Technical Report
RAL-TR-95-033

Production Techniques of Phase Plates for Laser Focal Spot Smoothing

D A Pepler C N Danson I N Ross and S A Edwards

July 1995

COUNCIL FOR THE CENTRAL LABORATORY OF THE RESEARCH COUNCILS

Enquiries about copyright, reproduction and requests for additional copies of this report should be addressed to:

The Central Laboratory for the Research Councils
Library and Information Services
Rutherford Appleton Laboratory
Chilton
Didcot
Oxfordshire
OX11 0QX
Tel: 01235 445384 Fax: 01235 446403
E-mail library@rl.ac.uk

ISSN 1358-6254

Production Techniques of Phase Plates for Laser Focal Spot Smoothing

DA Pepler, CN Danson, IN Ross and SA Edwards

Central Laser Facility, Rutherford Appleton Laboratory,
Chilton, Didcot, Oxon, UK.

Contents **Page**

1 Introduction 2

2 Theory 3
2.1 Random phase plates 3
2.2 Phase zone plates 4

3 Manufacture of phase plates 7
3.1 Mask generation 8
3.1.i Acetate production 8
3.1.ii Mask designs 8
3.1.iii Production of optical mask using chrome 11
3.1.iv Production of mask using optical resist 11
3.2 Manufacture of phase plate 13
3.2.i UV photo-resist 13
3.2.ii P10 photo-resist development 17
3.2.iii PM15 photo-resist development 17
3.3 Overcoat spinning 17

4 Phase plate examination 17

5 Conclusion 19

6 Acknowledgements 20

7 References 20

Appendix I Basic programming techniques using PostScript 21
Appendix II Random Number Generation 34
Appendix III Interference microscope measurements 36
Appendix IV Current suppliers 38

1 Introduction

In high power lasers such as the VULCAN Nd:glass laser system at the Rutherford Appleton Laboratory (RAL) the quality of the focal spot is a critical issue for laser-plasma interaction studies. Typically in these systems the beam passes through many optical components (such as Nd:glass amplifiers and focusing lenses) which can create undesirable phase and amplitude modulations. These can severely effect the focusability of the laser and beam smoothing techniques have therefore had to be developed to overcome these problems.

Kato et al[1] developed the random phase plate (RPP) in order to provide a means to scramble the phase of a laser beam and thereby cause a smoothing effect. This method was then employed in order to increase the quality of data gained from laser-plasma interaction experiments. An RPP consists of a random array of transmitting areas which induce a phase shift of either 0 or π radians into the beam in which it is placed.

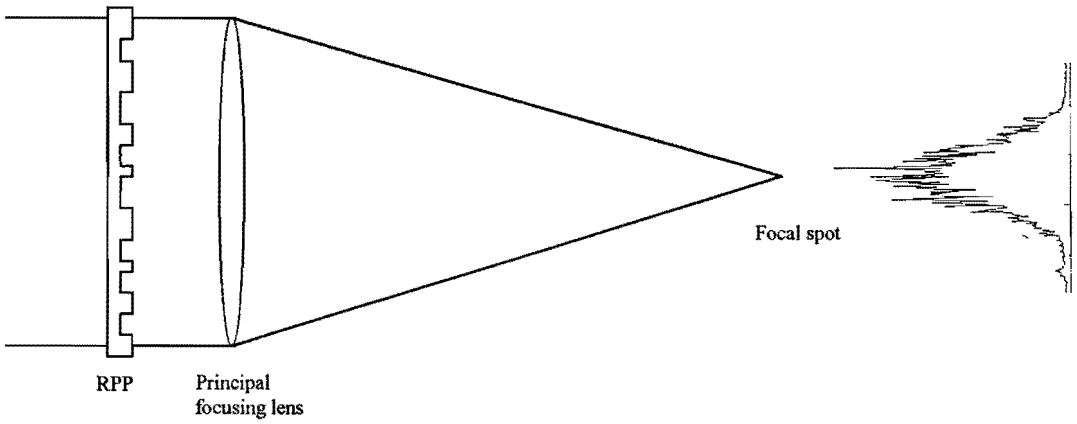


Figure 1. Diagram showing the implementation of an RPP.

When used in conjunction with a principal focusing lens, as shown in figure 1, the normal focal spot size is enlarged due to the diffractive nature of the RPP with a sinc^2 intensity profile superimposed on which is a speckle pattern containing many high spatial frequencies. These RPP's have been used successfully for many years but fail to produce the desired top hat intensity profiles. Deng et al[2] developed an alternative technique using an array of refracting lenses to split the beam into many separate beamlets. This array is focused onto a principal plane where all the beamlets overlap to produce a relatively large focal spot but with a top hat profile.

A combination of these two techniques, the phase zone plate array (PZP), was developed at RAL in collaboration with researchers from AWE plc[3]. This report describes the techniques developed at RAL to manufacture both the RPP and PZP plates.

2 Theory

2.1 Random phase plates

A random phase plate is a binary diffractive optic which has a glass or quartz substrate with a surface relief either etched directly into it's surface or into a thin film covering it's surface. The surface is tessellated in some basic shape, for example a square, with the etching depth made to be $\lambda/2(n-1)$; where n is the refractive index of the etched material, and λ is the operating wavelength. Each square acts as a diffracting aperture and the resulting focal profile is the superposition of diffracted light from all the apertures. The focal spot consists of a high spatial frequency speckle pattern with a scale length of the order of w'. The focal spot size and speckle size are given by

$$w \approx \frac{f\lambda}{d} \qquad \text{and} \qquad w' \approx \frac{f\lambda}{D} \tag{1}$$

where f is the focal length of the lens, d the element size and D is the beam size. The pattern is randomly formed with half of the squares producing a zero phase shift and the other half, a π radian phase shift. The equal areas of 0 and π destructively interfere exactly on axis, eliminating the central spike, the characteristic focal spot without the RPP.

Table 1. Random phase plate focal spot sizes (FWHM)

ELEMENT SIZE (mm)	**λ = 1.053 μm fl = 0.25 m**	**λ = 1.053 μm fl = 1 m**	**λ = 0.527 μm fl = 0.25 m**	**λ = 0.527 μm fl = 1 m**
10	25 μm	100 μm	25 μm	50 μm
5	50 μm	200 μm	25 μm	100 μm
2	125 μm	500 μm	67 μm	250 μm
1	250 μm	1 mm	125 μm	500 μm
0.5	500 μm	2 mm	250 μm	1 mm
0.25	1 mm	4 mm	500 μm	2 mm
0.1	2.5 mm	10 mm	1.25 mm	5 mm
0.05	5 mm	20 mm	2.5 mm	10 mm

It is therefore possible to control the size of the focal spot by using different RPP plates. Table 1 shows the plates required for a number of focal spot sizes for the two principle wavelengths on Vulcan and the two main focal length lenses.

The small scale speckle contains mainly high spatial frequencies which in many laser-plasma interaction experiments will be smoothed by energy transportation in the plasma.

To obtain satisfactory focal spot smoothing the thickness of the resist is critical to about 5%. A discrepancy in the thickness produces a coherent spike in the centre of the profile. With the current manufacturing technique this tolerance is unrealistic. It is possible however, by operating slightly out of the focal plane to reduce the thickness tolerance. An optimum working distance has been calculated to be

$$\Delta \approx \frac{f^2 \lambda}{2dD} \tag{2}$$

beyond the principal focal plane.

At this plane, no significant increase in the overall size of the focal spot occurs, but the coherent central spike will be defocused to approximately half the size of the smoothed spot. This degree of defocusing is possible due to the large depth of focus of the beam from the elements of the RPP compared to the depth of focus of the coherent spike.

