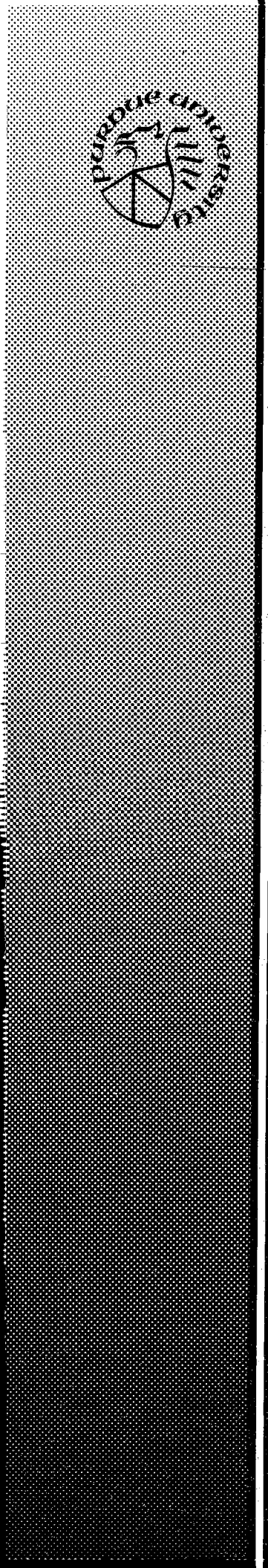
4-1-1984

The One Dimensional Signal Processing Laboratory. Hardware and Command Documentation

Jon C. Dattorro
Purdue University

John C. Stapleton
Purdue University

Steven C. Bass
Purdue University



ODSP - The One Dimensional Signal Processing Laboratory

Hardware and Command Documentation

Jon C. Dattorro
John C. Stapleton
Steven C. Bass

April 1984

TR-EE 84-21

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

The One Dimensional Signal Processing Laboratory

Hardware and Command Documentation

Jon C. Dattorro

John C. Stapleton

Steven C. Bass

April 1984

School of Electrical Engineering

Purdue University

West Lafayette, Indiana 47907

Table of Contents

Chapter 1
Hardware Description

Hardware Description

The purpose of this hardware description is to introduce the features of the equipment in the One Dimensional Signal Processing Lab and to illustrate the use of some of the standard lab setups. An overview of the lab equipment is presented. Interconnections between the lab terminals, the PDP 11-40 in the lab, and ECN are described. Detailed descriptions of the use of the terminals in controlling signal processing and file transfer software with the 11-40 are given. The basic equipment configurations for common signal processing tasks are shown. This introduction covers only a small part of the capabilities of the lab. The lab users are encouraged to extend these ideas and make use of the full flexibility of the lab for signal processing applications.

1. Equipment

1.1 PDP 11-40 Computer

The PDP 11-40 is the heart of the signal processing lab. Real time samples enter the computer by direct memory access (DMA) from the Kluge (FIFO queuing). Samples entering the 11-40 are typically routed immediately to disk store by DMA action. The DMA is more fully described in Appendix A. The software operating system (see "Switch II and DMC User's Manual"). provides file maintenance, signal processing operations, and file transfer between ECN and the 11-40 via a dedicated one megabaud communications line.

1.2 Disk Drives

The lab has two disk drives. The CDC 9766 Drive has 239 Mbytes available and is organized into 12 logical devices. The Aries Drive has 10 Mbytes organized into 5 Mbytes of disk cartridge and 5 Mbytes of system software. For further information on the organization of the drives see Appendix B.

1.3 Lear Siegler Adm3a Terminal

The Adm3a terminal is available for use with ECN or in controlling the PDP 11-40.

1.4 Tektronix 4025 Graphics Terminal

The 4025 programmable graphics terminal also can be used with ECN. It is the preferred terminal for controlling the 11-40 since many of the signal processing commands in the 11-40 software make use of its graphics and smart terminal capabilities.

1.5 Stereo Equipment

A Sony STR-V55 FM Stereo/FM-AM Receiver, an Akai Reel to Reel Tape Deck, a Technics SL-7 Direct Drive Automatic Turntable, and a Harman Kardon CD301 Cassette Deck are available in the lab. User's manuals are found in the top drawer of the file cabinet in the lab.

1.6 Test Equipment

Tektronix 2213 and RM503 Oscilloscopes, Hewlett-Packard 3580A Spectrum Analyzers, and Wavetek 185 Signal Generators are available in the lab. User's manuals are found in the top drawer of the file cabinet in the lab.

1.7 Rockland 852 Filter

The Rockland filter has two 8^{th} order high or low pass filters. These can be cascaded for bandpass or bandreject filters, or higher order high or low pass filters.

1.8 Bell Digital Filter Unit

The Bell Unit, constructed for Purdue by Bell Laboratories, is a serial, time-shared digital filter used as a teaching tool. It provides real-time programmable signal processing. The 11-40 is used in conjunction with eight terminals, each with an A/D and a D/A converter and a keypad for programming the filter.

1.9 Kluge Digital Filter

Much of the following information describing the Kluge is a direct copy or paraphrase of sections in the "Maintenance Manual for the Purdue Digital Filter", written by Dean Gibson, and available in the top drawer of the file cabinet in the lab. The information is included to illustrate the capabilities of the Kluge. The manual should be used during actual operation of the Kluge. The Kluge is a digital filter designed and built at Purdue. It contains a number of multiplier, adder, register, and memory modules, along with A/D and D/A converters and a complete nonrecursive filter. All connections to each module are brought out to sockets on the front panel. A digital filter is set up by patching the different modules together.

A/D Converter

The A/D converter samples the incoming analog signal to generate a sequence of 12-bit two's complement numbers at its output sockets. Two analog inputs are provided. The A/D converter can be time multiplexed between two different analog signals, or two signals can be summed and sampled. The A/D converter has an analog range of +/- 10 V. The output bits represent short integers in the range of -2048 to 2047. The 12 bits of output information are sign extended on the connector so the A/D is compatible with 12 bit or 15 bit cords.

D/A Converters

The Kluge has two D/A converters. Each converter expects a 12-bit two's complement input, representing an integer between -2048 and 2047. Each drives a 4^{th} order lowpass filter whose cutoff frequency is adjustable from the front panel. Pins 1 through 12 on the input connector are used for the 12 bit information; pins 13 through 15 are not connected. When the output of the nonrecursive filter or a 15 bit adder or multiplier is to be reconstructed, a 15 bit to 12 bit conversion cord is necessary if the number of active bits exceeds 12. These cords are identified by their black stripes.

Amplifier and Attenuator

An amplifier and attenuator are provided on the Kluge for controlling analog signal amplitudes. They are not short circuit protected. If an output is shorted to ground the output op amp will burn out and will have to be replaced.

Nonrecursive Filter

The nonrecursive filter is used to implement in real time those digital filters which can be described by the discrete convolution

$$y_k = \sum_{i=0}^{n} h_i x_{k-i}$$

where x_k and y_k are the input and output sequences respectively. The impulse response of the filter is h_k. The filter can be up to 256 taps in length. A 15 bit to 12 bit conversion cord is necessary if the number of active bits to be sent to the D/A exceeds 12.

Adders, Multipliers, Registers

The bottom row of Kluge multipliers, adders, and delays are 12 bit processors. The top row consists of 15 bit processors, and a conversion cord or the bit access port must be used to interface these processors with the converters.

Clock and Control Shift Register

The clock and the control shift register provide the control pulses for the other modules on the Kluge. The main reset switch must be pressed to start the clock when the Kluge is first powered up. The clock is a variable frequency oscillator which will provide an output in the range of .5 Hz to 5 MHz. The output of the clock is made available to the other modules through the control shift register. The register has two parts: a counter and the shift register proper. The counter generates a reset pulse every N clock pulses (as set by the thumbwheel switch marked SHIFT REGISTER LENGTH). At that time, the register is cleared and pulses are introduced at steps 1, N+1, 2N+1, and so on. A pulse moves down the shift register one step for each clock pulse until a reset pulse is generated. The cycle then repeats. Each of the 96 steps in the shift register is accessible at the front panel. For example, if the thumbwheel is set to 10, a logical one appears at outputs 1, 11, 21, and so on. After the next clock pulse the one appears at outputs 2, 12, 22, and so on (outputs 1, 11, 21, ... return to zero). If the frequency of the clock is 160 KHz, each of the outputs can be used as 16 KHz clocks with different phases depending on the relative position each has in the shift register.

2. The Line Select Box

The Line Select Box, located next to the terminals in the lab, is used to control the interconnections between ECN, the terminals, and the 11-40. The interconnections are shown in Figure 1. Figure 2 shows the connection layout and the schematic of the Line Select Box. The following options are available.

Line Select Box		
Switch Position	Function 1	Function 2
1	4025 to ECN 9600 baud (tty45)	ECN to 11-40 2400 baud (ea tty1b)
2	4025 to 11-40 2400 baud	ECN to 11-40 9600 baud (ea tty1b)
3	4025 to ECN 9600 baud (tty45)	ECN to 11-40 9600 baud (ea tty1b)
4	4025 to 11-40 2400 baud	ECN to 11-40 9600 baud (ea tty45)

In position 1 the 4025 is a terminal on ECN (tty45). In position 2 the 4025 is used to control the 11-40. In position 3 the 4025 is a terminal on ECN (tty45). In position 4 the 4025 is used to control the 11-40 during a download of software from ECN to the 11-40. The Adm3a is connected to ECN tty1b for all switch positions.

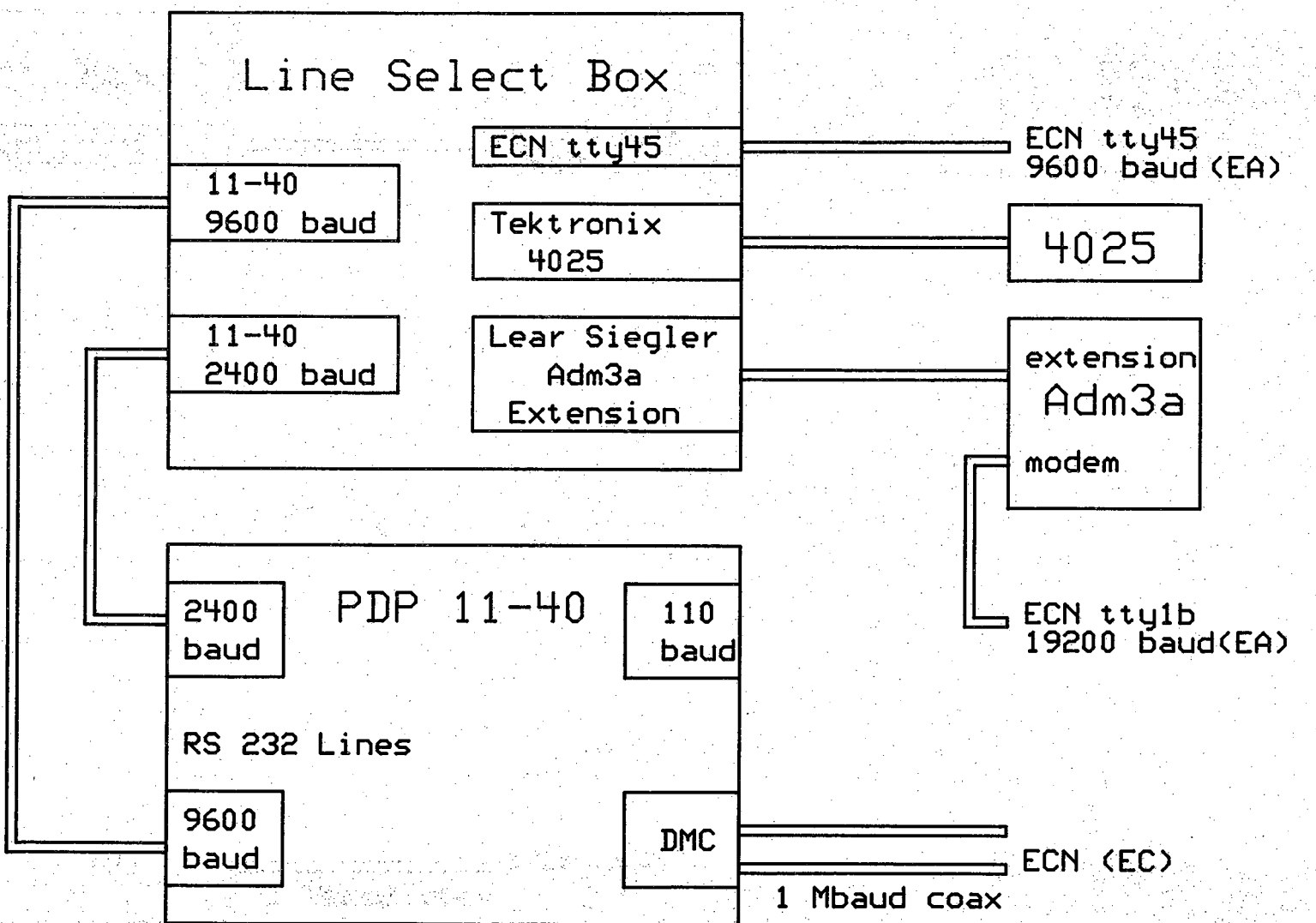
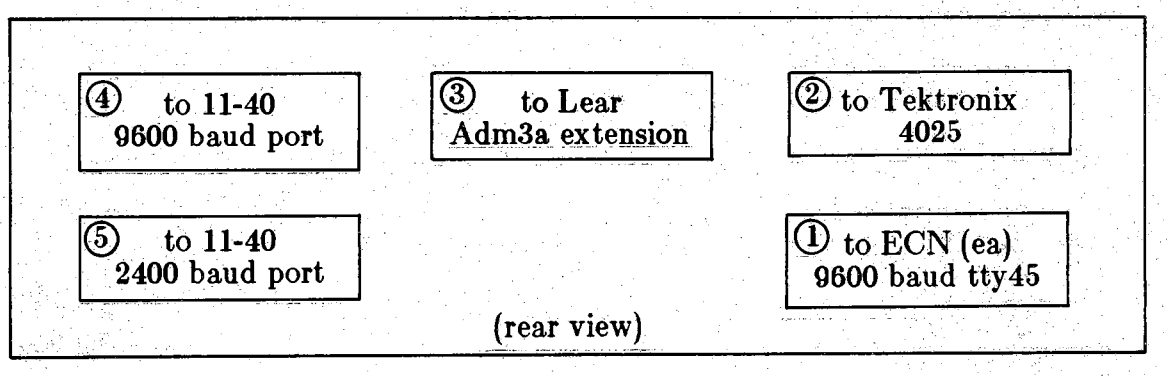


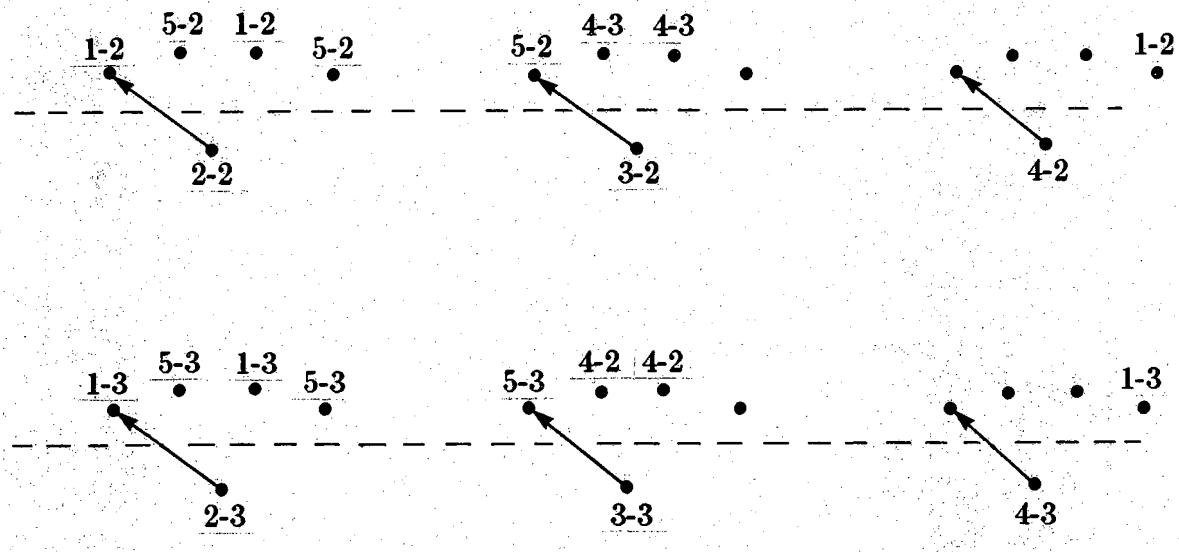
Figure 1. Connections between Line Select Box, ECN, 11-40, and Lab Terminals



Functions:

Switch pos.	Function 1	Function 2
1	① ↔ ②	③ ↔ ⑤
2	② ↔ ⑤	③ ↔ ④
3	① ↔ ②	③ ↔ ④
4	② ↔ ⑤	① ↔ ④

Notation for wiring: 1-2 → connector 1, pin 2
connectors ① and ⑤ are male; ② ,③ and ④ are female



(pin 7 of all connectors wired together)
switch is 6 pole - 4 pos. rotary switch

Figure 2. Internal Connections and Layout of the Line Select Box

3. The Tektronix 4025 Graphics Terminal

The 4025 programmable terminal is the preferred controller for the 11-40 when the graphics and other "smart" features of the terminal are needed. Many of the software commands in the Switch II program depend on these features. The manual for the 4025 is kept in the top right drawer of the desk on which the 4025 is located.

3.1 The Command Character

Once the 4025 is on (the switch, brightness, and focus are on the lower right side of the terminal), the current command character needs to be checked. The command character is typically the "@" or "!" character and precedes any terminal commands, such as changing the baud rate of the terminal. The current command character is found by pressing SHIFT STATUS (this key is in the upper right corner of the terminal). Something like U @1983 will appear. The second character ("@" here) is the command character. The command character may be changed with the "com" command. For example,

```
@com !
```

changes the command character from "@" to "!". Note that Switch II and DMC use the "@" command character. If the current command character is "!", Switch II or DMC will automatically change it to "@". Other command characters are not changed by Switch II and DMC.

3.2 Baud Rate

For communication with ECN, the baud rate must be 9600, and for communication with the 11-40, the baud rate must be 2400. To change the baud rate type

```
@baud xxxx
```

where xxxx is the desired baud rate (2400 or 9600) and "@" is the current command character. The Line Select Box is set to 1 for use with ECN and 2 for use with the 11-40.

3.3 Timeout

When using the 4025 with the 11-40 the user does not need to be logged in on an ECN computer. However, it is often convenient to be logged in for one reason or another. The default time for automatic logoff when no commands are issued to the ECN machine is usually 10 minutes. To change the timeout interval type

$TIMEOUT=xxxx

where $ is the ECN prompt and xxxx is the time in seconds (3600 is a one hour timeout).

3.4 Qplotting

The graphics capabilities of the terminal are used by many of the 11-40 signal processing programs. In addition, the 4025 can be used for qplotting on ECN. To set the terminal up for a qplot type

@wor 33
@gra 1,35
@shr

and then issue the qplot command using the "-t" option. To return the terminal to normal, type

@wor 0

These commands can be executed automatically using the programmable keys as discussed in the 4025 manual.

3.5 Editing with Vi

The 4025 can be set up for editing using vi. The command character must first be set to "!". Then use the "tset" command as follows.

$ TERM='tset -Q - 4025'
$ export TERM

As always, the lines must be typed as shown, without spaces next to the = sign. The two lines carry out the initialization of the 4025 for vi. No messages should appear if these steps are followed. The command character is automatically changed from "!" to "^_" (control-underscore, which appears as U_S when SHIFT STATUS is pressed). Vi may now be used for editing. After the editing session with vi is complete, the command character may be changed back to "!" or "@". Vi does not work well for the 4025 on the ea machine.

4. The Lear Siegler Adm3a

The Adm3a is normally connected directly to ECN (tty1b) via the MODEM port and to the Line Select Box via the EXTENSION port. The Adm3a can be used to control the 11-40 in the following way. The plug normally connected to the EXTENSION port of the terminal is moved to the MODEM port, replacing the ECN connection which is already there. The other end of the extension cable should already be plugged into the Adm3a port on the Line Select Box. The Line Select Box is set to position 1. The DIP switches on the left side of the terminal are set as follows.

ECN	11-40
Red	Red
	*	*	
	*		*
*		*	
	*		*
	*		*
*		*	

Blue	Blue
*		*	
*		*	
	*		*
*			*
	*		*
	*		*
	*	*	
	*		*
	*		*
	*		*
	*		*
	*		*
	*		*
	*		*
	*		*

The * represents the side of the switch that is pressed down. The terminal is

then set up for use with the 11-40. Note that the graphics capabilities of the signal processing commands in the 11-40 software cannot be used with the Adm3a.

5. Running the 11-40 Software - SW2, DMC, and BELLDF

The software written for the PDP 11-40 in the lab includes many signal processing and file manipulation routines. The following instructions detail the procedure for bringing up the 11-40 for use with the programs. Appendix C contains information on compiling and downloading the programs (not necessary in normal use).

5.1 11-40 Power Up Procedure

1) Before any power is applied, the Aries Disk Drive (above the 11-40) should already have its switches set as follows:

Disk Power	Prot Fixed	Prot Rmvl	Disk Drive
on	on	on	off

The disk power is turned on with the same switch that turns on the 11-40. A disk cartridge should already be in the disk drive.

The 11-40 is turned on by rotating clockwise the power switch key to the first point at which the power lights come on. Rotating the switch to the first stop or the lock position puts unnecessary pressure on the circuit board behind the front panel and should be avoided.

2) Wait for the SAFE light on the disk drive to come on. This takes approximately 10 seconds. When it is on, the disk cartridge may be changed if desired.

4) After the desired disk cartridge is in the drive, turn the Disk Drive switch on and wait for the READY light. This takes approximately 90 seconds. The Prot Rmvl switch is the write protect switch and should be left on unless your needs require writing to your disk. The 11-40 is then ready to be accessed by the Tektronix 4025 or Adm3a terminal.

5) The 4025 or Adm3a should be set up as described in the appropriate section of this handout ('The Tektronix 4025 Graphics Terminal' or 'The Lear Siegler Adm3a') for use with the 11-40. Logging on is not necessary.

For the 4025, type @baud 2400 (be sure to use the correct command character). This sets the baud rate of the 4025 to match that of the 11-40. Set the switch on the Line Select Box next to the 4025 to position 2. This changes the 4025 connection from ECN to the 11-40.

If you are logged onto ECN you will not be logged off for 10 minutes (unless you have changed your TIMEOUT).

For the Adm3a, the baud rate needs to be set with the DIP Switches and the ECN cable needs to be replaced with the 11-40 Line Select Box cable.

6) Press HALT (down) and then START (down) to clear the 11-40. Set the front panel switches of the 11-40 to octal 773110 (left to right). One is up and zero is down. Press LOAD ADDRESS (down), then ENABLE (up), and then START (down). An introductory message should appear on the terminal screen. Type control-C to stop the message, if desired.

5.2 SW2 and DMC

The SW2 and DMC programs act as shells on the 11-40 to execute many signal processing and file manipulation commands. A listing of the commands and instructions for their use are found in the documentation entitled "Switch II and DMC User's Manual." Most commands available in one program can be found in the other. Here are the commands that appear in only one of the two programs.

SW2	DMC
af	ul
ab	link
in	get
key	put
mf	cpu
out	con
prt	
r	
rnd	
sc	
sic	

The following instructions explain how to bring up the programs.

7) After the prompt (a lone period), type

R SW2
or R DMC

to start the appropriate program. SW2 returns a # prompt and

DMC returns a % prompt. If an error in starting the program is made, an error message such as NEXFIL appears and another try can be made.

8) To exit SW2 or DMC type log. To begin another program, return to step 7. To power down the 11-40, skip to step 15.

5.3 The Bell Unit and BELLDF

The following section includes information on bringing up the Bell Unit and setting up the passwords for instructional use. Information on actual use of the Bell Unit is contained in the "User's Manual for the Serial, Time-Shared Digital Filter Experimental Unit."

9) At the completion of step 6, the BELLDF program may be run instead of SW2 or DMC. The first step is to turn on the Bell Unit. The toggle switch is located under the 11-40 behind the white bordered black faceplate in the left corner. There is no power indication lamp. The displays of the blue terminals should light up when the terminals are turned on. The teletype with the yellow paper should be set to "line".

10) At the "." prompt of the RKDP monitor type

    L BELLDF

    and wait for the next prompt. Then type

    S 100

    and the cursor should move to the next line (no new prompt is issued). At this point the terminal used to control the 11-40 may be switched back to ECN if desired.

11) The teletype should now respond to the command (typed on the teletype)

    LP

    with an "@". This is the command to list passwords; at this point there aren't any. To enter a password type

    EP,MASTER,xxxx

    where xxxx is any number between 1 and 9999. A password is deleted by typing (typed on the teletype)

    DP,MASTER,xxxx

    Passwords can be toggled between read only and read/write access by

typing

RO,MASTER,xxxx

After each of the commands EP, DP, and RO the "@" is issued to indicate that the command has been processed. The response to LP is a listing of the passwords and how many lines of memory each password is currently using. The remaining commands that can be issued from the teletype are discussed in the "User's Manual for the Serial, Time-Shared Digital Filter Experimental Unit" under the heading "TTY Operations".

12) Note that if power is turned off all passwords and files are lost. The TTY sometimes locks up when the room becomes too warm. The system then has to be rebooted beginning with loading address 773110. Again, passwords and files are lost. To prevent losing information, files should be dumped on paper tape at frequent intervals.

13) Connecting the D/A converter to the output of a second order section is discussed in the user's manual but some clarification may be helpful. After the lines describing the second order sections or digital filter configuration, the D/A is connected to second order section number n by entering

-n0

and then pressing the COEF# key on the blue terminal.

14) At the completion of a session with the Bell Unit, the TTY is turned off and the 11-40 is powered down as follows.

5.4 Powering Down the 11-40

The following is the procedure for powering down the 11-40.

15) Turn off the disk drive and wait for the SAFE light. When the light comes on, the 11-40 may be powered off.

6. Using the 11-40 with ECN - File Transfers

The DMC program has commands which are used to transfer files to and from a computer on ECN. All electrical engineering machines currently support this file transfer capability.

Begin the file transfer by bringing up the DMC program as discussed earlier. The link, ul, get and put commands are used to connect to the ECN machine and get a file from or send a file to that machine.

6.1 Getting a file

First link to the ECN machine by using the link command. If you are not sure whether a link to the ECN machine exists, perform an unlink operation by typing

```
% ul
```

To link to the "ed" machine type

```
% link ed
```

The terminal should respond with

```
MIDMCO UP
connect to host:12
```

and prompt you for your login and password:

```
Login:
Password:
```

Once you have given the login and password information you can send and receive files only (you are not on the network). The get command is now used to get a file from the machine:

```
% get filethere filehere
```

where "filethere" is the path to the file on the ECN machine and "filehere" is any filename you wish. Any number of "gets" can be executed since the link is not broken after a "get". When you are through, use the "ul" command (unlink). This happens automatically at 11-40 power down.

6.2 Putting a File

Again, start by linking to the ECN machine if a link does not exist. Then use the put command as follows:

% put filehere filethere

where "filehere" is the file on the ODSP lab disk and "filethere" is the pathname to where the file will be placed on the ECN machine.

You may need to modify your filesize limit if the file to be sent is over your limit. The default limit is typically 600 blocks. This is accomplished by typing

$limit filesize xxxx

where $ is the ECN prompt and xxxx is the filesize in blocks. The current filesize limit is found by typing

$limit

(all limits are displayed). The limit modification returns to the default upon logging off and after each put. For more information on the SW2 and DMC commands see the "Switch II and DMC User's Manual."

6.3 Bugs

If any problems are experienced during a data transfer (such as the inability to create a link, or a hangup occurs during transfer), it may become necessary to power down the 11-40 (see section 5.1). The link can then be re-established. This is a last resort; the unlink command "ul" should be tried first.

7. The Patch Panel

The white Patch Panel under the Rockland Filter has terminations for the Sony receiver and the PDP 11-40. Figure 3 illustrates the terminations. Following are explanations of each of the terminations.

7.1 Speaker B

The power/speakers switch connects the output of the Sony receiver to these jacks when set to the "B" position.

7.2 Preamp Output/Power Amp Input

These connections are normally jumped together. They allow for split operation of the receiver, i.e., a power amplifier other than that of the Sony can be used between the preamplifier and the speakers.

7.3 Rec 2 Output/ Tape 2 Input

The Sony Receiver has a pair of inputs and a pair of outputs for use with two tape recorders. One tape recorder can be used to record from or play to a second tape recorder while being monitored by the Sony with these inputs and outputs. The Akai Tape Deck is connected to the Tape 1 Input and Rec 1 Output. The Tape 2 Input and Rec 2 Output is brought out to the Patch Panel. The Tape Copy selector switch on the Sony receiver controls the use of these inputs and outputs.

1) When set on Source, the Phono, Tuner, or Aux program signal selected by the Function switches is applied to both Rec Out 1 and 2. The Volume control does not affect the amplitude at Rec Out 1 and 2.

2) When set on Tape 2→1, the signal applied to Tape 2 Input is sent to Rec 1 Output for recording by the Akai. (Note that Rec 1 Output is connected to the Akai input.) The Monitor should be set to Tape 2 and all Function switches should be off. The Volume control affects only the amplitude at the headphone or speaker outputs of the Sony.

3) When set on Tape 1→2, the output of the Akai is made available at Rec 2 Output. The Monitor should be set to Tape 1 and the Function switches should be off. Again, the Volume control does not affect the Rec 1 and 2 outputs.

7.4 Aux Input/Phone Input

These inputs are used with the Function switches for auxiliary programs and disk programs with MM/MC cartridge operation.

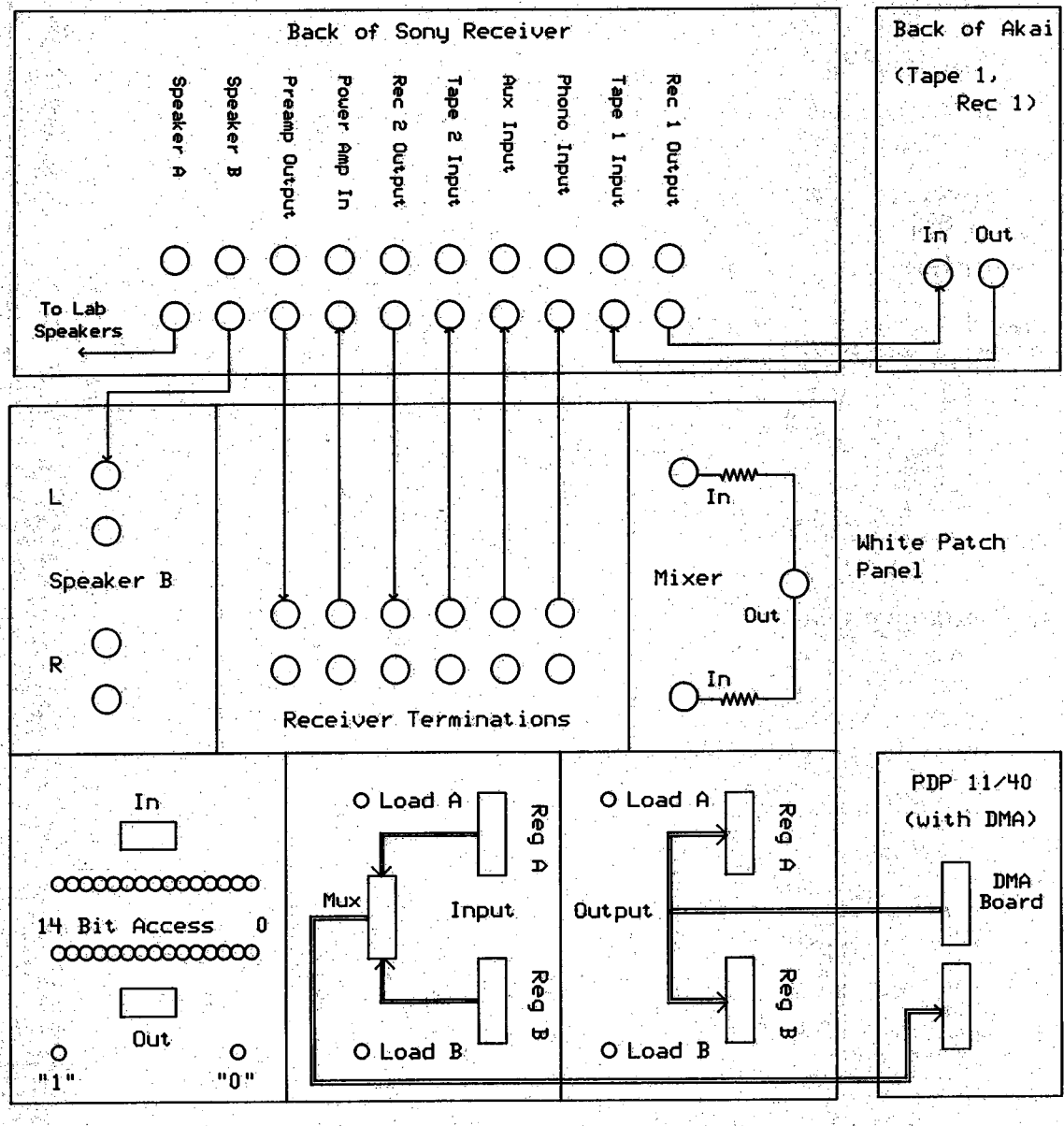


Figure 3. Connections between Patch Panel and Lab Equipment

7.5 Mixer

Two signals can be resistively summed by using these connections.

7.6 Bit Access

The Input and Output connections are brought out for access to the bits.

7.7 Input

The signals connected to the Input ports can be sent to the PDP 11-40 by using the "in" command in the SW2 program. There are two Input registers for stereo data transfer. The timing is controlled by inputs Load A and Load B. These timing inputs are ORed, which allows for sampling at a set rate, twice the set rate, or half the set rate.

To collect mono (one channel) data, the STEREO switch is set to the NO position. This disables Input B. The sample at Input A is loaded into register A whenever a pulse appears at Load A or Load B. Sampling occurs at the set rate determined by the Kluge clock when either Load A or Load B only is pulsed. If both Load A and Load B are pulsed, then channel A is sampled at twice the set rate. The samples are made available to the 11-40 via the "in" command.

To collect stereo data, the STEREO switch is set to the YES position A sample from each input is loaded into its proper register. Using the "in" command, alternate samples from Input A and Input B are stored on disk. If Load A and Load B are pulsed, then sampling occurs at the set rate determined by the Kluge clock. If Load A or Load B only is pulsed, then stereo sampling occurs at half the set rate.

7.8 Output

The PDP 11-40 is accessed at the Output ports by using the "out" command in the SW2 program. Samples are loaded into Registers A and B when the appropriate Load inputs are pulsed. Both registers are used in stereo applications; only one is needed for mono.

8. Using the 11-40 with the Kluge, Sony, and Akai

In this section we describe the standard setup for digitizing and reconstructing audio signals in stereo or mono. The interconnections for the complete stereo setup are shown in Figure 4. Mono processing is accomplished by using the left or right channel only. Stereo processing requires both channels.

The Patch Panel provides access to the inputs and outputs of the PDP 11-40 and the Sony and Akai. The PDP 11-40 controls the flow of the digital data between the lab disk and the In and Out ports on the right of the Patch Panel. The Sony and Akai are the analog source, playback, and storage devices. The Rockland filter is used to prevent aliasing. The Kluge controls the A/D and D/A conversions, and the analog signal amplitudes before and after the conversions.

Ground loops should be avoided when making the analog signal connections. The main ground is on the Kluge. By convention, the A/D and D/A converters have the analog signal grounds, while the amplifier and attenuator ground connectors (marked with an * in the diagram) are left disconnected. This prevents a ground loop through the Rockland filter. Inputs to the Akai or Sony on the Patch Panel should be terminated to prevent noise pickup on the high impedance inputs.

