NOTICE

**CERTAIN DATA
CONTAINED IN THIS
DOCUMENT MAY BE
DIFFICULT TO READ
IN MICROFICHE
PRODUCTS.**

CHEMICAL CHARACTERIZATION OF THE SURFACE SITES OF COAL

Technical Progress Report for the Period

January - March, 1990

RECEIVED BY USF

OCT 0 1990

Frederick M. Fowkes

with

David A. Cole.

Department of Chemistry

Lehigh University

Bethlehem, PA 18015

August 1990

PREPARED FOR THE UNITED STATES

DEPARTMENT OF ENERGY

Under Contract No. DE-FG22-87PC79925

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

CHEMICAL CHARACTERIZATION OF THE SURFACE SITES OF COAL

7 AUG 18 1964

This report was prepared as an account of work sponsored by the United States Government. Neither the United States, nor the United States Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

II. TECHNICAL PROGRESS

Currently our Microscal flow calorimeters and associated UV and refractive index detectors are interfaced to a MTU microcomputer. This computer has operated with very few problem for several years. However, as it has aged, reliability problems have surfaced. Many of the problems are intermittent such that identification and correction are quite difficult. The original instrument-computer interfaces were custom devices built in-house. Rather than repair them we opted to build new interfaces based on a more rugged design - see text below. We are not, however, in the position to repair the floppy disk drives of the MTU computer which currently corrupt data several time per month. Therefore we felt that it was appropriate to replace the MTU computer with a more reliable model. We also wished to use a *modern* operating system which is widely supported. We chose the MS-DOS operating system since both Lehigh University and the Chemistry Department support this system. The department has established a microcomputer laboratory with nine IBM PS-2 computers, several printers and a Hewlett Packard 7475 plotter in our building for student and faculty use.

Based on the desire to have as many of the laboratory instruments interfaced to this computer as possible, a DELL System 310 computer was selected. This computer is based on the Intel 80386 processor and runs at 20 MHz, fast enough to run Quarterdeck Office Systems' DESQview 386 multitasking environment. This was a prerequisite since we wished to collect data, copy files, and analyze data from at least eight instruments simultaneously. It is equipped with a VGA-16 color monitor with a maximum resolution of 800 by 600, a 40 Mbyte hard disk, two floppy drives (360 Kbyte 5.25" and 1.44 Mbyte 3.5") and a serial mouse.

A MetraByte DAS-8 analog to digital converter board was chosen to

perform A-D conversions. This is an IBM compatible, program selectable, eight channel, single-ended, 12-bit A-D converter with a maximum conversion time of 35 μ seconds and an input voltage range of ± 5 volts dc.

Instrument amplifier-interfaces are required for each of our instruments since the output signals from our Microscal calorimeters, Perkin-Elmer UV spectrophotometric detector, Erma and Knauer differential refractometers, and Setaram C80 calorimeter are all less than ± 5 volts dc. These interfaces are of our own design based on the following requirements.

- Variable signal amplification of up to 1000 times
- Variable signal offset from -5 to +5 volts dc
- Low current drain by the device - no loading of the source
- Low noise characteristics
- Rugged design.

The interface is shown in Figure 1. It is designed around two general purpose operational amplifier integrated circuits - four op-amps per IC. The first quad op-amp buffers the input signals ($+V_{in}$ and $-V_{in}$) and amplifies the differential signal up to 100 times based on the setting of the variable resistor, VR1. The second op-amp amplifies the intermediate signal by 10, filters noise with a RC circuit having a time constant of 0.1 seconds, adds the selected offset voltage, and buffers the output signal. The offset voltage is adjusted by the variable resistor, VR2. The sign of the offset +, -, or none is determined by the setting of dip switches. Five of these interfaces have been constructed so far. They are currently used to interface the Microscal, UV, and RI instruments to the MTU computer. They have been in operation for several months and have proven to be very reliable. The only modification necessary to use these interfaces with the Dell computer is to reset the offset voltage.

Figure 1. Wiring diagram for an instrument amplifier-interface capable of amplifying a dc input signal 1000 times and supplying a 0 or ± 5 volt dc offset. Components: - Quad operational-amplifiers IC1-IC2=LM348N - Capacitors C1-C7=0.01 μ f ceramic, C8-C11=4.7 μ f tantalum, C12=1 μ f ceramic, - Resistors (metal film, 1%) R1-R3=10 k Ω , R4-R5=1 k Ω , R6-R7=15k Ω , R8-R11=4.7 k Ω , R12=2.2k Ω - Variable resistors VR1-VR2=10 k Ω , ten turn - Dip switch SW, 1-on=no offset, 2-on=negative offset, 3-on=positive offset. Components are on top of the board as are dashed-line wires, solid line wires are on the bottom.

The data collection program was written in source code in order to maximize the execution speed and to take full advantage of the multitasking capabilities of DESQview. The code is actually three separate programs. The main program or *parent* task is called CHART. Initially CHART draws the display windows for each of four users. It then pawns two *child* tasks CONVERT and KID. CHART continues to execute the graphics display of current data until task KID signals CHART to exit to DOS, terminating the program. CHART uses a short routine DOT to write graphics data directly to video memory.

The child task CONVERT performs A-D conversions on each channel currently selected for data input based on a round-robin scheme. Data is collected for each channel every 0.5 seconds and written to disk every five minutes. Data collection timing is controlled by a software interrupt timer generated by the DESQview operating system. After all eight channels have been serviced the task is suspended until the next timing cycle begins.

The child task KID processes all user input. Input is required to log a new user, that is to request access to a given instrument (channel) and to open data files. Each of the four possible users is restricted from owning more than two instruments; however any available instrument can be selected. Each user is also restricted from modifying the operation of another users channels. Possible operations on an opened channel are: initiate/terminate data collection, set timing marks to mark notable events, set current voltage marks for visual display, and closing (freeing) a given channel. Comments can be added to the data file when the file is opened or when it is closed. A user can also elect to display the data collected in the past hour. Any of these options can be selected at any time without interfering with data collection. When the last user is finished collecting data he directs the child task KID to inform the parent task CHART to terminate the terminate both

child tasks KID and CONVERT and to exit to DOS. If a user attempts this while another user still owns an instrument, the request is denied.

It is nearly impossible for one user to inadvertently change the operation of another users channels. Before any command is permitted the user must login by typing his/her username. Invalid usernames are ignored. This is a significant advance over our present software. Moreover the ability to open and close files for any given instrument at any time allows a user to take his/her data out of the computer at any time for analysis at another site or in another DESQview window. Users are thus free to do as they wish without consulting others about when they want to begin collecting data or when the will be finished.

The source code for this program is given in the appendix. This code is not public domain or shareware. The code may not be used in part or in whole by anyone without the written consent of the author. It is provided simply as verification of the work done under this grant. Anyone wishing to use this code in any way should contact the author at Lehigh University, Department of Chemistry, Seeley G. Mudd Building #6, Bethlehem, PA.

III. CONCLUSIONS

We have begun the process of replacing our current data collection facilities with a new system based on an IBM compatible computer and a 12-bit analog-to-digital converter. Instrument amplifier-interface for five of our seven instruments have been constructed based on our own design. Assembly language code has been written for data collection. This code task full advantage of the multitasking capabilities of the DESQview 386 operating system. A data analysis program should be completed in the near future at which time the new system will be installed in the lab.

APPENDIX

COPYRIGHT 1990 DAVID A. COLE

This code is not public domain or shareware. The code may not be used in part or in whole by anyone without the written consent of the author. Anyone wishing to use this code in any way should contact the author at Lehigh University, Department of Chemistry, Seeley G. Mudd Building #6, Bethlehem, PA. No warranties, either expressed or implied are made with respect to the quality, performance, merchantability, or fitness for any particular application. Under no circumstances will the author be liable for direct, indirect, or incidental or consequential damages resulting from any defect in this code. Because some states do not allow the exclusion or limitation for incidental or consequential damages, some of the above limitations may not apply to you.

```

-SALL
TITLE << CHART.ASM>> Data acquisition code for use under DESQview
SUBTTL General program initialization
;CURRENT VERSION DATED --- March 23, 1990 ---
;
;
; .286
;use 80286 code for PUSH const in PUSHW and
;rotate right & left an 8-bit const times.

