This is to certify that the

thesis entitled

Precision Automated Measurements

presented by

Robert Jay Randel

has been accepted towards fulfillment
of the requirements for

Master's degree in Electrical Engineering

Major professor

Date Dec 17, 1990

PRECISION AUTOMATED MEASUREMENTS

By

Robert Jay Randel

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Electrical Engineering

1990

ABSTRACT

PRECISION AUTOMATED MEASUREMENTS

By

Robert Jay Randel

Due to their IEEE-488 GPIB capabilities, the latest series of Fluke/Philips instruments can be programmed to perform a wide variety of automated measurements. The TestTeam software package supplied by Philips contains drivers which make this programming directly possible from Microsoft QuickBASIC. Techniques and software are developed using these drivers to automate tedious steady-state measurements. The topics covered are bode magnitude and phase plots, locating cutoff (-3 dB) frequencies, measuring gain-bandwidth-products of op-amps, and finding the second order filter parameters f_c, H_0, and Q_0. This software is tested on numerous circuits.

ACKNOWLEDGEMENTS

My appreciation to Ken Noren for his support and advice.

Much thanks to Jim MacKay for his assistance and support.

My infinite gratitude to Professors G.M. Wierzba and C.R. MacCluer for their guidance and assistance in completing this document.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION TO TESTTEAM SOFTWARE

1.1 Introduction and Preview of Results

One tedious task a modern electrical engineer faces is the testing of a circuit. Despite advances in computer modelling and simulation, there is no substitute for building a circuit. Often problems are discovered that did not show up in the simulation. Many of the tests that are run on these circuits are very tedious measurements that can take even the most skilled of engineers many hours. Until recently automation of these measurements was not possible. However many newer test instruments now include some kind of computer interface. When properly connected to a computer, the computer can control the test equipment and perform the measurements for the engineer. Of course the computer and the test equipment both need to be programmed. One such set of instruments and software is the newest line of Fluke/Philips instruments and the Philips TestTeam software package.

TestTeam can be an extremely powerful tool to aid in the automation of circuit measurements. To unleash this power, however, it is necessary to program the package. The package is composed of a programming environment (called LabWindows) and a complete set of drivers which support many Fluke/Philips instruments. When these drivers are combined with some programming, they allow the experimenter to automate measurements. The documentation that accompanies this package can be described as sparse at best. It consists of descriptions of the drivers and an introduction to LabWindows, but contains no useful examples or techniques. The goal of this document is to indicate an approach to the development of software to automate precision measurements using these drivers. In Chapter 2, sample software is designed for the automated measurements of Bode magnitude and phase plots and cutoff frequencies of passive networks. This software is tested on low and high-pass filters. In Chapter 3, the software is expanded to include active circuits and perform such measurements as gain-bandwidth-products of op-amps and the second order filter parameters f_c, H_0, and Q_0. In Chapter 4, the software is utilized to perform measurements to analyze a high-Q band-pass filter. Complete listings of the developed programs are provided to assist the user in further development. The examples herein will enable a user to fully exploit the powerful device drivers included in TestTeam.

1.2 Choosing a Programming Language

The TestTeam drivers and the programming environment are compatible with two programming languages, Microsoft QuickBASIC and Microsoft C. This study will use Microsoft QuickBASIC. Although not a widely used language in the academic community, BASIC is widely used in industry. BASIC achieved great popularity in the late 1970's because it was included in the ROM of virtually every personal computer of that time. These older versions of BASIC are best suited to simple tasks. Critics complained that it is difficult or impossible to write structured, easy-to-follow code since there were no "while-until" or "repeat-until" commands nor were there subroutines with parameters. Also, since it is an interpreted language (as opposed to compiled) it could be too slow for many problems. Microsoft QuickBASIC has obviated all of these objections. It is a free-form language (requiring no line numbers) with a compiler, supporting subroutines and functions similar to Pascal, and has numeric coprocessor support. It would appear to be an ideal choice for a study whose goal is to make TestTeam more accessible to industry users.

1.3 Hardware Requirements

The TestTeam software requires an IBM PC/XT/AT, IBM PS/2, or compatible computer with one floppy disk drive, one hard disk drive, 640 KB of RAM, and the Philips PM 2201 IEEE-488 GPIB Bus Interface. For graphical output, a graphics card and monitor are also required. This study employed a 6 MHz Zenith 80286-based computer with monochrome CGA and an 80287 coprocessor.

1.4 Installation and Configuration of TestTeam Software

One should begin by making backup copies of the TestTeam distribution disks. To install TestTeam onto a hard drive, insert the first diskette, make floppy drive A the current drive and type

```
A>setup
```

and press the Enter key. The program now prompts the user to insert all of the TestTeam disks for installation. The directories into which the setup program will place the files are shown in Table 1.1 [1].

Table 1.1 - Directories of TestTeam Files

Directory Name	Contents
\LW	System files
\LW\FONTS	Font files required for graphics operations
\LW\LIBRARY	Library files for linking with standalone programs
\LW\INCLUDE	Include files associated with libraries
\LW\PROGRAMS	Source code to sample programs
\LW\INSTR	Instrument modules

The setup program also performs some necessary changes to the CONFIG.SYS file in the root directory of the hard disk. These include adding the lines FILES=20 and DEVICE=\LW\GPIB.COM. The former allows a maximum of 20 files to be open simultaneously, while the latter allows the system to be aware of the IEEE-488 port.

Next one must configure the TestTeam software to the instruments. The instruments used in this study are the Philips PM 3365 Oscilloscope, PM 5183 Programmable Synthesizer/Function Generator, PM 6666 Programmable Timer/Counter, and the Fluke 8840A Digital Multimeter. Before we can connect the instruments to the computer, their IEEE address must be set. This is done by using the front panel keys (PM 5193, PM 3365) or by setting DIP switches (PM 6666, Fluke 8840A). Any address can be chosen as long as there are no conflicts. Using standard IEEE-488 cables, connect each of the four instruments to the computer's interface. A typical way to do this is to "daisy-chain" the instruments to the computer; this makes the detection of a bad cable easier should that problem arise. Once this is complete turn on the instruments (making sure there are no front panel connections present) and type the following commands at the DOS prompt:

```
c:
cd\lw
config/a
```

The last command selects the configuration program, and the /a stands for automatic. The automatic configuration seems to work well. It recognizes the instruments attached to the GPIB bus and configure itself accordingly.

1.5 Analysis of the Configuration Program

One of the files created by the configuration program is called TESTTEAM.BAS. A sample listing is shown below:

```
REM File : TESTTEAM.BAS
REM Setup file for Philips/Fluke Instrument drivers
REM This will assign logical names to the instruments
REM and initialize the instruments

REM Following include statements can be removed in Interactive
window

REM     $INCLUDE: '\lw\instr\generato.inc'
REM     $INCLUDE: '\lw\instr\counter.inc'
REM     $INCLUDE: '\lw\instr\multimet.inc'
REM     $INCLUDE: '\lw\instr\scope.inc'
REM     $INCLUDE: '\lw\instr\general.inc'

DEFINT A-Z

COMMON SHARED /DMM1/ DMM1 AS INTEGER
COMMON SHARED /GNR1/ GNR1 AS INTEGER
COMMON SHARED /OSC1/ OSC1 AS INTEGER
COMMON SHARED /CNT1/ CNT1 AS INTEGER

CALL res.glb
CALL reset.config
CALL config(DMM1,"8840A,A 706,N DMM1")
CALL config(GNR1,"PM5193/V2.5,A 707,N GNR1")
CALL config(OSC1,"PM3365/V07V04,A 708,N OSC1")
CALL config(CNT1,"PM6666/22,A 710,N CNT1")
CALL allinit(DEFAULT.SET)
IF glb.stat > 1 THEN
         PRINT "Error: ",glb.str   :REM Print global error string
         call res.glb              :REM Reset global error status
ELSE
IF glb.stat = 1 THEN
         PRINT "Warning: ",glb.str :REM Print global warning string
         call res.glb              :REM Reset global error status
END IF
END IF
```

This file serves as the basis for all applications developed. Its purpose is to initialize the GPIB bus and the instruments so they will be ready for the commands.

1.6 Programming the Instruments

Within the QuickBASIC environment, the instruments are given integer variable names consistent with their function. For example the 8840A is given the variable name `dmm1%` (which stands for Digital Multi-Meter). If more than one of a given type of instrument is connected, the last digit will increase, e.g. `dmm2%`. A summary of these variable names is given in Table 1.2.

Table 1.2 - Variable Names for Instruments

Instrument	Sample Variable Name
Digital Multimeter	dmm1%
Function Generator	gnr1%
Timer/Counter	cnt1%
Oscilloscope	osc1%

1.7 Using the TestTeam Drivers

The real power of the TestTeam package lies in its driver functions. These drivers, when properly utilized, can be used to configure the instruments, perform measurements, and display graphs. The QuickBASIC syntax for these functions is

```
CALL function.name (...)
```

where the ellipses stand for whatever parameters are required for the function. A summary of these functions is provided in Table 1.3. [2]

Table 1.3 - Summary of Important TestTeam Functions

Function Name	Purpose
set.amplitude	Sets output amplitude of function generator
set.function	Sets function of instrument (e.g. VOLTS-DC or VOLTS-AC for multimeter)
set.speed	Sets speed of instrument (note that speed is inversely proportional to significant figures)
set.coupling	Allows AC or DC coupling to be selected
set.sensitivity	Sets sensitivity of counter
measure	Triggers instrument and performs measurement
grfreset	Clears the graphics display
setxdatatype	Sets data type for x-axis (integer, real, etc.)
setydatatype	Sets data type for y-axis
grfcurv2d	Creates x-y plot
grflreset	Resets graphics library
iolocal	Returns instrument to local control
setaxname	Labels x or y axis

1.8 Summary

In this chapter the goals of the document have been outlined. An introduction to using TestTeam software was given, and Microsoft QuickBASIC was chosen as the programming language to use. Finally the TestTeam software was installed, preparing us for what lies ahead.

CHAPTER 2

STEADY STATE ANALYSIS OF PASSIVE NETWORKS

2.1 Introduction

In this chapter routines are designed to employ TestTeam for steady state passive circuit analysis. In particular these routines automate common tedious measurements such as Bode magnitude and phase plots, and to automatically locate a -3 dB point.

2.2 Creating a Bode Magnitude Plot of a Passive Circuit

To use TestTeam to perform a measurement, one starts by envisioning how to perform that measurement manually, and then translate those steps into a program. To make a bode magnitude plot by hand, one would:

1. Connect the circuit to the function generator and multimeter.
2. Set the function generator to the beginning frequency.
3. Set the output of the function generator to 1 V rms.
4. Set the multimeter to read rms ac voltage.
5. Wait at least 5 τ for the circuit to settle.
6. Measure the output on the voltmeter.
7. Convert the output to dB.
8. Increase the frequency.
9. If not done, go to 5.

It would also be wise to choose the frequencies logarithmically if we intend to make a semilog plot. Translating these steps into QuickBASIC:

```
CALL set.amplitude (GNR1%, VRMS%, 1.0)      'set generator to 1V RMS
CALL set.function (DMM1%, VOLT.AC%)         'set DMM to read RMS AC V

ppd=5                                       'points per decade
strt!=1.5                                   'log of starting frequency
stp!=4.0                                    'log of stopping frequency

for i!=strt! to stp! step 1/ppd
  p=(i!-strt!)*ppd                          'offset in array
  x#(p)=10^i!                               'actual frequency
  CALL set.frequency (GNR1%, x#(p))         'set freq of generator
  CALL measure (DMM1%, y#(p))               'take voltage measurement
  y#(p)=20*log(y#(p))/log(10)               'convert to dB
  print x#(p),y#(p)                         'print frequency and dB
  x#(p)=i!                                  'store log of frequency
next i!

CALL GrfReset (4)                           'set up graphics screen
np=(stp!-strt!)*ppd                         'number of points read
CALL GrfCurv2D (x#(), y#(), np)             'do x-y plot/wait for key
CALL GrfLReset (0, 0, 1, 2)                 'return to text mode
```

This program may be run directly from the LabWindows environment, provided it is added to the configuration program, TESTTEAM.BAS. To enter the LabWindows program, type the following commands at the DOS prompt:

```
cd\lw

lw
```

This program is relatively straightforward in its operation. The first two lines set the function generator to 1 V rms and the multimeter to AC voltage, respectively. The next three set the number of points to take per decade and the starting and stopping frequencies. After this comes the loop which takes the frequencies. Since the FOR-NEXT loop chooses points linearly, if we take the antilog of each point, we will have the actual frequency. This is done in the line `x#(p)=10^i`. Next the frequency of the function generator is set and the measurement is taken. Note that the internal settling time of the multimeter (551 ms in VAC mode) [3] is sufficient for many circuits to settle, thus no delay is included. Then the measurement is converted to dB (log in QuickBASIC is to the base e, thus we must divide by log(10) to convert to base 10) and the result is displayed. The log of the frequency is stored so we can plot dB vs. log(frequency). Finally the data is plotted using the graphics library drivers.

A test run of this program was done using the first-order Butterworth low-pass filter shown in Figure 2.1. The actual values of the components (as measured by the HP 4284A Precision LCR Meter) are R = 1.0169 kΩ for the resistor, and R_S = 7.6 Ω and C_S = 100.11 nF (measured at 4 kHz) for the capacitor. The generated plot appears in Figure 2.2. This plot appears to be what would be expected from such a circuit.

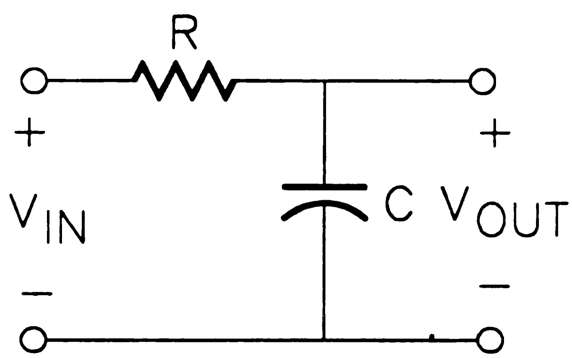

Figure 2.1 - Low-Pass Filter

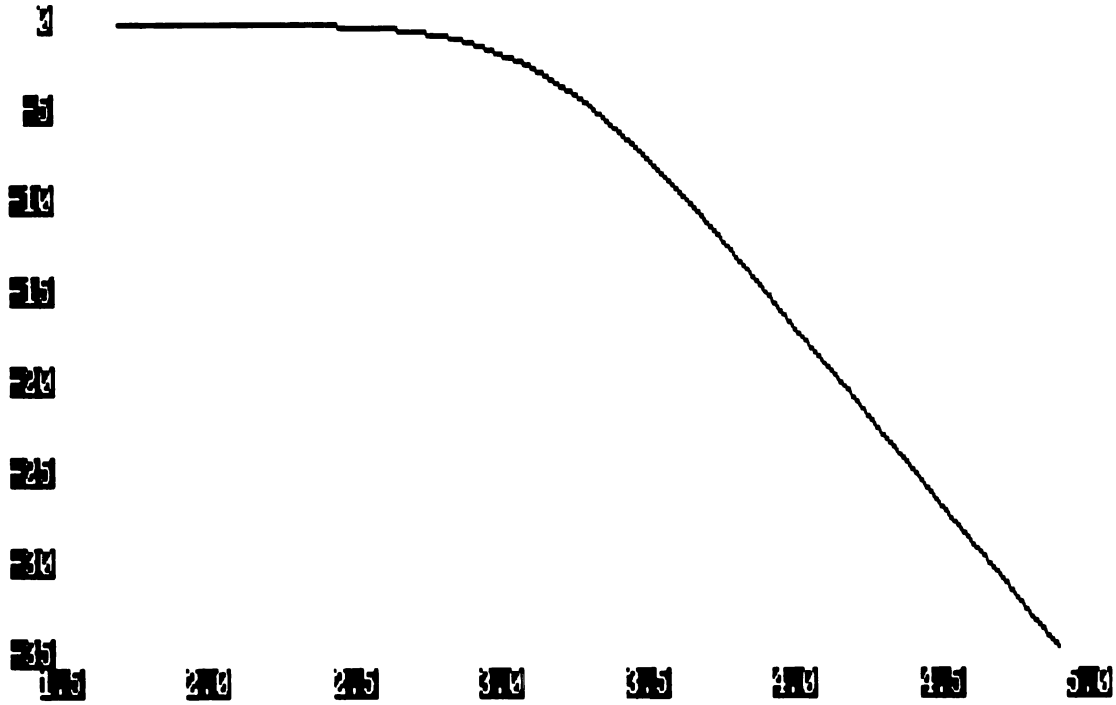

Figure 2.2 - Bode Magnitude Plot of Low-Pass Filter

2.3 Speeding Up Operation of This and Other Routines

The LabWindows interactive environment, an interpreted subset of QuickBASIC, is too slow by modern standards. Each measurement and calculation takes several seconds. For professional lab measurements, one must employ the full Microsoft QuickBASIC 4.5 compiler. An additional driver package is then required: the PM2233 Instrument Drivers. This package allows the interface of compiled QuickBASIC with the necessary drivers to control the instruments. The installation of these drivers is identical to the installation of the TestTeam software. It will place its files in the directory \DRIVERS. The only necessary changes to the program are in the `INCLUDE` statements at the beginning of the program. These changes can be seen in the listing of the program BODE3C.BAS in the next section.

