

ADA 085437

LEVEL III

②

⑪ 1976

⑨ FINAL REPORT

⑥ REPORT ON CONTRACT F05611-76-90203
Volume II

DTIC
ELECTE
JUN 16 1980
S D

Volume II of II

⑮ F05611-76-90203

⑫ 78

THE COPY FURNISHED TO DDC CONTAINED A
SIGNIFICANT NUMBER OF PAGES WHICH DO NOT
REPRODUCE LEGIBLY.

This document has been approved
for public release and sale; its
distribution is unlimited.

Prepared for:

Capt. D. R. Stevens, Chief
Inertial Guidance Research Division
The Frank J. Seiler Research Laboratory
USAF Academy, Colorado 80240

Prepared by:

⑩ Dr. G. J. Grimes
Kappa Systems, Inc.

1409 Potter Drive
Colorado Springs, Colorado 80909

80 6 11 006

gm

DISCLAIMER NOTICE

**THIS DOCUMENT IS BEST QUALITY
PRACTICABLE. THE COPY FURNISHED
TO DTIC CONTAINED A SIGNIFICANT
NUMBER OF PAGES WHICH DO NOT
REPRODUCE LEGIBLY.**

TABLE OF CONTENTS

	<u>Page</u>
1. A BRIEF SUMMARY OF THE DIGITAL FILTER WORK	1
2. PROJECT IDENTIFICATION	4
3. ANALYSIS FOR FILTER STABILITY AND FORMULATION OF DESIRED TRANSFER FUNCTION	7
4. DIGITAL FILTERS AND PHASE RELATIONSHIPS	10
5. RECURSIVE FILTERS	13
6. HARDWARE SYSTEM CONFIGURATION	18
7. VERIFICATION OF THE DIGITAL FILTER EXECUTION PROGRAM	20
8. THE VARIABLE TIME DELAY FILTERS	22
9. THE INFINITE IMPULSE RESPONSE FILTERS	30
10. THE FIR COEFFICIENT SYNTHESIS AND EXECUTION PROGRAMS	33
11. THE FIR FILTER RESULTS	46
12. RESULTS OF THE INFINITE IMPULSE RESPONSE FILTER COEFFICIENT SYNTHESIS PROGRAM	53
13. VERIFICATION OF THE INFINITE IMPULSE RESPONSE, FILTERS IN REAL-TIME	72
14. SUMMARY OF ISOPAD DIGITAL FILTER WORK FOR ISOPAD CONTROL	82
15. THE MOST SIGNIFICANT TECHNICAL CONTRIBUTIONS	86
16. SUMMARY OF THE PUBLICATIONS	88
17. SUMMARY OF THE PATENT PROCEEDINGS ON THE FIBER OPTIC SLIP RINGS	89
APPENDIX A: SUMMARY OF SOFTWARE	90
APPENDIX B: PROGRAM LISTING: INOUT.FTN	93
APPENDIX C: PROGRAM LISTING: GOOD.FTN	96
APPENDIX D: PROGRAM LISTING: ASKFIR.FTN	100
APPENDIX E: PROGRAM LISTING: DELAY13.FTN	113
APPENDIX F: PROGRAM LISTING: BAD.FTN, BAND.FTN, NINE9.FTN, DIFF12.FTN	117

V1

A085652

TABLE OF CONTENTS

	<u>Page</u>
APPENDIX G: PROGRAM LISTING: CASE8.FTN, CASE8F.FTN, HONEY.FTN, and BUTTER.FTN	130
APPENDIX H: THE DATEL 256 DRIVER PROGRAM: IDAC.MAC	143

Accession For	
NTIS GRA&I	
DDC TAB	
Unannounced	
Justification <i>has been</i>	
<i>50 on file</i>	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or special
<i>A</i>	<i>25 NH</i>

SECTION 13

VERIFICATION OF THE INFINITE IMPULSE RESPONSE FILTERS IN REAL-TIME

Two sets of data are presented here. Neither of them show the expected results, presumably because of the fixed time delay going through the filter.

Figure 13-1 shows the amplitude and phase portions of the transfer functions for the lowpass Butterworth filter. The magnitude portion of the transfer function is similar to that produced by the Burroughs 6700, but the phase portion of the transfer function does not agree at all. It resembles what might be expected from a lowpass analog filter.

Figures 13-2 and 13-3 show the results for the bandpass Chebychev filter. Here again, the results were different from what we expected after studying the Burroughs 6700 plots. Hopefully longer tests will verify the Burroughs 6700 results.

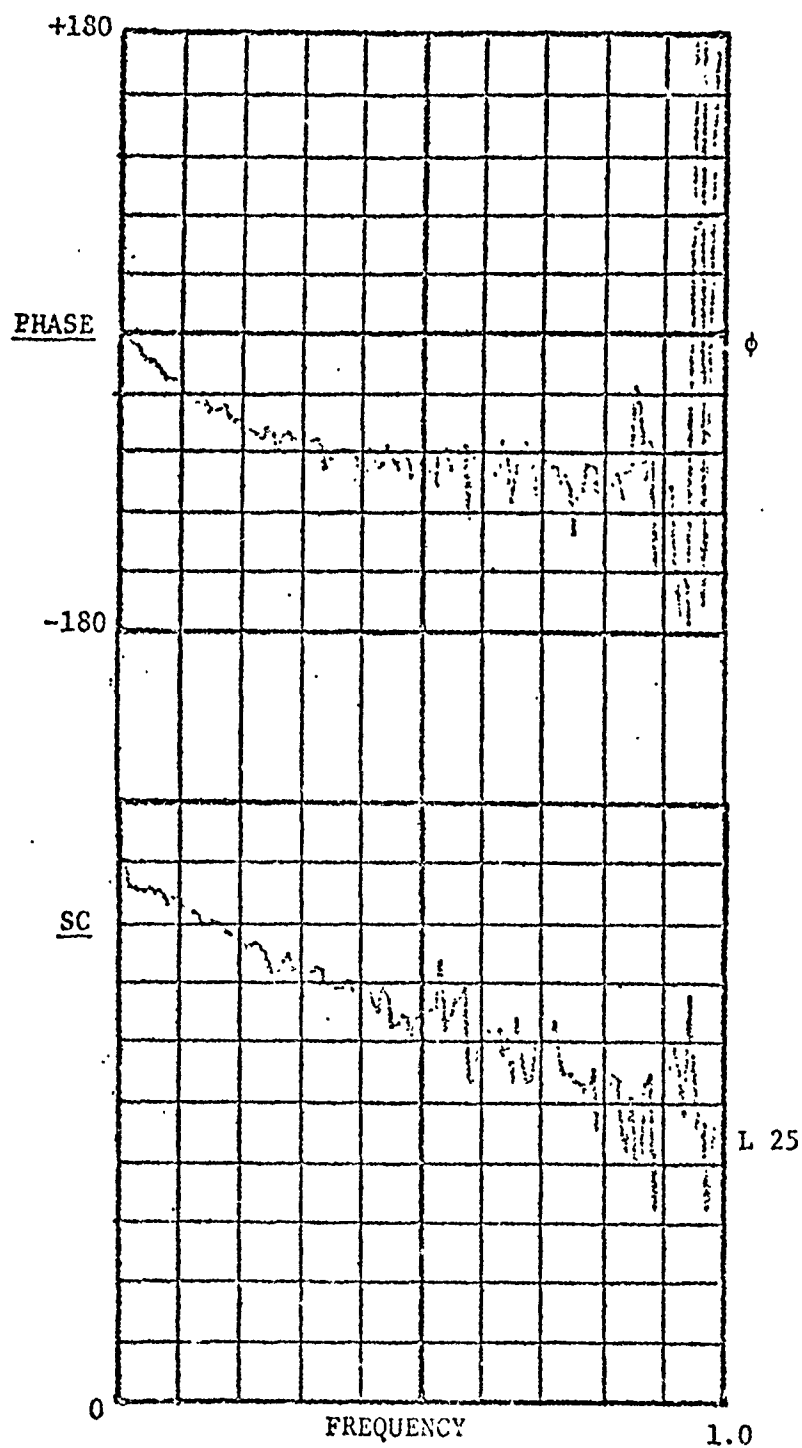


Figure 13-1 Transfer Function for Lowpass Butterworth.

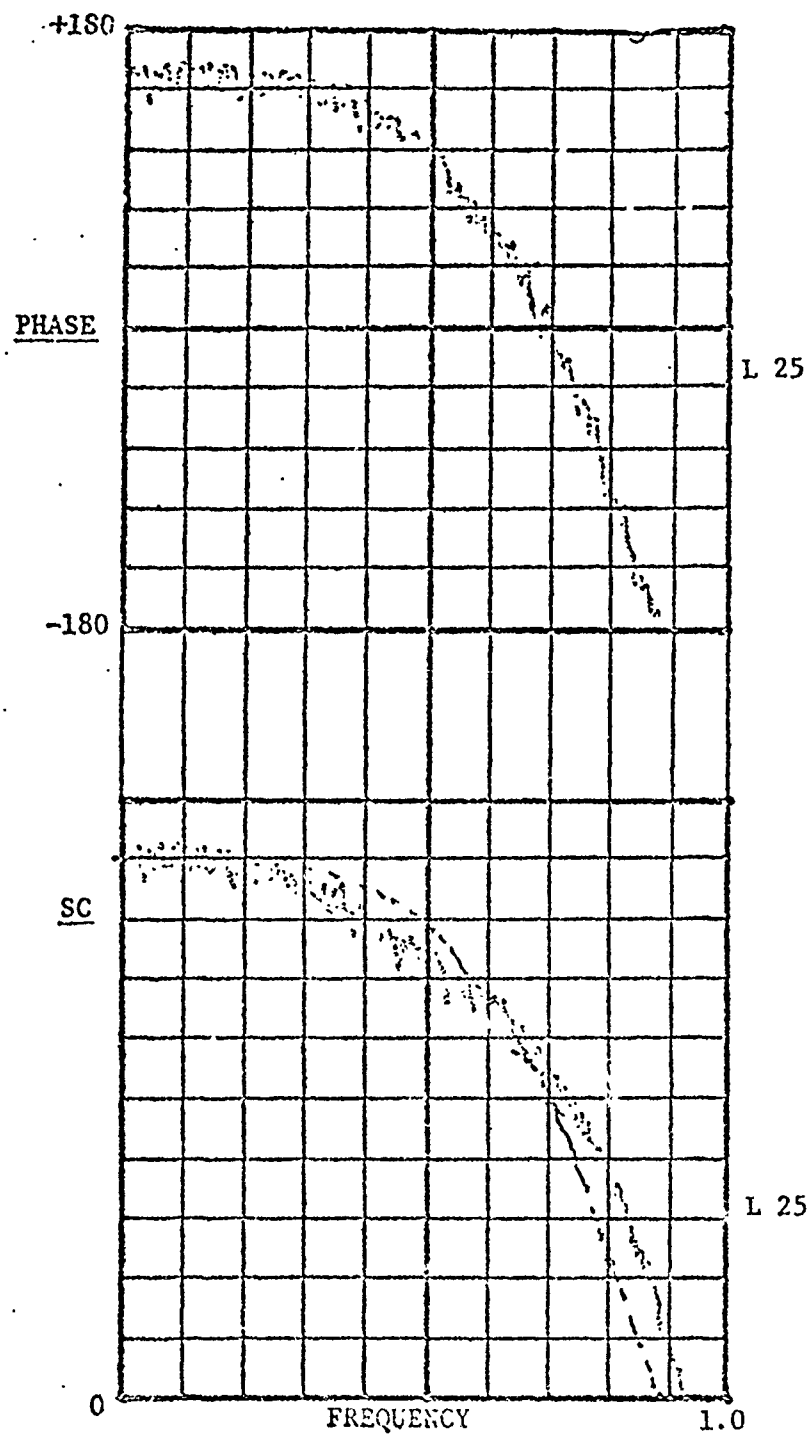


Figure 13-2 Transfer Function Magnitude for Bandpass Chebyshev.

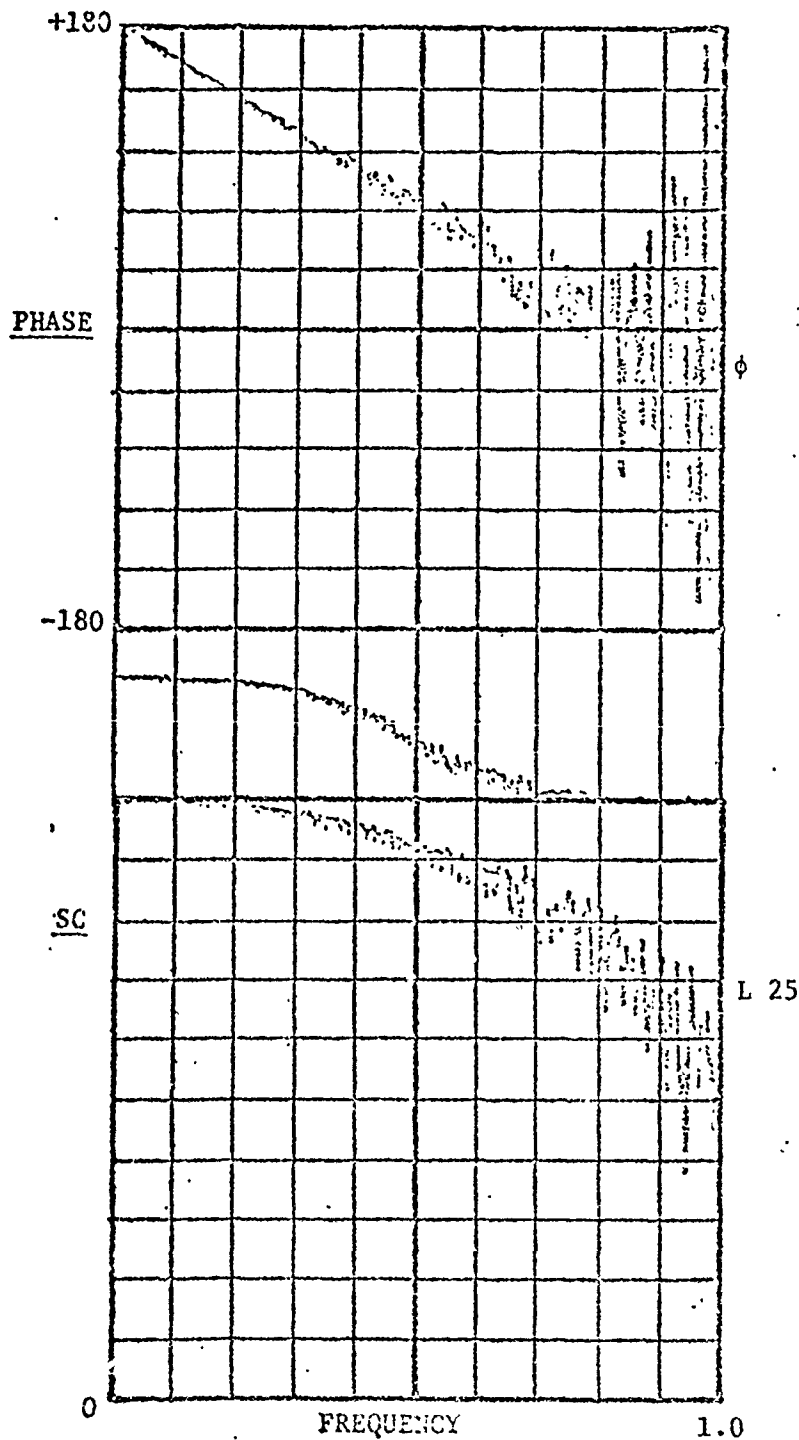


Figure 13-3 Transfer Function Phase for Chebyshev Bandpass.

Perhaps the most interesting of the IIR filters tested was a 0.015-0.045 Hz notch filter. Figure 13-4 shows the notch found in the overnight test. The notch did not appear in the short test in Figure 13-5.

The results shown in Figures 13-6 and 13-7 are remarkable. This was a long overnight test of the same 0.015 - 0.045 Hz IIR notch filter. The linear phase lead was the result of the Biomation anti-aliasing filter being used in a passive mode (bf). The lead is real. What is perhaps even more remarkable, although the data is not presented here, is that when the input and output filters were reversed, the result was a flat transfer function with a slight amount of lag, but not nearly so much lead as is shown in Figure 13-6. This shows that digital and passive analog filters may be used together to achieve remarkable (linear phase lead!) results. It also shows the nonlinear effects of using the two types of filters together. They obviously do not commute.

