A MOS PARAMETER EXTRACTION PROGRAM FOR THE BSIM MODEL

APPENDIX 5: BSIM1.0 PASCAL SOURCE CODE

by

J. R. Pierret

Memorandum No. UCB/ERL M84/100

21 November 1984

A MOS PARAMETER EXTRACTION PROGRAM FOR THE BSIM MODEL

APPENDIX 5: BSIM1.0 PASCAL SOURCE CODE

by

Joseph R. Pierret

Memorandum No. UCB/ERL M84/100

21 November 1984

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Research sponsored by the Defense Advanced Research Projects Agency
Contract No. N00039-84-C-0107.

bsim1.0.TEXT

```
$REF 325$
$UCSD$
program bsim1_0(input,output);                                        bsim1_0

{author: JOE PIERRET}
{this program is a modified version of the original csim3.2 by Brian Scott
 Messenger, January 1984.  before this version was finished, a copy of it
 was given to Tony Fung, who is adding the subthreshold parameter extraction.
 changes made to the program from that date forward are noted by capital JRP
 surrounded by brackets in the left most columns.  the small jrp in the left
 columns are noting changes made prior to giving tony a copy to work with}

import hpib_0;
import hpib_1;
import hpib_2;
import hpib_3;
import general_0;
import general_1;
import general_2;
import iodeclarations;
import iocomasm;
import dgl_lib;

{GLOBAL VARIABLES}
{last change made: NOV 14, 1984   By: JRP}

type linestring=string[80];

const   numbvb=6;
        numbvg=5;
        numbvd=4;
        number_small_vds_values=2;
        number_large_vds_values=2;
        betazeroparam=6;  {line that parameter betazero is on, in process file}
{THIS CONSTANT SHOULD BE MODIFIED WHEN THE NUMBER OF PARAMETERS IS CHANGED}
        const numbbsim=17;  {number of bsim parameters excluding vdd,tox,temp}

 {jrp}{variable continue_development has been changed to
        number_good_devices_error}

var   device,enhancement_depletion:integer;
      meas_20f:boolean;
      smud,smug,smus,smub:integer;
      mode:char;
      process,wafer,lot,date,operator:string[80];
      output_file,prober_file:string[15];
      vdd,temp,tox,cox,width,length:real;
      new_die,end_of_die,initial_jump,end_of_wafer,new_wafer:boolean;  {******
                                                          *******taken out}
      end_of_program:boolean;
      time_count:integer;
      {PROBER ORIENTED PARAMETERS}
      step_array:array[1..20,1..20] of char;   {this holds the automatic array}
      number_devices,present_device,number_die,present_die:integer;
      present_diex,present_diey:integer;  {x y die location}
      xdie_size,ydie_size,origin_diex,origin_diey:integer;
      sd,sg,ss,sb:array[1..40] of integer; {smus connected to various terminals}
      dt,ed,mx,my:array[1..40] of integer; {device types and internal die steps}
      w,l:array[1..40] of real;  {width and length arrays}
      meas_two_phif:array[1..40] of boolean;  {program decides to measure 20f}
      {BSIM PARAMETERS}
      c1_vfb,c2_phif2,c3_k1,c4_k2,c5_eta,c6_beta0,c7_u0,c8_u1,c9_x2beta0,
      c10_x2eta,c11_x3eta,c12_x2u0,c13_x2u1,c14_beta0sat,c15_x2beta0sat,
      c16_x3beta0sat,c17_x3u1,c9_x2mu0,c14_mu0sat,c15_x2mu0sat,c16_x3mu0sat:real;
```

```
    {PROCESS FILES AND PARAMETERS}
    {these parameters are the latest variations in k1 with length and width}
    nelk1,ndlk1,nzlk1,pelk1,pdlk1,pzlk1:real;
    newk1,ndwk1,nzwk1,pewk1,pdwk1,pzwk1:real;
    {these parameters are DeltaL and DeltaW for the previous die}
    ned1,ndd1,nzd1,ped1,pdd1,pzd1:real;
    nedw,nddw,nzdw,pedw,pddw,pzdw:real;
    {these parameters are the latest values of 20f for various devices}
    nephif2,ndphif2,nzphif2,pephif2,pdphif2,pzphif2:real;
    {these are files to store the parameters in}
    nchanenh,pchanenh,nchandep,pchandep,nchanzer,pchanzer,
    temporary,userout,userout2:text;
    {these are the number of good devices of a type on a die}
    numbne,numbpe,numbnd,numbpd,numbnz,numbpz:integer;
    {MEASUREMENT VALUES}
    ids:array[1..4,1..6,1..5] of real;
    vgs:array[1..4,1..6,1..5] of real;
    vds:array[1..4] of real;
    vbs:array[1..6] of real;
    vdindex,vgindex,vbindex:integer;
    {LEAST SQUARES ALGORITHMS}
    matrix:array[1..3,1..4] of real;
    parr:array[1..4] of real;
    initxval,inityval,initval1,initval2:real;
    {NEW VARIABLES ADDED TO SUMMER 1984 PROGRAM}
    hpib_analyzer_address:integer; {e.g. hpib analyzer might be HP4145A}
    hpib_prober_address:integer; {e.g. hpib prober might be ELECTROGLAS 2001X}
    continue:char; {move probes to new location, then enter CR to continue}
                   {also user inputs a 'y' or 'n' or CR to tell what to do
                    in prepare_for_iv_graphics procedure}
    colon_position:integer; {colon position in output_file name}
    period_position:integer; {period position in '.TEXT' in output_file name}
    first_time_thru_program:boolean; {allows 'graceful' ending of program
                    if user decides to quit before actually enterring program}
    all_die_are_bad:boolean; {= f when 1st good device on 1st good die is found}
    all_devices_are_bad:boolean; {= f when 1st good device on each die is found}
    first_good_die:integer; {= 1 when first good die on wafer is found.
                             this may or may not be the first die}
    first_good_device:integer; {= 1 when first good device is found on each
                                die.}

    fake_meas_20f:boolean;  {set true when 1st good device on a die is found.
                             this is needed in case 1st n devices on the die
                             are bad}
    graphics_selection:char;  {holds y or n which determines whether to enter
                               BSIM PARAMETER vs W or L graphics mode after
                               every die is completed}
    po_graphics_wanted:boolean;  {= true if graphics_selection=yes. if = true
                                  then the BSIM PARAMETER vs W or L mode will
                                  be enterred after every die}
    leastsq_divide_by_zero:boolean:  {in the past, if a divide by zero error was
                                      encountered while running the program, the
                                      pascal system would abort the program...
                                      possibly after an hour or so of computing.
                                      using this flag, if a divide by zero occurs,
                                      the data will not be saved and the program
                                      will move on to the next device}
    maxidsstring:linestring; {max of idsgzero or idsgvdd sent to analyzer}
{jrp}extract_error:integer;  {set=2 in saturation_region_data_extraction if
                              leastsq3 procedure is going to be called with
                              maxindex-initindex=2...leastsq3 doesn't work
                              for only 2 sets of data}
```

```
      measure_error:integer; {error flag for any errors encounterred while
                              measuring the device before extraction of
                              parameters}
{jrp}process_error:integer;  {error flag for any errors encountered while
                              developing the process files}
     processpar:array[1..6,1..17] of real;  {this holds the process data
                                             and was made global in this
                                             version}



{DEBUGGING PARAMETERS}
     iii,jjj,kkk:integer;  {used to read in all data values}
     idsstring:array[1..4,1..6,1..5] of string[15];
     vgsstring:array[1..4,1..6,1..5] of string[15];
     tempfile:text;  {to store ids array in once 120 values have been reduced}
     TRACE21,TRACE22,TRACE23,TRACE24,TRACE25,TRACE26,TRACE27,TRACE28,lsq3:boolean;
     TRACE29,TRACE29b,TRACE30,TRACE31,TRACE32,TRACE33,TRACE34,TRACE35:boolean;
{jrp}TRACE36,TRACE37,TRACE38,TRACE39:boolean;
{JRP}TRACE40,TRACE41,TRACEVTH:boolean;
     line_in_output_file:linestring;  {for writing output file to printer}
     {TRACE21--entering and leaving procedures, etc.}
     {TRACE22--      "       "      "      if-thens, etc.}
     {TRACE23--junk stuff}
     {TRACE24--for newton raphson iteration in saturation region}
     {TRACE25--only for entering-leaving some saturation procedures}
     {TRACE26--only for entering-leaving linear procedures}
     {TRACE27--only for debugging lsq3 divide-by-zero in sat_reg_data_extrac}
     {TRACE28--only for reading in ids array from outside files}
     {TRACE29--only for creating ids array to be used later}
     {TRACE29b-only to print out ids array which was created for TRACE29=true
               to the printer...just to verify values}
     {TRACE30--only in l_and_w_dependencies...will cause values
               paramvalue[i] and goodparamvalue[i] used in least
               square3 to be printed out on printer.  only u1,x2u1,and x3u1
               values are printed.}
     {TRACE31--to find divide by zero after parameters have been extracted
               probably in process_file_development}
     {TRACE32--only in l_and_w_dependencies to find out if leastsq procedures
               is finding correct values}
     {TRACE33--if t, then set lsq3 true for l_and_w dependencies}
     {TRACE34--if t, print out 17 calculated bsim params on the screen while
               plotting iv-graphics}
     {TRACE35--if t, write on printer the contents of the output file at 3
               places:  1)when it is created  2)before prepare_for_po_graphics
               and 3)after prepare_for_po_graphics}
{jrp}{TRACE36-only used in graphics to find out what is happening to vdterm,
               vgterm, etc. when user is asked for NEW SMU CONNECTIONS}
{jrp}{TRACE37-only used in linear region extraction procedures to detect
               real math UNDERflow}
{jrp}{TRACE38-only in measure_device_data to print out error code}
{jrp}{TRACE39-to find out why graphics mode was not entered in single mode}
{JRP}{TRACE40-to allow entire program to be skipped up to iv_graphics
               procedure.  NEDFILE.TEXT and bsimout.TEXT are read in from
               the program, and hence must exist first}
{JRP}{TRACE41-in prepare_for_iv_graphics, to see whats happening for single
               mode}
{JRP}{TRACEVTH-only in function bsimsim, used to output Vth at 4 different
               conditions.  used just to see what calculated Vth is!}
```

..bsim1_0

```
{*********************************************************************
**********************STRING CONVERSION ROUTINES*********************
*********************************************************************}
```

realtostr

```
function realtostr:linestring;
{this function will read a real number string, reading only legitimate
characters from the user}

var   buffer:string[80];
      value:char;
      i,j,max:integer;
      number:real;

begin
     realtostr:='0.0';
     i:=1;
     j:=1;
     readln(buffer);
     max:=strlen(buffer);
     if max <> 0 then
          realtostr:='                                                    ';
          for j:=1 to max do
               begin
                    value:=buffer[j];
                    if ((value>='0') and  (value<='9')) or (value='.')
                    or (value='E') or (value='e')or (value='+') or (value='-')then
                              begin
                                   realtostr[i]:=value;
                                   i:=i+1;
                              end;
           end;
end;
```

digit

```
function digit(number:char):integer;
{this function returns an integer given a character}
{last change made: JUNE 18, 1984   by: JRP}

begin
     if (number>'9') or (number<'0') then
          begin
               gotoxy(0,22);
               writeln('ERROR in function(digit).  Erroneous number = ',number);
          end;
     digit:=ord(number)-ord('0');     {this is the new function}
end;  {end of function digit}
```

strtoreal

```
function strtoreal(realstring:linestring):real;
{this function takes a real string and converts it into the
corresponding number}
{last change made: JUNE 18, 1984   by: JRP}

var   filestring : linestring; {string that will be treated as a file}
      pos : integer; {of no earthly value, required by I/O call}
      realnumber : real; {intermediate storage place before put into strtoreal}

begin
     if (strlen(realstring)<>0) then
          begin
```

```
            setstrlen(filestring,0); {clear out the 'file' filestring}
            strwrite(filestring,1,pos,realstring); {write realstring into filestring}
            strread(filestring,1,pos,realnumber); {read filestring into realnumber as a real}
            strtoreal:=realnumber; {put the real number into our function name strtoreal}

        end
    else
        strtoreal:=0.0;  {in case only a CR & LF are entered}
end;



{*****************************************************************}
{***************HPIB TALK, LISTEN & WAIT PROCEDURES***************}
{*****************************************************************}


procedure talk_to_hpib(var hpib_address:integer);                    talk_to_hpib
{this procedure sets up the hp9836 to talk to the hpib.  hpib_address is the
 current hpib address.  e.g.  14 for prober, or 17 for analyzer}
{last change made: JUNE 21, 1984  by: JRP}

begin
    untalk (7);
    unlisten (7);
    talk (7, my_address(7));
    listen (7, hpib_address);
end; {procedure talk_to_hpib}


procedure listen_to_hpib(var hpib_address:integer);                  listen_to_hpib
{this procedure sets up the hp9836 to listen to the hpib.  hpib_address is the
 current hpib address.  e.g.  14 for prober, or 17 for analyzer}
{last change made: JUNE 21, 1984  by: JRP}

begin
    untalk (7);
    unlisten (7);
    talk (7,hpib_address);
    listen (7, my_address(7));
end; {procedure listen_to_hpib}


procedure wait_till_hpib_ready(var hpib_address:integer);            wait_till_hpib_ready
{this procedure tells the hp9836 to wait until hpib is ready. hpib_address
 is the current hpib address.  e.g.  14 for prober, or 17 for analyzer}
{last change made: JUNE 21, 1984  by: JRP}

var statusbyte:integer; {status of serial poll of hpib}

begin
    statusbyte := spoll (700 + hpib_address);
    while statusbyte <> 1 do                    {keep polling until statusbyte=1}
        statusbyte := spoll (700 + hpib_address);
end; {procedure wait_till_hpib_ready}


procedure wait_till_bit7_IOSTATUS_set;                    wait_till_bit7_IOSTATUS_set
{this procedure polls bit 7 of the IOSTATUS register until it is set to '1'}
{last change made: JUNE 21, 1984  by: JRP}

var     bitset:boolean;

begin
```

...wait_till_bit7_IOSTATUS_set

```
    {not(bit_set(IOSTATUS(7,7),12)) returns the not-ready-for-data-bit}
    bitset:=false;
    while not(bitset) do
        bitset:=not(bit_set(IOSTATUS(7,7),12));
end;



{*****************************************************************}
{*********************GRAPHING FUNCTIONS**************************}
{*****************************************************************}


function test_point_on_screen1(yold,ynew,test:real):boolean;          test_point_on_screen1
{after the test point has been calculated, this function returns TRUE if test
is on the screen and between the old and new points on the left or right sides}

begin
    test_point_on_screen1:=false;
    test_point_on_screen1:=(((ynew>=yold) and (test>=yold) and (test<=ynew)) or
                            ((yold>=ynew) and (test>=ynew) and (test<=yold))) and
                                        ((test>=-0.8) and (test<=0.8));
end;


function test_point_on_screen2(xold,xnew,test:real):boolean;          test_point_on_screen2
{after the test point has been calculated, this function returns TRUE if test
 is on the screen and between the old and new points on the top or bottom}

begin
    test_point_on_screen2:=false;
    test_point_on_screen2:=(((xnew>=xold) and (test>=xold) and (test<=xnew)) or
                            ((xold>=xnew) and (test>=xnew) and (test<=xold))) and
                                        ((test>=-0.8) and (test<=0.8));
end;


{*****************************************************************
**********************USER INPUT FUNCTIONS************************
*****************************************************************}


procedure yes_no_selection_input(xpos,ypos:integer; var yes_no_answer:boolean);  yes_no_selection_input
{this procedure accepts only a yes or no answer from the user.  it will wait
 at xpos,ypos until it receives either a 'y' or 'Y' or 'n' or 'N'}
{last change made: THURSDAY SEPT 20, 1984   By: JRP}

var selection:char;

begin
    gotoxy(xpos,ypos);
    read(selection);
    while not( (selection='y') or (selection='Y') or
               (selection='n') or (selection='N') ) do
        begin
            gotoxy(xpos,ypos);
            write(' ');
            gotoxy(xpos,ypos);
            read(selection);
        end;
    if (selection='y') or (selection='Y') then
        yes_no_answer:=true
    else
        yes_no_answer:=false;
```

...yes_no_selection_input

```
end;
```

selection_input

```
procedure selection_input(xpos,ypos:integer; char1,char2:char; var selection:char);

begin
     gotoxy(xpos,ypos);
     read(selection);
     while (selection<char1) or (selection>char2) do
          begin
               gotoxy(xpos,ypos);
               write(' ');
               gotoxy(xpos,ypos);
               read(selection);
          end;
     gotoxy(xpos-3,ypos);
     write('    ');
     gotoxy(xpos-3,ypos);
     write(selection);
end;
```

```
{*************************************************************}
{*********************LEAST SQUARES PROCEDURES********************}
{*************************************************************}
```

leastsq2

```
procedure leastsq2(presindex,initindex,maxindex:integer;
                   xvalue,yvalue:real; var constant,linear:real);

{this program does a linear least square fit for an equation of
the form y=constant+(linear*x)}

var row,col:integer;
     det:real;
     rarr:array[1..4] of real;

begin
     if (presindex<initindex) or (presindex>maxindex) then
{this is the error handling for invalid presindex values}
          begin
               writeln('Error in Index For Routine leastsq2');
          end
     else
          begin
               if   presindex=initindex then
{this is the setup required for the initindex call to leastsq2}
                    begin
                         for row:=1 to 2 do
                              for col:=1 to 3 do
                                   matrix[row,col]:=0;
                         parr[1]:=1;
                         initxval:=xvalue;
                         inityval:=yvalue;
                    end;
```

...leastsq2

```
{all presindex values start execution here}
                parr[2]:=xvalue-initxval;
                parr[3]:=yvalue-inityval;
                for row:=1 to 2 do
                     for col:=1 to 3 do
                          matrix[row,col]:=matrix[row,col]+parr[row]*parr[col];
{this section does row conditioning on the final matrix}
                if presindex=maxindex then
                     begin
                          for row:=1 to 2 do
                               begin
                                    rarr[row]:=0;
                                    for col:=1 to 2 do
                                         if rarr[row]<abs(matrix[row,col]) then
                                              rarr[row]:=abs(matrix[row,col]);
                               end;
                          for row:=1 to 2 do
                               for col:=1 to 3 do
                                    if rarr[row]<>0.0 then
                                         matrix[row,col]:=matrix[row,col]/rarr[row]
                                    else
                                         leastsq_divide_by_zero:=true;
{this section does column conditioning on the final matrix}
                          for col:=1 to 2 do
                               begin
                                    rarr[col]:=0;
                                    for row:=1 to 2 do
                                         if rarr[col]<abs(matrix[row,col]) then
                                              rarr[col]:=abs(matrix[row,col]);
                               end;
                          for col:=1 to 2 do
                               for row:=1 to 2 do
                                    if rarr[col]<>0.0 then
                                         matrix[row,col]:=matrix[row,col]/rarr[col]
                                    else
                                         leastsq_divide_by_zero:=true;
{this section inverts the final matrix}
                          det:=matrix[1,1]*matrix[2,2]-matrix[2,1]*matrix[1,2];
                          if (det<>0.0) and (rarr[1]<>0.0) and (rarr[2]<>0.0) then
                               begin
                                    constant:=(matrix[1,3]*matrix[2,2]-matrix[2,3]*
                                              matrix[1,2])/det;
                                    linear:=(matrix[1,1]*matrix[2,3]-matrix[2,1]*
                                              matrix[1,3])/det;
                                    constant:=constant/rarr[1];
                                    linear:=linear/rarr[2];
                               end
                          else
                               leastsq_divide_by_zero:=true;
                          constant:=constant-linear*initxval+inityval;
                     end;  {end of handling for maximum index value}
          end;   {end of handling for good index values}
end;   {end of procedure leastsq2}
```

leastsq3

```
procedure leastsq3(presindex,initindex,maxindex:integer; var1value,var2value,
                   yvalue:real; var constant,linear1,linear2:real);


{this procedure does a linear least square fit for an equation of the form
y=constant+linear1*var1value+linear2*var2value}

var   row,col:integer;
      temp1index,temp2index:integer;
      det,temp:real;
      rarr:array[1..3] of real;
```

```
      sow:integer;  {for debugging*****************}

begin
    if (presindex<initindex) or (presindex>maxindex) then
{this is the error handling for invalid presindex values}
        begin
            writeln('error in index for routine leastsq3');
        end
    else
        begin
        {************************take out later*************************}
            if lsq3=true then
                    writeln(userout2,
                            'w /in lsq3 presindex=',presindex:2,
                            ' var1=',var1value,' var2=',var2value,
                            ' y=',yvalue);
        {************************take out later*************************}
            if presindex=initindex then
                {this is the setup for the initial call}
                begin
                    for row:=1 to 3 do
                        for col:=1 to 4 do
                            matrix[row,col]:=0;
                    initval1:=var1value;
                    initval2:=var2value;
                    inityval:=yvalue;
                end;
            parr[1]:=1.0;
            {all presindex values start here}
            parr[2]:=var1value-initval1;
            parr[3]:=var2value-initval2;
            parr[4]:=yvalue-inityval;
            for row:=1 to 3 do
                for col:=1 to 4 do
                    matrix[row,col]:=matrix[row,col]+parr[row]*parr[col];
        {************************take out later*************************}
            if lsq3=true then
                begin
                    writeln(userout2,'at 1st filling');
                    for row:=1 to 3 do
                        begin
                            for col:=1 to 4 do
                                write(userout2,matrix[row,col],' ');
                            writeln(userout2);
                        end;
                end;
        {************************take out later*************************}
            {final row and column conditioning is done here}
            if presindex=maxindex then
                begin
                    for row:=1 to 3 do
                        begin
                            rarr[row]:=0;
                            for col:=1 to 3 do
                                if rarr[row]<abs(matrix[row,col]) then
                                    rarr[row]:=abs(matrix[row,col]);
                        end;
                    for row:=1 to 3 do
                        for col:=1 to 4 do
                            begin
                                if rarr[row]<>0.0 then
                                    matrix[row,col]:=matrix[row,col]/rarr[row]
                                else
                                    leastsq_divide_by_zero:=true;
                            end;
```

```
{**********take out later**********}
    if lsq3=true then
        begin
            writeln(userout2,'at 2nd filling');
            for row:=1 to 3 do
                begin
                    for col:=1 to 4 do
                        write(userout2,matrix[row,col],' ');
                    writeln(userout2);
                end;
        end;
{**********take out later**********}
   {this is the column conditioning}
            for col:=1 to 3 do
                begin
                    rarr[col]:=0;
                    for row:=1 to 3 do
                        if rarr[col]<abs(matrix[row,col]) then
                            rarr[col]:=abs(matrix[row,col]);
                end;
            for col:=1 to 3 do
                for row:=1 to 3 do
                    begin
                        if rarr[col]<>0.0 then
                            matrix[row,col]:=matrix[row,col]/rarr[col]
                        else
                            leastsq_divide_by_zero:=true;
                    end;
{**********take out later**********}
    if lsq3=true then
        begin
            writeln(userout2,'at 3rd filling');
            for row:=1 to 3 do
                begin
                    for col:=1 to 4 do
                        write(userout2,matrix[row,col],' ');
                    writeln(userout2);
                end;
        end;
{**********take out later**********}
   {this section inverts the final matrix}
            for col:=1 to 4 do
                parr[col]:=0;
            temp1index:=2;
            temp2index:=3;
                for row:=1 to 3 do
                    begin
                        temp:=matrix[temp1index,2]*
   matrix[temp2index,3]-matrix[temp2index,2]*matrix[temp1index,3];
                        parr[4]:=parr[4]+matrix[row,1]*temp;
                        parr[1]:=parr[1]+matrix[row,4]*temp;
                        parr[2]:=parr[2]+matrix[row,4]*(
   matrix[temp1index,3]*matrix[temp2index,1]-matrix[temp2index,3]*
   matrix[temp1index,1]);
                        parr[3]:=parr[3]+matrix[row,4]*
   (matrix[temp1index,1]*matrix[temp2index,2]-
   matrix[temp2index,1]*matrix[temp1index,2]);
{**********take out later**********}
    if lsq3=true then
        begin
            writeln(userout2,'at parr calculation point & parr array=');
            for sow:=1 to 4 do
                write(userout2,'parr[',sow:1,']=',parr[sow],' ');
            writeln(userout2);
        end;
```

...leastsq3

```
        {**********************take out later**************************}
                                        templindex:=templindex+1;
                                        temp2index:=temp2index+1;
                                        if temp2index>3 then
                                            temp2index:=1;
                                        if templindex>3 then
                                            templindex:=1;
                                    end;
                            if (parr[4]<>0.0) and (rarr[1]<>0.0) and (rarr[2]<>0.0)
                                                                and (rarr[3]<>0.0) then
                                begin
                                    constant:=parr[1]/parr[4]/rarr[1];
                                    linear1:=parr[2]/parr[4]/rarr[2];
                                    linear2:=parr[3]/parr[4]/rarr[3];
                                    constant:=constant-linear1*initval1-linear2*initval2+
                                                inityval;
                                end
                            else
                                leastsq_divide_by_zero:=true;
        {**********************take out later**************************}
            if lsq3=true then
                begin
                    writeln(userout2,'leaving lsq3 and leastsq_divide_by_zero=',leastsq_divide_by_zero
                    writeln(userout2,
                    'constant=',constant,' linear1=',linear1,' linear2=',linear2);
                    writeln(userout2,
                    ' rarr[2]=',rarr[2],' rarr[3]=',rarr[3],'parr[1]=',parr[1]);
                    writeln(userout2,
                    ' parr[2]=',parr[2],'parr[3]=',parr[3],' parr[4]=',parr[4]);
                end;
        {**********************take out later**************************}
                end;
        end;
end;  {end of procedure leastsq3}


{*********************************************************************
*************************MAIN BSIM MENU ROUTINES**********************
*********************************************************************}

procedure initial_bsim_page:                                    initial_bsim_page
{this procedure shows the initial bsim information and takes the user inputs}
{last change made: JUNE 18,1984   by: JRP}

begin
    writeln(#12);  {clears the screen}
    gotoxy(12,0);
    write('BSIM AUTOMATIC MOS DEVICE CHARACTERIZATION PROGRAM');
    gotoxy(20,1);
    write('UC BERKELEY FALL 1984 VERSION 1.0');
    gotoxy(0,2);
    writeln('This Program can be used in any of the following modes:');
    write('[1] Fully Automatic. [2] Semi Automatic--with an automatic');
    writeln(' prober,');
    write('[3] Semi Automatic--with a manual prober, and [4] Single Device');
    writeln(' Operation.');
    writeln;
    write('FULLY AUTOMATIC OPERATION requires a prober file, and');
    writeln(' tests all devices ');
    write('in the file without interuption. This mode requires an');
    writeln(' automatic prober.');
    writeln;
    write('SEMI AUTOMATIC--[AUTOMATIC PROBER] OPERATION requires a prober');
```

...initial_bsim_page

```
        writeln(' file and auto-,');
        write('matically moves to each device in the file.  This mode stops at');
        writeln(' each device to');
        write('allow the user to switch connections. This mode requires an');
        writeln(' automatic prober.');
        writeln;
        write('SEMI AUTOMATIC--[MANUAL PROBER] OPERATION is similar to SEMI ');
        writeln('AUTOMATIC--');
        writeln('[AUTOMATIC PROBER], but does not require an automatic prober.');
        writeln;
        write('SINGLE DEVICE OPERATION allows the user to analyze an ');
        writeln('individual device,');
        write('extract BSIM parameters, and compare simulated');
        writeln(' versus measured data.');
        gotoxy(0,21);
        write('Select a Mode of Operation  >');
        gotoxy(32,19);
        write('[1]:FULLY AUTOMATIC');
        gotoxy(32,20);
        write('[2]:SEMI AUTOMATIC--[AUTOMATIC PROBER]');
        gotoxy(32,21);
        write('[3]:SEMI AUTOMATIC--[MANUAL PROBER]');
        gotoxy(32,22);
        write('[4]:SINGLE DEVICE');
        gotoxy(32,23);
        write('[5]:EXIT BSIM');
        selection_input(29,21,'1','5',mode);
end;
```

clear_output_file

```
procedure clear_output_file;
{this procdure simply clears out the output file.  if the user has chosen
 the same name output file as he did on the previous run, the old one will
 get erased.  this is done so that only the present devices under test will
 get recorded in the output file.  if an output file of the same name does
 not exist, it will be created first (so as not to get an operating system
 error) }
{last change made:  JULY 15, 1984  by: JRP}

begin
    {suppose the user chooses 'bsimout.TEXT', for example, for his
    output file, quits the program, and then runs it again
    and chooses 'bsimout.TEXT' again.  now if the original
    bsimout.TEXT was not destroyed by the user, all information
    to be put into bsimout.TEXT in the 2nd run will simply be
    added to what is already in bsimout.TEXT from the first run!
    that's why we must destroy the output file before it gets
    created, and in case it doesn't exist, we will create it
    first, and then destroy it...all to be followed by opening
    the file up and making it writeable!!  are you confused yet?}

    rewrite(userout,output_file);{create output_file if it doesn't}
    close(userout,'save');          {exist, or open and save if it does}
    reset(userout,output_file);{now make output_file readable and}
    close(userout,'purge');         {then destroy it}
    rewrite(userout,output_file); {now create new output_file and}
    close(userout,'save');          {make it writeable}
end;  {of procedure clear_output_file}
```

initialize_17_bsim_parameters_to_zero

```
procedure initialize_17_bsim_parameters_to_zero;
```

..initialize_17_bsim_parameters_to_zero

```
{this procedure sets all 17 bsim parameters to zero.  this is performed
 only on the first time thru the program, and subsequently every time the
 user decides to continue with another form of the program.  (e.g.  if the
 user decides to execute the "SINGLE MODE OF OPERATION", this procedure is
 called.  after completion of the "SINGLE" mode, if he decides to go again,
 say in the "AUTOMATIC MODE OF OPERATION", then it is called again at the
 the beginning of the mode.)  this procedure is needed because if a 2nd
 time thru the program is desired, all 17 parameters will be remembered
 from the first time thru, and c2_phif2 might have some value, say .800.
 then later on in the MAIN BSIM execution, where it asks ... if c2_phif2=0 ...
 this statement could be wrong.
{last change made: July 14, 1984   by: JRP}

 begin
      c1_vfb:=0.0;
      c2_phif2:=0.0;
      c3_k1:=0.0;
      c4_k2:=0.0;
      c5_eta:=0.0;
      c6_beta0:=0.0;
      c7_u0:=0.0;
      c8_u1:=0.0;
      c9_x2beta0:=0.0;
      c9_x2mu0:=0.0;
      c10_x2eta:=0.0;
      c11_x3eta:=0.0;
      c12_x2u0:=0.0;
      c13_x2u1:=0.0;
      c14_beta0sat:=0.0;
      c14_mu0sat:=0.0;
      c15_x2beta0sat:=0.0;
      c15_x2mu0sat:=0.0;
      c16_x3beta0sat:=0.0;
      c16_x3mu0sat:=0.0;
      c17_x3u1:=0.0;
 end;  {of procedure initialize_17_bsim_parameters_to_zero}
```

standard_input_display

```
procedure standard_input_display;
{this procedure prompts the user for all inputs common to
modes of operation}

{last change made: JUNE 19, 1984    by: JRP}

begin
      gotoxy(0,2);
      writeln('Process Name=?   >');
      writeln('Lot=?   >');
      if not((mode='1') or (mode='2') or (mode='3'))  then  {mode='4'}
           writeln('Wafer=?   >              XPOSITION=?   >              YPOSITON=?   >')
      else
           writeln('Wafer=?   >');
      writeln('Date=?   >');
      writeln('Operator=?   >');
      writeln('Output File=?   >');
      writeln('VDD(volts)=?   >');
      writeln('TEMPERATURE(deg. C)=?   >');
      writeln('TOX(angstroms)=?   >');
end;
```

input_standard_values

```
procedure input_standard_values;
```

..input_standard_values

```
{this procedure handles inputing values common to all modes of operation}
{last change made: SUNDAY OCT 14, 1984  By: JRP}

const blanking='                                                    ';

var i:integer;
    device_name:char;

begin
      gotoxy(17,2);  {process name input}
      readln(process);
      if strlen(process)>25 then
            strdelete(process,25,strlen(process)-24);  {deletes extra char.}
      gotoxy(13,2);
      write(blanking);
      gotoxy(13,2);
      write(process);
      gotoxy(8,3);  {lot input}
      readln(lot);
      if strlen(lot)>25 then
            strdelete(lot,25,strlen(lot)-24);  {deletes extra char.}
      gotoxy(4,3);
      write(blanking);
      gotoxy(4,3);
      write(lot);
      gotoxy(10,4);  {wafer input}
      readln(wafer);
      if strlen(wafer)>25 then
            strdelete(wafer,25,strlen(wafer)-24);  {deletes extra char.}
      gotoxy(6,4);
      write('                    ');
      gotoxy(6,4);
      write(wafer);
      if not((mode='1') or (mode='2') or (mode='3')) then {mode='4'}
            begin
                  gotoxy(34,4);
                  present_diex:=trunc(strtoreal(realtostr));
                  gotoxy(30,4);
                  write('                    ');
                  gotoxy(30,4);
                  write(present_diex:1);
                  gotoxy(57,4);
                  present_diey:=trunc(strtoreal(realtostr));
                  gotoxy(53,4);
                  write('                    ');
                  gotoxy(53,4);
                  write(present_diey:1);
            end;
      gotoxy(9,5);  {date input}
      readln(date);
      if strlen(date)>25 then
            strdelete(date,25,strlen(date)-24);  {deletes extra char.}
      gotoxy(5,5);
      write(blanking);
      gotoxy(5,5);
      write(date);
      gotoxy(13,6);  {operator input}
      readln(operator);
      if strlen(operator)>25 then
            strdelete(operator,25,strlen(operator)-24);  {deletes extra char.}
      gotoxy(9,6);
      write(blanking);
      gotoxy(9,6);
      write(operator);
```

..input_standard_values

```
        gotoxy(16,7)  {output_file input};
        readln(output_file);
        colon_position:=strpos(':',output_file);
        period_position:=strpos('.TEXT',output_file);
        if (period_position<>0) then        {'.TEXT' has been included in file name}
             strdelete(output_file,colon_position+11,period_position
                                                                    -colon_position-11)
        else                                {'.TEXT' has not been included in file name}
             strdelete(output_file,colon_position+11,strlen(output_file)
                                                                    -colon_position-10);
        if strlen(output_file)=0 then
              output_file:='bsimout.TEXT';
        gotoxy(12,7);
        write(blanking);
        gotoxy(12,7);
        write(output_file);
        gotoxy(15,8);
        vdd:=strtoreal(realtostr);  {reads in the value of vdd and cleans it up}
        gotoxy(11,8);
        write(blanking);
        gotoxy(11,8);
        write(vdd:1:2);
        gotoxy(24,9);
        temp:=strtoreal(realtostr);  {reads in temp and cleans it up}
        gotoxy(20,9);
        write(blanking);
        gotoxy(20,9);
        write(temp:1:2);
        temp:=temp+273.15;    {conversion of temperature to Kelvin}
        gotoxy(19,10);
        tox:=strtoreal(realtostr);  {reads in tox and cleans it up}
{jrp}while (tox=0.0) do
{jrp}     begin
{jrp}          gotoxy(19,10);
{jrp}          write('                PLEASE ENTER NON-ZERO VALUE...');
{jrp}          gotoxy(19,10);
{jrp}          tox:=strtoreal(realtostr);
{jrp}     end;
       cox:=3.9*8.854E-6/tox;
       gotoxy(15,10);
       write(blanking);
       gotoxy(15,10);
       write(tox:1:2);
end;

procedure automatic_mode_inputs;                          automatic_mode_inputs
{this procedure handles making the automatic mode display}
{last change made: THURSDAY SEPT 20. 1984   By: JRP}

var i:integer;

begin
       gotoxy(14,0);
       write('***AUTOMATIC OR SEMI-AUTOMATIC OPERATION***');
       standard_input_display;    {this displays the standard prompts}
       gotoxy(0,12);
       writeln('Prober File=?   >');
       writeln;
       writeln('At the end of EACH DIE, would you like to view plots of ');
       writeln('BSIM PARAMETER vs W or L?  (Y/N)   >');
       writeln;
       writeln('Probing Instructions');
       writeln('       The prober should be on, and the probes should be down');
       write('on the starting die, starting position.');
       writeln('   (see prober instructions)');
```

...automatic_mode_inputs

```
        writeln('HIT a "C" for changes, or any other key to start.  >');
        input_standard_values;
        gotoxy(16,12);
        readln(prober_file);
        if strlen(prober_file)>15 then
              strdelete(prober_file,16,strlen(prober_file)-15);
        if strlen(prober_file)=0 then
              prober_file:='bsimprob.TEXT';
        gotoxy(12,12);
        write('                                                    ');
        gotoxy(12,12);
        write(prober_file);
        yes_no_selection_input(35,15,po_graphics_wanted);
        gotoxy(27,15);
        write('          ');
        gotoxy(27,15);
        if po_graphics_wanted=true then
             write('YES')
        else
             write('NO');
        gotoxy(52,20);
end;
```

single_device_mode_inputs

```
procedure single_device_mode_inputs;
{this procedure handles device size inputs, and also needs to know
which SMUs are connected to each device terminal}
{last change made: JUNE 18, 1984    by: JRP}

const blanking='                                                ';

var   i:integer;
      phif2_or_nsub,k_t_q,ni:real;

begin
      gotoxy(26,0);
      write('***SINGLE DEVICE OPERATION***');
      standard_input_display;
      gotoxy(0,11);
      writeln('PHIF2 or NSUB=?  >');
      writeln('drawn width (microns)=?  >');
      writeln('drawn length (microns)=?  >');
      writeln('Device type=?  >     [1] enhancement, [2] zero-threshold, [3] depletion');
      writeln;
      writeln('SMU connected to DRAIN=?  >');
      writeln('SMU connected to GATE=?  >');
      writeln('SMU connected to SOURCE=?  >');
      writeln('SMU connected to BODY=?  >');
      writeln;
      writeln('Hit a "C" for changes or any other key to start.  >');
      input_standard_values;
      gotoxy(18,11);
      phif2_or_nsub:=abs(strtoreal(realtostr));  {reads value of phif2 or nsub}
      gotoxy(14,11);
      writeln(blanking);
      gotoxy(14,11);
      write(phif2_or_nsub);
      {here is where the input value of phif2 or nsub is read}
      if phif2_or_nsub>10 then
            begin
                  {the value is nsub}
                  k_t_q:=0.0258512*(temp/300.0);
                  ni:=1.45E10*exp(1.5*ln(temp/300))*exp(1.124/2.0/k_t_q*
                        (1-300/temp));
```

```
                c2_phif2:=2.0*k_t_q*ln(phif2_or_nsub/ni);
            end
        else
            c2_phif2:=phif2_or_nsub;
        gotoxy(26,12);
        width:=strtoreal(realtostr);   {reads in the value of width}
        gotoxy(22,12);
        writeln(blanking);
        gotoxy(22,12);
        write(width:2:2);
        gotoxy(27,13);
        length:=strtoreal(realtostr);   {reads in the value of length}
        gotoxy(23,13);
        writeln(blanking);
        gotoxy(23,13);
        writeln(length:2:2);
{jrp}enhancement_depletion:=5;   {set to bogus value so while loop is done}
{jrp}while ((enhancement_depletion<1) or (enhancement_depletion>3)) do
{jrp}     begin
              gotoxy(16,14);
              enhancement_depletion:=round(strtoreal(realtostr));
{jrp}         gotoxy(16,14);
{jrp}         write('      ');
{jrp}     end;
        gotoxy(12,14);                                  {read in devicetype value}
        write(blanking);
        writeln(blanking);
        gotoxy(12,14);
        case enhancement_depletion of
            1:begin                                      {user input enhancement}
                write('enhancement');
                enhancement_depletion:=1;
              end;
            2:begin                                      {user input zero-threshold}
                write('zero-threshold');
                enhancement_depletion:=0;
              end;
            3:begin                                      {user input depletiont}
                write('depletion');
                enhancement_depletion:=-1;
              end;
        end;   {of case}
        gotoxy(27,16);
        smud:=round(strtoreal(realtostr));   {reads in the value of drain smu}
        gotoxy(23,16);
        writeln(blanking);
        gotoxy(23,16);
        writeln(smud:1);
        gotoxy(26,17);
        smug:=round(strtoreal(realtostr));   {reads in the value of gate smu}
        gotoxy(22,17);
        writeln(blanking);
        gotoxy(22,17);
        writeln(smug:1);
        gotoxy(28,18);
        smus:=round(strtoreal(realtostr));   {reads in the value of source smu}
        gotoxy(24,18);
        writeln(blanking);
        gotoxy(24,18);
        writeln(smus:1);
        gotoxy(26,19);
        smub:=round(strtoreal(realtostr));   {reads in the value of body smu}
        gotoxy(22,19);
        writeln(blanking);
        gotoxy(22,19);
```

```
      writeln(smub:1);
      gotoxy(51,21);
end;
```

initial_status_inputs

```
procedure initial_status_inputs;
{this procedure handles getting the first set of inputs required by any
of the three modes of operation}
{last change made: JULY 14,1984  by: JRP}

var change:char;

begin
{**************************take out later**************************}
      if TRACE21=true then writeln(userout2,
            'about to enter initial_status_inputs');
{**************************take out later**************************}
      initialize_17_bsim_parameters_to_zero;  {in case c2_phif2 <> 0}
      change:='c';
      while (change='c') or (change='C') do
            begin
                  writeln(#12);
                  case mode of
                        '1','2','3':automatic_mode_inputs;
                        '4':single_device_mode_inputs;
                  end;  {end of case}
                  read(change);
            end;  {end of change loop}
      if ((mode='1') or (mode='2') or (mode='3')) then
            begin
                  gotoxy(52,20);
                  write('READING PROBER FILE');
            end;
{JRP} if TRACE40=false then
            clear_output_file;
end:   {end of procedure to get initial inputs}
```

initial_status_display

```
procedure initial_status_display;
{this procedure handles making the initial status chart which is
displayed during the progress of an extraction}
{last change made: JUNE 18,1984  by: JRP}

begin
      writeln(#12);    {clears the display}
      gotoxy(23,0);
      writeln('***BSIM EXTRACTION STATUS***');
      gotoxy(0,2);
      write('PROCESS=',process);
      gotoxy(40,2);
      write('VDD=',vdd:1:2,' VOLTS');
      gotoxy(0,3);
      write('LOT=',lot);
      gotoxy(40,3);
      temp:=temp-273.15;
      write('TEMP=',temp:1:2,' DEG C');
      temp:=temp+273.15;
      gotoxy(0,4);
      write('WAFER=',wafer);
      gotoxy(40,4);
      write('TOX=',tox:1:2,' ANGSTROMS');
      gotoxy(0,5);
      write('DATE=',date);
      gotoxy(40,5);
      write('XPOS=',present_diex:2,' YPOS=',present_diey:2);
      gotoxy(0,6);
```

