Microprocessor Based Analog Speech Scrambler/Descrambler

by

Dhiya Abdulatif Al-Dulaijan

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

ELECTRICAL ENGINEERING

May, 1986

Order Number 1355713

Microprocessor based analog speech scrambler/descrambler

Al-Dulaijan, Dhiya Abdulatif, M.S.
King Fahd University of Petroleum and Minerals (Saudi Arabia), 1986

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

MICROPROCESSOR BASED ANALOG SPEECH SCRAMBLER/DESCRAMBLER

Dhiya Abdulatif Al-Dulaijan

ELECTRICAL ENGINEERING

MAY 1986

UNIVERSITY OF PETROLEUM AND MINERALS

DHAHRAN : SAUDI ARABIA

This thesis, written by Mr. Dhiya Abdulatif Al-Dulaijan under the direction of his Thesis Committee, and approved by all its members, has been presented to and accepted by the Dean, College of Graduate Studies, in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering.

Dean, College of Graduate Studies

Date: 7/12/86

Department Chairman

Thesis Committee

Chairman

Member

Member

This thesis is dedicated to my parents

ACKNOWLEDGEMENTS

Acknowledgement is due to the University of Petroleum and Minerals for support of this research.

I wish to express my appreciation to Professor Gerhard F. Beckhoff who served as my major advisor. I also wish to thank the other members of my Thesis Committee, Dr. Heinz Koehler and Dr. Suleyman S. Penbeci.

Thanks are also due to Mr. Muhammad Khalid Butt, Secretary in EE Department for typing this manuscript.

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

الخلاصـــــــــة

في هذا البحث تم تطوير واختبار برامج الاجهزة المصممة على المعالج الالكتروني لاستخدامها في المخاطبات الخاصة .

تركز البحث على عملية التغيير النسبي لاجزاء الموجة الصوتية ، وذلك باستخدام طريقتـــين أساسيتين احداهما تجزء الموجة الصوتية وتغيير في اماكن الاجزاء .

والاخرى تعكس أجزاء من الموجة الصوتية ، وقد أجرى تطبيق واختبار هذه البرامج علـــــى كمبيوتر (IBM) الشخصي .

ABSTRACT

Algorithms for a microprocessor-based analog speech privacy device have been developed and tested. Emphasis is on time domain speech signal encoding such as time segment permutation (TSP) and reversed time segmentation (RTS). Implementation and test on the IBM personal computer is discussed. Feasibility of combining TSP and RTS scrambling algorithms to achieve an increased degree of privacy is investigated.

INTRODUCTION

There has been an expected rapidly growing interest in and development of secure communication techniques all over the world in recent years.

Many reasons contribute to this trend, such as the increasing demand for communication privacy in industrial and business areas as well as for military purposes, the replacement of paper media by telecommunications in conducting business, the fast development in computer technology, the rapid decrease of cryptanalysis cost, etc.

There is an increasing interest in analog scramblers due to a strong desire for secure speech communication over existing telephone channels with standard telephone bandwidth at acceptable speech quality and reasonable cost [3]. In order to scramble a speech signal, this interrelation between frequency, time and amplitude will have to be changed by transforming one or several of these three variables according to a specific algorithm [1].

In this research we are manipulating only the amplitude and time variables.

Speech scrambling methods can be grouped into two categories: digital and analog.

In digital scramblers, the voice signal is first converted to digital form, and the resulting bit stream is subsequently encrupted. This method has the advantage that signal quality is not degraded by the scrambling operation.

The digital scramblers commercially available today convert the signal by means of adaptive delta modulation which requires a minimum bit rate of 12 kb/s for acceptable speech quality.

Several papers described scrambling techniques for digital scramblers e.g. [9,16]. Analog scramblers modify the voice signal by operating on the amplitude, time, or frequency or any combination thereof. The scrambled signal can be transmitted within the same bandwidth as the original signal. The following scrambling methods fall into the analog category:

- Frequency inversion [4]
- DFT scrambling [3]
- FFT scrambling [15]
- Time segment permutation [4,7,8,14]
- Reverse time segmentation [4]
- Amplitude scrambling [10]

It is obvious that these methods are easily unscrambled when a single fixed code is used. The use of a rolling code (i.e., a code which continuously changes with time) improves security considerably.

The previous analog approaches achieve speech signal privacy by manipulating the time segment. When operating within the restriction of standard voice channel bandwidths, it becomes difficult to simultaneously achieve implementation simplicity, non-degraded voice quality, and a highly secured system.

A new time manipulation method is discussed in which individual time segments are transmitted in the forward or reversed order from their generation depending on the input key and a pseudo-random generator.

Implementation of these methods on an 8088 microprocessor increases the flexibility and security of the scrambling system.

The first chapter is a general overview of a microprocessor-based analog speech privacy device and supporting components.

The second chapter discusses implementing the IBM personal computer as an analog speech privacy device.

The third chapter describes the generation of the tables used for permutation. It also explains the automatic key change routine, which updates the key (or permutation pattern).

In the chapters (4-7) four scrambling methods are discussed. They are block permutation method I, block permutation method II, reversed segmentation and amplitude modification.

The last chapter describes several factors that could be used to relatively measure the effectiveness of various scrambling methods and thus the degree of communication security they provide.

CHAPTER 1

MICROPROCESSOR-BASED SPEECH SCRAMBLER SYSTEM

1.1 SYSTEM DESCRIPTION

The basic configuration for a microprocessor based analog speech privacy device (ASPD) is shown in Fig. 1.1.

The speech signal is lowpass filtered, changed from analog to digital format by a converter (ADC) and is applied to a digital processor.

The processor, under the control of the mode and key inputs, performs scrambling of the digitized samples of the speech signal. Depending on the algorithm used, the scrambling may consist of amplitude modification, reversed segmentation or time permutation, as well as any combination of these methods.

After completing the scrambling operation, the processor delivers the words representing the scrambled speech samples to the digital-to-analog converter (DAC).

The DAC changes the digital signal samples into a step-like sequence of voltage levels.

This sequence of voltage levels, after being passed through the lowpass filter that removes higher order harmonics, represents the scrambled analog speech signal.

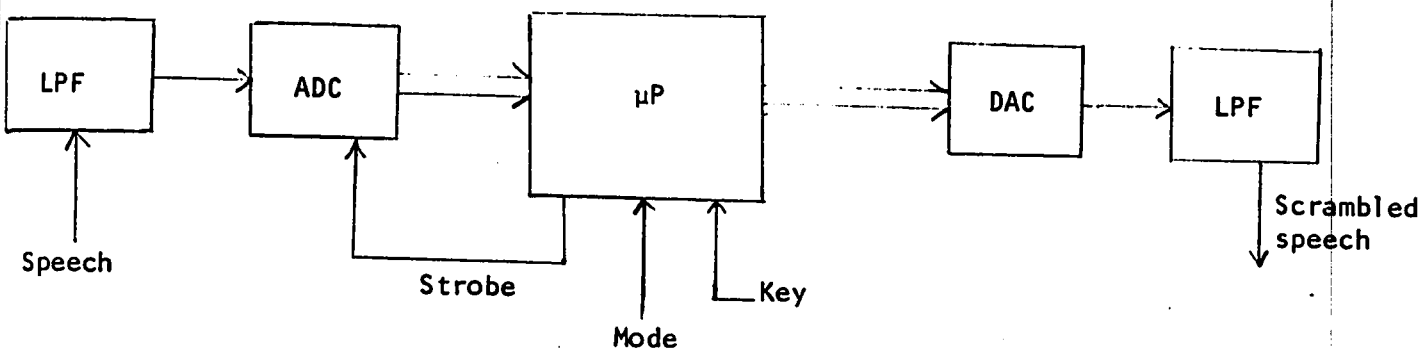


Figure 1.1. Basic configuration for a microprocessor-based ASPD.

1.2 MICROPROCESSOR CHOICE

The Intel 8088 microprocessor is an 8-bit data-bus microprocessor with 16-bit internal architecture, and an address bus of 20-bits which can address up to 1 Mbyte of memory.

It has been chosen for the microprocessor based scrambler/descrambler system for the following capabilities:

- Its ability to manipulate 16-bits as well as 8-bits data.
- It has very powerful data-transfer and string-manipulation instructions, which are used by the permutation algorithm, i.e. the following instructions move the data in TABLE 1 to TABLE 2.

```
MOV S1, OFFSET TABLE1
MOV D1, OFFSET TABLE2
MOV CX, TABLE LENGTH
REP MOVSB
```

- The 8088 microprocessor has a "TEST" pin which could be used with the "WAIT" instruction to synchronize the descrambler with the scrambler.

  If the TEST is low, execution continues; if TEST is high, the 8088 waits in an idle state until the pin goes low.
- It is a high speed microprocessor. For example, it adds two 16-bit registers in three clock cycles. For clock rate of 4.77 MHz, the addition takes

$$3 \times \frac{10^{-6}}{4.77} \text{ s} = 3 \times .21 \ \mu\text{s} = 0.63 \ \mu\text{s}.$$

1.3 ADC AND DAC

1.3.1 An analog voice signal is converted into a bipolar sequence of binary 8-bit words.

The rate of sampling is determined by the frequency of the strobe signal (generated by the microprocessor under program control) or by a free running clock which is typically at least twice the Nyquist frequency of the lowpass-filtered signal. Thus, for a 3.2 KHz lowpass filter cutoff frequency, the sampling rate should be 7.0 KHz or higher.

1.3.2 The microprocessor sends samples to the DAC at the same rate it receives them from the ADC.

Both ADC and DAC communicate with the microprocessor through a latched input/output port (eg. the ports of the 8255A Programmable Peripheral Interface Support Chip).

1.4 MICROCOMPUTER ARCHITECTURE OVERVIEW

When used as the central element of an analog speech privacy device, the microprocessor interacts with a number of support chips, thus forming a microcomputer unit.

The block diagram for such a microcomputer is shown in Fig. 1.2.

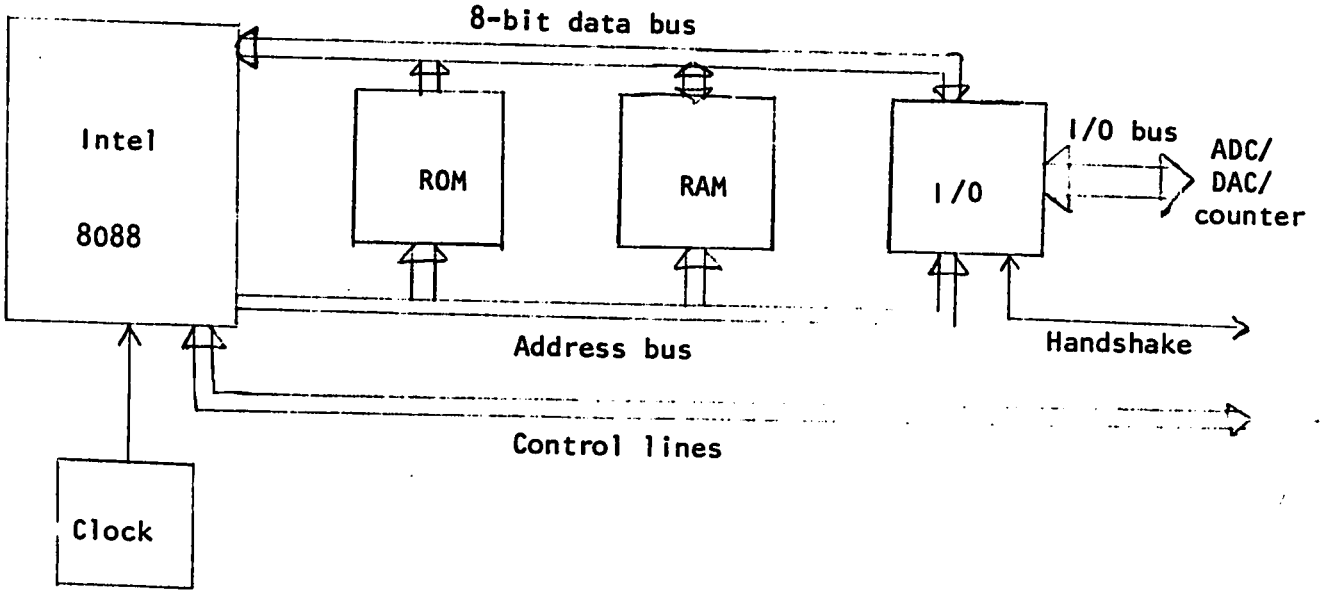


Figure 1.2. Microcomputer architecture.

1.4.1 The Random Access Memory (RAM) unit is used for storing the dynamically changing data such as speech samples, the variable elements and intermediate results of the scrambling/descrambling algorithm. For time domain scrambling, such as Time Segment Permutation (TSP) and Reversed Time Segmentation (RTS), the size of the RAM is determined primarily by the encoding interval (or frame). Depending on the sampling rate used, the RAM capacity requirement is calculated as follows:

1) For permutation

RAM capacity = 2 x encoding interval x sampling rate

since at least two buffers are needed in the permutation.

For an encoding interval of 0.5 sec. and a sampling rate of 8 KHz, the RAM buffer needed is

2 x 0.5 x 8000 = 8 Kbyte.

2) For reversed segmentation

RAM capacity = encoding interval x sampling rate

since only one RAM buffer is needed by this method.

For example, using an encoding interval of 0.2 second and a sampling rate of 7.00 KHz the RAM buffer needed is

0.2 x 7000 = 1.400 Kbyte.

1.4.2 The fixed elements of the scrambling algorithm, as well as the housekeeping functions, are stored in the Read-Only Memory (ROM) unit of the microcomputer. Consequently, the ROM capacity is a function of the algorithm complexity and operating modes required. The ability of a

microprocessor to address and manipulate the data stored in the RAM and ROM units is one of the major advantages of the microprocessor-based analog speech privacy device. For example, different encoding/decoding algorithms can be stored at various addresses within the ROM, thus providing the ASPD with multimode capability.

Moreover, a replacement of the ROM chip can provide for a performance update of the microprocessor-based ASPD equipment operated by the user.

1.4.3 The peripheral input/output device interfaces the microprocessor to ADC, DAC and the counter. The control and handshake lines carry the signals which determine the type and direction of the data exchange.

Interrupts are among the most important signals carried by the control lines.

The counter is used to interrupt the microprocessor for inputting a sample from the ADC and outputting a sample to the DAC.

CHAPTER 2

IMPLEMENTATION ON IBM PC

2.1 TEST SET UP

The scrambling and descrambling algorithm were tested on the IBM PC using the set up in Fig. 2.1. The scrambled speech samples as well as the synchronizing signal are passed from the scrambler to the descrambler through a programmable parallel port.

The cutoff frequency of the lowpass filters used was set to 3.4 KHz. The circuit diagram of the lowpass filter is shown in Fig. 2.2.

2.2 LAB TENDER BOARD INITIALIZATION

A Lab Tender Board was used with the IBM PC to provide the following functions:

- Analog to digital conversion
- Digital to analog conversion
- Programmable parallel ports
- Counting.

The base address of the board (the first I/O location occupied) is set with switches SW1 and SW2 to I/O address of 240 decimal. Lab Tender occupies 16 consecutive addresses in the I/O space of the PC. Each switch corresponds to one bit in the address as shown in Appendix A.

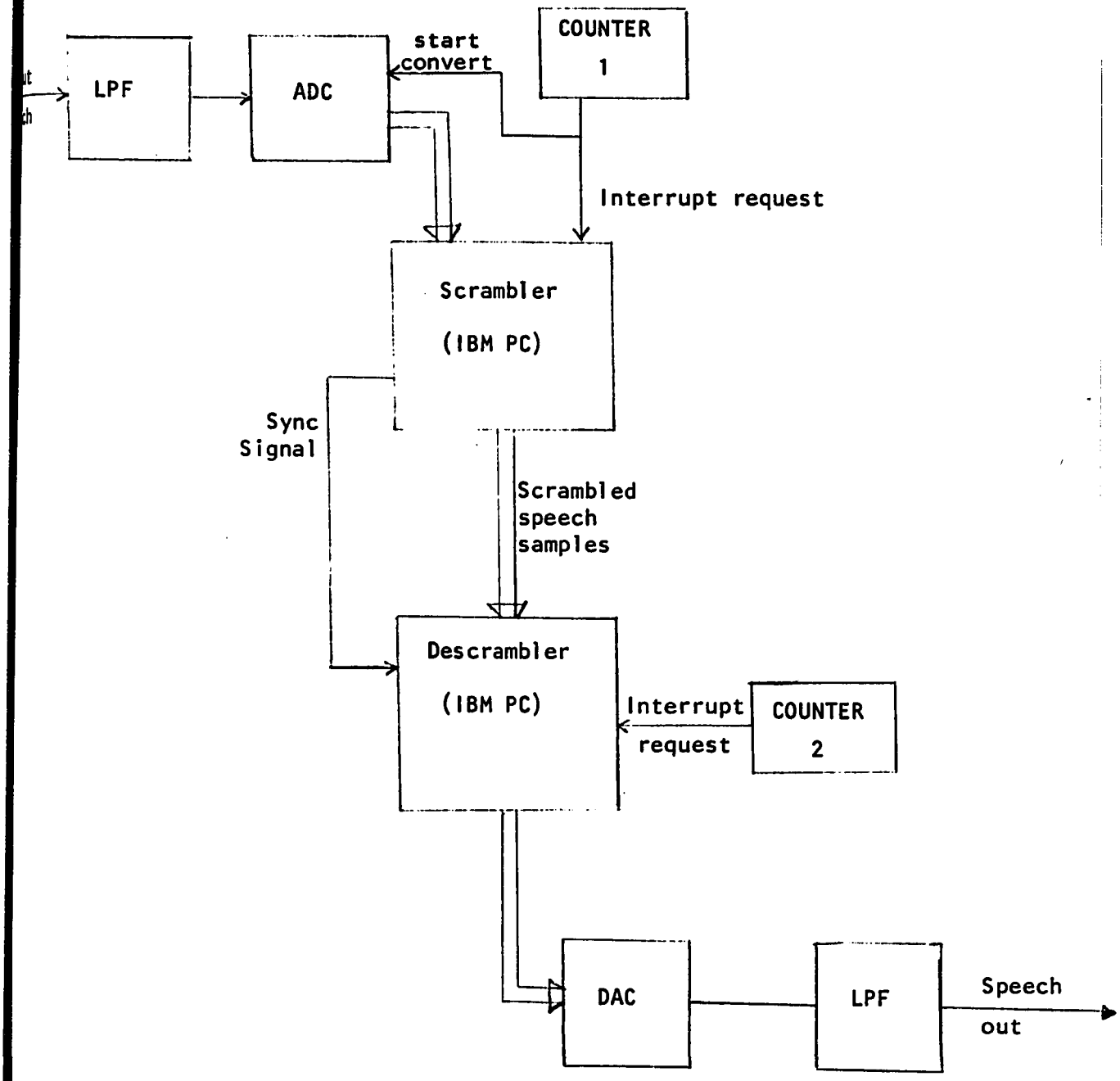


Figure 2.1. Test set-up.

