P R L

T E C H N I C A L R E P O R T

TN-90-70

AN ASSEMBLY LANGUAGE PROGRAM FOR IMPLEMENTATION OF FAST FOURIER TRANSFORM ON 8086/8088 MICROPROCESSOR

R. N. MISRA

&

AKHILESH KUMAR*

NOVEMBER 1990

PHYSICAL RESEARCH LABORATORY
NAVRANGPURA
AHMEDABAD 380 009

*IIT, Kharagpur - 721 302.

DOCUMENT CONTROL AND DATA SHEET

1.	Report No.	:	PRL-TN-90-70			
2.	Title and subtitle An Assembly Language Program for Implementation of Fast Fourier Transform on 8086/8088 Microprocessor	3.	Report Date	:	November 1990
		4.	Type of Report	:	Technical Report
		5.	Pages	:	26
		6.	Price	:	Unpriced
		7.	No. of References		9
8.	Author(s)	:	R.N. Misra and Akhilesh Kumar		
9.	A. Purpose	:	Details of FFT software for 8086		
	B. Useful for	:	Incorporating the above in other software modules		
10.	Originating Unit	:	Planetary Atmospheres and Aeronomy Area, Physical Research Laboratory Navrangpura, Ahmedabad 380 009		
11.	Sponsoring Agency	:	None		
12.	Abstract	:	FFT is widely used for estimation of spectrum of sampled data. Assembly language FFT program for 8086 can be embedded in other programs for real time applications		
13.	Key Words	:	Fast Fourier Transform, 8086/8088, Real Time		
14.	Distribution statement	:	Distribution unlimited		
15.	Security Classification	:	Unclassified		

AN ASSEMBLY LANGUAGE PROGRAM FOR IMPLEMENTATION OF FAST FOURIER TRANSFORM ON 8086/8088 MICROPROCESSOR

R.N.Misra and Akhilesh Kumar

ABSTRACT

Fast Fourier Transform (FFT) is a powerul technique used for digital signal processing .While off-line FFT can easily be performed on recorded data with the help of standard software,the real time FFT requires optimisation of hardware as well as software.Large number of multiplications and additions required in computation of FFT necessitate use of special multiplier accumulator hardware for realtime applications.Modern 16 bit microprocessors include instructions for binary multiplication which can be effectively utilized for real time FFT.A program written in the machine code of the microprocessor can be optimised to increase the speed of computation,and retain the resolution of the output data.The 256 point complex FFT implementation described here takes about 350 milliseconds on PC-XT operating at 4.77 MHz and 50 milliseconds on PC-AT running at 8 MHz.

KEYWORDS:Fast Fourier Transform,8086/8088,Real time.

Physical Research Laboratory,Navrangpura,Ahmedabad 380009 India.

Indian Institute of Technology,Kharagpur-721302,India

INTRODUCTION

Amplitude versus time plot of a time varying parameter does not give much information about its spectral characteristics.The fourier transform of a time varying signal maps it in the frequency domain and can be conveniently used for this purpose.Estimation of fourier transform is governed by the following equation;

$$S(w)=\int_{-\infty}^{\infty} s(t)\exp(-jwt)\ dt \quad \ldots\ldots\ldots\ldots\ldots\ldots(1)$$

Where $w = 2\pi f$, f being frequency

s(t)=The time varying signal under discussion

and S(w)= Fourier transform of s(t)

Though the real life quantities ,or their electrical analogs,are most often continuous in time ,the processing of such signals is being increasingly done with the help of digital hardware.The signals are therefore invariably sampled and only limited number of samples are available for extraction of further information.Number of samples is chosen to satisfy the requirement of the sampling theorem.The discrete form of the fourier transform ,the DFT, therefore,comes into picture.Now we are dealing with a variable x which can have n discrete values;

$$x(n)= x_{\emptyset} ,x_{1} ,x_{2} \ldots\ldots\ x_{N-1} \quad \ldots\ldots\ldots\ldots\ldots\ldots(2)$$

and we intend to estimate fourier transform of the above signal x(t) from the N samples .The integration operation is substituted by summation and a discrete fourier transform(DFT) having a set of values X(n) can be defined as follows;

$$x(n) = 1/N \sum_{k=0}^{N-1} X(k) \exp(j2\pi/N)kn \quad \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Where the coefficients X(k) are given by

$$X(k) = \sum_{n=0}^{N-1} x(n) \exp(-j2\pi/N)kn \quad \ldots\ldots\ldots\ldots\ldots\ldots (4)$$

The quantity $\exp(-j2\pi/N)$ is written as W to save space and the above equation takes the form

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W^{nk} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

The DFT can always be calculated with the help of the above equation but it would require NXN complex multiplications and N(N-1) complex additions and some means therfore need to be devised to save the computation time.

The fast fourier transform (FFT) is an algorithm to compute DFT, which reduces the number of multiplications to $N \log_2 N$, and therefore gives substantial time saving.It is,therfore,extensively used in spectral analysis of sampled data.The number of samples available for FFT operation are dictated by the required time resolution and the limitations posed by the hardware.Dyvik[1,2] and others have described a program to compute FFT of 256 data points .It was written for BBC Micro in BASIC and rewritten in machine code of 65Ø2 to reduce time of execution to about one second.Weysel Omer[3,4] devised a machine code program to implement FFT for a set of 128 data points with an accuracy of 32 bits on the same microprocessor.However the 65Ø2 does not have instructions for multiplication in its instruction repertoire and further speedup may not be possible.The present day 16 bit microprocessors like 8086 and 68000 do have instructions for multiplication of 16 bit data and storage of 32 bit result and can therefore be effectively utilised for FFT. The authors chose 8086 for its wide availability in the form of IBM PC compatibles.

THE PROGRAM

The off-line computation of FFT is extensively used and exellent programs are available in the literature. Burrus & Parks have described a large number of such programs optimised to different degree. Most of the programs described therein are in FORTRAN language and some in the assembly language of the 34010 digital signal processor of Texas Instruments. The authors have taken one of the FORTRAN programs(ibid ,page 59), as the basis for the development of machine code program for 8086. The original program has already been optimised to some extent and any further optimization is not very effective in reduction of computation time(ibid. page 60). Fig.1 shows the signal flow graph of a Decimation in Frequency decomposition of 8 point DFT ,which forms the basis of this program. The flowchart of the program written in the machine code of 8086 has been given in the fig.2 .

The 8086 microprocessor can handle 16 bit data with ease. Most of the real world signals are usually available in 10 to 12 bit resolution and can therefore be easily accomodated within 16 bit word. Two's complement form of data further reduces processing time as the signs of the intermediate results need not be adjusted at every step.The input data is scaled up to occupy the more significant bits to increase the accuracy of the result.For example 8 bit data is shifted 6 positions left to keep the data within 16 bits after one stage of operation.The sine and cosine tables are generated in binary form and +1 is normalised to 32767(7FFFH).Thus the coefficients are as large as practical and the results of multiplication and addition are shifted by one bit right at every step to avoid overflow and truncation in the subsequent operation.As the sines and cosines of one particular angle are used in one step,the sine cosine table has been generated in such a way that the values of cosine and sine of one angle are put together in four bytes as one table entry.This requires only one pointer and thus reduces the time of computation to some extent.The table starts with the angle 2π and arguments are decremented in the steps of $2\pi/256$ degrees. The program in the assembly language of 8086 and the trigoometric table have been given in appendices I and II respectively.

WINDOW FUNCTION

Calculation of fourier transform requires amplitude of the signal at all times.Since any measurement is for finite duration,it results in truncation of the signal in time domain.Truncation of the input signal in time brings inaccuracy in estimation of fourier transform.However the signal may be assumed to be periodic which repeats itself beyond the time of observation .Or else it may be assumed to be zero everywhere beyond the region of observation.The signal is often multiplied by a window function to isolate a zone. Estimate of DFT in such cases is the convolution of the DFT of the signal with that of the window function.A suitable window function can be chosen to multiply every sample of the signal before estimating the DFT to reduce the smearing effect caused by the rectangular window ,i.e. when the samples of the signal are used for estimation of DFT as it is. Window functions are symmetric and have side lobes lower than that of the basic rectangular window function.Triangular,Hamming and Hanning are some of the window functions and the ripple in the spectrum estimate gets reduced by their use.Multiplication with the window funcion would involve N more multiplications over and above $N \log_2 N$ required by FFT algorithm.Thus a drop of about 15 per cent in speed may be expected for a 256 point FFT.Reduction gets progressively lower for larger number of points .

ESTIMATION OF POWER SPECTRUM

The DFT is a map of the time domain input sequence(discrete set of data points) in to the frequency domain and is called its linear spectrum.Each element of the DFT has both real as well as imaginary parts which depend upon position of the signal.The square of the amplitude of the linear spectrum is called the power spectrum and may be obtained by squaring and adding the real and imaginary parts of the each element of the DFT .In fact it is more correctly called a periodogram and is only an approximation to the real power spectrum of the input sequence.The power spectrum is widely used for various signal processing applications.

PROGRAM USAGE

The program uses separate file for cosine and sine tables which is copied into RAM at the beginning.The data file is also copied into separate RAM area in the next step.The main program computes the FFT , unscrambles the results, and places the same into the RAM space that contained the input data .The unscrambling operation is done by a separate subroutine.Each of the data points in the data file consist of 8 bit real and 8 bit imaginary parts and the resultant FFT has 16 bit real and imaginary parts .The amplitude of each point is computed by a subroutine to facilitate generation of data points of spectrum.These may be plotted by usual plotting techniques available in the PC.

This program has been tested by computing the spectrum of artificially generated data comprising of single sine wave,sum of two sine waves,square wave as well as scintillation data .No window funtion has however been used .The input waveforms as well as computed spectra,plotted on linear amplitude as well as time/frequency scale, have been given in Figs 3 to 12A.