...initial_status_display

```
        write('OPERATOR=',operator);
        gotoxy(40,6);
        write('DEVICE=');
        gotoxy(0,7);
        write('OUTPUT FILE=',output_file);
        gotoxy(40,7);
        write('WIDTH=',width:1:2,' MICRONS');
        gotoxy(0,8);
        write('PROBER FILE=');
        if ((mode='1') or (mode='2') or (mode='3')) then
              write(prober_file)
        else
              write('SINGLE DEVICE OPERATION');
        gotoxy(40,8);
        write('LENGTH=',length:1:2,' MICRONS');
        gotoxy(0,10);
        write('MINUTES TO DIE COMPLETION=');
        gotoxy(40,10);
        write('MINUTES TO WAFER COMPLETION=');
        gotoxy(0,11);
        write('DEVICE EXTRACTION LOCATION ');
        gotoxy(64,11);
        write('FINISHED');
        gotoxy(0,13);
        writeln('PRESENT DEVICE BSIM PARAMETERS');
        gotoxy(40,13);
        writeln('X2U0=');
        write('VFB=');
        gotoxy(40,14);
        writeln('X2U1=');
        write('PHIF2=');
        if mode='4' then
              write(c2_phif2:1:3);
        gotoxy(40,15);
        writeln('X3U1=');
        write('K1=');
        gotoxy(40,16);
        writeln('X2BETA0=');
        write('K2=');
        gotoxy(40,17);
        writeln('X2ETA=');
        write('ETA=');
        gotoxy(40,18);
        writeln('X3ETA=');
        write('BETA0=');
        gotoxy(40,19);
        writeln('BETA0SAT=');
        write('U0=');
        gotoxy(40,20);
        writeln('X2BETA0SAT=');
        write('U1=');
        gotoxy(40,21);
        write('X3BETA0SAT=');
        gotoxy(0,23);
        write('message from program=');
end;
```

bsim_timer

```
procedure bsim_timer(var present_increment:integer);

const minutes_per_device=2.5;
      number_of_increments=37;

var   minutes_to_die_completion,minutes_to_wafer_completion:real;
      i:integer;
```

```
begin
      minutes_to_die_completion:=minutes_per_device*(number_devices-
                                    present_device+(number_of_increments-
                                    present_increment)/number_of_increments);
      minutes_to_wafer_completion:=(number_die-present_die)*number_devices*
                                  minutes_per_device+minutes_to_die_completion;
      {here is where the speedometer is written}
      if present_increment=1 then
            begin
                  gotoxy(26,11);
                  write('                                    ');
            end;
      gotoxy(27,11);
      for i:=1 to present_increment do
            write('X');
      present_increment:=present_increment+1;
      {here is where the time values are written}
      gotoxy(26,10);
      write('          ');
      gotoxy(68,10);
      write('          ');
      gotoxy(26,10);
      write(minutes_to_die_completion:1:1);
      gotoxy(68,10);
      write(minutes_to_wafer_completion:1:1);
      gotoxy(21,23);
end;  {end of bsim_timer}



{*********************************************************************}
{*******************BSIM MEASUREMENT ROUTINES*************************}
{*********************************************************************}

procedure measure_device(var device_type,error_code:integer;
                         vdterm,vgterm,vsterm,vbterm:integer; vdd:real);
{last change made: SUNDAY OCT 14, 1984   By: JRP}

var i,j:integer;                                    {loop counters}
    vdchan,vgchan,vschan,vbchan:string[20]; {to contain VCX where X is SMU}
    vdnumb,vgnumb,vsnumb,vbnumb:string[20]; {to contain SMU number as string}
    vddstr,vdd2str,mvddstr,vddplus1str:string[20];{to contain vdd strings}
    vdd2,minusvdd,vddplus1:real;                    {these are supply values}
    idsnch,idspch:real;
    idsgzero,idsgvdd:real;                          {currents for n and p conjig.}
    nextpos:integer;                                  {this is the next position in a
                                                            string that could be written to}
    temp:char;
    continue:char;  {if mode='4', and error occurs, press enter to continue}


procedure translate_terminals_to_strings;
{this procedure sets up string variables needed by the 4145}
{last change made: JUNE 20, 1984     by: JRP}

var i:integer;

begin
      bsim_timer(time_count);
      {here is where the terminal numbers are converted into channel strings}
      vdnumb:=' ';
      vgnumb:=' ';
      vsnumb:=' ';
      vbnumb:=' ';
```

measure_device

translate_terminals_to_strings

...translate_terminals_to_strings

```
        vddstr:='                                ';
        vdd2str:='                               ';
        mvddstr:='                               ';
        vddplus1str:='                             ';
        strwrite(vdnumb,1,nextpos,vdterm);
        strwrite(vgnumb,1,nextpos,vgterm);
        strwrite(vsnumb,1,nextpos,vsterm);
        strwrite(vbnumb,1,nextpos,vbterm);
        strwrite(vddstr,1,nextpos,vdd:2:1);
        vdd2:=2*vdd;
        strwrite(vdd2str,1,nextpos,vdd2:2:1);
        minusvdd:=-vdd;
        strwrite(mvddstr,1,nextpos,minusvdd:2:1);
        vddplus1:=vdd+1;
        strwrite(vddplus1str,1,nextpos,vddplus1:2:1);
        vdchan:='CH  ';
        strwrite(vdchan,3,nextpos,vdterm);   {sets up vd channel string}
        vgchan:='CH  ';
        strwrite(vgchan,3,nextpos,vgterm);   {sets up vg channel string}
        vschan:='CH  ';
        strwrite(vschan,3,nextpos,vsterm);   {sets up vs channel string}
        vbchan:='CH  ';
        strwrite(vbchan,3,nextpos,vbterm);   {sets up vb channel string}
end:    {end of procedure to translate_terminals_to_strings}
```

chan_definition_to_test_device_type

```
procedure chan_definition_to_test_device_type:
{this configuration is used to test for the type of device present by
checking the source,drain to body diodes}
{last change made: JUNE 20, 1984     by: JRP}

begin
        talk_to_hpib(hpib_analyzer_address);
        writestring(700+hpib_analyzer_address,'IT1 CA1 DR0 BC');  {auto calibration of 4145}
        {this is the source definition}
        writestring(700+hpib_analyzer_address,'DE  ');  {4145 to channel definition page}
        writestring(700+hpib_analyzer_address,vdchan);  {chan. def. for vd}
        writestring(700+hpib_analyzer_address,',''VDRAIN'',''IDRAIN'',1,3;');  {sets the DRAIN source}
        writestring(700+hpib_analyzer_address,vschan);  {chan. def. for vs}
        writestring(700+hpib_analyzer_address,',''VSOURC'',''ISOURC'',1,3;');  {sets the SOURC source}
        writestring(700+hpib_analyzer_address,vbchan);  {chan. def. for vb}
        writestring(700+hpib_analyzer_address,',''VBODY'',''IBODY'',1,1;');  {sets variable body}
        writestring(700+hpib_analyzer_address,vgchan);  {chan. def for vg}
        writestring(700+hpib_analyzer_address,',''VGATE'',''IGATE'',1,3;');  {sets the GATE source}
        writestring(700+hpib_analyzer_address,'VS1; VS2; VM1; VM2;');  {opens up extra sources}
end;    {end of procedure type_of_device_channel_definition}
```

source_setup_to_test_device_type

```
procedure source_setup_to_test_device_type;
{this procedure sets up the SMUs to measure the type of device}
{it steps the body from -VDD to +VDD and measures the gate and body currents}
{last change made: JUNE 24, 1984     by: JRP}

begin
        writestring(700+hpib_analyzer_address,'SS  VC');  {go to source setup page}
        writestring(700+hpib_analyzer_address,vdnumb);  {sets vd to 0 volts}
        writestring(700+hpib_analyzer_address,',0.0,0.01;');  {id compliance=10ma}
        writestring(700+hpib_analyzer_address,'VC');
        writestring(700+hpib_analyzer_address,vsnumb);  {sets vs to 0 volts}
        writestring(700+hpib_analyzer_address,',0.0,0.01;');  {is compliance=10ma}
        writestring(700+hpib_analyzer_address,'VC');
        writestring(700+hpib_analyzer_address,vgnumb);  {sets vg to 5 volts}
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,vddstr);
```

...source_setup_to_test_device_type

```
        writestring(700+hpib_analyzer_address,',1E-6;');  {sets vg compliance=1ua}
        writestring(700+hpib_analyzer_address,'VR1, ');
        writestring(700+hpib_analyzer_address,mvddstr);
        writestring(700+hpib_analyzer_address,', ');
        writestring(700+hpib_analyzer_address,vddstr);
        writestring(700+hpib_analyzer_address,', ');
        writestring(700+hpib_analyzer_address,vdd2str);
        writestring(700+hpib_analyzer_address,', 0.025;'); {sets ib compliance=25ma}
end;    {end of procedure setup_to_test_device_type}
```

measure_device_type

```
procedure measure_device_type(var devtype,error:integer);
{this procedure measures the currents to determine the type of device
present, and returns the device type}
{last change made: JUNE 20, 1984    by: JRP}

var bodyneg,bodypos:real;
    gateneg,gatepos:real;

begin
        bsim_timer(time_count);
        writestring(700+hpib_analyzer_address,'SM  DM2,LI  ''IBODY'','',''IGATE'';');
        writestring(700+hpib_analyzer_address,'IT1  DR0  BC');  {clears the buffer}
        writestring(700+hpib_analyzer_address,'MD  ME1;');  {make measurement}
        wait_till_hpib_ready(hpib_analyzer_address);
        writestring(700+hpib_analyzer_address,'DO  ''IBODY'';');
        listen_to_hpib(hpib_analyzer_address);
        readnumber(700+hpib_analyzer_address,bodyneg);
        readnumber(700+hpib_analyzer_address,bodypos);
        writestring(700+hpib_analyzer_address,'IT1  CA1  DR0  BC'); {clears the buffer} {does calibration}
        writestring(700+hpib_analyzer_address,'DO  ''IGATE'';');
        readnumber(700+hpib_analyzer_address,gateneg);
        readnumber(700+hpib_analyzer_address,gatepos);
        error:=0;
        if (abs(gateneg)>1E-7) or (abs(gatepos)>1E-7) then
                begin
                        devtype:=0;   {gate short}
                        error:=1;
                end
        else if (abs(bodyneg)>10E-6) and (abs(bodypos)>10E-6) then
                begin
                        devtype:=0;   {short between source,drain and body}
                        error:=2;
                end
        else if (abs(bodyneg)<10E-6) and (abs(bodypos)<10E-6) then
                begin
                        devtype:=0;
                        error:=3;   {open between source,drain and body}
                end
        else if (abs(bodyneg)>10E-6) then
                begin
                        devtype:=-1;     {pchannel device}
                        gotoxy(47,6);
                        write('PCHANNEL');
                end
        else
                begin
                        devtype:=1;     {nchannel device}
                        gotoxy(47,6);
                        write('NCHANNEL');
                end;
        talk_to_hpib(hpib_analyzer_address);
end;    {end of procedure measure_device_type}
```

...measure_device

procedure chan_definition_device_functionality; *chan_definition_device_functionality*

```
{this procedure sets up the channel names for the device functionality test,
and sets up the supply types}
{last change made: OCT 2, 1984     by: JRP}

begin
      writestring(700+hpib_analyzer_address,'IT1 DR0 BC'):    {auto calibration of 4145}
      {this is the source definition}
      writestring(700+hpib_analyzer_address,'DE '); {4145 to channel definition page}
      writestring(700+hpib_analyzer_address,vdchan); {channel definition for vd}
      writestring(700+hpib_analyzer_address,',',''VDRAIN'','IDRAIN',1.3;'): {sets up the DRAIN source}
      writestring(700+hpib_analyzer_address,vgchan); {channel definition for vg}
      writestring(700+hpib_analyzer_address,',','VGATE','IGATE',1,1;'):  {sets up the GATE source}
      writestring(700+hpib_analyzer_address,vschan); {channel definition for vs}
      writestring(700+hpib_analyzer_address,',','VSOURC','ISOURC',1.3;');{sets the SOURCE source}
      writestring(700+hpib_analyzer_address,vbchan); {channel definition for vb}
      writestring(700+hpib_analyzer_address,',','VBODY','IBODY',1.3;'):  {sets up the BODY source}
      writestring(700+hpib_analyzer_address,'VS1; VS2; VM1; VM2;'); {sets up
                        the  voltage sources and monitors as not being used}
end; {end of procedure channel_definition_device_functionality}
```

procedure source_setup_nchannel_device_functionality; *source_setup_nchannel_device_functionality*

```
{this procedure is called to setup the source values to test whether
 a device is an nchannel device.
 the device is diode connected with VB=VS=0. then VD=VG is stepped to VDD.
 We should get IDRAIN>>0}
{last change made: JUNE 20, 1984     by: JRP}

begin
      {this is the source setup}
      strwrite(vddstr,1,nextpos,vdd);  {puts vdd into a string format}
      writestring(700+hpib_analyzer_address,'SS '); {move to source setup page}
      writestring(700+hpib_analyzer_address,'VC'):  {set drain as voltage source}
      writestring(700+hpib_analyzer_address,vdnumb);
      writestring(700+hpib_analyzer_address,',');
      writestring(700+hpib_analyzer_address,vddstr);
      writestring(700+hpib_analyzer_address,'.0.1;'); {set id compliance=100ma}
      writestring(700+hpib_analyzer_address,'VR1,0.0,'); {set gate as variable source}
      writestring(700+hpib_analyzer_address,vddstr);
      writestring(700+hpib_analyzer_address,',');
      writestring(700+hpib_analyzer_address,vddstr);
      writestring(700+hpib_analyzer_address,',1E-6;'); {set gate compliance=1ua}
      writestring(700+hpib_analyzer_address,'VC');
      writestring(700+hpib_analyzer_address,vsnumb);  {set source as v-source}
      writestring(700+hpib_analyzer_address,'.0.0,0.1;'): {set is compliance=100ma}
      writestring(700+hpib_analyzer_address,'VC');
      writestring(700+hpib_analyzer_address,vbnumb);  {set body as v-source}
      writestring(700+hpib_analyzer_address,'.0.0,0.01;'): {set ib compliance=10ua}
end; {end of source_setup_nchannel_device_functionality}
```

procedure source_setup_pchannel_device_functionality; *source_setup_pchannel_device_functionality*

```
{this procedure is used to determine whether a device is a pchannel device.
 the device is diode connected with VS=VB=VDD and VG=VD is stepped to 0V.
 we should get IDRAIN<<0}
{last change made: JUNE 20, 1984     by: JRP}

begin
      writestring(700+hpib_analyzer_address,'SS');  {4145 to source setup page}
      writestring(700+hpib_analyzer_address,'VC'):  {set drain as v-source}
      writestring(700+hpib_analyzer_address,vdnumb);
```

...source_setup_pchannel_device_functionality

```
        writestring(700+hpib_analyzer_address,',0.0,0.1;'); {set id compliance=100ma}
        writestring(700+hpib_analyzer_address,'VR1,0.0,'); {set gate as variable source}
        writestring(700+hpib_analyzer_address,vddstr);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,vddstr);
        writestring(700+hpib_analyzer_address,',1E-6;'); {set gate compliance 1ua}
        writestring(700+hpib_analyzer_address,'VC'); {set source as v-source}
        writestring(700+hpib_analyzer_address,vsnumb);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,vddstr);
        writestring(700+hpib_analyzer_address,',0.1;'); {set is compliance=100ma}
        writestring(700+hpib_analyzer_address,'VC');
        writestring(700+hpib_analyzer_address,vbnumb);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,vddstr);
        writestring(700+hpib_analyzer_address,',0.01;'); {set ib compliance=10ma}
end:  {end of source_setup_pchannel_device_functionality}
```

measure_device_functionality

```
procedure measure_device_functionality(var devtype,error:integer);
{this procedure sets up the 4145 to measure a single value of ids at the
bias conditions previously setup, and reads the value from the 4145;
{last change made: OCT 2, 1984     by: JRP}

var i:integer;

begin
      {this is the measurement and display mode setup}
      {the following line sets up wait and hold times for measurement}
      writestring(700+hpib_analyzer_address,'SM   DM3,MX  "IDRAIN";');
      writestring(700+hpib_analyzer_address,'IT1  DR0  BC');    {clears the buffer}
      {this is the measurement setup}
      writestring(700+hpib_analyzer_address,'MD  ME1;'); {make measurement}
      wait_till_hpib_ready(hpib_analyzer_address);
      writestring(700+hpib_analyzer_address,'DO  "IDRAIN";');  {output data}
      listen_to_hpib(hpib_analyzer_address);
      readnumber(700+hpib_analyzer_address,idsgzero);
      readnumber(700+hpib_analyzer_address,idsgvdd);
      if devtype=1 then    {this is a nchannel device}
            begin
                  if idsgvdd<(width/length)*1E-6 then
                        begin
                              devtype:=0;  {this tests for an open drain source}
                              error:=4;
                        end;
                  if idsgzero>0.95*idsgvdd then
                        begin
                              devtype:=0;  {this tests for a shorted drain source}
                              error:=5;
                        end;
            end;
      if devtype=-1 then  {this is a pchannel device}
            begin
                  if abs(idsgzero)<(width/length)*1E-6 then
                        begin
                              devtype:=0;  {this tests for an open drain source}
                              error:=4;
                        end;
                  if abs(idsgvdd)>0.8*abs(idsgzero) then
                        begin
                              devtype:=0;    {this tests for a shorted drain source}
                              error:=5;
                        end;
            end;
      talk_to_hpib(hpib_analyzer_address);
      bsim_timer(time_count);
```

```
end; {end of procedure measure_device_functionality }
```

measure_device_data

```
{jrp}{added variable to procedure}
procedure measure_device_data(var measure_device_data_error:integer);
{this procedure sets up the strings to be used for the sources. sets up
the sources. and measures the data for n or p channel devices}
{last change made: SUNDAY OCT 14. 1984  By: JRP}

type testcode=string[20];

var   vsource,vgstart,vgend,vgstep,vbstart,vbstep:real;
      numbvbstep,numbvgstep:integer;
      vdrain:array[1..4] of real;
      svsource,svgstart,svgend,svgstep,svbstart,snumbvbstep,svbstep,
      svdrain1,svdrain2,svdrain3,svdrain4:string[20];
      multiplier:real;  {to scale the IDS vs VGS curves on the HP4145.
                         multiply the maximum ids value be the multiplier.
                         for vds=0.1 and 0.2 values, multiplier=0.1. so
                         the curves are more visible}
```

channel_definition_for_IDSvsVGS_data

```
procedure channel_definition_for_IDSvsVGS_data;
{this procedure sets up the channels with all being voltage sources and the
source being common.  The gate voltage and body voltage are set up to vary}
{last change made: JUNE 20. 1984  by: JRP}

begin
     writestring(700+hpib_analyzer_address,'IT2 DR0 BC');  {medium integration time}
     writestring(700+hpib_analyzer_address,'DE ');
     writestring(700+hpib_analyzer_address,vdchan);   {sets up drain as const}
     writestring(700+hpib_analyzer_address,',''VDRAIN'',''IDRAIN'',1,3;');
     writestring(700+hpib_analyzer_address,vgchan); {sets up gate as variable1}
     writestring(700+hpib_analyzer_address,',''VGATE'',''IGATE'',1,1;');
     writestring(700+hpib_analyzer_address,vschan); {sets up source as const}
     writestring(700+hpib_analyzer_address,',''VSOURC'',''ISOURC'',1,3;');
     writestring(700+hpib_analyzer_address,vbchan); {sets up body as variable2}
     writestring(700+hpib_analyzer_address,',''VBODY'',''IBODY'',1,2;');
     writestring(700+hpib_analyzer_address,'VS1; VS2; VM1; VM2;');
end; {end of procedure to define channels for data aquisition}
```

string_setup_measure_IDSvsVGS

```
procedure string_setup_measure_IDSvsVGS;
{this procedure calculates step sizes for the measurements, fills in the vdd
array. and makes the necessary strings for the 4145}
{last change made: JUNE 20. 1984  by: JRP}

var   nextposition:integer;  {for strwrite command}
      i:integer; {used for loop counter}

begin
      bsim_timer(time_count);
      {determine the value of VSOURCE}
      if device_type=1 then
            vsource:=0.0
      else
            vsource:=vdd;
      {determine the starting value of VGATE}
      if device_type=1 then
            vgstart:=0.0
      else
            vgstart:=vdd;
```

```
{determine the final value of VGATE}
if device_type=1 then
        vgend:=vdd
else
        vgend:=0.0;
{determine the step size for VGATE}
if vdd <=3 then
        vgstep:=0.05
else
        if vdd <=5 then
                vgstep:=0.1
        else
                if vdd <=10 then
                        vgstep:=0.2
                else
                        vgstep:=0.5;
vgstep:=vgstep*device_type;
{determine the number of steps for VGATE}
numbvgstep:=trunc(vdd /vgstep*device_type)+1; {plus 1 for the first point}
{determine the starting value for VBODY}
if device_type=1 then
        vbstart:=-vdd
else
        vbstart:=2*vdd;
{determine the number of VBODY steps}
numbvbstep:=numbvb;
{determine the value of the VBODY step}
vbstep:=device_type*(0.01*trunc(vdd /(0.01*(numbvbstep-1))));
{determine the values of VDRAIN}
{*****************************************************************}
{**********HERE IS WHERE THE DRAIN SOURCE BIASES CAN BE CHANGED******}
{*****************************************************************}
vds[1]:=0.1;
vds[2]:=0.2;
vds[3]:=vdd-0.5;
vds[4]:=vdd;
if device_type=1 then
        begin
                vdrain[1]:=vds[1];
                vdrain[2]:=vds[2];
                vdrain[3]:=vds[3];
                vdrain[4]:=vds[4];
        end
else
        begin
                vdrain[1]:=vdd-vds[1];
                vdrain[2]:=vdd-vds[2];
                vdrain[3]:=vdd-vds[3];
                vdrain[4]:=vdd-vds[4];
        end;
{here is where the strings are written}
svsource:='                    ';
strwrite(svsource,1,nextpos,vsource:2:3);
svgstart:='                    ';
strwrite(svgstart,1,nextpos,vgstart:2:3);
svgend:='                    ';
strwrite(svgend,1,nextpos,vgend:2:3);
svgstep:='                    ';
strwrite(svgstep,1,nextpos,vgstep:2:3);
svbstart:='                    ';
strwrite(svbstart,1,nextpos,vbstart:2:3);
snumbvbstep:='                    ';
strwrite(snumbvbstep,1,nextpos,numbvbstep);
svbstep:='                    ';
strwrite(svbstep,1,nextpos,vbstep:2:3);
```

...string_setup_measure_IDSvsVGS

```
        svdrain1:='                    ';
        strwrite(svdrain1,1,nextpos,vdrain[1]:2:3);
        svdrain2:='                    ';
        strwrite(svdrain2,1,nextpos,vdrain[2]:2:3);
        svdrain3:='                    ';
        strwrite(svdrain3,1,nextpos,vdrain[3]:2:3);
        svdrain4:='                    ';
        strwrite(svdrain4,1,nextpos,vdrain[4]:2:3);
        {here is where the values of vbs are stored in the array vbs}
        {realize that for a pchannel device. vbs is negated}
        for i:=1 to numbvbstep do
              vbs[i]:=device_type*((vbstart+(i-1)*vbstep-vsource));
end;    {end of procedure string_setup_measure_IDSvsVGS}
```

source_setup_measure_IDSvsVGS

```
procedure source_setup_measure_IDSvsVGS(svdrain:testcon
{this procedure sets up the sources to measure the data}
{last change made: JUNE 20. 1984   by: JRP}

begin
        writestring(700+hpib_analyzer_address,'SS  ');
        writestring(700+hpib_analyzer_address,'VR1,');  {setup for gate source}
        writestring(700+hpib_analyzer_address,svgstart);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,svgend);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,svgstep);
        writestring(700+hpib_analyzer_address,',1E-6;');
        writestring(700+hpib_analyzer_address,'VP  ');  {setup for body source}
        writestring(700+hpib_analyzer_address,svbstart);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,svbstep);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,snumbvbstep);
        writestring(700+hpib_analyzer_address,',0.001;');
        writestring(700+hpib_analyzer_address,'VC');  {setup for drain v-source}
        writestring(700+hpib_analyzer_address,vdnumb);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,svdrain);
        writestring(700+hpib_analyzer_address,',100E-3;');
        writestring(700+hpib_analyzer_address,'VC');
        writestring(700+hpib_analyzer_address,vsnumb);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,svsource);
        writestring(700+hpib_analyzer_address,',100E-3;');
        bsim_timer(time_count);
end;    {end of procedure source_setup_measure_IDSvsVGS}
```

measure_and_reduce_IDSvsVGS_data

```
{jrp}{changed variable error to ids_vgs_array_error}
procedure measure_and_reduce_IDSvsVGS_data(vdstartinteger;
                                           var ids_vgs_array_error:integer);
{this procedure reads the data from the 4145 for either a nchannel or
a pchannel device. The values of vds,ids,vbs and vgs are converted to
correspond to the convention for nchannel devices}
{last change made: SUNDAY OCT 14. 1984   By: JRP}

var     i,j,k,l:integer;
        idrain:array[1..100] of real;
        vgstep_at_vth:integer;
        vg_store_step:integer;
        threshold:real;
        notyet:boolean;
```

```
        maxidsval:real;              {maximum of idsgzero or idsgvdd}
        sign:linestring;

begin
        bsim_timer(time_count);
        {writestring(700+hpib_analyzer_address,'SM DM3 MX ''IDRAIN'';');
{*****************************take out later*******************************}
        if TRACE23=true then writeln(userout2,
             'idsgzero=',idsgzero,'idsgvdd=',idsgvdd);
{*****************************take out later*******************************}
        if abs(idsgzero)>abs(idsgvdd) then
             maxidsval:=multiplier*abs(idsgzero)
        else
             maxidsval:=multiplier*abs(idsgvdd);
        if maxidsval>1E0 then
             maxidsstring:='0.75E1'
        else if maxidsval>1E-1 then
             maxidsstring:='0.75E0'
        else if maxidsval>1E-2 then
             maxidsstring:='0.75E-1'
        else if maxidsval>1E-3 then
             maxidsstring:='0.75E-2'
        else if maxidsval>1E-4 then
             maxidsstring:='0.75E-3'
        else if maxidsval>1E-5 then
             maxidsstring:='0.75E-4'
        else if maxidsval>1E-6 then
             maxidsstring:='0.75E-5'
        else if maxidsval>1E-7 then
             maxidsstring:='0.75E-6'
        else if maxidsval>1E-8 then
             maxidsstring:='0.75E-7'
        else if maxidsval>1E-9 then
             maxidsstring:='0.75E-8'
        else if maxidsval>1E-10 then
             maxidsstring:='0.75E-9'
        else maxidsstring:='0.75E-10';
        if device_type=-1 then                    {if pmos, then add minus sign to}
             begin                                  {maxidsstring value sent to hp4145}
                  sign:='-';
                  strappend(sign,maxidsstring);
                  maxidsstring:=sign;
             end;
{*****************************take out later*******************************}
        if TRACE23=true then writeln(userout2,
             'maxidsstring =',maxidsstring);
{*****************************take out later*******************************}
        writestring(700+hpib_analyzer_address,'SM DM1, XN''VGATE'',1,0,');
        writestring(700+hpib_analyzer_address,vddstr);
{JRP}if device_type=1 then
{JRP}     begin
                  writestring(700+hpib_analyzer_address,':YA''IDRAIN'',1,0,');
                  writestring(700+hpib_analyzer_address,maxidsstring);
                  writestring(700+hpib_analyzer_address,';');
{JRP}     end
{JRP}else   {device is PMOS}
{JRP}     begin
{JRP}          writestring(700+hpib_analyzer_address,';YA''IDRAIN'',1,');
{JRP}          writestring(700+hpib_analyzer_address,maxidsstring);
{JRP}          writestring(700+hpib_analyzer_address,',0;');
{JRP}     end;
        writestring(700+hpib_analyzer_address,'IT2 DR0 BC');   {medium integration time}
        writestring(700+hpib_analyzer_address,'MD ME1');
        wait_till_hpib_ready(hpib_analyzer_address);
        bsim_timer(time_count);
```

```
      writestring(700+hpib_analyzer_address,'DO ''IDRAIN'';');
      bsim_timer(time_count);
      listen_to_hpib(hpib_analyzer_address);
      threshold:=0.1E-6*width/length;  {this is the threshold current level}
{jrp}if TRACE37=true then writeln(userout2,
{jrp}     'threshold=',threshold,' width=',width:2:2,' length=',length:2:2);
{jrp}ids_vgs_array_error:=0;
      for j:=1 to numbvbstep do
{jrp}    if ids_vgs_array_error=0 then
               {this steps through the values of vbs from highest in magnitude to
               the lowest value of vbs in magnitude}
               begin
                    notyet:=true;
{jrp}               vgstep_at_vth:=numbvgstep:  {in case all values}
                                                {are below threshold}
                    for i:=1 to numbvgstep do
                      {this steps through the values of vgs from lowest in
                       magnitude to highest in magnitude}
                         begin
{jrp}                         readnumber(700+hpib_analyzer_address,idrain[i]);
                              if (abs(idrain[i])>threshold) and (notyet=true) then
                                   begin
                                        notyet:=false;
                                        vgstep_at_vth:=i+5;
                                   end;
                         end;
                    vg_store_step:=(numbvgstep-vgstep_at_vth) div (numbvg-1);
                    if vg_store_step=0 then
                         ids_vgs_array_error:=6;   {this means that there are not enough points}
                    if ids_vgs_array_error=0 then
                         begin
                              for i:=1 to numbvg do
                                   begin
                                        ids[vdscount,j,i]:=(idrain[vgstep_at_vth+
                                                                   (i-1)*vg_store_step])*
                                                                   device_type;
                                        vgs[vdscount,j,i]:=(vgstep_at_vth-1+
                                                                   (i-1)*vg_store_step)*
                                                                   vgstep*device_type;
                                   end;
                         end;
               end;
      bsim_timer(time_count);
      talk_to_hpib(hpib_analyzer_address);
      bsim_timer(time_count);
end;

begin  {begin procedure to measure_device_data}
      channel_definition_for_IDSvsVGS_data;
      string_setup_measure_IDSvsVGS;
      multiplier:=0.1;
      source_setup_measure_IDSvsVGS(svdrain1);
      measure_and_reduce_IDSvsVGS_data(1,measure_device_data_error);
{jrp}if TRACE38=true then writeln(userout2,'within measure_and_reduce and',
           ' measure_device_data_error=',measure_device_data_error:1);
{jrp}if measure_device_data_error=0 then
{jrp}    begin
               source_setup_measure_IDSvsVGS(svdrain2);
               measure_and_reduce_IDSvsVGS_data(2,measure_device_data_error);
{jrp}          if TRACE38=true then writeln(userout2,
                    'within measure_and_reduce and',' measure_device_data_error=',
                    measure_device_data_error:1);
               multiplier:=1.0;
```

```
{jrp}          if measure_device_data_error=0 then
{jrp}               begin
                         source_setup_measure_IDSvsVGS(svdrain3);
                         measure_and_reduce_IDSvsVGS_data(3,
                              measure_device_data_error);
{jrp}                    if TRACE38=true then writeln(userout2,
                              'within measure_and_reduce and',
                              ' measure_device_data_error=',
                              measure_device_data_error:1);
{jrp}                    if measure_device_data_error=0 then
{jrp}                         begin
                                   source_setup_measure_IDSvsVGS(svdrain4);
                                   measure_and_reduce_IDSvsVGS_data(4,
                                        measure_device_data_error);
{jrp}                              end;
{jrp}               end;
{jrp}     end;
{jrp}if TRACE38=true then writeln(userout2,'within measure_and_reduce and',
          ' measure_device_data_error=',measure_device_data_error:1);
end;


begin {begin procedure measure_device}
      translate_terminals_to_strings;
      error_code:=0;
      chan_definition_to_test_device_type;
      source_setup_to_test_device_type;
      measure_device_type(device_type,error_code);
      if device_type=-1 then    {this is pchannel device test}
           begin
                chan_definition_device_functionality;
                source_setup_pchannel_device_functionality;
                measure_device_functionality(device_type,error_code);
           end;
      if device_type=1 then  {this is the nchannel device test}
           begin
                chan_definition_device_functionality;
                source_setup_nchannel_device_functionality;
                measure_device_functionality(device_type,error_code);
           end;
      if error_code=0 then
{jrp}      begin
{jrp}           measure_device_data(error_code);
{jrp}           if error_code=6 then  {error_code might become non-0}
{jrp}                begin
{jrp}                     gotoxy(0,22);
{jrp}                     write('NO PARAMETERS HAVE BEEN GENERATED OR SAVED');
{jrp}                     write(': IDS-VGS ARRAY HAS A ZERO IN IT');
{jrp}                end;
{jrp}      end
      else
           begin
                gotoxy(47,6);
                case error_code of
                     1:write(' ERROR **GATE SHORT**');
                     2:write(' ERROR *SHORTED JUNCTION*');
                     3:write(' ERROR **NO JUNCTION**');
                     4:begin
                            gotoxy(55,6);
                            write(' ERROR*OPEN DRAIN-SOURCE*');
                       end;
                     5:begin
                            gotoxy(55,6);
                            write(' ERROR*SHORTED DRAIN-SRC*');
                       end;
```

...measure_device

```
            end;  {end of case}
        end;
end;


{*********************************************************************}
{*******************BSIM EXTRACTION ROUTINES**************************}
{*********************************************************************}

procedure extract_device_parameters;                    extract_device_parameters
{this procedure extracts all of the pertinent device parameters}
{last change made: SUNDAY OCT 14, 1984  By: JRP}

var   u0,beta0,vth,u1,a,eta:array[1..4,1..6] of real;
      vt0,azerovds,u0zerovds:array[1..6] of real;
      k1,k2,vfb,phif2:array[1..4] of real;  {for linear region analysis}
      i,j,vdindex,vbindex,vgindex:integer;  {counters for extraction}
      {*******************************take out later*********************}
      iii,jjj,kkk:integer;  {used to print out all data values}
      {*******************************take out later*********************}


{************************LINEAR REGION EXTRACTION************************}

procedure linear_region_extraction;                     linear_region_extraction
{this procedure calculates the parameters VTX, SLOPE, and U0 using a first
pass to extract initial estimates, and newton-raphson iterations to derive
the final values at various biases}

const number_small_vgs_values=3;
      maximvar=1E-8;

var   vgsval:real;    {present value of vgs}
      nrindex:integer;  {present index for newton-raphson iteration}
      constant,linear,quadratic:real;
      idsdivvds:real;
      vgindex:integer;
      iterate:boolean;
      vgsminvtx:real;
      linear1,linear2:real;
      vtx,slope,u0x:real;

begin
      {*************************take out later*************************}
      if TRACE37=true then writeln(userout2,'entering linear_region_extraction');
      {*************************take out later*************************}
      for vgindex:=1 to number_small_vgs_values do
            begin
                  if TRACE37=true then lsq3:=true;
                  vgsval:=vgs[vdindex,vbindex,vgindex];
                  idsdivvds:=ids[vdindex,vbindex,vgindex]/vds[vdindex];
                  leastsq3(vgindex,1,number_small_vgs_values,vgsval,vgsval*vgsval,
                  idsdivvds,constant,linear,quadratic);
                  if TRACE37=true then lsq3:=false;
{jrp}             if TRACE37=true then writeln(userout2,'in 1st leastsq3 &',
{jrp}                   ' vgindex=',vgindex:1);
            end;
      vtx:=(-2.0*constant)/(linear+linear*sqr(1-(4*constant*quadratic)/
            (linear*linear)));
      slope:=linear+2*quadratic*vtx;
{jrp}if TRACE37=true then writeln(userout2,'after 1st leastsq3 & vtx=',
```

..linear_region_extraction

```
{jrp}      vtx,' slope=',slope);
      {this is the fit for (vgs-vtx)/g=1/slope+u0x/slope*(vgs-vtx) which
      is used to get an initial estimate for u0x}
      for vgindex:=(number_small_vgs_values+1) to numbvg do
            begin
                  vgsminvtx:=vgs[vdindex,vbindex,vgindex]-vtx;
                  leastsq2(vgindex,number_small_vgs_values+1,numbvg,vgsminvtx,
                        vgsminvtx/ids[vdindex,vbindex,vgindex]*vds[vdindex],
                        constant,linear);
            end;
      u0x:=linear/constant;
{jrp}if TRACE37=true then writeln(userout2,'after 2nd leastsq3 & u0x=',u0x);
      {this is the final fit for g=slope*(vgs-vtx)/(1+u0x*(vgs-vtx))}
      iterate:=true;
      nrindex:=1;
      while ((nrindex<15) and (iterate=true)) do
        begin
            for vgindex:=1 to numbvg do
                  begin
                        vgsminvtx:=vgs[vdindex,vbindex,vgindex]-vtx;
                        idsdivvds:=ids[vdindex,vbindex,vgindex]/vds[vdindex];
                        leastsq3(vgindex,1,numbvg,(u0x*idsdivvds-slope)/vgsminvtx,
                              -idsdivvds,idsdivvds/vgsminvtx+idsdivvds*u0x-slope,
                              constant,linear1,linear2);
                  end;
            if (u0x+linear2)<0 then
                  u0x:=u0x/10
            else
                  u0x:=u0x+linear2;
            slope:=slope+constant;
            vtx:=vtx+linear1;
            if (abs(constant/slope)<maximvar) and (abs(linear1)<maximvar) and
                  (abs(linear2/u0x)<maximvar) then
                        iterate:=false;
            nrindex:=nrindex+1;
{jrp}if TRACE37=true then writeln(userout2,'nrindex',
{jrp}      nrindex:2,' vtx=',vtx,' slope=',slope,' u0x=',u0x);
          end; {end of newton-raphson iteration}
      beta0[vdindex,vbindex]:=slope;
      vth[vdindex,vbindex]:=vtx;
      u0[vdindex,vbindex]:=u0x;
      {*************************take out later*************************}
      if TRACE37=true then writeln(userout2,'leaving linear_region_extraction');
      {*************************take out later*************************}
end;
```

large_device_20f_extraction

```
procedure large_device_20f_extraction;
{this procedure is called to analyze the first large device of the current
 die, and for each different device type, in the data structure,
 to determine 20f, K1, K2, and VFB.  this procedure NEVER gets called for
 SINGLE mode}
{last change made: THURSDAY SEPT 20, 1984   By: JRP}

const numbvb=6;
      maximvar=1E-8;

var   q_e_2,k_t_q:real;
      s,g,t,a,Na,ni:real;
      vp,sqroot_vp:real;
      constant,linear1:real;
      iterate:boolean;
      nrindex,vbindex:integer;
```

```
        temp1,temp2:real;
        test:integer;

begin
        {************************take out later**********************}
        if TRACE37=true then writeln(userout2,'entering large_device_20f_extraction');
        {************************take out later**********************}
        phif2[vdindex]:=0.6;
        k_t_q:=0.0258512*(temp/300.0);
        ni:=1.45E10*exp(1.5*ln(temp/300))*exp(1.124/2.0/k_t_q*(1-300/temp));
        q_e_2:=3.20438E-19*11.7*8.854E-14;
        k2[vdindex]:=0;
        {HERE IS WHERE s is calculated}
        {for the first die tested. s is set to 1}
        {all further dies use the process parameters for k1 on the previous
        die to determine s}
        test:=2*device+enhancement_depletion;
        if ( (present_die=1) or (first_good_die=1) ) then
                s:=1.0
        else
                case test of
                        -3:s:=1+pdlk1/(length+pdd1)+pdwk1/(width+pddw);    {pchan dep.}
                        -2:s:=1+pzlk1/(length+pzd1)+pzwk1/(width+pzdw);    {pchan zer.}
                        -1:s:=1+pelk1/(length+ped1)+pewk1/(width+pedw);    {pchan enh.}
                        1:s:=1+ndlk1/(length+ndd1)+ndwk1/(width+nddw);     {nchan dep.}
                        2:s:=1+nzlk1/(length+nzd1)+nzwk1/(width+nzdw);     {nchan zer.}
                        3:s:=1+nelk1/(length+ned1)+newk1/(width+nedw);     {nchan enh.}
                end;  {end of case statement to determine s}
        Na:=ni*exp(phif2[vdindex]/2.0/k_t_q);
        k1[vdindex]:=s*sqrt(q_e_2*Na)/cox;
        {the newton-raphson iteration begins here}
        nrindex:=1;
        iterate:=true;
        while ((nrindex<15) and (iterate=true)) do
                begin
{jrp}               if TRACE37=true then writeln(userout2,'vdindex',vdindex:1,
{jrp}                       ' nrindex=',nrindex:2);
                    for vbindex:=1 to numbvb do
                        begin
                            vp:=phif2[vdindex]-vbs[vbindex];
                            sqroot_vp:=sqrt(vp);
                            t:=1.744+0.8364*vp;
                            g:=1-1/t;
                            a:=1+g/2*k1[vdindex]/sqroot_vp;
                            temp1:=1-k2[vdindex]+k1[vdindex]/2/sqroot_vp
                            +k1[vdindex]/4*sqroot_vp/k_t_q+
                            vds[vdindex]/2*(-g/4*k1[vdindex]/
                            vp/sqroot_vp+k1[vdindex]/2/sqroot_vp/t/t*0.8364);
                            temp2:=vth[vdindex,vbindex]-a/2*vds[vdindex]
                            -phif2[vdindex]-k1[vdindex]*sqroot_vp;
                            leastsq3(vbindex,1,numbvb,temp1,-vp,temp2
                            ,vfb[vdindex],linear1,k2[vdindex]);
{jrp}                       if TRACE37=true then writeln(userout2,
{jrp}                           ' vb=',vbindex,' vfb[',vdindex:1,']=',
{jrp}                           vfb[vdindex],' k2[',vdindex:1,']=',k2[vdindex]);
                        end;
                    phif2[vdindex]:=phif2[vdindex]+linear1;
                    if phif2[vdindex]<0.2 then
                        phif2[vdindex]:=0.2;    {safety check}
                    Na:=ni*exp(phif2[vdindex]/2/k_t_q);
                    if Na<ni then
                        Na:=ni;  {guarantees that phif2 does not go negative}
                    k1[vdindex]:=s*sqrt(q_e_2*Na)/cox;
                    if abs(linear1)<maximvar then
                        iterate:=false;
```

...large_device_20f_extraction

```
                nrindex:=nrindex+1;
{jrp}           if TRACE37=true then writeln(userout2,
{jrp}                 'phif2[',vdindex:1,']=',phif2[vdindex],' k1[',vdindex:1,
{jrp}                 ']=',k1[vdindex]);
            end;    {end of newton-raphson iteration while loop}
            bsim_timer(time_count);
        {*************************take out later*************************}
        if TRACE37=true then writeln(userout2,'leaving large_device_20f_extraction');
        {*************************take out later*************************}
end:    {of procedure large_device_20f_extraction}
```

linear_region_threshold_analysis

```
procedure linear_region_threshold_analysis;
{this procedure extracts arrays for vfb,k1, and k2.  assuming that the
value of 20f is already known}
{last change made JUNE 18, 1984   by: JRP}

var   t,g,vp,sqroot_vp,vx:real;
      vbindex:integer;
      k_t_q,ni,q_e_2,s,Na:real;
      test:integer;

begin
      {*************************take out later*************************}
      if TRACE37=true then writeln(userout2,
          'entering linear_region_threshold_analysis');
      {*************************take out later*************************}
      test:=2*device+enhancement_depletion;
      {here is where the value of phif2 must be determined}
      if mode='4' then    {this is single device mode, and value has been input}
            phif2[vdindex]:=c2_phif2
      else
            case test of
                  -3:phif2[vdindex]:=pdphif2;   {pchannel depletion}
                  -2:phif2[vdindex]:=pzphif2;   {pchannel zero}
                  -1:phif2[vdindex]:=pephif2;   {pchannel enhancement}
                  1:phif2[vdindex]:=ndphif2;    {nchannel depletion}
                  2:phif2[vdindex]:=nzphif2;    {nchannel zero}
                  3:phif2[vdindex]:=nephif2;    {nchannel enhancement}
            end;  {end of case to determine phif2 values}
      k_t_q:=0.0258512*(temp/300.0);
      ni:=1.45E10*exp(1.5*ln(temp/300))*exp(1.124/2.0/k_t_q*(1-300/temp));
      q_e_2:=3.20438E-19*11.7*8.854E-14;
      {is only has to be an approximation here}
      s:=1;
      Na:=ni*exp(phif2[vdindex]/2.0/k_t_q);
      k1[vdindex]:=s*sqrt(q_e_2*Na)/cox;
      for vbindex:=1 to numbvb do
            begin
                  vp:=phif2[vdindex]-vbs[vbindex];
                  sqroot_vp:=sqrt(vp);
                  t:=1.744+0.8364*vp;
                  g:=1-1/t;
                  a[vdindex,vbindex]:=1+g/2*k1[vdindex]/sqroot_vp;
                  leastsq3(vbindex,1,numbvb,vds[vdindex]/2*g/2/sqroot_vp+
                              sqroot_vp,-vp,vth[vdindex,vbindex]-phif2[vdindex]-
                              a[vdindex,vbindex]/2*vds[vdindex],vfb[vdindex]
                              ,k1[vdindex],k2[vdindex]);
{jrp}                         if TRACE37=true then writeln(userout2,
{jrp}                               ' vb=',vbindex,' vfb[',vdindex:1,']=',
{jrp}                               vfb[vdindex],' k2[',vdindex:1,']=',k2[vdindex],
```

```
{jrp}                                  ' k1[',vdindex:1,']=',k1[vdindex]);
            end;
{jrp}if TRACE37=true then writeln(userout2,
{jrp}       'phif2[',vdindex:1,']=',phif2[vdindex]);
      bsim_timer(time_count);
      {************************take out later************************}
      if TRACE37=true then writeln(userout2,
            'leaving linear_region_threshold_analysis');
      {************************take out later************************}
end;    {end of the linear_region threshold analysis}
```

linear_region_data_reduction

```
procedure linear_region_data_reduction;
{this procedure reduces the data for the linear region variables
u0,vfb,20fk1,k2,vt0, and azerovds}
{last change made: THURSDAY SEPT 20, 1984   By: JRP}

var   vdindex,vbindex:integer;
      x3u0:real;
      linear1:real;
      vp,sqroot_vp,g,t,vx:real;
      test:integer;

begin
      {************************take out later************************}
      if TRACE21=true then writeln(userout2,
            'entering linear_region_data_reduction');
      {************************take out later************************}
      {clean up arrays}
      for vdindex:=1 to number_small_vds_values do
            for vbindex:=1 to numbvb do
                  begin
                        vp:=phif2[vdindex]-vbs[vbindex];
                        sqroot_vp:=sqrt(vp);
                        t:=1.744+0.8364*vp;
                        g:=1-1/t;
                        a[vdindex,vbindex]:=1+g/2*k1[vdindex]/sqroot_vp;
                        vx:=a[vdindex,vbindex]/2*vds[vdindex];
                        vth[vdindex,vbindex]:=vth[vdindex,vbindex]-vx;
                        u0[vdindex,vbindex]:=u0[vdindex,vbindex]/
                                          (1-u0[vdindex,vbindex]*vx);
                  end;
      {reduction of u0 to zero vbs values}
      for vbindex:=1 to numbvb do
            for vdindex:=1 to number_small_vds_values do
                  leastsq3(2*(vbindex-1)+vdindex,1,number_small_vds_values*numbvb,
                                    vbs[vbindex],vds[vdindex],u0[vdindex,vbindex],
                                    c7_u0,c12_x2u0,x3u0);
      gotoxy(3,20);
      write(c7_u0:1:3);
      gotoxy(45,13);
      write(c12_x2u0:1:6);
      for vbindex:=1 to numbvb do
            u0zerovds[vbindex]:=c7_u0+c12_x2u0*vbs[vbindex];
      for vdindex:=1 to number_small_vds_values do
            leastsq2(vdindex,1,number_small_vds_values,vds[vdindex],
                        vfb[vdindex],c1_vfb,linear1);
      gotoxy(4,14);
      write(c1_vfb:1:3);
      if (meas_20f=TRUE) or (fake_meas_20f=TRUE) then
            for vdindex:=1 to number_small_vds_values do
                  leastsq2(vdindex,1,number_small_vds_values,vds[vdindex],
                        phif2[vdindex],c2_phif2,linear1);
```

```
        {here is where the value of phif2 is stored for later devices}
        {if phif2 was not measured on the present device, the phif2 value
        stored will be the same as for the previous device}
        test:=2*device+enhancement_depletion;
        case test of
              -3:pdphif2:=c2_phif2;
              -2:pzphif2:=c2_phif2;
              -1:pephif2:=c2_phif2;
               1:ndphif2:=c2_phif2;
               2:nzphif2:=c2_phif2;
               3:nephif2:=c2_phif2;
        end:    {end of case to store away value of phif2}
        gotoxy(6,15);
        write(c2_phif2:1:3);
        for vdindex:=1 to number_small_vds_values do
              leastsq2(vdindex,1,number_small_vds_values,vds[vdindex],k1[vdindex],
                    c3_k1,linear1);
        gotoxy(3,16);
        write(c3_k1:1:3);
        for vdindex:=1 to number_small_vds_values do
              leastsq2(vdindex,1,number_small_vds_values,vds[vdindex],k2[vdindex],
                    c4_k2,linear1);
        gotoxy(3,17);
        write(c4_k2:1:3);
        for vbindex:=1 to numbvb do
              begin
                    if TRACE23=true then writeln(userout2,
                          'linear--vto array and vbindex=',vbindex);
                    if TRACE23=true then writeln(userout2,
                          'vbs[vbindex]=',vbs[vbindex],' and c2_phif2=',c2_phif2);
                    vp:=c2_phif2-vbs[vbindex];
                    sqroot_vp:=sqrt(vp);
                    vt0[vbindex]:=c1_vfb+c2_phif2+c3_k1*sqroot_vp-c4_k2*vp;
                    t:=1.744+0.8364*vp;
                    g:=1-1/t;
                    azerovds[vbindex]:=1.0+g/2.0*(c3_k1/sqrt(vp));
              end;
        for vdindex:=1 to number_small_vds_values do
              for vbindex:=1 to numbvb do
                    eta[vdindex,vbindex]:=(vth[vdindex,vbindex]-vt0[vbindex])/
                                                vds[vdindex];
        bsim_timer(time_count);
        {*************************take out later*************************}
        if TRACE21=true then writeln(userout2,
              'leaving linear_region_data_reduction');
        {*************************take out later*************************}
  end;

{jrp}{the entire following procedure is new}
procedure check_linear_parameters_validity(var linear_parameter_error:integer);
{this procedure checks each of the just-extracted linear  parameters to see if
 they are within a reasonable range of values.  if they are not, then the
 extraction will cease here, and the parameters will not be used when computing
 the process file.  a check of the saturation region parameters (and the
 linear parameters again) will be conducted in process_file_development}
{last change made: THURSDAY NOV 8, 1984  By: JRP}

begin
     linear_parameter_error:=0;
{JRP}if not( (c1_vfb >= -5.0) and (c1_vfb <= 1.0) and
               (c2_phif2 >= 0.2) and (c2_phif2 <= 1.5) and
               (c3_k1 >= 0.0) and (c3_k1 <= 5.0) and
```