```

```

.model small
include api.inc
include dos.inc
include bios.inc
extrn inpan:word
extrn linpan:word
; compile panel as follows
; source : c:\d\pdt\inpan.obj
; destination : c:\d\pdt\inpan.obj
; address label : inpan
; length label : linpan
; segment : _data
; group : $group
; class : data

```

```

.stack 100h

```

```

SUBTTL << DATA >>

```

```

PAGE +

```

```

.data

```

```

convwin dd 0
kidin dd 0
pinpan dd 0
panhan dd 0
inwin dd 0
inkey dd 0
pinbuf dd 0
linbuf dd 0

```

```

regone dd -1
zero dd 0
twohund dd 200
pkidname dd kidnamep
lkidname dd kidnamekl
kidnamep db 'MENU'
kidnamekl equ $-kidnamep

```

```

chartwin dd 0
inwinadc dd 0
inkeyadc dd 0
chartimer dd 0
chjunk dd 0
adtimer dd 0
junk dd 0
last dd 0
delay dd 50
quitdelay dd 100

```

```

;PANEL "INPAN" DATA

```

```

;ROUTE
rroule dd route
rroule dd route

```

```

route db 'route'
route equ $-route
;SETUP
psetup dd setup
lsetup dd setupl
setup db 'setup'
setupl equ $-setup
;OPENF
popenf dd openf
lopenf dd openfl
openf db 'openf'
openfl equ $-openf
;CLOSEF
pclof dd clof
lclof dd clofl
clof db 'clof'
clofl equ $-clof
;COLEC
pcolec dd colec
lcolec dd colecl
colec db 'colec'
colecl equ $-colec
;EXPRG
pexprg dd exprg
lexprg dd exprgl
exprg db 'exprg'
exprgl equ $-exprg
;ADC
padc dd adcp
ladc dd adcl
adc db 'adc'
adcl equ $-adcp
;GLOBAL TEMPORARY VARIABLES
temp0 dd 0
temp1 dd 0
temp2 dd 0
temp3 dd 0
;CHART
tempax dd 0
tempdx dd 0
chpos dd 0
chdi dd 0
chcount dd 0
initmode db 0
;temp storage for unit #, 1 to 8 in chart
;temp storage for Y offset drawing data line
;starting row for channel(i) pen lines
;pointer to channel#, 1 to 8 for chan1 or chan2
;=1=chan1, =2=chan2 being updated
;initial video mode et start up
;user input for route
;route flag, setup=0, colec=1, disp=2, exprg=3
;display old data flag, 0=no, 1=yes
;0=USER\
;8 characters followed by 00h
;usersum (2 bytes), # chan owned
;chan1, chan2, username (8 bytes)
;0=no unit, 1 to 8 = units.
;pointer to user(i)
;user sum accumulator area
;user login, 0=no, 1=yes
;current user #
;# channels currently available

```



```

d6      dw      1024 dup (0)
v1      db      226 dup (0)
v2      db      226 dup (0)
v3      db      226 dup (0)
v4      db      226 dup (0)
v5      db      226 dup (0)
v6      db      226 dup (0)
v7      db      226 dup (0)
v8      db      226 dup (0)
dv1     db      226 dup (0)
dv2     db      226 dup (0)
dv3     db      226 dup (0)
dv4     db      226 dup (0)
dv5     db      226 dup (0)
dv6     db      226 dup (0)
dv7     db      226 dup (0)
dv8     db      226 dup (0)
pdata   dw      d1,d2,d3,d4,d5,d6,d7,d8,d9,d10,d11,d12,d13,d14,d15,d16
pvld    dw      v1,v2,v3,v4,v5,v6,v7,v8
pdvid   dw      dv1,dv2,dv3,dv4,dv5,dv6,dv7,dv8
dataoff  dw      0,0,0,0,0,0,0,0 ;address to current data storage point
videoff  dw      0,0,0,0,0,0,0,0 ;offset to next point to be written
datebank db      0,0,0,0,0,0,0,0 ;current data bank, 0=A, 1=B
writebank db      0,0,0,0,0,0,0,0 ;write data to disk flag, 0=no, 1=yes
newdata  db      0,0,0,0,0,0,0,0 ;new data collected, flag
vidpen   dw      0,0,0,0,0,0,0,0 ;new pen position, +v=0, -v=0fffh
sum      dw      0,0,0,0,0,0,0,0 ;running sum for video display
sumcnt   dw      0,0,0,0,0,0,0,0 ;sum count for video display
olddata  db      0,0,0,0,0,0,0,0 ;last video data displayed
oldsetl  db      0,0,0,0,0,0,0,0 ;last video left mark displayed
oldsetr  db      0,0,0,0,0,0,0,0 ;last video right mark displayed
viddisp  dw      17,177,337,497 ;column offset from left for user 1 to 4
curnum   dw      0 ;current user number, 0 to 3
chansi   dw      0 ;temp storage for si pointer to channel #, 0 to 7
tempid   db      0 ;temp storage, unshifted video data
tempah   db      0 ;temp storage, 8 MSB

SUBTTL << Output Strings >>
PAGE *
;MAKE KID BACKGROUND TASK
kidback  db      1bh,10h
lkidback db      kidback+1
db      0c9h
kidbackl equ      $-lkidback-2

;HIDE WINDOW FRAME
phidwin  db      1bh,0
thidwin  dw      hidewirt
db      0d7h
hidewint equ      $-thidwin-2

;CHART
prody    db      13,10,'0'
db      '#####'
db      'This program must be run under DESQview by Quardtek, Ver. 2.2b or higher.'
db      'a',13,10
db      'a'
db      'Please press any key to return to DOS...'
db      '#####'

;SETUP
poutput  db      1bh,0 ;window stream to write to field
loutput  dw      output ;length of stream
db      0f3h,0 ;window write and field #
db      , ;
db      $-loutput-2 ;text to write

outputl  equ      $-loutput-2

;window stream to write to field
;length of command string
;make field 1 an input field
;erase old input
;move cursor to left of field 1
;use to move cursor to field 1

1bh,0
curnovel
$-lcurnovel-2

;disables all unit fields
;used initially to limit
;user input to user name
;or escape.
;2 to 9 = unit fields.

1bh,0
disabel  dw      disabel
dw      0f6h,2,0
db      0f6h,3,0
db      0f6h,3,0
db      0f6h,4,0
db      0f6h,5,0
db      0f6h,6,0
db      0f6h,7,0
db      0f6h,8,0
db      0f6h,9,0
equ      $-ldisabel-2

;enables all unit fields initially
;Change 0c0h to 0h when unit is
;selected and file is opened...
;No one can later chose unit(i).
;2 through 9 = units.

1bh,0
enablel  dw      enablel
dw      0f6h,2,0c0h
db      0f6h,3,0c0h
db      0f6h,4,0c0h
db      0f6h,5,0c0h
db      0f6h,6,0c0h
db      0f6h,7,0c0h
db      0f6h,8,0c0h
db      0f6h,9,0c0h
equ      $-lenable-2

;window stream
;display unit name
;unit name
;disable field 5, comments
;field 3 = input, move cursor to #3

1bh,0
outusel  dw      outusel
dw      0f3h,1
db      ,
db      0f6h,5,0
db      0f6h,3,80h
equ      $-loutusel-2

;field 3 = output
;write file name to field 3
;field 4 = output
;write file ext to field 4
;field 5 = input

1bh,0
dis34l  dw      dis34l
dw      0f6h,3,40h
db      0f3h,3,'
db      0f6h,4,40h
db      0f3h,4,'
db      0f6h,5,80h
equ      $-ldis34-2

1bh,0
en34l   dw      en34l
db      0f6h,5,0
db      0f6h,6,0ch
db      0f3h,3,'
db      0f3h,4,'
db      0f6h,4,80h
db      0f6h,3,80h
equ      $-ten34-2

;disable field 5, file comments
;enable field 6, escape
;clear field 3, file name
;clear field 4, file ext
;enable field 4, file ext
;enable field 3, file name

1bh,0
cdiss315l
db      0f6h,3,0,0f6h,4,0,0f6h,5,0,0f6h,6,0,0f6h,7,0,0f6h,8,0
db      0f6h,10,0,0f6h,11,0,0f6h,12,0,0f6h,13,0,0f6h,14,0,0f6h,15,0

;disables fields 3 through 8, 10 through 15
```

```

db 016h,1,80h
equ $-lcdis315-2

pcup
lcup

db 1bh,0
dw cpl
db 013h,3,'
db 016h,3,0
db 013h,4,' Down '
db 016h,4,0c0h
db 016h,1,80h
equ $-lcup-2

pcdown
lcpdown

db 1bh,0
dw cdownl
db 013h,4,'
db 016h,4,0
db 013h,3,' Up '
db 016h,3,0c0h
db 016h,1,80h
equ $-lcpdown-2

pcen58
lcp58

db 1bh,0
dw cen58l
db 016h,5,0c0h
db 016h,6,0c0h
db 016h,7,0c0h
db 016h,8,0c0h
equ $-lcp58-2

pcdisphan
lcpdisphan

db 1bh,0
dw dispchanl
db 013h,2,'
equ $-lcpdisphan-2

PAGE
;CLOSE FILE
pcout:
lcpout:

db 1bh,0
dw cloutl
db 013h,1,'
db 013h,2,'
db 016h,6,C:0h
db 016h,4,C:0h
db 016h,3,C:0h
db 016h,5,C
equ $-lcpout-2

pcldis34
lcpdis34

db 1bh,0
dw cldis34l
db 016h,3,0
db 016h,4,0
db 016h,6,0-0h
db 016h,5,80h
equ $-lcpdis34-2

;ADC
pcadcuser
lcpadcuser

db 1bh,0
dw adccuserl
db 013h,1,'
db 013h,2,'
db 013h,4,'
equ $-lcpadcuser-2

pcadcpen
lcpadcpen

db 1bh,0
dw adcpenl
db 013h,3,'
equ $-lcpadcpen-2

;disable and clear up field
;enable and write down field
;i.e. pen is up & can select down

;disable and clear down field
;enable and write up field
;i.e. pen is down & can select up

;enable mark, set1, set2, close

;display unit name for chan1 and 2

;unit name
;username, file name, file ext
;desselect ESCAPE
;enable NO
;enable YES
;comment field inactive

;disable YES
;disable NO
;desselect ESCAPE
;make comments input field

;user name
;unit name for channel 1
;unit name for channel 2

;display down arrow if pen down
;" " if pen up

;erase field 1 through 5
;of adc panel window