CONCLUSION

An assembly language program to compute 256 point complex FFT on 8086/8088 microprocessor has been written .It takes about 350 milliseconds on 8088 running at 4.77 MHz , 50 milliseconds on 80286 at 8 MHz and can therefore be used for real time signal processing applications.

ACKNOWLEDGEMENTS

The authors are thankful to Professors Satya Prakash and B.H.Subbaraya for their encouragement during the development of the software.The authors had helpful discussions with Dr.Harish Chandra who also provided the information about the basic input/output requirements of signal processing and sample ionospheric scintillation data and its spectrum to test the program.

References

1. G K Dyvik, Fast Fourier Transform, Electronics & Wireless World, Vol.91, No.1598, December 1985, pp.74-75.

2. T Larsen & G K Dyvik, Fast Fourier Transforms using a micro-computer, Electronics & Wireless World, Vol.91, No.1595, September 1985, pp.80-82.

3. Weysel Omer, Faster Fourier Transforms, Electronics & Wireless World, Vol.92, No.1604, June 1986, pp.23-25.

4. Weysel Omer, Faster Fourier Transforms, Electronics & Wireless World, Vol.92, No.1605, July 1986, pp.57-58.

5. C S Burrus & T W Parks, DFT/FFT and convolution algorithms, Theory & Implementation (Topics in Digital Signal Processing), A Wiley Interscience Publication, John Wiley & Sons, Texas Instruments, 1985.

6. S E Georgeoura, Fast Fourier Transforms of sampled waveforms, Electronics & Wireless World, Vol.94, No.1633, November 1988, pp.1122-1126.

