Volume II of II

Volume II of III-distribution is unhindered 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 80840

Prepared by:

Dr. E. J. Grimes
Kapner Systems, Inc.
1406 Potter Drive
Colorado Springs, Colorado 80909
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

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

APPENDIX G: PROGRAM LISTING: CASE8.FTN, CASE8F.FTN, HONEY.FTN, and BUTTER.FTN

APPENDIX H: THE DATEL 256 DRIVER PROGRAM: IDAC.MAC

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>Program Listing: CASE8.FTN, CASE8F.FTN, HONEY.FTN, and BUTTER.FTN</td>
<td>130</td>
</tr>
<tr>
<td>H</td>
<td>The DateL 256 Driver Program: IDAC.MAC</td>
<td>143</td>
</tr>
</tbody>
</table>
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. Thus 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 pdpl1/45 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 stairstep 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 result 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 Broderson 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 lots 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 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 slip ring 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 publications are as follows:


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 at 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 nonmodule 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, CASEBF.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.


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.


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. DTEL.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.
DIMENSION IBUF(6), IRATE(2), ISB(2)
COMMON IRATE, IDATA, Y, FREQ

WRITE (5, 80)
FORMAT("ENTER LOW PASS CUTOFF FREQUENCY IN HERTZ ")
READ (5, 81) FREQ

READ IN SAMPLE PERIOD IN MILLISECONDS
WRITE (5, 30)
FORMAT("ENTER SAMPLE PERIOD IN MILLISECONDS")
READ (5, 31) IRATE(2)
FORMAT (15)
C INITIALIZE FILTER FOR SUBROUTINE CUTOFF
Y=0.0
C INITIALIZE LABORATORY PERIPHERAL SYSTEM
CALL ASLSLN (1, ISB)
C DESIGNATE OUTPUT DAC CHANNEL FOR RATE 256
ICHAN=0
C DESIGNATE REGISTER FOR LPS11 FLAG SET
IFEN=7
C SPECIFY TIME BETWEEN SAMPLES IN MILLISECONDS
IRATE(2)=50
C PUT LPS11 RTS ROUTINE IN MILLISECOND SAMPLING MODE
IRATE(1)=2
C INITIATE SYNCHRONOUS SAMPLING
CALL RTS(IBUF,6.0,IRATE,IFEN,0.1,ISB)
INDEX=5
CALL WAITFR(IFEN)
CALL CLREF(IFEN)
SIGMA=IBUF(INDEX)
C SCALE IN VOLTS
IDATA=10000. * (SIGMA/2048.) -10000.
C CALL FILTER ROUTINE
CALL CUTOFF
C CALL LPS11 LED DISPLAY ROUTINE
CALL LED(IDATA)
C CALL ROUTINE TO OUTPUT RESULT TO DATEL 256
   CALL IDAC(IDATA, ICHAN)
C
C ADJUST POINTERS FOR LPSII
C AND CLEAR HALF BUFFER FOR NEXT SAMPLE
   CALL ADJLPS(IBUF,1)
C
C ...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
C...
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.
DIMENSION IBUF(6), IRATE(2), ISB(2)

THIS PROGRAM SYNTHESIZES AND EXECUTES
A FIRST ORDER RECURSIVE FILTER CORRESPONDING
TO A ONE-STAGE PASSIVE RC FILTER

THE ALGORITHM ON WHICH THIS FILTER IS BASED
IS VALID ONLY WHEN THE TIME CONSTANT IS MUCH
GREATER THAN THE PERIOD BETWEEN SAMPLES.
THE TIME CONSTANT SHOULD BE AT LEAST THREE
TIMES AS LARGE AS THE SAMPLE PERIOD

WRITE(5,80)
80 FORMAT('? ENTER LOW PASS CUTOFF FREQUENCY IN HERTZ-FLT.PT. ')
READ (5,81) FREQ

81 FORMAT(F4.2)

READ IN SAMPLE PERIOD IN MILLISECONDS
UPITE (5,30)
30 FORMAT('? ENTER SAMPLE PERIOD IN MILLISECONDS-INTEGER ')
READ (5,31) IRATE(2)
31 FORMAT (16)

INITIALIZE FILTER

Y=0.8

INITIALIZE LABORATORY PEKIPHERAL SYSTEM
CALL ASLSLN (1,139)

DESIGNATE OUTPUT DAC CHANNEL FOR DATEL 256
ICHAN=0

DESIGNATE REGISTER FOR LPSII FLAG SET
IEFN=7

PUT LPSII RTS ROUTINE IN MILLISECOND SAMPLING MODE
IRATE(1)=2

CALCULATE RC TIME CONSTANT IN SECONDS
RC=1./(FREQ*6.28)

CALCULATE SAMPLE INTERVAL IN SECONDS
ST=FLOAT(IRATE(2))/1000.
C CALCULATE FILTER PARAMETER
TCOH=ST/RC
TCONC=1.-TCON

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
IF (TCONL.GE.1) GO TO 301
GO TO 555
301 WRITE(S,201)
201 FORMAT(' UNSTABLE-SAMPLE RATE TOO SLOW FOR TIME CONSTANT')
GO TO 195
555 CONTINUE

C INITIATE SYNCHRONOUS SAMPLING
90 CALL PT$(IBUF,G,0,IRATE,IEFN,0.1,1SB)

INDEX=5
10 CALL WAITR(IEFN)
15 CALL CLREF(IEFN)
20 SIGMA=IBUF(INDX)

SCALE IN VOLTS
30 IDATA=10000.*(SIGMA/2048.)-10000.

BEGIN FILTER ROUTINE

Y=TCONC*Y+TCON*DATA
Y=DATA
CALL LPS11 LED DISPLAY ROUTINE
CALL LED(IDATA)

CALL ROUTINE TO OUTPUT RESULT TO DATEL 256
CALL IDAC(ICHAR,Y,DATA)

ADJUST POINTERS FOR LPS11
AND CLEAR HALF BUFFER FOR NEXT SAMPLE
CALL ADJLPS(ISUB,1)

INDEX=INDEX+1

CHECK STATUS REGISTER FOR PROPER I/O
AND TERMINATE IF STATUS NOT CORRECT
6 IF (INDEX.GT.C) INDEX=5
7 IF (ISB(2).GE.1) GO TO 10
IF (ISB(1).NE.0) GO TO 95
GO TO 10
95 CONTINUE
C
C IF PROGRAM CRASHES PRINT STATUS ON WAY OUT
WRITE (5,200) ISB(1),ISB(2),IDATA,IBUT(INDEX)
200 FORMAT(4I12)
195 CONTINUE
END

PROGRAM SECTIONS

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIZE</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODE1</td>
<td>000676</td>
<td>223</td>
</tr>
<tr>
<td>DATA</td>
<td>000224</td>
<td>10</td>
</tr>
<tr>
<td>WDATA</td>
<td>000320</td>
<td>168</td>
</tr>
<tr>
<td>WORKS</td>
<td>000070</td>
<td>28</td>
</tr>
</tbody>
</table>