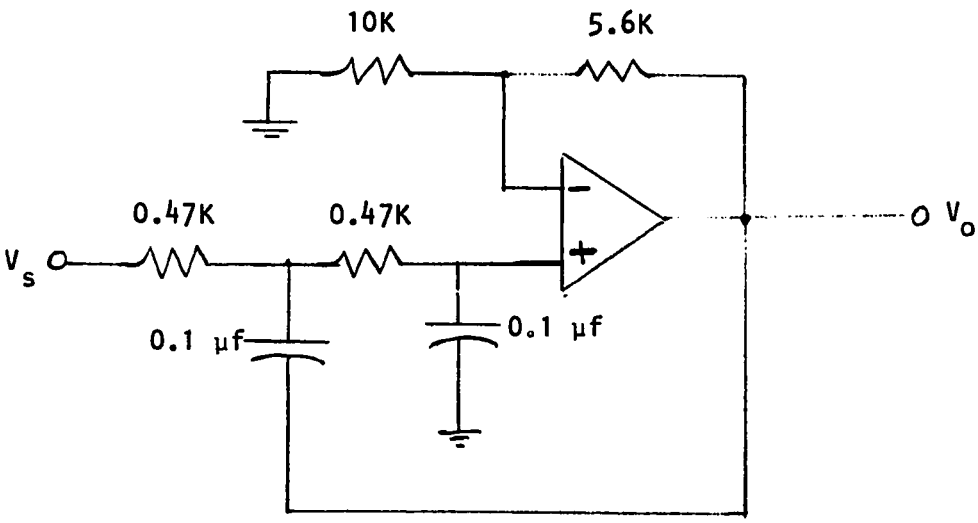


Figure 2.2. Second order Butterworth Lowpass filter.

2.2.1 The AM9513 system timing controller on the Lab tender board is a large scale integrated (LSI) circuit with five general-purpose, 16-bit counters. It also contains an internal oscillator with the associated frequency scaling circuitry for diving frequencies. The only counter we use in the AM9513 is counter 1. Counter 1 in Fig. 2.1 is connected to the interrupt flip-flop on the board, which in turn is connected to the interrupt request level 3 of the IBM PC.

It interrupts the processor periodically at a frequency of the sampling rate. The timer routine initializes the counter for an output frequency of 7.81 KHz.

This is the listing of the timer routine.

```
 TIMER PROC NEAR
; SET UP TIMER TO INTERRUPT
; EVERY 128 MICROSEC
MOV AL, 193     ; DISARM
OUT 249,AL      ; COUNTER 1

MOV AL,1        ; SELECT MODE
OUT 249,AL      ; REGISTER

MOV AL,34       ; SELECT COUNT REPETITIVELY,RELOAD FROM LOAD
OUT 248,AL      ; REGISTER AND TOGGLE OUTPUT

MOV AL,11       ; COUNT ON RISING EDGE, USE
OUT 248,AL      ; F1 AS A SOURCE (F1 FREQ. IS 1MHZ)
```

```
MOV AL,9        ; SELECT LOAD
OUT 249,AL      ; REGISTER

MOV AL,64       ; LOAD COUNT VALUE
OUT 248,AL      ; OF 64, GIVES 128 MICROSECONDS PERIOD

MOV AL,97       ; LOAD AND ARM
OUT 249,AL      ; COUNTER 1
RET
TIMER ENDP
```

2.2.2 The parallel ports in the scrambler are programmed as output ports by sending a hex value of (80H) to the parallel port control register address 255 decimal. The ports in the descrambler are programmed as input ports by sending a hex value of (9BH) to the parallel port control register.

2.2.3 ADC is initialized to run on external triggering by writing the value (40H) to the ADC control port of address 240 decimal. The output of Counter 1 is used as the external trigger for the ADC.

2.3 INTERRUPT VECTOR TABLE INITIALIZATION

The first 1024 bytes of memory in an 8088 microprocessor system are devoted to the interrupt function in an 8088 processor system. The 8088

MPU architecture will support a maximum of 256 interrupts. The low RAM area is used to store the addresses of each of the 256 possible interrupt-service routines. Each address is made up of two 16-bit values, an instruction pointer (IP), and a code segment (CS) value, when one of the 256 possible interrupts occurs, this table is accessed, and the associated CS and IP values are used to branch to the correct interrupt service routine.

Figure 2.3 illustrates the use of the low RAM area for all types of 8088 MPU interrupts.

Figure 2.4 shows the memory map for the interrupt vector table, as initialized by BIOS.

The scrambling and the descrambling algorithms use interrupt request 3 (IRQ3) to input and output speech samples every 125 µs. The following routine sets up the vector interrupt table for this function:

```
MOV AX,0
MOV ES,AX              ; Point extra segment at the
                         interrupt service routine
                         address table
MOV DI,44              ; Offset entry to IRQ3
MOV AX,OFFSET IOR      ; Offset of the service routine
CLD                    ; Set "FORWARD" string
                         operations
STOSW                  ; Place it in the table
MOV AX,CS              ; Segment of our service routine
STOSW                  ; Place it in the table
```

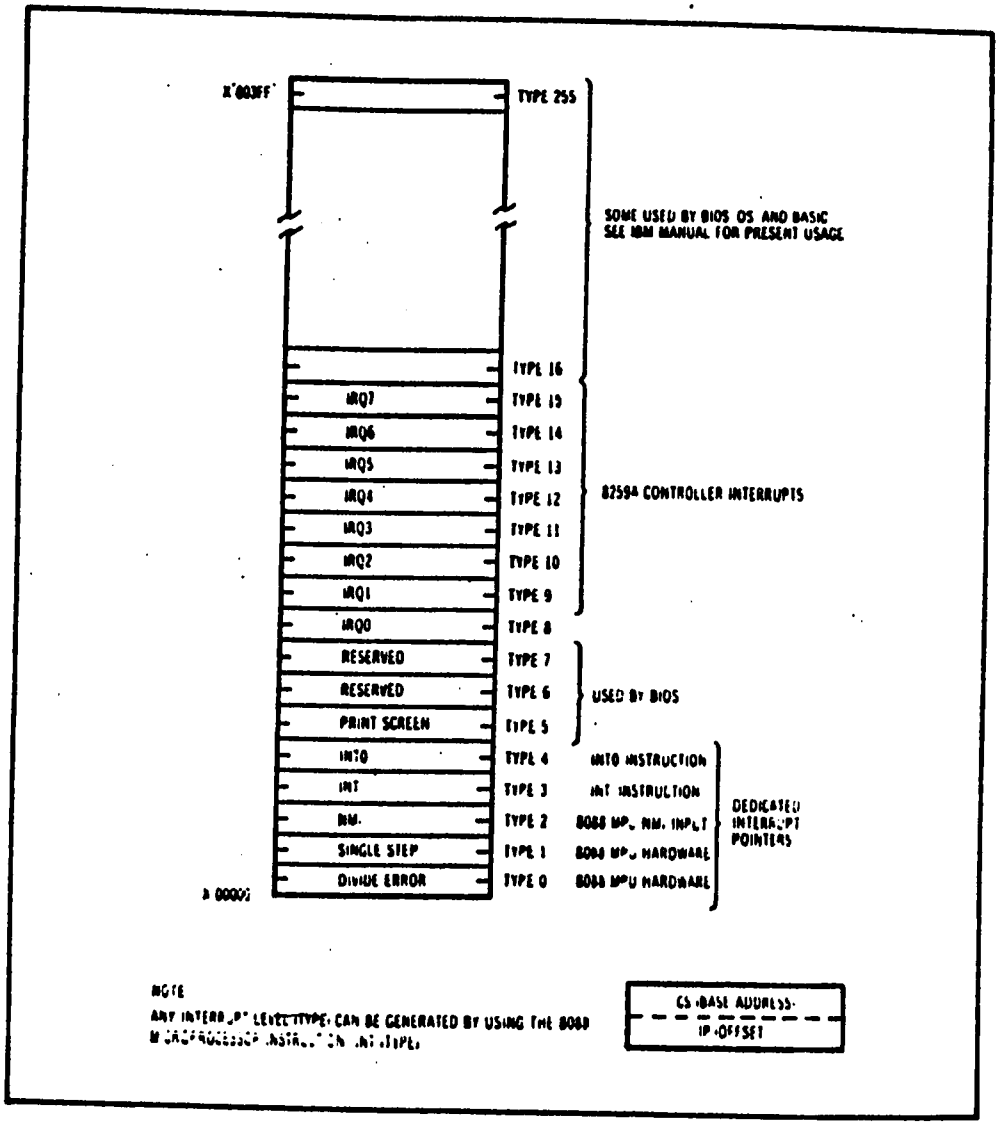


Figure 2.3. Vector pointers of the 8259A interrupt controller.

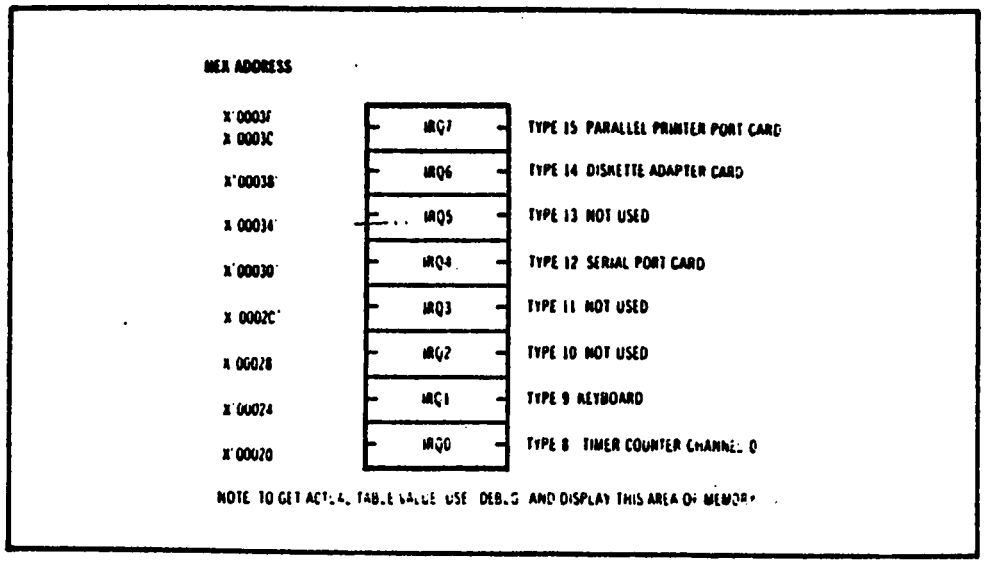


Figure 2.4. The 8259A interrupt controller vector table.

2.4. MASKING INTERRUPT REQUEST LEVEL 0

The PC has a timer counter mounted on the base system board which is used as a general system timer. The output of the timer counter is tied to interrupt level 0, the highest maskable interrupt level.

The timer is set up by the PC-BIOS to generate an interrupt request on level 0 every 54.936 millisecond. This interrupt request is masked by the Scrambling/Descrambling algorithm, since it adds a delay which affects the operation of the algorithm.

The interrupt is masked by outputting a hex value of BD to the port address 21 Hex, and re-enabled by outputting a hex value of BC to the same port address.

2.5. IBM PC SYSTEM DELAY

When running the Scrambling/Descrambling algorithm on the IBM PC, full speed of the microprocessor could not be utilized.

One reason is that in the PC design a wait cycle is inserted in every bus cycle.

This wait cycle reduces the speed of the microprocessor by about 20%.

Also, the refreshing of dynamic RAM of the IBM PC slows down the processor by about 7%.

CHAPTER 3

TABLE GENERATION

3.1 PSEUDO-RANDOM NUMBER GENERATORS (PRNG)

PRN generators are one of the most important cryptographic tools. These algorithms not only generate sequences of keys where the numbers are randomly distributed between 1 and some maximum N, but also - if set up appropriately- generate all N of these numbers in a fashion which "looks random" before they start their period over again.

One good congruential generator is the linear congruential PRN generator. This generator produces a sequence of pseudo-random numbers T_1, T_2, T_3, ... , T_N, ... , using the relation:

$$T_{i+1} = (A*T_i+B) \bmod N \qquad (3.1)$$

where A and B are constants, and T_0 is an initial value chosen as the seed number.

Equation (3.1) generates pseudo-random numbers of a certain period of repetition, depending on the values chosen for A and B.

3.2 MAXIMIZING THE LENGTH OF THE KEY SEQUENCE - CHOOSING A AND B

Pseudo-random generators have a maximum period N before the sequence starts to repeat. We would like to choose A and B in such a way that this period N (and thus the length of the key string) is maximized. It is shown in Knuth [6] that the sequence

$$T_{i+1} = (A*T_i+B) \bmod N$$

where $N = 2^K$ and K is an integer > 2, has length N if and only if

$$\left.\begin{array}{ll} (1) & \text{B is odd} \\ (2) & A \bmod 4 = 1 \end{array}\right\} \qquad (3.2)$$

3.3 KEY TABLE SUBROUTINE (KTS)

The block permutation method divides the buffer of the digital samples to be scrambled into blocks and permutes these blocks according to the key table.

The key table routine uses Eqn. (3.1) to generate a table of length N of starting addresses of the blocks in a scrambled order

$$T_{i+1} = (T_i*A+B) \bmod N$$

N is the number of blocks in the buffer to be permuted. The set A, B and T_0 is the input key for permutation.

The permutation routine reads input buffer blocks sequentially and writes them according to the addresses in the key table.

```
; GENERATION OF KEY TABLE
; FL IS THE NUMBER OF BLOCKS IN A FRAME
; BLL IS THE NUMBER OF SAMPLES IN A BLOCK

            MOV DI,OFFSET KT1  ; LOAD TABLE ADDRESS IN DI
            MOV AX,K1
            MOV CX,FL          ; LOAD FRAME LENGTH IN CX
RAND1   :   MUL A1             ; AX=K1*A1
ADD     :   AX,B1              ; AX=AX+B1
RED     :   CMP AX,FL          ; IF AX ≥ NUMBER OF BLOCKS IN A FRAME
            JAE RAND1          ; DO IT AGAIN
CONTINUE :  MOV AH,0
            MOV DX,AX
            MOV BL,BLL         ; LOAD BLOCK LENGTH IN BL REGISTER
            MUL BL             ; GENERATE BLOCK ADDRESS
            STOSW              ; STORE IT IN THE TABLE
            MOV AX,DX
            LOOP RAND1         ; IF ALL BLOCK ADDRESSES GENERATED
            RET                ; RETURN TO THE MAIN PROGRAM
```

where A1, B1 and K1 are part of the input key.

3.4 AUTOMATIC KEY CHANGE ROUTINE (AKCR)

AKCR enables the system to periodically change the permutation pattern without the user having to change the key setting.

The routine does this by generating pseudo-random numbers to be used as the starting point of the permutation key table.

The following equation is used to generate the pseudorandom numbers:

$$T_{i+1} = (A*T_i+B) \bmod 2^{16} \qquad (3.3)$$

The values of A and B must satisfy condition (3.2) to give a maximum repetition period.

So, if the condition (3.2) is satisfied, the sequence of numbers generated by Eqn. (3.3) is 2^{16} long for the fixeds value of A and B.

For the different values of A or B, a different sequence of numbers will be generated by Eqn. (3.3). So, if we keep changing A and B after each sequence of 2^{16} numbers generated, a longer sequence will be generated. The length of this sequence is

2^{16}x[values of A that satisfy (3.2) and $< 2^{16}$] x [values of B that satisfy (3.2) and $< 2^{16}$] $= 2^{16} \times 2^{14} \times 2^{15} = 2^{45}$.

CHAPTER 4

BLOCK PERMUTATION METHOD I

4.1. INTRODUCTION TO BLOCK PERMUTATION METHOD

Figure 4.1 illustrates the principle of Block Permutation method encoding. A portion of the speech samples is first loaded into memory of the scrambler. Upon completion of the storage phase, the speech signal is read out as a sequence of time-permuted segments (or blocks) of the original speech interval.

At the receiving end, the inverse procedure takes place and the segments arriving in a scrambled sequence are rearranged to yield the original speech signal.

The duration of the stored speech is typically 500 ms and the duration of the segments is about 30 ms.

The permutation order is determined by a preselected pseudorandom (PR) sequence. Consequently, code synchronization is required between the transmit and receive units.

4.2. SCRAMBLER FLOWCHART

Figure 4.2 shows a flowchart for a general scrambler. All

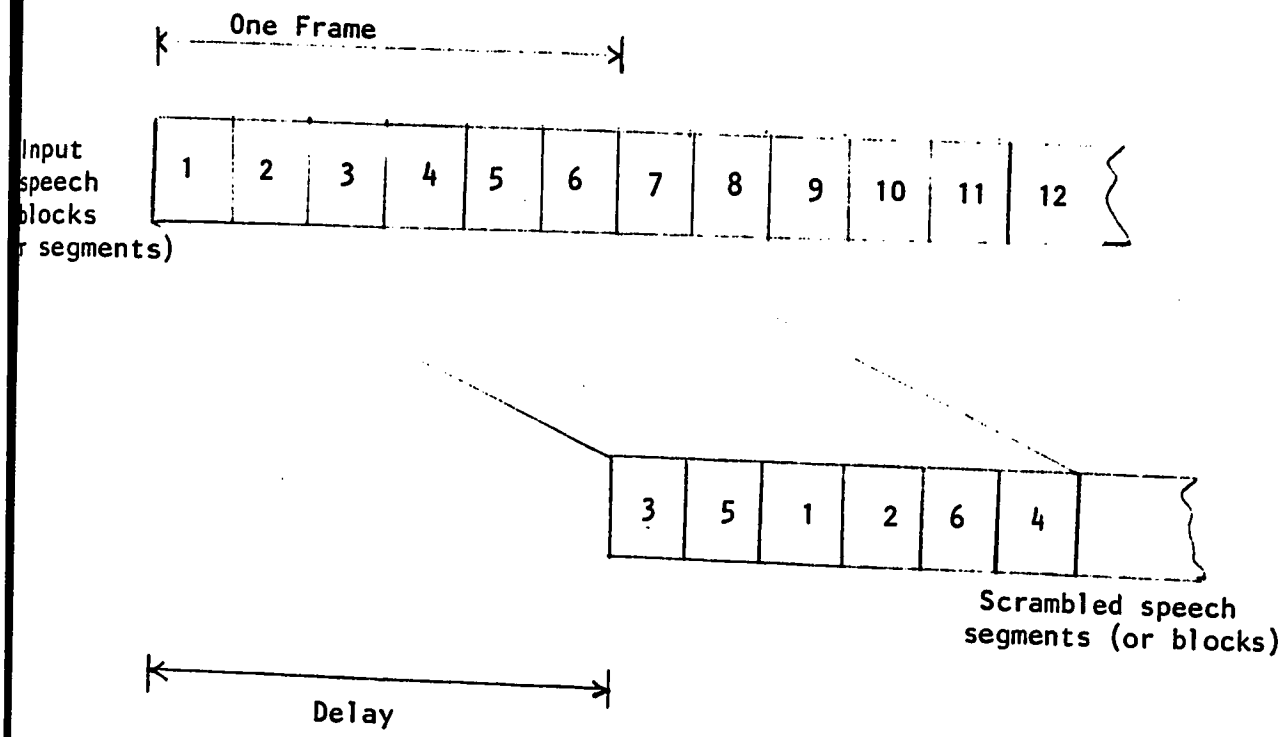


Figure 4.1. Block permutation.

