University of Canterbury
1985

Audio Response Systems

Research project for
Batchelor of Science with Honours
in Computer Science

by A. J. E. Dale

Abstract

This project details the implementation of an electronic voice synthesiser that can be programmed with a custom vocabulary and a custom voice. The technology used for the voice synthesis is Linear Predictive Coding, and the hardware is implemented using a Z80 microprocessor driving a TMS5220 LPC voice synthesiser chip.

Acknowledgements

This project was sponsered by Business Computers Limited, who provided funds for the purchase of the hardware components of the voice synthesiser. Thanks are due to John Kelly of BCL for obtaining this sponsorship for me.

I am also indebted to Nigel Brieseman and Steven Turner, for their part in my project. The digitised speech I use for the TMS5220 chip is provided by SGPRC, a general purpose speech-analysis program written by Nigel for his Ph.D thesis. The design for the hardware of my project is copied from Steven Turner's 3rd pro. engineering project.

Bill Kennedy and Dick Cooper, my supervisor, also assisted with my project.

Contents

1. Aims and Objectives

Data preperation operators do not, as a rule, spend much time looking at the screen of their computer terminal. Much of their time is spent entering data from printed forms etc, so that it is possible for them to type ahead of a mistake for quite a while before looking up at the screen and realising their error.

The aim of this project is to provide audible feedback to the operator, for error and other system messages, of a more informative nature than a beep from the terminal. In a word, Speech.

The speech generator should take the form of an "audible terminal" that is controlled via an asynchronous communications line:

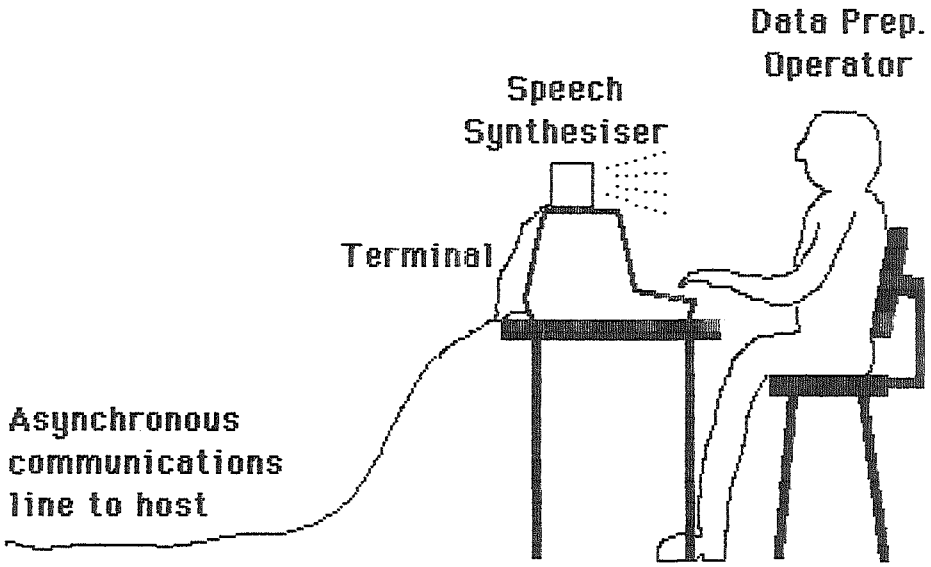


fig. 1: Proposed audio feedback system

The specialised nature of the application for the speech generator requires a custom vocabulary. For instance, if accounting data was being entered into a system from several companies it might be desirable to refer to a particular company by name.

Another desirable feature of the speech generator for New Zealand users would be for it to have a New Zealand accent, rather than the North American twang that commercial speech synthesisers invariably have. This feature could be extended, by giving the synthesiser the voice of, for instance, the chief operator at the site at which it is installed.

The design goals of the project can thus be summarised:

1. To produce an "audible terminal" that can be controlled from a host computer via an asynchronous communications line.

2. To program the audible terminal with a custom vocabulary.

3. To give the audible terminal a custom voice (accent).

2. Available Technologies

There are several well developed speech synthesis technologies available today. They all represent compromises between the bit rate of speech data they need, the intelligibility of the speech produced and its "naturalness".

2.1 Digital Recording

This is a very "brute force" approach to speech synthesis. The electrical signal produced from recorded speech is directly digitised and stored. Synthesised speech is produced by feeding the digitised speech to a digital to analogue converter to reproduce the original signal. An example of digital recording is the compact disc music reproduction system (digital recording can reproduce **any** sound).

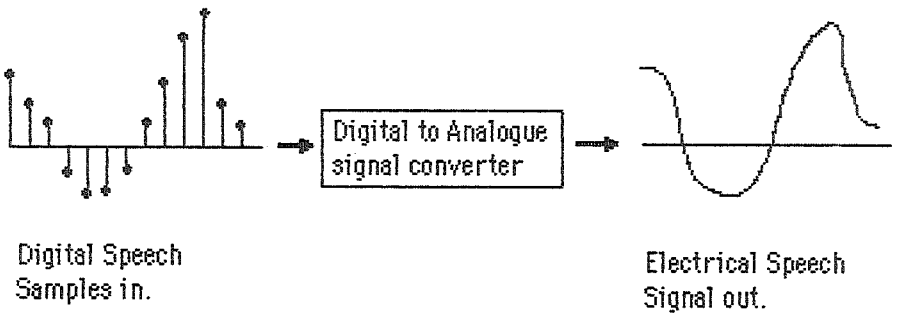


fig 2: Digital Recording speech synthesiser

Digital recording can produce very high quality speech (the compact disc music recording system has better performance than analogue systems such as records and cassettes), and can produce a custom voice and vocabulary. Text-to-speech is not really feasible with digital recording, since you must store every word you wish to synthesise in its entirety. If digital recording was used for the speech synthesiser each message would have to be stored in its entirety.

The main disadvantage of the Digital Recording method is the very high bit rate required to reproduce speech. Human speech uses the frequencies from 300 to 3000 Hertz, and to reproduce the highest frequencies requires data at the rate of at least 6000 bytes (48 K bits) per second (assuming each speech sample is to 8 bit precision). Digital recording speech synthesis systems typically require data rates from 16 K to 64 K bits per second. Five seconds of speech would require approximately 32 K bytes of memory, assuming a bit rate of 50 K bits per second.

2.2 Phoneme Synthesis

Phoneme synthesis builds words from their basic building blocks, or "phonemes" (see reference 1, chapter 11).

Phoneme speech synthesis systems take as their input data binary codes corresponding to phonemes, and produce synthesised speech as their output. The nature of the phoneme synthesis system means that a text-to-speech system is easily realisable, giving an infinitely variable vocabulary. (see reference 1, chapter 6).

An example of a phoneme speech synthesis system is the Votrax SC-01 speech synthesiser chip. The Votrax system uses an "alphabet" of sounds to produce its speech. For example, there are 6 "E" sounds in the Votrax aplhabet:

EH3, as in jackEt
EH2, as in Enlist
EH1, as in hEAvy
E, as in mEEt
EH, as in gEt
E1, as in bEfore

A word such as "**CAT**" is decomposed into its individual phonemes; **K AE1 AE1 T**. The binary codes for these phonemes are sent to the synthesiser to produce the spoken word.

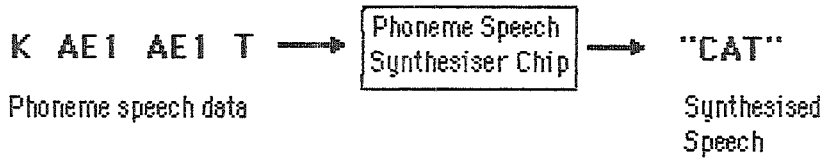


fig3: Phoneme speech synthesis system

Because an individual phoneme takes a good deal of time to speak (20 - 300 mS), the bit rate required to produce phoneme synthesised speech is very low, in the range of 30 to 100 bits per second. The storage required for a message of 5 seconds in length is only 32 bytes, assuming a data rate of 50 bits per second.