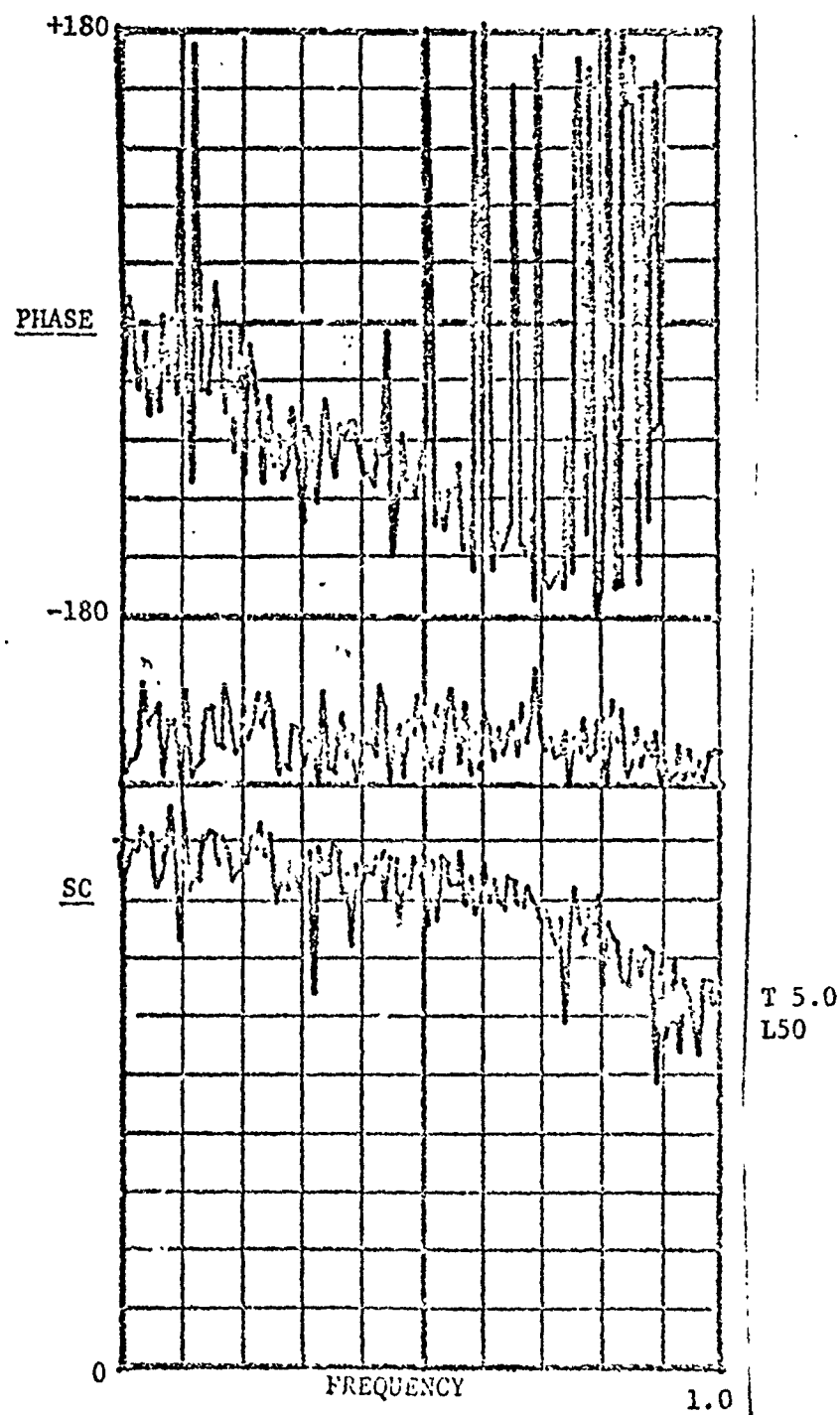


Figure 13-4. Overnight Test of
0.015 Hz to 0.045 Hz notch filter.

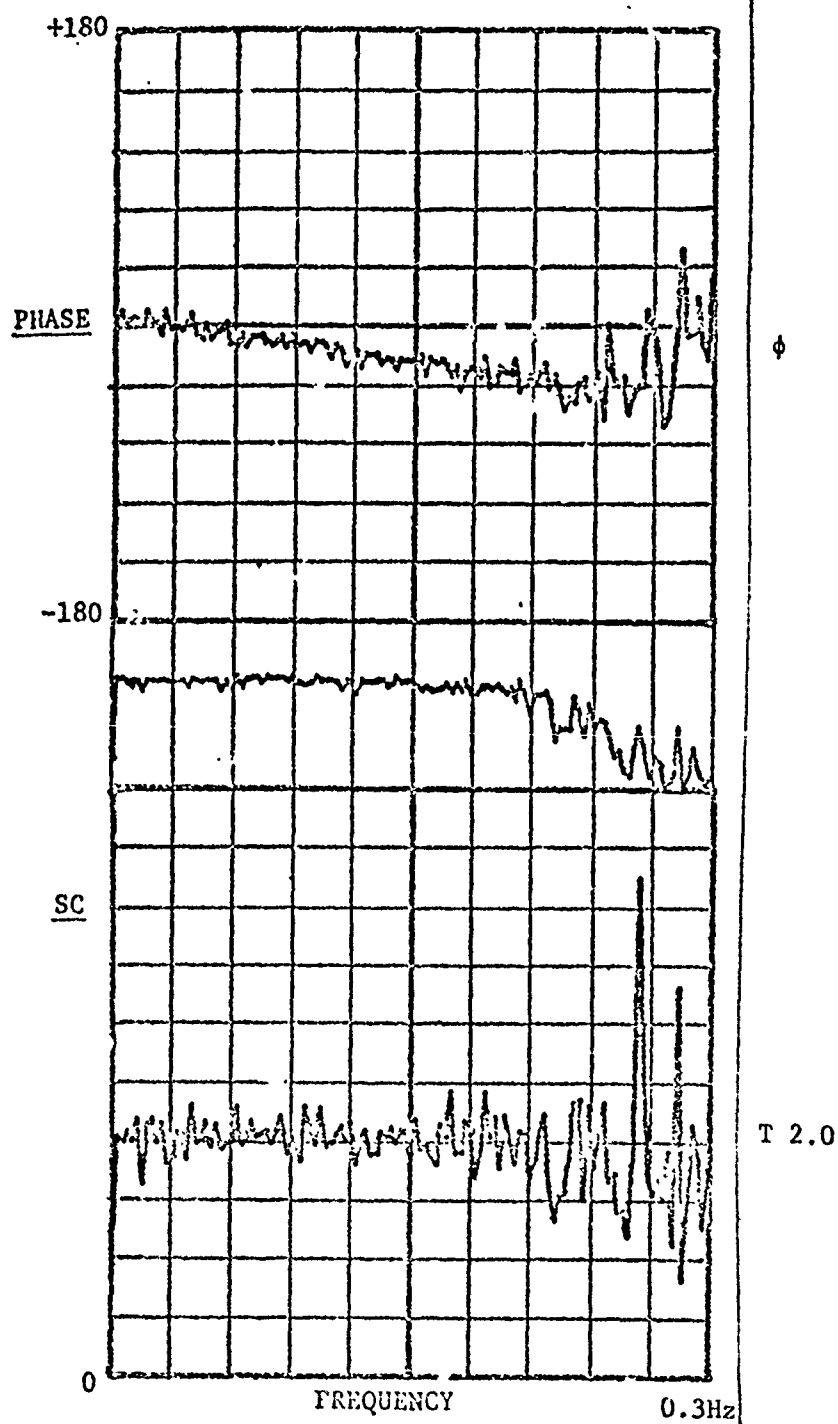


Figure 13-5. IIR Notch Filter, 0.015- 0.045 Hz, Short (40 min.) run.

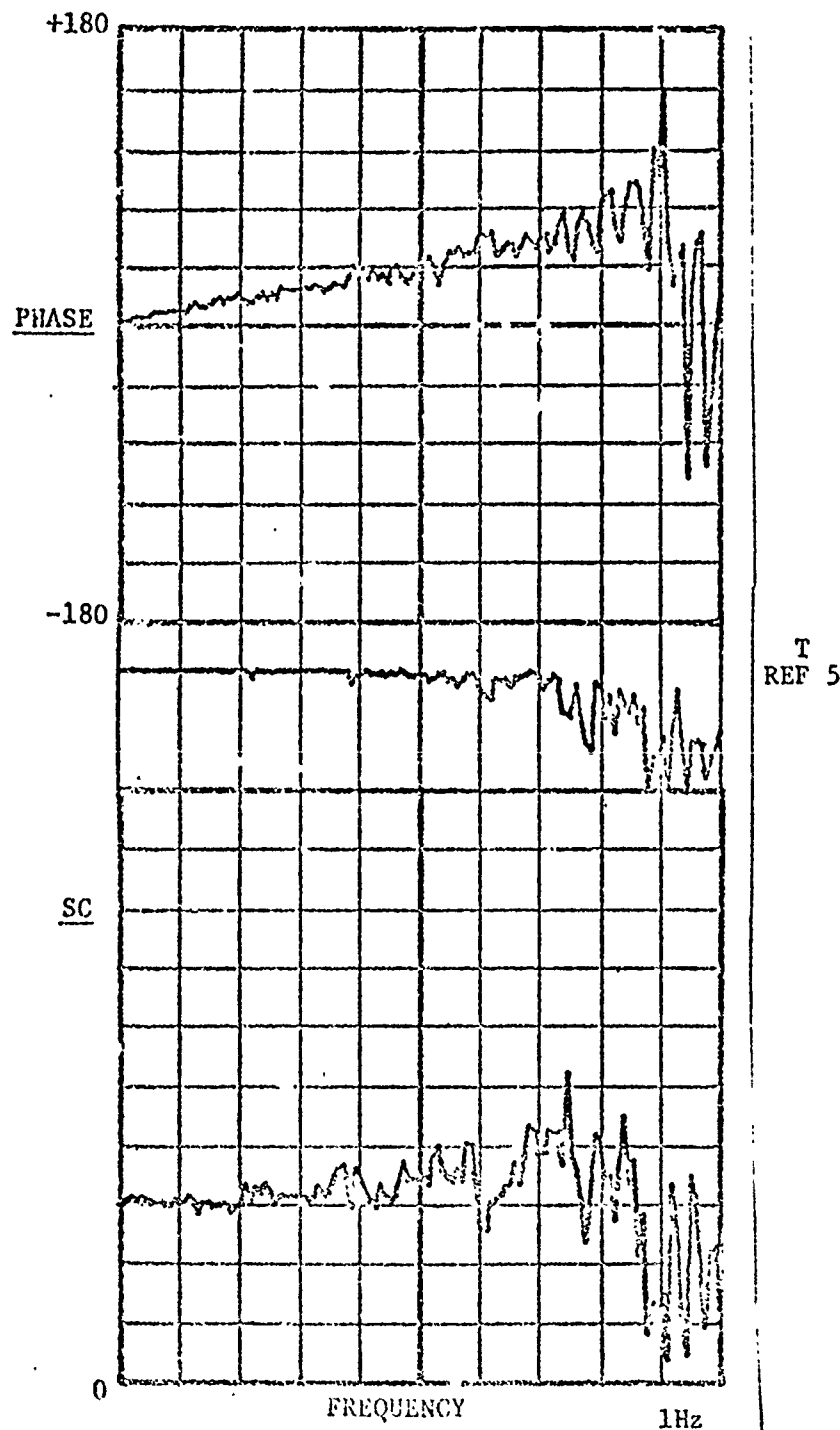


Figure 13-6. This remarkable linear phase lead was the result of the two-channel Biomation filter being used as a passive analog filter. The filter was the same IIR notch filter shown in Figures 13-4 and 13-5.

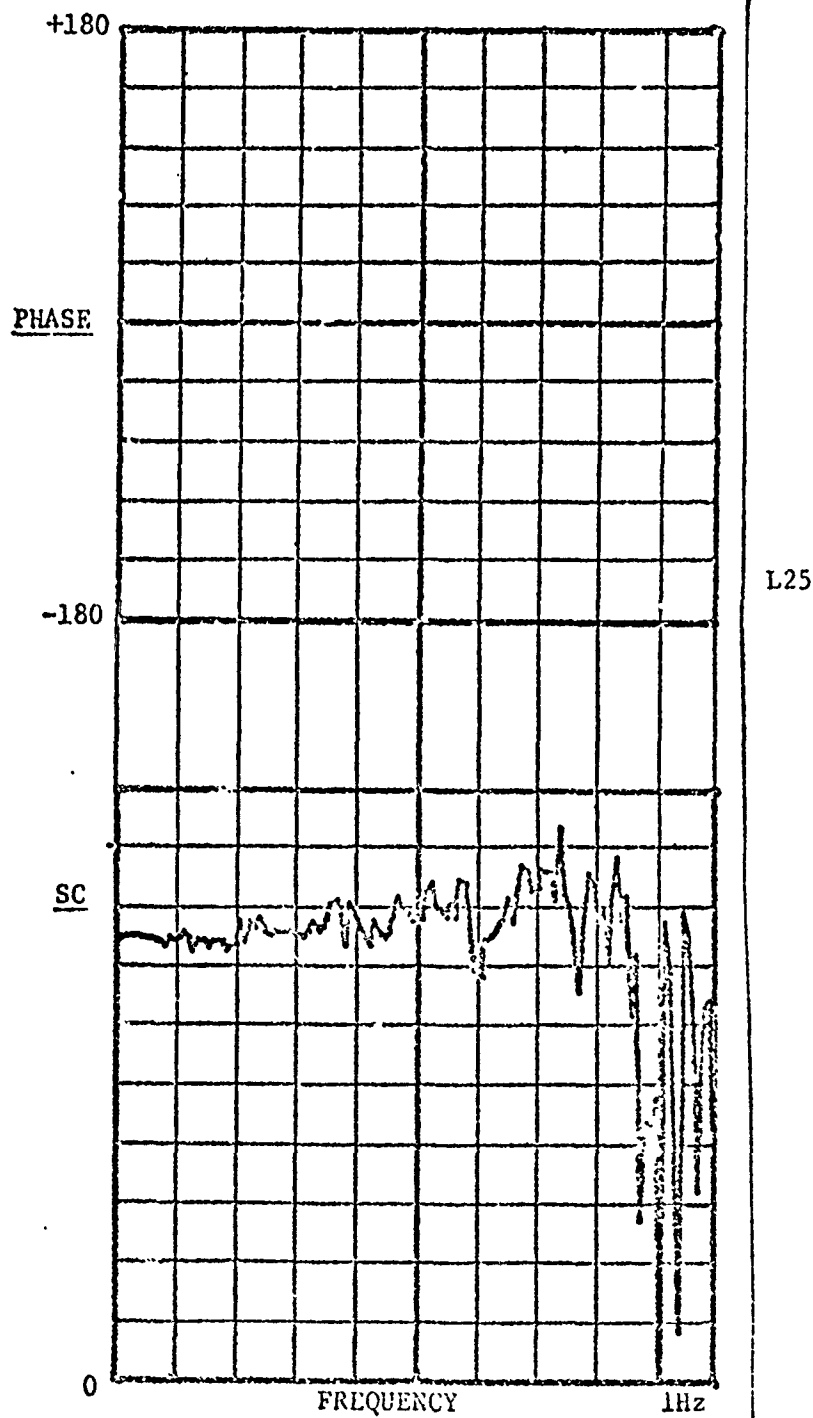


Figure 13-7. This is the amplitude data of Figure 13-6 shown in log form.

Optical slip rings are totally compatible with the modern laboratory trends toward digitizing the data as close to the source as possible and using automated mini-computer data acquisition systems.

The microprocessor based digital filter work for isopad seismic vibration control is summarized in this report.

SECTION 14

SUMMARY OF ISOPAD DIGITAL FILTER WORK FOR ISOPAD CONTROL

Using the four techniques as summarized below we were able to identify various types of digital filters which show significant promise for isopad control. These four techniques were:

1. Recursive digital filters analogous to common analog filters.
2. Variable time delay filters.
3. Finite Impulse Response Filters.
4. Infinite impulse response filters.

