AMSTRAD

Speech Synthesiser & Stereo Amplifier including Speakers

FOR THE AMSTRAD CPC464

SSA-1


Vocal
ateur
t-Parleurs
PC464
-1
AMSTRAD

AMSTRAD

Speech Synthesiser
& Stereo Amplifier
including Speakers

FOR THE AMSTRAD CPC464

SSA-1


SSA-1
Synthetiseur Vocal
Avec Amplificateur
Stereo Et Haut-Parleurs
AMSTRAD
Synthetiseur Vocal
Avec Amplificateur
Stereo Et Haut-Parleurs
POUR L' AMSTRAD CPC464
SSA-1


AMSTRAD
FABRIQUE EN HONG KONG
AMSTRAD


SSA-1
Manuel de L'utilisateur


Synthetiseur Vocal
Avec Amplificateur
Stereo Et Hau
POUR L'

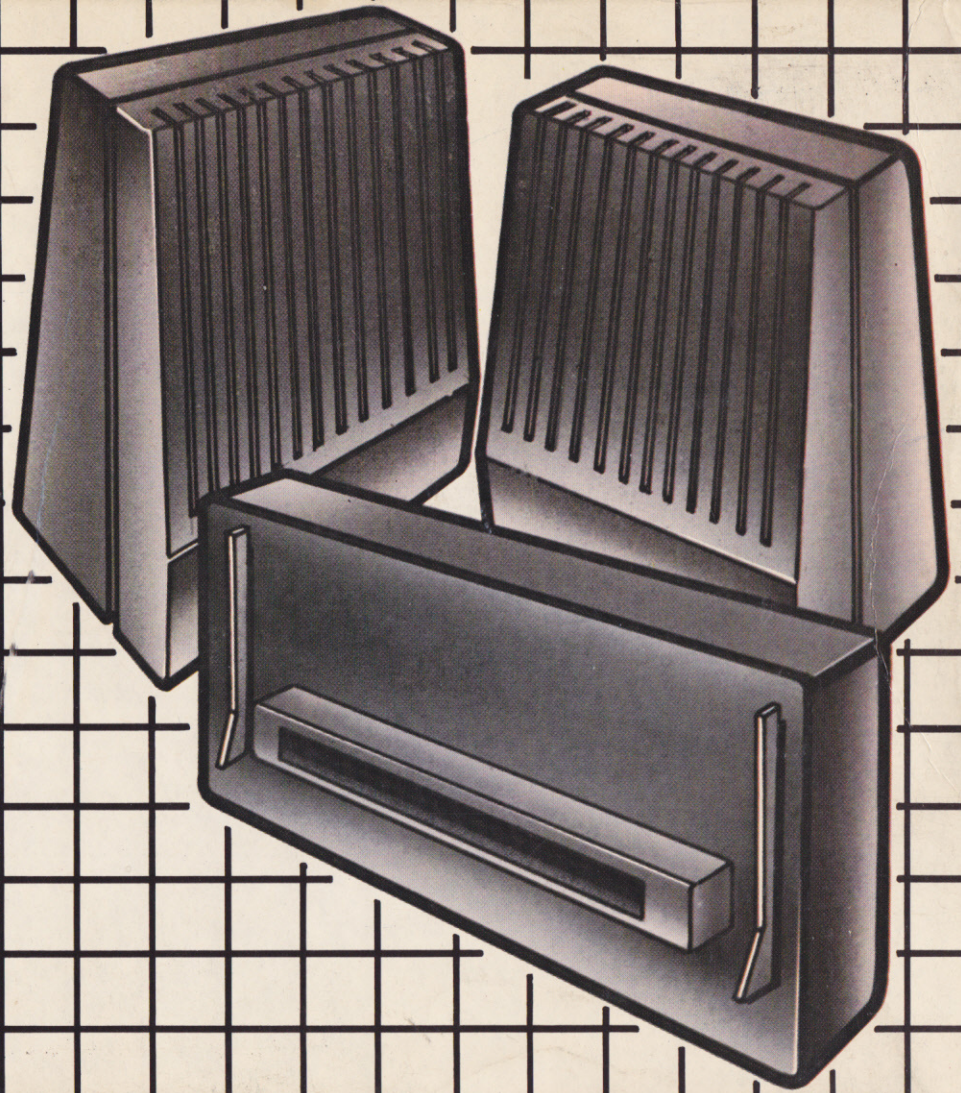

SSA-1

AMSTRAD

Speech Synthesiser & Stereo Amplifier including Speakers

FOR THE AMSTRAD CPC464

SSA-1



SSA-1

Speech Synthesiser & Stereo Amplifier including Speakers

Made in Hong Kong

SSA-1

AMSTRAD

Speech Synthesiser & Stereo Amplifier including Speakers

FOR THE AMSTRAD CPC464

SSA-1

AMSTRAD

RS232C Serial Interface

FOR THE AMSTRAD CPC464/664/6128

RS232C

RS232C

RS232C Serial Interface

FOR THE AMSTRAD CPC464/664/6128

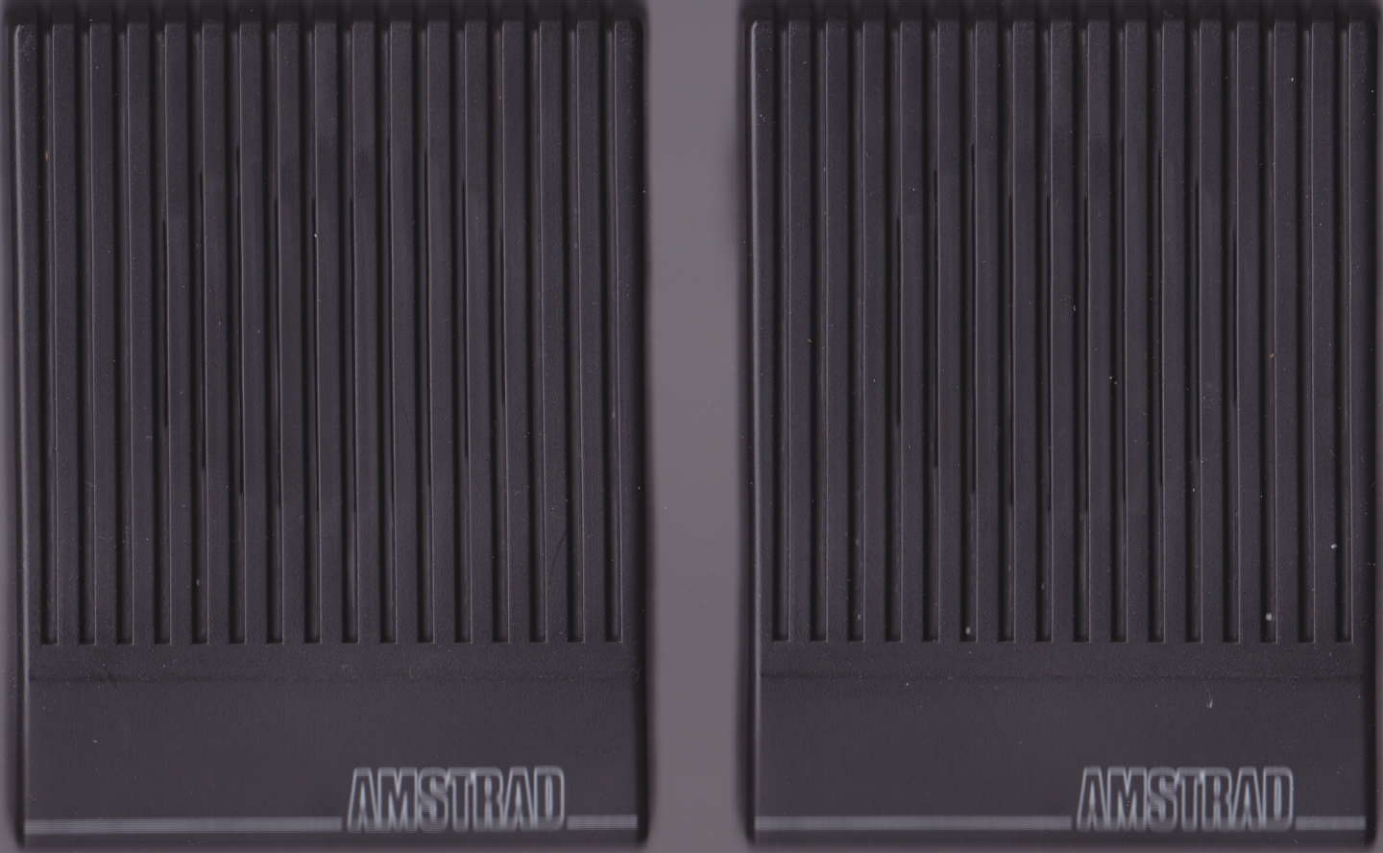
AMSTRAD

LIGHT PEN

LP-1

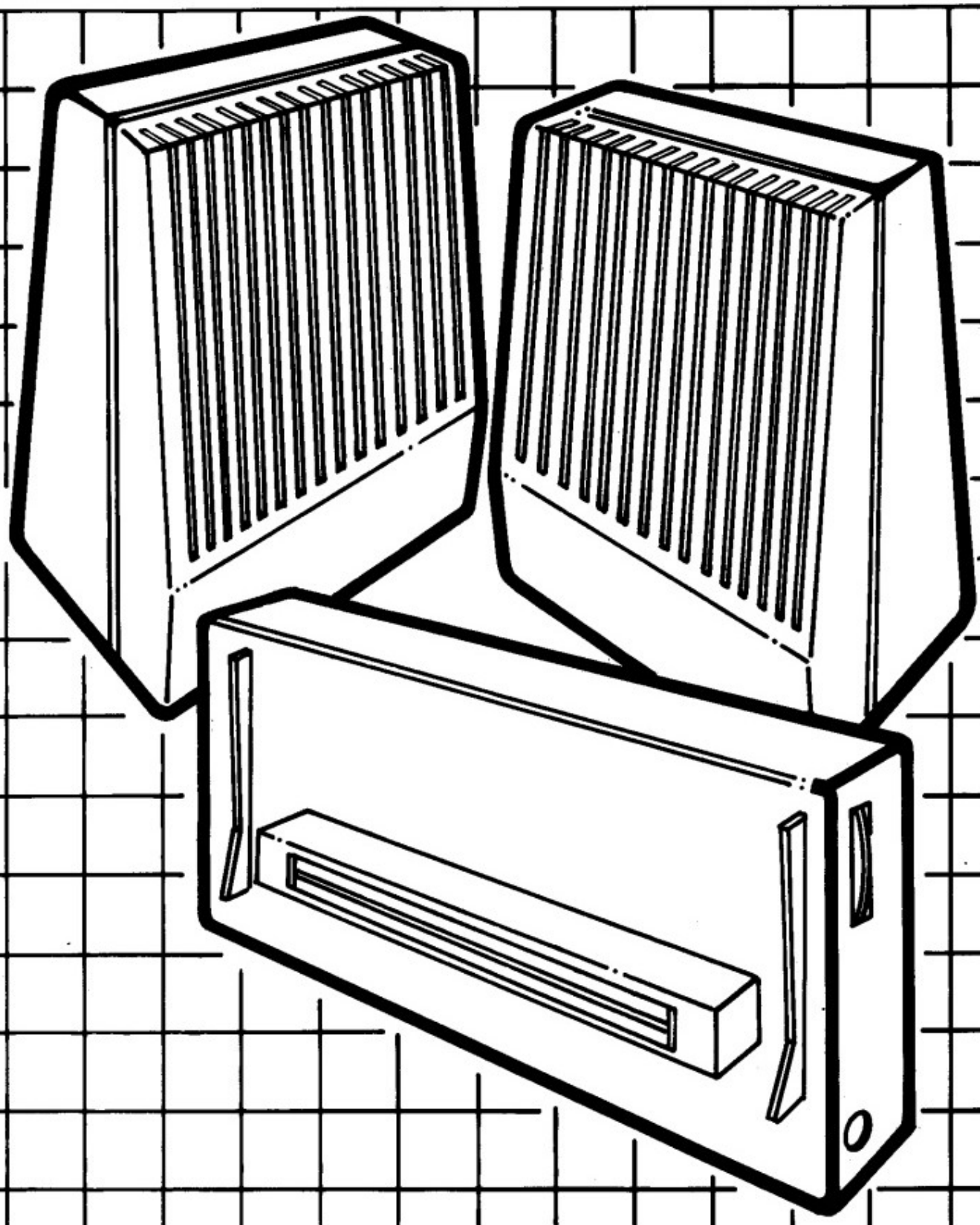
Colour Monitor Only


AMSTRAD
AMSTRAD


AMSTRAD
AMSTRAD


Q. C.
Passed


AMSTRAD
AMSTRAD



SSA-1

User Instruction Manual

CONTENTS

1 INTRODUCTION

Features

1. Natural speech
2. Allophone synthesis
3. Unlimited vocabulary
4. Audio mixer for stereo sound from CPC464
5. Stereo output, 200mW per channel into 4 ohms
6. 'Through Bus' for peripheral 'daisy chaining'
7. Nine extra commands including Text to Speech conversion
8. Interrupt driven speech output buffer

The Amstrad *SSA1* combined speech synthesiser and amplifier represents the latest state of the art in speech synthesis for the home micro.

Until recently most speech synthesisers could only manage a limited range of words, decided upon during manufacture. This was fine if the word you wanted to say what was available. In contrast, the *SSA1* can say any word or sentence in the English language. It does this by using a combination of discrete speech sounds called allophones. By combining allophones together intelligible speech is produced.

The *SSA1* also incorporates a sophisticated Text to Speech conversion system that allows most arbitrary English text to be spoken automatically. A stereo amplifier ensures crisp speech whilst allowing full use to be made of the CPC464's stereo sound.

2 SETTING UP THE SSA1

2.1 Installing the Hardware

Ensure that the computer and any other peripherals are switched off before connecting the interface. Connect the speakers into the sockets either side of the interface. When viewed from the front, the left speaker goes into the socket on the left hand side of the interface. Plug the interface into the Floppy Disc expansion port, then plug any other peripherals (disc drive) into the back of the SSA1.

A flying lead is provided to connect the *SSA1* to the stereo sound output of the CPC464, allowing all sound output to be heard via the stereo speakers. Connect the flying lead into the stereo sound output socket next to the joystick socket, on the left back edge of the computer. Turn the volume control on the right hand side of the interface to its central position.

Apply power to any peripherals and then switch on the computer. Press the green **[DEL]** key. A beep should be heard on both speakers. Re-adjust the volume control to suit.

When the *SSA1* is connected the internal loudspeaker of the computer is muted, although the loading of cassette programs may still be heard as normal through the internal loudspeaker.

2.2 Loading the Operating software

Before any speech can be generated the software supplied must be loaded into memory. Insert the tape and type:

```
RUN "SSA1 [ENTER]
```

Initially a short BASIC loader program will run that checks the *SSA1* hardware. If the hardware is faulty or not attached to the expansion port the message:

```
SSA1 hardware not operational
```

will be displayed and the program will end.

Otherwise a message asking for the address at which to load the software will be displayed. If you do not know what load address to use, simply type **[ENTER]** on its own, the software will then be loaded at the top of memory. The load address entered must be above 16384 and below `HIMEM`. The program will tell you how much memory is required by the software. Note that `HIMEM` will be set to prevent BASIC from using the same memory.

After the load address has been entered the main *SSA1* software will be loaded. When complete the message 'hello' should be heard. The speech synthesiser is now ready for use.

If you attempt to load the *SSA1* software in more than once, the message:

```
SSA1 software already loaded.
```

will be spoken and displayed.

The *SSA1* software you have just loaded is a special kind of machine code program. It allows extra commands for speech to be used in a BASIC program. To use the commands you simply type in their name (see Getting Started) together with any other variables as needed.

The *SSA1* software remains in memory, even after a NEW command, until a reset is performed, (using **[CTRL] [SHIFT] [ESC]**). In other words, the only time the *SSA1* software has to be installed is when the machine is switched on or reset.

2.3 Precautions In Use

The following precautions should be observed.

Always ensure that the computer and any other peripherals attached on the through bus (disc drives etc.) are switched off before connecting or disconnecting the interface.

Never short circuit the loudspeaker outputs for long periods of time.

Do not use loudspeakers with a lower impedance rating than 4 ohms per channel.

Loudspeakers generate magnetic fields. ALWAYS ensure that discs and cassettes are kept well away from the loudspeakers.

Never place the loudspeakers near the monitor screen, never place them on top of the monitor. (Permanent picture or colour distortion may result due to the magnetisation of part of the screen).

3 GETTING STARTED

The *SSA1* software provides nine external commands allowing speech to be generated for any application, the commands are:

```
|SAY,@‹STRING VARIABLE›
|ECHO,‹MODE›

|QUIET
|APHONE,‹INTEGER VARIABLE› [,‹INTEGER VARIABLE›]
|ROOM,@‹INTEGER VARIABLE›

|SPON
|SPOFF

|SPSTATUS,@‹INTEGER VARIABLE›
|SPOUT,‹INTEGER VARIABLE› [,‹INTEGER VARIABLE›]
```

Chapter 8 describes these commands in detail. The next section presents a few simple examples demonstrating how easy the *SSA1* is to use.

3.1 Simple Words and Phrases

The SSA1 contains a Text to Speech system that looks at text and converts characters into allophones, the fundamental sounds of speech, that the *SSA1* hardware can understand.

Two commands **|SAY** and **|ECHO** perform text to speech conversion. The **|SAY** command takes a string and says it. The **|ECHO** command can be set to look at text printed to the screen, so that listings and printed characters can be heard.

Type **NEW** and then the program:

```
10 a$="simple words"
20 |SAY,@a$
```

(The | character can be obtained with **[SHIFT] [@]**)

Now **RUN** the program and observe what happens. The program finishes before all the words are spoken. The software uses a speech queue, which works in a similar way to the BASIC SOUND queue. Essentially it allows the allophones to be spoken in order, transparently from BASIC. In other words it allows BASIC to continue a program without having to wait for speech to finish. The queue has room for a maximum of 64 allophones, around 8 seconds of speech. If there is no room left in the queue then the **|SAY** and **|ECHO** commands will wait for room to be made, it is only then that BASIC will pause, until all the speech has been sent to the buffer.

The string used with **|SAY** can contain any of the 256 ASCII characters, some characters such as **+ - &** are spoken, whilst others like **[]** are ignored. For a fuller explanation of the Text to Speech system refer to Chapter 5.

The **|ECHO** command is intended for saying phrases normally sent to the screen.

Type in the program:

```
10 |ECHO,1
20 INPUT "'How old are you '";age$
30 PRINT "'You are";age$;" years old'"
40 |ECHO,0
```

Line 10 instructs the SSA1 to say text printed to the screen between the delimiters (:), line 40 reverts printing back to normal. The (:) character is obtained with **[SHIFT]** [\], the key immediately above the green **[CTRL]** key.

RUN the program and enter your age as a number. Notice that numbers are spoken exactly as you would expect. Now type in:

```
|ECHO,3
LIST
```

The program will be listed and spoken, in fact anything printed by BASIC including syntax errors, messages etc. will be spoken. To cancel the speech echo enter:

```
|ECHO,0
```

Here are some more examples:

1. Say the alphabet

```
10 FOR letter = ASC("A") TO ASC("Z")
20  a$=CHR$(letter)
30  |SAY,@a$
40 NEXT
```

2. Say the numbers 1 to 10

```
10 |ECHO,3
20 FOR number = 1 to 10
30  PRINT number
40 NEXT
50 |ECHO,0
```

You should now be able to use speech from within any of your programs, for more details of the commands refer to Chapter 8.

3.2 Loading and Saving BASIC Programs

Essentially there is no difference between the way programs are normally loaded or saved, and the way they are loaded or saved with the *SSA1* installed. However, the the *SSA1* software must always be installed, as outlined in section 2.2, after switching on, or resetting, the computer.