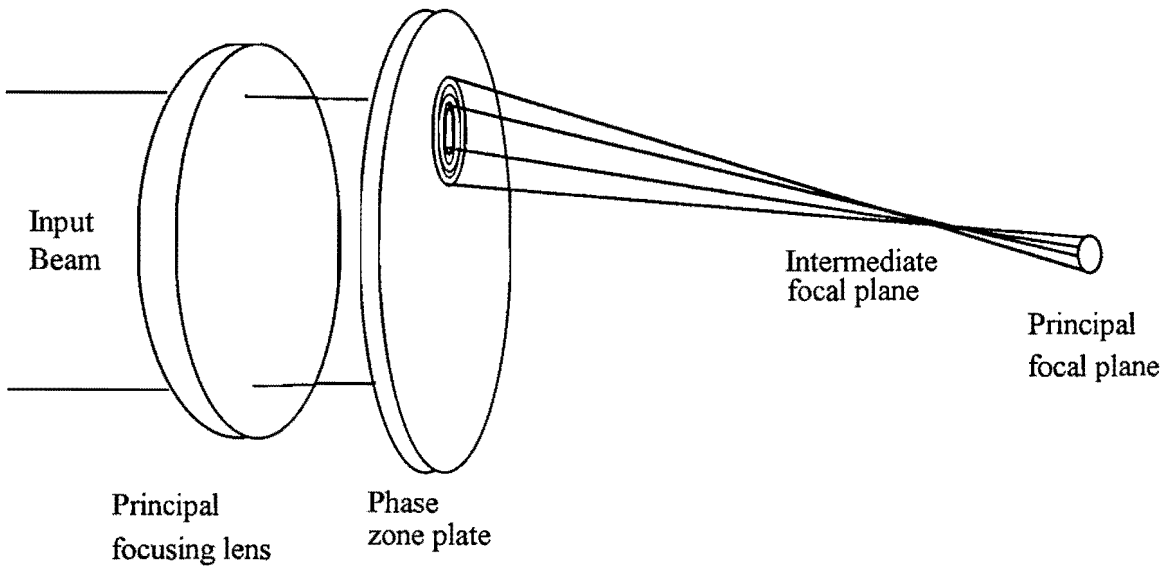


Figure 2. Schematic of the PZP focusing scheme

2.2 Phase zone plates

The PZP consists of an array of binary phase Fresnel zone plates where each zone plate behaves like a lens. A schematic of the PZP scheme is shown in Figure 2.

The PZP is positioned near to a principal focusing lens of focal length F_p and this will cause each beamlet to be focused at a distance corresponding to the combination of F_z and F_p, where F_z is the focal length of the PZP in the first order. The first order focal length can be calculated from the equation[4],

$$F_Z = \pm(R_m^2/m\lambda - m\lambda/4) \tag{3}$$

where F_z is the focal length, m is the order of the ring, R_m is the radius of that ring and λ is the wavelength of light.

The generated spot size in the principal focal plane can be easily calculated for the two situations when the PZP is situated either before or after the lens. When the PZP is placed at any distance before the lens the geometric spot size is given by,

$$W_S = W_Z \frac{F_P}{F_Z} \tag{4}$$

where W_S is the spot size and W_Z is the size of each zone.

The second case is when the PZP is placed a distance δ after the focusing lens, and the equation becomes,

$$W_S = W_Z \left(\frac{F_P - \delta}{F_Z} \right) \tag{5}$$

The focal length of a zone plate for each diffraction order is inversely proportional to the order number thus resulting in similar combination surfaces corresponding to both the positive and negative orders. This is shown schematically in Figure 3 with only the +/- 1st and +/- 3rd orders shown. The positive and negative orders respectively focus before and after the principal focal plane. In the principal plane, all the diffracted beamlets are concentric with the optic axis, and the beamlet overlaps are the same size for each pair of orders, but scale with the order number. The superposition of the beamlets is the basis for the intensity averaging to

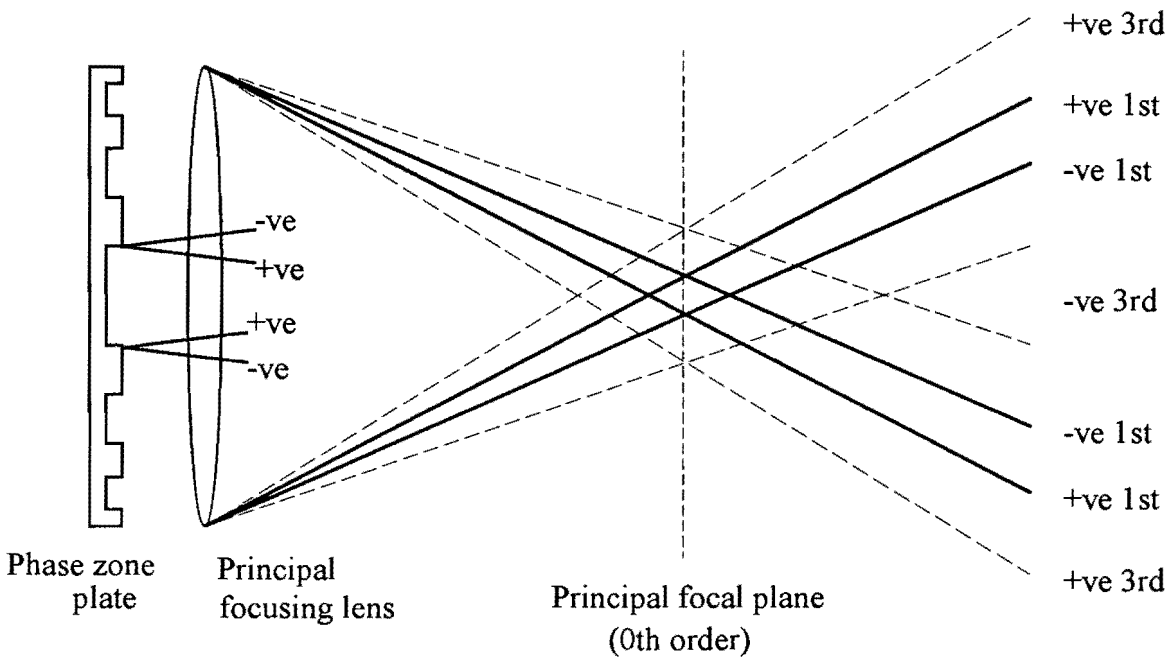


Figure 3. Diagram showing positive and negative orders of a PZP.

produce the final distribution. The final profile is independent of the input beam intensity distribution provided a sufficiently large number of beamlets are produced.

The range of focal spots that can be generated is limited by: the focal length of the principal focusing element; the size of the smallest elements of the zone which can be manufactured using the photo-lithographic process; and that the zone plate must contain a minimum number of elements. The graph shown in Figure 4 shows the focal spot sizes available for a focal length lens of 1 m at a wavelength of 1053 nm. The manufacturing boundary conditions of the PZP masks are given by the heavy lines. The solid line is where each Fresnel cell has only four zones, and the dotted line where the minimum structure size is 40 μm. The thin lines on the graph are lines of constant focal spot size, calculated from the Fresnel cell diameters and the focal length of the zone plate.

The graph can also be used with focal lengths other than 1 m, and with different wavelengths. From equation (4) the focal spot size is directly proportional to the principal lens focal length ie using a 0.25 m focal length lens would result in the spot sizes being reduced by a factor of 4. Similarly, from equations (3) and (4) the focal spot size is also directly proportional to the wavelength and so using the graph for the second harmonic (526 nm) results in the focal spots sizes reducing by a factor of 2.

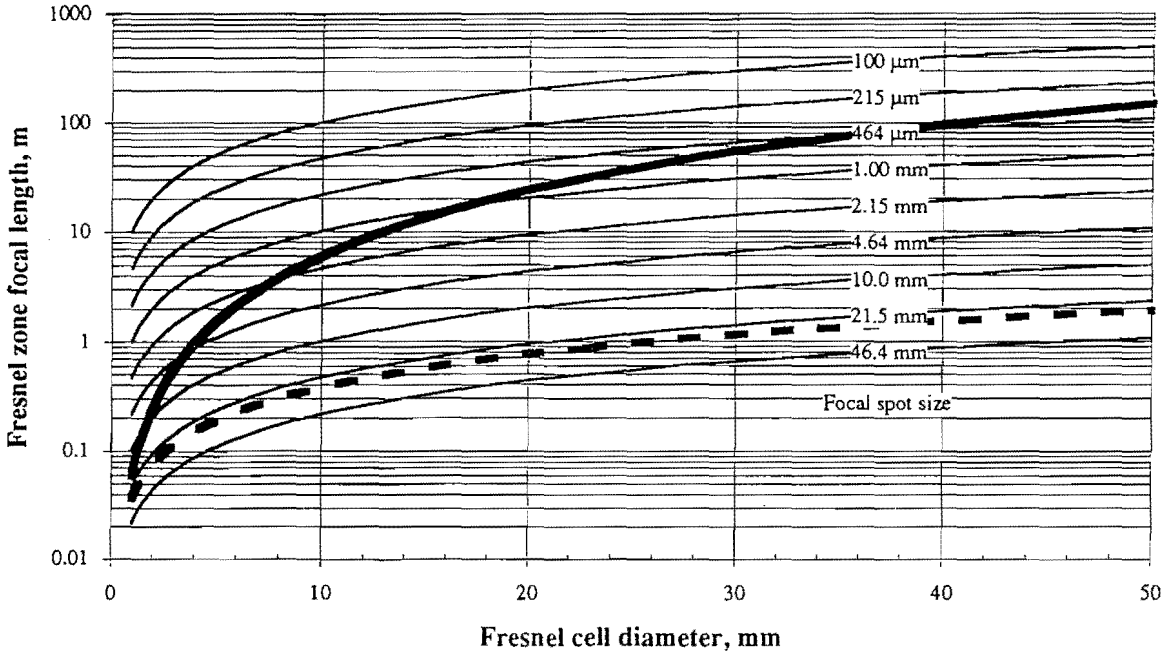


Figure 4. A graph showing the range of focal spot sizes that can be generated using PZPs

