CURRAH

µSPEECH

PROGRAMMING

MANUAL

CURRAH

µSPEECH

PROGRAMMING

MANUAL

By Mark Anson

1st Edition 1983

Acknowledgements to all at General Instrument Microelectronics Ltd. (UK) for generous applications assistance in the development of this product.

CONTENTS

Chapter 1 — Setting up the CURRAH MicroSpeech

Chapter 2 — Simple words and phrases

Chapter 3 — The Allophone Set

Chapter 4 — Intonation and "phrase libraries"

Chapter 5 — The speech buffer—using **CLEAR**

Chapter 6 — Using machine code with the unit

APPENDICES

I — Decimal & Hex codes for the allophones

II — Tape Recorder/Hi-Fi Output

III — LOAD, SAVE and BEEP

IV — Using the unit with the Sinclair ZX Microdrive

V — Commercial software on the MicroSpeech

VI — Your television picture

1. SETTING UP THE CURRAH MICROSPEECH

The CURRAH MicroSpeech is an allophone speech synthesizer, which means that it uses individual speech sounds strung together to make intelligible speech. This is different to some speech synthesizers available today which have a fixed vocabulary; unlike these systems the MicroSpeech has an unlimited vocabulary and can synthesize any word or sentence in the English language. The user merely has to become familiar with the speech sounds of English (which are different from letters) and the allophone symbols used to represent them.

To set up the unit, remove it from its packing tray if you have not already done so, **UNPLUG THE SPECTRUM FROM ITS POWER SUPPLY**, and plug the unit into the port on the back of the Spectrum, CURRAH logo uppermost. You will find that there are two short leads at the back of the MicroSpeech unit:

— A UHF lead (this is the one with the Phono plug)

— A line lead (this has a 3.5mm jack plug on it)

Remove the Spectrum's TV lead from the "TV" socket on the back of the Spectrum and plug in the short UHF lead instead. Then take the Spectrum's TV lead and plug it into the Phono socket on the back of the MicroSpeech. This now allows speech to be "Mixed" with the Spectrum's video signal so that it comes from the TV speaker.

Now take the line lead and plug it into the "MIC" socket on the back of the Spectrum. (You cannot do any harm if you plug it into the "EAR" socket by mistake). This lead allows the sound generated by the Spectrum's own internal loudspeaker to come from the TV speaker — simply remove this lead when you wish to use a cassette recorder with your Spectrum.

Plug in the power lead to the Spectrum. The Spectrum will initialize and the message:

"Speech System (c) CURRAH 1983"

will appear at the top of the screen, as well as normal Sinclair Research Ltd. copyright message. If this does not happen, unplug the power, check that you have connected the unit up correctly, and try again.

Turn up the volume on your television (about half-way will do) and press **ENTER**. The unit will say "enter". If it does not, there are two simple checks you can do:

— Make sure you have pressed **ENTER** — the unit will remain silent until the first **ENTER** is pressed after a power up, a **NEW**, or after a "report" has been printed at the bottom of the screen.

— If you can hear the voice but it is very quiet or a little distorted, the UHF trimmer on the top right corner of the case may need adjusting. Using a small screwdriver, carefully turn the trimmer until you get the best quality speech with the least interference — keep pressing **ENTER** as you do this to hear the voice. Don't worry about turning the screw too far it goes round and round without a stop — but you should find that an adjustment of a fraction of a turn is all that you need.

Now try pressing any of the keys. (Remember to press **ENTER** first if you have just switched on). All BASIC keywords and printable characters are voiced, (with the exception of the " " character) so when **LET** appears on the screen you will hear the word "let" — go on — try it! Note that all graphics characters are voiced as "graphic", whatever their shape.

All the keyvoices "repeat" — if you try holding down **ENTER** for a few repeats, the unit will say something like:

"E... E... E... E... Enter"

The old keyvoice is "chopped off" and a new one started each time the repeat occurs. If you type very fast the same sort of effect is obtained.

Try all of the keyvoices now. Apart from being useful in themselves, the keyvoices serve to illustrate what can be achieved by allophone speech synthesis.

When you are bored with this, go on the next chapter. If at any point you get stuck, remember that all you have to do is unplug the power, wait a moment, and start again — the unit initializes automatically each time.

2. SIMPLE WORDS AND PHRASES

You may keep the keyvoices on as long as you want, but you may find them a nuisance after a while, so enter:

LET keys = 0

And you will find that the keyvoices have vanished. Try entering:

LET keys = 1

And they reappear again. What you have done is to use a "reserved variable" to tell the unit what to do — the variable "keys" is "reserved" for the unit's use. You can put this in a program line, too — try entering this program:

10 LET keys = 0

After running this, the keyvoices are disabled again.

