

NASA Contractor Report 4117

High-Speed Assembly Language (80386/80387) Programming for Laser Spectra Scan Control and Data Acquisition Providing Improved Resolution Water Vapor Spectroscopy

Robert J. Allen
Allen Associates
Newport News, Virginia

Prepared by Allen Associates
for Vigyan Research Associates, Inc., for
NASA Langley Research Center
under Contract NAS1-17919

NASA

National Aeronautics
and Space Administration

Scientific and Technical
Information Division

1988

TABLE OF CONTENTS

| | <u>page</u> |
|---|-------------|
| ABSTRACT | v |
| INTRODUCTION | 1 |
| WATER VAPOR SPECTROSCOPY EXPERIMENTS | 1 |
| CHOICE OF ASSEMBLY LANGUAGE PROGRAMMING | 2 |
| DESCRIPTION OF PROGRAMS | 3 |
| FORMULATING THE PROGRAMS | 5 |
| ACKNOWLEDGMENT | 6 |
| REFERENCES | 6 |

LIST OF FIGURES

Figure

| | |
|---|----|
| 1. BLOCK DIAGRAM, WATER VAPOR SPECTROSCOPY EXPERIMENT | 7 |
| 2. FLOW DIAGRAMS: | |
| A. OF THE BASIC PROGRAM | 8 |
| B. OF THE ASSEMBLY LANGUAGE PROGRAM | 9 |
| 3. REPRESENTATIVE WATER (H ₂ O) SPECTRUM CURVES: | |
| A. Cross Section vs. Wavenumber showing pressure shift in air for the line centered at 13914.9502 cm ⁻¹ | 11 |
| B. Cross Section vs. Wavenumber showing pressure shift in air for the line centered at 13947.2608 cm ⁻¹ | 12 |
| 4. SCOPE PHOTOGRAPHS OF 5-VOLT PULSES GENERATED FROM BINARY OUTPUTS BO-0 TO BO-3. PULSE WIDTHS AND VARIOUS PROGRAM TIMES ARE INCLUDED | 13 |

LIST OF TABLES

| <u>Table</u> | <u>page</u> |
|---|-------------|
| I. PROGRAM TIMES DETERMINED USING BINARY OUTPUT PULSES SHOWN IN FIGURE 4 ----- | 15 |
| II. BASIC PROGRAM (ALLENB7R.BAS) | |
| A. LISTING OF PROGRAM ----- | 16 |
| B. DISPLAY PRODUCED BY RUNNING PROGRAM ----- | 19 |
| III. ASSEMBLY LANGUAGE PROGRAMS: | |
| A. ALLEN7R.LST (Microsoft Macro Assembler, Version 4.00) ----- | 20 |
| B. SYMDEB ALLEN7R.EXE (Microsoft Symbolic Debug Utility, Version 4.00) ----- | 27 |

ABSTRACT

An assembly language program using the Intel 80386 CPU and 80387 math co-processor chips was written to increase the speed of data gathering and processing, and provide control of a scanning CW ring dye laser system. This laser system is used in high resolution (better than 0.001 cm^{-1}) water vapor spectroscopy experiments. Laser beam power is sensed at the input and output of white cells and the output of a Fabry-Perot. The assembly language subroutine is called from Basic, acquires the data and performs various calculations at rates greater than 150 times faster than could be performed by the higher level language. The width of output control pulses generated in assembly language are 3 to 4 microseconds as compared to 2 to 3.7 milliseconds for those generated in Basic (about 500 to 1000 times faster).

Included are a block diagram and brief description of the spectroscopy experiment, a flow diagram of the basic and assembly language programs, listing of the programs, scope photographs of the computer generated 5-volt pulses used for control and timing analysis, and representative water spectrum curves obtained using these programs.

HIGH-SPEED ASSEMBLY LANGUAGE (80386/80387) PROGRAMMING
FOR LASER SPECTRA SCAN CONTROL AND DATA ACQUISITION
PROVIDING IMPROVED RESOLUTION WATER VAPOR SPECTROSCOPY

By

Robert J. Allen

INTRODUCTION

This report describes an Intel 80386 and 80387 assembly language program written to increase the speed of data gathering and processing, and provide control of a scanning CW ring dye laser system. This laser system is used in high resolution Water Vapor Spectroscopy Experiments. The 80386 is a 32-bit CPU chip operating at 16 MHz in a Compaq 386 computer, and the 80387 is a high speed math coprocessor. The assembly language subroutine is called from BASICA.

The program was originally written for the Compaq Portable 286 computer using the 80286 and 80287 chips. It was later updated for use with the faster Compaq 386.

WATER VAPOR SPECTROSCOPY EXPERIMENTS

A block diagram of the spectroscopy experiment is shown in Fig. 1. # An argon laser pumps a tunable CW ring dye laser. After receiving a Laser Scan Trigger, the Scanning Electronics will cause a very narrow line-width (0.0001 pm) laser beam to be outputted at wavelengths within pre-selected limits, for

Figures 1 and 3 were provided by Dr. Benoist Grossmann, project scientists conducting the water vapor spectroscopy experiments.

example, 726.000 to 726.050 nm. This 50 pm scanning range can be selected anywhere between 725 nm (13,793.1 cm⁻¹) to 730 nm (13,698 cm⁻¹). A portion of the output beam is directed by beamsplitters to a Monochrometer for absolute wavelength calibration, a Fabry-Perot and photodiode P4 to provide the wavelength markers, and P3 as the Power Reference. The remaining beam is routed through White Cells #1 and #2 and sensed by photodiodes P1 and P2.

The outputs from the four photodiodes, P1-P4, are amplified and routed to four analog input channels multiplexed into a 10 KHz analog-to-digital converter with 12-bit resolution (A/D Board). The four multiplexed digital outputs are then processed as described later in this report. The A/D Board is part of an IBM Data Acquisition and Control Adapter card that fits inside the Compaq. This Adapter also contains two analog output channels; a 16-bit digital input port; a 16-bit digital output port; a 32-bit timer; and a 16-bit, externally-clocked, timer/counter. One bit of the digital output port is available for use as the Laser Scan Trigger.

CHOICE OF ASSEMBLY LANGUAGE PROGRAMMING

Since the four readings from photodiodes P1-P4 averaged N times are a single point on a rapidly changing plot of a single water vapor line, this data set should be obtained as rapidly as the hardware allows. More time, if needed, can be used to start the repeat (R) of subsequent data sets since this time is the separation between data points. Separation time, however,

should also be kept to a minimum since it is associated with the resolution of the curve.

As indicated in the general literature and by Rollins (1985), "assembly language is by far the fastest, most flexible, and most compact of all programming languages. It shows how and why higher-level languages operate. It gives access to features of a machine that are inaccessible with other languages." An assembly language matrix multiplication routine published by Startz (1985) is about 150 times faster than pre-8087 Basic.

Fortunately, assembly language routines are easily combined with either interpreted or compiled Basic, as well as with programs written in other high-level languages. The main program illustrated in this report was written in Basic (BASICA.COM) calling the assembly language routine for acquiring N sets of the four photodiode (P1-P4) readings and performing required calculations.

DESCRIPTION OF PROGRAMS

The basic program (Table II) and assembly language program (Table III) were prepared as part of this task and provided to the Project Scientists. He incorporated them into his Basic program which included plotting features for use in the high resolution spectroscopy experiments.

Figure 2 contains a flow diagram of both programs. Basic is used first (prior to scanning) to calculate and display the 'mean' value of the background noise (W, X, Y & Z) from each of the four photodiodes (P1-P4). Scanning is initiated and the assembly language program called. The output from each of the

photodiodes are read and the corresponding background noise subtracted. The differences from P1, P2 and P4 are then normalized to the power reference difference P3. This process is repeated N times and the 'mean' of each of the three ratios $((P1-X)/(P3-W))$, $(P2-Y)/(P3-W)$, and $(P4-Z)/(P3-W)$ calculated. The process then returns to Basic where the 'mean' of these three ratios are displayed. Each of the ratios constitute a single data point on each of three curves. The above process is then repeated R times obtaining new data points while the laser continues to scan. BASICA line numbers are then displayed for the background sample size (M), data sample size (N) and number of repeats during laser scanning (R). The results, when coupled with the plotting program, are the curves shown in Figures 3A and B.

Five-volt pulses, BO-0 and BO-1, are generated in Basic prior to scanning and can be used to trigger a scope and to initiate scanning. Binary output pulses, BO-2 and BO-3, are generated in Assembly Language and were used to measure the time for different processes during program development such as the:

- (1) greater-than 20 microsecond delay time required to allow transients to settle following each multiplexed operation,
- (2) total time used to read the four analog input from the photodiodes and subtract the backgrounds from each, and
- (3) time to calculate the mean of the ratios.

Scope photographs of pulses BO-0 to BO-3, their pulse widths, and various times are shown in Figure 4 and Table I.

FORMULATING THE PROGRAMS

The assembly language program was written using WordStar, assembled using Microsoft Macro-Assembler version 4.00 (MASM ALLEN7R) and linked for high memory residence (LINK/H ALLEN7R). Debug was called (SYMDEB ALLEN7R.EXE) and Basic entered using N BASICA.COM. L and G. ALLENB7R.BAS was then loaded and a binary file created using DEF SEG = &H9FCA followed by BSAVE "ALLEN7R.BIN",0,&h178. Running ALLENB7.BAS produced the display shown in Table IIB.

ACKNOWLEDGMENT

Publications by Lafore (1984) and Startz (1985) provided excellent references supporting the assembly language portions of this task. The author wishes to thank Dr. Jerry Tucker of NASA Langley Research Center for his helpful suggestion in connection with the assembly language programming, Dr. Benoist Grossmann for providing the information concerning the experiments and review of this report, and Patrick Ponsardin for his review and comments concerning this report. Both Benoist and Patrick are affiliated with Old Dominion University Research Foundation (on leave from Electricite de France).

REFERENCES

- Rollins, Dan (1985), "IBM-PC 8088 MACRO Assembler Programming", Macmillan Publishing Co.
- Startz, Richard (1985), "8087 Applications and Programming for the IBM PC, XT, and AT", Brady Communications Co., Inc. NY, NY 10020.
- Lafore, Robert (1984), "Assembly Language Primer for the IBM PC & XT", A Plume/Waite Book.

