Copyright © 1981, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

SIMULATION OF DEVELOPED RESIST PROFILES

FOR ELECTRON-BEAM LITHOGRAPHY

by

Michael G. Rosenfield

Memorandum No. UCB/ERL M81/40

12 June 1981

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

SIMULATION OF DEVELOPED RESIST PROFILES FOR ELECTRON-BEAM LITHOGRAPHY

Michael G. Rosenfield

Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley, California, 94720

ABSTRACT

A program for the simulation of the time evolution of two dimensional electron-beam exposed resist profiles is presented. The implementation of an electron-beam machine in the user oriented computer program for the Simulation and Modeling of Profiles in Lithography and Etching (SAMPLE) is discussed. Complete software documentation as well as a study of new electron-beam writing strategies and resist development effects are included.

Acknowledgements

I would first like to thank my research advisor, Professor Andrew R. Neureuther, for first suggesting the interface of an electron-beam simulation program with SAMPLE. His guidance, support, and patience over the last two years were of immeasurable help and made even the difficult times seem easy. I would also like to thank Professor William G. Oldham for his support, advice, and for taking the time to talk with me that day in the Spring of 1979 before I came to Berkeley.

Special thanks also must go to my colleagues in the SAMPLE group - especially to Sharad N. Nandgaonkar for the countless times I asked him for advice and help. Thanks are also to be extended to John L. Reynolds and Dr. Michael M. O'Toole for their suggestions. I would also like to thank Dr. Y. C. Lin for his advice. I am grateful to Dr. Ilesanmi Adesida and Dr. David F. Kyser for their Monte Carlo calculations which make electron-beam simulation possible. I would also like to thank Dr. Chiu H. Ting for his help in the preparation of the publication in Appendix II. The expert drafting by Thomas King is also greatly appreciated.

Finally, I want to thank my family and friends (especially those 'back east') for their support and their faith in me.

Research sponsored by the National Science Foundation Grant Eng77-14660 and an IBM Graduate Fellowship.

1. Introduction

As critical feature sizes in integrated circuits decrease, the importance of electron-beam lithography (EBL) as an advanced processing technique has increased. Due to limitations in optical lithographic processes, electron-beam machines are currently being used in the production of experimental devices and circuits with micron and sub-micron minimum dimensions [1-4] as well as in the manufacture of masks for optical and X-ray lithographic processes [5-6]. The simulation of the time evolution of two dimensional electron-beam exposed resist profiles is a very useful tool in the understanding and exploration of electron-beam lithography for such applications.

This report deals with the implementation of an electron-beam machine in the user oriented computer program for the Simulation and Modeling of Profiles in Lithography and Etching (SAMPLE) developed at the University of California, Berkeley [7-10]. As an example of the insight which can be gained using simulation, new electron-beam writing strategies and resist development effects are also discussed in an appendix.

Section II deals with the EBL field in general and the important models used in the simulation. Section III is a detailed explanation of the algorithms and program code. Appendix I is a User's Guide for the SAMPLE e-beam machine. Appendix II is the manuscript of a paper in which the e-beam machine is applied to a comparison of various electron-beam exposure strategies as well as an investigation into the actual resist development model used in the simulation. Appendix III contains Common Block documentation and Appendix IV contains the program code.

2. Electron-Beam Lithography and Simulation Models

In order to achieve high resolution, EBL utilizes high energy (10-50 Kev) electron-beam exposure of polymeric resists in place of more conventional wavelength limited UV and deep UV optical techniques. The loss of energy by the electrons being scattered can affect the resist material in one of two ways: Generally, in negative resist, polymers are crosslinked making the resist insoluble in areas exposed to the beam. In positive resist, such as polymethylmethacrylate (PMMA), various mechanisms, such as depolymerization, make the resist soluble in areas exposed to the beam. Thus, the rate of development of a resist is a function of the energy absorbed by the resist [11].

In contrast to optical lithography, where an entire wafer or chip is exposed at once through a mask of the desired exposure pattern, e-beam systems must perform the desired exposure by serial data flow, i.e. writing out the complete pattern on the resist coated wafer. The two techniques for writing the pattern are vector scan and raster scan [5,6]. In raster scan, the beam or wafer holding table is moved through a regular pattern with the beam being blanked off where no exposure is desired. In vector scan, the beam is only positioned in areas where exposure is required.

The two most widely used types of beam used in e-beam systems are the Gaussian round beam and rectangular shaped beam [12-14]. The advantage of rectangular shaped beams is increased throughput as a larger part of the desired pattern is exposed at once [12].

In order to simulate the e-beam exposure and development processes, the absorbed energy density in the resist for a given exposure pattern must be known. Of the various methods by which the absorbed energy density can be determined [15-17], data from Monte Carlo (M/C) calculations [17,18] is used in the SAMPLE e-beam program. The M/C technique is based on simulating a large number of individual electron trajectories to calculate the spatial distribution of energy deposited in the resist [19]. A single scattering approach [17] is used where electrons from an idealized delta function electron-beam are elastically scattered by the atomic nuclei in the target (resist film and substrate material for example) with the angle of scattering being chosen in a random manner. In between these elastic scattering events, the electrons travel a distance of one mean free path and are assumed to lose energy continuously (i.e. linearly with path distance) due to inelastic scattering. This inelastic scattering is assumed to have negligible effect on the trajectory of the electron [20]. This is known as the continuous slowing down approximation (CSDA). It should be noted that although there are other M/C methods

which take into account other more exotic inelastic scattering events, such as core electron and plasmon excitation [18,21], the CSDA approach has been found to give adequate results for type of data used in simulating EBL [18].

The M/C data used in the SAMPLE e-beam program is a two dimensional array of the energy absorbed in the resist by a delta function line source. For a line source, the M/C computer program calculates the absorbed energy density at points in a plane in the resist perpendicular to the infinite line source. The plane is made up of two dimensional cells and the energy absorbed in each cell is output as the two dimensional array used in the SAMPLE e-beam program [17]. This data is valid only for the resist material and underlying substrate(s) used in the calculations. Different resist-substrate combinations yield different M/C energy absorption arrays.

From the M/C model of electron-beam exposure of resist, it can be seen that the scattering of the electrons in the exposure of one area of resist can affect other areas. Electrons can be backscattered from the resist and substrate and rexpose the same area as well as nearby areas. Thus, small isolated areas develop out slower than larger areas. This is the well known proximity effect [22] and is the fundamental limitation of the resolution of EBL since in close packed geometries, the exposure of one area inherently exposes nearby areas. It can be seen that M/C data, and thus the SAMPLE e-beam program, implicitly simulates the proximity effect since the output array includes the energy density absorbed 2.5 to 5 um away from the incident delta function line source [17,18].

Superposition is assumed to be valid. The absorbed energy density in the resist due to a delta function line source (i.e. the M/C data) can be convolved with the actual electron-beam exposure pattern to yield a two dimensional array which describes the absorbed energy density in a cross-section of resist. Knowing that the rate of development is a function of the absorbed energy density, the time evolution of two dimensional resist profiles can then be generated.

Early models assumed that development occured along equi-energy contours in the resist [17,39]. However, this assumption was found to be inadaquate [11,17,39,40] and the present model assumes that development is a surface etching phenomenon with the local etch rate determined by the energy absorbed in the resist being removed [19].

Using the present model, the resist can be developed using various etching algorithms. Both a cell removal model [23,24] and the string development model [23] are currently used. In the cell model, the resist is broken up into two

dimensional cells similar to those in the M/C calculations. The rate of dissolution of the resist in each cell that is in contact with the developer is determined by an etch rate previously assigned to that cell. The rate of dissolution of a cell depends on the etch rate as well as the number of sides exposed to the developer and whether or not these sides are adjacent [23].

The SAMPLE program uses the string model of development in which a string of straight line segments follows the developer-resist interface as a function of time [23]. During each development time step, each node (i.e. intersection of string segments) on the string is advanced at the local etch rate (which is a function of the absorbed energy density at that point) in the direction of the perpendicular bisector of the angle between adjacent segments. Segments are periodically added and deleted on the basis of length, and loops are deleted just prior to profile output times [19]. A more thorough explanation of the algorithms involved can be found in works by Jewett and O'Toole [25,8].

Both development models have given adequate agreement with experimental profiles [19,26-28]. The usefulness of simulation of EBL has been shown in a number of studies investigating proximity effects, resist materials, and writing strategies [19,28-34]. An example is contained in Appendix II.

At present, all simulation studies have been done with the more widely used positive resists as an adequate model has not been developed for the development of negative resist. Most work to date, including this report, has been in the simulation of two dimensional cross-sections of resist profiles. Recently, Jones et al have reported a computer program which simulates resist development for EBL in three dimensions [35].

3. Program Operation and Explanation of Code

The SAMPLE e-beam computer program consists of the following blocks of subroutines, written in standard ANSI Fortran 77 (Appendix 4), which inputs user defined data, acts as a controller, computes the absorbed energy density array for the input exposure pattern, develops the resist, and prints out useful information and messages.

INPUT

Subroutines EXTRA6, TRL101, TRL102, TRL104, TRL105, TRL108, TRL110, TRL111, TRL112, TRL113, TRL114, and TRL115 are used to initialize default parameters (TRL101) and input user defined data which sets the exposure and development conditions needed for simulation. The use of 'Trial' statements or 'Key-words' to input data to the program by way of these subroutines is discussed in Appendix 1.

CONTROLLER

Subroutine EBCTRL(numb) is the controlling subroutine which inputs the needed M/C data, initializes certain variables and directs other subroutines to form the exposure pattern, compute the absorbed energy density array, develop the exposed resist, and print out information.

CONVOLVER

Subroutines MLTSPT, EGAUSS, SQWGT, SPWGT, WEIGHT, MLTLIN(numb), BOUNDR, EARRAY, and function ERF(y) compute the overall exposure pattern and then convolve the M/C data under the pattern to give the final absorbed energy density array.

E-BEAM DEVELOPER

Subroutines EBDEV, ECYCLE, and function EBRATE(cz) along with subroutines LINEAR, CHKR, BNDARY, DELOOP(itype), PLOTOUT, PLOTHP(ioutpt), and PRTPTS(iouptp) from the SAMPLE program [8] develop the resist by means of the string algorithm and print out the resultant resist profiles.

MESSAGES AND INFORMATION

Subroutines EBMSG(numb), PRARRY(numb), and DEVMSG(numb) (which is from the SAMPLE program [8]) print out pertinent information and error messages for the user.

Common block documentation can be found in Appendix III.

An explanation of the code and algorithms used is as follows: Once default values are initialized, and user defined data for the exposure pattern is input by way of 'Trial' statements or 'Key-words' (Appendix I), subroutine EBCTRL(1) is called from subroutine TRL111. Information is read in from the file 'mcdat' and the M/C data is stored in array EMAT(80,500). Due to symmetry, all that is needed is the data from one side of the delta function line source (Figure 1). The M/C data is then multipied by the overall exposure dose and a constant to convert the data to units of J/cm**3. Variables are then initialized including the maximum number of array points which can be used in the arrays EMLT(1499) and ELIN(82,1002). This effectively limits the maximum distance between the first and last Gaussian shaped beams (spots) in a single line to 5 um and limits the 'window' of resist which can be developed to 10 or 20 um depending on the M/C cell size in the horizontal (x) direction.

The SAMPLE e-beam program is capable of simulating Gaussian shaped or rectangular electron-beams (Figure 2) [27]. If an exposure pattern is to be written with rectangular beams (each beam is considered one exposure line), subroutine SQWGT is called. SQWGT calculates the following expression which describes the rectangular beam profile [27]:

$$f(x) = \frac{\mathrm{erf}[(a-x)/SIGMA*SQ2]+\mathrm{erf}[(a+x)/SIGMA*SQ2]}{\sum_{x=-\infty}^{\infty} \mathrm{erf}[(a-x)/SIGMA*SQ2]+\mathrm{erf}[(a+x)/SIGMA*SQ2}$$

where A=(1/2)FWHM (full width half maximum), x is the distance from the center of the beam, SIGMA=edge width/$\sqrt{2\pi}$, and SQ2= $\sqrt{2}$. Each normalized ($\sum_{-\infty}^{\infty} f(x) = 1$) value of f(x) corresponds to a location in the array ETEM2(999) with the distance between f(x) values set equal to the size of the M/C cell in the x direction (x=0 corresponds to ETEM2(nrhcet+1=ncemat).

It was found that when x equals 2a (i.e. fwhm), f(x) is small (~0) and can be neglected. Thus, f(x) values need only be computed in this range. A warning is printed out if the fwhm of the beam is larger than the horizontal range of the M/C data. Values for the error function are found using an anlytical expression [36] in function ERF which calculates:

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

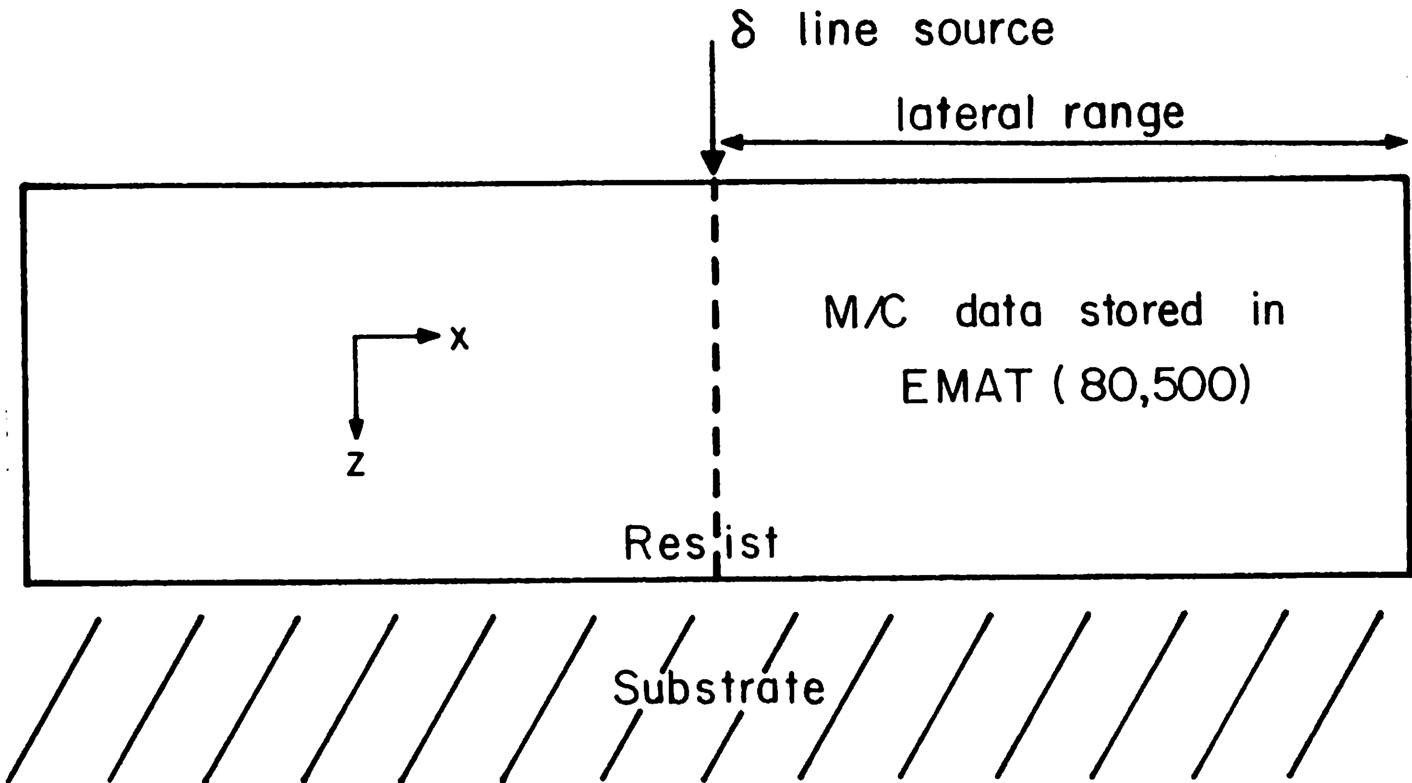
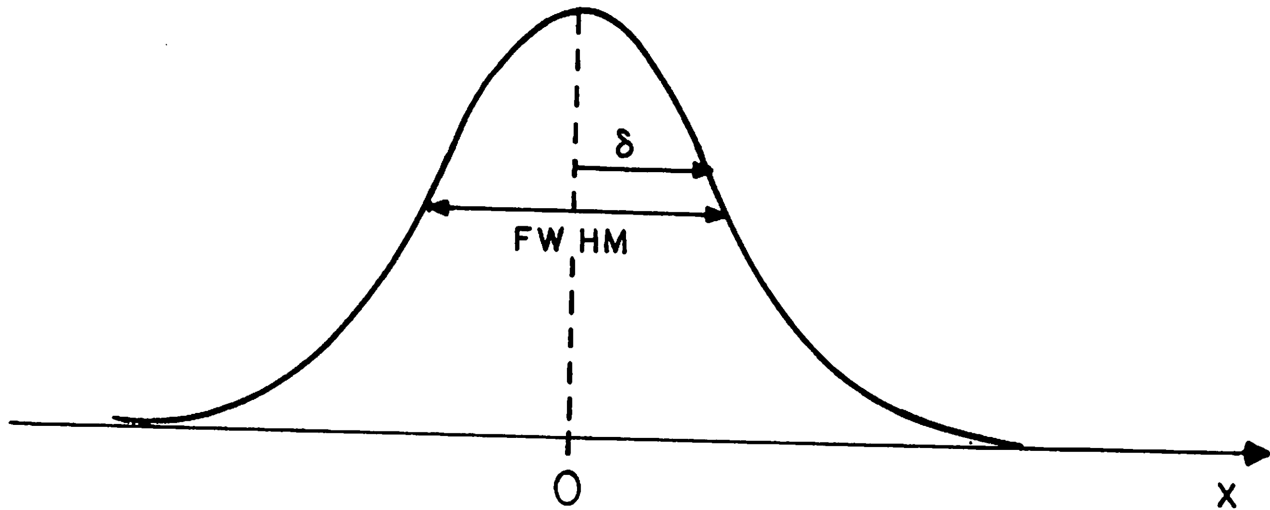


Figure 1. Symmetry of the Monte Carlo data. The program uses the data in the lateral range.



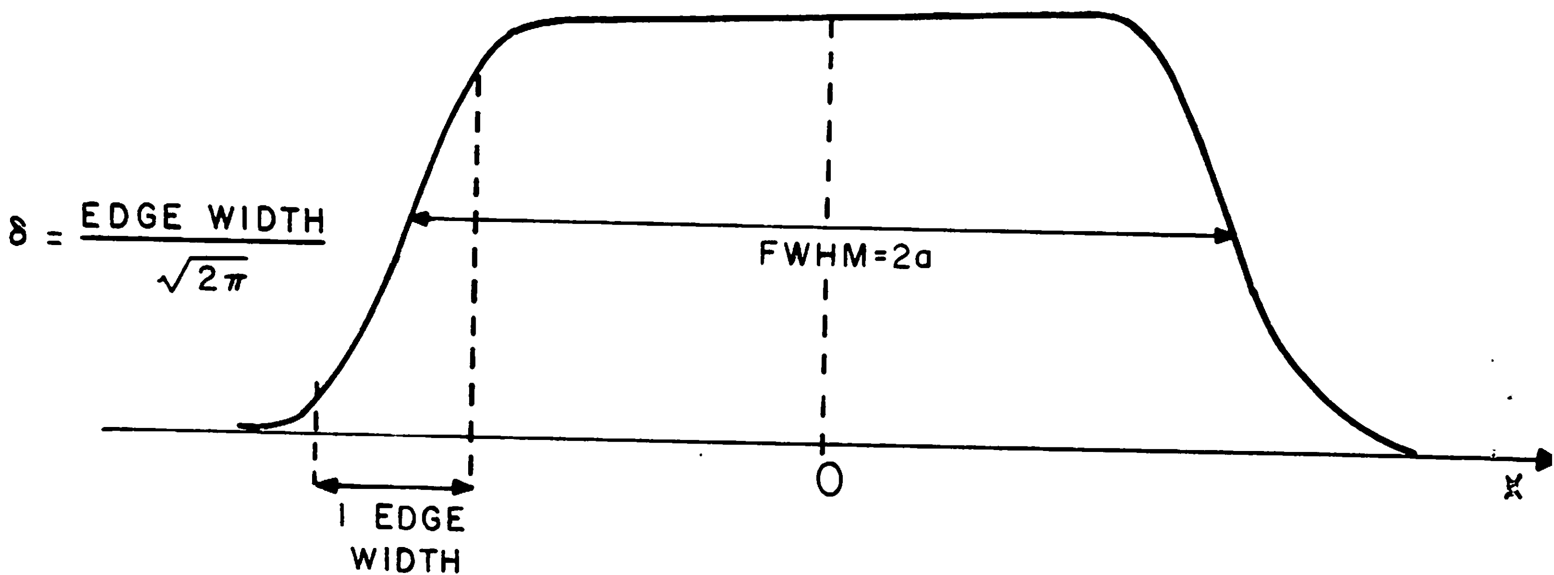


Figure 2. Simulated electron-beam shapes.

It should be noted that an overflow will result in function ERF if the argument given to it is approximately greater than 12. In fact if the argument is greater than or equal to approximately 4, ERF will return a value of 1. The right hand side, including the middle, of the profile is stored in ETEM2(999). Note that these 'weights' which describe the beam are also multiplied by the fwhm in order to assure that different sized beams with the same exposure dose develop out at approximately the same rate [37].

Subroutine EGAUSS is then called. In EGAUSS, the entire rectangular beam profile is entered into ETEM2(999) by entering right hand side profile points in the corresponding opposite left hand side ETEM2(999) positions. This information is then transferred to array EMLT(1499) which contains the profile for an exposure line.

If the exposure pattern is to be written with lines composed of one or more Gaussian shaped beams, the rectangular beam algorithim in SQWGT is skipped. The standard deviation of the present spot being calculated is stored in DEVTEM while that of the previous spot is stored in DEVUNI. Similarly, the 'weight' (see Appendix I) is first stored in WGHTO until it is compared with the weight of the previous spot, WGHT1. Subroutine SPWGT is called to calculate the profile for the Gaussian shaped beam. The beam is described by the normalized Gaussian:

$$f(x) = \frac{e^{-x^2/2*SIGMA^2}}{\sqrt{2\pi} * SIGMA}$$

where SIGMA is the standard deviation. The total area under the beam is 1 and thus, each point in the beam profile for convolution can be described as a fraction of the total area under the beam (Figure 3). The distance between each point describing the beam must be equal to the M/C cell size in the x direction. The area under a Gaussian is:

$$\frac{1}{SIGMA} * \sqrt{2\pi} \int_{x_1}^{x_2} e^{-x^2/(2*SIGMA^2)} dx = erf(\frac{x_2}{SIGMA}) - erf(\frac{x_1}{SIGMA})$$

where

$$erf(x) = \frac{1}{\sqrt{2\pi}} \int_0^x e^{-y^2/2} dy$$

and by a change of variables, the error function can be rewritten:

$$erf(x) = \frac{1}{\sqrt{\pi}} \int_0^{x/\sqrt{2}} e^{-t^2} dt$$

which is just 1/2 of the value returned from function ERF. Therefore, the profile for the Gaussian spot is calculated in SPWGT as:

$$F(x') = \frac{1}{2} [\mathrm{erf}(\frac{x'+CELLX/2}{DEVTEM*SQ2})-\mathrm{erf}(\frac{x'-CELLX/2}{DEVTEM*SQ2})] * WGHT1$$

where CELLX is the horizontal M/C cell size, DEVTEM is the standard deviation, SQ2= $\sqrt{2}$, and WGHT1 is the user defined weight for the spot. The middle of the spot (i.e. x=0) is located at ETEM2(ncemat). As with the rectangular beam, only the right hand side and middle of the Gaussian profile is calculated. A warning is printed out if the last array point of ETEM2(999) is not less than .0001, indicating that the standard deviation of the Gaussian is too large.

Upon returning from SPWGT, subroutine EGAUSS is called and the left hand side of the profile is calculated in the same manner as for a rectangular beam shape. The first spot profile in an exposure line is then entered into EMLT(1499).

If there is more than one spot per line, loop 408 is entered. The input distance between the present spot and the first spot (located at x=0.0) is changed from um to units where 1 unit is equal to 1 M/C cell size in the x direction. This distance, ISHIFT, is rounded off to the nearest cell. A check is made as to whether the distance between the present spot and x=0.0 requires more array space than is available in EMLT(1499). If so, an error message is printed and the program is stopped. The largest distance shifted is stored in ISLAR so that the actual number of array locations used in EMLT(1499) can be determined later in the program. If the standard deviation of the present spot is the same as the preceding, there is obviously no need to go through subroutines SPWGHT and EGAUSS again. If only the user defined 'weights' are different, subroutine WEIGHT is called and ETEM2(999) is simply multiplied by WGHTO/WGHT1 to calculate the profile. If the standard deviations are not equal, then subroutines SPWGT and EGAUSS are called and the new spot profile is stored in ETEM2(999). Subroutine MLTSPT then simply adds the new spot pattern to the previous pattern in EMLT(1499) with the new spot being offset from the first spot by the ISHIFT distance. The entire loop is then completed and the next spot, if any, is calculated and added to the line. The overall process is illustrated in Figure 4.

Once the profile for the exposure lines are calculated, the exposure profile in the user defined resist 'window of interest' is computed. This profile can be due to just one or up to twenty lines. The first line in an exposure pattern starts at x=0.0 (i.e. the same location as the middle

of the rectangular beam profile or the first Gaussian profile in a multiple spot line) and all other lines, if any, are referred to this point. First, all line shifts are rounded to the nearest M/C cell and then restored in the array DISLIN(20). NUMCOL is the number of array locations of EMLT(1499) which are used to contain the line pattern. If a symmetrical development (see Appendix I) is requested by the user, one-half of the total number of array points in the pattern, ITEMP, is computed. The variable SHIFT is the distance in um of the first spot or rectangular beam in the pattern from the left hand side resist window edge.

CPEDGE, used in the SAMPLE plotting routines, is set to 1/2 the size of the window (CPWIND) in um. ICPWIN is the number of columns in the final absorbed energy density array ELIN(82,1002) needed to develop the resist in the window. If this is too large, an error message is printed out and the program is stopped. NCLELU is the total number of columns in ELIN(82,1002), including boundaries, which are required. The distance SHIFT is then set to either the symmetrical development SHIFT or a user defined SHIFT plus the distance in um of one cell size less than the range of the M/C data in x (float(NRHCET)* CELLX-CELLX). This is because the total exposure pattern must be computed to + or - this extra distance from the window edges so that when the pattern is convolved, all possible contributions from outside the window of interest are added in the window. Thus, when calculating the exposure profile, the actual window size in array locations is (CPWIND/CELLX)+NCLMET (the number of array locations used in the ETEM2(999) array). Therefore the 'window' for calculating the exposure profile, hereafter referred to as the 'calculation window', is NCLMET larger than the ELIN(82,1002) window size as shown in Figure 5.

Subroutine MLTLIN(numb) calculates the input exposure pattern. If the first spot or rectangular beam in the pattern is to the right of the left calculation window edge, MLTLIN(1) is called. MAX is the number of locations in array ELNWGT(1999) which make up the calculation window. IPASS, DIPASS, and TSHIF are variables used in conjunction with MLTLIN(2). Loop 9 runs through each exposure line (starting with the first if MLTLIN(2) has not been called). ISTAR and ITEM define the array locations in EMLT(1499) which will be added to a corresponding location in ELNWGT(1999). If a line is so far to the right of the calculation window that there is no possibility of contributing exposure in the window, a warning is printed out to the user indicating that line and all lines which follow do not contribute (A return is then made to EBCTRL(1)).

Loop 20 adds the contribution for each line in the calculation window. It starts at the array location corresponding to the left hand edge of the window, ELNWGT(1). A check is first made on ITEM. If ITEM is

negative, this means that the first contribution to the calculation window will be to the right of the present ELNWGT(j) location and both the ITEM and ELNWGT(j) locations are then increased by 1. If ITEM becomes greater than the number of array locations used in EMLT(1499), there is no further contribution from that line and the next line is calculated. Otherwise, the contribution from EMLT(item) multiplied by the weight of that line, WGTLIN(k) (see Appendix I), is added to ELNWGT(j).