Once you have enabled or disabled the keyvoices, they remain either on or off until you enter a "**LET** keys =" command, either in a program line or entered directly as a command. (Note that the keyvoice status is preserved even after a **NEW**) There is another reserved variable, s$, (standing for "speech string") which is used for generating user-defined speech.

Try entering the following:

LET s$ = "he(ll)(oo)"

You will hear the unit say "hello". (If you do not, type it in again, checking that you get all the brackets in).

Now try entering this program:

10 LET s$ = "he(ll)(oo)" :PAUSE 1

Because of the way in which the MicroSpeech works, you MUST put a **PAUSE** statement after each "**LET s$** =" statement in a program to make sure that the unit detects each s$ as it occurs. (**PAUSE 1** will do).

If you **RUN** this program, you will again hear the word "hello". Every time you create a new **s$**, the contents of the string are examined by the allophone interpreter and, if the syntax is correct, the allophone symbols are converted into speech code and placed in a "buffer" (a sort of queue) near the top of memory. The code in the buffer is then automatically outputted to the speech chip without any further intervention from your BASIC program.

Your BASIC program can go away-and do other tasks once the speech has been stored, so you can fill the buffer up and leave it to get on with outputting

speech whilst you type in some data, for instance, or add more speech to the buffer.

User-defined speech has higher priority than key voices, so that if a sentence is being spoken and you press a key, the keys will not voice, even if they are enabled until the buffer has been emptied.

The unit has a self-diagnostic facility which reports on the status of the speech you have tried to output. **RUN** the above program again, and then enter:

PRINT s$

The computer will report back with:

***e(ll)(oo)**

The asterisk obliterating the first character tells you that the allophone symbols contained in **s$** were syntactically correct and that they were converted into speech code and loaded into the buffer for output. Now try changing the program to read:

10 LET s$ = "he(ll) ! (oo)" :PAUSE 1

No speech will be generated if you RUN this as the exclamation is an "illegal" character. RUN the program, then type in:

PRINT s$

And you will get the following report:

?e(ll)) (?oo)

The query (?) obliterating the first character tells you that there was an error in the string, and the second query shows you where the error occurred. In the case of an error occurring on the last character of the string, the second query obliterates it.

If you **BREAK** into a program whilst the voice is speaking the voice ceases at once — what is not so obvious is that any "Report" given by the Spectrum other than "**OK**" will also have this effect: e.g., **STOP statement, Out of DATA, RETURN without GOSUB,** etc. This is to ensure that if there has been an error in a program which may have left you with a buffer full of data you will not have to wait for the buffer to empty itself — remember a buffer only 200 bytes long can hold over 30 seconds of speech.

You have seen four examples of the allophone symbols used already, and you will probably be wanting to try constructing your own words and phrases. When you feel that you are familiar with the use of the reserved variables, go on to the next chapter, where the whole allophone set will be introduced.

3. THE ALLOPHONE SET

As you may have noticed, there are basically two sorts of allophone symbols, those that occur as single characters and those that have several characters enclosed by brackets. It is necessary to enclose some in brackets because "aa" for instance, is a different sound to "(aa)".

Here is a complete list of all the allophones. Try them to hear what they sound like by running this short program:

```
10 (INPUT a$: LET s$ = a$: PAUSE 1
20 (F s$ (TO 1) = "7" THEN PRINT "You have made a mistake"
30 IF a$ = "stop" THEN STOP
40 GOTO 10
```

Now whenever you come across an allophone you would like to hear, simply enter it when the "L" prompt is given and the unit will say it for you. In the following table, the allophone symbols are shown on the left, with examples of the sounds on the right:

1. PHONETIC allophones:

All single characters	All sound phonetic
a-z excepting x and q	e.g. like "a" in at, "b" in tab, etc.