The main drawback of phoneme synthesis is that the quality and intelligibility of the speech produced is very poor, because of the variability of human speech. The pitch, length and stressing of a word depends very much on the context in which it is spoken, but the phonemes stored in the synthesiser must be general purpose and so do not merge very well when forming words. Because of this, speech produced from a phoneme synthesis system sounds very flat and robot-like.

2.3 Linear Predictive Coding (LPC)

LPC is a speech synthesis system that uses an electrical model of the human vocal tract (see reference 1, chapters 5 and 10). The parameters that control the electrical model are continuously varied to give an electrical analogue of human speech. An example of a LPC speech synthesiser is the the Texas Instruments TMS5220 integrated circuit.

The muscles that control the human vocal tract change their position at a maximum rate of about once every 25 mS. Because of this, a new set of electrical parameters is supplied to the LPC model once every 25 mS. Each set of parameters supplied to the LPC model requires from 2 to 15 bytes on average, depending on the actual implementation of the LPC model. An LPC synthesiser therefore requires a bit rate of from 800 to 6000 bits per second. Most modern LPC synthesiser chips, such as the Texas Instruments TMS5220, require about 1200 bits per second. A message 5 seconds in length would thus require 750 bytes of storage.

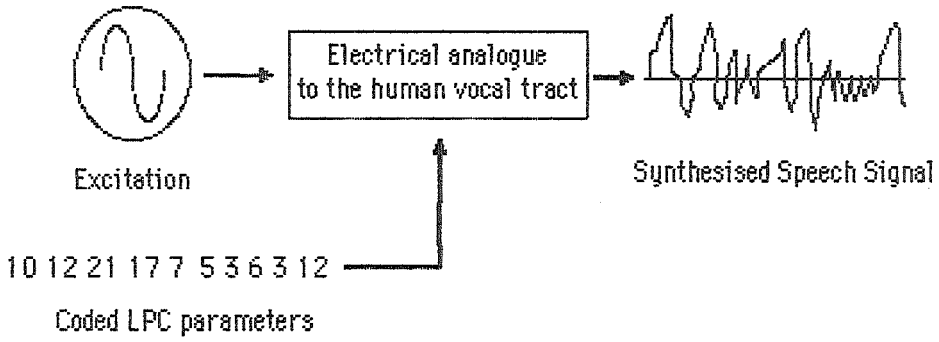


fig 4: LPC speech synthesis system

LPC speech synthesis can supply both a custom voice and vocabulary, although manfacturers supply ROM sets to program their LPC chips with a standard vocabulary. Transforming human speech into sets of LPC parameters is a complicated task, requiring at least minicomputer facilities.

2.4 Proprietary Systems

These systems use a technology developed by the manufacturer, with a semi-custom vocabulary of words available. Usually such a system consists of the speech synthesiser chip with the vocabulary available either in ROM, or on disc ready to be programmed into EPROM.

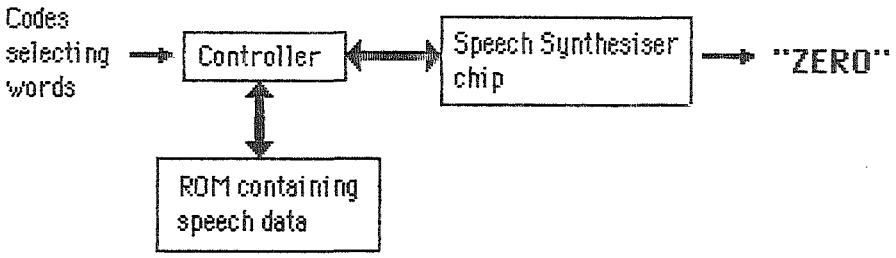


fig 5: Typical proprietary speech synthesiser system

An example of a proprietary system is National Semiconducters Digitalker II system. This system consists of a speech synthesiser chip, along with standard vocabularies in ROM and semi-custom vocabularies on disc (approximately 1000 words available, cost $US150). The quality of the speech produced by the Digitalker system is very high, albeit with an American accent.

Text-to-speech would be possible with the Digitalker system, provided the vocabulary of the text was a subset of the vocabulary available from the manufacturer. The bit rate of the Digitalker system is very low, in the order of 30 bits per second, since whole words are selected and spoken at once. An utterance of 5 seconds duration would thus require 20 bytes of storage.

3. The approach chosen

3.1 Description

3.1.1 Hardware

Linear Predictive Coding (LPC) was used to implement the speech synthesiser, the particular chip used being the Texas Instruments TMS5220.

Some way of controlling the TMS5220 chip was necessary, along with a method of communicating with the host computer for the audible terminal, via an asynchronous communications link. The best way to implement the controller is a microprocessor chip. LPC data for the TMS5220 chip could then be stored in EPROM and sent to the TMS5220 by the microprocessor chip.

3.1.2 Software

The audible terminal must be controlled from the host computer via an asynchronous communications link which could be either dedicated to the audible terminal, or tapped off the line to the operators visual terminal.

On the receipt of a trigger character(s), the audible terminal would speak a message. This message would be selected from a collection of messages stored in the audible terminal, by means of one or more index characters following the trigger sequence.

It was mentioned above that the asynchronous communications line for the audible terminal could be tapped off the line to the operators visual terminal was mentioned. In this case the operators terminal would be have to be disabled (by its "disable terminal" control character(s)) before the audible terminal could be activated. Both of these tasks (disabling the visual terminal and activating the audio terminal) could be performed at the same time by making the audible terminals trigger character(s) the same as the visual terminals disable character(s).

3.2 Justification

3.2.1 Hardware

LPC technology was used to implement the speech synthesiser for the following reasons:

(a) LPC is by far the most popular technology used today for speech analysis and synthesis. There are over 15 manufacturers offering speech synthesis chips based on the LPC system.

(b) It provides a reasonable compromise between bit rate and

intelligibility. Hand "tuned" LPC speech, such as is supplied in ROM by some manufacturers for standard vocabularies, can sound very human. The typical bit rate of 1200 bits per second means that about 7.5 minutes of speech could be stored in 64 K bytes of EPROM, which would cost about $100.

(c) The principles of LPC speech analysis are well known and documented. Producing LPC speech data from human speech is possible using minicomputer facilities.

(d) The most compelling reason for using LPC technology is that there is a large knowledge base of LPC technology here at the University of Canterbury. The School of Engineering have a group of Lecturers and Students who are implementing a system for correcting impaired speech (see references 2 and 3). The basic scheme of this system is to:

1. Read the impaired speech into the system, and transform it into LPC parameters.

2. Transform the LPC parameters to correct the impaired speech, using some predermined transformation.

3. Synthesise the corrected LPC parameters back into normal speech.

The eventual aim of the project is to perform the above three steps in real time, using a TMS320 microprocessor to perform steps 1 and 2, and a TMS5220 speech synthesiser chip to perform step 3.