Of these four approaches the last was by far the most successful. These were the filters designed using the MAC/FIL digital filter design software package from Agbabian Associates. It is perhaps somewhat ironic because infinite impulse response filters have a reputation for having wildly fluctuating and quite unpredictable phase transfer functions. I say reputation because very little information is available either from the literature or from those familiar with digital filters. It was found, though, that these digital filters had nice ramps of phase lead which would be ideal for isopad control.

Reviewing the results of the other types of filters, the recursive digital filters yielded results which were very similar as expected to the analog filters of the same difference equations. Also as expected, they produced the predictable lag of the same magnitude as the corresponding analog filters. This was predicted and served to verify the digital filter execution programs, which was the intended result of these programs. These filters also yielded valuable information relative to the highest speed that we could expect a digital filter to be executed in real time using the pdp11/45 computer.

The "raw" digital filter execution program, which actually consists of the FORTRAN execution program configured to put out exactly the same value to the Datel 256 as it reads from the LPS 11 in the fastest manner possible with no arithmetic manipulations or filtering, the IDAC.MAC, the assembly language Datel 256 driver, would run at about a 113 Hz sample rate. With a very simple digital filter, such as a simple recursive digital filter which only simulates a simple one-stage passive RC filter, the program runs at about 100 Hz. This speed is really not adequate for many applications for the isopad controller.

In particular, since the frequency range of most interest for isopad control is the range from 1 Hz to 20 Hz, this makes a very difficult situation, especially when lead is required. Suppose at 20 Hz we have a minimum time delay between sample in and control signal out of 10 milliseconds. This is an almost insurmountable handicap when lead is desired. For this reason we were limited to test the filters usually at a maximum sample rate of 1 Hz. This was done with the idea that, while the filter execution program could not be made more efficient in the way the FORTRAN code was compiled to object code by the optimized compiler by writing an assembly language program, perhaps the sampling could be scheduled in such a way that required much less software overhead. The sampling mode using the LPS software to schedule the samples was chosen because it offered most accurate sample spacing. Sample spacing uniformity is absolutely critical in digital filtering because any "jitter" can cause a very spurious frequency response. But, the LPS software uses a large amount of processor time relative to the time needed to execute the filter itself. The slowed sample rates were advantageous in that tests could be run without interfering with the multi-user aspects of the pdp11/45 system, but they made the filters more or less useless for real-time applications.

The variable time delay filters were also found to be of little value. The predicted lead turned out to be variable magnitudes of lag because the delay through the filter at reasonable control sampling frequencies was greater than the variable delay effect of the filter. These filters proved to be useful in producing all sorts of nonlinear sorts of lag filters of almost any rolloff and phase plot shape required, but were of little use for the real-time applications.

The finite impulse response (FIR) filters proved to be very stable, and the FIR filter coefficient synthesis program was successfully run on the pdp 11/45, which was never accomplished for reasons to be discussed with the IIR filter coefficient synthesis program. Suitable lead in real time was never realized with this class of filters. The phase response was smooth and linear as expected, but in real time the result was always lag.

The most surprising result of the FIR filter test program is that, like the IIR filters, the FIR filters produced unexpected results when used in conjunction or series with analog filters. Even when an ordinary analog lowpass filter was used to smooth the staircase output to the Datel 256 digital to analog converter channel with a one-pole filter, the phase was made more positive

than without the filter and the phase roll-off was far more gradual. This was true even when a 200 Hz analog filter was used, which was far above the active range of the filter. Even though lead was never achieved, this is one of the several examples of the combination of analog and digital filters where an unexpected results was obtained from the digital time series analysis.

The results of the infinite impulse response filters were perhaps the most impressive.

The results of testing the infinite response (IIR) filters generated from the MAC/FIL software package can be summarized very easily. The phase responses obtained in the simulated runs were much better than could have been hoped for, and the phase responses obtained in the real-time tests with the pdpll/45 control loop and the Time/Data fourier analyzer were very disappointing.

Even filters like a lowpass Butterworth filter, which would be naturally stable under all conditions showed lead. The lead was not only positive with respect to the 0° line, but actually had a positive slope over nearly two decades of frequency. This of course gave us high hopes of building a real-time controller, but because of delays through the processor or some unknown effect, the phase was never realized in real-time.

Another very interesting IIR filter was the bandpass Chebychev. This filter would be useful in giving gain to a particular band remote from an isopad resonance frequency. It gives nearly linear phase lead over nearly two decades of frequency. Used in conjunction with analog filters, this filter gave much promise, but again, its benefits were never realized in real time.

One of the filters which was of considerable interest because of the work of Emil Broderon was the notch filter. Although the phase naturally exhibits a sharp transition near the notch frequency, the phase usually exhibits lead before the amplitude falls off. Then, by using a sharp lowpass filter in conjunction with the filter, the lead can be utilized in a servo in a very beneficial sense. This was true of the analog filter that Emil designed and built. From the MAC/FIL simulation it appeared that this might be the case for the digital filter as well. This was never realized in real time, except in one case in which the two channels of the Biomation anti-aliasing filter were used as passive filters in a special configuration described in the past chapter.

This result was unique and showed that it was possible to utilize analog and digital filters together to get more lead than you would expect from either of them, or even the sum of their leads, taken individually. In particular, even when a digital filter with lead was used with a low pass analog filter with lag, the lead of the digital filter could be enhanced if the cutoff frequency of the analog filter was sufficiently above the Nyquist folding frequency of the digital filter. While we never found an analytical explanation for this, it was found to be very repeatable using the Time/Data fourier analyzer.

One of the frightening things that worried us at the onset of this project was that experts warned us that in order to get lead from a digital filter, we would probably have to use an infinite impulse response filter and that the phase transfer function of this class of filters would probably have too many wild fluctuations to be of any value whatsoever in a closed loop servo application and could probably be used to advantage in a post-processor configuration. This proved to be only half true. The infinite impulse response filters did indeed prove to be the only class of filters we implemented that showed promising phase characteristics. However, it is clearly not true that their phase transfer function plots show any wild fluctuations that would make them unsuitable for closed-loop servo applications

SECTION 15

THE MOST SIGNIFICANT TECHNICAL CONTRIBUTIONS

The major accomplishments of this work period can be briefly summarized as follows:

1. The invention, design, development, construction of a prototype, and testing of an electronic fiber optics communication system for bipolar analog signals of data acquisition quality.
2. The invention, development, design, construction of a prototype assembly, testing, and evaluation of a fiber optic slip ring.
3. The application of linear phase design techniques and finite impulse response (FIR) discrete-time or digital filters to the isopad stabilization problem.

The analog fiber optics communication system has several unique features. Among these are very low parts count and compactness, and more importantly, a novel encoding scheme. The encoding scheme utilizes a combination of pulse-width-modulation (PWM) and voltage-to-frequency-to-voltage (V/F/V) conversion to double the information transmittable through a fiber optic communication of a specified pulse repetition rate. This is especially important in the case of laser diodes since the pulse repetition rate is limited in most cases to frequencies less than the maximum repetition rates of commercially available voltage to frequency converters in integrated circuit form.

The high resolution fiber optics communications system (FOCS) uses the above techniques to reconstruct a ± 10 volt analog output signal from an identical ± 10 volt analog input signal.

Data collected on a very general V/F/V system is presented to give the designer information on V/F/V system dynamics at frequencies much higher than those considered to be in the V/F/V's system's useful range for data acquisition and communication purposes. This data shows that the V/F/V process may be useful for servo loop applications at much higher frequencies than it is for data acquisition and data communication applications.

The data on this system has proved to be of much interest and we have had a number of requests for information from both military and civilian organiza-

tions on how to determine the effective data rates on voltage-to-frequency-to-voltage systems and how to compare the informational content of the pulses to those of a digital analog-to-digital conversion and transmission system. The loop dynamics of this system are obviously complex and nearly impossible to analyze using a purely mathematical approach. This is why our results have proven to be so valuable. We apparently have the only actual data since we are apparently the only group which has had access to a Fourier analyzer while testing a voltage-to-frequency-to-voltage system. By using the digital time series analysis techniques of this device (primarily correlation, coherence, and transfer function analysis) we have some unique data on the performance of this very popular class of devices.

Our work on optical slip rings has been very well received, partially because of the success of our prototype optical slipring assembly and particularly because of the well known shortcomings of mechanical slip rings or commutators as they are sometimes called. The mechanical slip rings commonly used to collect data from rotating test fixtures suffer from such problems as friction, backlash, continuity failures, wear-limited lifetime, noise, the need for precious metals, and expensive and time-consuming preventive maintenance. This report discusses a multi-channel fiber optic slip ring assembly which avoids these problems. The assembly is interfaced with digital systems and a custom designed analog-in, analog-out high resolution fiber optics data communication system to provide the control and data acquisition functions during automated tests of inertial guidance components. We also did research and reported on design tradeoffs between multiplexing and providing additional optical channels for multi-channel operation and explored these alternatives relative to currently available electronic, electro-optic, and optical components.

The research and development effort on the mechanical slip rings shows that optical slip rings are a practical alternative to mechanical slip rings and offer cost and performance advantages for signal transmission using standard components. Mechanical and electrical isolation are among the advantages to be gained. Mechanical and optical slip rings can complement one another if mechanical slip rings are used for power transmission and optical slip rings are used for signal transmission.

SECTION 16

SUMMARY OF THE PUBLICATIONS

Three publications have resulted from our research in collaboration with FJSRL personnel in the areas of analog fiber optics data communications and the fiber optic slip rings. Two of them have been published and the third is now pending publication. The two that have been published have generated enthusiastic responses from both civilian and military agencies.

The references to these publicaions are as follows:

1. Grimes, G. J. and Stevens, D. R. "A High Resolution Analog Fiber Optics Data Communications System," Proceedings of the SPIE, Vol. 95, Modern Utilization of Infrared Technology II, 1976.
2. Grimes, G. J., Monaco, S. J., and Stevens, D. R. "Fiber Optic Slip Rings for Totating Test Fixture Data Acquisition," Proceedings of the 23rd International Instrumentation Symposium, Instrumentation in the Aerospace Industry, Proceedings of the ISA (Instrument Society of America), Vol. 23, 1977.
3. Grimes, G. J. and Stevens, D. R. "A High Resolution Analog Fiber Optics Data Communications System," to be published (accepted for publication) in Optical Engineering, the technical journal of the Society of Photo-Optical Instrumentation Engineers (SPIE).

SECTION 17

SUMMARY OF THE PATENT PROCEEDINGS ON THE FIBER OPTIC SLIP RINGS

Kappa Systems, Inc., is currently working with a law firm to file a patent on the fiber optic slip ring developed under this contract.

The search has been completed and there are no similar devices patented. Also, the law firm has verified that the device is patentable. We are currently working with Mr. Gene W. Stockman and Mr. Scott F. Partridge at the following law firm:

Schuyler, Birch, Swindler, McKie and Beckett
One Thousand Connecticut Avenue
Washington, D.C. 20036
(202) 296-5500

APPENDIX A

SUMMARY OF SOFTWARE

Two major types of programs were used in this effort: the filter coefficient synthesis programs and the real-time filter execution programs. The filter coefficient synthesis programs are large FORTRAN programs which calculate filter coefficients for a filter given parameters such as the corner frequencies and the weights for each band. The digital filter execution programs are small FORTRAN programs which call an assembly language driver to handle the D/A output.

The filter coefficient synthesis programs used were of two types. These were:

1. A Finite Impulse Response (FIR) filter synthesis program.
2. An infinite impulse response (IIR) filter synthesis program.

The FIR synthesis program was taken from Theory and Applications of Digital Signal Processing by Lawrence R. Rabiner, Prentice Hall, 1975. This program was adjusted to run in an interactive mode in the 11/45. A listing of this program is shown in Appendix D.

The IIR program used was one from the MAC/FIL software package. This package was purchased in 1974 for DFEE (Dean of Faculty: Department of Electrical Engineering) with about \$5000 of FJSRL funds. This package consists of three programs:

1. MAC/FIL: generates coefficients for lowpass, highpass, bandpass, and band-reject filters of many kinds, including Butterworth, Chebychev types I and II, and elliptic.
2. MAC/APX: generates filter parameters from specified gain functions input to it. Thus, it can be employed to generate matched filters, Weiner-Hopf filters, or any type of filter defined by its amplitude which cannot be generated by MAC/FIL.
3. MAC/SIM: simulates digital filters implemented in fixed point arithmetic. Through its use many hardware design problems can be answered without having to build special purpose devices.

The IIR filters reported on here were generated through the use of MAC/FIL. This program required numerous modifications to run properly. Most of these modifications were performed by Airman Gary Lear of FJSRL and Capt. Perry Cole of the computer center.

The MAC/APX program was never successfully run. Extensive modifications will probably be required to run this program.

The MAC/SIM program was not needed since we could easily test the filter response with the time/data fast fourier analyzer.

The MAC/FIL program was used exclusively on the Burroughs 6700 at the Computer Center.

The MAC/FIL, MAC/APX, and MAC/SIM programs are not reproduced in the appendices since the copyright of the software might be violated by this procedure.

The digital filter execution programs are very similar except for the actual filtering part which calculates the output from the appropriate sum of the products. The filter execution programs are all self contained except for the I/O drivers and handlers. Although it would have been more convenient to make the filter execution program be the same in all cases and merely call a subroutine to execute a specific filter, the nonmodular approach was taken to improve filter performance. The subroutine call was found to just add software overhead and increase the time delay through the filter.

The filter execution program called INOUT, FTN simply puts the input value to the DAC of the Datel 256 system with the absolute minimum time delay of the filter. The filter acts as a simple passive lowpass filter for frequencies of a sizeable fraction of the sample frequency. The program GOOD.FTN is a minor modification of INOUT.FTN to run the digital equivalent of simple one-stage passive RC filter in an interactive mode.

The programs which execute the variable time delay filters are very similar. An example is shown in Appendix E.

The programs which execute the FIR filters include the examples BAD.FTN, BAND.FTN, NINE9.FTN, and DIFF32.FTN shown in Appendix F.

The IIR filter execution programs include the programs CASE8.FTN, CASE8F.FTN, HONEYF.FTN, and BUTTER.FTN shown in Appendix G.

The assembly language driven IDAC.MAC is shown in Appendix H.

A list of all the programs included in the following appendices follows:

1. INOUT.FTN Program to test response of system with no filter.
2. GOOD.FTN One-stage passive RC filter (interactive).
3. ASKFIR.FTN FIR coefficient synthesis program from Raginer and Gold. Modified to run in an interactive mode on pdp 11/45.
4. BAD.FTN Linear phase FIR program, $N = 9$.
5. NINE9.FTN Linear phase FIR filter execution program, $N = 9$.
6. DIFF32.FTN Linear phase FIR differentiator, $V = 32$. Symmetrical filter.
7. BAND.FTN Fast execution bandpass linear phase FIR filter execution program, $N = 32$, (for N even, coefficients).
8. DELAY13.FTN Variable time delay filter. Unity gains for all frequencies. Time delay is progressively less for higher frequencies. Interactive.
9. CASE8.FTN IIR filter execution program for 8 recursive and 9 non-recursive coefficients. $N = 8$. 0.015 Hz to 0.045 Hz bandpass. H's are nonsymmetrical; G's are symmetrical.
10. CASE8F.FTN IIR execution program for $N = 8$.
11. HONEYF.FTN IIR filter execution program $N = 5$, nonsymmetrical coefficients.
12. BUTTER.FTN IIR filter execution program with coefficients for lowpass Butterworth filter. $N = 5$. One nonrecursive and four recursive weights.
13. IDAC.MAC The assembly driver for the Datel 256 system.
14. NOTCH.FTN IIR execution program with 6 recursive and 7 nonrecursive weights.
15. DATEL.FTN Program written by Capt. Lind to exercise the Datel 256 system by writing a digital triangle wave to it.