A special note of caution must be made. The amplifier on the Kluge is not short circuit protected. If the signal (red) is connected to ground (possibly by reversing the leads in a connecting cable), the output op amp will burn out and will have to be replaced.

8.1 Digitizing Data

The signal to be digitized should appear at the Rec 2 Output terminals. If the signal is from the Sony, then the Tape Copy switch is set to source. If the signal is from the Akai Recorder, the Tape Copy switch is set to Tape 1→2. The Sony can be used to monitor the signal by setting the Monitor switch to Tape 1. The Rec 2 Output is connected to the Kluge Amplifier for amplitude control. The signal is then passed through the left half of the Rockland filter for lowpassing to prevent aliasing. The cutoff of the lowpass filter should be half the sampling rate set by the clock on the Kluge. Once the signal is lowpassed, it is sampled by the A/D converter. The sampling consists of a convert and a load. As discussed earlier in the Equipment section concerning the Kluge, a clock period is split into N phases (N = 10 here). A sample from one channel is

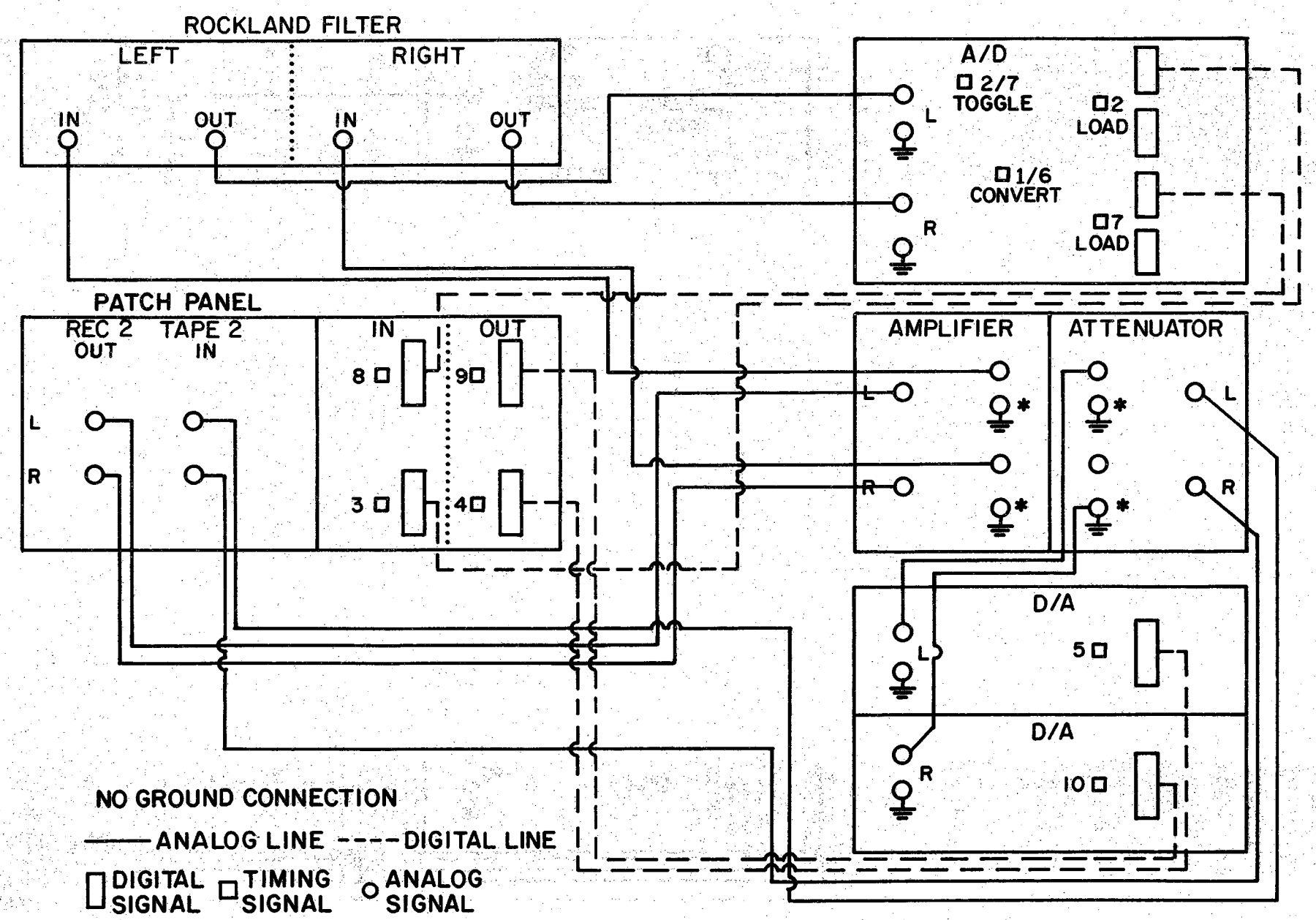


Figure 4. Stereo Signal Sampling and Reconstruction

converted on the first phase. It is loaded on the second phase. At least 7 μsec must elapse between the rising edges of the convert and load pulses in order for the A/D to perform a conversion. At the same time the A/D is toggled to look at the second channel. Using phases six and seven, the other channel is sampled and loaded, and the A/D is again toggled. In this way both channels are sampled and loaded during one clock cycle. The samples are made available for disk storage by loading them into Registers A and B on the Patch Panel Input section. This also occurs during the current clock cycle; phases three and eight are used to load the samples from the respective channels. The "in" command of the SW2 program is used to put this data on the disk.

8.2 Reconstructing the Signal

Digitized data stored on the disk with the 11-40 is reconstructed using the "out" command of the SW2 program. This command moves data from the disk to Registers A and B of the Output section of the Patch Panel. For stereo reconstruction, phases four and nine of a clock cycle are used to alternately load samples into their respective registers. For mono reconstruction, only one of the phases is needed. The next phase (five or ten) loads the sample into the appropriate D/A converter for reconstruction. Again, by using the different phases of the clock cycle, both stereo samples can be reconstructed in one sampling period. The lowpass filters on the D/A converters should be set to half the sampling rate to remove spectrum replication from the D/A conversion process. The analog output of each D/A converter is connected to half of the Attenuator section of the Kluge. The attenuator controls the amplitude of the reconstructed signal. The signal is placed at the Tape 2 Input, where it can be monitored by the Sony and recorded by the Akai. Monitoring is accomplished by setting the Monitor switch to Tape 2, and recording is made possible by setting the Tape Copy switch to Tape 2→1.

Chapter 2

The
SWITCH II
and
DMC
USER'S MANUAL

INTRODUCTION

This document provides the user with all the information necessary for the productive use of the digital signal processing software available in the ODSP lab. The ODSP software consists of utilities, which allow editing of audio files, as well as the standard (and some not so standard) mathematical operations which are often needed to be performed on audio files, such as filtering (linear and non-linear), the FFT, FM, correlations, convolutions, etc. The software was written by Chuck Binzel. Modifications were later made by Mark Yoder and Jon Dattorro.

The document is broken into two parts: a description of the command syntax, and a description of each command's operation. The syntax description contains examples to help explain the notation. The syntax itself is presented on the first page of the syntax description, and the conventions established there are used throughout the command synopses which head the description of each command. A first-time user should thoroughly understand the syntax description before attempting to use any of the commands.

There are numerous references to the Switch II (SW2) software. It should be known at the outset that there are two digital signal processing programs commonly run in the ODSP lab; they are called "SW2" and "DMC." Originally, the DMC program was intended to contain only commands which facilitated file transfers to and from UNIX, but, for various reasons, DMC eventually acquired some of the same commands as there are in SW2 as well as some commands that are not. Each command description usually informs the user of the program in which that command resides. The programs themselves, however, will give appropriate messages if the user attempts to invoke a command which exists in the *other* program.

Some of the command descriptions also give some insight, where appropriate, into the workings of some of the lab hardware. So, it would be useful to the first-time user to read all the command descriptions to gain a general knowledge of the ODSP lab.

The ODSP lab can be a very powerful tool in the hands of an experienced user. Good luck in your endeavors!

ODSP COMMAND SYNTAX

Table 1
Switch II Syntax

```
<src>                          = <pathname> [<window>]
    <pathname>                 = [<dev>/]<filename>
      <dev>                    = [a,h,b,c,d,e,f,g,i,j,k,l,m,n]
    <window>                   = [-<start>] [+<end>]
      <start>                  = [<NULL>,b,s,a,ab,as,e,eb,es]<NNNN>
        where;          {
                        NULL   = prior to mark in time
                        b      = prior to mark in blocks
                        s      = prior to mark in samples
                        }
      <end>                    = [<NULL>,b,s,a,ab,as,e,eb,es]<NNNN>
        where;          {
                        NULL   = subsequent to mark in time
                        b      = subsequent to mark in blocks
                        s      = subsequent to mark in samples
                        }
      and where;        {
                        a      = relative to beginning in time
                        ab     = relative to beginning in blocks
                        as     = relative to beginning in samples
                        e      = relative to ending in time
                        eb     = relative to ending in blocks
                        es     = relative to ending in samples
                        NNNN   = number of time units, blocks, or samples
                        }

<dest>                         = <pathname> [<attribute>]
    <pathname>                 = [<dev>/]<filename>
      <dev>                    = [a,h,b,c,d,e,f,g,i,j,k,l,m,n]
    <attribute>                = [-r<NNN>] [-m,-s,-d,-q,-p,-c]
        where;          {
                        NNN    = sampling rate
                        m      = monophonic file
                        s      = stereophonic file
                        c      = complex file (result of fft command)
                        p      = break-point file (see nlo command)
                        q      = queue file (see sos command)
                        d      = data file
                        }

s<NNNN>                        = NNNN tenths of seconds of silence (see out, tape, r, cat,
                                 and rev commands)
*                              = append to current stack (see r and out commands)

<dir>                          = pseudo-directory (see cd command)
<UNIX>                         = unix pathname
<host>                         = [ea, pa, pb, ka, fu, eeg, arpa, ec, ma, ga, ca, ed, ef, eg, ee]
                                 (network host)
```

Table 1 is the syntax of the Switch II (and DMC) command structure. Indented entries are related to entries immediately above. When character strings are flanked by < >, this means that the enclosed strings are not to be typed literally. Rather, the enclosure is to be interpreted as an English description of the data *type* that the user should enter. For example,

out <src>

(where we call <src> the "non-literal") means the following: we suppose the user to have a file called "junk" which the user wishes to hear via the "out" command. Then, on the command terminal, the user would type

out junk

(followed by a carriage return)* where "junk" is the name of the source (src) file whose samples are to be D/A-converted and passed to the speakers. Similarly, strings or non-literals flanked by [ ] are considered optional to the command. For example, if one looks at the synopsis of the "af" command, one finds

af <src> [<src> . . .] > <dest>

Here, the square brackets imply that the user may request the command to operate on more than one source file. (Notice that some UNIX conventions are employed; specifically, the "greater than" sign indicates that the output should be placed in <dest>, the destination file which will be created as a result of using this command.)

When no < > delimiters are specified in the syntax, this indicates that the ASCII character strings are to be typed literally. ASCII character strings which are preceded by a plus or minus sign are themselves parameters which are to be passed to the commands. These are usually optional. Keep in mind that not all parameters may have meaning to every command; e.g., not all commands can operate on windows which are specified in samples. Also, some commands ignore the <window> option altogether. The window option allows the user to have only a specified portion of the source file operated on by the given command. The following Table 2 shows the only commands which can operate on file windows specified, by the user, in samples:

*All commands must be terminated by using a carriage return, <cr>.

Table 2

fft
plot
bh3w
bh4w
hw
cm
crc

<NNNN> denotes a decimal (or sometimes octal) number which assigns to a parameter a specific numeric value.

When <src> appears in a command synopsis, we see from the syntax table that it may be composed of two types of information, a pathname possibly appended by a window. The "pathname" is defined such that by prepending a device reference letter (<dev>) to a filename, any file on any storage device (in any directory) may be accessed even though the user's current directory may reside on some other device. So, for example, the user's current directory may be on a device which is part of the CDC disk drive (the CDC is broken down into 12 separate devices: <dev> = b, c, d, e, f, g, i, j, k, l, m, n), but he or she may wish to play a file named "junk" which is stored on another device. (The Aries disk drive contributes two additional devices: <dev> = a, h; device "h" is not available for general use since it is used for the ODSP operating system.) Without specifying the pseudo-directory, the user simply issues the command

out g/junk

which will play the file "junk" on storage device "g" regardless of the current directory or device. If the current device were "g," the "g/" prefix would not be necessary. It is possible, therefore, to have simultaneously on all devices a file named "junk," and each one will be uniquely specified.

There are two types of directories for each device, the main directory and the pseudo-directory. A pseudo-directory is a means by which a user groups together a number of file*names* so that when a user lists his or her files (via the "ls" command), only the ones in the current pseudo-directory will be listed. One way to create a pseudo-directory is by using the "cd" command (see this command for more on pseudo-directories). The term "pseudo-directory" is used because the single ASCII character which identifies the pseudo-directory is *never* used in a pathname. In contrast to UNIX subdirectories, pseudo-directories afford the user no segregation of files. There may *not* coexist, therefore, two files having the same name on the same *device.* Listing (using "ls") from the main directory allows the user to list all files in all

pseudo-directories. Any files which have not been assigned to a pseudo-directory are considered to be residing in the main directory and will also be listed.

The pathname in the previous command example may optionally be followed by a window so that the user will hear only a specified portion of "junk." The window specification may contain a starting point reference, an ending point reference, or both, with respect to the "src" file in question. The starting point specification carries a minus sign prefix. This starting point may be relative to the beginning or end of the "src" file, or relative to a user-defined "MARK" in a file. These same reference options apply to the end point specification of the window, which is prefixed by a plus sign. The starting and ending point specifications may be made in terms of time (in tenths of seconds; see the "cti" command to change this), blocks (256 samples per block), or samples (2 bytes per sample), with respect to the indicated reference points. The default starting point specification is simply the first sample in the "src" file. The window endpoint specification defaults to an imaginary sample placed immediately after the last file sample. Any window specification is assumed to be *inclusive* of the starting point, but exclusive of the specified (or defaulted) endpoint. These same defaults and inclusions apply when the window is specified in terms of blocks, samples, time, or any *mixture* of the three possible reference types. Some examples of "src" file windows follow: For

out junk −3 +57

the portion of "junk" heard will be six seconds in length. It will begin 0.3 seconds prior to the "MARK" location, and end 5.7 seconds after the "MARK" location. The "MARK" defaults to the first file sample. Once the user specifies a new MARK via the "cm" command (see that command for further explanation of the MARK), that MARK is retained by the file as long as the file exists. If the "MARK" were default, the user would hear 0.3 seconds of the physically previous file (see the "ls" command) on the disk, and 5.7 seconds of "junk" from the beginning. No ASCII character is present in the starting or ending points of the window specification above. This means that for each part of the specification we have invoked the NULL parameter which connotes the "MARK" as well as time. So, −<start> is −3 in our example, and +<end> is +57. Note that the commas in the <start> and <end> definitions (and the other definitions) mean "or"; i.e., only one parameter can be chosen from the given set.

The following examples assume that the "out" command recognizes windows specified in units of samples. This is just for illustration and the reader is referred back to Table 2. (As may have been inferred from the previous example, <NNNN> is always an integer, in the syntax).

out junk −b3 +s57

will play 3 blocks previous to the "MARK" in "junk" and up to but not including the

57th sample subsequent to the MARK. Blocks and samples starting from the beginning of a file are numbered 0, 1, 2, ...

out junk −s57 +b3

will play 57 samples previous to the "MARK" in "junk" and just 3 blocks subsequent to it. Observe that it is not possible to specify a window, using the MARK exclusively, which does not flank the MARK (unless <NNNN> is a negative integer).

out junk −a2 +a3

plays from 0.2 seconds after the beginning of "junk" to 0.3 seconds after the beginning; i.e., 0.1 seconds of sound.

out junk −a2

plays from 0.2 seconds after the beginning of "junk" to the end of the file.

out junk −a2 +e3

plays from 0.2 seconds after the beginning to 0.3 seconds before the end of "junk."

out junk −e3 +e2

plays from 0.3 seconds before the end to 0.2 seconds before the end of "junk" (0.1 seconds of sound).

out junk −as40 +es40

plays from 40 samples after the beginning to 40 samples before the end of "junk." Finally, the most common type of window:

out junk −ab1 +eb1

plays "junk" except for the first and last blocks. Most of the time the user will prefer windows using *block* specifications.

When <dest> is indicated in a command synopsis, it refers to a user named file (new or existent) into which the output of that command will go. In the syntax of <dest>, the <pathname> and the <dev> are the same as they were for <src>. This allows a user to invoke a command, for example, on a file from a given device, and to send the output to another device.

It is not possible to specify a window in the destination file. The destination file may be assigned certain attributes, however. These attributes can be examined in the long listing of the user's files (see "ls" and "cfa" commands). For example,

cat junk > g/junk −r20000 −s

will copy the contents of junk on the current device into the main directory on the "g"

device. Note that it is not possible to specify a pseudo-directory for <dest>. If the long listing on device "g" is observed after the execution of this command, it will be found that "junk" has a sample "RATE" of 20000(Hz) and has been assigned the "TYPE" "S" for "stereo." The file types (a subset of <attribute>) can be assigned indiscriminately, and change nothing physically about the assigned file. Some attributes are used as parameters to some commands and must be set properly for the successful execution of those commands (see "fft," for example). The primary function of the <attribute> is to remind the user as to the nature and contents of his or her files. The default file "TYPE" is "D" for "data." Some commands will automatically set the proper attributes. For example, making a copy of a file (via the "cat" command) will carry the attributes over to the <dest> file. Some mathematical commands must have a "RATE" specified since it is used in the calculations. The default "RATE" is 0 and usually causes floating point errors in such commands. Any attribute can be set by using an <attribute> in the <dest> file of some command, *or* by using the "cfa" command.

When <dir> is indicated in a command, the pseudo-directory name may be required. Pseudo-directories have names which are one alphabetic character in length. See the "cd" and "copy" commands for examples of commands which use <dir>.

<UNIX> is the complete pathname needed to describe the whereabouts of a UNIX file. For example,

/b/dattorro/[<sub-directory>]/<filename>

might be a complete UNIX pathname, where "b" is the device name, "dattorro" is the user, followed by the hierarchy of sub-directories separated by slashes, if any, and finally the filename itself. When called for, it is wise to use complete pathnames. See "get" and "put" for examples of commands which use <UNIX>.

<host> is any one of the network machine names which are denoted by the indicated mnemonics. See "link" for an example of a command which uses <host>.

s<NNNN> means silence for <NNNN> tenths of a second. So, for example,

out file1 s40 file2

will play "file1" followed by 4 seconds of silence and then "file2." The default time increment of 0.1 seconds can be changed using the "cti" command. The syntax of this silence specification precludes the naming of files using the character "s" followed by a numeral. So, for example,

out s2df

would produce 0.2 seconds of silence, *not* the file named "s2df".

The character "*" has been given special meaning and is a directive to the "r" and "out" commands to append the specified source file-*names* and associated <window>'s

to the current output stack (or queue; see "pos"). The output stack is a list which facilitates the output of an intricate ordering of files (or file portions) possibly interspersed with silence. The output stack order is top-down and it determines the order of output to the D/A. The output stack itself can be saved for future use; see "sos," "ros." Using the "*" directive, the "r" and "out" commands will output those <src> files residing on the stack and those specified on the command line. If the "*" is not used, the output stack will be replaced with the <src> names and windows on the command line.

SYNTAX CONCLUSION

<esc>:

The 'escape' key halts printout to the terminal during any output until pressed again (or until any other key is pressed).

<break>:

The 'break' key unconditionally interrupts any command no matter where it is. This means that a file may be only partially altered. (The 'break' key must be hit twice in succession on the Tektronix terminal.)

<cr>:

All the commands which print any quantity of information can be halted by the use of the 'return' key. Hitting 'return' after the completion of the previous command will repeat that command.

Floating point errors:

At this time, some conditions can cause traps in the floating point processor. When an error occurs, a message is printed and the command is terminated.

Auto-help:

There is a provision in the system for auto-help files which work similarly to the UNIX '.profile'. When the user changes directories (with 'cd') the system looks for a file with the name '.start' and if found prints out its ASCII contents. This file can be used to identify disk packs using some sort of message or can be use to program function keys, or both. If this file is not found no action is taken. The user can type

help <command name>

for a brief description of each command.

help .start

will print the ASCII contents of the .start file, if it exists. See the "text" command for more information.

ODSP
SW2 and DMC
COMMANDS

NAME

ab – add a fixed bias to file(s)

SYNOPSIS

ab [−s] <NNNNN> <src> [<src>...]

DESCRIPTION

ab adds a prescribed constant, expressed by NNNNN as a (possibly signed) decimal integer, to each and every sample of the named files. The sum file replaces the named file, so the original file of data is lost. The file samples themselves are treated as 16-bit integers so that a NNNNN specification of 1 would add a value equal to one least significant bit to each file sample. <src> may employ window delimiter information thus biasing only a selected portion of that file.

Without the −s option, any sums exceeding the limits (defined to be −32767 to 32767) will "wrap around" as in normal two's complement addition rules. Specifying the −s option will cause sums exceeding these bounds to be saturated at 32767 or −32767, whichever is appropriate.

Approximate execution time for this command is 53 ms./block.

In the following example, we have requested the subtraction of 2^{10} from each word in the file "music".

ab −s −1024 music

The saturation option has been invoked. If the last few samples of "music" are zeros (possibly due to a previously applied padding operation), these will be affected by the bias also.

BUGS

A NNNNN specification of 0, +0, or −0 will produce horrible results.

NAME

af – form a linear combination of files (add weighted files)

SYNOPSIS

af <src> [<src> ...] > <dest>

DESCRIPTION

A weighted sum of all named source files is placed in the named destination file. <dest> will always be placed after the last physical file on the device. If <dest> existed prior to the af-command execution, it will be destroyed only if it was the last physical file. Otherwise, only the name of <dest> is destroyed at its former location in the index (see "rm"). The user is prompted for a multiplicative weight for each of the named source files. These weights need not be integers. If only one source file name is given, "af" simply scales that file by the user-prescribed multiplier. In any case, the block length of <dest> will be the length of the largest source file. The multiple-source file time-alignment is left justified, as would be intuitively expected. Entering a weight of −1 inverts a file; i.e., flips it over the time axis. After each weight is entered from the terminal, there will be a pause while that weight is applied to the corresponding source file and the weighted samples totalled in <dest>. Samples of the named source files are unchanged.

Multiplicative factors of one are checked for by the program. If found, no multiplication is actually performed with the result that unity scales are performed at roughly twice the speed of other scalings.

Some observed command execution times when operating on a single source file are:

272 ms/block	(Factor = 0.5)
230 ms/block	(Factor = 0.0)
132 ms/block	(Factor = 1.0)

When two source files of identical length were specified, the times observed were:

432 ms/block	(Each factor = 0.5)
383 ms/block	(Each factor = 0.0)
183 ms/block	(Each factor = 1.0)

When processing one file, three operations are required by the *af* command:

1) Read a buffer of <src> data,
2) Scale the samples within this buffer, and
3) Write the scaled buffer to <dest>.

However, successive passes to accommodate a second, third, or later source file must read not only the next consecutive source file, they must also read the <dest> file as it was written during the last pass. (By successively writing and re-reading the <dest> file, source files of indefinite length may be handled without requiring the disk drive heads to alternate rapidly back and forth between corresponding sections of several <src> files.) Thus, the "two-file" time quotes given above are somewhat more than those experienced in the single file case.

Rounding *is* performed on scaled <src> file samples before they contribute to the developing sums in <dest>. Also, the user should be advised that products exceeding the maximum dynamic range of −32767 to +32767 will "wrap around" these limits causing a gross error.

Note: SWITCH II is constructed to handle integers in the range of −32767 to 32767. This is so that −32768 would not have to be treated as a special case in some algorithms such as that found in the "sc" command.

NAME

ah – amplitude histogram of file samples

SYNOPSIS

ah [<# of bins>] <src> [<src> ...]

DESCRIPTION

An amplitude histogram of each of the named source files is calculated and listed on the terminal screen. The user is prompted for the upper and lower limits of the amplitude range that is to be partitioned into bins. (The default limits are both zero.) This range will be broken into the number of bins specified in the command line, with a maximum of 256 bins being allowed. If the number of bins is greater than the specified range, the number of bins is set equal to the range regardless of the user's wishes. A default of 32 bins is used in the absence of this specification. If several source files appear on the command line, a separate histogram will be calculated and listed for each file. When specifying amplitude range limits, the user should bear in mind that file sample values are considered to be integers in the defined range of −32767 to 32767. After the histogram is listed, the user will be asked if he/she wishes to see a plot of the histogram just listed. If the user responds with "no" (or <cr>), the program will begin calculating the histogram for the next named source file (if any). If the user responds with "y" a plot will appear whose vertical axis is the percentage of data points within a given bin with respect to the entire source file, while the x-axis represents the user-specified amplitude range. The user escapes after the plot with a <cr>.

Command execution time (exclusive of histogram listing time) has been observed to be approximately 329 ms/block.

NAME

bc – calculate block or logical numbers

SYNOPSIS

bc [<NNN>]

DESCRIPTION

This command is for "system" purposes and is rarely used in signal processing work. To describe its operation, the reader must be aware that SW2 refers to a given disk block (512 bytes) in either of two different ways. First by simple block number, where 0 refers to the first block of the first disk cylinder, 1 refers to the second block of the first cylinder, etc. The second type of reference to a disk block is by "logical" number (disk address). The formats of the disk address references for the Aries and CDC 9766 drives are shown in the next two figures. (For the Aries, the "drive select" number refers to one of eight RK-05 equivalents.) For the users' purposes, logical device "a" resides on the removable Aries platter (cartridge), while logical device "h" (the "help" file) and the system software library (maintained by DEC's RKDP program) occupy the lower (fixed) Aries platter. Logical devices b-g and i-n exist on the CDC drive.

If the user types

bc <NNN>

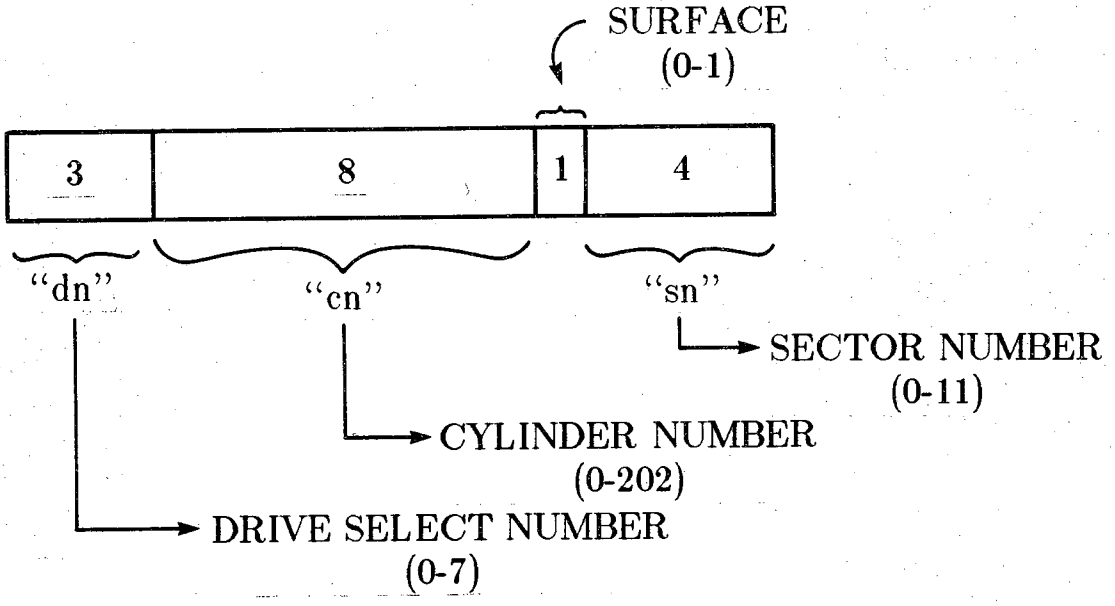
where NNN is a decimal block number, four quantities (five for the CDC drive) will be printed out, but only the *first* such quantity will have meaning. This will be an octal value that expresses the block reference in "logical number" disk address format as per the figures. NNN can be expressed by the user in octal form if he/she types <NNN> with a leading zero. Also, if NNN is zero, the current disk address (contained in the drive controller) will be returned in the first printed number.

If the user again types

bc <0NNN>

where NNN is now an octal representation (note the leading zero) of a valid "logical number" block reference, all but the first of the printed numbers will have meaning. (The first number appearing on the screen will now be garbage.) The second listed number will state the block number (in octal) of the block referred to by the NNN logical number. The third listed number will state this same block number, but now in decimal. The fourth listed number will reconvert the previously stated block number back to a "logical number" format (in octal). Thus,

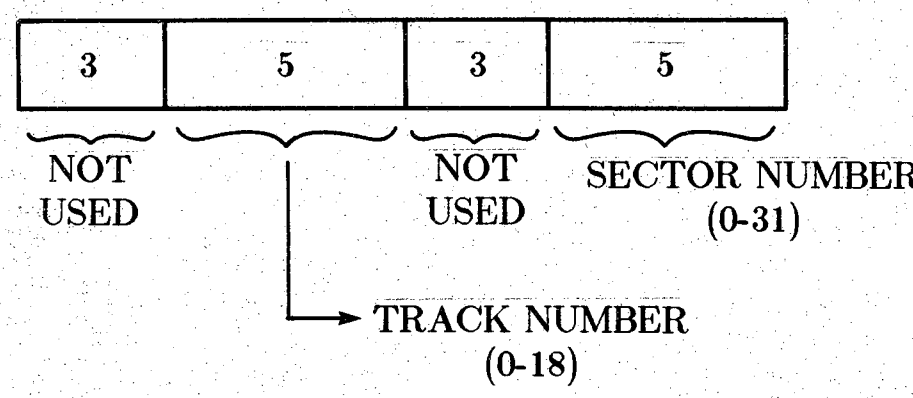
this fourth quantity should be identical to NNN. In the case of the CDC drive, a fifth quantity, listing the contents of the "desired cylinder" register, will be printed. In all these cases, the block and logical numbers will refer to blocks on the current physical drive in use. If no NNN quantity is given by the user, the four (or five) quantities listed will all be in reference to the last-addressed block on the current physical drive.



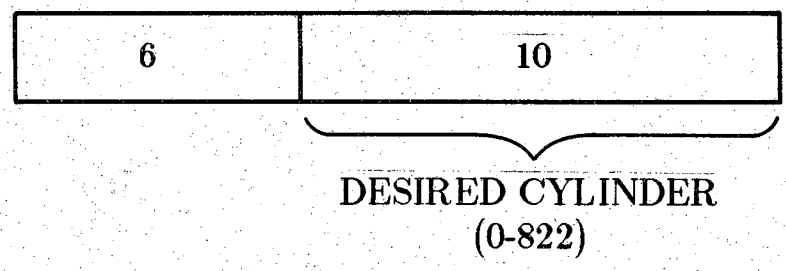
Format of a "logical number" (disk address) reference to a disk block on the Aries drive

Figure 1

DISK ADDRESS REGISTER (RMDA)



DESIRED CYLINDER REGISTER (RMDC)



Format of a "logical number " (disk address) reference to a disk block on the CDC-9766 drive

Figure 2

NAME

bh3w – 3-term Blackman-Harris window

SYNOPSIS

bh3w <src> [<src> ...]

DESCRIPTION

The 3-term Blackman-Harris (B-H) window is shown in the top figure. The purpose of this command is to weight the specified <window> of each separate source file by the B-H window. If the reader will imagine that the B-H window will first be adjusted to span a user-specified portion (<window>) of a named source file, and then superimposed upon that same file portion, then each sample in that portion of the file will be weighted by an amount equal to the precise value of the point on the B-H window immediately above. Keep in mind that all files in SW2 are multiples of 1 block in length. If a <window> is not specified for a source file and if that file has padded zeroes, then the B-H window will be adjusted to end in the last sample in the last block, *not* the last nonzero sample. This is not a problem because if desired, any portion of the source file can be <window>ed out using the options listed in the SW2 syntax description. In fact, any portion of a file may be specified down to the sample. For longer files, cutting out the last block will make little difference since the B-H window approaches zero towards the end of the file. Looking at the figure, $\frac{T_w}{2}$ will correspond to the end of the specified portion of the named file and $\frac{-T_w}{2}$, to the beginning. Note that the B-H window peak is equal to 1 which indicates that only the samples near the middle of the <window> in the source file will retain their original values. The output is placed back into the original source file so that only the specified portion of the original source file is destroyed. Each calculated point is rounded up (or down) to the nearest *integer* value.

Approximate run-time for this command is

1.2 sec/block.

Equation (1) (in the figure) is the continuous form of the B-H equation, but since our data is discrete, we are using the discrete form, (2). Equation (2) is shifted to the right by half the file length since we are dealing with causal signals. Because the equation index is "n" (not "nT"), it is not necessary for the file to have an associated sample rate (see "ls" command) in order for this command to

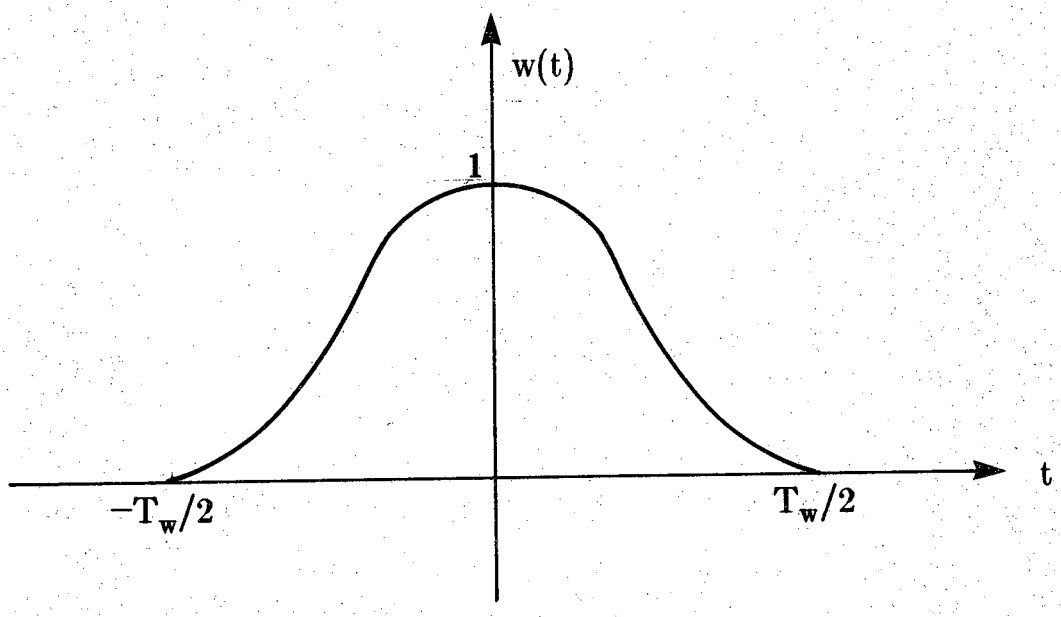
work properly.