TOTAL SPACE ALLOCATED = 001342 369

COD=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.
Program ASKFER
NEG=1
IF(JTYPE.EQ.1) NEG=0
NGRD=INTLFT/2
NODD=INTLFT-2*NODD
NFCHS=INTLFT/2
IF(NODD.EQ.1.AND.NEG.EQ.0) NFCHS=NFCHS+1

SET UP THE DENSE GRID. THE NUMBER OF POINTS IN THE GRID
IS (FILTER LENGTH + 1)*GRID DENSITY/2

GRID(1)=EDGE(1)
DELF=LGREN/FCHS
DELF=0.5/DELF
IF (NEG.EQ.0) GO TO 135
IF (EDGE(1).LT.DELF) GRID(1)=DELF

135 CONTINUE
J=1
L=1
LBAND=1
140 FUP=EDGE(L+1)
145 TEMP=GRID(J)

CALCULATE THE DESIRED MAGNITUDE RESPONSE AND THE WEIGHT
FUNCTION OF THE GRID

DES(J)=EFF(TEMP,FX,UTX,LBAND,JTYPE)
WT(J)=WATE(TEMP,FX,UTX,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,UTX,LBAND,JTYPE)
WT(J-1)=WATE(FUP,FX,UTX,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

SET UP A NEW APPROXIMATION PROBLEM WHICH IS EQUIVALENT
TO THE ORIGINAL PROBLEM

IF(NEG) 170, 170, 180
170 IF (NODD.EQ.1) GO TO 200
DO 175 J=1,NGRID
CHANGE=DCOS(P1GRID(1))

Program ASKFFR
DE5(J) = DE5(J)/CHANCE
175 UT(J) = UT(J)/CHANCE
   GO TO 200
190 IF (H000.EQ.1) GO TO 190
   DO 185 J = 1, NGRID
   CHANGE = DSIN(P1*GRID(J))
   DE5(J) = DE5(J)/CHANGE
   GO TO 200
185 UT(J) = UT(J)/CHANCE
   DO 190 J = 1, NGRID
   CHANGE = DSIN(P2*GRID(J))
   DE5(J) = DE5(J)/CHANGE

C
195 UT(J) = UT(J)/CHANGE
   INITIAL GUESS FOR THE EXTREMAL FREQUENCIES--EQUALLY
   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) = IGRID
   NM1 = NFCNS-1
   N2 = NFCNS+1
C
   CALL THE REMEZ EXCHANGE ALGORITHM TO DO THE APPROXIMATION
   PROBLEM
C
   CALL REMEZ(EDGE, NBANDS)
1000 FORMAT(' HERE WE GO')
   WRITE(5, 1000)
C
   CALCULATE THE IMPULSE RESPONSE
C
IF (HES) 300, 300, 320
300 IF (H000.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 = 1, 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 (H000.EQ.0) GO TO 330
   H(1) = 0.2*ALPHA(NFCNS)
   H(2) = 0.25*ALPHA(NM1)
   DO 325 J = 1, NM1
325 H(J) = 0.25*(ALPHA(NZ-J)-ALPHA(NFCNS+3-J))
   H(NFCNS) = 0.5*ALPHA(1)-0.25*ALPHA(3)
   H(N2) = 0.0
   GO TO 350
330 H(1) = 0.25*ALPHA(NFCNS)

Program ASKFiR
335 J=1,NFCHS
340 K=NFILT+1-J
345 CONTINUE
350 FORMAT(2X,'DEVIATION IN DB*5F15.9)
355 FORMAT(2X,'DEVIATION IN DB*5F15.9)

Program ASKFIR
450 CONTINUE
WRITE (6,455) (GRID(IEXT(J)),J=1,NZ)
455 FORMAT(2X,'EXTERNAL FREQUENCIES'/2X,5F12.7))
WRITE (6,460)
458 F::=AT(/1X,70(1H1)/1H1)
C IF(NFILT.NE.0) GO TO 100
STOP
END
C FUNCTION EFF(TEMP,FX,WTX,LBAND,JTYPE)
C FUNCTION TO CALCULATE THE DESIRED MAGNITUDE RESPONSE
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 FUNCTION WATE(TEMP,FX,WTX,LBAND,JTYPE)
C FUNCTION TO CALCULATE THE WEIGHT FUNCTION AS A
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 SUBROUTINE ERROR
WRITE (6,1)
1 FORMAT('********** ERROR IN INPUT DATA **********')
STOP
END
C SUBROUTINE RENEZ(EDGE,NBANDS)
C COMMON F12,AD,DEV.X,Y,GRID,DES,UT,ALPHA,EEXT,NFCNS,NGRID

Program ASK Fir
DIMENSION EDGE(20)
DIMENSION ICT(66), AD(66), ALPHA(65), X(66), Y(66)
DIMENSION DES(1045), GRID(1045), UT(1045)
DIMENSION A(66), F(66), D(65)
DOUBLE PRECISION P12, DNUM, DDEH, DTEM, A, P, Q
DOUBLE PRECISION AD, DEV, X, Y

C

ITERMAX=200
DEVL=-1.0
NZ=IFCN'S+1
WRITE(5, 222)
222 FOR M(STARTING REREMZ')
NZ=IFCN'S+2
NITER=0
100 CONTINUE
IEXT(NZ)=NGRID+1
NITER=NITER+1
IF(NITER.GT.ITERMAX) GO TO 400
DO 110 J=1, NZ
DTEMP=GRID(IEXT(J))
DTEMP=DCOS(DTEMP*P12)
110 X(J)=DTEMP
JET=(IFCN'S-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=AD(J)*UT(L)
DDEN=DDEN+DTEMP
130 K=-K
DEV=DNUM/DDEN
NU=1
IF(DEV.GT.0.0) NU=-1
DEV=-NU*DEV
K=KU
DO 140 J=1, NZ
L=IEXT(J)
DTEMP=K*DEV*UT(L)
Y(J)=DES(L)+DTEMP
140 K=-K
WRITE(5, 799)
799 FOR M(T REREMZ CHECKPOINT 2')
IF(DEV.GE.DEVL) GO TO 150
CALL DUCH
GO TO 400
150 DEVL=DCV
JCHINS=I
K1=IEXT(1)

Program ASKFIR
SEARCH FOR THE EXTREME FREQUENCIES OF THE BEST APPROXIMATION

200 IF (J.EQ.0) YNZ=COMP
   IF (J.GT.0) KUP=J+1
   L=J+1
   NUT=NUT
   IF (J.EQ.2) Y1=COMP
   COMP=DEV
   IF (L.GE.KUP) GO TO 300
   WRITE (5, 596)
   596 FORMAT (' WE GOT TO HERE')
   ERR=GEE(L,NZ)
   EPR=(ERR-DES(L)+LUT(L))
   DTEMP=NUT*ERR-COMP
   WRITE (5, 6398) ERR, DES(L), LUT(L), COMP, DTEMP, NUT, GEE(L, NZ)

8398 FORMAT (5F10.4, 13, 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)+LUT(L))
   DTEMP=NUT*ERR-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
   JCHANGE=JCHANGE+1
   GO TO 200