2. DOUBLE-VOWEL allophones:

(aa) or (ay)	"ay" in hay
(ee)	"ee" in see
(ii)	"i" in hive
(oo) or (eau)	"o" in stove

3. STRONG PHONETIC allophones (used at start of words or where extra emphasis is required):

(bb)	"b" in bat
(dd)	"d" in do
(gg)	"g" in got
(ggg)	"g" in big
(hh)	"h" in hoe
(ll)	"l" in let
(nn)	"n" in no
(rr)	"r" in run
(tt)	"t" in to
(yy)	"y" in yeah

4. COMPLEX ALLOPHONES:

(ar)	"ar" in arm
(aer)	"air" in repair
(ch)	"ch" in church
(ck)	"ck" in clock
(ear)	"ear" in clear
(eh)	"ar" in wary
(er)	"er" in leader
(err)	"ur" in purr
(ng)	"ng" in tongue
(or)	"or" in sore
(ou)	"oo" in root
(ouu)	"oo" in food
(ow)	"ow" in now
(oy)	"oy" in boy
(sh)	"sh" in ship
(th)	"th" in thin
(dth)	"th" in then
(uh)	"oo" in took
(wh)	"wh" in whig
(zh)	"z" in azure

5. PAUSES

apostrophe	very short pause, e.g. in can't
space	pause between words
comma	pause between phrases
full stop	pause between sentences

X and Q are missing from the set as the speech sounds they make may be made from combinations of other allophones—this is the key to allophone synthesis. For instance, the "queen" may be made by "kw(ee)n" and the word "box" by "boks".

Certain allophones run together well, for example:

LET s$ = "aaaaaaaa"

Comes out as a single "Aahhh" sound. Other allophones that exhibit this property are e, i, o, u, f, s, (eh), (th) and (uh).

To help you see how words are built up, here are some examples from the keyvoices:

Keyvoice	Voiced by using:
AT	a(tt)
'	apostruf(ee)
=	(ee) (ck)wulz
NEW	ny(ouu)
SIN	s(ii)n
CODE	c(oo)d
ACS	(ar)c'coz
STRING	stri(ng)
CIRCLE	s(er)cul
LLIST	e(ll)'list
SAVE	s(ay)v
5	f(ii)v
H	(ay) (ch)
LOG	lo(ggg)
THEN	(dth)en
MERGE	m(err)dj

You can see that a little care is needed to think in terms of how words are spoken rather than how they are written, but with a little practice you will be able to build up words very fast with few mistakes.
Try making up some words of your own now, and making short phrases using the various pauses to divide the words up. You should also try LISTing the BASIC demonstration program supplied with the unit to see how the various sentences are generated on that.

4. INTONATION AND "PHRASE LIBRARIES"

Intonation is necessary in allophone speech to add character to words or parts of words. Up to now you will have been entering speech strings in lower case letters. If you use UPPER case letters for any allophone, the intonation goes UP. Try this:

```
LET s$ = "aaAAaaAAaaAAaaAAaa"
```

And you will hear a sound rather like someone yodelling. Note that the pitch of the allophones does not go up or down suddenly — it is "ramped" so that the transition is smooth. Try this:

```
10 LET s$ = "sp(EE)k n(oo) (EE)vil" : PAUSE 1
```

You will be able to hear the intonation. Note that intonation will be most noticeable on vowels and "voiced" allophones, but try experimenting with it to see what sort of effects you can obtain:

In a bracketed allophone, it does not matter if you mix up upper and lower cases inside the bracket, as long as you realize that it is the LAST character inside the bracket which determines the intonation of the allophone. For example, "(oUu)" is not intoned, whilst "(ouU)" is intoned up.

When you are using the MicroSpeech unit to make sentences, think how much easier it would be if you could build up sentences from a "library" of words and phrases. Suppose you wanted to say "I'm sorry, but your answer was incorrect". Rather than put this in one long string, you might want to construct the sentence from a "library" of stock phrases, like this:

```
10 LET as = "(II)'m sor(ee)"
20 LET. b$ = ",but"
30 LET c$ = "y(OR) ans(er) woz"
40 LET d$ = " incurrect"
50 LET s$ = a$: PAUSE 1: LET s$ = b$: PAUSE 1:
   LET s$ = c$: PAUSE 1: LET s$ = d$: PAUSE 1
```

Whilst this will work very well, there is another solution. You could set up the four strings as before but join them together ("concatenate") them into s$ by changing line 50 to read:

```
50 LET s$ = a$ + b$ + c$ +. d$: PAUSE 1
```

Whilst this is more compact than the first method, it is still rather wasteful of strings. If you have to build up sentences from a library of more than about ten phrases, then you will probably find that setting up a string array and concatenating the elements into s$ will be a better solution. You will need to "slice" some elements before concatenation in order to suppress the trailing spaces:

```
10 DIM a$ (4, 18)
20 LET a$ (1) ="(II)'m sor(ee)"
30 LET a$ (2) = ",but"
40 LET a$ (3) = "y(OR) ans(er) woz"
50 LET a$ (4) = " incurrect"
60 LET s$ = a$ (1) (TO 14) + a$ (2) (TO 4) + a$ (3) + a$(4) (TO 10)
70 PAUSE 1
```

5. THE SPEECH BUFFER — USING CLEAR

It was explained earlier that speech data is stored near the top of memory until it is ready to be outputted.