..check_linear_parameters_validity

...check_linear_parameters_validity

```
{JRP}          (c4_k2 >= -1.0) and (c4_k2 <= 1.0) and
{JRP}          (c7_u0 >= -1.0) and (c7_u0 <= 1.0) and
               (c12_x2u0 >= -1.0) and (c12_x2u0 <= 1.0) ) then
     linear_parameter_error:=1;
end;
```

```
{************************SATURATION REGION*****************************}
```

initial_saturation_region_estimates

```
procedure initial_saturation_region_estimates(var beta0sat,u1sat,vthsat:real);
{this procedure handles the estimates for u1beta0, and vtl. for
 the saturation region analysis}
{last change made: THURSDAY SEPT 20. 1984   By: JRP}

var   etasat:real;

begin  {begin procedure initial saturation region estimates}
 {*******************************take out later************************}
      if TRACE25=true then writeln(userout2,
           'entering initial saturation region estimates');
 {*******************************take out later************************}
      if (vbindex=numbvb) and (vdindex=numbvd) then
           begin
                u1sat:=0.0;
                vthsat:=vt0[vbindex]-eta[2,vbindex]*vds[vdindex];
                beta0sat:=beta0[2,vbindex];
           end
      else if (vbindex=numbvb) and (vdindex<numbvd) then
           begin
                {use values for previous vdindex}
                etasat:=eta[vdindex+1,vbindex];
                u1sat:=u1[vdindex+1,vbindex];
                beta0sat:=beta0[vdindex+1,vbindex];
                vthsat:=vt0[vbindex]-etasat*vds[vdindex];
           end
      else {(vbindex<numbvb)}
           begin
                {use values from previous vbindex}
                etasat:=eta[vdindex,vbindex+1];
                u1sat:=u1[vdindex,vbindex+1];
                beta0sat:=beta0[vdindex,vbindex+1];
                vthsat:=vt0[vbindex]-etasat*vds[vdindex];
           end;
 {*******************************take out later************************}
      if TRACE25=true then writeln(userout2,
           'leaving intial saturation region estimates');
 {*******************************take out later************************}
end;  {end of procedure initial saturation region estimates}
```

saturation_region_data_extraction

```
{jrp}{next line}
procedure saturation_region_data_extraction(
                                        var saturation_extract_error:integer);
{this procedure extracts the parameters beta0sat, eta, and u1 from
 the data taken over large vds values}
{last change made: SUNDAY OCT 14. 1984   By: JRP}

var   nrindex,vgindex:integer;
      iterate:boolean;
      beta0sat,u1sat,vthsat,idssim,deltabeta0sat,
      deltavth,deltau1:real;
      didsdbeta0,dkdvth,dkdu1,didsdvth,didsdu1,dudvth:real;
      vgsminvth,u,vc,t,k,r,s,asat,u0sat,vdssat:real;
      temp1,temp2,temp3:array[1..6] of real;
```

```
    numbon,numblin,numbsat,goodcount:integer;
{*******************************take out later*************************}
    place:integer; {used for debugging newton-raphson loop}
{*******************************take out later*************************}
```

zero_u1_saturation_extraction

```
procedure zero_u1_saturation_extraction;
{this procedure is called if u1 is going negative.  In this case, u1 is
set to zero}
{last change made: THURSDAY SEPT 20, 1984  By: JRP}

var   nrindex,vgindex:integer;
      iterate:boolean;
      numbon,goodcount:integer;
      temp1,temp2:array[1..6] of real;
      {most variables are used as global variables within the routine
      saturation_region_data_extraction}

begin
{*******************************take out later*************************}
      if TRACE25=true then writeln(userout2,
            'entering zero_u1_saturation_extraction');
{*******************************take out later*************************}
      nrindex:=1;
      iterate:=true;
      while (nrindex<12) and (iterate=true) and (leastsq_divide_by_zero=false) do
            begin
                  numbon:=0;
                  for vgindex:=1 to numbvg do
                        begin
                              vgsminvth:=vgs[vdindex,vbindex,vgindex]-vthsat;
                              if vgsminvth<0.0 then
                                    vgsminvth:=0.0;
                              u:=1.0+u0sat*vgsminvth;
                              vdssat:=vgsminvth/asat;
                              if vds[vdindex]>vdssat then
                                    {this is saturation analysis}
                                    begin
                                          s:=beta0sat/u/asat;
                                          idssim:=s/2.0*sqr(vgsminvth);
                                          {this next line has been changed from
                                          original program.  (u+1)/2/u has been
                                          added to give exact derivative}
                                          didsdvth:=-(s*vgsminvth)*(u+1)/2/u;
                                    end
                              else
                                    {this is the linear region analysis}
                                    begin
                                          s:=beta0sat/u*vds[vdindex];
                                          idssim:=s*(vgsminvth-asat/2.0*
                                          vds[vdindex]);
                                          didsdvth:=-(s);
                                    end;
                              didsdbeta0:=idssim/beta0sat;
                              if vgsminvth>0 then
                                    begin
                                          numbon:=numbon+1;
                                          temp1[numbon]:=didsdvth/didsdbeta0;
                                          temp2[numbon]:=(ids[vdindex,vbindex,
                                                vgindex]-idssim)/didsdbeta0
                                    end;
                        end;
                  for goodcount:=1 to numbon do
                              leastsq2(goodcount,1,numbon,temp1[goodcount],
                                    temp2[goodcount],deltabeta0sat,deltavth);
                  beta0sat:=beta0sat+deltabeta0sat;
```

```
                vthsat:=vthsat+deltavth;
                if (abs(deltabeta0sat/beta0sat)+abs(deltavth))<1E-8 then
                    iterate:=false;
                nrindex:=nrindex+1;
            end;  {end of while loop}
 {*********************************take out later*************************}
      if TRACE25=true then writeln(userout2,
          'leaving zero_u1_saturation_extraction');
 {*********************************take out later*************************}
end:   {end of procedure zero_u1_saturation_extraction}


{procedure saturation_region_data_extraction begins here}
begin
 {*********************************take out later*************************}
      if TRACE21=true then writeln(userout2,
          'entering sat_reg_data_extrac vbindex=',vbindex:2,' & vdindex=',vdindex:2);
{*********************************take out later*************************}
{jrp}saturation_extract_error:=0;
      u0sat:=u0zerovds[vbindex];
      asat:=azerovds[vbindex];
      initial_saturation_region_estimates(beta0sat,u1sat,vthsat);
 {*********************************take out later*************************}
      place:=0;
      if TRACE25=true then writeln(userout2,
          're-entering intial saturation region estimates ');
 {*********************************take out later*************************}
{this section fits the curve (Id-Ids)=dIdsdBeta0*x+dIdsdVth*x2+dIdsdU1*x3
 where y=deltaBeta0 x2=deltaVth and x3=deltaU1}
      iterate:=true;
      nrindex:=1;    {newton-raphson index}
      while (nrindex<13) and (iterate=true) and (leastsq_divide_by_zero=false)
                          and (saturation_extract_error=0) do
        begin
            if TRACE24=true then
                write(userout2,
                    'nrindex=',nrindex:2,' place=');
          numbon:=0;
          numblin:=0;
          numbsat:=0;
          for vgindex:=1 to numbvg do
            begin
              vgsminvth:=vgs[vdindex,vbindex,vgindex]-vthsat;
              if vgsminvth<0 then
                  vgsminvth:=0.0;
              u:=1.0+u0sat*vgsminvth;
              vc:=u1sat/asat*vgsminvth;
              t:=sqrt(1.0+2.0*vc);
              k:=(1.0+vc+t)/2.0;
              dudvth:=-u0sat;
              vdssat:=vgsminvth/asat/sqrt(k);
              if vds[vdindex]>vdssat then
                  {this is saturation region}
                  begin
                      if vgsminvth<>0.0 then
                          numbsat:=numbsat+1;
                      s:=beta0sat/u/asat/k;
                      idssim:=s/2.0*sqr(vgsminvth);
                      t:=0.5+0.5/t;
                      dkdvth:=-u1sat/asat*t;
                      dkdu1:=vgsminvth/asat*t;
                      didsdvth:=-(s*vgsminvth+idssim/u*dudvth+idssim/k*
                                  dkdvth);
                      didsdu1:=-idssim/k*dkdu1;
                  end
```

```
            else
                {this is the linear region}
                begin
                    if vgsminvth<>0.0 then
                        numblin:=numblin+1;
                    r:=1.0+u1sat*vds[vdindex];
                    s:=beta0sat/u/r*vds[vdindex];
                    idssim:=s*(vgsminvth-asat/2.0*vds[vdindex]);
                    didsdvth:=-(s+idssim/u*dudvth);
                    didsdu1:=-idssim/r*vds[vdindex];
                end;
            didsdbeta0:=idssim/beta0sat;
            if vgsminvth>0 then
                begin
                    numbon:=numbon+1;
                    temp1[numbon]:=didsdvth/didsdbeta0;
                    temp2[numbon]:=didsdu1/didsdbeta0;
                    temp3[numbon]:=(ids[vdindex,vbindex,vgindex]
                            -idssim)/didsdbeta0;
                end;
        end;
    place:=1;
    if TRACE24=true then writeln(userout2,place:2);
    if TRACE27=true then lsq3:=true;
    if numbsat>numblin then
        begin
            if TRACE24=true then
                writeln(userout2,'numbsat>numblin and numbon=',numbon);
            for goodcount:=1 to numbon do
{jrp}           if (numbon<>2) then
                    leastsq3(goodcount,1,numbon,temp1[goodcount],
                            temp2[goodcount],temp3[goodcount],
{jrp}                       deltabeta0sat,deltavth,deltau1)
{jrp}               else
{jrp}                   {number of sets of data sent to leastsq3=2 which
                        will not give back valid data...leastsq3 needs
                        at least 3 sets of data}
{jrp}                   saturation_extract_error:=2;
        end
    else
        begin
            if TRACE24=true then writeln(userout2,
                'numbsat<numblin & numbon=',numbon,' numbsat=',numbsat);
            for goodcount:=numbsat+1 to numbon do
{jrp}           if (numbon-numbsat-1<>2) then
                    leastsq3(goodcount,1,numbon,temp1[goodcount],
                            temp2[goodcount],temp3[goodcount],
{jrp}                       deltabeta0sat,deltavth,deltau1)
{jrp}               else
{jrp}                   {number of sets of data sent to leastsq3=2 which
                        will not give back valid data...leastsq3 needs
                        at least 3 sets of data}
{jrp}                   saturation_extract_error:=2;
        end;
    if TRACE27=true then lsq3:=false;
    if (saturation_extract_error=0) then
        begin
            place:=2;
            if TRACE24=true then write(userout2,place:2);
            if (u1sat+deltau1)<0 then
                u1sat:=u1sat/10
            else
                u1sat:=u1sat+deltau1;
            place:=3;
            if TRACE24=true then write(userout2,place:2);
```

```
                    if vdindex<numbvd then
                        begin
                            if (beta0sat+deltabeta0sat)>beta0[vdindex+1,vbindex] then
                                beta0sat:=beta0[vdindex+1,vbindex]
                            else
                                beta0sat:=beta0sat+deltabeta0sat;
                        end
                    else
                        beta0sat:=beta0sat+deltabeta0sat;
                    place:=4;
                    if TRACE24=true then write(userout2,place:2);
                    vthsat:=vthsat+deltavth;
                    if (abs(deltabeta0sat/beta0sat)+abs(deltavth))<1E-8 then
                        iterate:=false;
                    place:=5;
                    if TRACE24=true then write(userout2,place:2);
                    if u1sat<1E-7 then
                        u1sat:=0.0;
                    place:=6;
                    if TRACE24=true then writeln(userout2,place:2);
                    if (u1sat<1E-7) and ((nrindex>6) or (iterate=false)) then
                        begin
                            iterate:=false;
                            zero_u1_saturation_extraction;
                        end;
                    nrindex:=nrindex+1;
                end;  {of 'if (extract_error=0) then.....}
            end:  {end of newton-raphson while loop}
                vth[vdindex,vbindex]:=vthsat;
                beta0[vdindex,vbindex]:=beta0sat;
                u1[vdindex,vbindex]:=u1sat;
                eta[vdindex,vbindex]:=(vt0[vbindex]-vthsat)/vds[vdindex];
 {*********************************take out later*************************}
      if TRACE21=true then writeln(userout2,
          'leaving saturation region data extraction');
 {*********************************take out later*************************}
end;  {end of procedure saturation_region_data_extraction}
```

saturation_region_data_reduction

```
procedure saturation_region_data_reduction;
{this procedure reduces the arrays for u1,eta and beta}
{last change made: THURSDAY SEPT 20, 1984  By: JRP}

var   vbindex,vgindex,vdindex:integer;
      x3beta0,u1lin:real;
{jrp} {variable count has been deleted and replaced with calculated value
        using vdindex and vbindex}
begin
 {*********************************take out later*************************}
      if TRACE21=true then writeln(userout2,
          'entering saturation region data reduction');
 {*********************************take out later*************************}
      {this is where the array for u1 is reduced}
      for vbindex:=1 to numbvb do
            for vdindex:=numbvd+1-number_large_vds_values to numbvd do
                leastsq3(2*(vbindex-1)+vdindex-2,1,
                    numbvb*(number_large_vds_values),vbs[vbindex],
                    vds[vdindex]-vdd,u1[vdindex,vbindex],c8_u1,c13_x2u1,c17_x3u1);
      gotoxy(3,21);
      write(c8_u1:1:3);
      gotoxy(45,14);
      write(c13_x2u1:1:6);
      gotoxy(45,15);
      write(c17_x3u1:1:6);
```

```
    {this is where the eta array is reduced}
    for vbindex:=1 to numbvb do
        for vdindex:=numbvd+1-number_large_vds_values to numbvd do
            leastsq3(2*(vbindex-1)+vdindex-2,1,
                    numbvb*(number_large_vds_values),c2_phif2-vbs[vbindex],
                    vds[vdindex]-vdd,eta[vdindex,vbindex],
                    c5_eta,c10_x2eta,c11_x3eta);
    gotoxy(4,18);
    write(c5_eta:1:3);
    gotoxy(46,17);
    write(c10_x2eta:1:6);
    gotoxy(46,18);
    write(c11_x3eta:1:6);
    {this is where the beta0 array is corrected for u1}
    for vdindex:=1 to number_small_vds_values do
        for vbindex:=1 to numbvb do
            begin
                u1lin:=c8_u1+c13_x2u1*vbs[vbindex]+c17_x3u1*
                        (vds[vdindex]-vdd);
                if u1lin<0.0 then
                    u1lin:=0.0;
                beta0[vdindex,vbindex]:=beta0[vdindex,vbindex]*
                                        (1+u1lin*vds[vdindex]);
            end;
    {this is where the beta0 array is modeled for the linear region}
    for vbindex:=1 to numbvb do
        for vdindex:=1 to number_small_vds_values do
            leastsq3(2*(vbindex-1)+vdindex,1,numbvb*number_small_vds_values,
                vbs[vbindex],vds[vdindex],beta0[vdindex,vbindex],
                c6_beta0,c9_x2beta0,x3beta0);
    gotoxy(6,19);
    write(c6_beta0:1:6);
    gotoxy(48,16);
    write(c9_x2beta0:1:6);
    {this is where beta0sat is modeled}
    for vbindex:=1 to numbvb do
        for vdindex:=numbvd+1-number_large_vds_values to  numbvd do
            leastsq3(2*(vbindex-1)+vdindex-2,1,number_large_vds_values*numbvb,
                    vbs[vbindex],vds[vdindex]-vdd,beta0[vdindex,vbindex],
                    c14_beta0sat,c15_x2beta0sat,c16_x3beta0sat);
    gotoxy(49,19);
    write(c14_beta0sat:1:6);
    gotoxy(51,20);
    write(c15_x2beta0sat:1:6);
    gotoxy(51,21);
    write(c16_x3beta0sat:1:6);
  {**********************************take out later**************************}
    if TRACE21=true then writeln(userout2,
        'leaving saturation region data reduction');
  {**********************************take out later**************************}
end;    {end of procedure saturation_region_data_reduction}

{**********************PARAMETER VALIDITY CHECK*****************************}

procedure check_all_parameters_validity(var saturation_parameter_error:integer);
{this procedure checks each of the just-extracted 17 parameters to see if
 they are within a reasonable range of values.  if they are not, then they
 will not be stored in the die file, and hence, not used when computing
 the process file}
{last change made: THURSDAY NOV 8, 1984   By: JRP}

begin
    saturation_parameter_error:=0;
```

...check_all_parameters_validity

```
{JRP}if not( (c1_vfb >= -5.0) and (c1_vfb <= 1.0) and
             (c2_phif2 >= 0.2) and (c2_phif2 <= 1.5) and
             (c3_k1 >= 0.0) and (c3_k1 <= 5.0) and
{JRP}        (c4_k2 >= -1.0) and (c4_k2 <= 1.0) and  {allowed negative for PMOS}
             (c5_eta >= -1.0) and (c5_eta <= 1.0) and
             (c6_beta0 >= 0.0) and (c6_beta0 <= 1.0) and
{JRP}        (c7_u0 >= -1.0) and (c7_u0 <= 1.0) and
             (c8_u1 >= -1.0) and (c8_u1 <= 5.0) and
             (c9_x2beta0 >= -1.0) and (c9_x2beta0 <= 1.0) and
             (c10_x2eta >= -1.0) and (c10_x2eta <= 1.0) and
             (c11_x3eta >= -1.0) and (c11_x3eta <= 1.0) and
             (c12_x2u0 >= -1.0) and (c12_x2u0 <= 1.0) and
             (c13_x2u1 >= -1.0) and (c13_x2u1 <= 1.0) and
             (c14_beta0sat >= 0.0) and (c14_beta0sat <= 1.0) and
             (c15_x2beta0sat >= -1.0) and (c15_x2beta0sat <= 1.0) and
             (c16_x3beta0sat >= -1.0) and (c16_x3beta0sat <= 1.0) and
             (c17_x3u1 >= -1.0) and (c17_x3u1 <= 1.0) ) then
    saturation_parameter_error:=3;
end;


begin    {main procedure extract_device_parameters begins here}
 {*********************************take out later*************************}
     if TRACE21=true then writeln(userout2,
          'entering extract_device_parameters');
     if TRACE29=true then
         begin
             if present_device=1 then
                 rewrite(tempfile,'file1.TEXT');
             if present_device=2 then
                 rewrite(tempfile,'file2.TEXT');
             if present_device=3 then
                 rewrite(tempfile,'file3.TEXT');
             if present_device=4 then
                 rewrite(tempfile,'file4.TEXT');
             for iii:=1 to numbvd do
               begin
                 for jjj:=1 to numbvb do
                   begin
                     for kkk:=1 to numbvg do
                       begin
                         writeln(tempfile,ids[iii,jjj,kkk]);
                       end;
                   end;
               end;
             writeln(tempfile,'**********');
             for iii:=1 to numbvd do
               begin
                 for jjj:=1 to numbvb do
                   begin
                     for kkk:=1 to numbvg do
                       begin
                         writeln(tempfile,vgs[iii,jjj,kkk]);
                       end;
                   end;
               end;
             writeln(tempfile,'**********');
             close(tempfile,'save');
         end;

    if (TRACE29b=true) or (TRACE37=true) then
         begin
             for iii:=1 to numbvd do
               begin
```

```
                    if iii=1 then writeln(userout2,'VD=0.1')
                    else if iii=2 then writeln(userout2,'VD=0.2')
                    else if iii=3 then writeln(userout2,'VD=4.5')
                    else if iii=4 then writeln(userout2,'VD=5.0');
                    for jjj:=1 to numbvb do
                      begin
                        write(userout2,'VB=',jjj-6:1);
                        for kkk:=1 to numbvg do
                          begin
                            write(userout2,' ',ids[iii,jjj,kkk],',');
                          end;
                        writeln(userout2);
                      end;
                  end;
          end;  {TRACE29b or TRACE37}
 {*********************************take out later*************************}
                for vdindex:=1 to number_small_vds_values do
                    begin
                        for vbindex:=1 to numbvb do
                            linear_region_extraction;
                        bsim_timer(time_count);
                    end;
                for vdindex:=1 to number_small_vds_values do
                    if ( (meas_20f=TRUE) or (fake_meas_20f=TRUE) ) then
                        large_device_20f_extraction
                    else
                        linear_region_threshold_analysis;
                linear_region_data_reduction;
{jrp}           check_linear_parameters_validity(extract_error);
{jrp}           if extract_error=0 then
                    begin
                        {saturation region extraction}
                        for vdindex:=numbvd downto numbvd+1-number_large_vds_values do
                            begin
                                for vbindex:=numbvb downto 1 do
{jrp}                               if extract_error=0 then
{jrp}                                   saturation_region_data_extraction(extract_error);
                                bsim_timer(time_count);
                            end;
{jrp}                   if extract_error=0 then
{jrp}                       begin
                                saturation_region_data_reduction;
{jrp}                           check_all_parameters_validity(extract_error);
{JRP}
{JRP}                           {3 lines omitted here}
{JRP}
{JRP}                           if extract_error=3 then  {saturation region error}
{jrp}                               begin
{jrp}                                   gotoxy(0,22);
{jrp}                                   write('THESE PARAMETERS ARE NOT WITHIN ',
                                            'ACCEPTABLE LIMITS & ARE NOT BEING SAVED');
{jrp}                               end;
{jrp}                       end
{jrp}                   else if extract_error=2 then  {saturation region error}
{jrp}                       begin
{jrp}                           gotoxy(0,22);
{jrp}                           write('THESE PARAMETERS ARE NOT BEING SAVED:',
                                    ' ONLY 2 SETS OF DATA FOR LEASTSQ3 PROC.');
{jrp}                       end;
                        bsim_timer(time_count);
{jrp}               end  {of "if extract_error=0 then....."}
{jrp}           else  {error messages}
```

..extract_device_parameters

```
{jrp}               begin
{jrp}                   gotoxy(0,22);
{jrp}                   if extract_error=1 then  {linear region error}
{jrp}                       write('THESE PARAMETERS ARE NOT WITHIN ACCEPTABLE',
                                  ' LIMITS & ARE NOT BEING SAVED');
{jrp}               end;
    {*****************************take out later************************}
        if TRACE21=true then writeln(userout2,
            'leaving extract_device_parameters');
    {*****************************take out later************************}
end;  {end of procedure extract_device_parameters}


{************************************************************
**********************BSIM IV GRAPHICS ROUTINES*****************
************************************************************}
```

iv_graphics

```
procedure iv_graphics;
{if the mode is not single device, then
 this procedure asks the user if he/she wants to view the standard IV curves
 for any location on the wafer.  it then enters the graphics mode if
 answer is yes
 if the mode is single device, the graphics mode is automatically entered}
{last change made NOV 8, 1984  by JRP}.

var   another_device:boolean;  {if = true, then do another graph}
      end_of_graphics:boolean;  {= true when no more graphing is desired}
      iv_curves_wanted:boolean;  {if = true, then enter IV graphics mode}
      iv_graphics_error:integer;  {if = 0, output file is not empty and type of
                                   device user is looking for does exist}
```

prepare_for_iv_graphics

```
procedure prepare_for_iv_graphics(var prepare_for_iv_graphics_error:integer);
{this procedure will search the output file.  This output file
contains 1 to ? process files, in sequence.  Each process file has
the x and y die positions from the wafer it was generated from.  This
program asks the user the following:
    do you want to enter the graphics mode?
    if so, what is the x-y die location?
    and what is the device length and width?
It then finds that position in the process file. IF IT EXISTS.  If it doesn't
exist, the graphics portion will not be entered, and the user will be warned.
The 54 bsim parameters are then read from the process file.  Subsequent
processing will prepare
the 54 parameters for graphing.  This includes calculating 17 electrical
parameters from the 51 process parameters using the formula:
bsimparameter = p0 + pl/Leff + pw/Weff , and also vdd, tox, and temp.}


const   numbgarbage=9;                    {# of lines of junk in each process file}
        position_of_xpos=5;                {xpos is the 5th line of the file}
        numb_junk_lines_between_parameters=2;  {# of lines between *YPOS= line
                                                and first parameter line}
        numb_chars_in_parameter=12;          {# of characters in 1 parameter}

var     found_correct_die:boolean:  {flag set true when correct die is found}
        number_process_groups:integer;  {# of sets of 17 bsim params in file}
        xposstring,yposstring:linestring;  {x&y strings we are looking for}
        wanted_xpos,wanted_ypos:linestring; {values input by user--x and y positions}
        real_wanted_xpos:real;  {real version of inputted wanted_xpos}
        real_wanted_ypos:real;  {real version of inputted wanted_ypos}
        xposline:linestring; {output file line with *XPOS= in it}
```

```
        yposline:linestring; {output file line with *YPOS= in it}
        devicenameline:linestring; {output file line with devicename in it}
        {output_file:linestring;  name of output file which contains groups of 17
                                  bsim parameters}
        i:integer; {loop counter}
        junk:linestring; {bogus variable used to determine # of groups of 17}
        devicename:linestring; {devicename entered in by user}
        devicename_selection:char;  {user selects type of device}
        zz:integer;  {counter to help debug repeat loop}
        parameterline:linestring; {holds entire line of params from proc. file}
        value:char;  {holds each character as it read from parameterline}
        parameter:linestring;  {if value is not ',' put value in parameter}
        k:integer; {position of character in parameter}
        j:integer; {loop counter for processing each line of process file}

        p0,pl,pw:real; {p0, pl, and pw parameters are read from
                        the process file into these 3 variables}
        temporary:array[1..3,1..20] of real:  {then they are placed into temporary
                                               array once it's determined which line
                                               of the proces file they came from}
        bsim:array[1..20] of real;  {after calculation of the electrical parameter
                                     using bsim[] = p0 + pl/Leff + pw/Weff , parameters
                                     are stored here.  Then they are put into proper
                                     names, like c1_vfb, c2_phif2, etc.}
        paramcount:integer; {loop counter for un-normalizing first 7
                             (except beta0) parameters}
{JRP}   pos:integer;  {bogus variable used if mode='4' for the strdelete
                       procedure call}

{jrp}{entire page is drawn on the screen first, then the inputs are taken}

begin
{JRP}if mode<>'4' then
{JRP}begin
        write(#12);                              {clear screen}
        gotoxy(0,0);
        writeln('                              ***PREPARATION FOR I-V GRAPHICS***');
        writeln;
        writeln('ENTER X DIE POSITION OF DEVICE TO BE GRAPHED=   >');
        writeln;
        writeln('ENTER Y DIE POSITION OF DEVICE TO BE GRAPHED=   >');
        writeln;
        writeln('                                                    [1] NMOS enhancement');
        writeln('SELECT THE NUMBER CORRESPONDING                      [2] NMOS depletion');
        writeln('TO THE DEVICE TYPE WHICH                             [3] NMOS zero-threshold');
        writeln('YOU WOULD LIKE TO GRAPH=     >                       [4] PMOS enhancement');
        writeln('                                                    [5] PMOS depletion');
        writeln('                                                    [6] PMOS zero-threshold');
        writeln;
        writeln('DEVICE WIDTH (microns) =   >');
        writeln;
        writeln('DEVICE LENGTH (microns) =   >');
        {the user will now input the desired x position of the die file}
        real_wanted_xpos:=100.0;  {bogus value so 'while loop' is executed}
        while ( (real_wanted_xpos>20.0) or (real_wanted_xpos<1.0) ) do
            begin
                gotoxy(48,2);
                write('                ');
                gotoxy(48,2);
                readln(wanted_xpos);
                real_wanted_xpos:=strtoreal(wanted_xpos);
                if (real_wanted_xpos>20.0) then{maximum die matrix size is 20}
                    begin
                        gotoxy(0,3);
```

```
{JRP}                    write('*** VALUE MUST BE BETWEEN 0 AND 20 ***');
                    end;
            end;
{jrp}gotoxy(45,2);
{jrp}write('                ');
{jrp}gotoxy(45,2);
{jrp}write(real_wanted_xpos:0:0);
      gotoxy(0,3);
      write('                                                    ');
      {the user will now input the desired y position of the die file}
      real_wanted_ypos:=100.0;  {bogus value so 'while loop' is executed}
      while ( (real_wanted_ypos>20.0) or (real_wanted_ypos<1.0) ) do
            begin
                  gotoxy(48,4);
                  write('                        ');
                  gotoxy(48,4);
                  readln(wanted_ypos);
                  real_wanted_ypos:=strtoreal(wanted_ypos);
                  if (real_wanted_ypos>20.0) then{maximum die matrix size is 20}
                        begin
                              gotoxy(0,5);
{JRP}                         write('*** VALUE MUST BE BETWEEN 0 AND 20 ***');
                        end;
            end;
{jrp}gotoxy(45,4);
{jrp}write('                ');
{jrp}gotoxy(45,4);
{jrp}write(real_wanted_ypos:0:0);
      gotoxy(0,5);
      write('                                                    ');
      {user will now select the devicename}
      selection_input(28,9,'1','6',devicename_selection);
      case devicename_selection of
            '1':devicename:='NMOSE';
            '2':devicename:='NMOSD';
            '3':devicename:='NMOSZ';
            '4':devicename:='PMOSE';
            '5':devicename:='PMOSD';
            '6':devicename:='PMOSZ';
            end;
{JRP}case devicename_selection of
{JRP}      '1','2','3':device:=1;
{JRP}      '4','5','6':device:=-1;
{JRP}      end;
      if TRACE23=true then writeln(userout2,
            'this is the devicename as determined by case --> ',devicename);
      gotoxy(27,13);
      width:=strtoreal(realtostr); {read in value of width}
      gotoxy(25,13);
      write('                                    ');
      gotoxy(25,13);
      write(width:3:2);
      gotoxy(28,15);
      length:=strtoreal(realtostr); {read in value of length}
      gotoxy(26,15);
      write('                                    ');
      gotoxy(26,15);
      write(length:3:2);
{JRP}end {of "if mode<>'4' then ...."}
{JRP}else   {mode='4'}
{JRP}{this "else" part simply gets the devicename, wanted_xpos, and
      wanted_ypos values from the already known enhancement_depletion,
      device, present_diex, and present_diey variables}
```

```
{JRP}     begin
{JRP}          if device=1 then                              {NMOS device}
{JRP}               case enhancement_depletion of
{JRP}                    1:devicename:='NMOSE';
{JRP}                    0:devicename:='NMOSZ';
{JRP}                    -1:devicename:='NMOSD';
{JRP}               end {of case}
{JRP}          else if device=-1 then                        {PMOS device}
{JRP}               case enhancement_depletion of
{JRP}                    1:devicename:='PMOSE';
{JRP}                    0:devicename:='PMOSZ';
{JRP}                    -1:devicename:='PMOSD';
{JRP}               end; {of case}
{JRP}          wanted_xpos:='     ';
{JRP}          strwrite(wanted_xpos,1,pos,present_diex);{change to string value}
{JRP}          while strpos(' ',wanted_xpos)<>0 do
{JRP}               strdelete(wanted_xpos,strpos(' ',wanted_xpos),1);
{JRP}          wanted_ypos:='     ';
{JRP}          strwrite(wanted_ypos,1,pos,present_diey);{change to string value}
{JRP}          while strpos(' ',wanted_ypos)<>0 do
{JRP}               strdelete(wanted_ypos,strpos(' ',wanted_ypos),1);
{JRP}     end; {of else part of "if mode<>'4' then..."}
{JRP}     if TRACE41=true then writeln(userout2,
{JRP}          'wanted_xpos=',wanted_xpos,'end',
{JRP}          'wanted_ypos=',wanted_ypos,'end',
{JRP}          'devicename=',devicename,'end');
     {determine how many lines are in the process file}
     number_process_groups:=0;
     reset(userout,output_file);
     while not(eof(userout)) do  {simply count the lines until at end of file}
          begin
               for i:=1 to numbbsim+numbgarbage do
                    readln(userout,junk);
               number_process_groups:=number_process_groups + 1;
          end;
     close(userout,'save');
     if TRACE23=true then write(userout2,
          'the number_of_process_groups is = ',number_process_groups);
     zz:=0;
     prepare_for_iv_graphics_error:=0;
     found_correct_die:=false;
     {we are about to get the process parameters from the output file.
      the output file might be empty--(i.e. no good devices were found)--
      and if this is the case, the user should be warned. this 'if' statement
      will go to "else" part if the file was empty, and the user will be
      warned in the main bsim program}
     if (number_process_groups<>0) then  {if = 0, then nothing is in file!!}
          begin
               reset(userout,output_file);
               xposstring:='*XPOS=';
               yposstring:='*YPOS=';
               strappend(xposstring,wanted_xpos);
               strappend(yposstring,wanted_ypos);
               if (TRACE23=true) or (TRACE41=true) then writeln(userout2,
                    'xposstring after appending inputted value is ',xposstring);
               if (TRACE23=true) or (TRACE41=true) then writeln(userout2,
                    'yposstring after appending inputted value is ',yposstring);
               readln(userout,devicenameline);
               if (TRACE23=true) or (TRACE41=true) then writeln(userout2,
                    '1st devicenameline read from file is = ',devicenameline);
               repeat
                    zz:=zz+1;
                    if(strpos(devicename,devicenameline)<>0) then
                         begin
                              if TRACE23=true then writeln(userout2,
```

```
                    'devicename found in devicenameline on',
                        zz,'-th time thru repeat loop');
                number_process_groups:=number_process_groups - 1;
                for i:=1 to (position_of_xpos - 1) do
                    begin
                        readln(userout,xposline);
                        if TRACE23=true then writeln(userout2,
                            'xposline as read from outputfile = ',
                            xposline);
                    end;
                readln(userout,yposline);
                if TRACE23=true then writeln(userout2,
                    'yposline as read from outputfile = ',yposline);
                if((strpos(xposstring,xposline)<>0) and
                    (strpos(yposstring,yposline)<>0) and
                    (strpos(devicename,devicenameline)<>0)) then
                        found_correct_die:=true
{*******************************take out later*********************
                        if TRACE23=true then writeln(userout2,
                            '3 strpos conditions are met where --->');
                        if TRACE23=true then writeln(userout2,
                            'xposline= ',xposline,' xposline= ',
                            yposline,' & devicenameline= ',devicenameline);
                        if zz=1 then
                        if TRACE23=true then writeln(userout2,
                            'found correct 3 things first time');
                            end
*******************************take out later*********************}
                else if (number_process_groups<>0) then
                    for i:=1 to (numbbsim + numbgarbage -
                                    position_of_xpos ) do
                        begin
                            readln(userout,devicenameline);
                            if TRACE23=true then writeln(userout2,
                                'devicenameline = ',devicenameline);
                        end;
            end
        else
            begin
                number_process_groups:=number_process_groups - 1;
                if (number_process_groups<>0) then
                    for i:=1 to (numbbsim + numbgarbage) do
                        readln(userout,devicenameline);
            end;
        if TRACE23=true then writeln(userout2,
            'number of process groups now is ',number_process_groups);
        if TRACE23=true then writeln(userout2,
            'number of time thru repeat is = ',zz);
    until ((number_process_groups=0) or (found_correct_die));
    if TRACE22=true then writeln(userout2,
        'thru with repeat loop and found_correct_die=',found_correct_die);
    if TRACE23=true then writeln(userout2,
        'found_correct die should be true because of 3 strpos');
    if (found_correct_die = true) then  {we have found correct file}
        begin
        {if we found the proper a)devicetype, b)xposition, and c)yposition, then
         to here, we are pointing at the *YPOS= line in the output file.}

            for i:=1 to numb_junk_lines_between_parameters do  {read 2 comment lines}
                begin
                    readln(userout,parameterline);
                    if TRACE23=true then writeln(userout2,
                        'junk parameter lines = ',parameterline);
                end;
```

bsim1.0.TEXT

```
for i:=1 to (numbbsim + 1) do  {17 parameter lines plus tox,temp,vdd line}
    begin
        readln(userout,parameterline);          {read line from output file}
        k:=1;                                   {initialize character position counter}
        parameter:='                ';          {zero out parameter string}
      for j:=1 to (numb_chars_in_parameter*3 + 2) do {# chars dealt with}
          begin
              value:=parameterline[j];
              if TRACE23=true then writeln(userout2,
                  'value = ',value);
              if (value <> ',') then                    {if = ',' ignore}
                  begin
                      parameter[k]:=value;
                      k:=k+1;
                      if TRACE23=true then writeln(userout2,
                          'parameter = ',parameter);
                  end;
              if j=numb_chars_in_parameter then {at end of 1st param}
                  begin
                      p0:=strtoreal(parameter);
                      k:=1;
                      parameter:='                ';
                  end;
              if j=(numb_chars_in_parameter*2 + 1) then {at end of 2nd
                  begin
                      pl:=strtoreal(parameter);
                      k:=1;
                      parameter:='                ';
                  end;
              if j=(numb_chars_in_parameter*3 + 2) then {at end of 3rd
                  pw:=strtoreal(parameter);
          end;
      if TRACE23=true then writeln(userout2,
          'about to fill up 1 line of temporary array');
      temporary[1,i]:=p0;           {fill up temporary array with}
      temporary[2,i]:=pl;           {process file params}
      temporary[3,i]:=pw;
end;
{tox,temp,&vdd are taken out of temporary [ ] into their own variables}
tox:=temporary[1,numbbsim+1] * 1E4;                 {translate tox from
                                                        microns to
temp:=temporary[2,numbbsim+1] + 273.15;             {translate temp from
                                                        celsius
vdd:=temporary[3,numbbsim+1];
{calculate electrical bsim values : parameter = p0 + pl/Leff + pw/Weff}
for i:=1 to numbbsim do
    if numbbsim <> betazeroparam then
        bsim[i]:=temporary[1,i] +
                (temporary[2,i]/(length+temporary[2,betazeroparam])) +
            (temporary[3,i]/(width+temporary[3,betazeroparam]));
c1_vfb:=bsim[1];
c2_phif2:=bsim[2];
c3_k1:=bsim[3];
c4_k2:=bsim[4];
c5_eta:=bsim[5];
if temporary[1,betazeroparam] < 1 then  {beta0 parameter is beta0}
    c6_beta0:=bsim[6]
else                                    {beta0 parameter is actually mobility}
    c6_beta0:=bsim[6]*cox*(width+temporary[3,betazeroparam])/
                                                (length+temporary[2,betaze
c7_u0:=bsim[7];
c8_u1:=bsim[8];
c9_x2mu0:=bsim[9];
c9_x2beta0:=c9_x2mu0*cox*
            (width+temporary[3,betazeroparam])/
```

```
                                (length+temporary[2,betazeroparam]);
                        c10_x2eta:=bsim[10];
                        c11_x3eta:=bsim[11];
                        c12_x2u0:=bsim[12];
                        c13_x2u1:=bsim[13];
                        c14_mu0sat:=bsim[14];
                        c14_beta0sat:=c14_mu0sat*cox*
                                (width+temporary[3,betazeroparam])/
                                (length+temporary[2,betazeroparam]);
                        c15_x2mu0sat:=bsim[15];
                        c15_x2beta0sat:=c15_x2mu0sat*cox*
                                (width+temporary[3,betazeroparam])/
                                (length+temporary[2,betazeroparam]);
                        c16_x3mu0sat:=bsim[16];
                        c16_x3beta0sat:=c16_x3mu0sat*cox*
                                (width+temporary[3,betazeroparam])/
                                (length+temporary[2,betazeroparam]);
                        c17_x3u1:=bsim[17];

    {**********************take out later**********************}
                        if TRACE23=true then
                            begin
                                writeln(userout2,'c1_vfb = ',c1_vfb);
                                writeln('c2_phif2 = ',c2_phif2);
                                writeln('c3_k1 = ',c3_k1);
                                writeln('c4_k2 = ',c4_k2);
                                writeln('c5_eta = ',c5_eta);
                                writeln('c6_beta0 = ',c6_beta0);
                                writeln('c7_u0 = ',c7_u0);
                                writeln('c8_u1 = ',c8_u1);
                                writeln('c9_x2beta0 = ',c9_x2beta0);
                                writeln('c10_x2eta = ',c10_x2eta);
                                writeln('c11_x3eta = ',c11_x3eta);
                                writeln('c12_x2u0 = ',c12_x2u0);
                                writeln('c13_x2u1 ',c13_x2u1);
                                writeln('c14_beta0sat = ',c14_beta0sat);
                                writeln('c15_x2beta0sat = ',c15_x2beta0sat);
                                writeln('c16_x3beta0sat = ',c16_x3beta0sat);
                                writeln('c17_x3u1 = ',c17_x3u1);
                            end;
    {**********************take out later**********************}

                    end
                else
                    begin {devicetype not found at wanted_xpos and wanted_ypos}
                        prepare_for_iv_graphics_error:=2;
                        if TRACE22=true then writeln(userout2,
                            'we got into error = 2 part somehow');
                    end;
            end
        else
            prepare_for_iv_graphics_error:=1;  {output file is empty!!}
{JRP}if TRACE40=true then
{JRP}           processpar[2,betazeroparam]:=temporary[2,betazeroparam];
end;  {of procedure prepare_for_iv_graphics}

{jrp}
procedure graphics(vdd,tox:real; var vdterm,vgterm,vsterm,vbterm:integer;
                   device_type:integer; var new_device:boolean);
{this is the main procedure to handle the BSIM I-V graphics capabilities}
{last change made: MONDAY OCT 15, 1984  By: JRP}


type graphlab=string[40];
```

graphics

```
var {the following are the data arrays that will be plotted, and counters}
    yaxsim,yaxmeas:array[1..100,1..10] of real;
    xaxisval:array[1..100] of real;
    zaxisval:array[1..10] of real;
    tvar:real;    {this is the constant voltage parameter}
    numbx,numbz,numby:integer;
    {the following are maximum and minimum array values, and graph labels}
    maximx,minimx,maximy,minimy:real;
    xaxlab,yaxlab,headerlab,zaxlab,xaxisunit,yaxisunit,tvarstr:graphlab;
    {the following are flags to determine what graph to generate}
    measonly,simonly,measandsim:boolean;
    maximum_error,rms_error:real;  {error values for graphs}
    selection:char;  {choice of menu items}
    newgraph,quit:boolean;  {menu redraw flags}
    xaxisexp,yaxisexp:integer;  {exponents for the axis}
    init_ok:boolean;  {for graphics initialization}
    error_return:integer;  {for graphic procedures}
    new_smu_connections:boolean;  {if = true, then get new smu values
                                   from user, otherwise, use last device
                                   tested connections}

{jrp}same_device_selection:char;  {determines what to do after initial graph
                                   has been drawn.  if='3', then user wants a
                                   new graph for the same device, so don't ask
                                   him for new SMU connections}
{jrp}smu_connections_ok:boolean;  {= false if user input number <0 or >4}
```

iv_grsetup

```
procedure iv_grsetup(var init_ok:boolean);
{this procedure is used to initialize the graphics library and a user
specified output device.}

var display_address :integer;
    control:integer;
    error_return:integer;

begin
    graphics_init;
    {#12 clears the CRT.  #10 generates a line feed}
    writeln(#12);
    display_address:=3;

    {initialize the graphics display device}
    display_init(display_address, control, error_return);
    if error_return<>0 then
        begin
            init_ok:=false;
            gotoxy(0,22);
            writeln('Display initialization error #',error_return:1)
        end
    else
        init_ok:=true;
end;  {end of graphics setup}
```

iv_maxminxy

```
procedure iv_maxminxy(var xmax,xmin,ymax,ymin:real; numbx,numbz:integer);
{this procedure uses measured data and/or simulated data in determining
the maximum and minimum values to be used for autoscaling}

var  i,j:integer;
     yvalue:real;

begin
     ymax:=0;
```

...iv_maxminxy

```
        ymin:=0;  {clears the maximum and minimum values for y}
        if (measonly=true) or (measandsim=true)  then
              begin
                    for i:=1 to numbx do
                       for j:=1 to numbz do
                             begin
                                   yvalue:=yaxmeas[i,j];
                                   if yvalue>ymax then
                                         ymax:=yvalue;
                                   if yvalue<ymin then
                                         ymin:=yvalue;
                             end;
              end;
        if (simonly=true) or (measandsim=true) then
              begin
                     for i:=1 to numbx do
                          for j:=1 to numbz do
                                begin
                                      yvalue:=yaxsim[i,j];
                                      if yvalue>ymax then
                                            ymax:=yvalue;
                                      if yvalue<ymin then
                                            ymin:=yvalue;
                                end;
              end;
        if xaxisval[numbx]>xaxisval[1] then
              begin
                    xmin:=xaxisval[1];
                    xmax:=xaxisval[numbx];
              end
        else
              begin
                    xmin:=xaxisval[numbx];
                    xmax:=xaxisval[1];
              end;
end;  {of procedure iv_maxminxy}
```

iv_autoscale

```
procedure iv_autoscale(xmax,xmin,ymax,ymin:real; xlabel,ylabel,top,zlabel,
                       xaxisunit,yaxisunit:graphlab);
{this procedure draws the tick marks for the graphic display, given the
maximum and minimum values for the graph}

type unitlab=string[7];

var   xunitlab,yunitlab:graphlab;
```

iv_find_exponent

```
procedure iv_find_exponent(var expon:integer; var units:graphlab; max,min:real);
{this procedure determines the exponent of the maximum data point for the
graph, and returns this value.  exponents in factors of 3 are used}

var nexttry:integer;
    newmaxim:real;
    maxormin:real;

begin
    expon:=0;
    if abs(max)>abs(min) then
          maxormin:=abs(max)
    else
          maxormin:=abs(min);
    newmaxim:=maxormin;
```