Example:

bh3w <src> +ab1

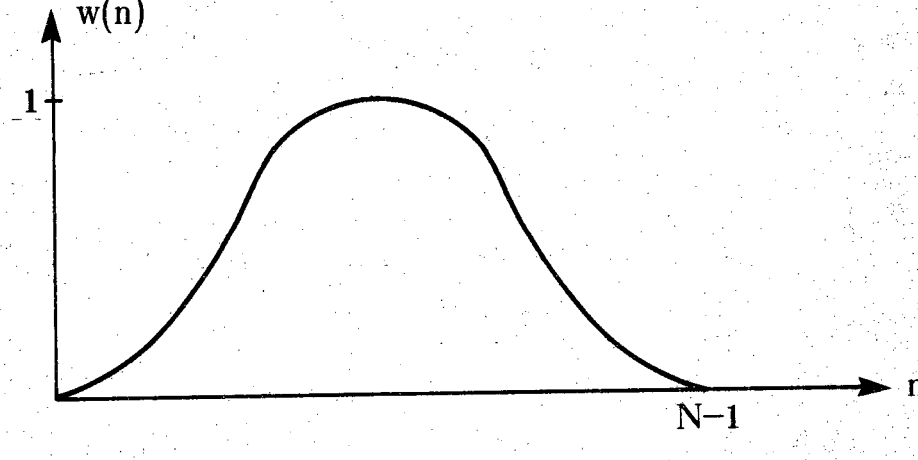
will apply the B-H window to only the first block of <src> and will place the result back into the first block of <src> leaving the original filelength and the remainder of the file unchanged.

The 3–term Blackman–Harris Window



$$(1)\quad w(t) = 0.42323 + 0.49755\cos(2\pi t/T_w) + 0.07922\cos(4\pi t/T_w)$$

NOTE: $T_w/2 \Longleftrightarrow \dfrac{N-1}{2}$



$$(2)\quad w(n) = 0.42323 + 0.49755\cos\left[\frac{2\pi\left(n-\left[\frac{N-1}{2}\right]\right)}{N-1}\right] + 0.07922\cos\left[\frac{4\pi\left(n-\left[\frac{N-1}{2}\right]\right)}{N-1}\right];$$

N = Total Number of Samples

n = nth sample = 0 → N–1

$\left[\frac{N-1}{2}\right.$ can be fractional.$\left.\right]$

Figure

NAME

bh4w – 4-term Blackman-Harris window.

SYNOPSIS

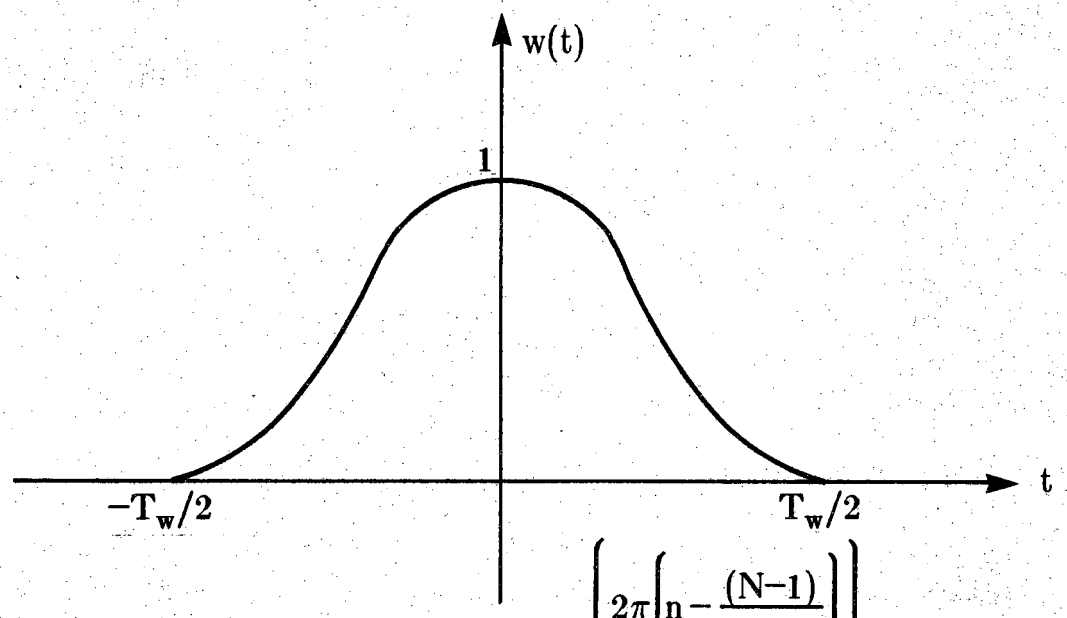
bh4w <src> [<src> ...]

DESCRIPTION

The 4-term Blackman-Harris window is shown in the figure. The description of this command is identical to that for "bh3w." The only difference is the weighting equation (see Fig.). Note that the sidelobes of the Fourier Transform of this window are 90dB below the main lobe, but the main-lobe width is twice the Hamming lobe width and four times the rectangular lobe width.

Run time is 1.61 sec/block

$$(1)\ 0.35875 + 0.48829\cos(2\pi t/T_w) + 0.14128\cos(4\pi t/T_w) + 0.01168\cos(6\pi t/T_w) = w(t)$$



$$(2)\ 0.35875 + 0.48829\cos\left[\frac{2\pi\left[n-\frac{(N-1)}{2}\right]}{N-1}\right] + 0.14128\cos\left[\frac{4\pi\left[n-\frac{(N-1)}{2}\right]}{N-1}\right] + 0.01168\cos\left[\frac{6\pi\left[n-\frac{(N-1)}{2}\right]}{N-1}\right] = w(n)$$

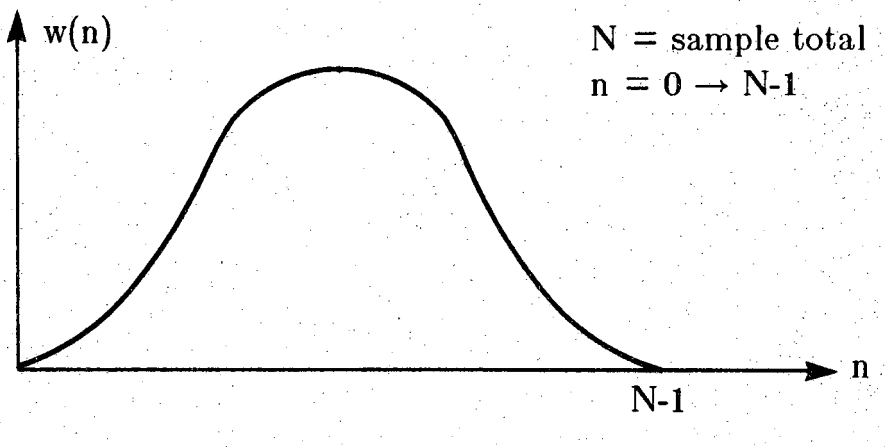


Figure: 4-term Blackman-Harris window

NAME

bwo – bit window operator

SYNOPSIS

bwo [<bit order>] < <src> [<src> ...]

DESCRIPTION

This command allows the user to directly manipulate the bits of each word of the named source files with the results placed back into the same respective files. This command operates differently than the "BIT ACCESS" module (located next to the PDP I/O ports) in so far as the manipulations are performed on the *sign + magnitude* representation of each source file word, whereas the module manipulates the two's complement representation (because that is what is available at the PDP output port). The following flowchart describes the overall command operations.

Essentially, the user has the ability to move selected bits to specified locations, duplicate selected bits into specified locations, set bits, and clear bits of the *sign + magnitude* representation of each word in his/her file. This ability is indicated in the synopsis by "[<bit order>]." In this part of the syntax may appear the letters "s" and "c" for set and clear respectively, and/or the ordinal decimal numbers of the bit locations in a 16-bit word (15 = MSB, 0 = LSB). A maximum of 16 of these symbols may be typed consecutively having the leftmost symbol correspond to the MSB (or leftmost bit) of the result and vice versa. If we think of the <bit order> part of the syntax as 16 bins, we can place any appropriate symbol into each bin. For example, Figure 2

bit15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
3	6	2	15	10	c	s	c	s	0	1	s	s	c	c	15

↑
"separator"

Figure 2

might be a <bit order> specification. The separators are: spaces between the decimal numbers and spaces or NULLs between the non-numeric characters. This particular specification commands the following to be done to each sample of the *S+M* representation of the two's complement binary input:

bit
15 – place bit 3 (of *S+M*) into bit 15 (of the result)
14 – place bit 6 into bit 14
13 – place bit 2 into bit 13
12 – etc.
11 – etc.
10 – clear bit 10 (of result)
9 – set bit 9 (of result)
8 – etc.
7 – etc.
6 – place bit 0 into bit 6
5 – etc.
4 – set bit 4
3 – etc.
2 – etc.
1 – etc.
0 – place bit 15 into bit 0

Notice that bit 15 of the *S+M* is placed into the result twice at two different locations.

If bit 15 of the result (or bit 3 of the *S+M*) is a one, then from the flowchart we see that the two's complement will be taken for the *final* result. Note that the *S+M* version of the input is preserved throughout so there are no recursive operations going on. An unfilled bin defaults to "c".

Let's look at some examples:

bwo <src>

has no effect whatsoever.

bwo < <src>

sets the entire file to zero since "c" is default and so is equivalent to

bwo ccccccccccccccc < <src>

which is equivalent to

bwo cccc < <src>

i.e., any number of c's.

Example 1:

x = 1111 1111 1111 0000 = −16 (2′s comp. input)

x′ = 1000 0000 0001 0000 = −16 ($S+M$)

bwo(x′) = 0000 1000 0000 0001 = (result) = (final result)

when

bwo 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 < <src>

is issued.

So, this command reversed all the bits of x′.

However,

x = 1111 1111 1111 0001 = −15 (2′s comp.)

x′ = 1000 0000 0000 1111 = −15 ($S+M$)

bwo(x′) = 1111 0000 0000 0001 = (result)

but MSB = 1 so MSB←0

$$2'\text{s comp. (bwo(x')}\Big|_{\text{msb}\leftarrow 0} = 1000\ 1111\ 1111\ 1111 = \text{(final result)}$$

when the same command is issued.

Notice the big difference in the result when only the LSB of the input is changed! This particular command is useful for making white noise out of any non-trivial audio file.

Example 2:

Another useful operation using this command is to yield the absolute values of all file data. This is accomplished using the following command:

bwo c 14 − 0 < <src>

Firstly, notice the introduction of the shorthand notation "14 − 0" which is equivalent to having typed

14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

This shorthand is not allowed except for descending order; i.e., "0 − 14" is illegal. This command says: convert the word to $S+M$, then clear the MSB and put the 14th to the 0th bits of the $S+M$ into the 14th to the 0th bit positions of the result (i.e., copy them as they were). Since the MSB is always cleared to zero, the two's complement at this step is equivalent to the present $S+M$

representation. The final result is that we have the absolute value *in two's complement* of whatever word this "bwo" command works on. Specifically,

$$x = 1111\ 1111\ 1111\ 0000 = -16 \quad (2'\text{s comp.})$$

$$x' = 1000\ 0000\ 0001\ 0000 = -16\ (S+M)$$

The "c" in the command clears the leftmost bit of x' (because it was typed in the leftmost bin in "<bit order>"). So,

$$\text{bwo}(x') = 0000\ 0000\ 0001\ 0000 = +16\ (S+M)$$

$$= \text{(final result)}$$

Now let's try +16 using the same command.

$$x = 0000\ 0000\ 0001\ 0000 \quad (2'\text{s comp. input})$$

$$x' = x \qquad (S+M)$$

$$\text{bwo}(x') = x \qquad \text{(MSB already clear)}$$

$$2'\text{s comp}\ (\text{bwo}(x')) = x = 16 \qquad (2'\text{s} = S+M)$$

Voilà.

This command may also be used to perform other useful operations. The user must be creative in his/her implementation of it.

Runtime ~ 204 ms/block

Note:

Certain operations using the "bwo" have inverses; i.e., you can recover the original file as it was before the "bwo" command was invoked. Example 1 given previously has an inverse: simply re-issue the command again exactly as given in the example to recover the original file. The types of operations which have inverses are those which do not originally destroy information; this class of operations consists of bit permutations. As an example of this, consider

bwo 1 3 5 7 9 11 13 15 0 2 4 6 8 10 12 14 < file

and its inverse,

bwo 8 0 9 1 10 2 11 3 12 4 13 5 14 6 15 7 < file'

After the second "bwo", file' ← file; where file' is the output of the first "bwo". It becomes clear that to recover the original file, every bit must be present in the modified file; therefore, no bit duplications, sets, or clears would be allowed in the first of the two "bwo" commands.

This technique could be used as a quick way to encrypt files of any sort.

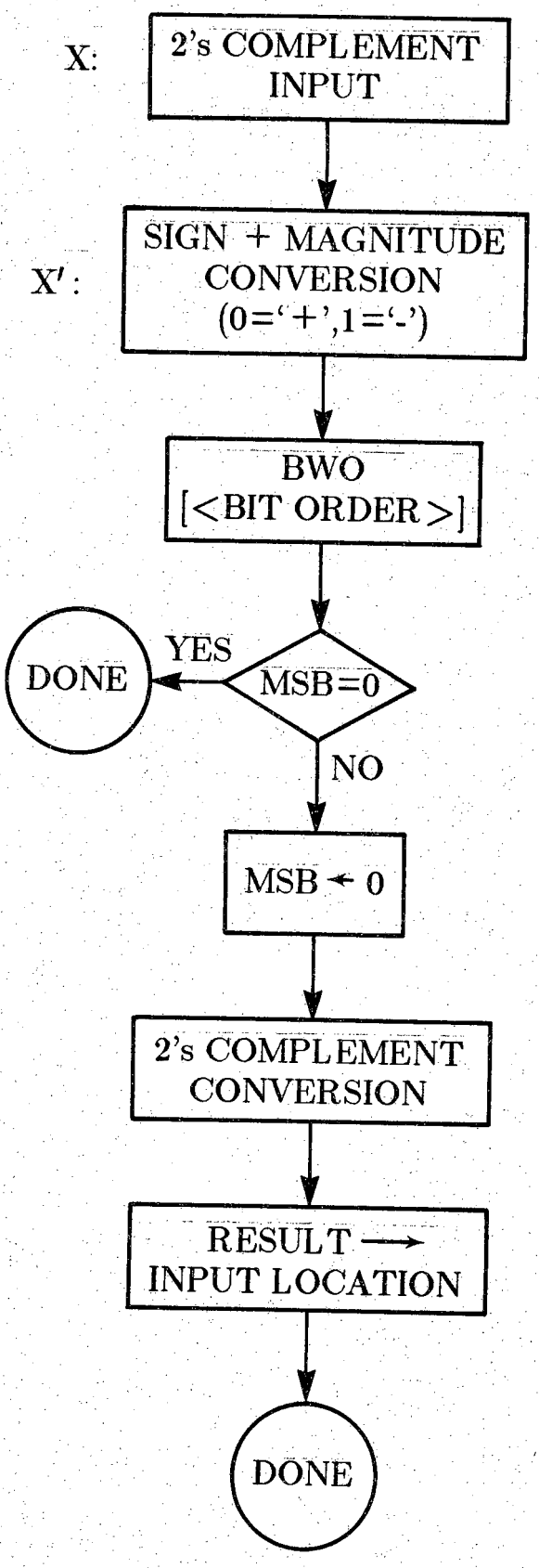


Figure 1
Flowchart—bwo command operation

NAME

cat -- concatenate files -- copy file

SYNOPSIS

cat <src> [<src> ...] > <dest>

DESCRIPTION

The named source files are concatenated and the result is stored in the destination. If no <dest> file is specified, a null file is "created". If no destination and no ">" are specified, an error message is generated. If only one source file is specified, it will be copied into the named destination. If a file having the same name as <dest> exists prior to the "cat" command, one of two things will happen:

A) If the file of the same name as <dest> is not physically the last file on disk, its name will be moved so as to make it last in the listing and the <dest> *file* will be created after the last physical file on the disk. Since only the name was moved, that pre-existing file will not be destroyed and can be recovered if necessary (see "rm" command).

B) If the file of the same name as <dest> is the last physical file, then it will be destroyed in the process of creating the <dest> file in its place.

In any case, the <dest> file is always placed after the last physical file and its name, therefore, is always last in the listing.

If the <dest> file is greater than the available disk space, then a file of length equal to the available disk space will be created having the name of the destination file. Thus, <dest> will be truncated.

This command differs from the UNIX command of the same name in so far as it can not be used to list ASCII file contents onto the terminal screen (see the "help" command, however).

This command is also useful to create many periods of a periodic waveform (one period of which is stored in some source file) by successive duplication. Specifically, suppose that you have used the pwl-command to create one period of a waveform, and then stored the result in "file". Then by issuing the command

cat file file> file

n times, you will have succeeded in duplicating the original source file 2^n times.

RUN TIME FOR THIS COMMAND IS

≃9.8ms/block

The "cat" command will carry over rate and type attribute information into the destination file.

This command will also honor the concatenation of silence (s<NNN>) with the <src> files. Any number of silences may be used, but at least one source file having a nonzero rate attribute must be given in the command line because each s<NNN> will need to be replaced with a file of zeroes of appropriate length. All files in the command line are assumed to be of the same sample rate although this is not necessary, in general, for the successful execution of this command.

NAME

cd -- change device and directory (create a pseudo-directory)

SYNOPSIS

cd [<dev>[/<dir>]]

DESCRIPTION

This command changes the current device, or the current device and directory and prints the new device and directory on the command terminal. The default device is "a" and the default directory is the main directory which has the NULL character identifier. When <dir> is not the NULL character it can be any *alphabetic* character in which case <dir> is called a pseudo-directory. In this manner, pseudo-directories can be brought into existence. (A pseudo-directory can also be created using the "copy" command.) A pseudo-directory will remain in existence only if files are stored in it. Once files are stored in a pseudo-directory, using this command will simply have the effect stated at the outset of this description. Hence, the user may "change" to an existent or non-existent pseudo-directory at any time.

The primary purpose of pseudo-directories is to make it convenient for the user to see only those files stored in some pseudo-directory when he/she issues the "ls" command from that pseudo-directory. Pseudo-directories afford the user no segregation of files, as in UNIX, and *the pseudo-directory alpha-identifier is never used in a pathname,* hence the term "pseudo". (The "copy" command will transfer the files in a given pseudo-directory from the device in which the pseudo-directory resides to any other directory and device the user wishes.) When the "ls" command is issued, the filenames will be prefixed by "<dir>/" if they reside in a pseudo-directory, or NULL if they reside in the main directory. All the existing pseudo-directories can be seen by issuing the "ls" command from the main directory. *Note that one never references a file, in a command, using a pseudo-directory* (<dir>) *prefix.* In contrast, one may always reference a file using a device (<dev>) prefix although it is redundant if the file exists in the current device. All files are always stored physically on the disk in the order in which they were created. Pseudo-directories have no effect whatsoever on the physical location of files. Therefore, the physical order of storage can only be determined from the main directory.

EXAMPLES

cd

typed by itself always changes to the main directory on the current device.

cd g/akh

changes to <dir> = "a" in the "g" device. "kh" is ignored.

cd /
or
cd /<dir>

yields an error message, but

cd g/
is equivalent to
cd g

which changes to the main directory in the "g" device. Creating a pseudo-directory or changing to an existent pseudo-directory are done using the same command; e.g.

cd a/x

brings pseudo-directory "x" into existence on the device "a" if "x" did not previously exist. Otherwise, the user simply changed the current directory to "x".

When the "cd" command is invoked having <dev> on the command line, the ASCII contents of the ".start" file, if existent on <dev>, are printed on the command terminal. The ".start" file is used to convey a particular message concerning that particular device; e.g., ownership or warning messages. (See the "text" and "help" commands.)

The "a" device corresponds to the top platter on the ARIES disk drive, while the "h" device corresponds to the top surface of the lower platter and is reserved for help texts; the "h" device is *not* for general use. The CDC disk drive adds twelve more devices which are all available for general use. The "a" device on the ARIES is capable of holding 4.989 Mbytes while each CDC device can hold 19.92 Mbytes. The ARIES "h" device can hold 2.494 Mbytes. The lower surface of the lower platter on the ARIES is where all the system programs are stored, and it has

the same capacity as the "h" device. It is accessible through the RKDP monitor PDP-program. The files stored on the devices comprising the CDC disk drive are considered permanent and should only be removed or altered by the owners. (The CDC is always powered except for maintenance or repair.) The top platter of the ARIES disk drive is removeable so that each user having their own "disk pack" may have a portable "a"-device storage medium for safe-keeping. A system "scratch" disk is usually in place on the ARIES top platter and is for general use. All files on the scratch disk are considered temporary and subject to removal at any time.

None of the devices are protected against unauthorized usage. In addition, all users have complete read-write access to all files stored on all devices. The ARIES disk drive does have two front-panel switches which protect the upper and lower platters, respectively, from file removal.

NAME

cfa -- change file attributes (rate and/or type)

SYNOPSIS

cfa <attribute> <filename> [<filename> ...]
[<attribute> <filename> [<filename> ...]]...

DESCRIPTION

<attribute> and <filename> correspond directly to the formal syntax; i.e., no "<dev>/" prefix may ever be used with <filename>. This means that the file attributes may be changed from only the device on which a particular file resides. Two attributes can be changed using this command:

→RATE; The actual sample rate of the file data, in Hz; indicated on the long listing (ls -l) under the "RATE" heading. From the syntax, the proper specification is -r<NNNN>.

→TYPE; The proper specification is -m or -s or ... or -d.

m = monophonic file (see ms-command)
s = stereophonic file (see sm-command)
c = complex file (see fft-command)
p = breakpoint file (see "nlo")
i = the index (see "init")
q = queue file (see "sos")
d = data file.

Any file can have its attribute changed to any TYPE but this does not affect the file in any way, physically. These TYPE designators serve to remind the user as to the nature of his/her files and have some use to the software as parameters. Certain commands which generate data output will automatically set the proper (RATE or) TYPE. "d" is the default TYPE. Note that some mathematical commands, such as the fft-command require that a RATE be specified (in the command line or the long listing) so *that* parameter can be used in the calculations. If not specified, floating point errors will develop, and the user will get an error message. In general, when floating point errors occur, the RATE is the first thing that the user should check. The "fft" also checks the file TYPE.

The TYPE and RATE of all files can be found in the long listing (ls -l) in the second two columns. The type designators are capitalized there.

The most common use of the RATE designator is to remind the user of the proper sample rate at which a file should be played (using the out-command). This is especially useful for files which have not been played for some time. Since

the rate at which the binary output port (from the PDP 11/40) produces data, is directly controlled by a continuously variable user-controlled external clock (located on the KLUGE), there is no simple way to predict the rate that any one user may have chosen. So, this information should be entered at some point by the creator of the file. Most all file manipulations thereafter, will carry over (or modify; e.g., in the case of the int-command) RATE (and TYPE) information and the user need not worry about resetting. For example;

cat prexist > newfile

will copy the RATE and TYPE information into the attributes of "newfile" as will most other commands which make new output files from old ones.

EXAMPLES

cfa -r10000 g/junk

will yield an error message because the device designator (g/) is not allowed even if the current device were "g".

in temp -r1000

takes in data from the A/D converter on the KLUGE and assigns a TYPE "M", for Monophonic (even if the stereo switch is set on the input port), and a RATE of 1000 (Hz).

cfa -s -r10000 temp

changes the file TYPE of "temp" to Stereo and the sample rate to 10000 (Hz).

cfa -r10000 -s temp

has the same effect.

cfa -s -m temp

changes the file TYPE to Mono (i.e., ignores -s).

BUGS Using -r<NNNN>, NNNN $\neq \phi$.

NAME

cm -- change the file mark

SYNOPSIS

cm <filename> <newmark> [<filename> <newmark> ...]

where;

<newmark> = [-<start>] OR [+<end>]

DESCRIPTION

The MARK always refers to some specific sample whose location is with respect to the beginning of the corresponding file. The default MARK is the file beginning; i.e., sample number ϕ. Changing the MARK, using "cm", allows one to mark a file in a desired location so as to conveniently command SW2 (or DMC) to perform operations on files with respect to their MARKs; e.g., assuming "junk" is much greater than 100 blocks in length,

out junk +b100

plays "junk" starting 100 blocks after the MARK. The current MARK for each file is shown in the long listing (see "ls").

cm junk -ab1000

places the MARK at the sample which is located precisely 1000 blocks after the beginning of the file "junk".

The choice of the "-" or "+" is only of concern when changing the MARK relative to the old one; the "-" means prior to the MARK while "+" means subsequent to the MARK, *in that case.*

<start> and <end> were defined in the syntax. <NNNN> must be an integer.

To reliably change the MARK back to the beginning of a file, type

cm <filename> +abϕ

Omitting <newmark> from the "cm" command has NULL effect.

There is one other way to mark a file; that is by using the "out" command.

BUGS

+s<NNN> = <newmark> does not work properly; "s" meaning "sample".

±b<NNN> = <newmark> moves the MARK in the appropriate direction by NNN blocks as it is supposed to, but will additionally move the MARK to the first sample of the *next* block if the old MARK was *not* located at the *first* sample of some block with respect to the beginning of the corresponding file; i.e., the MARK always arrives at the first sample of a block.

NAME

con -- connect to a network machine

SYNOPSIS

con <host>

DESCRIPTION

"con" is similar to the UNIX command of the same name.

The link-command must first be used.

("link" is a command in the DMC program.)

This command allows the user to login to UNIX network machines without having to kill *SW2* (or *DMC),* or to change the baud-rate on the Tektronix Terminal! The user is then talking to UNIX in the regular fashion, as if one were using a terminal which was not associated with the ODSP lab. To logoff type "log". A break* may need to be given from the terminal to get back to DMC.

BUGS

The password is not invisible!

*To "break" on the Tek terminal, hit the break key twice fast.

NAME

copy -- copy directory(s) from device to device

SYNOPSIS

copy <dev1>[/<dir>] <dev2>[/<dir>]

DESCRIPTION

All files on device 1 (only those in the pseudo-directory if specified) are copied to device 2 (and put in the specified pseudo-directory). This command also erases the specified files' names in the source directory index if the source and destination devices are the same. This command can be useful for shuffling the physical locations of files on a particular disk. The <dir> on device 2 does not necessarily need to be pre-existent.

BUGS

Puts you back into main directory if your working directory (wd) was a pseudo-directory. This command may result in file loss if there is inadequate available storage (see "rm" command for file recovery) on device 2.

NAME

cpu -- Print CPU (Central Processing Unit) of Network Link

SYNOPSIS

cpu [<UNIX command>]

DESCRIPTION

The command, "cpu", prints (on the command terminal) the current network link (see "link" command) and the working directory being used on that network machine. The <UNIX command> has limited capabilities. Essentially, it allows the user to issue UNIX commands from SWITCH II; e.g., run a UNIX program which might work perhaps on data sent from ODSP. For the "cpu" command to be successful, it must return some printed message to the user or it will hang. The safest use of the command is therefore:

cpu [<UNIX command> ; wd]

because the UNIX command "wd" returns the name of the current network directory. The "cpu" command is only available in the DMC program.

It is wise to include the entire UNIX pathname in the UNIX command, if applicable, for best results. The "cpu" command is also used to increase the filesize limit of a UNIX file since the link does not respect UNIX .profile directives. This is useful when using the "put" command for > 600 blocks. Another common use of the "cpu" command is after a "put" to see if the files sent were created properly; i.e., cpu ls -l.

NAME

crc -- cross correlation function

SYNOPSIS

crc [-b] [<#pts>] <src> [<src>] [> <dest>]

DESCRIPTION

This command implements the equation:

$$R(k) = \frac{1}{B} \sum_{n=0}^{L-1-k} x(n)y(n+k) \qquad (1)$$

	if "-b" flag:	$;k = \frac{-<\#pts>}{2} \rightarrow \frac{<\#pts>}{2} - 1$
; B = L - /k/	;NOT invoked	;L is the length, in samples, of
= L	;invoked	the smallest of the two source files
		;x(n) is the smallest source file
		;y(n) is the largest of the two

The user is prompted for a plot of R(k). The plot shows R(k) as a function of k.

<#pts> defaults to 256. The maximum is 512.

$$\text{Run time (roughly) in seconds} \simeq \frac{L \cdot (<\#pts>) \cdot 15\text{sec.}}{(256)^2}$$

The RATE attribute is not necessary for the successful operation of this command.

The cross correlation becomes an autocorrelation if the second <src> is not specified; i.e., $y(n+k) \rightarrow x(n+k)$ in R(k). R(k) is stored in <dest> if specified, otherwise it is discarded.

To perform a more complete correlation for the range $k = -\frac{<\#pts>}{2} \rightarrow \infty$, crc must be called multiple times varying the <window> on the "y" source each time and then concatenating the <dest> files.

BUGS

The <dest> files are not being created! The algorithm uses straight multiplication and does not incorporate the FFT, even though it resides in the same program file!

NAME

cti -- change the time increment

SYNOPSIS

cti [<1/Δt>]

DESCRIPTION

This command is rarely used, except in conjunction with "pwl". Its argument (inverse), Δt the time increment, is set by default to 0.1 seconds. Although Δt is not changed much, its value impacts many commands and parameters, and the descriptions of many commands make reference to "cti". For example, in the long listing of the index (see "ls"), lengths and marks may be shown in units of time. It is the "cti" command which determines those time units. In fact, in most all circumstances in which time is a parameter, "cti" has direct control of the time units.

If no argument is given on the command line, the current Δt is printed on the command terminal. If the argument is specified by the user as zero, the default time increment is used instead.

Note: 1/Δt is constrained to be an integer. The time increment is, of course, changed to be the specified argument inverse. Once specified, it remains fixed until changed or until DMC or SWITCH II is again booted.

NAME

eplot -- envelope plot (graphics)

SYNOPSIS

eplot [<# of bins>] [-z] [-i<N>] <src> [<src> ...]
; -z flag plots a zero reference line,
; -i parameter sets the number of y and x-axis tick marks to N+1 (default is N=4).

DESCRIPTION

Each <src> is assumed to be the same length. Each source file is divided (at the sample level) into the specified number of bins. Each bin is searched for both its maximum and its minimum, and that point pair is plotted. Adjacent max points and adjacent min points are respectively connected by a straight line. Each point is placed at the beginning sample of its bin on the graph. The x-axis coordinates are sample numbers, *not* bin numbers. The number of plot point pairs for each <src> is equal to the number of bins. The default number of bins is 256 so that if, for example, an envelope plot of a 1 block file were drawn using the bin default, the envelope plot would be the same as the plot of the original file itself. Therefore, smaller files tend to require fewer bins to characterize their envelopes. The entirety of each file is represented in only one plot, and the maximum number of bins is 2048, while the practical minimum is 2. The plot of each <src> is superimposed upon the others. The plots can be terminated prematurely by a <cr>. The time taken to plot is proportional to the number of bins specified.

BUGS

Floating point error on a constant value file. The ordinate max and min are taken from the first <src> and must be $\geq$ those of the other <src> files. If random video patterns appear, press the master reset on the RHS of the Tek Terminal.

NAME

fft -- Fast Fourier Transform

SYNOPSIS

fft [<# of transform points>] <src> [<src> ...] [> <dest>]

DESCRIPTION

The DFT is defined as

$$X(k) = \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi k}{N}n} \quad ;N = \#\text{transform points.}$$

This command can accommodate a maximum of N = 1024. N defaults to 256. Each source file is segmented into Q bins, each bin of length N. The resultant DFT is the phase adjusted sum of the DFT of each bin. If $X_q(k)$ is the DFT of bin number q, then it can be shown that

$$\sum_{q=0}^{Q-1} X_q(k) = \sum_{n=0}^{QN-1} x_n e^{-j\frac{2\pi k}{N}n}$$

$$= \sum_{q=0}^{Q-1} e^{-j2\pi kq} \left[\sum_{n=0}^{N-1} x_{n+qN} e^{-j\frac{2\pi k}{N}n} \right]$$

where the part in brackets is simply the DFT of the qth bin. ($N \neq 0$, $Q \neq 0$) Note that the frequency resolution remains at f_{sample}/N regardless of Q. Of course, the user can implicitly set Q to 1 by specifying the "number of transform points" as being equal to the windowed (see syntax) source file length, in samples. (This command will honor windows specified in samples.) Bins of sample length less than N are zero-padded.

If a destination file is indicated, the complex result will be stored there in floating point pairs; i.e., Re[X(0)],Im[X(0)],Re[X(1)],Im[X(1)],...,Re[X(N-1)],Im[X(N-1)]. Otherwise, the result is discarded. In any case, the user is prompted for a request to plot the magnitude or log magnitude of the DFT. (Phase plots are not available.) The user is also prompted for plot-frequency MAX and MIN; the default is the Nyquist and zero frequency which are used in the case of a lone carriage return. The maximum plot frequency honored is the Nyquist frequency.

If a source file is TYPE "C" (complex) (see long listing), it is assumed to be the result of a previous "fft" for which only a plot is now desired. To recover the original plot from a TYPE "C" source file, the number of transform points specified in the original command must be specified again, or else the plot will not be accurate with respect to the frequency scale. The plot is escaped with the use

of a <cr>.

This command does not employ any window functions (see "hw", "bh3w", "bh4w").

BUGS

If the source file is longer than approximately 64 blocks, a floating-point error will result. The solution is to use a <window> as defined in the syntax *(a zero RATE attribute for <src> in the long listing will also cause a floating-point error* (see "cfa" command)); e.g., fft 1024 junk +ab64

NAME

fm -- frequency modulation

SYNOPSIS

fm <src> [<src> ...] > <dest>

DESCRIPTION

The classical equation for FM is

$$x_c(t) = A_c \cos\left[\omega_c t + 2\pi f_\Delta \int_0^t x(\gamma)d\gamma\right] \tag{1}$$

; where $|x(t)| \leq 1$, $x(t)$ is the modulating signal, $x(t) = 0$ for $t < 0$, ω_c is the carrier radian-frequency, f_Δ is the frequency deviation constant, A_c is the carrier amplitude.

The trapezoidal rule for the approximation of an integral,

$$y(t) = \int_0^t x(\gamma)d\gamma$$

is

$$y(n\Delta T) \simeq \Delta T\left[\frac{x(0)}{2} + \sum_{k=1}^{n-1} x(k\Delta T) + \frac{x(n\Delta T)}{2}\right] \tag{2}$$

This command implements the equation:

$$x_c(nT) = 32767 \cos\left[\omega_c nT + I\{\sum_{k=0}^{n-1} x(kT) + \frac{x(nT)}{2}\}\right] \tag{3}$$

where T is the sample period. The user is prompted first for the carrier frequency, f_c, and then for the "modulation index," I. (By definition, "I" is not truly the modulation index which is only defined for (single) tone modulation of the carrier frequency.) The modulation index (entered in floating point) must be chosen carefully; for this, we must define its constituents.

In the ODSP Lab, *audio* files are composed of samples whose dynamic range is -2048 to +2047. Any file, however, may have values ranging from -32767 to +32767. To comply with the classical equation (1), we should normalize x(nT) by $|x(nT)|_{max}$. We can always find this using the "max" command. Next, we should rewrite (1) using (2).

$$x_c(nT)=A_c \cos\left[\omega_c nT + \frac{2\pi f_\Delta T}{|x(nT)|_{max}} \left\{\frac{x(0)}{2} + \sum_{k=1}^{n-1} x(kT) + \frac{x(nT)}{2}\right\}\right] \qquad (4)$$

where n = 1, 2, 3, ... Comparing Eq. (4) with (3), we see that A_c has been set to 32767. We can use the "sc" command to bring this down to audio levels. We also see that the x(0)/2 factor has been changed to x(0). The effect of this change will be to produce a carrier frequency shift equal to

$$\frac{f_\Delta x(0)}{2|x(nT)|_{max}} .$$

Also notice that (3) is defined for n=0 whereas (4) is not. (From (1), we see that $x_c(0) = A_c$.) If x(0) = 0, then (3) will satisfy (1) at t = 0 (n = 0), and it will be equivalent to (4) for n > 0 provided that

$$\boxed{I = \frac{2\pi f_\Delta T}{|x(nT)|_{max}}}$$

for which the software prompts using the nomenclature, "index of modulation".

The concatenated source files are used as the modulating signal (x(t)) to produce a frequency modulated cosinusoid of constant amplitude (see Eq. (3)) whose sampling rate is equal to that of the first <src> file, and whose length is equal to the sum of the time lengths of the <src> files. The result is put into <dest>.

If a zero modulation index is specified, (I=0) the user is further prompted for the desired <dest> file length in seconds (not tenths of) specified in floating point. This length must be less than or equal to the length of the concatenated <src> files. Note that a zero index of modulation provides a convenient way to produce a cosinusoid at the carrier frequency.