APPENDIX B

Program Listing: INOUT.FTN

This program tests the system response with no filtering done.

```

0001      DIMENSION IBUF(6),IRATE(2),ISB(2)
0002      COMMON IRATE, IDATA,Y,FREQ
      C
0003      WRITE (5,80)
0004      80  FORMAT('$ ENTER LOW PASS CUTOFF FREQUENCY IN HERTZ  ')
0005      READ (5,81) FREQ
0006      81  FORMAT(F4.2)
      C
      C    READ IN SAMPLE PERIOD IN MILLISECONDS
0007      WRITE (5,30)
0008      30  FORMAT('$ENTER SAMPLE PERIOD IN MILLISECONDS')
0009      READ (5,31)IRATE(2)
0010      31  FORMAT (16)
      C    INITIALIZE FILTER FOR SUBROUTINE CUTOFF
0011      Y=0.0
      C
      C    INITIALIZE LABORATORY PERIPHERAL SYSTEM
0012      CALL ASLSLN (1,ISB)
      C
      C    DESIGNATE OUTPUT DAC CHANNEL FOR DTEL 256
0013      ICHAN=0
      C
      C    DESIGNATE REGISTER FOR LPS11 FLAG SET
0014      IEFN=7
      C
      C    SPECIFY TIME BETWEEN SAMPLES IN MILLISECONDS
      C    IRATE(2)=50
      C
      C    PUT LPS11 RTS ROUTINE IN MILLISECOND SAMPLING MODE
0015      IRATE(1)=2
      C
      C    INITIATE SYNCHRONOUS SAMPLING
0016      90  CALL RTS(IBUF,6.0,IRATE,IEFN,0.1,ISB)
      C
0017      INDX=5
0018      10  CALL WAITFR(IEFN)
0019      15  CALL CLREF(IEFN)
0020      SIGMA=IBUF(INDX)
      C
      C    SCALE IN VOLTS
0021      IDATA=10000.*(SIGMA/2048.)-10000.
      C
      C    CALL FILTER ROUTINE
0022      CALL CUTOFF
      C
      C    CALL LPS11 LED DISPLAY ROUTINE
0023      CALL LED(IDATA)
      C

```

PORTAN IV-PLUS V02-51
 INOUT.FTN /TR:BLOCKS/WR

11:13:01 28-APR-78

PAGE 2

```

      C      CALL ROUTINE TO OUTPUT RESULT TO DATEL 256
0024      C      CALL IDAC(IDATA,ICHAN)
      C
      C      ADJUST POINTERS FOR LPS11
      C      AND CLEAR HALF BUFFER FOR NEXT SAMPLE
0025      C      CALL ADJLPS(IBUF,1)
      C
0026      C      INDX=INDX+1
      C
      C      CHECK STATUS REGISTER FOR PROPER I/O
      C      AND TERMINATE IF STATUS NOT CORRECT
0027      C      IF (INDX.GT.6) INDX=5
0028      C      IF (ISB(2).GE.1) GO TO 15
0029      C      IF (ISB(1).NE.0) GO TO 95
0030      C      GO TO 10
0031      95    CONTINUE
      C
      C      IF PROGRAM CRASHES PRINT STATUS ON WAY OUT
0032      C      WRITE (5,200) ISB(1),ISB(2),IDATA,IBUF(INDX)
0033      200    FORMAT(4I12)
0034      END
  
```

PROGRAM SECTIONS

NAME	SIZE	ATTRIBUTES
SCODE1	000524 170	RW,I,CON,LCL
SPDATA	000020 8	RW,D,CON,LCL
SIIATA	000222 73	RW,D,CON,LCL
SVARS	0' 0032 13	RW,D,CON,LCL
.S555.	000016 7	RW,D,CON,GBL

TOTAL SPACE ALLOCATED = 001036 271

.INOUT=INOUT

APPENDIX C

Program Listing: GOOD.FTN

This is a simple digital filter which is the digital analogue of a simple one-stage passive RC filter which runs on the pdp 11/45 in an interactive mode.

```

0001      DIMENSION IBUF(6),IRATE(2),ISB(2)
      C
      C      THIS PROGRAM SYNTHESIZES AND EXECUTES
      C      A FIRST ORDER RECURSIVE FILTER CORRESPONDING
      C      TO A ONE-STAGE PASSIVE RC FILTER
      C
      C      THE ALGORITHM ON WHICH THIS FILTER IS BASED
      C      IS VALID ONLY WHEN THE TIME CONSTANT IS MUCH
      C      GREATER THAN THE PERIOD BETWEEN SAMPLES.
      C      THE TIME CONSTANT SHOULD BE AT LEAST THREE
      C      TIMES AS LARGE AS THE SAMPLE PERIOD
      C
0002      WRITE(5,60)
0003      60 FORMAT('$ ENTER LOW PASS CUTOFF FREQUENCY IN HERTZ-FLT.PT. ')
0004      READ (5,81) FREQ
0005      81 FORMAT(F4.2)
      C
      C      READ IN SAMPLE PERIOD IN MILLISECONDS
0006      WRITE (5,30)
0007      30 FORMAT('$ ENTER SAMPLE PERIOD IN MILLISECONDS-INTEGERS ')
0008      READ (5,31) IRATE(2)
0009      31 FORMAT (I6)
      C
      C      INITIALIZE FILTER
      C
      C
0010      Y=0.0
      C      INITIALIZE LABORATORY PERIPHERAL SYSTEM
0011      CALL ASLSLN (1,ISB)
      C
      C      DESIGNATE OUTPUT DAC CHANNEL FOR DTEL 256
0012      ICHAN=0
      C
      C      DESIGNATE REGISTER FOR LPS11 FLAG SET
0013      IEFN=7
      C
      C
      C      PUT LPS11 RTS ROUTINE IN MILLISECOND SAMPLING MODE
0014      IRATE(1)=2
      C
      C      CALCULATE RC TIME CONSTANT IN SECONDS
0015      RC=1./(FREQ*6.28)
      C
      C
      C      CALCULATE SAMPLE INTERVAL IN SECONDS
0016      ST=FLOAT(IRATE(2))/1000.
      C

```

GOOD.FTN

/TR:BLOCKS/WR

```
      C      CALCULATE FILTER PARAMETER
0017      TCON=ST/RC
0018      TCONC=1.-TCON
      C
      C
      C      TEST TO SEE IF FILTER WILL BE STABLE;
      C      IF UNSTABLE PROGRAM WILL TERMINATE AND PRINT 'UNSTABLE-
      C      SAMPLE RATE TOO LOW FOR TIME CONSTANT'
      C
0019      IF (TCON.GE.1) GO TO 301
0020      GO TO 555
0021      301 WRITE(5,201)
0022      201 FORMAT(' UNSTABLE-SAMPLE RATE TOO SLOW FOR TIME CONSTANT')
0023      GO TO 195
0024      555 CONTINUE
      C
      C
      C      INITIATE SYNCHRONOUS SAMPLING
0025      90 CALL PTS(IBUF,6,0,IRATE,IEFN,0,1,ISB)
      C
0026      INDX=5
0027      10 CALL WAITR(IEFN)
0028      15 CALL CLREF(IEFN)
0029      SIGNA=IBUF(INDX)
      C
      C      SCALE IN VOLTS
0030      IDATA=10000.*(SIGMA/2048.)-10000.
      C
      C      BEGIN FILTER ROUTINE
      C
0031      Y=TCONC*Y+TCON*IDATA
0032      IDATA=Y
      C      CALL LPS11 LED DISPLAY ROUTINE
0033      CALL LED(IDATA)
      C
      C      CALL ROUTINE TO OUTPUT RESULT TO DATEL 256
0034      CALL IDAC(ICHAN,IDATA)
      C
      C      ADJUST POINTERS FOR LPS11
      C      AND CLEAR HALF BUFFER FOR NEXT SAMPLE
0035      CALL ADJLPS(IBUF,1)
      C
0036      INDX=INDX+1
      C
      C      CHECK STATUS REGISTER FOR PROPER I/O
      C      AND TERMINATE IF STATUS NOT CORRECT
0037      IF (INDX.GT.6) INDX=5
0038      IF (ISB(2).GE.1) GO TO 10
```

TRAN IV-PLUS V02-51

11:10:38

28-APR-78

PAGE 3

DD.FTN

/TR:BLOCKS/WR

```

0039      IF (ISB(1).NE.0) GO TO 95
0040      GO TO 10
0041      95  CONTINUE
      C
      C      IF PROGRAM CRASHES PRINT STATUS ON WAY OUT
0042      WRITE (5,200) ISB(1),ISB(2),IDATA,ISB(INDX)
0043      200  FORMAT(4I12)
0044      195  CONTINUE
0045      END

```

PROGRAM SECTIONS

NAME	SIZE	ATTRIBUTES
CODE1	000676 223	RW,I,CON,LCL
SDATA	000024 10	RW,D,CON,LCL
SDATA	000330 108	RW,D,CON,LCL
SDRS	000070 28	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 001342 369

.DD=GOOD

APPENDIX D

Program Listing: ASKFIR.FTN

This is the FIR coefficient synthesis program from Rabiner and Gold as modified to run in an interactive manner on the pdp 11/45.

```

      COMB IN PI2,AD,DEV,X,Y,GRID,DES,WT,ALPHA,TEXT,NFCNS,NGRID

      DIMENSION IEXT(66),AD(66),X(66),Y(66),ALPHA(66)
      DIMENSION H(66)
      DIMENSION DES(1045),GRID(1045),WT(1045)
      DIMENSION EDGE(20),FX(10),WTX(10),DEVIAT(10)
      DOUBLE PRECISION PI2,PI
      DOUBLE PRECISION AD,DEV,X,Y
      PI2=6.283185307179586
      PI=3.141592653489793
C     THIS PROGRAM IS SET UP FOR A MAXIMUM LENGTH OF 128, BUT
C     THIS UPPER LIMIT CAN BE CHANGED BY REDIMENSIONING THE
C     ARRAYS IEXT, AD,ALPHA, X,Y, H TO BE NFMAY/2+2.
C     THE ARRAYS DES, GRID, AND WT MUST DIMENSIONED
C     16 (NFMAY/2+2)
C
      NFMAY=128
100  CONTINUE
      JTYPE=0
C
      PROGRAM INPUT SECTION
      WRITE(5,4393)
4398  FORMAT(' ENTER FILTER LENGTH;TYPE OF FILTER:1=MULTIPLE')
      WRITE(5,4397)
4397  FORMAT(' PASSBAND/STOPBAND,2=DIFFERENTIATOR,3=HILBERT')
      WRITE(5,4396)
4396  FORMAT(' TRANSFORM FILTER;NUMBER OF BANDS, AND GRID DENSITY')
C
      READ (5,4444) NFILT,JTYPE,NBANDS,LGRID
4444  FORMAT(4I4)
C     IF(NFILT.GT.NFMAY.OR.NFILT.LT.3) CALL ERROR
      IF(NBANDS.LE.0) NBANDS=1
C
      GRID DENSITY IS ASSUMED TO BE 16 UNLESS SPECIFIED
      OTHERWISE
C
      IF(LGRID.LE.0) LGRID=16
      JB=2*NBANDS
      WRITE(5,4388)
4388  FORMAT(' ENTER BANDEDGES(FLOATING POINT) ')
      READ (5,3333) (EDGE(J),J=1,JB)
      WRITE(5,4387)
4387  FORMAT(' ENTER DESIRED FUNCTION FOR EACH BAND ')
      READ (5,3333) (FX(J),J=1,NBANDS)
      WRITE(5,4386)
4386  FORMAT(' ENTER WEIGHT FUNCTION IN EACH BAND ')
      READ (5,3333) (WTX(J),J=1,NBANDS)
3333  FORMAT(20F10.2)
      WRITE(5,4444) NFILT,JTYPE,NBANDS,LGRID
      WRITE(5,3333) (EDGE(J),J=1,JB)
      WRITE(5,3333) (FX(J),J=1,NBANDS)
      WRITE(5,3333) (WTX(J),J=1,NBANDS)
      IF(JTYPE.EQ.0) CALL ERROR

```

Program ASKFIR

```

      NEG=1
      IF (JTYPE.EQ.1) NEG=0
      NODD=NFILT/2
      NODD=NFILT-2*NODD
      NFCNS=NFILT/2
      IF (NODD.EQ.1.AND.NEG.EQ.0) NFCNS=NFCNS+1
C
C      SET UP THE DENSE GRID.  THE NUMBER OF POINTS IN THE GRID
C      IS (FILTER LENGTH + 1)*GRID DENSITY/2
C
      GRID(1)=EDGE(1)
      DELF=LGRID*NFCNS
      DELF=0.5/DELF
      IF (NEG.EQ.0) GO TO 135
      IF (EDGE(1).LT.DEF) GRID(1)=DELF
135  CONTINUE
      J=1
      L=1
      LBAND=1
140  FUP=EDGE(L+1)
145  TEMP=GRID(J)
C
C      CALCULATE THE DESIRED MAGNITUDE RESPONSE AND THE WEIGHT
C      FUNCTION OF THE GRID
C
      DES(J)=EFF(TEMP,FX,WTX,LBAND,JTYPE)
      WT(J)=WATE(TEMP,FX,WTX,LBAND,JTYPE)
      J=J+1
      GRID(J)=TEMP+DELF
      IF (GRID(J).GT.FUP) GO TO 150
      GO TO 145
150  GRID(J-1)=FUP
      DES(J-1)=EFF(FUP,FX,WTX,LBAND,JTYPE)
      WT(J-1)=WATE(FUP,FX,WTX,LBAND,JTYPE)
      LBAND=LBAND+1
      L=L+2
      WRITE(5,1000)
      WRITE(5,2765)
2765  FORMAT(' HERE WE GO AGAIN')
      IF (LBAND.GT.NBANDS) GO TO 160
      GRID(J)=EDGE(L)
      GO TO 140
160  NGRID=J-1
      IF (NEG.NE.NODD) GO TO 165
      IF (GRID(NGRID).GT.(0.5-DELF)) NGRID=NGRID-1
165  CONTINUE
C
C      SET UP A NEW APPROXIMATION PROBLEM WHICH IS EQUIVALENT
C      TO THE ORIGINAL PROBLEM
C
      IF (NEG) 170,170,180
170  IF (NODD.EQ.1) GO TO 200
      DO 175 J=1,NGRID
      CHANGE=DCOS(PI*GRID(1))

```

Program ASKFIR

```

      DES(J)=DES(J)/CHANGE
175  UT(J)=UT(J)*CHANGE
      GO TO 200
180  IF (NODD.EQ.1) GO TO 190
      DO 185 J=1,NGRID
      CHANGE=DSIN(PI*GRID(J))
      DES(J)=DES(J)/CHANGE
      GO TO 200
185  UT(J)=UT(J)*CHANGE
190  DO 195 J=1,NGRID
      CHANGE=DSIN(PI2*GRID(J))
      DES(J)=DES(J)/CHANGE
C
195  WT(J)=WT(J)*CHANGE
C  INITIAL GUESS FOR THE EXTREMAL FREQUENCIES--EQUALLY
C  SPACED ALONG THE GRID
C
200  TEMP=FLOAT(NGRID-1)/FLOAT(NFCNS)
      DO 210 J=1,NFCNS
210  IEXT(J)=(J-1)*TEMP+1
      IEXT(NFCNS+1)=NGRID
      NM1=NFCNS-1
      NZ=NFCNS+1
C
C  CALL THE REMEZ EXCHANGE ALGORITHM TO DO THE APPROXIMATION
C  PROBLEM
C
      CALL PENEZ(EDGE,NBANDS)
1000  FORMAT(' HERE WE GO')
      WRITE(5,1000)
C
C  CALCULATE THE IMPULSE RESPONSE
C
      IF (NEG) 300,300,320
300  IF (NODD.EQ.0) GO TO 310
      DO 305 J=1,NM1
305  H(J)=0.5*ALPHA(NZ-J)
      H(NFCNS)=ALPHA(1)
      GO TO 350
310  H(1)=0.25*ALPHA(NFCNS)
      DO 315 J=2,NM1
315  H(J)=0.25*(ALPHA(NZ-J)+ALPHA(NFCNS+2-J))
      H(NFCNS)=0.5*ALPHA(1)+0.25*ALPHA(2)
      GO TO 350
320  IF (NODD.EQ.0) GO TO 330
      H(1)=0.25*ALPHA(NFCNS)
      H(2)=0.25*ALPHA(NM1)
      DO 325 J=3,NM1
325  H(J)=0.25*(ALPHA(NZ-J)-ALPHA(NFCNS+3-J))
      H(NFCNS)=0.5*ALPHA(1)-0.25*ALPHA(3)
      H(NZ)=0.0
      GO TO 350
330  H(1)=0.25*ALPHA(NFCNS)