220 L=L-1
   WRITE (5, 789)
   789 FORMAT (' REZ 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)+LUT(L))
   DTEMP=NUT*ERR-COMP
   IF (DTEMP.GT.0.0) GO TO 230
   IF (JCHANGE.LE.0) GO TO 225

230 COMP=NUT*EPR
   WRITE (5, 975)
Program ASKFIR

999 FORMAT(' RENEZ CHECKPOINT 3.2')
235 L=L-1
   IF(L.LE.KLOW) GO TO 240
   E**=GEE(L,HE)
   ERR=(ERR-DES(L))*UT(L)
   DTEP=HUT*ERR-RES
   IF(DTEMP.LE.0.0) GO TO 240
   COMP=HUT*ERR
   GO TO 235
240 KLOW=IEXT(J)
   IEXT(J)=L+1
   J=J+1
   JCHNG=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,HZ)
   ERR=(ERR-DES(L))*UT(L)
   DTEP=HUT*ERR-RES
   IF(DTEMP.LE.0.0) GO TO 225
   COMP=HUT*ERR
   GO TO 210
260 KLOW=IEXT(J)
   J=J+1
   GO TO 200
300 IF(J.GT.HIZ) GO TO 320
   IF(K1.GT.IEXT(1)) K1=IEXT(1)
   IF(KHZ.LT.IEXT(HZ)) KHZ=IEXT(HZ)
   NUT1=HUT
   WRITE(5,432)
432 FORMAT(' RENEZ CHECKPOINT 3.5')
   NUT=P-HU
   L=0
   KUP=K1
   COMP=VIZ*(L,0.0)
   LUCK=1
310 L=L+1
   IF(L.GE.KUP) GO TO 315
   ERR=GEE(L,HZ)
   ERR=(ERR-DES(L))*UT(L)
   DTEP=HUT*ERR-RES
   IF(DTEMP.LE.0.0) GO TO 310
   COMP=HUT*ERR
   J=HIZ
   GO TO 210
315 LUCK=6
   GO TO 325
320 IF(LUCK.GT.9) GO TO 350
   IF(COMF.GT.Y1) Y1=COMP
   K1=IEXT(1)
325 L=HIZ+1
   KLOW=HZ
C
C  CALCULATION OF THE COEFFICIENTS OF THE BEST
C  APPROXIMATION USING THE INVERSE DISCRETE
C  FOURIER TRANSFORM
C
400 CONTINUE
N11=NFCNS-1
F5H=1.0E-06
GTEMP=GRID(1)
X(N22)=-2.0
CN=N4FCNS-1
DELF=1.0/CH
L=1
KJK=0
IF(EDGE(1).EQ.0.0.AND.EDGE(24).EQ.0.5) KKK=1
IF(NFCNS.LE.3) KKK=1
IF(KKK.EQ.1) GO TO 405
DTEMP=DCOS(P12+GRID(1))
DNUM=DCOS(P12+GRID(HGRID))
AA=2.0/(DTEMP-DNUM)
BB=-(DTEMP+DNUM)/(DTEMP-DNUM)
405 CONTINUE
DO 430 J=1,NFCNS
FT=(FLOAT(J-1))*DELF
XT=DCOS(P12+FT)
IF(KKK.EQ.1) GO TO 410
XT=(XT-BB)*AN
C ARCOS=ARAN(XT/SORT(1.0-XT*XT))
FT=DCOS(CT/P12

Program ASKFIR
LISTING

```
C FT=.5
410 XE=':'L)
  IF(YLT.XE) GO TO 420
  IF(KLTE.XT)LT.FSH) GO TO 415
  L=L+1
  GO TO 410
415 Y(J)=Y(L)
  GO TO 425
420 IF(KLT.XE)LT.FSH) GO TO 415
  GRID(1)=FT
  A(J)=$HE(1,1Z)
425 CONTINUE
  IF(L.LT.1) L=L-1
430 CONTINUE
  GRID(1)=GTEMP
  DDEN=P12/CH
  DO 510 J=1,NFCHS
    DTEMP=C.0
    DNUM=(FLOAT(J-1))*DDEN
    IF(HLT.1) GO TO 505
    DO 500 K=1,NM1
      DTEMP=DTEMP+A(K+1)*DCOS(DNUM*K)
      DO 505 1=1,2
        DTEMP=2.0*DTEMP+A(J)
      505 DTEMP=2.0*DTEMP+A(J)
  510 ALPHA(J)=DTEMP
  DO 550 J=2,NFCHS
  520 ALPHA(J)=2+ALPHA(J)/CN
  530 ALPHA(J)=ALPHA(J)/CN
  IF(NK.EQ.1) GO TO 545
  P(J)=2.0+ALPHA(NFCHS)BD+ALPHA(NM1)
  P(J)=2.0+ALPHA(NFCHS)
  545 CONTINUE
  DO 550 J=1,NFCHS
  550 ALPHA(J)=DTEP

  P(J+1)=0.0
  DO 520 K=1,J
    A(K)=P(K)
    P(K+1)=P(K)+2.0*BB*A(K)
    P(K+1)=P(K)+2.0*AA
    J=J-1
    DO 525 K=1,JM1
    P(K)=P(K)+0(K)+AA*A(K)
    JP1=K+1
    DO 550 K=3,JP1
    P(K)=P(K)*AA+A(K-1)
    IF(J.EQ.INI) GO TO 540
    DO 535 K=1,J
    535 P(K)=A(K)
    IF(J.EQ.INI) GO TO 540
    DO 540 CONTINUE
    DO 545 J=1,NFCHS
  545 ALPHA(J)=P(J)
```

Program ASKFINR
545 CONTINUE
    IF(NFCN5.GT.3) RETURN
    WRITE(5,765)
765 FORMAT(' PEIEZ CHECKPOINT 5')
    ALPHA(NFCN5+1)=0.0
    ALPHA(NFCN5+2)=0.0
    WRITE(5,556)
556 FORMAT(' LEAVING PEIEZ')
    RETURN
END

C
C
DOUBLE PRECISION FUNCTION D(K,N,M)
C
FUNCTION TO CALCULATE THE LAGRANGE INTERPOLATION COEFFICIENTS
FOR USE IN THE FUNCTION GEE.
C
COMMON P12,AD,DEV,X,Y,GRID,DES,UT,ALPHA,INDEX,NFCN5,NGRID
DIMENSION INDEX(65),AD(65),ALPHA(65),X(65),Y(65)
DIMENSION DES(1045),GRID(1045),UT(1045)
DOUBLE PRECISION AD,DEV,X,Y
DOUBLE PRECISION P12
DOUBLE PRECISION D
DOUBLE PRECISION P12

D=1.0
0=X(K)
DO 3 L=1,M
    DO 2 J=L,M
        IF((0-X(J))*2.1 1
        1 D=2.0*(0-X(J))
        CONTINUE
    2 CONTINUE
3 CONTINUE
    D=1.0/D
    RETURN
END

C
C
DOUBLE PRECISION FUNCTION GEE(K,N)
C
FUNCTION TO EVALUATE THE FREQUENCY RESPONSE USING THE
LAGRANGE INTERPOLATION FORMULA IN THE BARYCENTRIC FORM
C
COMMON P12,AD,DEV,X,Y,GRID,DES,UT,ALPHA,INDEX,NFCN5,NGRID
DIMENSION INDEX(65),AD(65),ALPHA(65),X(65),Y(65)
DIMENSION DES(1045),GRID(1045),UT(1045)
DOUBLE PRECISION P,C,D,XF
DOUBLE PRECISION P12
DOUBLE PRECISION AD,DEV,X,Y
F=0.0
XF=GRID(K)
XF=DCOS(P12)*XF
D=0.0
DO 12 J=1,M
    C+XF-X(J)
12 CONTINUE
Program ASJFIR
C WRITE (5,6666) X(J)
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
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
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.
DIMENSION BUF(6), ISRC(2), IRAT(2)

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 ISFCC CONTROL.