7. E O Brigham, The Fast Fourier Transform, Prentice Hall, Englewood Cliffs, New Jersey, 1974.

8. William D Stanley, Digital Signal Processing, Reston Publishing Co. Inc., Reston Virginia 22090, USA, 1975.

9. Alan V Oppenheim & Ronald W Schafer, Digital Signal Processing Prentice Hall, Englewood Cliffs, New Jersey, 1975.

APPENDIX I

```
;This program generates DFT from 256 discrete points of data
;Data is replaced by DFT

DATA_HERE       SEGMENT

POINTS        EQU 256
POWER         EQU 8
EVEN
TEMP           DW 1Ø DUP(Ø)
SQRL           DW ØØØØH
SQRH           DW ØØØØH
TRTAB          DW 4ØØH DUP(Ø)          ;TRIGONOMETRIC TABLE
DPTR           DW 4ØØH DUP(Ø)          ;DATA POINTER
RPTR           DW 2ØØH DUP(Ø)          ;RESULT POINTER
START          DB 'FFT Program starts',ØDH,ØAH,24H
DECIDE         DB 'Press P to proceed or Q to quit',ØDH,ØAH,24H
TR_FILE        DB 'Give trigonometric data file',ØDH,ØAH,24H
IN_FILE        DB 'Give input data file',ØDH,ØAH,24H
OUT_FILE       DB 'Give result file name',ØDH,ØAH,24H
NEXT           DB 'Do you want to proceed with another file',ØDH,ØAH
               DB 'If yes press P else press Q to quit',ØDH,ØAH,24H
FILENAME       DB 4Ø DUP(Ø)
READ_BUF       DB 4ØØH DUP(Ø)
ERR_PTR        DW Ø
               DW OFFSET ERR_MESS1
               DW OFFSET ERR_MESS2
               DW OFFSET ERR_MESS3
ERR_MESS1      DB 'INVALID FUNCTION NUMBER',ØDH,ØAH,24H
ERR_MESS2      DB 'FILE NOT FOUND',ØDH,ØAH,24H
ERR_MESS3      DB 'PATH NOT FOUND',ØDH,ØAH,24H

DATA_HERE       ENDS

STACK_HERE      SEGMENT

STACK     DW      1ØØ DUP(Ø)
ST_TOP    LABEL   WORD

STACK_HERE      ENDS

CODE_HERE       SEGMENT
                SM_SYS GROUP CODE_HERE, DATA_HERE, STACK_HERE
                ASSUME CS:SM_SYS, DS:SM_SYS, SS:SM_SYS
                MOV     AX,SM_SYS
                MOV     DS,AX
                MOV     SS,AX
                MOV     ES,AX
                MOV     SP,OFFSET SM_SYS:ST_TOP   ;Initialize stack.
                MOV     DX,OFFSET SM_SYS:START
                MOV     AH,Ø9H
                INT     21H
                MOV     DX,OFFSET SM_SYS:TR_FILE  ;Ask for trigonometric file
                MOV     AH,Ø9H
                INT     21H
                MOV     CX,POINTS                 ;File length of trigonometr
```

```
         SHL     CX,1                         ;data is 4*POINTS containg
         SHL     CX,1                         ;16 bit Re & Im values.
         CALL    LOAD
         MOV     CX,AX
         MOV     SI,OFFSET SM_SYS:READ_BUF    ;Copy the contents of trig-
         MOV     DI,OFFSET SM_SYS:TRTAB       ;-onometric file from read
         CLD                                  ;buffer to TRTAB.
    REP  MOVSB
         MOV     DX,OFFSET SM_SYS:DECIDE      ;Ask to proceed or quit.
         MOV     AH,Ø9H
         INT     21H
BACK:    MOV     AH,Ø
         INT     16H
         CMP     AL,51H                       ;If it is 'Q' or 'q' then quit.
         JE      QUIT
         CMP     AL,71H
         JE      QUIT
         CMP     AL,5ØH                       ;If it is 'P' or 'p' then
         JE      DO_IT                        ;read input file name.
         CMP     AL,7ØH
         JE      DO_IT
         JMP     BACK
QUIT:    MOV     AX,4CØØH                     ;Terminate the process and
         INT     21H                          ;exit to the calling procedure.
         NOP
DO_IT:   MOV     DX,OFFSET SM_SYS:IN_FILE     ;Ask for input file name.
         MOV     AH,Ø9H
         INT     21H
         MOV     CX,POINTS                    ;Input file length 2*POINTS
         SHL     CX,1                         ;containing both Re & Im data.
         CALL    LOAD
         MOV     CX,AX                        ;Store filelength in cx.
         MOV     DI,OFFSET SM_SYS:DPTR
         MOV     SI,OFFSET SM_SYS:READ_BUF
ADJST:   MOV     AX,Ø                         ;Adjust the input 8 bit data
         MOV     AL,[SI]                      ;by shifting it left 6 times.
         PUSH    CX
         MOV     CX,6
         SHL     AX,CL
         POP     CX
         INC     SI
         MOV     [DI],AX
         INC     DI
         INC     DI
         LOOP    ADJST
         MOV     CX,POINTS                    ;N=256
         MOV     AX,POWER                     ;M=8
         MOV     DX,ØØØ2H                     ;IE=1/2
         MOV     BX,ØØØ1H                     ;K=1
NEXTK:   PUSH    AX
         SHL     DX,1                         ;IE=2IE
         MOV     SI,OFFSET SM_SYS:TRTAB       ;IA=1
         MOV     BP,CX                        ;N1=N2
         SHR     CX,1                         ;N2=N2/2
         PUSH    CX
```

```
        PUSH    DX
        PUSH    BX
        MOV     AX,ØØØ1H                    ;J=1
NEXTJ:  PUSH    AX
        PUSH    AX
        MOV     BX,CX                       ;BX=N2
        PUSH    SI
        MOV     AX,[SI]                     ;AX=WR(IA)
        MOV     TEMP,AX                     ;TEMPØ=WR(IA)=C
        INC     SI
        INC     SI
        MOV     AX,[SI]
        MOV     TEMP+2,AX                   ;TEMP1=WI(IA)=S
        POP     SI
        ADD     SI,DX                       ;IA=IA+IE
        POP     AX                          ;AX=J
        CALL    MAIN
        POP     AX
        INC     AX                          ;Increment J.
        CMP     AX,CX                       ;Is J=<N2? If yes do another
        JLE     NEXTJ                       ;iteration.
        POP     BX
        POP     DX
        POP     CX
        INC     BX                          ;Increment K.
        POP     AX
        CMP     BX,AX                       ;If K=<M,do another iteration
        JLE     NEXTK                       ;otherwise FFT complete.
        CALL    UNSCR                       ;Unscramble the result.
        CALL    AMP                         ;Absolute values of result.
        CALL    MAXIM                       ;Maximize the result.
        JMP     WRITE                       ;Store result in a file.
LOAD:   MOV     DX,OFFSET SM_SYS:FILENAME   ;For loading the file.
        MOV     FILENAME,4Ø                 ;Max character for file name=4Ø.
        MOV     AH,ØAH                      ;Read input filename from
        INT     21H                         ;keyboard.
        MOV     DL,ØDH                      ;Output <cr> and linefeed to
        MOV     AH,Ø2H                      ;the display.
        INT     21H
        MOV     DL,ØAH
        MOV     AH,Ø2H
        INT     21H
        MOV     BL,FILENAME+1               ;Make last character in file
        ADD     BL,Ø2                       ;name read as Ø.
        MOV     BH,ØØ
        MOV     FILENAME[BX],ØØ
        MOV     DX,OFFSET SM_SYS:FILENAME
        ADD     DX,Ø2H
        MOV     AL,Ø
        MOV     AH,3DH                      ;Open the specified file to read
        INT     21H
heck for file error
        JNC     OK                          ;If file error then give
        ROL     AX,1                        ;error message.
        MOV     BX,AX
```

```
        MOV     DX,OFFSET SM_SYS:ERR_PTR[BX]
        MOV     BH,Ø
        MOV     AH,Ø9H
        INT     21H
        MOV     DL,ØDH
        MOV     AH,Ø2H
        INT     21H
        JMP     BACK
OK:     MOV     BX,AX                        ;Otherwise get filehandle & save
        PUSH    BX                           ;it for closing the same file.
        MOV     DX,OFFSET SM_SYS:READ_BUF
        MOV     AH,3FH                       ;Read the input file upto eof or
        INT     21H                          ;max characters specified in cx