```

Program ASKFIR

```

DO 335 J=2,NFIL
335 H(J)=2.25*(ALPHA(NZ-J)-ALPHA(NFCNS+2-J))
H(1)=0.5*ALPHA(1)-0.25*ALPHA(2)
NFI=5.12000)
C
C PROGRAM OUTPUT SECTION
C
350 WRITE (6,360)
360 FORMAT(1H1, 70(1H*)//25X, ' FINITE IMPULSE RESPONSE (FIR)'/
125X, ' LINEAR PHASE DIGITAL FILTER DESIGN'/
225X, ' REMEZ EXCHANGE ALGORITHM'//)
IF(JTYPE.EQ.1) WRITE (6,365)
365 FORMAT(25X, ' BANDPASS FILTER'//)
IF(JTYPE.EQ.2) WRITE (6,370)
370 FORMAT(25X, ' DIFFERENTIATOR'//)
IF(JTYPE.EQ.3) WRITE (6,375)
375 FORMAT(25X, ' HILBERT TRANSFORMER'//)
WRITE (6,378) NFILT
378 FORMAT(15X, ' FILTER LENGTH = ',I3//)
WRITE (6,380)
380 FORMAT(15X, ' ***** IMPULSE RESPONSE *****')
DO 381 J=1,NFCNS
K=NFILT+1-J
IF(NEG.EQ.0) WRITE (6,382) J,H(J),K
IF(NEG.EQ.1) WRITE (6,383) J,H(J),K
381 CONTINUE
382 FORMAT(20X, ' H(',I3,') = ',E15.8, ' = H(',I4,')')
383 FORMAT(20X, ' H(',I3,') = ',E15.8, ' = -H(',I4,')')
IF(NEG.EQ.1.AND.NODD.EQ.1) WRITE (6,384) NZ
384 FORMAT(20X, ' H(',I3,') = 0.0')
DO 430 K=1,NBANDS,4
KUP=K+3
IF(KUP.GT.NBANDS) KUP=NBANDS
WRITE (6,385) (J,J=K,KUP)
385 FORMAT(/24X,4(' BAND',I3,8X))
WRITE (6,390) (EDGE(2+J-1),J=K,KUP)
390 FORMAT(2X, ' LOWER BAND EDGE',5F15.9)
WRITE (6,395) (EDGE(2+J),J=K,KUP)
395 FORMAT(2X, ' UPPER BAND EDGE',5F15.9)
IF(JTYPE.NE.2) WRITE (6,400) (FX(J),J=K,KUP)
400 FORMAT(2X, ' DESIRED VALUE',2X,5F15.9)
IF(JTYPE.EQ.2) WRITE (6,405) (FX(J),J=K,KUP)
405 FORMAT(2X, ' DESIRED SLOPE',2X,5F15.9)
WRITE (6,410) (WTX(J),J=K,KUP)
410 FORMAT(2X, ' WEIGHTING',6X,5F15.9)
DO 420 J=K,KUP
420 DEVIAT(J)=DEV/WTX(J)
WRITE (6,425) (DEVIAT(J),J=K,KUP)
425 FORMAT(2X, ' DEVIATION',6X,5F15.9)
IF(JTYPE.NE.1) GO TO 450
DO 430 J=K,KUP
430 DEVIAT(J)=20.0*LOG10(DEVIAT(J))
WRITE (6,435) (DEVIAT(J),J=K,KUP)
435 FORMAT(2X, ' DEVIATION IN DB',5F15.9)

```

```

450 CONTINUE
  WRITE (6,455) (GRID(IEXT(J)),J=1,NZ)
455 FORMAT(2X,' EXTREMAL FREQUENCIES'/(2X,5F12.7))
  WRITE (6,460)
460 FORMAT(1X,70(IH1)/1H1)
C   IF (NFILT.NE.0) GO TO 100
  STOP
  END

C
  FUNCTION EFF(TEMP,FX,WTX,LBAND,JTYPE)
C
C   FUNCTION TO CALCULATE THE DESIRED MAGNITUDE RESPONSE
C   AS A FUNCTION OF FREQUENCY.
C
  DIMENSION FX(5),WTX(5)
  IF(JTYPE.EQ.2) GO TO 1
  EFF=FX(LBAND)
  RETURN
1 EFF=FX(LBAND)*TEMP
  RETURN
  END

C
C
C
  FUNCTION WATE(TEMP,FX,WTX,LBAND,JTYPE)
C
C   FUNCTION TO CALCULATE THE WEIGHT FUNCTION AS A
C   FUNCTION OF FREQUENCY.
C
  DIMENSION FX(5),WTX(5)
  IF(JTYPE.EQ.2) GO TO 1
  WATE=WTX(LBAND)
  RETURN
1 IF(FX(LBAND).LT.0.0001) GO TO 2
  WATE=WTX(LBAND)/TEMP
  RETURN
2 WATE=WTX(LBAND)
  RETURN
  END

C
C
C
  SUBROUTINE ERROR
  WRITE (6,1)
1 FORMAT(' ***** ERROR IN INPUT DATA *****')
  STOP
  END

C
C
C
  SUBROUTINE REINZ(EDGE,NBANDS)
C
C
COMMON P12,AD,DEV,X,Y,GRID,DES,WT,ALPHA,IEXT,NFCNS,NGRID

```

Program ASKFIR

```

DIMENSION EDGE(20)
DIMENSION IEXT(66),AD(66),ALPHA(66),X(66),Y(66)
DIMENSION DES(1045),GRID(1045),WT(1045)
DIMENSION A(66),P(66),Q(66)
DOUBLE PRECISION P12,DNUM,DDEN,DTEMP,A,P,Q
DOUBLE PRECISION AD,DEV,X,Y

C
C
      ITRMAX=200
      DEVL=-1.0
      NZ=NFCNS+1
      WRITE(5,222)
222  FOR MT(' STARTING REMEZ')
      NZZ=NFCNS+2
      NITER=0
100  CONTINUE
      IEXT(NZZ)=NGRID+1
      NITER=NITER+1
      IF(NITER.GT.ITRMAX) GO TO 400
      DO 110 J=1,NZ
      DTEMP=GRID(IEXT(J))
      DTEMP=DCOS(DTEMP*P12)
110  X(J)=DTEMP
      JET=(NFCNS-1)/15+1
      DO 120 J=1,NZ
120  AD(J)=D(J,NZ,JET)
      DNUM=0.0
      DDEN=0.0
      K=1
      DO 130 J=1,NZ
      L=IEXT(J)
      DTEMP=AD(J)*DES(L)
      DNUM=DNUM+DTEMP
      DTEMP=K*AD(J)/WT(L)
      DDEN=DDEN+DTEMP
130  K=-K
      DEV=DNUM/DDEN
      NU=1
      IF(DEV.GT.0.0) NU=-1
      DEV=-NU*DEV
      K=NU
      DO 140 J=1,NZ
      L=IEXT(J)
      DTEMP=K*DEV/WT(L)
      Y(J)=DES(L)+DTEMP
140  K=-K
      WRITE(5,799)
799  FORMAT(' REMEZ CHECKPOINT 2')
      IF(DEV.GE.DEVL) GO TO 150
      CALL DUCH
      GO TO 400
150  DEVL=DEV
      JCHNGE=0
      K1=IEXT(1)

```

Program ASKFIR

```

      NZ=IEXT(NZ)
      KLOW=0
      NUT=-NUT
      J=1
C
C      SEARCH FOR THE EXTREMAL FREQUENCIES OF THE BEST
C      APPROXIMATION
C
200  IF(J.EQ.NZ) YNZ=COMP
      IF(J.GE.NZ) GO TO 300
      KUP=IEXT(J+1)
      L=IEXT(J)+1
      NUT=-NUT
      IF(J.EQ.2) Y1=COMP
      COMP=DEV
      IF(L.GE.KUP) GO TO 220
      WRITE(5,596)
596  FORMAT(' WE GOT TO HERE')
      ERR=GEE(L,NZ)
      EPR=(ERR-DES(L))*WT(L)
      DTEMP=NUT*ERR-COMP
      WRITE(5,888)ERR,DES(L),WT(L),COMP,DTEMP,NUT,GEE(L,NZ)
888  FORMAT(5F10.4,18,F10.4)
      IF(DTEMP.LE.0.0) GO TO 220
      COMP=NUT*ERR
210  L=L+1
      IF(L.GE.KUP) GO TO 215
      ERR=GEE(L,NZ)
      EPR=(ERR-DES(L))*WT(L)
      DTEMP=NUT*EPR-COMP
      IF(DTEMP.LE.0.0) GO TO 215
      COMP=NUT*ERR
      GO TO 210
215  IEXT(J)=L-1
      J=J+1
      KLOW=L-1
      JCHNGE=JCHNGE+1
      GO TO 200
220  L=L-1
      WRITE(5,789)
789  FORMAT(' REVEZ CHECKPOINT 3')
225  L=L-1
      IF(L.LE.KLOW) GO TO 250
      WRITE(5,753)
753  FORMAT(' JUMP TO GEE')
      ERR=GEE(L,NZ)
      WRITE(5,6432)
6432 FORMAT(' COME BACK FROM GEE')
      EPR=(ERR-DES(L))*WT(L)
      DTEMP=NUT*ERR-COMP
      IF(DTEMP.GT.0.0) GO TO 230
      IF(JCHNGE.LE.0) GO TO 225
230  COMP=NUT*ERR
      WRITE(5,975)

```

Program ASXFIR

```

979 FORMAT(' REVEZ CHECKPOINT 3.2')
235 L=L-1
    IF(L.LE.KLOW) GO TO 240
    ERR=GEE(L,NZ)
    ERR=(ERR-DES(L))*WT(L)
    DTEMP=NUT*ERR-COMP
    IF(DTEMP.LE.0.0) GO TO 240
    COMP=NUT*ERR
    GO TO 235
240 KLOW=IEXT(J)
    IEXT(J)=L+1
    J=J+1
    JCHNGE=JCHNGE+1
    GO TO 200
250 L=IEXT(J)+1
    IF(JCHNGE.GT.0) GO TO 215
255 L=L+1
    IF(L.GE.KUP) GO TO 260
    ERR=GEE(L,NZ)
    ERR=(ERR-DES(L))*WT(L)
    DTEMP=NUT*ERR-COMP
    IF(DTEMP.LE.0.0) GO TO 225
    COMP=NUT*ERR
    GO TO 210
260 KLOW=IEXT(J)
    J=J+1
    GO TO 200
300 IF(J.GT.NZZ) GO TO 320
    IF(K1.GT.IEXT(1)) K1=IEXT(1)
    IF(KHZ.LT.IEXT(NZ)) KHZ=IEXT(NZ)
    NUT1=NUT
    WRITE(5,432)
432 FORMAT(' REVEZ CHECKPOINT 3.5')
    NUT=-NU
    L=0
    KUP=K1
    COMP=Y1Z*(1.00001)
    LUCK=1
310 L=L+1
    IF(L.GE.KUP) GO TO 315
    ERR=GEE(L,NZ)
    ERR=(ERR-DES(L))*WT(L)
    DTEMP=NUT*ERR-COMP
    IF(DTEMP.LE.0.0) GO TO 310
    COMP=NUT*ERR
    J=NZZ
    GO TO 210
315 LUCK=6
    GO TO 325
320 IF(LUCK.GT.9) GO TO 350
    IF(COMP.GT.Y1) Y1=COMP
    K1=IEXT(NZZ)
325 L=NGPID+1
    KLOW=KHZ

```

Program ASKFIR

```

      NUT=-NUT1
      COM1=Y1*(1.00001)
330  L=L-1
      IF(L.LE.KLOW) GO TO 340
      ERR=GEE(L,NZ)
      EFR=(ERR-DES(L))*WT(L)
      DTEMP=NUT*ERR-COM1
      IF(DTEMP.LE.0.0) GO TO 350
      J=NZZ
      COM1=NUT*ERR
      LUCK=LUCK+10
      GO TO 235
340  IF(LUCK.EQ.6) GO TO 370
      DO 345 J=1,NFCNS
345  IEXT(NZZ-J)=IEXT(NZ-J)
      IEXT(1)=K1
      GO TO 100
350  KN=IEXT(NZZ)
      DO 350 J=1,NFCNS
350  IEXT(J)=IEXT(J+1)
      IEXT(NZ)=KN
      WRITE(5,987)
987  FORMAT(' REMEZ CHECKPOINT 4')
      GO TO 100
370  IF(JCHANGE.GT.0) GO TO 100
C
C   CALCULATION OF THE COEFFICIENTS OF THE BEST
C   APPROXIMATION USING THE INVERSE DISCRETE
C   FOURIER TRANSFORM
C
400  CONTINUE
      NM1=NFCNS-1
      FSH=1.0E-06
      GTEMP=GRID(1)
      X(NZZ)=-2.0
      CN=2/NFCNS-1
      DELF=1.0/CN
      L=1
      KKK=0
      IF(EDGE(1).EQ.0.0.AND.EDGE(2+NBRANDS).EQ.0.5) KKK=1
      IF(NFCNS.LE.3) KKK=1
      IF(KKK.EQ.1) GO TO 405
      DTEMP=DCOS(PI2*GRID(1))
      DNUM=DCOS(PI2*GRID(NGRID))
      AA=2.0/(DTEMP-DNUM)
      BB=-(DTEMP+DNUM)/(DTEMP-DNUM)
405  CONTINUE
      DO 430 J=1,NFCNS
      FT=(FLOAT(J-1))*DELF
      XT=DCOS(PI2*FT)
      IF(KKK.EQ.1) GO TO 410
      XT=(XT-BB)/AA
C   ARCOS=ATAN(XT/SQRT(1.0-XT*XT))
      FT=ACOS(XT)/PI2

```

Program ASKFIR

```

C      FT=.5
410  XE=Y(L)
      IF(Y(L).GT.XE) GO TO 420
      IF((XE-XT).LT.FSH) GO TO 415
      L=L+1
      GO TO 410
415  A(J)=Y(L)
      GO TO 425
420  IF((XT-XE).LT.FSH) GO TO 415
      GRID(1)=FT
      A(J)=GFE(1,NZ)
425  CONTINUE
      IF(L.GT.1) L=L-1
430  CONTINUE
      GRID(1)=GTEMP
      DDEN=PI2/CN
      DO 510 J=1,NFCNS
      DTEMP=0.0
      DNUM=(FLOAT(J-1))*DDEN
      IF(NM1.LT.1) GO TO 505
      DO 500 K=1,NM1
500  DTEMP=DTEMP+A(K+1)*DCOS(DNUM*K)
505  DTEMP=2.0*DTEMP+A(1)
510  ALPHA(J)=DTEMP
      DO 550 J=2,NFCNS
550  ALPHA(J)=2*ALPHA(J)/CN
      A'(1)=ALPHA(1)/CN
      IF(KKK.EQ.1) GO TO 545
      P(1)=2.0*ALPHA(NFCNS)+BB+ALPHA(NM1)
      P(2)=2.0*AA*ALPHA(NFCNS)
      Q(1)=ALPHA(NFCNS-2)-ALPHA(NFCNS)
      DO 540 J=2,NM1
      IF(J.LT.NM1) GO TO 515
      AA=0.5*AA
      BB=0.5*BB
515  CONTINUE
      P(J+1)=0.0
      DO 520 K=1,J
      A(K)=P(K)
520  P(K)=2.0*BB+A(K)
      P(2)=P(2)+A(1)*2.0*AA
      JM1=J-1
      DO 525 K=1,JM1
525  P(K)=P(K)+Q(K)+AA*A(K+1)
      JP1=K+1
      DO 530 K=3,JP1
530  P(K)=P(K)+AA*A(K-1)
      IF(J.EQ.NM1) GO TO 540
      DO 535 K=1,J
535  Q(K)=-A(K)
      Q(1)=Q(1)+ALPHA(NFCNS-1-J)
540  CONTINUE
      DO 543 J=1,NFCNS
543  ALPHA(J)=P(J)