Your BASIC programs can be saved as usual with the SAVE command. For example:

```
SAVE "MYPROG"
```

When LOADing programs that use the speech commands you must ensure that the *SSA1* software has been installed first, otherwise the speech commands will not be present, and BASIC will give an error message when the program is run. Remember, the *SSA1* software is installed as explained in section 2.2.

After installing the *SSA1*, your BASIC programs may be loaded as usual with the LOAD command, for example:

```
LOAD "MYPROG"
```

3.3 Experimenting with sounds

The English language has many irregulations in pronounciation for example "Cough", "bough". The SSA1 software contains a large number of pronounciation rules but sometimes it will be necessary to experiment with character spelling to achieve the exact results.

Cough = COF
Bough = BOUH

4 ALLOPHONES

As mentioned previously, the Text to Speech system converts text into allophones. It is not possible to convert every word successfully into the correct allophones because of the many spelling-pronunciation irregularities in the English language. To allow difficult words to be spoken correctly, the |APHONE command can be used to speak allophones chosen by yourself.

Type NEW and then this program:

```
10 |SPON
20 |APHONE,42,15,16,9,49,22,13,51,3
```

When the program is RUN the word computer is spoken. The numbers after the |APHONE command are the allophones that make the word computer. Each allophone is stored in turn in the speech queue, ready to be spoken. Unlike the |SAY and |ECHO commands the allophones are not spoken immediately, but wait for the speech queue to be turned on with the command |SPON. Once the speech queue is on, any further allophones will be spoken until the queue is turned off, or held, with |SPOFF. To demonstrate the queueing effect, change line 10 to:

```
10 |SPOFF
```

This time when the program is **RUN** nothing is heard. This is because the queue is held. Now type in:

|SPON [ENTER]

Computer should now be heard. More information on these commands can be found in Chapter 8.

4.1 The Allophone Set

There are a total of 59 allophones and 5 pauses available on the *SSA1*. To use the allophone set successfully to synthesise words, the following points should always be remembered:

1. Speech sounds do not always appear acoustically the same, they may be spoken differently, even within the same word.
2. Like the pieces of a Jig-saw puzzle, the speech sounds are separate. Combinations of them must be used to make up the entire 'picture' of how a word sounds.
3. The letters of the alphabet do not have their own speech sound. Each letter may be spoken in a different way depending upon its position within a word.

To illustrate these points try saying the following words one after another:

```
call cup coop
```

You should feel the point of contact between the back of your tongue and the roof of your mouth move further back for each successive 'c' sound.

The 'c' at the start of each word is known as a phoneme. Phonemes themselves cannot be spoken. When the word is spoken a different sound of 'c' is produced, and these different sounds are known as allophones. Thus allophones are different ways of saying phonemes. Most phonemes are influenced by other phonemes around them, and this is why the same letter within a different word can make a different sound.

The allophone set (see Tables 4A and 4B) contains two or three sounds of some phonemes. It may be necessary to use one allophone of a particular phoneme at the start of a word or syllable, and another for the end of a word or syllable. A detailed set of guidelines for using the allophones are given in Table 4B. Note that these are suggestions, not rules.

As an example, /DD2/ sounds good in an initial position and /DD1/ sounds good in final positions, like in 'daughter' and 'collide'. One of the differences between the initial and final versions of a consonant may be that an initial version is longer. Therefore, to create an initial /SS/, two /SS/s can be used instead of the usual single /SS/ at the start of a word or syllable, as in 'sister'. Note that this can be done with /TH/, and /FF/, and the inherently short vowels (refer to Table 4B), but with no other consonants.

Generally it is best to experiment with some consonant clusters (strings of consonants such as str,cl), in order to discover which version works best in the cluster.

For example /KK1/ sounds good before /LL/ as in 'clown', and /KK2/ sounds good before /WW/ as in 'square'. One allophone of a particular phoneme may sound better before or after back vowels and another before or after front vowels. The /KK3/ allophone sounds good before /UH/, and /KK1/ sounds good before /IY/, as in 'cookie'.

Some sounds (/PP/ /BB1/ /BB2/ /TT1/ /TT2/ /DD1/ /DD2/ /KK1/ /KK2/ /KK3/ /GG1/ /GG2/ /GG3/ /CH/ and /JH/) require a brief duration of silence before them. For most of these, the silence is included in the allophone, but more may be added as desired. The pauses PA1 to PA5 may be used to provide silences of the appropriate duration.

When using allophones always think about how a word SOUNDS, not how it is spelt. For example, the /NG/ allophone obviously belongs at the end of the words 'sing' and 'long', but notice that the /NG/ sound is represented by the letter N in 'uncle'. Some sounds may not be represented in words by any letters, such as the /YY/ sound in 'computer'.

Most vowels can be doubled to make longer versions for stressed syllables. These are the inherently short vowels /IH/, /EH/, /AE/, /AX/, /AA/, and /UH/. For example, in the word 'extent' one /EH/ is used in the first syllable, which is unstressed, and two /EH/s are used in the second syllable, which is stressed. One of the inherently long vowels has a long and short version. The short one /UW1/, sounds good after /YY/ in 'computer'. The long version, /UW2/, sounds good in monosyllabic words like 'two'.

Included in the vowel set is a group called R-coloured vowels. These are vowel plus R combinations. For example, the /AR/ in 'alarm' and the /OR/ in 'score'. Of the R-coloured vowels there is one which has a long and short version. The short version is good for polysyllabic words with final /ER1/ sounds like 'letter', and the long version is good for monosyllabic words like 'fir'.

One final suggestion when creating sentences is to add a pause of 30-50msec between words and a pause of 100-200msec between sentences. Always remember to send a pause after the final allophone to suppress the last sound made by the speech synthesiser.

4.2 How To Use The Allophone Set

Table 4A should be used initially when trying to split words into their allophones. The allophones given, make the sound of the letters, and they can be used to make up any word in the English language.

For example to find the allophones for the word 'hello', start with the first letter of the word, thinking about how the word sounds. The first letter is 'h', and the allophone for 'h' is 27. The next letter is 'e', and its allophone is 7. The double 'll' sound can be made with (ll), whose allophone is 62. The 'o' at the end of the word makes the sound like the letter 'O' in the alphabet, the same sound is also made by (eau), whose allophone is 53. Lastly all the allophones must be put into order followed by a pause, a short pause PA1 is ideal. The corresponding **|APHONE** command would be:

```
IAPHONE,27,7,62,53,0
```

For more complicated words that sound 'funny' it may be necessary to refer to table 4B in addition to table 4A for extra information on the use of a particular allophone.

TABLE 4A - Quick Guide to the allophones

Sound	Allophone	Section in table 4B
a	24	4B.11
b	28	4B.51
c	8	4B.52
d	21	4B.51
e	7	4B.11
f	40	4B.42
g	36	4B.51
h	27	4B.42
i	12	4B.11
j	10	4B.2
k	42	4B.52
l	45	4B.3
m	16	4B.6
n	11	4B.6
o	23	4B.11
p	9	4B.52
r	39	4B.3
s	55	4B.42
t	17	4B.52
u	15	4B.11
v	35	4B.41
w	46	4B.3
y	49	4B.3
z	43	4B.41
(aa)/(ay)	20	4B.12
(ee)	19	4B.12
(ii)	6	4B.12
(oo)/(eau)	53	4B.12
(bb)	63	4B.51
(dd)	33	4B.51
(gg)	61	4B.51
(ggg)	34	4B.51
(hh)	57	4B.42
(ll)	62	4B.3
(nn)	56	4B.6
(rr)	14	4B.3
(tt)	13	4B.52
(yy)	25	4B.3
(ar)	59	4B.13
(aer)	47	4B.13
(ch)	50	4B.2
(ck)	41	4B.52
(ear)	60	4B.13
(eh)	26	4B.11
(er)	51	4B.13
(err)	52	4B.13
(ng)	44	4B.6
(or)	58	4B.13
(ou)	22	4B.12
(ouu)	31	4B.12
(ow)	32	4B.12
(oy)	5	4B.12
(sh)	37	4B.42
(th)	29	4B.41
(dth)	18	4B.41
(uh)	30	4B.11
(wh)	48	4B.42
(zh)	38	4B.41

TABLE 4B - Allophones by letter group

4B.1 VOWELS

4B.11 Short - These allophones can be doubled

Address	Allophone	Duration	Sounds-like	Example & when to use	Vowel Type
7	/EH/	50mS	e	bEnd	Mid front
12	/IH/	50mS	i	fItting	High front
15	/AX/	50mS	u	sUcceed	Mid central
23	/AO/	70mS	o	sOng	Low back
24	/AA/	60mS	a	hOt	Low central
26	/AE/	80ms	eh	End	Low front
30	/UH/	70mS	oo	cOOk	High Back

4B.12 Long

5	/OY/	290mS	oy	tOY	Mid back
6	/AY/	170mS	ii	skY	Low central
19	/IY/	170mS	ee	sEE	High front
20	/EY/	200mS	aa/ay	grEAt	Mid front
22	/UW1/	60mS	ou	tO	High back
31	/UW2/	170mS	ouu	monosyllabic fOOd	High back
32	/AW/	250mS	ow	OUt	Low central
53	/OW/	170mS	oo/eau	snOW	Mid back

4B.13 R- Coloured

47	/XR/	250mS	aer	repAIR	Mid front
51	/ER1/	110mS	er	lettER	Mid central
52	/ER2/	210mS	err	monosyllables: bIRd	Mid central
58	/OR/	240mS	or	stORE	Low back
59	/AR/	200mS	ar	fARm	Low central
60	/YR/	250mS	ear	clEAR	High front

4B.2 AFFRICATES

Address	Allophone	Duration	Sounds-like	Example & when to use
10	/JH/	100mS	j	Jury
50	/CH/	150mS	ch	CHurch

4B.3 RESONANTS

Address	Allophone	Duration	Sounds-like	Example & when to use
14	/RR1/	130mS	rr	Initial positions: Read
39	/RR2/	80mS	r	Initial clusters: bRown
49	/YY1/	90mS	y	Initial clusters: cUte
25	/YY2/	130mS	yy	Initial position: Yes
45	/LL/	80mS	l	heLLo
62	/EL/	140mS	ll	angLe
46	/WW/	140mS	w	We

4B.4 FRICATIVES

4B.41 Voiced

Address	Allophone	Duration	Sounds-like	Example & when to use
18	/DH1/	140mS	dth	Initial positions of words: THey
54	/DH2/	180mS	dth	Final positions and between vowels: baTHe
35	/VV/	130mS	v	Vest
43	/ZZ/	150mS	z	Zoo
38	/ZH/	130mS	zh	beiGE

4B.42 Voiceless

29	/TH/	130mS	th	Thin
40	/FF/	110mS	f	Fire
55	/SS/	60mS	s	Sat

(29,40,55 may be doubled for Initial positions and used singly in final positions)

27	/HH1/	90mS	h	before front vowels: Head
57	/HH2/	130mS	hh	before back vowels: Hoe
37	/SH/	120mS	sh	SHirt
48	/WH/	150mS	wh	WHim

4B.5 STOPS

4B.51 Voiced

Address	Allophone	Duration	Sounds-like	Example & when to use
28	/BB1/	40mS	b	riB
63	/BB2/	60mS	bb	Initial positions before vowels: Bug
21	/DD1/	50mS	d	Final positions: enD
33	/DD2/	80mS	dd	Initial positions and in clusters: Down, Drain
36	/GG1/	80mS	g	Before high front vowels: Guest
61	/GG2/	80mS	gg	Before high back vowels: Go; and clusters: Green
34	/GG3/	120mS	ggg	Before low vowels: Got; In medial clusters: anGer; and fina positions: peG

4B.52 Voiceless

17	/TT1/	80mS	t	Final clusters before /SS/: iTs
13	/TT2/	100mS	tt	To
42	/KK1/	120mS	k	Before front vowels: Computer
41	/KK2/	140mS	ck	Final positions: speaK Final clusters: sKy
8	/KK3/	80mS	kk	Before back vowels: Cork; and initial clusters: Crane
9	/PP/	150mS	p	Pub

4B.6 NASAL

Address	Allophone	Duration	Sounds-like	Example & when to use
16	/MM/	180mS	m	Milk
11	/NN1/	170mS	n	Before front and low vowels: Nice; Final clusters: earN
56	/NN2/	140mS	nn	Before back vowels: Noise
44	/NG/	200mS	ng	aNGer

4B.7 PAUSES

Address	Allophone	Duration	Example & when to use
0	PA1	10mS	Use before voiced stops and /JH/, or as an apostrophe (')
1	PA2	30mS	Use before voiced stops and /JH/, or as a colon (:) or semicolon (;)
2	PA3	50mS	Use before voiceless stops, and /CH/, and between words, or as spaces
3	PA4	100mS	Use between phrases or as a comma (,)
4	PA5	200mS	Use between sentences or as a period (.)

5 TEXT TO SPEECH

The commands **|SAY** and **|ECHO** incorporate a sophisticated text to speech system capable of converting most commonly occurring English text into allophones. The allophones represent the individual speech sounds that are found within words and are output to the SSA1 speech synthesiser hardware, to produce speech corresponding to written text.

The text to speech system uses the principle of Synthesis-by-Rule. This actually involves two separate tasks. The first task consists of text interpretation, the second the application of phonological rules.

Text interpretation is a non-trivial task. The spelling system of modern day English is awkward and sometimes confusing, as any school child might be aware. Words that sound the same may be spelt differently, like 'here' and 'hear', and equally words that sound differently may be spelt the same. For example 'lead' as in 'Lead us not into temptation' verses 'lead weight' meaning the metal. Similarly 'live' as in 'live wire' verses 'live a long time'. The only real solution to this problem would require a database containing the meanings of words. The context of the phrase could then be determined and the correct pronunciation deduced. Unfortunately this approach is restricted to mini-computers rather than personal computers. Text interpretation in the case of the *SSA1* is therefore restricted to the identification of word boundaries, the recognition of symbols and abbreviations, such as '@' and 'Dr', and the conversion of numbers. If one spelling of a word sounds incorrect then try its alternative, or experiment with misspelling the word.

Once the input text has been identified a set of rules are applied to determine the corresponding allophones for a given letter group within a word. The same letters may have different sounds depending where they occur within a word. Unfortunately the English language has many spelling peculiarities that are capable of deluding any logical set of rules. In fact some of the most common words contain blatant violations of the general rules. An instructive illustration of this comes from George Bernard Shaw, who used the letters 'ghoti' to spell 'fish', taking 'gh' from 'cough', 'o' from 'women', and 'ti' from 'nation'.

In order to make use of the spelling rules the English language does have, without causing havoc with those words that defy the rules, an exceptions dictionary is used that contains the correct set of allophones for troublesome words. Some of the BASIC keywords are examples of exceptions, the command **CREAL**, for instance, is actually pronounced 'c' followed by 'real'. For this reason the allophones for some keywords are held in the exceptions dictionary to prevent the word being said literally, and therefore sounding incorrect. The list of possible exceptions can be quite large and consequently the *SSA1* only has a limited set. Any words that still sound incorrectly may be tackled by splitting them into their allophones, as outlined in Chapter 4.

5.1 Recognised Abbreviations

OK	
BBC	
ITV	
Dr	'doctor'
Mr	'mister'
Mrs	'misses'
etc	'etcetera'
pm	'p' 'm'

5.2 Spoken Symbols

Symbol	ASCII	Text spoken
#	23H	'hash' when not followed by a digit, 'pounds' if immediately followed by a digit
$	24H	'string' when not followed by a digit, 'dollars' if immediately followed by a digit
%	25H	'percent'
&	26H	'and'
*	2AH	'times'
+	2BH	'plus'
-	2DH	'minus'
.	2EH	'point' if followed by a digit
/	2FH	'divided by'
<	3CH	'less than'
=	3DH	'equals'
>	3EH	'greater than'
@	4flH	'at'
\	5CH	'divided by'
↑	5EH	'power'
|	7CH	'bar'

The ASCII characters 23H ('#' pound or hash), and 24H ('$' dollar or string), are pronounced differently depending upon their context. If the characters are followed by a digit then the character is not spoken until the following number has been said. The character then takes on a monetary role and is pronounced 'dollar' or 'pound'.

Example:

```
#45      will say 'forty five pounds'
45#      will say 'forty five hash'
# 45     will say 'hash forty five'
$45      will say 'forty five dollars'
45$      will say 'forty five string'
$ 45     will say 'string forty five'
```

The ASCII character 2DH is always spoken as 'minus'. If this character is occurs as a hyphen between two words it will also be spoken as 'minus'.

Letters, symbols and numbers may be combined to form mathematical equations and so on:

```
3+(1/2)=3.5      spoken as 'three plus one divided by two equals three point five'.
a=34+b           spoken as 'a equals thirty four plus b'
BBC2             spoken as 'b b c two'
```

5.3 Numbers

The text interpreter will say numbers less than ten digits in full, numbers of more than nine digits will have each digit spoken separately. Any leading zeros will be suppressed, except for zeros encountered after a decimal point. Numbers longer than 40 digits will be split-up into sections of 40 digits and each section will be treated separately.

```
123456789        spoken as 'one hundred and twenty three million four hundred and fifty six thousand seven hundred and eighty
                 nine'.
00000001         spoken as 'one'
000002.002       spoken as 'two point zero zero two'
1111111111       spoken as 'one one one one one one one one one one'
```

5.4 Letters

The ASCII characters A-Z (41H to 5AH) and a-z (61H to 7AH) inclusive, are interpreted by a set of phonological rules. Words greater than 40 characters will be split-up into sections of 40 characters, and said separately.

5.5 Text Delimiters

All the punctuation symbols, tabs, spaces, carriage returns and line feeds are used to determine word boundaries. More than one occurrence of a space, tab or newline character is ignored, so that pauses do not seem too long. For example the following two lines will both have the same pause between the words.

```
Hello there!
Hello                                    there!
```

All other characters, including graphics, control codes (except for those used to determine word boundaries) and unpronounceable symbols, such as brackets, are ignored.

6 EXAMPLE PROGRAM

6.1 The Speaking Clock

This example program shows how simply text to speech may be incorporated within a program. The |ECHO command is used to read aloud the input prompts, whilst |SAY is used to speak the time. The time is spoken every five seconds, using a twelve hour cycle.

Before using the program, first make sure that the *SSA1* software is installed.

```
10 REM ****** The Speaking Clock *******
20 REM
30 REM ****** Assumes the SSA1 code has been loaded
40 REM
50 REM ****** Twelve hour cycle. Time spoken every
60 REM         five seconds
70 REM
75 ON ERROR GOTO 500
80 O$="o clock": P$="precisely": SEC$="second": SECS$=SEC$+"s"
90 A$="and"
100 MODE 1: |ECHO,1
110 HOURS=12
120 PRINT "`12 Hour Speaking Clock`": PRINT
130 INPUT "`Enter the time in HOURS,MINS`";HH,MM
140 IF HH>12 OR MM>60 THEN 130
150 S=0
160 MODE 0
170 PEN 3: PRINT "`The Speaking Clock`": PEN 1
180 |ECHO,0: GOSUB 270
190 EVERY 250 GOSUB 210
200 WHILE -1: WEND
210 REM ***** Update the time ************
220 SS=(SS+5) MOD 60230 IF SS<>0 THEN 270
240 MM=(MM+1) MOD 60
250 IF MM<>0 THEN 270
260 HH=(HH+1) MOD HOURS
270 LOCATE 7,11
280 IF HH=0 THEN HH=HOURS
290 PRINT USING "##:##:##";HH;MM;SS
300 H$=STR$(HH): |SAY,@H$
310 IF MM=0 THEN |SAY,@O$: GOTO 330
320 M$=STR$(MM): |SAY,@M$
330 IF SS=0 THEN |SAY,@P$: RETURN
340 |SAY,@A$
350 S$=STR$(SS): |SAY,@S$
360 IF SS>1 THEN |SAY,@SECS$ ELSE |SAY,@SEC$
370 RETURN
500 PRINT: PRINT "The SSA1 software has not been loaded"
510 PRINT: END
```

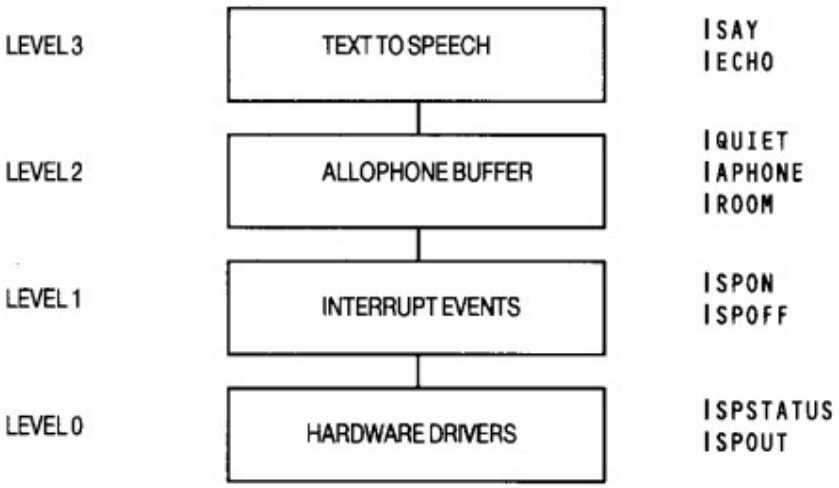
7 ADVANCED COMMAND OVERVIEW

The *SSA1* software provides nine external commands that are usable from BASIC programs. The commands are in the form of Resident System Extensions (RSXs) that must be loaded into memory, before they can be used. The RSXs are fully relocatable, and perform their own initialisation. A simple `LOAD` and a `CALL` from BASIC is all that is required to introduce the commands and initialise the *SSA1*.