```

[illegible]


```

mov ah,0fh
int 10h
mov initmode,al
;get current video mode
;save it
;clear screen
;is DESQview running?
;no
;API level 2.25 required
;DESQview up and running ok
;display no DV error
;ow=5, col=43
;return to DOS
;HIDE WINDOW FRAME
lea si,phidwin
push dssi
pushw hidenwin+4
;function to check free space
;drive D
;256 clusters needed, assuming 1096 bytes/cluster
;free space ok, check installed display
;not enough free space
;report free space error
;exit to DOS
;function
;subfunction
;vga or mega?
;CHANGE VIDEO MODE TO 640 BY 480, 16 COLORS
;DISPLAY GRAPHICS TICKS
;frame and ticks
;video page 0
;column
;row
;draw dot by writing directly to memory
mov ax,colert
mov bx,0
mov cx,11
mov dx,68
call near ptr dot
add dx,1
cmp dx,294
jb dvert1
add cx,160
cmp cx,651
jb dvert0
ja dvertdone
mov cx,149
jmp dvert0
mov temp0,0
mov temp1,0
mov temp2,0

```

```

mov ds,68
mov cx,12
call near ptr dot
inc cx
inc temp2
cmp temp2,5
tick1:
jb tick1
add cx,155
cmp cx,630
tick2:
ja tick2
jmp temp2,0
cmp temp0,1
je tick3
mov cx,145
temp0,1
mov temp2,0
jmp tick1
mov temp0,0
temp2,0
cmp temp1,6
tickdone:
je tickdone
dx,37
temp1,1
tick4:
je tick4
inc dx
inc temp1
jmp tick0
;OPEN PANEL FILE
tickdone:
asend new,panel
apop parhan
apush pinpan
apush linpan
asend open,parhan
;DISPLAY ADC PANEL
apush padc
apushw ladc
apush chartwin
asend apply,parhan
apop inwinadc
apop inkeyadc
;CREATE CONVERT TASK
apush zero
apush zero
apush zero
apush zero
apush negone
apush negone
lea si,convert
push cs
push si
asend new>window
apop convwin
;CREATE KID TASK
apush pkidname
apush lkidname
apush zero
apush zero
apush twound
apush negone
apush negone
;no name for convert window
;no length to name of convert window
;zero rows for convert window
;zero columns for convert window
;zero byte input buffer
;default stack size (256 bytes)
;new task
;get address of convert cs:[si]
;pointer to name of kid window
;length of name of kid window
;zero rows for kid window
;zero columns for kid window
;200 byte input buffer
;default stack size (256 bytes)
;new task