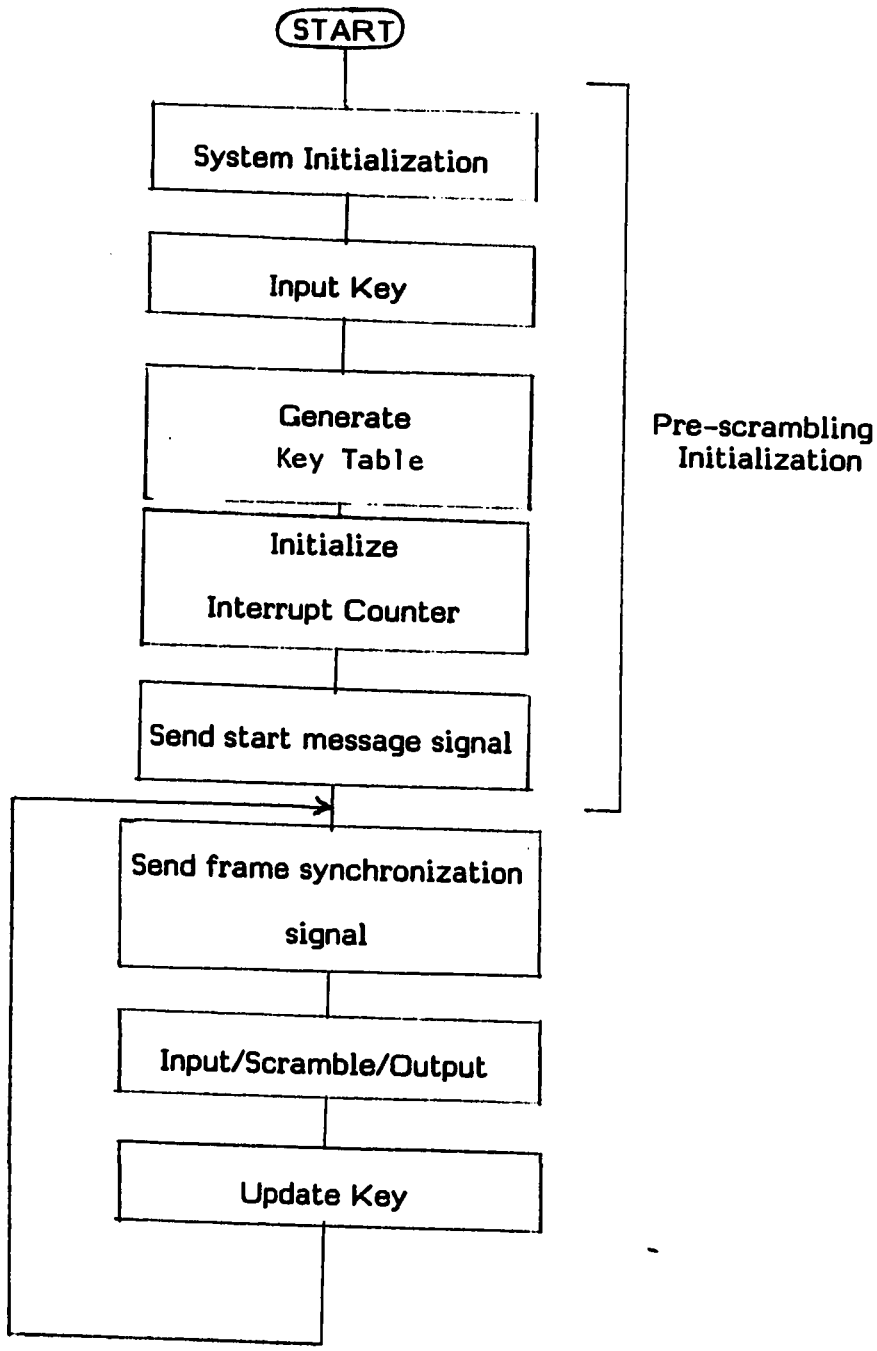


Figure 4.2. Flowchart of a scrambler.

scrambling methods described in the following chapters are using this general flowchart. Only scrambling algorithms will be discussed.

4.3. BLOCK PERMUTATION METHOD I

This algorithm uses two buffers, one for writing and one for reading. The elements which are to be scrambled are dealt within frames each containing N elements. While the elements in one buffer are being read out in scrambled order, the other buffer is being filled with incoming elements. At the end of each frame the two buffers change roles, the one that was being filled is emptied and vice versa. To be able to make any rearrangement involving N elements it is necessary to have enough storage for 2N elements.

Figure 4.3 shows a flowchart for implementation of the block permutation method I algorithm. The algorithm uses the BP base pointer for reading and the SI index register for writing.

At the beginning of each block the SI register is loaded with the address of the next block to be moved out.

The addresses of the blocks are stored in the Key Table in a permuted order.

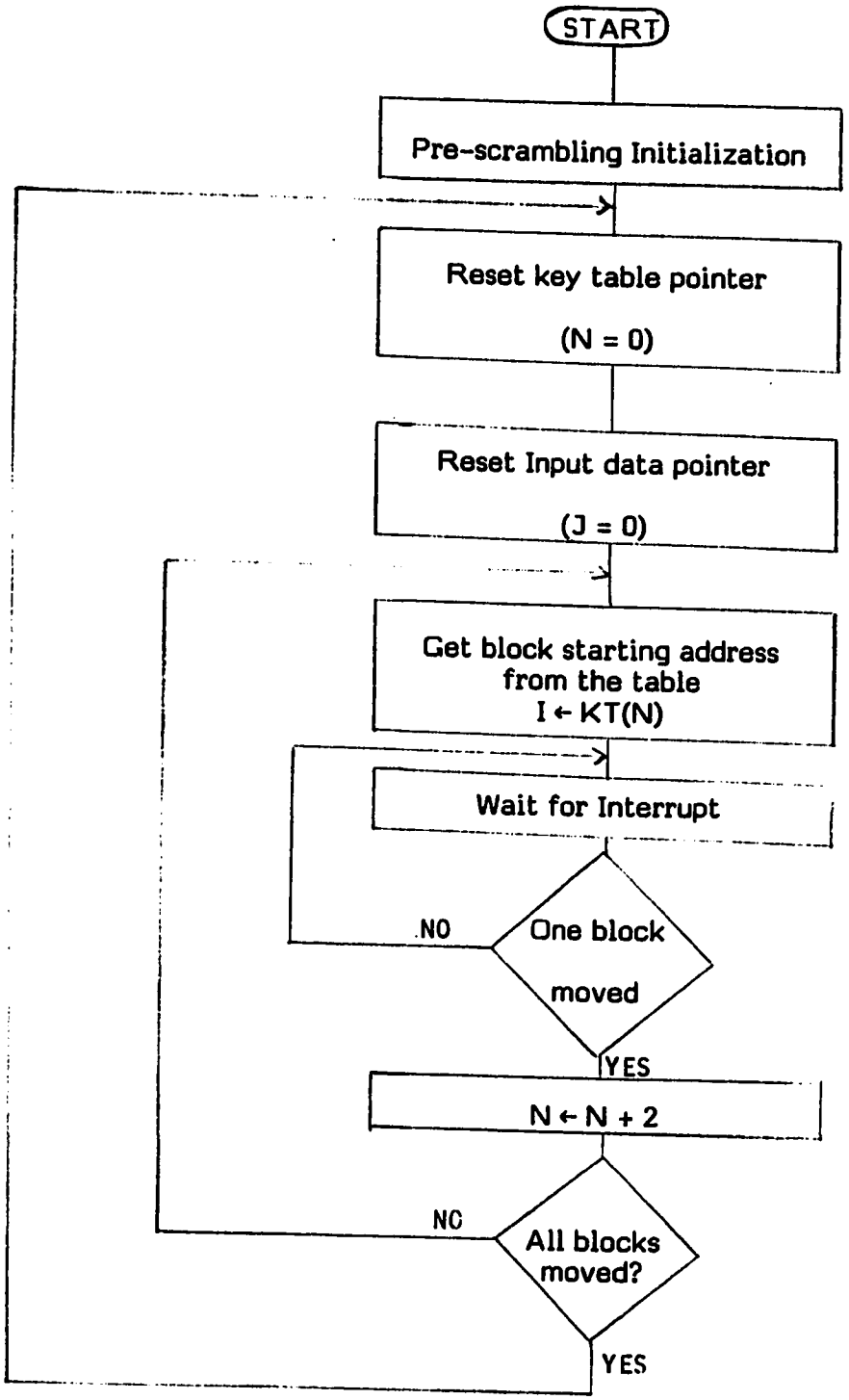


Figure 4.3. A flowchart of block permutation method I.

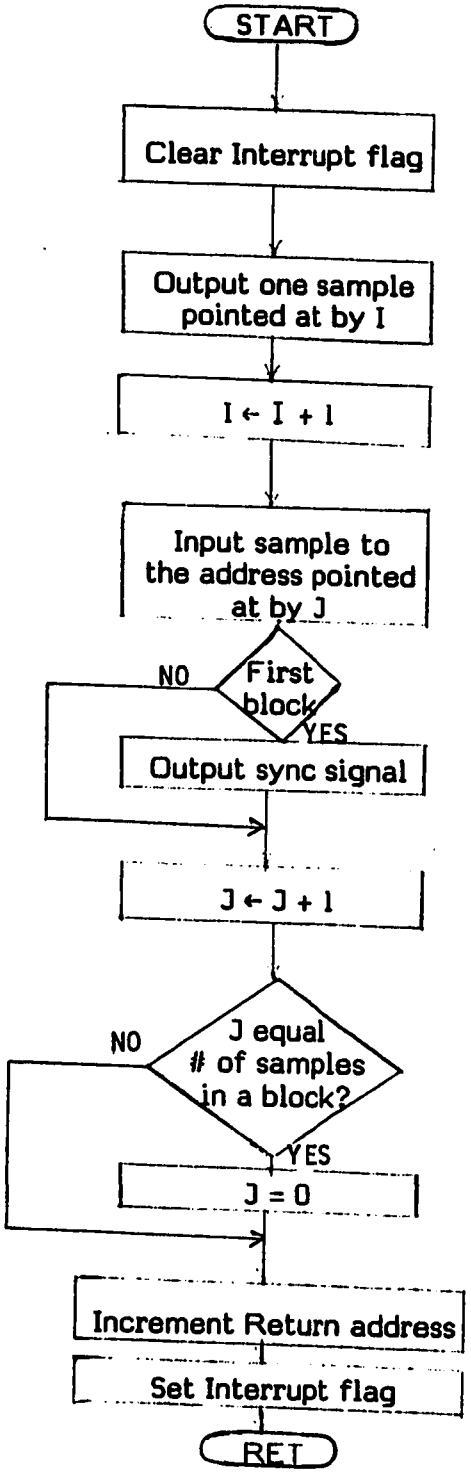


Figure 4.4. ISR Flowchart for block permutation method I.

This is the BLOCK PERMUTATION MAIN ROUTINE

```
; WB      is the work buffer
; L       is the number of samples in a frame
; BLL     is the number of samples in a block
; KT1     is the table starting address
; FFL     is four times the number of blocks in a frame
START:  MOV BX,0                ;  CLEAR KEY TABLE POINTER
        MOV BP,OFFSET WB
        ADD BP,L                ;  SET READING INDEX TO THE MIDDLE OF
                                ;  THE BUFFER
BACK:   MOV SI,OFFSET WB        ;  LOAD BUYFFER STARTING ADDRESS
                                ;  INTO WRITING INDEX
        MOV CX,BLL              ;  LOAD BLOCK LENGTH INTO SAMPLE
                                ;  COUNTER
        ADD SI,KT1[BX]          ;  GENERATE ADDRESS OF BLOCK TO BE
                                ;  WRITTEN
REPT:   JMP REPT                ;  WAIT FOR INTERRUPT
        LOOP REPT               ;  IF NOT ALL SAMPLES IN THE BLOCK
                                ;  ARE WRITTEN STAY IN THE WAIT LOOP
        ADD BX,2                ;  INCREMENT KEY TABLE POINTER
        CMP BX,FFL              ;  END OF BUFFER
        JNZ BACK                ;  IF NO, GO BACK
        JMP START
```

4.4 INTERRUPT SERVICE ROUTINE (ISR)

The interrupt service routine inputs one sample from ADC port and stores it in the buffer addressed by the BP index register (Fig. 4.4).

It also outputs one sample to the DAC port from the buffer addressed by the SI index register. So, the samples are inputted and outputted at the same rate by the ISR.

At the beginning of each frame ISR of the scrambler sends a synchronization signal to the descrambler.

The ISR increments the return address to skip the wait loop in the main routine.

INTERRUPT SERVICE ROUTINE

```
IOR PROC NEAR
CLI              ;  CLEAR INTERRUPT FLAG
PUSH AX
MOV AL,OB[BP]    ;  SEND DATA SAMPLE FROM OUTPUT BUFFER
OUT 253,AL       ;  TO THE DAC
IN AL,241        ;  INPUT SAMPLE FROM ADC
MOV INB[BP],AL   ;  STORE IT IN THE INPUT BUFFER
CMP BP,01        ;  TEST FOR THE BEGINNING OF NEW FRAME
JA GO
```

```
        MOV AX,BP
        NOT AX
        AND AL,01
        OUT 254,AL   ;  SEND SYNCHRONIZATION SIGNAL
GO    : INC BP       ;  INCREMENT I/O COUNTER

        CMP BP,L
        JB CON
        MOV BP,0
CON   : MOV AL,32    ;  END OF
        OUT 32,AL    ;  INTERRUPT (EOI)
        OUT 242,AL   ;  CLEAR INTERRUPT FLIP FLOP ON
                        THE LAB TENDER BOARD
        POP AX
        STI
        IRET
        IOR ENDP
```

4.5. METHOD'S ADVANTAGES

- The RAM memory used by this permutation method for buffering is half of that used by permutation method II.

For FL number of blocks in a frame and a block length of BLL, the RAM buffer needed is

2 * FL * BLL

- Speech delay of this method is minimum with respect to other permutation methods.

The delay (τ) that the scrambler or descrambler algorithm introduces is equal to the storage time of one buffer, which is

$$\tau = \text{block length} \times \text{number of blocks in a frame} \times \frac{1}{\text{sampling rate}} \qquad (4.1)$$

For example, the delay for a block length of 128 samples, 32 blocks in a frame and a sampling rate of 8 KHz is,

$$\tau = 128 \times 32 \times \frac{1}{8000\ (\text{Hz})} = 0.512 \text{ seconds.}$$

4.6. METHOD'S DISADVANTAGES

The ability of this permutation method to combine with other scrambling method is limited, due to the fact that scrambling is done between two incoming samples. So, the time available for scrambling is equal to one period of the sampling frequency. For example, when a 8 KHz sampling rate is used, the time available for scrambling is 125 µs only.

CHAPTER 5

BLOCK PERMUTATION METHOD II

5.1. ALGORITHM DESCRIPTION

In this method the algorithm uses four buffers, one for writing, one for reading and two for various manipulations. The input data is loaded in the input buffer (INB) and the output data is taken from the output buffer (OB) periodically by the interrupt service routine (ISR).

Figure 5.1 shows a flowchart for the scrambling routine.

As the input buffer (INB) is being filled and the output buffer (OB) is being emptied by ISR, the first part of the algorithm moves the new samples in the INB to work buffer 1 (WB1) and fills OB with samples to be sent to the DAC from work buffer 2 (WB2), until WB1 is full and WB2 is emptied. Then the data in WB1 are permuted and moved to WB2 as shown in Fig. 5.3.

The BP register is the input/output buffer-pointer. The key table (KT) contains the addresses of the blocks to be permuted in a scrambled order.

The BX register is used as an index to the Key Table. Since each address in the key table takes two bytes, the BX register is incremented by two.

FL is the number of blocks in a frame.

For the descrambler algorithm the operation is similar except that the data are inversely permuted in work buffer 1 and stored in work buffer 2.

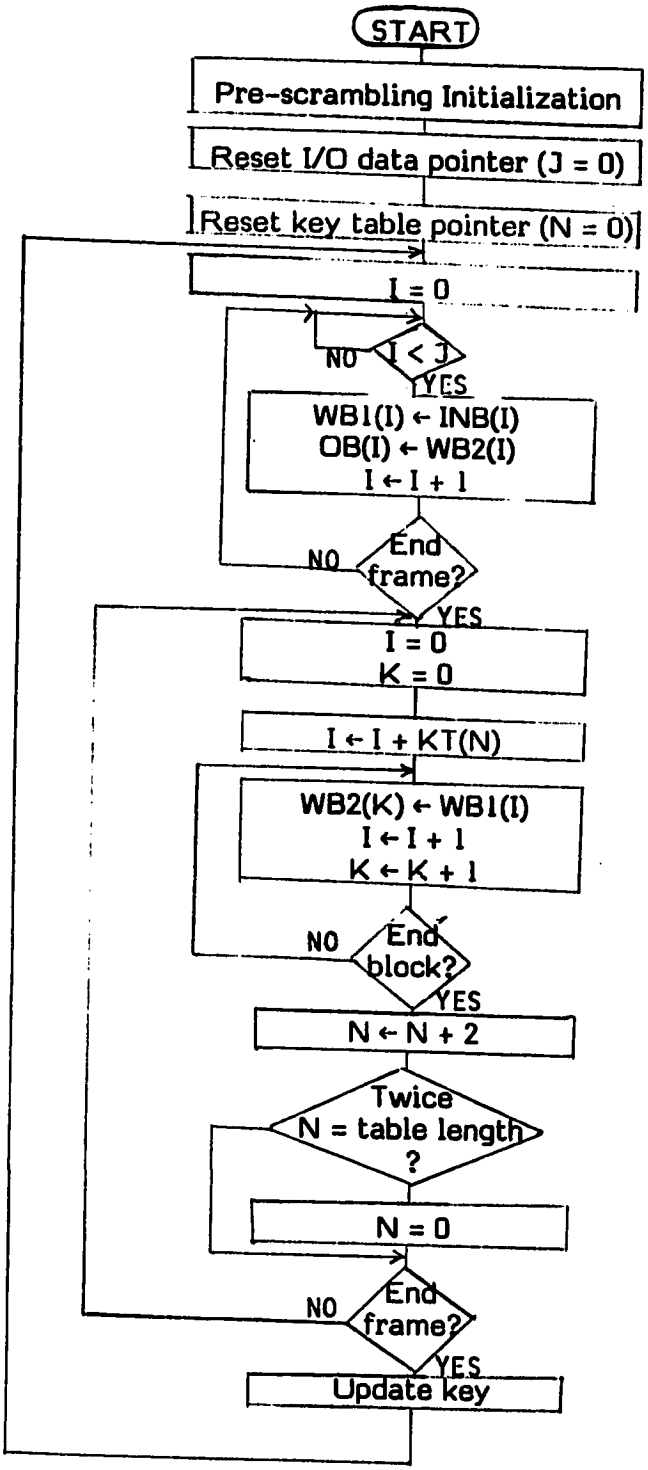


Figure 5.1. A flowchart of block permutation method II.

