new word or phrase to be added, the phrase will be ignored. You can get round this by using a delay loop in your program or by examining the buffer — but more of this later.

If you break into a program while it is running, any speech in progress is stopped and the buffer automatically emptied. This is to ensure that you will not have to wait for the buffer to empty itself — 30 seconds of unwanted speech can be quite tedious!

3. USING THE ALLOPHONES DIRECTLY.

Turning text into speech is quite a difficult procedure as there are many spelling-pronunciation irregularities. For instance, 'plough' and 'cough' are spelled similarly but the 'ough' component is spoken differently. SPEECH64 uses a rule-oriented system which can cope with most common spelling-pronunciation irregularities. There are some words which it cannot cope with as they do not obey any logical rules. You will probably fairly quickly find words that SPEECH64 will pronounce wrongly, depending on how determined you are to defeat it, but do not worry as it is very easy to get SPEECH64 to say even the most illogically-pronounced words.

Suppose you wanted to make the unit say 'Hawaii'. If you typed in **SAY 'HAWAII'** then SPEECH64 would respond with a word that would sound rather like 'Haway-i'. You could try deliberately misspelling the word, like this (the apostrophe will help to emphasise the 'EE' part):

SAY "HAWHY 'EE"

And this will work very well, but there is another way. All the speech in SPEECH64 is generated using 'allophones'. The Text to Speech routines convert text into the relevant allophones for output, but you can instruct the unit to say allophones directly. There are 58 speech sounds to choose from and four pauses of variable length, and you make words by stringing together the allophone symbols (the 'mnemonics' for the allophones) inside a **SAY** command. There are some points to note. Firstly, some of the mnemonics are single letters and some are several letters enclosed in ROUND BRACKETS — see the following table. Secondly, the **SAY** command must be informed that you are using allophones directly and you do this by enclosing the string of mnemonics inside SQUARE BRACKETS.

Suppose you wanted to make a single 'ai' sound (as in the 'ai' in 'Hawaii'). The allophone mnemonic for this sound (see the table below) is (ii). To make SPEECH64 say this sound, simply use the **SAY** command again, but enclose the allophone mnemonic you require inside SQUARE BRACKETS, like this:

SAY "[(ii)]"

Some allophones "run together" well — try SAY"[aaaaaa]". Other allophones that exhibit this property are e,i,o,u,f,s,(eh), (th) and (uh).

The unit will say something which sounds like 'eye'. Try this:
SAY "[haw(ii)(ee)]"

The unit will say 'Hawaii'. From the table given below, you can make up any word in the English language. Here is a complete list of all the allophones, and the four pauses.

Mnemonic	**Sounds like**	**Mnemonic**	**Sounds like**
Letters a-z	'phonetic'	(ar)	'ar' in arm
(excluding	like a in 'cat'	(aer)	'air' in repair
q and x)	'b' in 'tab' etc.	(ch)	'ch' in church
		(ck)	'ck' in clock
(aa) or (ay)	'ay' in hay	(ear)	'ear' in clear
(ee)	'ee' in see	(eh)	'ar' in wary
(ii)	'i' in hive	(er)	'er' in leader
(oo) or (eau)	'o' in stove	(err)	'ur' in purr
		(ng)	'ng' in tongue
(bb)	'b' in bat	(or)	'or' in sore
(dd)	'd' in do	(ou)	'oo' in root
(gg)	'g' in got	(ouu)	'oo' in food
(ggg)	'g' in big	(ow)	'ow' in now
(hh)	'h' in hoe	(oy)	'oy' in boy
(ll)	'l' in let	(sh)	'sh' in ship
(nn)	'n' in no	(th)	'th' in thin
(rr)	'r' in ruin	(dth)	'th' in then
(tt)	't' in to	(uh)	'oo' in took
(yy)	'y' in yeah	(wh)	'wh' in which
		(zh)	'z' in azure

' (apostrophe)	very short pause
SPACE	pause between words
, (comma)	pause between phrases
. (period)	pause between sentences

The double-letter allophones (bb) to (yy) inclusive are used where extra emphasis is required e.g. at the start of words. To illustrate this, try comparing SAY "[david]" to SAY "[(dd)avid]". The single-letter allophones are much softer.