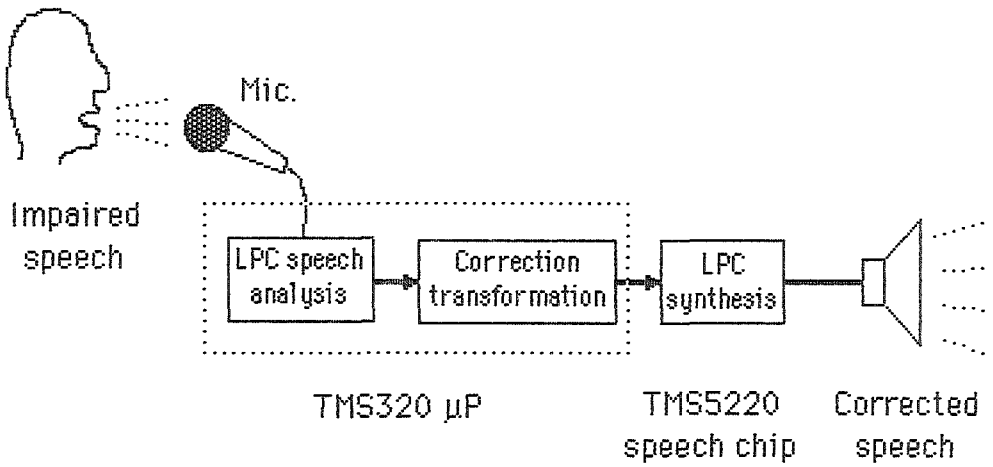


fig. 6: School of Engineering Impaired Speech system

At present the LPC speech analysis is performed on the School of Engineerings VAX 11/750 computer using SGPRC, a program written for LPC speech analysis (see reference 3). The LPC analysis is not performed in real time. The speech synthesiser part of the system has been implemented as a 3rd professional year Engineering project (see reference

2) by means of a Z80 microcomputer system driving a TMS5220 speech synthesiser chip:

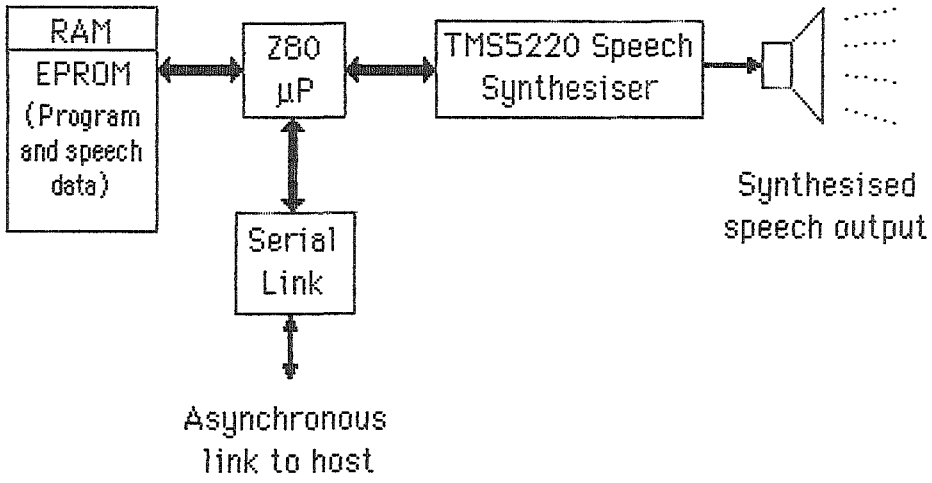


fig. 7: School of Engineering speech synthesiser

A monitor program (see reference 4) runs the serial communications link, and some software written as part of the 3rd pro. Engineering project (see reference 2) drives the TMS5220 speech synthesiser. The audible terminal is required to do much the same tasks as this system, and so the same microprocessor hardware should suffice.

The TMS5220 chip is one of the more modern LPC speech synthesiser chips available, and is capable of producing good quality speech. There was no reason to use a different LPC chip, because none available are markedly better.

Therefore, the technology to implement the audible terminal hardware already exists at the University of Canterbury. SGPRC can be used to produce the custom voice and vocabulary, and the hardware shown in fig. 7 can be used to implement the audible terminal of fig. 1, by writing a new software system.

3.2.2 Software

Because the audible terminal was to be controlled via an asynchronous communications line, the only possible way to control it was via a sequence of characters. A character sequence to activate the audible terminal was necessary because of the possibility of the asynychronous line to the audible terminal being shared with that of a visual terminal.

Complete messages were stored in the audible terminal, as opposed to using a text-to-speech algorithm, because the problems mentioned for text-to-speech algorithms (see section 2.2). A message to be spoken by the audible terminal therefore had to be selected by some means, and the only available way was to use characters sent down the asynchronous communications line, following the trigger character(s).

4. Implementation

4.1 Hardware (see also fig. 7)

The hardware used was developed by the School of Engineering (see reference 2). The Z80 CPU is driven by a program in EPROM, which takes approximately 200 bytes. Therefore, almost the entire 64 K bytes of memory will be available to store messages in, enough for 7.5 minutes of speech. Typically a system error message takes 3 - 5 seconds to speak, so over a hundred messages could be stored in EPROM. In the prototype speech synthesiser board only 24 K bytes of EPROM is available to store speech, enough for 1.5 minutes. This is more than adequate for testing purposes.

The prototype system also has a monitor program in EPROM (see references 2 and 4), that enables memory data to be examined and changed, and data to be sent to the devices controlled by the system. The monitor requires about 1 K bytes of EPROM space starting at address 0, plus some RAM located at address 8000_{16} to store system variables.

A serial communications chip interfaces to a terminal at 9600 baud, via an RS232 link.

The TMS5220 chip is driven off the system data bus, with the Z80 microprocessor sending it control bytes and data from EPROM.

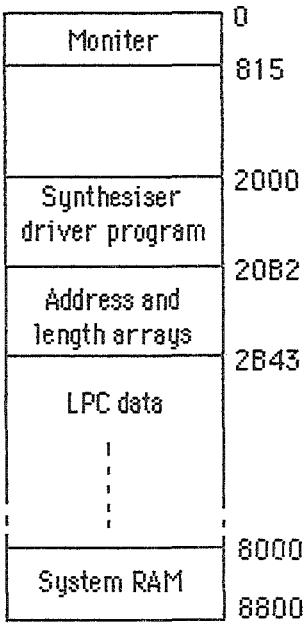


fig 8: Prototype hardware system memory map

4.2 The TMS5220 chip (see also appendix A)

The speech synthesiser hardware developed by the school of Engineering uses the TMS5220 speech synthesiser chip. The TMS5220 uses an electrical model of the human vocal tract to synthesise speech (see fig. 9). This electrical model is controlled by "frames" of LPC parameters:

	Energy	Repeat	Pitch	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10
Number of bits:	4	1	6	5	5	4	4	4	4	4	3	3	3

fig. 10: TMS5220 LPC frame

The energy parameter sets the volume of the output, for the current frame. The pitch parameter sets the timbre of the "voice". An unvoiced sound is made by selecting a pitch of zero. The ten K parameters set the characteristics of the electrical analogue of the human vocal tract.

Because the muscles of the human vocal tract change their position on average once every 25 mS, a new set of parameters is supplied to the TMS5220 chip once every 25 mS.

Ordinarly the TMS5220 chip would need 40 (frames/sec) * 50 (bits/frame) = 2000 bits per second, but several "tricks" are used to reduce the effective bit rate required:

(a) Unvoiced speech requires a less precise vocal tract model than voiced speech, because unvoiced speech does not use the larynx as its sound source. Most consonants, such as "b" and "t", are unvoiced, as opposed to vowels such as "e" and "o", and consonants such as "y", which are voiced (see also reference 1, pp 48-52). Only the Energy, Pitch and K1 - K4 parameters need be supplied to the TMS5220 to synthesise unvoiced speech.