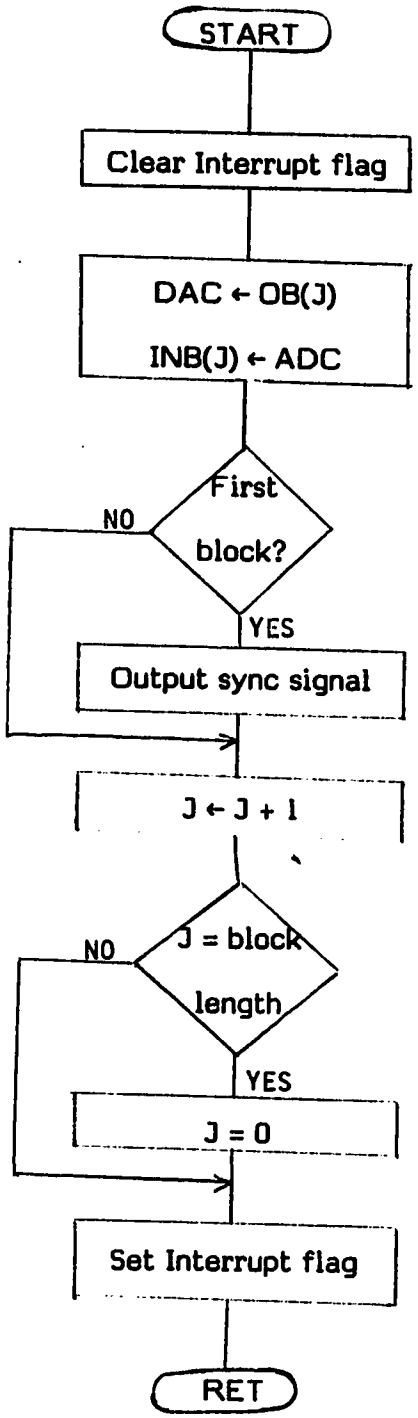


Figure 5.2. ISR flowchart for block permutation method II.

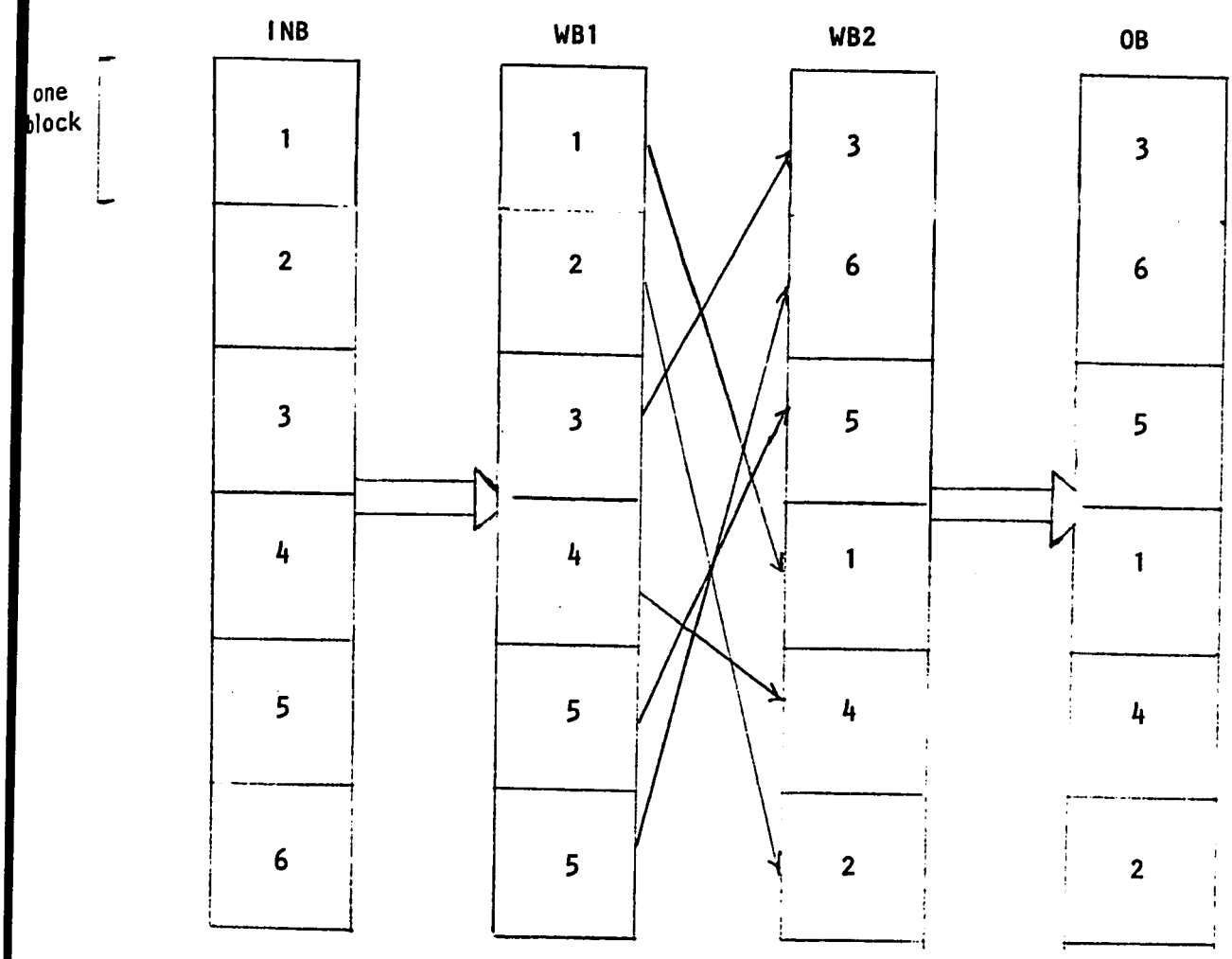


Figure 5.3. Illustration of block permutation method II.

THIS IS THE MAIN PROGRAM FOR BLOCK PERMUTATION METHOD II.

```
;   L is the number of samples in a frame
;   FL is the number of blocks in a frame
;   TFL = 2*FL
;   LM1 = L-1
;   Move input buffer to work buffer # 1 and work buffer # 2 to output buffer
START   : MOV SI,0          ;  CLEAR INDEX
TEST    : CMP BP,02
          JA RR
          CMP SI,LM1
          JZ DO
RR      : CMP BP,SI         ;  NEW INPUT SAMPLE
          JBE TEST          ;  IF NOT CHECK AGAIN
DO      : MOV AL,WB2[SI]    ;  MOVE ONE SAMPLE OF WORK BUFFER
                            ;  # 2 TO THE OUTPUT BUFFER
          MOV AL,INB[SI]    ;  MOVE ONE SAMPLE OF INPUT BUFFER
                            ;  TO WORK BUFFER # 1
          INC SI            ;  INCREMENT INDEX
          CMP SI,L          ;  ALL SAMPLES MOVED
          JNZ TEST          ;  IF NOT GO BACK
                            ;  PERMUTE WORK BUFFER # 1 AND MOVE
                            ;  IT TO WORK BUFFER # 2
```

```
        MOV DI,OFFSET WB2   ; SET WRITING INDEX TO WORK BUFFER #2
        MOV CNT,0           ; CLEAR COUNTER
BACK:   MOV SI,00           ; RESET READING INDEX
        MOV CX,BLL          ; LOAD BLOCK LENGTH INTO CX
        ADD SI,KT1[BX]      ; MODIFY READING INDEX TO THE BLOCK POINTER
                            ; AT BY THE KEY TABLE
REPT:   MOV AL,WB1[SI]      ; LOAD SAMPLE TO THE AL REGISTER
        STOSB               ; STORE IT IN WB #2
        INC SI
        LOOP REPT           ; IF ONE BLOCK MOVED CONTINUE
        ADD BX,2            ; INCRMENT KEY TABLE POINTER
        INC CNT             ; INCRMENT BLOCK COUNTER
        CMP BX, TFL         ; DEFINE TFL
        JNZ CONU
        MOV BX,0
CONU:   CMP CNT,FL          ; ALL BLOCKS MOVED
        JNZ BACK            ; IF NOT GO TO BACK
        CALL AKCR           ; SELECT KEY TABLE STARTING POINT
        JMP START
```

5.2 METHOD'S ADVANTAGES

- This permutation method utilizes only part of the time of the microprocessor, so it is possible to combine another method with it.

For example, when the number of samples in a frame is 4096, and the clock cycle time is 0.21 μs, the time available for scrambling (ST) one frame is

$$ST = \frac{\text{\# of sample in a frame}}{\text{sampling frequency}}$$

for a sampling rate of 8 KHz, the available time is

$$ST = \frac{4096}{8} = 512 \text{ ms.}$$

The permutation routine uses only 130 ms of the time available.

- The availability of microprocessor time makes it possible to use the automatic key change routine which generates a very long PRN period, like the one described in chapter 3.

5.3 METHOD'S DISADVANTAGES

- This method uses more RAM memory than the previous permutation method, since it uses four buffers for permutation. The buffer length is equal to the number of samples in a frame.
- The scrambler that uses this permutation method introduces a speech delay of twice the storage time of one buffer. For example, when the sampling rate is 8 KHz, and the number of samples in a frame is 4096, the scrambler speech delay (τ) is

$$\tau = 4096 \times \frac{1}{8000} \times 2 = 1.24 \text{ second}$$

which is twice the delay that permutation method 1 introduces.

5.4 CHOICE OF FRAME LENGTH (FL) AND BLOCK LENGTH (BLL)

The number of blocks in a frame can range from as few as 16 blocks to a large number limited only by the memory size, and the delay that the algorithm introduces.

The security of the scrambled speech increases with the increase of the number of blocks in a frame with a fixed block length.

The block length's lower limit is the distortion that the algorithm introduces, due to the discontinuities introduced by the scrambler between the blocks.

As the number of samples in a block reduces, the distortion in the recovered speech increases.

The upper limit of the block length is the number of samples that makes one syllable (around 36 ms).

For normal telephone conversation the acceptable delay is 0.5 second for the scrambler, which means that the maximum number of samples in a frame is 4096, when a 8 KHz sampling rate is used.

A good combination of frame length and block length for this application are:

BLL	FL
64	64
128	32
256	16

5.5 EXPERIMENTAL EVALUATION OF THE BLOCK PERMUTATION SCRAMBLING METHOD

The experimental evaluation on block permutation was performed using the set-up shown in Fig. 2.1.

The algorithm was used for sampling frequencies of 6.25 KHz, 7.1 KHz and 8 KHz.

The recovered speech was clearer as the sampling rate increases, but the speech was found still clear and understandable at a 6.25 KHz sampling rate. Thus, it is possible to use this sampling rate in some applications to decrease the RAM memory requirements.

The algorithm was tested with the duration of the permuted segments set to 2 ms, 8 ms, 16 ms and 32 ms, which corresponds to 16, 64, 128 and 256 samples respectively (if 8 KHz sampling rate is used).

The scrambled speech was intelligible when a duration of 2 ms was used for the permuted segments. But as the duration increased to 8 ms, 16 ms or 30 ms the scrambled speech became less intelligible and finally completely unintelligible. The duration of the permuted segments should not exceed the duration of a syllable (around 36 ms) except when the permutation is combined with another method which destroys the intelligibility of the speech.

(a)

(b)

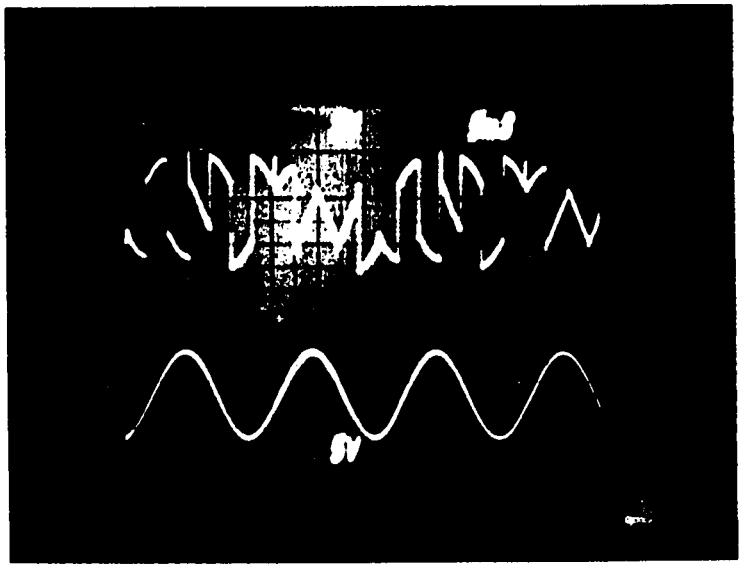


Figure 5.4. Block permutation scrambling.

(a) The scrambler output.

(b) The scrambler input.

CHAPTER 6

REVERSED TIME SEGMENTATION METHOD

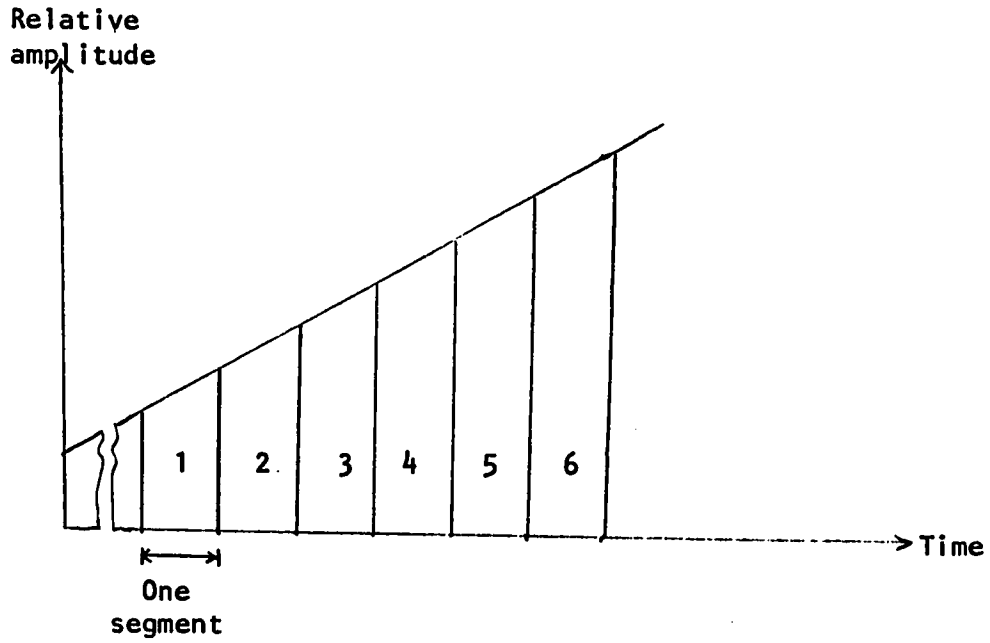
6.1. METHOD DESCRIPTION

Reversed Time Segmentation (RTS) is a time-domain analog speech privacy technique which adapts well to an implementation with a microprocessor. Also, it is one of the most efficient methods for destroying intelligibility of speech. In this method speech segments are reversed in time. Figure 6.1 shows qualitatively the speech signal scrambling pattern obtained with such a reversed time segment encoding method.

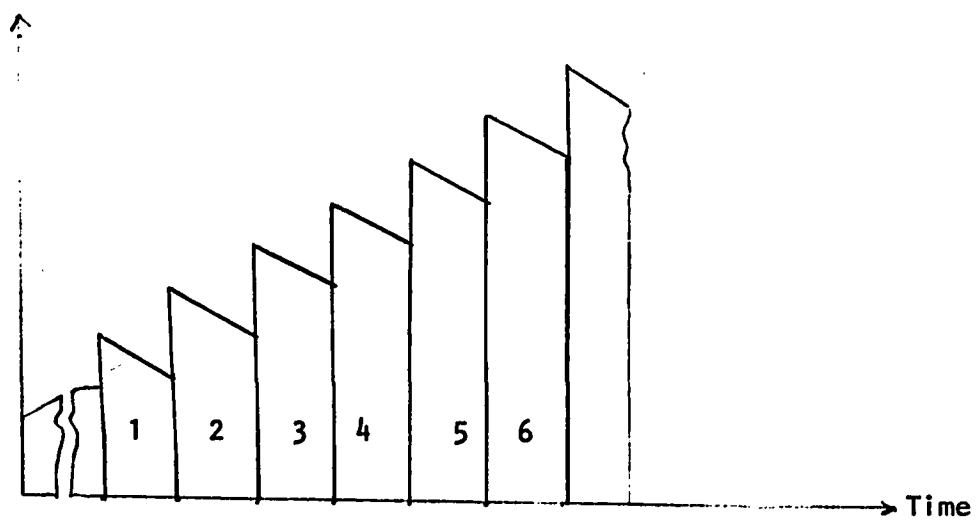
Part (a) of the figure shows a portion of a speech signal which is progressing in a normal direction, with its segments being generated in the sequence 1, 2, 3, 4, 5, 6. Part (b) of the same figure shows how the direction of each of the segments has been reversed, yet the order of their delivery has not been altered.

Depending on the encoding requirements, the duration of the segments ranges from 50 to 400 ms, with longer segments providing better scrambling at the expense of increased encoding delay. Another important feature of the RTS is that the spectral characteristics of the scrambled signals differ very little from those of the original signal.

The reversed segmentation method can be implemented with only one buffer.



(a) Original speech signal.



(b) Encoded speech signal.

Figure 6.1. RTS method.

On the first scan through memory, the buffer is loaded by the output of the ADC. During this time no signal readout is performed. As soon as the buffer is loaded, the signal readout can start. Immediately upon readout of each memory byte, the address is filled with a new sample of a digitized speech signal. Thus, when the readout of the inverted speech is completed, the buffer is full and ready for the next readout of the time-inverted speech. In this manner, the alternate readout scans of the buffer produce the required time inversion of speech signals. Since the RTS method is not a very secure method by itself we choose to combine it with the block permutation method.

6.2. THE REVERSED SEGMENTATION/ PERMUTATION ALGORITHM

This method combines the block permutation method II and the reversed segmentation method.