MLTLIN(2) is called when the first spot or rectangular beam is to the left of the left calculation window edge. ITEST and ITTEST correspond to variables ISTAR and ISTEM in MLTLIN(1). If a line is too far to the left of the calculation window to contribute any exposure, a warning is printed out and the next line is calculated. If the beginning of a line (i.e. the location of the first spot or middle of a rectangular beam) lies to the right of the left hand calculation window edge, that line (specified by IPASS) and all others following it are added using MLTLIN(1). TSHIF contains the new value for the shift, i.e. the distance from the left hand window edge, and DIPASS is used so that MLTLIN(1) sees the correct distance between lines (DISLIN(k) contains the distance between line 'k' and the first exposure line. DISLIN(k)-DIPASS is the distance between line 'k' and the first exposure line to the right of the left calculation window edge). Otherwise, loop 120 adds the contribution to ELNWGT(j) in a manner similar to that of MLTLIN(1). If ITTEST becomes greater than the number of array locations used in EMLT(1499), calculations for that line are completed and the next line is calculated.

The result of calling subroutine MLTLIN(numb) is a profile describing the exposure pattern as shown in Figure 5. The M/C data for a delta function line source can then be convolved under that profile to give the absorbed energy density in the resist window of interest.

Subroutine EARRAY is then called from EBCTRL(1) to perform the convolution. MAXCOL is the number of columns, excluding boundary columns, which are used in ELIN(82,1002) to describe the absorbed energy density in the window of interest. Each row of M/C data and therefore each row of ELIN(82,1002) corresponds to a depth in the resist with the distance between 'depths' equal to the M/C cell size in the z or vertical direction. Also, each row of the M/C data has values for the absorbed energy density which are symmetrical about the delta function line source as explained earlier and shown in Figure 6. The exposure pattern is convolved by constructing the exposure profile out of delta functions seperated by a distance equal to 1 M/C cell size in the x direction as illustrated in Figure 7. Each row of ELIN(82,1002) is computed by adding the contribution of absorbed energy density from each of the delta functions in

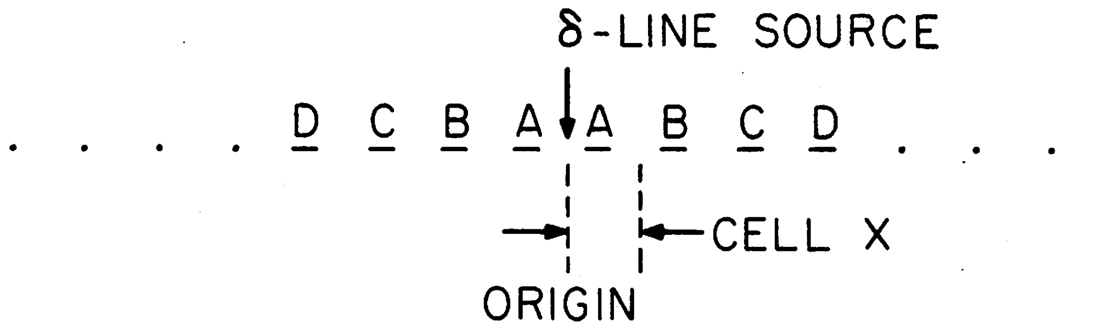


Figure 6. Illustration of the symmetry of the M/C data. 'A', 'B', etc. are values of absorbed energy density in a cell.

the window of interest. The convolution is complete when the procedure has been carried out for each row of the M/C data.

Contributions from the left and left edge of the window are first calculated. IMARK is the first array location of ELNWGT(1999) which contains a non-zero value. If no non-zero values are found, this section of EARRAY is skipped. Assuming a non-zero value is located, loops 99, 199, and 299 add the contribution from the delta functions to the left of the window, including the edge, in the window with each row being done seperately (loop 99). Loop 199 sets the correct location of the delta function (i.e the ELNWGT(1999) array location) and the corresponding location, ITEM, in the M/C array EMAT(999) at which loop 299 starts. Loop 299 performs the actual convolution addition by adding to each ELIN(i+1,l+1) column, starting at l=1, the aborbed energy density due to a delta function located at ELNWGT(j). The '+1' in the subscripts of the ELIN(82,1002) array allow for the addition of boundary rows and columns later in the program. If the contribution from delta function, j, goes past the right window edge, j is incremented and the next delta function is calculated.

Contributions from the right and right hand edge of the window are then calculated. Again, IMARK is the first location in ELNWGT(1999), starting from the right, which contains the first non-zero value. As before, if there are no contributions, this section of EARRAY is skipped. ITCOL is equal to the number of array locations used in ELNWGT(1999) plus one. Loops 399, 499, and 599 add the contributions from the right of the window in the window. Again, each row is done one at a time (loop 399). Loop 499 sets the parameters which determine the correct location of the delta function, i.e. ELNWGT(1999) position (J), and the corresponding location in the M/C array EMAT(80,500), ITEM, at which loop 599 starts. Loop 599 performs the convolution starting with the right-most delta function located at ELNWGT(itcol-imark). The contributions to ELIN(i+1,l+1), starting at column MAXCOL+1 and working left, from each delta function at ELNWGT(itcol-j) are calculated. The next delta function is calculated when the distance from the delta function to the ELIN(82,1002) location is greater than the range of the M/C data (NCEMAT) or when the contribution goes past the left hand window edge.

The convolution is then performed in the window of interest. This differs from the previous two cases in that contributions from both sides of the delta functions must be included. ISTART is the left-most ELNWGT(1999) location inside the window and IEND is the right-most location inside the window. Loops 201 and 202 control the convolution with each row again being done one at a time (loop 201). For each delta function at location ELNWGT(j), contributions

from the left and then the right are added in the window. A location is skipped if ELNWGT(j)=0.

ILFCON is the number of ELIN(82,1002) columns to the left of the delta function. In loop 204, the contributions from the left of a delta function at ELNWGT(j) are added to ELIN(i+1,k+1) starting with ELIN(i+1,2). IX is the corresponding location in EMAT(80,500). If IX is greater than the M/C range, it is decreased while the ELIN(82,1002) column is increased until it is in range.

Loop 205 adds the contribution to the right of the delta function located at ELNWGT(j). IX is again the correct EMAT(80,500) location to use. ISCON is the first column in ELIN(82,1002) to the right of the delta function. Contributions are added in the window starting at ELIN(i+1,iscon) and continuing until past the right window edge or the range of the M/C data.

Thus, at the end of EBCTRL(1), ELIN(82,1002) contains the absorbed energy density, due to an electron-beam exposure pattern, in a user defined window of resist. The resist is developed when EBCTRL(2) is called from subroutine TRL114.

Subroutine BOUNDR, called from EBCTRL(2), simply adds boundary rows and columns to the previously calculated ELIN(82,1002) array. The code is similar to that used in subroutine CLCMXZ in SAMPLE [8]. Boundary values are found by linear extrapolation with extrapolated values less than 0.0 set to 0.0. The boundaries are required in the development routines. Upon return from BOUNDR, parameters needed in the SAMPLE development algorithms are initialized.

The develop subroutines listed in Appendix IV are very similar to the SAMPLE optical develop routines explained in Mike O'Toole's Ph.D. thesis [8] and therefore only a brief explanation will be given. The development controller, subroutine EBDEV, is identical to subroutine DVELOP in SAMPLE [8] except that when estimating the time until developer breakthrough to the substrate, the maximum value of absorbed energy density in the top row (not including boundaries) of ELIN(82,1002) is found. This is because the rate of development increases with increasing energy density. There is also a slight difference when calculating the variable MAXPTS (i.e. a divison by the window size in um).

Function EBRATE(cz) is the same as function RATE(cz) [8] except that the rate equation (and background rate, BACRAT) applicable to EBL is used [19]:

$$R(D) = R1\left(c_m + \frac{D}{D_o}\right)^{ALPH}$$

(See Appendix I).

Subroutine ECYCLE is the same as subroutine CYCLE [8] except that an anisotropic development feature [28, see Appendix II] is incorporated. The user may input the fraction of horizontal development desired, FRAC, with the default value being, of course, 1.0. The string point XZ(m) is then advanced the normal length in the z direction; but, only FRAC of the normal length in the x direction.

The rest of the development subroutines come from the SAMPLE program [8] as explained earlier.

Subroutine EBMSG(numb) prints out various types of information and messages if called from other subroutines in the SAMPLE e-beam program. Subroutine PRARRY(numb) prints out the various arrays used in the program if instructed to do so by the user (See Appendix 1). Subroutine EPLOT prints out desired rows of ELIN(82,1002) into a file 'engpts' which can be plotted to give energy density distribution profiles for various depths in the resist (See Appendix I). The remainder of the code (the INPUT subroutines listed earlier) are the 'Trial' subroutines used to input exposure and development conditions to the program.

References

[1] J.R. Toker, M.R. Oliver, A.P. Lane, and R.H. Havemann, "Fabrication and characterization of e-beam defined mosfets with submicrometer gate lengths", 1980 IEDM Technical Digest, pp. 768-771, December 1980.

[2] R.K. Watts, W. Fichtner, E.N. Fuls, R.L. Johnston, and L.R. Thibault, "Electron beam lithography for small mosfets", 1980 IEDM Technical Digest, pp. 772-775.

[3] Y. Sakakibara, T. Ogawa, K. Komatsu, S. Moniya, M. Kobayashi, and T. Kobayashi, "Electron-beam direct writing technology for 1um VLSI fabrication", 1980 IEDM Technical Digest, pp. 425-428.

[4] S.A. Evans, S.A. Morris, L.A. Arledge, Jr., J.O. Englade, and C.R. Fuller, "A 1-um bipolar VLSI technology", IEEE Trans. Electron Devices, vol. ED-27, pp. 1373-1378, 1980.

[5] J.A. Dohertry, "Recent advances in electron beam systems for mask making", Solid State Tech., vol. 22, no. 5, pp. 83-88, 1979.

[6] J.A. Reynolds, "An overview of e-beam mask-making", Solid State Tech., vol. 22, no. 8, pp. 87-94, 1979.

[7] S.N. Nandgaonkar, "Design of a simulator program (SAMPLE) for IC fabrication", M.S. Report, Department of Electrical Engineering and Computer Sciences, U.C., Berkeley, 1978.

[8] M.M. O'Toole, "Simulation of optically formed images in positive photoresist", Ph.D. Thesis, Department of Electrical Engineering and Computer Sciences, U.C., Berkeley, 1979.

[9] W.G. Oldham, S.N. Nandgaonkar, A.R. Neureuther, and M.M. O'Toole, "A general simulator for VLSI lithography and etching processes: Part I -- application to projection lithography", IEEE Trans. Electron Devices, vol. ED-26, no. 4, pp. 717-722, 1979.

[10] W.G. Oldham, A.R. Neureuther, C. Sung, J.L. Reynolds, and S.N. Nandgaonkar, "A general simulator for VLSI lithography and etching processes: Part II -- application to deposition and etching", IEEE Trans. Electron Devices, vol. ED-27, no. 8, pp. 1455-1459, 1980.

[11] M. Hatzakis, C.H. Ting, and N. Viswanathan, "Fundamental aspects of electron beam exposure of polymeric resist system", Proc. 6th Int. Conf. on Electron and Ion Beam Sci. and Tech. (Electrochemical Society), pp.

542-579, 1974.

[12] H.C. Pfeiffer, "Recent advances in electron-beam lithography for the high-volume production of VLSI devices", IEEE Trans. Electron Devices, vol. ED-26, no. 4, pp. 663-674, 1979.

[13] E.V. Weber and R.D. Moore, "E-beam exposure for semiconductor device lithography", Solid State Tech., vol. 22, no. 5, pp. 61-67, 1979.

[14] H.C. Pfieffer and G.O. Langner, "Advanced beam shaping techniques for electron lithography", Proc. 8th Int. Conf. on Electron and Ion Beam Sci. and Tech. (Electrochemical Society), pp. 149-159, 1978.

[15] J.S. Greeneich and T. Van Duzer, "An exposure model for electron sensitive resist", IEEE Trans. Electron Devices, vol. ED-21, no. 5, pp. 286-299, 1974.

[16] R.J. Hawryluk, A.M. Hawryluk, and H.I. Smith, "Energy dissipation in a thin polymer film by electron beam scattering", J. Appl. Phys., vol. 45, no. 6, pp. 2551-2566, 1974.

[17] D.F. Kyser and K. Murata, "Monte Carlo simulation of electron beam scattering and energy loss in thin films on thick substrates", Proc. 6th Int. Conf. on Electron and Ion Beam Sci. and Tech. (Electrochemical Society), pp. 205-225, 1974.

[18] I. Adesida, "Electron energy dissipation in layered media", Ph.D. Thesis, Department of Electrical Engineering and Computer Sciences, U.C., Berkeley, 1979.

[19] A.R. Neureuther, D.F. Kyser, and C.H. Ting, "Electron-beam resist edge profile simulation", IEEE Trans. Electron Devices, vol. ED-26, no. 4, pp. 686-693, 1979.

[20] D.F. Kyser, "Monte Carlo Simulation in Analytical Electron Microscopy", in 'Introduction to Analytical Electron Microscopy', J.I. Goldstein, J. Hren, D.C. Joy, ed., Plenum Press, 1979, chap. 6.

[21] R. Shimizu, Y. Kataoka, T. Ikuta, T. Koshikawa, and H. Hashimoto, "A Monte Carlo approach to the direct simulation of electron penetration in solids", J. Phys. D: Appl. Phys., vol 9, pp. 101-114, 1976.

[22] T.H.P. Chang, "Proximity effect in electron-beam lithography", J. Vac. Sci. Tech., vol. 12, no. 6, pp. 1271-1275, 1975.

[23] R.E. Jewett, P.I. Hagouel, A.R. Neureuther, and T. Van Duzer, "Line-profile resist development simulation techniques", Polymer Eng. Sci., vol. 14, no. 6, pp. 381-384, 1977.

[24] F.H. Dill, A.R. Neureuther, T.A. Tuttle, and E.J. Walker, "Modeling projection printing of positive photoresist", IEEE Trans. Electron Devices, vol. ED-22, no. 7, pp. 456-464, 1975.

[25] R.E. Jewett, "A string model etching algorithm", M.S. Report, Department of Electrical Engineering and Computer Sciences, U.C., Berkeley, 1979.

[26] K. Murata, E. Nomura, K. Nagami, T. Kato, and H. Nakata, "Experimental and theoretical study of cross-sectional profiles of resist patterns in electron-beam lithography", J. Vac. Sci. Tech., vol. 16, no. 6, pp. 1734-1738, 1979.

[27] D.F. Kyser and R. Pyle, "Computer simulation of electron-beam resist profiles", IBM J. Res. Develop., vol. 24, no. 4, pp. 426-437, 1980.

[28] M.G. Rosenfield and A.R. Neureuther, "Exploration of electron-beam writing strategies and resist development effects", Proc. 9th Int. Conf. on Electron and Ion Beam Sci. and Tech. (Electrochemical Society), pp. 382-395, 1980.

[29] D.F. Kyser and C.H. Ting, "Voltage dependence of proximity effects in electron beam lithography", J. Vac. Sci. Tech., vol. 16, no. 6, pp. 1759-1763, 1979.

[30] J.S. Greeneich, "Impact of electron scattering on linewidth control in electron beam lithography", J. Vac. Sci. Tech., vol. 16, no. 6, pp. 1749-1753, 1979.

[31] J.S. Greeneich, "Proximity effects in electron-beam exposure of multi-layer resists, Proc. 9th Int. Conf. on Electron and Ion Beam Sci. and Tech. (Electrochemical Society), pp. 282-303, 1980.

[32] Y. Todokoro, "Double-layer resist films for submicrometer electron-beam lithography", IEEE Trans. Electron Devices, vol. ED-27, no. 8, pp. 1443-1448, 1980.

[33] M. Nakase and M. Yoshimi, "Quantitative evaluation of proximity effect in raster-scan exposure system for electron-beam lithography", IEEE Trans. Electron Devices, vol. ED-27, no. 8, pp. 1460-1465, 1980.

[34] M.P.C. Watts, P. Rissman, and J. Kahn, "Solubility ratio, sensitivity and line profile control in positive e-beam resists", Proc. 9th Int. Conf. on Electron and Ion Beam in Sci. Tech. (Electrochemical Society), pp. 375-381, 1980.

[35] F. Jones, J. Paraszczak, and M. Hatzakis, "A three dimensional model for resist development simulation", Proc. 9th Int. Conf. on Electron and Ion Beam Sci. and Tech. (Electrochemical Society), pp. 351-364, 1980.

[36] C. Hastings, Jr., 'Approximations for Digital Computers', Princeton, New Jersey: Princeton University Press, 1955, p. 187.

[37] D.F. Kyser, private communication.

[38] Papoulis, 'Probability, Random Variables, and Stochastic Processes', New York: McGraw Hill, 1965, pp. 65-66.

[39] J.C.H. Phang and H. Ahmed, "Line profiles in thick electron resist layers and proximity effect correction", J. Vac. Sci. Tech., vol. 16, no. 6, pp. 1754-1758, 1979.

[40] J.S. Greeneich, "Time evolution of developed contours in PMMA electron resist", J. Appl. Phys., vol. 45, pp. 5264-5268, 1974.

Appendix I -- USER'S GUIDE

SAMPLE E-BEAM USER'S GUIDE

Michael G. Rosenfield

Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley, California 94720

Version 1.1 February 9, 1981

Introduction

The SAMPLE (Simulation and Modeling of Profiles in Lithography and Etching) computer program, with the addition of an electron-beam machine, now has the capability to simulate the time evolution of two dimensional resist profiles for electron-beam lithography. Monte Carlo data, giving the spatial distribution of energy deposited in the resist by a delta-function line source, is convolved with a pattern of arrayed Gaussian (spot) or rectangular shaped electron-beams. This gives the energy density absorbed in a 'window' of resist for various patterns of exposure. The development is then simulated using a simple etch-rate versus dose curve and SAMPLE's string model of development. E-beam simulation can be implemented in SAMPLE through a series of 'trial' statements (as is done presently) or by their corresponding key-words for the general interface.

Overall Program Operation

Basically, the program is run by the user in the following manner: Default values are first initialized. An exposure line is then set. This line may be a single rectangular beam, a single Gaussian beam, or a group of arrayed Gaussians. This line can then be arrayed to form an exposure pattern consisting of one or more lines. The part of the exposed resist, relative to the exposed line pattern, which is to be investigated (i.e. the 'window of interest') is set by the user. The Monte Carlo data is then convolved with the exposure pattern in the window of interest at an overall exposure dose (uC/cm**2). The number of points in the development string, development times, and the constants for the development rate equation(the etch-rate as a function of absorbed energy density) are then inputted. The resist is then developed. The overall exposure dose can be changed without having to reconvolve the exposure pattern and another development can be run. Also, the various arrays used in the program as well as the energy absorbed at specified depths in the resist in the window of interest can be printed out if desired.

E-Beam Trial and Tenative Key-Word Summary

TRIAL 101 -- EBLITH

Default Parameters

Trial 101 (no arguments) initializes the default parameters and must be run first for correct initialization.

TRIAL 102 -- EBLPRINT

Output Printing Flags

Trial 102 arg2 arg3 arg4 arg5 arg6 sets flags which control the outputting of the various arrays used in the e-beam program as well as information pertaining to the exposure and development conditions. If arg2=1, the one-dimensional arrays EMLT(1499) (which contains the exposure pattern for a single line) and ELNWGT(1999) (which contains the exposure pattern needed to compute the absorbed energy in the window of interest) are outputted. If arg3=1 the final two dimensional energy density array, ELIN(82,1002), is printed out row by row (including boundaries). Note that ELIN(82,1002) will only be outputted if an actual development is requested. If arg4=1, the one dimensional array ETEM2(999) (which contains the exposure pattern for a single Gaussian or rectangular beam) is printed out. If arg5=1, the array EMAT(80,500) (which contains the inputted Monte Carlo data multiplied by the dose and in units of J/cm**3) is outputted. If arg6=1, information pertaining to the exposure and development conditions is printed out.

TRIAL 104 -- EBLRATE

Etch-Rate Parameters

Trial 104 arg2 arg3 arg4 arg5 sets the development rate equation constants. The rate equation used is [1]:

$$R(D) = R1(C_m + D/D_o)^{ALPH}$$

where R(D) is the etch-rate in A/sec, D is the absorbed energy density in J/cm**3, R1Cm is the background etch-rate, Cm is a constant inversely proportional to the initial number average molecular weight, DO is a reference or knee energy, and ALPH is the asymptotic slope of the etch-rate versus absorbed energy density curve at high dose. Arg2 changes the default value of R1. Arg3 changes the default

value of Cm. Arg4 changes the default value of DO and arg5 changes the default value of ALPH.

TRIAL 105 -- EBLPATSQ

Rectangular Beam

Trial 105 arg2 arg3 sets the full-width half maximum (fwhm) value and edge-width if a rectangular shaped beam is desired. Arg2 sets the fwhm and arg3 sets the edge-width (i.e. the lateral distance between the 10% and 90% points of the rectangular beam. An edge-width of 0.0 is not allowed. Note that one rectangular beam is considered one exposure line in the program.

TRIAL 106 -- EBLPATPS

Periodically Arrayed Gaussian Beams

Trial 106 arg2 arg3 arg4 arg5 arg6 . . . . sets the exposure pattern for a line made up of periodically arrayed, identical (same standard deviation) Gaussian beams (spots). Arg2 is the number of Gaussian beams in the line. Arg3 is the distance (negative numbers are not allowed) in microns between the center of the beams. Arg4 is the standard deviation of the spots. Arg5, arg6 and beyond are the 'weights' of each spot. Arg5 specifies the weight of the first spot, Arg 6 specifies the weight of the second spot, and so on. Each spot must have a corresponding weight. The weights indicate how much of the overall exposure dose is given to each Gaussian beam. For example, if there are 10 spots in a 1 um line, then the weight of each spot would be .1. The weights are very dependent on how the actual machine being simulated distributes the total dose of the electrons. Note that at least 2 spots must be specified to use trial 106. There is a maximum of 20 spots per line.

TRIAL 107 -- EBLPATNS

Non-Periodically Arrayed Gaussian Beams

Trial 107 arg2 arg3 arg4 arg5 . . . . . . sets the exposure pattern for a line made up of a non-periodic array of Gaussian beams. Each spot and its location relative to the first spot (which MUST be set at 0.0 microns) must be completeley specified. Arg2 is the number of spots. Arg3 is the position of the first spot (0.0). Arg4 is the standard deviation of the first spot. Arg5 is the weight of the

first spot. Similarly, args6-8 would describe the second spot, etc. At present, the maximum a spot may be shifted from the first spot at 0.0 microns is 5.0 microns-this holds for trial 106 as well. Negative distances are not allowed and the maximum number of spots per line is 20.

Note that trial 105 or trial 106 or trial 107 specifies an exposure line. Two or more of these trials being used will result in the last trial called being the one which specifies the line. Also note that the convolution accuracy is limited by the M/C cell size in the horizontal direction. Accuracy is degraded significantly when the standard deviation or edge-width is less than one M/C cell size in x.

TRIAL 108 -- EBLPLINE

Periodic Line Pattern

Trial 108 arg2 arg3 arg4 arg5 . . . . . . sets the exposure pattern for a series of periodically arrayed lines. Arg2 is the number of lines (2 or more). Arg3 is the periodic distance (negative numbers not allowed) between the center of two adjacent rectangular beam lines or the distance between the centers of the first spots in Gaussian beam lines. Arg4, arg5, etc. are the weights for each line. There must be one weight for each line. The weights allow the user to vary the relative overall exposure doses of each line. The maximum number of lines is 20.

TRIAL 109 -- EBLNLINE

Non-Periodic Line Pattern

Trial 109 arg2 arg3 arg4 . . . . . . sets the exposure patterns for one or more non-periodically arrayed lines. Each line and its location relative to the first line (which MUST be 0.0 microns) must be specified. Arg2 is the number of lines. Arg3 is the position of the first line (0.0). Arg4 is the weight of the first line. Similarly, arg5 and arg6 specify the second line and so on. There is no limit to the distances between lines. However, the Monte Carlo data only extends a finite distance. The program will warn the user when a line can not possibly contribute any energy density to the window of interest. Negative distances are not allowed and the maximum number of lines is 20. Note that this trial can be used to construct patterns of lines with different linewidths. For example, two 1 um rectangular beams can be arrayed to overlap at the half-maximum point to form a 2 um line.

TRIAL 110 -- EBLWIND

Window of Interest

Trial 110 arg2 arg3 arg4 sets the convolution and development window size and the position of the total exposure pattern in the window. Arg2 is the window size in microns. For Monte Carlo data with .02 um and less lateral cell size, the maximum size of the window is 10um. For cell size greater than .02 um, the maximum window size is 20um. Note that the larger the window, the larger the CPU time to run the program. If arg3=1, the left edge of the window will start at approximately (within one Monte Carlo cell size) the center of the exposure pattern (i.e. symmetrical development). If arg3=0, the position of the first spot in the first line or the position of the center of the first rectangular beam line in relation to the left window edge must be specified in arg4. If the first spot or rectangular beam is to the right of the left window edge, arg4 will be a positive number of microns. If to the left, arg4 will be negative.

TRIAL 111 -- EBLCNVLV

Convolution - Dose

Trial 111 arg2 runs the convolution and sets the overall exposure dose. Arg2 is the overall exposure dose in uC/cm**2. This overall dose is distributed over a line by way of the weights of each Gaussian as was explained previously. In the case of a rectangular beam, each beam receives the overall dose.

TRIAL 112 -- EBLSTRPTS

String Points - Anisotropic Development

Trial 112 arg2 arg3 sets the number of string points and the anisotropic development option. Arg2 is the number of points in the development string in the window of interest. 30-40 points/micron is usually sufficient. Accuracy and CPU time increase as the number of string points increases. If arg3 is set less than 1, there will be a reduction in the lateral motion of the string nodes by a factor of (1-arg3) [2]. If arg3 is set greater than 1 or less than 0, there will be erroneous results.

TRIAL 113 -- EBLNEWDOSE

Change Dose

Trial 113 arg2 changes the overall dose of the final energy array without reconvolving the entire exposure again. Arg2 is the new dose in uC/cm**2. Usually, trial 113 would be run after a development to change the dose for a subsequent development.

DEVTIME

Development Times

Devtime arg1 to arg2, arg3 sets the development times from arg1 to arg2 seconds in arg3 steps and is the same as used in the rest of the SAMPLE program.

TRIAL 114 -- EBLDEVELOP

Development

Trial 114 (no arguments) runs the development.

TRIAL 115 -- EBLENGPTS

Absorbed Energy Density Contours

Trial 115 arg2 arg3 arg4 sets the conditions for the printing out of various rows (depths) of the final energy array in the window of interest. If arg2=1, then the program will determine the maximum energy (ymax) printed in the output. If arg2 is any other positive integer (J/cm**3), that value will be used as the maximum energy. This maximum energy is printed out before the actual energy points and is only used for plotting purposes. Arg3 is the first row of the energy array to be printed out (1=top of the resist, # of rows of Monte Carlo data=bottom). Arg4 is the 'skip' number of rows-i.e. the number of rows added to the number of the first row to determine which row will be printed out. For example, for Monte Carlo data of 40 rows, arg3=1 and arg4=19 would result in rows 1,20,39 being printed out. This trial statement was inspired by a similar option in IBM's Lithography Modeling System (LMS) [3].

TRIAL 2

Output Options

Trial 2 arg2 arg3 arg4 from SAMPLE sets various printing options. If arg2=1, extra diagnostic printout is produced by the development machine. If arg3=1, points are printed for the developed plot suitable for plotting by HP graphics terminals or plotters at Berkeley. If arg4=1, a slower, more accurate algorithm is used which uses more string points and produces more accurate plots; little effect occurs for line printer plots.