```



```
leq si,kic
push cs
push si
push new,window
asend kidwin
;CREATE TIMER F
;HART
asend new,timer
;open chtimer
asend open chtimer

;get address of kid cs:[si]
;same as old point displayed?
jne mdel1
;no erase old point
;yes, test for left set point, chan 1
;DELETE OLD DATA POINT
mdel1: mov ax,colorb
mov bx,4
mov cl,olddata[si]
xor ch,ch
mov si,curnum
add cx,viddisp[si]
mov dx,chpos
inc dx
call near ptr dot
cmp bx,0
je msetd1
dec bx
add dx,2
jmp mdel2

;DISPLAY NEW DATA POINT
msetd1: mov ax,colorb
mov cl,tempvid
xor ch,ch
add cx,viddisp[si]
mov dx,chpos
inc dx
mov si,chan1
newdata[si],0
mov bl,tempvid
mov olddata[si],bl
mov bx,4
call near ptr dot
cmp bx,0
je mtest1
dec bx
add dx,2
jmp msetd2

;TEST IF LEFT SET POINT SELECTED
mtest1: shl si,1
mov di,adcf[si]
cmp byte ptr ds:[di+2],0 ;left set point flag clear?
jne mdel1
;no, change set point
;yes, test right set point flag
;TEST IF NEW LEFT SET POINT = OLD
mtds1: mov byte ptr ds:[di+2],0 ;clear left set point flag, chan 1
shr si,1
mov al,tempvid
cmp oldset[si],al
jne mdel1
;no, delete old set point
;yes, test right set point flag
;DELETE OLD LEFT SET POINT
mdel1: mov ax,colorb
mov bx,4
mov cl,oldset[si]
xor ch,ch
mov si,curnum
add cx,viddisp[si]
mov dx,chpos
call near ptr dot
cmp bx,0
je mset1
dec bx
add dx,2
jmp mdel2

;DISPLAY NEW LEFT SET POINT
mset1: mov di,chan1
mov al,tempvid

;get address of kid cs:[si]
;current user #, 0, 2, 4, 6, 8
;pointer to user(i) info space
;pointer to user(i) # chan owned
;user own any channels?
;yes
;chan1 data
;chan1 set points
;done 4 users yet?
;no, do next
;yes, test for write to disk
;initial row for chan1 pen lines
;updating chan1
;initialize chdi
;save it
;pointer to user(i) chan1 chan#, 1 to 8
;chan1 = no channel?
;no, test for new data
;done both channels yet?
;no, prep for chan2
;yes, do next user
;initial row for chan2 pen lines
;chan2 data
;chan2 set point
;=2, chan2
;prep for pointer to user(i) chan 2 chan #
;do chan2
;convert chan#, 0 to 7
;new data to display?
;yes
;no, test chan2
;word
;data point last collected
;=0, -v=127, in al
;save it
;byte
shr si,1
```

```
mov oldsetl(di),al      ;save new left set point
mov ax,colors           ;set point
mov bx,4               ;current video pen position
xor cl,tempvid         ;initial row for chan set lines
add cx,vldisp[s]
mov dx,chpos
call near ptr dot
cmp bx,0
je mtestsr1
dec bx
add dx,2
jmp msetsr2

;TEST IF RIGHT SET POINT SELECTED
mtestsr1: mov si,chan1
shl si,1
mov di,adeflags[s]
cmp byte ptr ds:[di+3],0 ;right set point flag clear?
jne midsr1
jmp mwidispa

;TEST IF NEW RIGHT SET POINT = OLD
midsr1: mov byte ptr ds:[di+3],0 ;clear right set point flag, chan 1
shr si,1
mov al,tempvid
cmp oldsetr(si),al
jne mdelr1
jmp mdelr2

;DELETE OLD RIGHT SET POINT
mdelr1: mov ax,colorb
mov bx,4
mov cl,oldsetr[s]
xor ch,ch
mov si,curnum
add cx,vldisp[s]
mov dx,chpos
call near ptr dot
cmp bx,0
je msetsr1
dec bx
add dx,2
jmp mdelr2

;DISPLAY NEW RIGHT SET POINT
msetsr1: mov al,tempvid
mov di,chan1
mov oldsetr(di),al
mov ax,colors
mov bx,4
mov cl,tempvid
xor ch,ch
add cx,vldisp[s]
mov dx,chpos
call near ptr dot
cmp bx,0
je mchancone
dec bx
add dx,2
jmp msetsr2
mchancone: jmp mwidispa

;WRITE DATA TO HARD DRIVE
mtestsr2: mov chan1,0
mov si,chan1
cmp writebank[s],0
jne mwrite
inc chan1
cmp

oldsetl(di),al      ;save new left set point
;set point
;current video pen position
;initial row for chan set lines
;done yet?
;yes, test right set point flag
;no
;do next
;byte, channel #
;word
;no, change it
;yes, test user(i) channel 2
;clear right set point flag, chan 1
;byte
;current video pen position
;same as old right set point?
;no, erase old
;yes, test user(i) channel 2
;background
;old right set point
;word, user #
;offset from left for user(i)
;initial row for chan pen lines
;done yet?
;yes
;no
;do next
;chan # 0 to 7, byte
;save new right set point
;set point
;current video pen position
;offset from left
;initial row for chan pen lines
;done yet
;yes
;no
;do next
;test user(i) channel 2
;channel #, 0 to 7
;write to disk?
;yes
;no
;done B yet?

mwrite: mov short mdispdata
mov writebank[s],0
mov ah,40h
shl si,1
mov bx,filehand[s]
mov cx,2048
shl si,1
mov dx,data[s]
mov di,pdata[s+2]
shr si,2
cmp databank[s],1
je mmbset
mov da,di
int 21h
jnc mwriteok
mov esdi,mwerr
mov bx,mwerr
mov cx,mwerr
mov da,word ptr chartwin+2
acall disperr
jmp mnextw
mwriteok: cmp cx,2048
jne msizerr
mov mnextw
mov bx,msizerr
mov cx,msizerr
mov da,word ptr chartwin+2
acall disperr
jmp mnextw
msizerr: mov esdi,msizerr
mov bx,msizerr
mov cx,msizerr
mov da,word ptr chartwin+2
acall disperr
jmp mnextw

mnextw: ;no, do next
;yes, check if old data requested
;reset writebank flag
;function to write to file
;0,2,4,6...
;file handle
;# bytes to write
;0,4,8,12...
;pointer to bank A
;pointer to bank B
;0,1,2,3...
;current bank = 8, i.e. write bank A7
;yes, write bank A, ds:[da]
;no, write bank B, ds:[dx]
;write data
;error in drive or handle?
;yes, report fatal error

mmbset: int 21h
jnc mwriteok
mov esdi,mwerr
mov bx,mwerr
mov cx,mwerr
mov da,word ptr chartwin+2
acall disperr
jmp mnextw
mwriteok: cmp cx,2048
jne msizerr
mov mnextw
mov bx,msizerr
mov cx,msizerr
mov da,word ptr chartwin+2
acall disperr
jmp mnextw
msizerr: mov esdi,msizerr
mov bx,msizerr
mov cx,msizerr
mov da,word ptr chartwin+2
acall disperr
jmp mnextw

;DISPLAY LAST 60 MIN OF DATA
;OR ERASE OLD LINE IF FILE IS CLOSED
mdispdata: cmp disptag,0
je mtestsr2
mov mtestsr2
mov di,adeflags[s]
cmp byte ptr ds:[di+4],1 ;close file(i)?
jne mnextw
mov mnextw
mov byte ptr ds:[di+5],0 ;channel un-owned?
je mnextw
add chan1,2
cmp chan1,16
jne mnextw
jmp mtestquit
;ERASE OLD DISPLAY DATA FOR CHANNEL JUST CLOSED
mnextw: mov ax,si
shr ax,1
inc al
mov si,0
mov di,user(s)
cmp byte ptr ds:[di+3],al ;user(i) chan1 = channel #?
je mdel1
cmp byte ptr ds:[di+4],al ;user(i) chan2 = channel #?
je mdel2
add si,2
cmp si,8
jne mnextw
jmp mnextw
mov byte ptr ds:[di+3],0 ;user(i) chan1 = none
mov tempda,49
;go erase current data line
short mdata
mov byte ptr ds:[di+4],0 ;user(i) chan2 = none
mnextw: mov
```



```

je vdispkip
mov ch,0
mov cl,al
add cx,vdispser
mov ax,colorb
call near ptr dot
mov ax,colorb
mov ch,0
mov cl,byte ptr ds:[si]
call near ptr dot
mov ax,vdispser
add add
mov al,byte ptr ds:[si]
mov al,byte ptr ds:[di],al
mov dx,68
vdispkip: cmp dx,68
je vdispdone
dec dx
dec di
cmp tempcx,225
jne vskipcx
mov tempcx,-1
sub si,226
vskipcx: inc tempcx
inc si
jmp vnextdot
vdispdone: ret

;HAS KID SIGNALLED QUIT WITH QUITSET=255
mtestquit: cmp quitset,0fffh
je chartend

;TEST IF CHART TIMER HAS EXPIRED YET
asend len,ctimer
apop chjunk
cmp word ptr chjunk,0
je chtimeck
call pause
jmp mainroute
ctimerck: asend read,ctimer
apop chjunk
jmp mainroute

;QUIT WAS SELECTED, DROP CHILD TASKS, EXIT TO DESVIEW
chartend: asend erase,ctimer
apush quitedelay
asend addto,ctimer
asend read,ctimer
apop chjunk
asend close,ctimer
asend free,ctimer
asend free,kidwin
asend free,corwin

;RESET WRITE MODE TO ZERO
mov dx,3cch
mov al,5
out dx,al
inc dx
mov al,0
out dx,al
;mode = 0
;set write mode to default
mov al,initcode
xor ah,ah
int 10h
exit 0

;***** END OF PARENT TASK "CHART" *****
;*****
;*****

```

```

;***** ROUTINE TO WRITE GRAPHICS DATA DIRECTLY TO VIDEO MEMORY *****
;*****
dot:
push dx
push cx
push bx
push ax
push ax
push cx
mov ax,dx
mov bx,80
mul bx
mov bx,ax
shr cx,3
add bx,cx
shl cx,3
pop ax
sub ax,cx
mov ax,ax
mov al,100000000b
shr al,cl
mov cl,al
mov dx,3cch
mov al,5
out dx,al
inc dx
mov al,2
out dx,al
mov ax,0a0000h
mov es,ax
dec dx
mov al,8
out dx,al
inc dx
mov al,cl
out dx,al
mov al,es:[bx]
pop ax
mov es:[bx],al
pop ax
pop bx
pop cx
pop dx
ret

SUBTTL << CONVERT >>
PAGE +
;***** START OF CHILD TASK "CONVERT" *****
;*****
convert: mov ax,adata
mov ds,ax
;HIDE CONVIN WINDOW FRAME
lea si,phidewin
apush dssi
apush hidewin+4
asend write,convwin
;OPEN CONVERT TIMER
asend new,timer
asend open,adtimer
asend quitset,0fffh
jne setdelay
jmp convquit

quit: program requested?
no, set timer delay and get data
yes, close timer and stop