It is possible to program directly in the QuickBASIC environment using these libraries, however it is limited to small programs. Thus the QuickBASIC editor was used to type the programs, and then they were compiled and linked from DOS. To automate this procedure, the batch file DOIT.BAT was created. Its listing is shown below.

```
bc/o %1;
\qb45\link %1,,@linkme.lnk
```

The command `bc` invokes the QuickBASIC compiler, and the `/o` parameter tells it to compile stand-alone, i.e. without a run-time module. The `%1` represents the parameter passed to the batch file. Finally the @linkme.lnk for the linker tells it to get its input from the file linkme.lnk. The contents of this file are shown below.

```
NUL.MAP/NOE/NOD/SEGMENTS:1000/STACK:10000,
\LW\LIBRARY\formatio+
\LW\LIBRARY\graphics+
\LW\LIBRARY\gpib_qb+\LW\LIBRARY\lwqb1+\qb45\bcom45+
\LW\LIBRARY\ttdrivqb+\DRIVERS\drivers+\LW\LIBRARY\lwqb2;
```

The first line tells the linker not to create a cross-reference file and to set aside 10 KB for the stack. The other lines tell the linker where to find the necessary libraries. For example, to compile and link the program BODE3C.BAS, type at the DOS prompt:

`doit bode3c`

and press ENTER. After the process is complete, to run the program simply type **`bode3c`** and press ENTER.

2.4 Creating a Bode Phase Plot

A method of measuring phase angle is via an oscilloscope by finding a common point on the two waveforms and measuring the time delay between them. The ratio of this time delay to the period of the waveform is then proportional to the phase angle. Symbolically this is

$$\frac{\Delta T}{T}=\frac{\theta}{360^{\circ}}.$$

The problem is obtaining an accurate measure of ΔT. Fortunately there is a simple solution. The PM6666 counter has a measurement mode called "TIME A-B". This mode gives the time delay between a positive slope on channel A and a positive slope on channel B [4]. Since the frequency (and thus the period), is known, the calculation of phase follows. The only other consideration is the coupling of the counter. For low frequencies, below about 500 Hz., the counter should be on DC coupling to avoid errors. Above 500 Hz. the counter should be set for AC coupling. The QuickBASIC implementation of this is shown below.

```
REM File : BODE3C.BAS

REM     $INCLUDE: '\lw\instr\generato.inc'
REM     $INCLUDE: '\lw\instr\counter.inc'
REM     $INCLUDE: '\lw\instr\multimet.inc'
REM     $INCLUDE: '\lw\instr\scope.inc'
REM     $INCLUDE: '\lw\instr\general.inc'
REM     $INCLUDE: '\lw\include\phildecl.inc'
REM     $INCLUDE: '\lw\include\graphics.inc'

DEFINT A-Z

COMMON SHARED /dmm1/ dmm1 AS INTEGER
COMMON SHARED /gnrl/ gnrl AS INTEGER
COMMON SHARED /osc1/ osc1 AS INTEGER
```

```
COMMON SHARED /cntl/ cntl AS INTEGER

CLEAR , , 2048

DIM SHARED x#(100), y#(100), ph#(100)
CALL getmem(20000)
overlay.memory.size& = 98304
memory.size& = SETMEM(640000)
memory.size& = SETMEM(-overlay.memory.size&)

CLS

CALL res.glb
CALL reset.config
PRINT "Initializing Multimeter..."
CALL config(dmm1, "8840A,A 706,N DMM1")
PRINT "Initializing Function Generator..."
CALL config(gnr1, "PM5193/V2.5,A 707,N GNR1")
PRINT "Initializing Oscilloscope..."
CALL config(osc1, "PM3365/V07V04,A 708,N OSC1")
PRINT "Initializing Counter..."
CALL config(cntl, "PM6666/22,A 710,N CNT1")
PRINT "Setting up defaults..."
CALL allinit(DEFAULT.SET)       'or ACTUAL.SET to leave unchanged
IF Glb.Stat > 1 THEN
      PRINT "Error: ", Glb.Str: REM Print global error string
      CALL res.glb: REM Reset global error status
      ELSE
IF Glb.Stat = 1 THEN
      PRINT "Warning: ", Glb.Str: REM Print global warning string
      CALL res.glb: REM Reset global error status
END IF
END IF

PRINT

StartOfProg:

CALL set.amplitude(gnr1%, VRMS%, 1!)
CALL set.function(dmm1%, VOLT.AC%)
CALL set.speed(dmm1, low)
CALL set.function(cntl%, TIMEINTERVAL.A.B)
CALL set.coupling(cntl%, chall%, dc%)

CALL set.sensitivity(cntl%, cha%, .02)
CALL set.sensitivity(cntl%, chb%, .02)

INPUT "Enter starting frequency ->", strt!
INPUT "Enter ending frequency ->", stp!
INPUT "Enter number of points per decade ->", ppd
PRINT
PRINT "Frequency       dB       Angle"
PRINT

strt! = LOG(strt!) / LOG(10)
stp! = LOG(stp!) / LOG(10)

FOR i! = strt! TO stp! STEP 1 / ppd
  p = (i! - strt!) * ppd
  x#(p) = 10 ^ i!
  CALL set.frequency(gnr1%, x#(p))
  CALL measure(dmm1%, y#(p))
```

```
  IF x#(p) > 500 THEN                'set AC coupling above 500 Hz
    CALL set.coupling(cntl%, chall%, ac%)
  ELSE
    CALL set.coupling(cntl%, chall%, dc%)
  END IF
  CALL measure(cntl%, diff#)

  ph#(p) = -diff# * x#(p) * 360#
  IF ph#(p) < -180 THEN
    ph#(p) = ph#(p) + 360#
  END IF

  y#(p) = 20 * LOG(y#(p)) / LOG(10)     'convert to dB
  LOCATE CSRLIN - 1
  PRINT USING "##.###^^^^   +###.###   +###.###"; x#(p); y#(p);
ph#(p)
  x#(p) = i!
NEXT i!

CALL GrfReset(4)
np = (stp! - strt!) * ppd              'number of points
CALL SetXDataType(4)
CALL SetYDataType(4)
CALL GrfCurv2D(x#(), y#(), np)         'x-y plot of dB vs. log(f)
CALL GrfReset(4)                       'clear the graphics display
CALL GrfCurv2D(x#(), ph#(), np)        'x-y plot of phase vs.log(f)
CALL GrfLReset(0, 0, 1, 2)             'return to text mode

CLS
PRINT "Again (y/n)?";
a$ = INPUT$(1)
PRINT : PRINT
IF UCASE$(a$) = "Y" THEN GOTO StartOfProg

REM   Now return to local control

CALL iolocal(dmm1%)
CALL iolocal(osc1%)
CALL iolocal(cntl%)
CALL iolocal(gnrl%)


END

REM $DYNAMIC
SUB ReportError
END SUB
```

The output of this program for a the high-pass R-C circuit in Figure 2.3 is shown in Figures 2.4 and 2.5. The values of the components used were C_s = 105.48 nF, R_s = 11.9 Ω (measured at 400 Hz), and R = 4.028 kΩ. The frequency range was from 20 Hz to 50 kHz, taking 9 points per decade.

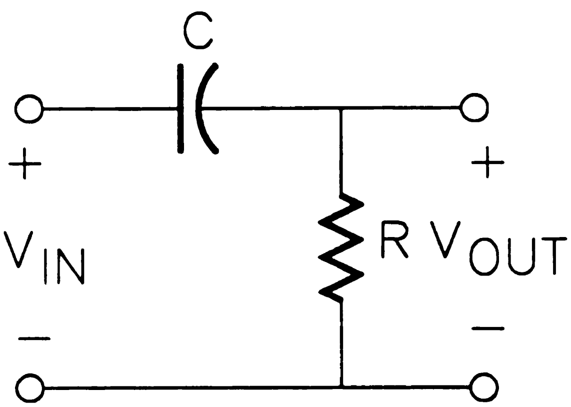

Figure 2.3 - High-Pass Filter

Figure 2.4 - Bode Magnitude Plot of High Pass Filter

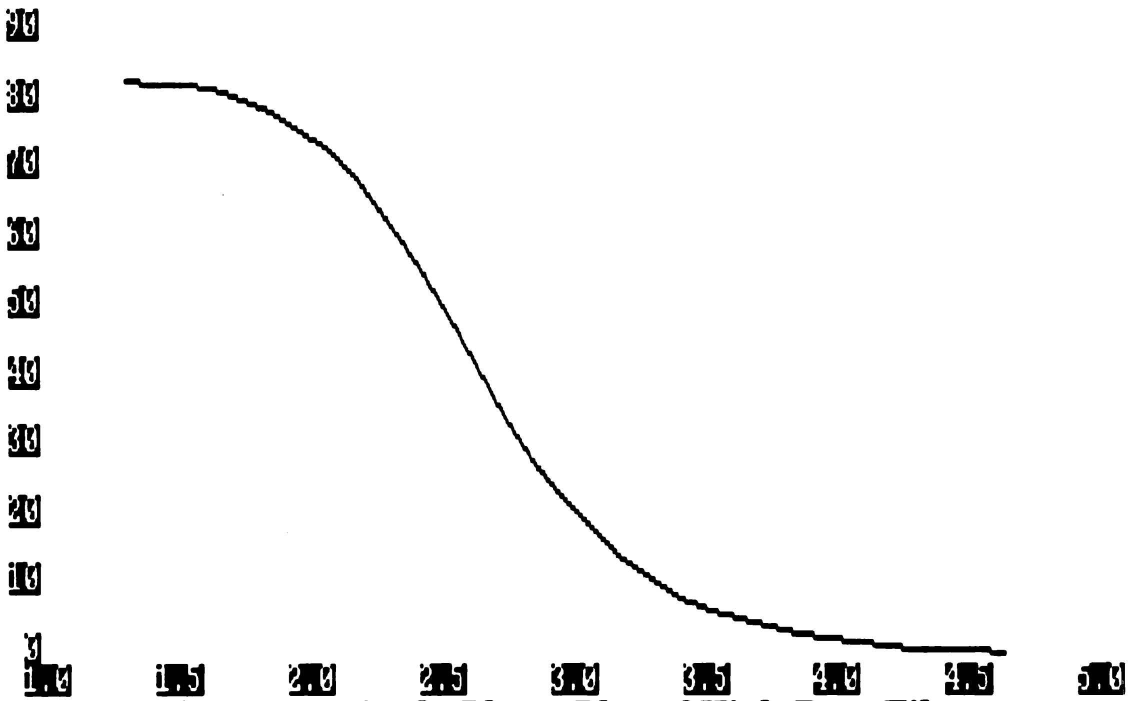

Figure 2.5 - Bode Phase Plot of High Pass Filter

2.5 Defects and Potential Improvements

The program BODE3C.BAS discussed above suffers from several defects. When measuring a low-pass filter, as the output voltage falls below a certain level, the counter can no longer lock onto the signal and erroneous data is taken. Also no access is given to the raw data in case an error is suspected. In addition the counter may occasionally give a time delay that is off by a period or two, e.g. yielding a phase angle of -362° as opposed to the correct value of -2°. Still another problem is that the output of the function generator may not be exactly 1 V RMS. It would also be useful to label the axes of the graph. Another desirable feature would be to implement a routine that would locate a -3 dB point.

2.6 Implementation of Improvements

2.6.1 Access to Raw Data

Providing the user with access to the raw data accumulated would be a convenient feature. This way the user could verify the data by hand if an error is suspected, or even use this data in another program. It would be best to supply all the calculated quantities as well as the measured values. The values to be stored in the file are: frequency, dB, phase, measured voltage, and measured time delay.

2.6.2 Correcting for Input Voltage Errors

When the PM5193 Function Generator is set for 1 V RMS output, the open-circuit output is not exactly 1 V RMS. This effect does depend upon frequency somewhat, although if we were to assume a constant error over a range of frequencies (which may or may not be accurate depending upon the range), we could divide all our voltage measurements by a correlation to approximate the actual transfer function. The optimum solution, of course, would be to use a second voltmeter, then the transfer function would simply be output divided by input. The former shall be implemented for the time being, assuming that access to a second voltmeter is not possible. The implementation of the second voltmeter is addressed in Section 3.2.1.

2.6.3 Disabling the Counter to Prevent Erroneous Data

When the output voltage falls below 10 mV RMS, the counter becomes unable to lock onto the signal even when it is set for maximum sensitivity. To solve this problem there are several options: (1) discard phase measurements when the output falls below the 10 mV threshold (although some value must be assigned to the array), or (2) we could stop taking phase measurements below the threshold. The latter shall be implemented.

2.6.4 Correction of Phase Angle

Occasionally the counter gives a time delay that is off by a period or two. This would result in a phase angle of say -722° or -362° instead of -2° (it always appears to be off in the negative direction). The simplest method to correct this is to add 360° to the phase angle as long as it is less that -180°. Translated to QuickBASIC:

```
while ph#(p) < -180
  ph#(p) = ph#(p) + 360#
wend
```

2.6.5 Finding a -3 dB Point

Finding a -3 dB point is a common required measurement. It may represent the bandwidth or a cutoff frequency of a circuit. By hand it is a tedious procedure, which makes it a particularly desirable feature to implement by machine. The method chosen here is the linear interpolation method. This method assumes a Bode plot has already been performed and that the data from this plot is stored in an array. The steps followed are detailed below.

1. Find two points whose magnitudes are "close" to $\sqrt{2}/2 \approx 0.70711$. If there are none, inform the user to re-run the bode plot to include such points.

2. Use the linear interpolation formula (from basic calculus)

$$f(x+\Delta x) \approx f(x) + f'(x)\Delta x$$

where f(x), f(x+Δx), f'(x), and x are known and Δx is the only unknown. ($f(x+\Delta x) = \sqrt{2}/2$ and f'(x) can be approximated by using the two points found in step 1). That is, if the two known frequencies are f_1 and f_2, and the two magnitudes are t_1 and t_2, respectively, then

$$f'(x) \approx \frac{t_2 - t_1}{f_2 - f_1}$$

and the target frequency is

$$f_0 = \frac{\frac{\sqrt{2}}{2} - t_1}{f'(x)} + f_1.$$

3. Set the target frequency to the value found in step 2. Read the output voltage from the voltmeter. If the voltage does not equal 0.70711 V (the 8840A displays 5 decimal points) then use one of

the original points and point found in step 2 and return to step 2.

Translating steps 2 and 3 into QuickBASIC:

```
DO
  slope# = (t2# - t1#) / (f2# - f1#)
  f0# = (0.70711 - t1#) / slope# + f1#
  CALL set.frequency(gnrl%, f0#)
  CALL measure (dmml%, t0#)
  IF (f2# - f0#) < (f0# - f1#) THEN
    f1# = f0#
    t1# = t0#
  ELSE
    f2# = f0#
    t2# = t0#
  END IF
LOOP UNTIL t0# = 0.70711
```

In practice this routine seems to require an average of 6 or 7 tries before it finds the cutoff frequency. Also we will want to include this value in the raw data file.