The E-BEAM Program at Berkeley

Presently, the E-BEAM-SAMPLE simulation program runs on a VAX PDP 11/780 compiled with the Fortran f77 compiler. Line printer output includes a plot of the developed contours. Through the use of f77's 'open' and 'close' commands, points for plotting the final energy profiles (trial 115) and the developed resist profiles are printed out into files 'engpts' and 'f77punch7' respectively.

Monte Carlo Data

At present, the E-Beam program does not have the capability to generate the Monte Carlo data needed for the simulation. As this requires large amounts of computer time, several Monte Carlo files [4] are supplied with the main program. This data is for PMMA resist coats of 1.5 and .5 um on Si at 10 and 20 KeV beam energies. The cell size is .01 um in x and .1 um in z. The data is read from file 'mcdat' in subroutine EBCTRL(numb) using 'open' and 'close' statements and has the following form:

resist thickness (um)
beam energy (KeV)
number to convert Monte Carlo data to J/cm**3
cell size in the x (lateral) direction in microns
cell size in the z (vertical) direction in microns
number of rows of data (i2 format, maximum=80)
number of columns of data (i3 format, maximum=500)
Monte Carlo data-8e10.4 format per line. One entire row immediately following another.

Miscellaneous

For use on smaller computers, the Monte Carlo (EMAT(80,500)) and final energy (ELIN(82,1002)) array sizes can be reduced. This is done by changing the array sizes in the common blocks /CNVLV2/ and /LINE1/. Also, the checks for the input EMAT(80,500) size and the window size settings in subroutine EBCTRL(numb) must be changed. In subroutine EBMSG(numb), message 6 must also be corrected for the new maximum number of columns in the EMAT(80,500) array.

Different rate equations can be used by changing the /RATDAT/ common block and the EBRATE (etch-rate) and BACRAT

(background etch-rate) expressions in function EBRATE(cz). Note that the SAMPLE development routines require the etch-rate in units of um/sec.

Default Parameters-set in Trial 101

[1] set printing flags so that only exposure and
    development info. is printed

[2] rate curve data for PMMA 2010; 1:1 MIBK:IPA developer:
    R1=1.0
    Cm=1.0
    D0=199
    ALPH=2.0

[3] exposure pattern:
    1.0 um rectangular beam with .25 um edge-width
    3 periodic lines, 2.0 um apart (center to center)
    dose=80 uC/cm**2

[4] development:
    symmetric with 2.0 um window, 60 string points
    40 to 160,4 development times

[5] energy plotting option:
    first row=1
    skip rows=7
    program sets ymax

[6] print out points for plot

Examples of SAMPLE Input Files for E-BEAM

The following input files are designed to illustrate the use of the e-beam simulation program in SAMPLE. The first input file is the simplest one possible:

```
**initialize default parameters
trial 101
**run convolution
trial 111
**print pts for energy contours
trial 115
**develop
trial 114
$
```

The above file simulates an exposure pattern made entirely from the default values previously discussed. Statements beginning with one or more '*' are regarded as comment lines by SAMPLE. The '$' at the end of the file tells SAMPLE to perform the operations indicated by the last trial statement and then stop as there is no more input. Figure 1 shows the developed profiles as plotted on an HP 2648 graphics

terminal at Berkeley. Monte Carlo data for 1.5 um PMMA on Si at 20 KeV was used. Figure 2 shows a plot of the energy absorbed in the resist in the development window for rows 1, 8, and 15 of the final energy array (i.e. the top, middle, and bottom of the resist).

The next input file illustrates many more aspects of the use of the various trial statements:

```
**initialize default parameters
trial 101
**set flags
trial 2 0 1 1
**specify line
trial 105 1.5 .3
**specify array of line(s)
trial 109 1 (0 1)
**specify convolution and development window
trial 110 3 0 1.5
**set dose and run convolution
trial 111 90
**set number of string points
trial 112 60
**set development times
devtime 10 to 90,9
**develop
trial 114
$
```

Trial 2 says to print out the developed profile points into the file 'f77punch7' and also to use a more exact development (for better plotted profiles from the HP graphics terminal). Trial 105 sets a rectangular beam to fwhm of 1.5 um and edge-width of .3 um. Trial 109 says to write only this one line with a weight of 1 (note that parentheses are ignored by SAMPLE and are only used for clarity). Trial 110 sets a 3.0 um window with the center of the rectangular beam 1.5 um from the left edge of the window. Trial 111 runs the convolution with an overall exposure dose of 90 uC/cm**2. Trial 112 says to use 60 string points and devtime says to develop from 10 to 90 seconds in 10 second intervals. Figure 3 shows the resulting profiles in .5 um of PMMA on Si at 20 KeV. Default PMMA 2010 developed in 1:1 MIBK:IPA is used as the resist.

The last two examples use Gaussian beams to form the exposure lines:

```
**initialize default parameters
trial 101
**set flags
trial 2 1 1 1
**specify line
trial 107 4 (0 .05 .2) (.2 .1 .2) (.4 .1 .2) (.6 .05 .2)
```

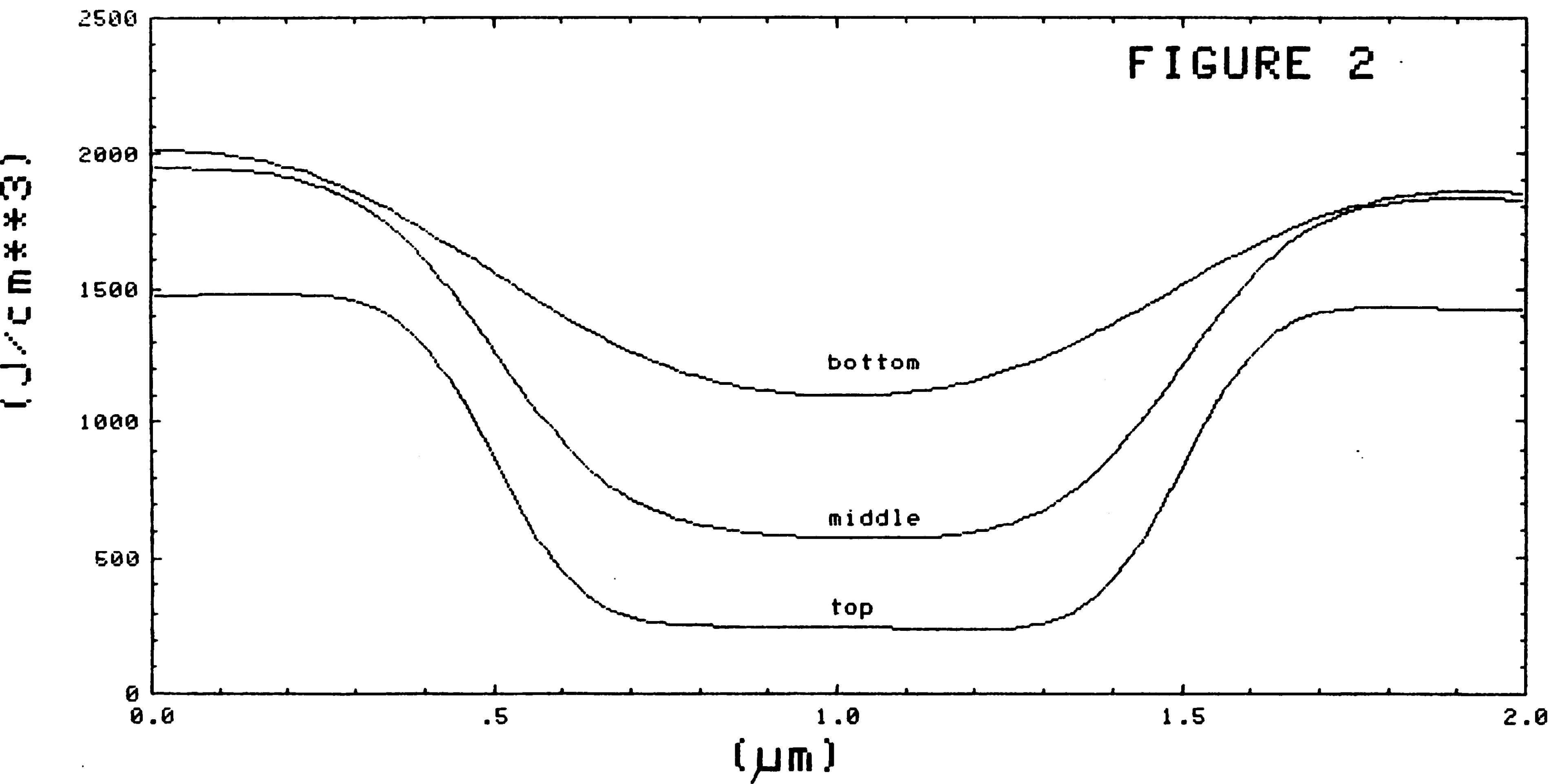


FIGURE 2
2500
2000
1500
1000
500
0
(J/cm**3)
bottom
middle
top
0.0
.5
1.0
1.5
2.0
(μm)

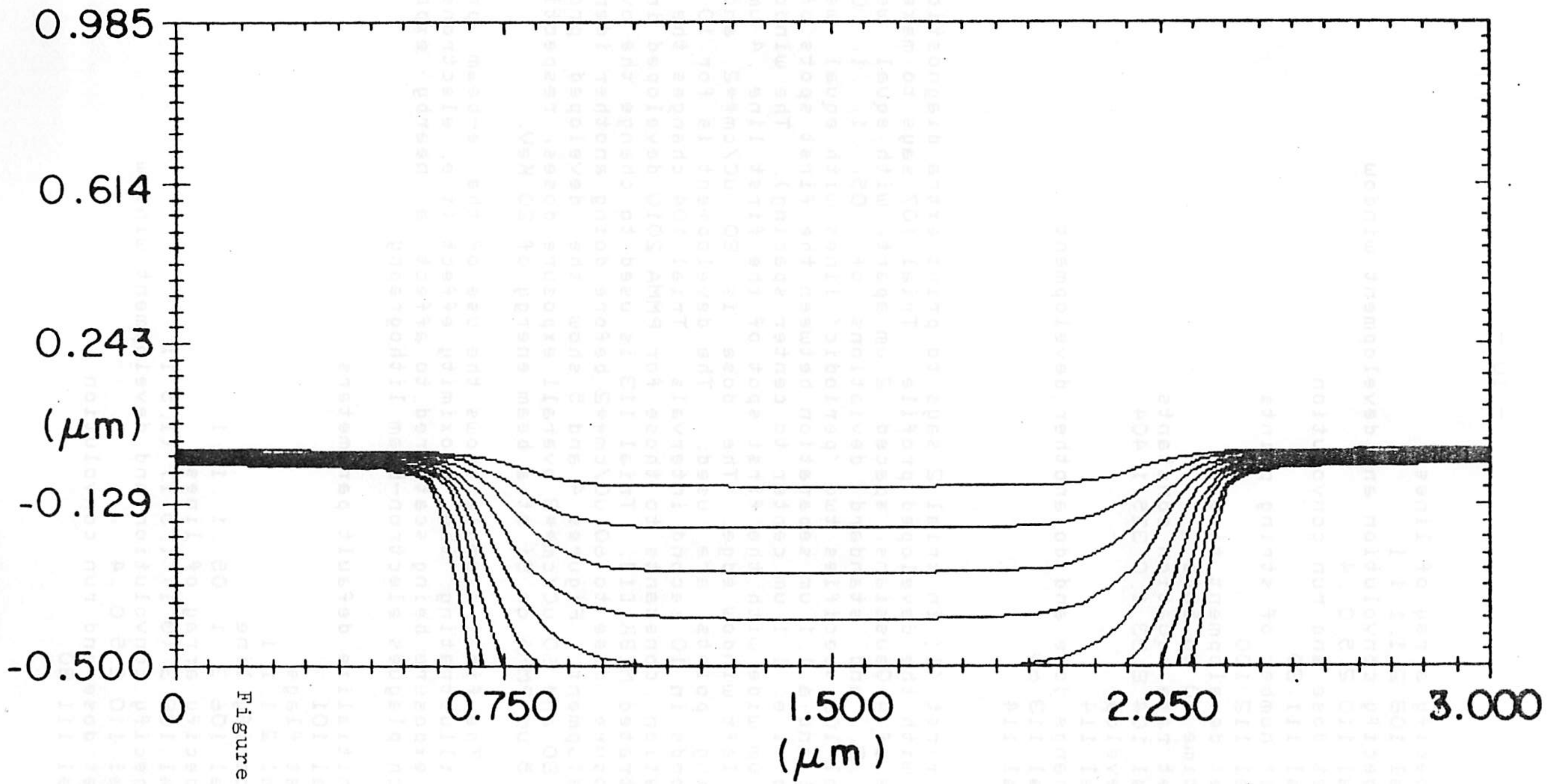


Figure 3

```
**specify array of lines
trial 108 2 1.1 1 1
**specify convolution and development window
trial 110 2.5 0 .4
**set dose and run convolution
trial 111 80
**set number of string points
trial 112 100
**set development times
devtime 10 to 50,5
**set rate equation constants
trial 104 8.33 1.0 325 1.404
**develop
trial 114
**change dose and do another development
trial 113 60
trial 114
$
```

The first '1' in trial 2 says to print extra diagnostic output with the developed profile. Trial 107 says to make up a line of 4 Gaussians, spaced .2 um apart, with equal weights of .2, and standard deviations of .05, .1, .1, .05 um. Trial 108 specifies two 'periodic' lines with equal weights of 1 and a 1.1 um separation between the first spots of each line (i.e. 1.1 um center to center spacing). The window is 2.5 um wide with the first spot of the first line .4 um from the left window edge. The dose is 80 uC/cm**2 and 100 string points are used. The development is for 10 to 50 seconds in 10 second intervals. Trial 104 changes the rate equation constants to those for PMMA 2010 developed in concentrated MIBK [1]. Trial 113 is used to change the overall exposure dose to 60 uC/cm**2 before doing another identical development. Figures 4 and 5 show the developed profiles for 80 and 60 uC/cm**3 overall exposure doses, respectively, in .5 um PMMA on Si at a beam energy of 20 KeV.

The final example shows the use of the e-beam program in illustrating the proximity effect (i.e. electrons from one exposure being scattered to affect a nearby exposure) which plagues electron-beam lithography:

```
**initialize default parameters
trial 101
**set flags
trial 2 1 1 1
**specify line
trial 106 3 .1 .05 .1 .1 .1
**specify array of lines
trial 109 3 (0 1) (.5 1) (1.5 1)
**specify convolution and development window
trial 110 2.5 0 .4
**set dose and run convolution
trial 111 80
```

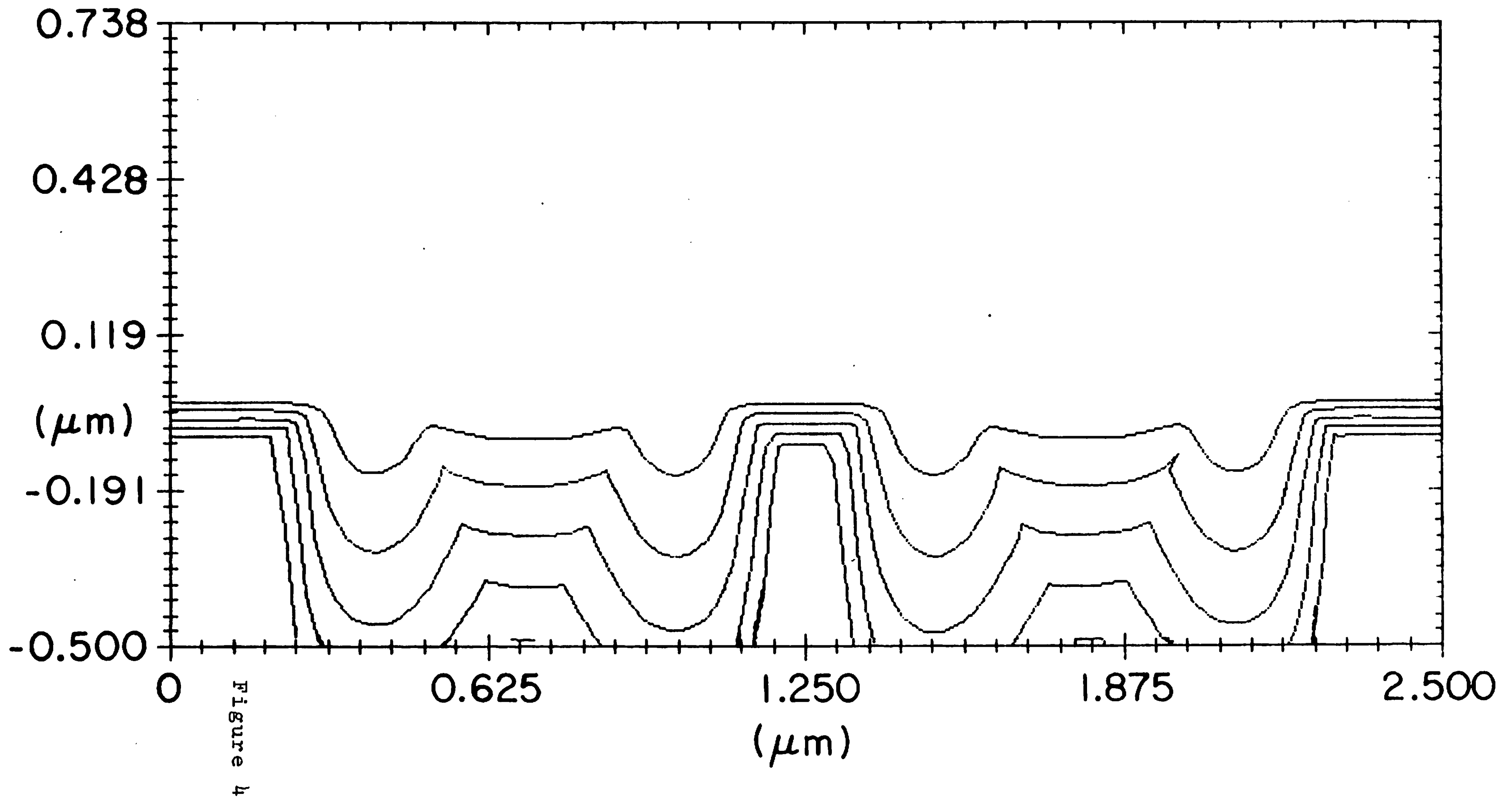


Figure 4

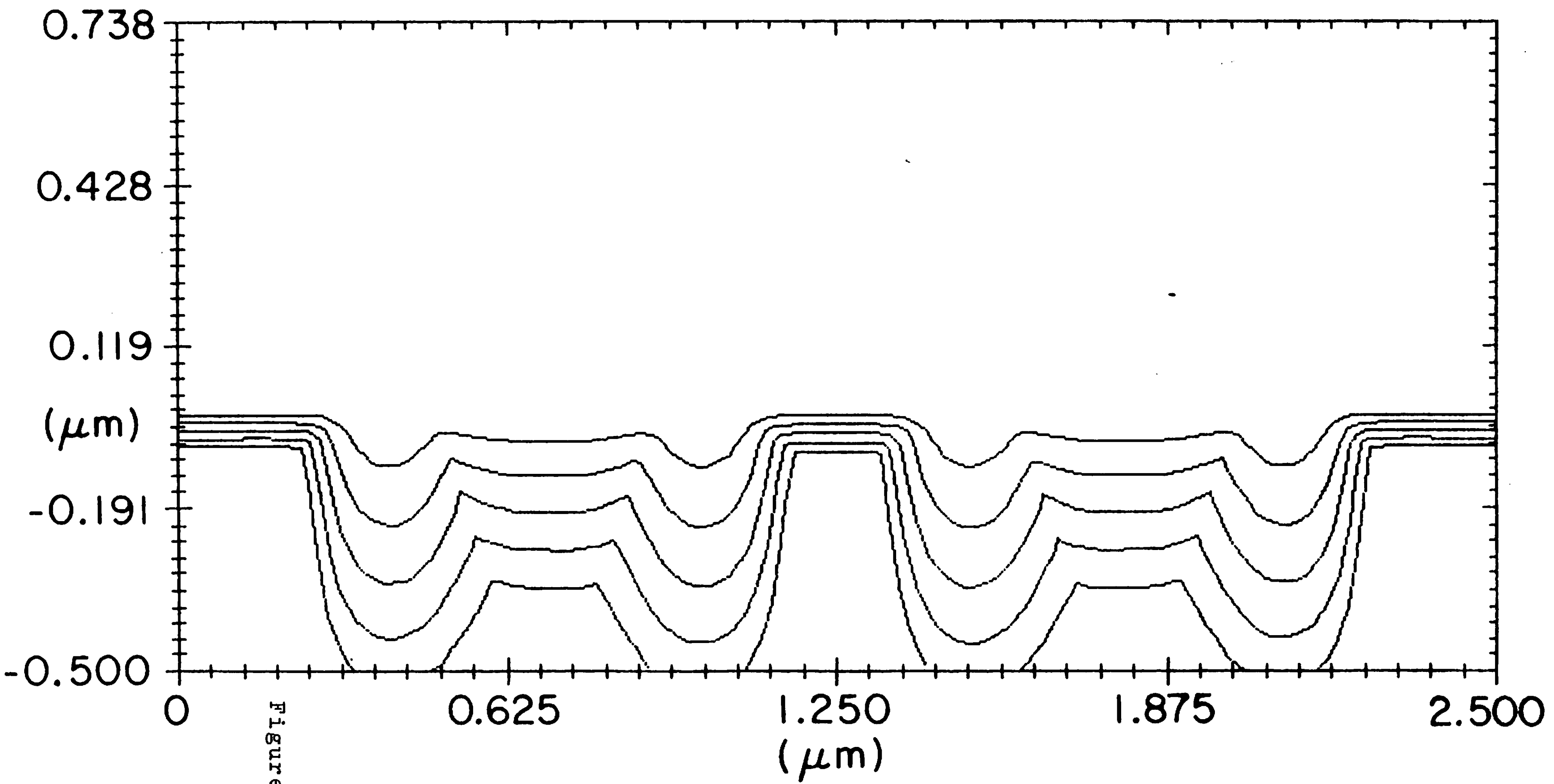


Figure 5

```
**print pts for energy contours
trial 115 1 1 7
**set number of string points
trial 112 60
**set development times
devtime 50 to 350,7
**develop
trial 114
$
```

Trial 106 specifies a line composed of 3 periodic Gaussians of .05 um standard deviation, spaced .1 um apart with weights of .1 apiece. Trial 109 is used to array the lines so that the distance between line 1 and line 2 is .5 um while the distance between line 2 and line 3 is 1.0 um. Figure 6 shows the developed profiles (50 to 350 seconds in 50 second intervals) using the default rate equation constants for PMMA 2010 in 1:1 MIBK:IPA developer. Notice how the two closely spaced lines interact to finally develop out to one large opening at the Si surface. Also, note that the third line is slightly skewed towards the other two-again due to the proximity effect. This interaction of the energy contributions of each line is shown in Figure 7-a plot of the energy absorbed in the resist at the top, middle, and bottom. The 1.5 um PMMA on Si Monte Carlo data was used in the simulation.

References

[1] A.R. Neureuther, D.F. Kyser, and C.H. Ting, IEEE Trans. Electron Devices, vol. ED-26, no. 4, pp. 686-693, 1979.

[2] M.G. Rosenfield and A.R. Neureuther, "Exploration of Electron Beam Writing Strategies and Resist Development Effects", presented at the Ninth Int. Conf. on Electron and Ion Beams in Sci. and Tech., The Electrochemical Society, St. Louis, May, 1980.

[3] D.F. Kyser and R. Pyle, IBM J. Res. Develop., vol. 24, no. 4, pp. 426-437, July, 1980.

[4] I. Adesida, "Electron Energy Dissipation in Layered Media", Ph.D. dissertation, Dept. of EECS, U.C. Berkeley, 1979.

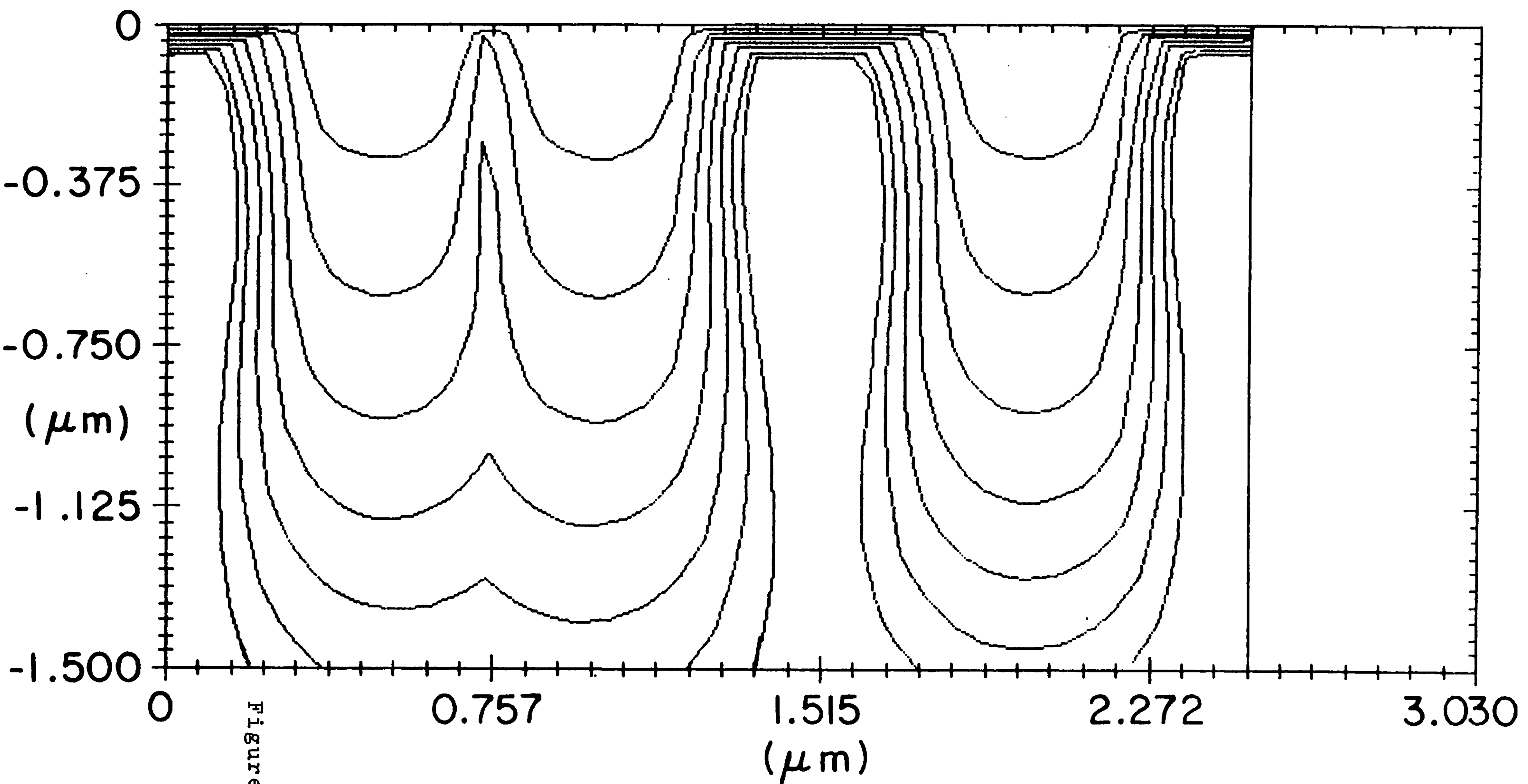


Figure 6

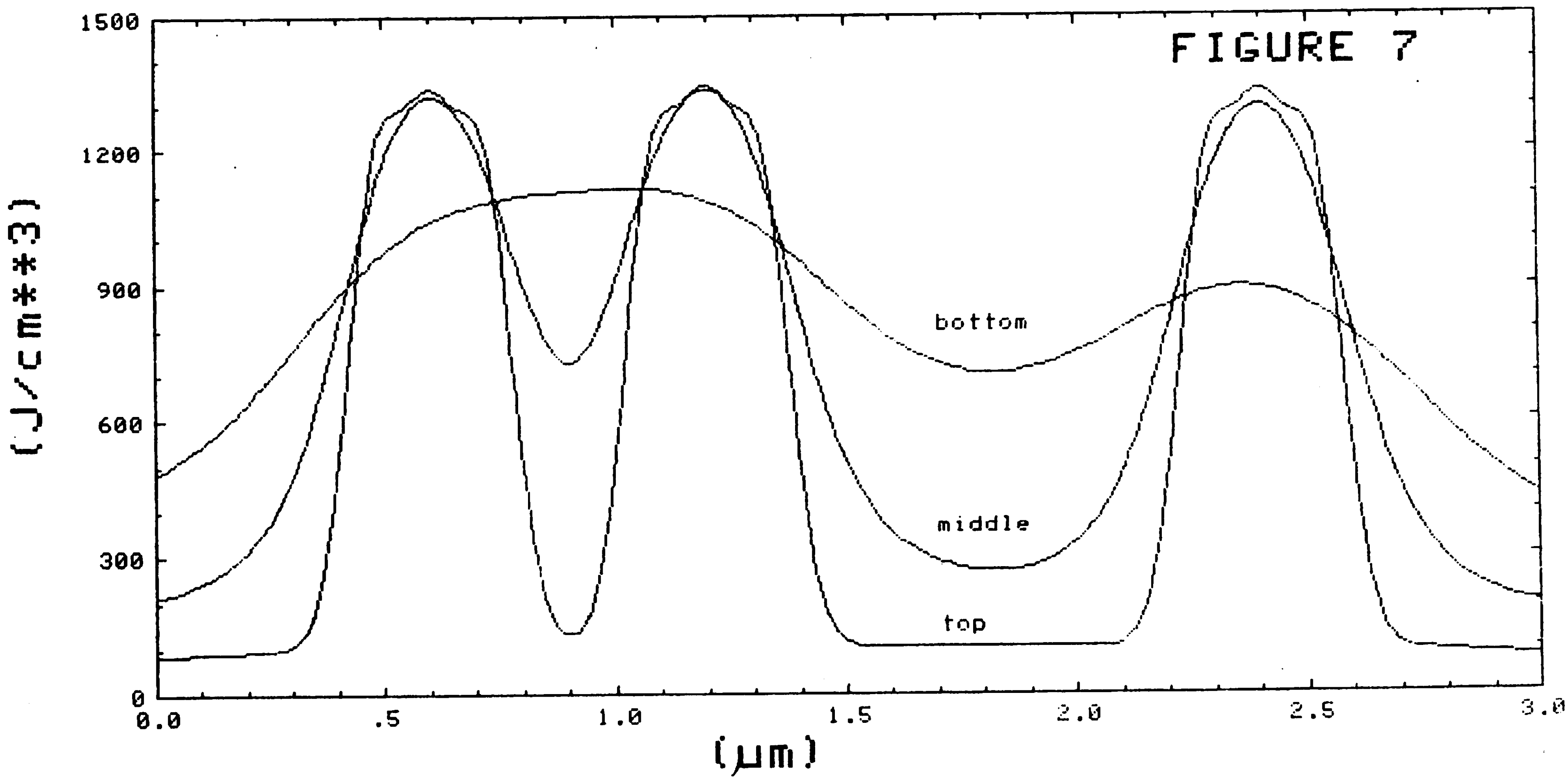


FIGURE 7
bottom
middle
top
(µm)
(J/cm**3)
0.0
.5
1.0
1.5
2.0
2.5
3.0
0
300
600
900
1200
1500

Appendix II -- Simulation Example

EXPLORATION OF ELECTRON-BEAM WRITING STRATEGIES AND RESIST DEVELOPMENT EFFECTS

Michael G. Rosenfield and Andrew R. Neureuther

Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley, California 94720

ABSTRACT

An electron-beam exposure program being developed in conjunction with program SAMPLE is used to explore writing strategies and a modification to the development model for electron-beam lithography. In particular, a processing approach based on an initial resist thinning followed by a secondary exposure is explored and compared with more conventional single exposure strategies. Profile description parameters are introduced to quantitatively describe the profile shape and sensitivity to development time. In addition, the introduction of an anisotropic component in the resist development model is shown to result in better agreement of simulated profiles with experiment.