Here are some examples of words made using allophones:

he(ll)(oo)	'hello'
(gg)(ou)(dd)b(ii)	'goodbye'
welkum	'welcome'
m(or)ni(ng)	'morning'
aft(er)n(ouu)n	'afternoon'
(ee)vni(ng)	'evening'
in(gg)lund	'England'
y(ouu)n(ii)(tt)id st(aa)ts	'United States'
compy(ouu)(tt)(er)	'computer'
sp(ee)(ch) sin(th)us(ii)z(er)	'speech synthesiser'
penisilin	'penicillin'
ny(ou)kl(ear)	'nuclear'
(oy)l	'oil'
electrisit(ee)	'electricity'

See if you can work out what this will sound like: (dth)u (bb)est w(ay) ov l(er)ni(ng) ab(ow)t aluf(oo)nz iz (tt)(uh) (tt)r(ii) m(aa)ki(ng) w(er)dz up y(or)self. (or)l(th)(oo) y(ouu) wil m(aa)k mist(aa)ks at f(er)st y(ouu) wil kwikl(ee) get (dth)u ha(ng) ov it and y(ouu) wil hav (aa) lot ov fun l(er)ni(ng).

Some more examples of the use of allophones can be found in Appendix II. Remember when using allophones that you have to think how words are spoken, rather than how they are written. It often helps to slowly speak the word you wish to pronounce, breaking it down into the individual speech sounds. Try making up some words now using allophones and compare your efforts with those of SPEECH64's text to speech routines. Note that if you make a syntax error (i.e. use an allophone that doesn't exist) the allophones will be ignored and the message:

? SYNTAX ERROR or **? SYNTAX ERROR IN XXX**

will be displayed, where **XXX** is the line number where the error occurred. SPEECH64 will, however, speak up to where the error occurred.

The really interesting thing about using allophones is that you can mix up normal text with allophones in a single **SAY** command, using the square brackets, like this:

SAY "I come from [haw(ii)(ee)]"

Note also that the **SAY** command can be assigned a string variable or an element of a string array — see Appendix III for examples of this.

4. USING BOTH VOICES AND INTONATION

So far you have just been using one voice to output speech. The **SAY** command uses Voice 1 (the higher one) by default, but you can use the lower voice by using **SAY 0**, like this:

```
SAY 0 "MY VOICE IS DEEPER"
```

SAY 1 uses voice 1, but as stated above the **SAY** command defaults to voice 1 anyway, so you will hardly ever need this. You can use both voices in a program, like this:

```
10 SAY "VOICE 1"
20 SAY 0 "VOICE 0"
```

Each voice has programmable intonation on the individual allophones. Intonation is helpful in allophonic speech as it adds expression and 'life' to what is otherwise rather robotic speech. Due to the complexity of the text-to-speech routines, you can only use intonation when you are using the allophones directly. You access intonation by using upper and lower case letters for the mnemonics — upper case is intoned UP, whilst lower case is not intoned. Try this to see what intonation sounds like.

SAY "[aaAAaaAAaaAA]" and **SAY "[hE(ll)(OO)]"**

And also:

SAY 0 "[aaAAaaAAaaAA]" and **SAY "[(gg)(OU)(dd)b(ii)]"**

You can see intonation being used in the example program in Appendix II. Remember that intonation will be most effective on vowels but you should try by experimenting with it to see what effects you can get. You might like to try adding intonation to the examples in the previous chapter.

An interesting effect is to use intonation and the two voices to get four different monotonic voices, like this:

```
10 SAY 0 "[he(ll)(oo)]"
20 SAY 0 "[HE(LL)(OO)]"
30 SAY 1 "[he(ll)(oo)]"
40 SAY 1 "[HE(LL)(OO)]"
```

5. THE SPEECH BUFFER

The speech buffer can hold 256 allophones, or about 25 seconds worth of speech. Normally this will not bother you, but if you wish to say a long sentence you may find that you can fill up the buffer faster than it can empty. The buffer can only empty as fast as SPEECH64 can speak the words, but BASIC can fill up the buffer in a few milliseconds. So what do you do if you have a long sentence — say one which when interpreted will generate more than 256 allophones?. To help you, SPEECH64 maintains a BASIC variable, **SP%**, which informs you how much space there is left in the buffer at any time. Try this program to see how the buffer fills up:

```
10 SAY "HELLO"
20 PRINT "□ " (shifted HOME)
30 PRINT SP%
40 IF SP%<6 THEN 40
50 GOTO 10
```

RUNning this program will show the decreasing amount of space left in the buffer as line **10** fills it up with 'hello's". Line **40** is a wait loop for the buffer to empty if there is no room at all. Remember that if SPEECH64 cannot fit all the allophones in for a particular word, it will put it what it can and lose the rest. So if you find your sentences are being "clipped" then simply insert a suitable wait loop for the buffer to empty, like this:

40 IF SP%<N THEN 40 (Where N is the number of allophones you guess the sentence will require)

Guessing the number of allophones a word will require is very easy if you have grasped the method of using allophones in Chapter 3, but as a very rough rule, allow one allophone for every letter in the input text, PLUS one. If you are lazy, you could get the computer to do this for you:

```
10 INPUT A$
20 A=LEN (A$)
30 IF SP%<(A+1) THEN 30
40 SAY A$
```

6. USING MACHINE CODE WITH THE UNIT

Before you can even think about programming the unit to speak in machine code, you must first work out which allophones you are going to use and convert them manually into the hexadecimal codes required, using the table in Appendix 1.