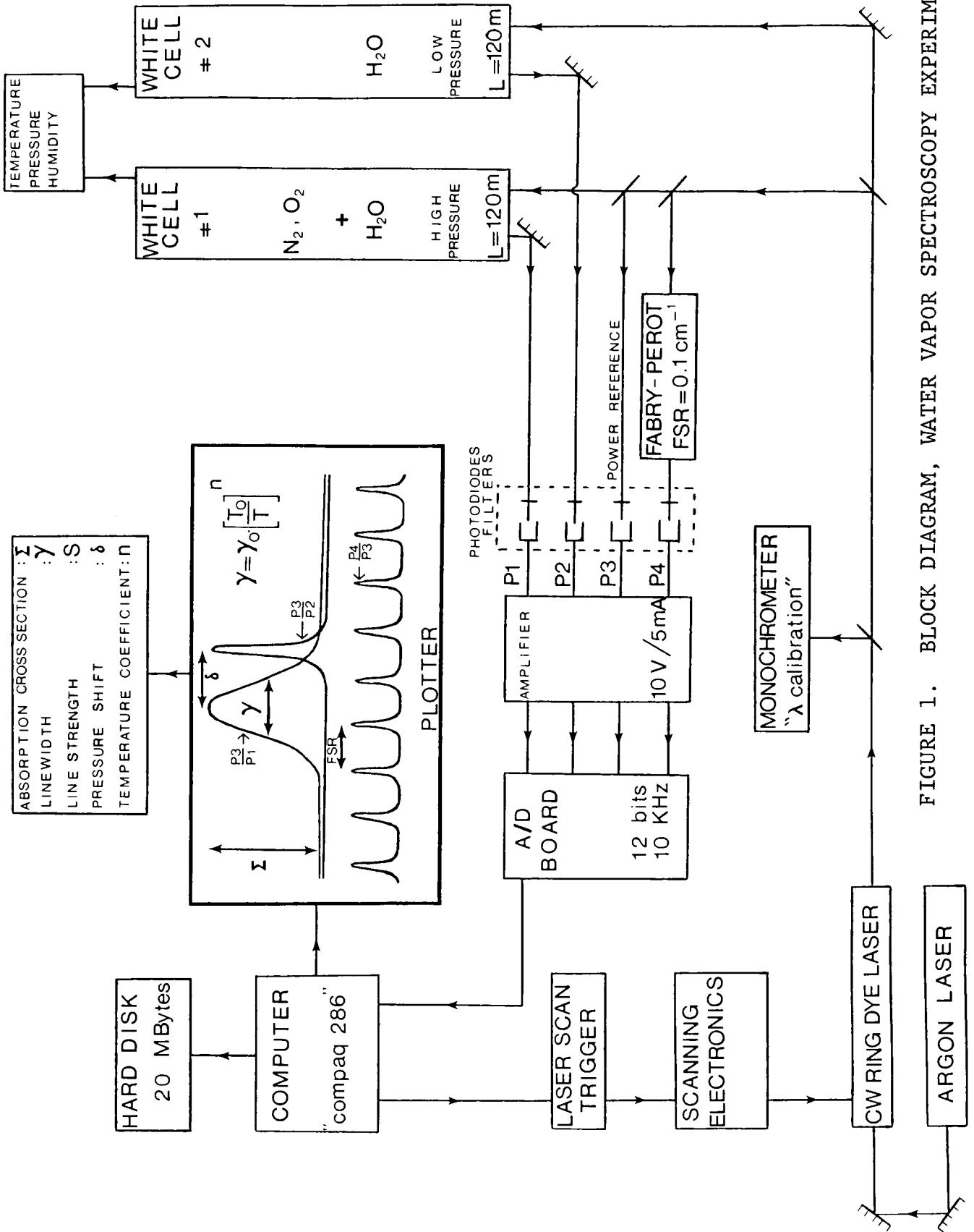


FIGURE 1. BLOCK DIAGRAM, WATER VAPOR SPECTROSCOPY EXPERIMENT.

ORIGINAL PAGE IS
OF POOR QUALITY

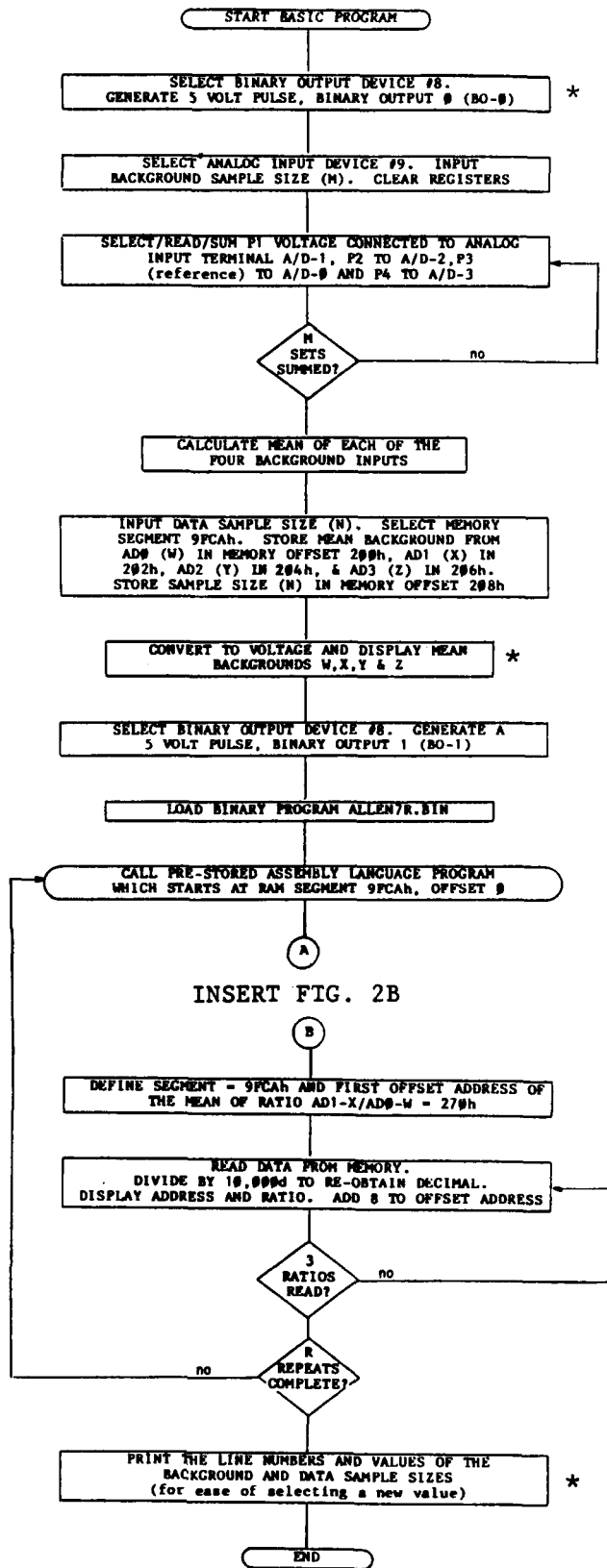
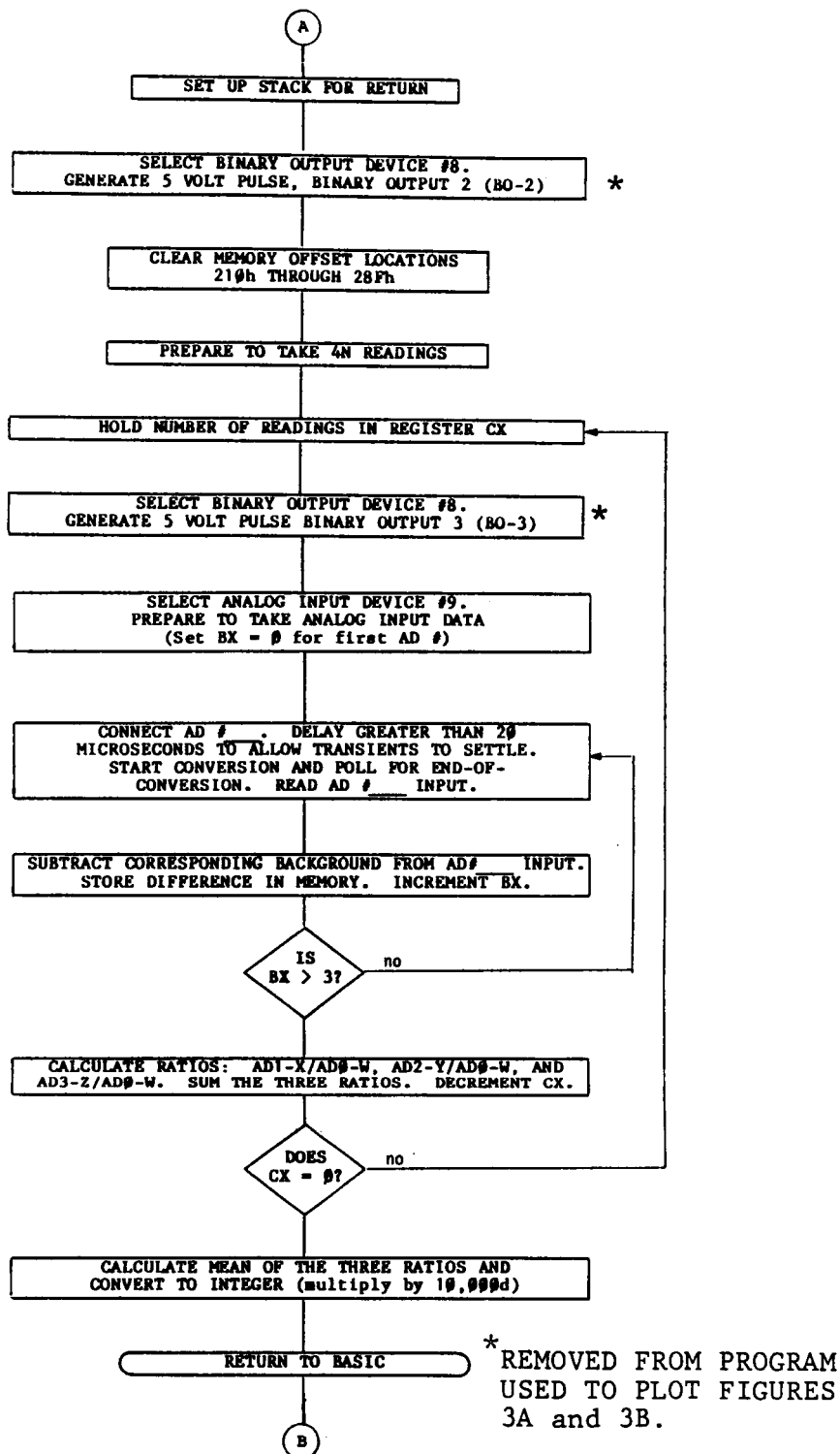


FIGURE 2A. FLOW DIAGRAM OF THE BASIC PROGRAM.