Introduction

As the complexity of circuits and processes increases, the performance tradeoffs of various lithographic techniques must be carefully considered. A comparison between lithographic tools or the optimization of a given tool can only be made in terms of a particular type of process and application. Simulation with suitable experimental verification is a very effective way to systematically explore key performance aspects such as resist profile quality and linewidth control. This paper uses simulation to characterize resist line edge quality for the particular case of direct electron-beam writing in a single layer of resist of sufficient thickness for step coverage. The results primarily consider various electron-beam writing strategies; but, some of the data such as linewidth sensitivity is also suitable for comparison with optical lithography.

In using thick resist coats, several problems are encountered which tend to reduce the practical performance well below that of the ability of the exposure tool to focus the exposing electrons. The lateral scattering of the electrons which increases with resist thickness as well as

proximity effects contribute to a broadening of the exposure with depth. The development process, which must be characterized by additional physical parameters, also introduces non-vertical motion in the time evolution of the developing resist profile which increases with resist thickness. To quantitatively describe the lithographic quality of the resulting thick resist line edges, profile description parameters are introduced. These parameters describe the developed profile shape, its deviation from a desired or optimum profile shape, and the sensitivity of the shape to development time.

Since it is well known that better dimension control can be achieved in a thinner resist layer, a potential strategy for improving critical dimension control and profile shape with thick resist is to locally expose and develop (thin) the initial resist and then make a secondary exposure of the critical features as shown in Figure 1. The desired features are then precisely detailed in the remaining thinner resist layer. In this paper, simulation is used to explore the quantitative advantages of this local initial resist thinning procedure as well as more conventional single exposure methods.

Another important issue for simulation is the quality of agreement of predicted profiles with experiment. Developed resist profile simulation results reported earlier [1-2], while giving first order agreement with experimental profiles, were found to be broader and more rounded than their experimental counterparts. This paper also explores these subtle differences using several adjustments and extensions of the exposure and development models. Variations in the standard exposure and development model parameters [1] are considered as well as a potentially more promising approach of introducing an anisotropic component in the development model.

The simulation approach used here is similar to that reported previously [1]. Monte Carlo data [3-4], giving the spatial distribution of energy deposited in the resist by a delta-function line source, is convolved with Gaussian shaped electron-beam spots. These spots are then periodically arrayed to give the energy density absorbed in the resist for various patterns of exposure. The development is simulated by using the curve fit of a simple etch rate versus dose curve [1]:

$$R(D) = R1(Cm+D/D0)^{\alpha}$$

where R(D) is the etch rate in A/sec, D is the absorbed energy density in J/cm3, $R1Cm^{\alpha}$ is the background etch-rate in A/sec, Cm is a constant inversely proportional to the initial number average molecular weight, D0 is a reference

or knee energy and α is the asymptotic slope at very high dose. The above equation, combined with the string model of development [5], is used to simulate the time evolution of two-dimensional resist line-edge profiles. These are implemented through a modified version of the Simulation And Modeling of Profiles in Lithography and Etching (SAMPLE) computer program [6,7] developed at Berkeley.

Local Initial Resist Thinning and Conventional Exposure Techniques

The initial resist thinning approach consists of first locally thinning the resist and then exposing the critical features. It can be implemented as illustrated in Figure 1. A primary exposure is made in the thick resist in the areas of the desired openings. The resist is then developed so that the resist thickness is significantly reduced only in the desired areas. A second exposure is made in these thinner resist regions and the critical dimensions are then precisely developed out. This approach is particularly suited to masking applications such as the opening of contact windows on a wafer with severe surface topography. In an attempt to somewhat realistically reflect the electron-beam writing throughput limitations, the combined dose of the primary and secondary exposures was constrained to be the same as the total dose used in the more conventional single exposure method.

Simulation of the initial resist thinning strategy can be performed by the superposition of the absorbed energy densities due to the primary and secondary exposures. This is done by adding the secondary exposure's energy density to the primary exposure's energy density starting at the initial resist thinning depth, d (Figure 2). In the case study presented here, 1.0 um PMMA 2010 on silicon was used with an initial resist thinning to a depth of .5 um. It was thus expedient to reuse the top .5 um of the 1.0 um Monte Carlo data to simulate the secondary exposure. The credibility of using this approximate simulation approach was reported earlier [8].

As mentioned previously, the comparison between the various exposure methods will be made with the constraint of the same total exposure for all techniques. A nominal .5 um linewidth at the resist-silicon interface in 1 um PMMA 2010 developed in 1:1 MIBK:IPA was chosen as the critical dimension for comparison purposes. The development parameters for the etch rate versus dose curve used in the modeling process have been reported earlier [1].

A conventional single exposure method used to open a .5 um line is shown in Figure 3a. The four electron-beam spots have a FWHM of .125 um and were thus spaced accordingly at 8 spots/um density. The dose was 45.6 uc/cm2 and each spot

was weighted equally (given an equal share of the total dose). The developed contours correspond to development times of 400 to 600 seconds in 20 second intervals. As can be seen, the contours about the .5 um nominal linewidth are overcut and rounded with the linewidth increasing at a rate of .0029 um/sec.

In an effort to improve on the developed profiles of Figure 3a, the relative doses of the four beam spots were adjusted while keeping the total dose the same. In effect, the two middle spots were given approximately four times the dose of the outer two spots. The developed contours, corresponding to development times of 260 to 400 seconds in 20 second intervals, are shown in Figure 3b. In this case, the sidewalls are more vertical and the linewidth is advancing at a lower rate of .0021 um/sec about the .5 um nominal linewidth.

In Figure 4, the reduced spot writing scheme of Greeneich [9] is simulated. This method uses only two spots spaced at a distance equal to twice their FWHM value (.125 um). The total dose was again 45.6 uc/cm2. The developed contours correspond to development times of 280 to 480 seconds in 20 second intervals. The contours are slightly improved over the contours of Figure 3a in that the sidewalls are slightly more vertical and that the linewidth is advancing at .0026 um/sec about the .5 um point.

Several versions of the initial resist thinning approach were explored in attempting to obtain greater linewidth control and improved shape. The first method simulated was simply a normal four spot initial exposure, as in Figure 3a, at half the dose (22.8 uc/cm2) and an identical secondary exposure at a depth in the resist of .5 um as illustrated in Figure 5a. The contours correspond to development times of 700 to 900 seconds in 20 second intervals. The sidewalls are more vertical and the linewidth is advancing at .0021 um/sec-which is an improvement over both the conventional technique of Figure 3a and the two spot approach of Figure 4.

This resist thinning process can be further improved by assigning different weights to each of the exposure spots. The method which appears to give the most significant improvement is shown in Figure 5b. The initial exposure is still four spots; however, the middle spots were given five times more dose than the outer spots. The secondary exposure then utilized only two spots directly below the initial two middle spots at a depth of .5 um. The secondary exposure was given three and one-half times more dose than the initial outer spots. The total dose was still constrained to be 45.6 uc/cm2. The developed contours correspond to development times of 280 to 480 seconds in 20 second intervals. The contours show significant improvement over

results of previous techniques. The sidewalls are vertical and the linewidth is developing out at a much slower rate, .0013 um/sec about the .5 um nominal linewidth.

For comparison purposes, the use of two spots similar to those used in the preceding secondary exposure were simulated using a conventional approach. Contours corresponding to development times of 200 to 400 seconds in 20 second intervals at a dose of 45.6 uc/cm2 are shown in Figure 6. The sidewalls are slightly curved and the linewidth is opening at a rate of .0018 um/sec about the nominal .5 um linewidth.

A plot of linewidth versus development time for the various exposures discussed is shown in Figure 7. As can be seen, the various strategies have slopes which differ by over a factor of 2. The optimized initial resist thinning procedure gives about 28% better linewidth control (defined as the slope of the curve about the .5 um nominal linewidth) over the two spot exposure of Figure 6, a 40% improvement over the weighted conventional exposure technique of Figure 3b, a 50% improvement over the technique of Greeneich in Figure 4, and a 55% improvement over the conventional technique of Figure 3a.

Profile Description Parameters

We now attempt to quantify the comparison of resist profiles by introducing a set of profile description parameters. The quantities x(0), x(.4), and x(.8) are defined as the half-linewidths at the relative remaining resist thicknesses of 0, .4, and .8, respectively. The definition of these parameters is illustrated in Figure [8]. Note that the the .8 level allows for reasonable top loss of resist.

From these three direct profile parameters, two independent quality parameters can be calculated. The slope or angle [10] of the sidewalls is specified by the 'tilt', T:

$$T = (x(.8)-x(0))/.8t_h$$

where t_h is the resist thickness. This is equivalent to the inverse slope of the straight line connecting the resist openings at x(.8) and x(0). The curvature of the sidewalls is described by the quantity, C:

$$C = (x(.4)-.5(x(.8)+x(0)))/.8t_h$$

C is the horizontal distance from the resist opening at x(.4) to the straight line which defines T. These quantities are also shown in Figure 8. Normalization of the above

quantities is made to resist thickness and not linewidth since the quality of a resist edge profile is primarily due to exposure beam quality, electron scattering, and developer effects and is only related to linewidth to second order.

In order to make comparisons with the optimum desired profiles for a particular process, a process performance index can be defined in terms of T and/or C. For example, the following definition can be made:

$$\Delta_{eff} = (|T-T_n|+|C-C_n|)$$

where Δ_{eff} is the deviation from the optimum, T_n is the tilt of the desired profile and C_n is the curvature of the desired profile Thus, a $\Delta_{eff} = 0$ indicates a perfect match between an experimental or simulated profile and the optimum. A figure of merit to describe a given profile is thus defined as:

$$Q = .8t_h/\Delta_{eff}$$

This is normalized to include the resist thickness. Thus, the closer a profile corresponds to the desired resist profile, the higher will be the Q value. It should be noted that the analysis allows almost any type of nominal profile shape to be used (for example, undercut profiles when comparing profiles for certain liftoff processes).

It is more difficult to define the bias or difference in half-linewidth between the actual and optimum profiles. A useful definition must reflect the position of the portion of the resist profile, X_R , which is critical to the pattern transfer process. More importantly, it must also implicitly define what is meant by the nominal exposure half-linewidth, X_E . One definition of X_E which we find convenient, and only slightly ambiguous, is the 50% exposure dose level of the nearest written spot or beam edge.

However, to emphasize the difference in bias for the various writing strategies in this paper, X_E is assumed to be .25 um for all the patterns. Furthermore, the desired optimum profile is chosen to have zero tilt and curvature. The bias, B, which in general is defined to be the distance:

$$B = X_R - X_E$$

becomes:

$$B = x(.8)\big|_{T=0} - .25\mu m$$

Since the desired profile is vertical, X_R , is chosen to be the value of X(.B) when the tilt is zero. That is, B is the over or under developed distance from the nominal .25 um half-linewidth at which the developed profile has zero tilt (Figure 8).

To complement the quantitative description of profile shape, an expression is needed which will reflect the sensitivity of the feature size to development time. The sensitivity can be defined as:

$$S = \frac{td}{x(0)} \frac{\partial(x(0))}{\partial(td)}$$

where td is the development time. A figure of merit for development time effects is thus:

$$FM = 1/S$$

These profile description parameters allow the discussion of resist profile quality for the various writing strategies to be restated more quantitatively (Table 1). As expected, the four-spot conventional approach of Figure 3a has both a low F.M. and Q-indicating poor shape quality and linewidth control. It also has a very large bias of more than 0.14 um. The weighted four spot approach shows significant improvement in linewidth sensitivity as well as in shape and bias. The two spot approach of Greeneich has a similar profile quality but a lower development figure of merit than the weighted four spot method. The four spot thinned approach of Figure 5a shows that the thinning technique can improve curvature and quality. However, the developer figure of merit is degraded slightly. The optimized thinned approach of Figure 5b shows a very significant improvement in both quality and developer figure of merit while maintaining a small bias. The two spot conventional exposure of Figure 6 also shows similar improvements but has a noticeably larger curvature.

From these six cases several general observations about writing strategies can be made. The initial resist thinning approach gives greater control over the latent energy deposition distribution in resist which should improve both resist profile quality and development figure of merit. The lower net energy deposition may, however, lengthen development time. Major improvements in both bias and quality can be made by reducing or eliminating the exposure near the edges of the line. This is analogous to using an over exposure and development in optical lithography and accepting a standard bias while applying windage to the exposure pattern. Finally, the relative merits of a particular approach are apt to be a strong function of the exposure and resist

development parameters.

Investigation of the Resist Model

Now let us turn to the problem of improving the agreement between simulation and experimental resist profiles presented earlier [1]. The initial approach was to further explore fine tuning the development model parameters; but, little impact could be made. For example, making the beam size smaller in an attempt to eliminate the rounded bottom of the simulated profile resulted in the top corners becoming too sharp with little change in the bottom. The shape of the etch rate versus dose curve was also adjusted by changing D0, α , and R1; but, again there was no significant improvement. Thus, we conclude that to achieve better agreement with experiment, the basic resist development model must be generalized.

As a second approach, the development model was generalized to allow the development of the resist to occur more rapidly approximately along the direction of the primary electron trajectories. This anisotropic effect was implemented approximately by a fractional reduction in the lateral motion of the nodes in the string algorithm as is illustrated in Figure 9. Figure 10 is a comparison of experimental and simulated profiles from [1] and simulated profiles with a 70% reduction in lateral development. The adapted profiles correspond to development times of 20 to 180 seconds in 20 second intervals. The experimental contours from [1] correspond to a 180 second development time while the simulated profiles from [1] correspond to development times of 20 to 200 seconds in 20 second intervals. All other parameters are the same as in [1].

For the case of isolated lines in Figure 10, the profiles with this simple reduction in lateral development rate more closely track the experimental contours. However, for arrays of lines in Figure 10, the correction was not as necessary. A possible explanation is that development tends to occur more rapidly along the trajectories of the high energy electrons. The physical mechanism causing such an effect might be related to the generation of volatile products and micropores during exposure [11]. This preferential development along trajectories is consistent with there being more of a need to correct the simulated profiles for isolated lines as compared to arrayed lines. For single lines, the electron trajectories would be fairly vertical with spreading increasing with depth in the resist. For arrayed lines, however, proximity effects contribute a significant mix of non vertical backscattered electrons which would tend to average out the preferential effects. Although the basic physical mechanism is not clear, the simple addition of a preferential development factor appears to be a simple procedure for significantly improving the

agreement of simulated profiles with experiment.

Conclusion

One of the advantages of simulation is its flexibility to explore potential processing approaches. Various assumptions have been used to explore writing strategies and an improved resist model for electron-beam lithography. An initial resist thinning and secondary exposure particularly applicable to contact opening has been suggested. Profile description parameters have been introduced to evaluate profile shape and linewidth sensitivity more quantitatively. These parameters show that by weighting the middle spots more heavily or reducing the number of spots/line improves the profile shape as well as the sensitivity to development time. This corresponds to designing a resist thickness dependent bias into the exposure pattern. The initial resist thinning approach was also shown to result in further improvements in both profile quality and linewidth sensitivity. Several possible exposure and resist development effects were explored to improve the detailed fit of simulation to experimental line-edge profiles. Introducing a preferred anisotropic etching approximately in the direction of the primary electron trajectories was found to give a significant improvement in the comparison, especially for isolated lines.

Acknowledgment

The authors would like to thank C.H. Ting, D.F. Kyser, S.N. Nandgaonkar, and Y.C. Lin for valuable discussions during the course of this work.

Research sponsored by the National Science Foundation Grant ENG77-14660 and IBM Fellowship.

Parts of this paper were originally presented at the Spring 1980 meeting of The Electrochemical Society, Inc. held in St. Louis, Missouri.

References

[1] A.R. Neureuther, D.F. Kyser, and C.H. Ting, IEEE Trans. Electron Devices, vol. ED-26, no. 4, pp. 686-693, 1979.

[2] K. Murata, E. Nomura, K. Nagami, T. Kato, and H. Nakata, J. Vac. Sci. Technol., vol. 16, no. 6, pp. 1734-1738, Nov./Dec. 1979.

[3] D.F. Kyser and K. Murata, Proc. 6th Int. Conf. on Electron and Ion Beam Sci. Tech. (Electrochemical Society, 1974), R. Bakish, ed., pp 205-223.

[4] I. Adesida, "Electron Energy Dissipation in Layered Media", Ph.D dissertation, Dept. of EECS, U.C. Berkeley, 1979.

[5] R.E. Jewett, P.I. Hagouel, A.R. Neureuther, and T. Van Duzer, Polymer Eng. Sci., vol. 17, no. 6, pp. 381-384, 1977.

[6] W.G. Oldham, S.N. Nandgaonkar, A.R. Neureuther, and M.M. O'Toole, IEEE Trans. Electron Devices, vol. ED-26, no. 4, pp. 717-722, 1979.

[7] W.G. Oldham, A.R. Neureuther, C. Sung, J.L. Reynolds, and S.N. Nandgaonkar, IEEE Trans. Electron Devices, vol. ED-27, no. 8, pp. 1455-1459, 1980.

[8] M.G. Rosenfield and A.R. Neureuther, Proc. 9th Int. Conf. on Electron and Ion Beam Sci. Tech. (Electrochemical Society, 1980), R. Bakish, ed., pp. 382-395.

[9] J.S. Greeneich, J. Vac. Sci. Technol., vol. 16, no. 6, pp. 1749-1753, Nov./Dec. 1979.

[10] M. Hatzakis, J. Vac. Sci. Technol., vol. 12, no. 6, pp. 1276-1279, Nov./Dec. 1975.

[11] A.C. Ouano, Polymer Eng. Sci., vol 18, no. 4, pp. 306-313, 1978.

Figure Captions

1. Local initial resist thinning strategy.

2. Local initial resist thinning simulation approach

3. Simulation of conventional exposure techniques. a) Equally weighted spots b) Weighted spots -- inner spots have 4 times the dose of the outer spots.

4. Simulation of reduced spot technique -- two spots equally weighted.

5. Simulation of local initial resist thinning techniques. a) Identical initial and secondary exposures b) Weighted initial exposure -- inner spots have 5 times the dose of outer spots, secondary exposure spots have 3.5 times the dose of the outer spots.

6. Simulation of a two spot conventional exposure similar to the secondary exposure of Figure 5b.

7. Linewidth versus development time for the various exposure techniques.

   Table 1 -- Profile description parameters for the various exposure techniques.

8. Illustration of some of the profile description parameters in Table 1.

9. Reduction of lateral motion of string nodes in the development model. $(1-\alpha)*$ 100% is the reduction.

10. Comparison of earlier simulated (left) and experimental (middle) resist profiles [1] with simulated profiles corresponding to a 70% reduction in lateral string motion (right). Simulated contours correspond to 20 second intervals. Experimental development time is 180 seconds.