The speech buffer is a "First in — first out" buffer (a FIFO buffer) which is initialized when you turn on the Spectrum with the MicroSpeech attached. In order to do this, the unit moves the "top of BASIC RAM" pointed (RAMTOP) down by 256 bytes (so that on 48K machine it will point to 65111) and the speech buffer is then established between the new RAMTOP and the user-defined graphics area. The buffer then fill from top (i.e. highest memory location) downwards as more speech is added to it. The top 6 bytes of the buffer (the "header") contain information on buffer and system status but the ones below it are free for speech data. You may make the buffer as large or as small as you want by use of the **CLEAR** command.

For instance, on a 48K machine, **CLEAR 65000** makes RAMTOP point to 65000, and the memory space free for the speech buffer will be increased by 111 bytes.

To safeguard against the buffer header being corrupted by a **CLEAR** command, the MicroSpeech unit monitors RAMTOP, and if a **CLEAR** is attempted into an illegal area (see below), the unit goes through its initialization sequence again.

Your BASIC program is not lost when this happens (unless it was so large than the new RAMTOP established by the unit truncates it), but you will find a new copyright message issued and RAMTOP once again established 256 bytes below the original RAMTOP.

If you try to add speech to the buffer and there is sufficient room for it to be fitted in, then the unit will ignore the speech string s$ until there is sufficient room in the buffer. If you **PRINT s$** under these conditions you will find it returned uncorrupted, i.e. without as asterisk or a query, to show that it has not been accepted into the buffer. If you find this occurring in a program, simply use **CLEAR** as described above to make the buffer a little larger, or alternatively use a **PAUSE** to allow the buffer to empty a little before attempting to add another string to it.

Since the keyvoices are outputted via the speech buffer, you can observe a similar effect if you enable the keyvoices when the buffer is very small. For instance, if you make the buffer only four bytes long, then any keyvoice longer than four allophones will not be voiced.

Summarising:

On a 16K machine, do not CLEAR greater than 32343

On a 48K machine, do not CLEAR greater than 65360

6. USING MACHINE CODE WITH THE UNIT

This section is intended to set out guidelines for writing machine code programs for the unit. It must be stressed that the MicroSpeech was primarily designed to be used with BASIC for wider popular appeal, and as the operating software is necessarily complex only the experienced programmer should attempt writing machine code for the unit.

In the following examples, the 48K version memory addresses will be used — simply subtract 32768 to convert to the 16K machine. Decimal is used for all numbers unless otherwise stated.

At the top of the speech buffer is the six-byte "header" — information which controls the action of the buffer and the speech system. This comprises:

ADDRESS (48K)	FUNCTION
65367	Flag Byte
65366	Spare Byte
65365	Hi byte of buffer pointer
65364	Lo byte of buffer pointer
65363	Spare
65362	Spare
65361	Next allophone to be voiced

The Buffer Pointer keeps a pointer to ONE BELOW the last item in the buffer which has to be voiced. It is incremented by one every time an allophone byte is outputted to the speech chip from the top of the buffer and the entire contents of the buffer are moved up by one ready for the next byte to leave. If this pointer points to 65361, as it does at startup, then the buffer is deemed empty and no allophone is issued.

The Spare Bytes are initialized to zero (00H) at startup and serve as safe locations for you to store vital data in—you may want to experiment with a dual buffer, for instance, and switch between the two by swapping pointers.

The Flag Byte contains various flags for system use.

The function of each bit is as follows:

BIT	FUNCTION
7	Copyright message flag — reset after issuing message

6	Spare
5	Spare
4	Used in keyboard scan — unlikely to be useful
3	Spare
2	Used in keyboard scan — unlikely to be useful
1	Key voice enable flag (set at first)
0	Spare

None of the flag bits will cause a crash if you alter them although the copyright message will appear if you set bit 7.

The important thing NOT to corrupt is the buffer pointer — if you make this point below the current RAMTOP you will crash the system.

If you would like to input speech data directly into the buffer rather than go via the BASIC variable s$ you must first fill the buffer with the allophones you want.

Each allophone in the buffer is represented by a byte; six bits of each byte is devoted to the allophone code, one bit is intonation, and bit 7 is always zero.

BIT:	7	6	5 4 3 2 1 0
FUNCTION:	zero	Intonation (1 for up)	---Allophone---

The decimal and hexadecimal codes for the allophones are given in Appendix I. As an example of machine code programming, let us suppose that you wanted to make the unit say "hello". First of all, you must work out the codes for "he(ll)(oo)" — they are 27,7,62 and 53.