3 Manufacture of phase plates

The phase plate manufacturing process uses standard photo-lithographic techniques and requires only three main elements:

1 A transparency with the mask design. A computer generated mask pattern is printed on a Linotronic laser printer to produce an acetate film, with the zero and π radian shift areas respectively transparent and opaque;

2 The production of a UV transparent mask which is produced from either a chrome plated quartz slide overcoated with a visible light photoresist or a blank quartz plate. Both of these products are commercially available from Hoya;

3 A glass substrate coated with a UV photo-resist to the required thickness. The photo-resist is spin coated onto the substrate and the thickness controlled by varying either the spin speed or the concentration of the resist. It is then fixed by baking in an oven at 160°C for 20 minutes.

3.1 Mask generation

There are several steps in the production process of the RPP mask. The design is computer designed and the image down loaded onto a high resolution laser printer producing an acetate mask. The mask pattern is then transferred onto a chrome coated quartz substrate, suitable for exposing the e-beam resist.

3.1.i Acetate production

The acetate masks for the phase plates are constructed using a high resolution (currently 10 μm) Linotronic laser printer. To communicate with the Linotronic printer a program is written in the PostScipt graphics language. The program can be written on any word processor that has the ability to save the written text in a 'text only' format and tested on any laser printer that has a PostScipt interpreter. The preferred equipment for this process is a Macintosh with the 'Word' word processor to generate the text of the program and the 'SendPS 2.0' utility to download the program into the laser printers PostScript interpreter. Details of how the code is written are given in Appendix I and the Random Number Generator described in appendix II.

3.1.ii Mask designs

Since phase plates were developed the number and variety of the mask designs has been rapidly increasing giving scope to a far greater range of focal spot sizes and far field images. Figure 5 shows 3 different mask designs for the manufacture of RPPs. Figure 5(a) shows a square structure [1]. In this case the individual elements are 3 mm square and designed to operate with a 1 m focal length lens at 527 nm producing a focal spot size of 167 μm. Figure 5(b) shows a 2 mm hexagonal structure [5], when used with 0.25 m focusing optics and 1053 nm radiation gives a focal spot size of 125 μm . Figure 5(c) shows a mask with 4 mm circular structure [6]. The RPP produced from this mask will produce a focal spot with a more circularly symmetric profile and also generates less intensity in the region outside the central spot. The focal spot size scales in the same manner as the square and the hexagonal structures.

Mask designs used to generate PZPs are shown in Figure 6. These masks are to scale. Figure 6(a) shows a mask for a PZP designed to be used in front of a principal lens of focal length 564 mm and gives a focal spot size of 0.4 mm. Figure 6(b) is of a PZP to be used with a principal lens of 4 m focal length and gives a spot size of 2.5 mm.

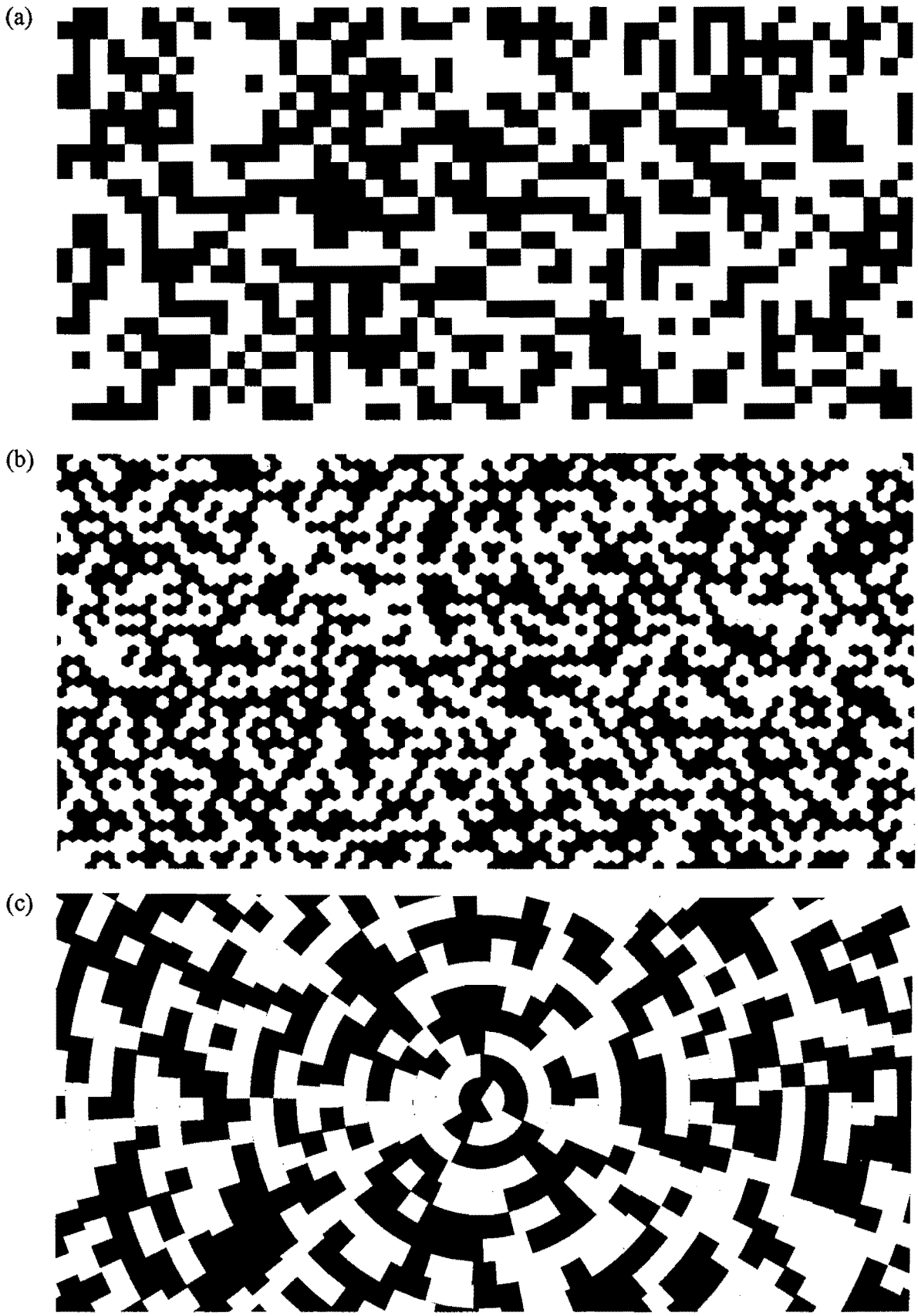
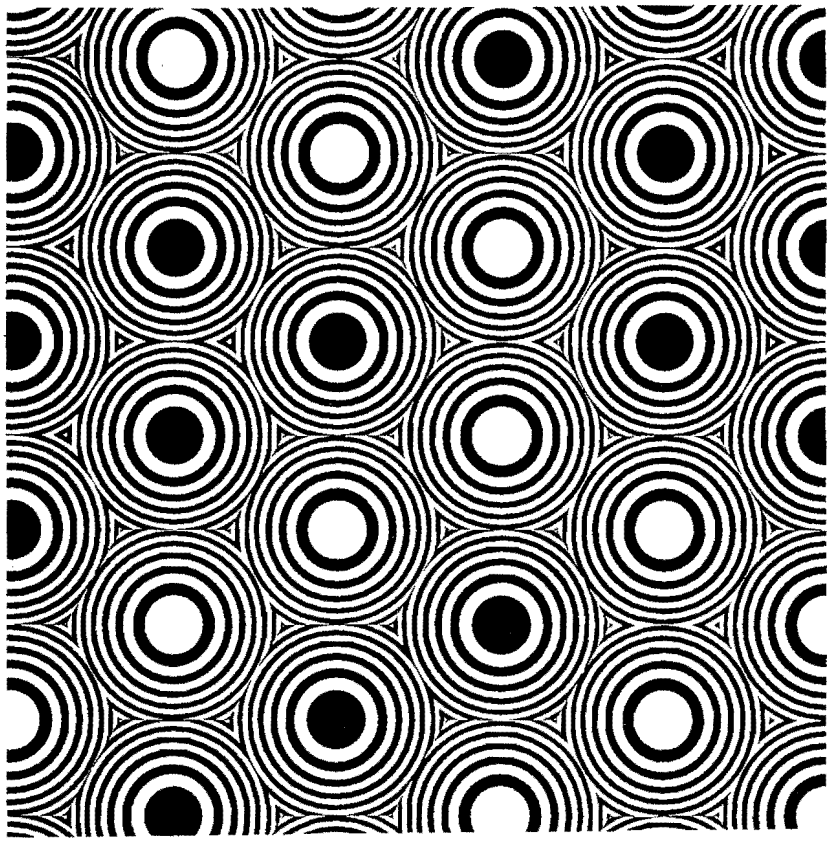