```

```

545 CONTINUE
    IF(NFCNS.GT.3) RETURN
    WRITE(5,765)
765 FORMAT(' REEZ CHECKPOINT 5')
    ALPHA(NFCNS+1)=0.0
    ALPHA(NFCNS+2)=0.0
    WRITE(5,556)
556 FORMAT(' LEAVING REEZ')
    RETURN
END

C
C
C      DOUBLE PRECISION FUNCTION D(K,N,M)
C
C      FUNCTION TO CALCULATE THE LAGRANGE INTERPOLATION COEFFICIENTS
C      FOR USE IN THE FUNCTION GEE.
C
COMMON P12,AD,DEV,X,Y,GRID,DES,WT,ALPHA,IEXT,NFCNS,NGRID
DIMENSION IEXT(66),AD(66),ALPHA(66),X(66),Y(66)
DIMENSION DES(1045),GRID(1045),WT(1045)
DOUBLE PRECISION AD,DEV,X,Y
DOUBLE PRECISION Q
DOUBLE PRECISION P12
D=1.0
Q=X(K)
DO 3 L=1,M
DO 2 J=L,M-1
IF(Q-X(J))1,2,1
1 D=2.0*(Q-X(J))
2 CONTINUE
3 CONTINUE
D=1.0/D
RETURN
END

C
C
C      DOUBLE PRECISION FUNCTION GEE(K,N)
C
C      FUNCTION TO EVALUATE THE FREQUENCY RESPONSE USING THE
C      LAGRANGE INTERPOLATION FORMULA IN THE BARYCENTRIC FORM
C
COMMON P12,AD,DEV,X,Y,GRID,DES,WT,ALPHA,IEXT,NFCNS,NGRID
DIMENSION IEXT(66),AD(66),ALPHA(66),X(66),Y(66)
DIMENSION DES(1045),GRID(1045),WT(1045)
DOUBLE PRECISION P,C,D,XF
DOUBLE PRECISION P12
DOUBLE PRECISION AD,DEV,X,Y
P=0.0
XF=GRID(K)
XF=DCOS(P12*XF)
D=0.0
DO 12 J=1,M
C=XF-X(J)

```

```

C      WRITE (5,6666) %F,X(J)
      IF (C) 4,12,4
      4 CONTINUE
C6666 FORMAT(2F10.5)
      C=AD(J)/C
      D=D+C
      1 P=P+C*Y(J)
      12 CONTINUE
      IF(D) 33,66,33
      33 CONTINUE
      GEE=P/D
      GO TO 77
      66 CONTINUE
      GEE=.5
      77 CONTINUE
      RETURN
      END

C
C
      SUBROUTINE OUCH
      WRITE (6,111)
      111 FORMAT(' *****FAILURE TO CONVERGE*****')
C      1' PROBABLE CAUSE IS MACHINE ROUNDING ERROR',
C      2' THE IMPULSE RESPONSE MAY BE CORRECT',
C      3' CHECK WITH A FREQUENCY RESPONSE')
      RETURN
      END

```

Program ASKFIR

APPENDIX E

Program Listing: DELAY13.FTN

This is a sample of a variable time delay filter. It has unity gain for all frequencies (up to frequencies near the sample frequency) and has a time delay which is progressively less for higher frequencies. It runs in an interactive mode on the pdp 11/45.

DELAY13.FTN /TR:BLOCKS/LR

```

0001      DIMENSION IBUF(5),IRATE(2),ISB(2)
      C
      C      THIS PROGRAM SYNTHESIZES AND EXECUTES
      C      A DIGITAL FILTER WHICH HAS UNITY GAIN AT
      C      ALL FREQUENCIES BUT THE TIME DELAY IS
      C      LESS FOR HIGHER FREQUENCIES. THE IDEA IS
      C      TO PROVIDE SOME LEAD FOR THE ISOPAD CONTROL.
      C
      C
      C
0002      WRITE(5,30)
0003      80 FORMAT('S ENTER TIME DELAY MULTIPLIER-FLT.PT. ')
0004      READ (5,81) FREQ
0005      81 FORMAT(F19.7)
0006      IFREQ=FREQ
      C
      C      READ IN SAMPLE PERIOD IN MILLISECONDS
0007      WRITE (5,30)
0008      30 FORMAT('S ENTER SAMPLE PERIOD IN MILLISECONDS-INTEG. ')
0009      READ (5,31) IRATE(2)
0010      31 FORMAT (I6)
      C
      C      INITIALIZE FILTER
      C
      C
0011      Y=0.0
      C      INITIALIZE LABORATORY PERIPHERAL SYSTEM
0012      CALL ASLOCN (1,150)
      C
      C      DESIGNATE INPUT DAC CHANNEL FOR DATA 256
0013      ICHP=
      C
      C      DESIGNATE REGISTER FOR LPS11 FLAG SET
0014      IEPN=7
      C
      C
      C      PUT LPS11 RTS ROUTINE IN MILLISEC. SAMPLING MODE
0015      IRATE(1)=2
      C
      C
      C      INITIATE SYNCHRONOUS SAMPLING
0016      90 CALL RTS(1200,6.0,IRATE,IEPN,C.1,150)
      C
      C
0017      INDM=5
0018      10 CALL UNITER(IEPN)
0019      15 CALL CLREF(IEPN)
0020      SIGMA=IBUF(INDM)
      C

```

```

C      SCALE IN VOLTS
0021      IDATA=12300.*(SIGMA/2048.)-10000.
C
C      BEGIN FILTER ROUTINE
0022      IDIFF=IABS(IDATA-IDATA)
0023      IDIFF=IABS(IDIFF)
0024      DIFF=IDIFF
C
0025      DLEP=200.-IDIFF
0026      DLEP=0.
0027      DLEP=50.-IDIFF
0028      IDATA2=IDATA
0029      IDLAY=IDIFF*IDLAY
0030      IF(IDLAY.LT.2) GO TO 9977
0031      DO 9977 I=1, IDLAY
0032      BLAP=1.
0033      WASTE=97*(BLAP)
C      WASTE=97*(BLAP)
0034      9977 CONTINUE
C      CALL LPS11 LED DISPLAY ROUTINE
0035      CALL LED(IDATA)
C
C      CALL ROUTINE TO OUTPUT RESULT TO DATEL 256
0036      CALL IDAC(ICHAN, IDATA)
C
C      ADJUST POINTER FOR LPS11
C      AND CLEAR HOLD BUFFER FOR NEXT SAMPLE
0037      CALL DULF(INDX)
C
0038      INDX=INDX+1
C
C      CHECK STATUS REGISTER FOR PROPER I/O
C      AND TERMINATE IF STATUS NOT CORRECT
0039      IF (INDX.GT.5) INDX=5
0040      IF (IS2(2).GE.1) GO TO 10
0041      IF (IS3(1).NE.0) GO TO 25
0042      GO TO 10
0043      95 CONTINUE
C
C      IF PROGRAM CRASHES PRINT STATUS ON WAY OUT
0044      WRITE (13,20) ISB(1), ISC(2), IDATA, IBUF(INDX)
0045      200 FORMAT(4I12)
0046      195 CONTINUE
0047      END
    
```

PROGRAM SECTIONS

TORTRAN IV-PLUS V02-51
DELAY13.FTN /TR:BLOCKS/WR

11:32:49 28-APR-78

PAGE 3

NAME	SIZE	ATTRIBUTES
SCODE1	000716 231	RW, I, CON, LCL
SPDATA	000020 8	RW, D, CON, LCL
SIDATA	000226 75	RW, D, CON, LCL
\$VARS	000076 31	RW, D, CON, LCL
\$STEMPS	000002 1	RW, D, CON, LCL

TOTAL SPACE ALLOCATED = 001264 345

.DELAY13=DELAY13

APPENDIX F

Program Listings: BAD.FTN, BAND.FTN, NINE9.FTN, DIFF32.FTN

These programs all execute FIR filters on the pdp 11/45.

SAD.FTH

/TR:ELDOCKS/LR

```
0001      DIMENSION ISUF(6),IRATE(2),ISB(2),H(30),Y(60)
      C
      C      THIS PROGRAM SYNTHESIZES AND EXECUTES
      C      LINEAR PHASE FINITE IMPULSE RESPONSE FILTER
      C      WITH COEFFICIENTS GENERATED USING THE REMEZ
      C      EXCHANGE ALGORITHM.
      C
      C      IT IS A SPECIAL MODIFICATION OF EASY.FTH
      C      FOR THE CASES WHERE N=AN ODD INTEGER
      C
      C      SET ORDER OF FILTER (CAN BE EVEN OR ODD AND EQUAL
      C      TO NUMBER OF H COEFFICIENTS
0002      N=9
      C
      C      INITIALIZE DATA MATRIX FOR FILTER
0003      DO 299 I=1,N
0004      Y(I)=0.0
0005      299 CONTINUE
      C
      C      SET H COEFFICIENTS
0006      H(1)=-.54497945
0007      H(2)=-.28219856
0008      H(3)=-.59043792
0009      H(4)=.32028612
0010      H(5)=.55130921
      C
      C
      C      INITIALIZE LABORATORY PERIPHERAL SYSTEM
0011      CALL ASLSLN (1,ISB)
      C
      C      DESIGNATE OUTPUT DAC CHANNEL FOR DATEL 255
0012      ICHAN=0
      C
      C      DESIGNATE REGISTER FOR LPS11 FLAG SET
0013      IEFH=7
      C
      C      SPECIFY TIME BETWEEN SAMPLES IN MILLISECONDS
0014      IRATE(2)=100
      C
      C      PUT LPS11 PTT ROUTINE IN MILLISECOND SAMPLING MODE
0015      IRATE(1)=3
      C
      C      CALCULATE SAMPLE INTERVAL IN SECONDS
0016      ST=FLOAT(IRATE(2))/1000.
```

FORTRAN IV-PLUS V03-51 11:11:36 28-APR-78 PAGE 2
 10.FTN /TS=CLOCK3/WR

```

      C
      C
      C
      C      INITIATE SYNCHRONOUS SAMPLING
0017  90      CALL RTS(18UF,6.8,DATL,144H,0,1,180)
      C
0018      INDX=5
0019  10      CALL WAITR(IEFN)
0020  15      CALL CLREF(IEFI)
0021      SIGMA=18UF(INDX)
      C
      C      SCALE IN VOLTS
0022      Y(1)=10000.*(SIGMA/2048.)~10000.
      C
      C      BEGIN FILTER ROUTINE
0023      DO 544 I=1,N-1
0024      Y(I+1)=Y(I)
0025  544      CONTINUE
      C
0026      SUM=0.0
0027      DO 20 I=1,N/2
0028      TERM=H(I)*(Y(I)+Y(N-I+1))
0029      SUM=SUM+TERM
0030  20      CONTINUE
0031      TERM=H(N/2+1)*Y(N/2+1)
0032      SUM=SUM+TERM
0033      IDATA=SUM
      C      CALL LPS11 LED DISPLAY ROUTINE
0034      CALL LED(IDATA)
      C
      C      CALL ROUTINE TO OUTPUT RESULT TO DATL 255
0035      CALL IDAC(CHAN, IDATA)
      C
      C      ADJUST POINTERS FOR LPS 1
      C      AND CLEAR H & F BUFFER FOR NEXT SAMPLE
0036      CALL ADJLDC(18UF,1)
      C
0037      INDX=INDX+1
      C
      C      CHECK STATUS REGISTER FOR PROPER I/O
      C      AND TERMINATE IF STATUS NOT CORRECT
0038      IF (INDX.GT.6) INIX=5
0039      IF (158(2).GE.1) GO TO 10
0040      IF (158(1).NE.0) GO TO 10
0041      GO TO 10
0042  95      CONTINUE
      C
      C      IF PROGRAM CRASHES PRIOR TO STATUS ON WAY OUT

```

JOE TRAN IV-PLUS V02-51
LAD.FTN /TR:BLOCKS/

11:11:36 28-APR-78

PAGE 3

0343 WRITE (5,200) ISB(1),ISB(2),IDATA,IBUF(INDX)
0344 200 FORMAT(4I12)
0345 195 CONTINUE
0346 END

PROGRAM SECTIONS

NAME	SIZE	ATTRIBUTES
\$CODE1	000732 237	RW.I.COM.LCL
\$PDATA	000044 18	RW.D.COM.LCL
\$IDATA	000060 24	RW.D.COM.LCL
\$VARS	000630 204	RW.D.COM.LCL
\$STEMPS	000002 1	PW.D.COM.LCL

TOTAL SPACE ALLOCATED = 001710 484

.BAD=BAD

END.FTH

/TR:BLOCKS/WP

```
0001      DIMENSION IBUF(6),IRATE(2),ISB(2),H(30),Y(60)
      C
      C      THIS PROGRAM SYNTHESIZES AND EXECUTES
      C      LINEAR PHASE FINITE IMPULSE RESPONSE FILTER
      C      WITH COEFFICIENTS GENERATED USING THE Remez
      C      EXCHANGE ALGORITHM.
      C
      C      SET ORDER OF FILTER (MUST BE EVEN AND EQUAL
      C      TO NUMBER OF H COEFFICIENTS
0002      N=32
      C
      C      INITIALIZE DATA MATRIX FOR FILTER
0003      Y(1)=0.0
0004      DO 290 I=1,N
0005      299  CONTINUE
      C
      C      SET H COEFFICIENTS
0006      H(1)=-0.57534121E-02
0007      H(2)=0.99027190E-03
0008      H(3)=0.75733545E-03
0009      H(4)=-0.65141192E-02
0010      H(5)=0.13800525E-01
0011      H(6)=0.22951468E-02
0012      H(7)=-0.10394067E-01
0013      H(8)=0.71769560E-02
0014      H(9)=-0.70657362E-01
0015      H(10)=0.11269114E-01
0016      H(11)=0.66233643E-01
0017      H(12)=-0.10497223E-01
0018      H(13)=0.85136133E-01
0019      H(14)=-0.10034907E 00
0020      H(15)=-0.09678577E 00
0021      H(16)=0.7041917E 00
      C      INITIALIZE LABORATORY PERIPHERAL SYSTEM
0022      CALL ASLSLN (1,ISB)
      C
      C      DESIGNATE OUTPUT I/O CHANNEL FOR DATA 230
0023      ICHAN=0
      C
      C      DESIGNATE REGISTER FOR LPS11 FLAG SET
0024      IEFH=7
      C
      C      SPECIFY TIME BETWEEN SAMPLES IN MILLISECONDS
0025      IRATE(2)=100
      C
      C      PUT LPS11 PTS ROUTINE IN MILLISECOND SAMPLE MODE
```

```

0026      IRATE(1)=2
          C
          C
          C      CALCULATE SAMPLE INTERVAL IN SECONDS
0027      ST=FLOAT(IRATE(2))/1000.
          C
          C
          C
          C      INITIATE SYNCHRONOUS SAMPLING
0028      90      CALL RTS(IBUF,6.0,IRATE,IEFN,0,1,133)
          C
0029      INDX=5
0030      10      CALL WAITR(IEFN)
0031      15      CALL CLREF(IEFN)
0032      SIGMA=IBUF(INDX)
          C
          C      SCALE IN VOLTS
0033      Y(1)=10000.*(SIGMA/2048.)-10000.
          C
          C      BEGIN FILTER ROUTINE
0034      DO 544 I=1,N-1
0035      Y(I+1)=Y(I)
0036      544      CONTINUE
          C
0037      SUM=0.0
0038      DO 20 I=1,N-2
0039      TERM=X(I)*Y(I)-Y(N-I+1))
0040      SUM=SUM+TERM
0041      20      CONTINUE
0042      IDATA=SUM
          C      CALL LPS11 LED DISPLAY ROUTINE
0043      CALL LER(IDATA)
          C
          C      CALL ROUTINE TO OUTPUT RESULT TO DATEL 256
0044      CALL IDAC(ICHAN,IDATA)
          C
          C      ADJUST POINTERS FOR LPS11
          C      AND CLEAR HALF BUFFER FOR NEXT SAMPLE
0045      CALL ADJLPS(IBUF,1)
          C
0046      INDX=INDX+1
          C
          C      CHECK STATUS REGISTER FOR PROPER I/O
          C      AND TERMINATE IF STATUS NOT CORRECT
0047      IF (INDX.GT.6) INDX=5
0048      IF (ISB(2).GE.1) GO TO 10
0049      IF (ISB(1).NE.0) GO TO 95
  