2.7 The Program BODE5C.BAS

A listing of the finished code with the modifications discussed in Sections 2.5 - 2.6 and some miscellaneous cosmetic changes is shown below:

```
REM File : BODE5C.BAS

REM     $INCLUDE: '\lw\instr\generato.inc'
REM     $INCLUDE: '\lw\instr\counter.inc'
REM     $INCLUDE: '\lw\instr\multimet.inc'
REM     $INCLUDE: '\lw\instr\scope.inc'
REM     $INCLUDE: '\lw\instr\general.inc'
REM     $INCLUDE: '\lw\include\phildecl.inc'
REM     $INCLUDE: '\lw\include\graphics.inc'

DEFINT A-Z

COMMON SHARED /dmml/ dmml AS INTEGER
COMMON SHARED /gnrl/ gnrl AS INTEGER
COMMON SHARED /oscl/ oscl AS INTEGER
COMMON SHARED /cntl/ cntl AS INTEGER
```

```
CLEAR , , 2048

DIM SHARED x#(500), y#(500), ph#(500)
CALL getmem(20000)
overlay.memory.size& = 98304
memory.size& = SETMEM(640000)
memory.size& = SETMEM(-overlay.memory.size&)

CLS

CALL res.glb
CALL reset.config
PRINT "Initializing Multimeter..."
CALL config(dmm1, "8840A,A 706,N DMM1")
PRINT "Initializing Function Generator..."
CALL config(gnr1, "PM5193/V2.5,A 707,N GNR1")
PRINT "Initializing Oscilloscope..."
CALL config(osc1, "PM3365/V07V04,A 708,N OSC1")
PRINT "Initializing Counter..."
CALL config(cnt1, "PM6666/22,A 710,N CNT1")
PRINT "Setting up defaults..."
CALL allinit(DEFAULT.SET)       'or ACTUAL.SET to leave unchanged
IF Glb.Stat > 1 THEN
      PRINT "Error: ", Glb.Str: REM Print global error string
      CALL res.glb: REM Reset global error status
      ELSE
IF Glb.Stat = 1 THEN
      PRINT "Warning: ", Glb.Str: REM Print global warning string
      CALL res.glb: REM Reset global error status
END IF
END IF

PRINT

StartOfProg:

OPEN "rawdata." FOR OUTPUT AS #1                  'save data in file

CALL set.amplitude(gnr1%, VRMS%, 1!)
CALL set.function(dmm1%, VOLT.AC%)
CALL set.speed(dmm1, low)
CALL set.function(cnt1%, TIMEINTERVAL.A.B)
CALL set.coupling(cnt1%, chall%, dc%)


CALL set.sensitivity(cnt1%, cha%, .02)
CALL set.sensitivity(cnt1%, chb%, .02)

INPUT "Enter starting frequency ->", strt!
INPUT "Enter ending frequency ->", stp!
INPUT "Enter number of points per decade ->", ppd
INPUT "Enter actual RMS output of function generator ->", ff#
IF ff# = 0 THEN ff# = 1  'assume 1 if they hit return
PRINT
PRINT "Frequency        dB        Angle"
PRINT

strt! = LOG(strt!) / LOG(10)
stp! = LOG(stp!) / LOG(10)
np = (stp! - strt!) * ppd                'number of magnitude points
nph = np                                  'number of phase points

FOR i! = strt! TO stp! STEP 1 / ppd
```

```
  p = (i! - strt!) * ppd
  x#(p) = 10 ^ i!
  CALL set.frequency(gnrl%, x#(p))
  CALL measure(dmml%, dvm#)
  dvm# = dvm# / ff#

  IF dvm# > .01 AND nph = np THEN       'take phase reading only if
voltage
                                    'is greater than 10 mV RMS
    IF x#(p) > 500 THEN
      CALL set.coupling(cntl%, chall%, ac%)
    ELSE
      CALL set.coupling(cntl%, chall%, dc%)
    END IF

    CALL measure(cntl%, diff#)

    ph#(p) = -diff# * x#(p) * 360#
    WHILE ph#(p) < -180
      ph#(p) = ph#(p) + 360#
    WEND
  ELSE
    IF nph = np THEN nph = p
  END IF

  y#(p) = 20 * LOG(dvm#) / LOG(10)      'convert to dB
  LOCATE CSRLIN - 1
  PRINT USING "##.###^^^^    +###.###    +###.###"; x#(p); y#(p);
ph#(p)
  PRINT #1, USING "##.###^^^^    +###.###    +###.###    +###.###
##.####^^^^"; x#(p); y#(p); ph#(p); dvm#; diff#
  x#(p) = i!
NEXT i!

PRINT

'Now look for -3 dB point

target# = .70711# / ff#
i = 1
WHILE ABS(EXP(LOG(10) * y#(i) / 20) - target#) / target# > .1 AND i
< np
  i = i + 1
WEND

IF i = np THEN
  PRINT "Couldn't find -3 dB frequency... run again with different
points"
  PRINT #1, ""
  PRINT #1, "Couldn't find -3 dB frequency... run again with
different points"
ELSE
  f1# = 10 ^ x#(i - 1)                  'starting frequencies
  f2# = 10 ^ x#(i)
  t1# = 10 ^ (y#(i - 1) / 20)           'starting target values
  t2# = 10 ^ (y#(i) / 20)
  DO
    slope# = (t2# - t1#) / (f2# - f1#)
    f0# = (target# - t1#) / slope# + f1#
    PRINT USING "Locating -3 dB frequency:  #####.##"; f0#
    LOCATE CSRLIN - 1
    'now find t0
    CALL set.frequency(gnrl%, f0#)
```

```
    CALL measure(dmm1%, t0#)
    IF (f2# - f0#) < (f0# - f1#) THEN
      f1# = f0#
      t1# = t0#
    ELSE
      f2# = f0#
      t2# = t0#
    END IF
  LOOP UNTIL t0# = target#

  PRINT USING "The cutoff frequency is #####.## Hz."; f0#
  PRINT #1, ""
  PRINT #1, USING "The cutoff frequency is #####.## Hz."; f0#

END IF

WHILE INKEY$ = "": WEND


CLOSE #1                                  'output raw data file

CALL GrfReset(4)
CALL SetAxGridVis(-1, 1)                 'enable grid lines for both
axes
CALL SetAxAuto(-1, 21)                   'auto-scale both axes every
time
CALL SetXDataType(4)
CALL SetYDataType(4)
CALL SetAxName(0, "log(f)")             'set name of x-axis
CALL SetAxName(1, "dB")                 'set name of y-axis
CALL GrfCurv2D(x#(), y#(), np)          'x-y plot of dB vs. log(f)
CALL GrfReset(4)                        'clear the graphics display
CALL SetAxName(1, "Phase Angle")        'set name of y-axis
CALL GrfCurv2D(x#(), ph#(), nph)        'x-y plot of phase vs. log(f)
CALL GrfLReset(0, 0, 1, 2)              'return to text mode

CLS
PRINT "Again (y/n)?";
a$ = INPUT$(1)
CLS
IF UCASE$(a$) = "Y" THEN GOTO StartOfProg

REM   Now return to local control

CALL iolocal(dmm1%)
CALL iolocal(osc1%)
CALL iolocal(cnt1%)
CALL iolocal(gnr1%)


END

REM $DYNAMIC
SUB ReportError
END SUB
```

2.8 Sample Data from BODE5C.BAS

The program was run for the high pass circuit shown in Figure 2.3. The resulting Bode plots are shown in Figures 2.6 - 2.7, and the raw data is shown in Table 2.1. The actual values of the components are R = 3.801 kΩ, R_S = 12.12Ω, and C_S = 105.31 nF (measured at 400 Hz.). The data was collected using 10 points per decade over a frequency range of 20 Hz to 50 kHz. The cutoff frequency was found to be 416.59 Hz by the program. Note that this is not the same as the theoretical cutoff frequency (396.34 Hz) but rather is the frequency at which the voltmeter read 0.70711. The error is due to the presence of R_S of the capacitor and the output impedance of the function generator. Correction of these errors is dealt with in the next chapter.

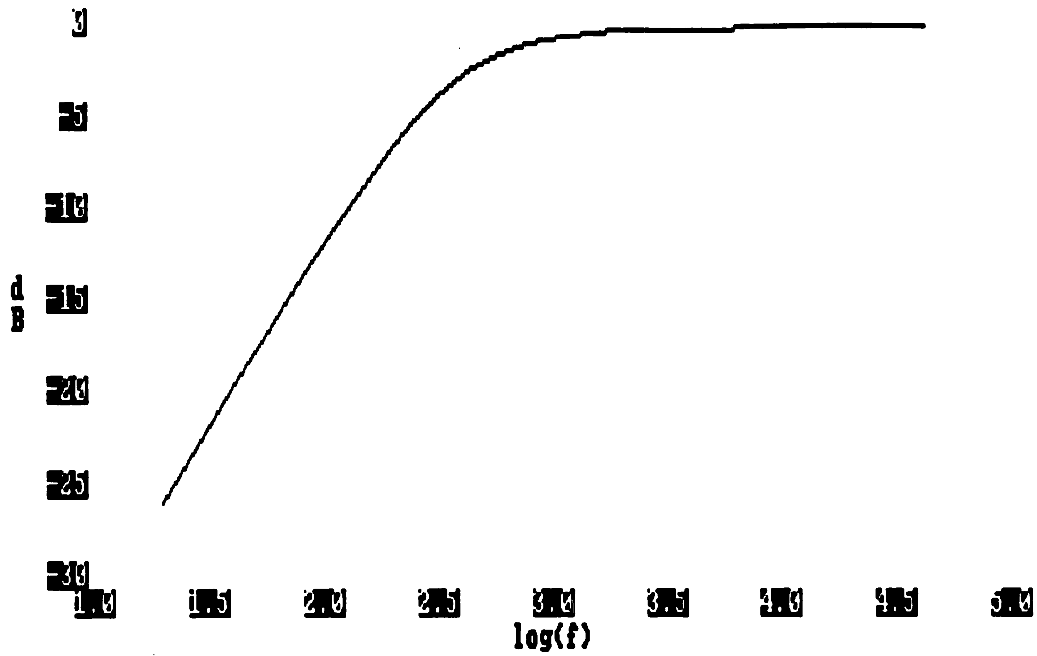

Figure 2.6 - Bode Magnitude Plot of High Pass Filter

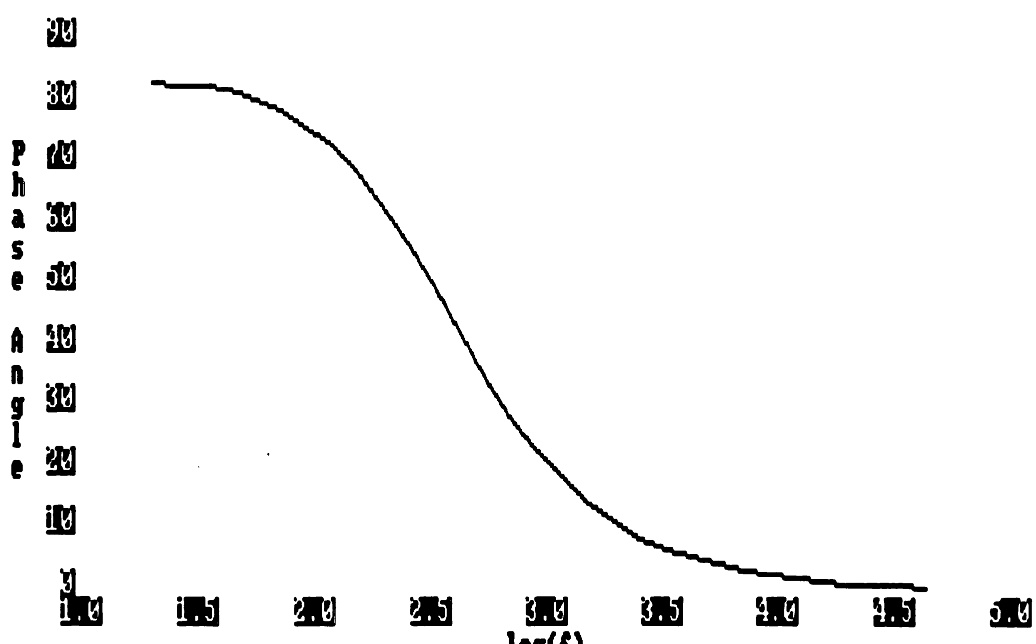

Figure 2.7 - Bode Phase Plot of High Pass Filter

Table 2.1 - Raw Data For High Pass Filter

2.000D+01	-26.225	+82.968	+0.049	3.8477D-02
2.518D+01	-24.164	+83.496	+0.062	3.0505D-02
3.170D+01	-22.159	+82.968	+0.078	2.4277D-02
3.991D+01	-20.167	+82.852	+0.098	1.9292D-02
5.024D+01	-18.187	+81.544	+0.123	1.5397D-02
6.325D+01	-16.225	+80.142	+0.154	1.2292D-02
7.962D+01	-14.288	+78.087	+0.193	9.8352D-03
1.002D+02	-12.386	+75.480	+0.240	7.8846D-03
1.262D+02	-10.543	+72.680	+0.297	6.3246D-03
1.589D+02	-8.775	+68.233	+0.364	5.1016D-03
2.000D+02	-7.120	+63.631	+0.441	4.1162D-03
2.518D+02	-5.613	+58.315	+0.524	3.3283D-03
3.170D+02	-4.294	+52.302	+0.610	2.6965D-03
3.991D+02	-3.191	+45.779	+0.693	2.1873D-03
5.024D+02	-2.315	+38.062	+0.766	1.7801D-03
6.325D+02	-1.653	+32.418	+0.827	1.4388D-03
7.962D+02	-1.174	+26.988	+0.874	1.1618D-03
1.002D+03	-0.842	+22.121	+0.908	9.3633D-04
1.262D+03	-0.616	+18.199	+0.932	7.5239D-04
1.589D+03	-0.465	+14.543	+0.948	6.0404D-04
2.000D+03	-0.366	+11.691	+0.959	4.8376D-04
2.518D+03	-0.299	+9.530	+0.966	3.8665D-04
3.170D+03	-0.255	+7.596	+0.971	3.0882D-04
3.991D+03	-0.224	+6.383	+0.975	2.4615D-04
5.024D+03	-0.203	+5.307	+0.977	1.9612D-04
6.325D+03	-0.189	+4.190	+0.979	1.5627D-04
7.962D+03	-0.177	+3.380	+0.980	1.2442D-04
1.002D+04	-0.168	+2.731	+0.981	9.9007D-05
1.262D+04	-0.159	+2.173	+0.982	7.8767D-05
1.589D+04	-0.150	+1.701	+0.983	6.2649D-05
2.000D+04	-0.140	+1.282	+0.984	4.9822D-05
2.518D+04	-0.138	+0.954	+0.984	3.9611D-05
3.170D+04	-0.147	+0.795	+0.983	3.1478D-05
3.991D+04	-0.154	+0.731	+0.982	2.5009D-05

The cutoff frequency is 416.59 Hz.

2.9 Summary

In this chapter TestTeam was utilized to analyze passive networks. Routines for creating Bode magnitude and phase plots were developed, and an algorithm for locating a -3 dB point was presented. These programs were tested as they were developed and sample data was provided.

CHAPTER 3

STEADY STATE ANALYSIS OF ACTIVE NETWORKS

3.1 Introduction

In the previous chapter we developed techniques to analyze passive networks. Now techniques are developed to measure active circuits. The topics covered include gain-bandwidth-products of op-amps and the second order filter parameters f_c, H_0, and Q_0.

3.2 Enhancements for the Measurement of Active Circuits

3.2.1 Addition of a Second Voltmeter

There are many circuits to which the 50Ω output impedance of the PM 5193 function generator is significant. That is, for the one-port network shown in Figure 3.1, V_S is significantly less than V_S. Until now, we have been assuming that $V_S \approx V_S$, which can result in significant errors. Consider the passive notch filter shown in Figure 3.2 with component values R=116.9 Ω, C_S = 206.2 nF, R_{SC} = 0.87 Ω, L_S = 907.34 μH, R_{SL} = 12.45 Ω (measured at 10 kHz). The output of the program as it is now is shown in Figures 3.3 - 3.4. The data was collected using 80 points per decade over a frequency range of 1 kHz to

100 kHz. The magnitude plot, however, is incorrect. At the center (notch) frequency, the input impedance of this circuit is $R+R_{SL}+R_{SC}$ = 130.22 Ω. Thus assuming the 50 Ω output impedance of the function generator is purely resistive,

$$V_S' = \frac{130.22}{130.22+50}(1\ V) = 0.723\ V$$

which is already nearly 3 dB down. The best solution to this problem is to add a second voltmeter to measure V_S'. As stated earlier, as long as the IEEE address is different, there will be no conflict on the bus. After re-running the configuration program, a new TESTTEAM.BAS file is generated.

```
REM File : TESTTEAM.BAS
REM Setup file for Philips/Fluke Instrument drivers
REM This will assign logical names to the instruments
REM and initialize the instruments

REM Following include statements can be removed in Interactive
window

REM     $INCLUDE: '\lw\instr\generato.inc'
REM     $INCLUDE: '\lw\instr\counter.inc'
REM     $INCLUDE: '\lw\instr\multimet.inc'
REM     $INCLUDE: '\lw\instr\scope.inc'
REM     $INCLUDE: '\lw\instr\general.inc'

DEFINT A-Z

COMMON SHARED /DMM1/ DMM1 AS INTEGER
COMMON SHARED /DMM2/ DMM2 AS INTEGER
COMMON SHARED /GNR1/ GNR1 AS INTEGER
COMMON SHARED /OSC1/ OSC1 AS INTEGER
COMMON SHARED /CNT1/ CNT1 AS INTEGER

CALL res.glb
CALL reset.config
CALL config(DMM1,"8840A,A 705,N DMM1")
CALL config(DMM2,"8840A,A 706,N DMM2")
CALL config(GNR1,"PM5193/V2.5,A 707,N GNR1")
CALL config(OSC1,"PM3365/V07V04,A 708,N OSC1")
CALL config(CNT1,"PM6666/22,A 710,N CNT1")
CALL allinit(DEFAULT.SET)
IF glb.stat > 1 THEN
        PRINT "Error: ",glb.str     :REM Print global error string
        call res.glb                :REM Reset global error status
```

```
ELSE
IF glb.stat = 1 THEN
        PRINT "Warning: ",glb.str :REM Print global warning string
        call res.glb              :REM Reset global error status
END IF
END IF
```

There are only three different lines in this code. First is the declaration of the new variable dmm2. TestTeam recognizes the two meters as dmm1 and dmm2. Then in the two CALL config lines the proper IEEE address is assigned to each meter (address 5 is assigned to the new meter). These three changes may be made directly to the BODE5C.BAS program.

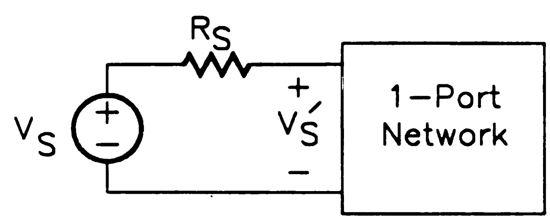

Figure 3.1 - Thevenin Equivalent of Function Generator

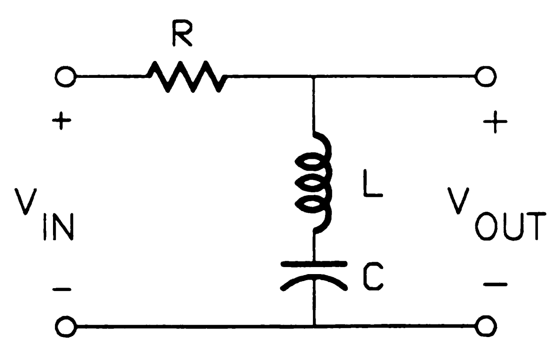

Figure 3.2 - Passive Notch Filter

The only necessary changes to the code to utilize the two meters are to take the ratio of the two measurements instead of taking only one measurement. The correct magnitude plot for the notch filter is shown in Figure 3.5.

Another advantage of using a second voltmeter is increased bandwidth. The upper frequency limit for the 8840A is 100 kHz. When using a second meter though, if its measurements roll off at the same rate as the first meter, their ratio will still be correct. For the set of instruments used, an effective increase in bandwidth to 400 kHz within 1% (0.1 dB) was observed.