Figure 5. Mask designs used in the production of RPP's

(a)



(b)

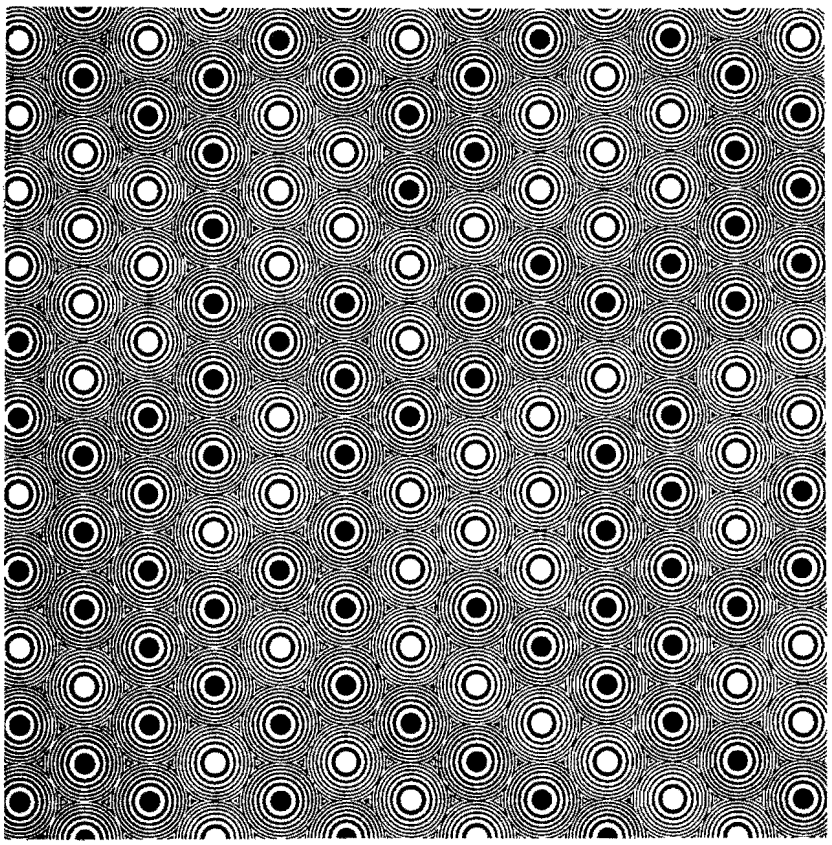


Figure 6. Mask designs used in the production of PZPs

3.1.iii Production of optical mask using chrome

The mask is manufactured using a quartz plate which has been pre-coated with a thin layer of chrome and a visible light photo-resist. Without being exposed to light the plate should be placed chrome side down onto the acetate mask within a light box and exposed for 10 seconds. For the best results the acetate mask should be used with its printed side facing upwards (this can be checked by gently scratching a small line in one of the corners). Develop for approximately 45 seconds and wash gently with water for another 45 seconds. While wearing rubber gloves place the plate in a bath of chrome etch (keeping the chrome side facing upwards) for 60 seconds to remove the chrome from the plate which has been exposed after the exposure and development of the resist. To complete the plate remove the resist which remains on top of the quartz by re-exposing and developing the whole plate.

3.1.iv Production of mask using optical resist

A mask is produced using a blank quartz plate (Hoya) and a visible photoresist (Shipley S1813 SP-15). Before the deposition of the photoresist ensure that the plate has been cleaned and drag wiped with acetone. A lab coat, protective gloves and goggles must be worn before any chemicals are handled. Due to the resist being reactive to visible light it is advisable to reduce the lighting in the room as much as is safely possible, although if the whole process (including exposure and development) is carried out within twenty minutes the resist will not be significantly affected.

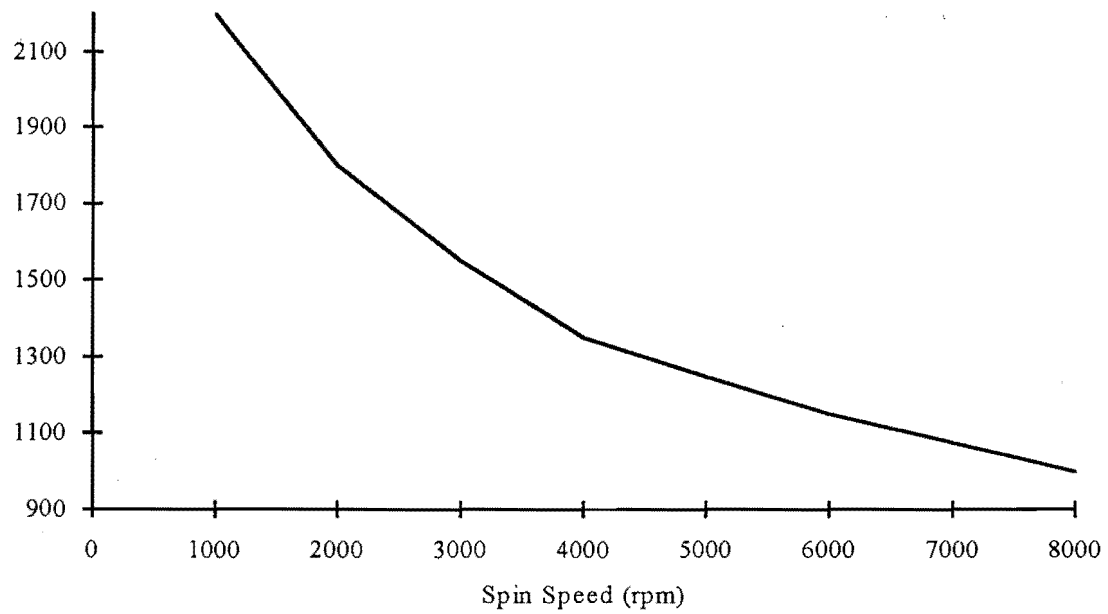


Figure 7. Graph showing spin speed against visible resist depth

The plate can now be set up on a spinner and centralised. Initially spin at 500 rpm, (with an acceleration of 2 for 5 seconds) check that the plate is central and if not switch off and adjust. The graph in Figure 7 shows the thickness of resist deposited for different spin speeds.

The visible resist can be removed from the chemical cabinet and a small amount poured into a beaker (within a fume cupboard). With the cover down on the front of the fume cupboard pour the resist onto the centre of the plate (cover approximately 1/3) and spin the plate. Too little resist will mean the plate won't be fully coated and too much can cause splashing of surplus resist back onto the plate. Inspect the plate and if it is not evenly coated reclean and start again. The standard coating is with the spinner set to 2000 rpm and for 60 seconds.

Once the quartz plate is coated carefully remove and place within a 115°C preheated oven for 4 minutes (being careful to only touch the edges). After this time remove using insulated oven gloves and leave on a bench to cool, this will only take a minute.

To expose the plate place the acetate mask in a light box with the coating side facing upwards as previously mentioned (trying to avoid touching or bending the part that will be used) and then gently rest the quartz plate on top with the resist side facing downwards (see Figure 8).