```

PAGE 3

```

2850      GO TO 10
2871      95      GO TO 10
C
C      IF PROGRAM CHANGES PRINT STATUS ON WAY OUT
C      OF THE (5,200) 100(1), 100(2), 100(3), 100(4), 100(5)
2852      200      PRINT(4112)
2853      200      PRINT(4112)
2854      195      GO TO 10
2855      195      GO TO 10

```

NAME	SIZE	ATTRIBUTES
SCORE1	001045	PW, I, CON, LCL
SPDATA	000120	PW, D, CON, LCL
SIDATA	000060	PW, D, CON, LCL
SWARS	000630	PW, D, CON, LCL
WTEIPS	000002	PW, D, CON, LCL

TOTAL SPACE ALLIED 360 = 002157 544

.. ID=BAND

FINES.FTH

/TR:BLOCKS/LR

```

0001      DIMENSION IBUF(6),IRATE(2),ISB(2),H(30),Y(60)
      C
      C      THIS PROGRAM SYNTHESIZES AND COMPUTES
      C      LINEAR PHASE FINITE IMPULSE RESPONSE FILTER
      C      WITH COEFFICIENTS GENERATED USING THE REMEZ
      C      EXCHANGE ALGORITHM.
      C
      C      IT IS A SPECIAL MODIFICATION OF LASY.FTH
      C      FOR THE CASES WHERE N=AN ODD INTEGER
      C
      C      SET ORDER OF FILTER (CAN BE EVEN OR ODD AND EQUAL
      C      TO NUMBER OF H COEFFICIENTS
0002      N=9
      C
      C
      C      INITIALIZE DATA MATRIX FOR FILTER
0003      DO 299 I=1,N
0004      Y(I)=0.0
0005      299 CONTINUE
      C
      C      SET H COEFFICIENTS
0006      H(1)=-.54107945
0007      H(2)=-.20316855
0008      H(3)=-.5904370
0009      H(4)=.32028611
0010      H(5)=.53120021
      C
      C
      C      INITIALIZE LABORATORY PERIPHERAL SYSTEM
0011      CALL LPLN(1,ISB)
      C
      C      DESIGNATE OUTPUT DAC CHANNEL FOR INTEL 255
0012      ICHANN=2
      C
      C      DESIGNATE REGISTER FOR LPS11 FLAG SET
0013      IEPH=1
      C
      C
0014      WRITE (5,80)
0015      80 FORMAT(' CUTOFF FREQ. IS 0.25 TIME SAMPLE FREQ. ')
0016      WRITE (5,81)
0017      81 FORMAT('S ENTER SAMPLE PERIOD IN MILLISECONDS- INTEGER: ')
0018      READ (5,82)
0019      82 FORMAT('IS')
      C      PUT LPS11 RTS ROUTINE IN MILLISECOND SAMPLING MODE
0020      IRATE(1)=2
      C

```

```

      C
      C      CALCULATE SAMPLE INTERVAL IN SECONDS
0021      ST=FLOAT(IRATE(2))/1000.
      C
      C
      C
      C
      C      INITIATE SYNCHRONOUS SAMPLING
0022      90      CALL RTS(IDUF,6.0,IRATE,IEFN,0,1,ISB)
      C
0023      INDX=5
0024      10      CALL WAITP(IEFN)
0025      15      CALL CLREF(IEFN)
0026      SIGMA=IBUF(INDX)
      C
      C      SCALE IN VOLTS
0027      Y(1)=10000.*(SIGMA/2048.)-10000.
      C
      C      BEGIN FILTER ROUTINE
0028      DO 544 I=1,N-1
0029      Y(I+1)=Y(I)
0030      544      CONTINUE
      C
0031      SUM=0.0
0032      DO 30 I=1,N/2
0033      TERM=(I)*(Y(I)+Y(I+1))
0034      SUM=SUM+TERM
0035      30      CONTINUE
0036      TERM=(N/2+1)*Y(N/2+1)
0037      SUM=SUM+TERM
0038      IDATA=SUM
      C      CALL LPS11 LED DISPLAY ROUTINE
0039      CALL LED(IDATA)
      C
      C      CALL ROUTINE TO OUTPUT RESULT TO DATEL 256
0040      CALL LPS(ICHAN,IDATA)
      C
      C      ADJUST POINTERS FOR LPS11
      C      AND CLEAR HALF BUFFER FOR NEXT SAMPLE
0041      CALL LPS(IRUF,1)
      C
0042      INDX=INDX+1
      C
      C      CHECK FOR END OF RUN FOR PROPER I/O
      C      AND TERMINATE IF IT IS NOT CORRECT
0043      IF (INDX.GT.6) INDX=5
0044      IF (ISB(2).GE.1) GO TO 10
0045      IF (ISB(1).NE.0) GO TO 95

```

CONTRAN IV-PLUS V02-51

11:32:30

28-APP-78

PAGE 3

INTE.FTH

/TP:BLOCKS/MR

```

0046      60 70 10
0047      95 01 105
          C
          C  IS PROGRAM STOPS PRINT STATUS ON WAY OUT
0048      WRITE (5,000) ISB(1),ISB(2),IDATA,IBUF(INDX)
0049      200
0050      195
0051

```

PROGRAM SECTIONS

NAME	SIZE	ATTRIBUTES
CODE1	001004	258 RW,I,CON,LCL
DATA	000044	18 RW,D,CON,LCL
SDATA	000202	78 RW,D,CON,LCL
SWAPS	000630	204 RW,D,CON,LCL
STEMPS	000002	1 RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 002124 554

NTINCS=NINES

00000.FTH

/TR:BLOCKS/WR

```

0001      DIMENSION IBUF(6),IRATE(2),ISF(2),Y(60)
      C
      C      THIS PROGRAM SYNTHESIZES AND SOLVES
      C      LINEAR PHASE FINITE IMPULSE RESPONSE FILTER
      C      WITH COEFFICIENTS GENERATED USING THE REMEZ
      C      EXCHANGE ALGORITHM.
      C
      C
      C      SET ORDER OF FILTER (MUST BE EVEN AND EQUAL
      C      TO NUMBER OF H COEFFICIENTS
0002      N=32
      C
      C
      C      INITIALIZE DATA MATRIX FOR FILTER
0003      Y(1)=0.0
0004      DO 299 I=1,N
0005      299 CONTINUE
      C
      C      SET H COEFFICIENTS
0006      H(1)=-0.77774121E-02
0007      H(2)=-0.9177198E-03
0008      H(3)=-0.75733545E-02
0009      H(4)=-0.60741152E-02
0010      H(5)=-0.1775297E-01
0011      H(6)=-0.2297471E-02
0012      H(7)=-0.1877777E-01
0013      H(8)=-0.71064558E-02
0014      H(9)=-0.39657333E-01
0015      H(10)=-0.11200114E-01
0016      H(11)=-0.66233043E-01
0017      H(12)=-0.10797221E-01
0018      H(13)=-0.65105133E-01
0019      H(14)=-0.12024007E-02
0020      H(15)=-0.02579472E-01
0021      H(16)=-0.3078172E-01
      C
      C      INITIALIZE LABORATORY PERIPHERAL SYSTEM
0022      CALL ASLSUB (1,199)
      C
      C
      C      DESIGNATE OUTPUT LOG CHANNEL FOR LABEL 256
0023      ICHAN=0
      C
      C
      C      DESIGNATE REGISTER FOR LPS11 PLATO SET
0024      IEFN=7
      C
      C
      C      SPECIFY TIME BETWEEN SAMPLES IN MICROSECONDS
0025      IRATE(2)=100
      C
      C
      C      PUT LPS11 RTS ROUTINE IN POLLING OR SAMPLING MODE

```

```

026      IRATE=1
      C
      C
      C      CALCULATE THE RATE IN SECONDS
027      ST=PLA
      C
      C
      C
      C      INITIATE THE SENSING CANTLING
028      90      CALL PRC (P, E, C, M, V, I, E, F, N, 0, 1, I, S, S)
      C
029      INDEX=
030      10      CALL PRC (P, E, F, N)
031      15      CALL PRC (P, E, F, N)
032      SIGN=1
      C
      C      SCALE THE RATE
033      YCI = (RATE/2048.0)-10000.
      C
      C      BEGIN THE SENSING
034      DO 300
035      YCI=YCI+1
036      544      CONTINUE
      C
037      SUM=0.0
038      DO 300
039      TEST=YCI*(YCI-1+1)
040      SUM=SUM+TEST
041      20      CONTINUE
042      IRATE=SUM
      C
      C      CALL THE LED DISPLAY ROUTINE
043      CALL LED (IRATE)
      C
      C      CALL THE ROUTINE TO OUTPUT RESULT TO DATA 256
044      CALL OUT (IRATE, 256)
      C
      C      ADJUST THE RATE FOR LPS
      C      AND CLEAR THE BUFFER FOR NEXT SAMPLE
045      CALL CLR (BUF, 1)
      C
046      INTR=1
      C
      C      CHECK THE STATUS REGISTER FOR PROPER I/O
      C      AND TEST THE STATUS NOT CORRECT
047      IF (INTR) GOTO 10
048      IF (INTR) GOTO 10
049      IF (INTR) GOTO 10
  
```

TORTPAN IV-PLUS VCC-01
DIFF32.FTH /TR:BLOCKS/WR

11:57:16 28-APR-78

PAGE 3

```
0050      GO TO 10
0051      95      CONTINUE
           C
           C      IF PROGRAM CRASHES PRINT STATUS ON WAY OUT
0052      WRITE (5,200) ISB(1),ISB(2),IDATA,IBUF(INDX)
0053      200      FORMAT(4I12)
0054      195      CONTINUE
0055      END
```

PROGRAM SECTIONS

NAME	SIZE	ATTRIBUTES
\$CODE1	001046 275	RW,I,CON,LCL
\$PDATA	000120 40	RW,D,CON,LCL
\$IDATA	000060 24	RW,D,CON,LCL
\$VARS	000630 204	RW,D,CON,LCL
\$TEMPS	000002 1	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 002100 544

DIFF32=DIFF32

APPENDIX G

Program Listings: CASE8.FTN, CASE8F.FTN, HONEYF.FTN, and BUTTER.FTN

These programs all execute IIR filters on the pdp 11/45 with coefficients from the MAS/FIL program.

LATES.FTH

/TR:BLOCKS/MR

```

0001      DIMENSION ICUF(5),IPATE(2),ISB(2),H(20),Y(20),X(20),S(20)
      C
      C      THIS PROGRAM SYNTHESIZES AND EXECUTES
      C      AN INFINITE IMPULSE RESPONSE FILTER WITH COEFFICIENTS
      C      GENERATED BY THE HPCFIL PROGRAM.  THE FILTERING LOOP
      C      IS FOR THE CASE WHERE THERE IS ONLY ONE NON-RECURSIVE
      C      WEIGHT AND FOUR RECURSIVE WEIGHTS.
      C
      C
      C      N IS THE NUMBER OF RECURSIVE WEIGHTS
0002      N=8
      C
      C
      C
      C      INITIALIZE DATA MATRIX FOR FILTER
0003      DO 299 I=1,N
0004          Y(I)=0.0
0005          X(I)=0.0
0006      299 CONTINUE
      C
      C      SET H COEFFICIENTS
0007      H(1)= 7.4013016
0008      H(2)=-24.070290
0009      H(3)= 44.926980
0010      H(4)=-52.639065
0011      H(5)= 39.645219
0012      H(6)=-18.744024
0013      H(7)= 5.0045574
0014      H(8)= -.921115
      C
      C
      C      SET G COEFFICIENT
0015      G(0)= .001457051
0016      G(1)=-.0087154355
0017      G(2)= .023574271
0018      G(3)=-.039814594
0019      G(4)= .046916724
0020      G(5)=G(7)
0021      G(6)=G(3)
0022      G(7)=G(1)
0023      G(8)=G(0)
      C
      C
      C      INITIALIZE LABORATORY PERIPHERAL SYSTEM
0024      CALL ADLSLN (1,ISB)
      C
      C      DESIGNATE OUTPUT DAC CHANNEL FOR DATEL 255
0025      ICHAN=0
      C

```

CASE8.FTH

/TR:ELC000000

```

      C
      C      DESIGNATE REGISTER FOR LPS11 FLAG SET
0025      IEFN=7
      C
      C
      C      READ IN SAMPLE PERIOD IN MILLISECONDS
0027      WRITE (5,30)
0028      30 FORMAT('ENTER SAMPLE PERIOD IN SECONDS-INTEGERS ')
0029      READ (5,31) IRATE(2)
0030      31 FORMAT (16)
      C
      C      PUT LPS11 RTS ROUTINE IN SECONDS SAMPLING MODE
0031      IRATE(1)=3
      C
      C
      C
      C
0032      90 CALL RTS(IBUF,6,0,IRATE,IEFN,0,1,158)
0033      INDX=5
0034      10 CALL WAITR(IEFN)
0035      15 CALL CLREF(IEFN)
0036      SIGMA=IBUF*INDX
      C
      C      SCALE IN VOLTS
0037      X(0)=10000.*(SIGMA/2048.)-10000.
      C
      C
      C
0038      SUM=0.0
0039      DO 20 I=1,N
0040      TERM=10*Y(I)
0041      SUM=SUM+TERM
0042      20 CONTINUE
0043      SUNE=SUM
0044      TERM=10*Y(1)
0045      SUM=SUM+TERM
0046      SUNE=SUNE+TERM
0047      SUM=SUM+TERM
0048      99 CONTINUE
0049      SUNE=SUNE+TERM
0050      Y(1)=SUNE
0051      IDATA=500.*Y(1)
0052      WRITE(5,345) SUM,SUNE,Y(1),TERM,TERM
0053      345 FORMAT (5F10.3)
      C
      C      SAVE RECURSIVE TERMS

```

```

0054      DO 544 I=0,N
0055      Y(I+2)=Y(I+1)
0056      X(I+1)=X(I)
0057      544 CONTINUE
      C
      C      CALL LPS11 LED DISPLAY ROUTINE
0058      CALL LED(IDATA)
      C
      C      CALL ROUTINE TO OUTPUT RESULT TO DATAL 256
0059      CALL IDATA(256)
      C
      C      ADJUST FORMATTERS FOR LPS11
      C      AND CLEAR LED DISPLAY FOR NEXT SAMPLE
0060      CALL LPS11
      C
0061      C
      C      CHECK STATUS REGISTER FOR ERROR I/O
      C      AND TEST STATUS NOT CORRECT
0062      IF (IN)
0063      IF (ISB(2).GE.1) GO TO 10
0064      IF (ISB(1).GE.0) GO TO 95
0065      GO TO 10
0066      95 CONTINUE
      C
      C      IF PROGRAM CRASHES PRINT STATUS ON WAY OUT
0067      WRITE (5,200) ISB(1),ISB(2),IDATA,IBUF(INDX)
0068      200 FORMAT(4I12)
0069      195 CONTINUE
0070      END
  
```

PROGRAM SECTIONS

NAME	SIZE	ATTRIBUTES
STONE1	001330 364	RW,I,CON,LCL
SPDATA	000104 34	RW,D,CON,LCL
TIMEATA	000150 52	RW,D,CON,LCL
WARS	000566 197	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 002372 637

.CASE8=CASE8

```

0001      DIMENSION IBUF(6),IFATE(2),ISB(2),H(20),Y(20),X(20),G(20)
      C
      C      THIS PROGRAM SYNTHESIZES AND EXECUTES
      C      AN INFINITE IMPULSE RESPONSE FILTER WITH COEFFICIENTS
      C      GENERATED BY THE IACFIL PROGRAM. THE FILTERING LOOP
      C      IS FOR THE CASE WHERE THERE IS ONLY ONE NON-RECURSIVE
      C      WEIGHT AND FOUR RECURSIVE WEIGHTS.
      C
      C
      C      N IS THE NUMBER OF RECURSIVE WEIGHTS
0002      N=8
      C
      C
      C
      C      INITIALIZE DATA MATRIX FOR FILTER
0003      DO 299 I=1,N
0004      Y(I)=0.0
0005      X(I)=0.0
0006      299 CONTINUE
      C
      C      SET H COEFFICIENTS
0007      H(1)= 7.4013016
0008      H(2)=-24.070098
0009      H(3)= 44.926880
0010      H(4)=-52.639065
0011      H(5)= 39.645219
0012      H(6)=-18.744094
0013      H(7)= 5.0865574
0014      H(8)= -.60660115
      C
      C
      C      SET G COEFFICIENTS
0015      G(0)= .0014973961
0016      G(1)=-.0087151755
0017      G(2)= .007571071
0018      G(3)=-.032814394
0019      G(4)= .046915724
0020      G(5)=G(3)
0021      G(6)=G(2)
0022      G(7)=G(1)
0023      G(8)=G(0)
      C
      C      INITIALIZE LABORATORY PERIPHERAL SYSTEM
0024      CALL ASUBLN (1,ISB)
      C
      C      DESIGNATE OUTPUT DAC CHANNEL FOR DATEL 256
0025      ICHAN=0
      C