        POP     BX                           ;and store in READ_BUF.
        PUSH    AX                           ;Save filelength.
        MOV     AH,3EH                       ;Getback filehandle and
        INT     21H                          ;close the file.
        POP     AX
        RET
MAIN:   SHL     BX,1
        SHL     BX,1                         ;BX=N2
NEXTI:  PUSH    AX
        SHL     AX,1
        SHL     AX,1
        ADD     AX,OFFSET SM_SYS:DPTR        ;AX=I
        SUB     AX,4
        MOV     DI,AX
        MOV     AX,[DI]                      ;X(I)
        PUSH    AX
        SUB     AX,[BX+DI]                   ;X(I)-X(L) : L=I+N2
        MOV     TEMP+4,AX                    ;TEMP2=X(I)-X(L)
        POP     AX
        ADD     AX,[BX+DI]                   ;X(I)+X(L)
        SAR     AX,1
        MOV     [DI],AX                      ;X(I)=[X(I)+X(L)]/2
        MOV     AX,[DI+Ø2]                   ;Y(I)
        PUSH    AX
        SUB     AX,[BX+DI+Ø2]
        MOV     TEMP+6,AX                    ;TEMP3=Y(I)-Y(L)
        POP     AX
        ADD     AX,[BX+DI+Ø2]
        SAR     AX,1
        MOV     [DI+Ø2],AX                   ;Y(I)=[Y(I)+Y(L)]/2
        PUSH    DX
        PUSH    CX
        PUSH    BX
        MOV     AX,TEMP
        IMUL    WORD PTR TEMP+4              ;TEMPØ*TEMP2
        PUSH    AX
        PUSH    DX
        MOV     AX,TEMP+2
        IMUL    WORD PTR TEMP+6              ;TEMP1*TEMP3
        MOV     BX,AX
        MOV     CX,DX
        POP     DX
```

```
        POP     AX
        SUB     AX,BX                   ;TEMPØ*TEMP2-TEMP1*TEMP3
        SBB     DX,CX
        PUSH    DX
        MOV     AX,TEMP+2
        IMUL    WORD PTR TEMP+4         ;TEMP1*TEMP2
        PUSH    DX
        PUSH    AX
        MOV     AX,TEMP
        IMUL    WORD PTR TEMP+6         ;TEMPØ*TEMP3
        MOV     BX,AX
        MOV     CX,DX
        POP     AX
        POP     DX
        ADD     AX,BX                   ;TEMP1*TEMP2+TEMPØ*TEMP3
        ADC     DX,CX
        MOV     CX,DX
        POP     DX
        SAR     DX,1
        SAR     CX,1
        POP     BX
        MOV     [BX+DI+Ø2],CX           ;Y(L)=[TEMP1*TEMP2+TEMPØ*TEMP3]/
        MOV     [BX+DI],DX              ;X(L)=[TEMPØ*TEMP2+TEMP1*TEMP3]/
        POP     CX
        POP     DX
        POP     AX
        ADD     AX,BP                   ;I=I+N1
        CMP     AX,POINTS               ;IF I<=N THEN REPEAT
        JLE     NEXTI
        RET
Subroutine for unscrambling the result of FFT from in place calculation.
UNSCR:  MOV     CX,POINTS               ;No. of points to unscramble.
        MOV     BX,Ø                    ;Set pointer at the start.
CONTD:  PUSH    CX                      ;Save no. of points to be
        PUSH    BX                      ;unscrambled & the pointer.
        PUSH    BX
        MOV     CX,8                    ;Reverse the order of the bits
ROTATE: RCL     BL,1                    ;in BL,which contains the
        RCR     BH,1                    ;pointer,and give the result
        LOOP    ROTATE                  ;in BH.
        MOV     CL,BH
        POP     BX                      ;If pointer is greater than
        CMP     BL,CL                   ;or equal to the bit reversed
        JGE     NEXT1                   ;pointer,data is not inter-
        SHL     CX,1                    ;-changed.
        SHL     CX,1                    ;Multiply both the pointers by
        SHL     BX,1                    ;4 to indicate actual memory
        SHL     BX,1                    ;location.
        MOV     DX,BX                   ;Interchange the data at the
        MOV     AX,DPTR[BX]             ;memory locations pointed by
        MOV     BX,CX                   ;the two pointers.First inter-
        XCHG    AX,DPTR[BX]             ;-change real part of data.
        MOV     BX,DX
        MOV     DPTR[BX],AX
        INC     BX                      ;Increment pointers to the
```

```
        INC     BX                      ;imaginary part of data.
        INC     CX
        INC     CX
        MOV     DX,BX                   ;Interchange imaginary parts.
        MOV     AX,DPTR[BX]
        MOV     BX,CX
        XCHG    AX,DPTR[BX]
        MOV     BX,DX
        MOV     DPTR[BX],AX
NEXT1:  POP     BX
        INC     BX                      ;Increment pointer.
        POP     CX
        LOOP    CONTD
        RET
ubroutine for amplitude of complex quantities.
AMP:    MOV     CX,POINTS               ;This subroutine takes the
        MOV     SI,OFFSET SM_SYS:DPTR   ;sqare of 16 bit Re & Im parts
        MOV     DI,OFFSET SM_SYS:RPTR   ;of complex number and after
REP:    MOV     AX,[SI]                 ;adding them takes the square
        MOV     BX,AX                   ;root of the sum by successive
        IMUL    BX                      ;approximation.
        PUSH    DX
        PUSH    AX
        INC     SI
        INC     SI
        MOV     AX,[SI]
        MOV     BX,AX
        IMUL    BX
        MOV     BX,AX
        POP     AX
        ADD     AX,BX
        MOV     BX,DX
        POP     DX
        ADC     DX,BX
        INC     SI
        INC     SI
        PUSH    CX
        MOV     SQRL,AX
        MOV     SQRH,DX
        MOV     BX,4ØØØH
        MOV     CX,BX
CONT:   MOV     AX,BX
        IMUL    BX
        CMP     SQRH,DX
        JL      RESET
        JZ      LOWER
MODIFY: SHR     CX,1
        MOV     AX,ØØØØH
        NOT     AX
        AND     AX,CX
        JZ      STORE
        OR      BX,CX
        JMP     CONT
STORE:  MOV     [DI],BX
        INC     DI
```

```
        INC     DI
        POP     CX
        LOOP    REP
        RET
RESET:  NOT     CX
        AND     BX,CX
        NOT     CX
        JMP     MODIFY
LOWER:  CMP     SQRL,AX
        JL      RESET
        JMP     MODIFY
WRITE:  MOV     DX,OFFSET SM_SYS:OUT_FILE
        MOV     AH,Ø9H                     ;Ask for the output file name.
        INT     21H
        MOV     DX,OFFSET SM_SYS:FILENAME
        MOV     FILENAME,4Ø
        MOV     AH,ØAH
        INT     21H                        ;Read output file name.
        MOV     DL,ØDH                     ;Output <cr> & linefeed to the
        MOV     AH,Ø2H                     ;display.
        INT     21H
        MOV     DL,ØAH
        MOV     AH,Ø2H
        INT     21H
        MOV     BL,FILENAME+1              ;Make the last character of
        ADD     BL,Ø2H                     ;the file name ØØ.
        MOV     BH,ØØ
        MOV     FILENAME[BX],ØØ
        MOV     DX,OFFSET SM_SYS:FILENAME
        ADD     DX,Ø2H
        MOV     CX,ØØH                     ;File attribute.
        MOV     AH,3CH                     ;Create file for write.
        INT     21H
        JNC     YES                        ;If error occurs give message.
        ROL     AX,1
        MOV     BX,AX
        MOV     DX,OFFSET SM_SYS:ERR_PTR[BX]
        MOV     AH,Ø9H
        INT     21H
        MOV     DL,ØDH
        MOV     AH,Ø2H
        INT     21H
        JMP     BACK
YES:    MOV     BX,AX                      ;Get filehandle and save it.
        PUSH    BX
        MOV     DX,OFFSET SM_SYS:RPTR
        MOV     CX,2ØØH                    ;Write data of RPTR in the
        MOV     AH,4ØH                     ;output file.
        INT     21H
        POP     BX
        MOV     AH,3EH                     ;Close output file.
        INT     21H
        MOV     DX,OFFSET SM_SYS:NEXT      ;Ask if another file is to be
        MOV     AH,Ø9H                     ;proceeded with FFT.
        MOV     BH,Ø
```