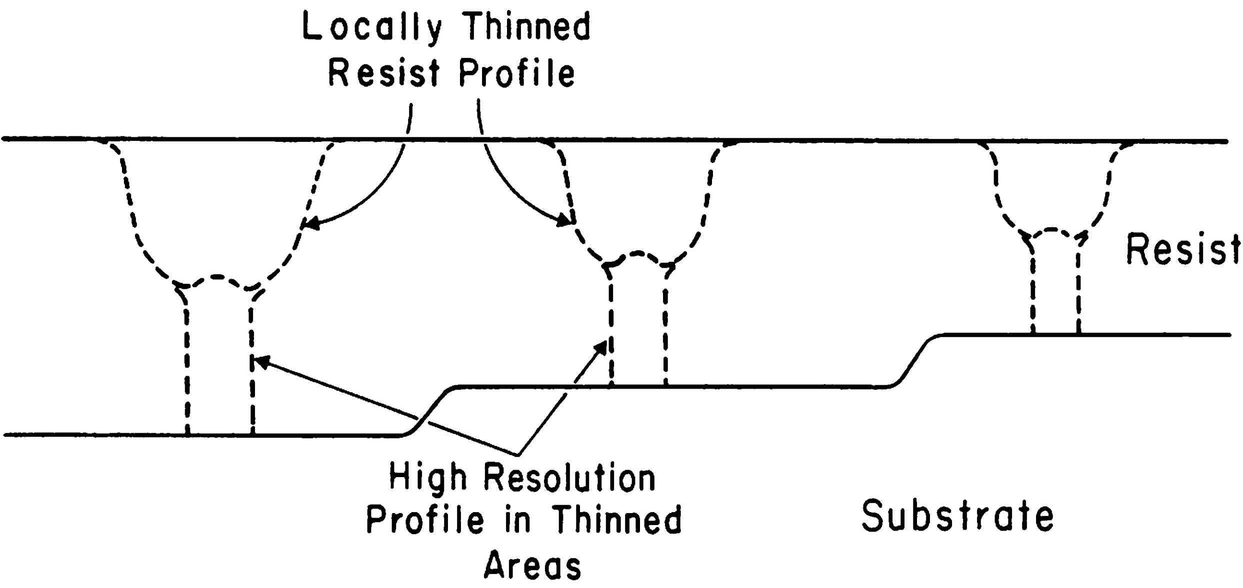


Figure 1

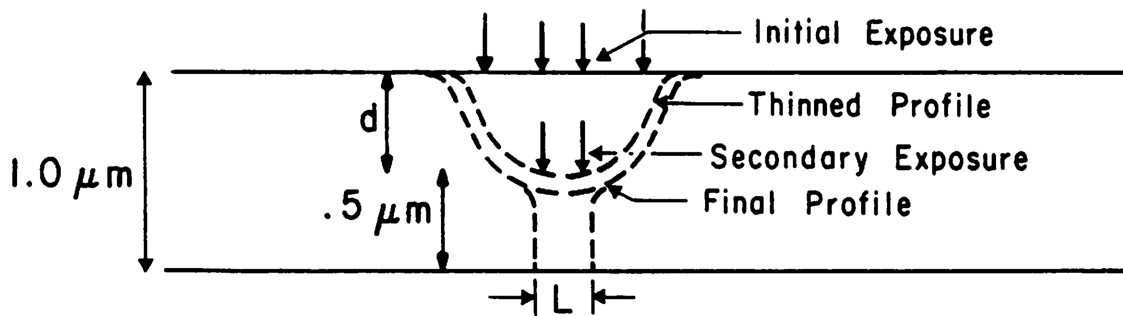


Figure 2

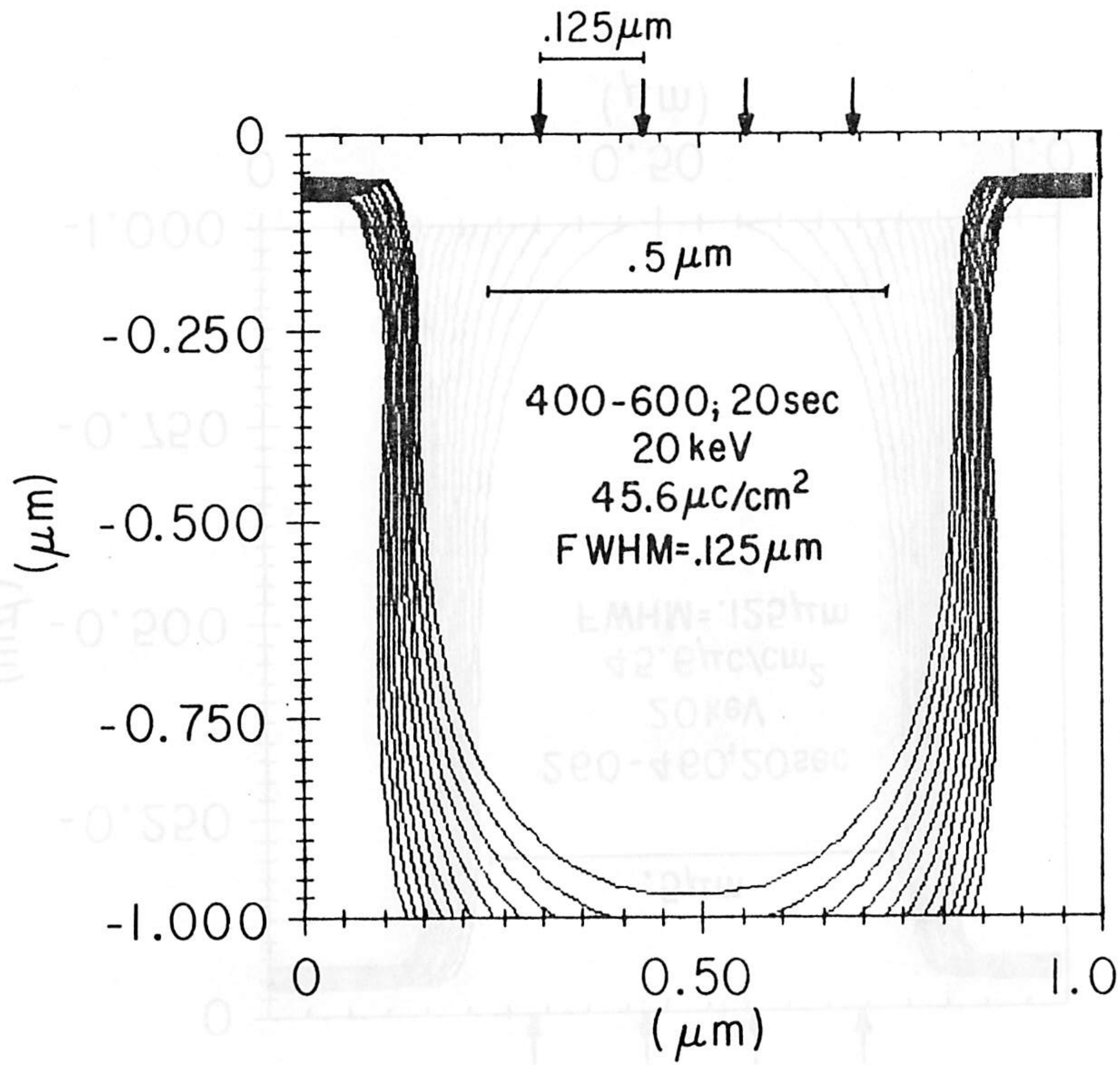


Figure 3a

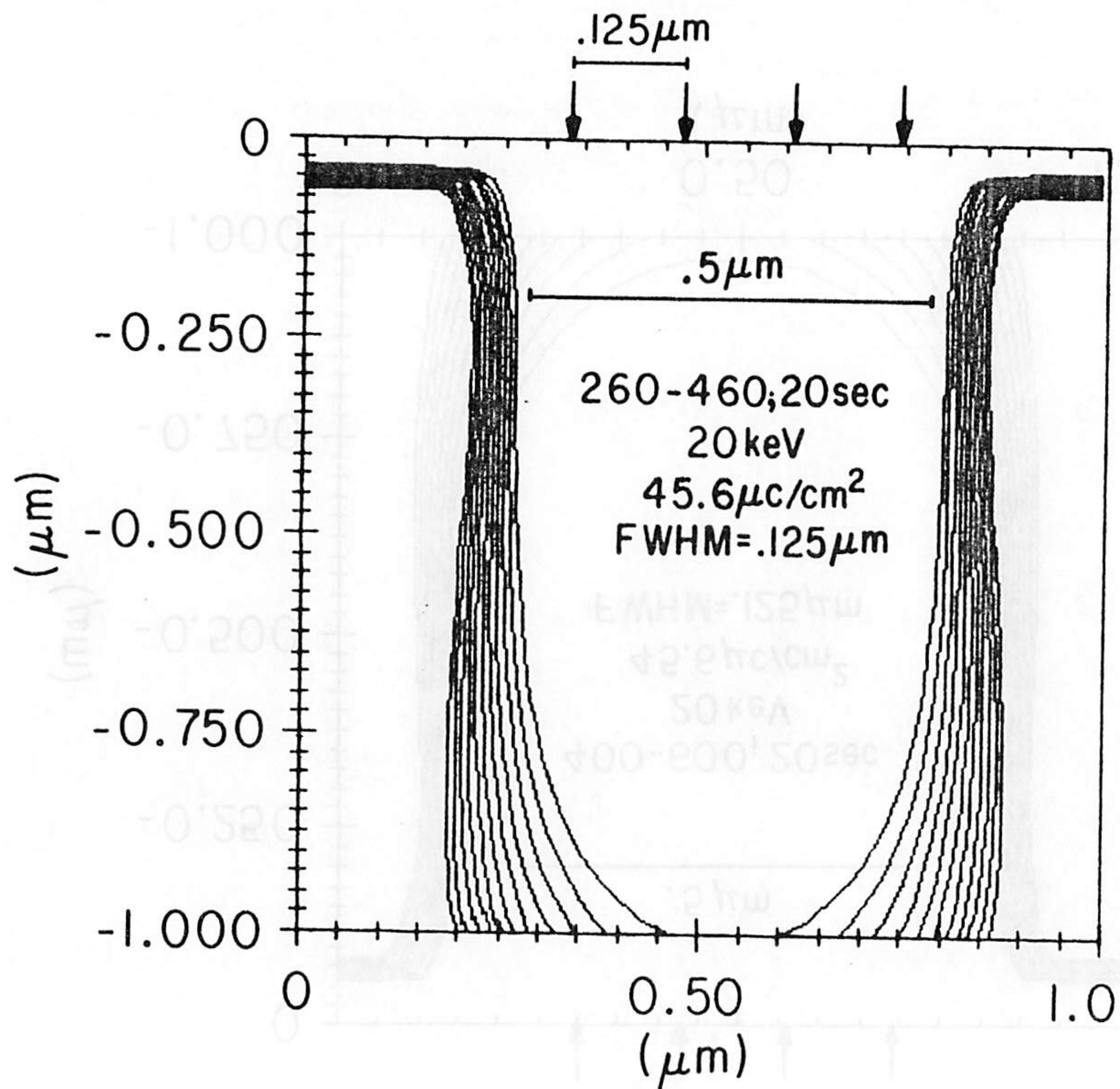


Figure 3b

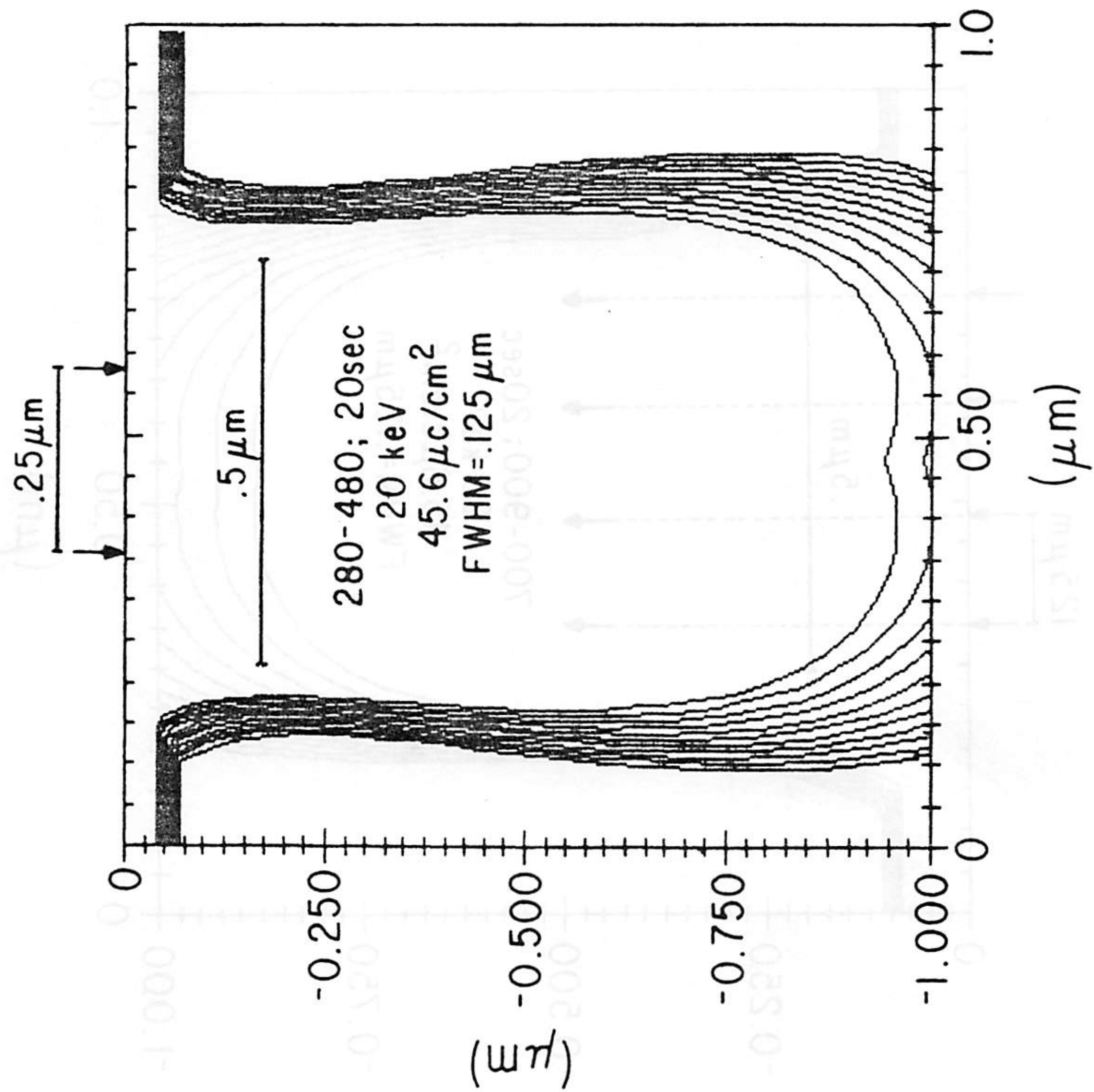


Figure 4

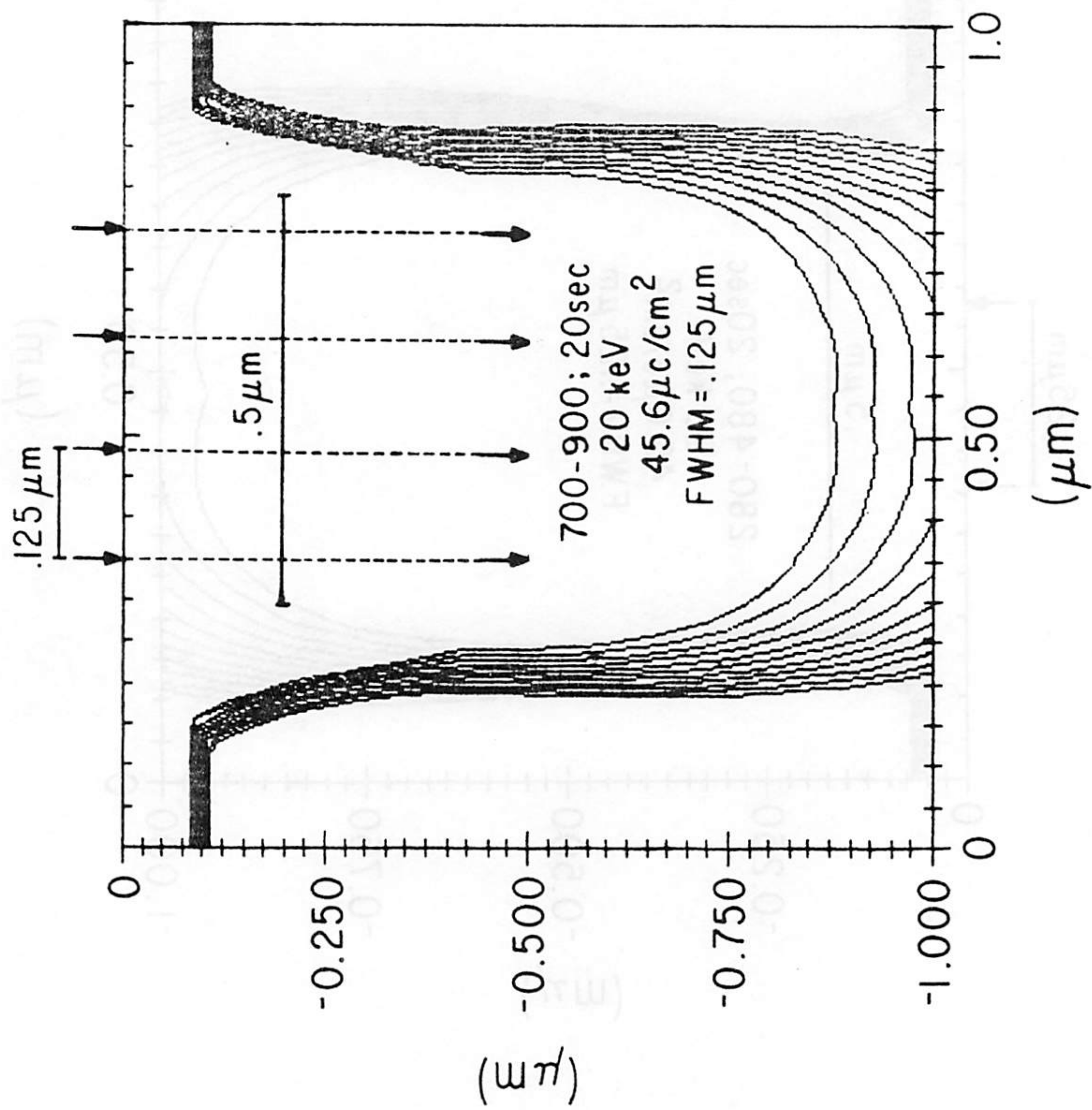


Figure 5a

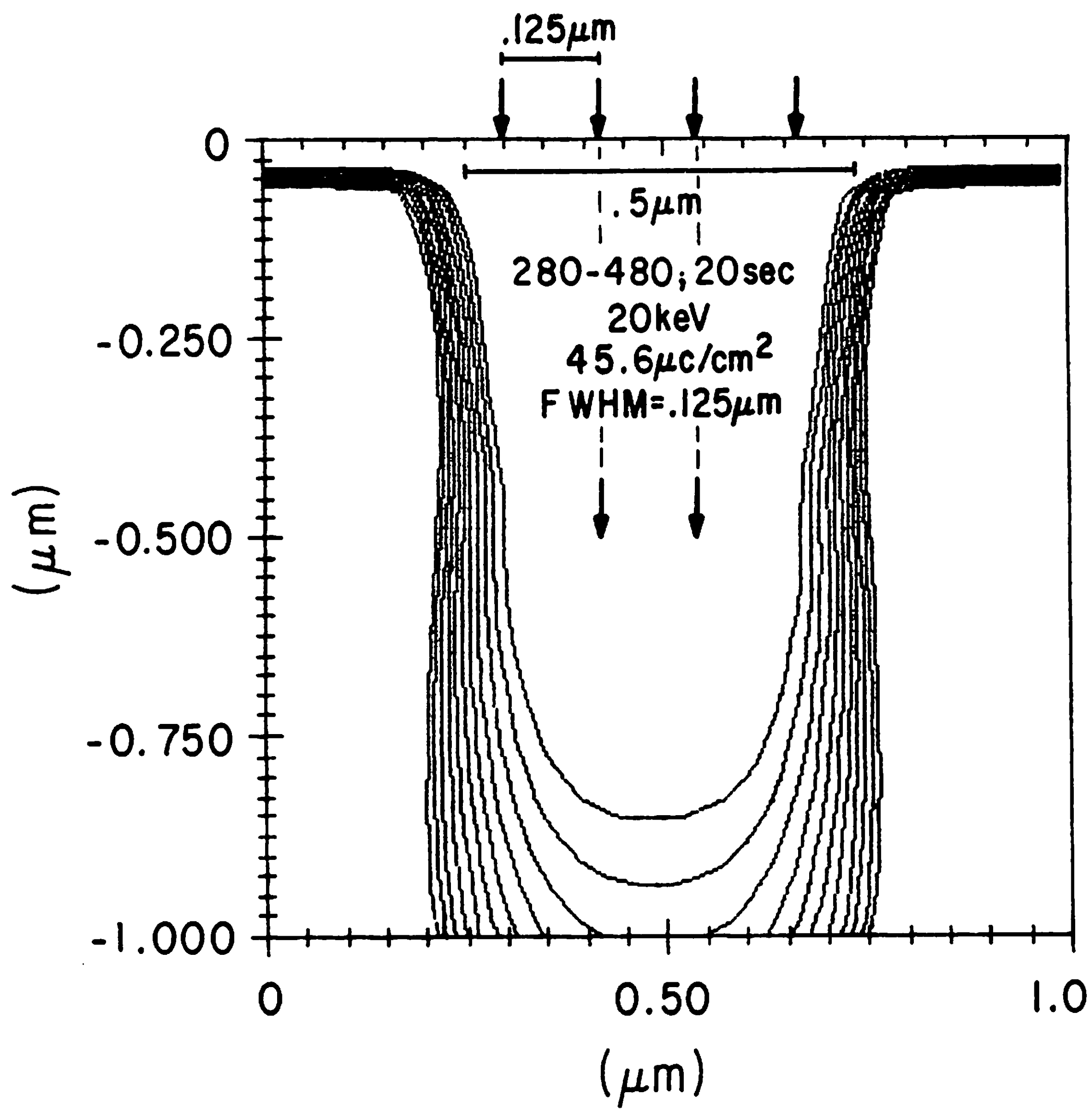


Figure 5b

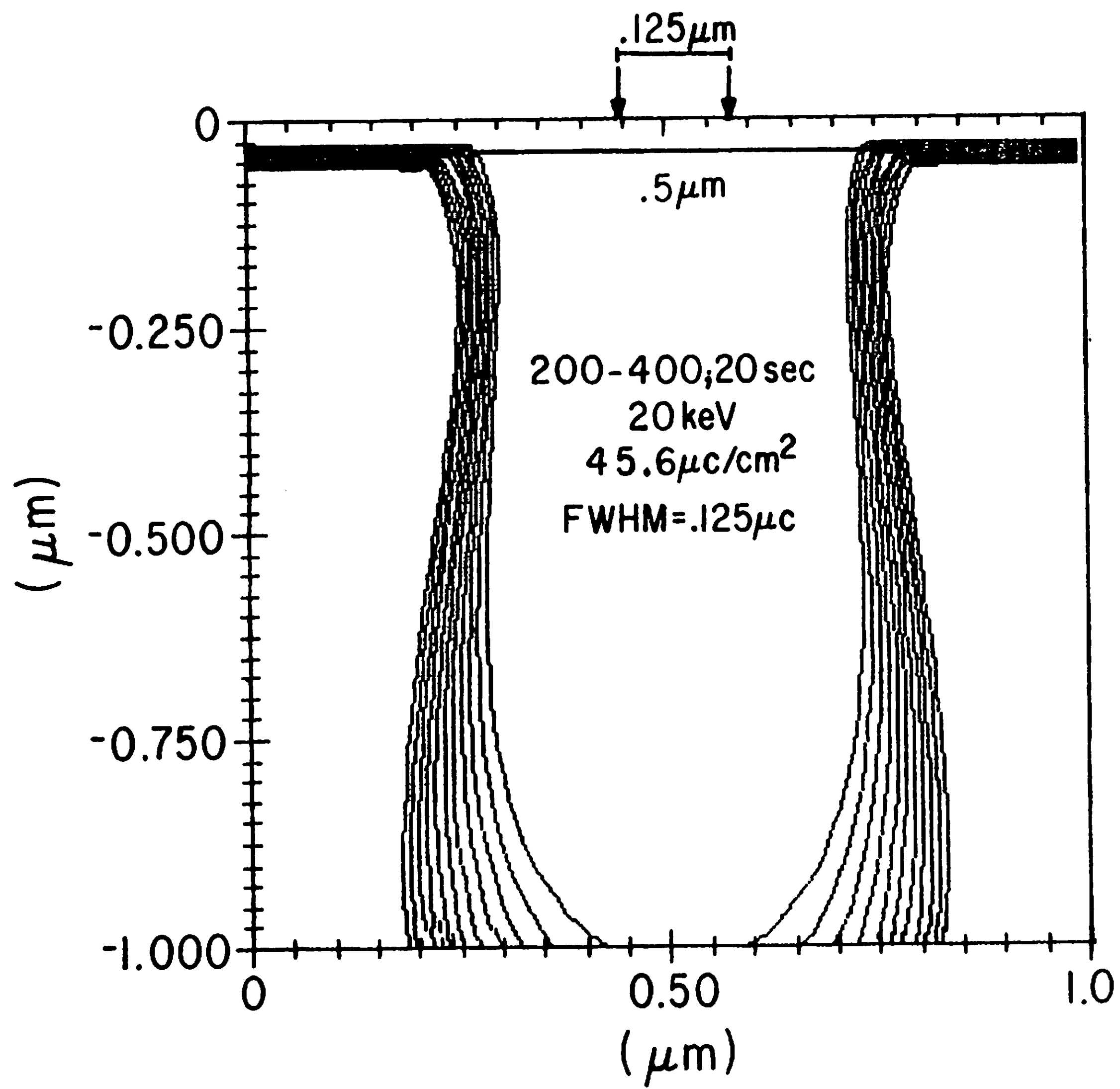


Figure 6

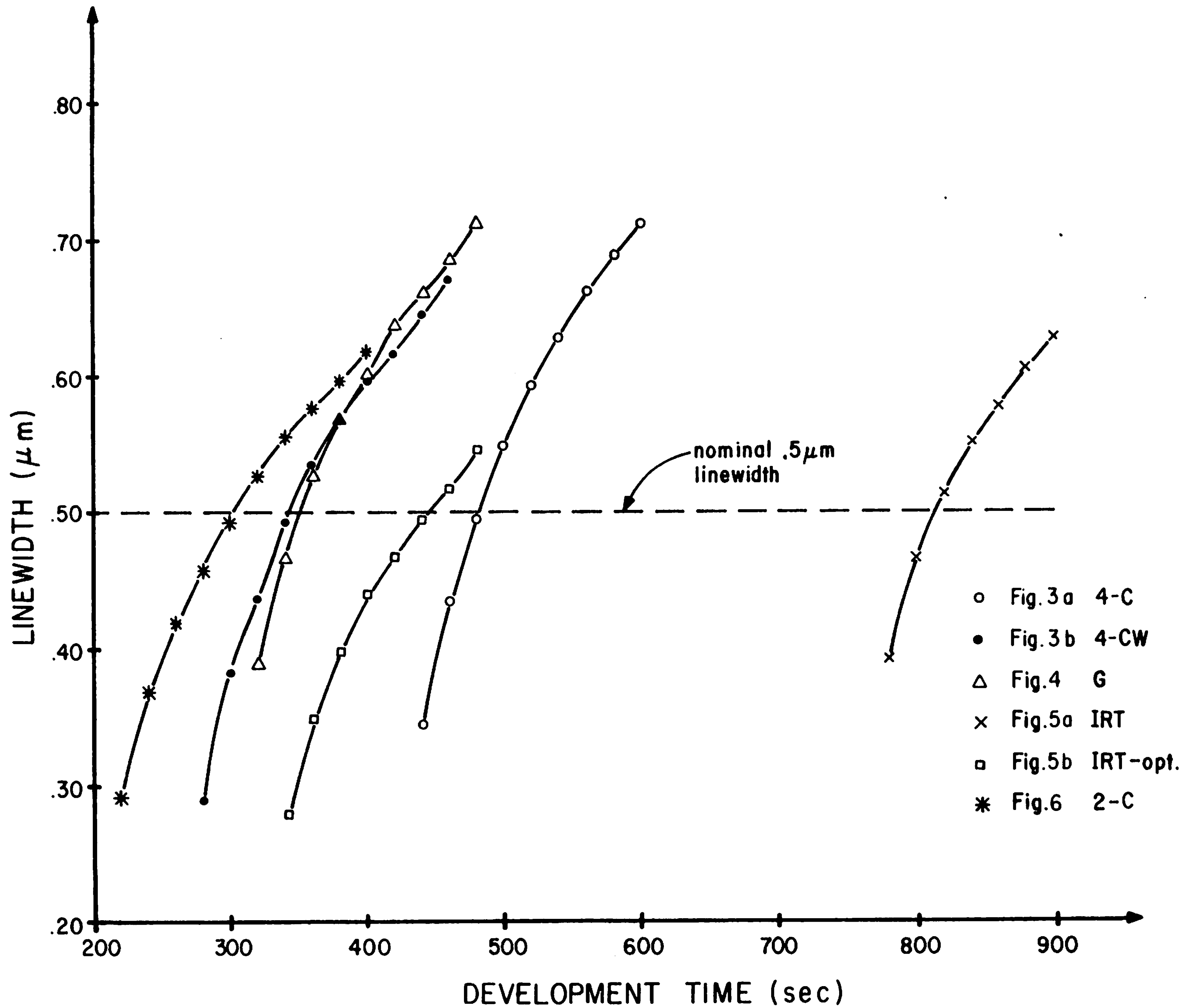


Figure 7

CASE	X(0) (μm)	X(.4) (μm)	X(.8) (μm)	T	C	B (μm)	$\partial(LW)/\partial(td)$ (μm/sec)	td (sec)	S	Δ_{Eff}	Q	FM
4 spot-conventional Figure 3a	.25	.35	.36	.14	.058	>.14	.0029	484	2.81	.20	4.1	.36
4 spot-weighted Figure 3b	.25	.32	.31	.081	.049	.074	.0021	345	1.45	.13	6.1	.69
2 spot-Greeneich Figure 4	.25	.33	.30	.068	.063	.080	.0026	350	1.82	.13	6.1	.55
4 spot-thinned Figure 5a	.24	.32	.37	.15	.020	>.13	.0021	825	3.47	.17	4.7	.29
Optimum-thinned Figure 5b	.25	.26	.27	.023	.009	.020	.0013	445	1.16	.032	25.8	.86
2 spot-conventional Figure 6	.25	.27	.24	-.012	.029	<.01	.0018	305	1.10	.041	19.5	.91