..iv_find_exponent

```
      if trunc(maxormin)>0 then  {number is greater than 1}
            while trunc(newmaxim/1000)>0 do
                  begin
                        expon:=expon+3;
                        newmaxim:=newmaxim/1000;
                  end
      else    {this means that the number is less than 1}
            while trunc(newmaxim)=0 do
                  begin
                        expon:=expon-3;
                        newmaxim:=newmaxim*1000;
                  end;
      units:=' ';
      if (expon<=-3) and (expon>=-9) then
            case expon of
                  -3:units:='milli';
                  -6:units:='micro';
                  -9:units:='nano';
            end;
end;  {end of procedure iv_find_exponent}
```

iv_draw_axis

```
procedure iv_draw_axis(xmax,xmin,ymax,ymin:real;xaxisexp,yaxisexp:integer);
{this procedure draws the axis, and makes and labels the tick marks}

var xaxislab,yaxislab:graphlab;
      maxerrstr,rmserrstr:string[50];
      count:integer;
      cnt:integer;  {also of no earthly value, required by strwrite call}
      wl_label:string[15];  {holds W, followed by L label}
```

iv_x_axis_labels

```
procedure iv_x_axis_labels;
{this procedure labels the x-axis}

var count:integer;
      numbxtck,numlxtck:integer;
      magbxtck:real;
      xpos:real;
      xstring:string[40];
      xtickval:real;
      cnt:integer;
      i:integer;
      littltck:real;
      range:real;
      xoffset:real;
      firstxtk,lastxtk:real;
      maglxtck,firslxtk:real;
      latelxtk:real;
      rangefac:real;


begin
      if xaxisexp>0 then
            for count:=1 to xaxisexp do
                  begin
                        xmax:=xmax/10;
                        xmin:=xmin/10;
                  end
      else if xaxisexp=0 then
            xmax:=xmax
      else
            for count:=1 to -(xaxisexp) do
                  begin
                        xmax:=xmax*10;
```

```
                xmin:=xmin*10;
            end;
range:=xmax-xmin;
rangefac:=1;
while range>10 do
      begin
            range:=range/10;
            rangefac:=rangefac*10;
      end;
while range<1 do
      begin
            range:=range*10;
            rangefac:=rangefac/10;
      end;
if range>6 then
      begin
            magbxtck:=1*rangefac;
            numlxtck:=3;
      end
else if range>3 then
      begin
            magbxtck:=0.5*rangefac;
            numlxtck:=4;
      end
else
      begin
            magbxtck:=0.25*rangefac;
            numlxtck:=4;
      end;
range:=range*rangefac;
{this is where the tick marks are drawn and labeled}
if xmax>0 then
      lastxtk:=magbxtck*trunc((xmax+magbxtck)/magbxtck)-magbxtck
else
      lastxtk:=-magbxtck*trunc((abs(xmax)+magbxtck)/magbxtck);
if xmin>0 then
      firstxtk:=magbxtck*trunc((xmin+magbxtck)/magbxtck)
else
      firstxtk:=-magbxtck*trunc((abs(xmin)+magbxtck)/magbxtck)+magbxtck;
xoffset:=((firstxtk-xmin)/range)*1.6;
numbxtck:=round((lastxtk-firstxtk)/magbxtck)+1;
xtickval:=firstxtk;
for i:=1 to numbxtck do
      begin
            xpos:=(i-1)*1.6/(numbxtck-1)*((lastxtk-firstxtk)/range);
            xpos:=xpos-0.8+xoffset;
            move(xpos,-0.8);   {move to position for big tick}
            line(xpos,-0.75);   {draws x tick}
            set_line_style(7);   {dotted line}
            line(xpos,0.8);
            set_line_style(1);   {full line}
            move(xpos-0.08,-0.88);   {moves to location for label}
            xstring:='      ';
            strwrite(xstring,4,cnt,xtickval:3:2);   {writes the string}
            set_char_size(0.02,0.05);
            gtext(xstring);   {writes out the string}
            xtickval:=xtickval+magbxtck;
      end;
maglxtck:=magbxtck/(numlxtck+1);
if xmin>0 then
      firslxtk:=maglxtck*trunc((xmin+maglxtck)/maglxtck) {little tick}
else
      firslxtk:=-maglxtck*trunc((abs(xmin)+maglxtck)/maglxtck)+maglxtck;
xoffset:=(firslxtk-xmin)/range*1.6;
latelxtk:=firslxtk;
```

..iv_x_axis_labels

```
      while latelxtk<xmax do
            begin
                  xpos:=-0.8+xoffset+(latelxtk-firslxtk)/range*1.6;
                  move(xpos,-0.8);
                  line(xpos,-0.78);
                  latelxtk:=latelxtk+maglxtck;
            end;
 end;  {end of procedure iv_x_axis_labels}
```

iv_y_axis_labels

```
procedure iv_y_axis_labels;
{this procedure labels the y-axis}

var count:integer;
    numbytck,numlytck:integer;
    magbytck:real;
    ypos:real;
    ystring:string[40];
    ytickval:real;
    cnt:integer;
    i:integer;
    littltck:real;
    range:real;
    yoffset:real;
    firstytk,lastytk:real;
    maglytck,firslytk:real;
    latelytk:real;
    rangefac:real;


begin
      if yaxisexp>0 then
            for count:=1 to yaxisexp do
                  begin
                        ymax:=ymax/10;
                        ymin:=ymin/10;
                  end
      else if yaxisexp=0 then
            ymax:=ymax
      else
            for count:=1 to -(yaxisexp) do
                  begin
                        ymax:=ymax*10;
                        ymin:=ymin*10;
                  end;
      range:=ymax-ymin;
      rangefac:=1;
      while range>10 do
            begin
                  range:=range/10;
                  rangefac:=rangefac*10;
            end;
      while range<1 do
            begin
                  range:=range*10;
                  rangefac:=rangefac/10;
            end;
      if range>7 then
            begin
                  magbytck:=2*rangefac;
                  numlytck:=3;
            end
```

```
        else if range>3 then   {3<range<7}
                begin
                        magbytck:=1*rangefac;
                        numlytck:=4;
                end
        else                   {1<range<3}
                begin
                        magbytck:=0.5*rangefac;
                        numlytck:=4;
                end;
        range:=range*rangefac;
        {this is where the tick marks are drawn and labeled}
        if ymax>0 then
                lastytk:=magbytck*trunc((ymax+magbytck)/magbytck)-magbytck
        else
                lastytk:=-magbytck*trunc((abs(ymax)+magbytck)/magbytck);
        if ymin>0 then
                firstytk:=magbytck*trunc((ymin+magbytck)/magbytck)
        else
                firstytk:=-magbytck*trunc((abs(ymin)+magbytck)/magbytck)+magbytck;
        yoffset:=((firstytk-ymin)/range)*1.6;
        numbytck:=round((lastytk-firstytk)/magbytck)+1;
        ytickval:=firstytk;

        for i:=1 to numbytck do
                begin
                        ypos:=(i-1)*1.6/(numbytck-1)*((lastytk-firstytk)/range);
                        ypos:=ypos-0.8+yoffset;
                        move(-0.8,ypos);   {position for y tick}
                        line(-0.75,ypos);  {draws y tick}
                        set_line_style(7);  {dotted line}
                        line(0.8,ypos);   {draws dotted line}
                        set_line_style(1);    {full line}
                        move(-0.93,ypos);    {moves to location for label}
                        ystring:='      ';
                        strwrite(ystring,1,cnt,ytickval:1:2);  {writes the string}
                        set_char_size(0.02,0.05);
                        gtext(ystring);    {plots the string}
                        ytickval:=ytickval+magbytck;
                end;
        maglytck:=magbytck/(numlytck+1);
        if ymin>0 then
                firslytk:=maglytck*trunc((ymin+maglytck)/maglytck)   {little tick}
        else
                firslytk:=-maglytck*trunc((abs(ymin)+maglytck)/maglytck)+maglytck;
        yoffset:=(firslytk-ymin)/range*1.6;
        latelytk:=firslytk;
        while latelytk<ymax do
                begin
                        ypos:=-0.8+yoffset+(latelytk-firslytk)/range*1.6;
                        move(-0.8,ypos);
                        line(-0.78,ypos);
                        latelytk:=latelytk+maglytck;
                end;
end;    {end of procedure iv_y_axis_labels}


begin {procedure iv_draw_axis begins here}
      {this is where the axis are drawn}
      move(-0.8,-0.8);
      line(-0.8,0.8);
      line(0.8,0.8);
      line(0.8,-0.8);
      line(-0.8,-0.8);
```

```
        iv_x_axis_labels;  {draws the tick marks and labels for the x axis}
        iv_y_axis_labels;  {draws the ticks and labels for the y axis}
        set_char_size(0.030,0.060);
        xaxislab:=' ';      {clears the label}
        strappend(xaxislab,xlabel);
        strappend(xaxislab,xunitlab);
        strappend(xaxislab,xaxisunit);
        move(-strlen(xaxislab)*0.0125,-1.0);
        gtext(xaxislab);
        set_text_rot(0.0,1.0);
        yaxislab:=' ';      {clears the label}
        strappend(yaxislab,ylabel);
        strappend(yaxislab,yunitlab);
        strappend(yaxislab,yaxisunit);
        move(-0.96,-strlen(yaxislab)*0.0125);
        gtext(yaxislab);
        set_text_rot(1.0,0.0);
        move(0.80,0.90);
        gtext(zlabel);
        move(-strlen(top)*0.025,0.90);
        set_char_size(0.05,0.1);
        gtext(top);
        set_char_size(0.035,0.075);
        wl_label:='W=';                                  {graph W=width}
        if width <= 10.0 then
            strwrite(wl_label,3,cnt,width:3:2)
        else
            strwrite(wl_label,3,cnt,width:3:1);
        move(-strlen(top)*0.025,0.83);
        gtext(wl_label);
        wl_label:='L=';                                  {graph L=length}
        if length <= 10.0 then
            strwrite(wl_label,3,cnt,length:3:2)
        else
            strwrite(wl_label,3,cnt,length:3:1);
        move(strlen(top)*0.025 - strlen(wl_label)*0.035,0.83);
        gtext(wl_label);
        move(-1.0,0.95);
        set_char_size(0.035,0.075);
        gtext('BSIM3.3');
{jrp}move(-1.0,0.88);
{jrp}date:=strrtrim(date);
{jrp}gtext(date);
        {here is where the rms error will be plotted}
        if (measandsim=true) then
            begin
                move(0.5,-1.0);
                set_char_size(0.030,0.06);
                rmserrstr:='RMS ERROR=      ';
                strwrite(rmserrstr,11,count,rms_error:2:2);
                strappend(rmserrstr,'%');
                gtext(rmserrstr);
            end;
        {here is where the fourth variable is plotted}
        move(-1.0,-1.0);
        set_char_size(0.035,0.075);
        strwrite(tvarstr,5,count,tvar:3:2);
        gtext(tvarstr);
end;    {end of procedure iv_draw_axis which draws and labels axis}


begin   {main procedure iv_autoscale begins here}
        iv_find_exponent(xaxisexp,xunitlab,xmax,xmin);
        iv_find_exponent(yaxisexp,yunitlab,ymax,ymin);
```

```
      iv_draw_axis(xmax,xmin,ymax,ymin,xaxisexp,yaxisexp);   {draws and labels}
end;  {end of procedure iv_autoscale}
```

iv_draw_the_data

```
procedure iv_draw_the_data(xmax,xmin,ymax,ymin:real; measured:boolean);
{this procedure draws in the measured and or simulated data}
{last change made: SUNDAY OCT 14, 1984  By: JRP}

var   i,j,k:integer;
      yval,xval:real;
      xnew,ynew:real;
      xold,yold:real;
      cnt:integer;
      zstring:string[6];
      labelz:boolean;
      slope,test:real;
      outplane:boolean;


begin
  set_char_size(0.03,0.06);
  for k:=1 to numbz do
    for i:=1 to numbx do
      begin
        xval:=xaxisval[i];
        if measured=true then
            yval:=yaxmeas[i,k]
        else
            begin
                yval:=yaxsim[i,k];
                set_line_style(1);
            end;
        if xaxisexp>0 then
            for j:=1 to xaxisexp do
                xval:=xval/10;
        if xaxisexp<0 then
            for j:=1 to xaxisexp do
                xval:=xval*10;
        if yaxisexp>0 then
            for j:=1 to yaxisexp do
                yval:=yval/10;
        if yaxisexp<0 then
            for j:=1 to yaxisexp do
                yval:=yval*10;
        xnew:=-0.8+1.6*((xval-xmin)/(xmax-xmin));
        ynew:=-0.8+1.6*((yval-ymin)/(ymax-ymin));
        {check whether point will be on screen}
        if ((xnew>=-0.8) and (xnew<=0.8) and (ynew>=-0.8) and (ynew<=0.8))then
          begin
            if i=1 then
              begin
                  outplane:=false;
                  move(xnew,ynew);
              end;
            if outplane=true then
              begin
{jrp}           if (xnew-xold<>0.0) then   {check for divide by zero}
{jrp}             slope:=(ynew-yold)/(xnew-xold)
{jrp}           else
{jrp}             slope:=1E100;    {about = infinity}
                {check for entering from left}
                test:=slope*(-0.8-xold)+yold;
                if test_point_on_screen1(yold,ynew,test) then
                    begin
```

```
                        move(-0.8,test);
                        if measured=true then
                            begin
                                move(xnew-0.01,ynew-0.015);
                                gtext('x');
                            end
                        else
                            line(xnew,ynew);
                    end;
                {check for entering from the right}
                test:=slope*(0.8-xold)+yold;
                if test_point_on_screen1(yold,ynew,test) then
                    begin
                        move(0.8,test);
                        if measured=true then
                            begin
                                move(xnew-0.01,ynew-0.15);
                                gtext('x');
                            end
                        else
                            line(xnew,ynew);
                    end;
                if slope<>0 then
                    test:=(0.8-yold)/slope+xold
                else
                    test:=-1.0;  {can not enter from top}
                {check for entering from top}
{jrp}           if test_point_on_screen2(xold,xnew,test) then
                    begin
                        move(test,0.8);
                        if measured=true then
                            begin
                                move(xnew-0.01,ynew-0.15);
                                gtext('x');
                            end
                        else
                            line(xnew,ynew);
                    end;
                if slope<>0 then
                    test:=(-0.8-yold)/slope+xold
                else
                    test:=-1.0;    {can not enter from bottom}
                {check for entering from bottom}
{jrp}           if test_point_on_screen2(xold,xnew,test) then
                    begin
                        move(test,-0.8);
                        if measured=true then
                            begin
                                move(xnew-0.01,ynew-0.15);
                                gtext('x');
                            end
                        else
                            line(xnew,ynew);
                    end;
                outplane:=false;
                {point plotted}
            end
        else
            {last point was on screen}
            begin
                if measured=true then
                    begin
                        move(xnew-0.01,ynew-0.015);
                        gtext('x');
                    end
```

```
                    else
                        line(xnew,ynew);
                end;
            end {end of handling for values on the screen}
        else {start handling points off screen}
            begin
                if i=1 then
                    begin
                        outplane:=true;
                    end;
                if outplane=false then {last point on screen}
                    begin
{jrp}                   if (xnew-xold<>0.0) then  {check for divide by zero}
{jrp}                       slope:=(ynew-yold)/(xnew-xold)
{jrp}                   else
{jrp}                       slope:=1E100;    {about = infinity}
                        {check for exiting from left}
                        test:=slope*(0.8-xold)+yold;
                        if test_point_on_screen1(yold,ynew,test) then
                            begin
                                if measured=false then
                                    line(0.8,test);
                                zstring:=' ';
                                strwrite(zstring,1,cnt,(zaxisval[k]):1:2);
                                move(0.85 ,test);
                                if ((measured=true) or (simonly=true))
                                and (i<>numbx)  then
                                    gtext(zstring);
                            end;
                        {check for exiting from right}
                        test:=slope*(-0.8-xold)+yold;
                        if test_point_on_screen1(yold,ynew,test) then
                            begin
                                if measured=false then
                                    line(-0.8,test);
                                zstring:=' ';
                                strwrite(zstring,1,cnt,(zaxisval[k]):1:2);
                                move(0.85,test);
                                if ((measured=true) or (simonly=true))
                                and (i<>numbx) then
                                    gtext(zstring);
                            end;
                        {check for exiting from top}
                        if slope<>0 then
                            test:=(0.8-yold)/slope+xold
                        else
                            test:=-1.0;    {can not exit from top}
{jrp}                   if test_point_on_screen2(xold,xnew,test) then
                            begin
                                if measured=false then
                                    line(test,0.8);
                            end;
                        {check for exiting from bottom}
                        if slope<>0 then
                            test:=(-0.8-yold)/slope+xold
                        else
                            test:=-1.0;    {can not exit from bottom}
{jrp}                   if test_point_on_screen2(xold,xnew,test) then
                            begin
                                if measured=false then
                                    line(test,-0.8);
                            end;
                        outplane:=true;
```

```
                end; {end for points going off screen}
              end; {end of handling points off screen}
            xold:=xnew;
            yold:=ynew;
            if (i=numbx) and ((measured=true) or (simonly=true))
            and (outplane=false) then
                  begin
                        move(0.85,ynew);
                        zstring:='  ';
                        strwrite(zstring,1,cnt,(zaxisval[k]):1:2);
                        gtext(zstring);
                  end;
        end;
end;
{end of procedure drawdata}
```

iv_zoom_graph_axis

```
procedure iv_zoom_graph_axis(var xpos2,xpos1,ypos2,ypos1:real);
{this procedure allows the user to zoom in on a portion of the graph by
varying the axis limits}

var   button:integer;
      xrange,yrange:real;
      oldxmin,oldymin:real;
      temppos:real;


{initialize the graphics locator as the knob}
begin
      xrange:=xpos2-xpos1;
      yrange:=ypos2-ypos1;
      oldxmin:=xpos1;
      oldymin:=ypos1;
      locator_init(2,error_return);
      if error_return=0 then
            begin
                  await_locator(2,button,xpos1,ypos1);
                  set_echo_pos(xpos1,ypos1);  {sets one diagonal of rectangle}
                  await_locator(8,button,xpos2,ypos2);
                  if xpos1>xpos2 then
                        begin
                              temppos:=xpos1;
                              xpos1:=xpos2;
                              xpos2:=temppos;
                        end;
                  if ypos1>ypos2 then
                        begin
                              temppos:=ypos1;
                              ypos1:=ypos2;
                              ypos2:=temppos;
                        end;
                  {this guarantees that the axis are not zoomed too small}
                  if (abs(xpos1-xpos2)<0.1) or (abs(ypos1-ypos2)<0.1) then
                        begin
                              xpos1:=-0.8;
                              xpos2:=0.8;
                              ypos1:=-0.8;
                              ypos2:=0.8;
                        end;
                  xpos1:=xrange/1.6*(xpos1+0.8)+oldxmin;
                  xpos2:=xrange/1.6*(xpos2+0.8)+oldxmin;
                  ypos1:=yrange/1.6*(ypos1+0.8)+oldymin;
                  ypos2:=yrange/1.6*(ypos2+0.8)+oldymin;
                  clear_display;
```

...iv_zoom_graph_axis

```
            end;
end;  {of procedure iv_zoom_graph_axis}
```

iv_initial_graph

```
procedure iv_initial_graph;
{this procedure uses autoscaling to determine what the minimum and maximum
values of the graph are, and plots the data}

var meas:boolean;
init_ok:boolean;

begin
      iv_grsetup(init_ok);
      set_aspect(210.0,160.0);
      set_display_lim(0.0,210.0,0.0,160.0,error_return);
      {this determines the maximum and minimum values of x and y}
      iv_maxminxy(maximx,minimx,maximy,minimy,numbx,numbz);
      iv_autoscale(maximx,minimx,maximy,minimy,xaxlab,yaxlab,
      headerlab,zaxlab,xaxisunit,yaxisunit);
      if (measonly=true) or (measandsim=true) then
            begin
                  meas:=true;
                  iv_draw_the_data(maximx,minimx,maximy,minimy,meas);
            end;
      if (simonly=true) or (measandsim=true) then
            begin
                  meas:=false;
                  iv_draw_the_data(maximx,minimx,maximy,minimy,meas);
            end;
end;  {end of procedure iv_initial_graph to plot graphics axis and data}
```

iv_zoom

```
procedure iv_zoom;
{this procedure handles zooming the graph axis}

var meas:boolean;

begin
      iv_zoom_graph_axis(maximx,minimx,maximy,minimy);
      clear_display;
      iv_autoscale(maximx,minimx,maximy,minimy,xaxlab,yaxlab,headerlab,
                  zaxlab,xaxisunit,yaxisunit);
      if (measonly=true) or (measandsim=true) then
            begin
                  meas:=true;
                  iv_draw_the_data(maximx,minimx,maximy,minimy,meas);
            end;
      if (simonly=true) or (measandsim=true) then
            begin
                  meas:=false;
                  iv_draw_the_data(maximx,minimx,maximy,minimy,meas);
            end;
end; {end of procedure iv_zoom to zoom in on axis}

{**********IV GRAPHICS MEASUREMENT AND SIMULATION**********}
```

measure_data

```
procedure measure_data(device_type,mtype,vdterm,vgterm,vsterm,vbterm:integer;
                              tvalue:real);
{mtype=1 then measure IDSvsVGS curves.  mtype=2 then measure IDSvsVDS curves}
{tvalue is the value of the constant supply}
```

...measure_data

```
var i,j:integer;                                    {loop counters}
    vdchan,vgchan,vschan,vbchan:string[20];  {to contain VCX where X is SMU}
    vdnumb,vgnumb,vsnumb,vbnumb:string[20];  {to contain SMU number as string}
```

translate_terminals_to_strings

```
procedure translate_terminals_to_strings;
{this procedure sets up string variables needed by the 4145}

var i:integer;
    nextpos:integer;

begin
    {here is where the terminal numbers are converted into channel strings}
    vdnumb:=' ';
    vgnumb:=' ';
    vsnumb:=' ';
    vbnumb:=' ';
    strwrite(vdnumb,1,nextpos,vdterm:1);
    strwrite(vgnumb,1,nextpos,vgterm:1);
    strwrite(vsnumb,1,nextpos,vsterm:1);
    strwrite(vbnumb,1,nextpos,vbterm:1);
    vdchan:='CH ';
    strwrite(vdchan,3,nextpos,vdterm:1);   {sets up vd channel string}
    vgchan:='CH ';
    strwrite(vgchan,3,nextpos,vgterm:1);   {sets up vg channel string}
    vschan:='CH ';
    strwrite(vschan,3,nextpos,vsterm:1);   {sets up vs channel string}
    vbchan:='CH ';
    strwrite(vbchan,3,nextpos,vbterm:1);   {sets up vb channel string}
end;  {end of procedure to translate_terminals_to_strings}

{**********IDSvsVGS DATA MEASUREMENT**********}
```

IDSvsVGS_data

```
procedure IDSvsVGS_data(vds:real);
{this procedure sets up the strings to be used for the sources, sets up
the sources, and measures the data for n or p channel devices}

type testcode=string[20];

var  vsource,vgstart,vgend,vgstep,vbstart,vbstep,vdrain:real;
     numbvbstep,numbvgstep:integer;
     svsource,svgstart,svgend,svgstep,svbstart,snumbvbstep,
     svbstep,svdrain:string[20];
```

channel_definition_for_IDSvsVGS_data

```
procedure channel_definition_for_IDSvsVGS_data;
{this procedure sets up the channels with all being voltage sources and the
source being common.  The gate voltage and body voltage are set up to vary}
{last change made: JUNE 20, 1984   by: JRP}

begin
    writestring(700+hpib_analyzer_address,'IT1 DR0 BC');
    writestring(700+hpib_analyzer_address,'DE ');
    writestring(700+hpib_analyzer_address,vdchan);  {sets up drain as const}
    writestring(700+hpib_analyzer_address,',''VDRAIN'',''IDRAIN'',1,3;');
    writestring(700+hpib_analyzer_address,vgchan);  {sets up gate as variable}
    writestring(700+hpib_analyzer_address,',''VGATE'',''IGATE'',1,1;');
    writestring(700+hpib_analyzer_address,vschan); {sets up source as const}
    writestring(700+hpib_analyzer_address,',''VSOURC'',''ISOURC'',1,3;');
    writestring(700+hpib_analyzer_address,vbchan); {sets up body as variable2}
    writestring(700+hpib_analyzer_address,',''VBODY'',''IBODY'',1,2;');
    writestring(700+hpib_analyzer_address,'VS1; VS2; VM1; VM2;');
```

```
end; {end of procedure to define channels for data aquisition}

procedure string_setup_measure_IDSvsVGS;
{this procedure calculates step sizes for the measurements, fills in the vdd
array, and makes the necessary strings for the 4145}

var   nextpos:integer;  {for strwrite command}
      i:integer; {used for loop counter}

begin
      {determine the value of VSOURCE}
      if device_type=1 then
            vsource:=0.0
      else
            vsource:=vdd;
      {determine the starting value of VGATE}
      if device_type=1 then
            vgstart:=0.0
      else
            vgstart:=vdd;
      {determine the final value of VGATE}
      if device_type=1 then
            vgend:=vdd
      else
            vgend:=0.0;
      {determine the step size for VGATE}
      if vdd<=3 then
            vgstep:=0.05
      else
            if vdd<=5 then
                  vgstep:=0.1
            else
                  if vdd<=10 then
                        vgstep:=0.2
                  else
                        vgstep:=0.5;
      vgstep:=vgstep*device_type;
      {determine the number of steps for VGATE}
      numbvgstep:=trunc(vdd/vgstep*device_type)+1; {plus 1 for the first point}
      {determine the starting value for VBODY}
      if device_type=1 then
            vbstart:=-vdd
      else
            vbstart:=2*vdd;
      {determine the value of the VBODY step}
      if vdd<=5 then
            vbstep:=1.0
      else if vdd<=10 then
            vbstep:=2.0
      else
            vbstep:=3.0;
      vbstep:=device_type*vbstep;
      {determine the number of VBODY steps}
      numbvbstep:=trunc(vdd/abs(vbstep))+1;
      {determine the values of VDRAIN}
      vds:=abs(vds);
      if device_type=1 then
            vdrain:=vds
      else
            vdrain:=vdd-vds;
      {here is where the strings are written}
      svsource:='                    ';
      strwrite(svsource,1,nextpos,vsource:2:3);
      svgstart:='                    ';
```

string_setup_measure_IDSvsVGS

...string_setup_measure_IDSvsVGS

```
        strwrite(svgstart,1,nextpos,vgstart:2:3);
        svgend:='                    ';
        strwrite(svgend,1,nextpos,vgend:2:3);
        svgstep:='                   ';
        strwrite(svgstep,1,nextpos,vgstep:2:3);
        svbstart:='                  ';
        strwrite(svbstart,1,nextpos,vbstart:2:3);
        snumbvbstep:='               ';
        strwrite(snumbvbstep,1,nextpos,numbvbstep);
        svbstep:='                   ';
        strwrite(svbstep,1,nextpos,vbstep:2:3);
        svdrain:='                   ';
        strwrite(svdrain,1,nextpos,vdrain:2:3);
end:    {end of procedure string_setup_measure_IDSvsVGS}
```

source_setup_measure_IDSvsVGS

```
procedure source_setup_measure_IDSvsVGS;
{this procedure sets up the sources to measure the data}
{last change made: JUNE 20, 1984   by: JRP}

begin
        writestring(700+hpib_analyzer_address,'SS  ');
        writestring(700+hpib_analyzer_address,'VR1,');  {setup for gate source}
        writestring(700+hpib_analyzer_address,svgstart);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,svgend);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,svgstep);
        writestring(700+hpib_analyzer_address,',1E-6;');
        writestring(700+hpib_analyzer_address,'VP  ');  {setup for the body source}
        writestring(700+hpib_analyzer_address,svbstart);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,svbstep);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,snumbvbstep);
        writestring(700+hpib_analyzer_address,',0.001;');
        writestring(700+hpib_analyzer_address,'VC');  {set up drain v-source}
        writestring(700+hpib_analyzer_address,vdnumb);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,svdrain);
        writestring(700+hpib_analyzer_address,',100E-3;');
        writestring(700+hpib_analyzer_address,'VC');
        writestring(700+hpib_analyzer_address,vsnumb);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,svsource);
        writestring(700+hpib_analyzer_address,',100E-3;');

end:    {end of procedure source_setup_measure_IDSvsVGS}
```

measure_IDSvsVGS_data

```
procedure measure_IDSvsVGS_data;
{this procedure reads the data from the 4145 for either a nchannel or
a pchannel device.  The values of vds,ids,vbs  and vgs are converted to
correspond to the convention for nchannel devices}
{last change made: NOV 8, 1984   by: JRP}

var   i,j:integer;

begin
{JRP}if device_type=1 then {NMOS}
{JRP}     begin
```

```
{jrp}          writestring(700+hpib_analyzer_address,'SM DM1, XN''VGATE'',1,0,');
{jrp}          writestring(700+hpib_analyzer_address,svgend);
{jrp}          writestring(700+hpib_analyzer_address,';YA''IDRAIN'',1,0,');
{jrp}          writestring(700+hpib_analyzer_address,maxidsstring);
{jrp}          writestring(700+hpib_analyzer_address,';');
{JRP}      end
{JRP}else  {PMOS}
{JRP}      begin
{jrp}          writestring(700+hpib_analyzer_address,'SM DM1, XN''VGATE'',1.0,');
{jrp}          writestring(700+hpib_analyzer_address,svgstart);
{jrp}          writestring(700+hpib_analyzer_address,';YA''IDRAIN'',1,');
{jrp}          writestring(700+hpib_analyzer_address,maxidsstring);
{jrp}          writestring(700+hpib_analyzer_address,',0;');
{JRP}      end;
      writestring(700+hpib_analyzer_address,'IT1 DR0 BC');
      writestring(700+hpib_analyzer_address,'MD ME1');
      wait_till_hpib_ready(hpib_analyzer_address);
      writestring(700+hpib_analyzer_address,'DO ''IDRAIN'';');
      listen_to_hpib(hpib_analyzer_address);
      for j:=1 to numbvbstep do
            for i:=1 to numbvgstep do
                  readnumber(700+hpib_analyzer_address,yaxmeas[i,j]);
      talk_to_hpib(hpib_analyzer_address);
      {here is where the xaxis and zaxis arrays are filled}
      numbx:=numbvgstep;
      for i:=1 to numbvgstep do
            xaxisval[i]:=(i-1)*vgstep;
      numbz:=numbvbstep;
      for j:=1 to numbvbstep do
            zaxisval[j]:=(vbstart+(j-1)*vbstep-vsource);
end;

begin  {begin procedure IDSvsVGS_data}
      channel_definition_for_IDSvsVGS_data;
      string_setup_measure_IDSvsVGS;
      source_setup_measure_IDSvsVGS;
      measure_IDSvsVGS_data;
end;

{**********IDS vs VDS MEASUREMENT**********}

procedure IDSvsVDS_data(vbs:real);                                   IDSvsVDS_data
{this procedure sets up the 4145 and measures data points for various VDS
and VGS values with a fixed VBS}

var   vsource,vdstart,vdend,vdstep,vgstart,vgstep,vbody:real;
      numbvgstep,numbvdstep:integer;
      svsource,svdstart,svdend,svdstep,svgstart,snumbvgstep,
      svbody,svgstep:string[20];

procedure channel_definition_for_IDSvsVDS_data;   channel_definition_for_IDSvsVDS_data
{this procedure sets up the channels with all being voltage sources and the
source being common.  The drain and gate voltages are set to vary}
{last change made: JUNE 20, 1984  by: JRP}

begin
      writestring(700+hpib_analyzer_address,'IT1 DR0 BC');
      writestring(700+hpib_analyzer_address,'DE ');
      writestring(700+hpib_analyzer_address,vdchan);  {drain is variable 1}
      writestring(700+hpib_analyzer_address,',''VDRAIN'',''IDRAIN'',1,1;');
      writestring(700+hpib_analyzer_address,vgchan);  {gate is variable 2}
      writestring(700+hpib_analyzer_address,',''VGATE'',''IGATE'',1,2;');
```

...channel_definition_for_IDSvsVDS_data

```
        writestring(700+hpib_analyzer_address,vschan); {source is constant}
        writestring(700+hpib_analyzer_address,',''VSOURC'',''ISOURC'',1,3;');
        writestring(700+hpib_analyzer_address,vbchan);  {body is constant}
        writestring(700+hpib_analyzer_address,',''VBODY'',''IBODY'',1,3;');
        writestring(700+hpib_analyzer_address,'VS1; VS2; VM1; VM2;');
end;    {end of procedure to define channels for data aquisition}
```

string_setup_measure_IDSvsVDS

```
procedure string_setup_measure_IDSvsVDS;
{this procedure calculates the step sizes for the measurements, fills in the
vdd array, and makes the necessary strings for the 4145}

var   nextpos:integer:  {used in the strwrite command}
      i:integer:  {used for loop counter}

begin
      if device_type=1 then
            vsource:=0.0
      else
            vsource:=vdd;
      {determine the starting values for VDRAIN}
      if device_type=1 then
            vdstart:=0.0
      else
            vdstart:=vdd;
      {determine the final value of VDRAIN}
      if device_type=1 then
            vdend:=vdd
      else
            vdend:=0.0;
      {determine the step size for VDRAIN}
      if vdd <=3 then
            vdstep:=0.05
      else
            if vdd <=5 then
                  vdstep:=0.1
            else
                  if vdd <=10 then
                        vdstep:=0.2
                  else
                        vdstep:=0.5;
      vdstep:=vdstep*device_type;
      {determine number of steps for VDRAIN}
      numbvdstep:=trunc(vdd /vdstep*device_type)+1;   {plus 1 for the first point}
      {determine the value of the VGATE STEP}
      if vdd <=5 then
            vgstep:=1.0
      else if vdd <=10 then
            vgstep:=2.0
      else
            vgstep:=3.0;
      vgstep:=vgstep*device_type;
      {determine the starting value for VGATE}
      if device_type=1 then
            vgstart:=vgstep
      else
            vgstart:=vdd+vgstep;
      numbvgstep:=trunc(vdd /abs(vgstep));
      {determine the value of VBODY}
      if device_type=1 then
            vbody:=-abs(vbs)
      else
            vbody:=vdd+abs(vbs);
```

...string_setup_measure_IDSvsVDS

```
        {here is where the strings are written}
        svsource:=                    ';
        strwrite(svsource,1,nextpos,vsource:2:3);
        svdstart:=                    ';
        strwrite(svdstart,1,nextpos,vdstart:2:3);
        svdend:=                      ';
        strwrite(svdend,1,nextpos,vdend:2:3);
        svdstep:=                     ';
        strwrite(svdstep,1,nextpos,vdstep:2:3);
        svgstart:=                    ';
        strwrite(svgstart,1,nextpos,vgstart:2:3);
        svgstep:=                     ';
        strwrite(svgstep,1,nextpos,vgstep:2:3);
        snumbvgstep:=                 ';
        strwrite(snumbvgstep,1,nextpos,numbvgstep:1);
        svbody:=                      ';
        strwrite(svbody,1,nextpos,vbody:2:3);
end;    {end of procedure string_setup_measure_IDSvsVDS}
```

source_setup_measure_IDSvsVDS

```
procedure source_setup_measure_IDSvsVDS;
{this procedure sets up the sources to measure the data}
{last change made: JUNE 20, 1984   by: JRP}

begin
        writestring(700+hpib_analyzer_address,'SS  ');
        writestring(700+hpib_analyzer_address,'VR1,');   {setup for drain source}
        writestring(700+hpib_analyzer_address,svdstart);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,svdend);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,svdstep);
        writestring(700+hpib_analyzer_address,',100E-3;');
        writestring(700+hpib_analyzer_address,'VP');  {setup for the gate source}
        writestring(700+hpib_analyzer_address,svgstart);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,svgstep);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,snumbvgstep);
        writestring(700+hpib_analyzer_address,',1E-6;');
        writestring(700+hpib_analyzer_address,'VC');   {setup for the body source}
        writestring(700+hpib_analyzer_address,vbnumb);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,svbody);
        writestring(700+hpib_analyzer_address,',10E-6;');
        writestring(700+hpib_analyzer_address,'VC');
        writestring(700+hpib_analyzer_address,vsnumb);
        writestring(700+hpib_analyzer_address,',');
        writestring(700+hpib_analyzer_address,svsource);
        writestring(700+hpib_analyzer_address,',100E-3;');
end;    {end of procedure source_setup_measure_IDSvsVDS}
```

measure_IDSvsVDS_data

```
procedure measure_IDSvsVDS_data;
{this procedure reads the data from the 4145 for either a nchannel or
a pchannel device.  The values of vds,ids,vbs and vgs are converted to
correspond to the convention of nchannel devices}
{last change made: NOV 8, 1984   by: JRP}

var   i,j,k:integer;

begin
{JRP}if device_type=1 then
{JRP}     begin
```

```
{jrp}          writestring(700+hpib_analyzer_address,'SM DM1, XN''VDRAIN'',1,0,');
{jrp}          writestring(700+hpib_analyzer_address,svdend);
{jrp}          writestring(700+hpib_analyzer_address,';YA''IDRAIN'',1.0,');
{jrp}          writestring(700+hpib_analyzer_address,maxidsstring);
{jrp}          writestring(700+hpib_analyzer_address,';');
{JRP}      end
{JRP}else   {device_type=-1 (PMOS)}
{JRP}      begin
{JRP}           writestring(700+hpib_analyzer_address,'SM DM1, XN''VDRAIN'',1,0,');
{JRP}           writestring(700+hpib_analyzer_address,svdstart);
{JRP}           writestring(700+hpib_analyzer_address,';YA''IDRAIN'',1,');
{JRP}           writestring(700+hpib_analyzer_address,maxidsstring);
{JRP}           writestring(700+hpib_analyzer_address,'.0;');
{JRP}      end;
      writestring(700+hpib_analyzer_address,'IT1 DR0 BC');
      writestring(700+hpib_analyzer_address,'MD ME1');
      wait_till_hpib_ready(hpib_analyzer_address);
      writestring(700+hpib_analyzer_address,'DO ''IDRAIN'';');
      listen_to_hpib(hpib_analyzer_address);
      for j:=1 to numbvgstep do
            for i:=1 to numbvdstep do
                  readnumber(700+hpib_analyzer_address,yaxmeas[i,j]);
      talk_to_hpib(hpib_analyzer_address);
      {here is where the xaxis and zaxis arrays are loaded up}
      numbx:=numbvdstep;
      for i:=1 to numbvdstep do
            xaxisval[i]:=(i-1)*vdstep;
      numbz:=numbvgstep;
      for j:=1 to numbvgstep do
            zaxisval[j]:=j*vgstep;
end;  {end of procedure to measure_IDSvsVDS_data}


begin {procedure IDSvsVDS data}
      channel_definition_for_IDSvsVDS_data;
      string_setup_measure_IDSvsVDS;
      source_setup_measure_IDSvsVDS;
      measure_IDSvsVDS_data;
end;  {end of procedure IDSvsVDS data}

{*********************************************************************}

begin {procedure measure_data begins here}
      translate_terminals_to_strings;
      if mtype=1 then
            IDSvsVDS_data(tvalue)
      else
            IDSvsVGS_data(tvalue);
end;

{*************************END OF MEASUREMENT ROUTINES*****************}
```

bsimsim

```
function bsimsim(vds,vbs,vgs:real):real;
{this function returns the value of ids for a given bias condition
as simulated using the latest bsim parameters.  when this function is called,
the beta values have already been calculated from the mobility (mu) values}
{last change made: WEDNESDAY OCT 31, 1984   By: JRP}

var eta,u0,u1,vgsminvth:real;
    a,g,t,vp,sqroot_vp,vth,vc,k,u,r,vdssat:real;
    temp,tempA,tempB,x3vddvdd,x2,x3,beta:real;
```

```
{JRP}leff:real;  {effective channel length for current device to be graphed}
{JRP}i,j:integer;  {counters to output VTH to printer}

begin
      {here is where the voltages are translated to fit n-channel designations}
      vds:=vds*device_type;
      vgs:=vgs*device_type;
      vbs:=vbs*device_type;
      eta:=c5_eta+c10_x2eta*(c2_phif2-vbs)+c11_x3eta*(vds-vdd);
{JRP}if eta<0.0 then
{JRP}      eta:=0.0;
      u0:=c7_u0+c12_x2u0*vbs;
{JRP}if u0<0.0 then
{JRP}      u0:=0.0;
      u1:=c8_u1+c13_x2u1*vbs+c17_x3u1*(vds-vdd);
      if u1<0.0 then
            u1:=0.0;
      {here is where the value of beta is calculated}
      temp:=c6_beta0+c9_x2beta0*vbs;
      tempA:=(c14_beta0sat+c15_x2beta0sat*vbs)/temp-1.0;
      tempB:=c16_x3beta0sat/temp;
      x3vddvdd:=tempB*vdd-tempA;
      x2:=(tempA-x3vddvdd)/vdd;
      x3:=x3vddvdd/vdd/vdd;
      beta:=temp*(1.0+x2*vds+x3*sqr(vds));
      {here is where the parameter a is calculated}
      vp:=c2_phif2-vbs;
      sqroot_vp:=sqrt(vp);
      t:=1.744+0.8364*vp;
      g:=1-1/t;
      a:=1+g/2*c3_k1/sqroot_vp;
      vth:=c1_vfb+c2_phif2+c3_k1*sqroot_vp-c4_k2*vp-eta*vds;
      vgsminvth:=vgs-vth;
      if vgsminvth<0.0 then
            vgsminvth:=0.0;
{JRP}leff:=length+processpar[2,betazeroparam];
{JRP}vc:=u1/a*(vgsminvth)/leff;
      u:=1+u0*(vgsminvth);
{JRP}r:=1+u1*vds/leff;
      t:=sqrt(1+2*vc);
      k:=(1+vc+t)/2;
      vdssat:=vgsminvth/a/sqrt(k);
      if vds<vdssat then
            {linear region}
            bsimsim:=device_type*beta/u/r*vds*(vgsminvth-a/2*vds)
      else
            bsimsim:=device_type*beta/u/a/k/2*vgsminvth*vgsminvth;
      if (vgs-vth)<0 then
            bsimsim:=0.0;
{JRP}{this next part will create Vth values at all 4 combinations of Vds
       and Vbs for Vds=0.1 and Vdd and for Vbs=0 and -Vdd}
{JRP}if (TRACEVTH=true) and (abs(vdd)=abs(vgs)) and (abs(vdd)=abs(vds)) then
{JRP}      begin
{JRP}      for i:=1 to 2 do
{JRP}          for j:=1 to 2 do
{JRP}               begin
{JRP}                    case i of
{JRP}                       1:vbs:=0.0;
{JRP}                       2:vbs:=-(abs(vdd));
{JRP}                    end; {of case}
{JRP}                    case j of
{JRP}                       1:vds:=0.1;
```

```
{JRP}                    2:vds:=abs(vdd);
{JRP}                  end; {of case}
{JRP}                  eta:=c5_eta+c10_x2eta*(c2_phif2-vbs)+c11_x3eta*(vds-vdd);
{JRP}                  if eta<0.0 then
{JRP}                     eta:=0.0;
{JRP}                  vth:=c1_vfb+c2_phif2+c3_k1*sqrt(c2_phif2-vbs)-
{JRP}                        c4_k2*(c2_phif2-vbs)-eta*vds;
{JRP}                  writeln(userout2,'VBS=',vbs:2:2,'  VDS=',vds:2:2,
{JRP}                                          '  VTH=',vth:2:2);
{JRP}              end;
{JRP}      end;
end;  {end of function to simulate bsim parameter current data}
```

error_calculations

```
procedure error_calculations(numbx,numbz:integer; var max,rms:real);
{this procedure calculates the maximum percent error, and the rms error for
all of the values of ids measured and simulated}

var   i,j:integer;  {these are the counter variables}
      error:real;

begin
      max:=0.0;
      rms:=0.0;
      for i:=1 to numbx do
            for j:=1 to numbz do
                  if (abs(yaxmeas[i,j])>1E-6) and (abs(yaxsim[i,j])>1E-6) then
                        begin
                              error:=abs((yaxmeas[i,j]-yaxsim[i,j])/yaxmeas[i,j]);
                              if (error>max) then
                                    max:=error;
                              rms:=rms+error*error;
                        end;
      max:=max*100;  {converts to percent error}
      rms:=100*sqrt(rms/numbx/numbz);  {converts to rms percent error}
end; {end of procedure to determine the maximum percent and rms errors}
```

IDvsVD

```
procedure IDvsVD(vbsconstant:real);
{this procedure loads the data arrays for measured and/or simulated
characteristics and generates the labels}

var   i,j:integer;


begin
      {this is where the labels are setup}
      writeln(#12);
      xaxlab:='VDS in ';
      yaxlab:='IDS in ';
      tvarstr:='VBS=        V';
      headerlab:='IDS versus VDS';
      zaxlab:='VGS(V)';
      xaxisunit:='volts';
      yaxisunit:='amps';
      if (measonly=true) or (measandsim=true) then
            measure_data(device_type,1,vdterm,vgterm,vsterm,vbterm,vbsconstant);
      if (simonly=true) then
            begin
                  if vdd<=5 then
                        numbx:=trunc(10*vdd)+1
                  else if vdd<=10 then
                        numbx:=trunc(5*vdd)+1
```