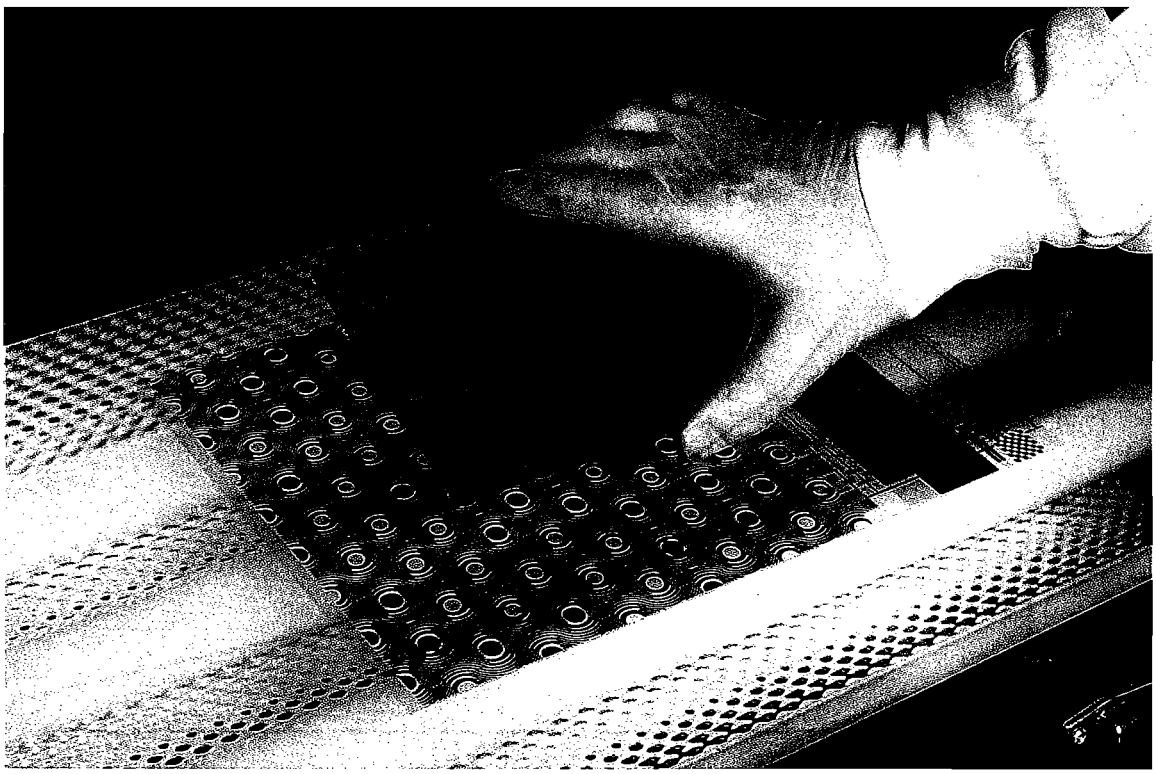


Figure 8. Photograph showing the exposure procedure of the quartz mask

When the lid is closed the plate and mask will be forced together to provide a tight seal, this will ensure a greater sharpness of detail on the quartz mask. Expose the plate for 10 seconds .

With minimal lighting in the room develop the visible resist in Universal Developer for 45 seconds. While agitating observe the effect of the developer on the resist, the pattern should appear very quickly and then gradually be seen to wash away. This gives an important indication of the rate of the developing process. When developed the plate can be gently washed in water for 45 seconds. Leave the plate to dry on a rack in a fume cupboard or respin at approximately 1000 rpm for 30 seconds to accelerate the drying process.

3.2 Manufacture of phase plate

3.2.i UV Photo-resist

The phase plate is spun in the same way as the quartz mask but uses a UV photo-resist (P10 or PM15) which is not sensitive to visible light and therefore does not need to be processed in the dark. The depth of the deposited resist and thus the wavelength of the beam for which the plate can be used can be varied by altering the spin speeds and dilution factors. The following three graphs in Figures 9, 10 and 11 show the results obtained for spinning plates with 50 / 50, 60 / 40 and 70 / 30 ratios of P10 to thinner respectively. The graph shown in Figure 12 represents 100% PM15 resist concentration.

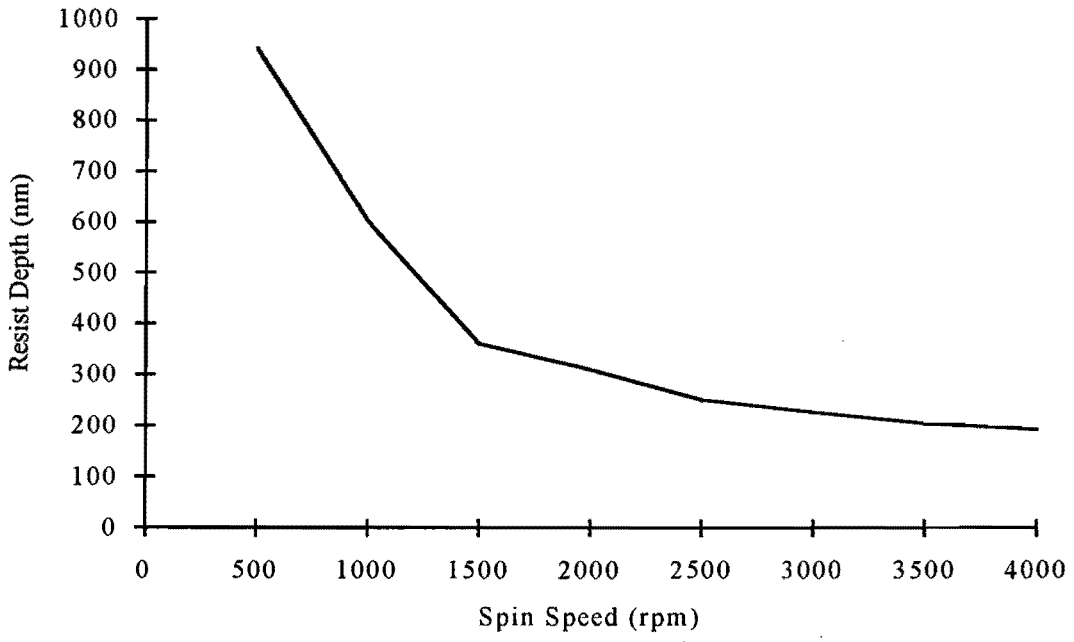


Figure 9. Graph showing spin speed against resist depth for 50% P10 / 50% thinner

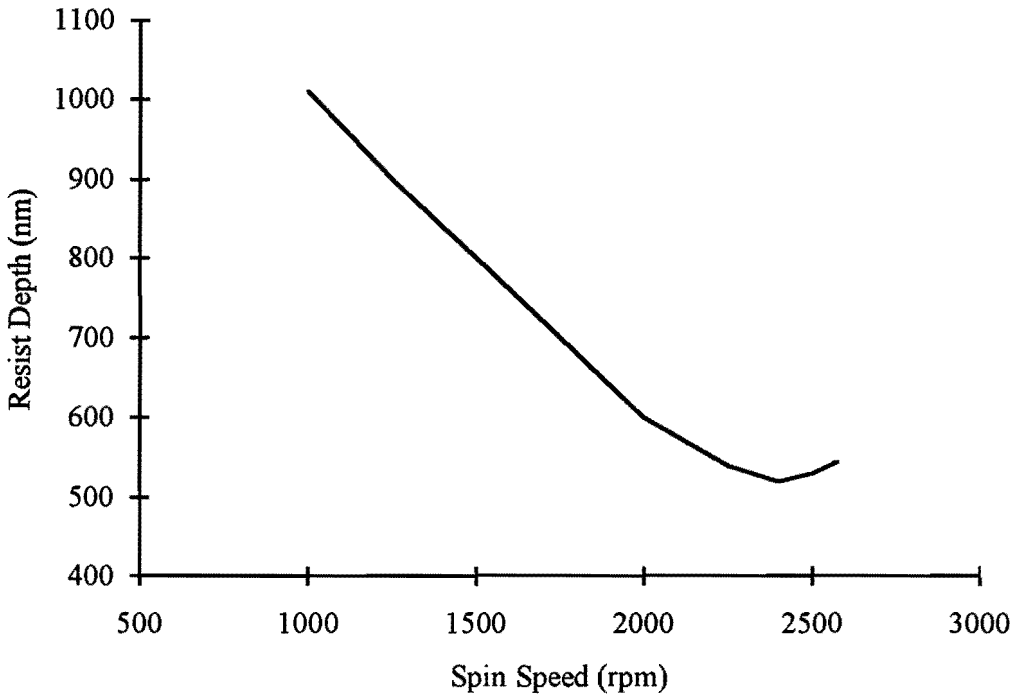


Figure 10. Graph showing spin speed against resist depth for 60% P10 / 40% thinner