Table 1

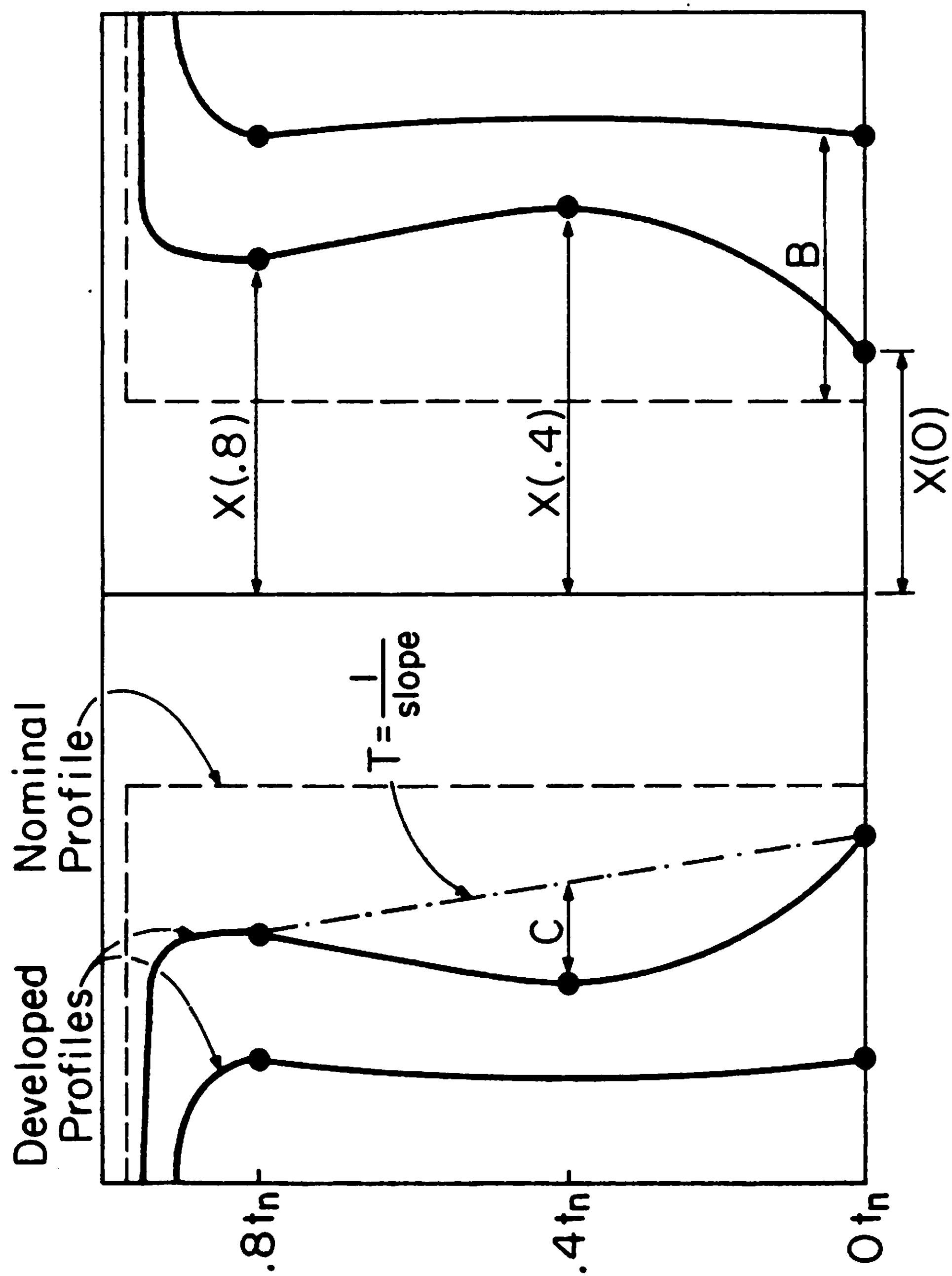


Figure 8

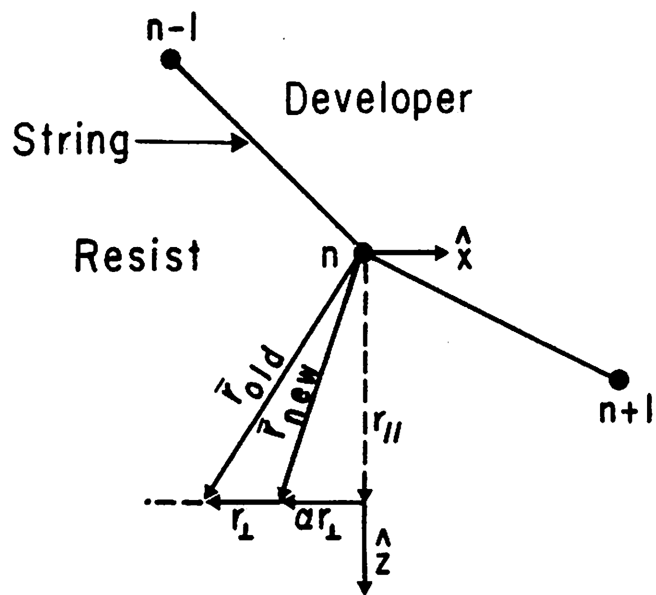
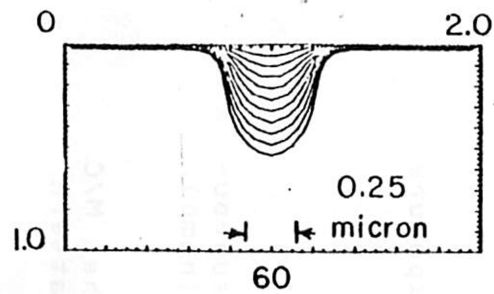
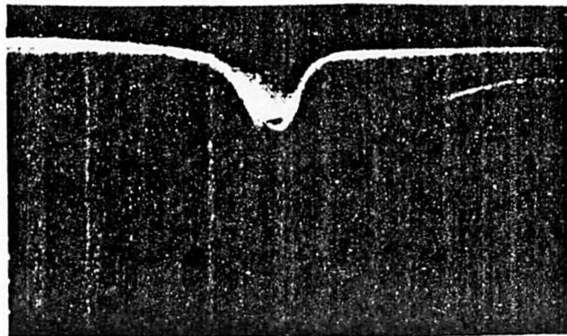
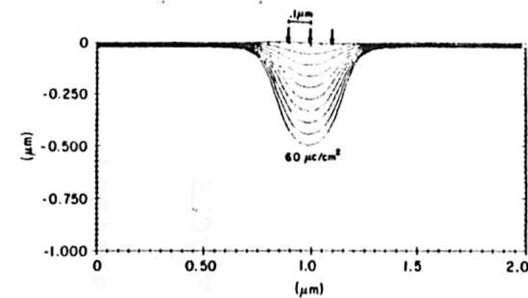
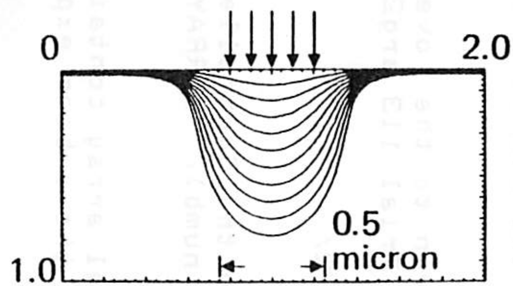


Figure 9

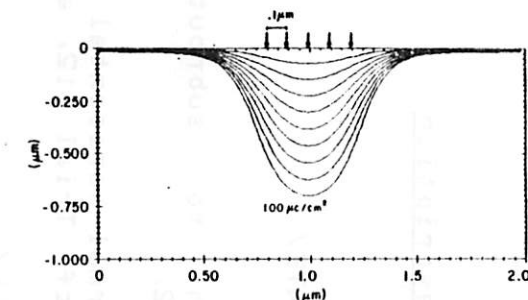


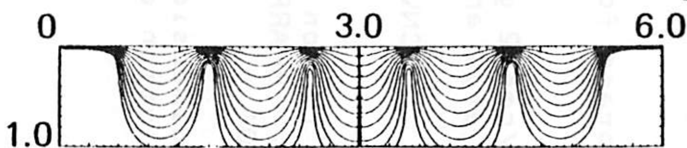




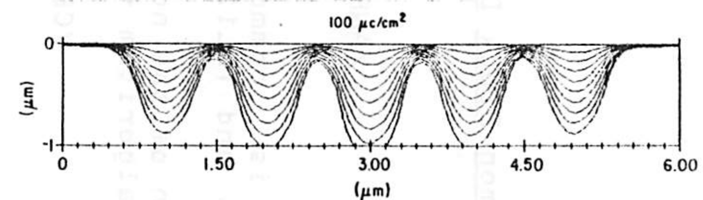


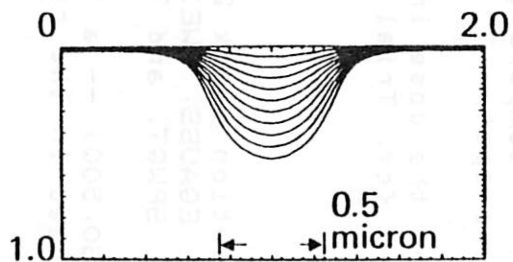
(a)



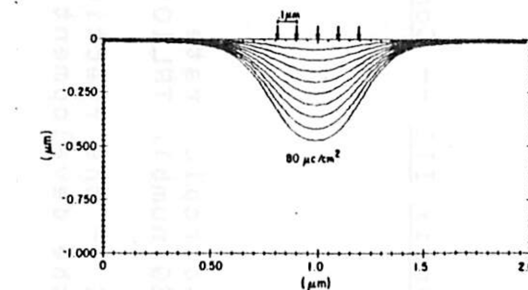


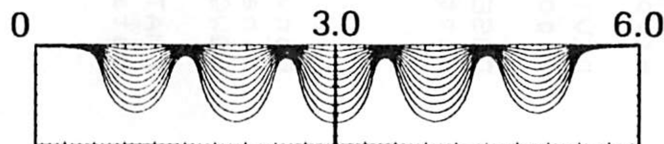
(b)





(c)





(d)

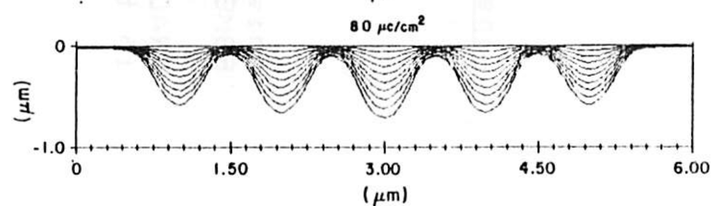


Figure 10

Appendix III -- Common Block Description

/ANRATE/

Anisotropic rate is common to subroutines ECYCLE, EBMSG(numb), TRL101, and TRL112.

FRAC -- the fraction of the normal horizontal motion desired in the development algorithm {cf. Trial 112, arg3}.

/CELL/

Cell size is common to the Controller and subroutines MLTLIN(numb), EBMSG(numb), EPLOT, SQWGT, SPWGT, and TRL101.

CELLX -- the M/C cell size in the horizontal x direction in microns.

CELLZ -- the M/C cell size in the vertical z direction in microns.

/CNVLV1/

Convolution block 1 is common to the Controller and subroutines MLTSPT, EGAUSS, WEIGHT, PRARRY, MLTLIN(numb), EARRAY, EBMSG(numb), SQWGT, SPWGT, TRL101, TRL111, and TRL113.

NCLMET -- the number of locations used in the array ETEM2(999).

DEVUNI -- temporary storage for the standard deviation of an exposure spot.

DEVTEM -- temporary storage for the standard deviation of an exposure spot.

DOSE -- the dose in uC/cm**2 given to the overall exposure pattern {cf. Trial 111 arg2 and Trial 113 arg2}.

/CNVLV2/

Convolution block 2 is common to the Controller and subroutines EGAUSS, WEIGHT, PRARRY(numb), EARRAY, EBMSG(numb), SQWGT, SPWGT, and TRL101.

EMAT(80,500) -- a two dimensional array containing the M/C data used in the convolution of the e-beam exposure pattern.

NUMSPT -- a counter used to keep track of the number of spots making up a line.

WGHTO -- temporary storage for the 'weight' given a spot.

NCEMAT -- the number of columns used in the array EMAT(80,500).

/COPYWD/

SAMPLE copy window [8] is common to the Controller and subroutines EBDEV, EPLOT, TRL101, and TRL110.

CPWIND -- the resist window of interest size in microns {cf. Trial 110 arg2}.

CPEDGE -- 1/2 CPWIND, used in SAMPLE plotting routines.

CPWORG -- not used by the e-beam machine.

/DEVFLG/

SAMPLE develop flag [8] is common to subroutines EBDEV and TRL101.

IDEVFL(5) -- a one dimensional array containing the flags for the development machine. 1 means 'yes' and 0 means 'no'. IDEVFL(1) -- print the number of string points, etc. and subroutine CHKR point totals? IDEVFL(2) -- print out points for each profile? IDEVFL(3) -- set equal to more points for more time-consuming but more accurate runs? {cf. Trial 2 arg1 arg2 arg3} IDEVFL(4) and IDEVFL(5) are not used by the e-beam machine.

/DEVTIM/

SAMPLE development times [8] is common to subroutines EBDEV, EBMSG(numb), and TRL101.

MXNDEV -- the maximum number of development contours that can be requested by the user -- currently 20 {cf. SAMPLE Key-word 'devtime' arg3}.

DEVSRT -- the development time in seconds of the first development contour {cf. SAMPLE key-word 'devtime' arg1}.

DEVEND -- the development time in seconds of the final development contour {cf. SAMPLE key-word 'devtime' arg2}

DEVINC -- the time in seconds between the intermediate development contours.

/DFALT1/

Default block 1 is common to the Controller and subroutines TRL101 and TRL110.

ISYM -- a flag in which '1' indicates a symmetrical development {cf. Trial 110 arg3}.

/DISTNC/

Distances is common to the Controller and subroutines MLTLIN(numb), EBMSG(numb), TRL101, TRL105, and TRL108.

SHFDIS(20) -- a one dimensional array containing the positions of the spots in an exposure line {cf. Trial 106 arg3 or Trial 107 arg3, arg6, etc.}.

DISLIN(20) -- a one dimensional array containing the positions of the lines in an exposure pattern {cf. Trial 108 arg3 or Trial 109 arg3, arg5, arg7, etc.}.

SPTWGT(20) -- a one dimensional array containing the weight for each spot in an exposure line {cf. Trial 106 arg5, arg6, etc. or Trial 107 arg5, arg8, etc.}.

WGTLIN(20) -- a one dimensional array containing the weight for each exposure line {cf. Trial 108 arg4, arg5, etc. or Trial 109 arg4, arg6, etc.}.

STDDEV(20) -- a one dimensional array containing the standard deviation of each spot in an exposure line {cf. Trial 106 arg4 or Trial 107 arg4, arg7, etc.}.

ITCOU -- the number of spots in an exposure line {cf. Trial 106 arg2 or Trial 107 arg2}.

/DVELP1/

SAMPLE development block 1 [8] is common to subroutines EBDEV, ECYCLE, TRL101, TRL112, and function EBRATE(cz).

XZ(1000) -- complex array containing the x and z positions of the 1000 possible development string points.

XMAX -- the real (not including boundary) maximum value of x in microns.

ZMAX -- the real maximum value of z in microns.

NPTS -- the current number of development string points {cf. Trial 112 arg2}.

CXZL -- the normalized left endpoint direction of development.

CXZR -- the normalized right endpoint direction

NADCHK -- the number of advances (calls on subroutine ECYCLE) between checks (calls on SAMPLE subroutine CHKR [8]).

NCKOUT -- the number of checks between outputs; the number of advances between outputs is NADCHK*NCKOUT.

/DVELP2/

SAMPLE development block 2 [8] is common to subroutines EBCYCLE and EBDEV.

TADV -- the current time between advances (for subroutine ECYCLE).

TCHK -- the current time between checks.

TTOT -- contains the total current development time during the execution of the SAMPLE-E-BEAM development machine.

IFLAG -- flag set in ECYCLE that tells EBDEV that some string segments are too long or too short. SAMPLE subroutine CHKR is then called from EBDEV.

SMAXX -- the maximum x string segment length allowed by CHKR (and ECYCLE).

SMINX -- the minimum x string segment length allowed by CHKR.

SMAXZ -- the maximum z string segment length allowed by CHKR.

/DVELP3/

SAMPLE development block 3 [8] is common subroutines ECYCLE and EBDEV.

NZFLG -- a flag used between subroutines EBDEV and ECYCLE in order to signal when the string has broken through the resist. The number of checks between outputs is then reduced.

NADFLG -- the output number of the string that broke through the resist.

/DVELP4/

SAMPLE development block 4 [8] is common to subroutine EBDEV.

BREAK -- the estimated time until resist breakthrough.

MAXPTS -- when the number of string points exceeds MAXPTS, SAMPLE subroutine DELOOP is called.

NADSAV -- contains the NADCHK before resist breakthrough.

NCKSV1 -- contains NCKOUT before resist breakthrough for the first development output.

NCKSV2 -- contains NCKOUT before resist breakthrough for the intermediate outputs.

NOUT -- the number of development outputs requested {cf. SAMPLE Key-word 'devtime' arg3}.

/ENGPLT/

Energy plotting is common to subroutines EBMSG(numb), EPLOT, TRL101, TRL115.

IDEP -- the first row (not including boundary) of the ELIN(82,1002) array to be printed {cf. Trial 115 arg3}.

ISKIP -- the skip number of rows of ELIN(82,1002) to be printed (i.e. rows IDEP, IDEP+ISKIP, IDEP+ISKIP+ISKIP, etc. are printed out) {cf. Trial 115 arg4}.

IR1 -- a flag in which a 1 indicates that in the output of the energy plotting option {cf. Trial 115}, the value of the maximum absorbed energy density in the desired rows of ELIN(82,1002) is printed in the output file. If IRI is any other positive integer number, that number is the maximum absorbed energy density printed. This maximum energy density is printed before the actual energy density values and is only used to give a range of the energy density for plotting purposes {cf. Trial 115 arg2}.

ENGMAX -- a maximum value of absorbed energy density printed for plotting purposes.

/HORIMG/

SAMPLE horizontal image [8] is common to the Controller, subroutine EBDEV, and function EBRATE.

DELTX -- the size of the M/C cell in the horizontal x direction in microns.

NMHPTS -- the number of columns (including boundary) used in the array ELIN(82,1002) to describe the absorbed energy density in the window of interest.

MNHPTS -- not used by the e-beam machine.

HORINT(50) -- not used by the e-beam machine.

/IO1/

SAMPLE input-output block 1 is common to the Controller and subroutines PRARRY(numb), MLTLIN(numb), EBDEV, EBMSG(numb), SQWGT, SPWGT, and all TRLxxx subroutines.

All of the variables in this common block are assigned numbers appropriate to the computer system in use and are used in input-output. For example, 'WRITE(IPRINT,#)', 'READ(IBULK,#) variable' are used to output and input information and data.

/LINE1/

Line block 1 is common to the Controller and subroutines PRARRY(numb), MLTLIN(numb), EARRAY, EBDEV, EBMSG(numb), BOUNDR, EPLOT, TRL101, TRL108, TRL113, and function EBRATE(cz).

ELIN(82,1002) -- a two dimensional array containing the absorbed energy density in the resist window of interest due to a specified electron-beam exposure.

LCOU -- the number of lines in the exposure pattern {cf. Trial 108 arg2 or Trial 109 arg2}.

LINCOU -- also the number of lines in the exposure pattern (redundancy due to changes in the original e-beam program; the variable was left in due to the author's laziness).

ELNWGT(1999) -- a one dimensional array containing the total exposure pattern needed to determine the absorbed energy density in the window of interest.

/LINE2/

Line block 2 is common to the Controller and subroutines PRARRY(numb), MLTLIN(numb), EARRAY(numb), BOUNDR, EBMSG(numb), EPLOT, TRL101, and TRL113.

NCELIN -- the maximum number of columns which can be used in the array ELIN(82,1002).

NUMCOL -- the number of array locations in EMLT(1499) used.

MAXCOL -- the number of columns (not including boundary) of ELIN(82,1002) used (Note that MAXCOL is used for a different purpose in the Controller -- to determine the location of the left window of interest edge for a symmetric development).

NCEMLT -- the maximum number of array locations in EMLT(1499) which can be used.

NCLELU -- the number of columns (including boundary) used in the array ELIN(82,1002).

NRELIN -- the number of rows in the array ELIN(82,1002) used.

/MCARLO/

Monte Carlo information is common to the Controller and subroutine EBMSG(numb).

THICK -- the thickness of the resist in microns.

EVENG -- the energy of the e-beam in KeV.

/PARSEM/

SAMPLE parsem is common to subroutine EXTRA6 and all TRLxxx subroutines except TRL101.

STNMLS(100) -- a one dimensional array which contains the values of each of the arguments in a Trial statement or Key-word.

NMINST -- the number of arguments set by the user in a Trial statement or Key-word.

ISTMTY -- not used in the e-beam program.

ISTKND -- not used in the e-beam program.

NMPNTR -- not used in the e-beam program.

/PRFLG/

Printing flags is common to the Controller and subroutines PRARRY(numb), WEIGHT, SQWGT, TRL101, and TRL102.

IWFLG(5) -- a one dimensional array containing the flags (1=yes) which influence the output of information and

messages by the program {cf. Trial 102 with arg2-6 corresponding to IWFLG(1-5)}.

/RATDAT/

Rate equation data is common to subroutines EBMSG(numb), TRL101, TRL104, and function EBRATE(cz).

(R1*CM) -- the background etch-rate in A/sec {cf. Trial 104 arg2 arg3}.

DO -- a reference or knee energy {cf. Trial 104 arg4}.

ALPH -- the asymptotic slope of the etch-rate versus absorbed energy density curve for the resist at high dose {cf. Trial 104 arg5}.

/SDIST/

Shift distance is common to the Controller and subroutines MLTLIN(numb), EBMSG(numb), and TRL110.

SHIFT -- the distance in microns of the first spot or rectangular beam from the left window edge {cf. Trial 110 arg4}.

/SIMPAR/

SAMPLE simulation parameters [8] is common to the Controller, subroutine EBDEV, and function EBRATE(cz).

NPRLYR -- the number of layers the resist is to be divided into (i.e. the number of rows used in ELIN(82,1002) not including boundary).

NPRPTS -- not used in the e-beam program.

NENDIV -- not used in the e-beam program.

DELTM -- not used in the e-beam program.

DELTZ -- the size of the M/C cell in the vertical z direction in microns.

/SPOT1/

Spot block 1 is common to the Controller and subroutines MLTSPT, EGAUSS, WEIGHT, PRARRY(numb), MLTLIN(numb), EARRAY, SPWGT, and SQWGT.

ETEM2(999) -- a one dimensional array containing the

exposure pattern profile of a single Gaussian spot or rectangular shaped electron-beam.

EMLT(1499) -- a one dimensional array containing the exposure pattern profile for a single line.

/SPOT2/

Spot block 2 is common to the Controller and subroutines MLTSPT, EGAUSS, WEIGHT, PRARRY(numb), MLTLIN(numb), EARRAY, EBMSG(numb), TRL101, TRL115, SPWGT, and SQWGT.

ISHIFT -- an integer counter used to keep track of the distance in array units between spots in a line.

WGHT1 -- temporary storage for the 'weight' given a spot.

NRHCET -- the number of array locations used in the right hand side (not including center) of ETEM2(999).

NREMAT -- the number of rows used in the M/C array EMAT(80,500).

/SQBEAM/

Square beam is common to the Controller and subroutines SQWGT, TRL101, and TRL105.

ISQ -- a flag in which 1 indicates a rectangular beam exposure line.

FWHM -- the full-width half maximum of the rectangular beam in microns {cf. Trial 105 arg2}.

EDGE -- the edge-width of the rectangular beam in microns {cf. Trial 105 arg3}.