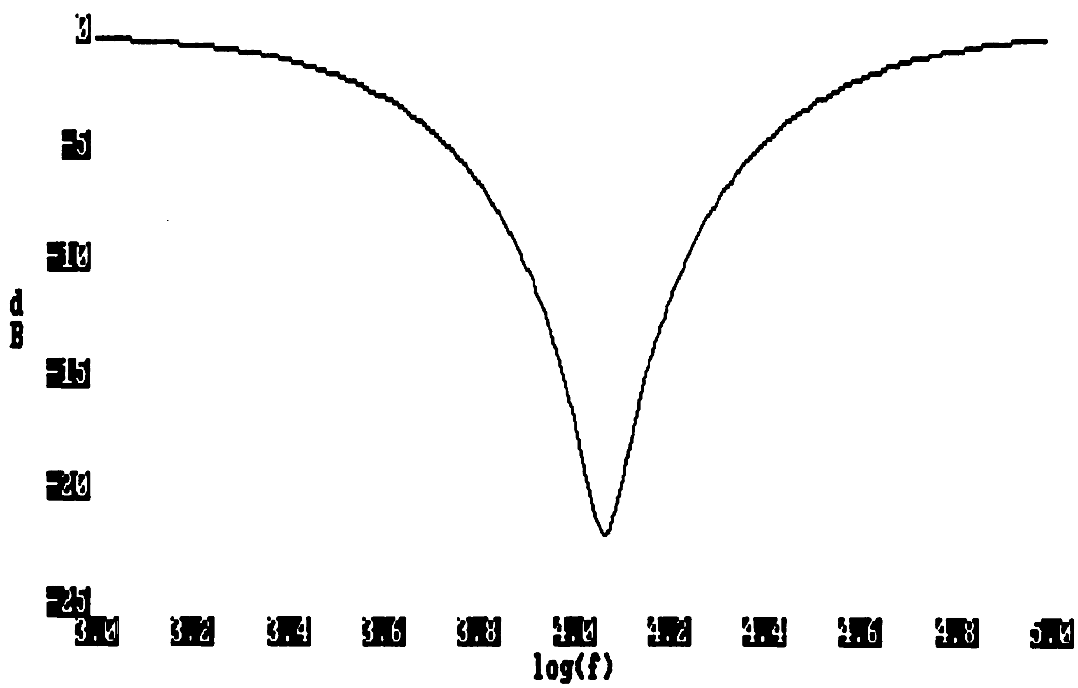

Figure 3.3 - Incorrect Bode Magnitude Plot for Notch Filter

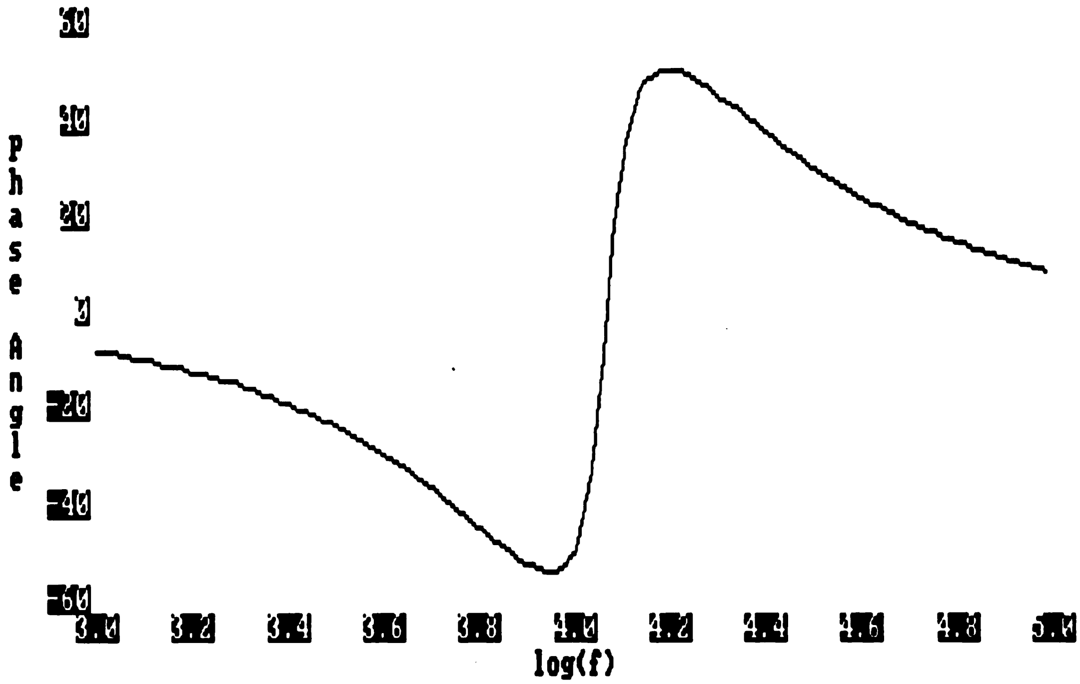

Figure 3.4 - Bode Phase Plot for Notch Filter

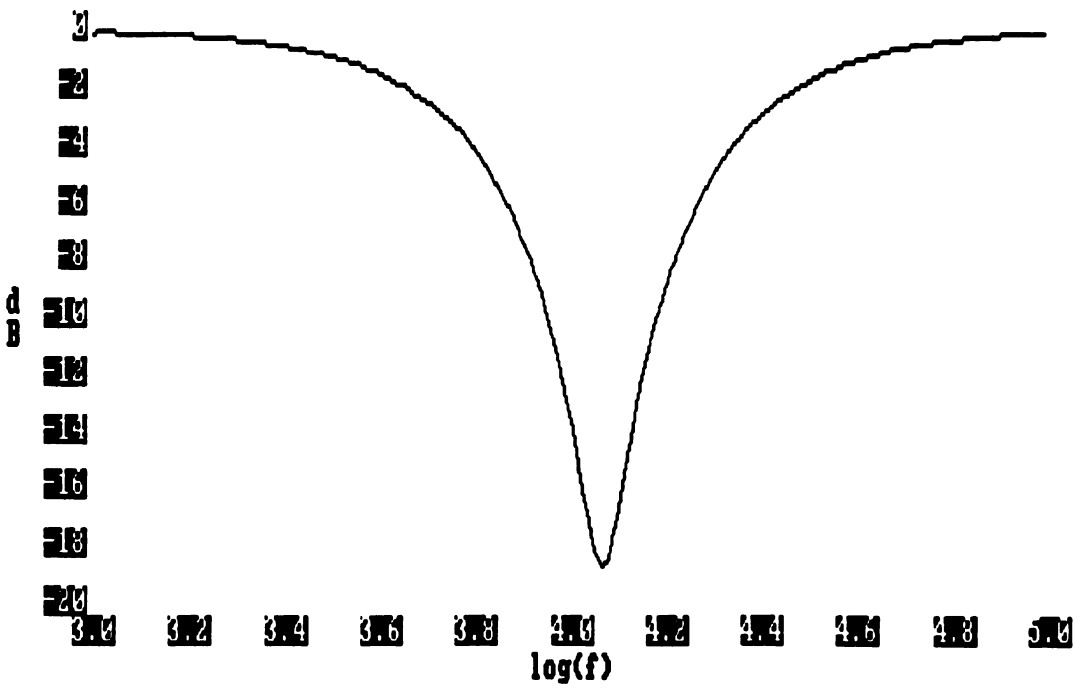

Figure 3.5 - Correct Bode Magnitude Plot for Notch Filter

3.2.2 Modification of -3 dB Routine for Active Circuits

In previous sections we have found the -3 dB point with respect to 1 V. For many circuits this may not be true. Let us examine a one-pole active low-pass filter. In this case we will measure the DC gain of the circuit, then find the -3 dB point with respect to it. Of course this will not work for high-pass or band-pass filters since their DC gain will be small. A solution to this problem is presented in section 3.6.3.

3.2.3 Measuring the Gain-Bandwidth-Product of an Op-Amp

Measuring the gain-bandwidth-product of an op-amp is another common required measurement. However certain circuit configurations make the measurement easier than others. Consider the inverting amplifier in Figure 3.6. By inserting a controlled source model of the op-amp (Figure 3.7) it can be shown that

$$\frac{V_2}{V_1} = -\frac{R_2}{R_1}\frac{1}{1+\frac{1}{A_{dm}}\frac{R_2+R_1}{R_1}}. \tag{3.1}$$

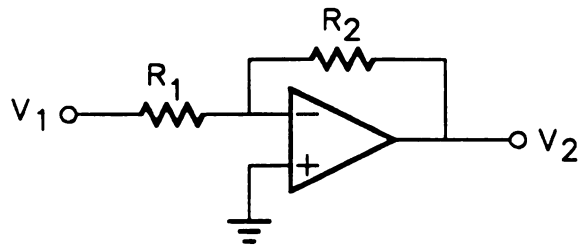

Figure 3.6 - Inverting Amplifier

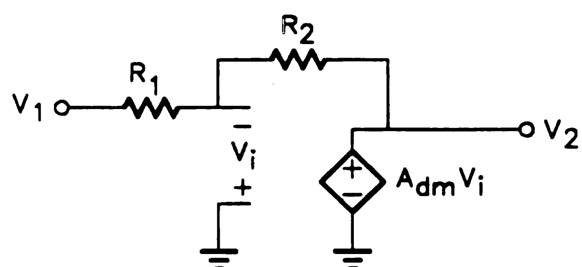

Figure 3.7 - Controlled Source Model of Inverting Amplifier

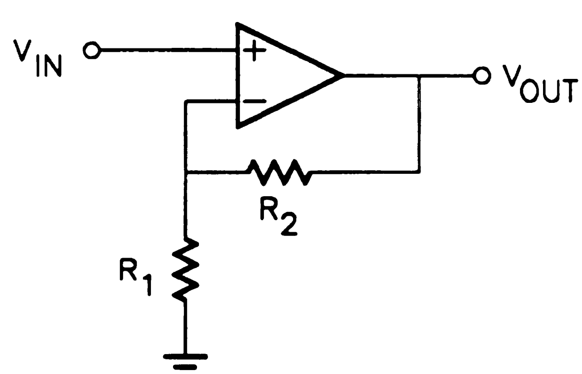

Figure 3.8 - Non-inverting Amplifier

Likewise for the non-inverting amplifier (Figure 3.8) it can be shown that

$$\frac{V_2}{V_1}=\frac{R_2+R_1}{R_1}\frac{1}{1+\frac{1}{A_{dm}}\frac{R_2+R_1}{R_1}}. \tag{3.2}$$

Note that both of these configurations (and indeed all one op-amp circuits [5]) have a transfer function of the form

$$F(s)=K\frac{1}{1+\frac{1}{A_{dm}}\frac{1}{\beta}}. \tag{3.3}$$

For low to mid frequencies, the open-loop differential mode gain A_{dm} is not constant, rather it acts as a one-pole transfer function. A one-pole model for A_{dm} is given by

$$A_{dm}=\frac{A_0\omega_0}{s+\omega_0}. \tag{3.4}$$

The frequency ω_0 is the frequency at which the gain is down by 3 dB. From the Fairchild data sheet for the 741 op-amp, A_0 is 200,000 and f_0 is 5 Hz. The product A_0f_0 is the gain-bandwidth-product. For frequencies much greater than f_0, A_{dm} can be approximated as

$$A_{dm}\approx\frac{A_0\omega_0}{s}. \tag{3.5}$$

Inserting this expression into Equation 3.3 yields

$$F(s)=K\frac{1}{1+\frac{s}{A_0\omega_0}\frac{1}{\beta}}, \tag{3.6}$$

and making the substitution s = jω gives

$$F(j\omega)=K\frac{1}{1+j\frac{\omega}{\omega_c}}, \tag{3.7}$$

where

$$f_c=\frac{GBP}{\frac{1}{\beta}}. \tag{3.8}$$

Note that if K = 1/ß, then the product $K{\cdot}f_c$ = GBP. The non-inverting amplifier has this property; the inverting amplifier does not. Thus to find the gain-bandwidth-product of an op-amp, configure it as a non-inverting amplifier. Then use BODE6C.BAS to find the -3 dB frequency of the circuit. The product of the DC gain and the cutoff frequency is then the gain-bandwidth-product. This is true as long as the one-pole approximation of A_{dm} holds. Thus K must be chosen large enough that the second pole of A_{dm} does not effect the results.

3.2.4 Aborting the Program

At certain times it may be desirable to abort operation of the program. Perhaps the circuit is not responding as expected, the wrong number of points per decade was selected, or a different sweep is desired. It would be convenient to simply press the ESCape key to stop operation of the program. This feature will also be implemented below.

3.3 The Listing of BODE6C.BAS

A listing of the program as it stands now is shown below.

```
REM File : BODE6C.BAS
REM        Support for second voltmeter added
REM        and support for active circuits

REM Following include statements can be removed in Interactive
window

REM     $INCLUDE: '\lw\instr\generato.inc'
REM     $INCLUDE: '\lw\instr\counter.inc'
REM     $INCLUDE: '\lw\instr\multimet.inc'
REM     $INCLUDE: '\lw\instr\scope.inc'
REM     $INCLUDE: '\lw\instr\general.inc'
REM     $INCLUDE: '\lw\include\phildecl.inc'
REM     $INCLUDE: '\lw\include\graphics.inc'

DEFINT A-Z

COMMON SHARED /dmm1/ dmm1 AS INTEGER
COMMON SHARED /dmm2/ dmm2 AS INTEGER
COMMON SHARED /gnr1/ gnr1 AS INTEGER
COMMON SHARED /osc1/ osc1 AS INTEGER
COMMON SHARED /cnt1/ cnt1 AS INTEGER

CLEAR , , 2048

DIM SHARED x#(500), y#(500), ph#(500)
CALL getmem(20000)
overlay.memory.size& = 98304
memory.size& = SETMEM(640000)
memory.size& = SETMEM(-overlay.memory.size&)

CLS

CALL res.glb
CALL reset.config
PRINT "Initializing Multimeters..."
CALL config(dmm1, "8840A,A 705,N DMM1")
CALL config(dmm2, "8840A,A 706,N DMM2")
PRINT "Initializing Function Generator..."
CALL config(gnr1, "PM5193/V2.5,A 707,N GNR1")
PRINT "Initializing Oscilloscope..."
CALL config(osc1, "PM3365/V07V04,A 708,N OSC1")
PRINT "Initializing Counter..."
```

```
CALL config(cntl, "PM6666/22,A 710,N CNT1")
PRINT "Setting up defaults..."
CALL allinit(DEFAULT.SET)        'or ACTUAL.SET to leave unchanged
IF Glb.Stat > 1 THEN
      PRINT "Error: ", Glb.Str: REM Print global error string
      CALL res.glb: REM Reset global error status
      ELSE
IF Glb.Stat = 1 THEN
      PRINT "Warning: ", Glb.Str: REM Print global warning string
      CALL res.glb: REM Reset global error status
END IF
END IF

PRINT


StartOfProg:

OPEN "rawdata." FOR OUTPUT AS #1                     'save data in file

CALL set.function(dmm1%, volt.ac%)
CALL set.function(dmm2%, volt.ac%)
CALL set.speed(dmm1, low)
CALL set.speed(dmm2, low)
CALL set.function(cntl%, TIMEINTERVAL.A.B)
CALL set.coupling(cntl%, chall%, dc%)

CALL set.sensitivity(cntl%, cha%, .02)
CALL set.sensitivity(cntl%, chb%, .02)

INPUT "Enter starting frequency ->", strt!
INPUT "Enter ending frequency ->", stp!
INPUT "Enter number of points per decade ->", ppd
INPUT "Enter desired RMS function generator voltage (default 1 V)
->", gnv!
IF gnv! = 0 THEN gnv! = 1

CALL set.offset(gnrl, 0#)
CALL set.amplitude(gnrl, vrms, gnv!)


PRINT
PRINT "Frequency       dB       Angle"
PRINT

strt! = LOG(strt!) / LOG(10)
stp! = LOG(stp!) / LOG(10)
np = (stp! - strt!) * ppd                  'number of magnitude points
nph = np                                   'number of phase points

FOR i! = strt! TO stp! STEP 1 / ppd
  p = (i! - strt!) * ppd
  x#(p) = 10 ^ i!

  IF INKEY$ = CHR$(27) THEN
    PRINT
    PRINT "Run aborted..."
    PRINT
    PRINT #1, ""
    PRINT #1, "Run aborted..."
    CLOSE
    GOTO Again
  END IF
```

```
  CALL set.frequency(gnrl%, x#(p))
  CALL measure(dmml%, dvml#)          'input
  CALL measure(dmm2%, dvm2#)          'output
  dvm# = ABS(dvm2# / dvml#)          'find transfer fn output/input

  IF dvm2# > 0 AND nph = np THEN      'take phase reading only if
output
                                    'is greater than 10 mV RMS
    IF x#(p) > 500 THEN
      CALL set.coupling(cntl%, chall%, ac%)
    ELSE
      CALL set.coupling(cntl%, chall%, dc%)
    END IF

    CALL measure(cntl%, diff#)

    ph#(p) = -diff# * x#(p) * 360#
    WHILE ph#(p) < -180
      ph#(p) = ph#(p) + 360#
    WEND
  ELSE
    IF nph = np THEN nph = p
  END IF

  y#(p) = 20 * LOG(dvm#) / LOG(10)     'convert to dB
  LOCATE CSRLIN - 1
  PRINT USING "##.###^^^^    +###.###    +###.###"; x#(p); y#(p);
ph#(p)
  PRINT #1, USING "##.###^^^^    +###.###    +###.###    +###.###
##.####^^^^"; x#(p); y#(p); ph#(p); dvm#; diff#
  x#(p) = i!
NEXT i!

PRINT

'Now find DC gain

CALL set.amplitude(gnrl, vrms, 0)     'turn off AC voltage from gen
CALL set.offset(gnrl, gnv!)           'turn on DC volts from gen

CALL set.function(dmml, volt.dc)
CALL set.function(dmm2, volt.dc)

CALL measure(dmml, dvml#)
CALL measure(dmm2, dvm2#)

hdc# = dvm2# / dvml#                  'DC gain (NOT dB)

CALL set.offset(gnrl, 0#)             'return DMM to its original
CALL set.function(dmml, volt.ac)      'state
CALL set.function(dmm2, volt.ac)
CALL set.amplitude(gnrl, vrms, gnv!)

PRINT
PRINT #1, ""

PRINT USING "DC Gain = ###.###"; hdc#;
IF hdc# <> 0 THEN PRINT USING " = ##.### dB"; 20 * LOG(ABS(hdc#)) /
LOG(10)
PRINT #1, USING "DC Gain = ###.###"; hdc#;
IF hdc# <> 0 THEN PRINT #1, USING " = ##.### dB"; 20 *
LOG(ABS(hdc#)) / LOG(10)
```

```
'Now look for -3 dB point from left side

PRINT
PRINT #1, ""

target# = hdc# / SQR(2#)
IF target# < .001 THEN
  PRINT "DC Gain is too low - Cannot sweep for -3 dB point from
left"
  PRINT #1, "DC Gain is too low - Cannot sweep for -3 dB point from
left"
  GOTO graphit
END IF
i = 1
WHILE ABS(EXP(LOG(10#) * y#(i) / 20#) - target#) / target# > .1# AND
i < np
  i = i + 1
WEND

IF i = np THEN
  PRINT "Couldn't find -3 dB frequency... run again with different
points"
  PRINT #1, ""
  PRINT #1, "Couldn't find -3 dB frequency... run again with
different points"
ELSE
  f1# = 10 ^ x#(i - 1)                  'starting frequencies
  f2# = 10 ^ x#(i)
  t1# = 10 ^ (y#(i - 1) / 20)           'starting target values
  t2# = 10 ^ (y#(i) / 20)

  DO
    slope# = (t2# - t1#) / (f2# - f1#)
    f0# = (target# - t1#) / slope# + f1#
    PRINT USING "Locating -3 dB frequency:  #####.##"; f0#
    LOCATE CSRLIN - 1

    IF INKEY$ = CHR$(27) THEN
      PRINT "Location of -3 dB frequency aborted...                "
      PRINT #1, ""
      PRINT #1, "Location of -3 dB frequency aborted..."
      GOTO graphit
    END IF

    'now find t0
    CALL set.frequency(gnr1%, f0#)
    CALL measure(dmm1, dvm1#)
    CALL measure(dmm2, dvm2#)
    t0# = dvm2# / dvm1#
    IF (f2# - f0#) < (f0# - f1#) THEN
      f1# = f0#
      t1# = t0#
    ELSE
      f2# = f0#
      t2# = t0#
    END IF
    hg& = 100000# * t0# / hdc#
  LOOP UNTIL hg& = 70711

  PRINT USING "The cutoff frequency is #####.## Hz."; f0#
  PRINT USING "GBP = #########.## Hz"; f0# * hdc#
  PRINT #1, USING "The cutoff frequency is #####.## Hz."; f0#
  PRINT #1, USING "GBP = #########.## Hz"; f0# * hdc#
```

```
  PRINT #1, "Actual cutoff ratio = "; t0# / hdc#

END IF

graphit:

CLOSE #1                            'output raw data file

CALL set.amplitude(gnrl, vrms, 0) 'turn off AC voltage from gen
                                  'to facilitate changing components

PRINT
PRINT "Press any key to see graphs...";

WHILE INKEY$ = "": WEND


CALL GrfReset(4)
CALL SetAxGridVis(-1, 1)                'enable grid lines for both
axes
CALL SetAxAuto(-1, 21)                  'auto-scale both axes every
time
CALL SetXDataType(4)
CALL SetYDataType(4)
CALL SetAxName(0, "log(f)")           'set name of x-axis
CALL SetAxName(1, "dB")               'set name of y-axis
CALL GrfCurv2D(x#(), y#(), np)        'x-y plot of dB vs. log(f)
CALL GrfReset(4)                      'clear the graphics display
CALL SetAxName(1, "Phase Angle")      'set name of y-axis
CALL GrfCurv2D(x#(), ph#(), nph)      'x-y plot of phase vs. log(f)
CALL GrfLReset(0, 0, 1, 2)            'return to text mode

CLS

Again:

PRINT "Again (y/n)?";
a$ = INPUT$(1)
CLS
IF UCASE$(a$) = "Y" THEN GOTO StartOfProg

REM   Now return to local control

CALL iolocal(dmm1%)
CALL iolocal(dmm2%)
CALL iolocal(osc1%)
CALL iolocal(cnt1%)
CALL iolocal(gnrl%)


END

REM $DYNAMIC
SUB ReportError
END SUB
```