In order to utilise the *SSA1* software to the full it is necessary to know how the commands work and what they do. The following sections detail the commands.

7.1 Hierarchy of Commands

There are four levels of hierarchy, these include low level firmware drivers that address the speech processor directly, interrupt driven allophone buffers, and text to speech conversion. The hierarchical diagram below summarises the levels. Each level is intended for a particular application. It is not recommended that the commands from levels 3, 2 and 0 be used simultaneously.



7.11 LEVEL 3 - Text to Speech Conversion

This is the highest level where ASCII text is converted into allophones by using a sophisticated Text to Speech system. The allophones produced are subsequently sent to level 2, the interrupt driven allophone buffer. When the the buffer becomes full, the commands at level 3 suspend BASIC until all the text has been converted.

7.12 LEVEL 2 - Interrupt Driven Allophone Buffer

This buffer is used to temporarily queue up to 64 allophones that are to be output to the speech processor. Its main purpose is to allow a foreground program to continue operation, whilst interrupt events ensure that any data, from the buffer, is output to the speech processor at the appropriate time.

The buffer can be filled from level 3 or by the command `|APHONE` at level 2, either source may insert allophones at any time into the buffer. If the buffer becomes full then the `|APHONE` command will discard any allophones that cannot be stored.

The command `|ROOM` may be used whenever it is necessary to determine the number of spaces left in the buffer. Some status information relating to levels 1 and 0 are also returned by `|ROOM`.

Any data in the buffer may be flushed, and a pause sent to the speech processor with the command `|QUIET`.

The buffer can only be emptied by speech interrupt events occurring at level 1. If the speech interrupts are off then the buffer will fill up. To allow text to speech to operate correctly the `|ECHO` and `|SAY` commands automatically enable the speech interrupts. The `|APHONE` command does not change the state of the interrupts, so care must be taken not to lose allophones if `|APHONE` is used when the buffer is full.

7.13 LEVEL 1 - Speech Interrupt Events

An event may be set up using `|SPON`. This event reads characters from the allophone buffer and sends them to the speech processor. The event may be disabled at any time with `|SPOFF`, this prevents any more characters being read from the buffer and also sends a pause to the speech processor, to ensure that the last allophone is not sounded for ever. When the event is disabled the buffer is left in its current state and may fill up from levels 3 or 2.

7.14 LEVEL 0 - Firmware Drivers

The allophones from level 2 are passed to this level via the interrupt events at level 1, to be output to the speech processor when it is ready to accept another allophone.

Two commands |SPSTATUS and |SPOUT, allow the user to interrogate the status of, and to send allophones directly to, the speech processor. It is recommended that speech interrupts are off, or that the buffer is empty, before using |SPOUT. Otherwise allophones from |SPOUT may be randomly interspersed with allophones being output by the speech event.

Level 0 is provided primarily for special effects and synchronisation. For example it is only possible to ascertain if a particular allophone has been spoken by using |SPSTATUS or |SPOUT.

8 THE COMMANDS IN DETAIL

8.1 |SAY

Use

To convert a string of ASCII text into speech.

Syntax

|SAY,@‹String Variable›

Where:

‹String Variable› represents the string to be converted.

example

Say 'the time is four thirty'.

```
a$ = "The time is 4:30"
|SAY, @a$
```

notes

The speech interrupts are automatically enabled. Any data in the allophone buffer will therefore be output before the allophones produced by the text to speech system. If the length of the string causes the allophone buffer to fill, then the operation of BASIC will be suspended until the entire string has beenconverted. No distinction is made between upper and lower case. Punctuation is converted into pauses of equivalent duration and a short pause is always sent at the end of each word.

Some common abbreviations, such as 'Mr' and 'Dr', are converted into their full word equivalents: 'mister' and 'doctor'. Numbers are spoken as full as possible such that '10' will be pronounced 'ten' instead of 'one', 'zero'. Chapter 5 gives a detailed description of the features of the text to speech system.
The following example will allow you to continually enter the text to be spoken:

```
10 CLS
20 LINE INPUT "What shall I say? ";a$
30 |SAY,@a$
40 PRINT
50 GOTO 20
```

8.2 |ECHO

use

Activates text to speech conversion on text printed to the screen. Five modes of operation are possible that result in printed characters being echoed to the speech synthesiser.

syntax

|ECHO, ‹Mode›

Where:

‹Mode› is an integer in the range 0..4 specifying the type of echo to be applied. The default mode after installation is mode 0. The modes available are:

Mode 0

This disables modes 1..4 and restores the original print condition as found during installation.

Mode 1

All print output enclosed between the delimiters (`) is spoken AND displayed.

Mode 2

All text output to the screen, including listings, syntax errors and so on is spoken, control codes are still sent to the screen.

Mode 3

All text output to the screen, including listings, syntax errors and so on is spoken AND displayed.

Mode 4

All print output enclosed between the delimiters (.) is spoken but NOT displayed.

example

Say a program instead of listing it.

```
|ECHO, 2
LIST
|ECHO, 0
```

notes

All 4 modes will wait if the allophone buffer fills up. The speech interrupts are automatically enabled. If programs are listed in modes 1 or 4 then any text between the delimiters will be acted upon. Care should be taken to ensure that there is always an even number of delimiters.

The delimiter is a (`) ASCII character 60H

Lines longer than 80 characters are automatically output. Any text is interpreted in the same way as the **|SAY** command, Chapter 5 gives a detailed account of the features of the text to speech system.

8.3 |APHONE

use

To send allophones directly to the interrupt driven allophone buffer, bypassing text to speech conversion.

syntax

|APHONE, ‹allophone› [,‹allophone›]

Where:

‹*allophone*› is an integer in the range 0..63. Values greater than 63 will be taken MOD 64. Consecutive allophones may be sent by separating each with commas. The maximum number of allophones that may be specified at any given time is limited by the length of a BASIC command line.

If the buffer becomes full, then the command will terminate. Any allophones encountered after the buffer becomes full will be discarded. A check should be made, using **|ROOM** to ensure there is enough space left in the buffer to accept the allophones, before using this command.

The allophones will only be heard if the speech interrupts are on.

example

Say the word 'hello' without using the text to speech system.

```
|APHONE,27,7,62,53,0
```

notes

Chapter 4 summarises the allophones available on the SSA1 and also presents some guidelines on how to select the allophones for particular words.

8.4 |ROOM

use

The **|ROOM** command may be used to find out how many free locations there are in the allophone buffer, whether the speech synthesiser is currently sounding an allophone, and if the speech interrupts are active.

syntax

|ROOM,@‹integer Variable›

Where:

‹Integer Variable› is the variable into which the result will be passed.
The result is bit significant:

Bits 0..5 the number of free locations in the buffer
Bit 6 indicates the state of the speech interrupts, when set the interrupts are disabled, the buffer will fill up if any more allophones are sent to it.
Bit 7 when set, the speech synthesiser is busy sounding an allophone.

example

Say the word 'hello' only if there are enough spaces in the allophone buffer.

```
10 a%=0
20 |ROOM,@a%
30 IF (a% and &3f) > 5 THEN |APHONE,27,7,62,53,0
40 END
```

notes

If the space returned is zero, then the buffer is full and any more allophones sent using **|APHONE** will be discarded.

8.5 |QUIET

use

To clear the allophone buffer to its empty condition and to send a pause to the speech processor.

syntax

```
|QUIET
```

No parameters should be given.

example

```
10 ON BREAK GOSUB 100
 ..
100 |QUIET : REM Stop generation of speech
110 RETURN
```

notes

This command may be issued at any time, regardless of the state of the speech interrupts, in order to prevent the generation of any speech, due to allophones in the speech buffer waiting to be spoken.

8.6 |SPON

use

To activate the speech interrupt event. The event allows allophones queued in the allophone buffer to be output independently of BASIC.

syntax

```
|SPON
```

No parameters should be given.

notes

No allophones from the buffer will be sent to the speech processor until speech events have been activated with this command. The machine will be slowed down slightly due to the overhead involved in dealing with the event. The events are synchronised with the fast ticker.

8.7 |SPOFF

use

To prevent the output of data from the allophone buffer. This has the reverse effect of **|SPON**, and disables the speech interrupt events.

syntax

```
|SPOFF
```

No parameters should be given.

example

```
10 |SPOFF
20 |APHONE,27,7,62,53,0
30 REM  Main program follows
 .
 .
100 |SPON : REM Say the previously stored word
```

notes

This command will have no effect unless used after a |SPON command. Any allophones remaining in the allophone buffer will remain intact. The output of the data from the allophone buffer may be re-commenced by issuing a |SPON command. After disabling events the |SPOFF command will wait for the currently voicing allophone to finish, before sending a pause, preventing the last allophone from sounding for ever.

It is recommended that |SPOFF is used whenever speech is not required for long durations.

8.8 |SPSTATUS

use

To read the hardware status of the speech processor.

syntax

|SPSTATUS,@‹Integer Variable›

Where:

‹Integer Variable› is a 16 bit integer into which the status will be placed. The status is bit significant:

Bit	Interpretation
0..5	Always returns 0, no meaning attached
6	This is a copy of /LOAD
7	This is a copy of /BUSY
8..15	Always returns 0, no meaning attached

A combination of bits 6 and 7 may be used to ascertain the condition of the speech processor as follows:

bit 7 6	speech condition
0 0	Voicing allophone,
0 1	Voicing allophone, DO NOT load next allophone address
1 0	Not voicing (silent), next allophone address may be loaded
1 1	Not produced by the SSA1

example

Wait for allophones to finish, before sending another.

```
10 a%=0
20 FOR i = 1 TO 5
30  |SPSTATUS,@a%
40  WHILE a% AND 64: |SPSTATUS,@a%: WEND
50  READ b%
60  |SPOUT,b% : REM No handshaking for |SPOUT
70 NEXT i
80 END
90 DATA 27,7,62,53,0
```

notes

This command may be used in conjunction with |SPOUT to test for the presence of the SSA1 hardware. A BASIC program may use |SPSTATUS to achieve synchronisation with spoken allophones, as may be required during animation.

use

To send allophones directly to the speech processor, bypassing the allophone buffer. One of three handshaking methods may be selected, according to the application.

syntax

|SPOUT, ‹allophone› [,‹allophone›]

Where:

‹allophone› is a 16 bit integer value in the range 0..255. Only the lower 6 bits are output as address data to the speech processor. The value is interpreted as follows:

Bit	Interpretation
0..5	Allophone in the range 0..63
6	When set, selects /BUSY mode of handshaking
7	When set, selects /LOAD mode of handshaking
8..15	Should be 0

Additionally bits 6 and 7 select one of 3 handshake modes in which to output the allophone address.

bit 6 7	Mode
0 0	Send allophone address immediately, regardless of /BUSY or /LOAD.
0 1	Send allophone address when /LOAD active, ignore /BUSY
1 0	Send allophone address when /BUSY active, ignore /LOAD
1 1	Send allophone address when /BUSY active, ignore /LOAD

example

Say 'hello' using /BUSY handshaking.

```
10 a%=0
20 FOR I=1 TO 5
30  READ b%: b%=b%+128
40  |SPOUT, b%
50 NEXT I
60 END
70 DATA 27,7,62,53,0
```

notes

Speech interrupts should be disabled with |SPOFF before using |SPOUT, otherwise it is possible that allophones may be randomly interspersed with those being output under interrupt from the allophone buffer. Although this may be an undesirable feature, it can be used to advantage to provide sound effects, in some applications.

9 ERROR MESSAGES

One general error message may be given by any of the commands, it is:

Bad command

The error may be due to either the incorrect number of parameters, an echo mode that is not in the range 0..4, or an attempt to say a null string.

APPENDIX A

A.1 Using the SSA1 Software from Machine Code

The commands available with the *SSA1* provide all the necessary facilities for any envisaged application, the commands however are not restricted for use with BASIC. As the commands are RSXs they can be called from within machine code programs, provided the correct parameters are passed, exactly the same as BASIC would pass them. By invoking the RSXs in this way, the programmer does not need to know the load address, or the hardware configuration of the *SSA1*, in order to produce speech.

From this point on the reader is assumed to have available a CPC464 Firmware Manual. To find out about the interface to an RSX, and how the parameters are passed read chapter 9.6 of the Firmware Manual.

The desired command is called from a program in a two stage process. Firstly KL FIND COMMAND is used to obtain the address of the command. (This firmware call may be used to indicate if the SSA1 software has in fact been installed.) Secondly once the address of the command has been established then the relevant registers should be set up to correspond to the parameters required by the command, and KL FAR PCHL called to invoke the command. It should be assumed that all the primary registers, including IY, are corrupted by the RSX commands.
The following example program shows how to use the ‡SPOFF command from machine code, it should be read in conjunction with the firmware manual.

```
EXAMPLE:    LD  HL, COMMAND$        ; READ ADDRESS OF COMMAND STRING
            CALL KL.FIND.COMMAND    ; SEARCH FOR THE RSX
            JR  NC, NOTFOUND        ; NO SUCH COMMAND LOGGED IN
;
            XOR A                   ; ZERO PARAMETERS REQUIRED
            CALL KL.FAR.PCHL        ; INVOKE THE COMMAND
            RET                     ; SUCCESSFUL
;
COMMAND$:   DEFW "SPOF"
            DEFB "F"+128            ; COMMAND NAME, LAST CHAR HAS BIT 7 SET
;
NOTFOUND:                           ; FAILED TO FIND COMMAND
```

The same procedure can be used for all the SSA1 commands. Remember that interrupts should not be disabled if the allophone buffer is to be used.

APPENDIX B

B.1 Driving the SSA1 Hardware Directly

In some cases it may be necessary to use the SSA1 without loading the RSX code, for example in a dedicated game. This section describes how the SSA1 looks to the programmer and the procedures needed to use it directly.

B.11 Memory Map

The *SSA1* occupies one address in the I/O map, as follows:

ADDRESS	INPUT	OUTPUT
0FBEEH	STATUS	ALLOPHONE DATA

Access to the speech processor is achieved during an I/O operation when A10, A4 and A0 are low. That is, I/O address 0FBEEH.

The SSA1 generates no hardware interrupts, and must be used in a polled fashion.

B.12 Status Register

During a read, the status of the speech processor is returned as a bit significant 8 bit value. The bit positions are assigned as follows:

BIT	INTERPRETATION
0-5	Not used. These bits are undefined
6	/LOAD Allophone addresses may be sent when LOW
7	/BUSY When LOW the processor is talking

B.13 Allophone Address Register

There are a total of 64 addresses, each corresponding to an allophone. Only bits 0-5 make up the address. Bits 6 and 7 should ALWAYS be sent as a LOW.

All the allophones in Table 4B are directly supported.

B.14 Handshaking

There are two possible methods of sending allophone addresses to the speech processor, both make use of polling the STATUS register.

1] Allophones may be written consecutively one after another whenever /LOAD is LOW. The speech processor will allow another allophone to be loaded whilst it is still sounding a previous allophone, its readiness to accept the next allophone is indicated by the /LOAD signal.

2] New data may be written after a preceding allophone has been completely spoken. This is achieved by writing data only when /BUSY is HIGH. In this case there is only ever one allophone outstanding, waiting to be spoken. The speech may be terminated at any time between successive allophones, by sending a pause to the processor. A suitable procedure for loading data using /BUSY handshaking is:

```
BEGIN
  READ STATUS
    WHILE /BUSY=Ø
      READ STATUS
    WEND
  OUTPUT ALLOPHONE
END
```

This method of handshaking is recommended.

The last part of the last allophone spoken will be continuously repeated, until another allophone is addressed. A PAUSE should be sent as the last allophone of any group, to stop the speech processor generating noise.

APPENDIX C

C.1 Hardware Operation

The operation of the *SSA1* can be functionally subdivided into two sections; one digital; the other analogue. Figure C1 outlines their association.

C.11 Digital

The digital section comprises four integrated circuits, one is the SP0256-AL2 allophone speech processor from GI. The remaining three complete the interface of the speech processor to the CPC464 expansion bus.

Whenever an i/o read is performed with A10, A4 and A0 low, two tri-state buffers connected to the data lines D7 and D6 are enabled. The inputs to these buffers are connected to the SBY and /ALD handshake lines of the speech processor. The SBY signal is referred to as /BUSY and /ALD is referred to as /LOAD throughout this guide.

Allophone addresses are loaded whenever an i/o write is performed with A10, A4 and A0 low.

C.12 Analogue

Two separate audio amplifiers are used to provide a stereo output of up to 200mW per channel into 4 ohms.

The stereo output from the CPC464 is mixed with the speech before being amplified. A SOUND command with a volume level of 15 produces an output of 200mW into a 4 ohm load at full volume.

The digital output from the speech processor, a 40kHz pulse width modulated signal, is low pass filtered at 5kHz and then fed symmetrically to each channel. At full volume the speech will generate 200mW into 4 ohms.

The operation of the computer may be heard through the loudspeakers on occasions, this is due stray pickup of computer generated noise manifesting itself in the same way as that heard on the computer's internal loudspeaker.