Appendix IV -- Program Code

----------INPUT---------

```
c SUBROUTINE EXTRIA calls the e-beam trial statements
c
      subroutine extria(iflag)
      common /parsem/ istmty,istknd,stnmls(100),nminst,nmpntr
c
      itest=int(stnmls(1))
      if (itest .ne. 101) go to 102
c trial 101 initializes default  parameters for e-beam simulation
      call trl101
      return
c
102   continue
      if (itest .ne. 102) go to 104
c trial 102 initializes printing flags for arrays,messages
      call trl102
      return
c
c103  continue
c     if (itest .ne. 103) go to 104
c trial 103 tells program which M/C file to use
c     call trl103
c     return
c
104   continue
      if (itest .ne. 104) go to 105
c trial 104 initializes rate eqn. parameters
      call trl104
      return
c
105   continue
      if (itest .ne. 105) go to 106
c trial 105 is for square shaped beams
      call trl105(1)
      return
c
106   continue
      if (itest .ne. 106) go to 107
c trial 106 is for gaussian shaped beams-periodic
      call trl105(2)
      return
c
107   continue
      if (itest .ne. 107) go to 108
c trial 107 is for gaussian shaped beams-non-periodic
      call trl105(3)
      return
```

```
c
108   continue
      if (itest .ne. 108) go to 109
c trial 108 is for periodic line arrays
      call trl108(1)
      return
c
109   continue
      if (itest .ne. 109) go to 110
c trial 109 is for non-periodic line arrays
      call trl108(2)
      return
c
110   continue
      if (itest .ne. 110) go to 111
c trial 110 sets type of develop, window size, and position in window
c if non-sym. develop
      call trl110
      return
c
111   continue
      if (itest .ne. 111) go to 112
c trial 111 sets the exposure dose and runs the convolution
      call trl111
      return
c
112   continue
      if (itest .ne. 112) go to 113
c trial 112 sets dev. times, npts, anrate fraction
      call trl112
      return
c
113   continue
      if (itest .ne. 113) go to 114
c trial 113 sets new dose for elin array
      call trl113
      return
c
114   continue
      if (itest .ne. 114) go to 115
c trial 114 runs the develop programs
      call trl114
      return
c
115   continue
      if (itest .ne. 115) go to 116
c trial 115 is the energy profile option
      call trl115
      return
c
116   continue
      iflag=0
      return
      end
```

```
c
c  SUBROUTINE TRL101 initializes default parameters
c
      subroutine trl101
      common /spot2/ ishift,wght1,nrhcet,nremat
      common /line2/ ncelin,numcol,maxcol,ncemlt,nclelu,nrelin
      common /cnvlv1/ nclmet,devuni,devtem,dose
      common /cnvlv2/ emat (80,500),numspt,wght0,ncemat
      common /ratdat/ r1,cm,d0,alph
      common /distnc/ shfdis(20),dislin(20),sptwgt(20),wgtlin(20),
     *                stddev(20),itcou
      common /prflg/ iwflg(5)
      common /io1/ itermi,ibulk,iprout,iresv1,iin,iprint,ipunch
      common /line1/ elin (82,1002),lcou,lincou,elnwgt (1999)
      common /devflg/ idevfl(5)
      common /devtim/ mxndev,devsrt,devend,devinc
      common /dvelp1/ cxzl,cxzr,xz(1000),xmax,zmax,npts,nadchk,nckout
      common /copywd/ cpwind,cpedge,cpworg
      common /anrate/ frac
      common /sqbeam/ isq,fwhm,edge
      common /dfalt1/ isym
      common /engplt/ idep,iskip,ir1,engmax
      integer d0
      complex xz,cxzl,cxzr
      iprint=6
c
c this is the default initialization routine
c set flags to print out only info.
      iwflg(1)=0
      iwflg(2)=0
      iwflg(3)=0
      iwflg(4)=0
      iwflg(5)=1
c rate curve data
      r1=1.0
      cm=1.0
      d0=199
      alph=2.0
c dose
      dose=80.0
c square beam option
      isq=1
      fwhm=1.0
      edge=.25
c periodic line option
      dislin(1)=0.0
      dislin(2)=2.0
      dislin(3)=4.0
      wgtlin(1)=1.0
      wgtlin(2)=1.0
      wgtlin(3)=1.0
      lincou=3
c develop flags
      idevfl(1)=0
```

```
      idevfl(2)=1
      idevfl(3)=0
      idevfl(4)=0
      idevfl(5)=0
c develop times
      devsrt=40.0
      devend=160.0
      devinc=40.0
      npts=60
c symmetric develop option
      isym=1
      cpwind=2.0
      frac=1.0
c energy plotting option
      idep=1
      iskip=7
      ir1=1
      call ebmsg(23)
      write (iprint,10)
10    format(1x,33hE-beam default values initialized/)
      return
      end
```

```
      SUBROUTINE TRL102
      common /prflg/ iwflg(5)
      common /parsem/ istmty,istknd,stnmls(100),nminst,nmpntr
      common /io1/ itermi,ibulk,iprout,iresv1,iin,iprint,ipunch
c
c this trial sets the e-beam printing flags for arrays, info.
c
      if (nminst .ge. 2 .and. nminst .le. 6) go to 20
      write(iprint,10)
10    format(1x,39hERROR-Trial 102 requires a minimum of 2,
     *        39h and a maximum of 6 arguments to change,/
     *        1x,32hthe default ebeam printing flags/)
      return
c
20    continue
      go to (1,1,2,3,4,5) nminst
5     iwflg(5)=int(stnmls(6))
4     iwflg(4)=int(stnmls(5))
3     iwflg(3)=int(stnmls(4))
2     iwflg(2)=int(stnmls(3))
1     iwflg(1)=int(stnmls(2))
      return
      end
c
      SUBROUTINE TRL104
      common /ratdat/ r1,cm,d0,alph
      common /parsem/ istmty,istknd,stnmls(100),nminst,nmpntr
      common /io1/ itermi,ibulk,iprout,iresv1,iin,iprint,ipunch
      integer d0
c
c this trial sets rate eqn. constants
c
      if (nminst .ge. 2 .and. nminst .le. 5) go to 20
      write(iprint,10)
10    format(1x,39hERROR-Trial 104 requires a minimum of 2,
     *        39h and a maximum of 5 arguments to change,/
     *        1x,36hthe default rate equation parameters/)
      return
c
20    continue
      go to (1,1,2,3,4) nminst
4     alph=stnmls(5)
3     d0=int(stnmls(4))
2     cm=stnmls(3)
1     r1=stnmls(2)
      return
      end
c
      SUBROUTINE TRL105(numb)
      common /parsem/ istmty,istknd,stnmls(100),nminst,nmpntr
      common /sqbeam/ isq,fwhm,edge
      common /distnc/ shfdis(20),dislin(20),sptwgt(20),wgtlin(20),
     *                stddev(20),itcou
      common /io1/ itermi,ibulk,iprout,iresv1,iin,iprint,ipunch
```

```
c
c this subroutine sets the incident exposure line shape (trials 105,
c 106, 107)
c
c initialize arrays to zero
      do 95 i=1,20
         shfdis(i)=0.0
         sptwgt(i)=0.0
         stddev(i)=0.0
95    continue
      go to (1,2,3) numb
c
c sq beam option
1     if (nminst .ge. 2 .and. nminst .le. 3) go to 20
      write(iprint,10)
10    format(1x,39hERROR-Trial 105 requires a minimum of 2,
     *       39h and a maximum of 3 arguments to change,/
     *       1x,34hthe default rectangular beam shape/)
      return
c
20    continue
      go to (4,4,5) nminst
5     if (stnmls(3) .le. 0.0) write(iprint,19)
19    format(1x,48hERROR-Trial 105-an edge width less than or equal,/
     *       49h to 0.0 is not allowed for rectangular beam shape/)
      if (stnmls(3) .le. 0.0) return
      edge=stnmls(3)
4     fwhm=stnmls(2)
      isq=1
      return
c periodic Gaussian option
2     if (nminst .ge. 6) go to 23
      write(iprint,21)
21    format(1x,39hERROR-Trial 106 requires a minimum of 6,
     *       37h arguments(i.e. at least 2 spots) for,/
     *       1x,38hthe periodic gaussian beamshape option/)
      return
23    itcou=int(stnmls(2))
      if (itcou .le. 20)  go to 25
      write(iprint,24)
24    format(1x,40hERROR-Trial 106-periodic gaussian option/)
      call ebmsg(7)
      return
25    if ((nminst-4) .eq. itcou) go to 30
      write(iprint,11)
11    format(1x,39hERROR-Trial 106 requires specifying the,
     *       37h same number of weights and spots for,/
     *       1x,38hthe periodic gaussian beamshape option/)
      return
30    if (stnmls(3) .ge. 0.0) go to 33
      write(iprint,24)
      write(iprint,26)
26    format(1x,31hnegative spot shift not allowed/)
      return
```

```
33     isq=0
       call ebmsg(8)
       sper=stnmls(3)
       sigma=stnmls(4)
c
c make shfdis,stddev,weight arrays
       stddev(1)=sigma
       sptwgt(1)=stnmls(5)
       do 99 n=2,itcou
          stddev(n)=sigma
          shfdis(n)=shfdis(n-1)+sper
          sptwgt(n)=stnmls(n+4)
99     continue
       return
c
c non-periodic Gaussian option
3      if (nminst .ge. 5) go to 34
       write(iprint,32)
32     format(1x,39hERROR-Trial 107 requires a minimum of 5,
      *       31h arguments for the non-periodic,/
      *       1x,25hgaussian beamshape option/)
       return
34     itcou=int(stnmls(2))
       if (itcou .le. 20) go to 35
       write(iprint,31)
31     format(1x,44hERROR-Trial 107-non-periodic gaussian option/)
       call ebmsg(7)
       return
35     itemp=(itcou*3)+2
       if (itemp .eq. nminst) go to 40
       write(iprint,12) itemp,itcou
12     format(1x,25hERROR-Trial 107 requires ,i2,10h arguments,
      *       16h to specify the ,i2,16h spots requested,/
      *       1x,45hin the non-periodic gaussian beamshape option/)
       return
40     itemp=itcou*3
       do 190 n=3,itemp,3
          if (stnmls(n) .ge. 0.0) go to 190
          write(iprint,31)
          write(iprint,26)
          return
190    continue
       i=3
       isq=0
       do 199 n=1,itcou
          shfdis(n)=stnmls(i)
          stddev(n)=stnmls(i+1)
          sptwgt(n)=stnmls(i+2)
          i=i+3
199    continue
       if (shfdis(1) .eq. 0.0) return
       write(iprint,50)
50     format(1x,46hWARNING-Trial 107-non-periodic gaussian option,/
      *       1x,34hfirst spot does not start at x=0.0/)
```

```
      return
      end
c
      SUBROUTINE TRL108(numb)
      common /parsem/ istmty,istknd,stnmls(100),nminst,nmpntr
      common /distnc/ shfdis(20),dislin(20),sptwgt(20),wgtlin(20),
     *                stddev(20),itcou
      common /line1/ elin (82,1002),lcou,lincou,elnwgt (1999)
      common /io1/ itermi,ibulk,iprout,iresv1,iin,iprint,ipunch
c
c this subroutine sets the positions of the exposure lines (trials
c 108 and 109)
c
      go to (1,2) numb
c
c periodic line array option
1     if (nminst .ge. 5) go to 5
      write(iprint,4)
4     format(1x,39hERROR-Trial 108 requires a minimum of 5,
     *        33h arguments(i.e. at least 2 lines),/
     *        1x,28hfor the periodic line option/)
      return
5     lincou=int(stnmls(2))
      if (lincou .ne. 0) go to 20
      write(iprint,10)
10    format(1x,42hERROR-Trial 108-periodic line array option/)
      call ebmsg(11)
      lincou=3
      return
20    if (lincou .le. 20) go to 30
      write(iprint,10)
      call ebmsg(12)
      lincou=3
      return
30    if ((nminst-3) .eq. lincou) go to 40
      write(iprint,13)
13    format(1x,39hERROR-Trial 108 requires specifying the,
     *       37h same number of weights and lines for,/
     *       1x,24hthe periodic line option/)
      lincou=3
      return
c
40    if (stnmls(3) .ge. 0.0) go to 41
      write(iprint,10)
      write(iprint,12)
12    format(1x,31hnegative line shift not allowed/)
      lincou=3
      return
41    call ebmsg(13)
      do 95 i=1,20
         dislin(i)=0.0
         wgtlin(i)=0.0
95    continue
      shfper=stnmls(3)
```

```
c
c put periodic line shifts in dislin(i), weights into wgtlin(i)
      do 99 n=2,lincou
         dislin(n)=dislin(n-1)+shfper
         wgtlin(n-1)=stnmls(n+2)
99    continue
      wgtlin(lincou)=stnmls(lincou+3)
      return
c
c non-periodic line option
2     if (nminst .ge. 4) go to 45
      write(iprint,43)
43    format(1x,39hERROR-Trial 109 requires a minimum of 4,
     *       31h arguments for the non-periodic,/
     *       1x,11hline option/)
      return
45     lincou=int(stnmls(2))
      if (lincou .ne. 0) go to 44
      write(iprint,42)
42    format(1x,46hERROR-Trial 109-non-periodic line array option/)
      call ebmsg(11)
      lincou=3
      return
44    if (lincou .le. 20) go to 48
      write(iprint,42)
      call ebmsg(12)
      lincou=3
      return
48    itemp=(lincou*2)+2
      if (itemp .eq. nminst) go to 50
      write(iprint,14)itemp,lincou
14    format(1x,25hERROR-Trial 109 requires ,i2,10h arguments,
     *       16h to specify the ,i2,16h lines requested,/
     *       1x,31hin the non-periodic line option/)
      lincou=3
      return
c
50    itemp=3 + (lincou-1)*2
      do 190 n=3,itemp,2
         if (stnmls(n) .ge. 0.0) go to 190
         write(iprint,42)
         write(iprint,12)
         lincou=3
         return
190   continue
      do 96 i=1,20
         dislin(i)=0.0
         wgtlin(i)=0.0
96    continue
      i=3
      do 199 n=1,lincou
         dislin(n)=stnmls(i)
         wgtlin(n)=stnmls(i+1)
         i=i+2
```

```
199   continue
      if (dislin(1) .eq. 0.0) return
      write(iprint,15)
15    format(1x,42hWARNING-Trial 109-non-periodic line option/)
      call ebmsg(14)
      return
      end
c
      SUBROUTINE TRL110
      common /parsem/ istmty,istknd,stnmls(100),nminst,nmpntr
      common /dfalt1/ isym
      common /sdist/ shift
      common /copywd/ cpwind,cpedge,cpworg
      common /io1/ itermi,ibulk,iprout,iresv1,iin,iprint,ipunch
c
c trial 110 sets the type of development (symmetric or not), window
c size, and 'shift' ( the distance first spot is from lhs window
c edge- if non-sym dev. specified)
c
      if (nminst .ge. 2 .and. nminst .le. 4) go to 20
      write(iprint,10)
10    format(1x,39hERROR-Trial 110 requires a minimum of 2,
     *       39h and a maximum of 4 arguments to change,/
     *       1x,30hthe default development window/)
      return
c
20    continue
      go to (1,1,2,2) nminst
2     isym=int(stnmls(3))
      if (isym .eq. 0 .and. nminst .ne. 4) write(iprint,25)
25    format(1x,34hERROR-Trial 110-development window/
     *       1x,43hnon-symmetric development requested without,
     *       17h specifying shift/)
      if (isym .eq. 0 .and. nminst .ne. 4) isym=1
      if (isym .eq. 0 .and. nminst .ne. 4) return
      if (isym .eq. 0) shift=stnmls(4)
1     cpwind=stnmls(2)
      return
      end
c
      SUBROUTINE TRL111
      common /parsem/ istmty,istknd,stnmls(100),nminst,nmpntr
      common /cnvlvl/ nclmet,devuni,devtem,dose
      common /io1/ itermi,ibulk,iprout,iresv1,iin,iprint,ipunch
c
c this subroutine sets the exposure dose and runs the convolution
c
      if (nminst .le. 2) go to 20
      write(iprint,10)
10    format(1x,39hERROR-Trial 111 requires a minimum of 1,
     *       36h and a maximum of 2 arguments to run,/
     *       1x,41hthe convolution and set the exposure dose/)
      return
c
```

```
20    continue
      go to (1,2) nminst
2     dose=stnmls(2)
1     call ebctrl(1)
      return
      end
c
      SUBROUTINE TRL112
      common /parsem/ istmty,istknd,stnmls(100),nminst,nmpntr
      common /dvelp1/ cxzl,cxzr,xz(1000),xmax,zmax,npts,nadchk,nckout
      common /anrate/ frac
      common /io1/ itermi,ibulk,iprout,iresv1,iin,iprint,ipunch
      complex xz,cxzl,cxzr
c
c this subroutine sets npts and anrate fraction
c
      if (nminst .ge. 2 .and. nminst .le. 3) go to 20
      write(iprint,10)
10    format(1x,39hERROR-Trial 112 requires a minimum of 2,
     *       39h and a maximum of 3 arguments to change,/
     *       1x,20hnumber of string pts,
     *       27h and anisotropic dev option/)
      return
c
20    continue
      go to (1,1,2) nminst
2     frac=stnmls(3)
1     npts=int(stnmls(2))
      return
      end
c
      SUBROUTINE TRL114
c
c trial 114 runs the develop routines
c
      call ebctrl(2)
      return
      end
c
      SUBROUTINE TRL115
      common /engplt/ idep,iskip,ir1,engmax
      common /parsem/ istmty,istknd,stnmls(100),nminst,nmpntr
      common /io1/ itermi,ibulk,iprout,iresv1,iin,iprint,ipunch
      common /spot2/ ishift,wght1,nrhcet,nremat
c
c this subroutine initializes and runs the energy contour option
c
      if (nminst .le. 4) go to 20
      write(iprint,10)
10    format(1x,39hERROR-Trial 115 requires a minimum of 1,
     *       36h and a maximum of 4 arguments to run,/
     *       1x,25hthe energy contour option/)
      return
c
```

```
20    continue
      go to (1,2,3,4) nminst
4     iskip=int(stnmls(4))
3     idep=int(stnmls(3))
      if (idep .le. nremat) go to 2
      write(iprint,30) idep
30    format(1x,37hERROR-Trial 115-energy contour option,/
     *       1x,15hfirst depth of ,i2,18h rows out of range,
     *          12h of M/C data/)
      iskip=39
      idep=1
      return
2     itest=int(stnmls(2)+.0001)
      if (itest .eq. 1) ir1=1
      if (itest .ne. 1) engmax=stnmls(2)
      if (itest .ne. 1) ir1=0
1     call eplot
      call ebmsg(22)
      return
      end
c
      SUBROUTINE TRL113
      common /cnvlv1/ nclmet,devuni,devtem,dose
      common /line1/ elin (82,1002),lcou,lincou,elnwgt (1999)
      common /line2/ ncelin,numcol,maxcol,ncemlt,nclelu,nrelin
      common /parsem/ istmty,istknd,stnmls(100),nminst,nmpntr
      common /io1/ itermi,ibulk,iprout,iresv1,iin,iprint,ipunch
c
c Trial 113 allows the user to change the overall exposure dose
c without having to reconvolve, the default dose is also set to
c the new inputted value
c
      if (nminst .eq. 2) go to 20
      write(iprint,10)
10    format(1x,39hERROR-Trial 113 requires 2 arguments to,
     *       33h change the overall exposure dose/)
      return
c
20    cdose=stnmls(2)/dose
      do 99 i=1,nrelin
         do 199 j=1,ncelin
            elin(i,j)=elin(i,j)*cdose
199      continue
99    continue
      call ebmsg(23)
      write(iprint,30) stnmls(2)
30    format(1x,39hnew dose for final energy array is now ,
     *       f6.2,9h uc/cm**2/)
      dose=stnmls(2)
      return
      end
```

----------CONTROLLER----------

```
c SUBROUTINE EBCTRL is the controlling subroutine which runs the
c convolution(1) and development(2)
c
      subroutine ebctrl(numb)
c
      common /spot1/ etem2 (999),emlt (1499)
      common /spot2/ ishift,wght1,nrhcet,nremat
      common /line1/ elin (82,1002),lcou,lincou,elnwgt (1999)
      common /line2/ ncelin,numcol,maxcol,ncemlt,nclelu,nrelin
      common /cnvlv1/ nclmet,devuni,devtem,dose
      common /cnvlv2/ emat (80,500),numspt,wght0,ncemat
      common /cell/ cellx,cellz
      common /prflg/ iwflg(5)
      common /io1/ itermi,ibulk,iprout,iresv1,iin,iprint,ipunch
      common /distnc/ shfdis(20),dislin(20),sptwgt(20),wgtlin(20),
     *                stddev(20),itcou
      common /sdist/ shift
      common /sqbeam/ isq,fwhm,edge
      common /dfalt1/ isym
      common /horimg/ deltx,mnhpts,nmhpts,horint(50)
      common /simpar/ nprlyr,nprpts,nendiv,deltm,deltz
      common /copywd/ cpwind,cpedge,cpworg
      common /mcarlo/ thick,eveng
c
      data ptfiv /.5000000/
c
      iprint=6
      go to (1,2) numb
c
c
c  read in M/C data
1     ibulk=2
      open (ibulk,file='mcdat')
      rewind ibulk
c
c read in resist thickness, energy, units to convert to J/cm**3 and cell
c sizes
c
      read (ibulk, 10) thick,eveng,units,cellx,cellz,nremat,ncemat
10    format (f6.4,/f6.3,/e10.4,/f6.4,/f6.4,/i2,/i3)
c
c check input emat size
      if (nremat .gt. 80) call ebmsg(5)
      if (ncemat .gt. 500) call ebmsg(6)
c
c if flag set, write out this info.
      if (iwflg(5) .eq. 1) call ebmsg(20)
c
c read in M/C data into array 'emat'
      read (ibulk,155) ((emat (i,j), j=1,ncemat),i=1,nremat)
```

```
155   format (8e10.4)
      close (ibulk)
c
      if (iwflg(5) .eq. 1) call ebmsg(4)
c
c change emat by multiplying by units and dose
c
      do 111 i=1,nremat
         do 211 j=1,ncemat
               emat(i,j)=emat(i,j)*dose*units
211      continue
111   continue
c
c if flag set, write out emat
c
      if (iwflg(4) .eq. 1) call prarry(4)
c
c set variables numspt,maxcol,lcou,islar, nrhcet, nclmet,array sizes
c
      maxcol=0
      lcou=1
      numspt=0
      islar=0
      nrhcet=ncemat-1
      nclmet=2*nrhcet+1
      nrelin=nremat+2
c allow for a 5 um spot shift in emlt
      item=int(5.000001/cellx) + nclmet
      if (item .gt. 1499) ncemlt=1499
      if (item .le. 1499) ncemlt=item
c allow for a 10-20 um window depending on cell size
      if (cellx .lt. .02) ncelin=int (10.00001/cellx)+2
      if (cellx .ge. .02) ncelin=int (20.00001/cellx)+2
c
      if (iwflg(5) .eq. 1) call ebmsg(1)
c
c initialize emlt,elin to zero
c
        do 97 j=1,ncemlt
         emlt (j) = 0.0
97      continue
       do 88 i=1,nrelin
            do 98 j=1,ncelin
               elin (i,j) = 0.0
98          continue
88     continue
c
      if (isq .eq. 1) call sqwgt
      if (isq .eq. 1) go to 600
c else.......
c form pattern of spots for convolution
c
      devuni=stddev(1)
      devtem=devuni
```

```
      wght1=sptwgt(1)
c
c call 'spwgt' to compute weights needed in convolution
c
      call spwgt
c call 'egauss' to add first spot to emlt
c
      call egauss
c
c now do the other spots, if any
      if (itcou .eq. 1) go to 510
      do 408 i=2,itcou
         numspt=numspt+1
         wght0=sptwgt(i)
c
c convert microns to units of distance = 1
         s2=shfdis(i)/cellx
         devtem=stddev(i)
c
c round off distances to nearest cell
c
         a1=aint(s2)
         q1=s2-a1
         if (q1 .ge. ptfiv) s2=a1+1.
         if (q1 .lt. ptfiv) s2=a1
         shfdis(i)=s2*cellx
         ishift=int(s2+.0001)
c
c check if shift distance is too large,write error message if necessary
c
      maxshf=ishift+nclmet
      if (maxshf .gt. ncemlt) call ebmsg(9)
c
c put largest value of ishift in islar
c
      if (ishift .gt. islar) islar=ishift
c
c if present spot's std. dev. is same as preceeding spot;
c but, weight constants are different, call subroutine weight
c
      if (wght0 .ne. wght1 .and. devtem .eq. devuni) call weight
      wght1=wght0
c
c if present spot's std. dev. is different from preceeding spot, a
c new spot exposure pattern array must be calculated
c
      if (devtem .ne. devuni) call spwgt
      if (devtem .ne. devuni) call egauss
c
c call mltspt to add new gaussian to emlt
c
      call mltspt
c
408   continue
```

```
c
c write out info. on spots
510   if (iwflg(5) .eq. 1) call ebmsg(10)
c
c round off line shifts to nearest cell
600   do 420 i=2,lincou
         s2=dislin(i)/cellx
         a1=aint(s2)
         q1=s2-a1
         if (q1 .ge. ptfiv) s2=a1+1.
         if (q1 .lt. ptfiv) s2=a1
         dislin(i)=s2*cellx
420   continue
c
c
c set numcol to # of columns emlt occupies
c
      numcol=islar+nclmet
      lcou=lincou
c initialize parameters needed to compute final 'elin' array
c
c  if sym develop.....
      if (isym .ne. 1) go to 200
      maxcol=numcol+int((dislin(lcou)+.0001)/cellx)
      itemp=maxcol/2
      wintem=float(itemp)*cellx
c
      shift=float(nrhcet)*cellx-wintem
c
c for non-sym develop cpwind should be divisible by cell size in x
200   cpedge=cpwind/2.
      icpwin=int((cpwind+.0001)/cellx)
      if (icpwin .gt. (ncelin-2)) call ebmsg(16)
      nclelu=icpwin+2
      shift=shift+float(nrhcet)*cellx
c
c call different parts of mltlin depending on whether shift
c is positive or negative
c
      if (shift .ge. 0.0) call mltlin(1)
      if (shift .lt. 0.0) call mltlin(2)
c print out data for lines
c
      if (iwflg(5) .eq. 1) call ebmsg(15)
c
c convolute pattern in window of interest
c
      call earray
      if (iwflg(1) .eq. 1 .and. numspt .ne. 0) call prarry(1)
      if (iwflg(1) .eq. 1 ) call prarry(5)
c
c reset shift to print out correct positions of lines in
c user-defined window
c
```

```
      shift=shift-float(nrhcet)*cellx
      if (iwflg(5) .eq. 1) call ebmsg(17)
c
      return
c
c ebctrl(2) controls the develop routines
c first, add boundary layer to elin array
c
2     call boundr
c
c if flag set, write out elin
c
      if (iwflg(2) .eq. 1) call prarry(2)
c
c initialize develop parameters
      deltx=cellx
      deltz=cellz
      nprlyr=nrelin-2
      mxndev=20
      nmhpts=nclelu
c develop exposed resist.
c call subroutine ebdev to develop.
c
      if (isym .eq. 1) call ebmsg(2)
      if (iwflg(5) .eq. 1) call ebmsg(24)
      call ebdev
      return
      end
```