```

```
serdelay: acall begin
           delay
           addto,adtimer
           mov si,channel
           mov di,adflags[si]
           cmp byte ptr ds:[di+5],0 ;channel owned?
           jne convyes
           convnext: add channel,2
                   cmp channel,16
                   je convpause
                   jmp convyes
           convpause: acall endc
                   add len,adtimer
                   add read,adtimer
                   jmp convyes
           convyes: word ptr last,61
                   testload
                   inc word ptr last+2
                   mov ax,word ptr junk+2
                   cmp ax,word ptr last+2
                   je overload
                   mov ax,word ptr junk
                   cmp ax,word ptr last
                   jne loadok
                   mov bx,262
                   mov cx,1
                   sound
                   loadok: mov ax,word ptr junk
                   mov word ptr last,ax
                   mov ax,word ptr junk+2
                   mov word ptr last+2,ax
                   jmp convset
           convset: mov dx,station
                   mov ax,channel
                   shr ax,1
                   out dx,al
                   dec dx
                   pushf
                   cli
                   out dx,al
                   inc cx
                   in al,dx
                   sti
                   popf
                   and and
                   jnz poletstat1
                   jmp staterr
           poletstat1: inc cx
                   cmp cx,doffh
                   jne poletstat2
                   jmp timeout
           poletstat2: in al,dx
                   and al,80h
                   poletstat1
                   dec dx
                   in al,dx
                   mov ah,al
                   mov tempah,ah
                   dec cx
                   in al,dx
                   shr al,4
                   shi al,4
           ;suspend multitasking until data collected
           ;restore ah, 8 MSB
           ;4 MSB in low nibble, zero fill high nibble
           ;channel # 0 to 14, word
           al,ah
           mov ah,tempah
           shr ah,4
           mov si,channel
           push ax
           xor ax,0ffffh
           mov vidpen[si],ax
           pop ax
           di,adflags[si]
           mov byte ptr ds:[di],0 ;pen() down?
           je prepbank
           ;yes
           ;no
           ;yes, prep to save, double word
           ;pointer to bank A
           ;pointer to bank B
           ;byte
           ;writing bank A7, =0
           ;yes, di ok
           ;no, set di to bank B
           ;word
           ;add current displacement
           word ptr ds:[di],ax ;save data point
           cmp dataoff[si],2046
           jne bankok
           ;no
           ;byte
           ;set write bank flag
           ;switch to new bank
           ;word
           ;reset displacement
           dataoff[si],doffh
           bankok: add dataoff[si],2
                   inc pen[si]
                   ;inc data count
                   ;make +v=0, -v=0ffff
                   ;double word
                   xor ax,0ffffh
                   shi si,1
                   cll
                   add ds[si],ax
                   mov nocarry
                   inc ds[si+2]
                   shr si,1
                   inc sumcount[si]
                   cmp sumcount[si],37
                   jne skipvid
                   mov sumcount[si],0
                   shi si,1
                   mov ax,ds[si]
                   mov bx,ds[si+2]
                   mov ds[si],0
                   mov ds[si+2],0
                   shr ah,2
                   shi bl,6
                   add ah,bl
                   shr si,1
                   mov di,pvid[si]
                   add di,videooff[si]
                   mov byte ptr ds:[di],ah
                   cmp videooff[si],225
                   jne videok
                   mov videooff[si],-1
                   inc videooff[si]
                   mov di,adflags[si]
                   inc di
                   cmp byte ptr ds:[di],0
                   je setnewdata
                   cmp markoff[si],60
                   je setnewdata
                   ;already marked 30 points?
                   ;yes, no more allowed
                   ;5 SECOND DELAY BEFORE MARKING POSITION AND SOUNDING TONE
                   ;8 LSB in al
                   ;restore ah, 8 MSB
                   ;4 MSB in low nibble, zero fill high nibble
                   ;channel # 0 to 14, word
                   al,ah
                   mov ah,tempah
                   shr ah,4
                   mov si,channel
                   push ax
                   xor ax,0ffffh
                   mov vidpen[si],ax
                   pop ax
                   di,adflags[si]
                   mov byte ptr ds:[di],0 ;pen() down?
                   je prepbank
                   ;yes
                   ;no
                   ;yes, prep to save, double word
                   ;pointer to bank A
                   ;pointer to bank B
                   ;byte
                   ;writing bank A7, =0
                   ;yes, di ok
                   ;no, set di to bank B
                   ;word
                   ;add current displacement
                   word ptr ds:[di],ax ;save data point
                   cmp dataoff[si],2046
                   jne bankok
                   ;no
                   ;byte
                   ;set write bank flag
                   ;switch to new bank
                   ;word
                   ;reset displacement
                   dataoff[si],doffh
                   bankok: add dataoff[si],2
                           inc pen[si]
                           ;inc data count
                           ;make +v=0, -v=0ffff
                           ;double word
                           xor ax,0ffffh
                           shi si,1
                           cll
                           add ds[si],ax
                           mov nocarry
                           inc ds[si+2]
                           shr si,1
                           inc sumcount[si]
                           cmp sumcount[si],37
                           jne skipvid
                           mov sumcount[si],0
                           shi si,1
                           mov ax,ds[si]
                           mov bx,ds[si+2]
                           mov ds[si],0
                           mov ds[si+2],0
                           shr ah,2
                           shi bl,6
                           add ah,bl
                           shr si,1
                           mov di,pvid[si]
                           add di,videooff[si]
                           mov byte ptr ds:[di],ah
                           cmp videooff[si],225
                           jne videok
                           mov videooff[si],-1
                           inc videooff[si]
                           mov di,adflags[si]
                           inc di
                           cmp byte ptr ds:[di],0
                           je setnewdata
                           cmp markoff[si],60
                           je setnewdata
                           ;already marked 30 points?
                           ;yes, no more allowed
                           ;5 SECOND DELAY BEFORE MARKING POSITION AND SOUNDING TONE
```