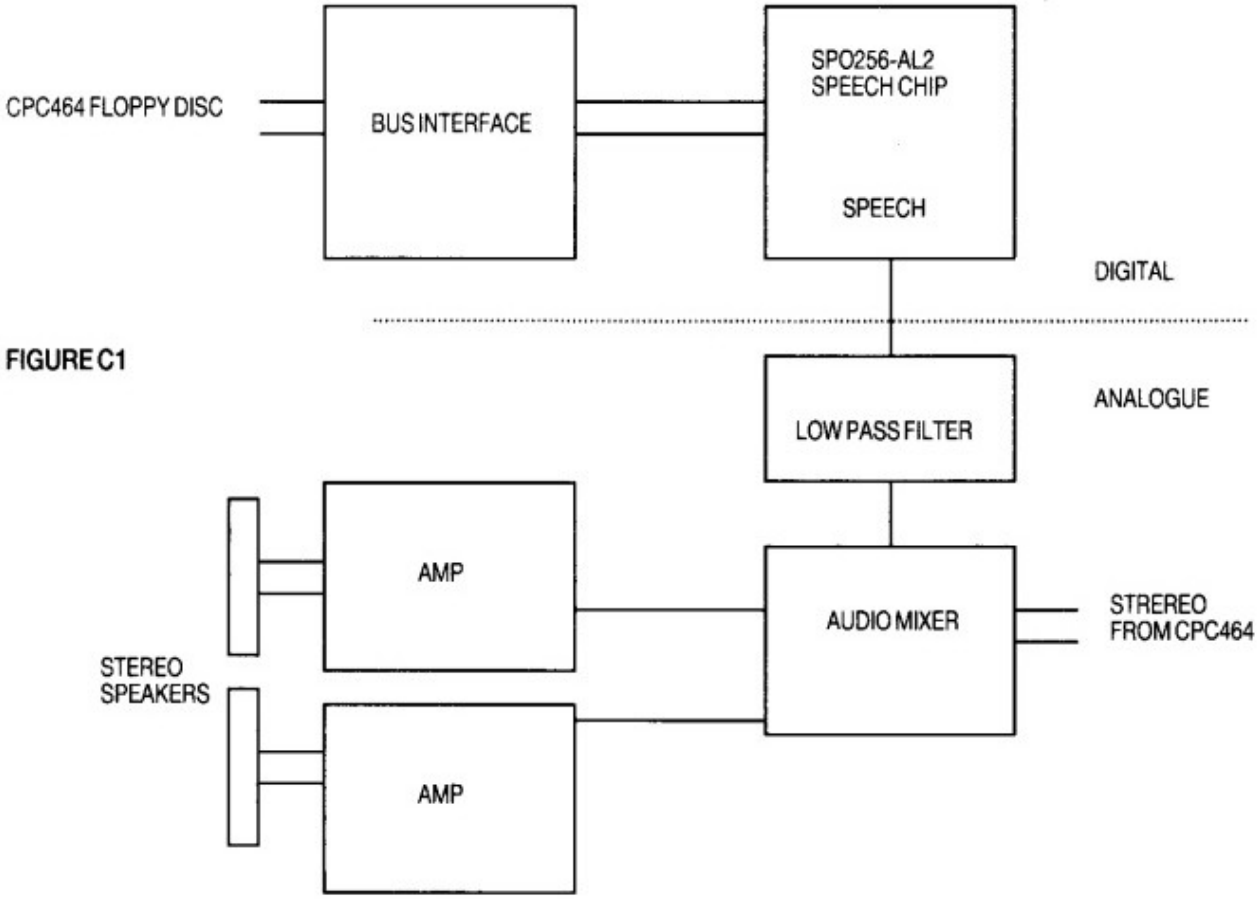


FIGURE C1

APPENDIX D

D.1 Initialising the SSA1 Software

It is possible to load the SSA1 software from within another program, in fact this is what the loader program does. To load the software you must first reserve an area of memory into which the code is to be loaded. The loader program will tell you how much space will be required. Let the load address be BASE, the following lines of BASIC will load, relocate and initialise the SSA1 software:

```
10 LOAD "SSA1.BIN",BASE
20 START=BASE+PEEK(BASE)+PEEK(BASE+1)*256
30 CALL START
```

The first word of the program is the offset from BASE to the start of the relocation routine. This is added to BASE to obtain START, which is then called to relocate and log on the software.

D.2 Testing for the SSA1 Hardware

If you are driving the hardware without the SSA1 software, or the software has been loaded without the using the loader program, then it is desirable to know if the SSA1 hardware is in fact connected. The following procedure outlines how to test for the hardware.

```
BEGIN
  READ STATUS REGISTER
  (* Bit 7 should be 1, bit 6 at Ø *)
  IF STATUS AND ØC0H = 80H THEN
    (* So far so good, now send some data *)
    DATA = 3 (* 100mS pause *)
    OUTPUT DATA TO ADDRESS REGISTER
    (* Bit 7 should now go to Ø *)
    READ STATUS REGISTER
    IF STATUS AND 80H = Ø THEN
       SUCCESS (* Hardware functions *)
    ELSE
       FAIL (* Not there ? *)
  ELSE
    FAIL (* Not there ? *)
END
```

The above procedure can be implemented either in machine code, BASIC, or by using the |SPOUT and |SPSTATUS commands. Obviously if the *SSA1* code has not been loaded, then it will be necessary to directly address the hardware. The program below checks for the hardware by sending a pause and testing to see if the /BUSY line has gone low.

```
10 ssa1%=&FBEE
20 d%= INP(ssa1%) AND &C0
30 IF d%<>&80 THEN 70
40 OUT ssa1%,3
50 d%= INP(ssa1%) AND &C0
60 IF d%=0 then PRINT "Hardware OK": END
70 PRINT "Hardware not working ???": END
```

D.3 Testing for the SSA1 Software

Sometimes it is useful for a program to know if the *SSA1* software has been installed. A BASIC program can determine if the software is present by issuing any one of the commands, and trapping any error messages. An error 28, 'Unknown command' will be generated if the software has not been installed. The program below shows how this is done.

```
10 ON ERROR GOTO 50
20 |QUIET
30 PRINT "Software installed"
40 END
50 IF ERR=28 THEN PRINT "No SSA1 software installed"
60 END
```

D.4 Further examples

Another example program lets you write up to 9 different phrases which can then either be spoken, or overwritten:

```
10 CLS
20 PRINT:PRINT "Press W to Write, press S to Say"
30 IF INKEY(59)<>-1 THEN 50
40 IF INKEY(60)<>-1 THEN 80 ELSE 30
50 PRINT "WRITE which";:GOSUB 110
60 PRINT "Now enter phrase number";n:LINE INPUT p$(n)
70 GOTO 20
80 PRINT "SAY which";:GOSUB 110
90 IF p$(n)="" THEN PRINT "Nothing to say!" ELSE |SAY,@p$(n)
100 GOTO 20
110 PRINT " phrase number (1 to 9)?";
120 a$=INKEY$:IF a$="" THEN 120
130 n=VAL(a$):IF n<1 THEN 120 ELSE PRINT n
140 RETURN
```

AMSTRAD SSA-1
Speech Synthesiser System for the CPC464

First edition 1985
Written by Chris Honey
Typeset by Amsoft Computer Graphics

CPC6128 Amstrad

vesta.homelinux.free.fr/wiki/cpc6128_amstrad.html

Sommaire

Description

L'Amstrad CPC 6128 etait dans les années 80 un ordinateur domestique de la gamme Amstrad CPC comportant 128 Ko de RAM, 48 Ko en ROM et utilisant le langage Locomotive BASIC 1.0. Il était l'un des premiers ordinateurs domestiques a être équipé de d'un écran couleur ainsi qu'un lecteur de disquette (3").

Il utilisait un processeur Zilog Z80A (8 bits) à 4 MHz

Voir L'Amstrad CPC 6128 sur Wikipedia

Documentation

- Documentation commandes BASIC (cpcinfo.txt)
- Amstrad Faq

Questions posées fréquemment de **comp.sys.amstrad.8bit** (http://genesis8.free.fr/amstrad/faq/french.php)

Périphériques

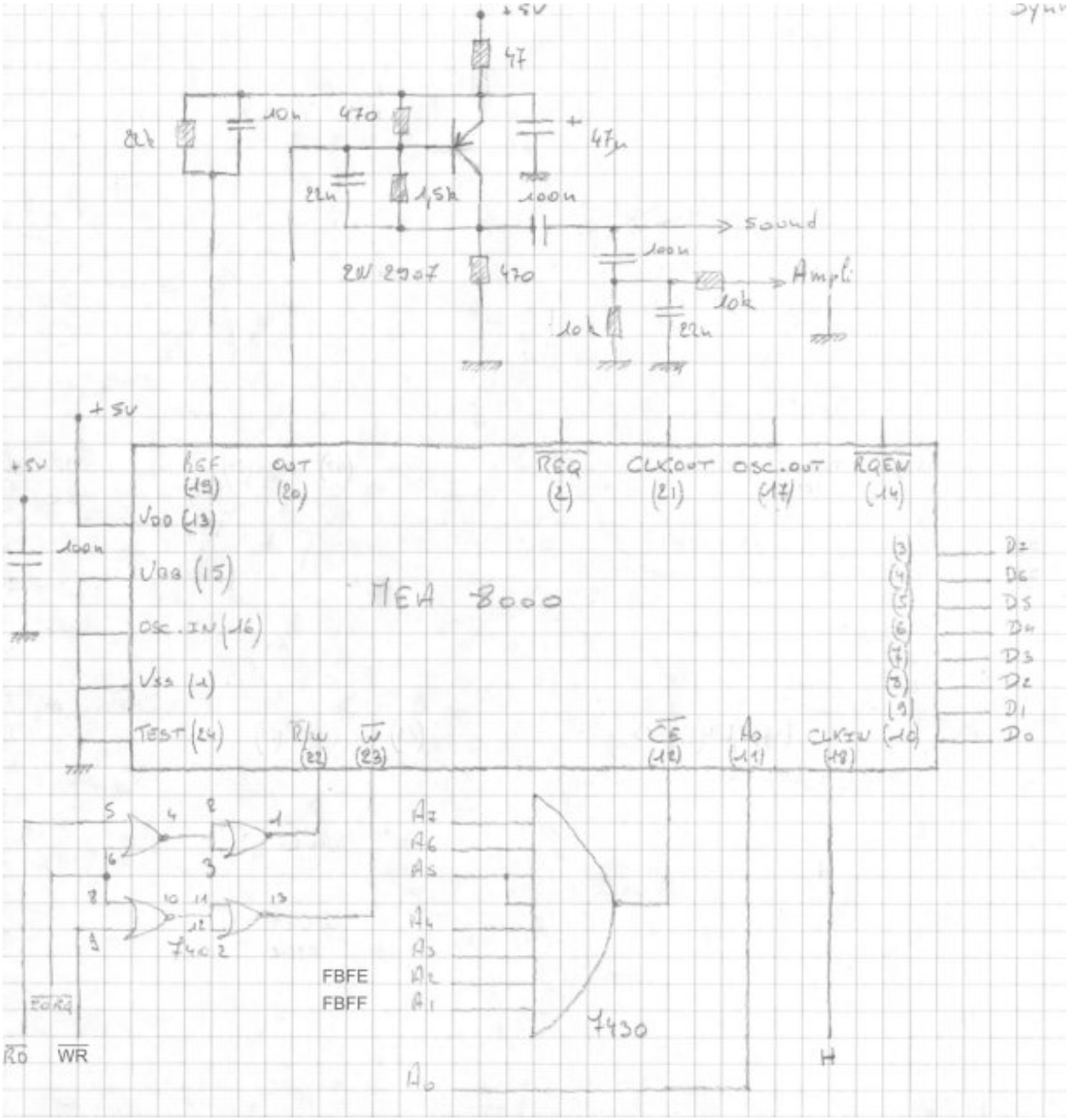
Périphériques commerciaux

Imprimante Amstrad DMP2000

joystick

Synthétiseur vocal Techni-musique

Shéma éléctronique du synthétiseur (utilise le CI ME8000):



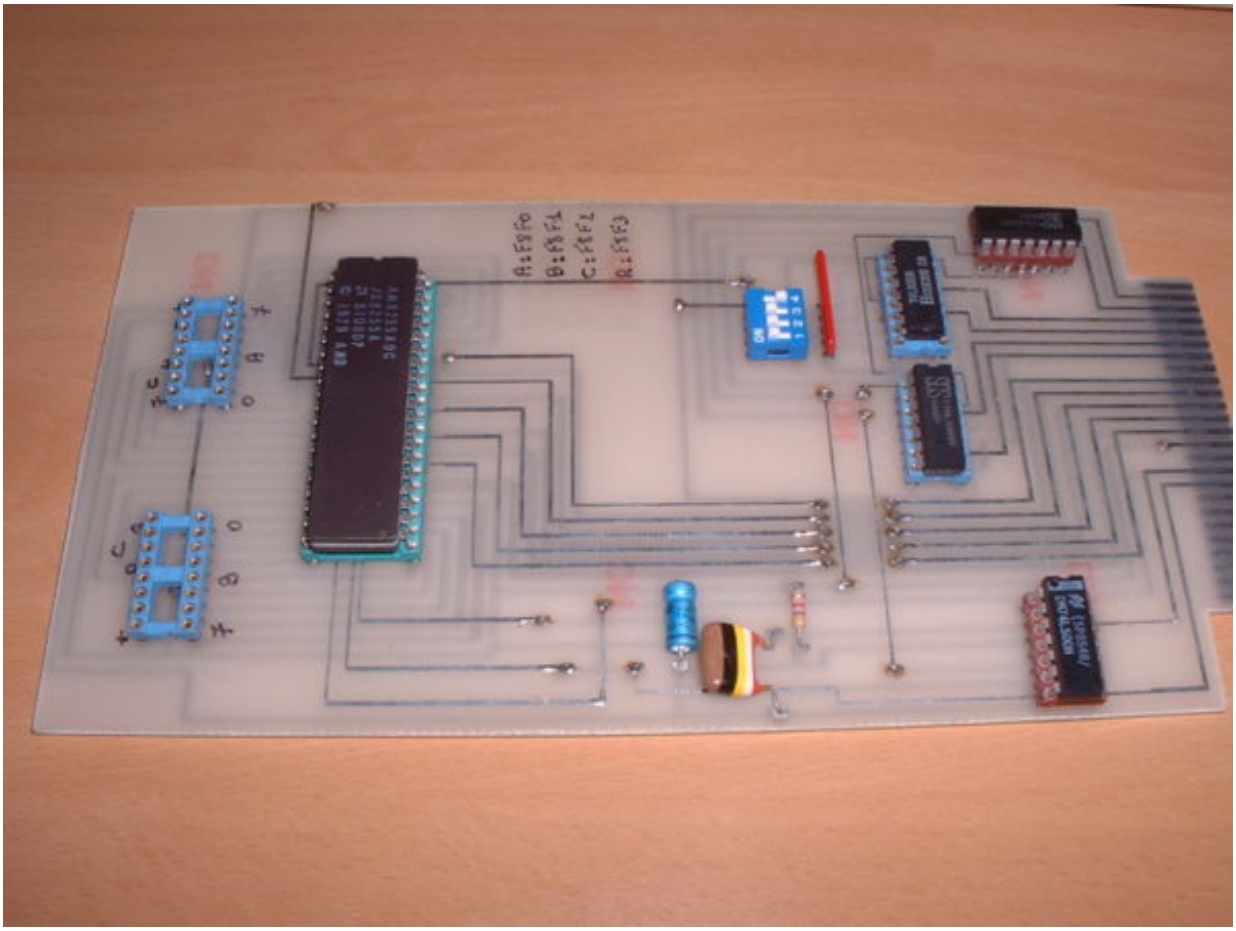
Shéma des photocopies du livre MEA8000.

Vocabulaire PHON100, (Page 1, 2, 3 , 4) logiciel fourni avec la carte.

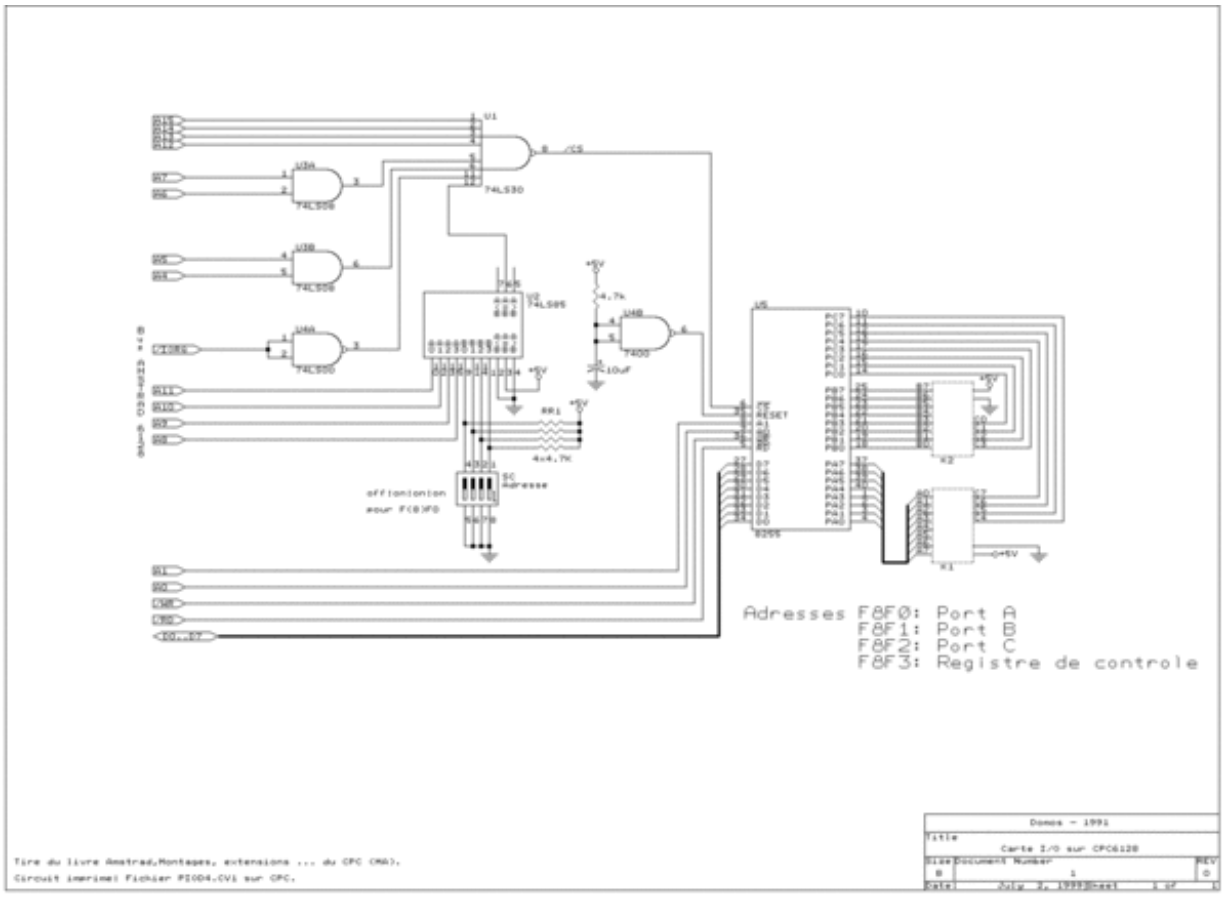
Montages personnels

Interface Entrées/Sorties à circuit 8255

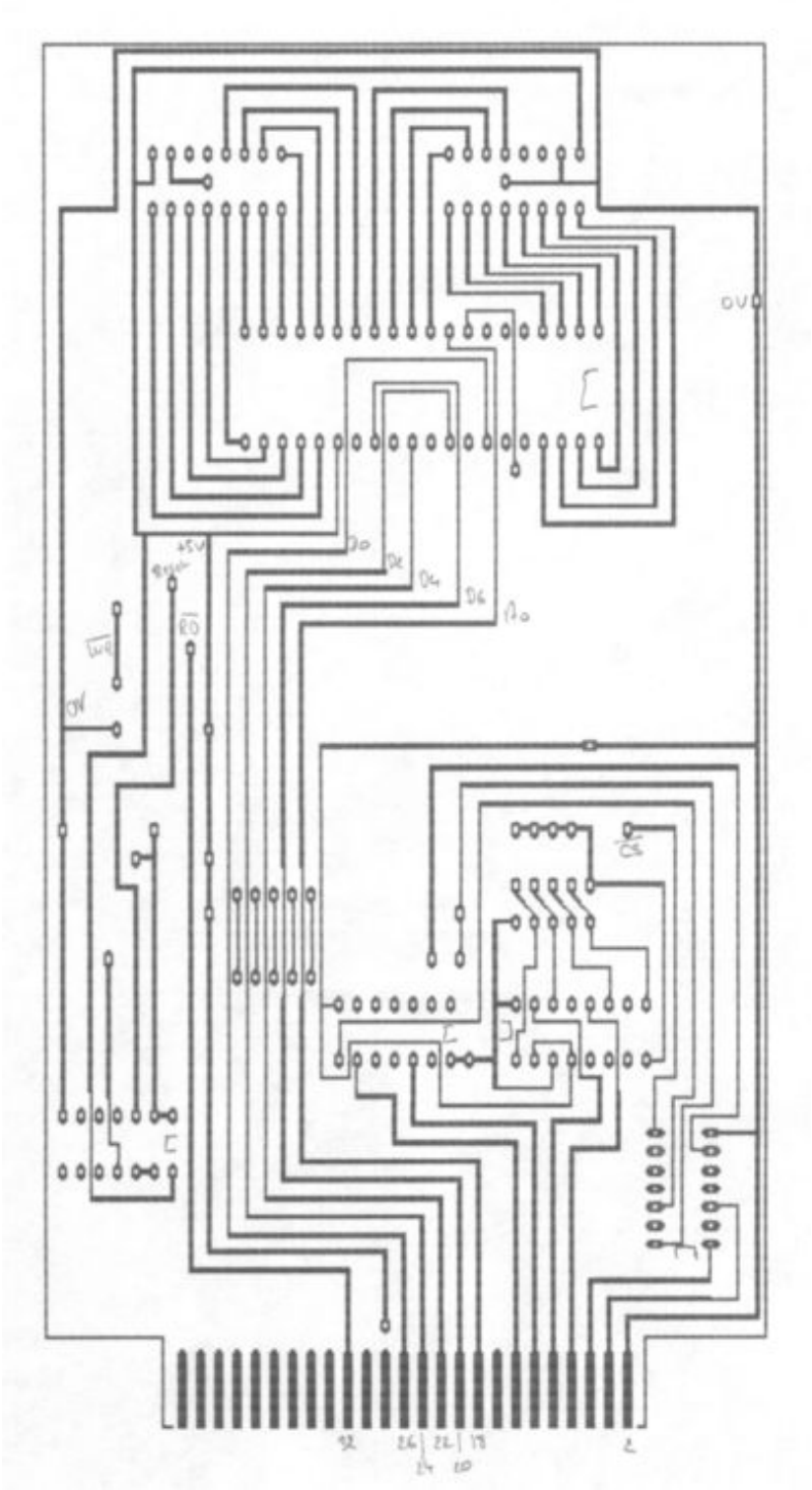
Tiré du livre "AMSTRAD Montages, extensions et périphérique du CPC", Edition Micro-Application. (Carte réalisatée en 1989 / 1990)