----------CONVOLVER----------

```
c
c FUNCTION ERF calculates the error function
c
      function erf(y)
      data a1,a2/7.05230784e-2,4.22820123e-2/
      data a3,a4/9.2705272e-3,1.520143e-4/
      data a5,a6/2.765672e-4,4.30638e-5/
      s=a6*y+a5
      s=s*y+a4
      s=s*y+a3
      s=s*y+a2
      s=s*y+a1
      s=s*y+1.
      s=s**16
      erf=1-(1./s)
      return
      end
c
c SUBROUTINE MLTSPT creates a multiple spot pattern array
c by adding the contribution from the spot described in 'etem2'
c to the multiple spot pattern array 'emlt'
c
      subroutine mltspt
      common /spot1/ etem2 (999),emlt (1499)
      common /spot2/ ishift,wght1,nrhcet,nremat
      common /cnvlv1/ nclmet,devuni,devtem,dose
c
c
       kmm1=ishift + 1
       kpm1=nclmet + ishift
       ic=1
         do 48 j=kmm1,kpm1
         emlt (j) = etem2 (ic) + emlt (j)
         ic=ic+1
48       continue
       return
       end
c
c SUBROUTINE EGAUSS completes the computation of the single
c Gaussian of square beam pattern array 'etem2'
c
      subroutine egauss
      common /cnvlv1/ nclmet,devuni,devtem,dose
      common /cnvlv2/ emat (80,500),numspt,wght0,ncemat
      common /spot1/ etem2 (999),emlt (1499)
      common /spot2/ ishift,wght1,nrhcet,nremat
      common /prflg/ iwflg(5)
c
c set devuni=devtem so that controller can decide whether a
c new spot needs to be computed
```

```
c
      devuni=devtem
c
c
c the following flips rhs of etem2 to lhs to get complete gaussian
c or square beam
c
      itemp=nclmet
        do 67 j=1,nrhcet
         etem2 (j) = etem2 (itemp)
         itemp=itemp-1
67      continue
c
c if flag set, write out etem2
c
      if (iwflg(3) .eq. 1) call prarry(3)
c
c if numspt greater than 0, return - only want to read
c etem2 into emlt for gaussian # 0
c
      if (numspt .gt. 0) return
c
c read in etem2 into emlt, for gaussian 0 only
c
        do 82 j=1,nclmet
         emlt (j)=etem2 (j)
82      continue
c
      return
      end
c
c
c SUBROUTINE WEIGHT changes wght1 constant of etem2
c it assumes devuni stays the same, then writes out new
c array by calling subprogram prarry (if flag is set)
c
      subroutine weight
      common /cnvlv1/ nclmet,devuni,devtem,dose
      common /cnvlv2/ emat (80,500),numspt,wght0,ncemat
      common /spot1/ etem2 (999),emlt (1499)
      common /spot2/ ishift,wght1,nrhcet,nremat
      common /prflg/ iwflg(5)
c
        do 222 j=1,nclmet
           etem2(j)=etem2(j)*(wght0/wght1)
222     continue
      wght1=wght0
      if (iwflg(3) .eq. 1) call prarry(3)
      return
      end
c
c SUBROUTINE MLTLIN(numb) creates the array 'elnwgt' which
c contains the exposure pattern which will be convolved to
c determine the energy absorbed in the user-specified window
```

```
c of interest
c
      subroutine mltlin(numb)
      common /line1/ elin (82,1002),lcou,lincou,elnwgt (1999)
      common /cnvlv1/ nclmet,devuni,devtem,dose
      common /line2/ ncelin,numcol,maxcol,ncemlt,nclelu,nrelin
      common /spot1/ etem2 (999),emlt (1499)
      common /spot2/ ishift,wght1,nrhcet,nremat
      common /cell/ cellx,cellz
      common /distnc/ shfdis(20),dislin(20),sptwgt(20),wgtlin(20),
     *                stddev(20),itcou
      common /sdist/ shift
      common /io1/ itermi,ibulk,iprout,iresv1,iin,iprint,ipunch
c
      max=(nclelu-2) + nclmet
      do 10 i=1,max
         elnwgt(i) = 0.0
10    continue
      go to (1,2) numb
1     ipass=1
      dipass=0.0000
      tshif=shift
c
c for positive shift, first spot is to right of left window edge:
c
3     do 9 k=ipass,lcou
c
c istar is a column counter used to add emlt to elin only in window
c of interest
       istar=nrhcet+1-int((dislin(k)+tshif+.0001-dipass)/cellx)
c
c do not add any more lines if they won't contribute in window
c
         if ( istar .ge. (-max+2)) go to 8
         call ebmsg(21)
         write(iprint,200) k
200      format(1x,7hline # ,i2,19h and greater do not,
     *            37h contribute in the window of interest/)
         return
8           item=istar
            do 20 j=1,max
c
c if istar > # col in emlt, do next line
c if istar is neg, do not add, just increase istar by 1
c
            if (item .gt. numcol) go to 9
            if (item .le. 0) go to 19
            elnwgt(j)=emlt(item)*wgtlin(k)+elnwgt(j)
19          item=item+1
20          continue
9     continue
            return
c
c the following is for neg shift - first spot is to left of left
```

```
c window edge
c
2        pshift=abs(shift)
       do 109 k=1,lcou
c go to next line if it will not add in window
         itest=nrhcet+1+int((pshift-dislin(k)+.0001)/cellx)
          if (itest .le. numcol) go to 105
          call ebmsg(21)
          write(iprint,201) k
201       format(1x,7hline # ,i2,20h does not contribute,
    *              22h in window of interest/)
          go to 108
c add lines using part (1) if the following
105       if ((shift+dislin(k)) .ge. 0.0) go to 100
             ittest=itest
             do 120 j=1,max
c go to next line if ittest becomes>numcol
             if (ittest .gt. numcol) go to 108
           elnwgt(j)=emlt(ittest)*wgtlin(k)+elnwgt(j)
             ittest=ittest+1
120          continue
108    ipass=k+1
109    continue
c
c use first part of mltlin to add lines which start to right of
c left window edge
100    if(ipass .gt. lcou) return
c else:
       tshif=dislin(ipass)-pshift
c must change so that part 1 see's correct dislin(k)
       dipass=dislin(ipass)
       go to 3
       end
c
c SUBROUTINE BOUNDR adds boundary rows and columns
c to elin by extrapolation
c
       subroutine boundr
       common /line1/ elin (82,1002),lcou,lincou,elnwgt (1999)
       common /line2/ ncelin,numcol,maxcol,ncemlt,nclelu,nrelin
c
c initialize subscripts
c
       nvp0=nrelin-2
       nvp1=nrelin-1
       nvp2=nrelin
       npp1=nclelu
       npp2=npp1-1
       npp3=npp1-2
c
c do left side column - starting with elin(2,1)
c
       do 10 iz=2,nvp1
          elin(iz,1)=elin(iz,2)+elin(iz,2)-elin(iz,3)
```

```
c
c if < 0, set equal to 0
c
            if (elin(iz,1) .lt. 0.) elin(iz,1)=0.
c
c now do right column-starting with elin(2,npp1)
c
            elin(iz,npp1)=elin(iz,npp2)+elin(iz,npp2)-elin(iz,npp3)
            if (elin(iz,npp1) .lt. 0.) elin(iz,npp1)=0.
10      continue
c
c do top and bottom rows of array
c
        do 20 ix=1,npp1
            elin(1,ix)=elin(2,ix)+elin(2,ix)-elin(3,ix)
            if (elin(1,ix) .lt. 0.) elin (1,ix)=0.
            elin(nvp2,ix)=elin(nvp1,ix)+elin(nvp1,ix)-elin(nvp0,ix)
            if (elin(nvp2,ix) .lt. 0.) elin(nvp2,ix)=0.
20      continue
        return
        end
c
c SUBROUTINE EARRAY computes the final 2-D energy absorption array,
c 'elin', by convolving the exposure pattern contained in 'elnwgt'
c with a delta function exposure contained in 'emat'
c
        subroutine earray
        common /spot1/ etem2 (999),emlt (1499)
        common /spot2/ ishift,wght1,nrhcet,nremat
        common /line1/ elin (82,1002),lcou,lincou,elnwgt (1999)
        common /line2/ ncelin,numcol,maxcol,ncemlt,nclelu,nrelin
        common /cnvlv1/ nclmet,devuni,devtem,dose
        common /cnvlv2/ emat (80,500),numspt,wght0,ncemat
c
c maxcol is the number of columns which make up
c elin (excluding boundary)
c
        maxcol=nclelu-2
c
c first do contributions from delta functions to the left of the
c window of interest
c find where to start convolution
c
        do 10 i=1,ncemat
             if (elnwgt(i) .eq. 0.0) go to 10
             imark=i
             go to 20
10      continue
c
c
c if imark=0, then there will be no contribution from this side since
c all the weights are zero
c
        go to 30
```

```
c
c the following convolves contributions from the left of the window
c of interest in the window
c
20     do 99 i=1,nremat
          do 199 j=imark,ncemat
             item=ncemat-j+1
             l=1
             do 299 k=item,ncemat
c
c go to next delta function if contribution past right window edge
c
                if (l .gt. maxcol) go to 199
                elin(i+1,l+1)=emat(i,k)*elnwgt(j)+elin(i+1,l+1)
                l=l+1
299           continue
199       continue
99     continue
c
c now do contributions from right of window after first finding where
c to start
c
30     imark=0
       item=0
       istr=ncemat+maxcol
       iend=maxcol+nclmet
       ix=iend+1
       do 40 i=istr,iend
          item=item+1
          ix=ix-1
          if (elnwgt(ix) .eq. 0.0) go to 40
          imark=item
          go to 45
40     continue
c
c again, skip if all weights are equal to 0.0
c
        go to 50
c
45      itcol=nclmet+maxcol+1
       do 399 i=1,nremat
          do 499 j=imark,ncemat
             item=ncemat-j+1
             l=maxcol
             do 599 k=item,ncemat
c
c if contribution goes past left window edge, do next delta function
c
                if (l .lt. 1) go to 499
                elin(i+1,l+1)=emat(i,k)*elnwgt(itcol-j)+elin(i+1,l+1)
                l=l-1
599          continue
499       continue
399    continue
```

```
c
c now convolve in window of interest
c set start and end points for elnwgt
c
50    istart=ncemat+1
      iend=nrhcet+maxcol
c
      do 201 i=1,nremat
         do 202 j=istart,iend
c
c skip if weight equals 0.0
c
            if (elnwgt(j) .eq. 0.0) go to 202
c
c set ending position for convolution in window
c
            ilfcon=j-ncemat
c
c do left of delta function
c
            ix=ilfcon
            do 204 k=1,ilfcon
c
c  decrease ix if greater than ncemat
c
               if (ix .gt. ncemat) go to 203
               elin(i+1,k+1)=emat(i,ix)*elnwgt(j)+elin(i+1,k+1)
203            ix=ix-1
204         continue
c
c do right of delta function
            ix=1
            iscon=ilfcon+1
            do 205 l=iscon,maxcol
c
c if out of M/C range do nest delta function
c
               if (ix .gt. ncemat) go to 202
               elin(i+1,l+1)=emat(i,ix)*elnwgt(j)+elin(i+1,l+1)
               ix=ix+1
205         continue
202      continue
201   continue
      return
      end
```

```
c SUBROUTINE SQWGT calculates the weights needed to convolute
c under a square beam profile. The array 'etem2' describes the
c square beam profile
c
      subroutine sqwgt
      common /cell/ cellx,cellz
      common /spot1/ etem2 (999),emlt (1499)
      common /spot2/ ishift,wght1,nrhcet,nremat
      common /cnvlv2/ emat (80,500),numspt,wght0,ncemat
      common /cnvlv1/ nclmet,devuni,devtem,dose
      common /io1/ itermi,ibulk,iprout,iresv1,iin,iprint,ipunch
      common /prflg/ iwflg(5)
      common /sqbeam/ isq,fwhm,edge
      dimension y(500)
      data sqpi2,sq2/2.506628,1.414214/
c
      do 10 j=1,nclmet
          etem2(j)=0.0
10    continue
c
      a=fwhm/2.0
      aincr=cellx
      sigma=edge/sqpi2
c
      numwgt=int((fwhm/aincr)+.0001)
c
c if numwgt is greater than the # of available m/c data columns
c write out warning
c
      if (numwgt .lt. ncemat) go to 16
      call ebmsg(21)
      write(iprint,15)
15    format(1x,43hWARNING, fwhm may exceed available m/c data/)
16    twait=0.0
      xx=0.0
c
      do 99 n=1,ncemat
         val1=(a-xx)/(sigma*sq2)
         val2=(a+xx)/(sigma*sq2)
         if (abs(val1) .gt. 4.0) go to 300
         t1=sign(erf(abs(val1)),val1)
100      if (abs(val2) .gt. 4.0) go to 400
         t2=sign(erf(abs(val2)),val2)
200      y(n)=t1+t2
         xx=xx+aincr
         twait=twait+y(n)
         go to 99
300      t1=sign(1.0,val1)
         go to 100
400      t2=sign(1.0,val2)
         go to 200
99    continue
c
      temwgt=twait-y(1)
```

```
      totwgt=(2.0*temwgt)+y(1)
c
c note that do loop only clculates rhs of weights
c
c      write(iprint,70) totwgt
c70    format(1x,38hThe sum total of wgts over the beam = ,f9.4/)
c
c  figure out rhs waits plus middle
c
      do 333 n=1,ncemat
           etem2(n+nrhcet)=y(n)*fwhm/totwgt
333   continue
c
c      write(iprint,75)(etem2(i),i=ncemat,nclmet)
c75    format(11(1x,f6.4))
c
      if (iwflg(5) .eq. 1) write(iprint,80)fwhm,edge
80    format(1x,24hsquare beam------fwhm = ,f6.4,8h microns/,
     *       1x,24h            edge width = ,f6.4,8h microns/)
      call egauss
      return
      end
c
c SUBROUTINE SPWGT finds weights for convolving Gaussian spots.
c The array 'etem2' describes the Gaussian shaped beam profile
c
      subroutine spwgt
      common /spot1/ etem2 (999),emlt (1499)
      common /spot2/ ishift,wght1,nrhcet,nremat
      common /cell/ cellx,cellz
      common /cnvlv1/ nclmet,devuni,devtem,dose
      common /cnvlv2/ emat (80,500),numspt,wght0,ncemat
      common /io1/ itermi,ibulk,iprout,iresv1,iin,iprint,ipunch
c
      data sq2 /1.414214/
c
c initialize etem2 to zero
c
      do 55 n=1,nclmet
         etem2(n)=0.0
55    continue
c
c the stepping distance must be the same as the cell size
c
      xx=cellx
      aincr=cellx/2.0
      do 99 n=2,ncemat
         aph=(xx+aincr)/(devtem*sq2)
         if (aph .gt. 5.0) go to 199
         amh=(xx-aincr)/(devtem*sq2)
         etem2(n+nrhcet)=((erf(aph)-erf(amh))/2.0)*wght1
         xx=xx+cellx
c         write(iprint,13)etem2(n+nrchet)
c13       format(1x,f12.10)
```

```
99        continue
c
c find middle weight
199   amh=aincr/(devtem*sq2)
      etem2(ncemat)=erf(amh)*wght1
c      write(iprint,13) etem2(ncemat)
c
c make sure that etem2(nclmet) is zero or else the stddev of the
c Gaussian may have been too large
      if (etem2(nclmet) .lt. .0001) return
      call ebmsg(21)
      write(iprint,20)
20    format(1x,43hWARNING-stddev of spot may exceed available,
     *       9h m/c data/)
      return
      end
c
```

----------E-BEAM DEVELOPER---------

```
      subroutine ebdev
      common /devflg/ idevfl(5)
      common /devtim/ mxndev,devsrt,devend,devinc
      common /dvelp1/ cxzl,cxzr,xz(1000),xmax,zmax,npts,nadchk,nckout
      common /dvelp2/ tadv,tchk,ttot,iflag,smaxx,sminx,smaxz
      common /dvelp3/ nzflg,ttotsv
      common /dvelp4/ break,maxpts,nadsav,ncksv1,ncksv2,nout
      common /horimg/ deltx,mnhpts,nmhpts,horint(50)
      common /io1   / itermi,ibulk,iprout,iresv1,iin,iprint,ipunch
      common /simpar/ nprlyr,nprpts,nendiv,deltm,deltz
      common /copywd/ cpwind,cpedge,cpworg
      common /line1/ elin (82,1002),lcou,lincou,elnwgt (1999)
      complex xz,cxzl,cxzr,ctz
c
c SUBROUTINE EBDEV is the sub-controller for the develop routines.
c idevfl(3)=1 for publication runs, which are more costly.
c
      call ebmsg(18)
      if(idevfl(5).eq.1) go to 2
      xmax = cpwind
      zmax = deltz*float(nprlyr)
c note there is no positional correspondence between the horizontal
c image points and the points on the developing string.
c breakthru estimation section
      jmax = 2
      elmax=elin (2,2)
      nmhp1 = nmhpts +1
      do 10 j=3,nmhp1
      if(elmax.ge.elin(2,j)) go to 10
      jmax = j
      elmax = elin(2,j)
   10 continue
c
c find estimated time for breakthru
      xjmax = deltx*float(jmax-2)
      delttm = 4.
      factor = .6
      tbreak = 1.e18
      kount = 0
      ncnvrg = 0
   11 tbsave = tbreak
      kount = kount + 1
c if kount > 20--stop - unable to converge
      if(kount.gt.20) call devmsg(4)
c
c lower factor if second estimate of tbreak<15 sec
c insurance against false convergence for high doses
      if( (kount.ne.3).or.(tbreak.ge.15.) ) go to 13
      factor = .4
      ncnvrg = 0
```

```
   13 ncnvrg = ncnvrg + 1
      tbreak = 0.
      delttm = factor*delttm
      z = 0.
   12 ctz = cmplx(xjmax,z)
      z = z + delttm*ebrate(ctz)
      tbreak = tbreak + delttm
      if(z.lt.zmax) go to 12
c
c if difference in previous 2 calculated breakthru times (say a and b)
c is > 5% of 'a'--no convergence
      if( abs(tbsave-tbreak).ge.(.05*tbsave) ) ncnvrg = 0
c
c if ncnvrg=4--convergence
      if(ncnvrg.eq.4) go to 14
      if(ncnvrg.eq.1) break = tbreak
      go to 11
c
c set delttm back to time for ncnvrg=1
   14 delttm = delttm/(factor*factor*factor)
      nadvan = int(devsrt/delttm + .5)
      nadchk = 3
      nckout = nadvan/nadchk + 1
      ncksv1 = nckout
      nadsav = nadchk
c
c npts-the number of points on string is initialized elsewhere
c make npts larger if idevfl(3)=1
      if(idevfl(3).eq.1) npts=int(1.4*float(npts))
      npttmp = npts
c
c set maxpts so that deloop is called when npts>maxpts
      maxpts = int( 1.6*sqrt(zmax*zmax+ xmax*xmax)*float(npttmp)/
     *           cpwind )
      if (maxpts .gt. 1000) call ebmsg(19)
c
c set directions into resist for left and right endpoints of string
      cxzl = (0.,1.)
      cxzr = (0.,1.)
c
c set endpoint positions of string
      xz(1) = (0.,0.)
      xz(npts) = cmplx(xmax,0.)
c normalize the starting string endpoint directions.
      cxzl=cxzl/cmplx(cabs(cxzl),0.)
      cxzr=cxzr/cmplx(cabs(cxzr),0.)
c set min and max string segment lengths for x; set max string
c length for z.
      call linear
      sdistx = abs( real(xz(2)) - real(xz(1)) )
      sminx = .70*sdistx
      smaxx = 1.8*sdistx
      smaxz = 1.8*deltz
      if(idevfl(3).ne.1) go to 1
```

```
      smaxx = 1.5*sdistx
      smaxz = 1.5*deltz
    1 ttot=0.
c intialize the pltout routine.
      nflg = 0
      nzflg = 0
      ttotsv = 0.
c
c determine # of contours requested, if too large--stop
    2 nout = int( (devend-devsrt)/devinc ) + 1
      if(nout.gt.mxndev) call devmsg(5)
      if(nout.lt.1) call devmsg(6)
      if(nckout.lt.1) call devmsg(10)
c time between checks = tchk and time between advances = tadv
      tchk = devsrt/float(nckout)
      tadv = tchk/float(nadchk)
      call devmsg(2)
c initialize ploting subroutine
      call pltout(26)
c
c enter loop 5, the main developing loop
      do 5 iout=1,nout
c
c if nzflag=1 (breakthru) or if this is first contour
c do not reset variables for next contour
      if( (nzflg.eq.1).or.(iout.eq.1) ) go to 30
      nadvan = int(devinc/delttm + .5)
      nckout = nadvan/nadchk + 1
      ncksv2 = nckout
      tchk = devinc/float(nckout)
      tadv = tchk/float(nadchk)
c
c loop 3 does one contour at a time
c
   30 do 3 ickout=1,nckout
      call ecycle
      if(iflag.eq.1) call chkr
      if(iflag.eq.1) call bndary
c call deloop when npts gets large and after every output.
c deloop(2) removes all loops, takes more time, and is reserved
c for the outputs.
      if(npts.ge.maxpts) call deloop(1)
    3 continue
c
c reset variables if breakthru or skip if variables have
c allready been reset
      if((nzflg.eq.0).or.(nflg.eq.1))go to 4
      nadchk = 3
      nckout = 8
      if(ncksv1.le.16) nckout = 4
      tchk = devinc/float(nckout)
      tadv = tchk/float(nadchk)
      nflg = 1
      call devmsg(3)
```

```
      maxpts = int(1.8*float(npts))
      if (maxpts .gt. 1000) call ebmsg(19)
    4 call deloop(2)
      iouttp = iout
      if(idevfl(2).eq.1) call prtpts(iouttp)
      if(idevfl(2).eq.1) call plothp(iouttp)
      call pltout(iouttp)
    5 continue
      if(idevfl(2).eq.1)call devmsg(13)
      write(iprint,22222)
22222 format(/1h1)
      return
      end
c
c FUNCTION EBRATE finds development rate in microns/sec
      function ebrate(cz)
      common /dvelp1/ cxzl,cxzr,xz(1000),xmax,zmax,npts,nadchk,nckout
      common /horimg/ deltx,mnhpts,nmhpts,horint(50)
      common /simpar/ nprlyr,nprpts,nendiv,deltm,deltz
      common /ratdat/ r1,cm,d0,alph
      common /line1/ elin (82,1002),lcou,lincou,elnwgt (1999)
      complex cz,xz,cxzl,cxzr
      integer d0
      data unit /1.e-4/
c function ebrate calculates a rate for the x,z position given.
c ebrate lets the string develop outside the boundary at a
c much reduced rate, in order to keep the string length down.
c chkr deletes the points outside of (0.,xmax).
c
c bacrat is the background etch rate
c
      bacrat=r1*(cm**alph)*unit
      truex = real(cz)
      truez = aimag(cz)
c
c 1.5 is the column of elin where xz(1) is located
c
      x = 1.50001 + truex/deltx
    5 z = 1.50001 + truez/deltz
      ix = int(x)
      iz = int(z)
      if(iz.lt.1)go to 40
      if((ix.ge.(nmhpts+2)).or.(ix.le.0))go to 40
      if(truez.gt.zmax) go to 50
c
c set fractions for weighting energy values in elin
c since xz(#) will seldom be exactly on an array point
      fracx = x - float(ix)
      fracz = z - float(iz)
      sfracx = 1. - fracx
      sfracz = 1. - fracz
c
c find interpolated value for amount of energy absorbed in
c resist at point (x,z). Note: subscripts inverted since
```

```
c (x,z) real space=(z,x) array space
      en = elin(iz,ix)*sfracx*sfracz + elin(iz,ix+1)*fracx*sfracz
     *   + elin(iz+1,ix)*sfracx*fracz + elin(iz+1,ix+1)*fracx*fracz
      ebrate = unit*r1*((cm +en/d0)**alph)
      return
c next stmt prevents the str from developing too far outside the bound.
   40 ebrate = bacrat
      return
c use reflective boundary conditions for 3 layers beyond the resist.
   50 if(truez.gt.(zmax+3.*deltz)) go to 40
      truez = zmax - (truez-zmax)
      go to 5

      end
```

```
      subroutine ecycle

      common /dvelp1/ cxzl,cxzr,xz(1000),xmax,zmax,npts,nadchk,nckout
      common /dvelp2/ tadv,tchk,ttot,iflag,smaxx,sminx,smaxz
      common /dvelp3/ nzflg,ttotsv
      common /anrate/ frac
      complex xz,cxzl,cxzr,dl,dr,dt
c SUBROUTINE ECYCLE takes the strings thru the
c no. of advances in between checks.

      ufrac=1.-frac
      iflag=0
      tminx = 1.e38
      tmaxx = 0.
      tmaxz = 0.
      zposmx = 0.
      nsave = 0

      nstop=npts-1
      do 1 n=1,nadchk
c calculate the advance of the left endpoint.
      dl=xz(2)-xz(1)
      xz(1) = xz(1)+cxzl*cmplx(tadv*ebrate(xz(1)),0.)
c t is the length between string points--tmax(x,z) and tmin(x,z) are
c the max and min length of the string segments for x and z.
      t = cabs(dl)
      dl=dl/cmplx(t,0.)
c calculate the advance for the middle string points based on the
c direction of the segments on either side of the points.
      do 2 m=2,nstop
      dr=xz(m+1)-xz(m)
      tminx = amin1( tminx,abs(real(dr)) )
```

```
      tmaxx = amax1( tmaxx,abs(real(dr)) )
      tmaxz = amax1( tmaxz,abs(aimag(dr)) )
      t=cabs(dr)
      dr=dr/cmplx(t,0.)
      dt=dl+dr
      dt = dt/cmplx(cabs(dt),0.)
c dt*sqrt(-1) is the normalized direction for the present point.
      xz(m) = xz(m) + dt*cmplx(0.,frac*tadv*ebrate(xz(m)))+
     *        cmplx(0.,ufrac*tadv*ebrate(xz(m)))
      zposmx = amax1(zposmx,aimag(xz(m)))
      dl=dr
    2 continue
c calculate the advance position for the right endpoint.
      xz(npts)=xz(npts)+cxzr*cmplx(tadv*ebrate(xz(npts)),0.)
      if(zposmx.ge.zmax) nsave=n
    1 continue
c if the segment lengths are too long/short, delete/add points(chkr).
      if( (tmaxx.gt.smaxx).or.(tminx.lt.sminx).or.(tmaxz.gt.smaxz) )
     *    iflag=1
      ttot=ttot+tchk
c if breakthrough has occured, set flag.
      if( (nsave.eq.0).or.(nzflg.eq.1) ) return
      nzflg = 1
      ttotsv = ttot-tchk+float(nsave)*tadv
      return

      end
```

----------MESSAGES AND INFORMATION----------

```
c SUBROUTINE EBMSG(numb) is the message subroutine for e-beam
c
      subroutine ebmsg(numb)
      common /devtim/ mxndev,devsrt,devend,devinc
      common /io1/ itermi,ibulk,iprout,iresv1,iin,iprint,ipunch
      common /spot2/ ishift,wght1,nrhcet,nremat
      common /line1/ elin (82,1002),lcou,lincou,elnwgt (1999)
      common /line2/ ncelin,numcol,maxcol,ncemlt,nclelu,nrelin
      common /cnvlv1/ nclmet,devuni,devtem,dose
      common /cnvlv2/ emat (80,500),numspt,wght0,ncemat
      common /cell/ cellx,cellz
      common /ratdat/ r1,cm,d0,alph
      common /distnc/ shfdis(20),dislin(20),sptwgt(20),wgtlin(20),
     *                stddev(20),itcou
      common /sdist/ shift
      common /anrate/ frac
      common /mcarlo/ thick,eveng
      common /engplt/ idep,iskip,ir1,engmax
      integer d0
      go to (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,
     *       21,22,23,24) numb
1     write(iprint,100) nremat,ncemat
100   format(1x,9hemat has ,i2,10h rows and ,i3,8h columns)
      write(iprint,110) nclmet
110   format(1x,10hetem2 has ,i3,8h columns)
      write(iprint,120) ncemlt
120   format(1x,9hemlt has ,i4,8h columns)
      write(iprint,130) nrelin,ncelin
130   format(1x,9helin has ,i2,10h rows and ,i4,8h columns/)
      return
2     write(iprint,1000)
      write(iprint,200)
200   format(1x,31h          symmetric development/)
      return
3     continue
      return
4     write(iprint,400) cellx,cellz
400   format(1x,12hcell size = ,f6.4,13h microns in x/,1x,
     *          12hcell size = ,f6.4,13h microns in z/)
      write(iprint,420) dose
420   format(1x,7hdose = ,f6.2,9h uc/cm**2/)
      return
5     write(iprint,2000)
      write(iprint,500)
500   format(1x,41hERROR - # of rows in emat greater than 80/)
      stop
6     write(iprint,2000)
      write(iprint,600)
600   format(1x,45hERROR - # of columns in emat greater than 500/)
      stop
```

```
7       write(iprint,700)
700     format(1x,39hERROR - # of spots/line greater than 20/)
        return
8       write(iprint,1000)
        write(iprint,800)
800     format(1x,29hperiodic spot shift specified/)
        return
9       write(iprint,2000)
        write(iprint,900) shift
900     format(1x,32hERROR - requested spot shift of ,f6.4,8h microns,
       *        31h requires extra columns in emlt/)
        stop
10      write(iprint,140) itcou
140     format(1x,i2,11h spots/line/)
        do 99 i=1,itcou
           write(iprint,150) i,stddev(i),sptwgt(i),shfdis(i)
150        format(1x,7hspot # ,i2,13h std. dev. = ,f6.4,8h microns,/
       *   1x,9hweight = ,f6.4,20h distance shifted = ,f6.4,8h microns/)
99      continue
        return
11      write(iprint,111)
111     format(1x,45hERROR - must have at least one line specified/)
        return
12      write(iprint,122)
122     format(1x,36hERROR - more than 20 lines requested/)
        return
13      write(iprint,1000)
        write(iprint,133)
133     format(1x,29hperiodic line shift specified/)
        return
14      write(iprint,144)
144     format(1x,50hWARNING-first line does not start at x=0.0 microns/)
        return
15      write(iprint,155) lincou
155     format(1x,i2,14h lines written/)
        do 199 i=1,lincou
           write(iprint,156) i,dislin(i),wgtlin(i)
156        format(1x,7hline # ,i2,18h shift distance = ,f7.4,8h microns/,
       *          1x,9hweight = ,f6.4/)
199     continue
        return
16      write(iprint,2000)
        write(iprint,166)
166     format(1x,48hERROR - window too large-not enough elin columns/)
        stop
17      do 299 i=1,lcou
           write(iprint,177) i
177        format(1x,16h-------- Line # ,i2,9h --------/)
           strtln=dislin(i) + shift
           write(iprint,178) strtln
178        format(1x,12hline starts ,f7.4,25h microns from window edge)
           do 399 j=1,itcou
              stgaus=strtln+shfdis(j)
              write(iprint,179) j,stgaus
```

```
179          format(1x,7hspot # ,i2,4h is ,f7.4,8h microns,
     *       17h from window edge)
399   continue
      write(iprint,175)
175   format(1x,1h )
299   continue
      write(iprint,174) frac
174   format(1x,18hanrate fraction = ,f7.5/)
      return
18    write (iprint,180) devsrt,devend,devinc
180   format (1x,28hdev time for first output = ,f5.1,8h seconds/
     *     ,1x,28hdev time for final output = ,f5.1,8h seconds/
     *   ,1x,32htime between intermediate dev = ,f5.1,8h seconds/)
      return
19    write(iprint,2000)
      write (iprint,190)
190   format (1x,31hERROR-in ebdev-maxpts too large/)
      stop
20    write(iprint,1000)
      write(iprint,220) thick,eveng
220   format(1x,19hresist thickness = ,f6.4,8h microns/,
     *        1x,14hbeam energy = ,f6.3,4h Kev/)
      return
21    write(iprint,3000)
      return
22    write(iprint,1000)
      write(iprint,230) idep,iskip
230   format(1x,26henergy profile pts printed/,
     *        1x,14hfirst depth = ,i3,14h skip depth = ,i3/)
      return
23    write(iprint,1000)
      return
24    write(iprint,410) r1,cm,d0,alph
410   format(1x,27hrate equation coefficients:/,
     *        1x,5hr1 = ,f6.4,9h    cm = ,f6.4,/
     *        1x,5hd0 = ,i3,11h    alph = ,f6.4/)
      return
1000  format(///,20x,45h---------- system message (e-beam) ----------/)
2000  format(///,20x,41h---------- fatal error (e-beam) ----------/)
3000  format(///,20x,38h---------- warning (e-beam) ----------/)
      end
```

```
c SUBROUTINE PRARRY prints out various arrays if flags are set
c
      subroutine prarry(numb)
      common /cnvlv1/ nclmet,devuni,devtem,dose
      common /cnvlv2/ emat (80,500),numspt,wght0,ncemat
      common /spot1/ etem2 (999),emlt (1499)
      common /spot2/ ishift,wght1,nrhcet,nremat
      common /line1/ elin (82,1002),lcou,lincou,elnwgt (1999)
      common /line2/ ncelin,numcol,maxcol,ncemlt,nclelu,nrelin
      common /io1/ itermi,ibulk,iprout,iresv1,iin,iprint,ipunch
c
      go to (1,2,3,4,5) numb
c
1     write (iprint,10)
10    format (1x,21hmultiple spot array:/)
         write (iprint,21)
21       format (1x,1h )
         write (iprint,22)(emlt(j),j=1,numcol)
22       format (1x,8(e10.4,1x))
      write(iprint,21)
      return
2     write(iprint,30)
30    format (1x,21hmultiple line array:/)
      do 40 i=1,nrelin
          write (iprint,21)
          write (iprint,22)(elin(i,j),j=1,nclelu)
40    continue
      return
c
c this part prints out etem2 and related info
c
c write out headings
c
3     write (iprint,112) numspt
112   format (1x,17hgaussian array # ,i1/)
      write (iprint,212)
212   format (1x,13hlhs gaussian/)
c
c print lhs gaussian
      istrt=nrhcet+1
c
      write (iprint,214)(etem2(j),j=1,nrhcet)
214   format (1x,8(e10.4,1x))
      write (iprint,21)
c
c print rhs gaussian--including center
c
      write (iprint,216)
216   format (1x,13hrhs gaussian/)
      write (iprint,218)(etem2 (j),j=istrt,nclmet)
218   format (1x,8(e10.4,1x))
      write (iprint,21)
      return
c
```

```
c the following prints out adjusted emat
c
c write out emat with heading
c
4       write (iprint,62)
62      format (1x,13hinput matrix:/)
        write (iprint,156) ((emat (i,j), j=1,ncemat), i=1,nremat)
156     format (1x,8e10.4)
        return
c
c print out elnwgt when emlt is printed out
c
5       write(iprint,70)
70      format(1x,7helnwgt:/)
        max=nclelu-2+nclmet
        write(iprint,218)(elnwgt(j),j=1,max)
        return
        end
```