WRITE(5,50)
66 FORMAT('ENTER TIME DELAY MULTIPLIER-FLT.PT. *)
0604 READ (5,61) FREQ
6855 81 FORMAT(F10.7)
0606 IFREQ=FREQ

C READ IN SAMPLE PERIOD IN MILLISECONDS
WRITE (5,50)
5208 30 FORMAT('ENTER SAMPLE PERIOD IN MILLISECONDS-INTEGER *)
0209 READ (5,31) IRAT(2)
310 FORMAT (16)

C INITIALIZE FILTER

011 C INITIALIZE LABORATORY PERIPHERAL SYSTEM
012 CALL ASCLNT (1.15D)

C DESIGN: INPUT PMU CHANNEL FOR DATEL 256

013 C DESIGNATE REGISTER FOR LPS11 FLAG SET
014 1EH=7

C PUT LPS11 RTS ROUTINE IN MILLISEC. SAMPLING MODE
IRATE(1)=2

C INITIATE SYNCHRONOUS SAMPLING
016 90 CALL RTS (IRAT, 6, 0, IRATE, 1EH, 0.15D)

C
017 IN=5
2100 1D CALL INPUT(1EH)
015 15 CALL CLEEP (IRAT)
0120 SEND-INPUT(1EH)
C SCALE IN VOLS.
0021 IDATA=16230.+*(SIGMA/2048.)-10000.
C
C BEGIN FILTER ROUTINE
0022 INDIFF=INDEC*(16384.-IDATA)
0023 IN=INDEC+INDIFF
0024 INF=IDATA
C
0025 DL=DP=25
0026 BL=EP=8
0027 INDIFF=-5000.
0028 IDATA=16384.
0029 INDEC=IDATA
0030 IDelay=IDP0E-IDPITY
0031 IF(IDelay.LT.2) G0 TO 9977
0032 DO 9977 IN=1.IDEALY
0033 BLAP=1.
C WASTE=.1*(BLAP)
C WASTE=.1*(AP)
0034 G0 TO 9977
C CALL LP311 LED DISPLAY ROUTINE
0035 CALL LED0PDATA)
C
C CALL ROUTINE TO OUTPUT RESULT TO DATEL 256
0036 CALL TRAC0PNT1D1M0DATA)
C
C ADJUST POINTERS FOR LP311
C "NO CLEAR HU. BUTTON FOR NEXT SAMPLE"
0037 CALL DULF
C
C
C C,C LUN
C C CHECK STATUS REGISTER FOR PROPER I/O
C AND TERMINATE IF STATUS NOT CORRECT
0039 IF (INDX.GT.5) G0 TO 995
0040 IF (152(2).GE.1) G0 TO 10
0041 IF (152(1).GE.0) G0 TO 25
0042 G0 TO 10
0043 95 CONTINUE
C
C C IF PROGRAM CRASHES PRINT STATUS ON WAY OUT
0044 WRITE (4,112) 152(1),152(2),152(0),153,155,156
0045 200 LUN(1)=4
0046 195 CONTINUE
0047 END

PROGRAM SECTIONS
<table>
<thead>
<tr>
<th>NAME</th>
<th>SIZE</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCODE1</td>
<td>000716</td>
<td>231</td>
</tr>
<tr>
<td>SPDATA</td>
<td>000028</td>
<td>8</td>
</tr>
<tr>
<td>SIDATA</td>
<td>000226</td>
<td>75</td>
</tr>
<tr>
<td>SVARS</td>
<td>000076</td>
<td>31</td>
</tr>
<tr>
<td>STEPS</td>
<td>000002</td>
<td>1</td>
</tr>
</tbody>
</table>

**TOTAL SPACE ALLOCATED = 001264**

.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.
DIMENSION ISUF(6), IRATE(2), ISB(2), H(30), Y(68)

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.FTN
C FOR THE CASES WHERE N = AN ODD INTEGER
C
C SET ORDER OF FILTER (Can be even or odd and equal
to number of H coefficients)
C
C N = 9
C
C INITIALIZE DATA MATRIX FOR FILTER
C
DO 299 I=1,9
Y(I)=0.0
299 CONTINUE
C
SET H COEFFICIENTS
H(1)=-.54497945
H(2)=-.20219056
H(3)=-.5042782
H(4)=.32078612
H(5)=.5510031
C
C INITIALIZE LABORATORY PERIPHERAL SYSTEM
CALL ASL5LN(1,15B)
C
DESIGNATE OUTPUT LOC CHANNEL FOR DATEL 296
ICHAN=0
C
DESIGNATE REGISTER FOR LPS11 FLAG SET
IEFH=7
C
SPECIFY TIME BETWEEN SAMPLES IN MILLISECONDS
IRATE(2)=100
C
PUT LPS11 PT ROUTINE IN MILLISECOND SAMPLING MODE
IRATE(1)=0
C
CALCULATE SAMPLE INTERVAL IN DEGREES
ST=FLOAT(IRATE(2))/1000.
C INITIATE SYNCHRONOUS SAMPLING
C
C017 98 CALL RTS(IPUSF.6.8.14ATE.1.4.0.6.1.15)
C
C018
C019 10 CALL WAIT Ferr(EFN)
C020 15 CALL CLEER(EFN)
C021
C SCALE IN VOLTS
C Y(I)=10000.*(SIGNAL/2048)-10000.
C
C C022
C
C C023 BEGIN FILTER ROUTINE
C DO 544 I=1,N-1
C Y(I+1)=Y(I)
C025 544 CONTINUE
C
C C026 SUM=0.0
C DO 20 I=1,N/2
C C026 TERMIN:=(Y(I)+Y(N-I+1))
C029 SUM=SUM+TERMIN
C C030 20 CONTINUE
C031 TERMIN=(Y(N/2)+Y(N/2+1))
C032 SUM=SUM+TERMIN
C C033 IDATA=SUM
C C CALL LPS11 LED DISPLAY ROUTINE
C C CALL LED(IDATA)
C C CALL ROUTINE TO OUTPUT RESULT TO DATEL 255
C C CALL IDAC(CHAIN,IDATA)
C
C C C035 ADJUST POINTERS FOR LP5
C C AND CLEAR IF IF BUFFER FOR NEXT SAMPLE
C C036 CALL ADJLPS(IPUSF.1)
C
C C C037 INDEX=INDEX+1
C
C C C038 CHECK STATUS REGISTER FOR PROPER I/O
C C AND TERMINATE IF STATUS NOT CORRECT
C C039 IF (INDEX.GT.6) INDEX=1
C C039 IF (IS8(2).LE.1) GO TO...
C C040 IF (IS8(1).NE.0) GO TO...
C C041 GO TO 10
C C042 95 CONTINUE
C
C C IF PROGRAM CRASHES PRIEST STATUS ON WAY OUT
WRITE (5, 200) ISO(I), ISQ(2), IDATA, IBUF(INDX)