A floating error will occur under several circumstances:

1) the first <src> sample rate is at default (ϕ) in the listing of the index (use "ls -l", look under *RATE)* (see "cfa"),
2) Case I=0; the specified destination-file time-length is greater than that of the concatenated <src> files,
3) I is too large,
4) x(nT) has a large DC component which is accumulating under the integral (use "stat" and "ab").

The source files remain intact. All calculations are performed in floating point with the results properly rounded to the nearest integer.

Reference: Communication Systems by A. B. Carlson, 2nd Edition, pgs. 220-233.

NAME

get -- get a file from a directory on a network host

SYNOPSIS

get [<host>] <UNIX> <dest>

DESCRIPTION

The "link" command must be used before this command can be invoked. This means that this command is only available when the DMC program is running. This command brings up one file from a UNIX network machine and places it in <dest>. The file size is not limited.

The PDP 11/40 in the ODSP lab has a direct 1 megabaud line to all network machines so that large files may be transferred in a relatively short time. This command does not destroy the link (see "ul") so that it may be used repeatedly.

EXAMPLE:

get /c/dattorro/<dir>/<sub dir>/<filename> <new filename>

The <host> option must be used whenever a link command was used where the user answered the login and password prompt with only a <cr>.

BUGS

This command has been known to drop a block occasionally. As a precaution, it is wise to specify entire UNIX pathnames although this is not strictly necessary. One should also use the CPU command to compare the filesize received with that sent; e.g.,

cpu ls -l

NAME

help -- prints command documentation for the user (prints ASCII files)

SYNOPSIS

help <command> (help <filename>)

DESCRIPTION

Documentation concerning the indicated command is printed on the terminal screen. The device h/ (on the bottom platter on the ARIES disk drive) is searched for the file name "command". If not found there, the current device is searched. The ASCII contents of the "command" file are then printed out for the user.

In conjunction with the "text" command, the user is able to create a personal help text. An application would be that of sending mail to other users; e.g., typing "text jon" allows the user to make an ASCII file named "jon". When the user, jon, finds this file in his pseudo-directory, he can then type

help jon

to reveal the message contents.

BUGS

If an ASCII file has appended zeroes (e.g., an ASCII UNIX file), "help" will repeat part of the original file (corresponding to the zeroes).

NAME

hw -- Hamming window

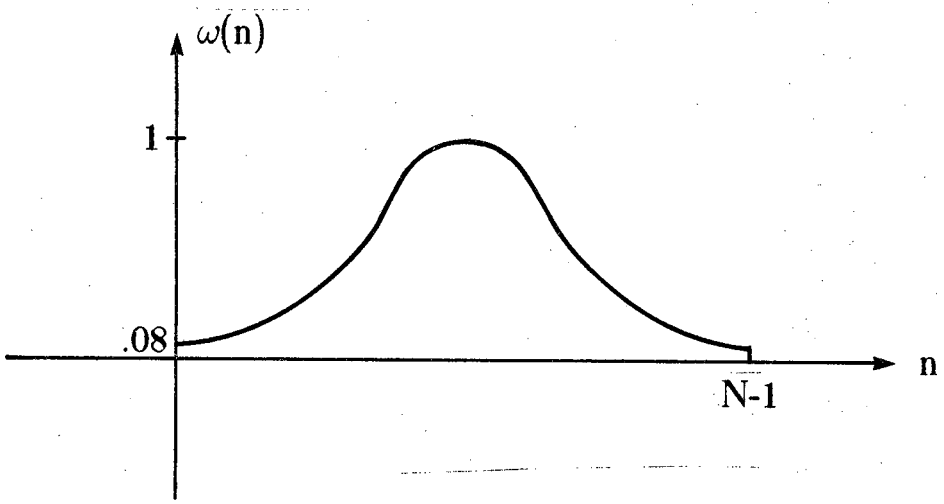
SYNOPSIS

hw <src> [<src> ...]

DESCRIPTION

The Hamming window is applied to each of the specified source files or any specified part thereof. (See bh3w for an expanded general discussion.) Of course, the default <window> is the entire file but may be specified down to the sample. ***The original source file is destroyed as a result of this command.*** Note that the mathematics for windows is, in general, carried out in floating point and then the proper rounding (down for negative samples) is applied to give the result to the nearest integer.

$$\omega(n) = .54 + .46 \cos\left[\frac{\pi(n - (\frac{N-1}{2}))}{(\frac{N-1}{2})}\right] \quad \begin{array}{l} ;n = 0 \rightarrow N - 1 \\ ;N = \text{total number of samples} \\ ;\left[(\frac{N-1}{2}) \text{ may be fractional}\right] \end{array}$$



Figure

NAME

in -- input (through DMA) to disk

SYNOPSIS

in [<dest>]

DESCRIPTION

This command enables users to A/D convert one-dimensional signals. Samples are taken from the DMA until another <cr> is typed. The DMA input port lies immediately below the panel labelled "receiver terminations" in the ODSP Lab. It takes in 16-bit binary numbers and stores them in the named destination file exactly as they appeared at that port. The number of blocks (1 block = 256 samples) taken in is shown in real time on the terminal screen during input. The rate of input is controlled by the shift register clock of the KLUGE in the ODSP Lab, whose output is used to strobe the "A" input port channel register. All files are stored in whole units of blocks so that the smallest file creatable is one block. This means that zeroes are appended, in general, to a file having a partially filled block at the end. The usefulness of the <window> specification in the syntax is apparent for some situations but, in general, the appended zeroes cause no problems in subsequent (different) commands.

The most frequent connection to the input port is from the A/D converter output. This is because we are most often interested in storing digitized audio files. (See the "get" command as an example of storing other file types.) If no destination file is specified, the destination from the previous "in" is used. The user should use the "cfa" command or the <attribute> of <dest> to assign a sample rate (and file type) since some of the mathematical commands (e.g., fft) require this information and floating point errors will result if it is set to zero (by default).

In general, one uses register "A" on the input port for monaural file generation. Even though the A/D is a twelve-bit device, it has sign extension so as to be compatible with the input port. Input files have a monaural (*M*ono) type assigned by default as shown in the long listing. Since each sample occupies two bytes worth of storage, the corresponding data type on the UNIX network is "short" for all network machines.

There are two channels on the single pdp input port. To input a stereo file, one alternatively strobes the two input port channel registers (which may each have separately incoming digital data), and sets the stereo (MUX) switch on the input port to "Yes". If one sets the stereo switch to "No," data will not be taken from the B channel. The strobe pulse to that channel, however, ***is not ignored*** and will cause data to be loaded from the A channel register to the input port, and then stored. (The strobe pulses always determine the *storage* rate.) Thus, *when*

storing mono files, be sure to disconnect the B channel strobe pulse! A further implication is that a stereo file might be created using only one strobe pulse while having the stereo switch set to "Yes". This is *not* true, because each strobe pulse must also load its respective channel register with incoming digital data. Normally each strobe is at the sample rate desired but 180 ° out of phase with each other. Note that it is not possible to use channel B for mono file input. Stereo files are stored such that the samples from each channel are interleaved in the <dest> file.

Provisions have been made for stereo to mono file conversion and vice versa; see "sm" and "ms". The file type is *not* set to stereo (S) during stereo input. Commands, such as "max," treat a stereo file as if it were a mono file.

Executing an "in" using a <dest> which already exists, moves the name of <dest> to the last position in the index. If that pre-existing <dest> file was physically last on the disk, then the further result of destroying that old <dest> file is incurred (see "rm").

NOTES:

A peculiarity of the ODSP Lab I/O Ports is that during an "in," the data at the input port also appears at the output port, and vice versa for "out". This can be used to advantage to monitor the digitized input signal by connecting the output port to the D/A (for the "in" case). In this manner, we can monitor the results of the digitization in real-time.

BUGS

If the DMA *output* port (located next to the input port) is not clocked, the first 1/2 block of <dest> will be equal to the last 1/2 block of the previous "in" destination; i.e. garbage. So make sure that the output port is always being clocked during input.

NAME

init -- clear logical device index, and optionally format Aries cartridge platter.

SYNOPSIS

init [<dev>] [-f]

DESCRIPTION

Use of the "-f" option will allow the Aries removable platter to be formatted. During execution, the user will be required to confirm that this formatting is indeed to be carried out. The CDC drive (logical devices b-g and i-n) cannot be formatted on this system. If need be, this can be done on one of the UNIX network machines.

Upon completion of this command, the index (directory) of the affected logical device will be "zeroed". Thus the system software will view this device as containing no files. (See the "rm" command for information on recovery of lost files.) Obviously, this command will be used sparingly during the course of a working disk's lifetime; usually when a cartridge disk changes ownership. Recall that the current contents of an index can be viewed using the "ls" command.

If the <dev> option is not used, the current logical device (as per "pwd") will be affected. Otherwise, the logical device mentioned (a, b, c, ...) will undergo initialization. Use of the "-f" option for devices other than "a" (Aries removable) will result only in the zeroing of the affected directory. Bear in mind that formatting is mandatory before attempting to use a brand new Aries cartridge.

NAME

int -- interpolate (change the sample rate of an audio file)

SYNOPSIS

int [-l<NNN>] [-i] [-m<NNN>] [<NNN>] <src> > <dest>

DESCRIPTION

This command, residing in the DMC program shares software with the "nrf" command. Consequently, the prompts have the same meaning. The purpose of this command is to change the sample rate of a digitized audio file, which means that new samples may need to be interspersed among the old ones, or perhaps some old samples need to be discarded altogether being ousted by new ones. One should read "A DIGITAL SIGNAL PROCESSING APPROACH TO INTERPOLATION" by R. W. Schafer and L. R. Rabiner, in the Proceedings of the IEEE, Vol. 61, No. 6, June 1973, to understand how FIR filters facilitate this process. The meanings of the elements in the synopsis are the same as those of the corresponding elements in the "nrf" synopsis. The two differences here are the "-l" and the "-m" parameters. Assuming that the <NNN> corresponding to each parameter gets assigned, respectively, to the variables "l" and "m", then

$$fs_{OLD}\left(\frac{l}{m}\right) = fs_{NEW} \quad ,$$

where fs_{OLD}, the old sample rate, is obtained from the <src> file rate attribute. The result is placed in <dest> which has a sample rate = fs_{NEW}. "l" and "m" default to 1. The size of <dest> is $\simeq(l/m) \cdot$ <src>. The user is not prompted for breakpoints as in "nrf" because the software determines the filter specs from "l" and "m". Be sure to choose a large enough abscissa when plotting. "-i" is not needed; see the "nrf" command.

The lone <NNN> specifies the number of tap weights used in the FIR filter design.

NAME

key -- piano keyboard simulation

SYNOPSIS

key [<suffix>]

DESCRIPTION

This command facilitates the output of files (via the DMA) by single keystroke. After the command has been invoked, the file having <filename> = <keystroke><suffix> is output. No carriage return is required after each keystroke and the user may use as many keystrokes as desired. A single carriage return will exit the command.

If <filename> is not found, nothing is output.

NAME

link -- connection to a UNIX-network host

SYNOPSIS

link [<host>]

DESCRIPTION

The default host is the ec-machine. <host> is the mnemonic for the network machines; e.g., ea, ec, ed, etc.

This command is used when it is desired to have the PDP-11/40 talk to one network machine for the purpose of, say, transferring data between the two machines; to or from some user's account. This command must be used as a preliminary to other DMC-Program commands such as "get" or "put". If no "put"'s are desired, the user can respond to the UNIX login prompts with carriage returns.

In the case of an unsuccessful link, two things can be tried:

1) Interrupt via break-break key, then unlink via "ul" command.
2) Restart the PDP via the "log" command or manually (773110).

-- then try linking again.

It is wise to use the "ul" command before any link is made if the user is unsure as to whether a previous link exists. Unsuccessful links are recognized by the failure of the login prompt to appear.

EXAMPLE

A typical transaction would appear like so:

```
% link ee<cr>                    (user)
MIDMCO up
connect to host:15

Login: dattorro                  (prompts)
Password:

Tue Mar 6    13:56:40  EST  1984

%
```

NAME

log -- return to PDP-system monitor

SYNOPSIS

log

DESCRIPTION

When this command is issued from either the SW2 or DMC program, the results are equivalent to having manually started the PDP using starting-address 773110. The effect is to return to the PDP system monitor. The main reason for the inclusion of this command into the SW2 repertoire is to facilitate changing programs (SW2 & DMC). The SW2 program is the main signal processing program used in the ODSP Lab, whereas the DMC program contains a subset of the SW2 commands in addition to commands which enable the direct connection of the ODSP lab PDP 11/40 to the UNIX network. Due to the size of the SW2 program, it was not possible to consolidate the two programs. The user need not remember which commands are germane to each program. Both SW2 and DMC will generate the appropriate message when a particular command is only available in the other program.

NAME

ls -- list index for current device and directory

SYNOPSIS

ls [-l, -t, -w] [[<filename> ....] , [<dir>/]]

DESCRIPTION

If "ls" alone is typed, the file names in the current directory (or only those in the current pseudo-directory (see "cd")) are listed. All files are always listed in their physical order of location on the disk which is the reason for referring to the complete long-listing as an "index". If filenames are specified, only those files appear in the listing. If <dir> is specified instead, then only those files in the specified pseudo-directory appear in the listing.

This command is similar to the UNIX command of the same name. If *any* of the options are invoked, the *long* listing will appear having the following format:

FILENAME	TYPE	RATE	START	LENGTH	MARK

If the current directory is the main directory, the long listing will be complete. The options only affect the units of LENGTH and MARK The -l option yields units of blocks, the -t yields time, and -w yields words. LENGTH refers to the length of the corresponding file, while MARK is a user prescribed sample location with respect to the beginning of the corresponding file (see "cm" command).

A block is 256 samples = 512 bytes. A word is 2 bytes = sample. The units of TIME are set by the "cti" command. The default unit is 0.1 seconds so that times usually can be read directly in tenths of seconds. A (default) sample RATE of zero will yield zero TIME. The units of RATE are Hz. START is the starting location of a file, in blocks, with respect to the physical beginning of storage (block location 1) on the device. (The START number is useful when recovering accidentally erased files (see "rm"), as also is the LENGTH.)

TYPE refers to the type <attribute> of the data stored in the corresponding file. The various types were discussed under the "cfa" command. FILENAME is determined by the user.

When "<alphabetic character>/" precedes a filename in a listing, this means that the file exists in a pseudo-directory which is identified by that alphabetic character; i.e., <dir> = <alphabetic character>.

BUGS

The word "FILENAME" is sometimes mis-spelled.

NAME

max -- find the maximum and minimum of each specified file

SYNOPSIS

max <src> [<src> ....]

DESCRIPTION

This command prints each specified filename followed by the values of the largest and smallest sample found in the respective file. Each file is considered to be composed of 2-byte two's complement integers so that this command is only meaningful for those types of files. A stereo file is treated as a monaural file.

Run time $\simeq$ 29 msec/block .

The source files are unaltered by this command.

NAME

mf -- multiply files together

SYNOPSIS

mf <src> [<src> ....] > <dest>

DESCRIPTION

This command is much like the "af" command except that here, the specified source files are *multiplied.* The result is stored in the named destination file, and the source files remain unchanged. If N = the number of specified source files, then

$$<\text{dest}> = \left\lfloor \frac{<\text{src1}> \cdot <\text{src2}> \cdot \ldots \cdot <\text{srcN}>}{(32768.)^{N-1}} \right\rfloor .$$

The length of <dest> will be the length of the *smallest* source file. If N = 1 then <src> will be multiplied by 1.

Execution time is $\simeq$ 330ms/block.

NAME

ms -- mono to stereo file conversion

SYNOPSIS

ms <src> <src> > <dest>

DESCRIPTION

A stereo file is created from the two source files specified, by interleaving the source file samples and then placing the result in the specified destination file. The sourcefiles are unchanged by this command. To play a stereo file in the ODSP lab, one alternately strobes the two channels of the single PDP output port, and sends each channel to a separate D/A converter. There is, therefore, a one-half sample constant time delay between the two channels.

The size of the destination file will be twice that of the *smallest* source file. This implies that the command terminates when it reaches EOF on either file. In this case, the smaller of the two source files should be padded with zeroes to make them equal in length (see "pwl"). Use "cat" to pad with a zero file.

NAME

mv -- change the name of a file

SYNOPSIS

mv <filename - old> <filename - new>

DESCRIPTION

This command changes the name of <filename - old> to <filename - new>, but does not change the physical location of <filename - old>. If <filename - new> is omitted, the filename will remain the same but the directory in which <filename - old> resides will be changed to the current directory. This is convenient since files may *never* be referenced using a pseudo-directory prefix. (Prefixes *always* refer to disk-drive devices.)

Alternatively, note that

mv <filename - old>

is equivalent to

mv <filename - old> < filename - old>

If <filename - new> exists before the command is executed, its name is deleted from the old index although its physical existence is not altered (see "rm" command). Note from the syntax that it is not possible to "mv" filenames across devices.

NAME

naf -- nonrecursive amplitude filter (median filtering)

SYNOPSIS

naf [<tap number>] <src> [<src> ...]

DESCRIPTION

This command performs a nonlinear operation on a file. The final results are placed back into the respective source file so that the original file is lost. The user is prompted for the weight of each tap in a tap window. The length (number of taps) of the tap window is determined by the number of weights entered, so all zero (tap) weights must be entered. All weights may be entered as floating point numbers separated by a space or a carriage return:

$$a_0\ a_1\ \ldots\ a_{w-2}\ a_{w-1}\text{<cr><cr>}$$

Two carriage returns terminate tap weight input.

The significance of <tap number> can best be understood by example: Suppose the user to have entered three tap weights,

(tap window) | a_0 | a_1 | a_2 |

and to have set <tap number> = 1. Further, assume <src> consists of the following samples:

(<src>) | 5 | 1 | 2 | 4 | 3 |

The algorithm begins by ordering the 1st three samples (tap-window-length samples) from highest to lowest and then placing that ranking in a buffer,

(buffer) | 5 | 2 | 1 |

The tap weights are then applied to that buffer

(buffer)		5 | 2 | 1
(tap window)	×	a_0 | a_1 | a_2
(result)	=	$a_0 5 + a_1 2 + a_2$

and the result is placed in a dummy file of the same length as <src>;

(dummy)	$a_0 5 + a_1 2 + a_2$				

On the second iteration, the tap window is shifted to the right *by one sample* and the process is repeated:

(buffer)	4	2	1
(tap window) ×	a_0	a_1	a_2

The dummy becomes

(dummy)	$a_0 5 + a_1 2 + a_2$	$a_0 4 + a_1 2 + a_2$			

The final result (in "dummy") gets put back into <src> so that at the end we have:

(<src>)	$a_0 5 + a_1 2 + a_2$	$a_0 4 + a_1 2 + a_2$	$a_0 4 + a_1 3 + a_2 2$	$a_0 4 + a_1 3$	$a_0 3$

Now, let us repeat the example above but with <tap number> = 2. The first iteration has the tap window shifted left with respect to the source file by 1 sample ((<tap number > -1) samples, in general). So, to the tap window, the source file now appears like so:

(Imaginary Source File)	(0)	5	1	2	4	3

The algorithm begins by ordering the 1st three samples (including the imaginary sample) from highest to lowest and then placing that ranking in a buffer,

(buffer)	5	1	0

The tap weights are then applied to that buffer

(buffer)		5	1	0
(tap window)	×	a_0	a_1	a_2
(result)	=	$a_0 5 + a_1$		

and the result is placed in a dummy file of the same length as <src> (the real <src>).

(dummy)

$a_0 5 + a_1$				

On all subsequent iterations,
the tap window is shifted by one sample to the right,
with respect to the source file,
and the process is repeated.
The number of iterations is always equal to the number
of samples in the source file.
For the example above,
the dummy file becomes the <src> file at command completion
and looks like

(<src>)

$a_0 5 + a_1$	$a_0 5 + a_1 2 + a_2$	$a_0 4 + a_1 2 + a_2$	$a_0 4 + a_1 3 + a_2 2$	$a_0 4 + a_1 3$

This command is nonrecursive because only the original <src> samples are involved in the ranking and weighting processes.

In general, we have seen that the effect of <tap number> is to shift the tap window to the left, with respect to the source file by <tap number> -1 places for the first iteration. The effect of changing <tap number> was observed in the preceding example, but the effects are more dramatic in the related command "raf" (recursive amplitude filter). Also, in general, the result of the nth iteration always gets placed back into the source file (at command termination; i.e., nonrecursively) at the nth location; n=0, 1, 2 .... Again, in general, each iteration encompasses only tap-window-length samples, and those samples are ranked from highest to lowest before weighting occurs.

It may be obvious, at this point, that this command will perform (nonrecursive) median filtering simply by making all taps equal to zero except for one tap, usually one in the middle, which is set to unit value.

<tap number> defaults to

$$\begin{cases} \lfloor w/2 \rfloor + 1 & ; \quad w \text{ odd} \\ w/2 & ; \quad w \text{ even} \end{cases}$$

; where w is the tap window length.

Further,

$$1 \leq \text{<tap number>} \leq w.$$

Note: N. C. Gallagher (Purdue EE Prof.) is an expert on the topic of median filtering.

NAME

nlf -- Nonrecursive Linear Filter (FIR Filtering)

SYNOPSIS

nlf [<tap>] <src> [<src> ...]

DESCRIPTION

This command implements the equation:

$$y(kT) = \sum_{n=1}^{w} a_{w-n} x((k+n-tap)T) \qquad (1)$$

; where

y ≡ output of FIR filter
tap ≡ tap window shift factor
w ≡ tap window length (# of filter coefficients)
x ≡ <src> input to FIR filter
a ≡ tap weights (filter coefs).

The final results are placed back into the respective source files so that the original contents are lost. The user is prompted for the FIR filter coefficients which should be entered in reverse order and separated by a blank (space) or a <cr>; i.e.,

$$a_{w-1}\ a_{w-2} \ldots a_1\ a_0 <cr><cr>$$

The coefficients may be entered in floating point. Note that filters having constant phase delay have symmetrical impulse responses; therefore the order of coefficient entry becomes irrelevant. Two carriage returns in succession terminate input. The number of coefficients, w, is determined by the number of coefficients entered, so all zero coefficients must be entered. "w" may also be thought of as the FIR filter length.

<tap> defaults to

$$\begin{cases} \lfloor w/2 \rfloor + 1 & ; \quad w \text{ odd} \\ w/2 & ; \quad w \text{ even} \end{cases}$$

and

$$1 \leq tap \leq w.$$

<tap> is not a critical parameter to this command and is included only because the "nlf" software is shared by "rlf," "naf," and "raf". By observation of (1) it can be seen that for <tap> = w

$$y(kT) = \sum_{n=1}^{w} a_{w-n}x(k+n-w)$$

$$= \sum_{n=1-w}^{o} a_{-n}x(k+n)$$

$$= \sum_{n=0}^{w-1} a_n x((k-n)T)$$

which is the classical convolution. For any other value of <tap>, the a_n sequence is convoluted with the time shifted x(n) sequence. This causes a linear phase shift of the output spectrum so that if phase is not important, one need not specify <tap>. One should see the "rlf" command for a discussion of the software implementation. In contrast to the "rlf" command, the results of each iteration are not placed back into the <src> file until the command reaches completion. It will *always* be found, however, that the results of the nth iteration were placed back into the nth sample location; n=0,1,2 .... Also, the number of iterations is equal to the number of source file samples.

For (1), remember that any terms whose arguments are negative, go to zero; e.g., y(-1) = 0. This command shares software with "rlf," "naf," and "raf".

NAME

nlo -- nonlinear operator

SYNOPSIS

nlo <src> [<src> ...] [[> <p-dest>] , [< <p-src>]]

DESCRIPTION

The user specifies a piece-wise linear function. This function then operates on the specified source file(s), and the result is placed back into <src> replacing the original contents. Looking at figure 1 we see a piece-wise linear function composed of 5 (linear) segments. This function is arbitrary (as is the number of segments) and does not necessarily pass through the origin. Suppose this function could be written in terms of x as f(x), then we could say that y = f(x) where x is a <src> file sample value and y is the sample value of the result. In this manner, all the <src> samples are transformed to new values specified by f(x). The user specifies f(x) by entering the breakpoints (the Cartesian coordinates, in the x-y plane, of the discontinuities) as integers. It is understood that each segment between adjacent breakpoints is linear, and that two breakpoints are always implicitly specified by the software; those are, (-32767,32767) and (32767,32767). This means that the last breakpoint on each side of the y-axis will be extrapolated, linearly. If the range of the specified breakpoints (range of x in (x,y)) is $\geq$ range of <src>, then the implicit breakpoints have no effect on the result.

Let's consider some examples: Suppose the user enters the breakpoints in figure 2, then f(x) = x and y = x; i.e., <src> is unchanged. Next consider Figure 3. The last graph can be represented by the equation

$$y = |x| \quad ,$$

so the absolute value of the <src> will be taken. Continuing in the same manner, consider Figure 4. Thus, the <src> samples will be replaced by their squared values as long as the |samples| are ≤ 3. The user needs to specify more breakpoints to approximate $y = x^2$ over a larger range.

The user is not constrained by the number or location of the breakpoints, nor by the order of entry. The user is prompted for the breakpoints. If <p-dest> is specified, the breakpoints will be stored there for future use. If <p-src> is specified, the user is *not* prompted for the breakpoints; rather the breakpoints are taken from <p-src> which was previously created by "nlo" having a <p-dest> on the command line. The <p-dest> type attribute is "P". The breakpoints are entered as "x_i y_i" with each number separated by a space or a <cr>. Two suc-

cessive <cr>'s terminate breakpoint input.

BUGS

One breakpoint does not produce dependable results. If the first breakpoint entered is (0,0), a floating error occurs.

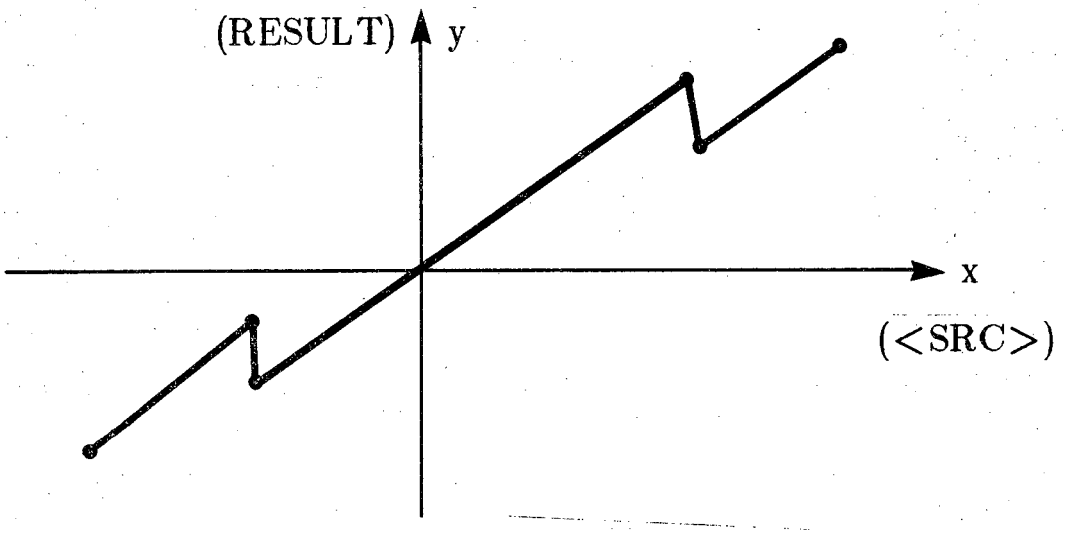


Figure 1

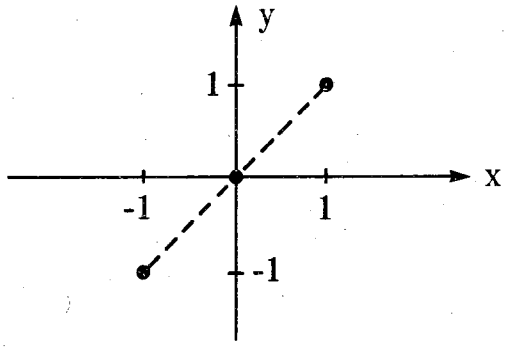


Figure 2

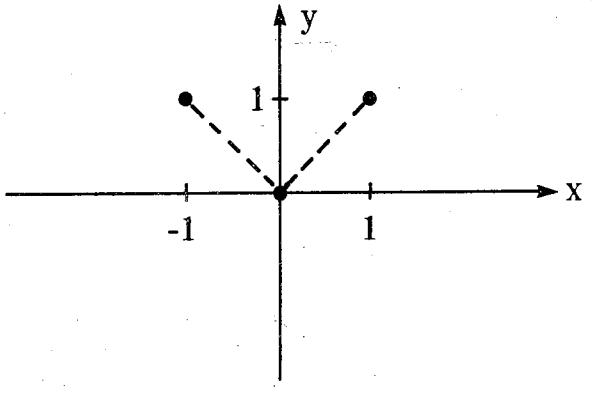


Figure 3

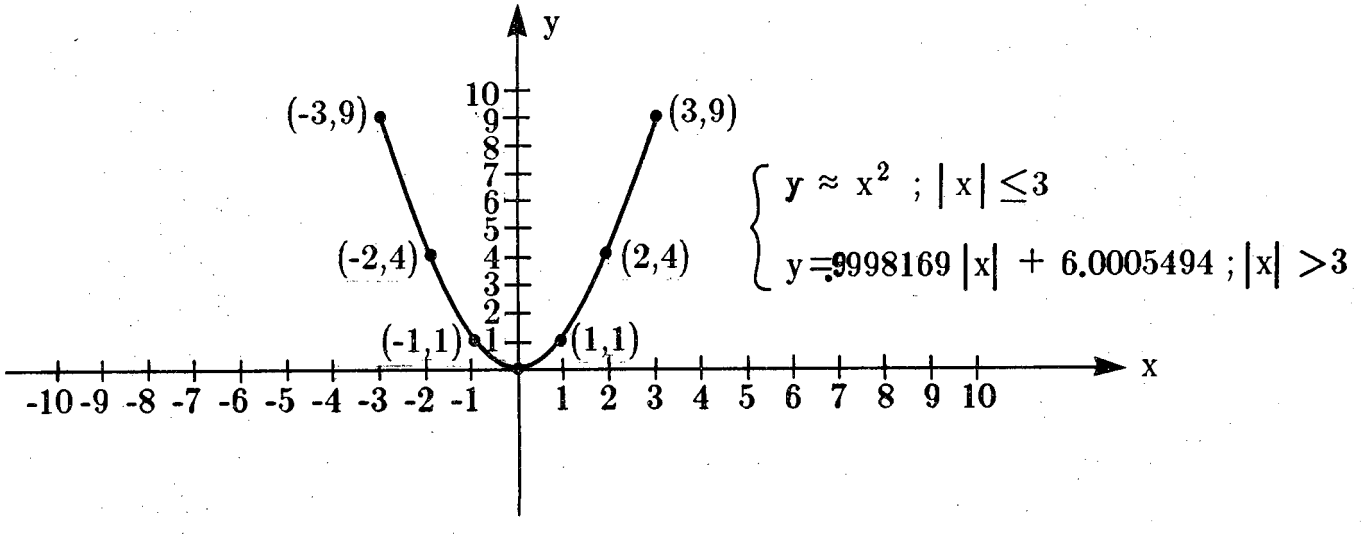


Figure 4

NAME

nrf -- Nonrecursive Filtering and Constant Delay FIR Design

SYNOPSIS

nrf [<NNN>] [-i] <src> > <dest> [< <tapsfile>]

DESCRIPTION

Residing in the DMC program, this command allows the user to give the (floating point) frequency domain specifications of a constant delay (linear phase) FIR filter, and to then filter <src> (placing the result in <dest>) using the FIR filter designed by the software which was given the constraint that the number of taps be equal to <NNN>. <NNN> *defaults to 101, and it must be odd having a maximum of 251.* The "-i" option informs the software that the frequency domain specifications, for which the user is prompted, are pure imaginary. Otherwise, the specifications are pure real. This is necessary to achieve the constant delay (linear phase) filter characteristic [1]. Also remember that for a real impulse response, a real spectrum must be even and an imaginary spectrum must be odd. Only the right hand side of the spectrum needs to be entered, however, remembering that the spectrum is periodic in the sample frequency. Before <src> is actually filtered, the user is prompted to see the magnitude of the transfer characteristic of the newly designed FIR filter, *or* to weight the taps of the newly designed FIR filter. If the user responds with "b", a 3-term Blackman-Harris window will be applied to FIR taps. If the user responds with "h", a Hamming window will be applied. If the user responds with "k", a Kaiser window will be applied after the user responds to a second prompt for the Kaiser parameter (w(a)nt) which is usually between 2 and 9. If the user responds with "o", the original taps (unweighted) will be recovered. If the user responds with "p", the FIR transfer magnitude will be plotted. If the user is not happy with the results of the plot because of the weighting, he/she should type "o" to recover the original unweighted taps, then weight the taps in any way desired, and then re-plot the transfer characteristic. In this way, the user can play with the weighting and graphically display the results before any filtering is actually performed. (To change the number of tap weights, the user must reissue the command.) For plotting ("p"), the user is further prompted for plot start and plot stop frequencies; these default to zero. The user is also prompted for the number of calculation points in the displayed transfer function; the greater this number is, the greater the resolution of the *graph* will be. This entry does *not* affect the FIR filter design in any way. For the plot, the user is additionally prompted for a plot in decibels ("d"), magnitude ("m"), or a numerical listing of the plot points, in both magnitude and decibels vs. frequency, on the command terminal ("t").

The sample rate used in the FIR design is obtained from the RATE attribute of <src>. A floating error will result if <src>'s rate is zero.

If <tapsfile> is specified, one of two things will happen dependent upon whether <tapsfile> existed prior to invoking the command:

Case 1: *(Nonexistent)* The (weighted) FIR taps will be placed in the (floating-point) <tapsfile> (having type attribute "P"),

Case 2: *(Pre-existent)* The taps in <tapsfile> will be used in the FIR filter (i.e., no FIR filter will be designed). The first prompt will be "window or plot ...".

Since the maximum number of taps is 251, and since each tap is a floating point (4 bytes) number, <tapsfile> is always 2 blocks long.

EXAMPLE (See Figure 1)

The user need only specify two breakpoints:

(0,1)

(f_c, 1)

The software assumes, in this case, a third breakpoint (f_c^+, 0) and interpolates to ($f_s/2$, 0).

A lone <cr> escapes prompts. In this command, a lone <cr> will, additionally, move the program to the next phase of processing.

This command should be compared with the "nlf" command. There, the actual FIR tap weights are entered by the user. This command shares software with the "int" command.

[1] *DIGITAL FILTERS:* by Antoniou, pub: McGraw, pgs 218-226

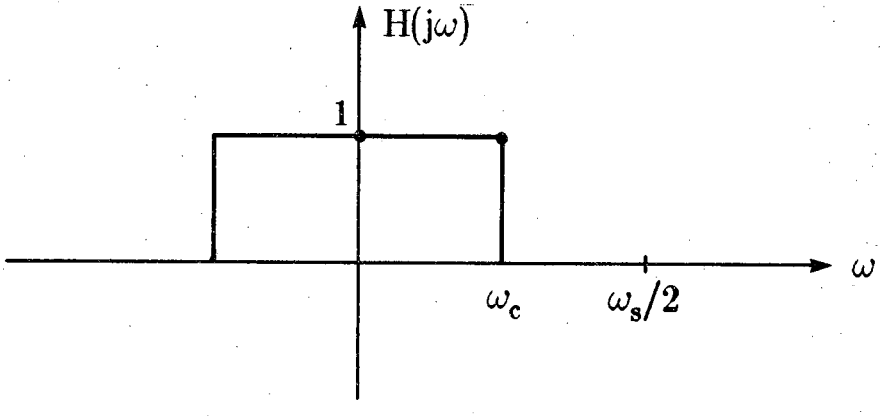


Figure 1

Name

od -- octal dump of a file, *or* of a group of blocks

SYNOPSIS

od [-i, -u, -f, -l] [<src> , <block number> [<block count>]]

DESCRIPTION

The octal values of the 16-bit words which comprise the specified source file *or* the specified blocks, are printed on the command terminal. The leftmost numbers on the terminal screen are always printed in octal and they represent the ordinal word numbers with respect to the beginning of the source, *or* the beginning of the block specified by <block number>. <block number> is an integer which specifies the physical block number with respect to the current device. The very first physical block is numbered 1 (not zero!). If <block count> is not specified, all the blocks starting from <block number> will be dumped, otherwise <block count> determines the number of blocks to be dumped. If no arguments are specified on the command line, "od" will begin dumping *one block ahead* of where it ended on the immediately previous "od" command; in this case the ordinal word numbers on the leftmost side of the screen will *not* reflect this jump.