3.4 Sample Data from BODE6C.BAS

As stated earlier, one useful application of this program is to measure the Gain-Bandwidth-Product of an op-amp. A non-inverting amplifier (Figure 3.8) is configured using an LM741 op-amp with R_1 = 1 kΩ and R_2 = 47 kΩ (nominal values). The resulting Bode plots are shown in Figures 3.9 - 3.10. Unfortunately, the bandwidth of the meters is only 400 kHz, since by examining the phase plot there appears to be a second pole around 4 MHz. The DC gain was found to be 48.01, with a GBP of 828.3 kHz. The Fairchild data sheet for the 741 lists a GBP of 1 MHz, however this characteristic varies widely from sample to sample. Thus the results are considered acceptable.

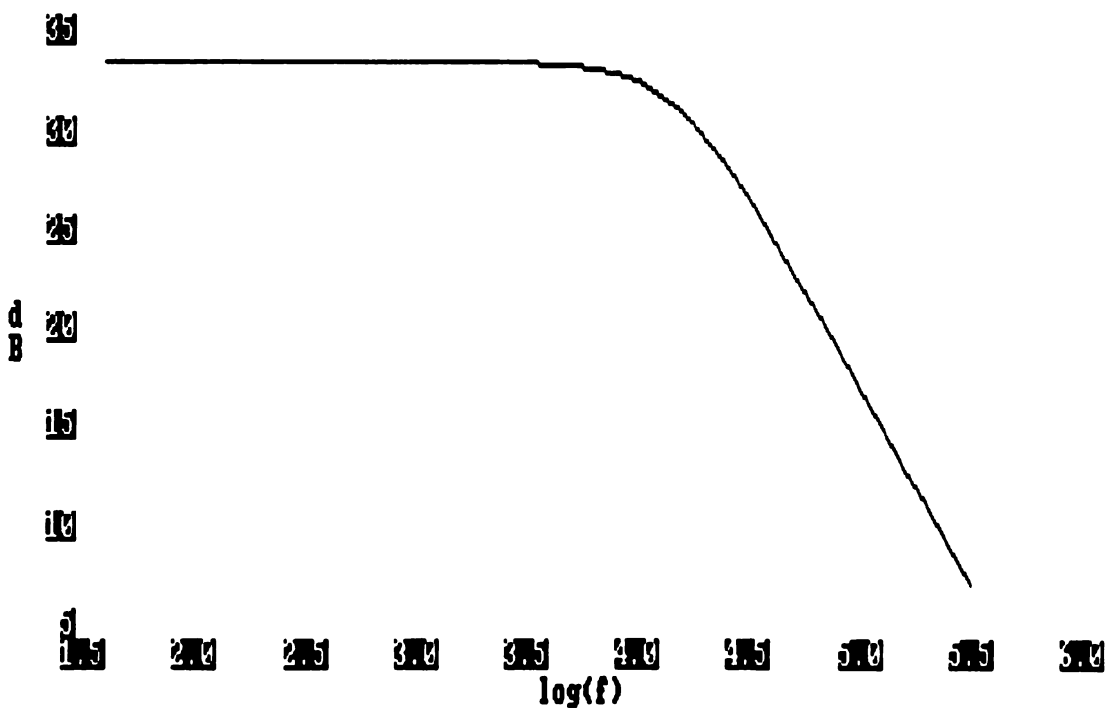

Figure 3.9 - Bode Magnitude Plot of Non-inverting Amplifier

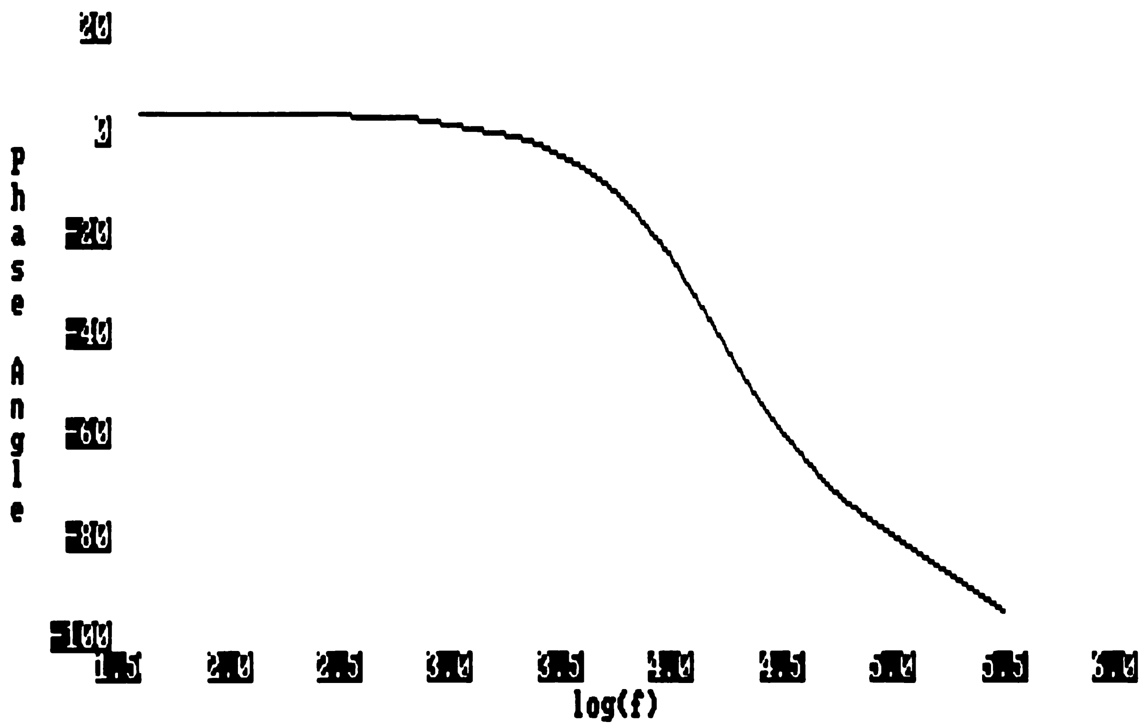

Figure 3.10 - Bode Phase Plot of Non-inverting Amplifier

3.5 Testing for Linearity

The measurements we have been taking are only valid when performed on linear systems. If a circuit is not behaving linearly, the measurements may be worthless. One possibility is that an op-amp in a circuit has saturated. A method of checking for linearity is to reduce the input by some ratio (perhaps 50%) and check that the output is also reduced by the same ratio. If not, a warning to the user is displayed. The QuickBASIC code to add to the program to do this is given below.

```
CALL set.amplitude(gnr1, vrms, gnv! / 2!)
CALL measure(dmm1, dvm1#)
CALL measure(dmm2, dvm2#)
IF ABS(dvm2# / dvm1# - dvm# / 2#) / (dvm# / 2#) > .01 THEN
```

```
  PRINT USING "Warning - Linearity Check Failed at f = ##.####^^^^";
x#(p)
  LOCATE CSRLIN - 1
  PRINT #1, USING "Warning - Linearity Check Failed at f =
##.####^^^^"; x#(p)
END IF
CALL set.amplitude(gnrl, vrms, gnv!)
```

It may also be desirable to do this only once every five or ten points, since it slows down the program considerably. Due to the delays imposed by this routine, it is not included in any other listings.

3.6 Automated Measurement of Filter Parameters

Another tedious measurement that would be useful to implement is the measurement of the second order filter parameters f_c, H_0, and Q_0. Finding Q_0 of a high-Q filter by hand is particularly difficult. When properly programmed, however, it becomes as simple as the touch of a button.

3.6.1 Location of a Second -3 dB point

If the program BODE6C.BAS was run on a notch filter, it would locate the -3 dB frequency to the left of the notch. This is because we begin our sweep from the left. If we were to sweep for a second -3 dB point from the right, we would be able to calculate the bandwidth of this filter. One question that rises immediately is what to find this second -3 dB point with respect to:

the DC value or the highest frequency point taken. There are some notch filters that do not return to the DC gain to the right of the notch. We will assume that all notch filters swept by this program do return to the DC gain. To ensure compatibility of this program with other filters, we will allow the user to look for 0, 1, or 2 cutoff frequencies.

3.6.2 Calculation of the Center Frequency of a Second-order Notch Filter

If the notch filter is a second-order filter, then the center frequency can be calculated by the formula

$$f_c=\sqrt{f_1\, f_2} \qquad (3.9)$$

where f_1 and f_2 are the two cutoff frequencies, and the Q of the filter can be found by

$$Q_0=\frac{f_c}{f_2-f_1}. \qquad (3.10)$$

Calculating f_c is certainly faster than searching for the center of the notch, and is sufficiently accurate for many applications. In the next section we will develop a technique for locating the peak of a band-pass filter; if desired that routine could be altered to locate the center of the notch.

3.6.3 Finding the Parameters of Band-pass Filters

Locating the cutoff frequencies of a band-pass filter is much more difficult than it was for the notch filter. The hardest part is finding the center frequency. Unlike the notch filter, where we could use the DC gain as the reference for the -3 dB frequencies, there is no reference to begin with. Thus the only way to find the cutoff frequencies is to first find the peak of the filter. An algorithm to find the peak is detailed below.

1. Sweep the bode plot as before.
2. Find the maximum value in the array y#. This will be the starting point. Call this point f_0.
3. Set the variable $\Delta\log f = 0.1$. This will be the logarithmic window in which we will look.
4. Measure the voltage at the two endpoints of the window,

$$f_{1,2} = 10^{(\log f_0 \pm \Delta\log f)}$$

5. If the voltage at either f_1 or f_2 is larger than at f_0, then set f_0 to the frequency at which the voltage is highest, and go to step 4.
6. Otherwise divide $\Delta\log f$ by 2.
7. If $\Delta\log f > 0.00001$ (or some acceptable tolerance), go to step 4. Otherwise, $f_c = f_0$ is the center frequency.

Once we have found f_c we can find H_0 by measuring the output voltage at that frequency, and we can find the cutoff frequencies relative to H_0 in the same manner as we have done all along. The formula given for Q_0 in Equation 3.10 is still valid, and now we have all the filter parameters. Note that once these modifications are made to the program, it can be used to locate the cutoff frequency of a high-pass filter by sweeping the bode plot to the highest allowable frequency (400 kHz for the set of instruments used), then finding one cutoff frequency with respect to the peak output.

3.7 Accuracy of These Routines

The accuracy of these routines are limited mainly by the accuracy of the instruments used. For a Fluke 8840A DMM calibrated within one year in True RMS AC volts mode, the worst case accuracy (from 50 - 100 kHz) is given by the manual to be ±(0.5% + 400 counts). In the 2 V range (full scale reading 1.99999 V), if the voltmeter reads 1 V, this says that the actual voltage lies in the range

$$0.991 \leq V \leq 1.009.$$

So the worst possible readings of a transfer function of 1 when using the ratio of two meters are

$$\frac{1.009}{0.991}=1.018$$

and

$$\frac{0.991}{1.009}=0.982.$$

Based upon the agreement of theoretical and experimental data presented in chapter 4, however, it appears that these numbers may be quite pessimistic. The program does appear to be quite consistent in its results, within one count at the fourth significant figure. Henceforth all measured values are reported to four significant figures.