The algorithm either reverses a segment (N blocks) of data or not depending on the input key. Every bit of the key corresponds to a segment in the frame of data samples. The algorithm reverses the segments that correspond to a binary 1 in the input key. It does not reverse the segments that correspond to a binary 0 in the input key.

If the segment is to be reversed, the direction flag is set, which causes the destination index register to decrement as data is moved from WB1 to WB2.

The direction flag is cleared if the segment is not to be reversed.

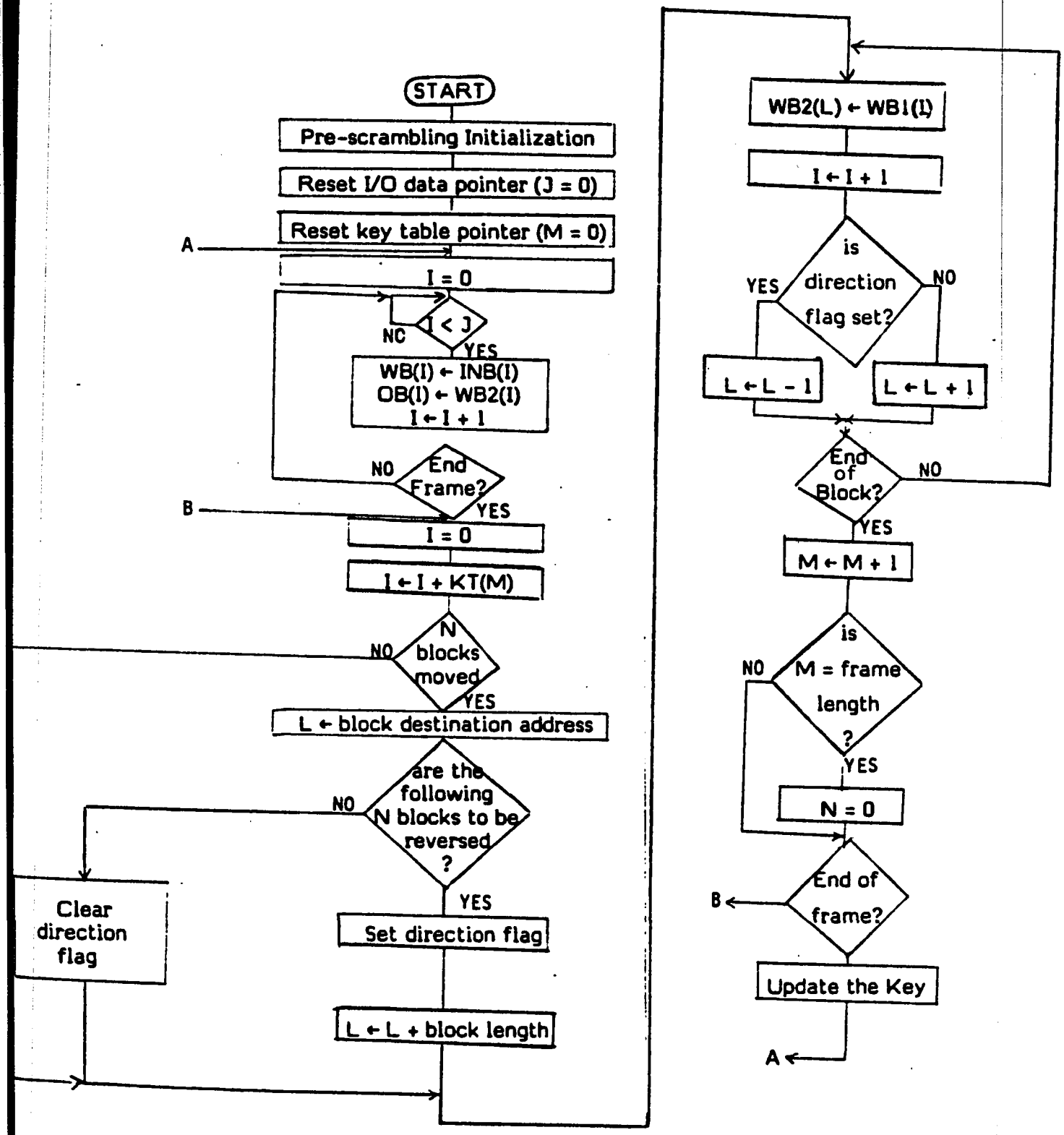


Figure 6.2. A flowchart of reverse segmentation/permutation method.

6.3. MAIN ROUTINE

The following is the listing of the Main Program for Permutation and Reversed Segmentation.

```
;   N is the number of blocks to be reversed
;   K3 is the key for segments reversing
;   BLL is the block length
;   FL is the number of blocks in a frame
;   TFL = 2 * FL
;   SL2 = N * BLL
;   SLM1 = SL2-1
;   Scramble work buffer # 1 and move it to work buffer # 2
        MOV CNT, 0        ; CLEAR COUNTER
        MOV IOC, 0        ; CLEAR DESTINATION BLOCK STARTING ADDRESS REG.
        MOV DL, N         ; LOAD NUMBER OF BLOCKS TO REVERSED  IN DL REG.
        MOV DH, K3        ; LOAD REVERSING KEY IN DH REG.
BACK:   MOV SI, 00
        MOV CX,BLL        ; LOAD BLOCK LENGTH INTO COUNTER
        ADD SI,KT1[XB]    ; MODIFY READING INDEX TO THE BLOCK
                          ; POINTED AT BY THE KEY TABLE
        CMP DL,N          ; N BLOCKS MOVED
        JB REPT           ; IF NO, JUMP DESTINATION ADDRESS MODIFICATION
                          ; DESTINATION ADDRESS MODIFICATION
```

```
        MOV DL, 0
        MOV DI,OFFSET WB2   ; LOAD BUFFER 2 ADDRESS IN DEST. INDEX REG
        ADD DI, IOC         ; ADD DISPLACEMENT
        ADD IOC, SL2
        RCR DH,1            ; CHECK IF SEGMENT (N BLOCKS) TO BE REVERSED
        JNB ON              ; IF NOT JUMPT TO ON
        STD                 ; SET AUTO DECREMENT MODE
        ADD DI, SLM1        ; CHANGE INDEX TO STORE FROM BOTTOM-UP
        JMP REPT
ON:     CLD                 ; SET AUTO INCREMENT MODE
                            ; START PERMUTATION
REPT:   MOV AL,WB1[SI]
        STOSB               ; DI IS INCREMENTED OR DECREMENTED
        INC SI
        LOOP REPT
        INCL DL
        ADD BX,2
        INC CNT
        CMP BX, TFL
        JNZ CONU
        MOV BX, 0
CONU:   CMP CNT, FL
        JNZ BACK
        CLD                 ; SET AUTO INCREMENT MODE
        CALL AKCR           ; CALL AUTOMATIC KEY CHANGE ROUTINE
        JMP START
```

6.4. METHOD'S DISADVANTAGE

Implementation of the Reversed Time Segmentation alone doesn't provide a very secure speech communication against analytical attack. This is due to the fact that in this method only two things are needed to be able to retrieve the original message; these are the length of the reversed time segments and the starting point of the segment.

6.5 METHOD'S ADVANTAGES

- The RTS method destroys the intelligibility of the speech.

- Only one RAM memory buffer is needed for the implementation of this method.

- The delay that the RTS method introduces is equal to that of the reversed segment, which ranges from 50 ms to 400 ms.

- RTS could be combined with other methods, like the block permutation method.

- The delay of the combined RTS/permutation method is equal to the longest delay of the two separate methods.

- The recovered voice quality of the RTS method is very good.

6.6 EXPERIMENTAL RESULTS

The algorithm was tested using the set up shown in Fig. 2.1.

A reversed segment duration of 8 ms and 16 ms was tried. It was found that the scrambled speech was intelligible. But as we increased the segment duration to 64 ms, the intelligibility of the scrambled speech reduced to practically zero.

So, for longer segments this method provides better scrambling.

(a)

(b)

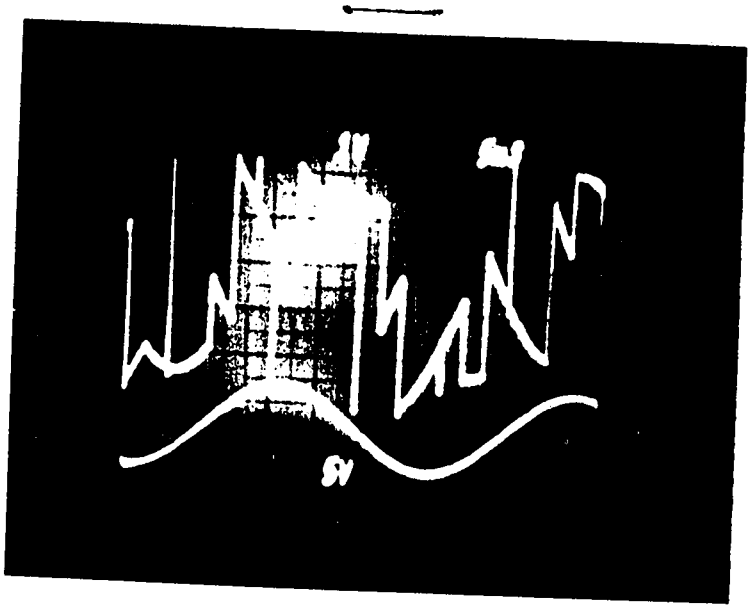


Figure 6.3. (a) Scrambled sinewave using the reverse segmentation method, when all segments are reversed.

(b) Input sine wave.

(a)

(b)

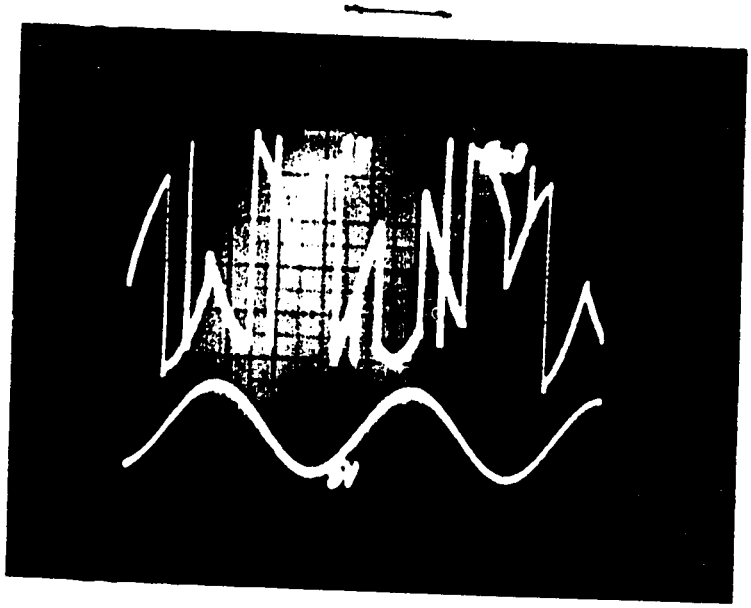


Figure 6.4. Combination of block permutation and reverse segmentation.

(a) The scrambler output.

(b) The scrambler input.

CHAPTER 7

AMPLITUDE MODIFICATION METHOD

7.1 DESCRIPTION OF THE METHOD

The amplitude modification technique performs scrambling by a modulo-n addition of pseudorandom numbers to the speech samples.

As each speech sample is read from the ADC, the microprocessor stores it in one of its registers. Then a pseudorandom number is added to it, and the modulo (256) equivalent of this sum is obtained. If X_n, Z_n and Y_n denote a speech sample, pseudorandom number and modulo (256) equivalent sum, respectively, the result is represented as

$$Y_n = (X_n + Z_n) \bmod (256) \tag{7.1}$$

The descrambler on the receiving end recovers the speech by a modulo-n subtraction of the pseudorandom numbers from the scrambled speech samples

$$X_n = (Y_n - Z_n) \bmod (256) \tag{7.2}$$

7.2 ALGORITHM DESCRIPTION

Figure 7.1 shows the flowchart of the scrambling program using the amplitude modification method.

The secret key is inputted from the keyboard, and used by the algorithm as the starting point of the pseudo-random sequence.

The key change routine described in chapter 3 is used for updating the key.

The interrupt service routine performs the inputting, scrambling and outputting of the speech samples.

7.3 PRACTICAL DIFFICULTIES

The practicality of maintaining the integrity of the sequence $\{Y_n\}$ during transmission is, of course the major question affecting the feasibility of this method. The channel distortion can substantially alter the transmitted sequence, and because of the modulo operation in the descrambler, increase the noise on the recovered samples.

To minimize the effects of distortion, it is necessary to (1) transmit analog samples at the slowest rate possible, and (2) provide adequate channel equalization at the receiver.

The computer will produce a carry-out if the sum of the key sequence to the speech samples exceeds 255.

Speech samples in Eqn. (7.1) do not produce many carry-outs when

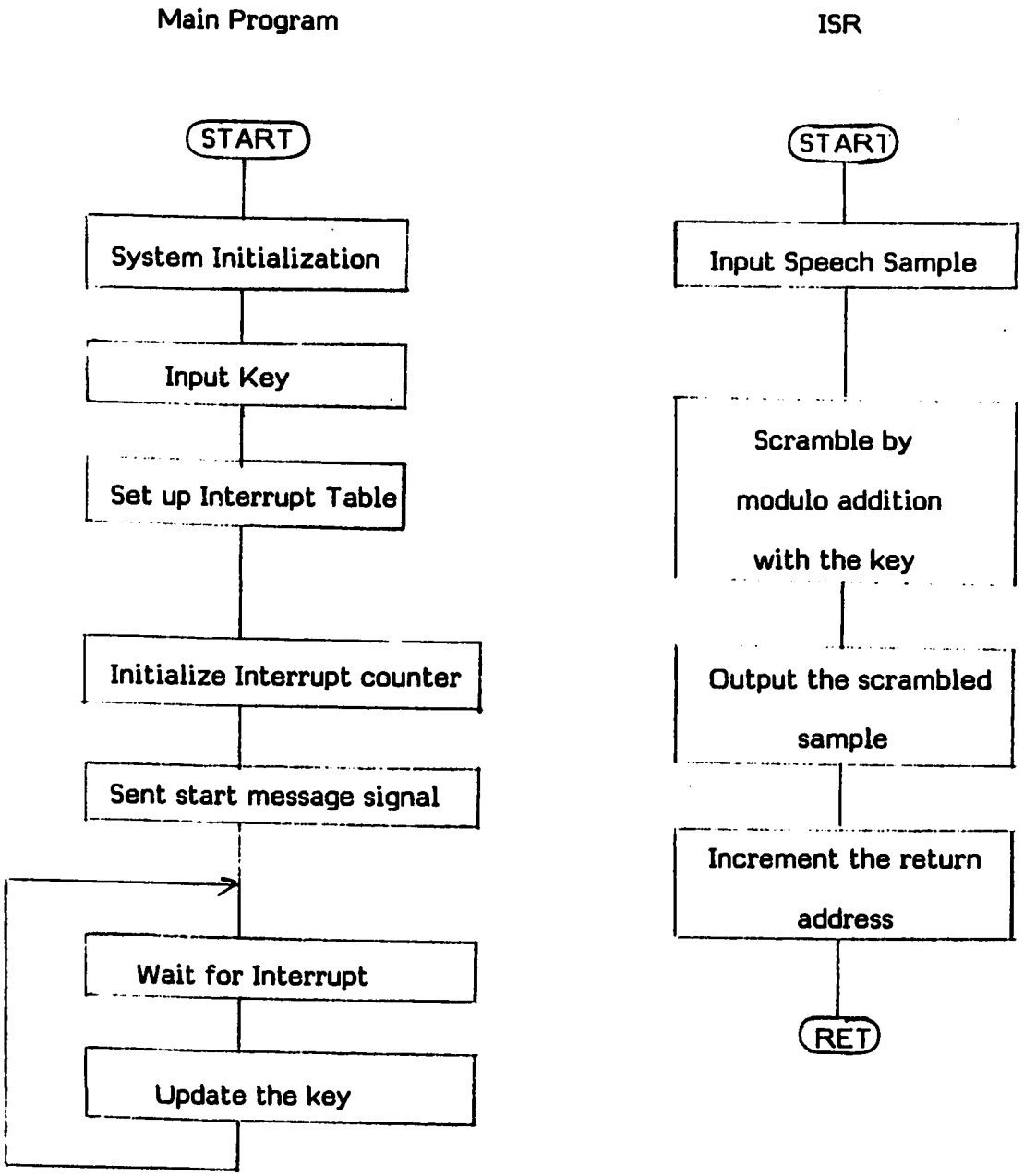


Figure 7.1. A flowchart of amplitude modification method.

added to the key sequence, because of their low average value. Differences among speakers affect the production of carry-outs.

To keep production of carry-outs high and eliminate variations between speakers, it is of value to compand the speech.

At the transmitter, the companding device compresses the speech samples. It does this by boosting the amplitude of the low level speech samples which carry much of the intelligibility. Subsequent noise in a communications sytem will have less masking effect on these important speech components.

At the receiver, the speech is expanded again to its original dynamic range.

7.4 METHOD'S ADVANTAGES AND DISADVANTAGES

One of the advantages of this method is that buffering of speech samples is not needed, because the algorithm scrambles one sample at a time. So, there is no delay of the speech inserted by the scrambler/descrambler system other than the processing delay, which is in the microsecond range.

This is a very highly secured scrambling method, because the scrambled speech spectra are non-speech-like.

The major disadvantage of this method is the distortion on the recovered speech.

The amplitude modification is a non-linear process that produces high-frequency components so that the scrambled speech has a lot of energy

at the high-frequency end of the band where the greatest distortion is likely to occur.

7.5 EXPERIMENTAL RESULTS

The algorithm was tested on the set-up shown in Fig. 2.1 with a small distortion inserted between the scrambler and the descrambler. The recovered speech was clear, but with a background distortion.

So, the small distortion added to the scrambled speech was enhanced by the scrambler.

For example, if the key $Z_N = 45$, speech sample $X_N = 2$, and channel distortion $\eta_N = -3$, the scrambled value is

$$Y_N = (X_N + Z_N) \bmod 256 = (45 + 2) \bmod 256 = 47$$

If the value received by the descrambler is

$$Y'_N = Y_N + \eta_N = 47 - 3 = 44$$

the recovered sample is

$$X'_N = (Y'_N - Z_N) \bmod 256 = (44 - 45) \bmod 256 = 255$$

which is not equal to the original value $X_N = 2$.

CHAPTER 8

SECURITY AND SYNCHRONIZATION

8.1 SECURITY

The degree of security inherent in any technique depends on the difficulty and the resources required to penetrate the system.

8.1.1 Security Measures

The security of a scrambled speech using block permutation method alone or combined with reversed segmentation method is affected by the following factors:

1. Frame length (FL) is the number of blocks in a frame. These blocks are permuted according to the input key. For a fixed block length, the more blocks are in a frame, the more secure the system will be. For example, for FL = 64 blocks, the number of possible permutation is $64! = 1.27 \times 10^{89}$, while for FL = 128 blocks, the number of possible permutation is $128! >> 64!$.