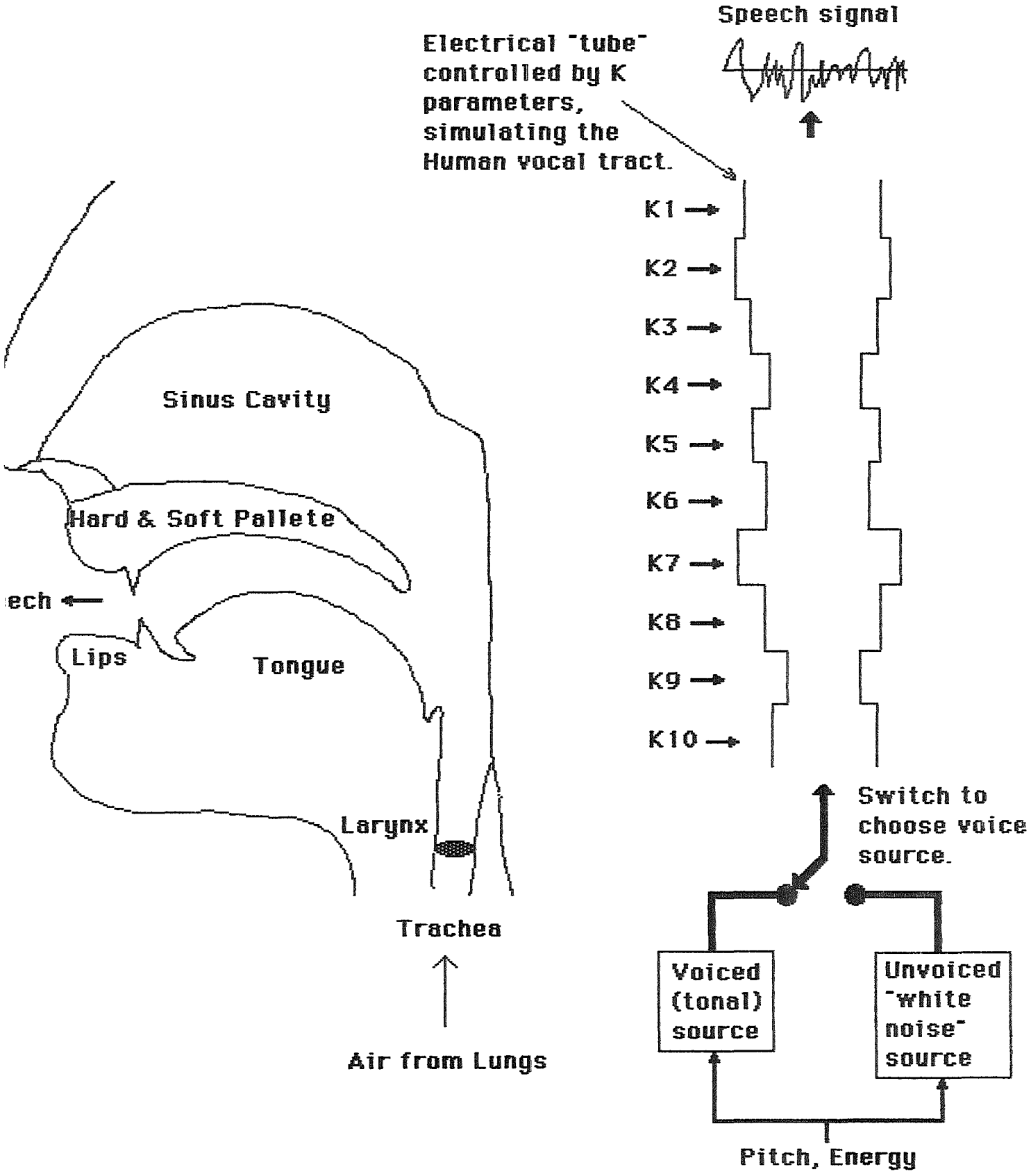
(b) The human vocal tract changes shape quite slowly, often slower than the maximum rate of once every 25 mS. Because of this, a "repeat" bit has been added to the TMS5220 frame. A repeat frame has only the energy and pitch specified, the previous K parameters being retained.

(c) When the energy is zero, silence is produced so there is no need to specify any other parameters.

(d) There is a stop code to tell the TMS5220 to stop synthesising speech; this has an energy of 15 and no other parameters.

Thus, the TMS5220 requires five different types of frames:

fig. 9: Comparisan of Human and LPC speech synthesisers



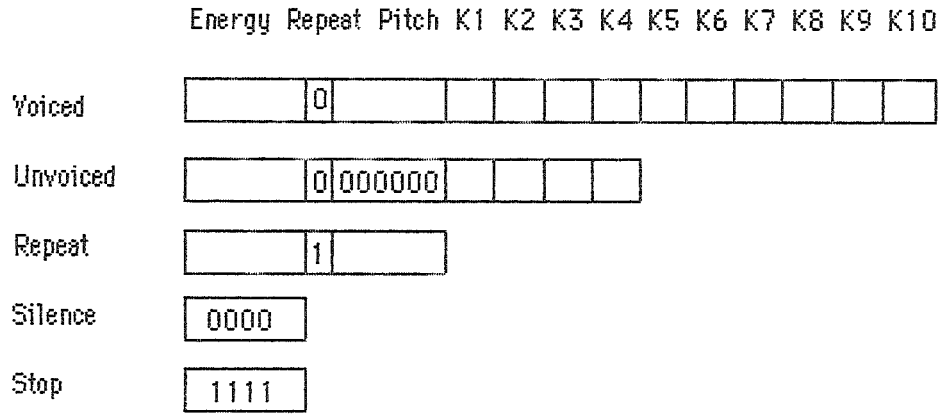


fig 11: Types of TMS5220 LPC frames

The TMS5220 is controlled via an 8 bit parallel interface, over which commands and data are sent (see appendix A).

4.3 Software

As shown in section 3.2.2, the audible terminal had to be controlled by characters. On the prototype audible terminal the trigger character is the ASCII "carriage return" character (character code 13).

Selection of the message to be spoken is done by using the ASCII code of the character following the trigger character as an index. Thus ASCII "null" character (character code 0) selects the first message stored in memory, the "start of header" character (code 1) selects the second message, and so on. Using the ASCII character set in this way 127 messages could be stored in the audible terminal. If this number was not enough up to 16129 messages could be stored by using two ASCII characters following the trigger character.

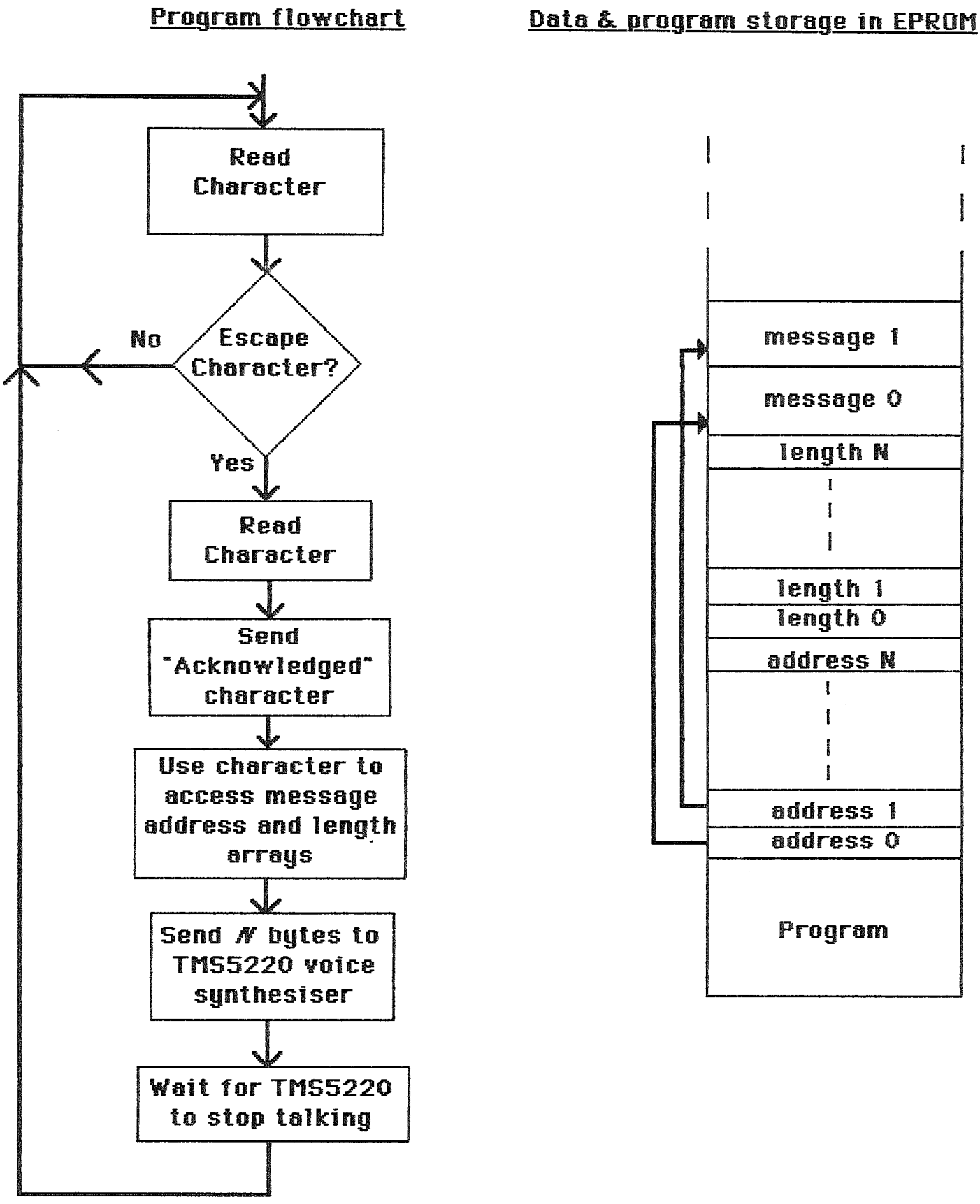
The software to drive the audible terminal is written in Z80 machine code. It works as follows (see also figure 12):