FROM FIG. 2A



TO FIG. 2A

FIGURE 2B. FLOW DIAGRAM OF THE 80386/80387 ASSEMBLY
LANGUAGE PROGRAM.

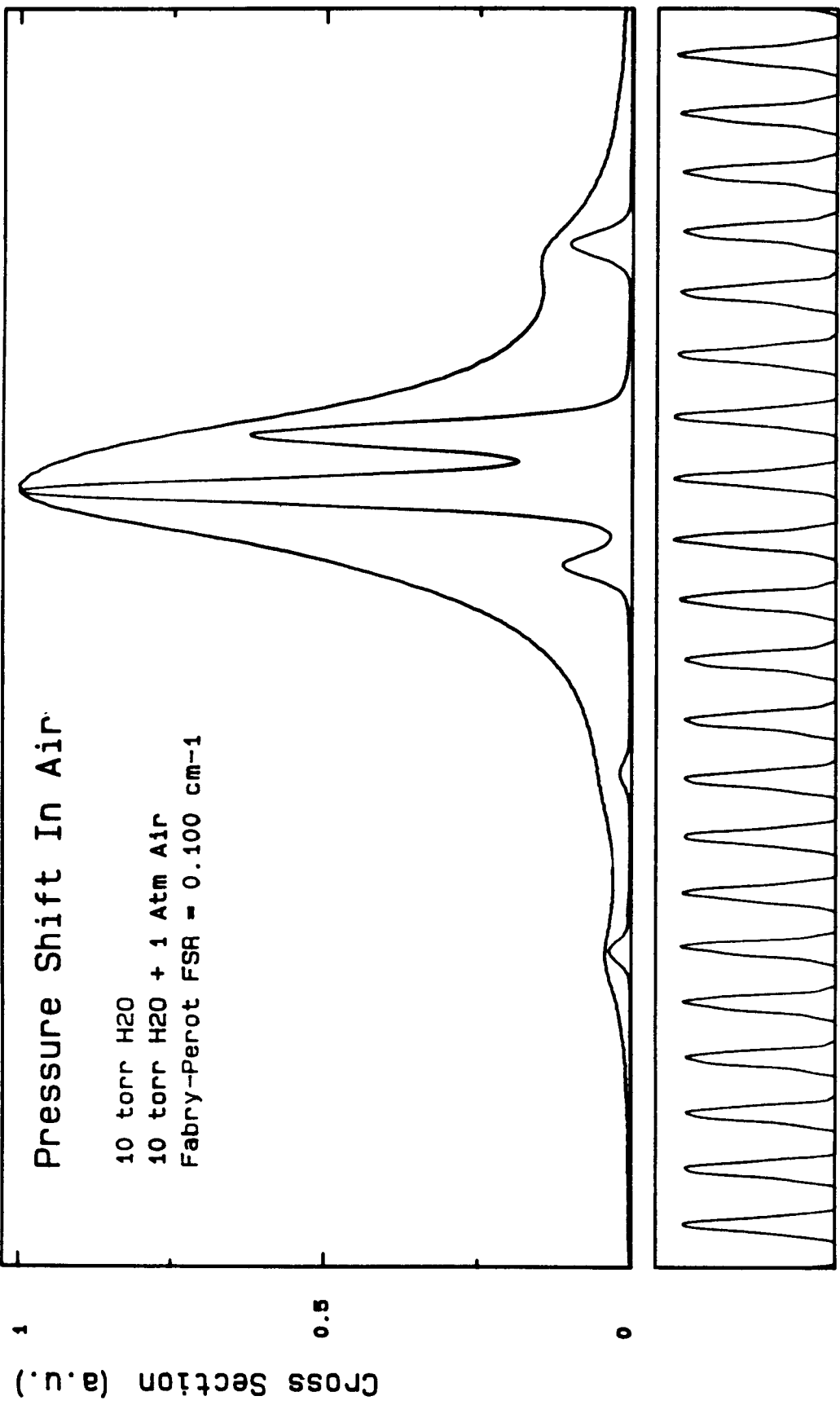


FIGURE 3A. CROSS SECTION VS. WAVENUMBER SHOWING PRESSURE SHIFT IN AIR FOR THE LINE CENTERED AT 13914.9502 cm-1.

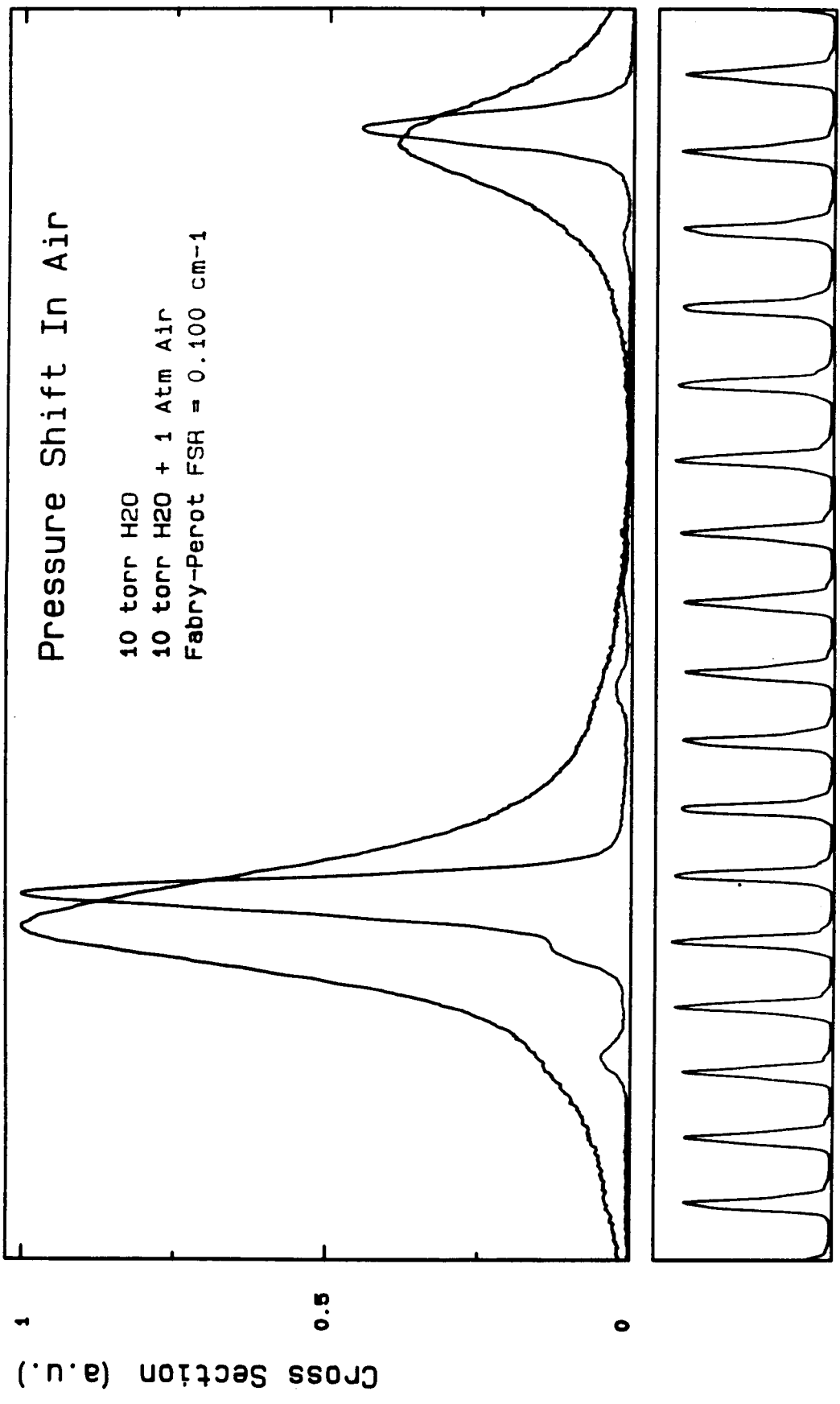
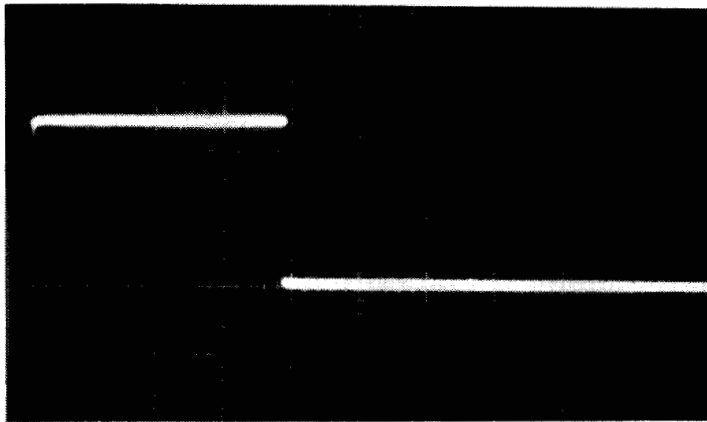


FIGURE 3B. CROSS SECTION VS. WAVENUMBER SHOWING PRESSURE SHIFT IN AIR FOR THE LINE CENTERED AT 13947.2608 cm⁻¹.