...IDvsVD

```
                else
                        numbx:=trunc(vdd)+1;
                if vdd<=5 then
                        numbz:=trunc(vdd)
                else if vdd<=10 then
                        numbz:=trunc(vdd/2)
                else
                        numbz:=trunc(vdd/3);
                for i:=1 to numbx do
                        xaxisval[i]:=(i-1)*vdd/(numbx-1)*device_type;
                for j:=1 to numbz do
                        zaxisval[j]:=j*vdd/numbz*device_type;
        end;
    if (simonly=true) or (measandsim=true) then
        begin
                for i:=1 to numbx do
                        for j:=1 to numbz do
                                yaxsim[i,j]:=bsimsim(xaxisval[i],vbsconstant,
                                        zaxisval[j]);
        end;
    if (measandsim=true) then
        error_calculations(numbx,numbz,maximum_error,rms_error);
end;
```

lnIDvsVD

```
procedure lnIDvsVD(vbsconstant:real);
{this procedure is very similar to IDvsVD with the exception that a minimum
current is allowed, and the logarithm is taken}

var   i,j:integer;

begin
    {this is where the labels are setup}
    writeln(#12);
    xaxlab:='VDS in ';
    yaxlab:='ln(IDS) in ';
    tvarstr:='VBS=          V';
    headerlab:='ln(IDS) versus VDS';
    zaxlab:='VGS(V)';
    xaxisunit:='volts';
    yaxisunit:='ln(amps)';
    if (measonly=true) or (measandsim=true) then
        begin
            measure_data(device_type,1,vdterm,vgterm,vsterm,vbterm,
                        vbsconstant);
            for j:=1 to numbz do
                for i:=1 to numbx do
                    begin
                        if yaxmeas[i,j]<1E-12 then
                            yaxmeas[i,j]:=1E-12;
                        yaxmeas[i,j]:=ln(yaxmeas[i,j]);
                    end;
        end;
    if simonly=true then
        begin
            if vdd<=5 then
                numbx:=trunc(10*vdd)+1
            else if vdd<=10 then
                numbx:=trunc(5*vdd)+1
            else
                numbx:=trunc(vdd)+1;
            if vdd<=5 then
                numbz:=trunc(vdd)
            else if vdd<=10 then
                numbz:=trunc(vdd/2)
            else
```

..lnIDvsVD

```
                numbz:=trunc(vdd /3);
            for i:=1 to numbx do
                xaxisval[i]:=(i-1)*vdd /(numbx-1)*device_type;
            for j:=1 to numbz do
                zaxisval[j]:=j*vdd /numbz*device_type;
        end;
    if (simonly=true) or (measandsim=true) then
        begin
            for j:=1 to numbz do
                for i:=1 to numbx do
                    begin
                        yaxsim[i,j]:=bsimsim(xaxisval[i],vbsconstant,
                                                    zaxisval[j]);
                        if yaxsim[i,j]<1E-12 then
                            yaxsim[i,j]:=1E-12;
                        yaxsim[i,j]:=ln(yaxsim[i,j]);
                    end;
        end;
    if (measandsim=true) then
        error_calculations(numbx,numbz,maximum_error,rms_error);
end;
```

IDvsVG

```
procedure IDvsVG(vdsconstant:real);
{this procedure loads the data arrays for measured and/or simulated data
for IDSvsVGS with VBS stepped and VDS held constant}

var   i,j:integer;

begin
    {this is where the labels are setup}
    writeln(#12);
    xaxlab:='VGS in ';
    yaxlab:='IDS in ';
    tvarstr:='VDS=        V';
    headerlab:='IDS versus VGS';
    zaxlab:='VBS(V)';
    xaxisunit:='volts';
    yaxisunit:='amps';
    if (measonly=true) or (measandsim=true) then
        measure_data(device_type,2,vdterm,vgterm,vsterm,vbterm,vdsconstant);
    if (simonly=true) then
        begin
            if vdd<=5 then
                numbx:=trunc(10*vdd)+1
            else if vdd<=10 then
                numbx:=trunc(5*vdd)+1
            else
                numbx:=trunc(vdd)+1;
            if vdd<=5 then
                numbz:=trunc(vdd)+1
            else if vdd<=10 then
                numbz:=trunc(vdd/2)+1
            else
                numbz:=trunc(vdd/3)+1;
            for i:=1 to numbx do
                xaxisval[i]:=(i-1)*vdd /(numbx-1)*device_type;
            for j:=1 to numbz do
                zaxisval[j]:=(-1)*(j-1)*device_type*vdd /(numbz-1);
        end;
    if (simonly=true) or (measandsim=true) then
        begin
```

...IDvsVG

```
                for i:=1 to numbx do
                    for j:=1 to numbz do
                        yaxsim[i,j]:=bsimsim(vdsconstant,zaxisval[j],
                        xaxisval[i]);
            end;
        if (measandsim=true) then
            error_calculations(numbx,numbz,maximum_error,rms_error);
end;
```

draw_menu

```
procedure draw_menu;
{this procedure draws the graphics menu}
{last change made: SUNDAY SEPT 30, 1984  By: JRP}

begin
      writeln(#12);
      writeln('                                ***BSIM I-V GRAPHICS MENU***');
      writeln;
      write('The BSIM I-V graphics routines will draw measured and/or ');
      writeln('simulated I-V data.');
      write('If the program is operating in the "SINGLE" mode, the 17 ELECTRICAL');
      writeln(' parameters');
      write('just extracted will be used. In the "AUTOMATIC" or "SEMI-AUTOMATIC"');
      writeln(' mode, the');
      write('17 ELECTRICAL parameters will be generated from the 54 parameter');
      writeln(' process file.');
      writeln;
      writeln('                      SELECT A NUMBER FOR A GIVEN DISPLAY MODE=  >');
      writeln('                            1)Measured Data Only');
      writeln('                            2)Simulated Data Only');
      writeln('                            3)Measured and Simulated Data');
      writeln;
      writeln('                       SELECT A NUMBER FOR A GIVEN GRAPH TYPE=  >');
      writeln('                            1)IDS versus VDS              VBS=?  >');
      writeln('                            2)IDS versus VGS              VDS=?  >');
      writeln('                            3)ln(IDS) versus VDS          VBS=?  >');
end;  {end of procedure draw_menu}
```

draw_iv_repeat_menu

```
procedure draw_iv_repeat_menu;
{this procedure draws the repeat menu}
{last change made: SUNDAY SEPT 30, 1984  By: JRP}

begin
      writeln(#12);
      gotoxy(0,2);
      writeln('                  SELECT A NUMBER FOR A GIVEN ACTION CAPABILITY=  >');
      writeln('                            1)Zoom Using Knob and Keys');
      writeln('                            2)Redraw Full Graph');
      writeln('                            3)Select New Graph for Current Device');
      writeln('                            4)Select New Device');
      write('                           5)Exit I-V Graphics Menu');
end;  {of procedure draw_iv_repeat_menu}
```

```
{*************************MAIN GRAPHICS PROGRAM**********************}

begin    {main procedure graphics begins here}
    iv_grsetup(init_ok);
    new_device:=false;
    if init_ok then
        begin
{jrp}       same_device_selection:='4';
            repeat
```

```
{jrp}          {**********take out later**********}
{jrp}          if TRACE36=true then
{jrp}               begin
{jrp}                    writeln(userout2,'beginning of repeat loop');
{jrp}                    writeln(userout2,
{jrp}                         'vdterm=',vdterm:1,' vgterm=',vgterm:1,
{jrp}                          ' vsterm=',vsterm:1,' vbterm=',vbterm:1);
{jrp}               end;
{jrp}          {**********take out later**********}
                writeln(#12); {clears the display}
                draw_menu;
                measonly:=false;  {flags to tell what data to plot}
                simonly:=false;
                measandsim:=false;
                selection_input(60,8,'1','3',selection);
                case selection of
                     '1':measonly:=true;
                     '2':simonly:=true;
                     '3':measandsim:=true;
                end;
                selection_input(59,13,'1','3',selection);
                case selection of
                     '1':begin
                              gotoxy(58,14);
                              tvar:=strtoreal(realtostr);
                              tvar:=-1*device_type*abs(tvar);
                         end;
                     '2':begin
                              gotoxy(58,15);
                              tvar:=strtoreal(realtostr);
                              tvar:=device_type*abs(tvar);
                         end;
                     '3':begin
                              gotoxy(58,16);
                              tvar:=strtoreal(realtostr);
                              tvar:=-1*device_type*abs(tvar);
                         end;
                end; {end of case statement to determine graph type}
                if ((measonly=true) or (measandsim=true)) and (mode<>'4') then
                     begin
{jrp}                   if same_device_selection='4' then
{jrp}                        begin
                                  gotoxy(0,18);
                                  writeln('                                        NEW SMU CONNECTIONS
                                  yes_no_selection_input(55,18,new_smu_connections);
{jrp}                        end
{jrp}                   else
{jrp}                       new_smu_connections:=false;
{jrp}                   smu_connections_ok:=false;
{jrp}                   while (smu_connections_ok=false) do
{jrp}                        begin
                                  gotoxy(24,18);
                                  write('                                                ');
                                  gotoxy(0,18);
                                  write('                                    SMU');
                                  writeln(' connected to DRAIN=?   >');
                                  write('                                    SMU');
                                  writeln(' connected to GATE=?   >');
                                  write('                                    SMU');
                                  writeln(' connected to SOURCE=?   >');
                                  write('                                    SMU');
                                  writeln(' connected to BODY=?   >');
                                  if new_smu_connections=true then
                                       begin
                                            gotoxy(52,18);
```

```
                    {reads in the value of drain smu}
                    vdterm:=round(strtoreal(realtostr));
                    gotoxy(48,18);
                    writeln('                    ');
                    gotoxy(48,18);
                    writeln(vdterm:1);
                    gotoxy(51,19);
                    {reads in the value of gate smu}
                    vgterm:=round(strtoreal(realtostr));
                    gotoxy(47,19);
                    writeln('                    ');
                    gotoxy(47,19);
                    writeln(vgterm:1);
                    gotoxy(53,20);
                    {reads in the value of source smu}
                    vsterm:=round(strtoreal(realtostr));
                    gotoxy(49,20);
                    writeln('                    ');
                    gotoxy(49,20);
                    writeln(vsterm:1);
                    gotoxy(51,21);
                    {reads in the value of body smu}
                    vbterm:=round(strtoreal(realtostr));
                    gotoxy(47,21);
                    writeln('                    ');
                    gotoxy(47,21);
                    writeln(vbterm:1);
                end
            else  {use old smu connections}
                begin
                    gotoxy(48,18);
                    writeln('                    ');
                    gotoxy(48,18);
                    writeln(vdterm:1);
                    gotoxy(47,19);
                    writeln('                    ');
                    gotoxy(47,19);
                    writeln(vgterm:1);
                    gotoxy(49,20);
                    writeln('                    ');
                    gotoxy(49,20);
                    writeln(vsterm:1);
                    gotoxy(47,21);
                    writeln('                    ');
                    gotoxy(47,21);
                    writeln(vbterm:1);
                end;
{jrp}       if (vdterm>=1) and (vdterm<=4) and
{jrp}          (vbterm>=1) and (vbterm<=4) and
{jrp}          (vsterm>=1) and (vsterm<=4) and
{jrp}          (vgterm>=1) and (vgterm<=4) then
{jrp}            smu_connections_ok:=true
{jrp}       else
{jrp}            begin
{jrp}                gotoxy(48,18);
{jrp}                write('                    ');
{jrp}                gotoxy(47,19);
{jrp}                write('                    ');
{jrp}                gotoxy(49,20);
{jrp}                write('                    ');
{jrp}                gotoxy(47,21);
{jrp}                write('                    ');
```

```
{jrp}                                   gotoxy(0,22);
{jrp}                                   write('SMU VALUES MUST BE BETWEEN');
{jrp}                                   write(' 1 AND 4.  PLEASE TRY AGAIN');
{jrp}                               end; {of else part}
{jrp}                          end; {of "while smu_connections_ok=false then....."}
{jrp}                     gotoxy(0,22);
{jrp}                     write('                                                    ');
{jrp}                     write('                                                    ');
                     end; {of "if measonly=true....."}
{jrp}                {**************************take out later*****************}
{jrp}                if TRACE36=true then
                       begin
{jrp}                      writeln(userout2,'after user selection');
{jrp}                      writeln(userout2,
{jrp}                            'vdterm=',vdterm:1,' vgterm=',vgterm:1,
{jrp}                            ' vsterm=',vsterm:1,' vbterm=',vbterm:1);
{jrp}                  end;
{jrp}                {**************************take out later*****************}
                     gotoxy(0,23);
{jrp}                if ((measonly=true) or (measandsim=true)) and (mode<>'4') then
                       begin
                            {here is where automatic movement to device will go}
                            write('Place probes on device and Press "ENTER"  >');
                            selection_input(43,23,chr(ord(32)),chr(ord(32)),continue);
                       end
                     else
                       begin
                            write('Press "ENTER" to continue  >');
                            selection_input(28,23,chr(ord(32)),chr(ord(32)),continue);
                       end;
                     case selection of          {from selection_input call}
                          '1':IDvsVD(tvar);     {just a few lines above}
                          '2':IDvsVG(tvar);
                          '3':lnIDvsVD(tvar);
                     end; {end of case statement to graph chosen graph}
{jrp}                {**************************take out later*****************}
{jrp}                if TRACE36=true then
{jrp}                   begin
{jrp}                      writeln(userout2,'before initial_graph is drawn');
{jrp}                      writeln(userout2,
{jrp}                            'vdterm=',vdterm:1,' vgterm=',vgterm:1,
{jrp}                            ' vsterm=',vsterm:1,' vbterm=',vbterm:1);
{jrp}                   end;
{jrp}                {**************************take out later*****************}
                     writeln(#12);
                     iv_initial_graph; {this draws out the autoscaled graph}
                     quit:=false;
                     newgraph:=false;
    {*********************************take out later*******************}
                     if TRACE34=true then
                        begin
                             writeln(#12);
                             writeln('Here are the 17 bsim parameters used to plot this graph: ');
                             writeln('c1_vfb = ',c1_vfb);
                             writeln('c2_phif2 = ',c2_phif2);
                             writeln('c3_k1 = ',c3_k1);
                             writeln('c4_k2 = ',c4_k2);
                             writeln('c5_eta = ',c5_eta);
                             writeln('c6_beta0 = ',c6_beta0);
                             writeln('c7_u0 = ',c7_u0);
                             writeln('c8_u1 = ',c8_u1);
                             writeln('c9_x2beta0 = ',c9_x2beta0);
                             writeln('c10_x2eta = ',c10_x2eta);
                             writeln('c11_x3eta = ',c11_x3eta);
                             writeln('c12_x2u0 = ',c12_x2u0);
```

```
                                writeln('c13_x2u1 ',c13_x2u1);
                                writeln('c14_beta0sat = ',c14_beta0sat);
                                writeln('c15_x2beta0sat = ',c15_x2beta0sat);
                                writeln('c16_x3beta0sat = ',c16_x3beta0sat);
                                writeln('c17_x3u1 = ',c17_x3u1);
                                writeln;
                                write('Press Enter to continue>');
                                read(selection);
                            end;  {of "if TRACE34=true then"}
{**********************************take out later*********************}
                        repeat
{jrp}                   {**************************take out later****************}
{jrp}                   if TRACE36=true then
{jrp}                        begin
{jrp}                             writeln(userout2,'just enterred 2nd repeat loop');
{jrp}                             writeln(userout2,
{jrp}                                  'vdterm=',vdterm:1,' vgterm=',vgterm:1,
{jrp}                                  ' vsterm=',vsterm:1,' vbterm=',vbterm:1);
{jrp}                        end;
{jrp}                   {**************************take out later****************}
                             draw_iv_repeat_menu;
{jrp}                        selection_input(62,2,'1','5',same_device_selection);
{jrp}                        case same_device_selection of
                                  '1':begin
                                           gotoxy(59,0);
                                           write('        ');
                                           gotoxy(59,3);
                                           write('SWITCH TO GRAPHICS');
                                           iv_zoom;
                                      end;
                                  '2':iv_initial_graph;
                                  '3':begin
                                           newgraph:=true;
                                           clear_display;
                                      end;
                                  '4':begin
                                           quit:=true;
                                           new_device:=true;
                                           clear_display;
                                      end;
                                  '5':begin
                                           quit:=true;
                                           clear_display;
                                      end;
                             end;
{jrp}                   {**************************take out later****************}
{jrp}                   if TRACE36=true then
{jrp}                        begin
{jrp}                             writeln(userout2,'just leaving 2nd repeat loop');
{jrp}                             writeln(userout2,
{jrp}                                  'vdterm=',vdterm:1,' vgterm=',vgterm:1,
{jrp}                                  ' vsterm=',vsterm:1,' vbterm=',vbterm:1);
{jrp}                        end;
{jrp}                   {**************************take out later****************}
                         until (quit=true) or (newgraph=true);
                    until newgraph=false;
               end;
          graphics_term; {terminate the graphics procedures}
     end;  {of procedure graphics}

{*************************MAIN IV_GRAPHICS PROGRAM*********************}

begin
```

```
{if mode<>'4' then
     begin}
          if po_graphics_wanted=false then
               begin
                    gotoxy(21,23);
                    write('Would you like to view I-V Curves?  (Y/N)  >');
                    yes_no_selection_input(65,23,iv_curves_wanted);
               end
          else
               begin
                    gotoxy(0,8);
                    write('Would you like to view I-V Curves?  (Y/N)  >');
                    yes_no_selection_input(44,8,iv_curves_wanted);
               end;
          end_of_graphics:=false;
          while not(end_of_graphics) do
               begin
                    if (iv_curves_wanted=false) then
                         end_of_graphics:=true          {graphics not wanted}
                    else
                         begin                                {graphics is wanted}
                              iv_graphics_error:=0;
                              if TRACE21=true then write(userout2,
                                   'just before entering prepare_for_iv_graphics. ');
                              if TRACE23=true then writeln(userout2,
                                   'length = ',length,'width = ',width);
                              prepare_for_iv_graphics(iv_graphics_error);
                              if TRACE21=true then writeln(userout2,
                                   'out of prepare procedure and iv_graphics_error = ',iv_graphics_e
                              if TRACE21=true then write(userout2,
                                   'just before entering graphics. ');
                              if TRACE23=true then writeln(userout2,
                                   'width = ',width,'length = ',length);
                              if (iv_graphics_error=0) then
                                   begin
                                        graphics(vdd,tox,smud,smug,smus,smub,device,another_device);
                                        write(#12);  {clear screen when finished}
                                   end
                              else if (iv_graphics_error=1) then
                                   begin
{JRP}                                   gotoxy(0,21);
{JRP}                                   write('                                                  ');
{JRP}                                   write('                                                  ');
{JRP}                                   gotoxy(0,22);
{JRP}                                   write('                                                  ');
{JRP}                                   write('                                                  ');
{JRP}                                   gotoxy(0,23);
{JRP}                                   write('                                                  ');
{JRP}                                   write('                                                  ');
{JRP}                                   gotoxy(0,21);
{JRP}                                   write('GRAPHICS MENU CAN NOT BE');
{JRP}                                   write(' ACCESSED BECAUSE THE OUTPUT');
{JRP}                                   writeln(' FILE IS EMPTY.');
{JRP}                                   write('(NO PROCESS FILES WERE CREATED AT');
{JRP}                                   writeln(' ANY DIE LOCATION)');
                                        write('Do you want to graph another device?  (Y/N)  >');
                                        yes_no_selection_input(46,23,another_device);
                                   end
                              else if (iv_graphics_error=2) then
                                   begin
{JRP}                                   gotoxy(0,21);
{JRP}                                   write('                                                  ');
```

```
{JRP}                                   write('                                                    ');
{JRP}                                   gotoxy(0,22);
{JRP}                                   write('                                                    ');
{JRP}                                   write('                                                    ');
{JRP}                                   gotoxy(0,23);
{JRP}                                   write('                                                    ');
{JRP}                                   write('                                                    ');
{JRP}                                   gotoxy(0,21);
{JRP}                                   write('GRAPHICS MENU CAN NOT BE');
{JRP}                                   write(' ACCESSED BECAUSE A PROCESS');
{JRP}                                   writeln(' FILE FOR THE SELECTED');
{JRP}                                   write('DEVICE TYPE WAS NOT');
{JRP}                                   write(' CREATED AT THE X AND Y DIE ');
{JRP}                                   writeln('POSITIONS SHOWN ABOVE.');
                                        write('Do you want to graph another device?  (Y/N)  >');
                                        yes_no_selection_input(46,23,another_device);
                                end;
                            if (another_device=false) then
                                end_of_graphics:=true;
                        end;
                end;
        {end
    else } {single mode, which implies--automatically go into graphics}
            {graphics(vdd,tox,smud,smug,smus,smub,device,another_device);}
    write(#12);
end;  {of procedure iv_graphics}
```

```
{**********************************************************
**********************BSIM PO GRAPHICS ROUTINES*******************
**********************************************************}
{BEGIN}
procedure po_graphics;                                          po_graphics

{LAST CHANGE MADE THURSDAY, OCT 4 BY JRP}

type graphlab=string[40];

const  epsilon=0.01;  {for draw_the_data2 (and possible draw_the_data1)....
                       it helps guarantee that data points will still be
                       within -.8 and +.8 screen limits.  e.g. if data point
                       is .80000000001 due to rounding, then it will still be
                       within the new limits of -.81 and +.81}

var error:integer;
    width,length:real;{tox  taken out due to const value above}
    {NEW VARIABLES ADDED TO SUMMER 1984 PROGRAM}
    po_graphics_error:integer;  {if = 0, output file is not empty and type of
                                 device user is looking for does exist}
    {**********these are added to this program**********}
    param:array[1..100,1..numbbsim] of real;  {holds all parameters extracted}
    ww:array[1..100] of real;  {holds W & L of devices}
    ll:array[1..100] of real;  {holds W & L of devices}
    param_to_graph:integer;  {number of bsim param in file}
    numb_sets_17_params:integer;
    graph_to_plot:char;
    repeat_selection:char;
    file_to_graph:char;  {number (1..6) of diefile to graph}
    file_to_graph_name:string[15]; {name of diefile to graph...NEDFILE,PZDFILE,...}
```

```
    third_variable_value:real;  {z axis value}
    xmax_until_newgraph_is_chosen:real;  {xmax for initial graph}
    have_not_zoomed_yet:boolean;  {=true until first graph is drawn}
    delta_l_or_w:real;  {= processpar[2or3,betazeroparam]}
{jrp}effective_size_error:integer;  {if al xvalues are less than the absolute value
                                     of delta_l_or_w, then set = 1 (error)}
{**********these are added to this program in graphics**********}
    {the following are maximum and minimum array values, and graph labels}
    maximx,minimx,maximy,minimy:real;
    xaxlab,yaxlab,headerlab,zaxlab1,zaxlab2,xaxisunit,yaxisunit,
        deltallab,deltawlab:graphlab;
    selection:char;  {choice of menu items}
    newgraph,quit:boolean;  {menu redraw flags}
    xaxisexp,yaxisexp:integer;  {exponents for the axis}
    init_ok:boolean;  {for graphics initialization}
    error_return:integer;  {for graphic procedures}
    {DEBUGGING VARIABLES...SET TRUE IF DEBUGGING IS DESIRED WITHIN PO_GRAPHS
    TRACE1,TRACE2,TRACE3,TRACE4,TRACE5,TRACE6,TRACE7,
    TRACE8,TRACE9,TRACE10,TRACE11,TRACE12,TRACE13,TRACE14:boolean: {for debugging...can turn off
                                                    to the printer by changing these}
{jrp}TRACE15,TRACE16:boolean; {for xmax-xmin procedures, and plot_the_data_1}

    {TRACE 1--shows when entering and leaving procedures}
    {TRACE 2--shows when entering and/or leaving if-else, etc. statements}
    {TRACE 3--shows miscellaneous values}
    {TRACE 4--only used in draw_the_data2 for finding xmax and xmin}
    {TRACE 5--only used in draw_the_data2 for debugging plotting}
    {TRACE 6--only used in maxminxy for determining max-min value accuracy}
    {TRACE 7--only used in prepare_for_po_graphics for debugging proces
              file data gathering}
    {TRACE 8--only used in prepare_for_po_graphics...lists every CHARACTER in
              process file}
    {TRACE 9--only used in x_axis_labels...for details of tick marks}
    {TRACE 10-only used in find_exponent...all details of it}
    {TRACE 11-only used in page1..to see if output file xpos,ypos,and
              devicename exist}
    {TRACE 12--not used}
    {TRACE 13--only in x_axis_labels...to get # of sig. dig. on x axis correct}
    {TRACE 14--only in plot_data1 to see yval and param[i,6]}
{jrp}{TRACE 15-only in max-min procedures to find out why parameter MUOSAT is
              having problems with it's maximum value}
{jrp}{TRACE 16-only in draw_the_data1 to see why zooming when the y exponent
              is positive doesn't work.  values just disappear}
```

prepare_for_po_graphics

```
procedure prepare_for_po_graphics(var prepare_for_po_graphics_error:integer;
{this procedure will search the output file.  This output file
contains 1 to ? process files, in sequence.  Each process file has
the x and y die positions from the wafer it was generated from.  The
procedure "draw_3_menu_pages" asks the user for the devicetype.  The
current x and y die location is known, since we are currently at
present_diex and present_diey.
This program then finds that position in the process file, IF IT EXISTS.  If
it doesn't exist, the graphics portion will not be entered, and the user will
be warned.  The 54 bsim parameters are then read from the process file
and put into the array processpar[].  Subsequent processing will prepare
the 54 parameters for graphing.  This includes calculating 17 electrical
parameters from the 51 process parameters using the formula:
bsimparameter = p0 + pl/Leff + pw/Weff , and also vdd, tox, and temp.}

const    numbgarbage=9;                {# of lines of junk in each process file}
         position_of_xpos=5;                  {xpos is the 5th line of the file}
```

```
        numb_junk_lines_between_parameters=2;  {# of lines between *YPOS= line
                                                  and first parameter line}
        numb_chars_in_parameter=12;            {# of characters in 1 parameter}

var     found_correct_die:boolean;  {flag set true when correct die is found}
        number_process_groups:integer;  {# of sets of 17 bsim params in file}
        xposline:linestring; {output file line with *XPOS= in it}
        yposline:linestring; {output file line with *YPOS= in it}
        xpos,ypos:integer;  {integer equivalent of xposline and yposline}
        devicenameline:linestring; {output file line with devicename in it}
        devicename:linestring; {NMOSE, NMOSD, ...}
        i:integer; {loop counter}
        junk:linestring; {bogus variable used to determine # of groups of 17}
        zz:integer;  {counter to help debug repeat loop}
        parameterline:linestring; {holds entire line of params from proc. file}
        value:char;  {holds each character as it read from parameterline}
        parameter:linestring;  {if value is not ' ' put value in parameter}
        k:integer; {position of character in parameter}
        j:integer; {loop counter for processing each line of process file}

        p0,pl,pw:real; {p0, pl, and pw parameters are read from
                        the process file into these 3 variables}


begin
    {determine how many lines are in the process file}
    number_process_groups:=0;
    reset(userout,output_file);
    while not(eof(userout)) do  {simply count the lines until at end of file}
        begin
            for i:=1 to numbbsim+numbgarbage do
                readln(userout,junk);
            number_process_groups:=number_process_groups + 1;
        end;
    close(userout,'save');
    {*********************************take out later*******************}
    if TRACE7=true then writeln(userout2,
        'the number_of_process_groups is = ',number_process_groups);
    zz:=0;
    {*********************************take out later*******************}
    prepare_for_po_graphics_error:=0;
    found_correct_die:=false;
    {we are about to get the process parameters from the output file.
     the output file might be empty--(i.e. no good devices were found!--
     and if this is the case, the user should be warned. this 'if' statement
     will go to "else" part if the file was empty, and the user will be
     warned in the main bsim program}
    if (number_process_groups<>0) then  {if = 0, then nothing is in file!!}
        begin
            case file_to_graph of
                '1':devicename:='NMOSE';
                '2':devicename:='NMOSD';
                '3':devicename:='NMOSZ';
                '4':devicename:='PMOSE';
                '5':devicename:='PMOSD';
                '6':devicename:='PMOSZ';
            end;  {of case selection}
            if TRACE7=true then writeln(userout2,'devicename = ',devicename);
            reset(userout,output_file);
            readln(userout,devicenameline);
            if TRACE7=true then writeln(userout2,
                '1st devicenameline read from file is = ',devicenameline);
            repeat
```

```
{*******************************take out later*******************}
              zz:=zz+1;
{*******************************take out later*******************}
              if(strpos(devicename,devicenameline)<>0) then
                   begin
                        if TRACE7=true then writeln(userout2,
                            'devicename found in devicenameline on',
                            zz:2,' time thru repeat loop');
                        number_process_groups:=number_process_groups - 1;
                        for i:=1 to (position_of_xpos - 1) do
                             begin
                                  readln(userout,xposline);
                                  if TRACE7=true then writeln(userout2,
                                       'xposline as read from outputfile = ',
                                       xposline);
                              end;
                        strdelete(xposline,1,6);
                        xposline:=strrtrim(xposline);
                        xpos:=trunc(strtoreal(xposline));
                        readln(userout,yposline);
                        strdelete(yposline,1,6);
                        yposline:=strrtrim(yposline);
                        ypos:=trunc(strtoreal(yposline));
                        if TRACE7=true then writeln(userout2,
                             'yposline as read from outputfile = >',
                             yposline);
                        if((xpos=present_diex) and (ypos=present_diey) and
                           (strpos(devicename,devicenameline)<>0)) then
                             found_correct_die:=true
{*******************************take out later*******************
                             if TRACE7=true then writeln(userout2,
                                  '3 strpos conditions are met where --->');
                             if TRACE7=true then writeln(userout2,
                                  'xposline= ',xposline,' yposline= ',
                                  yposline,' & devicenameline= ',devicenameline);
                             if zz=1 then
                             if TRACE7=true then writeln(userout2,
                                  'found correct 3 things first time');
                                end
*******************************take out later*******************}
                        else if (number_process_groups<>0) then
                             for i:=1 to (numbbsim + numbgarbage -
                                                  position_of_xpos ) do
                                  begin
                                       readln(userout,devicenameline);
                                       if TRACE7=true then writeln(userout2,
                                            'devicenameline = ',devicenameline);
                                  end;
                   end
              else
                   begin
                        number_process_groups:=number_process_groups - 1;
                        if (number_process_groups<>0) then
                             for i:=1 to (numbbsim + numbgarbage) do
                                  readln(userout,devicenameline);
                   end;
{*******************************take out later*******************}
         if TRACE7=true then writeln(userout2,
              'number of process groups now is ',number_process_groups:2);
         if TRACE7=true then writeln(userout2,
              'number of time thru repeat is = ',zz:2);
{*******************************take out later*******************}
         until ((number_process_groups=0) or (found_correct_die));
```

```
{*******************************take out later*********************}
        if TRACE7=true then writeln(userout2,
             'thru with repeat loop and found_correct_die=',
             found_correct_die);
        if TRACE7=true then writeln(userout2,
             'found_correct die should be true because of 3 strpos');
{*******************************take out later*********************}
        if (found_correct_die = true) then  {we have found correct file}
             begin
             {if we found the proper aldevicetype, blxposition, and clyposition, then
              to here, we are pointing at the *YPOS= line in the output file.}

                  for i:=1 to numb_junk_lines_between_parameters do  {read 2 comment lines}
                       begin
                            readln(userout,parameterline);
                            if TRACE7=true then writeln(userout2,
                                 'junk parameter lines = ',parameterline);
                       end;
                  for i:=1 to (numbbsim + 1) do  {17 parameter lines plus tox,temp,vdd line}
                       begin
                            readln(userout,parameterline);          {read line from output file}
{*******************************take out later*********************}
                            if TRACE7=true then writeln(userout2,
                                 'goodparameterline = ',parameterline);
{*******************************take out later*********************}
                            k:=1;                                   {initialize character position counter}
                            parameter:='                ';          {zero out parameter string}
                            for j:=1 to (numb_chars_in_parameter*3 + 2) do {# chars dealt with
                                begin
                                     value:=parameterline[j];
                                     if TRACE8=true then writeln(
                                          'value = ',value);
                                     if (value <> ',') then                    {if = ',' ignore it}
                                          begin
                                               parameter[k]:=value;
                                               k:=k+1;
                                               if TRACE8=true then writeln(
                                                    'parameter = ',parameter);
                                          end;
                                     if j=numb_chars_in_parameter then {at end of 1st param}
                                          begin
                                               p0:=strtoreal(parameter);
                                               k:=1;
                                               parameter:='                ';
                                          end;
                                     if j=(numb_chars_in_parameter*2 + 1) then {at end of 2nd p
                                          begin
                                               pl:=strtoreal(parameter);
                                               k:=1;
                                               parameter:='                ';
                                          end;
                                     if j=(numb_chars_in_parameter*3 + 2) then {at end of 3rd p
                                          pw:=strtoreal(parameter);
                                end;
                            if TRACE8=true then writeln(userout2,
                                 'about to fill up 1 line of temporary array');
                            if i<>numbbsim+1 then
                                 begin
                                      processpar[1,i]:=p0;          {fill up temporary array with}
                                      processpar[2,i]:=pl;              {process file params}
                                      processpar[3,i]:=pw;
                                 end;
                       end;
                  if (graph_to_plot='3') and (third_variable_value+
                                      processpar[2,betazeroparam]<=0.0) then
```

```
                    prepare_for_po_graphics_error:=3;
                if (graph_to_plot='4') and (third_variable_value+
                                        processpar[3,betazeroparam]<=0.0) then
                    prepare_for_po_graphics_error:=4;
            end
        else
            begin
                prepare_for_po_graphics_error:=2:  {devicetype not found at wanted_xpos and
                if TRACE7=true then writeln(userout2,
                    'we got into error = 2 part somehow');
            end;
    end
else
    prepare_for_po_graphics_error:=1;  {output file is empty!!}
if prepare_for_po_graphics_error=0 then  {convert beta params to}
    begin                                        {mu params * u1 params to}
        i:=0;                                   {u1 x length}
        repeat
            i:=i+1;
            for j:=1 to numbbsim do
{JRP}           begin
                    if (j=9) or (j=14) or (j=15) or (j=16) then
                        param[i,j]:=param[i,j] /cox*
                                        (11[i]+processpar[2,betazeroparam]) /
                                        (ww[i]+processpar[3,betazeroparam]);
{JRP}               if (j=8) or (j=13) or (j=17) then
{JRP}                   param[i,j]:=param[i,j]*
{JRP}                                   (11[i]+processpar[2,betazeroparam]);
{JRP}           end;
        until i=numb_sets_17_params;
    end;
end;  {of procedure prepare_for_po_graphics}
```

grsetup

```
procedure grsetup(var init_ok:boolean);
{this procedure is used to initialize the graphics library and a user
specified output device.}

var display_address :integer;
    control:integer;
    error_return:integer;

begin
    graphics_init;
    writeln(#12);                   {#12 clears the CRT}
    display_address:=3;
    {initialize the graphics display device}
    display_init(display_address, control, error_return);
    if error_return<>0 then
        begin
            init_ok:=false;
            gotoxy(0,22);
            writeln('Display initialization error #',error_return:1)
        end
    else
        init_ok:=true;
end;  {end of graphics setup}
```

maxminxy

```
procedure maxminxy(var xmax,xmin,ymax,ymin:real;var maxmin_error:integer);
```

...maxminxy

```
{this procedure checks all measured data and all simulated data in determining
the maximum and minimum values to be used for autoscaling}

var   i,j:integer;
      yvalue:real;
      xvalue:real;
      minl,minw,maxl,maxw:real:  {min & max W & L, such that l+deltal or
                                      w+deltaw is > 0, to be plotted this graph}
```

ymin_ymax_check

```
 procedure ymin_ymax_check:

 begin
      if yvalue>ymax then   {if it is larger than previous ones}
           ymax:=yvalue;    {then keep it}
      if yvalue<ymin then   {if it is smaller than previous ones}
           ymin:=yvalue;    {then keep it}
 end;



begin
     maxmin_error:=0;
     {*******************take out later**********************************}
     if TRACE15=true then writeln(userout2,'entering maxminxy');
     {*******************take out later**********************************}
     {1st find the smallest W or L.  Call it xmin.
      then set xmax=1/smallest=1/xmin.  No need to find largest device for
      scaling purposes because 1/largest =0.
      Therefore, xmin=0 always because we will plot from 0 to some value}
     xmin:=1.0E20:  {sets the minimum value for x > largest possible device}
     ymax:=0.0;
     ymin:=0.0;  {clears the maximum and minimum values for y}
     if TRACE2=true then
          begin
               for i:=1 to numb_sets_17_params do
                    begin
                         writeln('ww[i]= ',ww[i]:2:2);
                         writeln('ll[i]= ',ll[i]:2:2);
                    end;
          end; {of TRACE=2 ....will write out w and l values to screen}
     maxw:=0;
     maxl:=0;              {about to find largest and smaliest W and L}
     minw:=1E20;           {so set initial values to extremes}
     minl:=1E20;
     if (graph_to_plot='1') or (graph_to_plot='3') then
          delta_1_or_w:=processpar[3,betazeroparam]      {=delta W}
     else    {graph_to_plot='2' or '4'}
          delta_1_or_w:=processpar[2,betazeroparam];     {=delta L}
     for i:=1 to numb_sets_17_params do  {find max and min W & L}
          begin
               if ww[i]>maxw then
                    maxw:=ww[i];
               if ll[i]>maxl then
                    maxl:=ll[i];
               if (ww[i]<minw) and (ww[i]+processpar[3,betazeroparam]>0.0) then
                    minw:=ww[i];
               if (ll[i]<minl) and (ll[i]+processpar[2,betazeroparam]>0.0) then
                    minl:=ll[i];
          end;
{*******************take out later**********************************}
     if TRACE15=true then writeln(userout2,'minl=',minl:2,
                                  'maxl=',maxl:2,'minw=',minw:2,' maxw=',maxw:2);
     if TRACE15=true then writeln(userout2,'delta_1_or_w=',delta_1_or_w:4:4);
```

```
{*******************take out later*********************************}
     for i:=1 to numb_sets_17_params do
          begin
     {*******************take out later*********************************}
               if TRACE2=true then writeln(userout2,
                    'entering 1 of 17 & graph_to_plot=',
                     graph_to_plot,'& 3rd_var_val=',third_variable_value);
     {*******************take out later*********************************}
               if ((graph_to_plot='1') and (ww[i]+delta_1_or_w>0.0)) or
                  ((graph_to_plot='2') and (ll[i]+delta_1_or_w>0.0)) or
                  ((graph_to_plot='3') and (ll[i]=third_variable_value) and
                   (ww[i]+delta_1_or_w>0.0)) or
                  ((graph_to_plot='4') and (ww[i]=third_variable_value) and
                   (ll[i]+delta_1_or_w>0.0)) then
                  begin
                       if (param_to_graph=betazeroparam) and
                            ( (graph_to_plot='1') or (graph_to_plot='3') ) then
{jrp}                       if param[i,betazeroparam]=0.0 then
{jrp}                            yvalue:=1E100
{jrp}                       else
                                 yvalue:=1 /param[i,betazeroparam]   {plot 1/beta0}
                       else
                            yvalue:=param[i,param_to_graph];  {get parameter}
                        ymin_ymax_check:
                        if (graph_to_plot='1') or (graph_to_plot='3') then
                             xvalue:=ww[i]        {graph W on x-axis}
                        else    {graph_to_plot='2' or '4'}
                             xvalue:=ll[i];    {graph L on x-axis}
                        if (xvalue<xmin) then
                             xmin:=xvalue;    {then keep it}

                  end;
          end;  {of 1 of 17 for statement}
     {*******************take out later*********************************}
     if TRACE15=true then writeln(userout2,'after param[] check & ',
                                            'xmin= ',xmin,' xmax= ',xmax,
                                            ' ymin= ',ymin,' ymax= ',ymax);
     {*******************take out later*********************************}
     if xmin=1E20 then
          maxmin_error:=1;
     if maxmin_error=0 then
          begin
               xmax:=1 /(xmin+delta_1_or_w);
               xmin:=0;  {now set it =0 for graphing purposes}
          {*******************take out later*********************************}
               if TRACE2=true then writeln(userout2,
                         'OUT OF 1 of 17 & xvalue =',xvalue,'xmin= ',xmin,'xmax= ',xmax,
                                           'yvalue =',yvalue,'ymin= ',ymin,'ymax= ',ymax);
          {*******************take out later*********************************}
               {now get p0, pl, and pw from process file and generate min and max
                simulated values with them.}
               if (param_to_graph=betazeroparam) then   {beta0 parameter}
                    {since beta0 is always > 0. a plot of beta vs 1/L or a plot of
                     1/beta vs 1/W will always have one point at the origin 0,0.
                     therefore. only the maximum yvalue needs to be checked}
                    begin
                         if graph_to_plot='1' then   {plot 1/beta0 vs 1/W}
                              begin
                                   yvalue:=1 /processpar[1,betazeroparam] / cox * xmax *
                                             (max1+processpar[2,betazeroparam]);
                                   if yvalue>ymax then
                                        ymax:=yvalue;              {if > previous one. keep it}
```

```
{**********take out later**********}
    if TRACE2=true then writeln(userout2,'beta0param & gtp=',graph_to_plot,
                                ' yvalue =',yvalue,' ymin= ',ymin,' ymax= ',ymax);
{**********take out later**********}
                end;
            if graph_to_plot='2' then  {plot beta vs 1/L}
                begin
                    yvalue:=processpar[1,betazeroparam] * cox * xmax *
                            (maxw+processpar[3,betazeroparam]);
                    if yvalue>ymax then
                        ymax:=yvalue;              {if > previous one, keep it}
{**********take out later**********}
    if TRACE2=true then writeln(userout2,'beta0param & gtp=',graph_to_plot,
                                ' yvalue =',yvalue,' ymin= ',ymin,' ymax= ',ymax);
{**********take out later**********}
                end;
            if graph_to_plot='3' then  {plot 1/beta0 vs 1/W}
                begin
                    yvalue:=1 /processpar[1,betazeroparam] / cox * xmax *
                            (third_variable_value+processpar[2,betazeroparam]);
                    if yvalue>ymax then
                        ymax:=yvalue;              {if > previous one, keep it}
{**********take out later**********}
    if TRACE2=true then writeln(userout2,'beta0param & gtp=',graph_to_plot,
                                ' yvalue =',yvalue,' ymin= ',ymin,' ymax= ',ymax);
{**********take out later**********}
                end;
            if graph_to_plot='4' then  {plot beta0 vs 1/L}
                begin
                    yvalue:=processpar[1,betazeroparam] * cox * xmax *
                            (third_variable_value+processpar[3,betazeroparam]);
                     if yvalue>ymax then
                        ymax:=yvalue;              {if > previous one, keep it}
{**********take out later**********}
    if TRACE2=true then writeln(userout2,'beta0param & gtp=',graph_to_plot,
                                ' yvalue =',yvalue,' ymin= ',ymin,' ymax= ',ymax);
{**********take out later**********}
                end;
        end    {of beta0 checking}
    else   {param_to_graph <>beta0param}
        begin
            if graph_to_plot='1' then  {check all 4 combos of L & W}
                begin
{**********take out later**********}
                    if TRACE15=true then writeln(userout2,
                        'NOT beta0param & gtp=',graph_to_plot);
{**********take out later**********}
                    yvalue:=processpar[1,param_to_graph] +
                            (processpar[2,param_to_graph] /
                            (minl + processpar[2,betazeroparam])) +
                            (processpar[3,param_to_graph] * xmax);
                    ymin_ymax_check;
{**********take out later**********}
                    if TRACE15=true then writeln(userout2,
                        ' yvalue =',yvalue,' ymin= ',ymin,' ymax= ',ymax);
{**********take out later**********}
                    yvalue:=processpar[1,param_to_graph] +
                            (processpar[2,param_to_graph] /
                            (minl + processpar[2,betazeroparam]));
                    ymin_ymax_check;
{**********take out later**********}
                    if TRACE15=true then writeln(userout2,
                        ' yvalue =',yvalue,' ymin= ',ymin,' ymax= ',ymax);
```

```
{**********take out later**********}
                        yvalue:=processpar[1,param_to_graph] +
                                (processpar[2,param_to_graph] /
                                (maxl + processpar[2,betazeroparam])) +
                                (processpar[3,param_to_graph] * xmax);
                        ymin_ymax_check;
{**********take out later**********}
                        if TRACE15=true then writeln(userout2,
                            ' yvalue =',yvalue,' ymin= ',ymin,' ymax= ',ymax);
{**********take out later**********}
                        yvalue:=processpar[1,param_to_graph] +
                                (processpar[2,param_to_graph] /
                                (maxl + processpar[2,betazeroparam]));
                        ymin_ymax_check;
{**********take out later**********}
                        if TRACE15=true then writeln(userout2,
                            ' yvalue =',yvalue,' ymin= ',ymin,' ymax= ',ymax);
{**********take out later**********}
                    end; {of if graph_to_plot='1'}
                if graph_to_plot='2' then {check all 4 combos of L & W}
                    begin
                        yvalue:=processpar[1,param_to_graph] +
                                (processpar[2,param_to_graph] * xmax) +
                                (processpar[3,param_to_graph] /
                                (minw + processpar[3,betazeroparam]));
                        ymin_ymax_check;
                        yvalue:=processpar[1,param_to_graph] +
                                (processpar[3,param_to_graph] /
                                (minw + processpar[3,betazeroparam]));
                        ymin_ymax_check;
                        yvalue:=processpar[1,param_to_graph] +
                                (processpar[2,param_to_graph] * xmax) +
                                (processpar[3,param_to_graph] /
                                (maxw + processpar[3,betazeroparam]));
                        ymin_ymax_check;
                        yvalue:=processpar[1,param_to_graph] +
                                (processpar[3,param_to_graph] /
                                (maxw + processpar[3,betazeroparam]));
                        ymin_ymax_check;
{**********take out later**********}
                        if TRACE2=true then writeln(userout2,
                            'NOT beta0param & gtp=',graph_to_plot,
                            ' yvalue =',yvalue,' ymin= ',ymin,' ymax= ',ymax);
{**********take out later**********}
                    end; {of if graph_to_plot='2'}
                if graph_to_plot='3' then {check all 2 combos of L & W}
                    begin
                        yvalue:=processpar[1,param_to_graph] +
                                (processpar[2,param_to_graph] /
                                (third_variable_value + processpar[2,betazeroparam])) +
                                (processpar[3,param_to_graph] * xmax);
                        ymin_ymax_check;
                        yvalue:=processpar[1,param_to_graph] +
                                (processpar[2,param_to_graph] /
                                (third_variable_value + processpar[2,betazeroparam]));
                        ymin_ymax_check;
{**********take out later**********}
                        if TRACE2=true then writeln(userout2,
                            'NOT beta0param & gtp=',graph_to_plot,
                            ' yvalue =',yvalue,' ymin= ',ymin,' ymax= ',ymax);
{**********take out later**********}
                    end; {of if graph_to_plot='3'}
```