The TMS5220 is initialised when the program begins. The next steps are then repeated in an infinite loop:

Characters are read from the serial communications link until the trigger character(s) are received. Once this has happened, the next character is read and stored as an index. A character is then sent from the audible terminal to the host computer by way of an acknowledgement.

The index character is then used to access one of the messages stored in the audible terminal, as described above, and the message is spoken by the TMS5220 chip.

fig. 12: Program and Data structures



4.4 Producing a Vocabulary

At present a vocabulary is produced for the speech synthesiser as follows (see also figure 13):

1. Human speech is recorded on tape and digitised by SGPRC on the Engineering Schools VMS VAX to produce LPC encoded speech. The LPC files are transferred, via tape, to the Computer Science UNIX VAX.

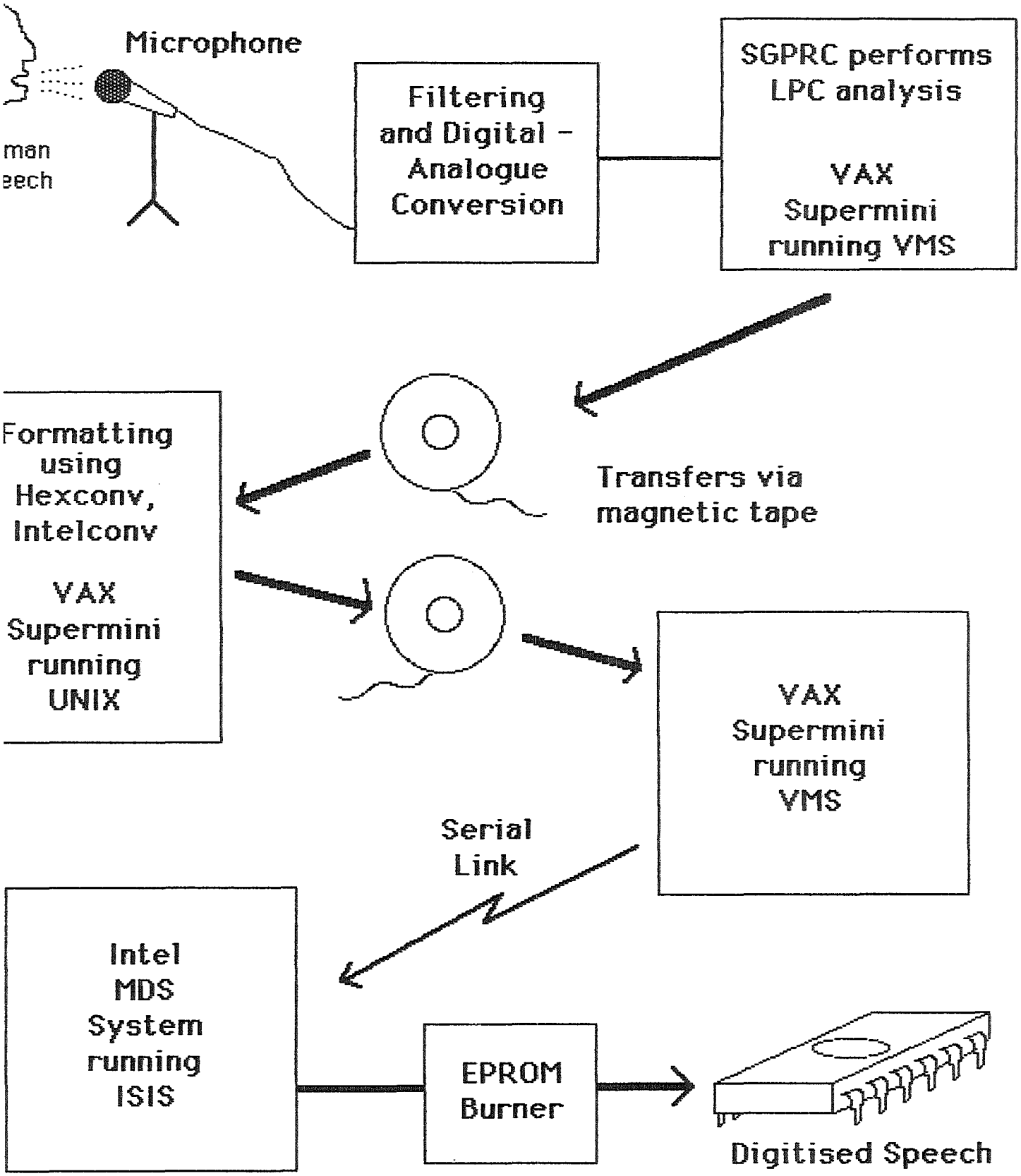
2. A program called **hexconv** is run on the files, to create the two arrays the program on the speech synthesiser board requires, and to concatenate the files together.

3. The data is appended to the end of the Z80 "audible terminal" program, ready to be burnt into EPROM.

4. **Intelconv** is run on the combined program/data files. This converts the files into the format required by the Engineering Schools EPROM burning program. The converted files are then transferred, via tape, back to the Engineering Schools VMS VAX.

5. The files are downloaded from Engineerings VAX to one of their Intel MDS microcomputer systems. These systems have an attached EPROM burner, which is what is finally used to burn the program/data files onto EPROM, ready to be installed in the speech synthesiser board.

fig. 13: Procedure for creating a vocabulary



5. Editor

5.1 *Raison d'être*

The speech output from the SGPRC program is not perfect. SGPRC has trouble knowing when speech is voiced or unvoiced, and a lot of background noise is introduced into the synthesised speech by the analysis.

An editor running on a speech synthesiser board would be useful while producing the vocabulary for the audible terminal. Because of the low data rate required by the TMS5220 (one frame every 25 mS) editing the LPC speech should provide reasonable results for the investment of time required.

The editor would be used on an audible terminal as a speech development system, to improve the output from SGPRC. The resulting edited speech could then be loaded into a dedicated audible terminal.

An LPC editor does exist on a Texas Instruments development board for the TMS5220 at the school of Engineering, but it does not work properly.

5.2 Planned Use

The editor would be used as follows:

1. Raw LPC speech is downloaded into RAM in the audible terminal/editor system from a host computer.

2. The speech is changed on a "trial and error" basis: The editor is used to alter the LPC data and the TMS5220 chip on the board is used to speak the changed data.

3. After an editing session the LPC speech is uploaded back to the host computer, ready for use in a dedicated audible terminal system.

5.3 Design Considerations

5.3.1 User Interface

The editor should be a screen-based interactive editor, with the added dimension of speaking the edited LPC speech data.