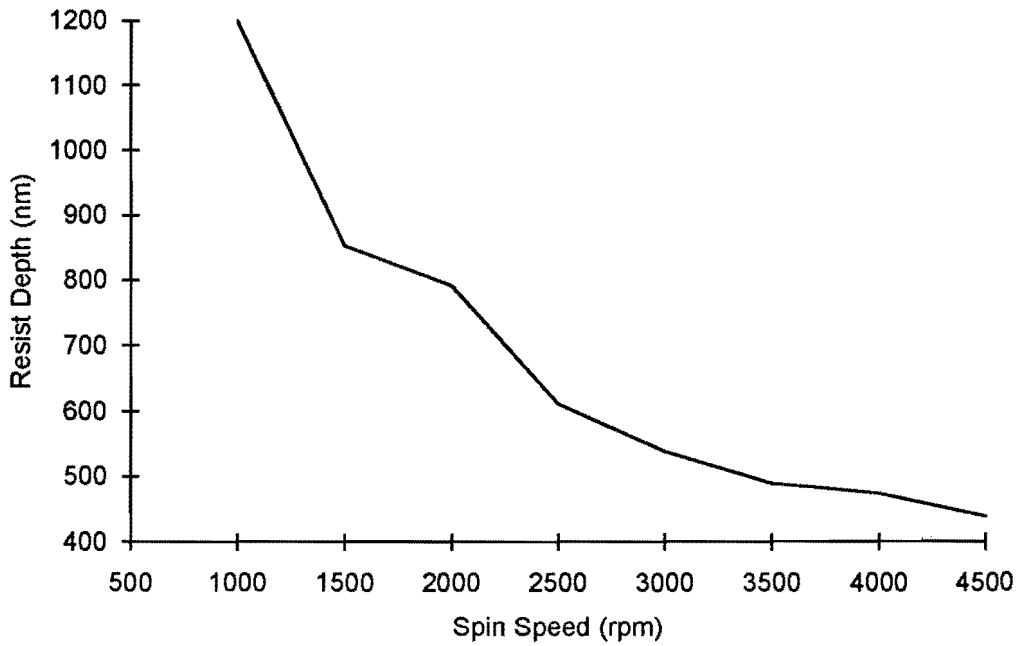


Figure 11. Graph showing spin speed against resist depth for 70% P10 / 30% thinner

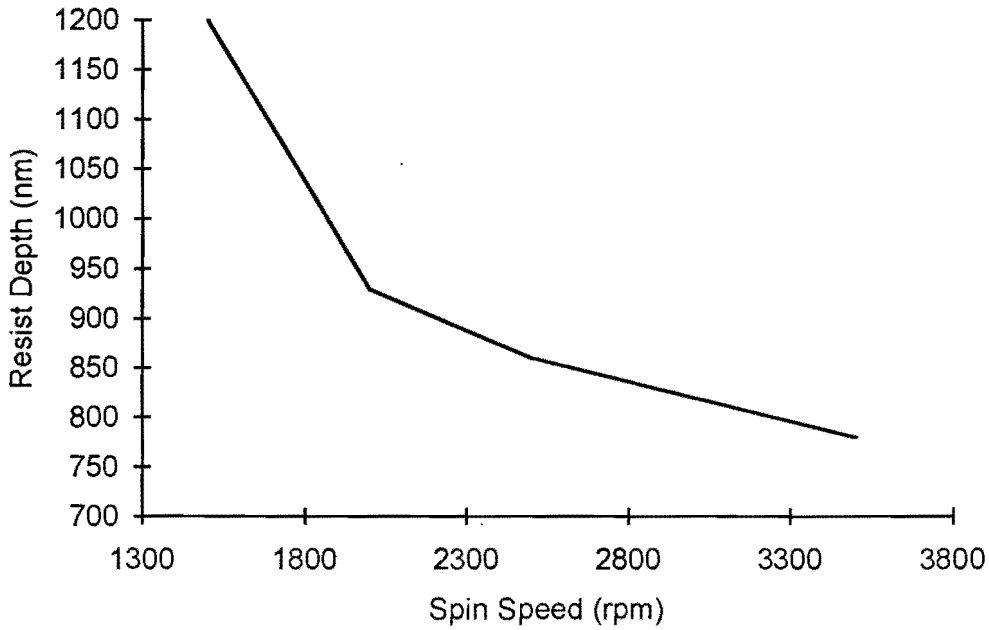


Figure 12. Graph showing spin speed against resist depth for 100% PM15

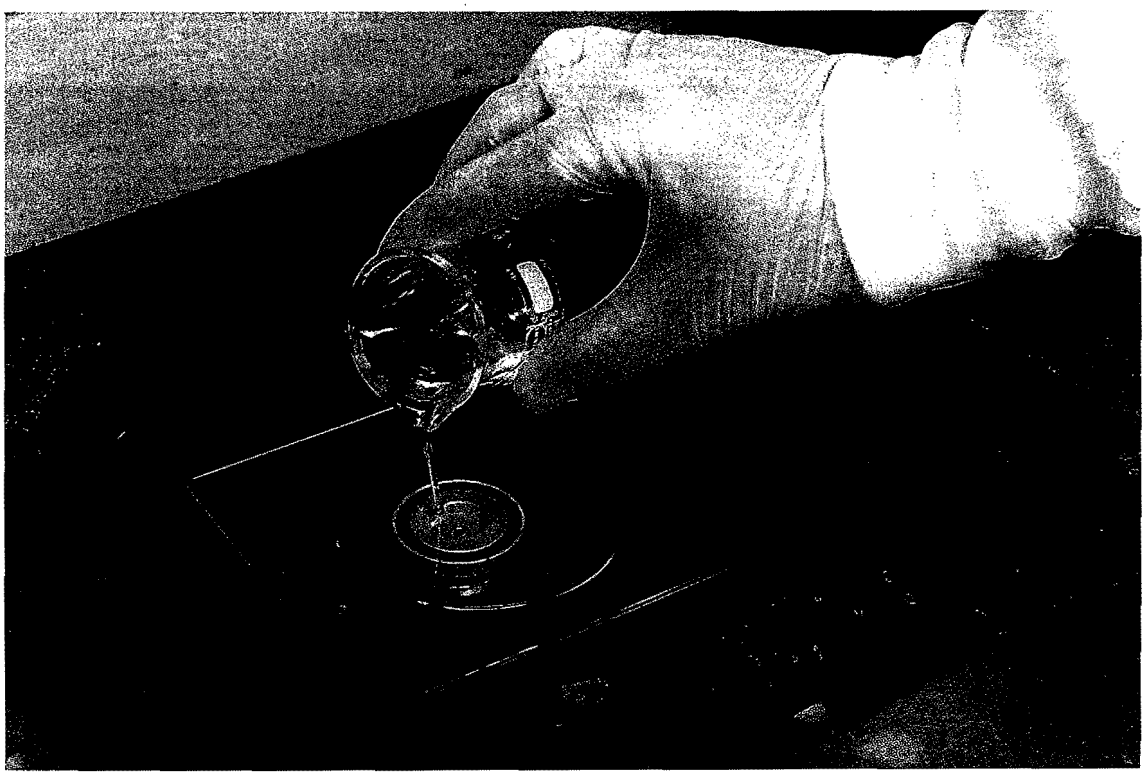


Figure 13. Photograph showing the spinning of the phase zone plate

After spinning the UV photo-resists are baked at a temperature of 160°C for 20 minutes. The plate must be left to cool on an oven rack to prevent rapid cooling, causing the resist to crack.

When cool expose the plate through the quartz mask for a period of 18 hours with a mercury lamp positioned 10 cm above as shown in Figures 14 and 15. It is important that the two layers of resist are in contact with each other as any gap between the two can result in a loss of edge definition.