200 FORMAT(4I12)

195 CONTINUE

END

PROGRAM SECTIONS

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIZE</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCODE1</td>
<td>000732</td>
<td>237</td>
</tr>
<tr>
<td>SPDATA</td>
<td>000044</td>
<td>18</td>
</tr>
<tr>
<td>SIDATA</td>
<td>000060</td>
<td>24</td>
</tr>
<tr>
<td>VHRS</td>
<td>000630</td>
<td>204</td>
</tr>
<tr>
<td>STEMS</td>
<td>000002</td>
<td>1</td>
</tr>
</tbody>
</table>

TOTAL SPACE ALLOCATED = 001710 484

BAD = BAD
DIMENSION BUF(5), IRATE(2), ISB(2), H(30), Y(60)

THIS PROGRAM SYNTHESIZES AND EXECUTES LINEAR PHASE FINITE IMPULSE RESPONSE FILTERS WITH COEFFICIENTS GENERATED USING THE PRGD EXCHANGE ALGORITHM.

SET ORDER OF FILTER (MUST BE EVEN AND EQUAL TO NUMBER OF H COEFFICIENTS)

N=32

INITIALIZE DATA MATRIX FOR FILTER

Y(1)=0.0

DO 299 I=1,N

CONTINUE

SET H COEFFICIENTS

H(1)=-0.575341216E-02
H(2)=-0.390271096E-03
H(3)=-0.229776452E-02
H(4)=-0.261411532E-02
H(5)=-0.132052829E-01
H(6)=-0.329518992E-02
H(7)=-0.123640671E-01
H(8)=-0.716335600E-02
H(9)=-0.72972855E-01
H(10)=+0.11260112E-01
H(11)=-0.66323503E-01
H(12)=+0.10497233E-01
H(13)=+0.851361278E-01
H(14)=+0.12738075E00
H(15)=-0.29678577E00
H(16)=+0.56419176E00

INITIALIZE LABORATORY PERIPHERAL SYSTEM

CALL ASLELH (1.19A)

DESIGNATE OUTPUT DAC CHANNEL FOR DATEL 225
ICHAN=0

DESIGNATE REGISTER FOR LPS11 FLAG SET
IEN=7

SPECIFY TIME BETWEEN SAMPLES IN MILLISECOND
INITS(2)=150

PUT LPS11 RTS ROUTINE IN MILLISECOND SAMPLE MODE
IRATE(1) = 2

CALCULATE SAMPLE INTERVAL IN SECONDS
ST = \text{FLOAT(IRATE(2))} / 1000.

INITIATE SYNCHRONOUS SAMPLING
CALL RTS(IBM, 0, IRATE, IEFN, 0, 1, 13)

INDEX = 5
CALL WAITF(IEFN)
CALL CLRDF(IEFN)
SIGMA = IBUF(INDEX)
SCALE IN VOLTS
Y(1) = 10(20.0 + SIGMA/2048) - 10000.

BEGIN FILTER ROUTINE
DO 544 I = 1, 11 - 1
   Y(I+1) = Y(I)
   CONTINUE

SUM = 0.0
DO 20 I = 1, 11
   TERM = Y(I) * (Y(I+1) + Y(I-1))
   SUM = SUM + TERM
20 CONTINUE
IDAT = SUM
CALL LP511 LED DISPLAY ROUTINE
CALL LEPF(IDATA)
CALL ROUTINE TO OUTPUT RESULT TO DATEL 256
CALL INAM(CHAN, IDATA)
ADJUST POINTERS FOR LP511
AND CLEAR HALF BUFFER FOR NEXT SAMPLE
CALL FLLP50(IBUF, 1)
INDEX = INDEX + 1
CHECK STATUS REGISTER TO PROPER I/C
AND TERMINATE IF STATUS NOT CORRECT
IF (INDEX .GT. 6) PIOS = 5
IF (ISRA(2) .GE. 1) GO TO 10
IF (ISB(1) .NE. 0) GO TO 95
```
FORTRAN IV-PLUS: 26-51  11:14:24  26-APR-78  PAGE 3
BAND: FTH  24:50:BLOCKS-41

2050 CONT 10 10
371 95 CONTINUE

C  C PROGRAM CONT'ING PRNT STATUS ON LAY OUT

515 UST.(5,280) I12(11), ISB(2), I1DATA, IBUF(INDX)

6065 200 F' CONT(4112)
6034 195 CONTINUE
3055 END

PROGRAM SECTION

NAME SIZE ATTRIBUTES

SCORE1 001045 15 PU, I, CON, LCL
STAT1 000120 10 PU, D, CON, LCL
DATA1 000060 4 PU, D, CON, LCL
SVRS 000530 1 PU, D, CON, LCL
WENPS 000002 1 PU, D, CON, LCL

TOTAL SPACE ALLRD = 002167

END=BAND
```
DIMENSION ISUF(6), IRATE(2), ISB(2), H(30), Y(60)

THIS PROGRAM SYNTHESIZES AND CULUTES
LINEAR PHASE FINITE IMPULSE RESPONSE FILTER
WITH COEFFICIENTS GENERATED USING THE REMEZ
EXCHANGE ALGORITHM.

IT IS A SPECIAL MODIFICATION OF CRAY.FTN
FOR THE CASE WHEN N=AN ODD INTEGER.

SET ORDER OF FILTER (CAN BE EVEN OR ODD AND EQUAL
to number of H COEFFICIENTS

INITIALIZE DATA MATRIX FOR FILTER

DO 299 I=1,11
   Y(I)=0.0
   CONTINUE

SET H COEFFICIENTS

H(1)=-.54107945
H(2)=-.20315355
H(3)=-.53043770
H(4)=.32528510
H(5)=.03120021

INITIALIZE LABORATORY PERIPHERAL SYSTEM

CALL *TNH (*ISB)

DESIGNATE OUTPUT DAC CHANNEL FOR INTEL 255

DDB = THE REGISTER FOR LPS11 PLG 1 SET

WRITE (5,80)

80 FORMAT (' CUTOFF FREQ. IS 0.25 TIME SAMPLE FREQ. '")

WRITE (5,81)

81 FORMAT (' ENTER SAMPLING PERIOD IN MILLISECONDS- INTEGER: '")

READ (5,82)

82 FORMAT (1E16.11)

PUT LPS11 REJ ROUTINE IN MILLISECONDS SAMPLING NODE

IRATE(1)=2
C CALCULATE SAMPLE INTERVAL IN SECONDS
6021 ST=FLOAT(IRATE(2)/1000).