3.8 The Program BODE7C.BAS

A listing of the program BODE7C is given below. The program will be tested in the next chapter.

```
REM File : BODE7C.BAS
REM        Added ability to search for two -3 dB points
REM        and calculate Ho, Qo for band-pass & notch filters

REM Following include statements can be removed in Interactive
window

REM     $INCLUDE: '\lw\instr\generato.inc'
REM     $INCLUDE: '\lw\instr\counter.inc'
REM     $INCLUDE: '\lw\instr\multimet.inc'
REM     $INCLUDE: '\lw\instr\scope.inc'
REM     $INCLUDE: '\lw\instr\general.inc'
REM     $INCLUDE: '\lw\include\phildecl.inc'
REM     $INCLUDE: '\lw\include\graphics.inc'

DEFINT A-Z

COMMON SHARED /dmm1/ dmm1 AS INTEGER
COMMON SHARED /dmm2/ dmm2 AS INTEGER
COMMON SHARED /gnr1/ gnr1 AS INTEGER
COMMON SHARED /osc1/ osc1 AS INTEGER
COMMON SHARED /cnt1/ cnt1 AS INTEGER

CLEAR , , 2048

DIM SHARED x#(500), y#(500), ph#(500)
CALL getmem(20000)
overlay.memory.size& = 98304
memory.size& = SETMEM(640000)
memory.size& = SETMEM(-overlay.memory.size&)

CLS

CALL res.glb
CALL reset.config
PRINT "Initializing Multimeters..."
CALL config(dmm1, "8840A,A 705,N DMM1")
CALL config(dmm2, "8840A,A 706,N DMM2")
PRINT "Initializing Function Generator..."
CALL config(gnr1, "PM5193/V2.5,A 707,N GNR1")
```

```
PRINT "Initializing Oscilloscope..."
CALL config(osc1, "PM3365/V07V04,A 708,N OSC1")
PRINT "Initializing Counter..."
CALL config(cntl, "PM6666/22,A 710,N CNT1")
PRINT "Setting up defaults..."
CALL allinit(DEFAULT.SET)        'or ACTUAL.SET to leave unchanged
IF Glb.Stat > 1 THEN
      PRINT "Error: ", Glb.Str: REM Print global error string
      CALL res.glb: REM Reset global error status
      ELSE
IF Glb.Stat = 1 THEN
      PRINT "Warning: ", Glb.Str: REM Print global warning string
      CALL res.glb: REM Reset global error status
END IF
END IF

PRINT


StartOfProg:

OPEN "rawdata." FOR OUTPUT AS #1                  'save data in file

CALL set.function(dmm1%, volt.ac%)
CALL set.function(dmm2%, volt.ac%)
CALL set.speed(dmm1, low)
CALL set.speed(dmm2, low)
CALL set.function(cntl%, TIMEINTERVAL.A.B)
CALL set.coupling(cntl%, chall%, dc%)

CALL set.sensitivity(cntl%, cha%, .02)
CALL set.sensitivity(cntl%, chb%, .02)

INPUT "Enter starting frequency ->", strt!
INPUT "Enter ending frequency ->", stp!
INPUT "Enter number of points per decade ->", ppd
INPUT "Enter desired RMS function generator voltage (default 1 V)
->", gnv!
IF gnv! = 0 THEN gnv! = 1

CALL set.offset(gnrl, 0#)
CALL set.amplitude(gnrl, vrms, gnv!)


PRINT
PRINT "Frequency       dB       Angle"
PRINT

strt! = LOG(strt!) / LOG(10)
stp! = LOG(stp!) / LOG(10)
np = (stp! - strt!) * ppd                'number of magnitude points
nph = np                                 'number of phase points

FOR i! = strt! TO stp! STEP 1 / ppd
  p = (i! - strt!) * ppd
  x#(p) = 10 ^ i!

  IF INKEY$ = CHR$(27) THEN
    PRINT
    PRINT "Run aborted..."
    PRINT
    PRINT #1, ""
    PRINT #1, "Run aborted..."
```

```
    CLOSE
    GOTO Again
  END IF
  CALL set.frequency(gnrl%, x#(p))
  CALL measure(dmml%, dvml#)        'input
  CALL measure(dmm2%, dvm2#)        'output
  dvm# = ABS(dvm2# / dvml#)         'find transfer fn output/input

  IF dvm2# > 0 AND nph = np THEN  'take phase reading only if output
                                  'is greater than 10 mV RMS
    IF x#(p) > 500 THEN
      CALL set.coupling(cntl%, chall%, ac%)
    ELSE
      CALL set.coupling(cntl%, chall%, dc%)
    END IF

    CALL measure(cntl%, diff#)

    ph#(p) = -diff# * x#(p) * 360#
    WHILE ph#(p) < -180
      ph#(p) = ph#(p) + 360#
    WEND
  ELSE
    IF nph = np THEN nph = p
  END IF

  y#(p) = 20 * LOG(dvm#) / LOG(10)     'convert to dB
  LOCATE CSRLIN - 1
  PRINT USING "##.###^^^^    +###.###    +###.###"; x#(p); y#(p);
ph#(p)
  PRINT #1, USING "##.###^^^^    +###.###    +###.###    +###.###
##.####^^^^"; x#(p); y#(p); ph#(p); dvm#; diff#
  x#(p) = i!
NEXT i!

PRINT

'Now find out what to do with this data

DO
  INPUT "Find (P)eak or (D)C gain? ", g$
  g$ = UCASE$(g$)
LOOP UNTIL g$ = "P" OR g$ = "D"

DO
  INPUT "Search for how many -3 dB frequencies (0,1,2) ? ", cf
LOOP UNTIL cf = 0 OR cf = 1 OR cf = 2

IF g$ = "D" THEN GOTO dcgain

'Find peak value of the points we took first

k = 1                             'pointer to maximum value
FOR i = 2 TO np - 1
  IF y#(i) > y#(k) THEN k = i   'reset pointer if we found a new
max.
NEXT i

IF k = np - 1 THEN                'if rightmost point is max, use it
as h0
  h0# = 10# ^ (y#(k) / 20#)
  GOTO sweepleft
END IF
```

```
PRINT

deltalogf# = .1
f0# = x#(k)                         'log of freq. of peak magnitude
t0# = y#(k)                         'dB of peak magnitude

WHILE deltalogf# > .00001
  PRINT USING "Locating center frequency:  ######.##"; 10# ^ f0#
  LOCATE CSRLIN - 1

  CALL set.frequency(gnr1, 10# ^ (f0# - deltalogf#))
  CALL measure(dmm1%, dvm1#)                        'input
  CALL measure(dmm2%, dvm2#)                        'output
  t1# = 20# * LOG(dvm2# / dvm1#) / LOG(10#)   'gain at lower freq.

  CALL set.frequency(gnr1, 10# ^ (f0# + deltalogf#))
  CALL measure(dmm1%, dvm1#)                        'input
  CALL measure(dmm2%, dvm2#)                        'output
  t2# = 20# * LOG(dvm2# / dvm1#) / LOG(10#)   'gain at upper freq.

  IF t1# < t0# AND t2# < t0# THEN
    deltalogf# = deltalogf# / 2#
  ELSE
    IF t1# > t0# THEN
      t0# = t1#
      f0# = f0# - deltalogf#                        'set new freq. lower
    END IF
    IF t2# > t0# THEN
      t0# = t2#
      f0# = f0# + deltalogf#                        'set new freq. higher
    END IF
  END IF
WEND

h0# = 10# ^ (t0# / 20#)                   'convert back from dB
fcn# = 10# ^ f0#

PRINT USING "fc = ######.### Hz.                              "; fcn#
PRINT USING "Ho = ##.#####"; h0#

PRINT #1, USING "fc = ######.### Hz."; fcn#
PRINT #1, USING "Ho = ##.#####"; h0#

GOTO sweepleft

'Now find DC gain

dcgain:

CALL set.amplitude(gnr1, vrms, 0)    'turn off AC voltage from gen
CALL set.offset(gnr1, gnv!)          'turn on DC volts from gen

CALL set.function(dmm1, volt.dc)
CALL set.function(dmm2, volt.dc)

CALL measure(dmm1, dvm1#)
CALL measure(dmm2, dvm2#)

h0# = dvm2# / dvm1#                  'DC gain (NOT dB)

CALL set.offset(gnr1, 0#)            'return DMM to its original
CALL set.function(dmm1, volt.ac)     'state
CALL set.function(dmm2, volt.ac)
```

```
CALL set.amplitude(gnrl, vrms, gnv!)

PRINT
PRINT #1, ""

PRINT USING "DC Gain = ###.#####"; h0#;
IF h0# <> 0 THEN PRINT USING " = ##.### dB"; 20 * LOG(ABS(h0#)) /
LOG(10)
PRINT #1, USING "DC Gain = ###.#####"; h0#;
IF h0# <> 0 THEN PRINT #1, USING " = ##.### dB"; 20 * LOG(ABS(h0#))
/ LOG(10)

'Now look for -3 dB point from left side

sweepleft:

IF cf = 0 THEN GOTO graphit

PRINT
PRINT #1, ""

target# = h0# / SQR(2#)
IF target# < .001 THEN
  PRINT "DC Gain is too low - Cannot sweep for -3 dB point from
left"
  PRINT #1, "DC Gain is too low - Cannot sweep for -3 dB point from
left"
  f0# = 0
  GOTO sweepright
END IF
i = 1
WHILE ABS(EXP(LOG(10#) * y#(i) / 20#) - target#) / target# > .2# AND
i < np
  i = i + 1
WEND

IF i = np THEN
  PRINT "Couldn't find left -3 dB frequency... run again with
different points"
  PRINT #1, ""
  PRINT #1, "Couldn't find left -3 dB frequency... run again with
different points"
ELSE
  f1# = 10 ^ x#(i - 1)                'starting frequencies
  f2# = 10 ^ x#(i)
  t1# = 10 ^ (y#(i - 1) / 20)         'starting target values
  t2# = 10 ^ (y#(i) / 20)

  DO
    slope# = (t2# - t1#) / (f2# - f1#)
    f0# = (target# - t1#) / slope# + f1#
    PRINT USING "Locating left -3 dB frequency:  #####.##"; f0#
    LOCATE CSRLIN - 1

    IF INKEY$ = CHR$(27) THEN
      PRINT "Location of left -3 dB frequency aborted...
  "
      PRINT #1, ""
      PRINT #1, "Location of left -3 dB frequency aborted..."
      f0# = 0
      GOTO sweepright
    END IF
```

```
    'now find t0
    CALL set.frequency(gnr1%, f0#)
    CALL measure(dmm1, dvm1#)
    CALL measure(dmm2, dvm2#)
    t0# = dvm2# / dvm1#
    IF (f2# - f0#) < (f0# - f1#) THEN
      f1# = f0#
      t1# = t0#
    ELSE
      f2# = f0#
      t2# = t0#
    END IF
    hg& = 100000# * t0# / h0#
  LOOP UNTIL hg& = 70711

  fcl# = f0#              'left cutoff frequency
  PRINT USING "The left cutoff frequency is #####.## Hz."; f0#
  PRINT #1, USING "The left cutoff frequency is #####.## Hz."; f0#

  IF cf = 1 THEN
    PRINT USING "GBP = #########.## Hz"; f0# * h0#
    PRINT #1, USING "GBP = #########.## Hz"; f0# * h0#
    GOTO graphit
  END IF

END IF

sweepright:

'Now look for -3 dB point from right side

PRINT
PRINT #1, ""

IF target# < .001 THEN
  PRINT "Rightmost Gain is too low - Cannot sweep for -3 dB point
from right"
  PRINT #1, "Rightmost Gain is too low - Cannot sweep for -3 dB
point from right"
  GOTO graphit
END IF
i = np - 1
WHILE ABS(EXP(LOG(10#) * y#(i) / 20#) - target#) / target# > .2# AND
i > 0
  i = i - 1
WEND

IF i = 0 THEN
  PRINT "Couldn't find right -3 dB frequency... run again with
different points"
  PRINT #1, ""
  PRINT #1, "Couldn't find right -3 dB frequency... run again with
different points"
ELSE
  f1# = 10 ^ x#(i - 1)                'starting frequencies
  f2# = 10 ^ x#(i)
  t1# = 10 ^ (y#(i - 1) / 20)         'starting target values
  t2# = 10 ^ (y#(i) / 20)

  DO
    slope# = (t2# - t1#) / (f2# - f1#)
    f0# = (target# - t1#) / slope# + f1#
    PRINT USING "Locating right -3 dB frequency:  #####.##"; f0#
```

```
    LOCATE CSRLIN - 1

    IF INKEY$ = CHR$(27) THEN
      PRINT "Location of right -3 dB frequency aborted...
   "
      PRINT #1, ""
      PRINT #1, "Location of right -3 dB frequency aborted..."
      GOTO graphit
    END IF

    'now find t0
    CALL set.frequency(gnrl%, f0#)
    CALL measure(dmm1, dvm1#)
    CALL measure(dmm2, dvm2#)
    t0# = dvm2# / dvm1#
    IF (f2# - f0#) < (f0# - f1#) THEN
      f1# = f0#
      t1# = t0#
    ELSE
      f2# = f0#
      t2# = t0#
    END IF
    hg& = 100000# * t0# / h0#
  LOOP UNTIL hg& = 70711

  fcr# = f0#              'right cutoff frequency
  PRINT USING "The right cutoff frequency is #####.## Hz."; f0#
  PRINT #1, USING "The right cutoff frequency is #####.## Hz."; f0#
  IF g$ = "D" THEN fcn# = SQR(fcl# * fcr#)
  PRINT
  PRINT USING "fc = #####.##    Qo = ###.####"; fcn#; fcn# / (fcr#
- fcl#)
  PRINT #1, ""
  PRINT #1, USING "fc = #####.##    Qo = ###.####"; fcn#; fcn# /
(fcr# - fcl#)

END IF

graphit:

CLOSE #1                                 'output raw data file

CALL set.amplitude(gnrl, vrms, 0)        'turn off AC voltage from gen
                                         'to facilitate changing
                                         'components

PRINT
PRINT "Press any key to see graphs...";

WHILE INKEY$ = "": WEND


CALL GrfReset(4)
CALL SetAxGridVis(-1, 1)                    'enable grid lines for both
axes
CALL SetAxAuto(-1, 21)                      'auto-scale both axes every
time
CALL SetXDataType(4)
CALL SetYDataType(4)
CALL SetAxName(0, "log(f)")                 'set name of x-axis
CALL SetAxName(1, "dB")                     'set name of y-axis
CALL GrfCurv2D(x#(), y#(), np)              'x-y plot of dB vs. log(f)
CALL GrfReset(4)                            'clear the graphics display
```

```
CALL SetAxName(1, "Phase Angle")     'set name of y-axis
CALL GrfCurv2D(x#(), ph#(), nph)     'x-y plot of phase vs. log(f)
CALL GrfLReset(0, 0, 1, 2)           'return to text mode

CLS

Again:

PRINT "Again (y/n)?";
a$ = INPUT$(1)
CLS
IF UCASE$(a$) = "Y" THEN GOTO StartOfProg

REM   Now return to local control

CALL iolocal(dmm1%)
CALL iolocal(dmm2%)
CALL iolocal(osc1%)
CALL iolocal(cnt1%)
CALL iolocal(gnr1%)


END

REM $DYNAMIC
SUB ReportError
END SUB
```

3.9 Summary

In this chapter techniques were developed to measure active circuits. A second voltmeter was added to improve accuracy, along with the necessary changes to the code. A method of measuring the gain-bandwidth-product of an op-amp was also given. Finally a program to measure the second order filter parameters f_c, H_0, and Q_0 was designed.

CHAPTER 4

EXPERIMENTAL AND THEORETICAL VERIFICATION

4.1 Introduction

In this chapter the routines developed in the previous chapters are used to investigate and verify the properties of a second-order active band-pass filter presented in a recent paper.

4.2 The Filters to be Tested

The most difficult band-pass filter to measure by hand is a high-Q filter. In a recent paper [6] a new relocated Tow-Thomas filter was presented (Figure 4.1) derived from the original Tow-Thomas filter (Figure 4.2). It claims to be insensitive to the gain-bandwidth-product of the op-amps used provided that they are matched. We will use BODE6C.BAS and BODE7C.BAS to attempt to verify these results.

The new filter was generated from the original by a method called op-amp relocation. This method generates circuits that have the same ideal characteristics but may react differently to a real (non-ideal) op-amp. The basis upon which this method is founded is that the voltage across the terminals of an ideal op-amp is zero (in the presence of feedback). So if one terminal of an op-amp is grounded, the other is forced to zero potential. Thus any other ground connection in the circuit may be made to the op-amp node instead of ground. Consider the original Tow-Thomas filter. The non-inverting node of op-amp 2 is connected to ground. Thus there will be no change in the ideal characteristics of the circuit if it is connected to the inverting node of op-amp 1 or op-amp 3 instead. Now the non-inverting node of op-amp 3 may be connected to the inverting node of either op-amp 1 or op-amp 2. Certain connections can make the circuit unstable, however. There are software programs that perform op-amp relocation and check for stability; one was used to design the relocated filter and led to the addition of R_7 and R_8. The non-ideal characteristics (due to finite GBP, et. al.) of a relocated circuit may be dramatically different. In fact, the original Tow-Thomas filter is unstable above about 6 kHz, whereas the relocated filter is stable to a much higher frequency.

4.3 Measuring the Gain-Bandwidth-Products

The paper [6] calls for the use of the Texas Instruments TL084 quad op-amp. This integrated circuit contains four op-amps whose gain-bandwidth-products are supposed to be "reasonably" close. The data sheet for this IC claims a typical GBP of about 3 MHz. Measurements were taken for six TL084's with varying gain-bandwidth-products. The summary of the measured results are shown in Table 4.1.