The minimum amount of information which can be dumped is one block, however a <cr> will prematurely terminate the dump.

If a flag is specified, the numerical format of the dump is given in the following table:

flag	format
-i	two's complement integer
-u	unsigned integer
-f	floating point
-l	two's complement long integer (32-bits)
NULL	Octal

To see the contents of an ASCII file in ASCII format, use the "help" command.

NAME

out -- output from disk (through DMA)

SYNOPSIS

out [*] [<src> ...]

DESCRIPTION

This command enables the user to D/A convert output files from disk for ODSP lab listening. The output samples appear at the PDP 11/40 output port. The rate at which they appear is determined by the rate at which the port is strobed by an output of the shift register clock on the KLUGE. For Monaural files, output port channel "A" (*only*) is strobed. For Stereo files, channels "A" and "B" are strobed alternately, which implies that there is always a constant 1/2-sample delay between the two channels.

Source file output can be separated in time using silences. In place of one <src> in the synopsis, one would substitute the specification, s<NNN>, where NNN = tenths of seconds (this default time increment can be changed using the "cti" command). If s<NNN> is used, at least one <src> must have a nonzero sample-rate attribute (see "cfa") so that a zero-file of appropriate length can be created. The output stack (see "pos") holds the filenames and *(<window>ed)* file lengths of the files and silences which were last output. If "*" alone is specified in the command line, only the (windowed) files on the output stack will be output.

out

will have the same result.

out [<src> ...]

will replace the output stack with [<src> ...], and [<src> ...] will be output.

out * [<src> ...]

will apend the current output stack with [<src> ...] and the current output stack files, followed by [<src> ...], will be output.

Output can be prematurely terminated with a <cr>. By depressing the space bar on the command terminal during the output of a file, the MARK attribute (see "cm") will be set to the sample number (with respect to the beginning of the file) which was being output at the time the bar was depressed. Marking a file in this manner also terminates output prematurely.

NAME

pk - pack the files on the current device

SYNOPSIS

pk

DESCRIPTION

Commands such as "rm" do not actually destroy files; rather, the file *name* itself is removed from the index. *Since files are stored in order of creation,* it is easy to understand how unused disk space sometimes appears between files. This situation can be detected by reading the bottom line of the long listing (ls -l) which states how many blocks are being used, how many blocks are available for use, and how many blocks are immediately useable. The difference between these last two quantities equals the total number of "empty" blocks which are interspersed between non-empty blocks (files). The "pk" command, then, consolidates all the disk files so that they are physically adjacent and in their original order. The time required for execution is dependent upon the particular disk file situation. Any information in the "empty" blocks is lost.

The reason we put "empty" in quotes is because an "empty" block is usually not empty (all zeros) but still has its original contents. This can be used to advantage in retrieving lost files (see "rm").

NAME

plot -- plot files (2D cartesian graphics)

SYNOPSIS

plot [<# plot points>] [-z] [-i<N>] [-a<NNN>] [-s<NNN>] <src> [<src> ...]

DESCRIPTION

This command uses the graphics capability of the Tektronix 4025 command terminal to enable the user to see a graphical representation of his/her files. The files are assumed to be comprised of two's complement 16-bit integers, and are plotted having the sample number on the abscissa and the sample value on the ordinate. The abscissa reference values are with respect to the first sample plotted, *not* with respect to the file-beginning unless the first sample plotted was also the first sample of the file.

The user is prompted for the vertical scale maximum and minimum. If the user responds, he/she must be sure that the (integer) values specified are $\geq$ those of the file portion to be plotted, otherwise poor results will be obtained. The software uses those values as the maximum and minimum of the ordinate on all the following graphs. After each graph is plotted, the user is prompted for the "next window?". <# plot points> determines the total number of samples (the "window") represented by the abscissa for each graph. It defaults to 256 and has a maximum of 4096. This "next window" comprises the ***next*** <# plot points> samples in the <src> file. The "window" always begins with the beginning of <src> (which can be respecified using the <window> in the syntax; this command responds to a <window> specified in samples) and shifts to the right by <# plot points> for each new graph until the end of <src> is reached (or the end specified by <window>). This "window" is not to be confused with the <window> in the syntax which places artificial boundaries on <src>. If the user responded to the very first prompt for the max with only a <cr>, the software will determine the ordinate max and min for each "window" based on only the samples encompassed by that "window".

When the user is prompted for the "next window?", he/she has several choices for a response. If the user responds with a <cr> or a "y", the next "window" will be plotted. If the user enters an integer less than or equal to 4096, then the <# plot points> will be changed to this new value for the next "window" *only*, and one "window" of this new size will be plotted. If the user responds with "r", then the graph just plotted will be plotted again. If the user responds with an "n", the plot will be terminated. At any time during a plot, a <cr> will prematurely terminate that graph and summon the "next window?" prompt.

The "-z" flag plots a zero reference line. The "-i<N>" parameter sets the number of reference points (or "tick marks") on both the axes to N + 1. Default is N = 8. These points are also labelled numerically. The "-a<NNN>" parameter causes NNN samples to be averaged for each plot point. The "-s<NNN>" parameter causes only one out of each NNN samples to be plotted. (This parameter should not be confused with s<NNN>, the specification of silence, which is alien to this command.)

The multiple <src> file specification in the synopsis is handled differently here than in other commands. Here, the user is able to plot several <src> files simultaneously! The previous discussion still applies but the user must now be careful with the max and min specification. If the user chooses to enter a max and min, he/she must be sure that those values exceed or equal the maxes and mins of the portions of all named <src> files to be plotted, or poor results will be obtained. Further, if the user chooses not to specify the max and min, the software will take them, on each window, from the *first* named source file. If these values do not exceed or equal the maxes and mins from the corresponding "windows" on the other named source files, poor results will, again, be obtained.

BUGS If the user responds to the max prompt but not to the min prompt (i.e., with a <cr>), min is set to 0.

Notes: If trouble arises, the TEK terminal command, @wor 0<cr>, should get the screen back to normal. ("@" is the command character; see TEK Manual.) If splotchy random noise type visual harrassment patterns appear on the screen during a plot, press the "master reset" on the right side of the TEK terminal.

NAME

pm -- phase modulation

SYNOPSIS

pm <src> [<src> ...] > <dest>

DESCRIPTION

This command shares software with the "fm" command. So, all the prompts are the same and many of the comments in the "fm" description apply here as well.

This command implements the equation:

$$x_c(nT) = 32767 \cos\left[\omega_c nT + Ix(nT)\right] \tag{1}$$

where T is the sample period (taken from the first <src>). The command prompts first for f_c, and then for I. x(nT) corresponds to the concatenated <src> files.

The classical equation for PM is:

$$x_c(t) = A_c \cos\left[\omega_c t + \phi_\Delta x(t)\right] \tag{2}$$

where x(t) is the modulating signal, $|x(t)| \leq 1$,
A_c is the carrier amplitude,
ϕ_Δ is the phase deviation constant,
f_c is the carrier frequency.

Comparing (2) to (1) we find that x(nT) needs to be normalized which leads to the conclusion that

$$\boxed{I = \frac{\phi_\Delta}{|x(nT)|_{max}}}$$

expressed in radians. $x_c(nT)$ goes into <dest>.

NAME

pos -- print the output stack

SYNOPSIS

pos

DESCRIPTION

· · · Prints the filenames, the starting and ending block numbers of the (windowed) files, and the (windowed) file lengths (in samples) of the most recent "r" or "out" command. These file names and lengths form what is called the "output stack" (or "queue"). This output stack can be referred to as "*" in subsequent "r" and "out" commands. For example, if "pos" showed the following stack:

Filename	*Starting*	*Ending*	*# of Samples*
junk	8560	8562	512

then

out * s<NNN> junk2 <window>

would output the files "junk" followed by NNN tenths of seconds of silence followed by the windowed portion of "junk2". The output stack would also be updated as follows:

Filename	*Starting*	*Ending*	*# of Samples*
junk	8560	8562	512
--NNN tenths of seconds of silence--			
junk2	3630	3634	1024

Note that since "junk2" was windowed, the entirety of junk2 is not listed on the stack. Also note that either "junk" or "junk2" must have a nonzero rate attribute (see "cfa") in order that the silence be formulated properly as an appropriate length zero-file.

The starting and ending block numbers are with respect to the first block on the device, <dev> .

NAME

prt -- Teletype on/off for hard copy

SYNOPSIS

prt [on, off]

DESCRIPTION

This command sets a flag so that all ASCII output directed to the command terminal is simultaneously printed on the AS-33R Teletype printer-terminal. This Teletype is generally not used except for this purpose. The default parameter is "off". The Teletype must be switched on to the "line" position for this command to work.

NAME

put -- put a file in a directory on a network host

SYNOPSIS

put <src>[<src> ...] <UNIX>

DESCRIPTION

The specified source file(s) is sent to that network machine which has specified in the previous "link" command. The [concatenated] source file(s) is placed in the specified UNIX directory and is given the filename specified in <UNIX>. This command is found in the DMC program. It is wise to use entire pathnames in <UNIX>, and the user should check his/her directory to verify that the complete file was sent, as a precaution. There is a 1 Mbaud link to network machines, so transfers occur rather quickly. Each source file arrives at <UNIX> in the same form as it was when it left the ODSP lab. So, for example, Monaural audio file samples arrive as "short" integers.

If it is desired to "put" a file which is greater than 600 blocks in length, the user must always perform the following command before each "put":

cpu limit file <NNNNN>; wd

"wd" is a dummy command (see "cpu"). NNNNN is the approximate maximum desired number of blocks to send. This must be done regardless of the user's UNIX ".profile". If a file named <UNIX> previously existed in the user's UNIX directory, it is written over (hence destroyed) by the new <UNIX> of the same name. Before sending large files the user should be sure that there exists adequate space on the system disks. This can be done from ODSP via

cpu df

BUGS

An unlink is performed as a result of this command. It is wise to use "ul" before "link" is used again, to help get things reinitialized properly.

An incomplete "put" will result in the inability to unlink via the "ul" command, and hence, to link. The DMC program must be restarted in this case.

NAME

pwd -- print working device and directory

SYNOPSIS

pwd

DESCRIPTION

This command prints the current device (<dev>) and the current working directory (<dir>) (NULL for the main directory, an alphabetic character (user determined; e.g., see "cd") for a pseudo-directory).

Example: pwd (user)

a/j (SW2)

"j" is the current pseudo-directory, while the current <dev> is "a". This command is similar to the UNIX command of the same name. In order for "pwd" to return "a/j", the user must have typed at some earlier time the command

cd a/j

NAME

pwl -- piece-wise linear file creation

SYNOPSIS

pwl [-a] <dest>

DESCRIPTION

The rate attribute is mandatory for the <dest> file in this command (see the syntax).

The user is prompted for break points, (x_i y_i). Each coordinate in the pairs must be separated by a space or a carriage return. Two <cr>'s in succession terminate input. "x_i" specifies the time length of the ith piece-wise linear segment while y_i is the final value of that segment; for $i = 1 \rightarrow \infty$. For $i = 0$, $x_o = 0$ and y_o is the initial value of the file. The default (Δt) unit of time, which is set by "cti" is 0.1 seconds. In order that the break-points be specified as integers, the way in which x_i specifies time length is via the following formula:

$$\text{piece-part time-length} = x_i \cdot \Delta t \ .$$

The y_i may be any integers representable in 16-bit two's complement, but remember that the ODSP A/D's and D/A's are 12-bit two's complement.

After breakpoint entry, the software performs linear interpolation of the Cartesian coordinates implied by the breakpoints, and the piece-wise linear file is placed in <dest>. If the "-a" flag is specified, "piece-part" in the formula above becomes "absolute", and $x_{i+1} > x_i$. In this case, the breakpoints, (x_i y_i), become the Cartesian coordinates of the discontinuities of the piece-wise linear file which the user wishes to create.

The minimum distance between breakpoints in time is Δt. The minimum number of samples between breakpoints is $f_s \cdot \Delta t$; where f_s is the sample rate specified in <dest>. In general, if part of a file is $N\Delta t$ seconds long, then it will have (N an integer) $f_s N\Delta t$ samples. The maximum number of samples between breakpoints which are separated in time by Δt is $2^{15} = 32768$. So, for example, if f_s were 40 KHz, Δt could not be equal to 1 because an interpolation error would result.

The user is not constrained to enter a breakpoint at every Δt. <dest> files can only be created in multiples of one block length (256 samples). After the user has specified all breakpoints, the fraction of a block length (in general) remaining immediately after the last breakpoint, will be (padded) filled with zeroes. This presents an inconvenience to the user who wishes to create one cycle of some arbitrary periodic waveform having a period, T. If the sample rate must be fixed for some reason, then, in general, it will be difficult to create one cycle of an *arbitrary*

waveform spanning exactly an integer number of blocks. If we say that

$$T = \frac{B \times 256 \text{ samples}}{f_s}$$

where B is the number of blocks constituting exactly one cycle, then if we are at liberty to choose f_s, we can have our desired T. But, we are still faced with the problem of filling an integer (B) number of blocks having *no* trailing zeroes blithely appended by the software. We reason that this is possible only if the user can specify the very first and very last sample values of the B blocks which constitute the waveform cycle. The simplest way to accomplish this feat is to set

$$f_s \cdot \Delta t = 1$$

which means that the minimum number of samples between breakpoints is 1; i.e., the user has within his/her power the ability to specify every sample value in the file, if desired, but remember that the user is not constrained to enter a breakpoint at every Δt. The "cti" command comes in handy to set Δt ***before*** the "pwl" command is executed.

Having generated one cycle of an arbitrary waveform, the user can then use the "cat" command n-times to replicate the cycle by a factor of 2^n.

In the above example, if the user is constrained to a particular f_s, then the user must find the number of breakpoints, P, such that

$$P/T = \text{int}(P/T)$$

where T is the cycle duration (in seconds). This is because $\frac{1}{\Delta t}$ in "cti" is constrained to be an integer. One then sets Δt to T/P. The user is now constrained to particular time locations for the breakpoints, although there will be no padded zeroes if the user specifies the P^{th} breakpoint.

As a final example, suppose the user wishes to create a file exactly one block long consisting of all 1's. The user should then type:

```
pwl junk -r2560
0   1
1   1<cr>
<cr>
```

assuming that the default Δt is in effect. "junk" will be created consisting of one block's worth of 1's. Since the default Δt is 0.1, the minimum number of samples between breakpoints is $f_s \Delta t$, or $(2560) \cdot (0.1) = 256$ samples $= 1$ block. Since the user indicated the second breakpoint to be only one Δt away, then the software interpolates between the Cartesian coordinates (0,1) and (1,1). (It is only coincidental that the breakpoints are numerically equal to the cartesian coordinates, because the "-a" flag was not specified in the command above.)

NAME

r -- repeat output from disk (through DMA)

SYNOPSIS

r [*] [<src> ...]

DESCRIPTION

This command is the same as the "out" command except that the output is repeated indefinitely. As in the "out" command, output is terminated via a <cr> or by hitting the space bar which marks the file. Note that by the rules for using the output stack, which were discussed in the "out" **Description**, the command sequence

```
out junk
r
```

results in the repeated output of "junk".

NAME

raf -- recursive amplitude filter (recursive median filtering)

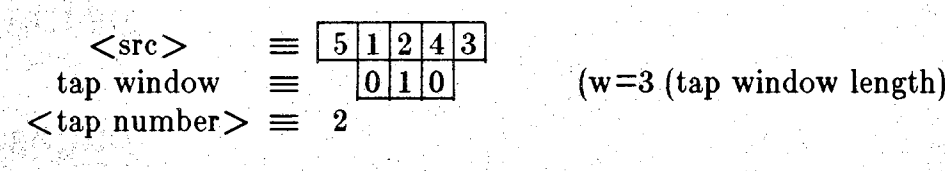
SYNOPSIS

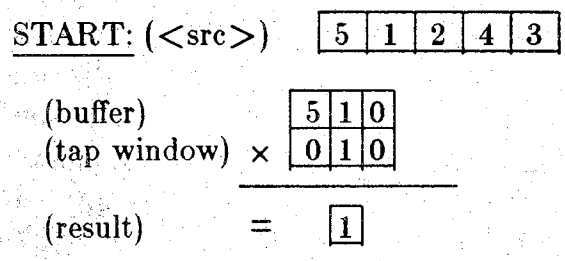
raf [<tap number>] <src> [<src> ...]

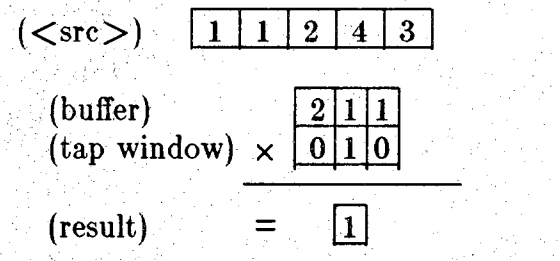
DESCRIPTION

This command is the same as the "naf" (nonrecursive amplitude filter) command with one exception. In fact, software is shared by these two commands (as well as with "nlf" and "rlf"). The exception is that the result of each iteration is placed directly back into the source file (essentially) immediately after each iteration. If one goes through the example given in "naf" and substitutes "<src>" for "dummy", the recursion should become apparent. It should also become clear that by the way we defined the effects of <tap number>, for <tap number> = 1 "naf" is equivalent, precisely, to "raf".

Let's go through a specific example:







(<src>)	1	1	2	4	3

(buffer)	4	2	1
(tap window) ×	0	1	0

(result) = [2]

(<src>)	1	1	2	4	3

(buffer)	4	3	2
(tap window) ×	0	1	0

(result) = [3]

(<src>)	1	1	2	3	3

(buffer)	3	3	0
(tap window) ×	0	1	0

(result) = [3]

(<src>)	1	1	2	3	3

This is the final result. Notice that the result of each iteration was placed back into the <src> as calculations proceeded, but the <src> samples themselves were never permuted. The permutation occurred only in the buffer. Also notice that there were only 5 iterations because this is the number of source file samples. <tap number> = 2 caused the tap window to be shifted left with respect to the source file by <tap number> − 1 samples on the first iteration. See "naf" for further insight into the algorithm.

Remember that the original file contents are lost.

NAME

rev -- concatenate files, then reverse

SYNOPSIS

rev <src> [<src> ...] > <dest>

DESCRIPTION

This command is the same as the "cat" command except that the resulting destination file is the (sample) reverse of the concatenated source files.

NAME

rlf -- recursive linear filter (IIR filtering)

SYNOPSIS

rlf [<tap>] <src> [<src> ...]

DESCRIPTION

This command implements the equation:

$$y(k) = \sum_{n=1}^{tap-1} a_{w-n}\, y(k+n-tap) + \sum_{n=tap}^{w} a_{w-n}\, x(k+n-tap) \qquad (1)$$

where, y ≡ output of IIR filter
tap ≡ tap window shift factor
w ≡ tap window length (# of coeffs.)
x ≡ input to IIR filter (<src>)
a ≡ tap weights (filter coefficients).

The results are placed back into the respective source files so that the original contents are lost. The user is prompted for the filter coefficients which should be entered in reverse order and separated by a blank (space) or a <cr>; i.e.,

$$a_{w-1} \quad a_{w-2} \quad \cdots \quad a_1 \quad a_0 <cr><cr>$$

The coefficients may be floating point numbers. Two carriage returns in succession terminate coefficient input. The number of coefficients, w, is determined by the number of coefficients entered, so all zero coefficients must be entered. <tap>, being optional, defaults to

$$\begin{cases} \text{int}(w/2) + 1 & ; \quad w \text{ odd} \\ w/2 & ; \quad w \text{ even} \end{cases}$$

and

$$1 \leq tap \leq w \ .$$

To understand the significance of <tap>, one must understand the algorithm. Suppose the user has entered four coefficients,

a_3	a_2	a_1	a_0

and has set <tap> = 1. Further assume that <src> consists of the following samples:

x_0	x_1	x_2	x_3	x_4	...

The algorithm begins with the multiplication of the samples and coefficients like so:

	x_0	x_1	x_2	x_3	...
×	a_3	a_2	a_1	a_0	

=	$a_3x_0 + a_2x_1 + a_1x_2 + a_0x_3$	x_1	x_2	...

This result shows the new contents of the source file after the first iteration. On the second iteration, the coefficients are shifted to the right like so:

	y(0)	x_1	x_2	x_3	x_4	...
×		a_3	a_2	a_1	a_0	

=	y(0)	$a_3x_1 + a_2x_2 + a_1x_3 + a_0x_4$	x_2	x_3	...

The number of iterations is always equal to the number of source file samples. In this case the final result in terms of "y" would be

y(0)	y(1)	y(2)	y(3)	...

Notice that this result is non-recursive (as indicated by (1)). This is always the case when <tap> = 1. Each iteration consists, then, of a shift and sum of products.

Now, let's go through the algorithm again for the case of <tap> = 2.

	x_0	x_1	x_2	x_3	...
× a_3	a_2	a_1	a_0		

=	$a_2x_0 + a_1x_1 + a_0x_2$	x_1	x_2	x_3	...

Notice that the effect of <tap> = 2 is to shift the coefficients to the left by <tap> − 1 places for the first iteration. This is always true in general. On the *next* iteration we find:

	y(0)	x_1	x_2	x_3	...
×	a_3	a_2	a_1	a_0	

=	y(0)	$a_3y(0) + a_2x_1 + a_1x_2 + a_0x_3$	x_2	x_3	...

We notice some recursion in this example, thus far. For this example we finally get:

$$y(kT) = \sum_{n=1}^{1} a_{4-n}\, y(k + n - 2) + \sum_{n=2}^{4} a_{4-n}\, x(k + n - 2)$$

(Any terms whose arguments are negative, go to zero; e.g., y(−1) = 0.)

We can start making some generalizations: The result of the n^{th} iteration always gets placed back (essentially) into the source file (immediately after that iteration) in the location, n; n = 0,1,2.... In other words, the 1^{st} source file sample always gets replaced by y(0), and y(1) for the 2^{nd} source file sample, etc., regardless of the values of <tap> or "w". Also, we see from observation of (1) that when <tap> = 1, a FIR (nonrecursive) filter results, and when <tap> = w, an all-pole IIR (recursive) filter results.

The question now arises as to what filter structure results in between these two limiting cases. The answer is best given by example: Suppose it is desired to implement the second order recursive filter section shown in the Figure. This filter is described in the frequency domain by

$$\frac{Y(z)}{X(z)} = \frac{a_0 + a_1z^{-1} + a_2z^{-2}}{1 - a_3z^{-1} - a_4z^{-2}} \qquad (2)$$

In the time domain,

$$y(nT) = a_4y(nT-2T) + a_3y(nT-T) + a_2x(nT-2T) + a_1x(nT-T) + a_0x(nT) \quad (3)$$

It is clear from (3) that on the RHS we need 2 "y" terms and 3 "x" terms. Using the limits of summation in (1) we set

$$\left.\begin{array}{l}\{(\text{tap}-1)-1\}+1=2\\ \{w-\text{tap}\}+1=3\end{array}\right\} \Rightarrow \begin{array}{l}<\text{tap}> = 3\\ w = 5\end{array}$$

The user should enter the 5 coefficients in reverse order ($a_4 \rightarrow a_0$) and indicate <tap> = 3 on the command line. Plugging into (1) we get:

$$y(kT) = \sum_{n=1}^{2} a_{5-n}\, y(k+n-3) + \sum_{n=3}^{5} a_{5-n}\, x(k+n-3)$$

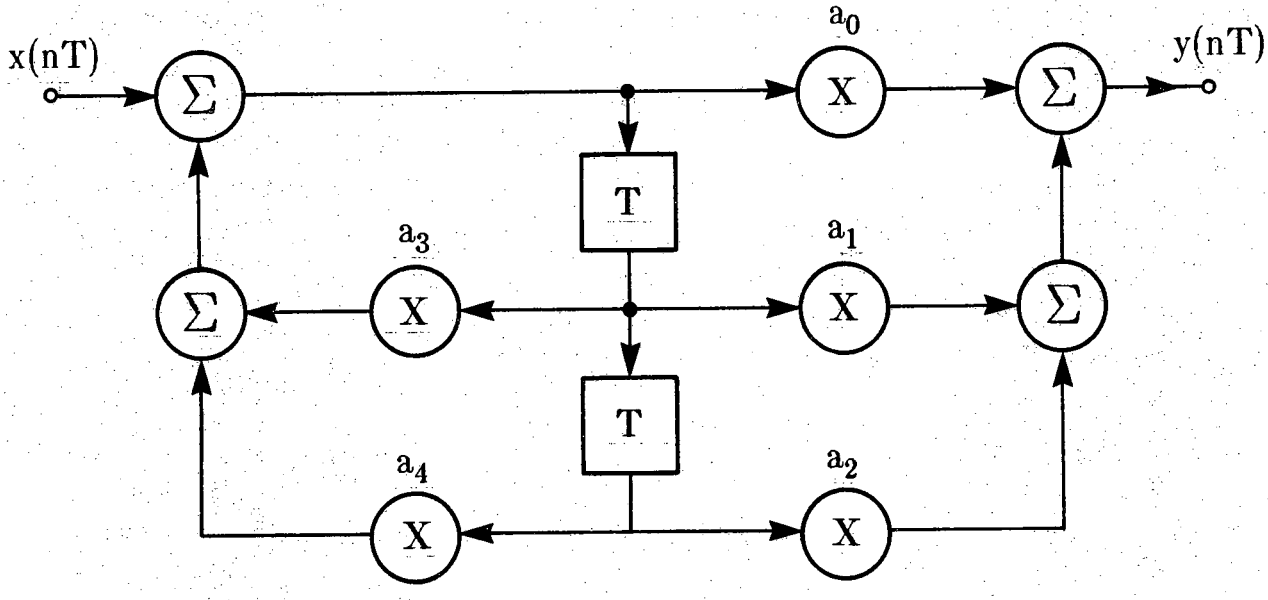
$$y(kT) = a_4\, y(k-2) + a_3\, y(k-1) + a_2x(k) + a_1x(k+1) + a_0x(k+2)$$

$$y(kT-2T) = a_4y(k-4) + a_3y(k-3) + a_2x(k-2) + a_1x(k-1) + a_0x(k)$$

$$\Longleftrightarrow z^{-2}\left(\frac{Y(z)}{X(z)}\right) = \frac{a_0 + a_1z^{-1} + a_2z^{-2}}{1 - a_3z^{-1} - a_4z^{-2}} \quad .$$

The magnitude of this filter response is the same as that of the Figure although the present equation requires future input samples. Of course, these samples can be provided since they are all stored on disk. This is a benefit of non-real-time processing.

One may ask why this particular implementation of "rlf" exists, and not one that is more straight-forward. The answer is that this command ("rlf"), "nlf", "naf," and "raf" all share the same software having only minor modifications to suit each command. These commands do *not* need the RATE attribute to be specified.



Figure

NAME

rm -- remove file(s), or an entire directory (from the index)

SYNOPSIS

rm [-p] [[<pathname> [<pathname> ...]], [<dir>/]]

DESCRIPTION

"rm" with no arguments has no effect.

If pathnames are specified, those files' *names* are removed from the device index. The physical contents of the files themselves remain intact unless the "-p" flag is specified. The "-p" flag invokes the "pk" command which causes the remaining files to be physically packed together, overwriting the files specified by the <pathname>'s.

This command is similar to the UNIX command of the same name. On the ECN at Purdue, when the "rm" command is issued, the removed file remains intact for a few hours on a back-up disk which is invisible to the user. Should the user have accidentally removed a file, it is therefore possible to retrieve it because the file was not immediately destroyed.

The same option is available in the ODSP Lab. The file called "index" which exists in the main directory of each device, holds the names and block locations of all current files. This information can be viewed using the "ls -l" command. Switch II (and DMC) will not allow this index file to be destroyed under any circumstances. Additionally, Switch II allows the user to give some other file the name "index". The device index file may disappear from its usual location, which is the very first file in the list, but it never actually gets destroyed. To bring it back into the listing, one needs to remove any files (which the user created) called "index", and then issue the "pk" command. To erase its contents, one must use the "init" command.

Example:

Suppose the following index was listed using the "ls -l" command:

Filename	*Type*	*Rate*	*Start*	*Length*	*Mark*
index	I	0	1	23	0
junk	D	2560	24	16	0
dexter	P	0	40	1	0

Then, the user removed junk ("rm junk") and listed again:

Filename	*Type*	*Rate*	*Start*	*Length*	*Mark*
index	I	0	1	23	0
dexter	P	0	40	1	0

The user now decides that he/she wants "junk" back. He/she issues:

cat dexter −b16 +b0 > junk

This clever user noticed that the MARK for "dexter" is precisely at the beginning of "dexter" and by making the "cat" command copy 16 blocks *before* the MARK up to zero blocks after the mark, he/she has recovered the original contents of "junk" which was 16 blocks in length. The listing now appears like so:

Filename	*Type*	*Rate*	*Start*	*Length*	*Mark*
index	I	0	1	23	0
dexter	P	0	40	1	0
junk	P	0	41	16	0

Notice that the rate and type attributes of "dexter" were carried over to "junk". These need to be changed back ("cfa"). Also notice that there is a 16 block gap between "index" and "dexter". We actually have two copies of "junk" on the disk now. Invoking "pk" will consolidate the files. Notice, again, that "junk" follows "dexter" whereas, before, "junk" preceded it.

The previous command could have equivalently been issued as

cat dexter −b16 +0 > junk
or, cat dexter −b16 +s0 > junk

The "+0" means 0 tenths of seconds after the mark, whereas "+s0" means 0 samples after the mark.

Using *this type* of <window> the user can devise more ways to recover lost files in some other situations. An idiosyncrasy of this window type is that SWITCH II will not traverse the right hand boundary of the named <src> file. Therefore, only "removed" files which are physically *previous* to an existent file can be easily retrieved. (Consider the use of negative numbers in <window>.)

If <dir>/ is specified *instead* of pathnames, all files' *names* in the indicated pseudo-directory are removed from the index.

NAME

rnd -- round file samples to prescribed number of bits

SYNOPSIS

rnd <number of bits> <src> [<src> ...]

DESCRIPTION

Each file sample will be masked (*AND*ed) with a 16-bit quantity whose <number of bits> MSB's are 1's and whose remaining bits are 0's. This command can be used to demonstrate the effect of n-bit quantization.

NAME

ros -- restore the output stack from disk

SYNOPSIS

ros [<src>]

DESCRIPTION

This command restores the previously saved output stack for use by the next "out" or "r" command. <src> must be a "q" file TYPE (a queue file) which was previously created using the "sos" command. If no <src> is specified, the current output stack (which can be seen via "pos") is cleared.

NAME

sc -- scale a file to a maximum magnitude

SYNOPSIS

sc [<maximum value>] <src> [<src> ...]

DESCRIPTION

The named source files are scaled such that the maximum magnitude in each respective file is changed to equal the maximum value specified in the command, while all other file values are scaled accordingly. The default maximum is 32767. Note that since the ODSP lab uses a twelve-bit two's complement A/D and D/A, one must assume a maximum audio file magnitude of 2047, the largest positive integer, *not* 2048, the largest negative integer magnitude.

Keep in mind that by scaling an audio file downwards, truncation noise will be introduced. The truncation occurs due to integer rounding. Therefore, to scale back upward would not reproduce exactly the original file.

This command displays the maximum and minimum found in the file, and the appropriate decimal scale factor. Although <maximum value> is usually positive, a negative <maximum value> may be used. This has the additional effect of inverting the source file.

The original source file is always destroyed by this command.

Use "af" to invert a file (i.e., multiply it by -1).

BUGS The maximum value may not be 0. Use the "af" command for scaling to 0. The absolute maximum magnitude is defined in SWITCH II to be 32767, not 32768.

NAME

sic -- sample the input port

SYNOPSIS

sic [-i, -u] [<number of samples>]

DESCRIPTION

This command is like the "in" command except that the sample values appear on the command terminal screen instead of getting placed in a destination file. This command is useful for checking the operation of the input channel. Since the samples come into the PDP via its input port, it must be clocked or the command will hang (a break from the command terminal will un-hang it).

If no parameter is specified, the samples will be printed in octal format. The -i flag (parameter) yields integer format while the -u flag yields unsigned integer format. The maximum "number of samples" is 256, while the default is 8 samples. The printout can be prematurely terminated by a <cr>.

NAME

sm -- Stereo-to-Monaural file conversion

SYNOPSIS

sm <weight1> <weight2> <src> > <dest>

DESCRIPTION

The source file is assumed to be a Stereo file regardless of its TYPE attribute. (See the "ms" command for the structure of a Stereo file.) Each sample of successive pairs of samples in the source file is first weighted by weight1 and weight2 respectively, and then the elements of the pair are summed. Each sum becomes a new sample which is placed consecutively in the destination file. Consequently, the destination file is half the size of the source file. The source file is not destroyed by the use of this command. Further, the destination file may not be of the same name as the source. The destination file should be scaled down, if necessary, to have a maximum value of 2047 for ODSP listening. The weights may be decimal fractions or integers.

This command may be used to decrease the sample rate of a file to 1/2 its original value by making either weight equal to zero. Beware that such a process may introduce aliasing.

BUGS If <src> is an odd number of blocks in length, the software uses data from the *next physical file* on disk to fill out the last block of <dest>. Solution: <src> must be even in length, or truncate <dest>.

NAME

sos -- save the output stack

SYNOPSIS

sos <dest>

DESCRIPTION

The current output stack, which can be displayed via the "pos" command, is stored for future use. The "ros" command restores that output stack which resides in a <dest> file created by "sos". Thus, the arguments of many complex "out" commands can be individually saved. <dest> is assigned the TYPE attribute, "Q" (for "queue").

NAME

stat - file statistics

SYNOPSIS

stat <src> [<src> ...]

DESCRIPTION

For each specified source file, the following information is displayed on the terminal screen:

- *Mean* of all sample values
- *Sum* of all sample values
- *Total* number of samples
- *Standard deviation*
- *Variance*

Remember that if no windowing is employed by the user when issuing this command (see syntax), the last block which is, in general, padded with zeroes to fill it, will be included in the calculations. (Also remember that the first block of an audio file may have been corrupted during input if the PDP output port was not strobed during data collection from the PDP input port (see "in").) This command operates on the source files at a rate of $\simeq$ 10 blocks/second.

This command can be useful for KLUGE maintenance. Specifically; for checking the non-recursive filter operation, or for a nifty offset null procedure for the gain/attenuator modules, see pgs. 91-92 and pgs. C2-C3 respectively, in the newer KLUGE manual.

NAME

tape -- audio tape back-up of files (multiple file output through DMA)

SYNOPSIS

tape [<src> ...]

DESCRIPTION

This command facilitates the audio tape recording of all files in the current (pseudo-) directory. Basically this command is equivalent to "out" where the "out" command-line argument would be a list of all the files in the current directory. The difference between "tape" and "out", in that case, is that ***all*** the named source files on the "tape" command line are output immediately after each respective file in the current directory is output; i.e., they are interleaved with the current directory files. <src>, in the "tape" argument, may be replaced with s<NNN> (see "cti") which denotes NNN tenths of seconds of silence. By taking the PDP output ports into the D/A converters and then into the tape recorder input, one can have an analog recording of all the digitized audio files in one's current directory. Although the is this primary reason for this command's existence, one is not constrained to use it for its intended purpose.