C INITIATE SYNCHRONOUS SAMPLING
6022 90 CALL RTS(IDUP, 6.0, IRATE, IEFH, 0, 1, ISS)

C 
6023 60 CALL UARTF(IFEH)
6024 10 CALL CLRF(IEFH)
6025 15 CALL CLRIF(IEFH)
6026 67 SIGA=IBUF(INC)

C SCALE IN VOLTS
6027 Y(I)=10000.*((SIGA/2048.)-1.0000).

C BEGIN FILTER ROUTINE
6028 DO 544 I=1,N-1
6029 544 Y(I+1)=Y(I)
6030 544 CONTINUE

C 
6031 SUM=0.0
6032 DO 29 I=1,N/C
6033 TERM=TERM+Y(I)*Y(I+1)
6034 SUM=SUM+TERM
6035 29 CONTINUE
6036 TMIN=(N/2+1)*Y(N/2+1)
6037 SUM=SUM-TERM
6038 INSTA=SUM

C CALL LPS1 LED DISPLAY ROUTINE
6039 CALL LED1(DATA)

C CALL ROUTINE TO OUTPUT RESULT TO DATEL 256
6040 CALL TRAC(IEFH, DATA)

C ADJUST POINTERS FOR LPS1
C AT END OF HALF CYCLE FOR NEXT SAMPLE
6041 CALL LPS1(IEFH, 1)

C 
6042 IF
C 
6043 IF (IND1.GT.0) IND1=0
6044 IF (IND2(2).GE.1) GO TO 10
6045 IF (IND1(3).NE.0) GO TO 55
IV-PUS V2-51: 2C, prc-Fi, Tp; BLOCSAR

PAGE 3

0090 G0 TO 10
0142 95 C0 12
0348 C IN DICTION SYMES PRINT STATUS ON WAY OUT
0379 UMAC (1) (2) (3) (4) INST. IUES (INHD)
0393 200
0430 195
0451

PROGRAM SECTIONS

NAME SIZE ATTRIBUTES
MODEL 001004 356 RM-I, CMH-LCL
SPTR 000044 18 RM-B, CMH-LCL
SIPA 000242 75 RM-B, CMH-LCL
SHAP 000630 205 RM-B, CMH-LCL
STEPS 000002 1 RM-B, CMH-LCL

TOTAL SPACE ALLOCATED = 032124 554

INITES=INITES
DIMENSION IBUF(6), IRATE(2, 15), X(-30), Y(50)

THIS PROGRAM SYNTHESIZES AND FILTERS LINEAR PHASE FINITE IMPULSE RESPONSE FILTER WITH COEFFICIENTS GENERATED USING THE REMEZ EXCHANGE ALGORITHM.

SET ORDER OF FILTER (MUST BE EVEN AND EQUAL TO NUMBER OF H COEFFICIENTS)
N=32

INITIALIZE DATA MATRIX FOR FILTER
Y(1)=0.0
DO 299 I=1,N
299 CONTINUE

SET H COEFFICIENTS
H(1)=0.63214112-02
H(2)=0.91781938-03
H(3)=0.752533456-02
H(4)=0.67211197-02
H(5)=0.19163568-01
H(6)=0.72676892-02
H(7)=0.10362407-01
H(8)=0.71335292-02
H(9)=0.395537338-01
H(10)=0.1122611-00
H(11)=0.693306477-1
H(12)=0.10397291
H(13)=0.65131323
H(14)=0.12020875
H(15)=0.23870672
H(16)=0.30094172

INITIALIZE INVERSE PERIODICAL FILTER
CALL INSLN (1.196)

DESIGNATE OUTPUT DCOS CHANNEL FOR DATE 256
ICHAN=0

DESIGNATE REGISTER FOR LPST31 FILTER
IEREN=7

SPECIFY TIME BETWEEN SAMPLES : "WALLSECONDS"
IRATE(1)=100

PUT LPST31 RTS ROUTINE IN MULTITASK SAMPLING MODE
CALC.

CALL RT0 IN SECONDS

CALL E609, E60A, E60B

INDEX

CALL E3F, E3F, E3F

SIGN = 

SCALPE

Y (i) = 1 

BEGIN

DO 2, Y(i+1)

544 CONT

SUM = 0.3

DO 1 = 1

TEST = 

CONT

20 CONT

I = I + 1

CALL RTRE + DISPLAY ROUTINE

CALL RT0

CALL RT0 TO OUTPUT RESULT TO PAPER TM

CALL RT0 TO LOAD DATA

ADJUST = 

AND CALL = 

BUFFED FOR NEXT SAMPLE

CALL 

I = I + 1

INDEX = 

CHECK = 

IF REGISTER FOR PROPER I/C

AND TEST I/F STATUS NOT CORRECT

IF CALL = 

IF CALL = 

IF CALL =
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,IDBF(INDX)
0053       200       FORMAT(4112)
0054       195       CONTINUE
0055       END

PROGRAM SECTIONS

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIZE</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>$CODE1</td>
<td>001046 275</td>
<td>RW,I,CON,LCL</td>
</tr>
<tr>
<td>$DATA</td>
<td>000120  40</td>
<td>RW,D,CON,LCL</td>
</tr>
<tr>
<td>$IDATA</td>
<td>000060   24</td>
<td>RW,D,CON,LCL</td>
</tr>
<tr>
<td>$VARS</td>
<td>000630  204</td>
<td>RW,D,CON,LCL</td>
</tr>
<tr>
<td>$TEMPS</td>
<td>000002   1</td>
<td>RW,D,CON,LCL</td>
</tr>
</tbody>
</table>

TOTAL SPACE ALLOCATED = 002100  544
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.
DIMENSION IC0H(6), FTOE(2), ISB(2), H(20), Y(20), X(20), G(22)

C THIS PROGRAM SYNTHESIZES AND EXECUTES
C AN INFINITE IMPULSE RESPONSE FILTER WITH COEFFICIENTS
C GENERATED BY THE INIFIL PROGRAM. THE FILTERING LOOP
C IS FOR THE CASE WHERE THERE IS ONLY ONE NON-RECURSIVE
C WEIGHT AND FOUR RECURSIVE WEIGHTS.
C
C N IS THE NUMBER OF RECURSIVE WEIGHTS
N=8

C INITIALIZE DATA MATRIX FOR FILTER

DO 299 I=1,N
  299 CONTINUE

C SET H COEFFICIENTS
H(1) = 7.4813016
H(2) = 2.0704500
H(3) = 44.928890
H(4) = 52.639065
H(5) = 35.653219
H(6) = 19.747994
H(7) = 9.006574
H(8) = 0.53

C SET G COEFFICIENTS
G(0) = 0.001437531
G(1) = 0.0087154255
G(2) = 0.023574271
G(3) = 0.03814594
G(4) = 0.05816724
G(5) = G(7)
G(6) = G(3)
G(7) = G(1)
G(8) = G(0)

C INITIALIZE LABORATORY PERIPHERAL SYSTEM
CALL AC5LH (1,158)

C DESIGNATE OUTPUT DAC CHANNEL FOR DATEL 255
ICHM1=0
DESIGNATE REGISTER FOR LPS11 FLAG SET IEFHIF

READ IN SAMPLE PERIOD IN MILLISECONDS
WRITE (5,30)
30 FORMAT ('ENTER SAMPLE PERIOD IN SECONDS-INTEGER')
READ (5,31) IRATE(?)
31 FORMAT (16)

PUT LPS11 RTS ROUTINE IN SECONDS SAMPLING MODE
IRATE(1)=3

CALL RTS(IBUF,6,0,IPATE,IEFH,0,1,158)
INDEX=5
CALL WAITFR(IEFH)
CALL CLOSEF(IEFH)
SUM=IPATE INDEX

SCALE IN VOLTS
X(0)=10000.*((SIGMA/2043.)-10000.)