[illegible]


```
je suspend ;yes
dec ci ;no
inc si
jap sfoundl
mov scuruser,0
mov user,0
mov switch,0
mov snewunit,0
mov sokesc,1
cmp sump,0
je sumproute
set user
mov sroute,0
jap short sfreewin
mov sroutel ;free SETUP window then goto ROUTE

; THIS CODE SETS UP SFREELUNIT TO INDICATE UNITS SELECTED
; 0=UNIT FREE, 1=UNIT SELECTED BUT FILE NO OPEN YET, 2=UNIT OWNED & FILE OPEN
gounit:
xor ah,ah
sub al,2
mov si,ax
cmp byte ptr esi,di+3i,"*";field selected?
je setunit
mov sfreewin[si],0
dec snewunit ;dec # units selected
;TEST IF # UNIT SELECTED IS <= TO THE # ALLOWED
stestesc: mov bl,snewunit ;# selected
cmp bl,schanavail ;# allowed
ja stoosany ;selected > allowed
mov sokesc,1 ;selected <= allowed, ok to escape
jap inssetup ;set next user input
;TOO MANY NEW UNITS SELECTED, DO NOT PERMIT ESCAPE
stoosany: mov sokesc,1
jap inssetup
;UNIT SELECTED, MARK IT AS SUCH IN SFREELUNIT, =1
setunit: inc snewunit ;inc # units selected
stestesc: mov bl,snewunit ;test # units selected
jap stestesc

; THIS CODE FREES THE SETUP WINDOW AND KEYBOARD THEN JUMPS TO ROUTE OR OPENF
; BASED ON THE VALUE OF SRROUTE, 0 OR 1 RESPECTIVELY.
sfreewin: send free,inkey ;free keyboard
send free,inwin ;free window
cmp sroute,1 ;goroute?
jne goopenf ;no, goopenf
jap goroute ;goto ROUTE

SUBTTL << OPEN FILE >>
PAGE *
goopenf: push popenf
push popenf
push kidwin
send apcyl,parhan
apop inwin
apop inkey
mov loutput 10
mov soutput 1,2
mov di,7
mov al,testuser[di]
cmp al,32
ja obl
mov al,' '
mov soutput di],al
cmp di,0
je osetuse

;length of command string
;field #2
;al = character?
;yes
;no, store space
;done 8 yet?
;yes
```

```
dec di ;no
jap obuilduser ;do next
lea si,poutput ;window stream
cscsi ;length of window stream
mov temp1,1c
pushw temp1
send write,inwin ;write user name in field #2
;CLEAR STORAGE OF FILE NAME AND EXTENSION
initcltr: mov si,0
oclname: mov oname[si],""
inc si
cmp si,8
jb ocrlname
mov si,0
mov ocrlname
ocrlfext: mov ocext[si],""
inc si
cmp si,3
jb ocrlfext
mov text,0
mov di,12
mov si,0
mov cx,0
mov framef,0
otestunit: mov al,sfreewin[si]
cmp al,1
je oselectu
cmp cx,7
jz all 87
jz onreset
add di,12
inc ci
inc si
jap otestunit
onreset: jmp osetadc
oselctuc: add di,11
mov si,11
mov temp0,cx
obuildunit: mov al,unit[di]
mov soutuser[si],al
cmp si,0
je osetunit
dec di
dec si
jap obuildunit
lea si,poutuser
cscsi
outuserl+4
pushw write,inwin
send write,0
mov lread,0
;USER INPUT OF FILE NAME AND EXT. IF FILE NAME NOT ENTERED AND ESCAPE THEN FREE UNIT
;IF ESCAPE AND FILE NAME ENTERED THEN MAKE FILE PATH-NAME AND OPEN
ogetname: send read,inkey
apop linbuf
apop pinbuf
apop esdi,pinbuf
mov al,es[di]
cmp al,4
je ofileext
jb ofilename
cmp framef,0
je oncname
jap onakefile
mov si,tempu
mov sfreewin[si],0
initcltr
jap
cmp cx,2
ocfilext: mov
```

```
add di,3 ;pointer to extension text
mov si,0 ;pointer to extension storage
mov text,3 ;default length of extension
add bread,6 ;increase length of string read
mov al,es:[di]
cmp al,32 ;was it a space?
je oextspace ;yes
mov oext[si],al ;no, save it
cmp cx,0 ;done 3 chars yet?
je oextdone ;yes
inc cx ;no
inc di
inc si
jmp oextnext

oextspace: dec cx
oextdone: je oextdone
inc di
inc si
jmp oextnext

oextnext: short otestmore
short otestmore
cx,7
di,3
add si,0
mov framef,1
mov name,8
mov bread,11
add al,es:[di]
cmp al,32 ;was it a space?
je oextspace ;yes
mov oext[si],al ;done 3 chars yet?
cmp cx,0 ;yes
je otestmore ;no
inc di
inc si
inc si
jmp oextnext

oextnext: jmp oextnext
;SEE IF ANOTHER FIELD WAS MODIFIED, IF SO THEN EITHER ESCAPE, OR EXT, OR EXT AND ESCAPE
otestmore: mov ax,word ptr linbuf
cmp ax,bread ;total length of input string
ja oore ;total > amount read?
jmp otestunit ;yes, get next field input
;nothing more to read and escape not yet entered

;SOME OTHER FIELD ALSO MODIFIED
oore: inc di
mov al,es:[di] ;pointer to next user input
cmp al,6 ;get next field modified
jne otestspace ;was it escape?
jmp otestspace ;no, file name or extension
;short makefile
al,4
cmp al,4 ;was it file extension?
jne oore ;yes, get file extension
;FATAL ERROR READING USER INPUT, THIS SHOULD NEVER HAPPEN
ooreerr: mov esdi,pkwoerrr
mov bx,looreerr
mov cx,wooreerr
jmp ooreerr

;BUILD FILE NAME
;FIRST GET USER #
makefile: cmp lname,c
```



```
;PREP TO RE-WRITE FILE EXTENSION TO SCREEN
odname:  mov cx,2
         mov di,22
         ;pointer to pdis34 file's ext string
         mov si,0
         mov al,odext[si]
         mov pdis34[di],al
         cmp cx,0
         je odoneext
         dec cx
         inc di
         inc si
         jmp odnext
;DISPLAY FILE NAME AND EXT, DISABLE NAME AND EXT FIELDS, MOVE POINTER TO COMMENT FIELD
odoneext: lea si,pdis34
          push dssi
          pushw di341+4
          ascnd write,imwin
;GET COMMENTS OR ESCAPE IF NONE
ocomint:  ascnd read,inkey
          apop linbuf
          apop pinbuf
          mov esdi,pinbuf
          mov al,es[di]
          cmp al,5
          je ogetcom
          jmp short overwriteinit
;TRANSFER COMMENTS TO WFILE(1) SPACE
ogetcom:  mov cx,151
          mov si,temp0
          add di,3
          mov al,es[di]
          mov ds:[si],al
          cmp cx,0
          je ocominput
          dec cx
          inc di
          inc si
          jmp onextcom
;WRITE HEADER INFO TO FILE
overwrite: mov ah,40h
          mov bx,tempand
          mov cx,412
          mov di,tempdi
          shl di,1
          mov dx,nfile[di]
          int 21h
          jc owiniterr
          mov ax,412
          mov owinittok
          jmp odiskfu1
          mov di,ouni:di
          mov sfreewunit[di],2
          inc foflag
          lea si,pen34
          push dssi
          pushw en341+4
          ascnd write,imwin
          mov si,tempdi
          shl si,1
          mov di,odcf ags[si]
          mov al,scuruser
          mov byte ptr ds:[di+5],al
          ;set user# flag
          jmp ointict
          ;go see if another file need opening
; THIS CODE IS USED TO REPORT THAT AN ERROR OCCURRED WHILE WRITING FILE HEADER INFO
; THIS IS SET UP TO BE UNRECOVERABLE
```

```
         esdi,pwiniterr
         mov bx,lwiniterr
         mov cx,winiterr
         mov dx,word ptr kidwin+2
         dispersor
         mov ah,5eh
         mov bx,tempand
         int 21h
         mov ah,41h
         lea dx,testpath
         int 21h
         overresc: mov cx,7
                   mov di,0
                   mov al,sfreewunit[di]