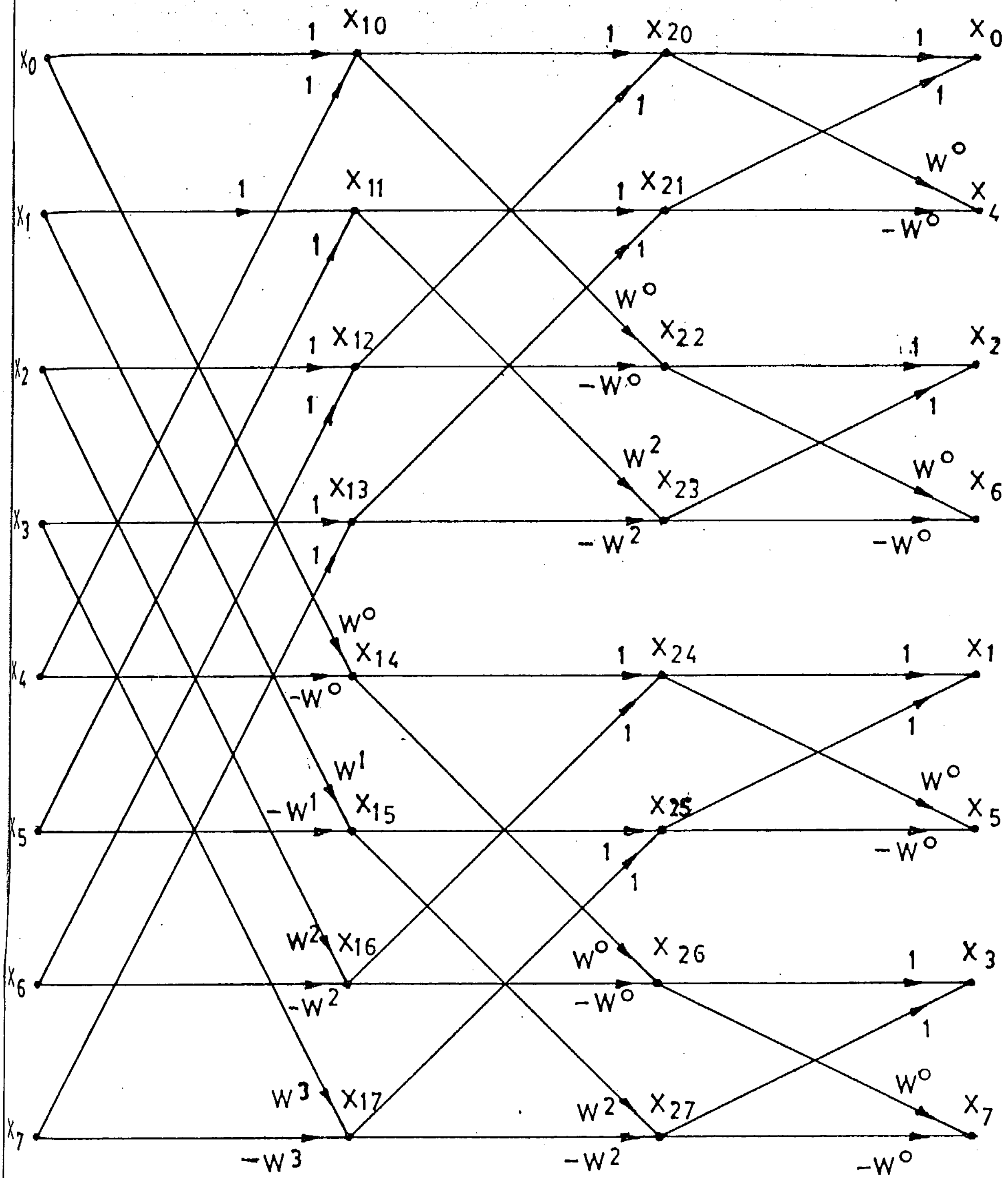
```
        INT     21H
        JMP     BACK
Subroutine for maximizing the result to facilitate plotting
MAXIM:  MOV     SI,OFFSET SM_SYS:RPTR
        MOV     DI,SI
        INC     SI
        INC     SI
        MOV     CX,POINTS
        SUB     CX,1
        SHL     CX,1
        PUSH    CX
        MOV     AX,ØØ
DO:     OR      AX,[SI]
        INC     SI
        INC     SI
        LOOP    DO
        MOV     BX,ØØ
        STC
        RCL     AX,1
ROT:    RCL     AX,1
        INC     BX
        JNC     ROT
        POP     CX
OTHER:  MOV     DX,CX
        MOV     CX,BX
        MOV     AX,[DI]
        SHL     AX,CL
        MOV     [DI],AX
        INC     DI
        INC     DI
        MOV     CX,DX
        LOOP    OTHER
        RET

DE_HERE ENDS
        END
```

APPENDIX II

TRIGONOMETRIC TABLE

Values of Cosine and Sine have been entered as 2's complement values in sequence each occupying two bytes. First entry is Cos 2 π and the last Sin 2 π /256. The argument is decremented by 2 π /256 in every step. Values are normalized to 7 FFF(H) = +1 and 8001(H) = -1.

```
ØBFD:Ø1ØØ  FF 7F ØØ ØØ F5 7F DC FC-D8 7F B8 F9 A6 7F 96 F6
ØBFD:Ø11Ø  61 7F 74 F3 Ø9 7F 55 FØ-9C 7E 38 ED 1D 7E 1E EA
ØBFD:Ø12Ø  89 7D Ø7 E7 E3 7C F5 E3-29 7C E6 EØ 5C 7B DD DD
ØBFD:Ø13Ø  7C 7A D8 DA 89 79 DA D7-84 78 E1 D4 6B 77 EF D1
ØBFD:Ø14Ø  41 76 Ø5 CF Ø4 75 21 CC-B5 73 46 C9 54 72 74 C6
ØBFD:Ø15Ø  E2 7Ø AA C3 5E 6F E9 CØ-C9 6D 32 BE 23 6C 86 BB
ØBFD:Ø16Ø  6D 6A E4 B8 A6 68 4C B6-CF 66 C1 B3 E8 64 41 B1
ØBFD:Ø17Ø  F1 62 CD AE EB 6Ø 65 AC-D7 5E ØB AA B3 5C BE A7
ØBFD:Ø18Ø  82 5A 7E A5 42 58 4D A3-F5 55 29 A1 9B 53 15 9F
ØBFD:Ø19Ø  33 51 ØF 9D BF 4E 18 9B-3F 4C 31 99 B4 49 5A 97
ØBFD:Ø1AØ  1C 47 93 95 7A 44 DD 93-CE 41 37 92 17 3F A2 9Ø
ØBFD:Ø1BØ  56 3C 1E 8F 8C 39 AC 8D-BA 36 4B 8C DF 33 FC 8A
ØBFD:Ø1CØ  FB 3Ø BF 89 11 2E 95 88-1F 2B 7C 87 26 28 77 86
ØBFD:Ø1DØ  28 25 84 85 23 22 A4 84-1A 1F D7 83 ØB 1C 1D 83
ØBFD:Ø1EØ  F9 18 77 82 E2 15 E3 81-C8 12 64 81 AB ØF F7 8Ø
ØBFD:Ø1FØ  8C ØC 9F 8Ø 6A Ø9 5A 8Ø-48 Ø6 28 8Ø 24 Ø3 ØB 8Ø
ØBFD:Ø2ØØ  ØØ ØØ Ø1 8Ø DC FC ØB 8Ø-B8 F9 28 8Ø 96 F6 5A 8Ø
ØBFD:Ø21Ø  74 F3 9F 8Ø 55 FØ F7 8Ø-38 ED 64 81 1E EA E3 81
ØBFD:Ø22Ø  Ø7 E7 77 82 F5 E3 1D 83-E6 EØ D7 83 DD DD A4 84
ØBFD:Ø23Ø  D8 DA 84 85 DA D7 77 86-E1 D4 7C 87 EF D1 95 88
ØBFD:Ø24Ø  Ø5 CF BF 89 21 CC FC 8A-46 C9 4B 8C 74 C6 AC 8D
ØBFD:Ø25Ø  AA C3 1E 8F E9 CØ A2 9Ø-32 BE 37 92 86 BB DD 93
ØBFD:Ø26Ø  E4 B8 93 95 4C B6 5A 97-C1 B3 31 99 41 B1 18 9B
ØBFD:Ø27Ø  CD AE ØF 9D 65 AC 15 9F-ØB AA 29 A1 BE A7 4D A3
ØBFD:Ø28Ø  7E A5 7E A5 4D A3 BE A7-29 A1 ØB AA 15 9F 65 AC
ØBFD:Ø29Ø  ØF 9D CD AE 18 9B 41 B1-31 99 C1 B3 5A 97 4C B6
ØBFD:Ø2AØ  93 95 E4 B8 DD 93 86 BB-37 92 32 BE A2 9Ø E9 CØ
ØBFD:Ø2BØ  1E 8F AA C3 AC 8D 74 C6-4B 8C 46 C9 FC 8A 21 CC
ØBFD:Ø2CØ  BF 89 Ø5 CF 95 88 EF D1-7C 87 E1 D4 77 86 DA D7
ØBFD:Ø2DØ  84 85 D8 DA A4 84 DD DD-D7 83 E6 EØ 1D 83 F5 E3
ØBFD:Ø2EØ  77 82 Ø7 E7 E3 81 1E EA-64 81 38 ED F7 8Ø 55 FØ
ØBFD:Ø2FØ  9F 8Ø 74 F3 5A 8Ø 96 F6-28 8Ø B8 F9 ØB 8Ø DC FC
ØBFD:Ø3ØØ  Ø1 8Ø ØØ ØØ ØB 8Ø 24 Ø3-28 8Ø 48 Ø6 5A 8Ø 6A Ø9
ØBFD:Ø31Ø  9F 8Ø 8C ØC F7 8Ø AB ØF-64 81 C8 12 E3 81 E2 15
ØBFD:Ø32Ø  77 82 F9 18 1D 83 ØB 1C-D7 83 1A 1F A4 84 23 22
ØBFD:Ø33Ø  84 85 28 25 77 86 26 28-7C 87 1F 2B 95 88 11 2E
ØBFD:Ø34Ø  BF 89 FB 3Ø FC 8A DF 33-4B 8C BA 36 AC 8D 8C 39
ØBFD:Ø35Ø  1E 8F 56 3C A2 9Ø 17 3F-37 92 CE 41 DD 93 7A 44
ØBFD:Ø36Ø  93 95 1C 47 5A 97 B4 49-31 99 3F 4C 18 9B BF 4E
ØBFD:Ø37Ø  ØF 9D 33 51 15 9F 9B 53-29 A1 F5 55 4D A3 42 58
ØBFD:Ø38Ø  7E A5 82 5A BE A7 B3 5C-ØB AA D7 5E 65 AC EB 6Ø
ØBFD:Ø39Ø  CD AE F1 62 41 B1 E8 64-C1 B3 CF 66 4C B6 A6 68
ØBFD:Ø3AØ  E4 B8 6D 6A 86 BB 23 6C-32 BE C9 6D E9 CØ 5E 6F
ØBFD:Ø3BØ  AA C3 E2 7Ø 74 C6 54 72-46 C9 B5 73 21 CC Ø4 75
ØBFD:Ø3CØ  Ø5 CF 41 76 EF D1 6B 77-E1 D4 84 78 DA D7 89 79
ØBFD:Ø3DØ  D8 DA 7C 7A DD DD 5C 7B-E6 EØ 29 7C F5 E3 E3 7C
ØBFD:Ø3EØ  Ø7 E7 89 7D 1E EA 1D 7E-38 ED 9C 7E 55 FØ Ø9 7F
ØBFD:Ø3FØ  74 F3 61 7F 96 F6 A6 7F-B8 F9 D8 7F DC FC F5 7F
ØBFD:Ø4ØØ  ØØ ØØ FF 7F 24 Ø3 F5 7F-48 Ø6 D8 7F 6A Ø9 A6 7F
ØBFD:Ø41Ø  8C ØC 61 7F AB ØF Ø9 7F-C8 12 9C 7E E2 15 1D 7E
ØBFD:Ø42Ø  F9 18 89 7D ØB 1C E3 7C-1A 1F 29 7C 23 22 5C 7B
ØBFD:Ø43Ø  28 25 7C 7A 26 28 89 79-1F 2B 84 78 11 2E 6B 77
ØBFD:Ø44Ø  FB 3Ø 41 76 DF 33 Ø4 75-BA 36 B5 73 8C 39 54 72
ØBFD:Ø45Ø  56 3C E2 7Ø 17 3F 5E 6F-CE 41 C9 6D 7A 44 23 6C
ØBFD:Ø46Ø  1C 47 6D 6A B4 49 A6 68-3F 4C CF 66 BF 4E E8 64
ØBFD:Ø47Ø  33 51 F1 62 9B 53 EB 6Ø-F5 55 D7 5E 42 58 B3 5C
ØBFD:Ø48Ø  82 5A 82 5A B3 5C 42 58-D7 5E F5 55 EB 6Ø 9B 53
ØBFD:Ø49Ø  F1 62 33 51 E8 64 BF 4E-CF 66 3F 4C A6 68 B4 49
ØBFD:Ø4AØ  6D 6A 1C 47 23 6C 7A 44-C9 6D CE 41 5E 6F 17 3F
ØBFD:Ø4BØ  E2 7Ø 56 3C 54 72 8C 39-B5 73 BA 36 Ø4 75 DF 33
ØBFD:Ø4CØ  41 76 FB 3Ø 6B 77 11 2E-84 78 1F 2B 89 79 26 28
ØBFD:Ø4DØ  7C 7A 28 25 5C 7B 23 22-29 7C 1A 1F E3 7C ØB 1C
ØBFD:Ø4EØ  89 7D F9 18 1D 7E E2 15-9C 7E C8 12 Ø9 7F AB ØF
ØBFD:Ø4FØ  61 7F 8C ØC A6 7F 6A Ø9-D8 7F 48 Ø6 F5 7F 24 Ø3
```