Each allophone is stored in the buffer as a 8-bit byte. Bits 0 to 5 are for the allophone code (decimal 0 to 63), bit 6 is for the voice, and bit 7 is for intonation. Bit 6 is set for Voice 1 and cleared for Voice 0, whilst bit 7 is set for intonation up and cleared for no intonation.

Now you can determine what the bytes are by looking up the relevant codes in the Appendix. For instance, the word 'hello' in Voice 0, with the 'e' intoned up comes out like this (in hex notation):

h, voice 0, no intonation = 1B+40 = 5B
e, voice 0, intonation UP = 07+40+80 = C7
(ll), voice 0, no intonation = 3E+40 = 7E
(oo), voice 0, no intonation = 35+40 = 75

There are three ways of programming SPEECH64 from machine code. The first two ways need the unit to be initialised from machine code. This can be achieved using the following program:

```
INIT    SEI           ;Disable interrupts
        LDA $A7F0     ;Select SPEECH64 ROM
        JSR $A121     ;Call set up routine
        LDA $A7F0     ;Disable SPEECH64 ROM
        CLI           ;Enable interrupts
*       LDA $02BE
*       ORA #$20
*       STA $02BE
        RTS
```

*These 3 lines turn the SPEECH64 BASIC commands on, but if you are not going to use SPEECH64 from BASIC then these lines can be omitted.

After initialisation you may want the keyvoices turned off — simply set bit 1 of $02BE (e.g. **LDA $02BE, ORA #$02, STA $02BE**) and if you want them turned on again subsequently simply clear bit 1 of $02BE with an **AND** instruction. The voice which the keyvoices use is controlled by bit 3 of $02BE — if bit 3 is set then the high

voice is used, if bit 3 is cleared then the low voice is used. Remember not to disturb the rest of the bits in $02BE. You can now begin to program the unit to speak using any of the following methods:

1. Using the Speech Buffer

The allophone which you want voiced must be put into the next available space in the 256 byte buffer, which is pointed to by the contents of $CE64, added to the buffer start address of $CF00. After an allophone has been put into the buffer the contents of $CE64 should be incremented by one.

Before putting an allophone into the buffer you should check that there is enough space. This can be done by looking at $CE5F which gives the number of free bytes in the buffer, PLUS ONE i.e. you should decrement by one to get the correct number of free bytes. The voice to be used is selected by bit 6 of each allophone byte, and intonation by bit 7, as explained above.

Example — to say 'hello'

```
        JSR INIT          ;Call initialisation routine
        LDY #$00
WAIT    LDX $CE5F         ;Is buffer full?
        DEX
        BEQ WAIT          ;Wait if so
        LDA DATA,Y        ;else get allophone
        BEQ FINISH        ;exit if finished
        LDX $CE64
        STA $CF00,X       ;Store allophone in buffer
        INC $CE64         ;and update pointer
        INY               ;Next allophone
        JMP WAIT

FINISH  RTS

DATA    DB $5B,$C7,$7E,$75 ;The four data bytes
        DB 0               ;End marker
```

2. Using the Text to Speech buffer

To use the Text to Speech routines from machine code you must store your text in the buffer at $CE70. Up to 80 characters may be stored in the buffer. The last character of your text MUST be a null byte (OOH) for correct operation. ONLY ASCII CODES $20 TO $60 SHOULD BE USED.

When all your text has been stored in the buffer you can initiate the conversion process by making the contents of $CE63 non-zero. The allophones generated in the process are automatically transferred to the speech buffer so you do not have to worry about them.

The voice (0 or 1) is controlled by bit 0 of $02BE. If bit 0 is set then the low voice will be used, if cleared then the high voice will be used.

Intonation is not possible in Text to Speech.

Example: To say "this is a demonstration of text to speech conversion', using the low voice.

```
            JSR INIT        ;call initialisation routine
            LDA $02BE
            AND #$FE
            STA $02BE       ;select low voice
NEXT        LDA DATA,X      ;get character
            STA $CE70,X     ;and store it in buffer
            BEQ FINISH      ;if last char (null byte)
            INX
            JMP NEXT
FINISH      INC $CE63       ;start conversion
            RTS
DATA        DB 'THIS IS A DEMONSTRATION OF TEXT TO
              SPEECH CONVERSION'
            DB 0
```

Remember to check that there is sufficient room in the speech buffer to hold the allophones which the Text to Speech routine generates — once again you could check the contents of $CE5F to see how much room there is left in the buffer.