2. A fixed permutation pattern will result in a low security

system. So, if the permutation pattern changes during scrambling the system will be more secure.

3. Code depth is the time required for the code (or permutation pattern) to repeat itself (for a changing permutation pattern). The system with longer code depth will be more secure.

4. The rate of change of the key is another factor that affects the security of the system.
   The greater the rate of change, the more secure the system will be.

5. A system in which all blocks (or segments) are of the same length will not be as secure as one in which the length is variable.

8.1.2 Block Permutation Methods Security

The frame length of the block permutation algorithms (described in chapters 4 and 5) could be chosen to provide sufficient security, but the longer the frame length the more delay will be added by the system. The code depth for block permutation method II is calculated as follows: Code depth = Pseudorandom number sequence length x $\frac{\text{number of samples in a frame}}{\text{sampling rate}}$

For a 1000 samples in a frame and a sampling rate of 8 KHz, the code depth is $2^{45} \times \frac{1000}{8000} = 2^{42}$ seconds.

The rate of change of the permutation pattern in block permutation methods I and II is maximum, since each frame is permuted using different pattern. The block length in method I and II is fixed, but, when implemented, two or more programs could be stored in the ROM, each one with a different block length. The scrambler (or descrambler) will switch from one to the other program to provide a variable block length system.

8.1.3 The factors that effect the security of amplitude modification scrambling are:

1. Code depth.
2. The rate of change of the key.

The algorithm described in chapter 7 for the amplitude modification method has a very high rate of key change since each sample is modified by a different key.

The code depth is

$$\text{pseudorandom number sequence length} \times \frac{1}{\text{sampling rate}}$$

For a sampling rate of 8 KHz, the code depth is

$$2^{45} \times \frac{1}{8000} = 2^{39} \text{ seconds.}$$

8.2. SYNCHRONIZATION

Two basic features are needed in a synchronization system for time-division, reversed segmentation and amplitude modification scramblers.

First, a common frequency reference is needed which will define the rate at which the elements are generated in the scrambler and the descrambler.

Second, a way of starting the descrambling process at exactly the start of the message is necessary so that the code generators run in step, and the elements in the scrambler and the descrambler start at the same instant relative to the start of the message.

CONCLUSION

The microprocessor-based implementation of analog voice privacy equipment provides for good signal processing capability required for achieving a high degree of privacy. Furthermore, with a microprocessor, the signal-handling capability is available at a reasonable cost, small size, and a low level of power consumption. Design updates and improvements can also be easily performed with microprocessor-based equipment by simply changing the ROM unit.

Successful implementation and experimental verification of several microprocessor-based algorithms clearly demonstrate the advantages offered by using a microprocessor for the design of the analog voice privacy device.

REFERENCES

[1] Edward Brunner, "Efficient scrambling techniques for speech signals", Proc. Int. Conf. Comm. Seattle, WA, June 1980, pp. 16.1.1-6.

[2] Hary Katzan, "The Standard Data Encryption Algorithm", Petrocelli Books, Inc. 1977.

[3] Ger-Chih Chou and Lin-Shan Lee, "A new efficient and practical DFT scrambling system for secure speech communication", Proc. Int. Conf. Comm., 1981, pp. E8.5.1 - E8.5.5.

[4] Sergei Udalov, "Microprocessor-based techniques for analog voice privacy", Proc. Int. Conf. Comm. Seattle, WA, June 1980, pp. 16.4.1-5.

[5] Lance Hoffman, "Modern Methods for Computer Security and Privacy", Prentice-Hall, Inc., 1977.

[6] D.E. Knuth, The Art of Computer Programming, Addison-Wesley, Inc. 1971, Vol. 2.

[7] E. Belland and N. Bryg, "Speech signal privacy system based on time manipulation", 1978 Carnahan Conference on Crime Countermeasures, University of Kentucky at Lexington, pp. 37-40, May 1978.

[8] R.C. French, "Speech scrambling and synchronization," Philips Reports, No. 9, pp. 1-115, 1973.

[9] H.P. Hartmann, "Analog scrambling vs. digital scrambling in police telecommunication networks," 1978 Carnahan Conference on Crime Countermeasures, University of Kentucky, Lexington, pp. 47-51, May 1979.

[10] W.N. Butler and D.L. Cohn, "A simplified voice privacy system," 1983 Conference on Crime Countermeasures, University of Kentucky, Lexington, Kentucky, May 1983, pp. 71-77.

[11] A.M. McCalmont, "Measuring security in analog speech communications security devices," Proc. Int. Conf. Comm. Seattle, WA, June 1980, pp. 16.5.1 - 16.5.4.

[12] N. Jayant, B. McDermott, S. Christensen, and A. Quinn, "A comparison of four methods for analog speech encryption," Proc. Int. Conf. Comm., Seattle, WA, June 1980, pp. 16.6.1-5.

[13] N.S. Jayant, R.V. Cox, B.J. Mcdermott, and A.M. Quinn, "Analog scramblers for speech based in time and frequency," The Bell System Technical Journal, Vol. 62, No. 1, January 1983, pp. 25-46.

[14] S.T. Hong and W. Kuebler, "An analysis of time segment permutation methods in analog voice privacy system," 1981 Carnahan Conference on Crime Countermeasures, University of Kentucky, Lexington, Kentucky, May 1981, pp. 167-171.

[15] L.S. Lee, Y.P. Harn, and Y.C. Chen, "A simple sample value scrambler using FFT algorithms for secure voice communications," Proc. Int. Conf. Comm. Seattle, WA, June 1980, pp. 49.4.1 - 4.5.

[16] S.C. Kak and N.S. Jayant, "On speech encryption using waveform scrambling," The Bell System Technical Journal, Vol. 56, No. 5, May 1977, pp. 781-808.

[17] S.P. Morse, The 8086 Primer, Hayden Book Company, Inc. 1980.

[18] L.C. Eggebrecht, Interfacing to the IBM Personal Computer, Howard W. Sams & Co., Inc. 1983.

[19] D.R. McGlynn, Modern Microprocessor System Design, John Wiley & Sons, Inc. 1980.

[20] D. Willen and J. Krantz, 8088 Assembler Language Programming: The IBM PC, Howard W. Sams & Co., Inc. 1983.

[21] The IBM Personal Computer Disk Operating System Manual.

[22] The IBM Personal Computer Macro Assembler Manual.

APPENDIX A

LAB. TENDER BOARD DESCRIPTION AND PROGRAMMING

Switch Use

SW1 and SW3 - I/O port address select.

The base address of the board (the first I/O location occupied) is set with switches SW1 and SW2. Lab Tender occupies 16 consecutive addresses in the I/O space of the PC. Each switch corresponds to one bit in the address as follows:

Address Bit	Switch	Binary Value
A0	Not selected	1
A1	Not selected	2
A2	Not selected	4
A3	Not selected	8
A4	SW3-5	16
A5	SW3-4	32
A6	SW3-3	64
A7	SW3-2	128
A8	SW1-1	256
A9	SW1-2	512
A10	SW1-3	1024
A11	SW1-4	2048
A12	SW1-5	4096
A13	SW1-6	8192
A14	SW1-7	16384
A15	SW1-8	32768

SW3-1, 6, 7 and 8 have no function.

When a switch is 'open' it contributes a value of one to the board address. A switch that is 'closed' contributes a value of 'zero' to the board address. An 'open' switch contributes a value of 'one' at the address bit. For example to address the board to base address 330 Hex (816 Dec) as done by Tecmar, SW1-1,2 and SW3-4,5 should be open and all other switches on SW1 and SW3 should be closed.

Switch	Binary Value
SW1-2	512
SW1-1	256
SW1-5	32
SW1-4	16
	816 = base address

SW3

SW1

To view this as a binary operation (816 decimal = Ø330 Hex = 0000 0011 0011 0000 binary), the starting address would appear as follows:

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A2	A0
0	0	0	0	0	0	1	1	0	0	1	1	0	0	0	0

The switches representing A9, A8, A5, and A4 (represented by binary 1) are 'open' or 'off', so they contribute to the board address.

The base address is now 816 Dec (330 Hex) and will extend up to 831 Dec (33F Hex) since A0, A1, A2 and A3 are not selectable. The base address of 816 Dec is maintained as a reference in the remainder of this manual. Other addresses may be selected in a similar way but care must be taken not to use an address that is already used by some other board or the IBM Personal Computer. Note that the address selected may only be an even multiple of 16 such as 816 Dec (330 Hex), or 832 Dec (340 Hex) and so on.

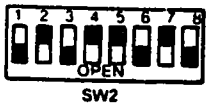
SW2: This set of switches is used to select either single ended or differential mode for the A to D converter. There are only two valid settings.

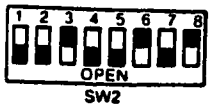
For single-ended mode SW2-2,4,5,-7
are closed SW2-1,3,6,8 are open.

For differential mode SW2-3,6,8
are closed, SW2-1,2,4,5,7 are open.

In single-ended mode there are 32 individual analog input channels.

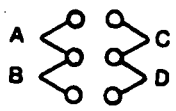
In differential mode there are 16 analog input channels made up of the voltage on one of the channels on connector P1 minus the voltage on the corresponding channel on connector P2.





Jumper Use

J1: Parallel Port (C) Bit 1 and 5, Inverted/Non-Inverted.

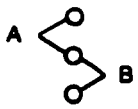


This jumper is for selecting either inverted or non-inverted outputs fr parallel port C bits 1 and 5. The inverted outputs are needed only if t parallel IC (the 8255) is to be connected to another 8255 somewhere el: in the strobed data mode (mode 1). For most other uses the jump should be in the non-inverted output positions.

Position A	Port C	Bit 1	Non-inverted
Position B	Port C	Bit 1	Inverted
Position C	Port C	Bit 5	Non-inverted
Position D	Port C	Bit 5	Inverted

The usual setting will be with jumpers in positions A and C, which is ho the board is shipped.

J2: Timer Out 1/Out 2 as Interrupt Source



This jumper selects either the timer output OUT1 or OUT2 to be a possi ble interrupt source.

Position A	OUT2 as source
Position B	OUT1 as source

I3: Exteral A/D Start Convert Input

```
     1        2
nput -O    O - GND
```

This jumper is actually a connector for an external 'start conversion' signal. When the A to D converter is set up for external conversion control (see programming section) the transition from high to low of a TTL ogic level on pin 1 of J2 will start an A to D conversion. Pin 2 of J3 is a ground connection to be used as a signal return line for the external convert signal. When external conversion control is not being used, a jumper hould be placed over these two pins; this is the way the board is shipped.

Connector Organization

There are five ribbon cable connectors on the Lab Tender foı connections to the outside world. The pin numbering for all conr the same:

```
Top of
Board
1 o o 18
2 o o 19
3 o o 20
4 o o 21
5 o o 22
6 o o 23
7 o o 24
8 o o 25
9 o o 26        Side of board
10 o o 27       Back of IBM-PC
11 o o 28
12 o o 29
13 o o 30
14 o o 31
15 o o 32
16 o o 33
17 o o 34
```

P1 and P2: These two connectors are for the A to D converter inpu There are 32 analog input channels if the board is set up for single-end (SE) use and 16 pairs of channels if the board is set up for differential i put (DI) use. In differential mode two channels are taken, one chann each from P1 and one channel from P2. The functions of the pins on P and P2 on the following page.

P1

Pin Number	Function
1	A to D Channel 0
2	A to D Channel 1
3	A to D Channel 2
4	A to D Channel 3
5	A to D Channel 4
6	A to D Channel 5
7	A to D Channel 6
8	A to D Channel 7
9	A to D Channel 8
10	A to D Channel 9
11	A to D Channel 10
12	A to D Channel 11
13	A to D Channel 12
14	A to D Channel 13
15	A to D Channel 14
16	A to D Channel 15
17-34	Ground

P2

Pin Number	Function
1	A to D Channel 16 (Channel 0 Differential)
2	A to D Channel 17 (Channel 1 Differential)
3	A to D Channel 18 (Channel 2 Differential)
4	A to D Channel 19 (Channel 3 Differential)
5	A to D Channel 20 (Channel 4 Differential)
6	A to D Channel 21 (Channel 5 Differential)
7	A to D Channel 22 (Channel 6 Differential)
8	A to D Channel 23 (Channel 7 Differential)
9	A to D Channel 24 (Channel 8 Differential)
10	A to D Channel 25 (Channel 9 Differential)
11	A to D Channel 26 (Channel 10 Differential)
12	A to D Channel 27 (Channel 11 Differential)
13	A to D Channel 28 (Channel 12 Differential)
14	A to D Channel 29 (Channel 13 Differential)
15	A to D Channel 30 (Channel 14 Differential)
16	A to D Channel 31 (Channel 15 Differential)
17-34	Ground

P3: This connector is for the D to A outputs. There are 16 analog outpu
organized as:

P3

Pin Number	Function
1	D to A Channel 0
2	D to A Channel 1
3	D to A Channel 2
4	D to A Channel 3
5	D to A Channel 4
6	D to A Channel 5
7	D to A Channel 6
8	D to A Channel 7
9	D to A Channel 8
10	D to A Channel 9
11	D to A Channel 10
12	D to A Channel 11
13	D to A Channel 12
14	D to A Channel 13
15	D to A Channel 14
16	D to A Channel 15
17-34	Ground

P4: This connector is the parallel port connector. It has three 8 bit parallel ports organized as:

Pin Number	P4 Function
1	Port A bit 7
2	Port A bit 5
3	Port A bit 3
4	Port A bit 1
5	Port C bit 7
6	Port C bit 5
7	Port C bit 3
8	Port C bit 1
9	Port B bit 7
10	Port B bit 5
11	Port B bit 3
12	Port B bit 1
13	No function
14	Port A bit 6
15	Port A bit 4
16	Port A bit 2
17	Port A bit 0
18-26	Ground
27	Port B bit 0
28	Port B bit 2
29	Port B bit 4
30	Port B bit 6
31	Port C bit 0
32	Port C bit 2
33	Port C bit 4
34	Port C bit 6

P5: This connector is for the Timer. The timer has 16 external outputs. These are organized as:

P5

Pin Number	Function
1	Gate 1
2	Gate 2
3	Gate 3
4	Gate 4
5	Gate 5
6	OUT1
7	OUT2
8	OUT3
9	OUT4
10	OUT5
11	Source 1
12	Source 2
13	Source 3
14	Source 4
15	Source 5
16	Frequency Out
17-34	Ground

Connector Summary

Connector	Function	# Pins	Compatible Mating Connector Mfgr.	Connector Part #
P1	A/D input	34	BERG	65611-234
P2	A/D input	34	BERG	65611-234
P3	D/A outputs	34	BERG	65611-234
P4	Parallel Port I/O	34	BERG	65611-234
P5	Timer I/O	34	BERG	65611-234
Interrupt Header	Interrupt Jumpering	16	T&B Ansley	609-M161H

Use of the Interrupt Header

The jumper header in the 16 pin socket can be used to connect any of the interrupt sources on the board to any of the IBM IRQ lines. The header looks like:

16	15	14	13	12	11	10	9
1	2	3	4	5	6	7	8

Interrupt Source	Source Header Pin
Overrun Error Condition	1
A/D done	2
Timer Interrupt	3
Parallel Interrupt 1	4
Parallel Interrupt 2	5
Overrun Error Condition or	6
A/D done	6

IRQ Lines	Header Pin Numbers
IRQ 2	16
IRQ 3	15
IRQ 4	14
IRQ 5	13
IRQ 6	12
IRQ 7	11

Header Pins 7, 8, 9, 10 have no function.

An interrupt is connected by soldering a wire from the interrupt source pin to the desired IRQ line. If no interrupts will be used, then make no connections on the header.

Programming the Lab Tender

This board uses 16 consecutive I/O locations. The starting I/O address is selected with switches SW1 and SW3. See Switch, Jumper, and Connector Summary. All 16 address lines A15-AØ are decoded, giving the full range of starting I/O addresses from ØØØØH to FFFØH. The board comes from the factory with starting address Ø33ØHex (816 decimal).

The following is a list of the functions and their locations relative to the starting location of the board.

Read or Write	Location Equals Starting Location Plus	Function
Write	Ø	A/D control port: D0: A '1' initiates a new A/D conversion. A '0' inhibits a new conversion. D1-D5: Selects one of the 32 analog channels in single ended mode, or, D1-D4: Selects one of the 16 pairs of channels in differential mode. In both cases the binary value of these bits (as if they were data bits D0-D4 or D0-D3) is the channel number. D6: A '1' allows conversions to be initiated through the external conversion connector. A '0' disables the external conversion feature, allowing conversions only by data bit D0. The board powers-up with external input disabled (a '0' in this bit). D7: No function.
Read	Ø	A/D Status Port: DØ: Error Status: A '1' indicates that a new conversion was initiated before the data from the previous conversion was read. A '0' indicates that there is no error. The error bit is reset if an A/D data read is done. D1-D6: These bits reflect the settings of D1-D6 by the write to this port so that the channel selected may be checked at any time.

		D7: Done status: A '1' indicates that the A/D converter has finished a conversion. A '0' indicates it is still converting (this takes 18 micro-sec.). The done bit is reset to '0' when an A/D data read takes place; it is set to a '1' at the end of an A/D conversion.
Read	1	A/D Data Port: This is the 8 bit value from the most recent A/D conversion. It is in offset binary which means that 0000 0000 = −5.00V, 10000 0000 = 0V, and 1111 1111 = +4.96V, and each increment of one (one bit) is equal to 0.04 Volts.
Write	2	Timer Interrupt Clear: A write (output) to this location will clear the interrupt from the timer if the timer has been previously setup and the timer interrupt source has been connected to one of the IBM IRQ lines with a connection on the interrupt header. Data D0-D7 may have any value.
Write	4	D/A Multiplexer Control: D0-D2: Selects one of eight D/A channels in either multiplexer 0 or multiplexer 1. D3: A '1' selects multiplexer 0 (channels 0-7). A '0' disables multiplexer 0. D4: A '1' selects multiplexer 1 (channels 8-15). A '0' disables multiplexer 1. D5-7: No function.

Programming the A to D Converter:

Using the A/D converter is quite simple. The program must first select the input channel by outputing the channel number to the A/D control port (Base +Ø). If bit D0 of the A/D control byte is a '1' a conversion will be immediately initiated, otherwise only the multiplexer channel select will be changed. This will remain set until a '1' is written to DØ of the A/D control port. After a conversion has been initiated the program must wait until a read of the A/D status port (Base +Ø) produces a '1' in bit D7 of the input byte (DONE STATUS). After this, the data must be read from the A/D Data Port (Base +1) before a new conversion is started or it will be lost and the error bit will be set. If the done bit is being used to generate an interrupt, then the data may be read as soon as the interrupt occurs. The done bit is set to zero when an A/D data read is performed: this is automatic.