Note: $x_o - x_7$: DATA ARRAY (LEFT)

$X_o - X_7$: SCRAMBLED OUTPUT (RIGHT)

W : exp (−j2 π /N)

Fig.1 : SIGNAL FLOW GRAPH FOR DECIMATION IN FREQUENCY DECOMPOSITION OF 8 POINT DFT (Ref.9, p.304)

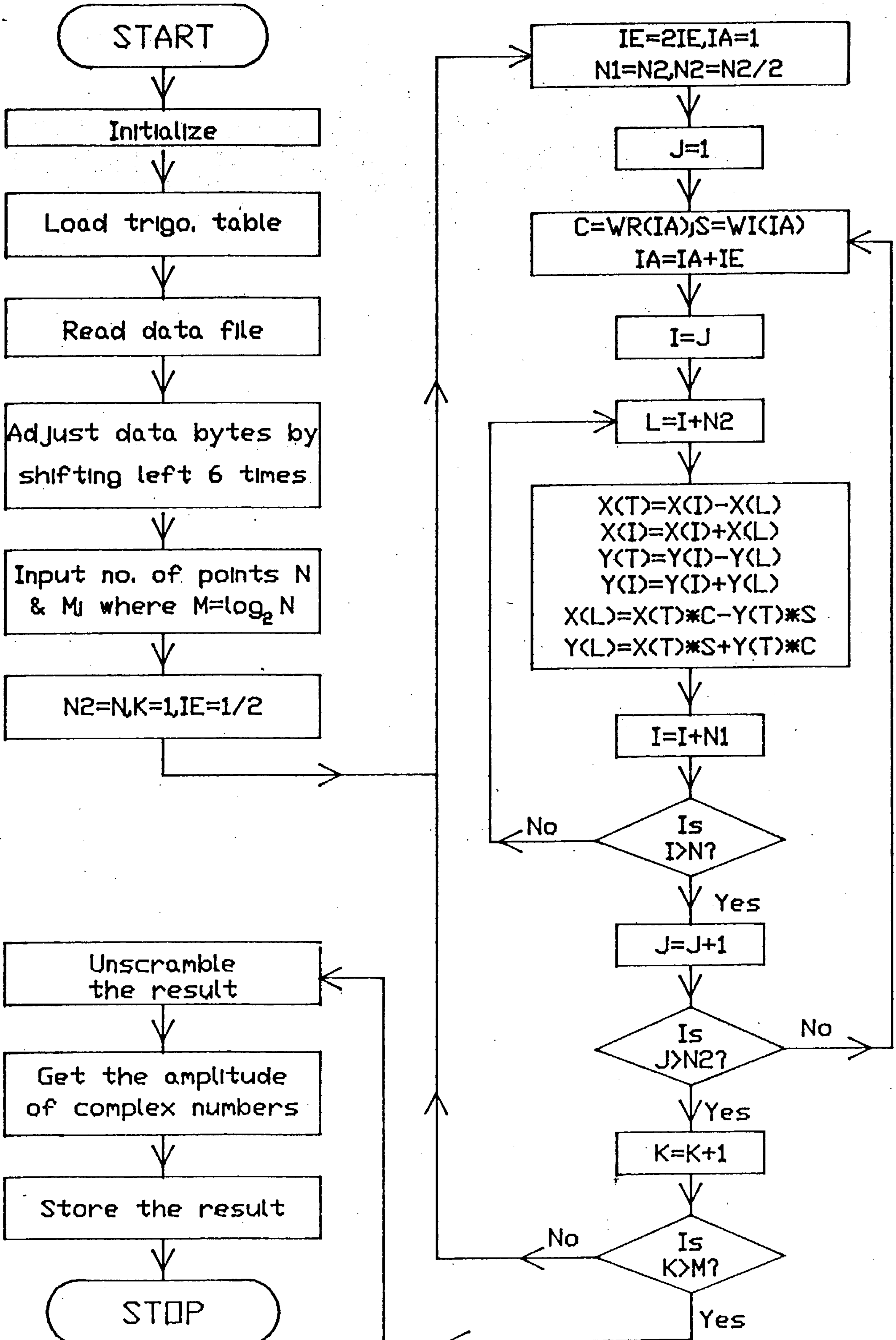


FIG.2 FLOW CHART OF FFT PROGRAM

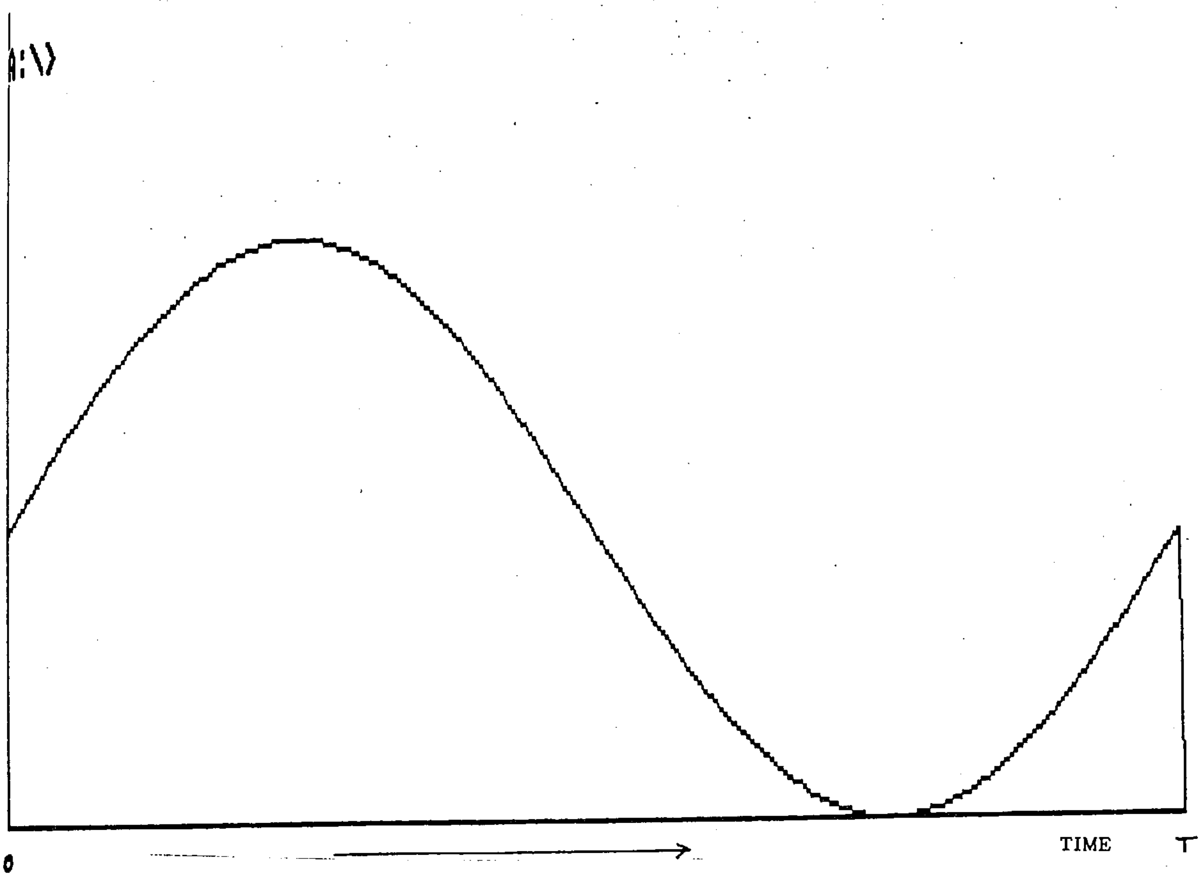


FIG.3 : AMPLITUDE DURING ONE CYCLE OF SINE WAVE (SAMPLED IN 256 PARTS)

FIG.4 : FIRST 64 ELEMENTS OF DFT FOR THE WAVEFORM OF FIG.3.

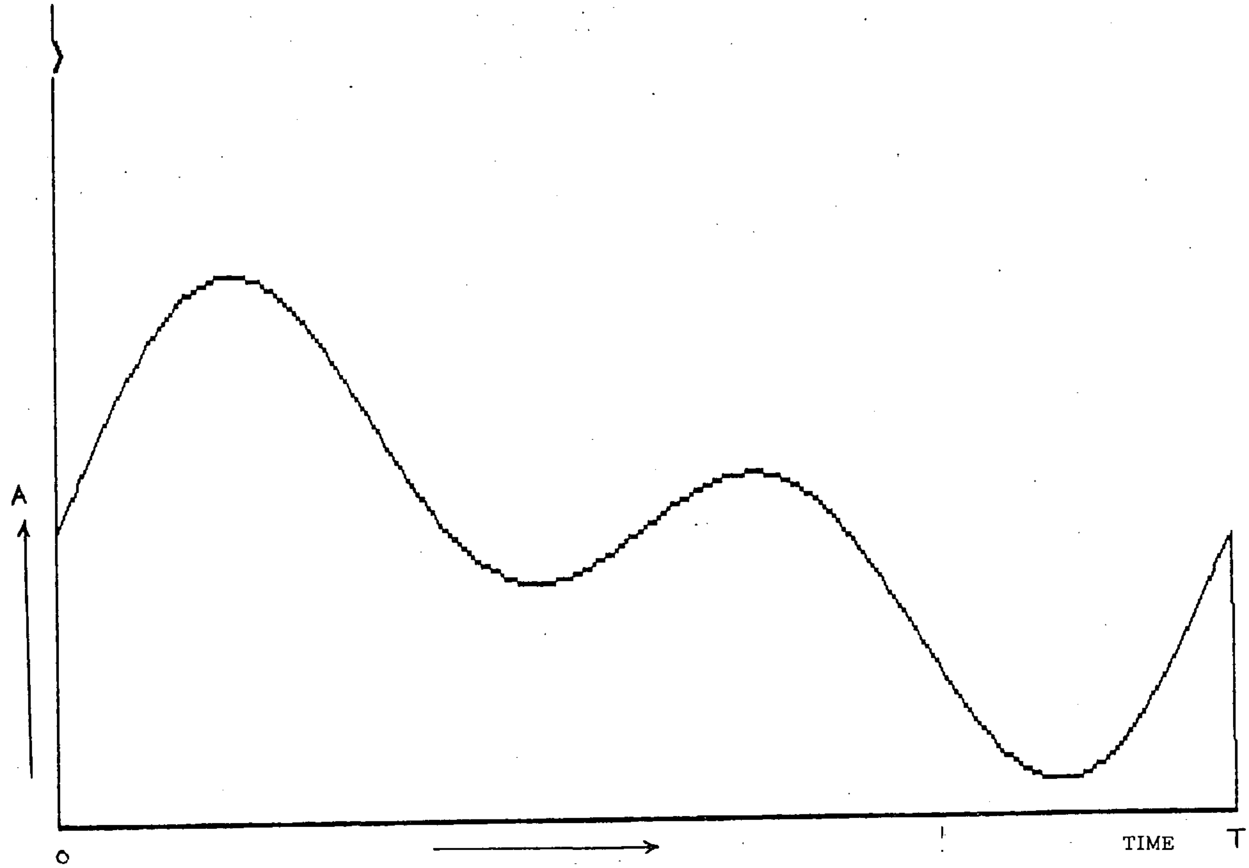


FIG.5 : AMPLITUDE DURING ONE CYCLE OF SINE WAVE ADDED TO ITS SECOND HARMONIC.

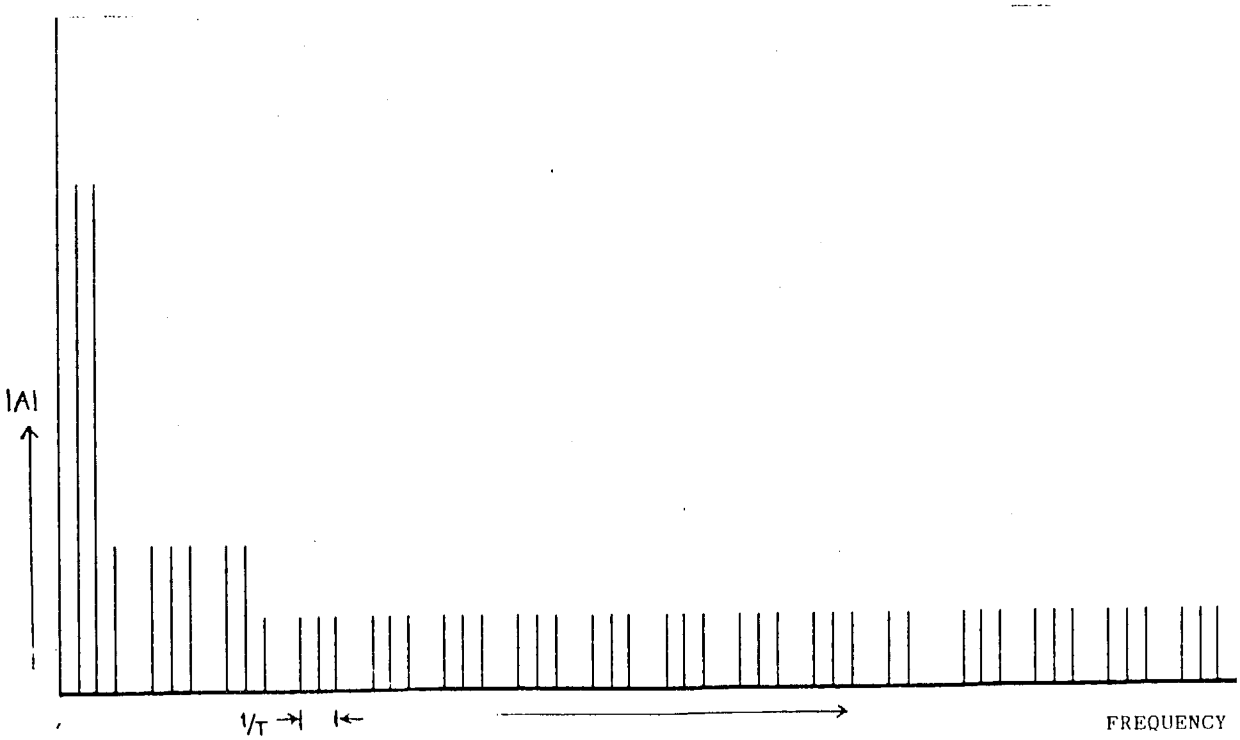


FIG.6 : FIRST 64 ELEMENTS OF DFT OF WAVEFORM IN FIG.5.