3. Using the Speech Chip Directly.

In this mode there is no need to initialise the unit using the **INIT** routine detailed above. The allophones are sent to location $DE00 for the high voice and location $DE01 for the low voice. The status of the speech chip is found by reading location $DE00 - if bit 7 is set then the speech chip is ready for another allophone. If intonation is required on an allophone then bit 7 should be set in the byte sent to the speech chip. Note that bit 6 is irrelevant as regards which voice is used — the voice is determined by which location you send the byte to. The volume on the C64's sound

chip should be set to a suitable level to allow the speech to be heard. (This function is performed automatically using the **INIT** program detailed above).

Example: To say 'hello' with the low voice and intonation on the 'e':

```
         LDA #$0F
         STA $D418        ;set sound to maximum
         LDX #$00
STATUS   LDA $DE00        ;is speech chip ready?
         BPL STATUS       ;wait if not
         LDA DATA,X       ;else get allophone
         STA $DE01        ;and send to low voice
         BEQ FINISH       ;end if last allophone
         INX
         JMP STATUS       ;get next one
FINISH   RTS
DATA     DB $1B, $87, $3E, $35
         DB 0             ;end of data
```

Notes on the Hardware of the SPEECH64

1. If the **IRQ** is disabled then any speech being outputted will cease or 'hang' on an allophone. This is because the **IRQ** is responsible for outputting allophones to the speech chip from the speech buffer. The **IRQ** is used for this task so as not to slow down the computer.
2. The SPEECH64 unit uses the **IRQ** 'vector' at $0314, which points to the SPEECH64's **IRQ** routines. Should you require the **IRQ**, use the SPEECH64 'vector' at $CE68 (just like the Commodore 'vector' at $0314) to point to your routine, which should be terminated by a **JMP** to $EA31.
3. SPEECH64 uses the following areas of memory:
   $02A7 to **$02BF**
   $CE5E to **$CEFF**
   $0314 to **$0315**
   $00FB to **$00FC**

APPENDIX I
Decimal and Hex Codes for the Allophones

Allophone	Decimal	Hex	Allophone	Decimal	Hex
a	24	18	(aa)/(ay)	20	14
b	28	1C	(ee)	19	13
c	8	08	(ii)	6	06
d	21	15	(oo)/(eau)	53	35
e	7	07			
f	40	28	(bb)	63	3F
g	36	24	(dd)	33	21
h	27	1B	(gg)	61	3D
i	12	0C	(ggg)	34	22
j	10	0A	(hh)	57	39
k	42	2A	(ll)	62	3E
l	45	2D	(nn)	56	38
m	16	10	(rr)	14	0E
n	11	0B	(tt)	13	0D
o	23	17	(yy)	25	19
p	9	09			
r	39	27	(ar)	59	3B
s	55	37	(aer)	47	2F
t	17	11	(ch)	50	32
u	15	0F	(ck)	41	29
v	35	23	(ear)	60	3C
w	46	2E	(eh)	26	1A
y	49	31	(er)	51	33
z	43	2B	(err)	52	34
			(ng)	44	2C
			(or)	58	3A
			(ou)	22	16
			(ouu)	31	1F
			(ow)	32	20
			(oy)	5	05
			(sh)	37	25
' (apostrophe)	1	01	(th)	29	1D
(space)	3	03	(dth)	18	12
, (comma)	4	04	(uh)	30	1E
. (full stop)	This is a hybrid of two pauses (04 + 04)		(wh)	48	30
			(zh)	38	26

Note that these only add up to 62 allophones. The two remaining ones are not implemented in the interpreter and are unlikely to be useful, although you can try them if you want — code 0 gives a 10ms pause, whilst code 54 (36 Hex) gives an allophone practically identical to the (dth) allophone.