SUM=0.0
DO 20 I=1,2
20 CONTINUE

CONTINUE

SL

99 CONTINUE
SUM=SUM
Y(1)=SUMX
IPATE=SUMC.X(I)

WRITE(5,55) SUM,SUNE,Y(1),TERM.TURN

345 FORMAT (5F10.3)

SAME RECURSIVE TERMS
DO 544 I=0,N
555 Y(I+2)=Y(I+1)
556 X(I+1)=X(I)
557 544 CONTINUE
C
CALL LPS11 LED DISPLAY ROUTINE
CALL LPS1 CUT (IDATA)
C
CALL ROUTINE TO OUTPUT RESULT TO DASEL 256
CALL IDRAV (EXCH (RT))
C
ADJUST PARAMETERS FOR LPS11
C
AN 'CLEARING' BLOCK FOR NEXT SAMPLE
C
CALL ASH (RT)
C
C
CHECK THAT REGISTER FOR indeed I/O
C
WRITE OUT STATUS NOT CORRECT
3562 IF (IN = 'O') 
3563 IF (I$EC(1).NE.1) GOTO 93
3564 IF (I$CE(2).NE.0) GOTO 93
3565 G0 T0 10
3566 90 CONTINUE
C
IF PROGRAM CRASHES PRINT STATUS ON WAY OUT
3567 WRITE (5,200) I$EC(1), I$EC(2), IDATA, IBUF (INDX)
0*38 200 FORMAT (A112)
0*39 195 CONTINUE
0*39 END
C
PROGRAM SECTIONS
NAME SIZE ATTRIBUTES
C
SFORE1 001330 344 RW.I.DOH.LCL
SCDATA 00104 34 RW.D.DOH.LCL
SCDATA 00150 52 RW.D.DOH.LCL
WARS 000566 197 RW.D.DOH.LCL
C
TOTAL SPACE ALLOCATED = 002372 6ST
C
.CASE8=CASE8
DIMENSION BUF(6), FATE(2), ISA(2), H(20), Y(20), X(20), G(20)

THIS PROGRAM SYNTHESIZES AND EXECUTES AN INFINITE IMPULSE RESPONSE FILTER WITH COEFFICIENTS GENERATED BY THE INChef PROGRAM. THE FILTERING LOOP IS FOR THE CASE WHERE THERE IS ONLY ONE NON-RECURSIVE WEIGHT AND FOUR RECURSIVE WEIGHTS.

N IS THE NUMBER OF RECURSIVE WEIGHTS

N=8

INITIALIZE DATA MATRIX FOR FILTER

DO 299 I=1,N
  Y(I)=0.0
  X(I)=0.0
299 CONTINUE

SET H COEFFICIENTS

H(1)= 7.4013016
H(2)= -24.079098
H(3)= -44.869999
H(4)= -22.639065
H(5)= 33.645219
H(6)= 10.741094
H(7)= 5.0865574
H(8)= -60660115

SET G COEFFICIENTS

G(0)= .001495761
G(1)= .008111755
G(2)= .00755041
G(3)= .033814324
G(4)= .046919724
G(5)= G(3)
G(6)= G(2)
G(7)= G(1)
G(8)= G(0)

INITIALIZE LABORATORY PERIPHERAL SYSTEM

CALL ASL(W,158)

DESIGNATE OUTPUT DAC CHANNEL FOR DATEL 256

ICH=0
C
C DESIGNATE REGISTER FOR LPSII Flags Set
0026 IEFN=7

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 LPSII RTS ROUTINE IN MILLISECONDS SAMPLING MODE
0031 IRATE(1)=2

C
0032 50 CALL RTS(IBUF,6,0,IRATE,IEFN,0,1,ISB)
0033 INDX=5
0034 10 CALL WRITFRI(IEFN)
0035 15 CALL CLRLF(IEFN)
0036 SIGMA=IEFN(INDX)

C
C SCALE IN VOLTS
0037 X(0)=10000.0*(SIGMA/2048.0)-10000.

C

C 0330 SUM=0.0
0339 C DO 30 I=1,N
0340 TERM=H(I)*Y(I)

0341 SUM=SUM+TERM
0342 20 CONTINUE

0343 SUM=0.0
0344 MN=(N/2)-1
0345 DO 49 I=0,MN

0346 SUM=SUM+G(I)*H(I)*X(N-I)
0347 49 CONTINUE

0351 I_DATA=500.0*Y(I)

0352 WRITE (5,345) SUM,SUNE,Y(I),TERM,TURN
0353 345 FORMAT (SF10.3)

C
C SUM COLLECTIVE TERMS
C   CALL LPS11 LED DISPLAY ROUTINE
C   CALL LED(IDATA)
C   CALL ROUTINE TO OUTPUT RESULT TO DATEL 256
C   CALL IDAC(ICHAN,IDATA)
C   ADJUST POINTERS FOR LPS11
C   AND CLEAR HALF BUFFER FOR NEXT SAMPLE
C   CALL ADJLPS(IBUF,1)
C
C   INDEX=INDEX+1
C   CHECK STATUS REGISTER FOR PROPER I/O
C   AND TERMINATE IF STATUS NOT CORRECT
C   IF (INDEX.GT.6) INDEX = 0
C   IF (ISB(2).GE.1) GO TO 10
C   IF (ISB(1).NE.0) GO TO 95
C   GO TO 10
C   CONTINUE
C   IF PROGRAM CRASHES PRINT STATUS ON WAY OUT
C   WRITE (5,200) VAR(1), IDAC(IDATA), IBUF(INDEX)
C   WRITE (200) 'STATUS=112'
C   CONTINUE
C   END

PROGRAM SECTION

NAME     SIZE     ATTRIBUTES
SCODE1   001336   264     RW.L.COM.LCL
SFILE1   000104   34      RW.D.COM.LCL
SIN1     000154   54      RW.D.COM.LCL
SVAR1    000566   161     RW.D.COM.LCL

TOTAL SPACE ALLOCATED = 002376   639
DIMENSION M(6),IRATE(2),ISS(2),H(30),Y(60)

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.