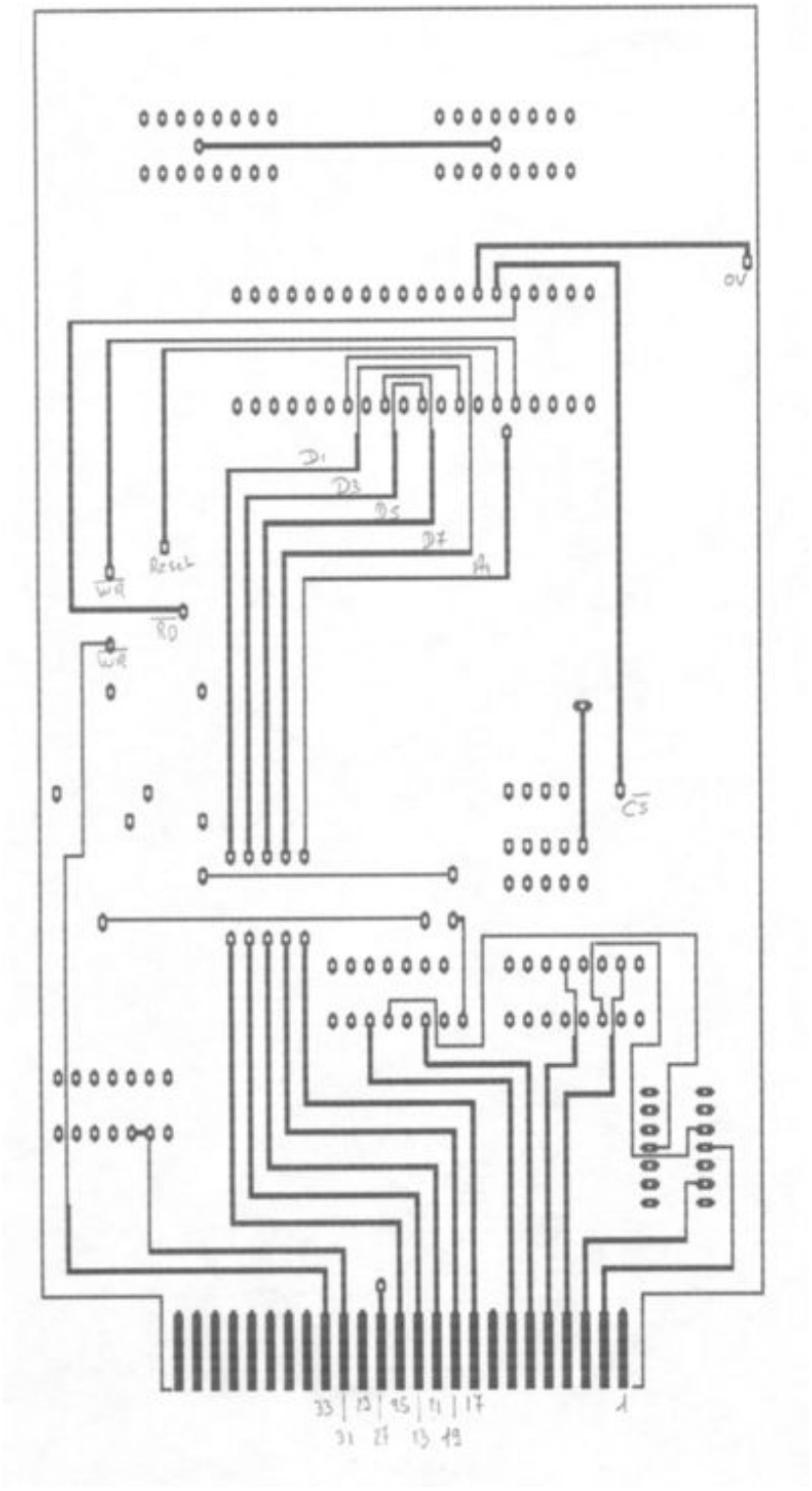


Shéma:


74LS30
74LS05
74LS00
4x4.7K
Adresses F8F0: Port A
F8F1: Port B
F8F2: Port C
F8F3: Registre de controle
Carte I/O sur CPC6128

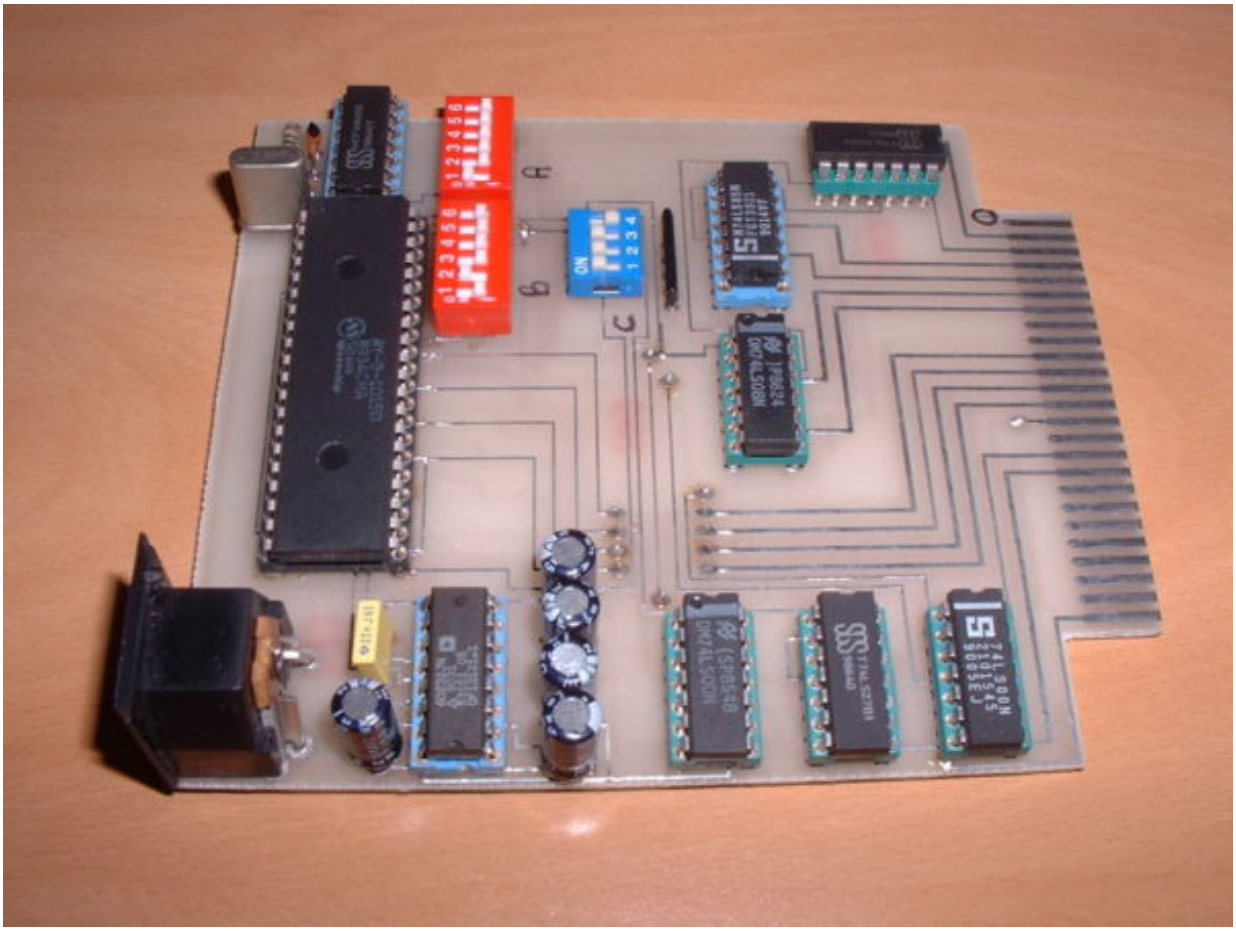
Images CI réalisées avec **CIAO** sur Amstrad (PIOD4.CV1 à 4, disquette fichier emulateur CI_PIO.DSK):



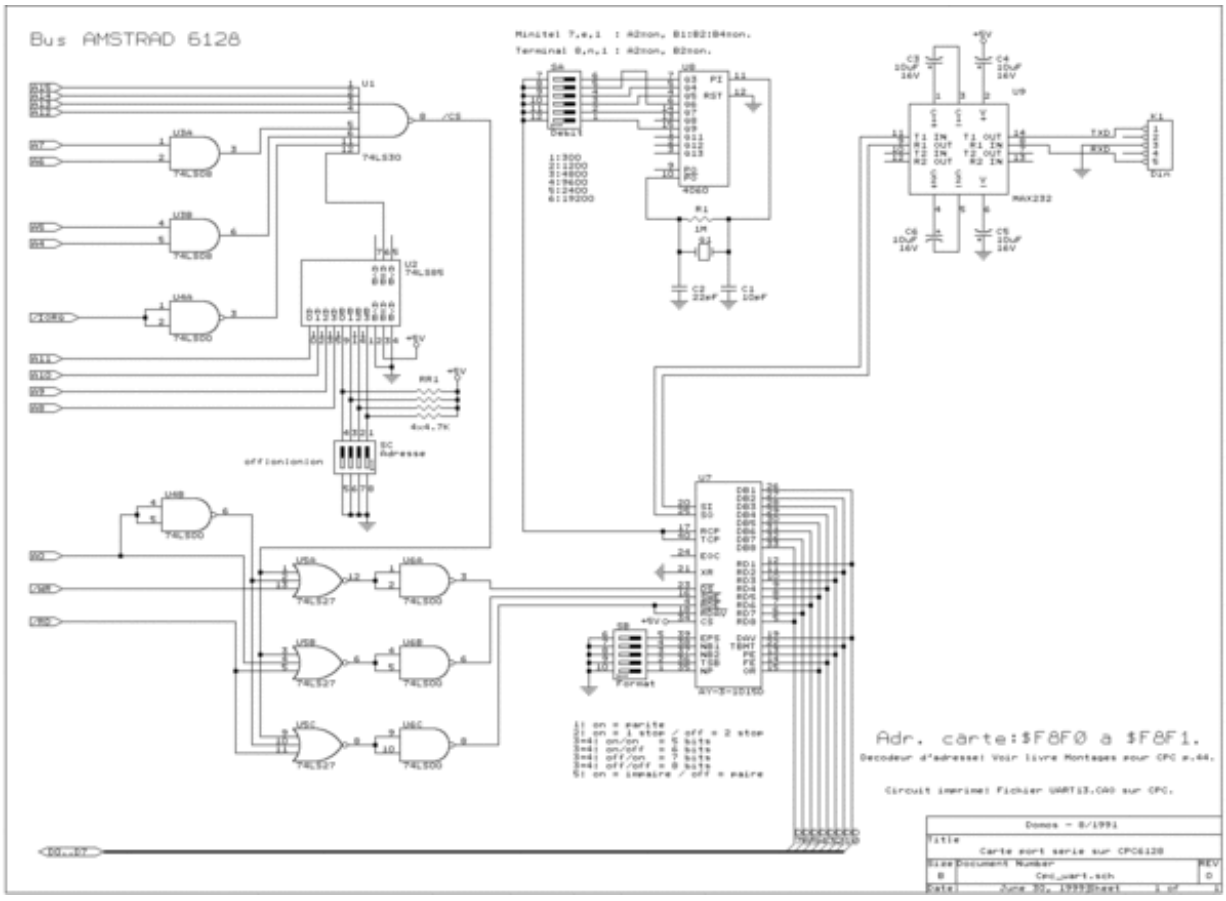


Interface RS232 à CI

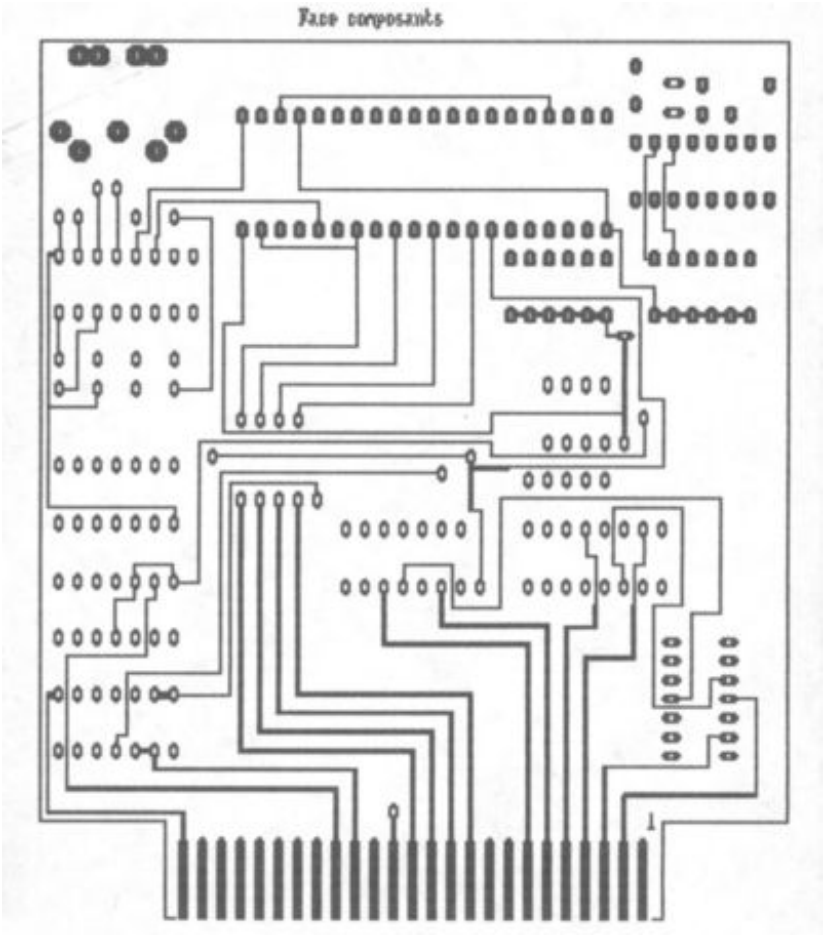
Cette carte est un montage personnel à base du circuit **AY-3-1015D**, seul l'interface à été reprise de la carte I/O à 8255 (Réalisée vers 1989 / 1990).

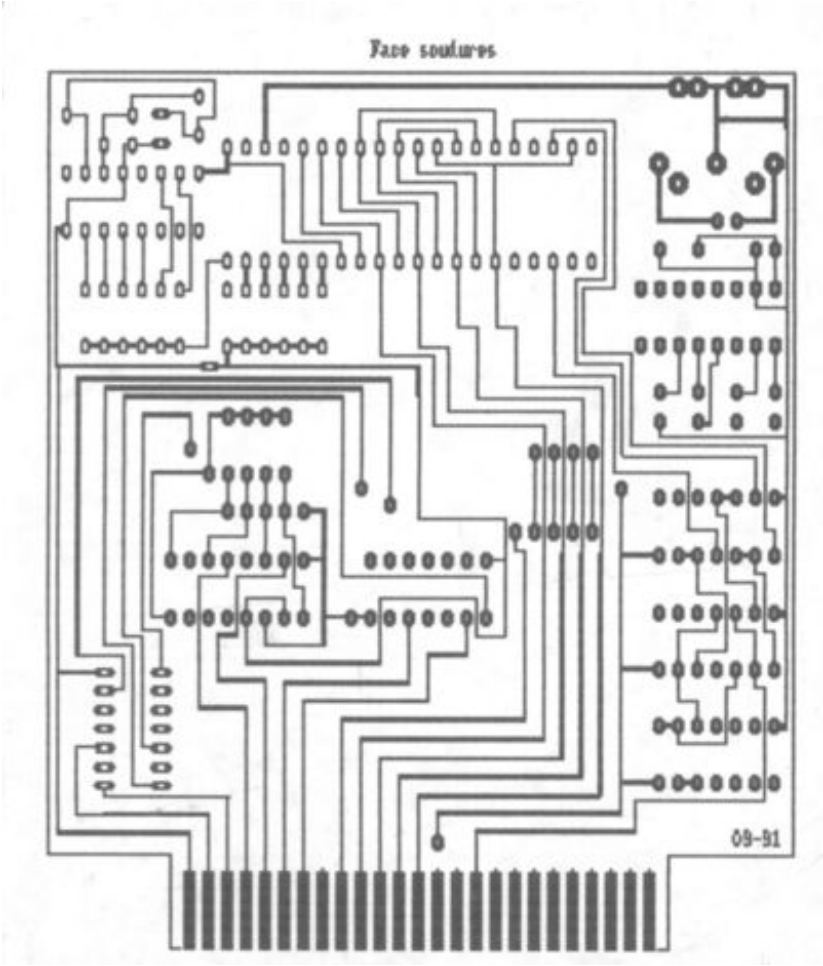


Shéma:


Bus AMSTRAD 6128
Adr. carte:$F8F0 à $F8F1.
Carte sort serie sur CPC6128

Images du CI réalisé avec **CIAO** sur Amstrad CPC 6128 : (UART14 .CC1 à 4, disquette fichier emulateur CI_UART2.DSK)