...maxminxy

```
                    if graph_to_plot='4' then  {check all 2 combos of L & W}
                      begin
                        yvalue:=processpar[1,param_to_graph] +
                                (processpar[2,param_to_graph] * xmax) +
                                (processpar[3,param_to_graph] /
                                (third_variable_value + processpar[3,betazeroparam]));
                        ymin_ymax_check;
                        yvalue:=processpar[1,param_to_graph] +
                                (processpar[3,param_to_graph] /
                                (third_variable_value + processpar[3,betazeroparam]));
                        ymin_ymax_check:
{******************take out later**********************************}
                        if TRACE2=true then writeln(userout2,
                          'NOT beta0param & gtp=',graph_to_plot,
                           ' yvalue =',yvalue,' ymin= ',ymin,' ymax= ',ymax);
{******************take out later**********************************}
                          end;  {of if graph_to_plot='4'}
                  end;  {of non-beta0 values}
{******************take out later**********************************}
              if TRACE15=true then writeln(userout2,
                'leaving maxminxy & ymax = ',ymax,' ymin = ',ymin);
{******************take out later**********************************}
        end;  {of if maxmin_error=0 then.....}
end;{of procedure maxminxy}
```

autoscale

```
procedure autoscale(xmax,xmin,ymax,ymin:real; xlabel,ylabel,top,topleft,
                    topright,zlabel1,zlabel2,xaxisunit,yaxisunit:graphlab);
{this procedure draws the tick marks for the graphic display, given the
maximum and minimum values for the graph}

type unitlab=string[7];

var   xunitlab,yunitlab:graphlab;
      we_are_doing_x_exponent:boolean;  {true if we are calling find_exponent
                                         for x axis}

{************************************************************************}
```

find_exponent

```
procedure find_exponent(var expon:integer; var units:graphlab; max,min:real;
                                                    this_is_xaxis:boolean);
{this procedure determines the exponent of the maximum data point for the
graph, and returns this value.  exponents in factors of 3 are used}

var newmaxim:real;
    maxormin:real;

begin
    {******************take out later**********************************}
    if TRACE1=true then writeln(userout2,'entering find_exponent');
    {******************take out later**********************************}
    expon:=0;
    if abs(max)>abs(min) then
        maxormin:=abs(max)
    else
        maxormin:=abs(min);
    newmaxim:=maxormin;
    if TRACE10=true then writeln(userout2,'set 2 values equal');
    while newmaxim>1.0E+9 do
      begin
        newmaxim:=newmaxim/1000;
        maxormin:=maxormin/1000;
        expon:=expon+3;
      end;
    if TRACE10=true then writeln(userout2,'after while statement');
    if TRACE10=true then writeln(userout2,'max =',max,'min=',min,'maxormin = ',maxormin);
```

..find_exponent

```
    if trunc(maxormin)>0 then  {number is greater than 1}
        begin
            if TRACE10=true then writeln(userout2,
                'entering "if" part of if-else and maxormin s/b > 1');
            while trunc(newmaxim/1000)>0 do
                begin
                    expon:=expon+3;
                    newmaxim:=newmaxim/1000;
                end;
            if TRACE10=true then writeln(userout2,
                'leaving "if" part of if-else and expon=',
                expon,' & newmaxim=',newmaxim);
        end
    else   {this means that the number is less than 1}
        begin
            if TRACE10=true then writeln(userout2,
                'entering "else" part of if-else and maxormin s/b < 1');
            while trunc(newmaxim)=0 do
                begin
                    expon:=expon-3;
                    newmaxim:=newmaxim*1000;
                end;
            if TRACE10=true then writeln(userout2,
                'leaving "else" part of if-else and expon=',
                expon,' & newmaxim=',newmaxim);
        end;


    units:=' ';
    if this_is_xaxis=true then
        begin
            case expon of
                0:units:='xE-3 ';          {note that exponent is offset by 3}
               -3:units:='';               {this is offset in x_axis_labels}
               -6:units:='x1000 ';
               -9:units:='x1000000 ';
            end;
        end
    else  {find_exponent was called to label the y axis}
        begin
            case expon of
                15:units:='xE+15';
                12:units:='xE+12';
                9:units:='xE+9';
                6:units:='xE+6';
                3:units:='xE+3';
                0:units:='';
                -3:units:='xE-3';
                -6:units:='xE-6';
                -9:units:='xE-9';
                -15:units:='xE-15';
            end;
        end;
    {*******************take out later*********************************}
    if TRACE1=true then writeln(userout2,'leaving find_exponent');
    {*******************take out later*********************************}
end;  {end of procedure find_exponent}
```

draw_axis

```
procedure draw_axis(xmax,xmin,ymax,ymin:real;xaxisexp,yaxisexp:integer);
{this procedure draws the axis, and makes and labels the tick marks}

var xaxislab,yaxislab,toplab,topunit,deltalab:graphlab;
    count:integer;
    cnt:integer;
```

x_axis_labels

```
procedure x_axis_labels;
{this procedure labels the x-axis}

var count:integer;
    numbxtck,numlxtck:integer;
    magbxtck:real;
    xpos:real;
    xstring:string[40];
    xtickval:real;
    cnt:integer;
    i:integer;
    littltck:real;
    range:real;
    xoffset:real;
    firstxtk,lastxtk:real;
    maglxtck,firslxtk:real;
    latelxtk:real;
    rangefac:real;
    temp_x_value:real;  {= 1/xstring = W or L so x-axis values are printed
                           as microns instead of 1/microns}
    newxstring:string[40];  {contains new 1/xstring to be plotted}
    j:integer;  {loop counter}
    temp_delta_l_or_w:real;  {=delta_l_or_w * E to the exponent}


begin
    {*******************take out later***********************************}
    if TRACE1=true then writeln(userout2,'entering xaxis_labels');
    {*******************take out later***********************************}
     temp_delta_l_or_w:=delta_l_or_w*1000;
     if xaxisexp>0 then
          for count:=1 to xaxisexp do
               begin
                    xmax:=xmax/10;
                    xmin:=xmin/10;
                    temp_delta_l_or_w:=temp_delta_l_or_w*10;
               end
     else if xaxisexp=0 then
          xmax:=xmax
     else
          for count:=1 to -(xaxisexp) do
               begin
                    xmax:=xmax*10;
                    xmin:=xmin*10;
                    temp_delta_l_or_w:=temp_delta_l_or_w/10;
               end;
     range:=xmax-xmin;
     rangefac:=1;
    {*******************take out later***********************************}
    if TRACE9=true then writeln(userout2,'at point 1');
    {*******************take out later***********************************}
     while range>10 do
          begin
               range:=range/10;
               rangefac:=rangefac*10;
          end;
     while range<1 do
          begin
               range:=range*10;
               rangefac:=rangefac/10;
          end;
     if range>6 then
          begin
```

```
            magbxtck:=1*rangefac;
            numlxtck:=3;
        end
  else if range>3 then
        begin
            magbxtck:=0.5*rangefac;
            numlxtck:=4;
        end
  else
        begin
            magbxtck:=0.25*rangefac;
            numlxtck:=4;
        end;
  range:=range*rangefac;
{*****************take out later*********************************}
if TRACE9=true then writeln(userout2,'at point 2');
{*****************take out later*********************************}
  {this is where the tick marks are drawn and labeled}
  if xmax>0 then
        lastxtk:=magbxtck*trunc((xmax+magbxtck)/magbxtck)-magbxtck
  else
        lastxtk:=-magbxtck*trunc((abs(xmax)+magbxtck)/magbxtck);
  if xmin>0 then
        firstxtk:=magbxtck*trunc((xmin+magbxtck)/magbxtck)
  else
        firstxtk:=-magbxtck*trunc((abs(xmin)+magbxtck)/magbxtck)+magbxtck;
  xoffset:=((firstxtk-xmin)/range)*1.6;
  numbxtck:=round((lastxtk-firstxtk)/magbxtck)+1;
  xtickval:=firstxtk;
{*****************take out later*********************************}
  if TRACE9=true then
       begin
            writeln(userout2,'firstxtk=',firstxtk,'lastxtk=',
                 lastxtk,'range=',range,'numbxtck=',numbxtck);
            writeln(userout2,'THIS VALUE S/B 25.0--magbxtck=',magbxtck,'xoffset=',
                 xoffset);
            writeln(userout2,'xpos is now given a bogus value to start with');
       end;
xpos:=120.0;
{*****************take out later*********************************}
  set_char_size(0.02,0.05);
  move(-1.00,-0.85);
  gtext('EFFECTIVE');
  move(-1.00,-0.92);
  gtext('DRAWN');
  for i:=1 to numbxtck do
        begin
{*****************take out later*********************************}
  if TRACE9=true then
       begin
            writeln(userout2,'at the beginning of the for loop, xpos= ',xpos,
                 'AND');
            writeln(userout2,'i=',i:2,'xoffset=',xoffset,'numbxtck=',
                 numbxtck);
            writeln(userout2,'lastxtk=',lastxtk,'firstxtk=',firstxtk,'range=',range);
       end;
{*****************take out later*********************************}
              xpos:=(i-1)*1.6/(numbxtck-1)*((lastxtk-firstxtk)/range);
              xpos:=xpos-0.8+xoffset;
{*****************take out later*********************************}
              if TRACE9=true then writeln(userout2,
                   'created a new xpos ---- xpos= ',xpos);
```

```
{*******************take out later*************************************}
                move(xpos,-0.8);  {move to position for big tick}
                line(xpos,-0.75);   {draws x tick}
                set_line_style(7);  {dotted line}
                line(xpos,0.8);
                set_line_style(1);    {full line}
                move(xpos-0.08,-0.85);    {moves to location for label}
                set_char_size(0.02,0.05);
                temp_x_value:=xtickval:    {used to generate x label in microns}
{*******************take out later*************************************}
                if TRACE13=true then writeln(userout2,
                    'numbxtck=',numbxtck:3,'xmax=',xmax:3:3);
{*******************take out later*************************************}
                if temp_x_value<>0.0 then
                    begin  {label EFFECTIVE x axis sizes}
                        temp_x_value:=1000.0/temp_x_value;
                        newxstring:='      ';
                        if (numbxtck>6) and (temp_x_value>=100) then   {writes the string}
                            strwrite(newxstring,4,cnt,temp_x_value:3:1)
                        else   {writes the string}
                            strwrite(newxstring,4,cnt,temp_x_value:3:2);
{*******************take out later*************************************}
                        if TRACE9=true then writeln(userout2,
                                'newxstring= ',newxstring);
{*******************take out later*************************************}
                        gtext(newxstring);  {writes out the string}
                        {now convert to DRAWN SIZE and label.
                         scale is linear with respect to EFFECTIVE size}
                      move(xpos-0.08,-0.92);    {moves to location for label}
                        temp_x_value:=xtickval:    {used to generate x label in microns}
                        {look above for temp_delta_l_or_w calculation}
                        temp_x_value:=1000.0/temp_x_value - temp_delta_l_or_w;
                        newxstring:='      ';
                        if (numbxtck>6) and (temp_x_value>=100) then   {writes the string}
                            strwrite(newxstring,4,cnt,temp_x_value:3:1)
                        else   {writes the string}
                            strwrite(newxstring,4,cnt,temp_x_value:3:2);
{*******************take out later*************************************}
                        if TRACE9=true then writeln(userout2,
                                'newxstring= ',newxstring);
{*******************take out later*************************************}
                        gtext(newxstring);  {writes out the string}
                    end;  {if temp_x_value<>0}
                xtickval:=xtickval+magbxtck;
{*******************take out later*************************************}
                if TRACE9=true then writeln(userout2,
                    'at the END of the for loop, xpos= ',xpos,'xtickval=',xtickval);
{*******************take out later*************************************}
        end;
 maglxtck:=magbxtck/(numlxtck+1);
 if xmin>0 then
        firslxtk:=maglxtck*trunc((xmin+maglxtck)/maglxtck) {little tick}
 else
        firslxtk:=-maglxtck*trunc((abs(xmin)+maglxtck)/maglxtck)+maglxtck;
 xoffset:=(firslxtk-xmin)/range*1.6;
 latelxtk:=firslxtk;
 while latelxtk<xmax do
        begin
                xpos:=-0.8+xoffset+(latelxtk-firslxtk)/range*1.6;
                move(xpos,-0.8);
                line(xpos,-0.78);
                latelxtk:=latelxtk+maglxtck;
        end;
```

...x_axis_labels

```
    {*******************take out later*********************************}
    if TRACE1=true then writeln(userout2,'leaving xaxis_labels');
    {*******************take out later*********************************}
 end;  {end of procedure x_axis_labels}
```

y_axis_labels

```
procedure y_axis_labels;
{this procedure labels the y-axis}

var count:integer;
    numbytck,numlytck:integer;
    magbytck:real;
    ypos:real;
    ystring:string[40];
    ytickval:real;
    cnt:integer;
    i:integer;
    littltck:real;
    range:real;
    yoffset:real;
    firstytk,lastytk:real;
    maglytck,firslytk:real;
    latelytk:real;
    rangefac:real;


begin
    {*******************take out later*********************************}
    if TRACE1=true then writeln(userout2,'entering yaxis_labels');
    {*******************take out later*********************************}
     if yaxisexp>0 then
          for count:=1 to yaxisexp do
               begin
                    ymax:=ymax/10;
                    ymin:=ymin/10;
               end
     else if yaxisexp=0 then
          ymax:=ymax
     else
          for count:=1 to -(yaxisexp) do
               begin
                    ymax:=ymax*10;
                    ymin:=ymin*10;
               end;
     range:=ymax-ymin;
     rangefac:=1;
     while range>10 do
          begin
               range:=range/10;
               rangefac:=rangefac*10;
          end;
     while range<1 do
          begin
               range:=range*10;
               rangefac:=rangefac/10;
          end;
     if range>7 then
          begin
               magbytck:=2*rangefac;
               numlytck:=3;
          end
```

```
        else if range>3 then   {3<range<7}
                begin
                        magbytck:=1*rangefac;
                        numlytck:=4;
                end
        else                    {1<range<3}
                begin
                        magbytck:=0.5*rangefac;
                        numlytck:=4;
                end;
        range:=range*rangefac;
        {this is where the tick marks are drawn and labeled}
        if ymax>0 then
                lastytk:=magbytck*trunc((ymax+magbytck)/magbytck)-magbytck
        else
                lastytk:=-magbytck*trunc((abs(ymax)+magbytck)/magbytck);
        if ymin>0 then
                firstytk:=magbytck*trunc((ymin+magbytck)/magbytck)
        else
                firstytk:=-magbytck*trunc((abs(ymin)+magbytck)/magbytck)+magbytck;
        yoffset:=((firstytk-ymin)/range)*1.6;
        numbytck:=round((lastytk-firstytk)/magbytck)+1;
        ytickval:=firstytk;

        for i:=1 to numbytck do
                begin
                        ypos:=(i-1)*1.6/(numbytck-1)*((lastytk-firstytk)/range);
                        ypos:=ypos-0.8+yoffset;
                        move(-0.8,ypos);   {position for y tick}
                        line(-0.75,ypos);  {draws y tick}
                        set_line_style(7);  {dotted line}
                        line(0.8,ypos);   {draws dotted line}
                        set_line_style(1);     {full line}
                        move(-0.93,ypos);     {moves to location for label}
                        ystring:='      ';
                        strwrite(ystring,1,cnt,ytickval:1:2);  {writes the string}
                        set_char_size(0.02,0.05);
                        gtext(ystring);     {plots the string}
                        ytickval:=ytickval+magbytck;
                end;
        maglytck:=magbytck/(numlytck+1);
        if ymin>0 then
                firslytk:=maglytck*trunc((ymin+maglytck)/maglytck)   {little tick}
        else
                firslytk:=-maglytck*trunc((abs(ymin)+maglytck)/maglytck)+maglytck;
        yoffset:=(firslytk-ymin)/range*1.6;
        latelytk:=firslytk;
        while latelytk<ymax do
                begin
                        ypos:=-0.8+yoffset+(latelytk-firslytk)/range*1.6;
                        move(-0.8,ypos);
                        line(-0.78,ypos);
                        latelytk:=latelytk+maglytck;
                end;
    {**********take out later**********}
    if TRACE1=true then writeln(userout2,'leaving yaxis_labels');
    {**********take out later**********}
end;    {end of procedure y_axis_labels}


begin {procedure draw_axis begins here}
```

```
        {this is where the axis are drawn}
        move(-0.8,-0.8);
        line(-0.8,0.8);
        line(0.8,0.8);
        line(0.8,-0.8);
        line(-0.8,-0.8);
        x_axis_labels;  {draws the tick marks and labels for the x axis}
        y_axis_labels;  {draws the ticks and labels for the y axis}
        set_char_size(0.030,0.060);
        xaxislab:=' ';      {clears the label}
        strappend(xaxislab,xlabel);
        strappend(xaxislab,xunitlab);
        strappend(xaxislab,xaxisunit);
        move(-strlen(xaxislab)*0.0125,-1.0);
        gtext(xaxislab);
        set_text_rot(0.0,1.0);
        yaxislab:=' ';      {clears the label}
        strappend(yaxislab,ylabel);
        strappend(yaxislab,yunitlab);
        strappend(yaxislab,yaxisunit);
        move(-0.96,-strlen(yaxislab)*0.0125);
        gtext(yaxislab);
        set_text_rot(1.0,0.0);
        move(0.80,0.95);
        gtext(zlabel1);
        move(0.70,0.90);
        gtext(zlabel2);
        if (graph_to_plot='1') or (graph_to_plot='3') then
            topunit:='W'
        else
            topunit:='L';
        toplab:=' ';        {clears the label}
        strappend(toplab,top);
        strappend(toplab,topunit);
        move(-strlen(toplab)*0.025,0.95);
        set_char_size(0.05,0.1);
        gtext(toplab);
        move(-1.0,0.95);
        set_char_size(0.035,0.075);
        gtext('BSIM1.0');
{jrp}move(-1.0,0.88);
{jrp}date:=strrtrim(date);
{jrp}gtext(date);
        deltalab:=' ';
        strappend(deltalab,topleft);
        strwrite(deltalab,9,cnt,processpar[2,betazeroparam]:3:2);
        set_char_size(0.030,0.060);
        move(-strlen(toplab)*0.025,0.88);
        gtext(deltalab);
        deltalab:=' ';
        strappend(deltalab,topright);
        strwrite(deltalab,9,cnt,processpar[3,betazeroparam]:3:2);
        move(strlen(toplab)*0.025-strlen(deltalab)*0.030,0.88);
        gtext(deltalab);
end;    {end of procedure draw_axis which draws and labels axis}


begin   {main program autoscale begins here}
    {*******************take out later*********************************}
        if TRACE1=true then writeln(userout2,'entering autoscale');
    {*******************take out later*********************************}
        we_are_doing_x_exponent:=true;
        find_exponent(xaxisexp,xunitlab,xmax,xmin,we_are_doing_x_exponent);
        we_are_doing_x_exponent:=false;
        find_exponent(yaxisexp,yunitlab,ymax,ymin,we_are_doing_x_exponent);
```

...autoscale

```
      draw_axis(xmax,xmin,ymax,ymin,xaxisexp,yaxisexp);  {draws and labels}
     {*******************take out later**********************************}
      if TRACE1=true then writeln(userout2,'leaving autoscale');
     {*******************take out later**********************************}
end;  {end of procedure autoscale}
```

draw_the_data1

```
procedure draw_the_data1(xmax,xmin,ymax,ymin:real);
{this procedure draws in the measured data}
{last change made: MONDAY OCT 15, 1984   By: JRP}

var   i,j:integer;
      yval,xval:real;
      xnew,ynew:real;
      xold,yold:real;
      cnt:integer;
      zstring:string[6];
      label:boolean;
      slope,test:real;
      outplane:boolean;
      {***************************these were added**********}
      next_plot_char,plot_char:string[1];  {character used for plotting}
      size,next_size:real;  {z axis size}
      number_checked:integer;
      number_different_sizes:integer;
      plot_char_check:integer;  {= # diff sizes or # diff sizes - 10}
      still_looking_for_next_size:boolean:  {more z values to plot?}
      first_time_for_this_size:boolean:  {set false after we find the 1st
                                          occurence of the current size}
      plotted_point:array[1..100] of boolean:  {if true, point has been plotted
                                                --or attempted to be plotted}
      zstring_pointer:integer;  {number of zstrings already labeled}
      current_size_plotted:boolean:  {= true if at least one point with current
                                      zvalue has been plotted}
```

plot_point

```
procedure plot_point;
{this procedure simply moves the graphics position to xnew, ynew (with a
 little offset due to character size) and plots the point}

begin
     move(xnew-0.0115,ynew-0.020);
     gtext(plot_char);
     current_size_plotted:=true;
end;


begin  {procedure plot_the_data1}
     {*******************take out later**********************************}
  if TRACE16=true then writeln(userout2,
       'entering draw_the_data1 & xmax=',xmax:2:2,' ymax=',ymax:2:2,
                                  'xmin=',xmin:2:2,' ymin=',ymin:2:2);
     {*******************take out later**********************************}
  set_char_size(0.05,0.12);
  if (graph_to_plot='3') or (graph_to_plot='4') then
       next_size:=third_variable_value          {user input the z value to be plotted}
  else if (graph_to_plot='1') then
       next_size:=ll[1]                            {use first value of L[] array}
  else if (graph_to_plot='2') then
       next_size:=ww[1];                           {use first value of W[] array}
  number_different_sizes:=1;
  number_checked:=0;
  next_plot_char:='x';
  still_looking_for_next_size:=false;
  zstring_pointer:=0;
  for i:=1 to numb_sets_17_params do
```

```
      plotted_point[i]:=false;
    repeat{ until all sets of data points are finished }
      current_size_plotted:=false;
      size:=next_size;
      first_time_for_this_size:=true;
      plot_char:=next_plot_char;
      for i:=1 to numb_sets_17_params do
        begin
      {**********take out later**********}
            if TRACE2=true then writeln(userout2,
                 'i=',i:2,' FOR LOOP BEGIN ww[i]=',
                 ww[i]:2,' ll=',ll[i]:2,' size=',size:2);
      {**********take out later**********}
            if ( ((graph_to_plot='2') or (graph_to_plot='4')) and (ww[i]=size) ) or
               ( ((graph_to_plot='1') or (graph_to_plot='3')) and (ll[i]=size) ) then
                 begin
{jrp}              set_char_size(0.05,0.12);
      {**********take out later**********}
                  if TRACE2=true then writeln(userout2,
                       'i=',i:2,' & found graph & size ww[i]=',
                       ww[i]:2,' ll[i]=',ll[i]:2,' size=',size:2);
      {**********take out later**********}
                  if (still_looking_for_next_size=false) and
                     (first_time_for_this_size=true) then {found 1st one of}
                       begin                                                {next size}
                            still_looking_for_next_size:=true;
                            first_time_for_this_size:=false;
                       end;
                  if (graph_to_plot='2') or (graph_to_plot='4') then {get xval}
                       xval:=1.0 /(ll[i] + delta_l_or_w)
                  else    { (graph_to_plot='1') or (graph_to_plot='3') }
                       xval:=1.0 /(ww[i] + delta_l_or_w);
                  if (param_to_graph=betazeroparam) and
                     ( (graph_to_plot='1') or (graph_to_plot='3') ) then
                       {plot 1/beta0 vs 1/W}
{jrp}                  if param[i,betazeroparam]=0.0 then
{jrp}                       yval:=1E100
{jrp}                  else
                            yval:=1 /param[i,betazeroparam]
                  else
                       yval:=param[i,param_to_graph];{JOEJOEJOE}
    {**********take out later**********}
                  if TRACE16=true then writeln(userout2,
                       'ww[',i:1,']=',ww[i]:2:2,' ll[',i:1,']=',ll[i]:2:2,
                       ' yval=',yval);
    {**********take out later**********}
                  {if xaxisexp>0 then        change xval due to x axis exponent
                   for j:=1 to xaxisexp do
                        xval:=xval/10;}
                  {if xaxisexp<0 then        change xval due to x axis exponent
                   for j:=1 to xaxisexp do
                        xval:=xval*10;}
                  {if yaxisexp>0 then        change yval due to y axis exponent
                   for j:=1 to yaxisexp do
                        yval:=yval/10;}
                  {if yaxisexp<0 then        change yval due to y axis exponent
                   for j:=1 to yaxisexp do
                        yval:=yval*10;}
                  {if (param_to_graph=betazeroparam) and
                      ( (graph_to_plot='1') or (graph_to_plot='3') ) then
                      plot 1/beta0 vs 1/W
                        begin
                            if yaxisexp<0 then
                                 yval:=yval/1000;
                            if yaxisexp>0 then
```

```
                                    yval:=yval*1000;
                                end;}
    {*********************************take out later*********************}
                        if TRACE14=true then writeln(userout2,
                            'after exponent division/multplication',
                            ' param[',i:2,',6]=',param[i,betazeroparam],
                            ' yval=',yval);
    {*********************************take out later*********************}
                        xnew:=-0.8+1.6*((xval-xmin)/(xmax-xmin)); {change to world}
                        ynew:=-0.8+1.6*((yval-ymin)/(ymax-ymin)); {coordinate system}
    {*********************************take out later*********************}
                        if TRACE16=true then writeln(userout2,
                            'after exponent adjust & ',' yval=',yval,' ynew=',ynew);
    {*********************************take out later*********************}
                        {check whether point will be on screen}
                        if ((xnew>=-0.8) and (xnew<=0.8) and (ynew>=-0.8) and (ynew<=0.8))then
{jrp}                       plot_point;
{jrp}{deleted large section here and left only the above line}
                        xold:=xnew;
                        yold:=ynew;
                        number_checked:=number_checked+1;
                        plotted_point[i]:=true;    {shows that this point has been
                                                          plotted or attempted to be plotted}
                    end {of "if" part of     if ww[i]=size...else...}
                else if (still_looking_for_next_size=true) then
                    begin
    {*********************take out later*************************************}
                        if TRACE2=true then writeln(userout2,
                            'i=',i:2,
                            ' & didnt find correct graph & size & still_looking is true');
    {*********************take out later*************************************}
                        if (plotted_point[i]=false) then
                            begin
                                if ((graph_to_plot='2') or (graph_to_plot='4')) then
                                    next_size:=ww[i]          {graph is vs. 1/L}
                                else  {graph_to_plot = '1' or '3'}
                                    next_size:=ll[i];         {graph is vs. 1/W}
                                still_looking_for_next_size:=false;
                                number_different_sizes:=number_different_sizes+1;
                            end;
                        plot_char_check:=number_different_sizes;
                        while plot_char_check>=10 do  {maximum of 10 diff plot chars}
                            plot_char_check:=plot_char_check-10; {so subtract 10 to}
                        case plot_char_check of                  {yield single digit}
                          1:next_plot_char:='x';
                          2:next_plot_char:='o';
                          3:next_plot_char:='e';
                          4:next_plot_char:='a';
                          5:next_plot_char:='c';
                          6:next_plot_char:='s';
                          7:next_plot_char:='u';
                          8:next_plot_char:='v';
                          9:next_plot_char:='n';
                          0:next_plot_char:='z';
                        end;  {of case of plot_char_check}
                    end;  {of "else" part of     if ww[i]=size...else...  }
            end;
        if current_size_plotted=true then
            begin
{jrp}           set_char_size(0.05,0.12);
                zstring:='  ';
                strwrite(zstring,1,cnt,size:5:1);
                move(0.81,0.75-(0.06*(zstring_pointer-1)));
                gtext(plot_char);
```

...draw_the_data1

```
{jrp}          set_char_size(0.03,0.06);
                 move(0.85,0.75-(0.06*(zstring_pointer-1)));
                 gtext(zstring);
                 zstring_pointer:=zstring_pointer+1;
               end;  {of if current_size_plotted=true}
     until (((graph_to_plot='1') or (graph_to_plot='2')) and
                                          (number_checked=numb_sets_17_params)) or
               (graph_to_plot='3') or (graph_to_plot='4');
                                          {...until all lines are finished}
     {******************take out later**********************************}
     if TRACE16=true then writeln(userout2,'leaving draw_the_data1');
     {******************take out later**********************************}
end;{end of procedure drawdata1}
```

draw_the_data2

```
procedure draw_the_data2(xmax,xmin,ymax,ymin:real);
{this procedure draws in the simulated data.  It does this by finding the
 yvalue at the extreme left and right side of the graph, and connecting
 the 2 points.  This is acceptable since the equation being plotted is
 param=p0 + pL/Leff + pW/Weff which is a straight line for either fixed W or L.}
{last change made: SUNDAY OCT 14, 1984  By: JRP}

var   i,j:integer;
      yval,xval:real;
      xnew,ynew:real;
      xold,yold:real;
      cnt:integer;
      size:real;  {z axis size}
      zstring:string[6];
      label:boolean;
      slope,test:real;
      outplane:boolean;
      number_checked:integer;
      plotted_point:array[1..140] of boolean:  {if true, point has been plotted}
      first_point_found:boolean;  {= true when 1st of 2 points to be plotted
                                   has been found--used only if zoom does not
                                   include either of the 2 simulated points}
      found_third_variable_value:boolean;  {we will do repeat loop until this}
                                           {is finally set true.  this shows}
                                           {that for graph_to_plot=3or4, we}
                                           {have now found a line to draw}
```

draw_line_and_zstring

```
procedure draw_line_and_zstring(xline,yline,xstrg,ystrg:real);
{this procedure draws the data line and then labels the zvalue size and
 the plot}

begin
     line(xline,yline);
     zstring:='  ';
     if (graph_to_plot='1') or (graph_to_plot='3') then
          strwrite(zstring,1,cnt,ll[i]:5:1)
     else {(graph_to_plot='2') or (graph_to_plot='4')}
          strwrite(zstring,1,cnt,ww[i]:5:1);
     move(xstrg,ystrg);
     gtext(zstring);
end;


begin
     {******************take out later**********************************}
  if TRACE1=true then writeln(userout2,
       'entering draw_the_data2 and xmax=',xmax:2:2,' ymax=',ymax:2:2);
```

```
    {**********take out later**********}
  found_third_variable_value:=false;
  set_char_size(0.03,0.06);
  number_checked:=0;
  for i:=1 to numb_sets_17_params do
      plotted_point[i]:=false;
  i:=0;  {initialize # of times thru repeat loop}
  repeat {until all values of w and l have been plotted}
    {**********take out later**********}
      if TRACE2=true then writeln(userout2,
          'entering repeat and number checked=',number_checked:2);
      if TRACE2=true then writeln(userout2,
          'When number checked=number_sets_17_params, s/b last time thru repeat');
    {**********take out later**********}
      i:=i+1;
      if (plotted_point[i]=false) and
         ( (graph_to_plot='1') or
           (graph_to_plot='2') or
           ((graph_to_plot='3') and (third_variable_value=ll[i])) or
           ((graph_to_plot='4') and (third_variable_value=ww[i])) ) then
         begin  {AAAA}
           found_third_variable_value:=true;
           if (graph_to_plot='2') or (graph_to_plot='4') then
              begin  {BBBB}
                size:=ww[i];
{**********take out later**********}
                if TRACE5=true then writeln(userout2,
                    'Entering gtp=2or4 & size=',size:2);
{**********take out later**********}
                for j:=i to numb_sets_17_params do
                    begin
                      if size=ww[j] then
                        begin
                          plotted_point[j]:=true;
                          number_checked:=number_checked+1;
                        end;
                    end;
              end  {BBBB}  {of 'if' part of "graph_to_plot"='2'.....}
           else  {graph_to_plot='1' or '3'}
              begin  {BBBB}
                size:=ll[i];
{**********take out later**********}
                if TRACE5=true then writeln(userout2,
                    'Entering gtp=1or3 & size=',size:2);
{**********take out later**********}
                for j:=i to numb_sets_17_params do
                    begin
                      if size=ll[j] then
                        begin
                          plotted_point[j]:=true;
                          number_checked:=number_checked+1;
                        end;
                    end;
              end;  {BBBB}  {of 'else' part of "graph_to_plot"='2'.....}
{**********take out later**********}
           if TRACE5=true then writeln(userout2,
                'xmin=',xmin:3:3,' xmax=',xmax:3:3,
                ' ymin=',ymin:3:3,' ymax=',ymax:3:3);
{**********take out later**********}
           if ( ((graph_to_plot='1') or (graph_to_plot='3')) and
                (ll[i]+processpar[2,betazeroparam]>0.0) ) or
              ( ((graph_to_plot='2') or (graph_to_plot='4')) and
                (ww[i]+processpar[3,betazeroparam]>0.0) ) then
```

```
                begin  {CCCC}
                  xval:=0;
                  if param_to_graph<>betazeroparam then
                       begin
                            if (graph_to_plot='2') or (graph_to_plot='4') then
                                 yval:=processpar[1,param_to_graph] +
                                       processpar[3,param_to_graph] /
                                       (processpar[3,betazeroparam] + size)
                            else  {graph_to_plot='1' or '3'}
                                 yval:=processpar[1,param_to_graph] +
                                       processpar[2,param_to_graph] /
                                       (processpar[2,betazeroparam] + size);
                       end
                  else  {param_to_graph=betazeroparam}
                       yval:=0.0;  {beta0 or 1/beta0 minimum always = 0}
                  xnew:=-0.8+1.6*((xval-xmin)/(xmax-xmin));
                  ynew:=-0.8+1.6*((yval-ymin)/(ymax-ymin));
                  if ((xnew>=-0.8-epsilon) and (xnew<=0.8+epsilon) and
                       (ynew>=-0.8-epsilon) and (ynew<=0.8+epsilon)) then
                       begin  {point is on the screen}
                            move(xnew,ynew);
                            outplane:=false;
                       end
                  else       {point is NOT on the screen}
                       outplane:=true;
                  xold:=xnew;
                  yold:=ynew;
{*****************take out later*********************************}
                  if TRACE5=true then
                       begin
                            writeln(userout2,'xval=',xval:4:4,' yval=',
                                 yval:4:4,' xold=',xold:3:3,' yold=',yold:3:3);
                            writeln(userout2,'outplane=',outplane);
                       end;
{*****************take out later*********************************}
                  xval:=xmax_until_newgraph_is_chosen;
                  if param_to_graph<>betazeroparam then
                       begin
                            if (graph_to_plot='2') or (graph_to_plot='4') then
                                 yval:=processpar[1,param_to_graph] +
                                       processpar[3,param_to_graph] /
                                       (processpar[3,betazeroparam] + size) +
                                       processpar[2,param_to_graph] * xval
                            else  {graph_to_plot='1' or '3'}
                                 yval:=processpar[1,param_to_graph] +
                                       processpar[2,param_to_graph] /
                                       (processpar[2,betazeroparam] + size) +
                                       processpar[3,param_to_graph] * xval;
                       end
                  else  {param_to_graph=betazeroparam & plot beta0 vs. 1/L}
                       if (graph_to_plot='2') or (graph_to_plot='4') then
                            yval:=processpar[1,betazeroparam]*cox*
                                  (size + processpar[3,betazeroparam])*xval
                  else    {param_to_graph=betazeroparam & plot 1/beta0 vs. 1/W}
                       yval:=1/processpar[1,betazeroparam]/cox*xval*
                             (size + processpar[2,betazeroparam]);
                  xnew:=-0.8+1.6*((xval-xmin)/(xmax-xmin));
                  ynew:=-0.8+1.6*((yval-ymin)/(ymax-ymin));
{*****************take out later*********************************}
                  if TRACE5=true then writeln(userout2,
                       'xval=',xval:4:4,' yval=',yval:4:4,
                       ' xnew=',xnew:10:10,' ynew=',ynew:10:10);
```

```
        {**********take out later**********}
                        {check whether point will be on screen}
        {**********take out later**********}
                        if ((TRACE5=true) and (xnew>=-0.8)) then
                            writeln(userout2,'xnew>=-0.8');
                        if ((TRACE5=true) and (xnew<=0.8)) then
                            writeln(userout2,'xnew<=0.8');
                        if ((TRACE5=true) and (ynew>=-0.8)) then
                            writeln(userout2,'ynew>=-0.8');
                        if ((TRACE5=true) and (ynew<=0.8)) then
                            writeln(userout2,'ynew<=0.8');
        {**********take out later**********}
                        if ((xnew>=-0.8-epsilon) and (xnew<=0.8+epsilon) and
                            (ynew>=-0.8-epsilon) and (ynew<=0.8+epsilon))then
                          begin       {point is on the screen}
        {**********take out later**********}
                            if TRACE5=true then writeln(userout2,
                                'This point will be ON screen');
        {**********take out later**********}
                            if outplane=true then  {last point was off the screen}
                                begin
        {**********take out later**********}
                                    if TRACE5=true then writeln(userout2,
                                        'and Last point OFF screen');
        {**********take out later**********}
{jrp}                               if (xnew-xold<>0.0) then  {check for divide by zero}
{jrp}                                   slope:=(ynew-yold)/(xnew-xold)
{jrp}                               else
{jrp}                                   slope:=1E100:    {about = infinity}
                                    {check for entering from left}
                                    test:=slope*(-0.8-xold)+yold;
        {**********take out later**********}
                                    if TRACE5=true then writeln(userout2,
                                        'TEST A , test=',test:2:2);
        {**********take out later**********}
                                    if test_point_on_screen1(yold,ynew,test) then
                                        begin
        {**********take out later**********}
                                        if TRACE5=true then writeln(userout2,
                                            'test_point_on_screen is TRUE');
        {**********take out later**********}
                                            move(-0.8,test);
                                            draw_line_and_zstring(xnew,ynew,-0.75,test);
                                        end;
                                    {check for entering from the right}
                                    test:=slope*(0.8-xold)+yold;
        {**********take out later**********}
                                    if TRACE5=true then writeln(userout2,
                                        'TEST B , test=',test:2:2);
        {**********take out later**********}
                                    if test_point_on_screen1(yold,ynew,test) then
                                        begin
        {**********take out later**********}
                                        if TRACE5=true then writeln(userout2,
                                            'test_point_on_screen is TRUE');
        {**********take out later**********}
                                            move(0.8,test);
                                            draw_line_and_zstring(xnew,ynew,0.60,test);
                                        end;
                                    {check for entering from top}
                                    if slope<>0 then
                                        test:=(0.8-yold)/slope+xold
                                    else
```

```
                                        test:=-1.0;  {can not enter from top}
{**********take out later**********}
                                    if TRACE5=true then writeln(userout2,
                                        'TEST C , test=',test:2:2);
{**********take out later**********}
                                    if test_point_on_screen2(xold,xnew,test) then
                                        begin
{**********take out later**********}
                                    if TRACE5=true then writeln(userout2,
                                        'test_point_on_screen is TRUE');
{**********take out later**********}
                                            move(test,0.8);
                                            draw_line_and_zstring(xnew,ynew,test-0.2,0.78);
                                        end;
                                    {check for entering from bottom}
                                    if slope<>0 then
                                        test:=(-0.8-yold)/slope+xold
                                    else
                                        test:=-1.0;    {can not enter from bottom}
{**********take out later**********}
                                    if TRACE5=true then writeln(userout2,
                                        'TEST D , test=',test:2:2);
{**********take out later**********}
                                    if test_point_on_screen2(xold,xnew,test) then
                                        begin
{**********take out later**********}
                                    if TRACE5=true then writeln(userout2,
                                        'test_point_on_screen is TRUE');
{**********take out later**********}
                                            move(test,-0.8);
                                            draw_line_and_zstring(xnew,ynew,test-0.2,-0.78);
                                        end;
                                    {point has been plotted}
                                end
                            else    {last point was on screen so plot this one}
                                begin
{**********take out later**********}
                                    if TRACE5=true then writeln(userout2,
                                        'and Last point ON screen');
{**********take out later**********}
                                    draw_line_and_zstring(xnew,ynew,xnew-0.20,ynew);
                                end;  {of else part}
                        end {end of handling for values on the screen}
                    else {start handling points off screen}
                        begin
{**********take out later**********}
                            if TRACE5=true then writeln(userout2,
                                'This point WILL NOT be on screen');
{**********take out later**********}
                            if outplane=false then {last point was on the screen}
                                begin
{**********take out later**********}
                                    if TRACE5=true then writeln(userout2,
                                        'and Last point was ON screen');
{**********take out later**********}
{jrp}                               if (xnew-xold<>0.0) then   {check for divide by zero}
{jrp}                                   slope:=(ynew-yold)/(xnew-xold)
{jrp}                               else
{jrp}                                   slope:=1E100;    {about = infinity}
                                    {check for exiting from left}
                                    test:=slope*(0.8-xold)+yold;
{**********take out later**********}
                                    if TRACE5=true then writeln(userout2,
                                        'TEST E , test=',test:2:2);
```

```
{**********take out later**********}
                    if test_point_on_screen1(yold,ynew,test) then
                      begin
{**********take out later**********}
                    if TRACE5=true then writeln(userout2,
                      'test_point_on_screen is TRUE');
{**********take out later**********}
                        draw_line_and_zstring(0.8,test,0.60,test);
                      end;
                    {check for exiting from right}
                    test:=slope*(-0.8-xold)+yold;
{**********take out later**********}
                    if TRACE5=true then writeln(userout2,
                      'TEST F , test=',test:2:2);
{**********take out later**********}
                    if test_point_on_screen1(yold,ynew,test) then
                      begin
{**********take out later**********}
                    if TRACE5=true then writeln(userout2,
                      'test_point_on_screen is TRUE');
{**********take out later**********}
                        draw_line_and_zstring(-0.8,test,-0.75,test);
                      end;
                    {check for exiting from top}
                    if slope<>0 then
                      test:=(0.8-yold)/slope+xold
                    else
                      test:=-1.0;    {can not exit from top}
{**********take out later**********}
                    if TRACE5=true then writeln(userout2,
                      'TEST G , test=',test:2:2);
{**********take out later**********}
                    if test_point_on_screen2(xold,xnew,test) then
                      begin
                        draw_line_and_zstring(test,0.8,test-0.2,0.78);
{**********take out later**********}
                        if TRACE5=true then writeln(userout2,
                          'test_point_on_screen is TRUE');
{**********take out later**********}
                      end;
                    {check for exiting from bottom}
                    if slope<>0 then
                      test:=(-0.8-yold)/slope+xold
                    else
                      test:=-1.0;    {can not exit from bottom}
{**********take out later**********}
                    if TRACE5=true then writeln(userout2,
                      'TEST H , test=',test:2:2);
{**********take out later**********}
                    if test_point_on_screen2(xold,xnew,test) then
                      begin
                        draw_line_and_zstring(test,-0.8,test-0.2,-0.78);
{**********take out later**********}
                        if TRACE5=true then writeln(userout2,
                          'test_point_on_screen is TRUE');
{**********take out later**********}
                      end;
                    outplane:=true;
                  end   {end for points going off screen}
                else    {last point was not on screen & neither is this one}
                  begin
{**********take out later**********}
                    if TRACE5=true then writeln(userout2,
                      'and Last point was OFF screen');
```

```
        {**********take out later**********}
                                first_point_found:=false;
{jrp}                           if (xnew-xold<>0.0) then  {check for divide by zero}
{jrp}                               slope:=(ynew-yold)/(xnew-xold)
{jrp}                           else
{jrp}                               slope:=1E100;    {about = infinity}
                                {JOEJOEJOE}
                                {check for entering from left}
                                test:=slope*(-0.8-xold)+yold;
        {**********take out later**********}
                                if TRACE5=true then writeln(userout2,
                                    'TEST I , test=',test:2:2);
        {**********take out later**********}
                                if test_point_on_screen1(yold,ynew,test) then
                                    begin
        {**********take out later**********}
                                if TRACE5=true then writeln(userout2,
                                    'test_point_on_screen is TRUE');
        {**********take out later**********}
                                        if first_point_found=false then
                                            begin
                                                first_point_found:=true;
                                                move(-0.80,test);
                                            end
                                        else
                                            draw_line_and_zstring(-0.8,test,-0.75,test);
                                    end;
                                {check for entering from the right}
                                test:=slope*(0.8-xold)+yold;
        {**********take out later**********}
                                if TRACE5=true then writeln(userout2,
                                    'TEST J , test=',test:2:2);
        {**********take out later**********}
                                if test_point_on_screen1(yold,ynew,test) then
                                    begin
        {**********take out later**********}
                                        if TRACE5=true then writeln(userout2,
                                            'test_point_on_screen is TRUE');
        {**********take out later**********}
                                        if first_point_found=false then
                                            begin
                                                first_point_found:=true;
                                                move(0.80,test);
                                            end
                                        else
                                            draw_line_and_zstring(0.80,test,0.60,test);
                                    end;
                                {check for entering from top}
                                if slope<>0 then
                                    test:=(0.8-yold)/slope+xold
                                else
                                    test:=-1.0;  {can not enter from top}
        {**********take out later**********}
                                if TRACE5=true then writeln(userout2,
                                    'TEST K , test=',test:2:2);
        {**********take out later**********}
                                if test_point_on_screen2(xold,xnew,test) then
                                    begin
        {**********take out later**********}
                                        if TRACE5=true then writeln(userout2,
                                            'test_point_on_screen is TRUE');
        {**********take out later**********}
                                        if first_point_found=false then
                                            begin
                                                first_point_found:=true;
```

...draw_the_data2

```
                                                move(test,0.80);
                                            end
                                        else
                                            draw_line_and_zstring(test,0.80,test-0.2,0.78);
                                    end;
                                {check for entering from bottom}
                                if slope<>0 then
                                    test:=(-0.8-yold)/slope+xold
                                else
                                    test:=-1.0;    {can not enter from bottom}
        {******************take out later*********************************}
                                if TRACE5=true then writeln(userout2,
                                    'TEST L , test=',test:2:2);
        {******************take out later*********************************}
                                if test_point_on_screen2(xold,xnew,test) then
                                    begin
        {******************take out later*********************************}
                                        if TRACE5=true then writeln(userout2,
                                            'test_point_on_screen is TRUE');
        {******************take out later*********************************}
                                        if first_point_found=false then
                                            begin
                                                first_point_found:=true;
                                                move(test,-0.8);
                                            end
                                        else
                                            draw_line_and_zstring(test,-0.8,test-0.2,-0.78);
                                    end;
                                {line has been drawn}
                                {JOEJOEJOE}
                            end;
                        end; {end of handling current point off screen}
                    end;  {CCCC} {if 3rd_variable_value+other_delta_l_or_w
                                <0, we don't plot the line because this
                                would give false plots}
            end; {AAAA} {of 'if' statement...if plotted_point[i]=false...}
    until (((graph_to_plot='1') or (graph_to_plot='2')) and
                                        (number_checked=numb_sets_17_params)) or
          (((graph_to_plot='3') or (graph_to_plot='4')) and
                                        (found_third_variable_value=true));
    {******************take out later*********************************}
    if TRACE1=true then writeln(userout2,'leaving draw_the_data2');
    {******************take out later*********************************}
end;{end of procedure drawdata2}
```

zoom_graph_axis

```
procedure zoom_graph_axis(var xpos2,xpos1,ypos2,ypos1:real);
{this procedure allows the user to zoom in on a portion of the graph by
varying the axis limits}

var   button:integer;
      xrange,yrange:real;
      oldxmin,oldymin:real;
      temppos:real;


{initialize the graphics locator as the knob}
begin
      xrange:=xpos2-xpos1;
      yrange:=ypos2-ypos1;
      oldxmin:=xpos1;
      oldymin:=ypos1;
      locator_init(2,error_return);
      if error_return=0 then
```