5.3.2 Compiler

The editor should be written in a high level language (HLL), as an aid to portability between systems. Writing in a HLL is considerably easier than writing in assembler, and portability between systems is greatly enhanced. The use of a HLL also enables the editor to be developed and debugged on the Computer Science VAX, and later compiled to Z80 code.

C is probably the best language for writing the editor. C has extensive facilities for low level bit manipulation such as shifts and bitwise ANDs and ORs, and good facilities for pointer manipulation. Bit manipulation is essential for handling the TMS5220 LPC frames. C also produces compact object code, an important consideration for the editor whose eventual destination will be a microprocessor system.

5.4 Implementation

5.4.1 User Interface

In order to make the editor as independant of terminal type as possible, only the most basic terminal operations were used: printing characters, linefeeds and carriage returns, and vertical tabs, to move the cursor upwards. In view of these restrictions, the screen format used by the Texas Instruments' editor seemed to be quite a good one to use:

FRAME	Energy	Rpt	Pitch	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10
0	1	0	8	32	1	8	0	8	0	3	0	5	2
1	4	0	6	20	4	9	3	2	0	5	1	3	3
2	0												
3	1	0	9	24	12	6	4	6	5	3	0	2	0
4	6	0	0	12	9	7	3						
5	4	0	5	8	6	4	5						
6	0												
7	0												
8	0												
9	0												
10	0												
11	0												
12	0												
13	0												
14	0												
15	0												
16	0												
17	0												
18	0												
19	0												
20	30	0	0	6	7	12	4						
21	18	0	13	7	9	8	2	6	6	5	4	1	2

fig. 15: Typical editor screen display

The top row of the screen identifies the different fields of the LPC frames (see section3, "TMS5220 chip"), which are displayed one after the other, down the screen. Each frame is numbered, for selection when editing the LPC data. The screen represents a window on the LPC data that is being

edited, which can be shifted around in memory.

5.4.1.1 Screen Mode Editing

The basic operation to be performed on the LPC data are to make small changes in the coefficients. This is accomplished by first selecting the frame to be changed, then the LPC parameter to be changed. The frames are selected by entering the number of the frame at the side of the display. The LPC parameter is selected as follows:

"e" for the energy parameter.
"p" " " pitch "
"r" " " repeat "
"1" to "9" for K parameters 1 to 9.
"0 for K parameter 10.

Once an LPC parameter has been selected, the terminal cursor is moved to the position on the screen of the parameter, and the replacing value can be typed in. Adding, deleting and changing LPC frames on the screen are done with the "a", "d" and "c" characters respectively, followed by fields specifying the number of the frame to be operated on and the type of frame to add/change.

The screen display can be scrolled up or down by means of the "f" and "b" keys. The "f" key is used to move the screen forward in memory from the current position, and the "b" key is used to move it backwards.

See appendix C for more detailed information on the operation of the editor.

5.4.1.2 Non Screen Mode Operations

Other operations of the LPC editor are:

Loading a file to edit. This operation is selected with an "r" (read file) character. In the current implementation of the editor, on the Computer Science VAX, the r command is followed by the filename of the LPC data to edit. In the proposed implementation on the speech synthesiser board, the LPC file will be downloaded via the asynchronous communications link.

The "w" (write file) writes the edited data. The current implementation of the editor overwrites the original LPC file with the edited data. The proposed implementation of the editor, on the speech synthesiser board, will send the edited file along the asynchronous link.

The character "q" quits from the editor, back to the UNIX shell on the current system. This operation will be unnecessary on the implementation on the speech synthesiser board, since the whole system will be dedicated to editing speech.

5.4.2 Compiler

At present, the only C compiler in the Computer Science Department that produces Z80 code is for a subset of C, written by J E Hendrix (see reference 5). This compiler at the moment produces 8080 assembler mnemonics, but fortunately the Z80 instruction set is a superset of the 8080 instruction set, so only the code-producing section of the compiler need be changed to produce the equivalent Z80 mnemonics.

The C compiler has an associated runtime library consisting of about 350 lines of 8080 assembler, and this must also be changed to Z80 mnemonics. I/O procedures such as **getchar()** and **putchar(c)** must also written, to communicate with the terminal link on the speech synthesiser.

5.5 Possible Improvements

The user interface discussed above does not present information in a very "user-friendly" way. The screen display could be made to relate more closely to the actual speech being reproduced. For instance, it would be useful to know what sort of sound was being reproduced by the sets of LPC parameters, and this sound could be displayed alongside the parameters on the screen.

5.5.1 Graphics

A graphical display of the parameters would be easy to implement, as an aid to detecting poorly reproduced speech. Because each of the Pitch, Energy and K parameters change value smoothly, it would be possible to present them in a graphical way, and so detect any irregularities visually. Large irregularities would probably be due to errors in the LPC analysis process, and smoothing them out would improve the quality of the speech.

One way of implementing a graphical display might be an option on the editor that displayed a chosen parameter as a line graph for the LPC data being analysed. Each of the LPC parameters could thus be examined visually.

5.5.2 Speech Recognition

An editor that was able to display what sounds the LPC frames it was editing represented would be a very powerful tool, making precise alterations to the LPC speech data much easier than the editor currently implemented.

The main difficulty of implementing such an editor is the variability of human speech. This is also the main difficulty with speech recognition, and in fact the problems faced by the editor would be much the same (see reference 1, chapter 7).

6. Results

6.1 Speech Synthesiser

The speech synthesiser has been successfully implemented as described in section 4. At present the speech is produced from "raw" LPC parameters, straight from SGPRC. The resulting speech is recognisable, but the quality is low.

6.1.1 Intelligibility of the speech produced

The low quality of the LPC synthesised speech is due to the difficulty that SGPRC has in extracting the characteristics of human speech used by the LPC model. Human speech is very variable, and no fixed set of rules exists for LPC analysis that work 100% of the time. As result of this, all the standard LPC vocabularies sold by manufacturers are produced by hand tuning the output from speech analysis programs.

The intelligibility of the LPC speech can be improved by improving the input to SGPRC as follows:

Low pitched male voices generally provide the best results, because of the large low-frequency component. However, women generally speak more clearly than men, so some female voices provide a good source for SGPRC.

SGPRC is more sensitive to vowels (voiced sounds) as opposed to the consonants (unvoiced), so emphasising the consonants helps the quality of the synthesised speech.

The speech used should be spoken slowly and clearly.

6.1.2 Considerations in producing the vocabulary

When translating the test vocabulary for the prototype voice synthesiser from a visual messages to audio, the nature of some of the system messages required that they be altered:

For instance, the message "Monthly (M) or Annual (A) accounts?" is nearly meaningless if spoken verbatim. This sentence was broken up into two consecutive sentences: "Enter M for Monthly accounts". "Enter A for Annual accounts".

Thus, it is not feasible to translate some sentences directly from the written form to speech. In the above example this is because the two letters **(M)** and **(A)** are intermixed with the text, but in fact they constitute a simple illustration added to the sentence. The information represented by this illustration must be converted to text form if the sentence is to be spoken. The old adage "a picture is worth a thousand words" is graphically demonstrated by the extra words that had to be

added to the original message.

6.2 Editor

The editor at present runs on the Computer Science VAX, as described in section 5. It can edit data, but no graphical facilities are available, and the editor is still in the prototype stage. The work still to be done for the editor includes:

Adding some of the facilities mentioned in section 5.5.

Changing the C compiler to produce Z80 code, and altering the runtime library.

Porting the editor to the speech synthesiser board.

7. Conclusions

Although the speech synthesiser has been successfully implemented it is not really suitable for commercial use in its present form, because the quality of the "raw" LPC speech is too low. Edited LPC speech may be suitable, although the amount of time necessary to edit speech may not be cost-effective.

The goals of a custom vocabulary and a custom voice have been realised, but largely negated by the quality of the speech produced.