Let us also suppose you wanted the "e" to be intoned up. Simply add 64 to the basic code (7) to give 71. Now that you. have your data, you have to load it into the buffer in the right order (top down) and then update the buffer point to ONE BELOW the last allophone you wish to be voiced. Here is a suggested program which would do this, written as an assembly language subroutine:

```
Start:  ld hl,Data
        ld de,65361
        ld b,4              ;load count with 4
Loop:   ld a,(hl)
        ld (de),a           ;load allophone into buffer
        inc hl
        dec de
        djnz Loop           ;loop to load four bytes
        ex de,hl            ;point hl to 1 below the
                            ;last byte in buffer
        ld (65364) ,hl      ;update pointer
        ret                 ;and return

Data:   defb 27,71,62,53    ;the four data bytes
```

APPENDIX 1 — Decimal and Hex codes for the Allophones

ALLOPHONE	DECIMAL	HEX	ALLOPHONE	DECIMAL	HEX
a	24	18	(aa) or (ay)	20	14
b	28	1C	(ee)	19	13
c		08	(ii)	6	06
d	21	08	(oo) or (eau)	53	35
e	7	07			
f	40	28	(bb)	63	3F
g	36	24	(dd)	33	21
h	27	1B	(gg)	61	3D
	12	0C	(ggg)	34	22
	10	0A	(hh)	57	39
k	42	2A	(H)	62	3E
	45	2D	(nn)	56	38
m	16	10	(rr)	14	0E
n	11	08	(tt)	13	0D
o	23	17	(yy)	25	19
	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
	1	01	(th)	29	1D
(space)	3	03	(dth)	18	12
	4	04	(uh)	30	1E
	This is a	hybrid of	(wh)	48	30
	2 pauses	(04+04)	(zh)	33	21

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.

II — Tape Recorder/Hi-Fi Output

The speech signal is present as a high impedance output on the line lead as well as a UHF output. To record some speech, plug the line lead into the "MIC" input on your tape recorder, set it to "RECORD" and any speech produced by the unit will be recorded on tape. To output speech to an external amplifier, connect the line lead to the "AUX" or "TAPE" input on your amplifier, via a suitable connecting lead.

III — LOAD, SAVE and BEEP

Before attempting to **LOAD** from a tape recorder, it is sensible to disable the keyvoices, as otherwise you will get a continuous "e" allophone outputted during the program load. This is due to the tape handling routines taking complete control of the Spectrum's CPU and preventing the MicroSpeech from updating its speech buffer. Upon completion of the **LOAD**, the "enter" keyvoice is completed normally.

If you leave the keyvoices on during a **SAVE**, you will hear a keyvoice after the preliminary data block has been written to the tape recorder (the keyvoice you will hear is the one you press after the "press any key" prompt).

If you **BEEP** in the middle of an allophone, you will "draw out" the allophone and combine it with a musical tone, as the **BEEP** command prevents updates of the speech buffer for the duration of the tone. If you are very clever, you may be able to use this effect to make the MicroSpeech "sing".

IV — Using the unit with the Sinclair ZX Microdrive

The CURRAH MicroSpeech unit is fully compatible with the Sinclair ZX Microdrive and Interface I. Simply follow the setting up instructions supplied with Interface I, and plug the MicroSpeech unit into the port on the back of Interface (BEFORE plugging in the power.

Again, be sure to disable the keyvoices during all data transfer operations to prevent the "enter" keyvoice from being drawn out.

V — Games software on the MicroSpeech

Most games programs will run on the CURRAH MicroSpeech, and even if the program does not use the speech facility you will find the game is enhanced as the Spectrum's sound will come from your television speaker. Leading software houses are currently writing games software which will utilize the speech output — watch out for these as they become available.

VI — Your television picture

The CURRAH MicroSpeech "mixes" sound into the UHF output of the Spectrum modulator, and consequently on some makes of TV set you may find some "patterning" where large areas of strong colour are being displayed. This is not due to a fault in your unit, but is an inevitable consequence of mixing sound into the UHF output; the technically-minded amongst you might be interested to know that the patterning is due to "sidebands" generated during the mixing process interfering with the colour information being transmitted to your television.

Currah Computor Components Ltd.
Hollymount, Wooler Road, Hartlepool, Cleveland TS26 0HA
Tel. (0429) 72996/34511/36181. Telex. 58127 CURRAH G

Printed by Harrison's of Hartlepool