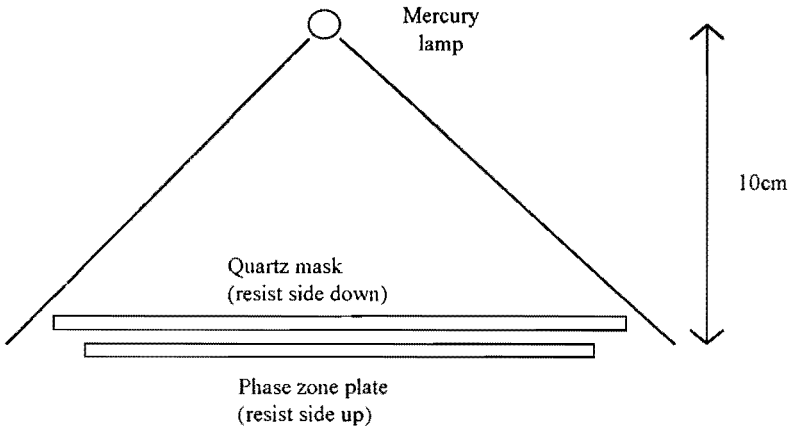


Figure 14. Diagram showing the exposure layout for the phase plate

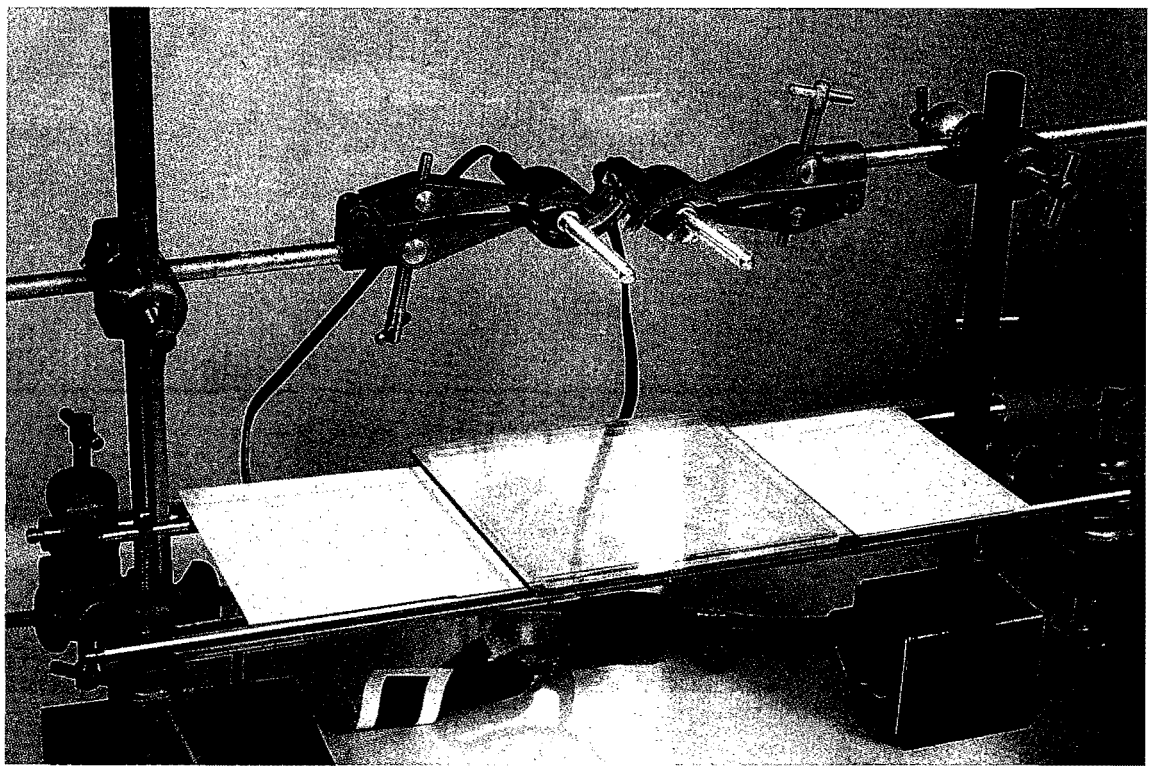


Figure 15. Photograph showing the phase plate exposure procedure

3.2.ii P10 Photo-resist development

The phase plate is developed in 50% Hexone (MIBK) and 50% Isopropyl Alcohol (IPA). While wearing gloves agitate the plate gently for 30 seconds and then wash in 100% IPA for 30 seconds to stop the developing process. The phase zone plate is completed and left to dry in a fume cupboard, the IPA remaining on the plate should evaporate rapidly. If the back of the plate seems smeared or marked carefully clean with lens tissue and acetone making sure not to touch the front surface.

3.2.iii PM15 Photo-resist development

Develop the PM15 photo-resist face down in a bath of acetone for 30 seconds and spin dry at 1000 rpm for 30 seconds. Wash the plate with acetone and drag wipe on the reverse side to leave a cleaner finish.

3.3 Overcoat spinning

A process developed for the manufacture of large aperture phase plates uses two resists spun on the same plate. This negates the requirement of using a large aperture quartz plate where the cost of purchasing a quartz plate would be prohibitive. This requires a PM UV photo-resist to be spun onto the plate and baked and then in darkness a visible resist spun directly onto the surface of the PM resist. The plate now has to be exposed using the same acetate mask and developed as before. This procedure will have resulted in the glass plate being coated in the E-beam resist and then partially coated in the visible resist according to the required mask structure. The plate is now ready to be re-exposed using a mercury lamp for 18 hours and again developed using the 50 / 50 mixture of MIBK/IPA. Remove the remaining visible resist by exposing in the light box and developing again. The plate is now left to dry.

4 Phase plate examination

The phase plate now has to be analysed. The three stages which can be included in this procedure are visual inspection, coating verification and performance:

1) Visual inspection:- hold the plate up to the light and check the quality and uniformity of the resist and of the pattern (ie look for flaking on the surface);

2) Coating verification:- measure the thickness of the resist using an interference microscope (see appendix III). The photographs in Figures 16 and 17 show the

interference fringes obtained using the microscope and a plate coated with UV photo-resist for a wavelength 532 nm;

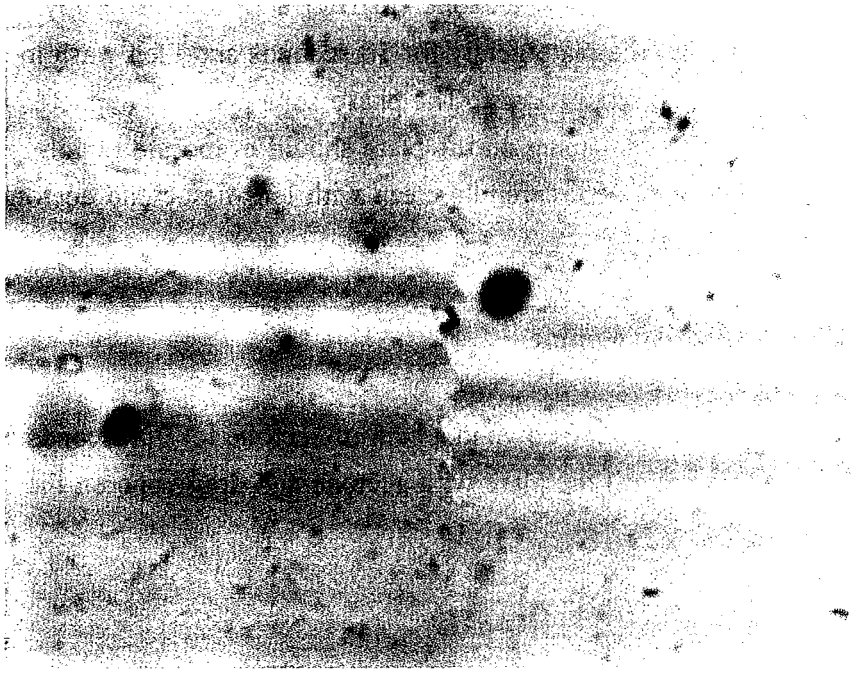
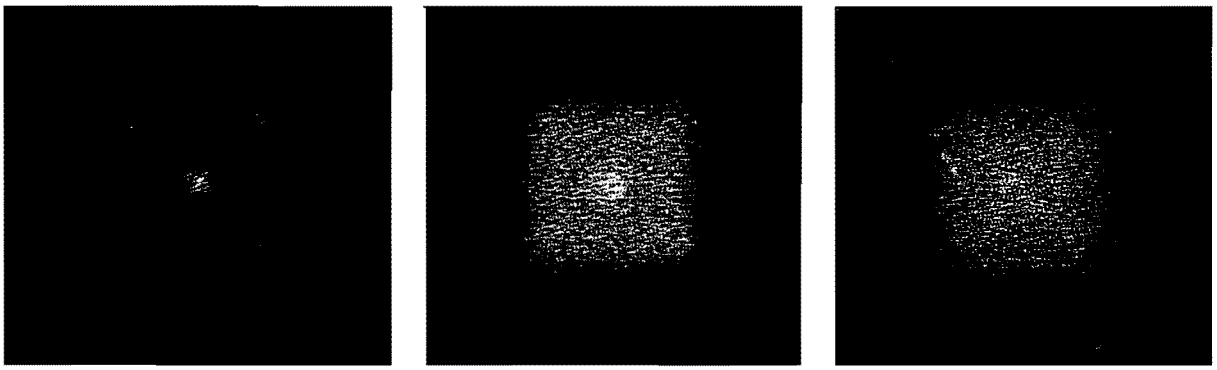