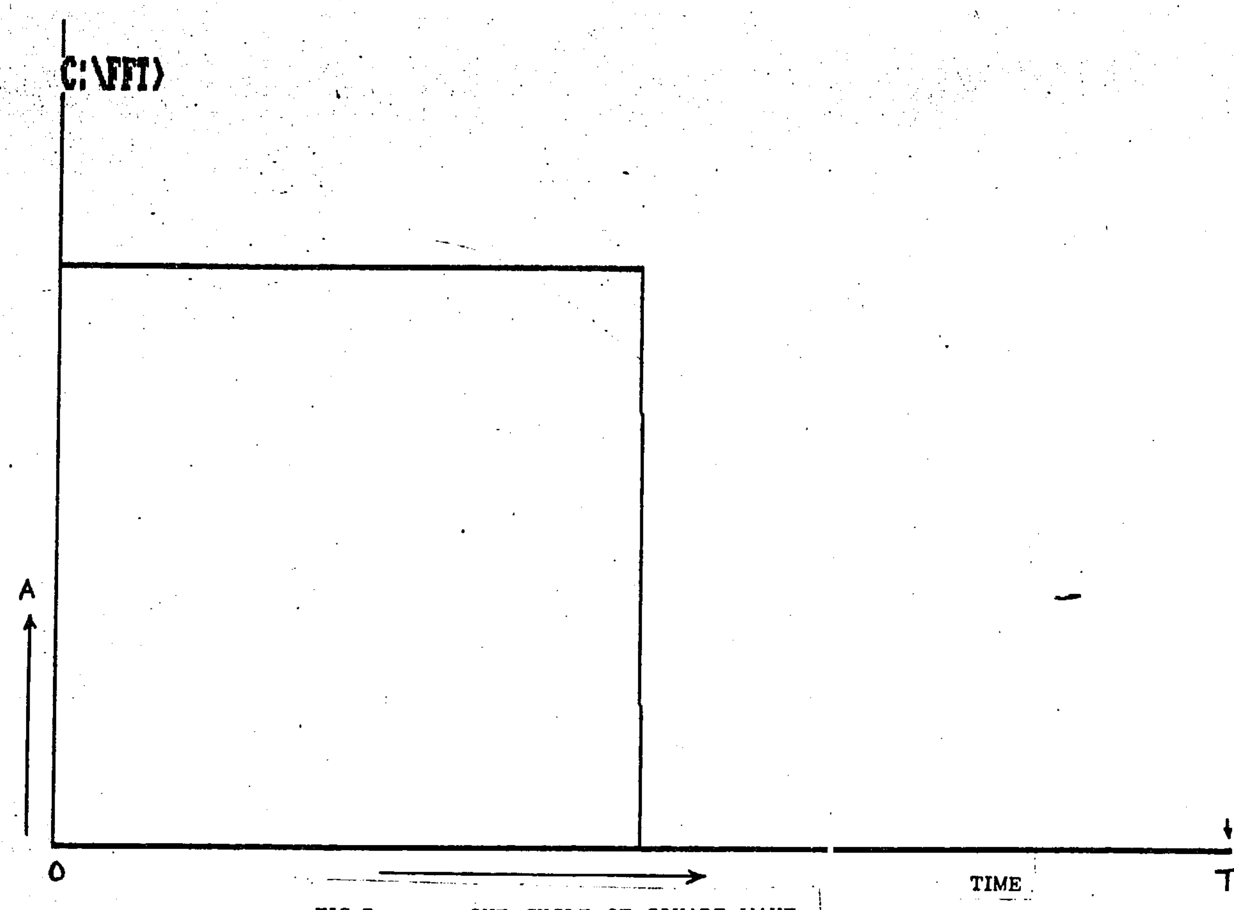


FIG.7 : ONE CYCLE OF SQUARE WAVE.

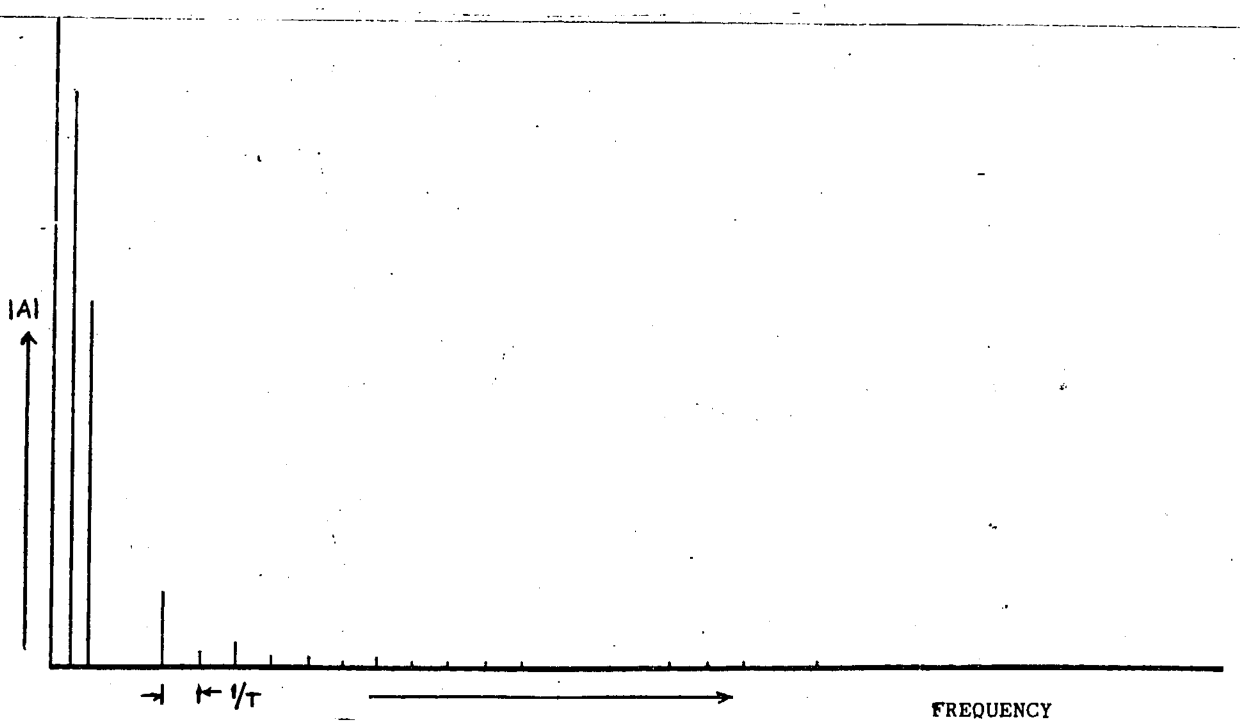


FIG.8 : FIRST 64 ELEMENTS OF DFT OF WAVEFORM IN FIG.7.

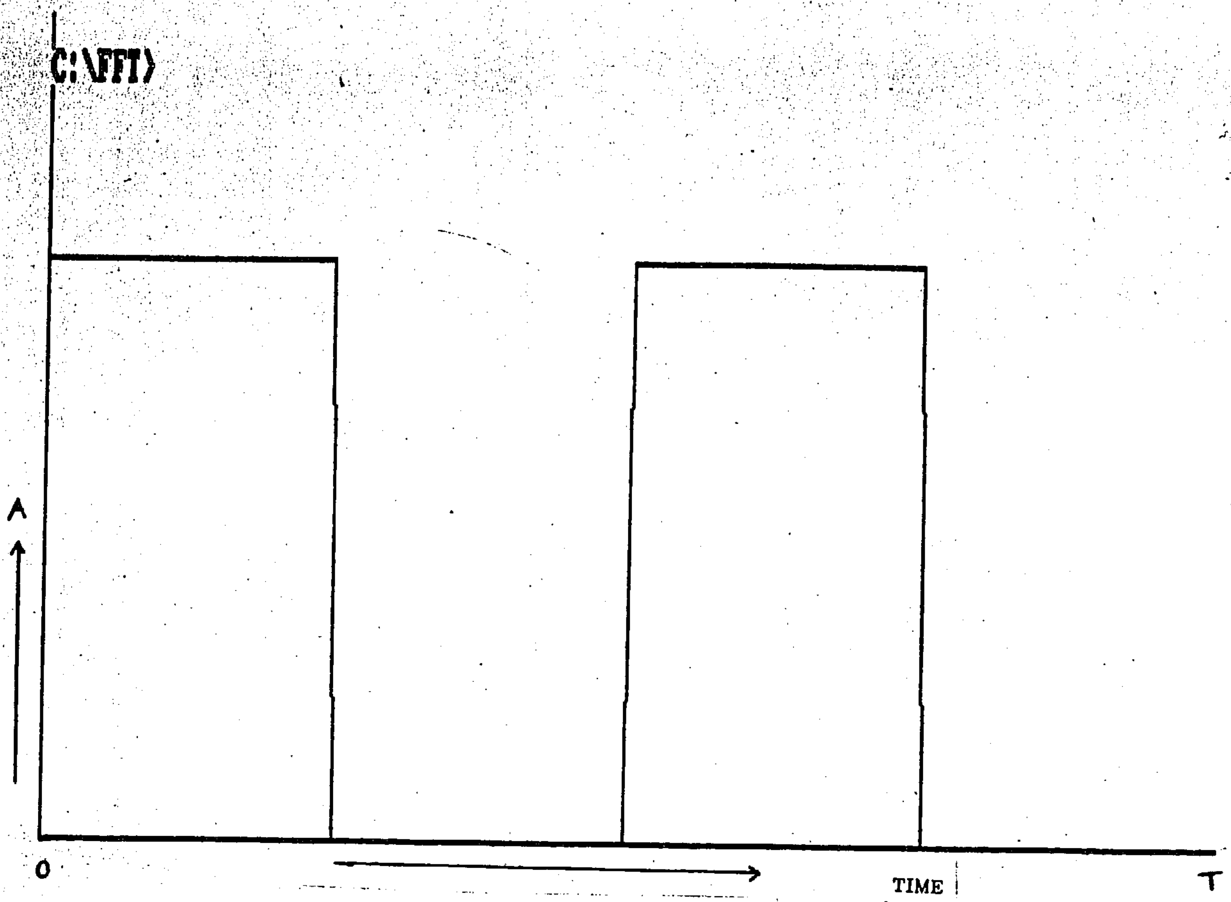


FIG.9 : TWO CYCLES OF SQUARE WAVE.