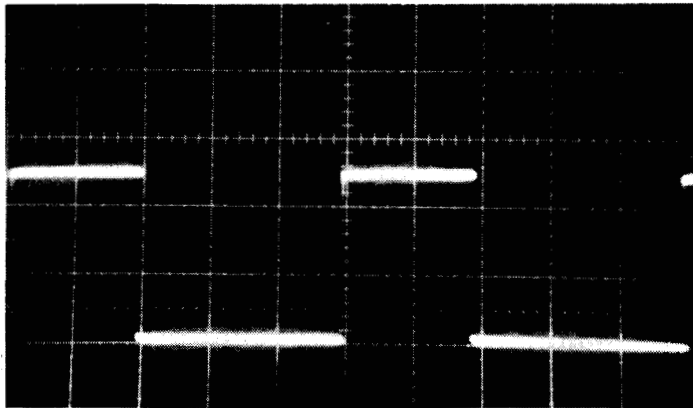


(A) 5V 3.7 ms wide
Binary Output
pulse BO-0

2V/div 0.5 ms/div

(added program:
1035 GOTO 80)

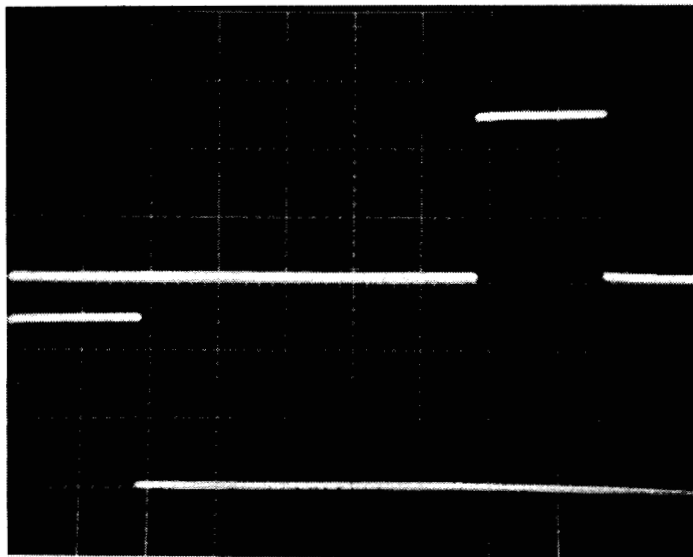
ORIGINAL PAGE IS
OF POOR QUALITY



(B) 5V 2 ms wide
Binary Output
pulse BO-1

2V/div 1 ms/div

(added program:
1105 GOTO 1060)



(C) BO-0

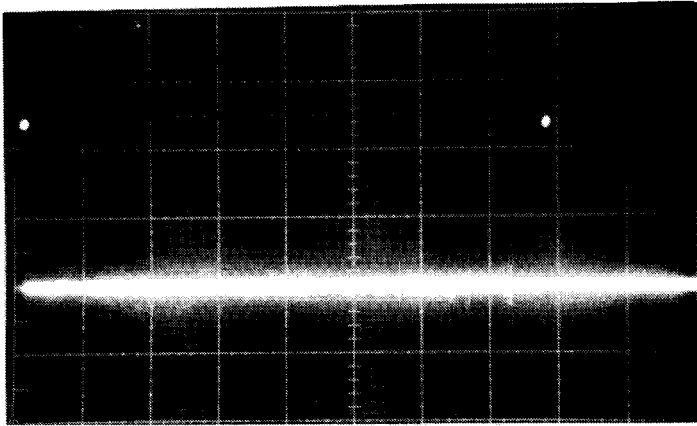
2V/div 1 ms/div

(D) BO-1 (scope trig)

2V/div 1 ms/div

(added program:
1105 GOTO 80)

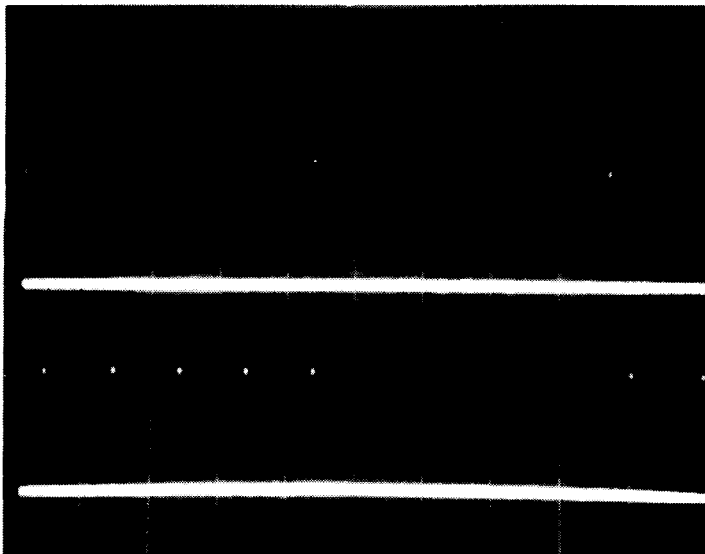
FIGURE 4. SCOPE PHOTOGRAPHS OF 5-VOLT PULSES GENERATED FROM BINARY OUTPUTS BO-0 TO BO-3. PULSE WIDTHS AND VARIOUS PROGRAM TIMES ARE LISTED IN TABLE I.



(E) BO-1

2V/div 100 ms/div

(added program:
1395 GOTO 1060)



(F) BO-2

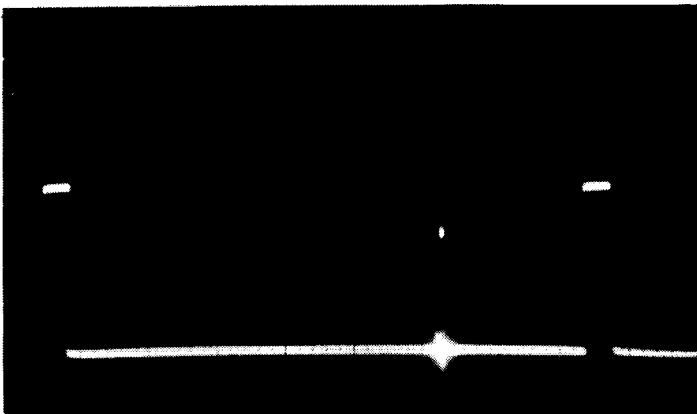
2V/div 1 ms/div

(G) BO-3

2V/div 1 ms/div

(added program:
1185 GOTO 1160)

ORIGINAL PAGE IS
OF POOR QUALITY,



(H) BO-1 ADDED TO
INTENSIFIED BO-2

(added program:
1185 GOTO 1060)

TABLE I. PROGRAM TIMES DETERMINED USING BINARY OUTPUT PULSES SHOWN IN FIGURE 4

| BASICA LINE NUMBERS | ADDED PROGRAM STEP * | GENERATED BINARY OUTPUT PULSE | TIME FROM START OF BO-i TO START OF BO-j |
|--|-----------------------------|--|--|
| 80-140 | 1035 GOTO 80 | BO-0, 5V 3.7 ms wide (FIG. 4A) | BO-0 to BO-0: 200 ms (M=1) 412 ms (M=10) 875 ms (M=30) 2500 ms (M=100) |
| 1060-1100 | 1105 GOTO 1060 | BO-1, 5V 2 ms wide (FIG. 4B) | BO-1 to BO-1: 3 ms |
| 80-1100 | DELETE 1040 1105 GOTO 80 | BO-0 and BO-1 (FIG. 4C&D) | BO-0 to BO-1: 200 ms (M=1) BO-1 to BO-0: 6.8 ms |
| 1060-1390 | 1395 GOTO 1060 | BO-1 (FIG. 4E) | BO-1 to BO-1: 770 ms |
| 1160-1185 includes total assembly language subroutine | 1185 GOTO 1160 | BO-2 & BO-3 3.5V into 10K WIDTHS: 3 us (BO-2) 4 us (BO-3) (FIG. 4F&G) | BO-2 to BO-2: 4.5 ms (N=1) 6.5 ms (N=3) 8.5 ms (N=5) 43.2 ms (N=40) FOR N=5: BO-2 to BO-3 -- 0.31 ms BO-31 to BO-32 -- 1.0 ms BO-35 to BO-2 -- 4.2 ms |
| 1060-1185 includes total assembly language subroutine | 1185 GOTO 1060 | BO-1 ADDED TO INTENSIFIED BO-2. (BO-2 into 10K) (FIG. H) | BO-1 to BO-2: -- 29.6 ms BO-2 to BO-1: -- 10.4 ms BO-1 to BO-1: -- 40 ms |

*Added program to produce a repetitive pulse for timing analyses and scope photographic purposes.

TABLE IIA BASIC PROGRAM ALLENB7R.BAS

```

10 CLS '--- WATER VAPOR SPECTROSCOPY EXPERIMENT, CONTROL & DATA PROCESSING ---
20 '
30 'GENERATES BINARY OUTPUT PULSE BO 0, CALCULATES THE MEAN OF M SETS OF THE
BACKGROUND DATA FROM AD0-AD3, POKES THE FOUR BACKGROUND VALUES & THE DATA
SAMPLE SIZE N INTO MEMORY, CONVERTS THE FOUR INTO A VOLTAGE, AND DISPLAYS THEIR
VALUES.
40 ' CALLS THE ASSEMBLY LANGUAGE PROGRAM. AFTER RETURNING TO BASIC, DIVIDES
THE THREE CALCULATED DATA RATIOS (corrected for background noise) BY 10,000,
AND DISPLAYS THESE RATIOS.
50 ' THE ASSEMBLY LANGUAGE PROGRAM IS AGAIN CALLED AND THE PROCESS REPEATED
R TIMES. THE PROGRAM LISTING FOR THE BACKGROUND SAMPLE SIZE (M), DATA SAMPLE
SIZE (N) AND THE NUMBER OF REPEATS DURING LASER SCANNING (R) ARE DISPLAYED FOR
EASE OF CHANGE.
60 '
70 ' ***** GENERATE BINARY OUTPUT PULSE BO 0 *****
80 OUT 49890!,8 'Selects Reg. 12, Device 8
90 OUT 8930,0 'Zero volts from BO 0 (low byte)
100 OUT 8931,0 'Zero volts from BO 0 (high byte)
110 OUT 8930,1 '5 volts from BO 0 (low byte)
120 OUT 8931,0 '(high byte) SCOPE TRIG +
130 OUT 8930,0 'Zero volts from BO 0 (low byte)
140 OUT 8931,0 '(high byte) START MEASURING BACKGROUND
150 ' ***** TAKE THE MEAN OF BACKGROUND READINGS FROM AD0-AD3 *****
160 OUT 49890!,9 'BASIC ADDRESS OF DATA ACQ ADAPTER 0 = 2E2H (AD# INPUTS)
170 M = 30 'BACKGROUND SAMPLE SIZE, M
180 AL=0:AH=0:BL=0:BH=0:CL=0:CH=0:DL=0:DH=0
190 WL=0:WH=0:XL=0:XH=0:YL=0:YH=0:ZL=0:ZH=0
200 FOR J=1 TO M
210 OUT 738,0
220 OUT 738,0 'SELECT AD 0 (P1)
230 OUT 739,0
240 OUT 738,1
250 OUT 739,0
260 OUT 738,0
270 OUT 739,0
280 AL=INP (8930) 'READ AD 0
290 WL=WL+AL
300 AH=INP (8931)
310 WH=WH+AH 'SUM AD 0
320 OUT 738,0 'SELECT/READ/SUM AD 1 (P2)
330 OUT 739,1
340 OUT 738,1
350 OUT 739,1
360 OUT 738,0
370 OUT 739,1
380 BL=INP (8930)
390 XL=XL+BL
400 BH=INP (8931)
410 XH=XH+BH
420 OUT 738,0 'SELECT/READ/SUM AD 2 (P3)
430 OUT 739,2
440 OUT 738,1
450 OUT 739,2
460 OUT 738,0
470 OUT 739,2
480 CL=INP (8930)
490 YL=YL+CL
500 CH=INP (8931)
510 YH=YH+CH