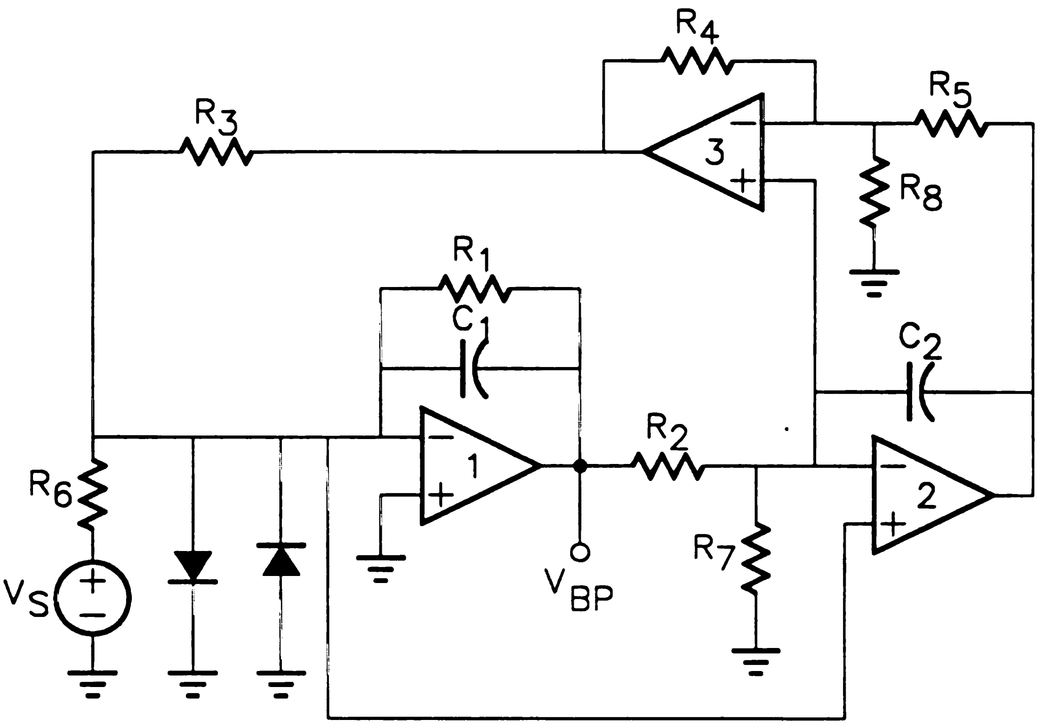

Figure 4.1 - Relocated Tow-Thomas Filter

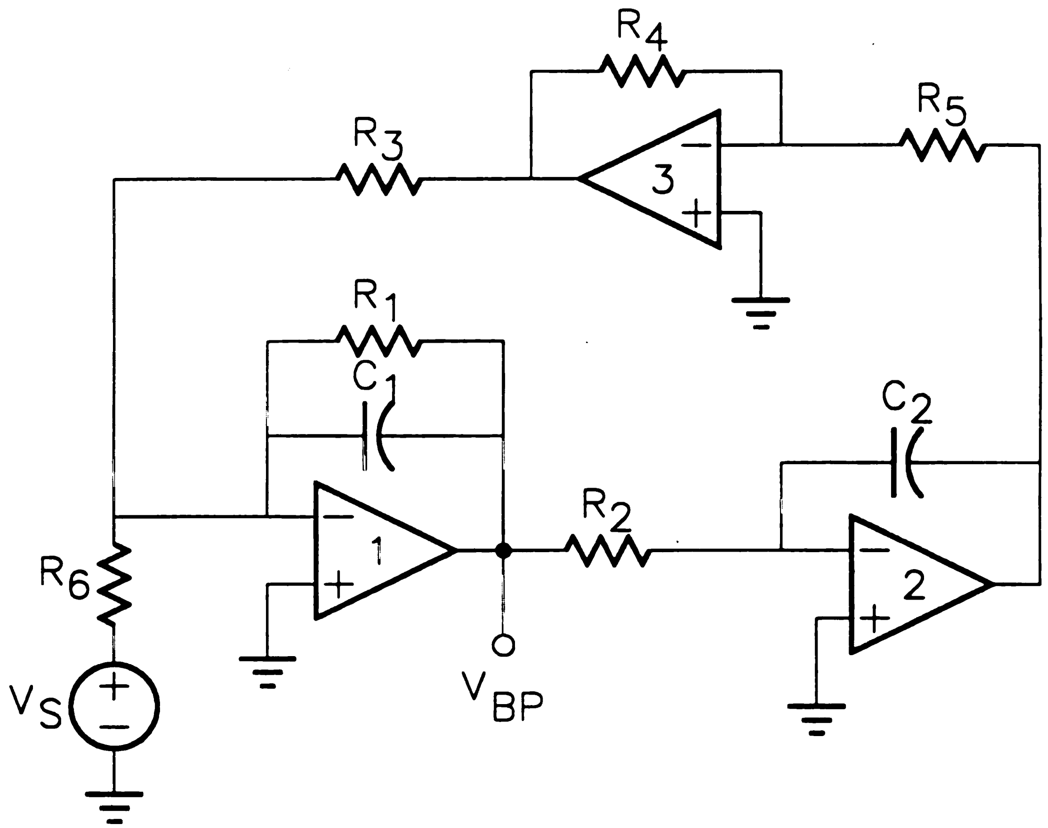

Figure 4.2 - Original Tow-Thomas Filter

Table 4.1 - Gain-Bandwidth-Products of Measured Op-Amps

Op-Amp	GBP 1 (MHz)	GBP 2 (MHz)	GBP 3 (MHz)	Room Temp. (°C)
1	2.872	2.941	2.968	24
2	2.971	2.876	2.909	24
3	2.761	2.903	2.690	24
4	2.979	2.930	2.857	24
16	3.356	3.613	3.479	23
21	2.878	2.716	2.805	24

While measuring these gain-bandwidth-products a crucial error was almost made. The chips were being measured by placing them into the circuit, applying power, taking the measurement, and then removing them. While doing this, it was noticed that subsequent runs of the program were producing significantly different results - as much as 10%. This was caused by the failure to allow the IC to reach its normal operating temperature. From the data sheet for the TL084, 168 mW is being dissipated by the biasing current when using ±15 V power supplies. Since the thermal resistance is 131.6° C/W, we can expect an increase of 22° C above ambient. Out of curiosity, several runs were performed at varying intervals after applying power. These results are shown in Table 4.2. They are not conclusive since they only apply to one op-amp on one IC, but they are interesting nonetheless. Based upon these results, all op-amps were allowed at least 15 minutes to reach operating temperature before any measurements were taken.

Table 4.2 - Gain-Bandwidth-Product vs. Time

Time After Power Application (min)	GBP (MHz)
1	2.915
5	2.882
10	2.878
15	2.877
20	2.876
25	2.876
45	2.878

4.4 Testing the Filter

Now we are ready to build and test the filter. A list of the component values recommended in the paper and the actual values used are shown in Table 4.3. The results of the measurements are summarized in Table 4.4. The filter performed as the authors claimed. Despite the varying gain-bandwidth-products of the op-amps used, the filter parameters remained relatively constant. The consistency of the center frequency was most impressive, varying only in the least significant digit. The maximum deviation in Q_0 among the filters, between op-amps 1 and 16, was only 2.94%, and the maximum deviation of H_0 was 2.49%. A typical set of generated Bode plots are shown in Figures 4.3 - 4.4.

Table 4.3 - Component Values for the Tow-Thomas Filter

Component	Recommended Value	Actual Value
R_1	100 kΩ	101.05 kΩ
R_2	4 kΩ	3.855 kΩ
R_3	1 kΩ	1.030 kΩ
R_4	1 kΩ	1.020 kΩ
R_5	1 kΩ	1.014 kΩ
R_6	100 kΩ	104.25 kΩ
R_7	2.47 kΩ	2.240 kΩ
R_8	1.175 kΩ	1.208 kΩ
C_1	7.958 nF	C_P = 7.870 nF R_P = 4.67 MΩ @ 10 kHz.
C_2	7.958 nF	C_P = 7.951 nF R_P = 4.58 MΩ @ 10 kHz.

Table 4.4 - Measured Results of Relocated Tow-Thomas Filter

Op-Amp	f_c (kHz)	Q_0	H_0	Room Temp. (°C)
1	10.12	49.61	0.9518	24
2	10.13	49.48	0.9501	24
3	10.13	49.60	0.9582	24
4	10.12	49.54	0.9539	24
16	10.13	48.15	0.9343	24
21	10.12	49.38	0.9573	23

As a measure of how good this circuit is, the original Tow-Thomas circuit was constructed and tested with two TL084s. Unfortunately it is unstable at 10 kHz, simply acting as an oscillator. The solution to this is to lower f_c by making the two capacitors larger. In this case the capacitors were increased to C_1 = 19.73 nF (R_P = 2.04 MΩ @ 4 kHz) and C_2 = 19.88 nF (R_P = 2.90 MΩ @ 4 kHz). The results are given in Table 4.5. In the next section, it is shown that when using an ideal op-amp model, Q_0 and H_0 are not a function of the capacitor values. Thus it is clear that in the original design Q_0 and H_0 are dependent upon the gain-bandwidth-products of the op-amps used.

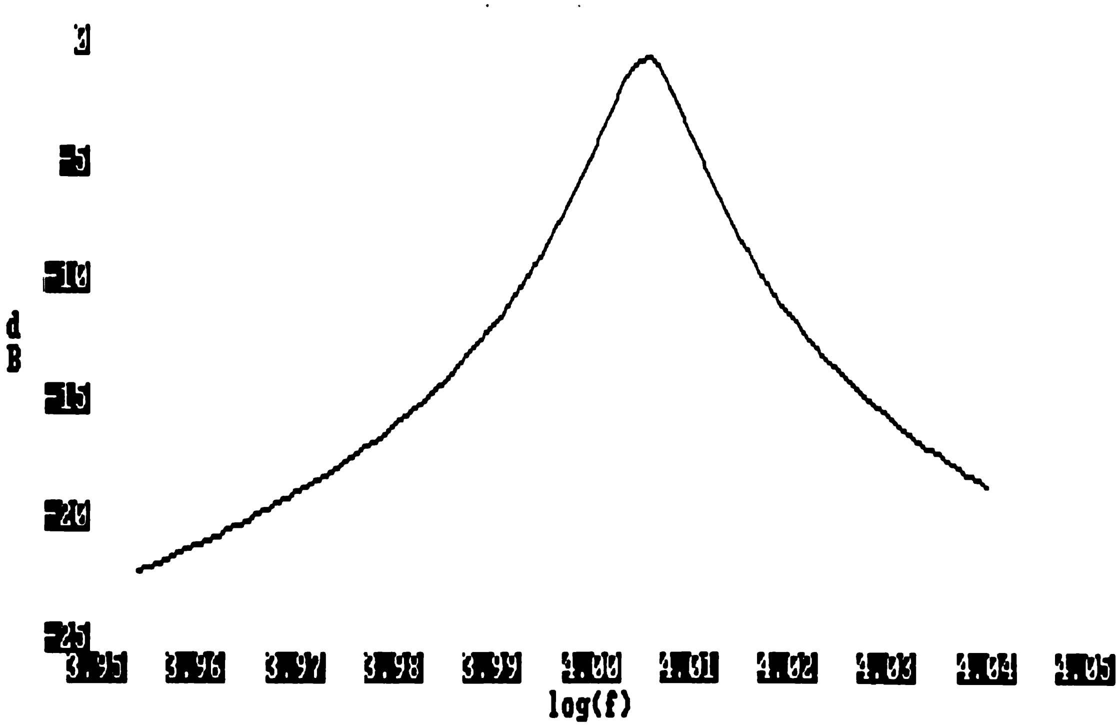

Figure 4.3 - Bode Magnitude Plot of Relocated Tow-Thomas Filter

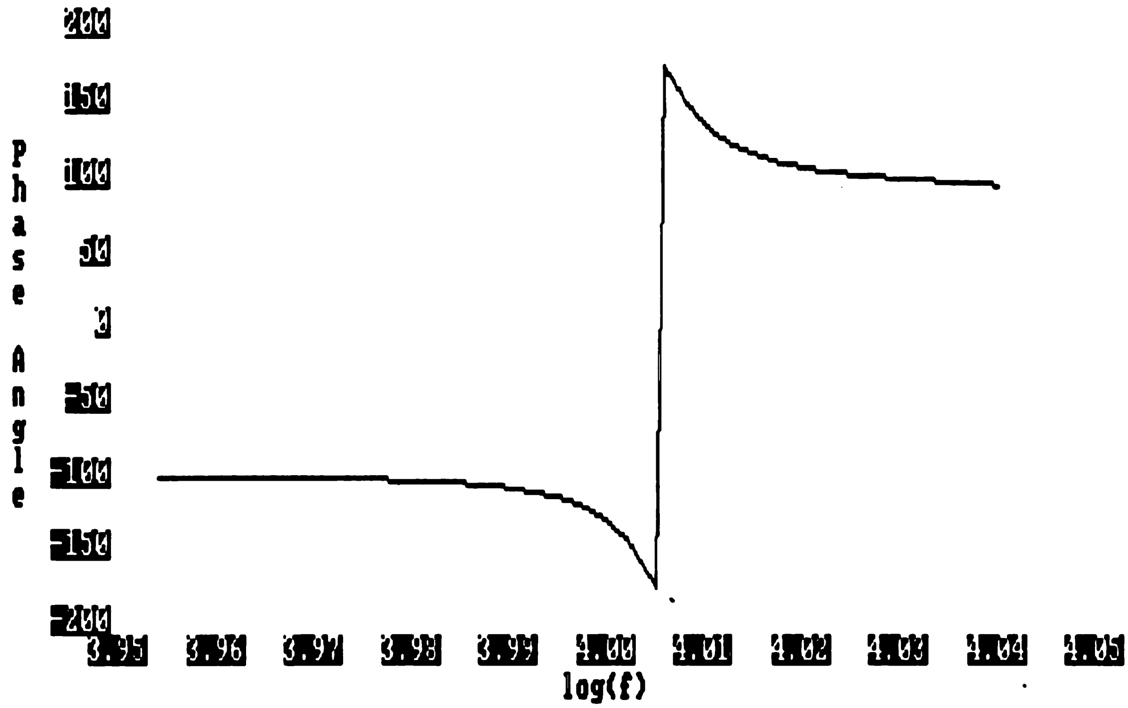

Figure 4.4 - Bode Phase Plot of Relocated Tow Thomas Filter

Table 4.5 - Measured Results of Original Tow-Thomas Filter

Op-Amp	f_c (kHz)	Q_0	H_0	Room Temp. (°C)
16	4.039	71.03	1.357	23
21	4.035	82.35	1.589	23

4.5 Theoretical Verification of Measured Results

There is a symbolic SPICE (Sspice) computer program under development at Michigan State University [7]. We will use this program to verify the measured results of the relocated Tow-Thomas filter. The input to

Sspice is similiar to PSpice, and is given below.

```
Relocated Tow-Thomas Filter
vs    10 0 ac
r1    1  2 101.05k
r2    2  3 3.855k
r5    4  5 1.014k
r4    5  6 1.020k
r3    1  6 1.030k
r6    10 1 104.25k
r7    3  0 2.240k
r8    5  0 1.208k
c1    1  2 7.870n
rp1   1  2 4.67MEG
c2    3  4 7.951n
rp2   3  4 4.58MEG
xoa1 0   1 7
xoa2 1   3 8
xoa3 3   5 9
r0    2  7 200
r0    4  8 200
r0    6  9 200
.end
```

Note the addition of the r0 resistors, representing the output resistance of the TL084. They are included because later in this section we will show that they affect the filter parameters in the non-ideal case. These values do not affect the filter parameters in the ideal case so they may be ignored for now. Also note that since an ideal op-amp model is being used, the formulas found for this filter will also apply to the original Tow-Thomas (R_7 and R_8 do not appear in the formulas for the parameters). The Sspice output file is given below.

```
Relocated Tow-Thomas Active Filter

[0  ] [Y 1,1   0       -G3      0       0       0       -G6    ][V2 ]
[0  ] [Y 2,1   0       0        -G0     0       0       0      ][V4 ]
[0  ] [-G2     Y 3,2   0        0       0       0       0      ][V6 ]
[0  ]=[0       Y 4,2   0        0       -G0     0       0      ][V7 ]
[0  ] [0       -G5     -G4      0       0       0       0      ][V8 ]
[0  ] [0       0       Y 6,3    0       0       -G0     0      ][V9 ]
[1  ] [0       0       0        0       0       0       1      ][V10]

Y( 1,1 ) = -sC1-GP1-G1
Y( 2,1 ) = +sC1+GP1+G2+G1+G0
Y( 3,2 ) = -sC2-GP2
Y( 4,2 ) = +sC2+GP2+G5+G0
```

```
Y( 6,3 ) = +G4+G3+G0
*Ignore nodes 11 and higher if present. They are used for internal
numbering.

Numerator of: v2

TERMS SORTED ACCORDING TO POWERS OF s

s**1 terms:

 + sC2*G6*G4

s**0 terms:

 + GP2*G6*G4

************************************************

NUMERICAL VALUE OF ABOVE SYMBOLIC RESULT

+ 7.47731e-017 * s**1 + 2.05333e-015 * s**0

************************************************

Denominator of: v2

TERMS SORTED ACCORDING TO POWERS OF s

s**2 terms:

 - sC2*sC1*G4

s**1 terms:

 - sC2*GP1*G4 - sC2*G4*G1 - sC1*GP2*G4

s**0 terms:

 - GP2*GP1*G4 - GP2*G4*G1 - G5*G3*G2

************************************************

NUMERICAL VALUE OF ABOVE SYMBOLIC RESULT

- 6.13474e-020 * s**2 - 8.04948e-017 * s**1 - 2.48373e-010 * s**0

************************************************


Relocated Tow-Thomas Active Filter

SECOND ORDER FILTER PARAMETERS:

Qo is:

SQRT{( + C2*C1)*( + GP2*GP1*G4 + GP2*G4*G1 + G5*G3*G2)}
-------------------------------------------------------------------
( + C2*GP1 + C2*G1 + C1*GP2)*SQRT{ + G4}

 = 48.4933
```

```
Wo**2 is:
( + GP2*GP1*G4 + GP2*G4*G1 + G5*G3*G2)
------------------------------------------------
( + C2*C1*G4)
fo = 10126.8Hz
*****************************************
```

The important parameters in the Sspice output have been highlighted. Sspice does not give H_0, so a hand calculation was done using the given transfer function. This calculation yields $|T(j2\pi \cdot 10126.8)| = 0.928918$.

Sspice also has the ability to predict the shifting of f_c and Q_0 due to finite gain-bandwidth-products. It was stated earlier that the output resistance also effects these shifts. Thus it is necessary to measure those values, and they are given in Table 4.6. The Sspice results are tabulated in Table 4.7.

Table 4.6 - Measured Output Resistance of Op-Amps[1]

Op-Amp	R_{OUT} 1 (Ω)	R_{OUT} 2 (Ω)	R_{OUT} 3 (Ω)
1	285.0	288.6	306.8
2	273.8	272.3	267.4
3	279.3	340.6	296.2
4	295.5	287.5	295.5
16	263.5	325.4	256.0
21	312.5	293.9	287.5

[1]This data was collected by Bassam Hindeleh

Table 4.7 - Parameter Shifting Predicted by Sspice

Op-Amp	$\Delta f_c/f_c$	$\Delta Q_0/Q_0$
1	-0.000916259	0.033277500
2	-0.001009520	0.020787300
3	-0.001061240	0.072818000
4	-0.001027560	0.061305000
16	-0.000740414	0.000161558
21	-0.001259700	0.022055900

Table 4.8 - Predicted Parameters and Errors

Op-Amp	Prediced f_c (kHz)	% Error	Predicted Q_0	% Error
1	10.12	0.000	50.11	1.008
2	10.12	0.099	49.50	0.040
3	10.12	0.099	52.02	4.879
4	10.12	0.000	51.47	3.896
16	10.12	0.099	48.50	0.727
21	10.11	0.099	49.56	0.365

The experimental results agree well with theory for the most part. The most glaring inconsistencies are the Q_0 errors for op-amps 3 and 4. In section 3.7 it was noted that the results of the program are consistent to within one count at the fourth significant figure. In this case, a change in that last significant figure of one count in one of the cutoff frequencies is enough to cause the error in the calculation of Q_0.