If the external conversion bit (D6) is set in the A/D control word then the falling edge of a TTL strobe applied to the external conversion connector J4 will trigger an A/D conversion. For external conversion triggering, using the done bit as an interrupt is recommended. The external conversion signal can come from any TTL clock source, even from the 9513 timer, but must not occur faster than every 20 μsec. or the external convert signal will be going faster than the converter can convert.

Programming the D to A Converter:

To control only one channel of the D/A, the program must merely first select the channel to be used. After that, any value sent to the D/A converter location will appear as a voltage at that D/A channel pin on connector P3.

Controlling multiple channels is a little more complex. First, before changing channels, the multiplexers must be disabled by outputting 0's to bits D3 and D4 of the D/A multiplexer control port (Base +4) while keeping the channel select bits the same as they were. While the multiplexers are disabled, the channel number may be changed, the D/A data changed, then either multiplexer 0 or 1 may be enabled. If the multiplexers are not dissabled first, glitching may occur on one of the D/A outputs.

Counters

The counters of the AM9513 can be used for count accumulation, frequency division, or a combination of both, depending on the application.

In count accumulation, each counter keeps track of the number of transitions that occur on its input. Without disturbing the counting process, the counter value can be checked at any time by the CPU, or it can be compared with an independent value. The input to the counter can be varied by using different types of inputs and controls.

In frequency division, a particular output frequency is desired and the value of the counter is of less interest. To get the frequency desired, the input frequency and the 'length' being counted are controlled. The output polarities are programmable, and controls allow the output to be pulsed or to be output as a waveform.

The counters in the AM9513 can be programmed to count up or down in either binary or binary coded decimal (BCD). Sixteen counter sources are available for each counter, and the input polarity is individually selectable as active-high or active-low. Gating functions allow direct hardware and software control of the count accumulation. Adjacent counters can be internally concatenated to form an effective counter length of up to 80 bits (5 16-bit counter).

Counters 1 and 2 in the AM9513 can be configured to provide a 24-hour real-time clock. Auxiliary alarm registers on these counters allow the contents to be compared with previously entered values.

Data Bus

At power-up or reset, the external data bus is configured for 8-bit width. The data bus lines act as inputs during a READ operation and outputs during a WRITE.

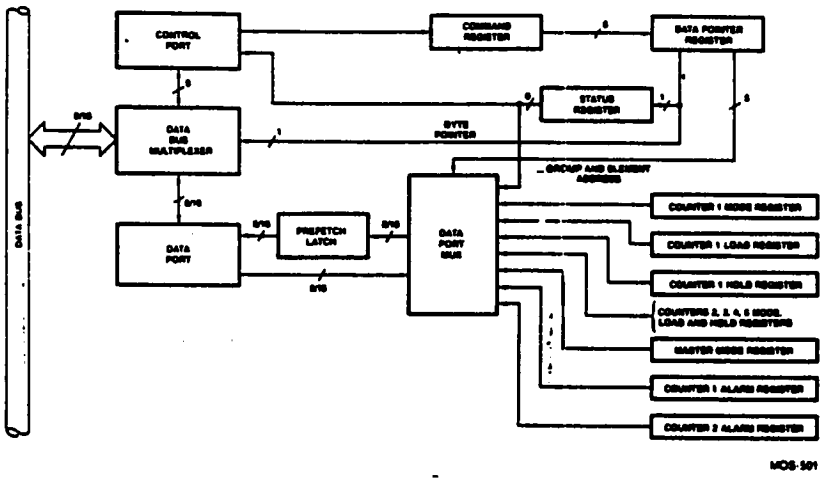
The control and data port registers (discussed on pp. 5-8) are all connected to a common internal 16-bit bus. When the external bus is in the 8-bit mode, the internal 16-bit information is multiplexed.

Frequency Sources

The internal oscillator is one source of frequency inputs to the counters. An external crystal, RC (Resistive Capacitive), IC (Inductive Capacitive), or other type of reactive network can be used to control the frequency of the oscillator. (This is a 1.000 MHz crystal on Tecmar boards). A Frequency Scaler divides the oscillator output to provide several subfrequencies. Any of these frequencies can be selected as an input for an individual counter. These frequencies can also be used by another part of the system.

Functional Description

The AM9513 is addressed by the external system as only two locations: the control port and the data port. See Figure 1.



Am9513 Register Access
Figure 1

Control Port

The control port allows direct access to the Command register when writing, and to the Status register when reading. (All other available internal locations are accessed for both reading and writing via the data port). Transfers to and from the control port are always 8 bits wide.

Note: Each access to the control port will transfer between the Command register (during a WRITE) or Status register (during a READ) and the Data Bus.

Command Register

Control over the five counters of the AM9513 is provided by the Command register. It also controls access through the data port by allowing the user to update the Data Pointer register.

Pages 9-14 of this document explain each of the commands in detail; Figure 3 (P. A22) gives a brief summary.

Six of the command types are used for direct software control of the counting process. Each of these commands contains a 5-bit S field. In a linear-select fashion, the first through the fifth bits in the S field each correspond to one of the five counters. (S1 = Counter 1, S2 = Counter 2, S3 = Counter 3, S4 = Counter 4, and S5 = Counter 5). When an S bit is a one, the specified operation is performed on the designated counter. When an S bit is a zero, no operation occurs for the corresponding counter.

Data Pointer Register

The 6-bit Data Pointer register is loaded by issuing the appropriate command through the control port to the Command register. The contents of the Data Pointer register are used to control the data port multiplexer and select which internal register is to be accessible through the data port.

As shown in Figure 4 (p. A34) the Data Pointer register consists of a 3-bit Group Pointer, a 2-bit Element Pointer, and a 1-bit Byte Pointer.

The Element and Group Pointers are used to select which internal register is to be accessible through the data port. The Byte Pointer bit indicates which byte of a 16-bit register is to be transferred on the next access through the data port. If the Data Pointer is loaded, the Byte Pointer bit is set to one, indicating a least-significant byte is expected. The Byte Pointer toggles following each 8-bit data transfer with an 8-bit data bus configuration; it remains set with a 16-bit data bus configuration.

The Byte Pointer Status is available only through the Status Register.

Random access to any available internal data location can be accomplished by simply loading the Data pointer using the command shown in Figure 3 and then initiating a data read or data write. This procedure can be used at any time, regardless of the setting of the Data Pointer Control bit (MM14). When the 8-bit data bus configuration is being used (MM13 = 0), two bytes of data would normally be transferred following the issuing of the 'Load Data Pointer' command.

To permit the host processor to rapidly access the various internal registers, automatic sequencing of the Data Pointer is provided. Sequencing is enabled by clearing Master Mode bit 14 (MM14) to zero. As shown in Figure 10, several types of sequencing are available depending on the data bus width used and the initial Data Pointer value entered by command.

When E1 = 0 or E2 = 0 and G4, G2, G1 point to a Counter Group, the Data Pointer will proceed through the Element cycle. The Element field will automatically sequence through the three values 00, 01 and 10 starting with the value entered. When the transition from 10 to 00 occurs, the Group field will also be incremented by one. Note that the Element field in this case does not sequence to a value of 11. The group field circulates only within the five Counter Group codes.

If E2, E1 = 11 and a Counter Group is selected, then only the Group field is sequenced. This is the Hold cycle. It allows the Hold registers to be sequentially accessed while bypassing the Mode and Load registers. The third type of sequencing is the Control cycle. If G4, G2, G1 = 111 and E2, E1 = 11, the Element Pointer will be incremented through the values 00, 01 and 10, with no change to the Group Pointer.

When G4, G2, G1 = 111 and E2, E1 = 11, no incrementing takes place and only the Status register will be available through the data port. Note that the Status register can also always be read directly through the Control port.

For all of these auto-sequence modes, if an 8-bit bus is used, the Byte pointer will toggle after every data transfer to allow the least and most significant bytes to be transferred before the Element or Group Fields are incremented.

The auto-sequence modes are illustrated in Figure 10.

Data Port

The data port is used to communicate with the addressable internal locations that cannot be addressed through the command port. To understand the various registers that are accessible through the data port, it is best first to discuss the Counter Logic Group.

Counter Logic Group

The five counter logic groups that exist in an AM9513 each consist of a 16-bit counter with associated control and output logic, a 16-bit Load register, a 16-bit Hold register, and a 16-bit Mode register. Additionally, counter logic groups 1 and 2 each contain a 16-bit comparator and a 16-bit Alarm register. The CPU has both Read and Write access to all the registers in the counter logic group through the data port. The counters themselves cannot be directly accessed.

Counter Mode registers: Five registers, one for each counter, that allow the user to select one of 16 inputs to the counter, gating and repitition modes, type of counting, and input and output polarity.

Master Mode register: Controls the options that are not controlled by the individual Counter Mode registers. See the next subsection for more information about this register.

Load register: Five registers, one for each counter, that control the count period by resetting the counter to a user-defined value. A value is written into the Load register and transferred to the counter when a Terminal Count occurs. The value in either the Load or Hold register is transferred on a Terminal Count in all operating modes. For count accumulation, the transfer of the Load register value can be transparent if it is filled with zeros.

Hold registers: Five registers, one for each counter, that save the counter values without interrupting the counting process. The CPU can then check these values. These registers can also be used as extra Load registers to generate complex output waveforms.

Alarm registers: Two 16-bit alarm registers, one each on counters 1 and 2, that together with its comparator can signal a particular count.

Comparators: Two comparators, one each on counters 1 and 2, whose output goes true when the value in the alarm register equals the value of the associated counter value.

Master Mode Register

The 16-bit Master Mode register controls the internal activities not controlled by the Counter Mode registers. With the Master Mode register, frequency, time-of-day operations, comparator controls, binary or BCD division, and data bus width can be specified. A choice of Data Pointer increment enable or disable is also available. Figure 5 on page 23 shows the bit assignments for the Master Mode Register.

Additionally, three of the Master Mode register bits can be set independently of the rest through commands to the Command register. These bits control the FOUT gate, the data bus width, and the Data Pointer.

Prefetch Circuit

In order to minimize the read access time to internal Am9513 registers, a prefetch circuit is used for all read operations through the Data Port. Following each read or write operation through the Data Port, the Data Pointer register is updated to point to the next register to be accessed. Immediately following this update, the new register data is transferred to a special prefetch latch in the 9513 I.C. When the user performs a subsequent read of the Data Port, the data bus drivers are enabled, outputting the prefetched data on the bus. Since the internal data register is accessed prior to the start of the read operation, its access time is transparent to the user. In order to keep the prefetched data consistent with the data pointer, prefetches are also performed after each write to the Data Port and after execution at the 'Load Data Pointer' command. The following rules should be kept in mind regarding Data Port Transfers.

1. The Data Pointer register should always be reloaded before reading from the Data Port if a command other than 'Load Data Pointer' was issued to the Am9513 following the last Data Port read or write. The Data Pointer does not have to be loaded again if the first Data Port transaction after a command entry is a write, since the Data Port write will automatically cause a new prefetch to occur.

2. Operating modes N, O, Q and R allow the user to save the counter contents in the Hold register by applying an active-going gate edge. If the Data Pointer register had been pointing to the Hold register in question, the prefetched value will not correspond to the new value saved in the Hold register. To avoid reading an incorrect value, a new 'Load Data Pointer' command should be issued before attempting to read the saved data. A Data Port write (to another register) will also initiate a prefetch; subsequent reads will access the recently saved Hold register data. Many systems will use the 'saving' gate edge to interrrupt the host CPU. In systems such as this the interrupt service routine should issue a 'Load Data Pointer' command prior to reading the saved data.

Status Register

The status register is an 18-bit read-only register that can be accessed either by a read at the control port address or a read of the data port with the Data Pointer Register set to point to the Status Register. This register indicates the state of the Byte Pointer Bit in the Data Poiinter register and the state of the OUT signal for each of the counter groups. The OUT signals status reported is the state of the signal after the polarity select logic and just before the 3-state buffer interface circuitry.

Each bitin the Status register corresponds to one of the OUT signals and will reflect the exact state of that signal; for example, if the OUT 3 line = 1 then the OUT3 status bit = 1.

For counters 1 and 2 the OUT pins and status bits will reflect the comparator output if the comparators are enabled. The status register bit and OUT pin are active high if Counter Mode bit 2 = 0 and active low if Counter Mode bit2 = 1. When the Low Impedence to Ground Option is selected and the comparator is enabled, status bit will reflect an active low comparator output. If the comparator is not enabled then the status bits for OUT1 and/or OUT2 will function like the other status bits (reflect the state of the OUT pin). Refer to Figure 9 for the Status Register bit assignments.

Commands

AM9513 commands are entered through the control port into the 8-bit command register. The commands allow the CPU to use a variety of operating modes and other options, depending on the application. With these modes and options, the CPU can initialize and update the internal data and control information, and it can manipulate operating bits during operation.

As described earlier, six of the command types are used for direct software control of the counting process. Each of these command types contains a 5-bit S field, with each of the bits corresponding to counters 1 through 5 (S1 = Counter 1, S2 = Counter 2, etc). When an S bit is a 1, the specified operation is performed on the designated counter. When an S bit is a 0, no operation occurs for that counter.

Note: This command format is advantageous because by using only one command, any combination of counters can be specified and acted on.

Figure 3, a summary of the commands and command codes, is shown on page 14.

In the following tables the Data bits in the command word and their corresponding functions are indicated.

Arm Counters Command

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	S5	S4	S3	S2	S1

Counters specified in the S field will be enabled for counting. Counters must be armed before counting can commence. Hardware gating facilities may then enable or disable the counting process. Software gating is performed using the ARM and DISARM commands. The Arm Counters command can arm a counter or do nothing. A zero in the S field will not disarm the counter.

NOTE: To arm Counters 2 and 3, for example, the command code would be 0010, 0110, or 38 decimal. The section titled 'Using the AM9513' gives specific examples of how to program the circuit using the BASIC programming language.

Load Counters Command

D7	D6	D5	D4	D3	D2	D1	D0
0	1	0	S5	S4	S3	S2	S1

Counters specified in the S field will be loaded with previously entered values from either of the associated Load or Hold registers. (The Counter Mode register determines whether the value will come from the Load or Hold register. See Page A35). This command does not change the value(s) of the Load or Hold register(s). It is often used as a counter initialization prior to active hardware gating or as a software trigger.

Load and Arm Counters Commands

D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	S5	S4	S3	S2	S1

Couunters specified in S field will be armed then loaded, just as if an Arm command and then a Load command were performed.

Disarm Counters Command

D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	S5	S4	S3	S2	S1

Counters specified in S field will be disabled from counting. A disarmed counter will stop counting regardless of other control conditions unless it is in the TC state. If in the TC state, it will count once before disarming. The Arm command will resume the counting.

Save Counters Command

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	S5	S4	S3	S2	S1

Counters specified in the S field will have their contents moved into the corresponding Hold register. The counting process will not be interrupted, and the contents being transferred will overwrite any previous Hold register contents. This allows the accumulated count to be preserved.

Disarm and Save Counters Command

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	S5	S4	S3	S2	S1

Counters specified in the S field will be disarmed and the contents will be moved to the corresponding Hold registers. A Disarm command followed by a Save command will accomplish the same thing.

Set Output Command

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	0	1	N4	N2	N1

(001 < or = N > or = 101)

This command sets the toggle for counter N. The OUTN signal will be driven high unless a Terminal Count output is specified.

Clear Output Command

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	0	0	N4	N2	N1

(001 < or = N < or = 101)

This command resets the toggle for counter N. The OUTN signal will be driven low unless a TC output is specified.

Step Counter Command

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	0	N4	N2	N1

(001 < or = N < or= 101)

This command increments or decrements counter N by 1, depending on the operating configuration. If the correspondong Counter Mode register CM3 bit is cleared to zero, the counter will decrement by 1. The counter will inrement by 1 if the CM3 bit is set to a logic high. Even a disarmed counter can be incremented or decremented by the Step command.

Load Data Pointer Register Command

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	E2	E1	G4	G2	G1

(G4, G2, G1 ≠ 000, ≠ 110)

The bits specified in the E (Element) and G (Group) fields of the command code will be sent to the corresponding E and G fields of the Data Pointer register. Figure 4 on page A34 gives the bit assignments for the Element and Group Pointers of the Data Pointer register. The Byte pointer bit of the Data Pointer register is set. G values of 000 and 110 should not be used.

Disable Data Pointer Sequencing Command

D7	D6	D5	D4	D3	D2	D1	
1	1	1	0	1	0	0	= E8 Hex

This commands sets bit 14 of the Master Mode register without changing any other bits in the register. Bit MM14 controls the sequencing of the Data Pointer register. Using this command disables the sequecing and thus allows the host processor to continue accessing a particular internal location without having to update the Data Pointer each time. Loading a full word into the Master Mode register can also change or control MM14.

Enable Data Pointer Sequencing Command

D7	D6	D5	D4	D3	D2	D1	D0	
1	1	1	0	0	0	0	0	= E0 Hex

This command clears bit 14 of the Master Mode register without changing any other bits in the register. Bit MM14 controls the sequencing of the Data Pointer register. Using this command enables the sequencing and thus allows the host processor to access several internal locations without repetitive updating of the Data Pointer. Loading a full word into the Master Mode register can also change or control MM14.

Enable 16-bit Data Bus Command

D7	D6	D5	D4	D3	D2	D1	D0	
1	1	1	0	1	1	1	1	=EF Hex

This command sets bit 13 of the Master Mode register without changing any of the other bits in the register. Bit MM13 controls the multiplexer in the data bus buffer. No multiplexing occurs and the sixteen external data bus lines are used to transfer information into and out of the AM9513. Loading the full Master Mode register in parallel can also change or control MM13. The 16-bit bus mode will never be used in the IBM PC.

Enable 8-bit Data Bus Command

D7	D6	D5	D4	D3	D2	D1	D0	
1	1	1	0	0	1	1	1	= E7 Hex

This command clears bit 13 of the Master Mode register without changing any of the other bits in the register. Bit MM13 controls the multiplexer in the data bus buffer. The multiplexer is enabled and 16-bit internal information is transferred eight bits at a time to the eight low-order external data bus lines. Loading the full Master Mode register in parallel will also control MM13. The 9513 powers up with this bit cleared.

Gate off FOUT Command

D7	D6	D5	D4	D3	D2	D1	D0	
1	1	1	0	1	1	1	0	= EE Hex

This command sets bit 12 of the Master Mode register without changing any of the other bits in the register. Bit MM12 controls the output state of the FOUT signal. When this bit is set the FOUT line will exhibit a low impedance to ground. Loading the full Master Mode register in parallel can also change or control MM12.

Gate On FOUT Command

D7	D6	D5	D4	D3	D2	D1	D0	
1	1	1	0	0	1	1	0	= E6 Hex

This commands clears bit 12 of the Master Mode register without changing any of the other bits. Bit MM12 controls the output state of the FOUT signal. the FOUT line will become active and drive output (on the Frequency Out line) the selected and divided FOUT signal. Loading the full Master Mode register in parallel will also control MM12. A transient pulse may be generated on the FOUT signal at the time FOUT is gated on or off.