NAME

text -- create a help-command ASCII file

SYNOPSIS

text <dest>

DESCRIPTION

ASCII text is entered from the terminal by the user and is then stored in the named destination file. The ASCII contents of this file can be displayed on the terminal via the "help" command; e.g.,

help <filename>

The user terminates the creation of the <dest> file by placing the character "ctrl-d" at the beginning of a line followed by a carriage return.

In this manner the .start file can be created. .start is a special file whose ASCII contents are displayed on the terminal whenever the user changes to a new directory via the cd-command. The .start will only appear when it exists on the device in which the new directory resides.

NAME

ul -- disconnect network link (unlink)

SYNOPSIS

ul

DESCRIPTION

This command undoes the connections, rather, unlinks the PDP 11/40 from the network host which was established by the "link" command. Once a link has been established, it is ***not*** necessary to unlink in order to carry on with other types of SWITCH-II file manipulations or most other SWITCH-II commands. It is neither necessary to unlink after a "put" command since "put" automatically unlinks after its execution.

It ***is*** necessary to unlink before a link to a new host can occur, if an old link was previously established; i.e., only one link at a time is permitted. If the user is unsure as to whether a link exists, it will not upset anything to perform an "ul" to assure a broken connection. In fact, ***it is wise to use "ul" before every "link" to insure successful links*** because the "link" command has been known to malfunction for no apparent reason. The "ul" command helps to get things re-initialized.

"ul" normally takes approximately 6 seconds to execute successfully. It has also been known to fail, in which case a keyboard interrupt (break-break) will usually get the user back to DMC. However, it is usually then necessary to restart the DMC program via the "log" command which will always successfully break the link. If the interrupt fails to get back to DMC, a manual restart (773110) will then be mandatory.

BUGS This command causes the device and pseudo-directory to revert back to Aries' main directory "a" if a link actually existed prior to the command execution.

Chapter 3
Bell Digital Filter User's Manual

USER'S MANUAL
FOR THE
SERIAL, TIME-SHARED
DIGITAL FILTER
EXPERIMENTAL UNIT

This digital filter system* is a teaching and research tool for digital signal processing. It provides a large real-time programmable signal processing capability which may be accessed through each of several easy-to-use terminals. The system consists of a PDP 11/40 minicomputer (mini) and teletype (TTY), a general purpose digital signal processor (DF), and a number of terminals (1 to 8) which are used to enter data that programs the DF. These terminals also provide analog signal paths to and from the A/D and D/A modules in the DF. The mini's function is to interpret entries made at the terminals, and to make requested data transfers between the mini's memory and the DF memory.

The terminal accesses the DF directly as well as through the mini. The terminal has an analog input and output (±10 volt range). An analog signal source connected to the input terminal is sampled and converted to a 12 bit digital signal by a sample and hold-analog to digital converter (A/D) in the DF. An 8 KHz sampling rate is nominal,

*Constructed for Purdue by Bell Telephone Laboratories

and will be assumed throughout this manual although higher sampling rates are available. The digital signal is processed by the DF according to the data entered by the user through the terminal keyboard. The processed digital signal may then be converted back to analog by a digital to analog converter (D/A) in the DF, and this signal may be observed at the analog output of the terminal. There is no pre- or post-conversion filtering done to the analog signals, so that all the effects of sampling a continuous signal may be observed.

The DF basically performs the second order section (SOS) calculation shown in Figure 1. It uses 16 bit information signals between sections and in the sample delays, but carries 18 bit numbers at the multiplier outputs. The multiplier coefficients β_1, β_2, and α_1 may have values between -1.77... and +1.77... (octal number base) with 16, 13, and 9 bits of resolution respectively. For example α_1 has a 7 bit fractional part, and its values are -1.774, -1.770, -1.764,..., -.0, +.0,..., 1.770, and 1.774. α_2 is restricted to values of -1, 0, or +1. This restriction presents no problem for most practical second order filters, and as will be shown later, there are ample options available to correct many short comings which may arise. The DF performs the second order section calculation in slightly less than 1µsec, using serial "pipeline" arithmetic. Since a calculation is needed only every 125 µsec for an 8 KHz sampling rate, the processor may be time multiplexed into 128 independent "time slots," providing 128 separately programmable second order sections. One sample of the data for the first SOS is processed during the first time slot, using the coefficients (β_1, α_1, etc.)

for that time slot (which are stored in a memory within the DF). Then, one sample of data for the second SOS is processed during the second time slot, using a different set of coefficients, and so on until one sample is processed for each of the 128 time slots. This process is continually repeated for each sample in the 128 independent SOS. Thus the DF effectively provides 128 of the processes shown in Figure 1, each operating at an 8 KHz rate. These are divided among the active terminals, so that each user has access to a large number of SOS.

The DF memory provides 64 bits of program data for each time slot. Sixty of these bits are programmed by the user by entering six coefficients using the terminal. Three of the remaining bits control the assignment of a time slot to a particular terminal. The user is unaware of these bits since they are automatically set by the computer software. The remaining bit is currently unused.

Coefficients 1, 2, 3, and 4 (which account for 40 of these 64 bits) are respectively the β_1, β_2, α_1, and α_2 multiplier coefficients shown in Figure 1. Coefficients 5 and 6 control the options <u>within</u> a SOS, and the connectivity <u>between</u> SOS's.

<u>Input</u>, <u>Output</u>, <u>and Connectivity</u>

Coefficient No. 6 is made up of three octal digits: "1's", "10_8's", and "100_8's", each controlling a set of options. The complete coefficient is formed by adding together the code numbers for each option desired. A zero is always the normal option. The 1's digit controls the input to a SOS, and the 10_8's digit controls the output. Combined they may be used to connect the SOS's together in a wide variety of configurations. Figure 2 illustrates the options available.

Typically, SOS's are simply cascaded to accomplish higher order filtering. Thus, the output of the previous time slot is always available as the input to the next SOS, and this is specified by setting the 1's digit to zero.

A "4" specifies the A/D signal as the SOS input. This signal is available to all time slots assigned to the terminal.

A "5" specifies the "computer loaded register" as input. Terminal keyboard operation (which will be described later) loads a value in this register,* and this may be used as a constant (D.C.) input. However, every user has access to this register, and someone else may alter the value, so it must be used with care.

A "6" specifies a (minus) zero input.

A "7" specifies connection to a white noise source. The noise is psuedo random, with a repeat period of several seconds, and has a full scale amplitude (16 bits).

Three "temporary data registers" (TDR) are available to serve as signal processing paths. That is, the output of one SOS may be connected to the input of several other SOS. A "1", "2", or "3" connects the SOS input to TDR1, TDR2, or TDR3.

The 10_8's digit of coefficient 6 controls what (if anything) is done with the output of the SOS. A "10," "20," or "30" specifies that the output is to be loaded into TDR1, TDR2, TDR3.

TDR1 and TDR2 may also be used as accumulators, so that several separately processed signals may be added together. A "60"

---

*See the discussion of the PULSE key below.

("70") specifies that the output is to be added to the contents of TDR1 (TDR2). There is no overflow protection for these additions, and the usual "wrap around" to the opposite sign will occur if the result becomes too positive or negative.

As a result of the serial arithmetic used by the DF, there is a one time slot delay from the time a TDR is loaded until when it may be used as an input to another SOS. Thus, if the third time slot accumulates its output in TDR1, the first time slot which may use this accumulated value as an input is number 5.

A "40" (for coefficient 6) loads the nine most significant bits of the SOS output into the "coefficient register" (CR). The contents of this resister may be used as the α_1 coefficient of a later time slot. This option will be described later.

A "50" adds the SOS output to both TDR1 and TDR2.

In general, a TDR may not be used to connect the output of a time slot to the input of a lower numbered time slot, unless all other users are not using that TDR. There is only one set of these registers, and all users may access them during their time slots. Thus, there is no guarantee that data has not been loaded in a TDR during the time slots assigned to some other user.

The output of any one of the SOS's assigned to a user may be output to the D/A associated with his terminal. A special keyboard operation described later accomplishes this connection. The computer software guarantees that only one SOS output is connected to the D/A. The D/A converts the twelve least significant bits of the (16 bit) data, and saturates on any signal which is out of this

12-bit range. Scaling (described later) may be used to attenuate the 16 bit signal to within the conversion range of the D/A.

The 100_8's digit of coefficient 6 is used to specify one of three nonlinear functions which may be applied to the SOS output.

A "100" specifies full wave rectification (FWR). All negative numbers in the data stream are converted to positive.

A "200" specifies the "signum" function. With this function, the output may have only one of two different values; +k or -k, the sign is fixed by the data stream, and the value of k is fixed by any scaling that is applied (see below).

A "300" specifies half wave rectification (HWR). All negative values in the data stream are converted to -0.

A note regarding the polarity of data: The A/D and D/A are both buffered with <u>inverting</u> unity gain amplifiers, so that a FWR signal from the DF will have <u>negative</u> polarity when observed at the terminal. However, a signal converted by the A/D, processed by the DF with no polarity sensitive functions, and converted by the D/A will have undergone <u>two</u> inversions, and will thus have the proper polarity.

Coefficient 5 is made up of four octal digits, and controls a variety of special functions which greatly increase the versatility of a SOS.

The 1000_8's digit of coefficient 5 controls a power of 2 scaling at the output of a SOS. This is used to compensate for excessive gain within a SOS, and to keep the signals within the dynamic range of the DF or the D/A. The data is scaled <u>after</u> the

SOS calculation and nonlinear functions (FWR, signum, and HWR) but before the output options (Figure 2). The data is scaled by 2^{-j}, where j is the digit in the 1000_8's place of coefficient 5. Thus a 3000 specifies that the data stream magnitude be reduced by 2^{-3} or 1/8. Attenuation as great as 2^{-7} may be specified. When the signum function is specified by coefficient 6 ("200" digit), the scaling determines the amplitude of the two "saturated" outputs, $\pm k$; $k = 2^{15-2j}-1$, so that with "7000" scaling, the amplitude is one LSB (least significant bit), and with "0000" scaling, the saturated amplitude is 15 "ones" (full scale).

Configuration Options

A variety of other options are available so that signal processing using something other than the basic second-order section format (Figure 1) may be accomplished. These options alter the structure shown in Figure 1 and may also change the source or nature of the delayed samples (Z^{-1} and Z^{-2}). All of these options increase the DF hardware complexity only slightly, and are possible principally because of the serial format of the data representation. (A data path may be switched using a single multiplexer, or broken using a single AND gate. Often, a special feature cost no additional hardware because of unused gates or functions in IC packages already needed for the basic processor.)

Figure 3 is a more accurate illustration of how the processor is actually implemented. With the logical switches in the positions shown, the SOS process is the same as that shown in Figure 1. The

utility of this organization will be apparent as the options are described.

The 1's digit of coefficient 5 controls the dual selector switch P_{1A} and P_{1B} as well as switch S_2. S_2 is in the closed position for all odd codes, and open for all even codes.

A "1" code selects the input signal as the sample to be delayed rather than the sum of β_1 and β_1 and the input signal. That is, the SOS feedback path is broken. This has the effect of placing β_1 and α_1 in parallel and β_2 and α_2 in parallel. This makes the SOS an all transversal filter, and eliminates the resolution limitations of α_2. Some transversal filters require coefficients with a range greater than ± 2, which is the normal range of β_1, β_2, and α_1. However, the delayed samples may be multiplied by a power of two using the 100_8's digit of coefficient 5, which controls selector P_2. Thus, with β_1 and α_1 in parallel, coefficients in the range of ± 32 are possible.

Switch S_{1A} and S_{1B} are controlled by the 10_8's digit of coefficient 5. A "40" code breaks the signal paths into adders A_1 and A_2. The SOS signal may still find its way to the output, however, through the delays and the α_1 and α_2 multipliers.

All odd codes of the 1's digit of coefficient 5 are obviously meant to use the β multipliers in some way, because the effect of S_{1B} is defeated by S_2 for these codes.

A "2" (for coefficient 5) selects the SOS output as the sample to be delayed (through P_{1B}). This makes the filter all recursive.

Thus α_1 becomes equivalent (in its effect) to β_1. The special options available with α_1 (which will be described later) make this a powerful feature.

A "3" selects the twice-delayed sample from the next time slot as the sample to be delayed during the current time slot. This makes it possible to implement higher than second order filters directly. Figure 4a illustrates a fourth order (direct realization) recursive filter implemented using this feature. Figure 4b is an equivalent form of this arrangement. Note that this is different from two cascaded second order sections.

Figure 5 shows how a 6th order all transversal (nonrecursive) filter might be directly implemented, using the accumulator feature of TDR1. There are six samples of delay before α_2 and β_2 of time slot 1 can operate on the input data. As we now know, the accumulation in TDR1 is not ready for output until time slot 5. Note that the input goes to time slot 3. This is the only case where data may be passed "backward" in time, i.e. to an earlier time slot without using a TDR.

A "4" or "5" code specifies that a full scale signum function is to be applied to the delayed data. A "5" specifies the nonrecursive mode (due to switch P_{1a}). This allows a SOS to perform functions which require hysteresis, such as a Schmitt trigger.

Codes "6" and "7" bypass the overflow protection on the delayed sample, so that wrap around arithmetic is in effect. A "7" disconnects the β_2 signal path, and restricts the SOS to a first order recursive section. Thus, a code "7" may be used for making a "resetting" integrator.

With $\beta_1 = 1$, the output is equal to the sum of all the inputs until the result reaches full scale, then the result goes to negative full scale. This is useful for generating long period ramp functions, which may later be changed to square waves or impulse strings for testing the step or impulse responses of filters made with other SOS.

α_1 Options

The α_1 and α_2 coefficient circuitry is really more complex than is shown in Figure 3. Figure 6 illustrates their options. These options are controlled by the 10's digit of coefficient 5.

Either coefficient 3 or the value stored in the coefficient register, CR (described earlier in the section on output options) may be used as the α_1 coefficient. Switch S1 (Figure 6) does this selection, a code "20" specifying the CR. Thus, two data streams may be multiplied together to perform a variety of modulation functions. Even when the CR is selected, the sign of coefficient No. 3 (α_1) is in effect, so that a positive or negative multiply may be done. Since the α_1 multiplier may be put in either the transversal or recursive part of the SOS, amplitude or frequency modulation may be accomplished.

The coefficient selected by S1 may also be used as an additive constant. This option is chosen by S2, a code "10" selecting a constant ("1"), as the multiplier term rather than the delayed data. This is a useful feature for threshold detection. Note also that two data paths may be added or subtracted (according to the sign of coefficient 3) by selecting CR with S1.

Coefficient 3 and CR have only 9 bit resolution, since α_1 controls only a 9 bit multiplier. CR is loaded (see Figure 2) with the sign and eight most significant bits of the output data. Some care must be exercised in scaling to maintain a reasonable dynamic range.

Figure 7 illustrates how these options may be used to program a sweep frequency oscillator. Time slot 1 is used to generate a small constant. Using a zero input (coefficient 6 = 6), a small value for α_1 (coefficient 3 = .004 for example), and a scaling of 2^{-7} (coefficient 5 = 7000), a constant appears at the output. Time slot 2 is used as a wrap around integrater. With coefficient 5 = 47, coefficient 3 and scaling may be used to add just the peak to peak amplitude of the ramp (saw tooth) output signal. Time slot 3 uses coefficient 3 as an additive constant to produce a level shift, and the output is stored in CR.

Since serial arithmetic is used and sizable delays occur in the multipliers, the data loaded in CR may not be used until four time slots later. Time slot 7 is used as a programmable oscillator. β_2 is set slightly more negative than -1 (-1.001) so that the oscillations will grow in amplitude until limited by the overflow protection. Coefficient 5 = 23 so that the α_1 multiplier is CR and is in the recursive path. The output of time slot 7 will be a full scale sine wave oscillation, which may be scaled by time slot 8 before it is finally used. The center frequency is chosen by α_1

(additive constant) in time slot 3. The sweep range is fixed by α_1 and the scale used in time slot 2. The sweep rate is determined by the constant generated by time slot 1.

One other option is available using the α_1 and α_2 multipliers. A "-0" code for α_2 is redundant, and is used to select the signum functions shown in Figure 6 through S3. The saturation constant is 1/2 full scale (a one followed by 14 zeros). If the α_1 multiplier is positive, the output from the adder will be nonzero only when the delayed data changes sign, which will cause a full scale output from the adder for one sample. Thus, it operates as a zero crossing detector. If β_1 and β_2 are set to produce an oscillation and coefficient 5 = 40, then the output will be a sequence of full amplitude pulses of alternating sign. If the FWR option is used, they will all have the same sign, with twice the frequency of the oscillation produced by β_1 and β_2. A time slot such as this (with scaling) may be used to drive the wrap around integrater in Figure 7 to make a step-sweep frequency oscillator, where the output has constant frequency for a period of time, and then steps to a new frequency.

This completes the description of options which are controlled by the six coefficients that program each time slot. A great deal of flexability is available to produce a variety of filter configurations. The examples should illustrate to the user that some of the time slots assigned to him may be used to generate test signals for other time slots which have been programmed as filters. Then still

other time slots may be used to measure the response of the filter under test. A large class of test instruments may be constructed in this way.

Appendix 1 contains a list of the coefficients, and a brief description of all the options which are available. The remaining part of this manual will describe the use of the terminal keyboard.

Digital Filter Programming Using a Terminal

The user terminal has connectors on the rear for an analog input (±10 volt differential input, ~ 20 kΩ impedance) and an analog output (± 10 volt, ~ 100 Ω impedance). Both of these signals have 12 bit resolution (smallest step = 5mv).

Figure 8 illustrates the front of the terminal. There are two displays: a three digit display on the left called the coefficient number display (CND), and a five digit display on the right called the data display (DD). A calculator-like keyboard is used to enter the data displayed by CND and DD, transfer the data to the DF, and to display data from the DF. It is also used to access a file system for the storage of program data.

The ON/OFF switch on the keyboard is used to activate the terminal, and execute initialization functions. The available time slots (128 for 8 KHz sampling) are nominally divided equally among the 8 terminals connected to the system. However, if a terminal is off, its time slots are assigned to the terminal with the next lower physical number (modulo the number of terminals in the system).

Thus, if only one terminal is on, all 128 time slots are assigned to it. (All terminals function identically except for this assignment of unused time slots). When the terminal is turned on, all the coefficients of the time slots assigned to that terminal (including those contributed by other off terminals) are set to their "nominal" values. That is, coefficients 1 through 5 are set to zero, and coefficient 6 is set equal to 6 (zero input) for all time slots except the first. Coefficient 6 for the first time slot is set to 4, the A/D input, and the output of this time slot is connected to the D/A. Thus, the DF is simply a "wire", digitizing the input analog signal, performing no processing, and converting the signal back to analog form. The effects of simple digitization may thus be observed. There is no filtering done before or after the digital/analog conversions. Note: Turning the terminal off, then on again is the fastest way to reset all coefficients, and erase all programming which may have been done.

The CND addresses the time slot number and the coefficient number for data transfer. The least significant digit is the coefficient number, and may only have values 1,2,...,6. The other two digits represent the time slot number, the first is numbered "1". Since more than 99 time slots may be assigned to a terminal, the symbols "0", "□", and "-" for the most significant digit are used to indicate time slot numbers 100 through 109, 110 through 119, and 121 through 128, respectively.

The keyboard keys in the right column and the point (".") key have dual functions. The alternate functions (yellow lower labels) are selected using the PASSWORD key, and will be discussed later. The primary functions (the upper white labels) are used for most DF programming.

Pressing the EXAM key causes the coefficient specified by the CND to be displayed by the DD. If the coefficient number is 1, 2, or 3, the display will have a "point" after the first digit, and a four digit fractional part. The number base is nominally octal (set when terminal is turned on) but may be changed to decimal using the PASSWORD key (see later). Coefficient 4 may be "1.", ".", "-1.", or "-.". Coefficients 5 and 6 are 4 and 3 digit octal numbers. Each time the EXAM key is pressed consecutively, the CND is automatically incremented to the next coefficient, and its value displayed. After coefficient 6, coefficient 1 of the next time slot is displayed.

The digit keys place data in the DD. Digits are accepted until the display is filled, then additional digits are ignored. The "." key will only place the point after the first digit, and is ignored any time it is pressed when not appropriate. The digit and "." keys cause the DD to be automatically cleared if pressed immediately after the EXAM key.

The "Ch.S.", key changes the sign of DD. There is no restriction on when or how often it may be pressed.

The "CLEAR DISP" key clears DD, producing a null display.

A particular time slot and coefficient is addressed by entering their numbers in the DD, and pressing the "COEF #" key. The value in DD is moved to the CND, and DD is cleared. If the specified coefficient number is zero, it will be changed to 1. If it is greater than 6, it will be changed to 6. If the time slot number specified is zero, it will be changed to 1 (a null value in DD will cause "11" to be entered into CND). If the time slot number is greater than the number of time slots assigned to that terminal, the CND and DD will blink at about a 1 second rate. This is a general error signal which occurs whenever an illegal or impossible request is made. All keys are ignored until the CLEAR DISP key is pressed. This clears DD, the error signal, and the correct value may be entered.

The D/A connection to a time slot is also controlled by the COEF # key. If a negative time slot number is entered in CND, the D/A is thereby connected to the output of the specified time slot. The D/A connection to any other time slot is broken. (Notice, the coefficient number part of the specification is not significant for the D/A connection, only the part specifying the time slot number is used.) This D/A connection remains the same until another negative coefficient number is entered.

The DF is programmed using the DEP key. The coefficient number specified by the CND is loaded with the value in DD whenever the DEP key is pressed. DD will automatically clear after the transfer

is completed. Coefficients 1, 2, and 3 must have a point, even if their values are zero. Normally, their fractional parts are specified in octal, but may be specified in decimal using an option described later. Coefficients 5 and 6 must always be octal, and not contain a point. DEPositing an illegal number will cause an error signal (blinking CND and DD), and the CLEAR DISP key must be pressed before the correct value may be entered.

These key functions are arranged so that normal data programming is convenient and self-checking if the coefficients are entered in sequence (coefficient 1 through 6 for each time slot, starting with the lowest time slot number). After the data is entered (DEPosit), pressing the EXAM key will display the just-entered coefficient as it exists in the DF memory. Thus, any rounding, truncation, or missing bit fields can be observed. Pressing the EXAM key again will increment the coefficient number in CND, and display its current value. If the value happens to be correct, the EXAM key may be pressed again to display yet the next coefficient. If a change is necessary, the DD will be automatically cleared if any digit key or the "." key is pressed. If just the sign needs to be changed, the Ch.S. key may be pressed, and then the DEP key. The magnitude need not be reentered in DD.

These key functions are all that is necessary to program completely the DF, and observe the processed signals. However, other functions have been provided to increase the flexability and power of the system.

It is often useful to pulse a filter network, in order to alter a limit cycle, start a process, or test the response of the filter. The PULSE key may be used for this purpose. The time slot which receives the pulse is that specified by the contents of CND. The amplitude and sign of the pulse is specified by the contents of DD, which must contain a point. The "1." part places a one in the most significant bit position. When the PULSE key is pressed, the value specified in DD is loaded in the "computer loaded register" (see Figure 2). Coefficient 6 of the time slot specified by CND is read, and a "5" (connect input to computer loaded register) is temporarily placed in the 1's digit. This modified coefficient 6 is loaded in the DF, and used for one sample. Then the original coefficient 6 is loaded, and the process is complete. It is repeated each time the PULSE key is pressed. This sequence replaces one of the data samples in the normal signal by a value specified by DD. If the normal signal input is a zero, the effect is an impulse. A variety of effects may be observed using this function.

Password and File System

Approximately 4K words of the mini core memory are available to store DF program data; and this memory may be accessed by the user to store and/or retrieve DF coefficients. A particular file is accessed using a "password" number assigned to a set of mini storage locations. This number may have any value between -19999 and 19999. The specific number is assigned by the instructor through the teletype (TTY) connected to the mini. A password has access to 256 lines

of storage, one line being the data necessary to program one time slot (64 bits of data, so there is enough storage for ~1000 time slots). Transfers to and from a file are made on a line basis, so that a complete time slot is the smallest storable unit of data.

A psssword is specified by entering the password number in DD, and then pressing the PASSWORD key. If the password is valid, (one that has been previously entered by the instructor) DD will clear. Six of the keys now have alternate functions (the lower, yellow labels). The terminal will be in the password mode (with the yellow alternate functions in effect) until a password function is completed, or until the CLEAR DISP key is pressed twice consecutively. If the same password is used for another password function, it need not be reentered. Pressing the PASSWORD key when there is a null DD display has the effect of specifying the last used password number (Note: There is no "last used" password defined after the terminal has been turned off and then on again.)

The alternate function of the "." key is to change the number base of the fractional part of the coefficients. If DD is null, and the key pressed while in the password mode, the base is octal. If a 9 is in DD, the base will be decimal. After this key is pressed, all keys again have their normal functions (the terminal is out of the password mode). The number base thus specified will be in effect until respecified, or until set to octal by turning the terminal off and then on.

Four things must be specified to make a transfer of data between the DF and a file: 1) the first DF time slot where the

transfer is to occur, 2) the password of the file, 3) the first line of that file where the transfer is to occur, and 4) the number of lines (time slots) of data that are to be transferred. The DF time slot is specified by the contents of CND (the coefficient number part is ignored), and must be set to the proper value before entering the password mode. The password is specified as described earlier, and the alternate key functions then come into effect. The first line of the file is specified by entering the line number in DD and pressing the FIRST LINE (Ch.S.) key. The file lines are numbered 1 through 256. An error signal will occur if the "first line" number is out of this range, and CLEAR DISP key must then be pressed. (This does not take the terminal out of password mode. The CLEAR key must be pressed a second time to accomplish this.) The correct number may then be reentered. If the first line is not specified, "1" is assumed. Pressing the FIRST LINE key causes DD to clear. The number of lines to be transferred is then entered in DD. These operations define the range of the data transfer.

If the WRITE FILE key is pressed, the data will be written into the file from the DF memory. If the READ FILE key is pressed, the file will be read into the DF memory. If an attempt is made to read a file which has not been written, or if a file line number greater than 256 is requested during transfer, an error signal will result. After transfer, the terminal is out of the password mode.

In summary, then

TO READ OR WRITE PASSWORD FILES:

1) Make sure the time slot number of the first SOS to be affected by data transfer is now residing in CND. If this is not the case, use the COEF # key to properly load CND.
2) Key in password number of the file involved.
3) Depress the PASSWORD key.
4) Key in the line number of the first line of this memory file to be affected by the data transfer.
5) Depress FIRST LINE key.
6) Key in the number of SOS's (or, equivalently, line numbers) to be transferred.
7) a) To read a file from the computer memory to your DF sections, depress READ FILE.
   b) To write some (or all) of your DF sections to computer memory, depress WRITE FILE.

The file space of a negative password number may not be written into. The instructor may change the sign of a password number from the TTY, and thus make a file read only. This allows some generally useful processing programs to be available to all users, without danger of having the file damaged.

The data stored in a file contains the same D/A connection information that existed when the file was written. When the file is read back into the DF, the D/A connection specified in the file

will be made, and any other connection broken. After the file has been read into the DF, the CND will indicate the time slot which is connected to the D/A. If no connection is specified in the data, the existing D/A connection is maintained, and the contents of CND will be unchanged.

A special feature is provided to compare different DF's in quick succession. When the TAB FILE key is pressed (with a null DD display), the first eight lines of the password file previously referred to is read into the DF, starting at the time slot specified by CND. The terminal does not leave the password mode, does not indicate any D/A connection by modifying CND, and displays "0" in DD. The next time the TAB FILE key is pressed (no other keys pressed in between), eight more lines of the password file are read into the same time slots, but this time, starting with line 10, and a "1" is then displayed in DD. Each successive time the TAB FILE key is pressed, eight more lines of the file are read into the same DF time slots, starting with lines 20,30,...,70,1,10,... and so on with the numbers 2,3,...,7,0,1,... being displayed in DD. Thus, eight different DF's each using up to eight time slots may be entered successively using a single key stroke. If a digit key (0-7) is pressed, that number will be entered into DD. Then, the next time the TAB FILE key is pressed, that group of eight lines will be read. The second time the key is pressed, the digit will be incremented (modulo 8), and the next group of eight lines will be read in.

Obviously, the password file needs to be prepared carefully for this feature to be effective, but a variety of interesting effects may be produced and/or observed. For example, if the first

time slot of each group is programmed to be an oscillator at a frequency of the chromatic musical scale, simple tunes may be played using the digit keys and the TAB FILE key. Using one of the nonlinear functions available at the output of the oscillator section, and the seven remaining sections for filtering, the timbre of many conventional instruments may be imitated.

The instructor may change the tab file structure if necessary. The number of time slots programmed, the start line of each group, and the total number of groups are all programmable by a TTY entry.

The contents of a file may be listed on the TTY using the TTY LIST key. The first line and number of lines are specified just as for a data transfer (the contents of CND are irrelevant) and then the "TTY LIST" key is pressed. The TTY will now begin listing the file in an easy to read format. The data typed on any line is that which is in the file when the line printing begins, so transfers to the file while a list is in progress may alter the listing. The number base of the fractional parts appearing in a listing is the same as specified at the terminal. A "*" is typed at the end of the line which is connected to the D/A. If a terminal request for a list is made while the TTY is already in use, an error signal will result. A listing may be stopped by typing a "BELL" (control-G) on the TTY.

TTY Operations

Three file operations are available to the user from the TTY: 1) a file may be listed (just as from a terminal), 2) a file

may be dumped in compressed form on paper tape, and 3) a file may be entered from paper tape. The format of the instructions typed at the TTY is:

LF,
DF, } password #, first line, number of lines (carriage return)
EF,

LF is for list file, DF is for dump file, and EF is for enter file.

The password #, first line, and number of lines are the same as would be used from a terminal.

The list normally has octal fractional parts. Decimal fractional parts may be specified by typing a ",9" after the "number of lines". If an error is made while typing, a "RUB OUT" may be typed to erase the entire line. The TTY responds with "??" if an instruction cannot be interpreted. Any TTY process may be stopped by typing a BELL (control-G).

When doing a dump or enter file, the appropriate settings must be made on the paper tape punch or paper tape reader. A leader is automatically punched at the beginning and end of any punched tapes. A "RUB OUT" (all holes) at the beginning of the tape signals the start of data. Whenever eight null characters (no holes) are encountered, the end of tape is assumed, and the reading process will stop, even if the specified number of lines have not been read. Only eight punch frames per line (time slot) are required to code the data, so these dump tapes are typically quite short.

Conclusions

The DF processor and software driving the terminals provides the user access to a powerful, real time signal processing system. The imaginative user can implement a wide variety of filters, and special purpose signal processors. Things which may be programmed include:

1) Sample and hold amplifier
2) Peak signal detector
3) Spectrum analyzer
4) Linear to log voltage converter
5) Pulse counter
6) Phase sensitive detector
7) Hartly modulator
8) Music "box"

This list is by no means complete, and is intended to stimulate the user.

APPENDIX I

Coefficients and Functions

Coefficient 1: β_1

Range	-1.7777 to 1.7777	octal
	-1.9999 to 1.9999	decimal

Coefficient 2: β_2

Range	-1.7776 to 1.7776	octal
	-1.9995 to 1.9995	decimal

Coefficient 3: α_1

Range	-1.774 to 1.774	octal
	-1.9922 to 1.9922	decimal

Coefficient 4: α_2

Values	1, 0, -1	normal multiplier
	-0	1/2 full scale signum at output of α_1 and α_2

Coefficient 5: CONTROL

Made up of 4 octal digits which control special options.

1's digit; delayed sample options.

0 - normal second order section

1 - all transversal nonrecursive filter (FIR)

2 - all recursive filter (IIR)

$\left\{\begin{array}{l}\beta_1 \text{ and } \alpha_1 \text{ in parallel} \\ \beta_2 \text{ and } \alpha_2 \text{ in parallel}\end{array}\right.$

3 - Z^{-2} output from next time slot are used for the delayed samples

4 - Full scale signum function is appled to the delayed samples

5 - Same as 4, but all transversal (FIR)

6 - Defeat overflow protection (allow wrap-around) and signum function on delayed samples

7 - Same as 6, but β_2 term disconnected

Note: Code 1, 3, 5, and 7 always add β_1 and β_2 terms to the input, defeating the effect of a code 40 on these terms.

10_8's digit: multiplier options.

00 - α_1 is normal multiplier

10 - α_1 is additive constant

20 - CR used as multiplier (α_1 sign still effective)

30 - CR used as additive constant (α_1 sign still effective)

40 - Break direct data path from input to output

(Disconnect β_1 and β_2 terms; defeated by odd 1s digit)

Note: A 10, 20, or 30 may be added to 40 for combined options.

100_8's digit: multiply delayed values (effectively modifies β_1, β_2, α_1 and α_2 range).

0 - normal

100 - multiply by 2 ($-4 < \beta_1, \beta_2, \alpha_1 < 4$), ($-2 \leq \alpha_2 \leq 2$)

200 - multiply by 4 ($-8 < \beta_1, \beta_2, \alpha_1 < 8$), ($-4 \leq \alpha_2 \leq 4$)

300 - multiply by 8 ($-16 < \beta_1, \beta_2, \alpha_1 < 16$), ($-8 \leq \alpha_2 \leq 8$)

Note: A 400 bit does not appear in this digit.

1000_8's digit: Scale output from time slot by a power of two. Set amplitude of saturated or signum output.

	Scale Value	Limits Output (Information Line) Peaks to	Output Signum Function Saturates at:
0	(x1)	$\pm(2^{15}-1)$ (normal for 16 bit information line)	$\pm(2^{15}-1)$
1000	$(x2^{-1})$	$\pm(2^{14}-1)$	$\pm(2^{13}-1)$
2000	$(x2^{-2})$	$\pm(2^{13}-1)$	$\pm(2^{11}-1)$
3000	$(x2^{-3})$	$\pm(2^{12}-1)$	$\pm(2^{9}-1)$
4000	$(x2^{-4})$	$\pm(2^{11}-1)$	$\pm(2^{7}-1)$
5000	$(x2^{-5})$	$\pm(2^{10}-1)$	$\pm(2^{5}-1)$
6000	$(x2^{-6})$	$\pm(2^{9}-1)$	$\pm(2^{3}-1)$
7000	$(x2^{-7})$	$\pm(2^{8}-1)$	±1 (LSB)

Coefficient 6: I/O

Made up of three octal digits which control input, output, and nonlinear functions.

1's digit: input selection.

0 - input taken from previous time slot

1 - input taken from TDR1 } Temporary data Registers loaded by earlier time slot

2 - input taken from TDR2

3 - input taken from TDR 3

4 - input taken from A/D (12 least significant bits of 16 bit information line are loaded.)

5 - input taken from COMPUTER LOADED REGISTER

6 - zero input

7 - 16 bit white noise used as input.

10_8's digit: output destination. Recall that output is <u>always</u> available to the next SOS.

0 - none of the registers modified

10 - load TDR1 with output

20 - load TDR2 with output

30 - load TDR3 with output

40 - load CR <u>and</u> COMPUTER LOADED REGISTER with output (sign and 8 most significant bits)

50 - add output to TDR1 <u>and</u> TDR2 (no overflow protection)

60 - add output to TDR1 (no overflow protection)

70 - add output to TDR2 (no overflow protection)

100_8's digit: nonlinear function.