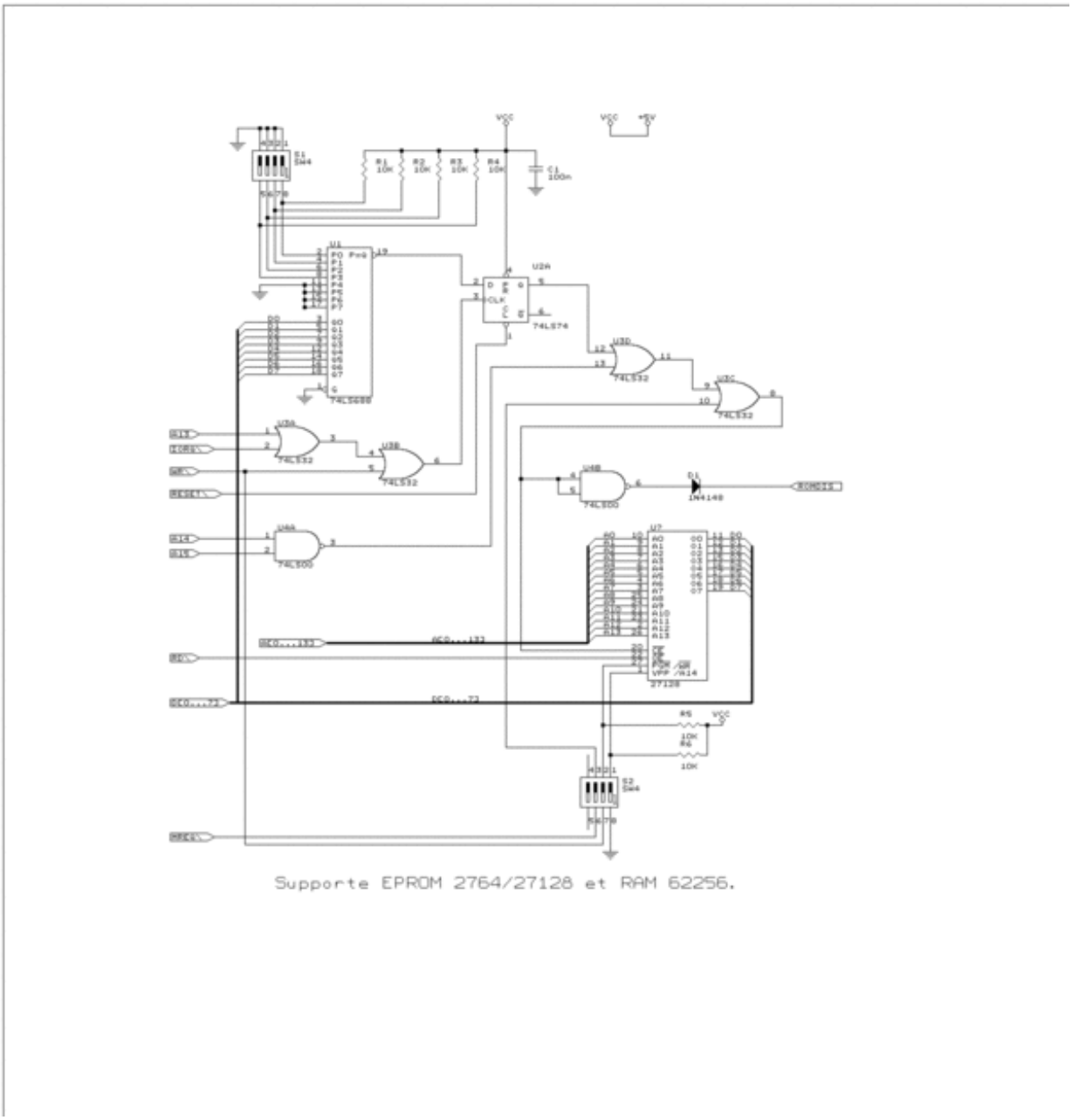




Extension ROM

Le montage a été testé sur plaquette d'essais, le CI dessiné mais jamais réalisé. Dessin du CI réalisé sous **CIAO** sur Amstrad (ROMEX12.CC1 à 4, disquette fichier emulateur CI_ROMEX.DSK)

Shéma:



Lecteur de disquettes 5"1/4

Emulateur CPC6128 sur PC

- CPCemu (http://www.cpc-emu.org/download.html)

Emulateur pour PC version Windows et Linux.

Photos

CPC6128 + Lecteur disquette 5"1/4

Clavier/UC

Lecteur disquette 5"1/4

Page assembleur DAMS

Lecteur disquette 3" interne

CPC6128

Clavier

Interface i/o et série

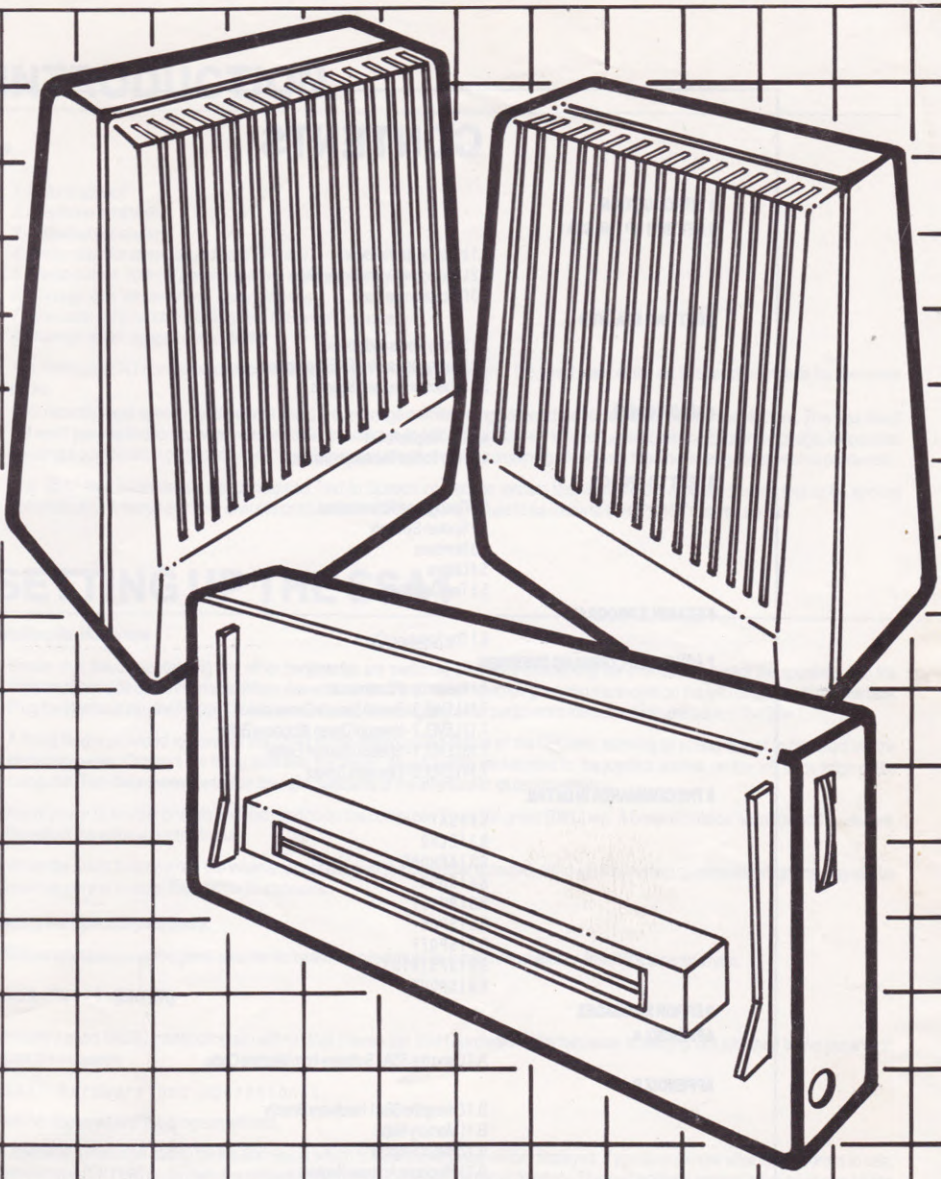
Divers

- Hardware problems

```
Internal drive 07/22/2000

  If you have the error : 'disk missing', the drive belt should be the
 problem. The best solution is to come with your old belt in an
 electronics shop and see the available belts. You should look for
 one with the dimensions 72mm x 3 mm x 0.5 mm (although I believe it
is
 OK to use belts in the length range of 69-72 mm long and either 3 or
 4mm wide).

  You can find belts at Paris (75011), reference Koenig 7093.00 for
 22 FF at Espace Composants Electronique, 66 rue de Montreuil,
 métro Nation, phone 01 43 72 30 64, fax 01 43 72 30 67,
 web http://www.ibcfrance.fr

  Cibotronic at Paris (France) used to sell them, but they
 don't have them anymore. The reference was MASTER type CR 4092,
 dimensions 71.0 x  0.6 x 2.8 mm.
```

1989

Catégorie : Micro



SSA-1

User Instruction Manual

CONTENTS

1 INTRODUCTION

Features

1. *Natural speech*
2. *Allophone synthesis*
3. *Unlimited vocabulary*
4. *Audio mixer for stereo sound from CPC464*
5. *Stereo output, 200mW per channel into 4 ohms*
6. *'Through Bus' for peripheral 'daisy chaining'*
7. *Nine extra commands including Text to Speech conversion*
8. *Interrupt driven speech output buffer*

The Amstrad *SSA1* combined speech synthesiser and amplifier represents the latest state of the art in speech synthesis for the home micro.

Until recently most speech synthesisers could only manage a limited range of words, decided upon during manufacture. This was fine if the word you wanted to say what was available. In contrast, the *SSA1* can say any word or sentence in the English language. It does this by using a combination of discrete speech sounds called allophones. By combining allophones together intelligible speech is produced.

The *SSA1* also incorporates a sophisticated Text to Speech conversion system that allows most arbitrary English text to be spoken automatically. A stereo amplifier ensures crisp speech whilst allowing full use to be made of the CPC464's stereo sound.

2 SETTING UP THE SSA1

2.1 Installing the Hardware

Ensure that the computer and any other peripherals are switched off before connecting the interface. Connect the speakers into the sockets either side of the interface. When viewed from the front, the left speaker goes into the socket on the left hand side of the interface. Plug the interface into the Floppy Disc expansion port, then plug any other peripherals (disc drive) into the back of the SSA1.

A flying lead is provided to connect the *SSA1* to the stereo sound output of the CPC464, allowing all sound output to be heard via the stereo speakers. Connect the flying lead into the stereo sound output socket next to the joystick socket, on the left back edge of the computer. Turn the volume control on the right hand side of the interface to its central position.

Apply power to any peripherals and then switch on the computer. Press the green **[DEL]** key. A beep should be heard on both speakers. Re-adjust the volume control to suit.

When the *SSA1* is connected the internal loudspeaker of the computer is muted, although the loading of cassette programs may still be heard as normal through the internal loudspeaker.

2.2 Loading the Operating software

Before any speech can be generated the software supplied must be loaded into memory. Insert the tape and type:

```
RUN "SSA1 [ENTER]
```

Initially a short BASIC loader program will run that checks the *SSA1* hardware. If the hardware is faulty or not attached to the expansion port the message:

```
SSA1 hardware not operational
```

will be displayed and the program will end.

Otherwise a message asking for the address at which to load the software will be displayed. If you do not know what load address to use, simply type **[ENTER]** on its own, the software will then be loaded at the top of memory. The load address entered must be above 16384 and below `HIMEM`. The program will tell you how much memory is required by the software. Note that `HIMEM` will be set to prevent BASIC from using the same memory.

After the load address has been entered the main *SSA1* software will be loaded. When complete the message 'hello' should be heard. The speech synthesiser is now ready for use.

If you attempt to load the *SSA1* software in more than once, the message:

```
SSA1 software already loaded.
```

will be spoken and displayed.

The *SSA1* software you have just loaded is a special kind of machine code program. It allows extra commands for speech to be used in a BASIC program. To use the commands you simply type in their name (see Getting Started) together with any other variables as needed.

The *SSA1* software remains in memory, even after a NEW command, until a reset is performed, (using **[CTRL] [SHIFT] [ESC]**). In other words, the only time the *SSA1* software has to be installed is when the machine is switched on or reset.

2.3 Precautions In Use

The following precautions should be observed.

Always ensure that the computer and any other peripherals attached on the through bus (disc drives etc.) are switched off before connecting or disconnecting the interface.

Never short circuit the loudspeaker outputs for long periods of time.

Do not use loudspeakers with a lower impedance rating than 4 ohms per channel.

Loudspeakers generate magnetic fields. ALWAYS ensure that discs and cassettes are kept well away from the loudspeakers.

Never place the loudspeakers near the monitor screen, never place them on top of the monitor. (Permanent picture or colour distortion may result due to the magnetisation of part of the screen).

3 GETTING STARTED

The *SSA1* software provides nine external commands allowing speech to be generated for any application, the commands are:

```
|SAY,@‹STRING VARIABLE›
|ECHO,‹MODE›

|QUIET
|APHONE,‹INTEGER VARIABLE› [,‹INTEGER VARIABLE›]
|ROOM,@‹INTEGER VARIABLE›

|SPON
|SPOFF

|SPSTATUS,@‹INTEGER VARIABLE›
|SPOUT,‹INTEGER VARIABLE› [,‹INTEGER VARIABLE›]
```

Chapter 8 describes these commands in detail. The next section presents a few simple examples demonstrating how easy the *SSA1* is to use.

3.1 Simple Words and Phrases

The SSA1 contains a Text to Speech system that looks at text and converts characters into allophones, the fundamental sounds of speech, that the *SSA1* hardware can understand.

Two commands `|SAY` and `|ECHO` perform text to speech conversion. The `|SAY` command takes a string and says it. The `|ECHO` command can be set to look at text printed to the screen, so that listings and printed characters can be heard.

Type `NEW` and then the program:

```
10 a$="simple words"
20 |SAY,@a$
```

(The `|` character can be obtained with **[SHIFT] [@]**)

Now `RUN` the program and observe what happens. The program finishes before all the words are spoken. The software uses a speech queue, which works in a similar way to the BASIC SOUND queue. Essentially it allows the allophones to be spoken in order, transparently from BASIC. In other words it allows BASIC to continue a program without having to wait for speech to finish. The queue has room for a maximum of 64 allophones, around 8 seconds of speech. If there is no room left in the queue then the `|SAY` and `|ECHO` commands will wait for room to be made, it is only then that BASIC will pause, until all the speech has been sent to the buffer.

The string used with `|SAY` can contain any of the 256 ASCII characters, some characters such as `+ - &` are spoken, whilst others like `[]` are ignored. For a fuller explanation of the Text to Speech system refer to Chapter 5.

The `|ECHO` command is intended for saying phrases normally sent to the screen.

Type in the program:

```
10 |ECHO,1
20 INPUT "'How old are you '";age$
30 PRINT "'You are";age$;" years old'"
40 |ECHO,0
```

Line 10 instructs the SSA1 to say text printed to the screen between the delimiters (:), line 40 reverts printing back to normal. The (:) character is obtained with **[SHIFT]** [\], the key immediately above the green **[CTRL]** key.

RUN the program and enter your age as a number. Notice that numbers are spoken exactly as you would expect. Now type in:

```
|ECHO,3
LIST
```

The program will be listed and spoken, in fact anything printed by BASIC including syntax errors, messages etc. will be spoken. To cancel the speech echo enter:

```
|ECHO,0
```

Here are some more examples:

1. Say the alphabet

```
10 FOR letter = ASC("A") TO ASC("Z")
20  a$=CHR$(letter)
30  |SAY,@a$
40 NEXT
```

2. Say the numbers 1 to 10

```
10 |ECHO,3
20 FOR number = 1 to 10
30  PRINT number
40 NEXT
50 |ECHO,0
```

You should now be able to use speech from within any of your programs, for more details of the commands refer to Chapter 8.

3.2 Loading and Saving BASIC Programs

Essentially there is no difference between the way programs are normally loaded or saved, and the way they are loaded or saved with the *SSA1* installed. However, the the *SSA1* software must always be installed, as outlined in section 2.2, after switching on, or resetting, the computer.

Your BASIC programs can be saved as usual with the SAVE command. For example:

```
SAVE "MYPROG"
```

When LOADing programs that use the speech commands you must ensure that the *SSA1* software has been installed first, otherwise the speech commands will not be present, and BASIC will give an error message when the program is run. Remember, the *SSA1* software is installed as explained in section 2.2.

After installing the *SSA1*, your BASIC programs may be loaded as usual with the LOAD command, for example:

```
LOAD "MYPROG"
```

3.3 Experimenting with sounds

The English language has many irregulations in pronounciation for example "Cough", "bough". The SSA1 software contains a large number of pronounciation rules but sometimes it will be necessary to experiment with character spelling to achieve the exact results.

Cough = COF
Bough = BOUH

4 ALLOPHONES

As mentioned previously, the Text to Speech system converts text into allophones. It is not possible to convert every word successfully into the correct allophones because of the many spelling-pronunciation irregularities in the English language. To allow difficult words to be spoken correctly, the |APHONE command can be used to speak allophones chosen by yourself.

Type NEW and then this program:

```
10 |SPON
20 |APHONE,42,15,16,9,49,22,13,51,3
```

When the program is RUN the word computer is spoken. The numbers after the |APHONE command are the allophones that make the word computer. Each allophone is stored in turn in the speech queue, ready to be spoken. Unlike the |SAY and |ECHO commands the allophones are not spoken immediately, but wait for the speech queue to be turned on with the command |SPON. Once the speech queue is on, any further allophones will be spoken until the queue is turned off, or held, with |SPOFF. To demonstrate the queueing effect, change line 10 to:

```
10 ISPOFF
```

This time when the program is RUN nothing is heard. This is because the queue is held. Now type in:

ISPON [ENTER]

Computer should now be heard. More information on these commands can be found in Chapter 8.

4.1 The Allophone Set

There are a total of 59 allophones and 5 pauses available on the *SSA1*. To use the allophone set successfully to synthesise words, the following points should always be remembered:

1. Speech sounds do not always appear acoustically the same, they may be spoken differently, even within the same word.
2. Like the pieces of a Jig-saw puzzle, the speech sounds are separate. Combinations of them must be used to make up the entire 'picture' of how a word sounds.
3. The letters of the alphabet do not have their own speech sound. Each letter may be spoken in a different way depending upon its position within a word.

To illustrate these points try saying the following words one after another:

```
call cup coop
```

You should feel the point of contact between the back of your tongue and the roof of your mouth move further back for each successive 'c' sound.

The 'c' at the start of each word is known as a phoneme. Phonemes themselves cannot be spoken. When the word is spoken a different sound of 'c' is produced, and these different sounds are known as allophones. Thus allophones are different ways of saying phonemes. Most phonemes are influenced by other phonemes around them, and this is why the same letter within a different word can make a different sound.

The allophone set (see Tables 4A and 4B) contains two or three sounds of some phonemes. It may be necessary to use one allophone of a particular phoneme at the start of a word or syllable, and another for the end of a word or syllable. A detailed set of guidelines for using the allophones are given in Table 4B. Note that these are suggestions, not rules.

As an example, /DD2/ sounds good in an initial position and /DD1/ sounds good in final positions, like in 'daughter' and 'collide'. One of the differences between the initial and final versions of a consonant may be that an initial version is longer. Therefore, to create an initial /SS/, two /SS/s can be used instead of the usual single /SS/ at the start of a word or syllable, as in 'sister'. Note that this can be done with /TH/, and /FF/, and the inherently short vowels (refer to Table 4B), but with no other consonants.

Generally it is best to experiment with some consonant clusters (strings of consonants such as str,cl), in order to discover which version works best in the cluster.

For example /KK1/ sounds good before /LL/ as in 'clown', and /KK2/ sounds good before /WW/ as in 'square'. One allophone of a particular phoneme may sound better before or after back vowels and another before or after front vowels. The /KK3/ allophone sounds good before /UH/, and /KK1/ sounds good before /IY/, as in 'cookie'.

Some sounds (/PP/ /BB1/ /BB2/ /TT1/ /TT2/ /DD1/ /DD2/ /KK1/ /KK2/ /KK3/ /GG1/ /GG2/ /GG3/ /CH/ and /JH/) require a brief duration of silence before them. For most of these, the silence is included in the allophone, but more may be added as desired. The pauses PA1 to PA5 may be used to provide silences of the appropriate duration.

When using allophones always think about how a word SOUNDS, not how it is spelt. For example, the /NG/ allophone obviously belongs at the end of the words 'sing' and 'long', but notice that the /NG/ sound is represented by the letter N in 'uncle'. Some sounds may not be represented in words by any letters, such as the /YY/ sound in 'computer'.