N IS THE NUMBER OF RECURSIVE WEIGHTS
N=5

SUBROUTINE FILTER
    DO 999 I=1,4
    Y(I)=0.0
END

C SET H COEFFICIENTS
HA=0.5247
HB=-0.6052
HC=-0.171

C SET G COEFFICIENT
G=0.5227

C INITIALIZE LABORATORY PERIPHERAL SYSTEM
CALL ASLSEL (1,153)

C DESIGNATE OUTPUT DRO CHANNEL FOR DATEL 256
ICHAN=0

C DESIGNATE REGISTER FOR LPS11 FLAG SET
IFEN=7

C READ IN SAMPLE PERIOD IN MILISECONDS
WRITE (5,30)
C FORMAT('SAMPLE PERIOD IN MILLISECONS-INTEGER ')
READ (5,31) IRATE(2)
C FORMAT (16)

C 'PUT LPS11 RTS ROUTINE IN SECONDS SAMPLING MODE
IRATE(1)=2
0020 90 CALL RTS(1,BUF,6.0,IRATE,IEFI,E,1,ISB)
0021 - INDEX=5
0022 10 CALL WIARF(IEFI)
0023 15 CALL CLOEF(IEFI)
0024 SCALE IN VOLTS
0025 X=10600.*(SIGMA/2048.)-1030.
0026 SUM=0.0
0027 DO 20 I=1,N
0028 TERN=H(I)*Y(I)
0029 SUM=SUM+TERM
0030 CONTINUE
0031 Y(I)=SUM+H(I)
0032 DATA=Y(I)
0033 CONSERVE RECURSIVE TERMS
0034 DO 54 I=1,I+1
0035 Y(I+1)=Y(I)
0036 CONTINUE
0037 CALL LED DISPLAY ROUTINE
0038 CALL LED(CAP))
0039 CALL ROUTINE TO OUTPUT RESULT TO PATEL 256
0040 CALL IMPS(char,DATA)
0041 ADJUST PRINT FOR LED11
0042 AND CLEAR HALF BUFFER FOR NEXT SAMPLE
0043 CALL ADJLPS(IEFI,1)
0044 INDEX=INDEX+1
0045 CHECK STATUS REGISTER FOR PROPER I/O
0046 AND TERMINATE IF STATUS NOT CORRECT
0047 IF (INDEX.EQ.0) GO TO 45
0048 IF (ISB.EQ.1) GO TO 10
0049 IF (ISB(1).NE.0) GO TO 95
0050 GO TO 10
0051 CONTINUE
C C IF PROGRAM CRASHES PRINT STATUS ON WAY OUT
0345 WRITE (5, 260) ISB(I), ITB(J), IDATA, IEUF(INDX)
3046 200 FORMAT (4112)
3047 195 CONTINUE
3048 - END

PROGRAM SECTIONS

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIZE</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODE1</td>
<td>000740</td>
<td>249 RW, I, CON, LCL</td>
</tr>
<tr>
<td>DATA</td>
<td>000050</td>
<td>20 RW, D, CON, LCL</td>
</tr>
<tr>
<td>IDATA</td>
<td>000145</td>
<td>51 RW, I, CON, LCL</td>
</tr>
<tr>
<td>SVARS</td>
<td>000634</td>
<td>206 RW, I, CON, LCL</td>
</tr>
<tr>
<td>STEMEP</td>
<td>000032</td>
<td>1 RW, D, CON, LCL</td>
</tr>
</tbody>
</table>

TOTAL SPACE ALLOCATED = 082014 516

.HONEYF=HONEYF
DIMENSION BUF(6), IPATE(2), ISB(2), H(30), Y(60)

THIS PROGRAM SYNTHESIZES AND EXECUTES AN INFINITE IMPULSE RESPONSE FILTER WITH COEFFICIENTS GENERATED BY THE MACFIL PROGRAM. THE FILTERING LOOP IS FOR THE CASE WHERE THERE IS ONLY ONE NON-RECURSIVE WEIGHT AND FOUR RECURSIVE WEIGHTS.

N IS THE NUMBER OF RECURSIVE WEIGHTS

N=5

INITIALIZE DATA MATRIX FOR FILTER

DO 299 I=1,N
Y(I)=0.0
299 CONTINUE

SET H COEFFICIENTS

H(1)=.92476531
H(2)=-.68528953
H(3)=.30471970
H(4)=.27438119
H(5)=.05302164

SET G COEFFICIENT

G=.52299771

INITIALIZE INPUT PERIPHERAL SYSTEM

CALL AIC55H (.155)

DESIGNATE OUTPUT DAC CHANNEL FOR DATA. 255
ICHAN=0

DESIGNATE REGISTER FOR LPSII FLAG SET
IEFH=7

SPECIFY TIME BETWEEN SAMPLES IN MILLISECONDS
IRATE(2)=1

PUT LPSII PTS ROUTINE IN MILLISECOND SAMPLING MODE
IRATE(1)=3

CALCULATE SAMPLE INTERVAL IN SECONDS
ST=FLOAT(IRATE(2))/1000.
C
C
0016 90 CALL RTS(I2,F.6.0,IRATE,IEFN,.I,ISB)
0019 INDEX=5
0020 10 CALL U1TFP(IEFN)
0021 15 CALL CLRFP(IEFN)
0022 SIGMA=IBUF(INDEX)
C
C SCALE IN VOLTS
0023 X=10000.*(SIGMA/2048.1)-10000.
C
C
0024 SUM=0.0
0025 DC 20 I=1,N
0026 TERM=H(I)*Y(I)
0027 SUM=SUM+TERM
0028 20 CONTINUE
0029 Y(I)=TERM+Y(I)
0030 IDATA=Y(I)
C
C SAVE REST OF 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 IRLAC(IEFN,IDATA)
C
C ADJUST POINTERS FOR LPS11
C AND CLEAR HALF BUFFER FOR NEXT SAMPLE
0036 CALL ADJLPS(IEFN,IBUF.1)
C
C
0037 INDEX=INDEX+1
C
C CHECK STATUS REGISTER FOR PROPER I/O
C AND TERMINATE IF STATUS NOT CORRECT
0038 IF (INDEX.GT.6) INDEX=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
```plaintext
PROGRAM SECTIONS

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIZE</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODE1</td>
<td>000716</td>
<td>RW, I, CON, LCL</td>
</tr>
<tr>
<td>PDATA</td>
<td>000850</td>
<td>RW, D, CON, LCL</td>
</tr>
<tr>
<td>GDATA</td>
<td>000960</td>
<td>RW, D, CON, LCL</td>
</tr>
<tr>
<td>SVMPS</td>
<td>000640</td>
<td>RW, D, CON, LCL</td>
</tr>
<tr>
<td>STEMPS</td>
<td>000002</td>
<td>RW, D, CON, LCL</td>
</tr>
</tbody>
</table>

TOTAL SPACE ALLOCATED = 001710  484

JATTER=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
MOV #20,#160010 ; PUT IN RANDOM MODE
MOV 0(R5)+,0#160012 ; SPECIFY CHANNEL 1
MOV 0(R5)+,0#160C:4 ; 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 LPS11 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.