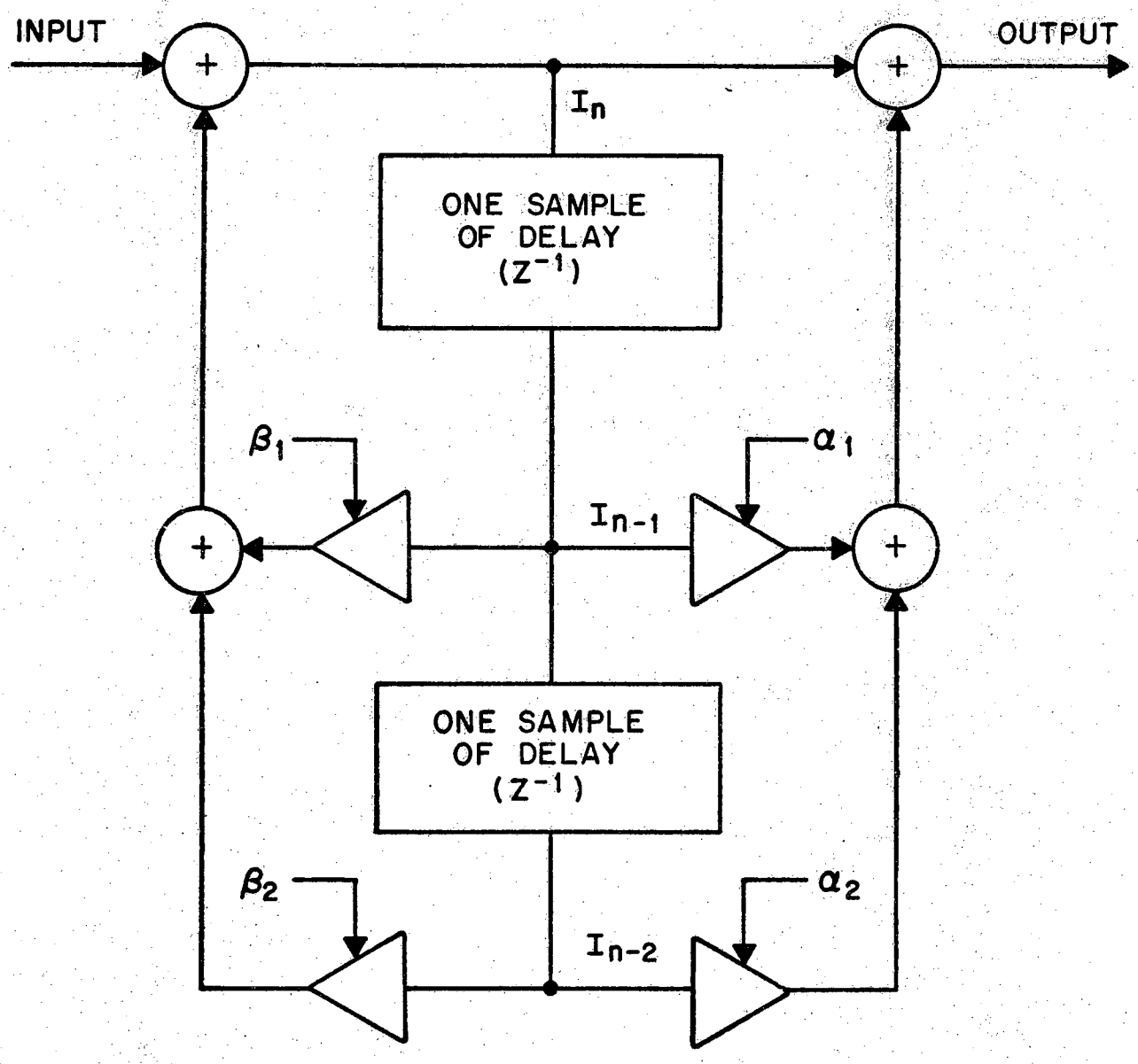
0 - normal output

100 - full wave rectify output

200 - signum function applied to output (saturation values determined by 1000's digit of coefficient 5)

300 - half wave rectify output

Note: The 400 bit is not used.



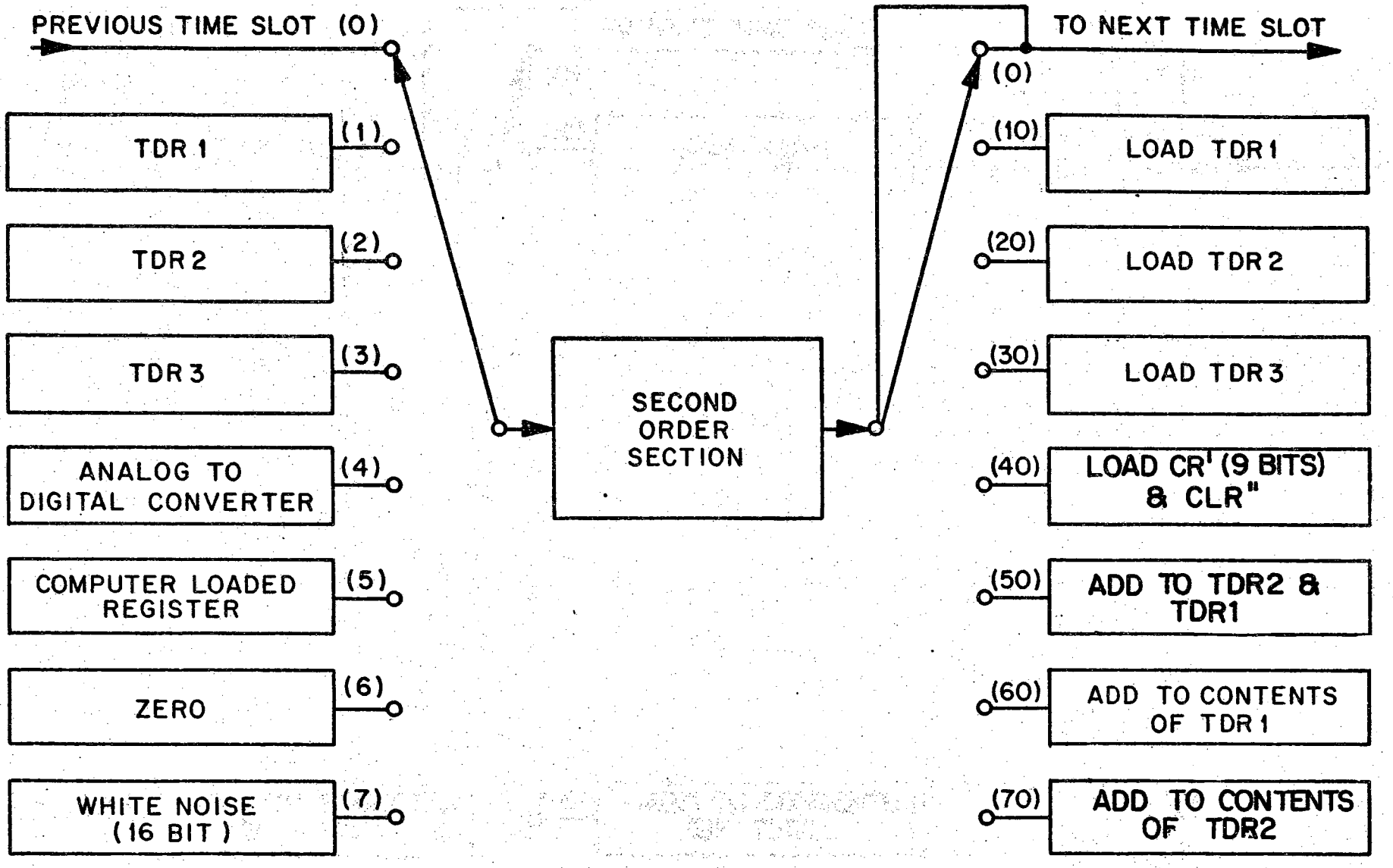
$$I_n = INPUT_n + \beta_1 \ I_{n-1} + \beta_2 \ I_{n-2}$$

$$OUTPUT_n = I_n + \alpha_1 \ I_{n-1} + \alpha_2 \ I_{n-2}$$

$$I_{n-2} = Z^{-1} \ I_{n-1}$$

$$I_{n-1} = Z^{-1} \ I_n$$

FIG.1 BASIC SECOND ORDER FILTER



INPUT SELECTOR
1's DIGIT, COEFFICIENT 6

OUTPUT SELECTOR
10_8's DIGIT, COEFFICIENT 6

' COEFICIENT REGISTER
" COMPUTER LOADED REGISTER

FIG 2 INPUT/OUTPUT SELECTOR

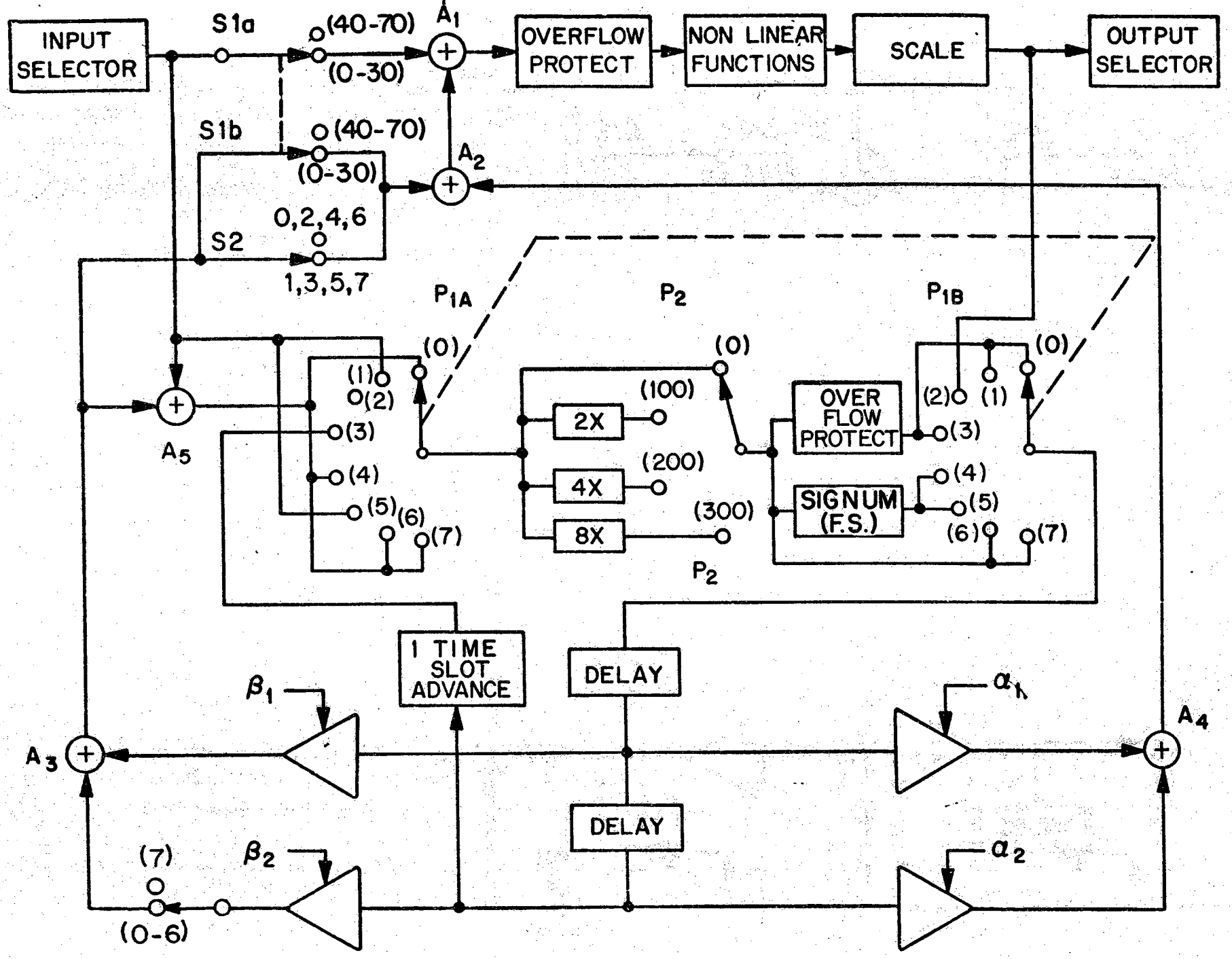
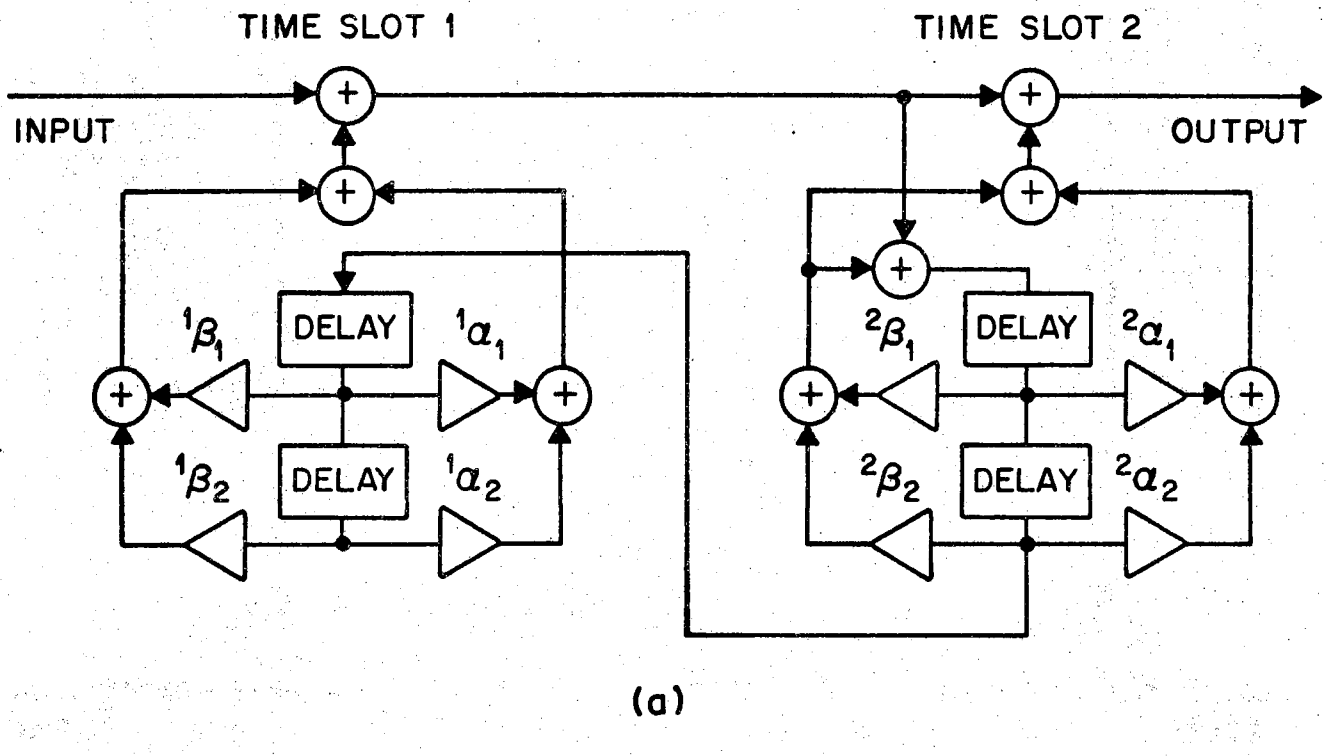
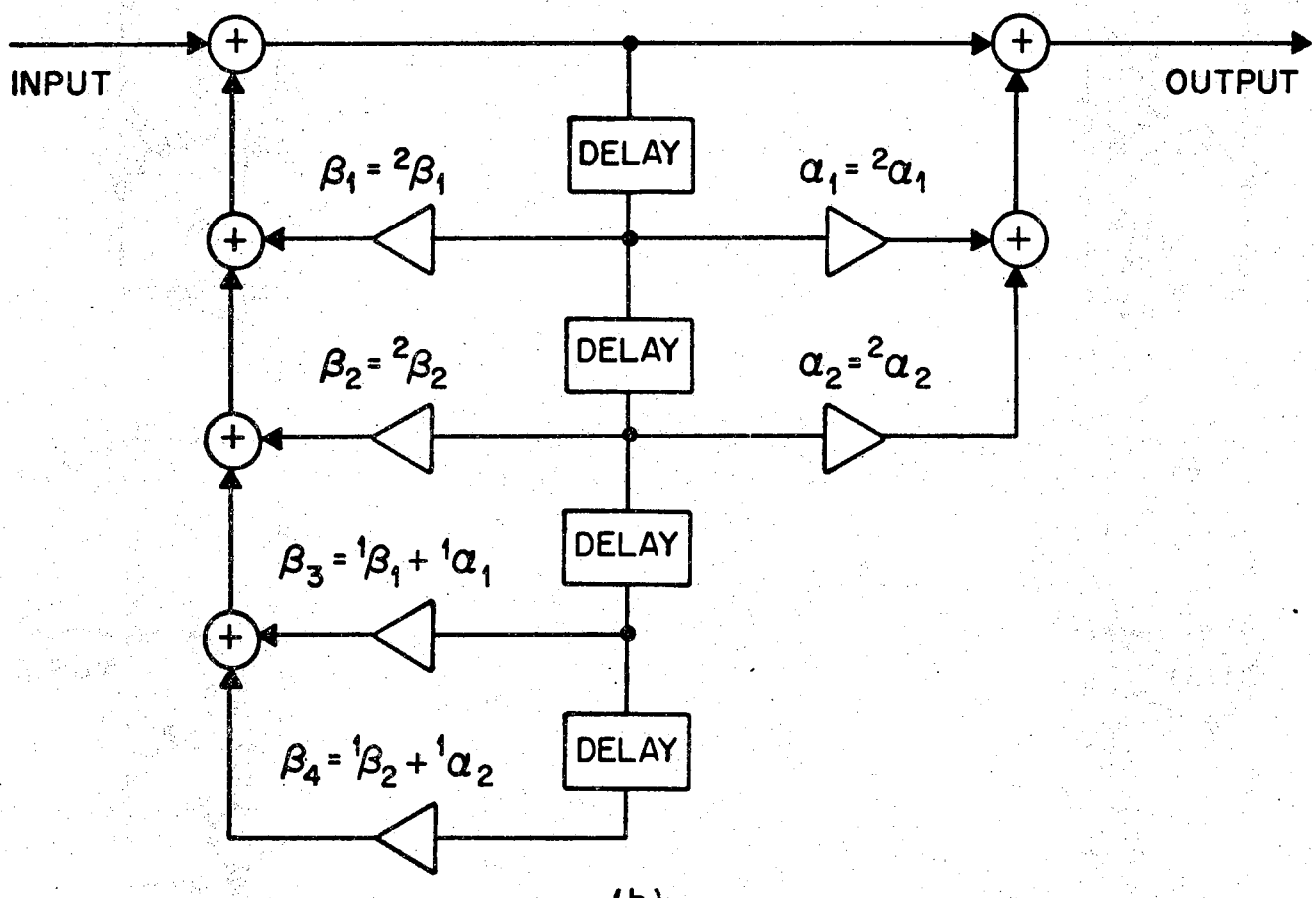


FIG.3 COEFFICIENT 5 DELAY OPTIONS



(a)



(b)

FIG.4 FOURTH ORDER IIR FILTER

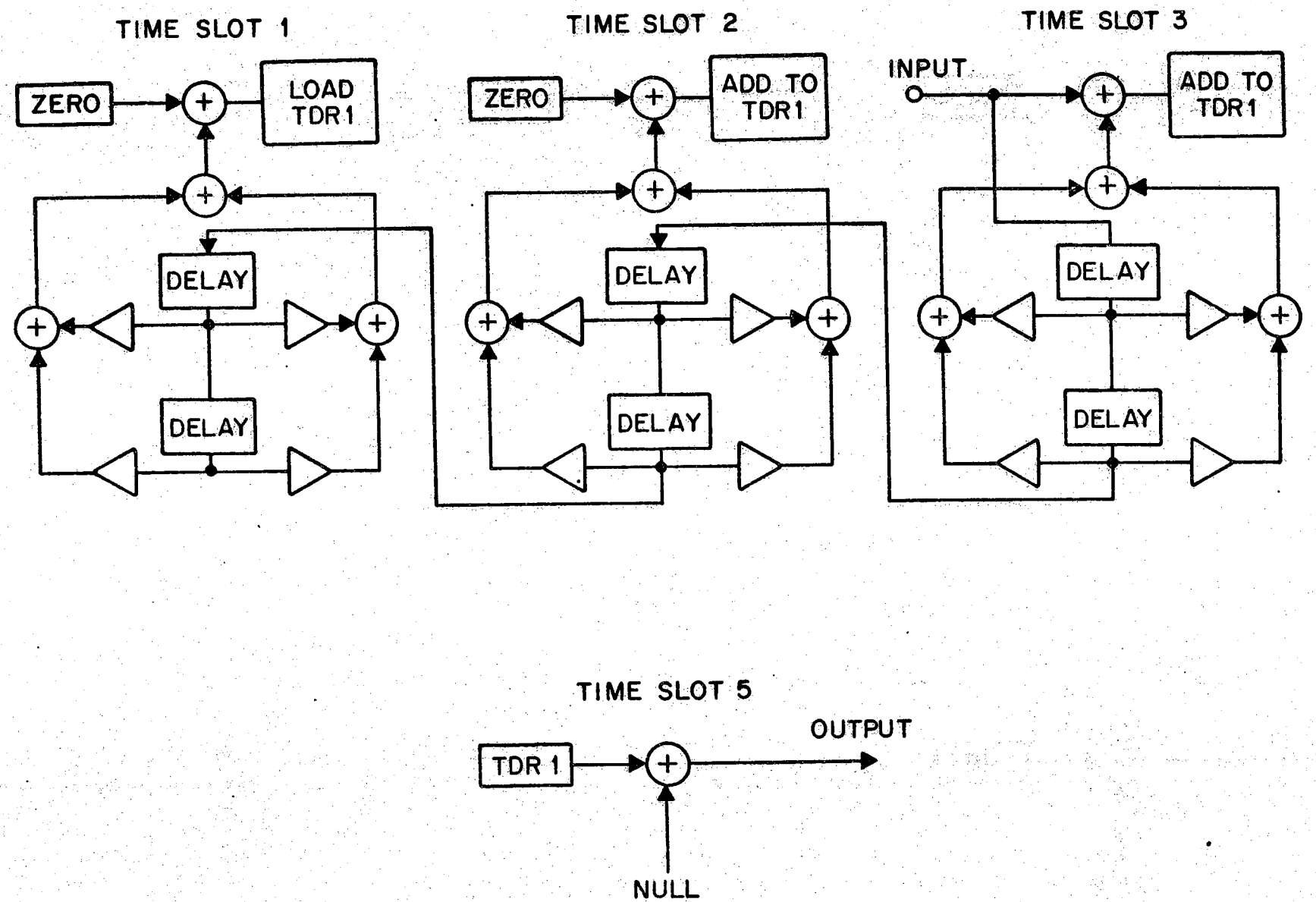


FIG. 5 SIXTH ORDER FIR FILTER

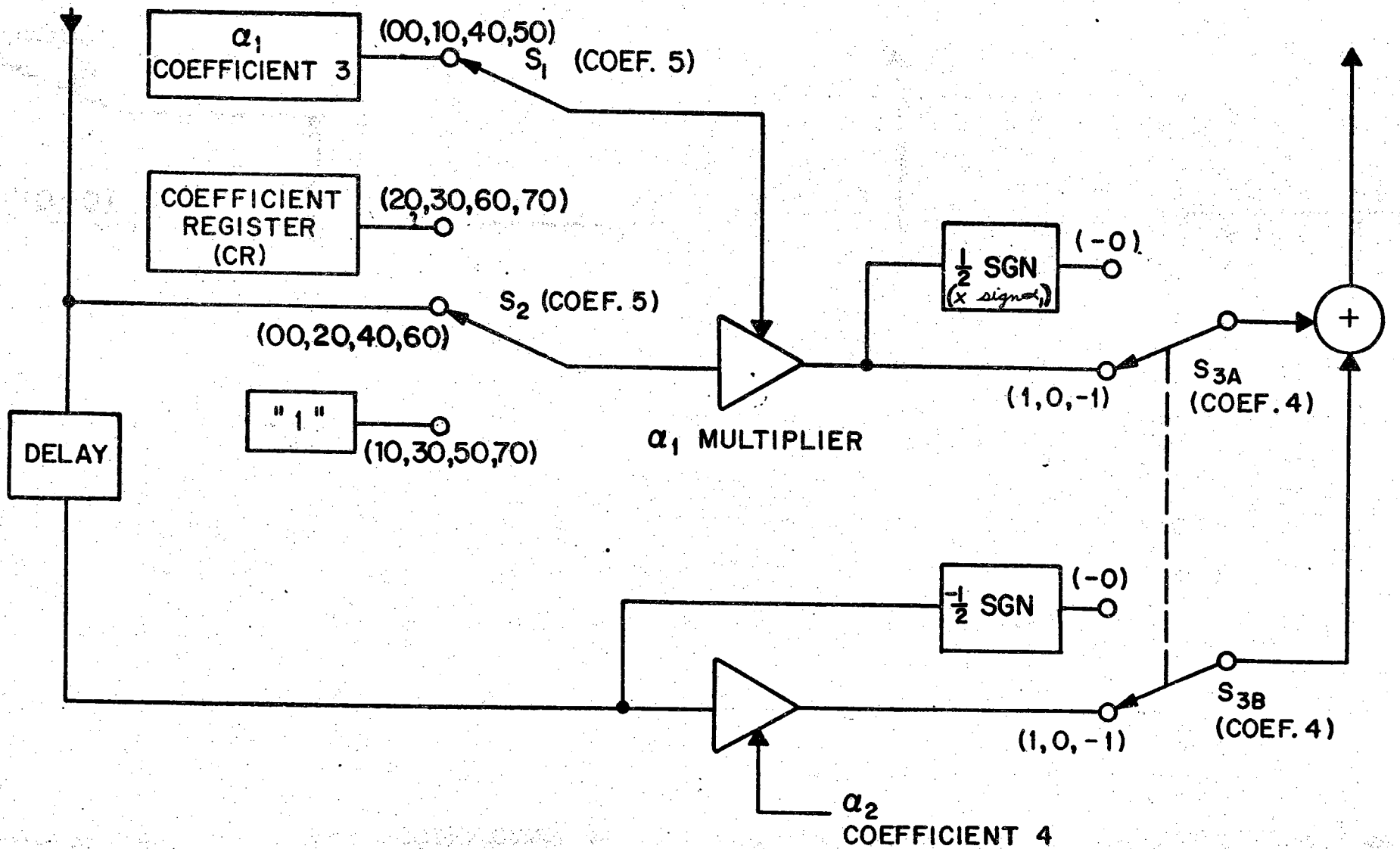


FIG.6 α_1 AND α_2 OPTIONS

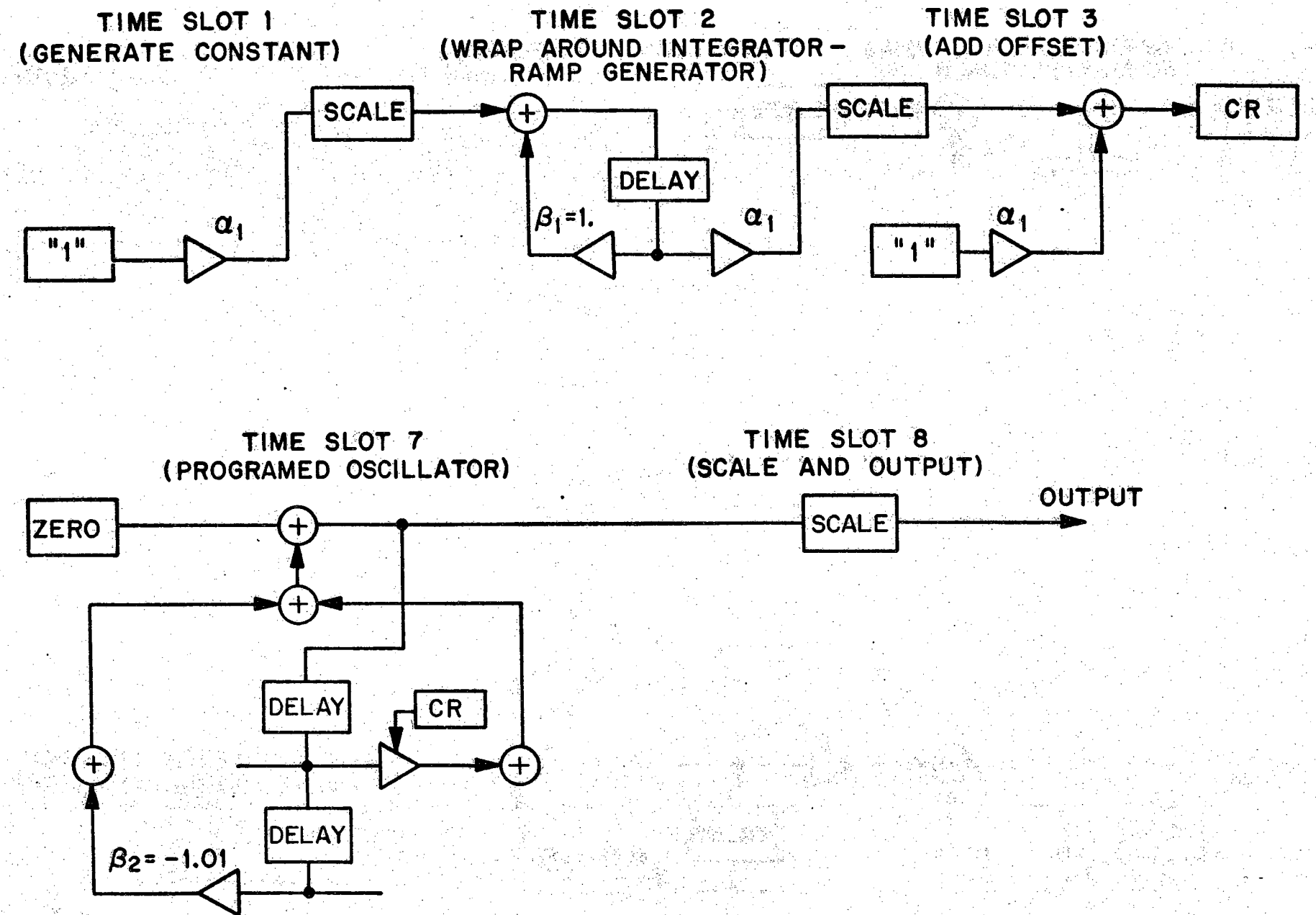


FIG. 7 SWEEP FREQUENCY OSCILLATOR

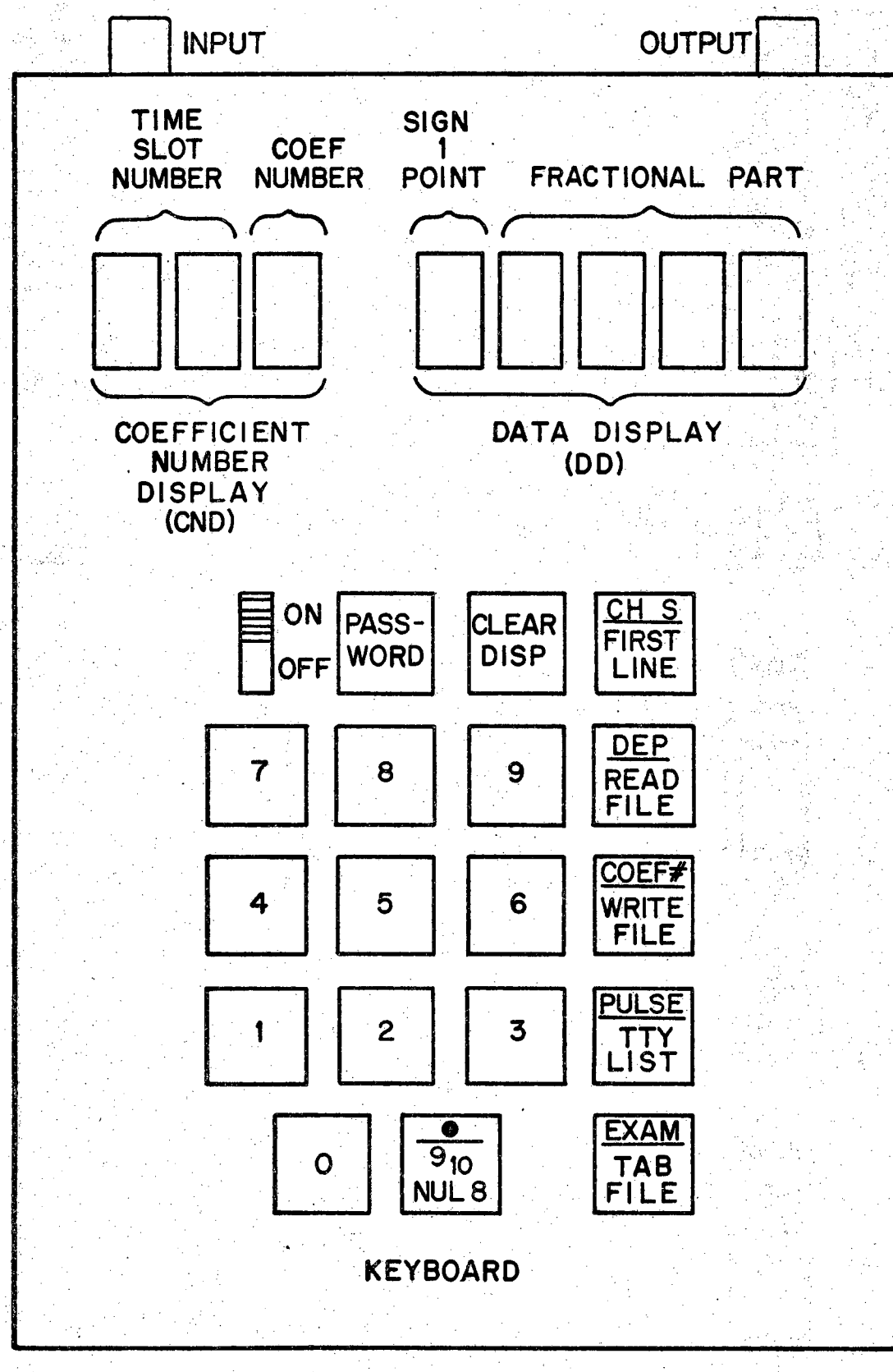
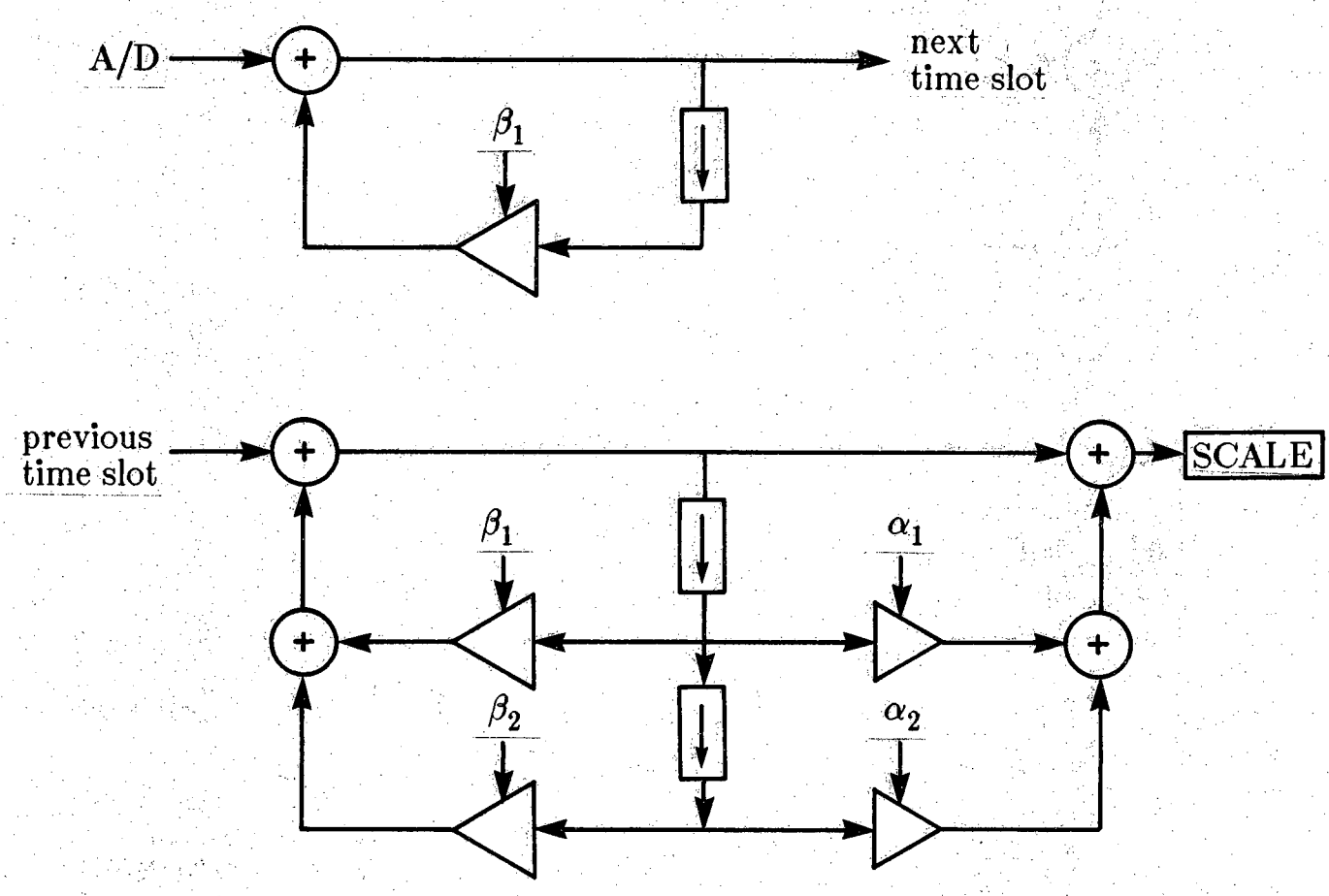


FIG. 8 TERMINAL KEYBOARD AND DISPLAYS

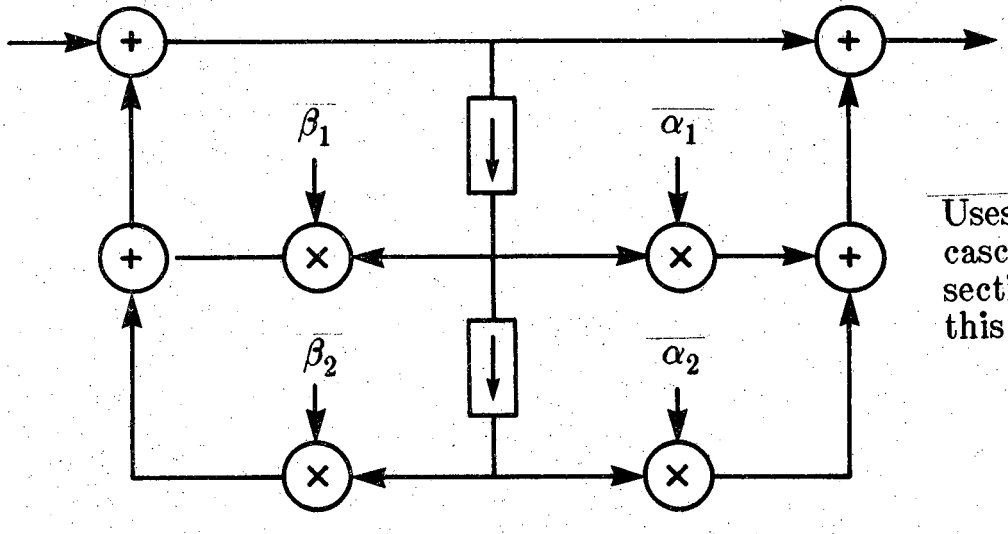
3rd order Cheb. Low-pass

3dB ripple Cutoff at 1.27 kHz



Bell Unit Coefficients (Octal)						
section	β_1	β_2	α_1	α_2	cntrl	i-o
1	0.5763	0.0000	0.0000	0.0	0	4
2	1.0414	-0.5763	1.1600	1.0	1040	0*

6th Order Band-Reject (Lee Jackson)
3dB cutoffs at 500 Hz and 890 Hz
3 stop-band zeros



Uses 3 cascaded sections of this type

Bell Unit Coefficients (Octal)						
section	β_1	β_2	α_1	α_2	cntrl	i-o
1	1.6070	-0.7164	-1.6340	1.0	0	4
2	1.3425	-0.6604	-1.5140	1.0	1000	0
3	1.4210	-0.6050	-1.5740	1.0	0	0*

Appendix A
DMA

The PDP-11 DMA Interface

This interface is designed to use the MDB-11B DMA controller and a custom user control and status register. The MDB-11B interface is covered in the MDB-11B instruction manual and will not be covered here.