```

```

520 OUT 738,0 'SELECT/READ/SUM AD 3 (P4)
530 OUT 739,3
540 OUT 738,1
550 OUT 739,3
560 OUT 738,0
570 OUT 739,3
580 DL=INP (8930)
590 ZL=ZL+DL
600 DH=INP (8931)
610 ZH=ZH+DH
620 NEXT J 'REPEAT SELECT/READ/SUM DATA SET AD 0 - AD 3
630 '
640 WL=WL/M 'MEAN OF BACKGROUND READINGS FROM AD 0
650 WH=WH/M
660 XL=XL/M 'MEAN OF BACKGROUND READINGS FROM AD 1
670 XH=XH/M
680 YL=YL/M 'MEAN OF BACKGROUND READINGS FROM AD 2
690 YH=YH/M
700 ZL=ZL/M 'MEAN OF BACKGROUND READINGS FROM AD 3
710 ZH=ZH/M
720 '
730 N = 5 'DATA SAMPLE SIZE, N
740 '
750 ' ***** STORE MEAN BACKGROUND READINGS AND N IN MEMORY *****
760 DEF SEG = &H9FCA 'SEGMENT ADDRESS (page 426)
770 ADDR = &H200
780 POKE ADDR,WL 'SAVE AD 0 BACKGROUND IN MEMORY 200H
790 ADDR=ADDR+1
800 POKE ADDR,WH
810 ADDR=ADDR+1
820 POKE ADDR,XL 'SAVE AD 1 BACKGROUND IN MEMORY 202H
830 ADDR=ADDR+1
840 POKE ADDR,XH
850 ADDR=ADDR+1
860 POKE ADDR,YL 'SAVE AD 2 BACKGROUND IN MEMORY 204H
870 ADDR=ADDR+1
880 POKE ADDR,YH
890 ADDR=ADDR+1
900 POKE ADDR,ZL 'SAVE AD 3 BACKGROUND IN MEMORY 206H
910 ADDR=ADDR+1
920 POKE ADDR,ZH
930 ADDR = &H208
940 POKE ADDR,N 'SAVE DATA SAMPLE SIZE, N, IN MEMORY 208H
950 '
960 ' ***** CONVERT TO VOLTAGE AND DISPLAY *****
970 W=.00244*(256*WH+WL)
980 X=.00244*(256*XH+XL)
990 Y=.00244*(256*YH+YL)
1000 Z=.00244*(256*ZH+ZL)
1010 PRINT "BACKGROUND","W","X","Y","Z"
1020 PRINT ,W,X,Y,Z
1030 PRINT
1040 STOP
1050 '***** GENERATE BINARY OUTPUT PULSE BO1 *****
1060 OUT 49890!,8 'Selects Reg. 12, Device 8
1070 OUT 8930,2 '5 volts from BO 1 (low byte)
1080 OUT 8931,0 '(high byte) 4-CHANNEL MARKER
1090 OUT 8930,0 'Zero volts from BO 1 (LOW BYTE)
1100 OUT 8931,0 'Zero volts from BO 1 (high byte)

```

```

1110 '##### PREPARE FOR ASSEMBLY LANGUAGE PROGRAM #####
1120 BLOAD "ALLEN7R.BIN",0      '(R. L. Lafore, page 428)
1130 R = 300      'NUMBER OF REPEATS DURING LASER SCANNING, R
1140 R = 3        'NUMBER OF REPEATS DURING LASER SCANNING, R
1150 FOR S=1 TO R
1160 CALL ALLEN7R
1170 ' ##### GO TO ASSEMBLY LANGUAGE PROGRAM #####
1180 '----- RETURN FROM ASSEMBLY LANGUAGE PROGRAM -----
1190 '***** DISPLAY DATA CALCULATED IN ASSEMBLY LANGUAGE *****
1200 DEF SEG = &H9FCA
1210 ADDR = &H270
1220 PRINT "DATA OFFSET ADDRESS",,"RATIO"
1230 FOR K=1 TO 3
1240 EL = PEEK(ADDR)
1250 EH = PEEK(ADDR+1)
1260 E=(256*EH+EL)
1270 PRINT "          " HEX$(ADDR),
1280 PRINT USING " #.#.#" ;E
1290 F=E/10000
1300 PRINT ",,,"F
1310 ADDR = ADDR+8
1320 NEXT K
1330 NEXT S
1340 PRINT
1350 ' ***** DISPLAY SAMPLE SIZE OF CALCULATED DATA *****
1360 PRINT "170 M ="M "          'BACKGROUND SAMPLE SIZE, M
1370 N = PEEK(&H208)
1380 PRINT "730' N ="N "          'DATA SAMPLE SIZE, N
1390 PRINT "1140 R ="R"          'NUMBER OF REPEATS DURING LASER SCANNING, R
1400 END
1410 'SAVE"ALLENB7R 11/3/87
Ok

```

TABLE IIB DISPLAY PRODUCED BY RUNNING ALLENB7R.BAS

Break in 1040

Ok

CONT

| DATA OFFSET | ADDRESS | | RATIO |
|-------------|---------|----------|--------|
| | 270 | %21154.0 | |
| | | | 2.1154 |
| | 278 | %31532.0 | |
| | | | 3.1532 |
| | 280 | %42287.0 | |
| | | | 4.2287 |
| DATA OFFSET | ADDRESS | | RATIO |
| | 270 | %21053.0 | |
| | | | 2.1053 |
| | 278 | %31361.0 | |
| | | | 3.1361 |
| | 280 | %42071.0 | |
| | | | 4.2071 |
| DATA OFFSET | ADDRESS | | RATIO |
| | 270 | %21055.0 | |
| | | | 2.1055 |
| | 278 | %31418.0 | |
| | | | 3.1418 |
| | 280 | %42143.0 | |
| | | | 4.2143 |

170 M = 30

730' N = 5

1140 R = 3

Ok

'BACKGROUND SAMPLE SIZE, M

'DATA SAMPLE SIZE, N

'NUMBER OF REPEATS DURING LASER SCANNING, R

TABLE IIIA ASSEMBLY LANGUAGE PROGRAM ALLEN7R.LST

Microsoft (R) Macro Assembler Version 4.00

10/29/87 14:49:17
Page 1-1

```

;ALLEN7R --.EXE file to be controlled by BASIC
;
; Generates a BINARY OUTPUT PULSE (BO2) to
; indicate the START of taking data, follow-
; ed by a BO3 pulse to be used as a marker
; at the start of each set of ADC INPUTS.
; Reads data from AD0 and subtracts W (poked
; into RAM previously from BASICA) and holds
; the DIFFERENCE, then reads data from AD1
; and subtracts X and holds this DIFFERENCE,
; etc. for AD2-Y and AD3-Z. This is repeat N
; time where 2>N>100 which was previously
; poked from BASICA. A maximum of four
; memory locations (200H, 202H, 204H, & 206H)
; are required for the BACKGROUND data pre-
; viously averaged in BASIC; and four memory
; locations (210H, 218H, 220H, & 228H) for
; subtracted data, three memory locations
; (230H, 238H, & 240H) for the RATIO data,
; one (248H) for the value of N, three
; (250H, 258h, & 260H) for the summation of
; RATIOS, and the same three (250H, 258H,
; & 260H) for the MEAN of the RATIOS all
; obtained in this ASSMEBLY LANGUAGE program.
; The values are then multiplied by 10,000d
; to eliminate the decimal point. The prog-
; ram then returns to basic.
;          BASICA PROGRAM -- ALLENB7R.BAS
;-----
.287

;*****

0000          ST_SEG  SEGMENT STACK  ;DEFINE STACK SEGMENT
0000 0014{    DB          20 DUP ('STACK  ')
           53 54 41 43 4B
           20 20 20
           ]

00A0          ST_SEG  ENDS

;*****

0000          PROGNAM SEGMENT          ;DEFINE CODE SEGMENT
;-----
0000          MAIN    PROC    FAR      ;MAIN PART OF PROGRAM
           ASSUME    CS:PROGNAM,DS:NOTHING

0000          START:          ;START EXECUTION ADDR