Most vowels can be doubled to make longer versions for stressed syllables. These are the inherently short vowels /IH/, /EH/, /AE/, /AX/, /AA/, and /UH/. For example, in the word 'extent' one /EH/ is used in the first syllable, which is unstressed, and two /EH/s are used in the second syllable, which is stressed. One of the inherently long vowels has a long and short version. The short one /UW1/, sounds good after /YY/ in 'computer'. The long version, /UW2/, sounds good in monosyllabic words like 'two'.

Included in the vowel set is a group called R-coloured vowels. These are vowel plus R combinations. For example, the /AR/ in 'alarm' and the /OR/ in 'score'. Of the R-coloured vowels there is one which has a long and short version. The short version is good for polysyllabic words with final /ER1/ sounds like 'letter', and the long version is good for monosyllabic words like 'fir'.

One final suggestion when creating sentences is to add a pause of 30-50msec between words and a pause of 100-200msec between sentences. Always remember to send a pause after the final allophone to suppress the last sound made by the speech synthesiser.

4.2 How To Use The Allophone Set

Table 4A should be used initially when trying to split words into their allophones. The allophones given, make the sound of the letters, and they can be used to make up any word in the English language.

For example to find the allophones for the word 'hello', start with the first letter of the word, thinking about how the word sounds. The first letter is 'h', and the allophone for 'h' is 27. The next letter is 'e', and its allophone is 7. The double 'll' sound can be made with (ll), whose allophone is 62. The 'o' at the end of the word makes the sound like the letter 'O' in the alphabet, the same sound is also made by (eau), whose allophone is 53. Lastly all the allophones must be put into order followed by a pause, a short pause PA1 is ideal. The corresponding IAPHONE command would be:

```
IAPHONE,27,7,62,53,0
```

For more complicated words that sound 'funny' it may be necessary to refer to table 4B in addition to table 4A for extra information on the use of a particular allophone.

TABLE 4A - Quick Guide to the allophones

Sound	Allophone	Section in table 4B
a	24	4B.11
b	28	4B.51
c	8	4B.52
d	21	4B.51
e	7	4B.11
f	40	4B.42
g	36	4B.51
h	27	4B.42
i	12	4B.11
j	10	4B.2
k	42	4B.52
l	45	4B.3
m	16	4B.6
n	11	4B.6
o	23	4B.11
p	9	4B.52
r	39	4B.3
s	55	4B.42
t	17	4B.52
u	15	4B.11
v	35	4B.41
w	46	4B.3
y	49	4B.3
z	43	4B.41
(aa)/(ay)	20	4B.12
(ee)	19	4B.12
(ii)	6	4B.12
(oo)/(eau)	53	4B.12
(bb)	63	4B.51
(dd)	33	4B.51
(gg)	61	4B.51
(ggg)	34	4B.51
(hh)	57	4B.42
(ll)	62	4B.3
(nn)	56	4B.6
(rr)	14	4B.3
(tt)	13	4B.52
(yy)	25	4B.3
(ar)	59	4B.13
(aer)	47	4B.13
(ch)	50	4B.2
(ck)	41	4B.52
(ear)	60	4B.13
(eh)	26	4B.11
(er)	51	4B.13
(err)	52	4B.13
(ng)	44	4B.6
(or)	58	4B.13
(ou)	22	4B.12
(ouu)	31	4B.12
(ow)	32	4B.12
(oy)	5	4B.12
(sh)	37	4B.42
(th)	29	4B.41
(dth)	18	4B.41
(uh)	30	4B.11
(wh)	48	4B.42
(zh)	38	4B.41

TABLE 4B - Allophones by letter group

4B.1 VOWELS

4B.11 Short - These allophones can be doubled

Address	Allophone	Duration	Sounds-like	Example & when to use	Vowel Type
7	/EH/	50mS	e	bEnd	Mid front
12	/IH/	50mS	i	fItting	High front
15	/AX/	50mS	u	sUcceed	Mid central
23	/AO/	70mS	o	sOng	Low back
24	/AA/	60mS	a	hOt	Low central
26	/AE/	80ms	eh	End	Low front
30	/UH/	70mS	oo	cOOk	High Back

4B.12 Long

5	/OY/	290mS	oy	tOY	Mid back
6	/AY/	170mS	ii	skY	Low central
19	/IY/	170mS	ee	sEE	High front
20	/EY/	200mS	aa/ay	grEAt	Mid front
22	/UW1/	60mS	ou	tO	High back
31	/UW2/	170mS	ouu	monosyllabic fOOd	High back
32	/AW/	250mS	ow	OUt	Low central
53	/OW/	170mS	oo/eau	snOW	Mid back

4B.13 R- Coloured

47	/XR/	250mS	aer	repAIR	Mid front
51	/ER1/	110mS	er	lettER	Mid central
52	/ER2/	210mS	err	monosyllables: blRd	Mid central
58	/OR/	240mS	or	stORE	Low back
59	/AR/	200mS	ar	fARm	Low central
60	/YR/	250mS	ear	clEAR	High front

4B.2 AFFRICATES

Address	Allophone	Duration	Sounds-like	Example & when to use
10	/JH/	100mS	j	Jury
50	/CH/	150mS	ch	CHurch

4B.3 RESONANTS

Address	Allophone	Duration	Sounds-like	Example & when to use
14	/RR1/	130mS	rr	Initial positions: Read
39	/RR2/	80mS	r	Initial clusters: bRown
49	/YY1/	90mS	y	Initial clusters: cUte
25	/YY2/	130mS	yy	Initial position: Yes
45	/LL/	80mS	l	heLLo
62	/EL/	140mS	ll	angLe
46	/WW/	140mS	w	We

4B.4 FRICATIVES

4B.41 Voiced

Address	Allophone	Duration	Sounds-like	Example & when to use
18	/DH1/	140mS	dth	Initial positions of words: THey
54	/DH2/	180mS	dth	Final positions and between vowels: baTHe
35	/VV/	130mS	v	Vest
43	/ZZ/	150mS	z	Zoo
38	/ZH/	130mS	zh	beiGE

4B.42 Voiceless

29	/TH/	130mS	th	Thin
40	/FF/	110mS	f	Fire
55	/SS/	60mS	s	Sat

(29,40,55 may be doubled for Initial positions and used singly in final positions)

27	/HH1/	90mS	h	before front vowels: Head
57	/HH2/	130mS	hh	before back vowels: Hoe
37	/SH/	120mS	sh	SHirt
48	/WH/	150mS	wh	WHim

4B.5 STOPS

4B.51 Voiced

Address	Allophone	Duration	Sounds-like	Example & when to use
28	/BB1/	40mS	b	riB
63	/BB2/	60mS	bb	Initial positions before vowels: Bug
21	/DD1/	50mS	d	Final positions: enD
33	/DD2/	80mS	dd	Initial positions and in clusters: Down, Drain
36	/GG1/	80mS	g	Before high front vowels: Guest
61	/GG2/	80mS	gg	Before high back vowels: Go; and clusters: Green
34	/GG3/	120mS	ggg	Before low vowels: Got; In medial clusters: anGer; and fina positions: peG

4B.52 Voiceless

17	/TT1/	80mS	t	Final clusters before /SS/: iTs
13	/TT2/	100mS	tt	To
42	/KK1/	120mS	k	Before front vowels: Computer
41	/KK2/	140mS	ck	Final positions: speaK Final clusters: sKy
8	/KK3/	80mS	kk	Before back vowels: Cork; and initial clusters: Crane
9	/PP/	150mS	p	Pub

4B.6 NASAL

Address	Allophone	Duration	Sounds-like	Example & when to use
16	/MM/	180mS	m	Milk
11	/NN1/	170mS	n	Before front and low vowels: Nice; Final clusters: earN
56	/NN2/	140mS	nn	Before back vowels: Noise
44	/NG/	200mS	ng	aNGer

4B.7 PAUSES

Address	Allophone	Duration	Example & when to use
0	PA1	10mS	Use before voiced stops and /JH/, or as an apostrophe (')
1	PA2	30mS	Use before voiced stops and /JH/, or as a colon (:) or semicolon (;)
2	PA3	50mS	Use before voiceless stops, and /CH/, and between words, or as spaces
3	PA4	100mS	Use between phrases or as a comma (,)
4	PA5	200mS	Use between sentences or as a period (.)

5 TEXT TO SPEECH

The commands |SAY and |ECHO incorporate a sophisticated text to speech system capable of converting most commonly occurring English text into allophones. The allophones represent the individual speech sounds that are found within words and are output to the SSA1 speech synthesiser hardware, to produce speech corresponding to written text.

The text to speech system uses the principle of Synthesis-by-Rule. This actually involves two separate tasks. The first task consists of text interpretation, the second the application of phonological rules.

Text interpretation is a non-trivial task. The spelling system of modern day English is awkward and sometimes confusing, as any school child might be aware. Words that sound the same may be spelt differently, like 'here' and 'hear', and equally words that sound differently may be spelt the same. For example 'lead' as in 'Lead us not into temptation' verses 'lead weight' meaning the metal. Similarly 'live' as in 'live wire' verses 'live a long time'. The only real solution to this problem would require a database containing the meanings of words. The context of the phrase could then be determined and the correct pronunciation deduced. Unfortunately this approach is restricted to mini-computers rather than personal computers. Text interpretation in the case of the *SSA1* is therefore restricted to the identification of word boundaries, the recognition of symbols and abbreviations, such as '@' and 'Dr', and the conversion of numbers. If one spelling of a word sounds incorrect then try its alternative, or experiment with misspelling the word.

Once the input text has been identified a set of rules are applied to determine the corresponding allophones for a given letter group within a word. The same letters may have different sounds depending where they occur within a word. Unfortunately the English language has many spelling peculiarities that are capable of deluding any logical set of rules. In fact some of the most common words contain blatant violations of the general rules. An instructive illustration of this comes from George Bernard Shaw, who used the letters 'ghoti' to spell 'fish', taking 'gh' from 'cough', 'o' from 'women', and 'ti' from 'nation'.

In order to make use of the spelling rules the English language does have, without causing havoc with those words that defy the rules, an exceptions dictionary is used that contains the correct set of allophones for troublesome words. Some of the BASIC keywords are examples of exceptions, the command CREAL, for instance, is actually pronounced 'c' followed by 'real'. For this reason the allophones for some keywords are held in the exceptions dictionary to prevent the word being said literally, and therefore sounding incorrect. The list of possible exceptions can be quite large and consequently the *SSA1* only has a limited set. Any words that still sound incorrectly may be tackled by splitting them into their allophones, as outlined in Chapter 4.

5.1 Recognised Abbreviations

OK	
BBC	
ITV	
Dr	'doctor'
Mr	'mister'
Mrs	'misses'
etc	'etcetera'
pm	'p' 'm'

5.2 Spoken Symbols

Symbol	ASCII	Text spoken
#	23H	'hash' when not followed by a digit, 'pounds' if immediately followed by a digit
$	24H	'string' when not followed by a digit, 'dollars' if immediately followed by a digit
%	25H	'percent'
&	26H	'and'
*	2AH	'times'
+	2BH	'plus'
-	2DH	'minus'
.	2EH	'point' if followed by a digit
/	2FH	'divided by'
<	3CH	'less than'
=	3DH	'equals'
>	3EH	'greater than'
@	4flH	'at'
\	5CH	'divided by'
↑	5EH	'power'
|	7CH	'bar'

The ASCII characters 23H ('#' pound or hash), and 24H ('$' dollar or string), are pronounced differently depending upon their context. If the characters are followed by a digit then the character is not spoken until the following number has been said. The character then takes on a monetary role and is pronounced 'dollar' or 'pound'.

Example:

```
#45       will say 'forty five pounds'
45#       will say 'forty five hash'
# 45      will say 'hash forty five'
$45       will say 'forty five dollars'
45$       will say 'forty five string'
$ 45      will say 'string forty five'
```

The ASCII character 2DH is always spoken as 'minus'. If this character is occurs as a hyphen between two words it will also be spoken as 'minus'.

Letters, symbols and numbers may be combined to form mathematical equations and so on:

```
3+(1/2)=3.5     spoken as 'three plus one divided by two equals three point five'.
a=34+b          spoken as 'a equals thirty four plus b'
BBC2            spoken as 'b b c two'
```

5.3 Numbers

The text interpreter will say numbers less than ten digits in full, numbers of more than nine digits will have each digit spoken separately. Any leading zeros will be suppressed, except for zeros encountered after a decimal point. Numbers longer than 40 digits will be split-up into sections of 40 digits and each section will be treated separately.

```
123456789       spoken as 'one hundred and twenty three million four hundred and fifty six thousand seven hundred and eighty nine'.
00000001        spoken as 'one'
000002.002      spoken as 'two point zero zero two'
1111111111      spoken as 'one one one one one one one one one one'
```

5.4 Letters

The ASCII characters A-Z (41H to 5AH) and a-z (61H to 7AH) inclusive, are interpreted by a set of phonological rules. Words greater than 40 characters will be split-up into sections of 40 characters, and said separately.

5.5 Text Delimiters

All the punctuation symbols, tabs, spaces, carriage returns and line feeds are used to determine word boundaries. More than one occurrence of a space, tab or newline character is ignored, so that pauses do not seem too long. For example the following two lines will both have the same pause between the words.

```
Hello there!
Hello                                   there!
```

All other characters, including graphics, control codes (except for those used to determine word boundaries) and unpronounceable symbols, such as brackets, are ignored.

6 EXAMPLE PROGRAM

6.1 The Speaking Clock

This example program shows how simply text to speech may be incorporated within a program. The |ECHO command is used to read aloud the input prompts, whilst |SAY is used to speak the time. The time is spoken every five seconds, using a twelve hour cycle.

Before using the program, first make sure that the *SSA1* software is installed.

```
10 REM ****** The Speaking Clock *******
20 REM
30 REM ****** Assumes the SSA1 code has been loaded
40 REM
50 REM ****** Twelve hour cycle. Time spoken every
60 REM         five seconds
70 REM
75 ON ERROR GOTO 500
80 O$="o clock": P$="precisely": SEC$="second": SECS$=SEC$+"s"
90 A$="and"
100 MODE 1: |ECHO,1
110 HOURS=12
120 PRINT "`12 Hour Speaking Clock`": PRINT
130 INPUT "`Enter the time in HOURS,MINS`";HH,MM
140 IF HH>12 OR MM>60 THEN 130
150 S=0
160 MODE 0
170 PEN 3: PRINT "`The Speaking Clock`": PEN 1
180 |ECHO,0: GOSUB 270
190 EVERY 250 GOSUB 210
200 WHILE -1: WEND
210 REM ***** Update the time ************
220 SS=(SS+5) MOD 60230 IF SS<>0 THEN 270
240 MM=(MM+1) MOD 60
250 IF MM<>0 THEN 270
260 HH=(HH+1) MOD HOURS
270 LOCATE 7,11
280 IF HH=0 THEN HH=HOURS
290 PRINT USING "##:##:##";HH;MM;SS
300 H$=STR$(HH): |SAY,@H$
310 IF MM=0 THEN |SAY,@O$: GOTO 330
320 M$=STR$(MM): |SAY,@M$
330 IF SS=0 THEN |SAY,@P$: RETURN
340 |SAY,@A$
350 S$=STR$(SS): |SAY,@S$
360 IF SS>1 THEN |SAY,@SECS$ ELSE |SAY,@SEC$
370 RETURN
500 PRINT: PRINT "The SSA1 software has not been loaded"
510 PRINT: END
```

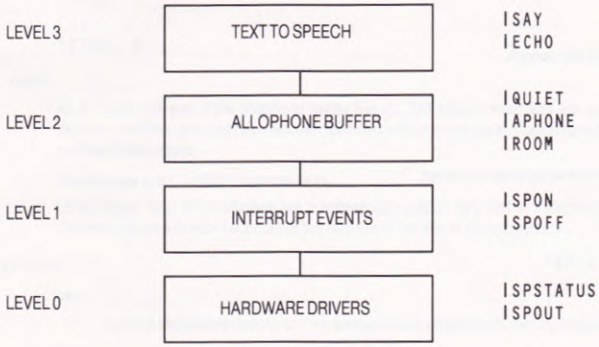
7 ADVANCED COMMAND OVERVIEW

The *SSA1* software provides nine external commands that are usable from BASIC programs. The commands are in the form of Resident System Extensions (RSXs) that must be loaded into memory, before they can be used. The RSXs are fully relocatable, and perform their own initialisation. A simple `LOAD` and a `CALL` from BASIC is all that is required to introduce the commands and initialise the *SSA1*.

In order to utilise the *SSA1* software to the full it is necessary to know how the commands work and what they do. The following sections detail the commands.

7.1 Hierarchy of Commands

There are four levels of hierarchy, these include low level firmware drivers that address the speech processor directly, interrupt driven allophone buffers, and text to speech conversion. The hierarchical diagram below summarises the levels. Each level is intended for a particular application. It is not recommended that the commands from levels 3, 2 and 0 be used simultaneously.



7.11 LEVEL 3 - Text to Speech Conversion

This is the highest level where ASCII text is converted into allophones by using a sophisticated Text to Speech system. The allophones produced are subsequently sent to level 2, the interrupt driven allophone buffer. When the the buffer becomes full, the commands at level 3 suspend BASIC until all the text has been converted.

7.12 LEVEL 2 - Interrupt Driven Allophone Buffer

This buffer is used to temporarily queue up to 64 allophones that are to be output to the speech processor. Its main purpose is to allow a foreground program to continue operation, whilst interrupt events ensure that any data, from the buffer, is output to the speech processor at the appropriate time.

The buffer can be filled from level 3 or by the command `|APHONE` at level 2, either source may insert allophones at any time into the buffer. If the buffer becomes full then the `|APHONE` command will discard any allophones that cannot be stored.

The command `|ROOM` may be used whenever it is necessary to determine the number of spaces left in the buffer. Some status information relating to levels 1 and 0 are also returned by `|ROOM`.

Any data in the buffer may be flushed, and a pause sent to the speech processor with the command `|QUIET`.

The buffer can only be emptied by speech interrupt events occurring at level 1. If the speech interrupts are off then the buffer will fill up. To allow text to speech to operate correctly the `|ECHO` and `|SAY` commands automatically enable the speech interrupts. The `|APHONE` command does not change the state of the interrupts, so care must be taken not to lose allophones if `|APHONE` is used when the buffer is full.

7.13 LEVEL 1 - Speech Interrupt Events

An event may be set up using `|SPON`. This event reads characters from the allophone buffer and sends them to the speech processor. The event may be disabled at any time with `|SPOFF`, this prevents any more characters being read from the buffer and also sends a pause to the speech processor, to ensure that the last allophone is not sounded for ever. When the event is disabled the buffer is left in its current state and may fill up from levels 3 or 2.

7.14 LEVEL 0 - Firmware Drivers

The allophones from level 2 are passed to this level via the interrupt events at level 1, to be output to the speech processor when it is ready to accept another allophone.

Two commands |SPSTATUS and |SPOUT, allow the user to interrogate the status of, and to send allophones directly to, the speech processor. It is recommended that speech interrupts are off, or that the buffer is empty, before using |SPOUT. Otherwise allophones from |SPOUT may be randomly interspersed with allophones being output by the speech event.

Level 0 is provided primarily for special effects and synchronisation. For example it is only possible to ascertain if a particular allophone has been spoken by using |SPSTATUS or |SPOUT.

8 THE COMMANDS IN DETAIL

8.1 |SAY

Use

To convert a string of ASCII text into speech.

Syntax

|SAY,@‹String Variable›

Where:

‹String Variable› represents the string to be converted.

example

Say 'the time is four thirty'.

```
a$ = "The time is 4:30"
|SAY, @a$
```

notes

The speech interrupts are automatically enabled. Any data in the allophone buffer will therefore be output before the allophones produced by the text to speech system. If the length of the string causes the allophone buffer to fill, then the operation of BASIC will be suspended until the entire string has beenconverted. No distinction is made between upper and lower case. Punctuation is converted into pauses of equivalent duration and a short pause is always sent at the end of each word.