For commercial use a proprietary system such as National Semiconductors "Digitalker" may be best, supplying a high quality, semi-custom vocabulary at a reasonable price (in the $100 range). Using this system requires the sacrifice of one of the goals of the project: the ability to synthesise a custom voice. Because of the semi-custom nature of the vocabulary, some system messages that use words not available from the manufacturers may have to be altered.

The editor runs on the Computer Science VAX at the moment, and when ported to the speech synthesiser board should make a useful LPC speech development system.

8. Where to from here?

From a commercial point of view, the speech synthesis technology used in the speech synthesiser should probably be changed to a proprietary system, which would provide a feasible audible terminal at a reasonable cost.

From a research point of view, the combination of the speech synthesiser board and the editor should provide a quite powerful tool for LPC research, providing a very interactive environment. Another dimension could be

added to this system by the TMS320 LPC analysis system used for the School of Engineerings speech project (see section 3). With the TMS320 system interfaced to the speech synthesiser board, speech could be LPC analysed by the TMS320, then edited and spoken by the system, all interactively.

Appendix A: TMS5220 chip

The speech synthesiser chip used by the School of Engineering group is the Texas Instruments TMS5220. It implements an electrical model of the human vocal tract (see fig. 9). The model consists of a series of electrical filters, with excitation provided from two sources:

(a) A sawtooth waveform for voiced sounds (vowels, and consonants such as z).

(b) A white noise generator for unvoiced sounds (consonants such as k and s).

The filters change the character of the sound sent through them from the excitation source, in a way that mirrors the way sounds from the lungs and vocal cords are changed by the human vocal tract. Parameters to these filters and to the excitation source are sent to the TMS5220 chip from an external source, to produce speech output.

The basic form of these parameters is a "frame", which contains several categories of information:

	Energy	Repeat	Pitch	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10
Number of bits:	4	1	6	5	5	4	4	4	4	4	3	3	3

fig. a1: TMS5220 LPC frame

The energy parameter sets the volume of the output, for the current frame. The pitch parameter sets the timbre of the excitation source. An unvoiced source is selected by a pitch of zero. The ten K parameters set the characteristics of the electrical filters that the excitation is passed through.

Because the muscles of the human vocal tract change their position on average once every 25 mS, a new set of parameters is supplied to the TMS5220 chip once every 25 mS.

The TMS5220 chip uses several "tricks" to reduce the bit rate that it needs to synthesise speech:

(a) The minimum possible number of bits is used for each parameter. The higher order K parameters have less effect on the speech produced, so they are assigned fewer bits for their representation.

(b) Unvoiced speech requires a less precise vocal tract model than voiced speech, so that only the Energy, Pitch and K1 - K 4 parameters need be

supplied to the TMS5220 for unvoiced speech.

(c) The human vocal tract changes shape quite slowly, often slower than the maximum rate of once every 25 mS. Because of this, a "repeat" bit has been added to the TMS5220 frame. A repeat frame has only the energy and pitch specified, the previous K parameters being retained.

(d) When the energy is zero, silence is produced so there is no need to specify any other parameters.

(e) There is a stop code to tell the TMS5220 to stop synthesising speech; this has an energy of 15 and no other parameters.

Thus, the TMS5220 requires five different types of frames:

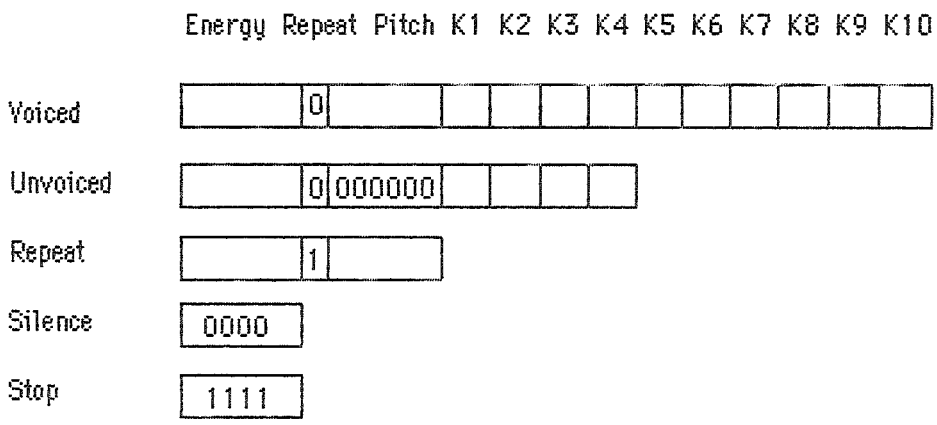


fig a2: Types of TMS5220 LPC frames

The TMS5220 chip is interfaced to its controller by an 8 bit parallel bus. The bus connects to an internal FIFO which buffers data. This is essential because the LPC parameters supplied to the TMS5220 don't necessarily start and end on a byte boundary

The TMS5220 is controlled by sending a control byte, specifying the operation it is to perform, followed by any data required by the operation:

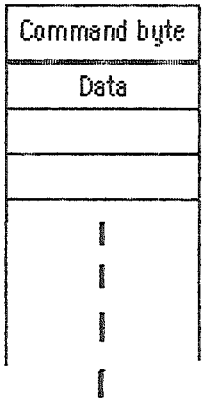


fig. a3: TMS5220 command and data sequence

There are six different commands the TMS5220 can execute. The only commands used in the audible terminal are the "reset", "read byte" and "speak external" commands. The other commands are for use with Texas Instruments ROM sets, that have custom vocabularies. The custom ROM is the "VSM" (Voice Synthesis Memory) that is referred to in these commands:

Command	Code	Operation
Read Byte	XXXX100X	Read byte from VSM
Speak External	XXXX011X	Speak data from external source
Read and Branch	XXXX110X	Read byte from VSM & branch to new VSM address
Load Address	AAAA001X	Alters VSM address
Speak	XXXX101X	Speak data from VSM
Reset	XXXX111X	Reset the TMS5220

fig. a3: TMS5220 commands

The "speak external" command makes the TMS5220 accept LPC data from an external source via the 8 bit bus, and use it to synthesise speech.

The "reset" command makes the TMS5220 do an internal reset.

The TMS5220 has a status byte that can be read from the 8 bit bus, which has the following format:

Bit #: 7 6 5 4 3 2 1 0

| TS | BL | BE | X | X | X | X | X |

fig. a4: TMS5220 Status Byte

The Talk Status (TS) bit is set if the TMS5220 is talking, and is cleared if it is idle. The Buffer Low (BL) bit is set if the TMS5220s internal FIFO buffer is less than half full. The Buffer Empty (BE) bit is set if the TMS5220 runs out of data while synthesising speech (ie if it is not kept "topped up" with fresh LPC frames).

Appendix B: Software

1. Speech synthesiser software operation

The software written to operate the audible terminal operates as follows (see also fig. 12):

1. The TMS5220 is initialised by sending 10 "reset" commands to it.

2. The serial communications chips status is continually read until it signals that it has a character ready.

3. The character is examined to see if it is the escape character. At present the escape character is the ASCII carriage return character. If the received character isn't the escape character the program returns to step 2.

4. If the character is the escape character, the next character received by the serial communications chip is assumed to be a binary index to the message to speak.

5. An character is sent back down the asynchronous communications line, to signal to the host computer that the character sequence was received.

6. The chosen message is sent to the TMS5220 chip from EPROM.

7. The TMS5220 "TS" status bit (see figure a2) is repeatedly examined until it signals that speech synthesis has finished.

8. The program returns to step 2, waiting for the next character.

2. Data Structures

The messages are stored in EPROM as follows (see also fig.12):

The first 1024 bytes after the program contain two arrays, indexed by the message number to be spoken:

An array of 256 16 bit addresses. These are the addresses of the messages stored in EPROM.

An array of 256 16 bit integers. These are the lengths of the messages, in bits.