Master Reset Command

D7	D6	D5	D4	D3	D2	D1	D0	
1	1	1	1	1	1	1	1	= FF Hex

This command duplicates the action of the power-on reset circuitry. It disarms all counters, enters 0000 in the Master Mode, Load, and Hold registers, and enters 0B00 (hex) in the Counter Mode registers. The Load command should be performed after a power-up or software reset to clear any counters that may be in a Terminal Count state. Because reset does not initialize it, the Data Pointer register must be set to a legal value. To reset, the command code 1111 1111, or 255 decimal is used. Once a reset has been performed, the Master Mode, Counter Mode, Load, and Hold registers can be initialized to the desired values.

Unused Command Codes

Table 1 shows the chart of unused command codes, that is, command codes that should not be used.

Table 1

C7	C6	C5	C4	C3	C2	C1	C0	
1	1	1	1	0	0	0	0	= F0 Hex
1	1	1	1	0	1	1	0	= F6 Hex
1	1	1	1	0	1	1	1	= F7 Hex
0	0	0	X	X	1	1	0	
0	0	0	X	X	0	0	0	
1	1	1	1	1	X	X	X	

Note: 'X' means that the indicated bit may be either a '1' or a '0'.

Command Code								Command Description
128 C7	64 C6	32 C5	16 C4	8 C3	4 C2	2 C1	1 C0	
0	0	0	E2	E1	G4	G2	G1	Load Data Pointer register with contents of E and G fields (G ≠ 000 G ≠ 110)
0	0	1	S5	S4	S3	S2	S1	Arm counting for all selected counters
0	1	0	S5	S4	S3	S2	S1	Load contents of specified source into all selected counters
0	1	1	S5	S4	S3	S2	S1	Load and Arm all selected counters
1	0	0	S5	S4	S3	S2	S1	Disarm and Save all selected counters
1	0	1	S5	S4	S3	S2	S1	Save all selected counters in hold register
1	1	0	S5	S4	S3	S2	S1	Disarm all selected counters
1	1	1	0	1	N4	N2	N1	Set output bit N (001 ≤ N ≤ 101)
1	1	1	0	0	N4	N2	N1	Clear output bit N (001 ≤ N ≤ 101)
1	1	1	1	0	N4	N2	N1	Step counter N (001 ≤ N ≤ 101)
1	1	1	0	1	0	0	0	Set MM14 (Disable Data Pointer Sequencing)
1	1	1	0	1	1	1	0	Set MM12 (Gate off FOUT)
1	1	1	0	1	1	1	1	Set MM13 (Enter 16-bit bus mode)
1	1	1	0	0	0	0	0	Clear MM14 (Enable Data Pointer Sequencing)
1	1	1	0	0	1	1	0	Clear MM12 (Gate on FOUT)
1	1	1	0	0	1	1	1	Clear MM13 (Enter 8-bit bus mode)
1	1	1	1	1	1	1	1	Master reset

Figure 3
Summary of Command Codes

Operating Modes

Figure 7 on page 35 shows the bit assignments for the Counter Mode register. Bits 5-7 and 13-15 of the Counter Mode register select the operating mode for each counter. For simplicity, the eighteen different modes that can be used each have an assigned letter name from A through X.

Notes:

1. Operating Modes M, P, T, U, W, and X are reserved and should not be used.

2. Figure 8 on page 22 summarizes the operating modes described in this section. It is suggested that you read through all the descriptions prior to trying to understand the table.

3. 'Terminal Count' or TC, which is referred to often in this section, is the length of time when a counter's contents would have been zero if a reload had not occurred. See the Terminal Count section for the special conditions immediately before and during TC.

MODE A
Software Triggered Strobe with No Hardware Gating

In this mode, the counter is available for counting source edges when it is issued an ARM command. On Each Terminal Count, the counter reloads itself from the Load register and automatically disarms itself, inhibiting further counting. When a new Arm command is issued, counting resumes.

MODE B
Software Triggered Strobe with Level Gating

The counter is available for counting source edges only when the assigned gate is active. The counter must be armed for counting to occur.

MODE C
Hardware Triggered Strobe

Mode C is similar to Mode B except that a gate edge must be applied to the armed counter for the counting of source edges to begin. The counter must be armed before application of the triggering gate edge. Gate arms applied to a disarmed counter are disregarded. The counter will start counting on the first source edge after the triggering gate edge and will continue counting until TC. At TC, the counter will reload from the Load register and automatically disarm itself. Counting will then remain inhibited until a new ARM command and a new gate edge are applied in that order. Note that after application of a triggering gate edge, the gate input will be disregarded for the remainder of the count cycle. This differs from Mode B where the gate can be modulated throughout the count cycle to stop and start the counter.

MODE D
Rate Generator with No Hardware Gating

In Mode D, the gate input does not influence the counter operation. After a counter is armed, it will count to TC repetitively. When it reaches TC, the counter reloads itself from the Load register and thus the Load register determines the time between TCs. By specifying the TC Toggled output mode in the Counter Mode register, a square wave generator can be produced.

MODE E
Rate Generator with Level Gating

Similar to Mode D, in Mode E only the source edges occuring while the gate input is active will be counted. This allows the counting process to be enabled and disabled under hardware control. By specifying the TC Toggled output mode, a square wave generator can be produced.

MODE F
Non-Retriggerable One-Shot

This mode provides a non-retriggerable one-shot timing function. In Mode F, the counter must be armed in order to function. Counting is enabled by applying a gate edge to the armed counter. At TC, the counter reloads from the Load register and waits for another gate edge. Mode F differs from Mode C in that only a new gate is required, not a new ARM command. Once the triggering gate edge is applied, the gate input is ignored until the next TC.

MODE G
Software-Triggered Delayed Pulse One-Shot

In this mode, counter operation is not affected by the gate. After the counter is armed, it counts to TC twice and then disarms itself. The counter is usually loaded the first time from the Load register by either a Load command or by the last TC of a timing cycle. When the counter reaches the first new TC, it reloads itself from the Hold register. At the second TC, the counter reloads itself from the Load register and disarms, thus stopping counting. A new ARM command starts counting again. Specifying the TC Toggled output mode in the Counter Mode register generates the software-triggered, delayed-pulse-one-shot. The counter's initial contents control the amount of time between the arm command and the output pulse. The pulse duration is controlled by the contents of the Hold register.

MODE H
Software-Triggered Delayed Pulse One-Shot with Hardware Gating

Mode H is similar to Mode G except that the gate input determines which source edges are to be counted. After the counter is armed, it counts only the source edges that occur while the gate is active; source edges that occur while the gate is inactive are ignored. The counting process, therefore, can be turned on and off by the gate. On the first TC, the counter reloads from the Hold register; on the second TC, it reloads from the Load register and disarms itself. In this mode, both the pulse width and the initial output delay time can be controlled by the gate.

MODE I

Hardware-Triggered Delayed Pulse Strobe

Mode I is also similar to Mode G except that counting cannot begin until a gate edge is applied to an armed counter. Gate edges applied to an unarmed counter are ignored. On the first source edge after the gate edge, the armed counter starts counting. On the first TC, the counter reloads from the Hold register; on the second TC, it reloads from the Load register and disarms itself. An ARM command and the application of a gate edge are necessary to restart counting. Unlike Mode H, gate edges after the initial gate edge are ignored until the second TC. This means the gate cannot turn on and off the count process.

MODE J

Variable Duty Cycle Rate Generator with No Hardware Gating

In Mode J, an armed counter will count continuously until given a DISARM command. On the first TC, the counter reloads from the Hold register; on the second TC, it reloads from the Load register. Counting continues, the counter reloading first from the Hold register, then from the Load register, until the counter is given a DISARM command. Specifying the TC Toggled output in the Counter Mode register generates a variable duty cycle output. In this mode, the Load and Hold register values control the output duty cycle, giving high resolution when the count values are relatively high.

MODE K

Variable Duty Cycle Rate Generator with Level Gating

Mode K is similar to Mode J except that the source edges are counted only when the gate is active, thus permitting the gate to turn on and off the count process. After the counter is armed it counts all source edges occuring while the gate is active and ignores all source edges while the gate is inactive. The reload source alternates as in Mode J, starting with the Hold register on the first TC, and then the Load register. In this mode, the gate can control the duty cycle of the output waveform (including both the high and low portions of the output waveform) if the TC Toggled output is used.

MODE L

Hardware-Triggered Delayed Pulse One-Shot

Mode L is also similar to Mode J, except that a gate edge must be applied to an armed counter for counting to begin. After the first gate edge is applied, the counter will count source edges until the second TC. In this mode, the gate input after the triggering gate edge is disregarded until the next count cycle begins, and thus, the count process cannot be turned on and off by the gate. On the first TC, the counter is reloaded from the Hold register; on the second TC, it is reloaded from the Load register and counting is terminated until a new gate edge is applied to the counter. Note that counting will not continue without a new gate edge after the second TC.

MODE N

Software-Triggered Strobe with Level Gating and Hardware Retriggering

In Mode N, an ARM command must be given to a counter for counting to commence. After the counter is armed, it counts only the source edge occurring while the gate is active (qualified source edges), thus allowing the count process to be turned on and off by the gate. Unqualified source edges are ignored. After an ARM command and the application of a gate edge, the counter counts to TC, reloads itself from the Load register, and disarms itself, stopping counting. Counting starts again with an ARM command. If an active-going gate edge is applied to the counter, the counter contents are saved in the Hold register. After the application of the retriggering gate edge, the first qualified source edge will transfer the contents of the Load register into the counter. The second qualified source edge will resume the counting process.

MODE O

Software-Triggered Strobe with Edge Gating and Hardware Retriggering

In Mode O, the gate level cannot turn on and off the count process and counting cannot begin until the armed counter receives an active-going gate edge. Regardless of the gate level, all source edges are counted until the first TC. The counter is reloaded from the Load register and disarmed on the first TC. For a new count cycle to begin, first an ARM command must be given and then a gate edge must be applied. Mode O differs from Modes C, F, I and L in that the count process is retriggered on all active-going gate edges, including the first. The contents of the counter are transferred into the Hold register following each retriggering gate edge. The first source edge that follows transfers the Load register contents into the counter. Counting resumes on the second source edge after a retrigger.

MODE Q

Rate Generator with Synchronization (Event Counter with Auto-Read/Reset)

A counter in Mode Q can be used as a rate generator with synchronization or an event counter with auto-read/reset. For counting to occur, the counter must first be given an ARM command. It will then count all source edges that occur while the gate is active (qualified) and ignore all unqualified source edges. This allows the count process to be turned on and off by the gate. The counter reloads from the Load register on each TC, and is retriggered by the next qualified gate edge. The retriggering gate edge transfers the contents of the counter into the Hold register, and the first source edge that follows transfer the contents of the Load register into the counter. The secound qualified source edge resumes counting.

MODE R
Retriggerable One-Shot

Mode R is similar to Mode Q except that the gate edges start the counting. When an armed counter has a gate edge applied, it will count all the source edges until it reaches TC, regardless of the gate level. At TC, the counter is reloaded from the Load register and halted. Counting cannot resume unless a new gate edge is applied first. This transfers the contents of the counter to the Hold register. The first source edge that follows transfer the contents of the Load register to the counter. The second source edge that follows start the counting.

MODE S
Reload Source Control

In Mode S, the gate input selects the reload source for LOAD commands (armed or disarmed) and for TC-initiated reloads. The gate input does not control or cause counting. The Load register is used as the reload source when the gate is low; the Hold register is used when it is high. This is easy to remember as low-Load and high-Hold. After the counter is armed, it counts to TC twice and disarms. The counter reloads at each TC, by using the source selected by the gate. An ARM command is necessary to resume a new counting cycle after the second TC.

MODE V
Frequency-Shift Keying

Mode V and Mode S have identical gate operation, making the reload sources the Load register for a low gate and the Hold register for a high gate. After a counter is armed, it will count to TC repetitively. Using the source selected by the gate, the counter is reloaded at each TC. Until the counter is given a DISARM command, counting continues. Specifying a TC Toggled output mode in the Counter Mode register provides frequency shift keying. Modulation of the gate can be used to switch frequencies.

Terminal Count

As mentioned earlier, Terminal Count or TC is the length of time a counter's contents would have been zero if a reload had not occurred. There are some special conditions associated with the counter operation immediately before and during TC. They are as follows:

1. During the clock cycle prior to a TC, an internal signal is generated that makes the counter go to TC on the next count. Retriggering by a hardware Gate edge or issuing a Load or Load and Arm command does not extend the length of time until TC. However, a count source edge, a Load or Load and Arm command, or a Step command may cause the 'next count' that drives the counter to TC.

2. During the cycle that preceeds a TC, execution of a Load or Load and Arm command immediately forces the counter to TC. TC terminates if a Load or Load and Arm command is issued during the TC state.

3. With an active TC, the next source edge is always counted, regardless of whether or not the counter is disarmed or gated off during the TC.

Note: The contents of a counter after a TC will always differ by 1 from the value after reload. This is because the counter is always reloaded at the start of TC and it always counts at the end of TC.

Using the AM9513

The following figures will be helpful in doing the three examples in this secton.

Figure 3 Command Summary
Figure 4 Data Pointer Register
Figure 5 Master Mode Register Bit Assignments
Figure 6 Frequency Scaler Ratios
Figure 7 Counter Mode Register Bit Assignments

The examples are described step-by-step to help familiarize you with the use of the commands.

Notes:

1. All the examples use decimal numbers and the BASIC programming language.

2. For all the examples, XXX should be replaced by the control port address you have designated for control port, and YYY by the data port address, which is set by the address selection switches.

Using the Programmable Parallel Port I.C. (The 8255)

The 8255 is a general purpose programmable I/O device designed for implementing a variety of parallel port configurations. It has 24 I/O lines which are programmed in two groups of 12 lines and may be used in 3 major modes of operation. The 24 I/O lines form three 8-bit ports labelled A, B, and C. The use of these three ports depends on the mode being used. The three modes of operation are called Mode 0, Mode 1 and Mode 2.

In Mode 0 each group of 12 I/O lines may be programmed in a group of eight and a group of four, each independently available for input or output.

In Mode 1 each group of 12 I/O lines may be programmed to have eight lines of input or output. Of the remaining four lines, three are used for handshaking and interrupt control signals.

Mode 2 is a bidirectional bus mode which uses eight lines from one group of 12 for a bidirectional bus, and five lines, borrowing one from the other group, for handshaking.

The two groups of 12 I/O lines are formed by the lines of port A with half of port C for one group, and the lines of port B with the other half of port C for the second group. These are called group A and group B respectively.

The 8255 is controlled through four consecutive I/O locations with the IBM PC's Input or Output instructions. These locations are Port A, Port B, Port C and the Control port. A description of each of the ports is below.

Port A: This port is an Input and/or Output location for transferring 8-bit data between the I/O lines of Port A and the CPU. Port A is an Input port for Mode 0 and Mode 1 input, an Output port for Mode 0 and Mode 1 output, and an Input/Output port for Mode 2. Each of the 8 I/O lines of Port A is capable of sourcing 200 micro-amps and sinking 1.7 ma when in the output condition.

Port B: This port is an Input or Output location for transfering 8-bit data between the I/O lines of Port B and the CPU. Port B is an Input port for Mode 0 and Mode 1 input, and an Output port for Mode 0 and Mode 1 output. Port B cannot be used in Mode 2. Each of the 8 I/O lines of Port B is capable of sinking 1.7 ma and can source 1.0 ma at 1.5 volts. The high current source capability allows Port B to directly drive the Darlington type drivers used in some display devices.

Port C: This port is an Input or Output port that is connected to the I/O lines of Port C. In Mode 0 the lower four data bits (D0-D3) and the upper four data bits (D4-D7) are separately programmable as either input or output lines. In Mode 1 and Mode 2 some of the lines are dedicated as control or interrupt lines and the lines left over may be programmed as either input or output. The control lines and interrupt lines may be read (input) at any time. These I/O lines can source 200 micro-amps and sink 1.7 ma when functioning as outputs.

Control Port: This port is an Output only port used for controlling the 8255. If bit D7 of the 8 bit control word output to this port is a '1' then the rest of the control word selects the mode and input/output set-up. If D7 is a '0' the control word allows the direct setting to '1' or resetting to '0' of any bit in Port C. The complete format of the control word is described below.

With D7 = 1 (Mode definition format)

D6 and D5: Mode definition for group A. (Port A and control lines).

D6	D5	
0	0	Mode 0
0	1	Mode 1
1	0	Mode 2
1	1	Mode 2

D4: Port A definition

D4 = 1	Input
D4 = 0	Output

D3: Port C upper definition (C4-C7)

D3 = 1	Input
D3 = 0	Output

D2: Mode definition for group B (Port B and control lines)

D2 = 1	Mode 1
D2 = 0	Mode 0

D1: Port B definition

D1 = 1	Input
D1 = 0	Output

D0: Port C lower definition (C0-C3)

D0 = 1	Input
D0 = 0	Output

With D7 = 0 (Bit Set/Reset format)

D6, D5, D4: These bits have no function in this control word and their setting has no effect.

D3, D2, D1: These three bits select the bit of Port C that will be set or reset.

D3	D2	D1	Port C Bit Selected
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

D0: This bit selects whether the bit of Port C will be set to a '1' or reset to a '0'.

D0 = 1	Set
D0 = 0	Reset

Upon power-up the IBM Personal Computer power-on reset line resets the 8255 to Mode 0 with all I/O lines set as inputs. The 8255 could be used this way without further programming. For other set-ups the mode and configuration will have to be programmed by sending the correct Mode definition command to the Control Port. A complete description of the three modes follows.

Mode 0 (Basic Input/Output):

This configuration provides simple input and output operations for each of the three ports. No 'handshaking' is required; data is simply written to or read from a specified port. The data bits D0-D7 correspond directly to the I/O lines A0-A7, B0-B7 and C0-C7.

Mode 0 Basic Functional Definitions

1. Two 8 bit ports and two 4 bit ports.
   Port A (A0-A7)
   Port B (B0-B7)
   Port C upper (C4-C7)
   Port C lower (C0-C3)

2. Any port can be input or output.

3. Outputs are latched.

4. Inputs are not latched.

5. 16 different Input/Output configurations are possible in this mode.

The command output to the Control Port for any of the 16 configurations may be chosen from the following table. In the table I = input, O = output, and all commands are given in hexadecimal.

Command	Port A	Port B	Port C lower	Port C upper
80	0	0	0	0
81	0	0	I	0
82	0	I	0	0
83	0	I	I	0
88	0	0	0	I
89	0	0	I	I
8A	0	I	0	I
8B	0	I	I	I
90	I	0	0	0
91	I	0	I	0
92	I	I	0	0
93	I	I	I	0
98	I	0	0	I
99	I	0	I	I
9A	I	I	0	I
9B	I	I	I	I