Some common abbreviations, such as 'Mr' and 'Dr', are converted into their full word equivalents: 'mister' and 'doctor'. Numbers are spoken as full as possible such that '10' will be pronounced 'ten' instead of 'one', 'zero'. Chapter 5 gives a detailed description of the features of the text to speech system.
The following example will allow you to continually enter the text to be spoken:

```
10 CLS
20 LINE INPUT "What shall I say? ";a$
30 |SAY,@a$
40 PRINT
50 GOTO 20
```

8.2 |ECHO

use

Activates text to speech conversion on text printed to the screen. Five modes of operation are possible that result in printed characters being echoed to the speech synthesiser.

syntax

|ECHO, ‹Mode›

Where:

‹Mode› is an integer in the range 0..4 specifying the type of echo to be applied. The default mode after installation is mode 0. The modes available are:

Mode 0

This disables modes 1..4 and restores the original print condition as found during installation.

Mode 1

All print output enclosed between the delimiters (`) is spoken AND displayed.

Mode 2

All text output to the screen, including listings, syntax errors and so on is spoken, control codes are still sent to the screen.

Mode 3

All text output to the screen, including listings, syntax errors and so on is spoken AND displayed.

Mode 4

All print output enclosed between the delimiters (.) is spoken but NOT displayed.

example

Say a program instead of listing it.

```
|ECHO, 2
LIST
|ECHO, 0
```

notes

All 4 modes will wait if the allophone buffer fills up. The speech interrupts are automatically enabled. If programs are listed in modes 1 or 4 then any text between the delimiters will be acted upon. Care should be taken to ensure that there is always an even number of delimiters.

The delimiter is a (`) ASCII character 60H

Lines longer than 80 characters are automatically output. Any text is interpreted in the same way as the |SAY command, Chapter 5 gives a detailed account of the features of the text to speech system.

8.3 |APHONE

use

To send allophones directly to the interrupt driven allophone buffer, bypassing text to speech conversion.

syntax

|APHONE, ‹allophone› [,‹allophone›]

Where:

‹*allophone*› is an integer in the range 0..63. Values greater than 63 will be taken MOD 64. Consecutive allophones may be sent by separating each with commas. The maximum number of allophones that may be specified at any given time is limited by the length of a BASIC command line.

If the buffer becomes full, then the command will terminate. Any allophones encountered after the buffer becomes full will be discarded. A check should be made, using |ROOM to ensure there is enough space left in the buffer to accept the allophones, before using this command.

The allophones will only be heard if the speech interrupts are on.

example

Say the word 'hello' without using the text to speech system.

```
|APHONE,27,7,62,53,0
```

notes

Chapter 4 summarises the allophones available on the SSA1 and also presents some guidelines on how to select the allophones for particular words.

8.4 |ROOM

use

The |ROOM command may be used to find out how many free locations there are in the allophone buffer, whether the speech synthesiser is currently sounding an allophone, and if the speech interrupts are active.

syntax

|ROOM,@‹Integer Variable›

Where:

‹Integer Variable› is the variable into which the result will be passed.
The result is bit significant:

Bits 0..5 the number of free locations in the buffer
Bit 6 indicates the state of the speech interrupts, when set the interrupts are disabled, the buffer will fill up if any more allophones are sent to it.
Bit 7 when set, the speech synthesiser is busy sounding an allophone.

example

Say the word 'hello' only if there are enough spaces in the allophone buffer.

```
10 a%=0
20 |ROOM,@a%
30 IF (a% and &3f) > 5 THEN |APHONE,27,7,62,53,0
40 END
```

notes

If the space returned is zero, then the buffer is full and any more allophones sent using **|APHONE** will be discarded.

8.5 |QUIET

use

To clear the allophone buffer to its empty condition and to send a pause to the speech processor.

syntax

```
|QUIET
```

No parameters should be given.

example

```
10 ON BREAK GOSUB 100
 ..
100 |QUIET : REM Stop generation of speech
110 RETURN
```

notes

This command may be issued at any time, regardless of the state of the speech interrupts, in order to prevent the generation of any speech, due to allophones in the speech buffer waiting to be spoken.

8.6 |SPON

use

To activate the speech interrupt event. The event allows allophones queued in the allophone buffer to be output independently of BASIC.

syntax

```
|SPON
```

No parameters should be given.

notes

No allophones from the buffer will be sent to the speech processor until speech events have been activated with this command. The machine will be slowed down slightly due to the overhead involved in dealing with the event. The events are synchronised with the fast ticker.

8.7 |SPOFF

use

To prevent the output of data from the allophone buffer. This has the reverse effect of **|SPON**, and disables the speech interrupt events.

syntax

```
|SPOFF
```

No parameters should be given.

example

```
10 |SPOFF
20 |APHONE,27,7,62,53,0
30 REM  Main program follows
 .
 .
100 |SPON : REM Say the previously stored word
```

notes

This command will have no effect unless used after a |SPON command. Any allophones remaining in the allophone buffer will remain intact. The output of the data from the allophone buffer may be re-commenced by issuing a |SPON command. After disabling events the |SPOFF command will wait for the currently voicing allophone to finish, before sending a pause, preventing the last allophone from sounding for ever.

It is recommended that |SPOFF is used whenever speech is not required for long durations.

8.8 |SPSTATUS

use

To read the hardware status of the speech processor.

syntax

|SPSTATUS,@‹Integer Variable›

Where:

‹Integer Variable› is a 16 bit integer into which the status will be placed. The status is bit significant:

Bit	Interpretation
0..5	Always returns 0, no meaning attached
6	This is a copy of /LOAD
7	This is a copy of /BUSY
8..15	Always returns 0, no meaning attached

A combination of bits 6 and 7 may be used to ascertain the condition of the speech processor as follows:

bit 7 6	speech condition
0 0	Voicing allophone,
0 1	Voicing allophone, DO NOT load next allophone address
1 0	Not voicing (silent), next allophone address may be loaded
1 1	Not produced by the SSA1

example

Wait for allophones to finish, before sending another.

```
10 a%=0
20 FOR i = 1 TO 5
30  |SPSTATUS,@a%
40  WHILE a% AND 64: |SPSTATUS,@a%: WEND
50  READ b%
60  |SPOUT,b% : REM No handshaking for |SPOUT
70 NEXT i
80 END
90 DATA 27,7,62,53,0
```

notes

This command may be used in conjunction with |SPOUT to test for the presence of the SSA1 hardware. A BASIC program may use |SPSTATUS to achieve synchronisation with spoken allophones, as may be required during animation.

8.9 ISPOUT

use

To send allophones directly to the speech processor, bypassing the allophone buffer. One of three handshaking methods may be selected, according to the application.

syntax

```
ISPOUT, ‹allophone› [,‹allophone›]
```

Where:

‹allophone› is a 16 bit integer value in the range 0..255. Only the lower 6 bits are output as address data to the speech processor. The value is interpreted as follows:

Bit	Interpretation
0..5	Allophone in the range 0..63
6	When set, selects /BUSY mode of handshaking
7	When set, selects /LOAD mode of handshaking
8..15	Should be 0

Additionally bits 6 and 7 select one of 3 handshake modes in which to output the allophone address.

bit 6 7	Mode
0 0	Send allophone address immediately, regardless of /BUSY or /LOAD.
0 1	Send allophone address when /LOAD active, ignore /BUSY
1 0	Send allophone address when /BUSY active, ignore /LOAD
1 1	Send allophone address when /BUSY active, ignore /LOAD

example

Say 'hello' using /BUSY handshaking.

```
10 a%=0
20 FOR I=1 TO 5
30  READ b%: b%=b%+128
40  ISPOUT, b%
50 NEXT I
60 END
70 DATA 27,7,62,53,0
```

notes

Speech interrupts should be disabled with ISPOFF before using ISPOUT, otherwise it is possible that allophones may be randomly interspersed with those being output under interrupt from the allophone buffer. Although this may be an undesirable feature, it can be used to advantage to provide sound effects, in some applications.

9 ERROR MESSAGES

One general error message may be given by any of the commands, it is:

```
Bad command
```

The error may be due to either the incorrect number of parameters, an echo mode that is not in the range 0..4, or an attempt to say a null string.

APPENDIX A

A.1 Using the SSA1 Software from Machine Code

The commands available with the *SSA1* provide all the necessary facilities for any envisaged application, the commands however are not restricted for use with BASIC. As the commands are RSXs they can be called from within machine code programs, provided the correct parameters are passed, exactly the same as BASIC would pass them. By invoking the RSXs in this way, the programmer does not need to know the load address, or the hardware configuration of the *SSA1*, in order to produce speech.

From this point on the reader is assumed to have available a CPC464 Firmware Manual. To find out about the interface to an RSX, and how the parameters are passed read chapter 9.6 of the Firmware Manual.

The desired command is called from a program in a two stage process. Firstly KL FIND COMMAND is used to obtain the address of the command. (This firmware call may be used to indicate if the SSA1 software has in fact been installed.) Secondly once the address of the command has been established then the relevant registers should be set up to correspond to the parameters required by the command, and KL FAR PCHL called to invoke the command. It should be assumed that all the primary registers, including IY, are corrupted by the RSX commands.
The following example program shows how to use the ‡SPOFF command from machine code, it should be read in conjunction with the firmware manual.

```
EXAMPLE:    LD  HL, COMMAND$        ; READ ADDRESS OF COMMAND STRING
            CALL KL.FIND.COMMAND    ; SEARCH FOR THE RSX
            JR  NC, NOTFOUND        ; NO SUCH COMMAND LOGGED IN
;
            XOR A                   ; ZERO PARAMETERS REQUIRED
            CALL KL.FAR.PCHL        ; INVOKE THE COMMAND
            RET                     ; SUCCESSFUL
;
COMMAND$:   DEFW "SPOF"
            DEFB "F"+128            ; COMMAND NAME, LAST CHAR HAS BIT 7 SET
;
NOTFOUND:                           ; FAILED TO FIND COMMAND
```

The same procedure can be used for all the SSA1 commands. Remember that interrupts should not be disabled if the allophone buffer is to be used.

APPENDIX B

B.1 Driving the SSA1 Hardware Directly

In some cases it may be necessary to use the SSA1 without loading the RSX code, for example in a dedicated game. This section describes how the SSA1 looks to the programmer and the procedures needed to use it directly.

B.11 Memory Map

The *SSA1* occupies one address in the I/O map, as follows:

ADDRESS	INPUT	OUTPUT
0FBEEH	STATUS	ALLOPHONE DATA

Access to the speech processor is achieved during an I/O operation when A10, A4 and A0 are low. That is, I/O address 0FBEEH.

The SSA1 generates no hardware interrupts, and must be used in a polled fashion.

B.12 Status Register

During a read, the status of the speech processor is returned as a bit significant 8 bit value. The bit positions are assigned as follows:

BIT		INTERPRETATION
0-5	Not used.	These bits are undefined
6	/LOAD	Allophone addresses may be sent when LOW
7	/BUSY	When LOW the processor is talking

B.13 Allophone Address Register

There are a total of 64 addresses, each corresponding to an allophone. Only bits 0-5 make up the address. Bits 6 and 7 should ALWAYS be sent as a LOW.

All the allophones in Table 4B are directly supported.

B.14 Handshaking

There are two possible methods of sending allophone addresses to the speech processor, both make use of polling the STATUS register.

1] Allophones may be written consecutively one after another whenever /LOAD is LOW. The speech processor will allow another allophone to be loaded whilst it is still sounding a previous allophone, its readiness to accept the next allophone is indicated by the /LOAD signal.

2] New data may be written after a preceding allophone has been completely spoken. This is achieved by writing data only when /BUSY is HIGH. In this case there is only ever one allophone outstanding, waiting to be spoken. The speech may be terminated at any time between successive allophones, by sending a pause to the processor. A suitable procedure for loading data using /BUSY handshaking is:

```
BEGIN
  READ STATUS
    WHILE /BUSY=Ø
      READ STATUS
    WEND
  OUTPUT ALLOPHONE
END
```

This method of handshaking is recommended.

The last part of the last allophone spoken will be continuously repeated, until another allophone is addressed. A PAUSE should be sent as the last allophone of any group, to stop the speech processor generating noise.

APPENDIX C

C.1 Hardware Operation

The operation of the *SSA1* can be functionally subdivided into two sections; one digital; the other analogue. Figure C1 outlines their association.

C.11 Digital

The digital section comprises four integrated circuits, one is the SP0256-AL2 allophone speech processor from GI. The remaining three complete the interface of the speech processor to the CPC464 expansion bus.

Whenever an i/o read is performed with A10, A4 and A0 low, two tri-state buffers connected to the data lines D7 and D6 are enabled. The inputs to these buffers are connected to the SBY and /ALD handshake lines of the speech processor. The SBY signal is referred to as /BUSY and /ALD is referred to as /LOAD throughout this guide.

Allophone addresses are loaded whenever an i/o write is performed with A10, A4 and A0 low.

C.12 Analogue

Two separate audio amplifiers are used to provide a stereo output of up to 200mW per channel into 4 ohms.

The stereo output from the CPC464 is mixed with the speech before being amplified. A SOUND command with a volume level of 15 produces an output of 200mW into a 4 ohm load at full volume.

The digital output from the speech processor, a 40kHz pulse width modulated signal, is low pass filtered at 5kHz and then fed symmetrically to each channel. At full volume the speech will generate 200mW into 4 ohms.

The operation of the computer may be heard through the loudspeakers on occasions, this is due stray pickup of computer generated noise manifesting itself in the same way as that heard on the computer's internal loudspeaker.

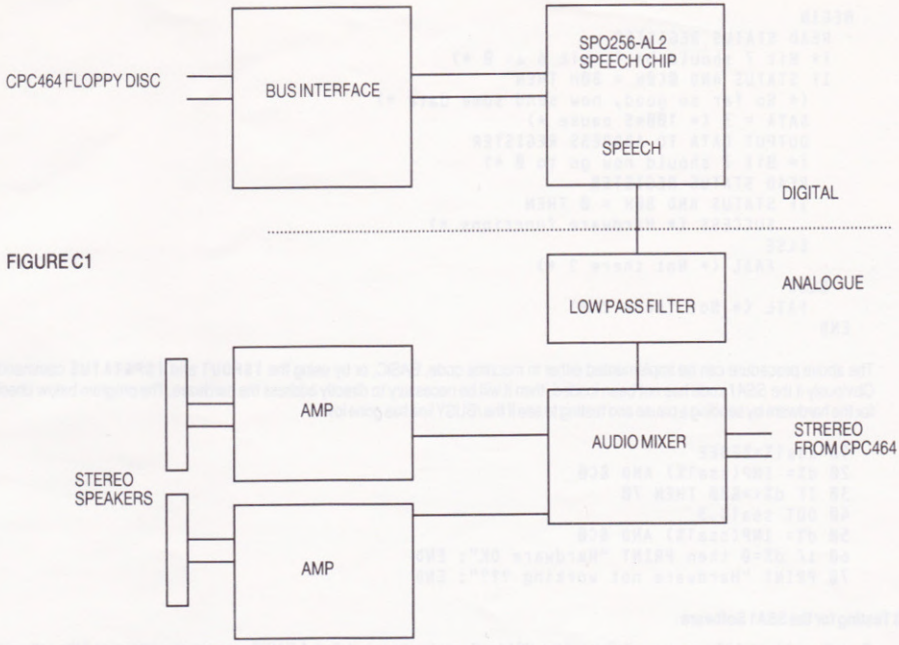


FIGURE C1

APPENDIX D

D.1 Initialising the SSA1 Software

It is possible to load the SSA1 software from within another program, in fact this is what the loader program does. To load the software you must first reserve an area of memory into which the code is to be loaded. The loader program will tell you how much space will be required. Let the load address be BASE, the following lines of BASIC will load, relocate and initialise the SSA1 software:

```
10 LOAD "SSA1.BIN",BASE
20 START=BASE+PEEK(BASE)+PEEK(BASE+1)*256
30 CALL START
```

The first word of the program is the offset from BASE to the start of the relocation routine. This is added to BASE to obtain START, which is then called to relocate and log on the software.

D.2 Testing for the SSA1 Hardware

If you are driving the hardware without the SSA1 software, or the software has been loaded without the using the loader program, then it is desirable to know if the SSA1 hardware is in fact connected. The following procedure outlines how to test for the hardware.

```
BEGIN
  READ STATUS REGISTER
  (* Bit 7 should be 1, bit 6 at Ø *)
  IF STATUS AND ØC0H = 80H THEN
    (* So far so good, now send some data *)
    DATA = 3 (* 100mS pause *)
    OUTPUT DATA TO ADDRESS REGISTER
    (* Bit 7 should now go to Ø *)
    READ STATUS REGISTER
    IF STATUS AND 80H = Ø THEN
       SUCCESS (* Hardware functions *)
    ELSE
       FAIL (* Not there ? *)
  ELSE
    FAIL (* Not there ? *)
END
```

The above procedure can be implemented either in machine code, BASIC, or by using the |SPOUT and |SPSTATUS commands. Obviously if the *SSA1* code has not been loaded, then it will be necessary to directly address the hardware. The program below checks for the hardware by sending a pause and testing to see if the /BUSY line has gone low.

```
10 ssa1%=&FBEE
20 d%= INP(ssa1%) AND &C0
30 IF d%<>&80 THEN 70
40 OUT ssa1%,3
50 d%= INP(ssa1%) AND &C0
60 IF d%=0 then PRINT "Hardware OK": END
70 PRINT "Hardware not working ???": END
```

D.3 Testing for the SSA1 Software

Sometimes it is useful for a program to know if the *SSA1* software has been installed. A BASIC program can determine if the software is present by issuing any one of the commands, and trapping any error messages. An error 28, 'Unknown command' will be generated if the software has not been installed. The program below shows how this is done.

```
10 ON ERROR GOTO 50
20 |QUIET
30 PRINT "Software installed"
40 END
50 IF ERR=28 THEN PRINT "No SSA1 software installed"
60 END
```

D.4 Further examples

Another example program lets you write up to 9 different phrases which can then either be spoken, or overwritten:

```
10 CLS
20 PRINT:PRINT "Press W to Write, press S to Say"
30 IF INKEY(59)<>-1 THEN 50
40 IF INKEY(60)<>-1 THEN 80 ELSE 30
50 PRINT "WRITE which";:GOSUB 110
60 PRINT "Now enter phrase number";n:LINE INPUT p$(n)
70 GOTO 20
80 PRINT "SAY which";:GOSUB 110
90 IF p$(n)="" THEN PRINT "Nothing to say!" ELSE |SAY,@p$(n)
100 GOTO 20
110 PRINT " phrase number (1 to 9)?";
120 a$=INKEY$:IF a$="" THEN 120
130 n=VAL(a$):IF n<1 THEN 120 ELSE PRINT n
140 RETURN
```

AMSTRAD SSA-1
Speech Synthesiser System for the CPC464

First edition 1985
Written by Chris Honey
Typeset by Amsoft Computer Graphics


Synthetiseur Vocal
Avec Amplificateur
Stereo Et H
POUR L'
SSA
Amsoft ©
SYNTHETISISEUR DE PAROLE
FACE 1
SSA-1
Appuyez sur (CTRL) et petite touche (ENTER),
pressez PLAY sur le Datacorder puis une touche quelconque.
© AMSOFT ET HONEYSOFT 1985.


Amsoft ©
SYNTHETISISEUR DE PAROLE
FACE 2
SSA-1
Appuyez sur (CTRL) et petite touche (ENTER),
pressez PLAY sur le Datacorder puis une touche quelconque.
© AMSOFT ET HONEYSOFT 1985.

Amsoft ©

SPEECH SYNTHESISER

SIDE
1

SSA-1

To LOAD press CTRL and small ENTER keys simultaneously
press PLAY on Datacorder then any key

Amsoft ©

SPEECH SYNTHESISER

SIDE
2

SSA-1

To LOAD press CTRL and small ENTER keys simultaneously.
press PLAY on Datacorder then any key