4.6 Summary

In this chapter the previously developed routines were tested on a high-Q band-pass filter presented in a recent paper. Sspice was used to provide theoretical verification of the measured results. The results were found to compare favorably with theory.

CHAPTER 5

CONCLUSIONS AND FUTURE RESEARCH

5.1 Conclusions

The goal of this document was to indicate an approach to the development of software using TestTeam to automate precision measurements. TestTeam has been found to be a powerful and useful tool in this study. In Chapter 2, routines were developed using one voltmeter to analyze passive networks. In particular, a program to read data and create Bode magnitude and phase plots and -3 dB frequencies was designed. In Chapter 3, a second voltmeter was added to correct errors due to the output impedance of the function generator. Routines were then developed to analyze active circuits, such as finding the gain-bandwidth-products of op-amps and finding the second-order parameters f_c, H_0, and Q_0 of band-pass and band-stop filters. Although not explicitly developed for this purpose, the routines presented could be altered to measure the aforementioned parameters of low and high-pass filters also. Finally in Chapter 4 the routines were successfully tested on a high-Q band pass filter.

5.2 Future Research

The next logical step in the progression of automated measurements is to move from steady state to transient measurements. Measurements such as step response and calculations such as slew rate, step response, and FFTs can be made from digitized oscilloscope data. In order to use TestTeam with an oscilloscope, another driver package is required: The PM2235 Oscilloscope Drivers. An attempt to write a program to capture oscilloscope data was made. However when feeding the same signal to both channels and performing a measurement, there was a voltage shift between the two channels, and neither one had a correct ground. Attempts with other oscilloscopes yielded different shifts, but no correct results. Phone consultations with a Fluke engineer have not provided a satisfactory solution as of this writing.

The code written to capture the oscilloscope data is shown below. The header on this program is slightly different due to a replacement for TESTTEAM.BAS supplied with the oscilloscope drivers.

```
'************* PM2233 INSTRUMENT DRIVERS INITIALIZATION PROCEDURES
'* File name : SCOPE3C.BAS                                      *
'****************************************************************

'  $DYNAMIC $INCLUDE: '\drivers\DRIVERS.INC'
'  $DYNAMIC $INCLUDE: '\lw\include\graphics.inc'

'****************************************************************
'*                           DRIVERS INITIALIZATION
'****************************************************************

CLEAR , , 10000          ' Reserve 10K bytes stack
Drivers.Initialization
GOTO Application

Drivers.Report.Error:
  IF (ERR <> 11) THEN
    PRINT : PRINT "BASIC error"; ERR; "in line"; ERL
  ELSE
    ReportError
  END IF
  RESUME NEXT
'END Drivers.Report.Error

Application:
' USER APPLICATION STARTS HERE
```

```
DIM    SHARED    ca%(512),    cb%(512),    ma#(512),    mb#(512),
reg.settings#(89), t#(512)
CALL getmem(20000)
overlay.memory.size& = 98304
memory.size& = SETMEM(640000)
memory.size& = SETMEM(-overlay.memory.size&)

CLS


PRINT


StartOfProg:

PRINT "Into my routine..."

CALL Set.Frequency(GNR1, 1000#)
CALL Set.Amplitude(GNR1, PP, 8!)
CALL Set.Waveform(GNR1, triangular)


PRINT "Calling Auto Set..."
CALL Auto.Set(osc1)
CALL set.polarity(osc1, chall, positive)
PRINT "Setting DC coupling..."
CALL Set.Coupling(osc1, chall, DC)
PRINT "Setting channel format..."
CALL set.chan.trans.format(osc1, 1, real.data, 512)
PRINT "Setting horizontal mode..."
CALL Set.Horizontal.Mode(osc1, SINGLE.SHOT)

PRINT "Measuring scope data..."
CALL measure.chan.data(osc1, accu, cha, ca%())    'measure forces
trigger
CALL read.chan.data(osc1, accu, chb, cb%())      'read does not

PRINT "Scaling data..."
CALL get.reg.settings(osc1, accu, reg.settings#())
CALL scale.chan.data(osc1, cha, reg.settings#(), ca%(), ma#())
CALL scale.chan.data(osc1, chb, reg.settings#(), cb%(), mb#())

FOR i = 0 TO 512
  t#(i) = i
NEXT i


graphit:

CALL Set.Amplitude(GNR1, VRMS, 0)   'turn off AC voltage from gen

PRINT
PRINT "Press any key to see graphs...";

WHILE INKEY$ = "": WEND


CALL GrfReset(4)
CALL SetAxAuto(-1, 21)                  'auto scale both axes
CALL SetPlotMode(0)                     'no wait for keypress
CALL SetXDataType(4)
CALL SetYDataType(4)                    'eight-byte floating point
CALL GrfCurv2D(t#(), ma#(), 512)
```

```
CALL SetPlotMode(2)                    'wait for keypress
CALL GrfCurv2D(t#(), mb#(), 512)
CALL GrfLReset(0, 0, 1, 2)             'return to text mode

CLS

Again:

PRINT "Again (y/n)?";
a$ = INPUT$(1)
CLS
IF UCASE$(a$) = "Y" THEN GOTO StartOfProg

REM   Now return to local control

CALL Iolocal(DMM1%)
CALL Iolocal(DMM2%)
CALL Iolocal(osc1%)
CALL Iolocal(CNT1%)
CALL Iolocal(GNR1%)


END

REM $DYNAMIC

DEFSNG A-Z
SUB Drivers.Initialization
'*******************************************************************
'*                    DRIVERS INITIALIZATION PROCEDURE
'*******************************************************************

  ON ERROR GOTO Drivers.Report.Error
  CLS
  CALL Reset.Config
  overlay.memory.size& = 141312
  memory.size& = SETMEM(640000) ' Try to get all memory now
available
  memory.size& = SETMEM(-overlay.memory.size&) ' Free memory for
instrument overlays
  PRINT "Initializing Multimeters..."
  CALL Config(DMM1, "8840A,N DMM1,A 705")
  CALL Config(DMM2, "8840A,N DMM2,A 706")
  PRINT "Initializing Function Generator..."
  CALL Config(GNR1, "PM5193/V2.5,N GNR1,A 707")
  PRINT "Initializing Oscilloscope..."
  CALL Config(osc1, "PM3365/V07V04,N OSC1,A 708")
  PRINT "Initializing Counter..."
  CALL Config(CNT1, "PM6666/22,N CNT1,A 710")
  PRINT "Setting up defaults..."
  CALL Allinit(actual.SET)
END SUB 'Drivers.Initialization

SUB ReportError
'*******************************************************************
'*                       DEFAULT ERROR HANDLER
'*******************************************************************

  GPIB.ERR = Err.Num
  IF (GPIB.ERR <> 0) THEN
    GPIB.STAT = Err.Stat
    GPIB.ERR$ = Err.Str$
    GPIB.GLBSTAT = Glb.Stat
```

```
    GPIB.GLBERR$ = Glb.Str$
    PCIB.ERR = IOErr.Num
    PCIB.ERR$ = IOErr.Str$
    status = GPIB.STAT
    SELECT CASE status
      CASE 0: PRINT "IOdrivers error in line"; ERL: PRINT GPIB.ERR$
      CASE 1: PRINT "Warning in line"; ERL: PRINT GPIB.ERR$
      CASE 2: PRINT "Error in line"; ERL: PRINT GPIB.ERR$
      CASE 3: PRINT "Fatal error in line"; ERL: PRINT GPIB.ERR$: END
    END SELECT
  END IF
END SUB 'ReportError
```

The output of this program with the same signal fed to both channels is shown in Figures 5.1 - 5.3. It is clear from these plots the offset of the two channels is not the same, nor is it consistent among different waveforms.

If the two outputs were correct, one measurement which could be made is the slew rate of an op-amp. The first step is to read the oscilloscope data when the time base of the oscilloscope was set see the slewing. Then two points could then be used to calculate the slope, perhaps at 30% and 80% of the peak voltage values. The sampling rate can be read from the oscilloscope using a driver, and the calculation of the slew rate follows.

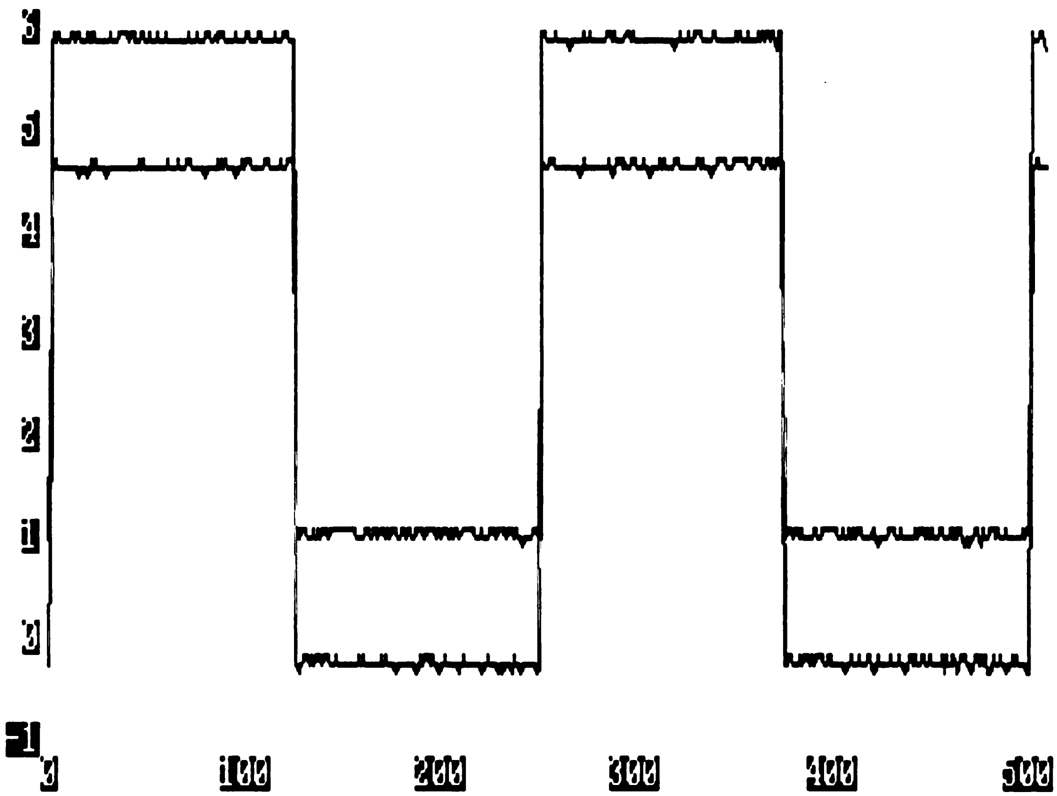

Figure 5.1 - Oscilloscope Output for Positive Pulse

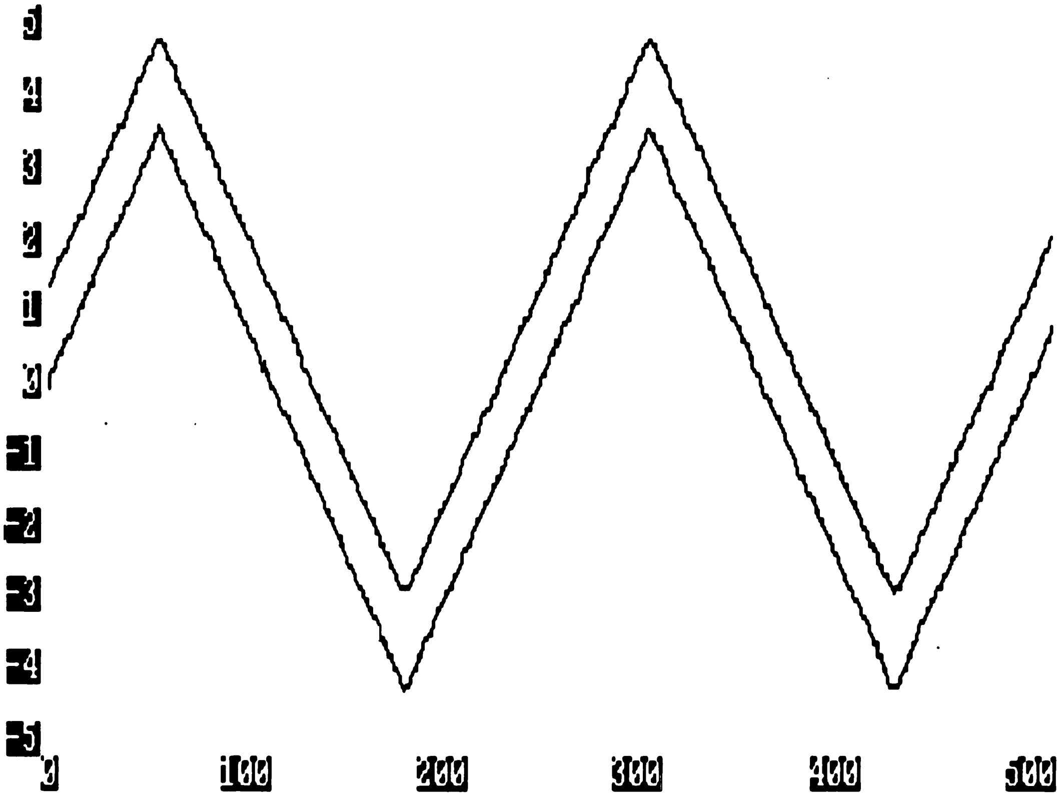

Figure 5.2 - Oscilloscope Output for Triangle Wave

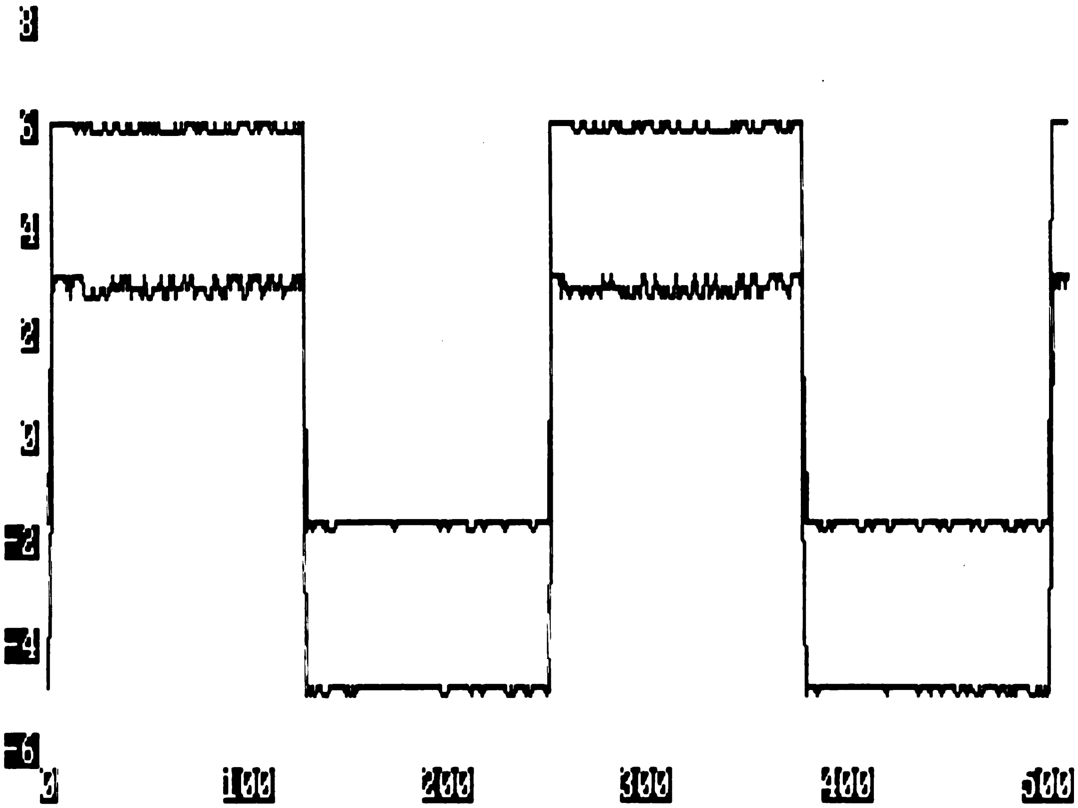

Figure 5.3 - Oscilloscope Output for Square Wave

LIST OF REFERENCES

LIST OF REFERENCES

[1] "Getting Started with TestTeam," National Instruments Corporation, 1989.

[2] "TestTeam User Manual," National Instruments Corporation, 1988.

[3] "8840A Digital Multimeter Instruction Manual," John Fluke Mfg. Co., Everett Washington, 1985.

[4] "Programmable Timer/Counter PM 6666 Operator's Manual," Philips Export B.V., The Netherlands, 1988.

[5] G.M. Wierzba, Designing Electronic Circuits Using Analog ICs, Texas Instruments Inc., Dallas, TX and Prentice-Hall, Englewood Cliffs, NJ, in press.

[6] G.M. Wierzba and J.A. Svoboda, "An op-amp relocated bandpass filter with zero center frequency sensitivity to gain-bandwidth-product," Invited Paper, Proc. of the 29th Midwest Symposium on Circuits and Systems, Lincoln, NE, pp. 28-32, August, 1986.

[7] G.M. Wierzba, V. Joshi, A. Srivastava, and J.A. Svoboda, "Sspice - Symbolic SPICE for Linear Active Circuits," Invited Paper, Proc. of the 32nd Midwest Symposium on Circuits and Systems, Urbana, IL, pp. 1197-1201, August, 1989.