```

```

;SET UP STACK FOR RETURN (p 439)
0000 55          PUSH BP          ;SAVE BP
0001 1E          PUSH DS          ;SAVE OLD DATA SEGMENT
0002 56          PUSH SI          ;SAVE OLD SI
0003 8C C8       MOV AX,CS       ;DATA SEGMENT SAME AS
0005 8E D8       MOV DS,AX       ; CODE SEGMENT
0007 2B C0       SUB AX,AX       ;ZERO AX
0009 50          PUSH AX          ;SAVE IT ON STACK

;SELECT/HOLD DATA ACQ/CTRL ADAPT--BINARY
000A BA C2E2     MOV DX,0C2E2H  ;REG 12,ADPTR 0,LO B
000D B8 0008     MOV AX,8       ;BINARY DEVICE 8
0010 EE          OUT DX,AL       ; LOW BYTE OUTPUT
0011 BA C3E3     MOV DX,0C3E3H  ;REG 12,ADPTR 0,HI B
0014 B8 0000     MOV AX,0       ;BINARY DEVICE 8
0017 EE          OUT DX,AL       ; HIGH BYTE OUTPUT

;Select Binary Output (Write), register 2
; Set BO 2 TTL output low
0018 BA 22E2     MOV DX,22E2H  ;REG 2,BIN OUT,LO B
001B B8 0000     MOV AX,0       ;BO 2 = 0 (LO & HI)
001E EE          OUT DX,AL       ;OUTPUT LOW BYTE
001F BA 22E3     MOV DX,22E3H  ;REG 2,BIN OUT,HI B
0022 EE          OUT DX,AL       ;OUTPUT HIGH BYTE

; Set BO 2 TTL output high
0023 BA 22E2     MOV DX,22E2H  ;REG 2,BIN OUT,LO B
0026 B0 04       MOV AL,4       ;BO 2 = 1, LO B
0028 EE          OUT DX,AL       ;OUTPUT LOW BYTE
0029 BA 22E3     MOV DX,22E3H  ;REG 2,BIN OUT,HI B
002C B0 00       MOV AL,0       ;HIGH BYTE OF BO 2=0
002E EE          OUT DX,AL       ;OUTPUT HIGH BYTE

; Set BO 2 TTL output low
002F BA 22E2     MOV DX,22E2H  ;REG 2,BIN OUT,LO B
0032 B0 00       MOV AL,0       ;BO 2 = 0 (LO & HI)
0034 EE          OUT DX,AL       ;OUTPUT LOW BYTE
0035 BA 22E3     MOV DX,22E3H  ;REG 2,BIN OUT,HI B
0038 EE          OUT DX,AL       ;OUTPUT HIGH BYTE

;Fill memory offset locations
0039 BB 0210     MOV BX,0210H  ;OFFSET FILL ADDR.
003C 8B FB       MOV DI,BX      ; INTO REG. DI
003E B9 0080     MOV CX,80H    ;# LOCATIONS TO FILL
0041 B8 0000     MOV AX,0000H  ;PUT # INTO AX
0044 89 05       MOV [DI],AX   ;FILL MEMORY WITH #s
0046 47         INC DI        ;INCREASE ADDR. BY 1
0047 E2 FB       LOOP FILL     ;REPEAT FILLING

;Prepare to Take 4N readings
0049 BB 0208     MOV BX,0208H  ;N POKED FROM BASIC is
004C 8B 0F       MOV CX,[BX]   ;# OF READ. TO AVERAGE
004E BB 0210     MOV BX,0210H  ;OFFSET DATA ADDR.
0051 8B FB       MOV DI,BX     ; INTO REG. DI

```



```

0053 B0 04          MOV AL,4          ;# OF ADC CHANNELS
0055 51            NEWSET: PUSH CX        ;HOLD # OF READING 4*N

;SELECT/HOLD DATA ACQ/CTRL ADAPT--BINARY
0056 BA C2E2      MOV DX,0C2E2H    ;REG 12,ADPTR 0,LO B
0059 B8 0008      MOV AX,8         ;BINARY DEVICE 8
005C EE          OUT DX,AL         ; LOW BYTE OUTPUT
005D BA C3E3      MOV DX,0C3E3H    ;REG 12,ADPTR 0,HI B
0060 B8 0000      MOV AX,0         ;BINARY DEVICE 8
0063 EE          OUT DX,AL         ; HIGH BYTE OUTPUT

;Select Binary Output (Write), register 2
; Set BO 3 TTL output high (WAS LOW)
0064 BA 22E2      MOV DX,22E2H    ;REG 2,BIN OUT,LO B
0067 B0 08        MOV AL,8        ;BO 3 = 1, LO B
0069 EE          OUT DX,AL         ;OUTPUT LOW BYTE
006A BA 22E3      MOV DX,22E3H    ;REG 2,BIN OUT,HI B
006D B0 00        MOV AL,0        ;HIGH BYTE OF BO 3=0
006F EE          OUT DX,AL         ;OUTPUT HIGH BYTE

; Set BO 3 TTL output low
0070 BA 22E2      MOV DX,22E2H    ;REG 2,BIN OUT,LO B
0073 B0 00        MOV AL,0        ;BO 3 = 0 (LO & HI)
0075 EE          OUT DX,AL         ;OUTPUT LOW BYTE
0076 BA 22E3      MOV DX,22E3H    ;REG 2,BIN OUT,HI B
0079 EE          OUT DX,AL         ;OUTPUT HIGH BYTE

;SELECT/HOLD DATA ACQ/CTRL ADAPT--ANALOG INPUT
007A BA C2E2      MOV DX,0C2E2H    ;REG 12,ADPTR 0,LO B
007D B0 09        MOV AL,9        ;DEVICE 9 LOW BYTE
007F EE          OUT DX,AL         ;OUTPUT LOW BYTE
0080 BA C2E3      MOV DX,0C2E3H    ;REG 12,ADPTR 0,HI B
0083 B0 00        MOV AL,0        ;DEVICE 9 HIGH BYTE
0085 EE          OUT DX,AL         ;OUTPUT HIGH BYTE

;PREPARE TO TAKE ANALOG INPUT DATA
0086 2B DB        SUB BX,BX        ;ZERO BL (AD# = AD0)

;Select Analog Input Control Reg 0 (AD#)
; Connect AD#. Hold conv (reset bit #0 to 0)
NEXTAD: 0088 BA 02E2      MOV DX,02E2H    ;REG 0,AD# INPUT,LO
008B B0 00        MOV AL,0        ;HOLD CONVERT
008D EE          OUT DX,AL         ;OUTPUT LOW BYTE
008E BA 02E3      MOV DX,02E3H    ;REG 0, AD# INPUT, HI
0091 8A C3        MOV AL,BL       ;CODE FOR AD# 0 thru 3
0093 EE          OUT DX,AL         ;OUTPUT HIGH BYTE

; Delay >20 us (allow transients to settle)
0094 B9 0022      MOV CX,0022H    ;MEASURED DELAY: 26us

0097 E2 FE        TRANS: LOOP TRANS ;EXECUTE DELAY #1

; Start Conversion (set bit #0 to 1)
0099 BA 02E2      MOV DX,02E2H    ;REG 0, AD# INPUT, LO

```

```

009C B0 01          MOV AL,1          ;START CONVERT,LOW B
009E EE            OUT DX,AL         ;OUTPUT LOW BYTE
009F BA 02E3       MOV DX,02E3H     ;REG 0, AD# INPUT, HI
00A2 8A C3         MOV AL,BL        ;CODE FOR AD# 0 thru 3
00A4 EE            OUT DX,AL         ;OUTPUT HIGH BYTE

; Wait for BUSY STATE bit 0 = 0 (Polling)
00A5 BA 02E2       MOV DX,02E2H     ;REG 0, LOW BYTE
00A8 ED            CONV: IN AX,DX          ;IN FROM AI STATUS REG
00A9 25 0001       AND AX,0001H     ;MASK ALL BUT BIT 0
00AC 75 FA         JNZ CONV        ;REPEAT IF BIT 0 NOT 0

; Enable Reading of AD# Channel
00AE BA 02E2       MOV DX,02E2H     ;REG 0, AD# INPUT, LO
00B1 B8 0000       MOV AX,0         ;ENABLE READING, LO B
;ZERO AH FOR FUTURE IN
00B4 EE            OUT DX,AL         ;OUTPUT LOW BYTE
00B5 BA 02E3       MOV DX,02E3H     ;REG 0,A/D 0 INPUT,HI
00B8 8A C3         MOV AL,BL        ;CODE FOR AD# 0 thru 3
00BA EE            OUT DX,AL         ;OUTPUT HIGH BYTE

; Read AD# Input (register 2--see p71)
00BB BA 22E2       MOV DX,22E2H     ;REG 2, AD# INPUT, LO
00BE ED            IN AX,DX          ;INPUT READING

;Subtract background from A/D inputs
00BF 57            PUSH DI         ;SAVE DATA ADDR.
00C0 BA 0200       MOV DX,0200H     ;BACKGROUND DATA
00C3 03 D3         ADD DX,BX        ; ADDRESS
00C5 03 D3         ADD DX,BX        ; + 2 TIMES
00C7 8B FA         MOV DI,DX        ; BL (AD#)
00C9 8B 15         MOV DX,[DI]     ;GET BACKGROUND
00CB 2B C2         SUB AX,DX        ;DIFFERENCE IN AX

;Store Difference in memory
; AD0-W, AD1-X, AD2-Y, & AD3-Z
00CD 5F            POP DI         ;DIFFERENCE ADDRESS
00CE 89 05         MOV [DI],AX     ;DIFFERENCE IN MEMORY
00D0 83 C7 08      ADD DI,8        ;INCREASE ADDR BY 8

;Obtain another AD# (channel # 1, 2 or 3)
00D3 FE C3         INC BL          ;INCREASE AD# BY 1
00D5 80 FB 04     CMP BL,4        ;IS BL > 3 i.e. (= 4)
00D8 75 AE         JNZ NEXTAD     ;NO? THEN REPEAT

;Take three ratios:
; AD1-X/AD0-W, AD2-Y/AD0-W, AD3-Z/AD0-W
00DA BB 0210       MOV BX,0210H    ;ADDR OF DIVISOR AD0-W
00DD 8B C3         MOV AX,BX       ;COPY INTO AX
00DF 05 0008       ADD AX,8        ;ADDRESS OF DIVIDEND
00E2 8B F8         MOV DI,AX       ; AD1-X IN DI
00E4 8B CB         MOV CX,BX       ;ADDR OF AD1-X/AD0-W
00E6 83 C1 20      ADD CX,20H     ; OFFSET FROM DIVISOR