_zoom_graph_axis

```
        begin
            await_locator(2,button,xpos1,ypos1);
            set_echo_pos(xpos1,ypos1);  {sets one diagonal of rectangle}
            await_locator(8,button,xpos2,ypos2);
            if xpos1>xpos2 then
                begin
                    temppos:=xpos1;
                    xpos1:=xpos2;
                    xpos2:=temppos;
                end;
            if ypos1>ypos2 then
                begin
                    temppos:=ypos1;
                    ypos1:=ypos2;
                    ypos2:=temppos;
                end;
            {this guarantees that the axis are not zoomed too small}
            if (abs(xpos1-xpos2)<0.1) or (abs(ypos1-ypos2)<0.1) then
                begin
                    xpos1:=-0.8;
                    xpos2:=0.8;
                    ypos1:=-0.8;
                    ypos2:=0.8;
                end;
            xpos1:=xrange/1.6*(xpos1+0.8)+oldxmin;
            xpos2:=xrange/1.6*(xpos2+0.8)+oldxmin;
            ypos1:=yrange/1.6*(ypos1+0.8)+oldymin;
            ypos2:=yrange/1.6*(ypos2+0.8)+oldymin;
            clear_display;
        end;
end;
```

initial_graph

```
procedure initial_graph(var maxminxy_error:integer);
{this procedure uses autoscaling to determine what the minimum and maximum
values of the graph are, and plots the data}

var init_ok:boolean;

begin
    {*****************************take out later*********************}
    if TRACE1=true then writeln(userout2,
        'NOW ENTERING initial_graph & 3rd_var_val= ',third_variable_value);
    {*****************************take out later*********************}
    grsetup(init_ok);
    set_aspect(210.0,160.0);
    set_display_lim(0.0,210.0,0.0,160.0,error_return);
    {this determines the maximum and minimum values of x and y}
    maxminxy(maximx,minimx,maximy,minimy,maxminxy_error);
    if maxminxy_error=0 then
        begin
            if have_not_zoomed_yet=true then
                begin
                    have_not_zoomed_yet:=false;
                    xmax_until_newgraph_is_chosen:=maximx;
                end;
            autoscale(maximx,minimx,maximy,minimy,xaxlab,yaxlab,headerlab,
                    deltallab,deltawlab,zaxlab1,zaxlab2,xaxisunit,yaxisunit);
            draw_the_data1(maximx,minimx,maximy,minimy);
            draw_the_data2(maximx,minimx,maximy,minimy);
        end;  {of if maxminxy_error=0 then.....}
    {*****************************take out later*********************}
    if TRACE1=true then writeln(userout2,'NOW LEAVING initial_graph');
```

..initial_graph

```
      {*****************************take out later*****************}
end;  {end of procedure initial_graph to plot graphics axis and data}
```

zoom

```
procedure zoom;
{this procedure handles zooming the graph axis}

begin
      zoom_graph_axis(maximx,minimx,maximy,minimy);
      clear_display;
      autoscale(maximx,minimx,maximy,minimy,xaxlab,yaxlab,headerlab,
                  deltallab,deltawlab,zaxlab1,zaxlab2,xaxisunit,yaxisunit);
      draw_the_data1(maximx,minimx,maximy,minimy);
      draw_the_data2(maximx,minimx,maximy,minimy);
end; {end of procedure to zoom in on axis}
```

make_xyz_axis_labels

```
procedure make_xyz_axis_labels;
{this procedure generates the labels}
{last change made OCT 31, 1984    by   JRP}

var   i,j:integer;


begin
      {this is where the labels are setup}
      writeln(#12);
      if (graph_to_plot='1') or (graph_to_plot='3') then
           begin
                xaxlab:='WIDTH in ';
                zaxlab1:='LENGTH';
           end
      else
           begin
                xaxlab:='LENGTH in ';
                zaxlab1:='WIDTH';
           end;
      zaxlab2:='in microns';
      case param_to_graph of
        1:yaxlab:='VFB in ';
        2:yaxlab:='2PHIF in ';
        3:yaxlab:='K1 in ';
        4:yaxlab:='K2 in ';
        5:yaxlab:='ETA ';
        6:if (graph_to_plot='1') or (graph_to_plot='3') then
                yaxlab:='1/BETA0 in '
          else
                yaxlab:='BETA0 in ';
        7:yaxlab:='U0 in ';
        8:yaxlab:='U1 in ';
        9:yaxlab:='X2MU0 in ';
        10:yaxlab:='X2ETA in ';
        11:yaxlab:='X3ETA in ';
        12:yaxlab:='X2U0 in ';
        13:yaxlab:='X2U1 in ';
        14:yaxlab:='MU0SAT in ';
        15:yaxlab:='X2MU0SAT in ';
        16:yaxlab:='X3MU0SAT in ';
        17:yaxlab:='X3U1 in ';
      end;  {of case to determin yaxis label}
      headerlab:='BSIM PARAMETER vs ';
      xaxisunit:='microns';
      case param_to_graph of
        1:yaxisunit:='volts';
```

...make_xyz_axis_labels

```
        2:yaxisunit:='volts';
        3:yaxisunit:=' /sqrt(volts)';
        4:yaxisunit:=' /volts';
        5:yaxisunit:=' /volts';
        6:if (graph_to_plot='1') or (graph_to_plot='3') then
              yaxisunit:='volts^2/amps'
          else
              yaxisunit:='amps/volts^2';
        7:yaxisunit:=' /volts';
{JRP}   8:yaxisunit:='um/volts';
        9:yaxisunit:='cm^2/volts^2-sec';
        10:yaxisunit:=' /volts^2';
        11:yaxisunit:=' /volts^2';
        12:yaxisunit:=' /volts^2';
{JRP}   13:yaxisunit:='um/volts^2';
        14:yaxisunit:='cm^2/volts-sec';
        15:yaxisunit:='cm^2/volts^2-sec';
        16:yaxisunit:='cm^2/volts^2-sec';
{JRP}   17:yaxisunit:='um/volts^2';
      end; {of case to determine yaxis unit}
   deltallab:='deltaL=';
   deltawlab:='deltaW=';
end;
```

draw_repeat_menu

```
procedure draw_repeat_menu;

begin
     write(#12);
     gotoxy(0,2);
     writeln('SELECT A NUMBER FOR A GIVEN ACTION CAPABILITY=  >');
     writeln('            [1] Zoom Using Knob and Keys');
     writeln('            [2] Redraw Full Graph');
     writeln('            [3] Select A New Graph');
     writeln('            [4] Exit BSIM PARAMETER vs L or W Menu');
end;
```

draw_3_menu_pages

```
procedure draw_3_menu_pages;
{this procedure draws 3 menu pages and gets the desired graph to plot info
 from the user.}

var     change:char;  {if = 'c', then re-do all inputs}
```

page1

```
procedure page1;

const   numbgarbage=9;  {number of junk lines in a single process file}

var     i,j:integer;    {loop counters}
        line_in_file:string[15];  {line in die_file}
        nmose_present,nmosz_present,nmosd_present,
        pmose_present,pmosz_present,pmosd_present:boolean;  {=true if this
                          devicename has a process file in the output file}
        xpos,ypos:integer;  {x and y die position read back from the output
                             file}
        xposline,yposline:string[8];  {same as above but includes *XPOS= or *YPOS=}
        devicenameline:linestring;  {= devicename as read from output file}
{jrp}   file_selection_error:boolean;  {=t if user selected file is not one
                                        of the ones which has a process file
                                        developed for it}

begin
```

```
nmose_present:=false;          {check the output file to see what type}
nmosz_present:=false;          {of devices have been SUCCESSFULLY tested}
nmosd_present:=false;          {at the current die location}
pmose_present:=false;
pmosz_present:=false;
pmosd_present:=false;
reset(userout,output_file);
while not(eof(userout)) do  {simply count the lines until at end of file}
    begin
        readln(userout,devicenameline);
{*********************************take out later**********************}
        if TRACE11=true then
            writeln('devicenameline=',devicenameline,'<---end');
{*********************************take out later**********************}
        for i:=1 to 4 do
            readln(userout,xposline);                {read *XPOS= line}
        strdelete(xposline,1,6);                     {delete *XPOS= from it}
        xposline:=strrtrim(xposline);
        xpos:=trunc(strtoreal(xposline));          {convert to an integer}
{*********************************take out later**********************}
        if TRACE11=true then
            writeln('xpos=',xpos,'<---end');
{*********************************take out later**********************}
        readln(userout,yposline);                    {read *YPOS= line}
        strdelete(yposline,1,6);                     {delete *YPOS= from it}
        yposline:=strrtrim(yposline);
        ypos:=trunc(strtoreal(yposline));          {convert to an integer}
{*********************************take out later**********************}
        if TRACE11=true then
            writeln('ypos=',ypos,'<---end');
{*********************************take out later**********************}
        if ((xpos=present_diex) and (ypos=present_diey)) then
            begin
                if (strpos('NMOSE',devicenameline)<>0) then
                    nmose_present:=true
                else if (strpos('NMOSZ',devicenameline)<>0) then
                    nmosz_present:=true
                else if (strpos('NMOSD',devicenameline)<>0) then
                    nmosd_present:=true
                else if (strpos('PMOSE',devicenameline)<>0) then
                    pmose_present:=true
                else if (strpos('PMOSZ',devicenameline)<>0) then
                    pmosz_present:=true
                else if (strpos('PMOSD',devicenameline)<>0) then
                    pmosd_present:=true;
            end;
        for i:=1 to numbbsim+numbgarbage-6 do
            readln(userout,devicenameline);
    end;
close(userout,'save');
write(#12);
gotoxy(0,0);
writeln('                          ***BSIM PARAMETER vs. W or L GRAPH***');
writeln;
write('This graphics mode allows one to compare extracted, size-');
writeln('DEPENDENT parameters ');
write('from the 17-parameter ELECTRICAL file, to size-INDEPENDENT values');
writeln(', approximated');
writeln('from the 54-parameter PROCESS file.');
writeln;
write('If you plot W on the x-axis, then L becomes the 3rd');
writeln(' variable, and vice versa.');
write('You may choose to plot only one third-variable value, or you');
writeln(' may plot all of');
write('them.  Choosing only one allows finer details to be analyzed.');
```

```
      writeln('   The x-axis');
      writeln('values are scaled linear with respect to 1/EFFECTIVE SIZE.');
      writeln;
      writeln('You will choose:  1) the type of device to plot');
      writeln('                  2) the BSIM parameter to plot on the y-axis');
      writeln('                  3) whether W or L will be plotted on the x-axis');
      writeln('                  4) and whether all sizes or one size device will');
      writeln('                     be plotted for the third parameter');
      writeln;
      write('SELECT THE DEVICE TYPE YOU WANT TO PLOT=   >');
      i:=0;
      if (nmose_present=true) then
           begin
                gotoxy(10,18+i);
                i:=i+1;
                writeln('[1] NMOS enhancement');
           end;
      if (nmosd_present=true) then
           begin
                gotoxy(10,18+i);
                i:=i+1;
                writeln('[2] NMOS depletion');
           end;
      if (nmosz_present=true) then
           begin
                gotoxy(10,18+i);
                i:=i+1;
                writeln('[3] NMOS zero-threshold');
           end;
      if (pmose_present=true) then
           begin
                gotoxy(10,18+i);
                i:=i+1;
                writeln('[4] PMOS enhancement');
           end;
      if (pmosd_present=true) then
           begin
                gotoxy(10,18+i);
                i:=i+1;
                writeln('[5] PMOS depletion');
           end;
      if (pmosz_present=true) then
           begin
                gotoxy(10,18+i);
                i:=i+1;
                writeln('[6] PMOS zero-threshold');
           end;
{jrp}file_selection_error:=true;
{jrp}while (file_selection_error=true) do  {user might have input wrong value}
{jrp}      begin
                selection_input(43,17,'1','6',file_to_graph);
{jrp}           case file_to_graph of
{jrp}             '1':file_selection_error:=not(nmose_present);
{jrp}             '2':file_selection_error:=not(nmosd_present);
{jrp}             '3':file_selection_error:=not(nmosz_present);
{jrp}             '4':file_selection_error:=not(pmose_present);
{jrp}             '5':file_selection_error:=not(pmosd_present);
{jrp}             '6':file_selection_error:=not(pmosz_present);
{jrp}           end;  {of case}
{jrp}           if (file_selection_error=true) then
{jrp}                begin
{jrp}                     gotoxy(46,17);
```

...page1

```
{jrp}                    write('ERROR IN YOUR SELECTION');
{jrp}                    gotoxy(46,18);
{jrp}                    write('PLEASE SELECT AGAIN');
{jrp}                    gotoxy(42,17);
{jrp}                    write('>');
{jrp}                end;
{jrp}      end;  {of while file_selection_error=true...}
{jrp}gotoxy(46,17);
{jrp}write('                                        ');
{jrp}gotoxy(46,18);
{jrp}write('                                        ');
     gotoxy(47,22);
     write('PLEASE WAIT...');
     case file_to_graph of
          '1':file_to_graph_name:='NEDFILE.TEXT';
          '2':file_to_graph_name:='NZDFILE.TEXT';
          '3':file_to_graph_name:='NDDFILE.TEXT';
          '4':file_to_graph_name:='PEDFILE.TEXT';
          '5':file_to_graph_name:='PZDFILE.TEXT';
          '6':file_to_graph_name:='PDDFILE.TEXT';
        end;  {of case selection}
     reset(userout,file_to_graph_name);
     i:=0;  {counter to count # groups of 17 parameters}
     repeat
          i:=i+1;
          readln(userout,line_in_file);  {width}
          ww[i]:=strtoreal(line_in_file);
          readln(userout,line_in_file);  {length}
          ll[i]:=strtoreal(line_in_file);
          for j:=1 to numbbsim do
               begin
                    readln(userout,line_in_file);  {1 of 17 bsim parameters}
                    if TRACE3=true then
                         writeln('line_in_file =  ',line_in_file);
                    param[i,j]:=strtoreal(line_in_file);
                    if TRACE3=true then
                         writeln('param[i,j] =  ',param[i,j]);
               end;
          readln(userout,line_in_file);  { '**********' line}
     until eof(userout);
     numb_sets_17_params:=i;
end;  {of procedure page1}
```

```
procedure page2;
```

page2

```
var      fraction_part:real;
         numbrows,numbcolumns:integer;  {# rows or cols for this screen full}
         row,column:integer;  {present row or column we are on}
         total_numb_of_rows:integer;  {# of rows left to list}
         numb_rows_left:integer;  {# of rows left to be plotted}
         numb_times_repeated:integer;  {# times thru repeat loop.  needed so
                                        correct w and l values are pulled from array}
         selection:char;  {= CR when user decides to continue}

begin
  {*********************************take out later*********************}
    if TRACE1=true then writeln(userout2,'entering PAGE1');
  {*********************************take out later*********************}
    {the W/L ratios will be listed in groups of 10 across the screen.}
    total_numb_of_rows:=trunc(numb_sets_17_params / 10) + 1;  {# of lines of 10 params}
```

```
                                                    {+1 for fractional part}
fraction_part:=(numb_sets_17_params / 10) - trunc(numb_sets_17_params / 10)/1;
{pascal does not have a "fraction" function which is similar to "trunc".
 hence, the line above has a needless divide by 1 in it.  you see, pascal
 does allow 2 integers to be divided with a real result, but does not
 allow 2 integers to be subtracted with a real result!!! nice, huh!?!?}
if fraction_part=0.0 then  { / by 10 came out even, so subtract out added 1}
        total_numb_of_rows:=total_numb_of_rows - 1;
numbrows:=4;
numbcolumns:=10;
numb_rows_left:=total_numb_of_rows;  {# rows left to list}
numb_times_repeated:=0; {each repeat cycle, w and l array line pointer
                              must be incremented}
repeat  {until numb_rows_left=0}
    write(#12);
    gotoxy(0,0);
    writeln('W/L ratios of devices successfully tested are listed here:');
{**********************take out later*****************************}
    {gotoxy(0,18);}
    {writeln('numbrows = ',numbrows);}
{**********************take out later*****************************}
    if (numb_rows_left<4) then
        numbrows:=numb_rows_left;
    for row:=1 to numbrows do
        begin
            gotoxy(0,3*row - 2);
            write('W');
            gotoxy(0,3*row - 1);
            write('L');
            if (numb_rows_left<=4) and (row=numbrows) then
                                        {if last row, # columns might not = 10}
                numbcolumns:=numb_sets_17_params - (10*(total_numb_of_rows-1));
            for column:=1 to numbcolumns do
                begin
                    gotoxy(7*column,3*row - 2);
                    write(ww[40*numb_times_repeated+(10*(row - 1)) + column]:7:1);
                    gotoxy(7*column,3*row - 1);
                    write(ll[40*numb_times_repeated+(10*(row - 1)) + column]:7:1);
                end;
        end;
        numb_times_repeated:=numb_times_repeated + 1;{# times thru repeat loop}
        numb_rows_left:=numb_rows_left - numbrows;
        if (numb_rows_left<>0) then
            begin
                gotoxy(0,15);
                write('There are more W/L values to view.  Press ');
                writeln('"ENTER" to view them.  >');
                selection_input(66,15,chr(ord(32)),chr(ord(32)),selection);
            end;
until (numb_rows_left=0);
gotoxy(0,16);
writeln('SELECT DESIRED GRAPH=? >');
writeln('          [1] BSIM PARAMETER vs. W --- for all values of L');
writeln('          [2] BSIM PARAMETER vs. L --- for all values of W');
writeln('          [3] BSIM PARAMETER vs. W --- for single value of L.  L=? >');
writeln('          [4] BSIM PARAMETER vs. L --- for single value of W.  W=? >');
selection_input(24,16,'1','4',graph_to_plot);
if graph_to_plot='3' then
    begin
        gotoxy(68,19);
        readln(third_variable_value);
        gotoxy(65,19);
        write('          ');
        gotoxy(65,19);
        write(third_variable_value:5:2);
```

...page2

```
    {*******************************take out later*******************}
                if TRACE3=true then writeln(userout2,
                    'graph_to_plot=3 & 3rd_var_value=',third_variable_value);
    {*******************************take out later*******************}
            end;
        if graph_to_plot='4' then
            begin
                gotoxy(68,20);
                readln(third_variable_value);
                gotoxy(65,20);
                write('          ');
                gotoxy(65,20);
                write(third_variable_value:5:2);
    {*******************************take out later*******************}
                if TRACE3=true then writeln(userout2,
                    'graph_to_plot=4 & 3rd_var_value=',third_variable_value);
    {*******************************take out later*******************}
            end;
    {*******************************take out later*******************}
        if TRACE1=true then writeln(userout2,'leaving PAGE1');
    {*******************************take out later*******************}
end;  {of procedure page2}
```

page3

```
procedure page3;

begin
      write(#12);
      gotoxy(0,0);
      writeln('SELECT THE PARAMETER TO BE GRAPHED=   >');
      writeln('                [1] VFB');
      writeln('                [2] 2PHIF');
      writeln('                [3] K1');
      writeln('                [4] K2');
      writeln('                [5] ETA');
      writeln('                [6] BETA0');
      writeln('                [7] U0');
      writeln('                [8] U1');
      writeln('                [9] X2MU0');
      writeln('               [10] X2ETA');
      writeln('               [11] X3ETA');
      writeln('               [12] X2U0');
      writeln('               [13] X2U1');
      writeln('               [14] MU0SAT');
      writeln('               [15] X2MU0SAT');
      writeln('               [16] X3MU0SAT');
      writeln('               [17] X3U1');
      gotoxy(38,0);
      readln(param_to_graph);
      while (param_to_graph<0) or (param_to_graph>17) do
            begin
                gotoxy(38,0);
                write('          ');
                gotoxy(38,0);
                read(param_to_graph);
            end;
      gotoxy(35,0);
      write('              ');
      gotoxy(35,0);
{jrp}if param_to_graph<10 then
{jrp}     write(param_to_graph:1,' ')
{jrp}else
            write(param_to_graph:2,' ');
      gotoxy(0,19);
      write('Press a "c" to make any changes to your choices, or');
```

```
     writeln(' press "ENTER" to begin   >');
     gotoxy(77,19);
end;  {of proecdure page3}


begin   {procedure draw_3_menu_pages}
     change:='c';
     while (change='c') or (change='C') do
          begin
               page1;
               page2;
               page3;
               read(change);
          end; {while loop}
     clear_display;
end:  {procedure draw_3_menu_pages}


{*************************MAIN PO_GRAPHICS PROGRAM*********************}

begin    {main procedure po_graphics begins here}
       TRACE1:=false;
       TRACE2:=false;
       TRACE3:=false;
       TRACE4:=false;
       TRACE5:=false;
       TRACE6:=false;
       TRACE7:=false;
       TRACE8:=false;
       TRACE9:=false;
       TRACE10:=false;
       TRACE11:=false;
       TRACE12:=false;
       TRACE13:=false;
       TRACE14:=false;
       TRACE15:=false;
       TRACE16:=false;
       {TRACE1:=true;
       TRACE2:=true;
       TRACE3:=true;
       TRACE4:=true;
       TRACE5:=true;
       TRACE6:=true;
       TRACE7:=true;
       TRACE8:=true;
       TRACE9:=true;
       TRACE10:=true;
       TRACE11:=true;
       TRACE12:=true;
       TRACE13:=true;
       TRACE14:=true;
       TRACE15:=true;
       TRACE16:=true;}
       grsetup(init_ok);
       newgraph:=true;
       if init_ok then
            begin
                 repeat
                      writeln(#12); {clears the display}
                      draw_3_menu_pages;
                      prepare_for_po_graphics(po_graphics_error);
```

```
{**************************take out later**************************}
                    if TRACE35=true then
                        begin
                            reset(userout,output_file);
                            write(userout2,
                                'Just after prepare_for_po_graphics and here');
                            writeln(userout2,' is output file:');
                            while not(eof(userout)) do
                                begin
                                    readln(userout,line_in_output_file);
                                    writeln(userout2,line_in_output_file);
                                end;
                            writeln(userout2,'That was it');
                            close(userout,'save');
                        end;
{**************************take out later**************************}
                    if (po_graphics_error=0) then
                        begin
                            if TRACE1=true then writeln(userout2,
                                'about to enter make_xyz_axis_labels');
                            make_xyz_axis_labels;
                            have_not_zoomed_yet:=true;
                            initial_graph(effective_size_error);
                            newgraph:=true;
                            if effective_size_error=0 then
                                begin
{jrp}                               write(#12);
                                    quit:=false;
                                    newgraph:=false;
                                    repeat
                                        draw_repeat_menu;
                                        selection_input(49,2,'1','4',repeat_selection);
                                            case repeat_selection of
                                                '1':begin
                                                        gotoxy(52,3);
                                                        write(' ');
                                                        gotoxy(52,3);
                                                        write('SWITCH TO GRAPHICS...');
                                                        zoom;
                                                    end;
                                                '2':initial_graph(effective_size_error);
                                                '3':begin
                                                        newgraph:=true;
                                                        clear_display;
                                                    end;
                                                '4':begin
                                                        quit:=true;
                                                        clear_display;
                                                    end;
                                            end;
                                    until (quit=true) or (newgraph=true);
                                end
                            else  {effective_size_error<>0}
                                begin
                                    gotoxy(0,21);
                                    write('                                                                    ');
                                    write('                                                                    ');
                                    gotoxy(0,21);
                                    write('                                                                    ');
                                    write('                                                                    ');
                                    gotoxy(0,21);
                                    write('                                                                    ');
                                    write('                                                                    ');
                                    gotoxy(0,21);
                                    if (graph_to_plot='1') or (graph_to_plot='3') then
```

```
                                        write('DELTA W = ',processpar[3,betazeroparam]:3:2)
                                    else  {graph_to_plot='2' or '4'}
                                        write('DELTA L = ',processpar[2,betazeroparam]:3:2);
                                    write('.  THIS YIELDS A NEGATIVE EFFECTIVE');
                                    writeln(' SIZE FOR ALL VALUES OF');
                                    if (graph_to_plot='1') or (graph_to_plot='3') then
                                        write('WIDTH. ')
                                    else  {graph_to_plot='2' or '4'}
                                        write('LENGTH.');
{jrp}                               writeln('Do you want to select another graph?  (Y/N)  >');
{jrp}                               yes_no_selection_input(54,22,newgraph);

                                end;
                        end
{jrp}               else
{jrp}                   begin
                            gotoxy(0,21);
                            if (po_graphics_error=1) then
                                begin
                                    writeln;
                                    write('GRAPHICS MENU CAN NOT BE ACCESSED ');
                                    writeln('BECAUSE THE OUTPUT FILE IS EMPTY.');
                                end
                            else if (po_graphics_error=2) then
                                begin
                                    write('GRAPHICS MENU CAN NOT BE ACCESSED ');
                                    writeln('BECAUSE A PROCESS FILE FOR THE TYPE OF');
                                    write('DEVICE YOU REQUESTED WAS NOT CREATED AT');
                                    writeln(' THE CURRENT DIE POSITION.');
                                end
                            else if (po_graphics_error=3) then
                                begin
                                    write('DELTA L = ',processpar[2,betazeroparam]:3:3,
                                        '.  YOU HAVE SELECTED A DEVICE WITH AN ');
                                    writeln('EFFECTIVE CHANNEL');
                                    writeln('LENGTH LESS THAN ZERO.');
                                end
                            else if (po_graphics_error=4) then
                                begin
                                    write('DELTA W = ',processpar[3,betazeroparam]:3:3,
                                        '.  YOU HAVE SELECTED A DEVICE WITH AN ');
                                    writeln('EFFECTIVE CHANNEL');
                                    writeln('WIDTH LESS THAN ZERO.');
                                end;
{jrp}                       write('Do you want to select another graph?  (Y/N)  >');
{jrp}                       yes_no_selection_input(47,23,newgraph);
{jrp}                   end;  {of else part of "if po_graphics_error=0....}
            until newgraph=false;
        end;
    clear_display;
    graphics_term; {terminate the graphics procedures}
end;

{*****************************************************************}
{******************BSIM FILE CREATION AND STORAGE*****************}
{*****************************************************************}


procedure process_file_development(new_die,end_of_die,new_wafer:boolean);
{last change made: SUNDAY OCT 14, 1984   By: JRP}

type linestring=string[80];
```

process_file_development

...process_file_development

```
{**********DIE FILE CREATION**********}

procedure die_file_creation;                                    die_file_creation
{this procedure resets all of the temporary files and their pointers}

begin
      numbne:=0;
      numbpe:=0;                                  {zero out}
      numbnd:=0;          {number of good devices of particular device type}
      numbpd:=0;
      numbnz:=0;
      numbpz:=0;
      rewrite(nchanenh,'NEDFILE.TEXT');      {clear out all 6 of these files}
      close(nchanenh,'SAVE');
      rewrite(pchanenh,'PEDFILE.TEXT');
      close(pchanenh,'SAVE');
      rewrite(nchandep,'NDDFILE.TEXT');
      close(nchandep,'SAVE');
      rewrite(pchandep,'PDDFILE.TEXT');
      close(pchandep,'SAVE');
      rewrite(nchanzer,'NZDFILE.TEXT');
      close(nchanzer,'SAVE');
      rewrite(pchanzer,'PZDFILE.TEXT');
      close(pchanzer,'SAVE');
      rewrite(temporary,'TEMP.TEXT');
      close(temporary,'SAVE');
end;  {end of procedure to create die files and zero counters}


{**********17 PARAMETER STORAGE**********}

procedure store_parameters_in_die_files(device_type:integer);  store_parameters_in_die_files

{this procedure handles storing the extracted bsim parameters in a file
based on the device type that they represent}

{there are 6 basic types of device types recognized}
{1-Nchannel Enhancement}   {NEDFILE}
{2-Pchannel Enhancement}   {PEDFILE}
{3-Nchannel Depletion}     {NDDFILE}
{4-Pchannel Depletion}     {PDDFILE}
{5-Nchannel Zero-Threshold}  {NZDFILE}
{6-Pchannel Zero-Threshold}  {PZDFILE}

{if single device operation is used, the data is stored as follows.}
{the threshold voltage and the device type (n or p) from the
measurement routines are used to determine the file to store the data in.}
{In fully automatic operation, device types are specified by the user.}

var   diefile:string[20];
      linevar:string[80];
      devnumber:integer;

begin
      rewrite(temporary,'TEMP.TEXT');
      if (device_type=1) and (enhancement_depletion=1) then
            begin
                  devnumber:=1;                                {devicetype}
                  diefile:='NEDFILE.TEXT';
                  reset(nchanenh,diefile);
                  while not(eof(nchanenh)) do    {read file into temporary file}
                        begin
                              readln(nchanenh,linevar);
                              writeln(temporary,linevar);
                        end;
```

```
        end
    else if (device_type=-1) and (enhancement_depletion=1) then
        begin
            devnumber:=2;
            diefile:='PEDFILE.TEXT';
            reset(pchanenh,diefile);
            while not(eof(pchanenh)) do
                begin
                    readln(pchanenh,linevar);
                    writeln(temporary,linevar);
                end;
        end
    else if (device_type=1) and (enhancement_depletion=-1) then
        begin
            devnumber:=3;
            diefile:='NDDFILE.TEXT';
            reset(nchandep,diefile);
            while not(eof(nchandep)) do
                begin
                    readln(nchandep,linevar);
                    writeln(temporary,linevar);
                end;
        end
    else if (device_type=-1) and (enhancement_depletion=-1) then
        begin
            devnumber:=4;
            diefile:='PDDFILE.TEXT';
            reset(pchandep,diefile);
            while not(eof(pchandep)) do
                begin
                    readln(pchandep,linevar);
                    writeln(temporary,linevar);
                end;
        end
    else if (device_type=1) and (enhancement_depletion=0) then
        begin
            devnumber:=5;
            diefile:='NZDFILE.TEXT';
            reset(nchanzer,diefile);
            while not(eof(nchanzer)) do
                begin
                    readln(nchanzer,linevar);
                    writeln(temporary,linevar);
                end;
        end
    else if (device_type=-1) and (enhancement_depletion=0) then
        begin
            devnumber:=6;
            diefile:='PZDFILE.TEXT';
            reset(pchanzer,diefile);
            while not(eof(pchanzer)) do
                begin
                    readln(pchanzer,linevar);
                    writeln(temporary,linevar);
                end;
        end;
    writeln(temporary,width);
    writeln(temporary,length);
    writeln(temporary,c1_vfb);
    writeln(temporary,c2_phif2);
    writeln(temporary,c3_k1);
    writeln(temporary,c4_k2);
    writeln(temporary,c5_eta);
    writeln(temporary,c6_beta0);
    writeln(temporary,c7_u0);
```

```
writeln(temporary,c8_u1);
writeln(temporary,c9_x2beta0);
writeln(temporary,c10_x2eta);
writeln(temporary,c11_x3eta);
writeln(temporary,c12_x2u0);
writeln(temporary,c13_x2u1);
writeln(temporary,c14_beta0sat);
writeln(temporary,c15_x2beta0sat);
writeln(temporary,c16_x3beta0sat);
writeln(temporary,c17_x3u1);
writeln(temporary,'**************');
close(temporary,'save');
{this is where the diefie is written to}
reset(temporary,'TEMP.TEXT');
case devnumber of
      1:begin
            rewrite(nchanenh,diefile);
            while not(eof(temporary)) do
                  begin
                        readln(temporary,linevar);
                        writeln(nchanenh,linevar);
                  end;
            close(nchanenh,'save');
            numbne:=numbne+1;
         end;
      2:begin
            rewrite(pchanenh,diefile);
            while not(eof(temporary)) do
                  begin
                        readln(temporary,linevar);
                        writeln(pchanenh,linevar);
                  end;
            close(pchanenh,'save');
            numbpe:=numbpe+1;
         end;
      3:begin
            rewrite(nchandep,diefile);
            while not(eof(temporary)) do
                  begin
                        readln(temporary,linevar);
                        writeln(nchandep,linevar);
                  end;
            close(nchandep,'save');
            numbnd:=numbnd+1;
         end;
      4:begin
            rewrite(pchandep,diefile);
            while not(eof(temporary)) do
                  begin
                        readln(temporary,linevar);
                        writeln(pchandep,linevar);
                  end;
            close(pchandep,'save');
            numbpd:=numbpd+1;
         end;
      5:begin
            rewrite(nchanzer,diefile);
            while not(eof(temporary)) do
                  begin
                        readln(temporary,linevar);
                        writeln(nchanzer,linevar);
                  end;
            close(nchanzer,'save');
            numbnz:=numbnz+1;
         end;
```

```
            6:begin
                  rewrite(pchanzer,diefile);
                  while not(eof(temporary)) do
                        begin
                              readln(temporary,linevar);
                              writeln(pchanzer,linevar);
                        end;
                  close(pchanzer,'save');
                  numbpz:=numbpz+1;
               end;
      end;  {end of case statement}
end;  {end of procedure store_parameters_in_die_files}


{**********PROCESSS FILE CREATION AND STORAGE**********}

procedure make_process_files;                                   make_process_files
{this is the main procedure that handles making a process file}

var   ignores,paramvals:string[80];
      paramvalue:real;
      j:integer;
{jrp}i:integer;  {loop counter}
{jrp}number_good_devices:integer;  {# of good devices which have had parameters
                                    extracted for them.  this number will
                                    generally be different for each type of
                                    device}


procedure load_up_process_parameters(filenumb,numbval:integer;load_up_process_parameters
                                     var number_good_devices_error:integer);
{this procedure handles analyzing any of the six data files (NEDFILE,
PEDFILE, NDDFILE, PDDFILE, NZDFILE, and PZDFILE) to determine as many
process_parameters as permitted. the process parameters are loaded in the
array processpar[1..3,1..17] which is a global array}

type  processarray=array[1..20] of real;

var   paramcount,skip:integer;
      widths,lengths,paramvalues,betazeros,ignores:string[80];
      width,length,paramvalue:processarray;
      numbdevice:integer;
      mindL,uncox,dWuncox:real;
      {************************************************************}
      i:integer;  {loop counter----------take out later}


procedure l_and_w_dependencies(var width,length,paramvalue:processarray;l_and_w_dependencies
                               constant,linear1,linear2:real; entries:integer;
                               betazero:boolean; var goodcount:integer);
{this procedure takes all of the values of a parameter across various values
of l and w and determines the variations.  Checks are made to be sure that
the same values are not entered, and that 0 values are ignored}
{the betazero flag tells whether this is for the beta extraction (true)}
{last change made: OCT 9, 1984  by: JRP}

var goodwidth,goodlength,goodparamvalue:array[1..20] of real;
    i,j:integer;
    duplicate,different1,differentw:boolean;

begin
      {**********take out later**********}
      if TRACE30=true then
         begin
            if (paramcount=8) or (paramcount=13) or (paramcount=17) then
```

```
                begin
                    writeln(userout2,'paramcount=',paramcount:2);
                    for i:=1 to entries do
                        write(userout2,' paramvalue[',i:1,']=',paramvalue[i]);
                end;
        end;
    {**********take out later**********}
    goodcount:=0;
    {check for duplicate w and l values, and eliminate the first one tested}
    for i:=1 to entries do  {this is the latest pair of l and w}
        begin
            duplicate:=false;
            for j:=i to entries do  {this scans over remaining  values}
                if ((i<>j) and (width[i]=width[j]) and
                  (length[i]=length[j])) then
                    duplicate:=true;
            {eliminate next if-then statement to see if this is causing
             some error}
            {if paramvalue[i]=0.0 then
                duplicate:=true;}
            if duplicate=false then
                begin
                    goodcount:=goodcount+1;
                    goodwidth[goodcount]:=width[i];
                    goodlength[goodcount]:=length[i];
                    goodparamvalue[goodcount]:=paramvalue[i];
                end;
        end;  {end of check for zeros and duplications}
    {**********take out later**********}
    if TRACE30=true then
        begin
            if (paramcount=8) or (paramcount=13) or (paramcount=17) then
                begin
                    writeln(userout2,'paramcount=',paramcount:2);
                    for i:=1 to goodcount do
                        write(userout2,' goodparamvalue[',i:1,']=',
                        goodparamvalue[i]);
                end;
        end;
    {**********take out later**********}
    linear1:=0.0;
    linear2:=0.0;
    constant:=0.0;
    if (goodcount<>0) and (goodcount<>1) and (goodcount<>2) then
        begin
            {check for two different l, and/or two different w  values}
            differentl:=false;
            differentw:=false;
            for i:=1 to goodcount do
                for j:=1 to goodcount do
                    begin
                        if goodlength[i]<>goodlength[j] then
                            differentl:=true;
                        if goodwidth[i]<>goodwidth[j] then
                            differentw:=true;
                    end;
            {here is where the least-square procedures are called}
            if betazero=true then  {this is a check for betazero}
                begin
{**********take out later**********}
                    if TRACE32=true then
                    begin
                        writeln(userout2,'w/in l_and_w, BETA0 param');
                        for i:=1 to goodcount do writeln(userout2,
                          'gdparval[',i:1,']=',goodparamvalue[i],' gdwidth=',
```

```
                    goodwidth[i]:2:1,' gdlength=',goodlength[i]:2:1);
                end;
{*****************************take out later*********************}
                if differentl=false then
                    begin
                        constant:=goodparamvalue[1];
                        if TRACE32=true then writeln(userout2,
                            'diffl=false and constant=',constant);
                    end
                else if (differentl=true) and (differentw=false) then
                    begin
                     for i:=1 to goodcount do
                            leastsq2(i,1,goodcount,goodwidth[i] /
                            goodparamvalue[i],goodlength[i],constant,
                            linear1) ;
                     if TRACE32=true then writeln(userout2,
                          'diffl=true & diffw=false & constant=',constant,
                          ' linear1=',linear1);
                    end
                else
                    begin
                     for i:=1 to goodcount do
                            leastsq3(i,1,goodcount,goodwidth[i] /
                            goodparamvalue[i],1 /goodparamvalue[i],
                            goodlength[i],constant,linear1,linear2);
                     if TRACE32=true then writeln(userout2,
                          'diffl=true & diffw=true & constant=',constant,
                          ' linear1=',linear1,' linear2=',linear2);
                    end
            end
        else  {this is a non-betazero parameter}
            begin
{*****************************take out later*********************}
                if TRACE32=true then
                begin
                    writeln(userout2,'w /in l_and_w, NON-BETA0 param');
                    for i:=1 to goodcount do
                      writeln(userout2,'gdparval[',i:1,']=',goodparamvalue[i],' gdwidth=',
                        goodwidth[i]:2:1,' gdlength=',goodlength[i]:2:1);
                end;
{*****************************take out later*********************}
                if (differentl=false) and (differentw=true) then
                    {varying widths only}
                    begin
                     for i:=1 to goodcount do
                            leastsq2(i,1,goodcount,1 /goodwidth[i],
                            goodparamvalue[i],constant,linear1);
                     if TRACE32=true then writeln(userout2,
                          'diffl=false & diffw=true & constant=',constant,
                          ' linear1=',linear1);
                    end;
                if (differentl=true) and (differentw=false) then
                    {varying lengths only}
                    begin
                     for i:=1 to goodcount do
                            leastsq2(i,1,goodcount,1 /goodlength[i],
                            goodparamvalue[i],constant,linear1);
                     if TRACE32=true then writeln(userout2,
                          'diffl=true & diffw=false & constant=',constant,
                          ' linear1=',linear1);
                    end;
                if (differentl=true) and (differentw=true) then
```

```
                                    {varying lengths and widths}
                                   begin
                                    for i:=1 to goodcount do
                                          leastsq3(i,1,goodcount,1 /goodlength[i],
                                          1 /goodwidth[i],goodparamvalue[i],
                                          constant,linear1,linear2);
                                    if TRACE32=true then writeln(userout2,
                                         'diffl=true & diffw=true & constant=',constant,
                                         ' linear1=',linear1,' linear2=',linear2);
                                   end;
                            end;
              end   {end of goodcount>1}
        else if goodcount=1 then
              if betazero=true then
                    linear1:=cox*goodparamvalue[1]
                    {this is done so that betazero will be processpar[1,betazeroparam]
                     i.e. because values are divided by cox in load_up_process_
                     parameters to get mobility.  but if goodcount=1. then only
                     one good device exists and that parameter = beta.  so
                     multiply by cox so it is later divided back out}
              else
                    constant:=goodparamvalue[1];
          {else goodcount=2 and leastsq3 procedure doesn't work for 2 maxindex=2.
           (leastsq3 gives back bad values for goodcount=2}
end;    {end of procedure l_and_w_dependencies}

procedure process_parameter_errors(width,length,paramvalue:processarray;
                                   constant,linear1,linear2:real; entries:integer;
                                   var maxerror,errorwidth,errorlength:real);
{this procedure is called after the process parameters have been determined,
and it returns the worst error, and the w and l for that device}

var   count:integer;
      simparam:real;    {this is the simulated value of the parameter}
      errorparam:real;  {this is the error for a given device}

begin
     maxerror:=0.0;
     errorwidth:=width[1];
     errorlength:=length[1];
     for count:=1 to entries do
          begin
               simparam:=constant+1 /length[count]*linear1
                                  +1 /width[count]*linear2;
               if paramvalue[count]<>0.0 then
                    errorparam:=100*abs((paramvalue[count]-simparam)/
                                paramvalue[count])
               else if simparam=0 then   {paramvalue=0 and simparam=0}
                    errorparam:=0.0
               else                                {paramvalue=0 and simparam<>0}
                    errorparam:=100.0;
               if errorparam>maxerror then
                    begin    {this is worst error so far}
                         maxerror:=errorparam;
                         errorwidth:=width[count];
                         errorlength:=length[count];
                    end;
          end;
end;  {end of procedure to determine process parameter errors}
```

```
begin    {procedure load_up_process_parameters}
    {*****************************take out later*********************}
    if TRACE31=true then writeln(userout2,'load_up_proc_params--entering');
    {*****************************take out later*********************}
    {this is where the parameters beta, deltaW and deltaL are evaluated}
    case filenumb of
        1:reset(nchanenh,'NEDFILE.TEXT');
        2:reset(pchanenh,'PEDFILE.TEXT');
        3:reset(nchandep,'NDDFILE.TEXT');
        4:reset(pchandep,'PDDFILE.TEXT');
        5:reset(nchanzer,'NZDFILE.TEXT');
        6:reset(pchanzer,'PZDFILE.TEXT');
    end;  {end of case to reset files}
    for numbdevice:=1 to numbval do
        begin
            case filenumb of
                1:readln(nchanenh,widths);
                2:readln(pchanenh,widths);
                3:readln(nchandep,widths);
                4:readln(pchandep,widths);
                5:readln(nchanzer,widths);
                6:readln(pchanzer,widths);
            end;  {end of case to read width}
            case filenumb of
                1:readln(nchanenh,lengths);
                2:readln(pchanenh,lengths);
                3:readln(nchandep,lengths);
                4:readln(pchandep,lengths);
                5:readln(nchanzer,lengths);
                6:readln(pchanzer,lengths);
            end;  {end of case to read length}
            for skip:=1 to betazeroparam-1 do
                case filenumb of
                    1:readln(nchanenh,ignores);
                    2:readln(pchanenh,ignores);
                    3:readln(nchandep,ignores);
                    4:readln(pchandep,ignores);
                    5:readln(nchanzer,ignores);
                    6:readln(pchanzer,ignores);
                end;  {end of case to skip values up to betazero}
            case filenumb of
                1:readln(nchanenh,betazeros);
                2:readln(pchanenh,betazeros);
                3:readln(nchandep,betazeros);
                4:readln(pchandep,betazeros);
                5:readln(nchanzer,betazeros);
                6:readln(pchanzer,betazeros);
            end;  {end of case to read betazero string}
            for skip:=1 to numbbsim+1-betazeroparam do
                case filenumb of
                    1:readln(nchanenh,ignores);
                    2:readln(pchanenh,ignores);
                    3:readln(nchandep,ignores);
                    4:readln(pchandep,ignores);
                    5:readln(nchanzer,ignores);
                    6:readln(pchanzer,ignores);
                end;  {end of case to skip values after betazero}
            {here is where the strings are converted to real numbers}
            {the width and length here are MASK values}
            width[numbdevice]:=strtoreal(widths);
            length[numbdevice]:=strtoreal(lengths);
            paramvalue[numbdevice]:=strtoreal(betazeros);
        end;
```

```
{betazero values are now in array paramvalue}
{*****************************take out later*********************}
if TRACE33=true then lsq3:=true;
if TRACE31=true then writeln(userout2,
     'load_up_proc_params--entering beta 1_and_w depen');
{*****************************take out later*********************}
1_and_w_dependencies(width,length,paramvalue,mindL,uncox,
                        dWuncox,numbval,true,number_good_devices_error);
{*****************************take out later*********************}
if TRACE31=true then writeln(userout2,
     'just after 1&w for BETA & number_good_devices_error=goodcount=',
     number_good_devices_error:3);
if TRACE31=true then writeln(userout2,
     'load_up_proc_params--came out of beta 1_and_w depen');
{*****************************take out later*********************}
if number_good_devices_error<>2 then
  begin
      processpar[1,betazeroparam]:=uncox/cox;
      processpar[2,betazeroparam]:=-mindL;
      if uncox<>0 then
            processpar[3,betazeroparam]:=dWuncox/uncox
      else
            processpar[3,betazeroparam]:=0.0;
    {*****************************take out later*********************}
      if TRACE32=true then writeln(userout2,
                                          'processpar[1,6]=',processpar[1,6],
                                          ' processpar[2,6]=',processpar[2,6],
                                          ' processpar[3,6]=',processpar[3,6]);
    {*****************************take out later*********************}
      processpar[4,betazeroparam]:=0.0;
      processpar[5,betazeroparam]:=0.0;
      processpar[6,betazeroparam]:=0.0;
      {this is where the remaining parameters are evaluated}
      for paramcount:=1 to numbbsim do
            if paramcount<>betazeroparam then
              begin
                case filenumb of
                      1:reset(nchanenh,'NEDFILE.TEXT');
                      2:reset(pchanenh,'PEDFILE.TEXT');
                      3:reset(nchandep,'NDDFILE.TEXT');
                      4:reset(pchandep,'PDDFILE.TEXT');
                      5:reset(nchanzer,'NZDFILE.TEXT');
                      6:reset(pchanzer,'PZDFILE.TEXT');
                end;  {end of case to open files}
                for numbdevice:=1 to numbval do
                      begin
                            case filenumb of
                                  1:readln(nchanenh,widths);
                                  2:readln(pchanenh,widths);
                                  3:readln(nchandep,widths);
                                  4:readln(pchandep,widths);
                                  5:readln(nchanzer,widths);
                                  6:readln(pchanzer,widths);
                            end;  {end of case to read width}
                            case filenumb of
                                  1:readln(nchanenh,lengths);
                                  2:readln(pchanenh,lengths);
                                  3:readln(nchandep,lengths);
                                  4:readln(pchandep,lengths);
                                  5:readln(nchanzer,lengths);
                                  6:readln(pchanzer,lengths);
                            end;  {end of case to read length}
```

```
                    {width and length here are changed to EFFECTIVE values}
                        width[numbdevice]:=strtoreal(widths) +
                                                processpar[3,betazeroparam];
                        length[numbdevice]:=strtoreal(lengths) +
                                                processpar[2,betazeroparam];
                        for skip:=1 to paramcount-1 do
                            case filenumb of
                                1:readln(nchanenh,ignores);
                                2:readln(pchanenh,ignores);
                                3:readln(nchandep,ignores);
                                4:readln(pchandep,ignores);
                                5:readln(nchanzer,ignores);
                                6:readln(pchanzer,ignores);
                            end;  {end of case to skip values}
                        case filenumb of
                            1:readln(nchanenh,paramvalues);
                            2:readln(pchanenh,paramvalues);
                            3:readln(nchandep,paramvalues);
                            4:readln(pchandep,paramvalues);
                            5:readln(nchanzer,paramvalues);
                            6:readln(pchanzer,paramvalues);
                        end;  {end of case to read value}
                        for skip:=1 to numbbsim+1-paramcount do
                            case filenumb of
                                1:readln(nchanenh,ignores);
                                2:readln(pchanenh,ignores);
                                3:readln(nchandep,ignores);
                                4:readln(pchandep,ignores);
                                5:readln(nchanzer,ignores);
                                6:readln(pchanzer,ignores);
                            end;  {end of case to get to next parameter set}
                        paramvalue[numbdevice]:=strtoreal(paramvalues);
        {>>>>>>>>>>>>>>>>>>>>>>HERE IS WHERE BETA TERMS ARE CONVERTED<<<<<<<<<}
    {>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>TO MOBILITY TERMS.<<<<<<<<<<<<<<<<<<<<<<<<<<}
                        {now convert 4 beta terms to mobility terms}
                        if ((paramcount=9) or (paramcount=14) or
                            (paramcount=15) or (paramcount=16)) then
                            paramvalue[numbdevice]:=paramvalue[numbdevice]/
                                cox/width[numbdevice]*length[numbdevice];
{JRP}{>>>>>>>>>>>>>>>>>>>>>>HERE IS WHERE U1 TERMS ARE CONVERTED<<<<<<<<<<<<<<<<<}
        {JRP}{>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>TO U1 X LENGTH TERMS.<<<<<<<<<<<<<<<<<<<<}
        {JRP}                       {now convert 3 U1 terms to U1 X LENGTH terms}
        {JRP}                       if ((paramcount=8) or (paramcount=13) or
        {JRP}                           (paramcount=17)) then
        {JRP}                         paramvalue[numbdevice]:=paramvalue[numbdevice]
        {JRP}                                                 *length[numbdevice];
                                end;
                {*****************************take out later*********************}
                            if TRACE31=true then writeln(userout2,
                                'load_up_proc_params--entering other l_and_w depen');
                            if TRACE30=true then
                                begin
                                  if (paramcount=8) or (paramcount=13) or (paramcount=17) then
                                    begin
                                      writeln(userout2,
                                      'before calling l_and_w & paramcount=',
                                      paramcount:2);
                                      for i:=1 to numbval do
                                          write(userout2,' paramvalue[',i:1,']=',paramvalue[i]);
                                    end;
                                end;
                  {*********************take out later**********************}
                            l_and_w_dependencies(width,length,paramvalue,processpar[1,
                                                    paramcount],processpar[2,paramcount],
                                                processpar[3,paramcount],numbval,false,
```

```
                                        number_good_devices_error);
    {*****************************take out later*********************}
                if TRACE31=true then writeln(userout2,
                    'just after l&w for',paramcount:2,
                     ' param & number_good_devices_error=goodcount=',
                    number_good_devices_error:3);
                if TRACE33=true then lsq3:=false;
                if TRACE31=true then writeln(userout2,
                    'load_up_proc_params--came out of other l_and_w depen');
    {*****************************take out later*********************}
                process_parameter_errors(width,length,paramvalue,
                                        processpar[1,paramcount],
                                        processpar[2,paramcount],
                                        processpar[3,paramcount],numbval,
                                        processpar[4,paramcount],
                                        processpar[5,paramcount],
                                        processpar[6,paramcount]);
    {*****************************take out later*********************}
                if TRACE31=true then writeln(userout2,
                    'load_up_proc_params--came out of proc_param_errors',
                     ' for paramcont=',paramcount);
    {*****************************take out later*********************}
                {change max error device width and length to mask dimensions}
                processpar[5,paramcount]:=processpar[5,paramcount]-
                                        processpar[3,betazeroparam];
                processpar[6,paramcount]:=processpar[6,paramcount]-
                                        processpar[2,betazeroparam];
              end;
    {*****************************take out later*********************}
      if TRACE31=true then writeln(userout2,
          'FINISHED WITH ALL 17 PARAMS & number_good_devices_error=goodcount=',
          number_good_devices_error:3);
    {*****************************take out later*********************}
 {here is where the k1 variations with L and W are stored}
      case filenumb of
            1:nelk1:=processpar[2,3];
            2:pelk1:=processpar[2,3];
            3:ndlk1:=processpar[2,3];
            4:pdlk1:=processpar[2,3];
            5:nzlk1:=processpar[2,3];
            6:pzlk1:=processpar[2,3];
      end;  {end of case to fill in k1 variations with L}
      case filenumb of
            1:newk1:=processpar[3,3];
            2:pewk1:=processpar[3,3];
            3:ndwk1:=processpar[3,3];
            4:pdwk1:=processpar[3,3];
            5:nzwk1:=processpar[3,3];
            6:pzwk1:=processpar[3,3];
      end:   {end of case to fill in k1 variations with W}
      {here is where deltaL and deltaW values are stored}
      case filenumb of
            1:ned1:=processpar[2,betazeroparam];
            2:ped1:=processpar[2,betazeroparam];
            3:ndd1:=processpar[2,betazeroparam];
            4:pdd1:=processpar[2,betazeroparam];
            5:nzd1:=processpar[2,betazeroparam];
            6:pzd1:=processpar[2,betazeroparam];
      end;  {end of case to store deltaL values}
      case filenumb of
            1:nedw:=processpar[3,betazeroparam];
            2:pedw:=processpar[3,betazeroparam];
            3:nddw:=processpar[3,betazeroparam];
            4:pddw:=processpar[3,betazeroparam];
            5:nzdw:=processpar[3,betazeroparam];
```