                   cmp al,1
                   je overrfree
                   mov cx,0
                   je overrdone
                   dec cx
                   inc di
                   jmp overrunit
;UPDATE USER INFO IF UNIT MARKED AS SELECTED AND USER INFO ALREADY CHANGED
overrfree: mov ax,di
           inc al
           mov si,cuserdi
           cmp user[si+3],al
           jne overrdone1
           mov user[si+3],0
           mov user[si+4],al
           jne overrdone2
           mov user[si+4],0
           ;FREE UNIT IN SFREEWUNIT SPACE
           overrdone2: mov sfreewunit[di],0
                     jmp overrnext
           overrdone:  jmp ofreewin
; THIS CODE IS USED TO REPORT THAT LESS THAN 352 BYTES WERE WRITTEN WHILE WRITING
; FILE HEADER INFO. SUGGESTION IS TO CHECK DISK SPACE AND RETRY. IT IS SET UP TO
; BE UNRECOVERABLE.
odiskfull: mov esdi,pfulldisk
           mov bx,lfulldisk
           mov cx,wfulldisk
           jmp oabandon
; THIS CODE IS USED TO REPORT THAT THE FILE COULD NOT BE ASSIGNED TO A UNIT.
; THIS SHOULD NEVER HAPPEN AND IS THEREFORE A FATAL ERROR
oerrorass: mov esdi,passerr
           mov bx,lasserr
           mov cx,wasserr
           jmp oabandon
;abandon units selected and close/delete files
; THIS CODE IS USED TO REPORT THAT THE FILE ALREADY EXISTS. ENTERING *
; WILL OVERWRITE THE OLD FILE, ENTERING A NEW FILE NAME WILL ATTEMPT AGAIN.
ofilefound: mov esdi,poverwrite
            mov dx,word ptr kidwin+2
            mov bx,loverwrite
            mov cx,moverwrite
            acall dispersor
            mov pcurnmove+5,3
            mov pcurnmove+8,3
            mov pcurnmove+18,3
            lea si,pcurnmove
            apushw curmove+4
            ascnd write,imwin
            mov pcurnmove+5,1
            ;move cursor
            ;reset to field #1, username in SETUP
```

```
mov pcumove+8,1
mov pcumove+16,1
asend read,inkey
apop linbuf
apop pinbuf
amov esdi,pinbuf
mov al,es:[di]
cmp al,3
je ocheckedl
jmp ofilefound
add di,3
mov al,es:[di]
cmp al,'*'
je oyesdel
jmp ofilefound
mov ah,4h
lea dx,testfile
int 21h
jc odelerr
jmp otestit
amov esdi,pdsierror
mov dx,word ptr kidwin+2
mov bx,ldelerror
mov cx,wdelerror
acall disper
jmp osetunit

; THIS CODE IS USED TO REPORT AN ERROR DURING FILE CREATION AND OPENING
obadopen: cmp ax,4
je oallbuf
ja ofulldir
amov esdi,poferror
mov dx,word ptr kidwin+2
mov bx,loferror
mov cx,woferror
acall disper
jmp osetunit
amov esdi,pallbuf
mov dx,word ptr kidwin+2
mov bx,lallbuf
mov cx,wllbuf
acall disper
jmp ofreewin

; THE ALLBUF ERROR IS UNCORRECTABLE WITHOUT CLOSING SOME FILE OR CHANGING THE
; NUMBER OF FILES IN THE CONFIG.SYS FILE.
ofulldir: amov esdi,pfulldir
mov dx,word ptr kidwin+2
mov bx,fulldir
mov cx,wfulldir
acall disper
jmp ofreewin

; THE DIRFULL ERROR IS UNCORRECTABLE WITHOUT DELETING SOME FILE
; UPDATE ADC WINDOW
osetadc: mov al,scuruser
mov ah,1
cmp al,1
je oadcsuset
add ah,5
dec al
jmp oadcsux
mov padcuser 5j,ah
inc ah
mov padcuser 15j,ah
inc ah
inc ah

padcuser[29],ah
mov si,cuserdi
add si,5
mov di,6
mov ci,7
mov al,user[si]
mov padcuser[di],al
cmp ci,0
je oadcsusend
dec ci
inc si
inc di
jmp oadcsnext
mov dx,0
mov di,16
mov si,cuserdi
mov al,user[si+3]
xor ah,ah
mov bl,12
mul bl
mov si,ax
mov cl,11
mov al,unit[si]
mov padcuser[di],al
cmp ci,0
je oadcdone
dec ci
inc si
inc di
jmp oadclnext
cmp dx,1
je oadcsnow
mov dx,1
mov si,cuserdi
inc si
mov di,30
mov oadc2
lea si,padcuser
apush dssi
apushw adcsuserl+4
asend write,inwinadc

;free window and keyboard before returning
ofreewin: asend frce,inkey
asend free,inwin
;ALL SELECTED UNIT MUST OWNED OR FREE AT THIS POINT
mov di,7
ofreeunit: cmp sfreecunit[di],1
jne ofreemore
mov sfreecunit[di],0
ofreemore: cmp di,7
je ofreedone
dec di
jmp ofreeunit
ofreedone: jmp goroute
;return to ROUTE

SUBTTL << COLLECT >>
PAGE +
gocolec: apush pcolec
apushw lcolec
apush kidwin
asend apply,parban
apop inwin
apop inkey
;CLEAR CLOSE FILE FLAG
mov cc,close,0
```



```
;CLOSE CHANNEL REQUESTED, FREE COLLECT WINDOW, ROUTE TO CLOSE FILE
cc.closechan: @send free,inkey ;free collect keyboard
@send free,inkey ;free collect window
jmp short gotoeof ;route to CLOSE FILE

;UPDATE A/D WINDOW TO REFLECT PER UP OR DOWN
@move: mov al,scuser
mov ah,3
ad.pennext: cmp al,1
je ad.pen1
add ah,5
dec al
jmp .ad.pennext
.ad.pennext: @mp0,si
mov temp1,di
mov temp2,ax
mov padopen15,ah
mov al,clupf
mov padopen16,al
mov di,cluserdi
mov ax,si
inc al
cmp user[di-3],al
je ad.penok
add padopen15,2
lea si,padopen
@push ddsi
@pushw ad.pen1-4
@send write,inkeyadc
mov cx,temp2
mov di,temp1
mov si,temp0
ret

SUBTITLE << CLOSE FILE >>
PAUSE +
gotoeof: @push pclosetf
@pushw kiowin
@send apply,parhan
@pop inwin
@pop inkey
mov al,cchan1
xor ah,ah
mov tempchan,ax
cl.test: cmp cclose,1
je cclose1
cmp cclose,3
je cclose1
cmp cclose,2
jne clerror1
mov al,cchan2
xor ah,ah
mov tempchan,ax
jmp short cclose1
;FATAL ERROR, COULD NOT DETERMINE WHICH FILE TO CLOSE
cl.error1: @mov esdi,pc:termone
mov dx,word ptr kiowin-2
mov bx,cl:termone
mov cx,wc:termone
@call disperr
jmp cl:free n
;CLOSE FILE FOR UNIT SPECIFIED IN CCHAN1 IF JE CLOSE1 OR CCHAN2 IF JMP CLOSE1
;FIRST CLEAR UNIT NAME AND FILE NAME IN OUTPUT STRING
cl.set: mov di,6
```

```
cl:run: mov pclosetf[di],''
inc di
cmp di,18
jne cl:run
add di,2
cl:trfn: mov pclosetf[di],''
inc di
cmp di,40
jne cl:trfn
;GET NAME OF UNIT TO CLOSE
mov ax,tempchan
mov di,ax
shl di,1
mov si,tf:file[di]
add si,30
mov di,6
cl:unitnext: mov al,ds:[si]
mov pclosetf[di],al
cmp di,17
je cl:unitdone
inc si
inc di
jmp cl:unitnext
;GET USER NAME, FILE NAME, AND FILE EXTENSION
cl:unitdone: sub si,33
add di,3
mov al,ds:[si]
cmp al,0
je cl:unitdone
mov pclosetf[di],al
inc si
inc di
jmp cl:unitnext
;DISPLAY UNIT NAME, AND FILE NAME AND EXTENSION
cl:unitdone: lea si,pclosetf
@push ddsi
@pushw clout+4
@send write,inwin
mov ckey[0],0
mov ckey[1],0
;GET USER INPUT, YES OR NO TO CLOSE FILE, OR ESCAPE ALSO = NO
cl:input: @send read,inkey
@pop linbuf
@pop pinbuf
@mov esdi,pinbuf
cmp byte ptr es:[di],6 ;was it escape
jne clyorn ;no, yes or no entered
jmp short clesc ;yes, test routing
mov al,es:[di] ;get field modified
sub al,3
xor ah,ah
mov si,ax
mov al,es:[di-3]
mov ckey[si],al
jmp cl:input
cl:esc: mov al,ckey[0]
add al,ckey[1]
cmp al,178
je cl:input
cmp ckey[0],""
je cclose
jmp cl:free n
;UPDATE ADFC(1)
cl:close: mov si,tempchan
shl si,1
mov di,adcf[si]
```