```

```

00E9 8B F1          MOV SI,CX          ; BY 20H (IN SI)
00EB B9 0003       MOV CX,3           ; 3 RATIOS
00EE 9B DB 05      NXRAT: FILD DWORD PTR [DI] ;LD DIVIDEND-80287
00F1 9B DA 37      FIDIV DWORD PTR [BX];DIVIDE INTEGER
00F4 9B D9 1C      FSTP DWORD PTR [SI] ;STORE REAL & POP
00F7 83 C7 08      ADD DI,8           ;NEXT AD# DIFFERENCE
00FA 83 C6 08      ADD SI,8           ;NEXT AD# RATIO ADDR
00FD E2 EF          LOOP NXRAT         ;CALCULATE NEXT RATIO

;Sum the three ratios
00FF 83 C3 20      ADD BX,20H         ;ADDR OF AD1-X/ADO-W
0102 8B FB          MOV DI,BX          ; IN DI
0104 8B C3          MOV AX,BX          ;ADDR SUM AD1-X/ADO-W
0106 05 0020       ADD AX,20H         ; 20H HIGHER ADDR
0109 8B F0          MOV SI,AX          ; IN SI
010B B9 0003       MOV CX,3           ; 3 SUMS
010E 9B D9 04      NXSUM: FLD DWORD PTR [SI] ;LD REAL SUM
0111 9B D8 05      FADD DWORD PTR [DI] ;ADD REAL
0114 9B D9 1C      FSTP DWORD PTR [SI] ;STORE REAL & POP
0117 83 C7 08      ADD DI,8           ;ADDR OF NEXT RATIO
011A 83 C6 08      ADD SI,8           ;ADDR OF NEXT SUM
011D E2 EF          LOOP NXSUM         ;CALCULATE NEXT SUM

;Prepare to obtain another set of 4 AD# read.
011F BB 0210       MOV BX,0210H       ;OFFSET DATA ADDR.
0122 8B FB          MOV DI,BX          ; INTO PTR DI
0124 BB 0000       MOV BX,0000H       ;SELECT AD# = ADO
0127 59             POP CX              ;# READINGS REMAINING
0128 49             DEC CX              ; DECREASE BY 1
0129 E3 03          JCXZ MEAN          ;END IF CX = 0
012B E9 0055 R     JMP NEWSET         ;OTHERWISE LOOP

;Calculate mean (divide each sum by N)
012E B9 0003       MEAN: MOV CX,3        ; 3 GROUPS OF SUMS
0131 BB 0208       MOV BX,00208H      ;ADDRESS
0134 8B FB          MOV DI,BX          ; PTR FOR N
0136 BB 0250       MOV BX,0250H       ;OFFSET SUM ADDR.
0139 8B F3          MOV SI,BX          ; INTO PTR SI
013B 9B D9 04      NXMEAN: FLD DWORD PTR [SI] ;LOAD INTO 80287
013E 9B DA 35      FIDIV DWORD PTR [DI] ;DIVIDE BY N
0141 9B D9 1C      FSTP DWORD PTR [SI] ;STORE & POP MEAN
0144 83 C6 08      ADD SI,8           ;NEXT OFFSET SUM ADDR
0147 E2 F2          LOOP NXMEAN        ;REPEAT FOR NEXT MEAN

;Multiply by 10000d and change to integer
0149 BB 0268       MOV BX,0268H       ;ADDRESS OF MULTIPLIER
014C B8 2710       MOV AX,2710H       ;MOV+ 10000d MULTIPL.
014F 89 07          MOV [BX],AX        ; INTO ADDR. 0268H
0151 BA 0250       MOV DX,0250H       ;OFFSET MEAN ADDR
0154 8B FA          MOV DI,DX          ; POINTED TO BY DI
0156 BA 0270       MOV DX,0270H       ;ADDRESS OF ANSWER
0159 8B F2          MOV SI,DX          ; POINTED BY SI
015B B9 0003       MOV CX,3           ;REPEAT FOR 3 MEANS

```

```

015E 9B D9 05          CONINT: FLD DWORD PTR [DI] ;LOAD MEAN IN 80287
0161 9B DA 0F          FIMUL DWORD PTR [BX];MULTIPLY X 10000D
0164 9B DB 1C          FISTP DWORD PTR [SI] ;STORE & POP ANS
0167 83 C7 08          ADD DI,8 ;POINT NEXT MEAN
016A 83 C6 08          ADD SI,8 ;POINT NEXT ANSWER
016D E2 EF          LOOP CONINT ;NEXT CONVERT2INTEGER

```

```

;Stop After 3 ANSWERS in memory--short integer
016F 58          POP AX
0170 5E          POP SI
0171 1F          POP DS
0172 5D          POP BP
0173 90          NOP
0174 90          NOP
0175 CB          RET ;RTN BASIC
0176 90          NOP
0177 90          NOP

```

```

0178 0190[  MAIN ENDP ;END OF MAIN PART OF PROGRAM
      ??   DB 400D DUP(?) ;RESERVE DATA AREA

```

]

0308

```

;-----
PROGNAM ENDS
;*****

```

END START ;END ASSEMBLY

Segments and Groups:

| Name | Size | Align | Combine Class |
|-------------------|------|-------|---------------|
| PROGNAM | 0308 | PARA | NONE |
| ST_SEG | 00A0 | PARA | STACK |

Symbols:

| Name | Type | Value | Attr |
|-------------------|--------|-------|-----------------------|
| CONINT | L NEAR | 015E | PROGNAM |
| CONV | L NEAR | 00A8 | PROGNAM |
| FILL | L NEAR | 0044 | PROGNAM |
| MAIN | F PROC | 0000 | PROGNAM Length = 0178 |
| MEAN | L NEAR | 012E | PROGNAM |
| NEWSSET | L NEAR | 0055 | PROGNAM |
| NEXTAD | L NEAR | 0088 | PROGNAM |
| NXMEAN | L NEAR | 013B | PROGNAM |
| NXRAT | L NEAR | 00EE | PROGNAM |
| NXSUM | L NEAR | 010E | PROGNAM |
| START | L NEAR | 0000 | PROGNAM |
| TRANS | L NEAR | 0097 | PROGNAM |

288 Source Lines
 288 Total Lines
 35 Symbols

49350 Bytes symbol space free

0 Warning Errors
 0 Severe Errors

C:\BG>

TABLE IIIB ASSEMBLY LANGUAGE PROGRAM ALLEN7R.EXE

SYMDEB ALLEN7R.EXE

Microsoft (R) Symbolic Debug Utility Version 4.00

Copyright (C) Microsoft Corp 1984, 1985. All rights reserved.

Processor is [80286]

-R

```

AX=0000 BX=0000 CX=03A8 DX=0000 SP=00A0 BP=0000 SI=0000 DI=0000
DS=2E17 ES=2E17 SS=9FC0 CS=9FCA IP=0000 NV UP EI FL NZ NA PO NC
9FCA:0000 55          PUSH  BP
-U0,177
9FCA:0000 55          PUSH  BP
9FCA:0001 1E          PUSH  DS
9FCA:0002 56          PUSH  SI
9FCA:0003 8CC8        MOV   AX,CS
9FCA:0005 8ED8        MOV   DS,AX
9FCA:0007 2BC0        SUB   AX,AX
9FCA:0009 50          PUSH  AX
9FCA:000A BAE2C2      MOV   DX,C2E2
9FCA:000D B80800    MOV   AX,0008
9FCA:0010 EE          OUT   DX,AL
9FCA:0011 BAE3C3      MOV   DX,C3E3
9FCA:0014 B80000    MOV   AX,0000
9FCA:0017 EE          OUT   DX,AL
9FCA:0018 BAE222      MOV   DX,22E2
9FCA:001B B80000    MOV   AX,0000
9FCA:001E EE          OUT   DX,AL
9FCA:001F BAE322      MOV   DX,22E3
9FCA:0022 EE          OUT   DX,AL
9FCA:0023 BAE222      MOV   DX,22E2
9FCA:0026 B004        MOV   AL,04
9FCA:0028 EE          OUT   DX,AL
9FCA:0029 BAE322      MOV   DX,22E3
9FCA:002C B000        MOV   AL,00
9FCA:002E EE          OUT   DX,AL
9FCA:002F BAE222      MOV   DX,22E2
9FCA:0032 B000        MOV   AL,00
9FCA:0034 EE          OUT   DX,AL
9FCA:0035 BAE322      MOV   DX,22E3
9FCA:0038 EE          OUT   DX,AL
9FCA:0039 BB1002    MOV   BX,0210
9FCA:003C 8BFB        MOV   DI,BX
9FCA:003E B98000    MOV   CX,0080
9FCA:0041 B80000    MOV   AX,0000
9FCA:0044 8905        MOV   [DI],AX
9FCA:0046 47          INC   DI
9FCA:0047 E2FB        LOOP  0044
9FCA:0049 BB0802    MOV   BX,0208
9FCA:004C 8B0F        MOV   CX,[BX]
9FCA:004E BB1002    MOV   BX,0210
9FCA:0051 8BFB        MOV   DI,BX
9FCA:0053 B004        MOV   AL,04
9FCA:0055 51          PUSH  CX
9FCA:0056 BAE2C2      MOV   DX,C2E2
9FCA:0059 B80800    MOV   AX,0008
9FCA:005C EE          OUT   DX,AL
9FCA:005D BAE3C3      MOV   DX,C3E3
9FCA:0060 B80000    MOV   AX,0000
9FCA:0063 EE          OUT   DX,AL