bsim1.0.TEXT

...load_up_process_parameters

```
                6:pzdw:=processpar[3,betazeroparam];
            end;  {end of case to store deltaL values}
        end;  {of if number_good_device_error<>2....}
end;  {of procedure load_up_process_parameters}
```

write_into_the_output_file

```
procedure write_into_the_output_file(devicetype:integer);
{this procedure handles storing the present user process file into a
temporary file, and adding the latest process file to it}

var   i,j:integer;
      userfile:string[20];
      values:string[80];
      devicename:string[80];

begin
      reset(userout,output_file);
      rewrite(temporary,'TEMP.TEXT');
      while not(eof(userout)) do
            begin
                  readln(userout,values);
                  writeln(temporary,values);
            end;
      {here is where the new additions are made}
      case devicetype of
            1:devicename:='NMOSE';
            2:devicename:='PMOSE';
            3:devicename:='NMOSD';
            4:devicename:='PMOSD';
            5:devicename:='NMOSZ';
            6:devicename:='PMOSZ';
      end;  {end of statement to name the device type}
      writeln(temporary,devicename);
      writeln(temporary,'*PROCESS=',process);
      writeln(temporary,'*RUN=',lot);
      writeln(temporary,'*WAFER=',wafer);
      writeln(temporary,'*XPOS=',present_diex:1);
      writeln(temporary,'*YPOS=',present_diey:1);
      writeln(temporary,'*OPERATOR=',operator);
      writeln(temporary,'*DATE=',date);
      for j:=1 to numbbsim do
            begin
                  write(temporary,processpar[1,j]);
                  write(temporary,',');
                  write(temporary,processpar[2,j]);
                  write(temporary,',');
                  write(temporary,processpar[3,j]);
                  write(temporary,',');
                  write(temporary,processpar[4,j]);
                  write(temporary,',');
                  write(temporary,processpar[5,j]);
                  write(temporary,',');
                  writeln(temporary,processpar[6,j]);
            end;
      tox:=tox*1E-4;  {translate tox into microns from angstroms}
      write(temporary,tox);
      tox:=tox*1E4;
      write(temporary,',');
      temp:=temp-273.15;  {translates temp into celcius}
      write(temporary,temp);
      temp:=temp+273.15;
      write(temporary,',');
      writeln(temporary,vdd);
      close(temporary,'save');
```

...write_into_the_output_file

```
        reset(temporary,'TEMP.TEXT');
        rewrite(userout,output_file);
        while not(eof(temporary)) do
              begin
                    readln(temporary,values);
                    writeln(userout,values);
              end;
        close(userout,'save');
{*****************************take out later*****************************}
        if TRACE35=true then
             begin
                  reset(userout,output_file);
                  writeln(userout2,'Just created output file and here it is:');
                  while not(eof(userout)) do
                       begin
                            readln(userout,line_in_output_file);
                            writeln(userout2,line_in_output_file);
                       end;
                  writeln(userout2,'That was it');
                  close(userout,'save');
             end;
{*****************************take out later*****************************}
end:  {end of procedure write_into_the_output_file}
```

process_error_message

```
procedure process_error_message(filenumber:integer);
{this procedure displays an error message on the screen if an error
 is discovered while in load_up_process_parameters}

begin
     gotoxy(0,22);
     write('ONLY 2 GOOD DEVICES FOR');
     case filenumber of
        1:write(' NMOS-ENHANCEMENT.');
        2:write(' PMOS-ENHANCEMENT.');
        3:write(' NMOS-DEPLETION.');
        4:write(' PMOS-DEPLETION.');
        5:write(' NMOS-ZERO-THRESHOLD.');
        6:write(' PMOS-ZERO-THRESHOLD.');
     end: {of case}
     write(' NO PROCESS FILE DEVELOPED.');
     if (mode='2') or (mode='3') then
          begin
               gotoxy(21,23);
               write('Press "ENTER" to continue  >');
               selection_input(49,23,chr(ord(32)),chr(ord(32)),continue);
          end;
     gotoxy(0,22);
     write('                                                                    ');
end;



begin  {begin the main procedure to make a process file}
      gotoxy(21,23);
      write('DEVELOPING PROCESS FILES');
      for i:=1 to 6 do
           begin
                case i of
                   1:number_good_devices:=numbne;
                   2:number_good_devices:=numbpe;
                   3:number_good_devices:=numbnd;
                   4:number_good_devices:=numbpd;
                   5:number_good_devices:=numbnz;
                   6:number_good_devices:=numbpz;
```

```
                end; {of case}
                if number_good_devices>0 then
                    begin
                        load_up_process_parameters(i,
                                                   number_good_devices,process_error);
                        if process_error<>2 then
                            write_into_the_output_file(i)
                        else
                            process_error_message(i);
                    end;
            end;
end:  {end of procedure to make a process file}

{************************MAIN FILE CREATION ROUTINE*********************}

begin   {program process_file_development begins here}
        if new_wafer then
            begin
                rewrite(userout,output_file);
                close(userout,'save');
            end;
        if new_die then
             die_file_creation;    {this makes up the die files}
{jrp}if (measure_error=0) and (leastsq_divide_by_zero=false) and
           (extract_error=0) then    {do ONLY if the device tested was not defective}
                store_parameters_in_die_files(device)  {this stores device parameters}
        else if (leastsq_divide_by_zero=true) then
            begin
                gotoxy(0,22);
                write('THESE PARAMETERS ARE NOT BEING SAVED');
{jrp}           write(': DIVIDE BY 0 IN LEAST SQUARE PROCEDURE');
{jrp}       end;
        if end_of_die then
            begin
                if TRACE28=true then
                    begin
                        writeln(userout2,'within p-f-d and at end of die');
                        maxidsstring='7.5E-5';  {set a value if reading in
                        ids array from a file instead of measuring devices}
                    end;
                make_process_files;
            end;
end;

{************************ERASE TEMPORARY FILES**************************}

procedure erase_temporary_files;                          erase_temporary_files
{this procedure is called at the end of the bsim program to purge files that
are not needed by spice}  {to avoid purging files, the program should be
terminated with 'stop' after it returns to the main menu}

begin
        reset(nchanenh,'NEDFILE.TEXT');
        close(nchanenh,'PURGE');
        reset(pchanenh,'PEDFILE.TEXT');
        close(pchanenh,'PURGE');
        reset(nchandep,'NDDFILE.TEXT');
        close(nchandep,'PURGE');
        reset(pchandep,'PDDFILE.TEXT');
        close(pchandep,'PURGE');
        reset(nchanzer,'NZDFILE.TEXT');
        close(nchanzer,'PURGE');
        reset(pchanzer,'PZDFILE.TEXT');
```

..erase_temporary_files

```
    close(pchanzer,'PURGE');
    reset(temporary,'TEMP.TEXT');
    close(temporary,'PURGE');
end;

{************************************************************************}
{**********************AUTOMATIC PROBER ROUTINES**********************}
{************************************************************************}
```

read_prober_file

```
procedure read_prober_file(var xdie_size,ydie_size,origin_diex,origin_diey,
                           number_die,number_devices:integer);
{the origin x and y die locations are determined while reading the die step
array.  The number of die, and number of devices are also calculated}
{last change made: JUNE 19, 1984  by: JRP}

var   xdie_sizestr,ydie_sizestr,command:linestring;
      i,j:integer;
      prober:text;

begin
      {************************take out later*****************************}
      if TRACE21=true then writeln(userout2,'entering read_prober_file');
      {************************take out later*****************************}
      reset(prober,prober_file);    {opens up the prober file for reading}
      number_die:=0;    {resets die counter}
      if mode<>'3' then                {if mode='3', die sizes are not included}
           begin
                {here is where the die size is read}
                readln(prober,xdie_sizestr);
                readln(prober,ydie_sizestr);
                if TRACE23=true then writeln(userout2,
                     'about to do strtoreal on xdie_sizestr');
                xdie_size:=trunc(strtoreal(xdie_sizestr));
                ydie_size:=trunc(strtoreal(ydie_sizestr));
                if TRACE23=true then writeln(userout2,
                     'finished with strtoreal on ydie_sizestr');
           end; {end of mode<>'3' check}
      {now read in the step array into step_array}
      {the array is read from left to right, top to bottom.  location [1,1] is
      in the upper left corner, and location [20,20] is the lower right corner}
      for j:=1 to 20 do  {scans rows}
           begin
                for i:=1 to 19 do  {scans columns}
                     read(prober,step_array[i,j]);
                readln(prober,step_array[20,j]);
           end;
      if TRACE23=true then writeln(userout2,
           'finished with reading in step array');
      {determine the number of die, and the origin die}
      for j:=1 to 20 do
           for i:=1 to 20 do
                begin
                     if step_array[i,j]<>'0' then
                          number_die:=number_die+1;
                     if (step_array[i,j]='X') or (step_array[i,j]='x') then
                          begin
                               origin_diex:=i;
                               origin_diey:=j;
                          end;
                end;
```

```
{**********take out later**********}
if TRACE23=true then write(userout2,'number_die = ',number_die);
if TRACE22=true then writeln(userout2,
     'about to start internal die stepping');
i:=0;
j:=0;
{**********take out later**********}
{here is where the internal die stepping is done}
number_devices:=0;
{sd,ss,sg,sb,w,l,dt,ed,mx, and my are valid command lines}
while not(eof(prober)) do
      begin
            readln(prober,command);
            if (command[1]='s') or (command[1]='S') then
                  begin
                        if (command[2]='d') or (command[2]='D') then
                              sd[number_devices]:=digit(command[4])
                        else if (command[2]='g') or (command[2]='G') then
                              sg[number_devices]:=digit(command[4])
                        else if (command[2]='s') or (command[2]='S') then
                              ss[number_devices]:=digit(command[4])
                        else if (command[2]='b') or (command[2]='B') then
                              sb[number_devices]:=digit(command[4]);
{##########take out later##########}
if TRACE23=true then write(userout2,i,'finish with s commands ');
i:=i+1;
{##########take out later##########}
                  end
            else if (command[1]='w') or (command[1]='W') then
                  begin
                        strdelete(command,1,2);
                        w[number_devices]:=strtoreal(command);
{##########take out later##########}
if TRACE23=true then write(userout2,i,'finish with w commands ');
i:=i+1;
{##########take out later##########}
                  end
            else if (command[1]='l') or (command[1]='L') then
                  begin
                        strdelete(command,1,2);
                        l[number_devices]:=strtoreal(command);
{##########take out later##########}
if TRACE23=true then write(userout2,i,'finish with l commands ');
i:=i+1;
{##########take out later##########}
                  end
            else if ((command[1]='d') or (command[1]='D')) and
            ((command[2]='t') or (command[2]='T')) then
                  begin
                        strdelete(command,1,3);
                        dt[number_devices]:=trunc(strtoreal(command));
{##########take out later##########}
if TRACE23=true then write(userout2,i,'finish with d commands ');
i:=i+1;
{##########take out later##########}
                  end
            else if ((command[1]='e') or (command[1]='E')) and
            ((command[2]='d') or (command[2]='D')) then
                  begin
                        strdelete(command,1,3);
                        ed[number_devices]:=trunc(strtoreal(command));
{##########take out later##########}
if TRACE23=true then write(userout2,i,'finish with e commands ');
i:=i+1;
```

..read_prober_file

```
{############################take out later##############################}
                    end
                else if ((command[1]='m') or (command[1]='M')) then
                    begin
                        if (command[2]='x') or (command[2]='X') then
                            begin
                                strdelete(command,1,3);
                                mx[number_devices]:=trunc(strtoreal(command));
                            end
                        else if (command[2]='y') or (command[2]='Y') then
                            begin
                                strdelete(command,1,3);
                                my[number_devices]:=trunc(strtoreal(command));
                            end;  {<-------take out ; later}
    {############################take out later##############################}
    if TRACE23=true then writeln(userout2,i,'finish with m commands ');
    i:=i+1;
    {############################take out later##############################}
                    end
                else if (command[1]='*') and (command[2]='*') then
                    number_devices:=number_devices+1;
    {############################take out later##############################}
    if TRACE23=true then writeln(userout2,i,'finish with * commands ');
    i:=i+1;
    {############################take out later##############################}
        end;  {end of while not eof}
    number_devices:=number_devices-1;  {this handles final **}
    if TRACE23=true then writeln(userout2,'number_devices = ',number_devices);
    close(prober,'save');
    {************************take out later*****************************}
    if TRACE21=true then writeln(userout2,'leaving read_prober_file');
    {************************take out later*****************************}
end; {end of procedure to read the prober file}
```

load_measure_two_phif_array

```
procedure load_measure_two_phif_array;
{this procedure handles the order of all devices on a die so that the
largest and best for measuring 20f comes first for every type}

var   i:integer;
```

find_and_switch_best_device

```
procedure find_and_switch_best_device(dtpresent,edpresent:integer);
{this procedure scans all devices of a given type, and selects the most
optimal in terms of geometry for extracting 20f on.}
{last change made: NOV' 8, 1984   by: JRP}

var   largeindex,firstindex:integer;
      area,small_dimension,present_area,present_small_dimension:real;
      count,tempinteger:integer;
      tempreal:real;

begin
      largeindex:=0;
      firstindex:=0;
      area:=0.0;
      small_dimension:=0.0;
      {scan for first occurrence of device type}
      for count:=1 to number_devices do
        begin
          if (ed[count]=edpresent) and (dt[count]=dtpresent) then
            begin
              if firstindex=0 then
                  begin
                      firstindex:=count;
```

```
                                largeindex:=count;
                                area:=w[count]*l[count];
                                if w[count]>l[count] then
                                    small_dimension:=l[count]
                                else
                                    small_dimension:=w[count];
                            end
                        else
                            begin
                                if ((w[count]*l[count])>area) and
                                    (w[count]>small_dimension) and
                                    (l[count]>small_dimension) then
                                        begin
                                            if w[count]>l[count] then
                                                small_dimension:=l[count]
                                            else
                                                small_dimension:=w[count];
                                            area:=w[count]*l[count];
                                            largeindex:=count;
                                        end;
                            end;
                    end;
            end;
        {here is where the switch is done to make the best device first}
        if firstindex>0 then
            begin
{JRP}           if mode<>'3' then  {don't switch in manual probe station mode}
{JRP}               begin
                        tempinteger:=sd[firstindex];
                        sd[firstindex]:=sd[largeindex];
                        sd[largeindex]:=tempinteger;
                        tempinteger:=sg[firstindex];
                        sg[firstindex]:=sg[largeindex];
                        sg[largeindex]:=tempinteger;
                        tempinteger:=ss[firstindex];
                        ss[firstindex]:=ss[largeindex];
                        ss[largeindex]:=tempinteger;
                        tempinteger:=sb[firstindex];
                        sb[firstindex]:=sb[largeindex];
                        sb[largeindex]:=tempinteger;
                        tempinteger:=mx[firstindex];
                        mx[firstindex]:=mx[largeindex];
                        mx[largeindex]:=tempinteger;
                        tempinteger:=my[firstindex];
                        my[firstindex]:=my[largeindex];
                        my[largeindex]:=tempinteger;
                        tempreal:=w[firstindex];
                        w[firstindex]:=w[largeindex];
{JRP}                   w[largeindex]:=tempreal;
                        tempreal:=l[firstindex];
                        l[firstindex]:=l[largeindex];
                        l[largeindex]:=tempreal;
{JRP}               end;
                meas_two_phif[firstindex]:=TRUE;
            end;
    end;    {end of procedure to set up 2of measurement order}

    begin   {begin procedure to handle various device types and select order}
        {############################take out later################################}
        if TRACE21=true then writeln(userout2,
            'entering load_measure_two_phif_array');
        {############################take out later################################}
        for i:=1 to number_devices do
            meas_two_phif[i]:=FALSE;
        find_and_switch_best_device(-1,-1);
```

```
        find_and_switch_best_device(-1,0);
        find_and_switch_best_device(-1,1);
        find_and_switch_best_device(1,-1);
        find_and_switch_best_device(1,0);
        find_and_switch_best_device(1,1);
        {the following values of phif2 are stored in case the first device
        which is tested for phif2 is a bad device}
        pdphif2:=0.6;
        pzphif2:=0.6;
        pephif2:=0.6;
        ndphif2:=0.6;
        nzphif2:=0.6;
        nephif2:=0.6;
        {############################take out later##############################}
        if TRACE23=true then writeln(userout2,'pdphifs etc... values set to 0.6');
        if TRACE21=true then writeln(userout2,'leaving load_meas_2_phif_array');
        {############################take out later##############################}
end;    {end of procedure to handle 20f array}
```

load_automatic_parameters

```
procedure load_automatic_parameters(device_number:integer);
{this procedure loads up the smu connections, w, l, device type etc.
from the global arrays which have the information for the device}

begin
        smud:=sd[device_number];
        smug:=sg[device_number];
        smus:=ss[device_number];
        smub:=sb[device_number];
        width:=w[device_number];
        length:=l[device_number];
        device:=dt[device_number];
        enhancement_depletion:=ed[device_number];
        meas_20f:=meas_two_phif[device_number];
end;
```

prober_move

```
procedure prober_move(deltax,deltay:integer);
{this routine handles all movement of the prober}
{the step array size must be previously set}
{last change made: JUNE 20, 1984   by: JRP}

var     ready:boolean;
        i,spoll_byte,nextposition:integer;
        message:array[1..10] of char;
        deltaxstr,deltaystr:string[6];

begin
        {here is where deltax, and deltay are converted to strings}
        deltaxstr:='      ';
        deltaystr:='      ';
        strwrite(deltaxstr,1,nextposition,deltax:1);
        strwrite(deltaystr,1,nextposition,deltay:1);
        {here is where the trailing blanks are trimmed}
        deltaxstr:=strrtrim(deltaxstr);
        deltaystr:=strrtrim(deltaystr);
        talk_to_hpib(hpib_prober_address);
        {Z down before move}
        writestringln(700+hpib_prober_address,'ZD');
        wait_till_bit7_IOSTATUS_set;
        writestring(700+hpib_prober_address,'MOX');
```

..prober_move

```
        writestring(700+hpib_prober_address,deltaxstr);
        writestring(700+hpib_prober_address,'Y');
        writestringln(700+hpib_prober_address,deltaystr);
        wait_till_bit7_IOSTATUS_set;
        {check for the signal that message is to be sent from prober}
        ready:=false;
        while not(ready) do
                ready:=requested(7);
        spoll_byte:=spoll(700+hpib_prober_address);
        {here is where the prober message is read}
        listen_to_hpib(hpib_prober_address);
        i:=1;
        repeat
                begin
                        readchar(7,message[i]);
                        i:=i+1;
                end;
        until end_set(7);
        {here is where the ZUP is issued}
        talk_to_hpib(hpib_prober_address);
        writestringln(700+hpib_prober_address,'ZU');
        wait_till_bit7_IOSTATUS_set;
        untalk(7);
        unlisten(7);
end;    {end of procedure to move prober}
```

set_die_size

```
procedure set_die_size(xmicron,ymicron:integer);
{this procedure is called to set the die size of the wafer.  For internal
die stepping, the procedure sets the die size at 1 micron}
{last change made: JUNE 20, 1984   by: JRP}

var   xmicronstr,ymicronstr:string[6];
      nextposition:integer;

begin
      {here is where the die size strings are setup}
      xmicronstr:='      ';
      ymicronstr:='      ';
      strwrite(xmicronstr,1,nextposition,xmicron:1);
      strwrite(ymicronstr,1,nextposition,ymicron:1);
      {trim the trailing blanks}
      xmicronstr:=strrtrim(xmicronstr);
      ymicronstr:=strrtrim(ymicronstr);
      talk_to_hpib(hpib_prober_address);
      writestring(700+hpib_prober_address,'SP1X');  {this does die size preset}
      writestring(700+hpib_prober_address,xmicronstr);
      writestring(700+hpib_prober_address,'Y');
      writestringln(700+hpib_prober_address,ymicronstr);
      wait_till_bit7_IOSTATUS_set;
      writestringln(700+hpib_prober_address,'SP2X0Y0');  {this does origin preset}
      wait_till_bit7_IOSTATUS_set;
      untalk(7);
      unlisten(7);
end;    {end of procedure to set die size}
```

set_up_prober_initial_conditions

```
procedure set_up_prober_initial_conditions;
{this procedure sets up the prober to use microns instead of mils,
 and it sets up the proper coordinate system quadrant (quadrant 2)}
{last change made: JUNE 20, 1984   by: JRP}

begin
```

```
    {##########################take out later##############################}
    if TRACE21=true then writeln(userout2,
        'entering set_up_prober_initial_conditions');
    {##########################take out later##############################}
    talk_to_hpib(hpib_prober_address);
    writestringln(700+hpib_prober_address,'SM1U1');
    wait_till_bit7_IOSTATUS_set;
    writestringln(700+hpib_prober_address,'SM2Q2');  {sets up quadrant=2}
    wait_till_bit7_IOSTATUS_set;
    untalk(7);
    unlisten(7);
    {##########################take out later##############################}
    if TRACE21=true then writeln(userout2,
        'leaving set_up_prober_initial_conditions');
    {##########################take out later##############################}
end; {end of procedure to set micron units}
```

step_to_next_die

```
procedure step_to_next_die(var present_diex,present_diey:integer;
                           var initial_jump,end_of_wafer:boolean);

var   i,j:integer;
      next_diex,next_diey:integer;

begin
      {if initial jump then find the first die}
      next_diex:=0;
      next_diey:=0;
      end_of_wafer:=false;
      if initial_jump=true then
            begin
                  for j:=1 to 20 do
                        for i:=1 to 20 do
                              if (step_array[i,j]<>'0') and (next_diex=0) then
                                    begin
                                          next_diex:=i;
                                          next_diey:=j;
                                    end;
                  initial_jump:=false;
            end   {end of handling for initil jump}
      else
            begin
                  {check present row}
                  for i:=present_diex+1 to 20 do
                        if (step_array[i,present_diey]<>'0') and (next_diex=0) then
                              begin
                                    next_diex:=i;
                                    next_diey:=present_diey;
                              end;
                  if next_diex=0 then  {next die not in the same row}
                        for j:=present_diey+1 to 20 do
                              for i:=1 to 20 do
                                    if (step_array[i,j]<>'0') and (next_diex=0) then
                                          begin
                                                next_diex:=i;
                                                next_diey:=j;
                                          end;
            end; {end of getting next_die locations for non-initial jump}
       if next_diex=0 then
            end_of_wafer:=true
       else
            begin
```

...step_to_next_die

```
                {now if using an automatic prober, move to next die}
                if (mode<>'3') then
                    prober_move(next_diex-present_diex,next_diey-present_diey);
                present_diex:=next_diex;
                present_diey:=next_diey;
            end;
end;  {end of procedure to step to next die}
```

unload_wafer

```
procedure unload_wafer;
{this procedure brings the wafer back to the unload position}
{last change made: JUNE 20, 1984  by: JRP}

begin
    talk_to_hpib(hpib_prober_address);
    writestringln(700+hpib_prober_address,'HO');
    wait_till_bit7_IOSTATUS_set;
    writestring(700+hpib_prober_address,'MEOVER 1 BILLION BSIM EXTRACTION  ');
    writestringln(700+hpib_prober_address,'          CUSTOMERS SERVED');
    wait_till_bit7_IOSTATUS_set;
    untalk(7);
    unlisten(7);
end;  {end of procedure to unload wafer}
```

```
{*************************************************************************}
{*********************MAIN BSIM PROGRAM*********************************}
{*************************************************************************}


 begin  {this is the main anne3 program}
    lsq3:=false;
    TRACE21:=false;
    TRACE22:=false;
    TRACE23:=false;
    TRACE24:=false;
    TRACE25:=false;
    TRACE26:=false;
    TRACE27:=false;
    TRACE30:=false;
    TRACE31:=false;
    TRACE32:=false;
    TRACE33:=false;
    TRACE34:=false;
    TRACE35:=false;          {TRACE28,29, and 29b set by user}
{jrp}TRACE36:=false;
{jrp}TRACE37:=false;
{jrp}TRACE38:=false;
{jrp}TRACE39:=false;
{JRP}TRACE40:=false;
{JRP}TRACE41:=false;
{JRP}TRACEVTH:=false;
    {TRACE21:=true;
    TRACE22:=true;
    TRACE23:=true;
    TRACE24:=true;
    TRACE25:=true;
    TRACE26:=true;
    TRACE27:=true;
    TRACE30:=true;
    TRACE31:=true;
    TRACE32:=true;
    TRACE33:=true;
    TRACE34:=true;}
```

```
        {TRACE35:=true;}
    {jrp}{TRACE36:=true;}
    {jrp}{TRACE37:=true;}
    {jrp}{TRACE38:=true;}
    {jrp}{TRACE39:=true;}
    {JRP}{TRACE40:=true;}
    {JRP}{TRACE41:=true;}
    {JRP}{TRACE\TH:=true;}
        rewrite(userout2,'#6:');

    {*********************************************************************}
{HERE IS WHERE HPIB ANALYZER AND PROBER ADDRESSES CAN BE CHANGED GLOBALLY }
    {*********************************************************************}
        hpib_analyzer_address:=17;
        hpib_prober_address:=14;
        first_time_thru_program:=true;
        end_of_program:=false;
        all_die_are_bad:=true;        {initially, no good devices have been found}
        first_good_die:=0;
        ioinitialize;  {initializes all of the I/O devices and interfaces}
        repeat
          all_devices_are_bad:=true; {initially, at present die, no good devices}
          initial_bsim_page;  {provides menu, and determines operation mode}
          first_good_device:=0;     {= 1 when found first good device}
          fake_meas_20f:=false;     {in case 1st device is defective}
          if mode<>'5' then
                begin
                    initial_status_inputs;  {this requests all necessary inputs}
                    first_time_thru_program:=false; {decided not to quit immediately}
                end;
           case mode of     {this calls sequence of routines to service mode}
             '1','2':begin     {this is for automatic prober and
                                semi-automatic--(with automatic prober) modes}
    {*****************************take out later**********************
                    gotoxy(0,0);
                    write('                                                    ');
                    gotoxy(0,0);
                    write('want to create 120 ids value array');
                    write(' instead of measuring? (Y/N)  >');
                    yes_no_selection_input(66,0,TRACE29);
                    if TRACE29=true then
                        begin
                            gotoxy(0,1);
                            write('                                                    ');
                            write('want to print out 120 ids value array? >');
                            yes_no_selection_input(40,1,TRACE29b);
                        end;
                    if TRACE29=false then
                        TRACE28:=true
                    else
                        TRACE28:=false;
    ****************************take out later*********************}
                    read_prober_file(xdie_size,ydie_size,origin_diex,
                                     origin_diey,number_die,number_devices);
                    load_measure_two_phif_array;
                    {all probing information is now known}
                    initial_jump:=true;                {go from origin to first die}
                    end_of_wafer:=false;
                    set_up_prober_initial_conditions;
                    set_die_size(xdie_size,ydie_size);
                    present_diex:=origin_diex;
                    present_diey:=origin_diey;
                    step_to_next_die(present_diex,present_diey,
                                     initial_jump,end_of_wafer);
```

```
                {now the prober is at the first die to be tested}
                present_die:=1;
                while not(end_of_wafer) do
                  begin
                      {here is where a die is tested}
                      set_die_size(1,1);  {micron units}
                      for present_device:=1 to number_devices do
                          begin
                              leastsq_divide_by_zero:=false;
                              prober_move(mx[present_device],my[present_device]);
                              {for first device of die, user has already been
                               asked to check probes, so don't ask again}
                               if ((mode='2') and (present_device<>1)) then
                                   begin
                                        gotoxy(22,23);
                                        write('Are probes on next device? If so,');
                                        write(' Press "ENTER"   >');
                                        selection_input(72,23,chr(ord(32)),chr(ord(32)),continue);
                                   end;
                              device:=0;
                              enhancement_depletion:=0;  {loads up device types}
                              load_automatic_parameters(present_device);
                              initial_status_display;
                              time_count:=1; {for bsim_timer routine}
                              {now at present device}
{*****************************take out later*********************}
                              if TRACE28=true then
                                   begin
                                        if present_device=1 then
                                            reset(tempfile,'file1.TEXT');
                                        if present_device=2 then
                                            reset(tempfile,'file2.TEXT');
                                        if present_device=3 then
                                            reset(tempfile,'file3.TEXT');
                                        if present_device=4 then
                                            reset(tempfile,'file4.TEXT');
                                        for iii:=1 to numbvd do
                                          begin
                                            for jjj:=1 to numbvb do
                                              begin
                                                for kkk:=1 to numbvg do
                                                  begin
                                                      readln(tempfile,
                                                        idsstring[iii,jjj,kkk]);
                                                      ids[iii,jjj,kkk]:=strtoreal(idsstring[iii,jjj,kkk]);
                                                  end;
                                              end;
                                          end;
                                        readln(tempfile,vgsstring[1,1,1]);  {***** line}
                                        for iii:=1 to numbvd do
                                          begin
                                            for jjj:=1 to numbvb do
                                              begin
                                                for kkk:=1 to numbvg do
                                                  begin
                                                      readln(tempfile,
                                                        vgsstring[iii,jjj,kkk]);
                                                      vgs[iii,jjj,kkk]:=strtoreal(vgsstring[iii,jjj,kkk]);
                                                  end;
                                              end;
                                          end;
                                        close(tempfile,'save');
                                        device:=1;
                                        measure_error:=0;
                                        vds[1]:=0.1;
```

```
                                        vds[2]:=0.2;
                                        vds[3]:=4.5;
                                        vds[4]:=5.0;
                                        vbs[1]:=-5.0;
                                        vbs[2]:=-4.0;
                                        vbs[3]:=-3.0;
                                        vbs[4]:=-2.0;
                                        vbs[5]:=-1.0;
                                        vbs[6]:=0.0;
                                    end;
                                if TRACE28=false then
        {*****************************take out later*********************}
                                measure_device(device,measure_error,smud,smug,smus,smub,vdd);
                                new_die:=false;
                                end_of_die:=false;
                                if (all_devices_are_bad=true) and (measure_error=0) then
                                    begin       {found 1st good device of present die}
                                        all_devices_are_bad:=false;
                                        first_good_device:=1;
                                        fake_meas_20f:=TRUE;
                                    end;
                                if (all_die_are_bad=true) and
                                                    (all_devices_are_bad=false) then
                                    begin     {found first good die of wafer}
                                        all_die_are_bad:=false;
                                        first_good_die:=1;
                                    end;
                                if (present_device=1) or (first_good_device=1) then
                                      new_die:=true; {either 1st device or 1st good device}
                                if present_device=number_devices then
                                      end_of_die:=true;
                                if (new_die=true) and
                                    ( (present_die=1) or (first_good_die=1) ) then
                                      new_wafer:=true;  {either 1st device or 1st
                                            good device) and (1st die or 1st good die)
                                            ...simply put...1st good device on wafer}
                                if (measure_error=0) then
                                      extract_device_parameters;
                                {within process_file_development, if measure_error<>}
{jrp}                           {0 & extract_error<>0 then do nothing.  however, if}
                                {end_of_die is true, make_process_files IS CALLED}
                                process_file_development(new_die,end_of_die,new_wafer);
                                first_good_device:=0;
                                first_good_die:=0;
                                fake_meas_20f:=false;
                            end;  {end of testing all devices on the die}
{*****************************take out later*************************}
    if TRACE35=true then
        begin
            reset(userout,output_file);
            writeln(userout2,
                'Just before entering po_graphics and here is output file:');
            while not(eof(userout)) do
                begin
                    readln(userout,line_in_output_file);
                    writeln(userout2,line_in_output_file);
                end;
            writeln(userout2,'That was it');
            close(userout,'save');
        end;
{*****************************take out later*************************}
                        if po_graphics_wanted=true then
```

```
{jrp}                    begin
{jrp}                        if mode='2' then
{jrp}                            begin
{jrp}                                gotoxy(21,23);
{jrp}                                write('Press "ENTER" to continue  >');
{jrp}                                selection_input(49,23,chr(ord(32)),
                                                    chr(ord(32)),continue);
{jrp}                            end;
                             po_graphics;
{jrp}                    end;
                    {return to die origin}
                    prober_move(0,0);
                    set_die_size(xdie_size,ydie_size);
                    step_to_next_die(present_diex,present_diey,
                                     initial_jump,end_of_wafer);
                    present_die:=present_die+1;
                    {stop for semi-automatic mode if not at end of wafer}
                    if ((mode='2') and (not(end_of_wafer))) then
                      begin
                        gotoxy(21,23);
                        write('Are probes on 1st device of next die?');
                        write(' If so, Hit "ENTER" >');
                        selection_input(80,23,chr(ord(32)),chr(ord(32)),continue);
                      end;
                  end;  {now prober is at next die or wafer is done}
                  {the wafer is now complete}
                iv_graphics;
                {unload_wafer;**********put back in later}{sends wafer to home position}
              end;  {end of automatic probing mode}
            '3':begin    {semi-automatic (with manual prober) mode}
                if TRACE21=true then writeln(userout2,'entering mode "3" loop');
                read_prober_file(xdie_size,ydie_size,origin_diex,
                                 origin_diey,number_die,number_devices);
                if TRACE21=true then writeln(userout2,
                    'entering load_measure_two_phif_array');
                load_measure_two_phif_array;
                {all probing information is now known}
                if TRACE23=true then writeln(userout2,
                     'Phif2 now equals ',c2_phif2:1:3);
                initial_jump:=true;
                end_of_wafer:=false;
                present_diex:=origin_diex;
                present_diey:=origin_diey;
                {stepping does not actually occur for mode='3', but bookkeeping
                 of die location is done in step_to_next_die}
                step_to_next_die(present_diex,present_diey,
                                 initial_jump,end_of_wafer);
                {the prober should have been placed at the 1st die to be tested}
                present_die:=1;
{JRP}           if TRACE40=false then begin
                while not(end_of_wafer) do
                  begin
                    {here is where a die is tested}
                    for present_device:=1 to number_devices do
                      begin
                        leastsq_divide_by_zero:=false;
                        {for first device of die, user has already been
                         asked to check probes, so don't ask again}
                        if (present_device<>1) then
                          begin
                            gotoxy(22,23);
                            write('Move the probes to the next device');
                            write(' and Press "ENTER"  >');
                            selection_input(77,23,chr(ord(32)),chr(ord(32)),continue);
                          end;
```

```
                    device:=0;
                    enhancement_depletion:=0;  {loads up device types}
                    load_automatic_parameters(present_device);
                    if TRACE21=true then writeln(userout2,
                         'entering initial_status_display');
                    initial_status_display;
                    time_count:=1; {for bsim_timer routine}
                    {now at present device}
                    if TRACE21=true then writeln(userout2,
                         'entering measure_device');
                    measure_device(device,measure_error,smud,smug,smus,smub,vdd);
                    new_die:=false;
                    end_of_die:=false;
                    new_wafer:=false;
                    if (all_devices_are_bad=true) and (measure_error=0) then
                         begin        {found 1st good device of present die}
                              all_devices_are_bad:=false;
                              first_good_device:=1;
                              fake_meas_20f:=TRUE;
                         end;
                    if (all_die_are_bad=true) and
                                        (all_devices_are_bad=false) then
                         begin      {found first good die of wafer}
                              all_die_are_bad:=false;
                              first_good_die:=1;
                         end;
                    if (present_device=1) or (first_good_device=1) then
                         new_die:=true; {either 1st device or 1st good device}
                    if present_device=number_devices then
                         end_of_die:=true;
                    if (new_die=true) and
                       ( (present_die=1) or (first_good_die=1) ) then
                         new_wafer:=true;  {(either 1st device or 1st
                              good device) and (1st die or 1st good die)
                              ...simply put...1st good device on wafer}
                    if (measure_error=0) then
                         extract_device_parameters;
{jrp}               {within process_file_development, if measure_error<>}
{jrp}               {0 & extract_error<>0 then do nothing.  however, if}
{jrp}               {end_of_die is true, make_process_files IS CALLED}
                    process_file_development(new_die,end_of_die,new_wafer);
                    first_good_device:=0;
                    first_good_die:=0;
                    fake_meas_20f:=false;
               end;  {end of testing all devices on the die}
          if po_graphics_wanted=true then
{jrp}          begin
{jrp}               gotoxy(21,23);
{jrp}               write('Press "ENTER" to continue  >');
{jrp}               selection_input(49,23,chr(ord(32)),chr(ord(32)),
                                        continue);
                    po_graphics;
{jrp}          end;
          {stepping does not actually occur for mode='3', but
           bookkeeping of die location is done in step_to_next_die}
          step_to_next_die(present_diex,present_diey,
                         initial_jump,end_of_wafer);
          present_die:=present_die+1;
          {stop for move to new die location}
          if (not(end_of_wafer)) then
               begin
                    gotoxy(21,23);
                    write('Move probes to 1st device of next die.');
                    write(' Then Hit "ENTER"  >');
                    selection_input(79,23,chr(ord(32)),chr(ord(32)),continue);
```

```
                        end;
                    end;  {now prober is at next die or wafer is done}
{JRP}               end   {of if TRACE40=false.....}
{JRP}               else
{JRP}                   po_graphics;
                {the wafer is now complete -- let's go into graphics mode}
                iv_graphics;
                end;  {end of semi-automatic [with manual prober] probing mode}


            '4':begin   {this is the single device probing mode}
                    initial_status_display;
                    time_count:=1;
                    number_devices:=1;
                    number_die:=1;
                    present_device:=1;
                    present_die:=1;
                    device:=0;
                    {enhancement_depletion:=0; *****determined in input section}
                    meas_20f:=FALSE;
                    measure_device(device,measure_error,smud,smug,smus,smub,vdd);
                    new_die:=true;
                    end_of_die:=true;
                    new_wafer:=true;
{JRP}               if (all_devices_are_bad=true) and (measure_error=0) then
                        all_die_are_bad:=false;
                    if (measure_error=0) then
                        extract_device_parameters;
                    process_file_development(new_die,end_of_die,new_wafer);
                    gotoxy(21,23);
                    if TRACE39=true then writeln(userout2,
                                                 'measure_error=',measure_error:1,
                                                 'extract_error=',extract_error:1,
                                                 'process_error=',process_error:1);
{JRP}               iv_graphics;
                end;
            '5':end_of_program:=true;
          end;  {end of case statement}
        until end_of_program=true;
        if (first_time_thru_program=false) and
{JRP}       ((all_die_are_bad=false) ) then
      {if true, user quit immediately or entire wafer was bad=>no files created}
          erase_temporary_files;  {this clears out all of the temporary files}
{JRP}   if TRACE34=true then writeln(userout2,'1st_time_thru_program=',
{JRP}       first_time_thru_program,' all_die_are_bad=',all_die_are_bad,
{JRP}       ' mode=',mode);
        writeln(#12);  {clears the display}
        iouninitialize;  {uninitializes all of the I/O interfaces}
 end.
```