;adjust pointer for retrieving dwords

;end of name = ASCII zero byte

;clear input, field YES
;clear input, field NO
;GET USER INPUT, YES OR NO TO CLOSE FILE, OR ESCAPE ALSO = NO

;pointer to ckey YES or NO field
;get Y or N
;save it
;get next key
;get Yes flag, Y, N, or zero if not chosen
;get No flag, Y, N, or zero if not chosen
;both Yes and No selected? Y+Y=178
;yes, invalid entry, try again
;Yes field selected?
;Yes, go close file
;user must have selected No or pressed ESCAPE only

;channel #, 9 to 7

```
;UPDATE # CHANNELS OWNED
mov     byte ptr ds:[di+5],0
mov     ax,cuserdi
dec     byte ptr user[di+2]
;DISABLE YES AND NO FIELDS, ENABLE COMMENTS
lea     si,pc1dis34
pushw   ds:[si]
pushw   cdis34i+4
asend   write,inwin
;FLUSH THE REMAINING DATA COLLECTED TO DISK
mov     ax,tempchan
mov     si,ax
shl     si,1
mov     ah,40h
mov     bx,filehand[si]
tempchand,bx
shl     si,1
mov     dx,pdata[si]
mov     di,pdata[si+2]
shr     si,2
mov     writebank[si],0
cptestbank
xor     databank[si],i
mov     writebank[si],0
shl     si,1
mov     dataoff[si],2048
shr     si,1
cptestbank: mov     databank[si],0
je     ctestbank
mov     dx,di
cubset:  shl     si,1
mov     cx,dataoff[si]
int     21h
cletbankab
mov     esdi,pc1usherr
mov     dx,word ptr kidwin2
mov     bx,icflusherr
mov     cx,wclusherr
call     disperso
shl     si,1
mov     di,pdata[si]
mov     cx,4095
mov     byte ptr ds:[di],0
cdata:  cmp     cx,0
je     csetsetvi
dec     cx
inc     di
cdata:  jmp     cdata
shl     si,1
mov     di,pvid[si]
mov     cx,225
mov     byte ptr ds:[di],0
cpvid:  cmp     cx,0
je     cpviddone
dec     cx
inc     di
cpvid:  jmp     cpvid
shl     si,1
mov     dsun[si],0
mov     dsun[si+2],0
shr     si,1
mov     pent[si],0
mov     dataoff[si],0
mov     videoff[si],0
mov     sunent[si],0
;CLEAR COMMENTS ENTERED FLAG FOR LATER
mov     si,1
mov     databank[si],0
mov     writebank[si],0
mov     newdata[si],0
;GET COMMENTS OR ESCAPE ENTRY
mov     ccomflag,0
cclinin: read,inkey
pop     linbuf
pop     pinbuf
mov     esdi,pinbuf
cmp     byte ptr es:[di],6
jne     cclincom
short cclinodone
mov     ax,tempchan
mov     si,ax
shl     si,1
mov     ax,nfile[si]
mov     si,ax
add     si,201
mov     cl,151
add     di,3
cclincom: mov     al,es:[di]
mov     ds:[si],al
cmp     cl,0
je     cclincomset
dec     cl
inc     di
inc     si
cclincomset: jmp     cclincom
mov     ccomflag,1
;PREPARE TO WRITE FINAL COMMENTS TO DISK, SKIP IF NULL ENTERED
cclinodone: cmp     ccomflag,0
je     cclincomok
mov     al,0
mov     ah,42h
mov     bx,tempchand
mov     cx,0
mov     dx,201
int     21h
jnc     cmoveok
jmp     cmoveerr
;WRITE COMMENTS TO DISK
cmoveok: mov     dx,si
sub     dx,151
mov     ah,40h
mov     bx,tempchand
mov     cx,152
int     21h
jc     clerrcom
cmp     cx,152
jne     clerrcom
jmp     short cmoveok
clerrcom: jmp     cclincomerr
;PREPARE TO WRITE MARKED DATA POINTS TO DISK
cmoveok: mov     bx,tempchand
mov     al,0
mov     ah,42h
mov     cx,0
mov     dx,353
int     21h
jnc     cmoveokmark
jmp     cmoveerr
;WRITE MARKED DATA POINTS TO DISK
cmoveokmark: mov     ax,tempchan
mov     si,ax
;byte
;clear comments entered flag for later
;Escape entered, i.e. no comments
;no, comments entered
;yes, no comments or comments already done
;setup to store comments in nfile(i)
;adjust pointer for retrieving duords
;final comments string space
;beginning of input string
;char(i) of comments
;store it
;done yet?
;yes
;no
;do next
;set comments flag to yes
;get new comments or escape to flag comments done
;write comments to disk7, 0=no, 1=yes
;no, none entered
;beginning of file code
;function to set file pointer
;offset=DX:DX
;final comments space
;move file pointer
;moved w/o error
;error in moving pointer
;pointer to end comments
;pointer to beginning of comments
;function to write data
;# bytes to write
;write comments
;error in writing final comments
;wrote 152 bytes?
;no
;yes, comments written w/o error
;error in writing final comments
;beginning of file code
;function to set file pointer
;offset = DX:DX
;marked data point space
;move file pointer
;moved w/o error
;error in moving pointer
```

```
shl si,1
mov dx,ptr kidiin+2
mov ah,40h
mov bx,temp0
mov cx,60
int 21h
jc cmarkbad
cmp ax,60
jne cmarkbad
jmp short cmarkok
cmarkbad: jmp cmarkerr
;CLOSE FILE
cmarkok: mov ah,3eh
mov bx,temp0
int 21h
jmp cmarkerr1
;CLEAN UP ALLOCATION
;FIRST RESET SFREENIT
closeok: mov ax,tempchan
mov dx,ax
mov sfreenit[di],0
;CLEAR NFILE(1) HEADER INFO
shl di,1
mov ax,nfile[di]
add ax,8
mov si,ax
mov cx,404
byte ptr ds:[si],0
clearfile: cmp cx,0
je cmarkflag
dec cx
inc si
jmp clearfile
;DECREASE FILE OPEN FLAG
clearflag: dec foflag
;CLEAR FILE HANDLE
mov filehand[di],0
;CLEAR DISABLE UNIT FOR SETUP
mov ax,tempchan
mov di,0
mov ax,0
cmarkerr: je cmarkerrable
dec ax
add di,3
jmp cmarkerrnext
cmarkerrable: mov ienable[di+2],0
;CHECK IF CHANNEL 2 SHOULD BE CLOSED
cmp cclose,3
je csetc12
jmp cmarkerrin
csetc12: dec cclose
jmp cmarkerr
;REPORT FATAL ERROR IN MOVING FILE POINTER TO WRITE FINAL COMMENTS
cmarkerr: mov esdi,pcmarkerr
mov dx,word ptr kidiin+2
mov temp0,b:
mov bx,temp0
mov cx,temp0
call disperr
jmp cmarkok
;restore file handle
;go try and close file
;REPORT FATAL ERROR IN MOVING FILE POINTER TO WRITE MARKED DATA POINTS
;FATAL ERROR IN MOVING FILE, TRY IT AGAIN
cmarkerr1: mov ah,3eh
mov bx,temp0
int 21h
jc cmarkerr2
jmp closeok
;REPORT ERROR IN CLOSING FILE
cmarkerr2: mov esdi,pcmarkerr
mov dx,word ptr kidiin+2
mov bx,temp0
mov cx,temp0
call disperr
jmp closeok
;try closing other file
;UPDATE A/D WINDOW TO REFLECT CURRENT USE
cmarkerr3: mov cclose,0
mov ccloseflag,0
mov ci,4
mov di,5
mov si,cuserdi
cmp userisi+21,0
je cmarkerrall
;USER STILL HAS ONE CHANNEL, DISPLAY "NO CHANNEL" FOR ONE CSAN
jmp osetade
;no, display units owned in ininade
;routine is at the end of OPEN FILE
;USER DOES NOT OWN ANY CHANNELS, ERASE ALL FIELDS FOR USER(1)
cmarkerrall: mov al,scuser
mov ah,1
cmarkerrnext: cmp al,1
je cmarkerrset
dec al
add ah,5
jmp cmarkerrnext
;test next
;set field #
;done setting all 5 fields?
yes
no
;next field #
;pointer to next field command string
;set next
cmarkerrdone: mov padc1[di],ah
cmp ci,0
je cmarkerrdone
dec ci
inc ah
add di,2
jmp cmarkerrset
cmarkerrdone: lea si,padc1r
```


END

DATE FILMED

01 / 30 / 91