APPENDIX II — AN EXAMPLE PROGRAM

THE SPEAKING CLOCK by D. BROWN

This program will speak the time in TI$ using the high voice. (Don't forget to initialise TI$). The program makes use of allophones, text to speech and intonation.

```
10   INIT
20   DIM T$(25)
30   T$( 1)="[won]"
40   T$( 2)="[(tt)(OUU)]"
50   T$( 3)="[(th)(er)(EE)]"
60   T$( 4)="[f(OR)]"
70   T$( 5)="[f(II)v]"
80   T$( 6)="[slks]"
90   T$( 7)="[sev(ER)n]"
100  T$( 8)="[(AA)(tt)]"
110  T$( 9)="[n(II)n]"
120  T$(10)="[(tt)En]"
130  T$(11)="[(ee)levun]"
140  T$(12)="[(tt)welv]"
150  T$(13)="[(th)(ER)t(EE)n]"
160  T$(14)="[f(OR)t(EE)n]"
170  T$(15)="[fift(EE)n]"
180  T$(16)="[sikst(EE)n]"
190  T$(17)="[sevEnt(EE)n]"
200  T$(18)="[(AA)t(EE)n]"
210  T$(19)="[n(II)nt(EE)n]"
220  T$(20)="[(tt)wEnt(EE)]"
230  T$(21)="[(th)(ER)t(EE)]"
240  T$(22)="[f(or)t(EE)]"
250  T$(23)="[fift(EE)]"
260  SAY"THE TIME IS"
270  H=VAL(MID$(TI$,1,2))
280  M=VAL(MID$(TI$,3,2))
290  S=VAL(MID$(TI$,5,2))
300  IF H>12 THEN H=H-12
310  IF H=0 THEN H=12
320  X=H : GOSUB 1000
330  IF M>0 THEN 360
340  SAY "O CLOCK"
350  GOTO 370
360  X=M : GOSUB 1000
370  IF S>0 THEN 400
380  SAY"[pr(ee)s(ii)s](ee)]"
390  GOTO 440
400  SAY "AND"
410  X=S : GOSUB 1000
420  SAY "[secund]"
430  IF S>0 THEN SAY "[S]"
440  FOR A=0 TO 5000 : NEXT
450  GOTO 260
1000 IF X-50>=0 THEN X=X-50 :
     A$=T$(23) : GOTO 1060
1010 IF X-40>=0 THEN X=X-40 :
     A$=T$(22) : GOTO 1060
1020 IF X-30>=0 THEN X=X-30 :
     A$=T$(21) : GOTO 1060
1030 IF X-20>=0 THEN X=X-20 :
     A$=T$(20) : GOTO 1060
1040 SAY T$(X)
1050 RETURN
1060 SAY A$
1070 IF X=0 THEN RETURN
1080 GOTO 1040
```

APPENDIX III — A Description of the Extra Basic Commands for reference purposes

GENERAL

All these extra commands can be used in a multi-statement line, but note that any statements FOLLOWING one of these commands on a line will be ignored — rather like the BASIC REM statement.

SAY

Format: **SAY n** (user string)
SAY n (string variable)
SAY n (element of string array)

Where n is either 0 or 1. If n is not specified it defaults to 1. Restrictions: User string must be enclosed in quotation marks. String concatenation is not allowed; the input string must be a single string, element of a string array, or string variable.

Examples of valid use of **SAY** command:

```
SAY "hello"
SAY 0 A$
SAY 1 V$ (1,3)
SAY 0 R$ (2,3,6)
```

INIT

Format: **INIT**

Initialises Speech64 operating system and issues CURRAH copyright message. The copyright message is not issued if INIT is invoked in a program line.

BYE

Format: **BYE**

Temporarily suspends Speech64 operating system. No messages issued. BASIC runs as normal and the Extended BASIC commands will be processed as errors, until INIT is again invoked.

KON

Format: **KON n**

Where **n** is either 0 or 1. If **n** is not specified it defaults to 1. Enables the automatic key voicing in either high or low voices.

Examples:

```
KON
KON 0
KON 1
```

KOFF

Format: **KOFF**

Disables automatic key voicing. (Note that **KOFF 0** and **KOFF 1** are accepted but you will probably never need this)

WELWYN CURRAH WARRANTY

This product is warranted against defects for one year from date of purchase from appointed CURRAH dealers. Within this period it will be replaced or repaired free of charge. **Within 30 days of purchase**, simply return it to the place of purchase with your sales receipt. **After 30 days from date of purchase**, please return it with your sales receipt to:

WELWYN CURRAH, 104 West Fourth Street, Suite 208-9, Royal Oak, Michigan 48067.

Warranty does not cover transportation costs. Nor does it cover accidental damage or a unit used contrary to the operating instructions or any damage caused as a consequence of such misuse.

EXCEPT AS PROVIDED HEREIN, WELWYN CURRAH MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Some states do not permit limitation or exclusion of implied warranties; therefore, the aforesaid limitation(s) or exclusion(s) may not apply to the purchaser.

This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

WELWYN CURRAH LTD
104 WEST 4TH ST.
SUITE 208
ROYAL OAK
MICH 48067
U.S.A.