The function of this interface is to provide a custom command and status register (CSR) and to handle BA and WC updating and interrupt processing.

There are 4 registers for controlling this device. They are the command and status register, the word count register, the bus address register, and the data register. The address of the first register is 767770 octal. The interrupt vector is 310 octal and the interrupt priority is set at level 5.

The first register is the word count register (wc). It must be loaded with the twos compliment of the number of words of data to be transferred.

The second register is the bus address register and it must be loaded with the starting memory address for the transfer.

The third register is the command and status register. This register controls all operations of the interface and must be loaded last if the GO bit will be set.

The last register is the data register. This register gives the programer access to the data lines to and from the user device on a programed i/o basis and will not affect the dma transfer of data.

All registers are 16 bits long and do not work properly with the DATOB unibus transfer so stay away from byte mode instructions.

The layout of the CSR is as follows.

bit 15 - transfer error: stops transfer on bus timeout error also clears go bit,sets ready bit, and causes an interrupt if interrupt enable is set. bit is read only but is cleared by any write to CSR.

bit 14,13,12 - device select:these bits are read/write and are available to the user as latched ttl signals.

bit 11,10,9 - status inputs:these bits are read only and represent unlatched ttl signals from the user device.

bit 8 - read/write:this bit controls the direction of the data transfer. Clear transfers data from the user device to memory and set transfers data from memory to the user device. This bit is read/write.(of course)

bit 7 - ready:this bit is set when the interface has completed a data transfer or an error has occurred during a unibus data transfer. This bit is read only but is cleared when the go bit is set.

bit 6 - interrupt enable:If this bit is set, the MDB-11B will cause a single unibus interrupt when ready is set by the hardware. This bit is read/write and cleared by unibus init.

bit 5,4 - address extension bits A17, A16:These bits allow dma transfers to extended memory on any PDP-11 unibus. Note that unlike the DEC DR-11B dma interface, data transfers can cross 32k word boundaries. These bits are read/write.

bit 3,2,1 - function select:these bits are available to the user device as latched ttl signals.These bits are read/write.

bit 0 - go bit:setting this bit will start dma transfers to or from memory as defined by bit 8. Decrementing the csr will stop transfers without reseting the word count or bus address registers and transfers may be continued by incrementing the csr. Decrementing and incrementing will preserve the device select,read/write,interrupt enable,memory extension, and function select bits. This bit is read/write and is cleared by word count equal to zero or error bit set.

The CSR:

The csr is loaded by the anding of device sync, csr select and c1 control. This produces cs load in ic D2 which clocks data from tne unibus into latches and counters. The function select and device select bits are latched in ic F1 and F2 and sent to the output connector P2. The bus address extension bits are latched in one half of ic F2 which is connected as a counter so that the dma transfer can pass over 32k boundaries. The error latch has a zero clocked into it by cs load and the go bit loads bit 0. The error bit is set by the signal bus time out which is caused by a dma cycle where slave sync is not returned from the memory with in 30 microseconds of tne start of a transfer. This signal is developed on the MDB-11B. The go bit is set by program control and cleared by any one of three things. Done which is generated by the word count register being equal to zero and the completion of a dma cycle, MR which is a buffered unibus init signal, and error which indicates that something is wrong with either the hardware or software.

The control signals.

There are three transfer control signals that handle all of the handsnaking between the user device and the MDB-11B. These signals are Data ReQueST (DRQST), Data DONE (DDONE), and Data ReQuest ReaDY (DRQRDY). Data request is a signal from tne user device which indicates that the user device is ready for a data transfer. If the transfer is from the memory to the device, this signal indicates that the device is ready to receive data. If the transfer is from the device to memory, then this signal indicates that data from the device is stable and ready to be sent out on the unibus. Data done is the response to a data request and is generated by the MDB-11B. In a transfer from memory to the device, this signal indicates that the data from memory is now stable and the device can store it. In a transfer from the device to memory, it indicates that the data has been transferred. The last control signal is data request ready. This signal indicates tnat tne MDB-11B is ready for a data request from the user device and that the word count is non-zero and the go bit is set.

The Interrupt Logic

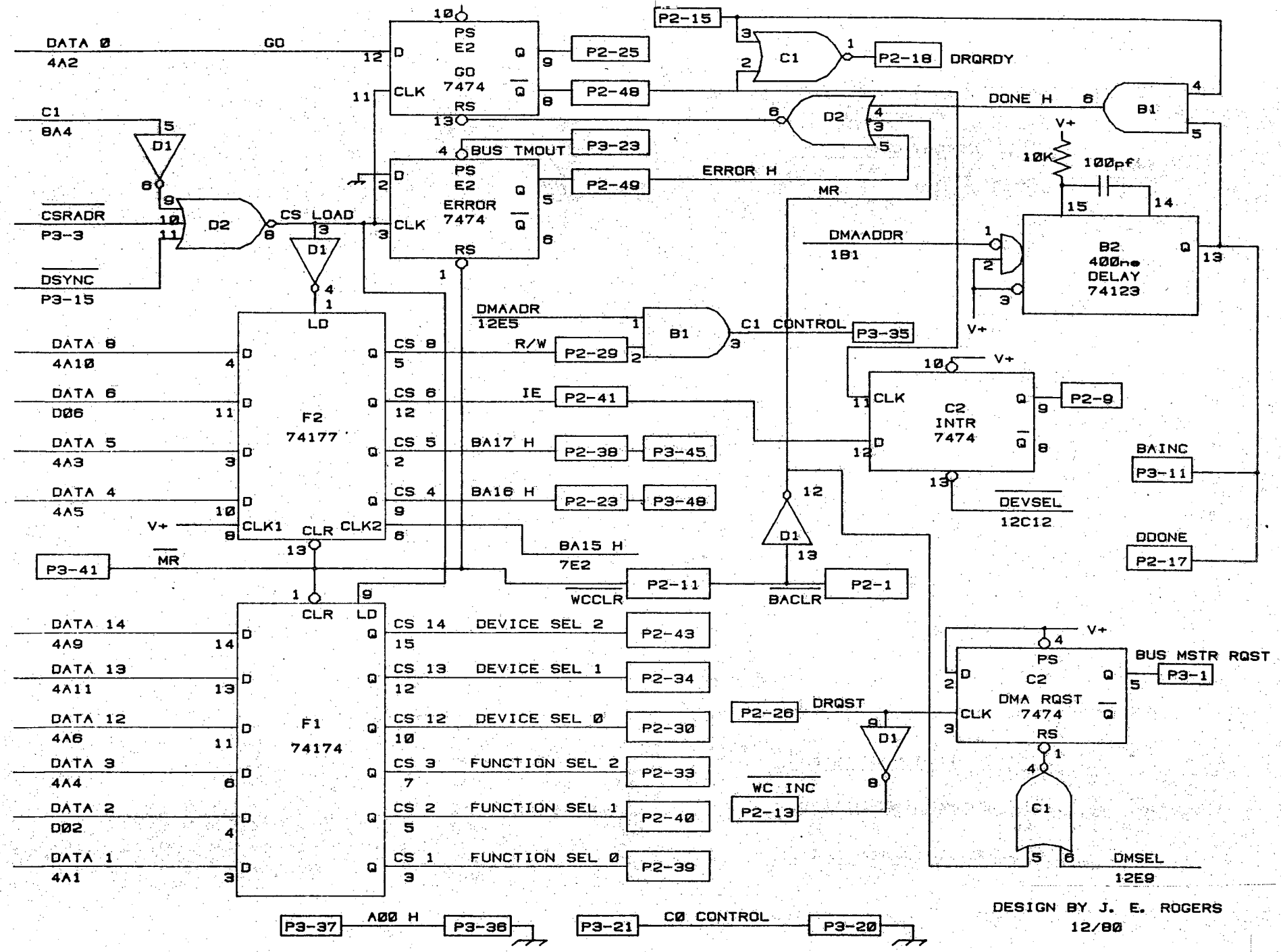
Interrupt enable is latched in ic F2 and becomes the data input to ic C2 which is tne interrupt flip-flop. When tne go latch is reset at the end of a transfer, it clocks a one into ic C2 which starts a unibus interrupt. The acknowledgment from the cpu clears interrupt via tne device select signal to prevent multiple interrupt requests.

The DMA Sequence

The DMA sequence is started when DRQST clocks a one into ic C2 and sets bus master request. This signal also increments the word count register so that if word count equal zero is true then DRQRDY to the user device will go false. DMA RQST is reset when the DMA request is granted by the cpu with the signal DMSEL. When the DMA is completed, the signal DMAADR going false will trigger the one-shot ic B2 which sends DDONE to the user device. This also increments the bus address register and if word count equals zero, it will clear the go bit.

Control and Data Signal Connections

SIGNAL NAME	CONNECTOR and PIN NUMBER
DRQST	P2-26
DDONE	P2-17
DRQRDY	P2-18
MR-	P3-41
STATUS 0	P2-37
STATUS 1	P2-42
STATUS 2	P2-50
FUNCTION 0	P2-39
FUNCTION 1	P2-40
FUNCTION 2	P2-33
DEVICE SELECT 0	P2-30
DEVICE SELECT 1	P2-34
DEVICE SELECT 2	P2-43
DATO 0	P3-49
DATO 1	P3-5
DATO 2	P3-26
DATO 3	P3-6
DATO 4	P3-7
DATO 5	P3-25
DATO 6	P3-28
DATO 7	P3-29
DATO 8	P3-27
DATO 9	P3-39
DATO 10	P3-43
DATO 11	P3-38
DATO 12	P3-44
DATO 13	P3-47
DATO 14	P3-50
DATO 15	P3-46
DATI 0	P2-3
DATI 1	P2-4
DATI 2	P2-6
DATI 3	P2-5
DATI 4	P2-24
DATI 5	P2-27
DATI 6	P2-7
DATI 7	P2-28
DATI 8	P2-31
DATI 9	P2-35
DATI 10	P2-32
DATI 11	P2-36
DATI 12	P2-45
DATI 13	P2-47
DATI 14	P2-44
DATI 15	P2-46


DATA Ø
4A2
GO
PS
E2
GO
7474
CLK
RS
P2-25
P2-48
P2-15
C1
P2-18
DRQRDY
DONE H
B1
C1
8A4
D1
D2
BUS TMOUT
P3-23
ERROR H
P2-49
ERROR
V+
10K
100pf
CSRADR
P3-3
DSYNC
P3-15
CS LOAD
MR
DMAADDR
1B1
B2
400ns
DELAY
74123
DMAADDR
12E5
R/W
P2-29
C1 CONTROL
P3-35
LD
DATA 8
4A10
DATA 6
D06
DATA 5
4A3
DATA 4
4A5
F2
74177
CS 8
CS 6
CS 5
CS 4
IE
P2-41
BA17 H
P2-38
P3-45
BA16 H
P2-23
P3-48
CLK1
CLR
CLK2
C2
INTR
7474
P2-9
BAINC
P3-11
DEVSEL
12C12
DDONE
P2-17
BA15 H
7E2
P3-41
MR
WCCLR
P2-11
BACLR
P2-1
DATA 14
4A9
DATA 13
4A11
DATA 12
4A6
DATA 3
4A4
DATA 2
D02
DATA 1
4A1
F1
74174
CS 14
DEVICE SEL 2
P2-43
CS 13
DEVICE SEL 1
P2-34
CS 12
DEVICE SEL Ø
P2-30
CS 3
FUNCTION SEL 2
P2-33
CS 2
FUNCTION SEL 1
P2-40
CS 1
FUNCTION SEL Ø
P2-39
P2-26
DRQST
WC INC
P2-13
BUS MSTR RQST
P3-1
C2
DMA RQST
7474
DMSEL
12E9
P3-37
A00 H
P3-36
P3-21
C0 CONTROL
P3-20
DESIGN BY J. E. ROGERS
12/80

DMA Custom Interface Board Annotations

The schematic (located in the second from top drawer in the lab file cabinet) that this annotation accompanies is the custom interface contained in the PDP 11/40 computer. This board is activated whenever an "in" or "out" command is initiated by the user.

The board itself has two 40 pin and two 50 pin berg connectors. These berg connectors are referred to as P1, P2, P3 and P4 on the custom interface schematic. Each pin in any 40 or 50 pin berg connector is denoted as P1/2/3/4-xx where xx is a number between 1 and 40.

The abbreviations DIR, DIL, DOR, and DOL stand for data input right, data input left, data output right, and data output left, respectively. Berg connector P4 contains the data input lines, and berg connector P1 contains the data output lines. Data input and output lines originate at the left and right "in" and "out" sockets on the white connector panel located on the right-hand side of the equipment rack. Other signals on P4 are input load A, load B, "stereo/mono", "1" (+5V), "0" (0V), and grounds. P1 also contains output load C, load D, "1", "0", and grounds. Table 1 summarizes the pin assignments of connectors P1 and P4.

Table 1 Pin Assignments for 40 Pin Berg Connectors P1 and P4		
Pin Number	P4 (Input)	P1 (Output)
1 - 15	DIR0 - DIR14	DOR0 - DOR14
16 - 30	DIL0 - DIL14	DOL0 - DOL14
31	GRD	GRD
32	LOAD A (Right)	LOAD C (Right)
33	GRD	GRD
34	LOAD B (Left)	LOAD D (Left)
35	GRD	GRD
36	Stereo/Mono (1/0)	-
37	GRD	GRD
38	"1"	"1"
39	GRD	GRD
40	"0"	"0"

Berg connectors P2 and P3 connect this custom interface board with the PDP 11-40 DMA interface board. The abbreviations DATI and DATO stand for data input and data output respectively. Additionally, berg P2 contains DMA control signals data request (DRQST), data done (DDONE), data request ready (DRQRDY), status 0, and INPUT. Berg P3 also contains control signal $\overline{MR}$. The control signals status 1, status 2, function 0, function 1, function 2, device select 0, device select 1, and device select 2 are not used on this custom board. Table 2 summarizes the pin assignments of connectors P2 and P3. No connection exists where a pin number is omitted.

Chip location on the board is given by a letter-number combination. The letters are A, B, C, D, E, F, and H, and the numbers are 1, 2, 3, 5, 6, 7, 9, and 10. These represent row-column information.

Table 2
Pin Assignments for 50 Pin Berg
Connectors P2 and P3

Pin Number	P2 (Input)	Pin Number	P3 (Output)
3	DATI 0	5	DATO 1
4	DATI 1	6	DATO 3
5	DATI 3	7	DATO 4
6	DATI 2	25	DATO 5
7	DATI 6	26	DATO 2
17	DDONE	27	DATO 8
18	DRQRDY	28	DATO 6
24	DATI 4	29	DATO 7
26	DRQST	38	DATO 11
27	DATI 5	39	DATO 9
28	DATI 7	41	$\overline{\text{MR}}$
29	INPUT	43	DATO 10
30	DEV SEL 0		
31	DATI 8	44	DATO 12
32	DATI 10	46	DATO 15
33	FUNCTION 2		
34	DEV SEL 1		
35	DATI 9	47	DATO 13
36	DATI 11	49	DATO 0
37	STATUS 0	50	DATO 14
39	FUNCTION 0		
40	FUNCTION 1		
42	STATUS 1		
43	DEV SEL 2		
44	DATI 14		
45	DATI 12		
46	DATI 15		
47	DATI 13		
50	STATUS 2		

Appendix B
Disk Drives

Disk Drives

Logical Devices on the CDC-9766 Drive

Logical Device Letter	Logical Device Number (Internal)	Cylinders
b	2	0 - 63
c	3	64 - 127
d	4	128 - 191
e	5	192 - 255
f	6	256 - 319
g	7	320 - 383
i	8	384 - 447
j	9	448 - 511
k	10	512 - 575
l	11	576 - 639
m	12	640 - 703
n	13	704 - 767

Each logical device holds 19.922944 Mbytes. There are

64 cylinders / logical device
19 tracks / cylinder
32 sectors / track
512 bytes / sector
608 sectors / cylinder

There are 239.07533 Mbytes total used on the drive.

Logical Devices on the Aries Drive

Logical Device	Description
a	top and bottom surfaces of upper disk; accessible by user
h	top surface of lower disk; not to be used but is accessible by user (stores help files)
system (shell)	bottom surface of lower disk;
software	not accessible by user

There are

4 surfaces
406 tracks / surface
12 sectors / track
512 bytes (256 samples) / sector.

Appendix C
Compilation and Permanent Lab Disk Storage of DMC and SW2 Programs

Compilation and Permanent Lab Disk Storage of DMC and SW2 Programs

July 12, 1982

This document describes the procedure for compiling and downloading the Switch II and DMC software. This procedure is used only when changes must be made to that software. The object code is stored on the bottom surface of the lower platter on the Aries disk drive. That surface is accessible by this procedure. That surface also stores PDP 11-40 system software known as the RKDP monitor. The software is written in the C programming and can be found on the EA machine (ECN) in

/c/bass/ldlab

The C language software is a compendium of subroutines in separate files which, when linked, form the Switch II or DMC programs.

To begin with, your .profile must have the following contents:

```
PATH=:$HOME/bin$PATH:bin
export PATH
```

Compilation of All Subroutines (from main directory "base"):

Type the following commands from LSI terminal:

```
cc3 -c -O *.s *.c
(CAPITAL Oh not zero)
```

then

```
catd dmc <dummy>
```

or

```
catr sw2 <dummy>
```

where dummy is any group of ascii characters ≠ existing filename. No errors or warnings should be returned by the compiler! If errors are returned see the README file in the lib directory.

If it is desired to make changes to only one of the subroutines, it is not necessary to perform the entire compilation procedure (which takes a long time,

by the way). Having all the .o files already in the "base" directory as a result of having previously issued the cc3 command, it is only necessary to replace dummy with the name of the newly modified subroutine file and to proceed from either the catd or catr commands ("cat" the catd and catr command files in the directory base/bin to see which subroutine files are associated with those commands).

Assuming that you have successfully compiled the programs, you should find that new files have been placed in the directory "base" called dmc or sw2, as a result of catd or catr, respectively. These are the two object files we wish to download into the lab PDP 11-40. Note that it is not necessary to compile and download sw2 if only dmc is changed, and vice versa.

Downloading:

One must be acquainted with the Partial Listing of the RKDP Monitor Manual kept (usually) in the top drawer of the Tektronix Terminal's desk. This is necessary because the procedure for dumping the compiled programs onto the aries disk might require the understanding of the somewhat cryptic error messages generated by the RKDP software provided by DEC.

Bring up the PDP by manually setting 773110 on the front panel. First halt the processor and simultaneously load the address. Then start the processor, enable the processor, and start the processor again. The Tektronix terminal should be set up (previously) for operation at 2400 baud and the metal RS-232 switch-box should be set on the 4th position. The PDP should type a whole mess of instructions and finally a lone period.

On the Tektronix terminal type:

```
r upd2
```

Enter the date in the proper format:

```
(ex.:  21-feb-82)
```

Type

```
load dk0:ldr1.bin
```

Restart the PDP at location 122500. (This is the starting location of the ldr1 program just loaded into core.) On the LSI terminal type

```
dl dmc
```

or

```
dl sw2
```

(The PDP 11-40 should have stopped at the address 122712, otherwise dmc (sw2) must have been too big.) When this command has finished restart the PDP at location 131444. (This is the restart address of the upd2 system program.) On the Tektronix terminal type:

```
locore 0
xfr
1000
```

At this point we are ready to do the dump onto disk. But if the program we wish to dump has the same name as one on the disk an error message will come up which looks like DELOLD and means; delete the old program of the same name. It is recommended that the old program be saved as a precaution until the new program has been verified. This can be done via the RENAME command. Later the old program "old" can be deleted via del dk0:old.bin . Note that the ".bin" is an appendage to every aries PDP file name (in this case "old"), and the "dk0" stands for disk 0 on the aries disk drive. Alternately, load the old program into core (before executing the download dl) and then dump it back onto the disk under a new name "new". The command sequence would be

```
load dk0:old.bin
dump dk0:new.bin
del dk0:old.bin
```

The results of this sequence are equivalent to having used RENAME. At this point, type

```
dump dk0:dmc.bin
```

or

```
dump dk0:sw2.bin
```

If you get a device error (DEVERR) this means that the "prot fixed" switch is set on the Aries disk drive. Turn it off. (Do the dump again.) The program has now been dumped into the file on the aries called dmc. To examine the directory to see that it was in fact dumped, type:

```
dir dk0:
```

To run the program, restart the 11-40 at 773110 then type:

```
r dmc
```

or

r sw2

Checking

Don't forget to put the "prot fixed" switch back on. Check for proper operation of the dmc program by attempting a "link" command. Check for proper operation of the sw2 program by attempting an "fft" command. If the link doesn't work in the dmc, see README in the subdirectory lib. If the fft doesn't work, you did something wrong, try again.

Running from Core with no Download to Disk

If it is desired to run either program from core without having to dump them onto disk, after the download (dl) restart the PDP at location 1000. There is another way to download from the network host (A-machine) into the PDP RAM; i.e., restart at 773110 and

r ldr1

then

dl dmc

or

dl sw2

Then restart the PDP from 1000.

Appendix D
ECN PDP-11 Bootstrap Procedures

ECN PDP-11 Bootstrap Procedures

Introduction

Several different stages are involved when "deadstarting" UNIX. These can be roughly classified as follows:

1. **Hardware boot.** This is accomplished via the front panel of the computer. The end result is that block zero from some device (disc or tape) is loaded into core memory at location zero and started.
2. **Primary bootstrap.** On the V7 PDP-11's, the hardware bootstrap loads a one-block program which, by itself, cannot handle large files (such as the UNIX kernel). As a result, the primary bootstrap automatically locates and loads the program *boot* and transfers control to it. The operation of the primary bootstrap is invisible to the operator (unless an error occurs).
3. **Secondary bootstrap.** The standalone program *boot* comprises the secondary bootstrap. It is a general program which is capable of loading and patching other standalone programs (or a kernel). Typically, it is used to load the file "/unix".
4. **UNIX kernel.** Once *boot* has loaded the file "/unix" and transferred control to it, the UNIX kernel is running. Finally, it brings up process 1 ("/etc/init") which is used to bring up normal single- or multi-user operation.

It is the intent of this short manual to describe how to bring up UNIX on the PDP-11's. First, the general details of the hardware and software are discussed. Second, a "cookbook" procedure is given for each ECN PDP-11 system.

Hardware Bootstrap

A hardware bootstrap may be accomplished in one of two ways:

1. The bootstrap ROM may be used.
2. Using the front panel, a READ may be functioned into the device register for the bootstrap device.

The first alternative is simpler and less prone to error. Unfortunately, non-DEC devices (e.g. the SI/CDC discs) cannot be booted in this fashion. Note also that the boot procedure varies from machine to machine (because of different device register locations and different machine types).

The 11/70's can be booted from the RP04 by starting the processor at 17765000 (octal). [The bootstrap on the PDP-11/70 also includes some quick internal diagnostic checks.]

The 11/45's may be booted from the RK05 (or Pertec) discs by starting the processor at 773110.

A boot from the SI/CDC drives can be performed by functioning a read into the device register. However, the address of this register differs from machine to machine. The general procedure is as follows:

1. WRITE PROTECT THE DISC. DO NOT SKIP THIS STEP!
2. Halt the processor and perform a bus reset (press START with HALT down).
3. Load the address of the disc device register. This is 17771700 on the 11/70's, 770700 at ARPA and 776700 at EEG.

4. Deposit 21, examine, deposit 71.
5. Unprotect the disc.
6. Start the processor at zero.

Primary and Secondary Bootstraps

In most cases the primary bootstrap is invisible. This is not the case, however, on FU (where there is no secondary bootstrap) or when deadstarting from tape. These special cases are dealt with in a different section.

Since the primary bootstrap is invisible, the first visible indication of a successful boot is the secondary bootstrap. Once the secondary bootstrap has been loaded, it prints

```
Boot
:
```

and awaits input. If nothing is typed it will eventually time out and load a default boot file. (The timeout interval is machine dependent; it ranges from about five seconds on an 11/70 to about 15 seconds on a cacheless 11/45.) Typing any character will disable the timeout. All input is mapped to lower-case. Both a pound-sign and a control-H act as erase characters; both an at-sign and a control-X act as line kill characters.

Normally, a device-filename specification is entered in response to the colon prompt. The full form of such a specification is:

```
xx(y,z)name
```

where "y" is the unit number (e.g. drive 0, drive 1), "z" is a starting record number (block number for discs, file number for tapes) and "xx" is a device name selected from the following:

rp DEC RP03
hp DEC RP04, device regs at 776700
hq DEC RP04, device regs at 770700
rk DEC RK05
tm DEC TU10 or similar tape (an Aviv/Telex drive or a [sigh] digidata is "similar")
ht DEC TU16
si SI/CDC 9766 disc, 33 sectors, device regs at 771700
sj SI/CDC 9766 disc, 33 sectors, device regs at 770700
sm SI/CDC 9766 disc, 32 sectors
sk SI/CDC 9762 disc, device regs at 776300
sl SI/CDC 9762 disc, device regs at 776700

As an example, to boot from RK05 unit zero, with a filesystem beginning at block zero (e.g. at ARPA), the specification is "rk(0,0)unix". When the root filesystem does not begin at block zero the block number must be calculated; this has been done at each site for the various boot devices.

The *boot* program will provide defaults. If a carriage return is typed (no specification at all), a default device-file specification will be used. If only a filename is typed, the default device will be supplied. For instance, if the default specification is "sm(0,262656)unix", typing "new-unix" is equivalent to typing "sm(0,262656)new-unix". **[Note: This is not true on the VAXes.]**

The *boot* program is capable of patching a loaded program before transferring control to it. The patch facility is invoked by typing a minus-sign instead of a file specification. *boot* will then prompt for symbol names. Enter the names of any symbols to be patched, one per line, (the names "_rootdev", "_swapdev", "_pipedev", "_swplo", and "_nswap" are automatically provided). You need not worry about typing the leading underscore. After all of the symbols have been entered, type only a carriage return (i.e. a blank line). *Boot* will reprompt with a colon. At

this time, the device-file specification should be entered. After the file has been loaded, *boot* will print

```
Patch
->
```

Typing the name of a variable (or typing a numeric constant specifying a machine address) will display the value of that variable. [Note that all operations are performed on 16-bit words.] Typing "variable=value", where "value" is a numeric constant, will change "variable". Constants may be octal (default), decimal (leading "0t") or hexadecimal (leading "0x"). When a blank line is typed, *boot* exits patch mode and starts the loaded binary.

Magnetic/DECtape Bootstraps

Magnetic tape and DECtape boots are not common on the ECN; probably the only time a tape will be booted (outside of running diagnostics) is to completely restore the system from tape after a disastrous crash. The recovery procedure will accompany the "deadstart" tape.

Bootable magtapes and DECtapes are normally written in *tp* format (a holdover from version 6 UNIX). A hardware boot is performed to load block 0 of the tape into memory. This block (the primary bootstrap) is then capable of loading any ordinary executable file. The *tp* bootstrap prompts with an equals sign ("="). To load a file, type its name.

It is not possible to discuss to any great depth the procedure to be used when performing a tape boot. Consult the deadstart tape for the necessary information about which files are available, etc.

The AARL (V6) System

In version 6 there is no secondary bootstrap ("boot" program); instead, the primary bootstrap is capable of loading the kernel. When performing a hardware boot on the AARL machine, the value in the console switch register is important. If the switches are all set to zero, the primary bootstrap will automatically load the file "/unix" and start executing it. Also, the system will come up multi-user immediately. If the switches are set to (octal) 70, the bootstrap will prompt with an at-sign ("@"). The name of the file to be loaded can then be entered. The system will come up single-user but will go multi-user when a Control-D is typed. Finally, if the switches are set to (octal) 173030, the bootstrap will prompt with an at-sign, and the system will come up single-user. It will remain single user (even if the console is "logged out"); to start multi-user it is necessary to set the switches to zero and type a Control-D on the console.

UNIX Cookbook

<or>

How to Boot in 10 Steps or Less

This section will give step-by-step hardware bootstrap instructions for the various ECN machines. Before beginning this task, however, a few comments are needed concerning the operation of the PDP-11 front panel.

All of the PDP-11 hosts on the ECN have old-style "lights and switches" front panels. While the exact configuration varies somewhat from model to model, the operation of all is very similar. The panel consists of a row of switches and a bank of lights. The switches are usually organized into two principal groups: the group of switches to the left are used for entering 18-bit (PDP-11/45) or 22-bit (PDP-11/70) addresses and 16-bit data, while the switches on the right are used to select various front panel functions (e.g. halt the processor, examine memory, etc.). The lights are roughly arranged into three groups: a row of 18 (11/45) or 22 (11/70) "address" lights, a row of 16 "data" lights, and another set of lights labelled "kernel", "super", "user", "data", etc.

The address and data lights and the address/data switches are color-coded in groups of three. Each group represents one octal digit. The switches are down for 0 (off) and up for 1 (on).

The switches to the right are:

LOAD ADRS — This is a momentary switch. Depressing this switch causes the value in the switches to be transferred into the row of address lights. This defines an address which can be examined, modified, or used to start the processor.

EXAM — This is a momentary switch. When EXAM is depressed, the contents of the current memory location (as determined by the last LOAD ADRS) is displayed in the data lights. [Note: there are two rotary switches on the front panel, each associated with a set of lights. If necessary, adjust these so that the lights "CONS PHY" and "DATA PATHS" are lit.] The normal way to examine memory is to perform a LOAD ADRS, and then a series of EXAMs. The first EXAM will examine the memory address specified by the LOAD ADRS; each successive EXAM will advance the address by 2 so that consecutive words in memory may be displayed without having to re-enter the address for each one.

DEP — This is a momentary switch. This switch, unlike the other function switches, must be raised (rather than depressed). The DEP switch causes the contents of the data switches (switches 15-0) to be deposited into the current memory address. Like EXAM, consecutive deposits are performed on successive addresses.

ENABLE/HALT — This is a two-position switch. If this switch is down, the processor is halted. If this switch is up, the processor may be restarted (see the CONT switch below). Note that raising the switch alone will not cause the processor to restart.

CONT — This is a momentary switch, used in conjunction with the HALT/ENABLE switch. If the processor is halted and the HALT/ENABLE switch is up, pressing CONT will cause the processor to "continue". If the HALT/ENABLE switch is down (the processor will be halted) and CONT is pressed, one instruction will be executed ("single-step" execution).

S INST/S BUS CYCLE — This is a two-position switch. It has no effect if the ENABLE/HALT switch is up. If the ENABLE/HALT switch is down, then the S INST/S BUS CYCLE switch determines the action taken when CONT is pressed. If this switch is up, depressing CONT causes a single instruction to be executed (as described above). If this switch is down, depressing CONT causes a single UNIBUS cycle to be performed. For the purposes of bootstrapping UNIX, this switch should always be up.

START — This switch is used to initialize the UNIBUS and start the processor at a specified address. If the ENABLE/HALT switch is down, then pressing START will initialize the UNIBUS. If the ENABLE/HALT switch is up, pressing START will initialize the UNIBUS and start the processor at the last address specified by a LOAD ADRS.

Bootstrap Procedure:
ODSP PDP-11/40

This machine can be booted from the Dynex (Aries which is an RK05-type drive). The Dynex boot procedure is the same as an RK05 boot:

1. Put the HALT/ENABLE switch down, and press START, To reset the devices on the UNIBUS.
2. Load address 773110 (place the value 773110 into the switches and press LOAD ADRS)
3. Raise ENABLE/HALT and press START.

ACKNOWLEDGEMENTS

ACKNOWLEDGEMENTS

The authors wish to express their thanks to Tom Goeddel and Todd Wilson for their invaluable participation in both the design of the original ODSP hardware and software, and the preparation of this document.