Thus, the program knows how many bytes to send to the TMS5220 chip, and from what address, for a particular message. The rest of the memory following these two arrays is available for storing the LPC encoded messages.

3. Hexconv

Hexconv is a program that concatenates the individual LPC message files together and creates the address and length arrays required by the Z80 program in the speech synthesiser. Hexconv runs on the Computer Science VAX under UNIX.

When it is run, hexconv asks for the start address of the data in memory. This enables the LPC data that follows the Z80 program to be relocated. Hexconv then repeatedly asks for files of LPC data, until it is indicated that there are no more.

Each LPC data file is sequentially assigned a location in the address and length arrays. The lengths of the LPC files are used to set up the address arrays. Each LPC data file is appended on to the current block of data. In this way the data structure of fig. 12 is built up.

4. Intelconv

Intelconv is a program that runs on the Computer Science VAX. It converts files of hexadecimal characters into the format required by the Intel MDS system EPROM burner program. This format is:

:<line length> <address> <data> <checksum>

<line length> is the 2 digit hexadecimal number of bytes of data in the line. This number must be either 10_{16}, 20_{16} or 40_{16}.

<address> is a 4 digit hexadecimal number with the current address of the line of data. <address> must be increase in increments of <line length>.

<data> is <line length> bytes of hexadecimal bytes.

<checksum> is the negation of the sum of all the preceeding 2 digit hexadecimal numbers, modulo 100_{16}.

Appendix C: Using the LPC Editor

c1. Invoking the Editor

The editor can be accessed on the Computer Science VAX under the directory:

/usr/users/cosc802/dale/ed

c2. User Interface

Once the editor is running, the screen is set up in the format of fig. a1. When the editor first runs, all the data fields are blank, since there is as yet no data in the editors workspace. All editor commands are single characters, followed by parameters for the command. Commands can be either upper or lower case letters.

FRAME	Energy	Rpt	Pitch	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10
0		1	0	8	32	1	8	0	8	0	3	0	
5	2												
1		4	0	6	20	4	9	3	2	0	5	1	
3	3												
2		0											
3		1	0	9	24	12	6	4	6	5	3	0	
2	0												
4		6	0	0	12	9	7	3					
5		4	0	5	8	6	4	5					
6		0											
7		0											
8		0											
9		0											
10		0											
11		0											
12		0											
13		0											
14		0											
15		0											
16		0											
17		0											
18		0											
19		0											
20		30	0	0	6	7	12	4					
21		18	0	13	7	9	8	2	6	6	5	4	
1	2												

fig. a1: Screen format of the editor

The editor requires that all numerical values entered are 2-digit decimal numbers. For example, the value 2 is entered as 02. This is done so that no delimiter characters (such as the carriage return key) are needed for numbers.

Mistakes, such as an unknown command, or a letter entered where a number was expected, are signalled by a beep from the terminal. The editor then waits for the next command.

c2.1 Loading a file

The character "r" selects this operation. After this command, the editor will ask for the name of the file to load. If the file is accessible, it will be loaded, and if not the editor will reply "<filename> does not exist" and ask again for a file to load.

c2.2 Writing a file

A file is written by the "w" (write file) operation. The name of the file that was loaded (see section c2.1) is overwritten with the edited data.

2.3 Screen-mode Editing

Screen-mode editing is done using single letter commands. A command is followed by the number of the LPC frame to be operated on (from the FRAME field at the left of the display, see fig. a1), and optionally by a frame type or LPC field specifier.

c2.3.1

LPC data is changed using the "e" (edit) command. The syntax of the edit command is:

e<frame number><LPC field>

<frame number> is one of the numbers of the frame field at the left of the screen (see fig. a1).

<LPC field> selects one of the individual parameters in the LPC frame to be edited, as follows:

e: selects the energy parameter
r: " " repeat parameter
p: " " pitch parameter
1: " " K1 parameter
2: " " K2 "

..... etc .....

9: " " K9 "
0: " " K10 "

Once the LPC frame and field have been selected, the terminals cursor is moved to this position on the screen. The new value for the field can then be typed in.

c2.3.2 Adding a frame

This operation inserts a new LPC frame at a specified positions. The add operation has the following syntax:

a<frame number><frame type>

<frame number> is specified as in section c2.3.1, and selects the frame before which the new frame is to be inserted.

<frame type> is one of the following:

v: selects a voiced frame
u: " an unvoiced frame
r: " a repeat frame
s: " a silence "
e: " an end "

The parameters of the new frame are initialised to zero except where they must be non-zero to specify the frame type, in which case they are initialised to one. For instance, an unvoiced frame must have its energy field non-zero, or it will be interpreted (and displayed) as a silence frame (see also fig. 11). The energy field is therefore initialised to one.

c2.3.3 Deleting a frame

The "d" (delete) command removes a frame from the LPC data, at a specified position.

The syntax of the delete command is:

d<frame number>

Where <frame number> (specified in section c2.3.1) selects the position on the screen of the frame to be deleted.

c2.3.4 Changing the type of a frame

The type of a frame is changed using the "c" (change) command, which has the following syntax:

c<frame number><frame type>

<frame number> is specified as for section c2.3.1, and selects which frame on the screen is to be changed.

<frame type> selects the type that the frame is to be changed to, and is specified as for section c2.3.2.

When a frame is changed, the parameters of the old frame type are retained for the new type as much as possible. For instance, if a voiced frame is changed to a repeat frame, the energy and pitch parameters of the voiced frame are retained, and their values used in the energy and pitch parameters of the repeat frame.

c2.5 Manipulating the screen

The screen display represents a "window" on to the LPC data being edited, that can be moved forwards and backwards, from the start of the data to its end. There are two single letter commands that move the screen around in memory:

f (forward): This command moves the screen forward 10 LPC frames.

b(back): This command moves the screen backwards 10 LPC frames.

In case the screen display is corrupted from an external source, the "p" (print screen) command redraws it.

c3. File formats

The editor expects its input files to be ASCII characters. These characters must be in the range 0-9, and A - F (or a - f), and characters outside this range are ignored when reading in a file.

The characters are assumed to represent hexadecimal nibbles, in the range

0 to F_{16}. Pairs of nibbles are assumed to be bytes, the most significant nibble first. If there are an odd number of nibbles in the file that is read in then the last nibble is discarded.

When an edited file is written by the editor, the output format is ASCII characters representing hexadecimal bytes, as described above.

c4. Command Summary

a <frame number> <frame type>: Add a frame of type <frame type> at position <frame number> on the screen.

Example: To insert an unvoiced frame at frame number 2 on the screen, enter "a02u".

b: Move the screen display forward 10 LPC frames.

c <frame number> <frame type>: Change a frame at position <frame number> to <frame type>. All parameters compatible with the frame are conserved.

Example: To change an unvoiced frame to a voiced frame at frame number 12 on the screen, type "c12v".

d <frame number>: Delete a frame at position <frame number> on the screen.

Example: To delete frame number 3 on the screen, enter "d03".

e <frame number> <LPC field spec>: Edit an LPC parameter selected by <LPC field spec> at frame <frame number> on the screen.

Example: To change the K9 parameter of frame number 3 on the screen to the value 7, enter "e03907".

f: Move the screen display forward 10 LPC frames.

p: Redraw the screen display.

q: Quit from the editor.

r: Read a file of LPC coefficients.

Example: To read in the file "LPC.data" enter "rLPC.data <carriage return>".

w: Write the edited file previously read.

References:

[1] Bristow, G. et. al.: *Electronic Speech Synthesis;*

[2] Turner, S. G.: *"LPC Speech Synthesiser, A report submitted for 3rd pro. Electrical and Electronic Engineering, 1984";*

[3] Breiseman, N. B.: *"Impaired Speech Recognition" (an Elec. Eng. Ph.D paper, yet to be published);*

[4] Wignall, T.: *"S. D. Man Listing";*

[5] Hendrix, J. E.: *The Small-C handbook;*