```

CASE8F.FTN

/TR:BLOCKS/WR

```

      C
      C      DESIGNATE REGISTER FOR LPS11 FLOPS SET
0026      IEFN=7
      C
      C
      C
      C      READ IN SAMPLE PERIOD IN MILLISECONDS
0027      WRITE (5,30)
0028      30 FORMAT('ENTER SAMPLE PERIOD IN MILLISECONDS-INTEGER ')
0029      READ (5,31) IRATE(2)
0030      31 FORMAT (15)
      C
      C      PUT LPS11 RTS ROUTINE IN MILLISECONDS SAMPLING MODE
0031      IRATE(1)=2
      C
      C
      C
      C
0032      90 CALL RTS(IBUF,6.0,IRATE,IEFN,0.1,15B)
0033      INDX=5
0034      10 CALL WAITR(IEFN)
0035      15 CALL CLRLF(IEFN)
0036      SIGMA=IBUF(INDX)
      C
      C      SCALE IN VOLTS
0037      X(0)=10000.*(SIGMA/2048.)-10000.
      C
      C
      C
0038      SUM=0.0
0039      DO 20 I=1,N
0040      TERM=H(I)*Y(I)
0041      SUM=SUM+TERM
0042      20 CONTINUE
0043      SUNE=0.0
0044      NN=(N/2)-1
0045      DO 99 I=0,NN
0046      TURN=G(I)*(X(I)+X(N-1))
0047      SUNE=SUNE+TURN
0048      99 CONTINUE
0049      SUNE=SUNE+G(N/3)*X(N/2)
0050      Y(1)=SUM+SUNE
0051      IDATA=500.*Y(1)
0052      WRITE (5,345) SUM,SUNE,Y(1),TERM,TUPM
0053      345 FORMAT (5F10.3)
      C
      C      SAVE RECURSIVE TERMS

```

LPTPAN IV-PLUS V02-51
CASE8F.FTH /TR:BLOCKS/WR

11:13:13

23-APR-72

PAGE 3

```

0054      DO 544 I=0,N
0055          Y(I+2)=Y(I+1)
0056          X(I+1)=X(I)
0057      544 CONTINUE
      C
      C      CALL LPS11 LED DISPLAY ROUTINE
0058      CALL LED(IDATA)
      C
      C      CALL ROUTINE TO OUTPUT RESULT TO DATEL 256
0059      CALL IDAC(ICHAN, IDATA)
      C
      C      ADJUST POINTERS FOR LPS11
      C      AND CLEAR HALF BUFFER FOR NEXT SAMPLE
0060      CALL ADJLPS(IBUF,1)
      C
0061      INDX=INDX+1
      C
      C      CHECK STATUS REGISTER FOR PROPER I/O
      C      AND TERMINATE IF STATUS NOT CORRECT
0062      IF (INDX.GT.6) INDX=5
0063      IF (ISB(2).GE.1) GO TO 10
0064      IF (ISB(1).NE.0) GO TO 95
0065      GO TO 10
0066      95  CONTINUE
      C
      C      IF PROGRAM CRASHED PRINT STATUS ON WAY OUT
0067      WRITE (5,200) ICHAN(1), ICHAN(2), IDATA, IBUF(INDX)
0068      200  FORMAT(4I12)
0069      195  CONTINUE
0070      END

```

PROGRAM SECTIONS

NAME	SIDE	SIZE	ATTRIBUTES
SCODE1	001330	364	RW,1,CON,LCL
SPDATA	000104	34	RW,D,CON,LCL
SIDATA	000154	54	RW,D,CON,LCL
\$IIRS	000566	187	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 002376 639

.CASE8F=CASE8F

```

0001      DIMENSION IOUT(6),IRATE(2),ISB(2),H(30),Y(60)
      C
      C      THIS PROGRAM SYNTHESIZES AND EXECUTES
      C      AN INFINITE IMPULSE RESPONSE FILTER WITH COEFFICIENTS
      C      GENERATED BY THE MACFIL PROGRAM. THE FILTERING LOOP
      C      IS FOR THE CASE WHERE THERE IS ONLY ONE NON-RECURSIVE
      C      WEIGHT AND FOUR RECURSIVE WEIGHTS.
      C
      C
      C      N IS THE NUMBER OF RECURSIVE WEIGHTS
0002      N=5
      C
      C
      C      INITIALIZE DATA MATRIX FOR FILTER
0003      DO 299 I=1,N
0004      Y(I)=0.0
0005      299 CONTINUE
      C
      C      SET H COEFFICIENTS
0006      H(1)=.52474401
0007      H(2)=-.68504857
0008      H(3)=.30071078
0009      H(4)=-.17432219
0010      H(5)=.63304264
      C
      C
      C      SET G COEFFICIENT
0011      G=.52271373
      C
      C      INITIALIZE LABORATORY PERIPHERAL SYSTEM
0012      CALL ASLSLN (1,ISB)
      C
      C      DESIGNATE OUTPUT DAC CHANNEL FOR DTEL 255
0013      ICHAN=0
      C
      C      DESIGNATE REGISTER FOR LPS11 FLAG SET
0014      IEFH=7
      C
      C      READ IN SAMPLE PERIOD IN MILLISECONDS
0015      WRITE (5,30)
0015      30 FORMAT('ENTER SAMPLE PERIOD IN MILLISECONDS-INTEGER ')
0017      READ (5,31) IRATE(2)
0018      31 FORMAT (16)
      C
      C
      C      PUT LPS11 RTS ROUTINE IN SECONDS SAMPLING MODE
0019      IRATE(1)=2

```

WIEYF.FTH

/TR:BLOCKS-WR

```

      C
      C
      C
      C
0020  90  CALL RTS(1BUF,6.0,IRATE,IEFH,0.1,ISB)
0021      INDX=5
0022  10  CALL WRITER(IEFH)
0023  15  CALL CLREF(IEFH)
0024      SIGMA=1BUF(INDX)
      C
      C  SCALE IN VOLTS
0025      X=10000.*(SIGMA/2048.)-10000.
      C
      C
0026      SUM=0.0
0027      DO 20 I=1,N
0028          TERM=H(I)*Y(I)
0029          SUM=SUM+TERM
0030  20  CONTINUE
0031      Y(I)=SUM+G*Y
0032      IDATA=Y(I)
      C
      C  SAVE RECURSIVE TERMS
0033      DO 544 I=1,N-1
0034          Y(I+1)=Y(I)
0035  544  CONTINUE
      C
      C  CALL LED DISPLAY ROUTINE
0036      CALL LED(IDATA)
      C
      C  CALL ROUTINE TO OUTPUT RESULT TO PATEL 256
0037      CALL IDPO(ICHAN,IDATA)
      C
      C  ADJUST PLINTER FOR LPS11
      C  AND CLEAR HALF BUFFER FOR NEXT SAMPLE
0038      CALL ADJLPS(IEFH,1)
      C
0039      INDX=INDX+1
      C
      C  CHECK STATUS REGISTER FOR PROPER I/O
      C  AND TERMINATE IF STATUS NOT CORRECT
0040      IF (INDX.GT.5) INDX=5
0041      IF (ISB(2).GE.1) GO TO 10
0042      IF (ISB(1).NE.0) GO TO 95
0043      GO TO 10
0044  95  CONTINUE

```

104TRAN IV-PLUS V02-01
HONEYF.FTH /TR:BLOCKS/WR

11:14:29

28-APR-78

PAGE 3

```
      C
      C      IF PROGRAM CRASHES PRINT STATUS ON WAY OUT
0045      WRITE (5,200) ISS(1),ISS(2),IDATA,IBUF(INDX)
0046      200  FORMAT(4I12)
0047      195  CONTINUE
0048      END
```

PROGRAM SECTIONS

NAME	SIZE	ATTRIBUTES
SCODE1	000740 240	RW,I,CON,LCL
IPDATA	000050 20	RW,D,CON,LCL
IDATA	000146 51	RW,D,CON,LCL
SVARS	000634 206	RW,D,CON,LCL
STEMPS	000002 1	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000014 516

,HONEYF=HONEYF

```

0001      DIMENSION IBUF(5),IPATE(2),ISB(2),H(30),Y(60)
      C
      C      THIS PROGRAM SYNTHESIZES AND EXECUTES
      C      AN INFINITE IMPULSE RESPONSE FILTER WITH COEFFICIENTS
      C      GENERATED BY THE MACFIL PROGRAM. THE FILTERING LOOP
      C      IS FOR THE CASE WHERE THERE IS ONLY ONE NON-RECURSIVE
      C      WEIGHT AND FOUR RECURSIVE WEIGHTS.
      C
      C
      C      N IS THE NUMBER OF RECURSIVE WEIGHTS
0002      N=5
      C
      C
      C
      C      INITIALIZE DATA MATRIX FOR FILTER
0003      DO 299 I=1,N
0004      Y(I)=0.0
0005      299 CONTINUE
      C
      C      SET H COEFFICIENTS
0006      H(1)=.92476581
0007      H(2)=-.68524953
0008      H(3)=.30671978
0009      H(4)=-.77488119
0010      H(5)=.05301254
      C
      C
      C      SET G COEFFICIENT
0011      G=.52271771
      C
      C      INITIALIZE LOGICATORY PERIPHERAL SYSTEM
0012      CALL AILSUB(1,ISB)
      C
      C      DESIGNATE OUTPUT DAC CHANNEL FOR DATAL 255
0013      ICHAN=0
      C
      C      DESIGNATE REGISTER FOR LPS11 FLAG SET
0014      IEFH=7
      C
      C      SPECIFY TIME BETWEEN SAMPLES IN MILLISECONDS
0015      IRATE(2)=1
      C
      C      PUT LPS11 PTS ROUTINE IN MILLISECOND SAMPLING MODE
0016      IRATE(1)=3
      C
      C
      C      CALCULATE SAMPLE INTERVAL IN SECONDS
0017      SI=FLOAT(IRATE(2))/1000.

```

: UTTER.FTH

/TP:ELCCYB/JT

```

      C
      C
0018  90  CALL RTS(ISBF,6.0,IRATE,IEFN,0.1,ISB)
0019      INDX=5
0020  10  CALL WAITF(IEFN)
0021  15  CALL CLREF(IEFH)
0022      SIGNA=ISBF(INDX)
      C
      C  SCALE IN VOLTS
0023      X=10000.*(SIGNA/2048.)-10000.
      C
      C
0024      SUM=0.0
0025      DO 20 I=1,N
0026          TERM=H(I)*Y(I)
0027      SUM=SUM+TERM
0028  20  CONTINUE
0029      Y(I)=TERM*G
0030      IDATA=Y(I)
      C
      C  SAVE RECURSIVE TERMS
0031      DO 544 I=1,N-1
0032          Y(I+1)=Y(I)
0033  544  CONTINUE
      C
      C  CALL LPS11 LED DISPLAY ROUTINE
0034      CALL LED(IDATA)
      C
      C  CALL ROUTINE TO OUTPUT RESULT TO DATEL 256
0035      CALL IDAC(IEFH,IDATA)
      C
      C  ADJUST POINTERS FOR LPS11
      C  AND CLEAR HALF BUFFER FOR NEXT SAMPLE
0036      CALL ADJLPS(ISBF,1)
      C
0037      INDX=INDX+1
      C
      C  CHECK STATUS REGISTER FOR PROPER I/O
      C  AND TERMINATE IF STATUS NOT CORRECT
0038      IF (INDX.GT.6) INDX=5
0039      IF (ISB(2).GE.1) GO TO 10
0040      IF (ISB(1).GE.0) GO TO 95
0041      GO TO 10
0042  95  CONTINUE
      C
      C  IF PROGRAM CRASHES PRINT STATUS ON WAY OUT

```

FORTRAN IV-PLUS V02-51

11:28:55

28-APR-78

PAGE 3

BUTTER.FTH /TR:BLOCKS/WR

```
0043      WRITE (5,200) ISB(1),ISB(2),IDATA,ISUF(INDX)
0044      200  FORMAT(4I12)
0045      195  CONTINUE
0046      END
```

PROGRAM SECTIONS

NAME	SIZE	ATTRIBUTES
SCODE1	000716 231	RW,I,CON,LCL
SPLATA	000050 20	RW,D,CON,LCL
SIDATA	000060 24	RW,D,CON,LCL
SVARS	000640 208	RW,D,CON,LCL
STEMPS	000002 1	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 001710 484

.BUTTER=BUTTER

APPENDIX H

The Datel 256 Driver Program: IDAC.MAC

The following assembly language program IDAC takes values from the filter execution programs (EASY, JUNK, and NINE) and writes the results to a digital to analog channel of the Datel 256 system. The call is:

CALL IDAC (ICHAN, IDATA)

where ICHAN is the channel number (0 to 15) and IDATA is the integer data (-2048 to +2048). The channel used now is 0, and the output pins are 12 = high and 11 = common. The channel or channels are addressed in the random mode.

```

      .TITLE  IDAC
IDAC::  MOV    (R5)+,R1      ;GET ARGUMENT FOR DAC
      MOVB   #20,@#160010  ;PUT IN RANDOM MODE
      MOV    @ (R5)+,@#160012 ;SPECIFY CHANNEL 1
      MOV    @ (R5)+,@#160014 ;WRITE VALUE TO DAC,START DAC
      RTS    PC
      .END

```

Program IDAC

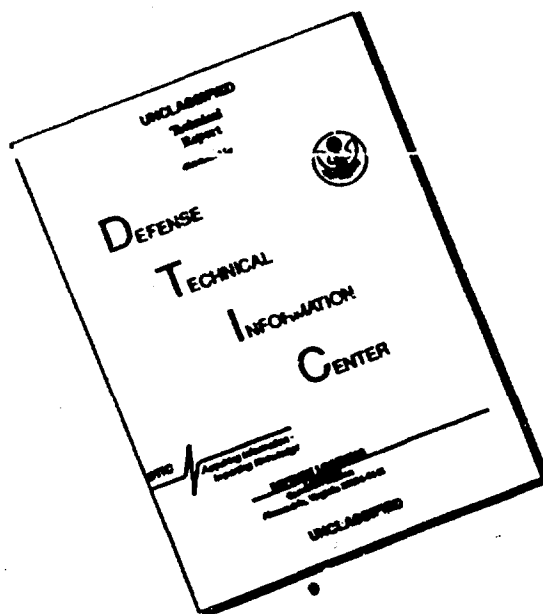
The original program for the Datel 256 system is shown here only for the record. In order to do what needed to be done, the Datel had to be programmed to constantly switch between the block mode for the synchronous inputs and the register mode for the asynchronous outputs. For this reason, this plan was abandoned and the LPS11 system was used instead. The Datel 256 system is a good general purpose piece of equipment, but unfortunately, the particular interface to the pdp 11/45 is not particularly suitable for the digital filter task. At the time the Datel 256 system was originally purchased to work with the SEL810B or the HP2114B, it could not have been foreseen that its use in a closed loop system in conjunction with the pdp 11/45 would be so difficult.

However, since both the Datel 256 system and LPS11 system are available, it appears now that a suitable functional configuration can be obtained by using the LSP11 system with its ADC's and real-time clock for input, and the pdp 11/45 for processing, and the Datel 256 system for the DAC output. A signal flowchart of this configuration is shown in figure 4-1. The original code written to drive the Datel 256 system consisting of the following key instructions:

MOV #2000, R1	Load DMA memory start address into register 1
MOV #1, R2	Load word count into register 2

MOV #60,@#760010	Put into block mode and load status register 2
MOV #1000,@#760016	Put into block mode, and load starting and final addresses
MOV R1, @#760012	Load memory address registers
MOV R2,@#760014	Load word count and start block conversion

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.