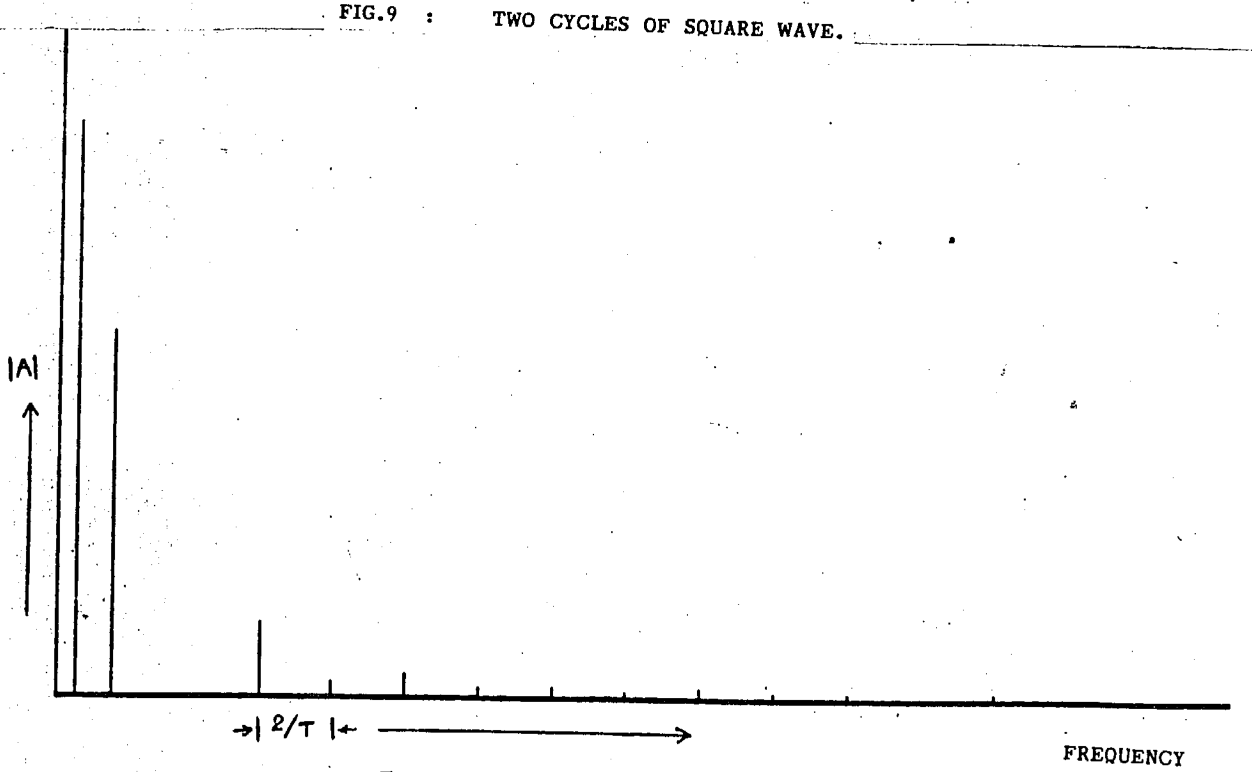


FIG. 10 FIRST 64 ELEMENTS OF DFT FOR TWO CYCLES OF SQ. WAVE (FIG 9)

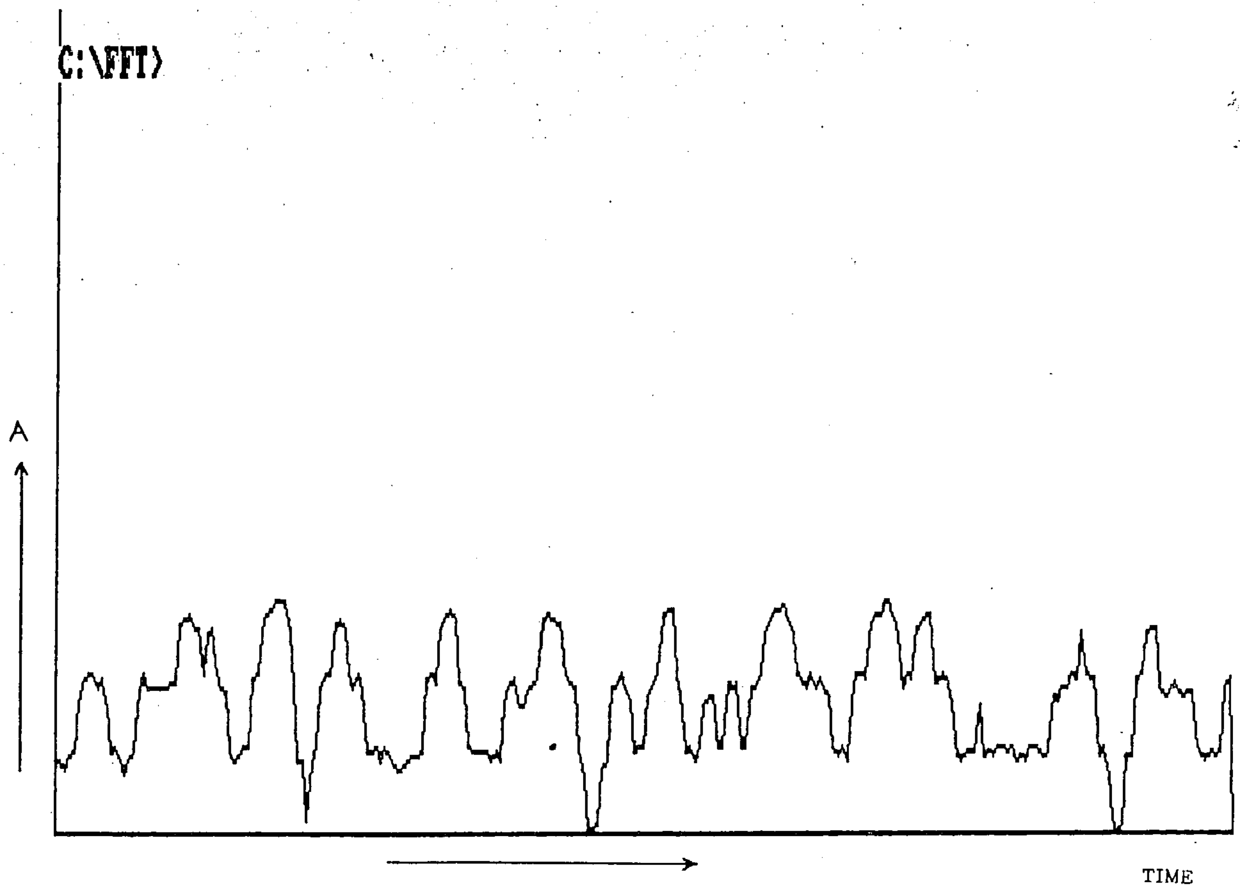


FIG.11 : IONOSPHERIC SCINTILLATION DATA.

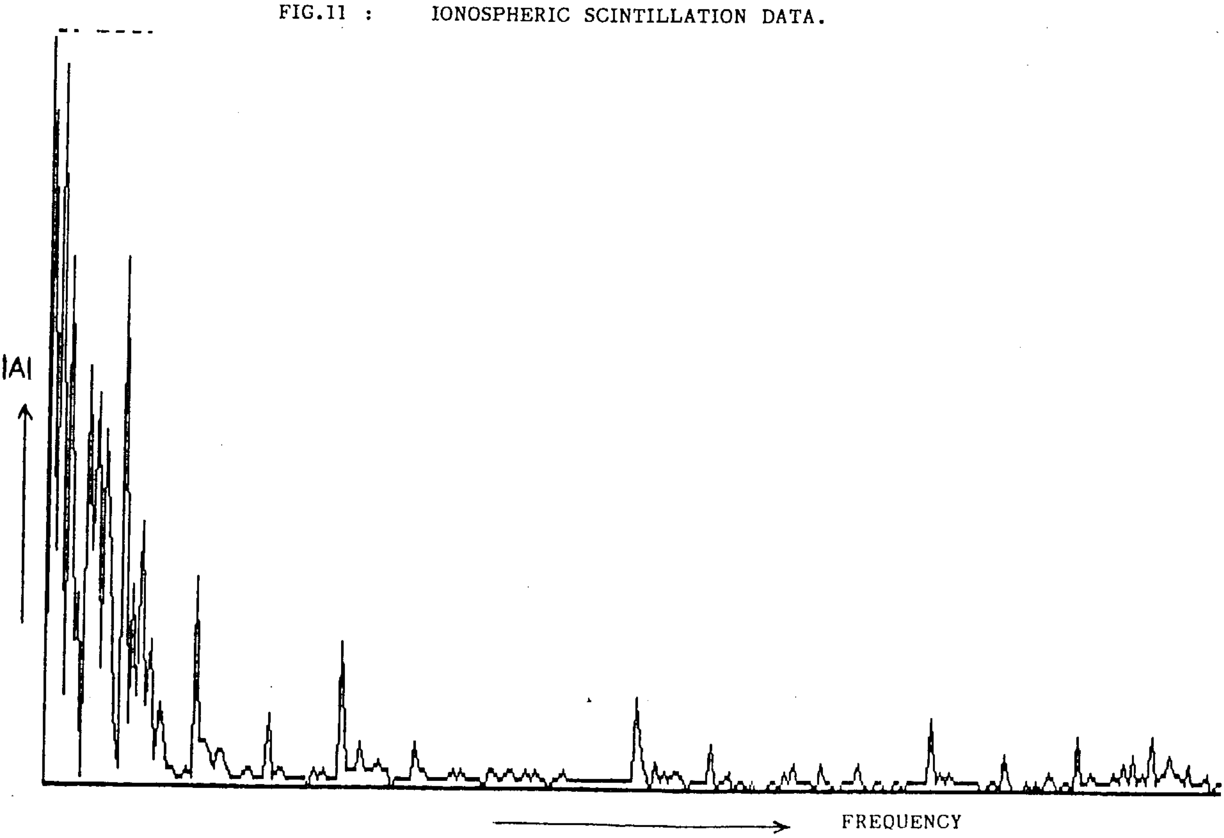


FIG.12 : PLOT OF AMPLITUDE OF DFT FOR ALL THE 256 ELEMENTS.