```

| | | | |
|-----------|--------|------|---------|
| 9FCA:0064 | BAE222 | MOV | DX,22E2 |
| 9FCA:0067 | B008 | MOV | AL,08 |
| 9FCA:0069 | EE | OUT | DX,AL |
| 9FCA:006A | BAE322 | MOV | DX,22E3 |
| 9FCA:006D | B000 | MOV | AL,00 |
| 9FCA:006F | EE | OUT | DX,AL |
| 9FCA:0070 | BAE222 | MOV | DX,22E2 |
| 9FCA:0073 | B000 | MOV | AL,00 |
| 9FCA:0075 | EE | OUT | DX,AL |
| 9FCA:0076 | BAE322 | MOV | DX,22E3 |
| 9FCA:0079 | EE | OUT | DX,AL |
| 9FCA:007A | BAE2C2 | MOV | DX,C2E2 |
| 9FCA:007D | B009 | MOV | AL,09 |
| 9FCA:007F | EE | OUT | DX,AL |
| 9FCA:0080 | BAE3C2 | MOV | DX,C2E3 |
| 9FCA:0083 | B000 | MOV | AL,00 |
| 9FCA:0085 | EE | OUT | DX,AL |
| 9FCA:0086 | 2BDB | SUB | BX,BX |
| 9FCA:0088 | BAE202 | MOV | DX,02E2 |
| 9FCA:008B | B000 | MOV | AL,00 |
| 9FCA:008D | EE | OUT | DX,AL |
| 9FCA:008E | BAE302 | MOV | DX,02E3 |
| 9FCA:0091 | 8AC3 | MOV | AL,BL |
| 9FCA:0093 | EE | OUT | DX,AL |
| 9FCA:0094 | B92200 | MOV | CX,0022 |
| 9FCA:0097 | E2FE | LOOP | 0097 |
| 9FCA:0099 | BAE202 | MOV | DX,02E2 |
| 9FCA:009C | B001 | MOV | AL,01 |
| 9FCA:009E | EE | OUT | DX,AL |
| 9FCA:009F | BAE302 | MOV | DX,02E3 |
| 9FCA:00A2 | 8AC3 | MOV | AL,BL |
| 9FCA:00A4 | EE | OUT | DX,AL |
| 9FCA:00A5 | BAE202 | MOV | DX,02E2 |
| 9FCA:00A8 | ED | IN | AX,DX |
| 9FCA:00A9 | 250100 | AND | AX,0001 |
| 9FCA:00AC | 75FA | JNZ | 00A8 |
| 9FCA:00AE | BAE202 | MOV | DX,02E2 |
| 9FCA:00B1 | B80000 | MOV | AX,0000 |
| 9FCA:00B4 | EE | OUT | DX,AL |
| 9FCA:00B5 | BAE302 | MOV | DX,02E3 |
| 9FCA:00B8 | 8AC3 | MOV | AL,BL |
| 9FCA:00BA | EE | OUT | DX,AL |
| 9FCA:00BB | BAE222 | MOV | DX,22E2 |
| 9FCA:00BE | ED | IN | AX,DX |
| 9FCA:00BF | 57 | PUSH | DI |
| 9FCA:00C0 | BA0002 | MOV | DX,0200 |
| 9FCA:00C3 | 03D3 | ADD | DX,BX |
| 9FCA:00C5 | 03D3 | ADD | DX,BX |
| 9FCA:00C7 | 8BFA | MOV | DI,DX |
| 9FCA:00C9 | 8B15 | MOV | DX,[DI] |
| 9FCA:00CB | 2BC2 | SUB | AX,DX |
| 9FCA:00CD | 5F | POP | DI |
| 9FCA:00CE | 8905 | MOV | [DI],AX |
| 9FCA:00D0 | 83C708 | ADD | DI,+08 |
| 9FCA:00D3 | FEC3 | INC | BL |
| 9FCA:00D5 | 80FB04 | CMP | BL,04 |
| 9FCA:00D8 | 75AE | JNZ | 0088 |

| | | | |
|-----------|--------|-------|----------------|
| 9FCA:00DA | BB1002 | MOV | BX,0210 |
| 9FCA:00DD | 8BC3 | MOV | AX,BX |
| 9FCA:00DF | 050800 | ADD | AX,0008 |
| 9FCA:00E2 | 8BF8 | MOV | DI,AX |
| 9FCA:00E4 | 8BCB | MOV | CX,BX |
| 9FCA:00E6 | 83C120 | ADD | CX,+20 |
| 9FCA:00E9 | 8BF1 | MOV | SI,CX |
| 9FCA:00EB | B90300 | MOV | CX,0003 |
| 9FCA:00EE | 9B | WAIT | |
| 9FCA:00EF | DB05 | FLD | DWord Ptr [DI] |
| 9FCA:00F1 | 9B | WAIT | |
| 9FCA:00F2 | DA37 | FIDIV | DWord Ptr [BX] |
| 9FCA:00F4 | 9B | WAIT | |
| 9FCA:00F5 | D91C | FSTP | DWord Ptr [SI] |
| 9FCA:00F7 | 83C708 | ADD | DI,+08 |
| 9FCA:00FA | 83C608 | ADD | SI,+08 |
| 9FCA:00FD | E2EF | LOOP | 00EE |
| 9FCA:00FF | 83C320 | ADD | BX,+20 |
| 9FCA:0102 | 8BFB | MOV | DI,BX |
| 9FCA:0104 | 8BC3 | MOV | AX,BX |
| 9FCA:0106 | 052000 | ADD | AX,0020 |
| 9FCA:0109 | 8BF0 | MOV | SI,AX |
| 9FCA:010B | B90300 | MOV | CX,0003 |
| 9FCA:010E | 9B | WAIT | |
| 9FCA:010F | D904 | FLD | DWord Ptr [SI] |
| 9FCA:0111 | 9B | WAIT | |
| 9FCA:0112 | D805 | FADD | DWord Ptr [DI] |
| 9FCA:0114 | 9B | WAIT | |
| 9FCA:0115 | D91C | FSTP | DWord Ptr [SI] |
| 9FCA:0117 | 83C708 | ADD | DI,+08 |
| 9FCA:011A | 83C608 | ADD | SI,+08 |
| 9FCA:011D | E2EF | LOOP | 010E |
| 9FCA:011F | BB1002 | MOV | BX,0210 |
| 9FCA:0122 | 8BFB | MOV | DI,BX |
| 9FCA:0124 | BB0000 | MOV | BX,0000 |
| 9FCA:0127 | 59 | POP | CX |
| 9FCA:0128 | 49 | DEC | CX |
| 9FCA:0129 | E303 | JCXZ | 012E |
| 9FCA:012B | E927FF | JMP | 0055 |
| 9FCA:012E | B90300 | MOV | CX,0003 |
| 9FCA:0131 | BB0802 | MOV | BX,0208 |
| 9FCA:0134 | 8BFB | MOV | DI,BX |
| 9FCA:0136 | BB5002 | MOV | BX,0250 |
| 9FCA:0139 | 8BF3 | MOV | SI,BX |
| 9FCA:013B | 9B | WAIT | |
| 9FCA:013C | D904 | FLD | DWord Ptr [SI] |
| 9FCA:013E | 9B | WAIT | |
| 9FCA:013F | DA35 | FIDIV | DWord Ptr [DI] |
| 9FCA:0141 | 9B | WAIT | |
| 9FCA:0142 | D91C | FSTP | DWord Ptr [SI] |
| 9FCA:0144 | 83C608 | ADD | SI,+08 |
| 9FCA:0147 | E2F2 | LOOP | 013B |
| 9FCA:0149 | BB6802 | MOV | BX,0268 |
| 9FCA:014C | B81027 | MOV | AX,2710 |
| 9FCA:014F | 8907 | MOV | [BX],AX |

| | | | |
|-----------|--------|-------|----------------|
| 9FCA:0151 | BA5002 | MOV | DX,0250 |
| 9FCA:0154 | 8BFA | MOV | DI,DX |
| 9FCA:0156 | BA7002 | MOV | DX,0270 |
| 9FCA:0159 | 8BF2 | MOV | SI,DX |
| 9FCA:015B | B90300 | MOV | CX,0003 |
| 9FCA:015E | 9B | WAIT | |
| 9FCA:015F | D905 | FLD | DWord Ptr [DI] |
| 9FCA:0161 | 9B | WAIT | |
| 9FCA:0162 | DA0F | FIMUL | DWord Ptr [BX] |
| 9FCA:0164 | 9B | WAIT | |
| 9FCA:0165 | DB1C | FISTP | DWord Ptr [SI] |
| 9FCA:0167 | 83C708 | ADD | DI,+08 |
| 9FCA:016A | 83C608 | ADD | SI,+08 |
| 9FCA:016D | E2EF | LOOP | 015E |
| 9FCA:016F | 58 | POP | AX |
| 9FCA:0170 | 5E | POP | SI |
| 9FCA:0171 | 1F | POP | DS |
| 9FCA:0172 | 5D | POP | BP |
| 9FCA:0173 | 90 | NOP | |
| 9FCA:0174 | 90 | NOP | |
| 9FCA:0175 | CB | RETF | |
| 9FCA:0176 | 90 | NOP | |
| 9FCA:0177 | 90 | NOP | |

-



Report Documentation Page

| | | | | | |
|---|--|---|---|---|-------------------------|
| 1. Report No. NASA CR-4117 | | 2. Government Accession No. | | 3. Recipient's Catalog No. | |
| 4. Title and Subtitle High-Speed Assembly Language (80386/80387) Programming for Laser Spectra Scan Control and Data Acquisition Providing Improved Resolution Water Vapor Spectroscopy | | | | 5. Report Date February 1988 | |
| | | | | 6. Performing Organization Code | |
| 7. Author(s) Robert J. Allen | | | | 8. Performing Organization Report No. 87101 | |
| | | | | 10. Work Unit No. 176-40-04-70 | |
| 9. Performing Organization Name and Address Vigyan Research Associates, Inc. 28 Research Drive Hampton, VA 23666 | | | | 11. Contract or Grant No. NAS1-17919 (Task 28B) | |
| | | | | 13. Type of Report and Period Covered Contractor Report May 1, 1986 - Oct. 31, 1987 | |
| 12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225 | | | | 14. Sponsoring Agency Code | |
| | | | | | |
| 15. Supplementary Notes Langley Technical Monitor: Edward V. Browell Final Report in support of the LASE Project to redefine the water vapor lines with improved accuracy in the 725-730 nm region. ALLEN ASSOCIATES, Subcontractor Robert J. Allen: Allen Associates, 205 Tabb Lane, Newport News, VA 23602 | | | | | |
| 16. Abstract <p>An assembly language program using the Intel 80386 CPU and 80387 math co-processor chips was written to increase the speed of data gathering and processing, and provide control of a scanning CW ring dye laser system. This laser system is used in high resolution (better than 0.001 cm-1) water vapor spectroscopy experiments. Laser beam power is sensed at the input and output of white cells and the output of a Fabry-Perot. The assembly language subroutine is called from Basic, acquires the data and performs various calculations at rates greater than 150 faster than could be performed by the higher level language. The width of output control pulses generated in assembly language are 3 to 4 microseconds as compared to 2 to 3.7 milliseconds for those generated in Basic (about 500 to 1000 times faster).</p> <p>Included are a block diagram and brief description of the spectroscopy experiment, a flow diagram of the basic and assembly language programs, listing of the programs, scope photographs of the computer generated 5-volt pulses used for control and timing analysis, and representative water spectrum curves obtained using these programs.</p> | | | | | |
| 17. Key Words (Suggested by Author(s)) 80386 and 80387 Assembly Language programming. Improved resolution water vapor spectroscopy. Assembly language programming for Laser Spectra control and high speed data acquisition/processing. | | | 18. Distribution Statement Unclassified - Unlimited Subject Category 60 | | |
| 19. Security Classif. (of this report) Unclassified | | 20. Security Classif. (of this page) Unclassified | | 21. No. of pages 38 | 22. Price A03 |