Figure 16. Photograph showing fringe shift from an interference microscope

Figure 17. Photograph showing fringes using a green interference filter

3) Performance:- place the plate in an equivalent plane monitor (EPM) and use a laser of a similar wavelength to that for which the plate is designed to operate and examine the generated focal profile. Figure 18 shows photographs from the EPM in several different planes.



(a) (b) (c)

Figure 18. EPM outputs of a PZP
(a) in the principal focal plane (b) 6 mm beyond the principal focal plane
(c) 12 mm beyond the principal focal plane

5 Conclusions

Focal spot smoothing techniques have been used in a range of scheduled laser-plasma interaction experiments on the VULCAN Nd:glass laser facility.

One of the major advantages of RPP's and PZP's is the ease of use and speed of manufacture. With current mask designs it is possible to produce a phase plate in only one day. Further there are a very wide range of wavelengths that the phase plate can be manufactured for (1.053 μm and 0.531 μm are the most common), and focal spot sizes can be easily manipulated by choice of the element size and focusing optic.

Provided in this report is the necessary information to enable the reader to gain a knowledge of the background and theory involved in the development of the phase plate method of laser beam smoothing. We have outlined the complete procedure needed to manufacture either a RPP or PZP from first principles to completion.

6 Acknowledgements

The authors wish to acknowledge the development work of Bob Bann and the efforts of a long succession of undergraduate students working on this programme: Michelle Barraclough, Jim Exley, Scott Rivers, Michael Dooley and Stephanie Sails.

7 References

[1] Y. Kato, K. Mima, N. Miyanaga, S. Arinaga, Y. Kitagawa, M. Nakatsuka and C Yamanaka, "Random phasing of high power lasers for uniform target acceleration and plasma-instability suppression.", Phys Rev Lett, Vol 53, No 11, p1057 (1984)

[2] X. Deng, X. Liang, Z. Chen, W. Yu and R. Ma, "Uniform illumination of large targets using a lens array", Appl. Optics, Vol.25, No.3, p377 (1986)

[3] R.M. Stevenson, M.J. Norman, T.H. Bett, D.A. Pepler, C.N. Danson and I.N. Ross, "Binary phase zone plate arrays for the generation of uniform focal profiles", Opts Lett, Vol 19, No 6, 363-365, March 15 1994.

[4] J.R. Meyer-Arendt, *Introduction to Classical and Modern Optics* (Prentice-Hall, Englewood Cliffs, N.J., 1984)

[5] J. Kelly (Editor), "Omega phase conversion with distributed phase plates", LLE Review Quarterly Report, DOE/DP40200-65, Vol 33, Oct-Dec 1987.

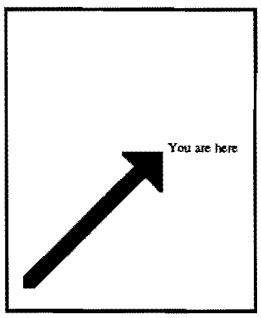
[6] D.A. Pepler, C.N. Danson, R. Bann, I.N. Ross, R.M. Stevenson, M.J. Norman, M. Desselberger and O. Willi, "Focal spot smoothing and tailoring for high-power laser applications", SPIE Conference Proceedings Vol 1870, p 76-87, 1993.

Appendix I

Basic Programming Technique Using PostScript

The PostScript language is a very powerful means for generating complex graphics. The basic concept behind the language is to produce marks on a page either in lines, circles, other more complex blocks or in text forms in various shades of grey and then to print them. However, the simplest of printed forms can seem cumbersome to generate even though the language has great flexibility.

The default co-ordinate system defines the origin to be the bottom left of the page and scales drawings to 72 units per inch (approximately the same size as the printing industry 'point'). So, for example, to generate an arrowed line from the co-ordinate (1 inch, 1 inch) to say (7 inch, 7 inch) and place the text 'You are here' at the end of the arrow as shown at right requires the setup of a number of items, as follows (all text after the '%%' character string is treated as a comment and is not executed)




```
newpath                 %% declares a new drawing segment
        72 72 moveto    %% moves the cursor to (72,72)
        504 504 lineto  %% marks the line (72,72) to (504,504) on the page
closepath               %% terminates the drawing segment
0 setgray               %% sets the current drawing colour to black
10 setlinewidth         %% defines the line width
stroke                  %% colours the marked path with the current colour

newpath                 %% declares a new drawing segment
        504 504 moveto  %% moves the cursor without marking the page
        0 -72 rlineto   %% marks the line from (504,504) to (504,432)
        -72 72 rlineto  %% marks the line from (504,432) to (432,504)
        72 0 rlineto    %% marks the line from (432,504) to (504,504)
closepath               %% terminates the drawing segment
0 setgray               %% sets the current drawing colour to black
fill                    %% fills in the outline of the triangle

/Helvetica findfont 10 scalefont setfont    %% select a font and scale for printing
504 504 moveto (You are here) show          %% position the cursor and prints text

showpage                %% forces the printer to print the page
```

This short program breaks down into 4 sections. Drawing the line; drawing the arrowhead; placing the text; and printing the page:-

1. Drawing the line :- drawing of any shape generates a path and should always start with the command or operator *newpath.* This simply instructs the interpreter to initialise the current path to be empty which causes the current point to be undefined. Thus the next instructions

should redefine the current point, describe various movements and be terminated with the operator *closepath*. The simple movements in this example were '*72 72 moveto*' and '*504 504 lineto*'. The PostScript language works in a 'reverse-Polish' form where numbers or objects are placed on a stack and pulled off as operators require. The program above placed the numbers *72* (x) and *72* (y) onto the stack and the *moveto* operator pulled these back off, interpreting them as the y and x co-ordinates. The *x y moveto* directly defines the current point as being 1 inch (72 points) horizontally and vertically from the origin. Likewise the *x y lineto* operator pulled the co-ordinates off the stack and *marked* a line from the current point to another point 7 inches horizontally and vertically from the origin. Marking a line is not like using a pen on paper as it does not actually produce any discernible image but rather indicates a potential for an image. The *n setgray* and *n setlinewidth* operators control the shade of grey (0.0 black; 1.0 white) and the linewidth of the constructed path respectively and the *stroke* operator records the image.

2. Drawing the arrowhead :- again the requirement is to start the construction of the arrow shape with a *newpath* command and an *x y moveto* to establish the starting point. The three subsequent commands are all based on the *dx dy rlineto* operator which allows a relative movement by *dx* and *dy* from the current point to a new point which then itself becomes the current point. The path is terminated with the *closepath* operator the grey level set with the *n setgray* operator and then the shape blocked in with the *fill* operator.

3. Placing the text :- Writing text requires the selection of a font style and the *fontname findfont* operator locates the font in the font library, *10 scalefont* scales the font to 10 points and *setfont* defines this as the current font style. It should be noted that PostScript is case sensitive and in general all program commands should be written in lowercase to denote user written code. There are a number of system functions which can be accessed and these are capitalised as in this example with the name of the selected font - 'Helvetica'. The current point is set using the *x y moveto* operator and the text is recorded on the page using the *textstring show* operator. The text *textstring* is defined by placing it within rounded parenthesis as in the current example *(You are here) show*. The colour of the recorded text in this example will be black as the *0 setgray* operator is still current.

4. Printing the page :- The preceding instructions only record positions on the page which have been written in the selected colours. In order to obtain the output on paper it is necessary to force the laser printer to throw a page and this is simply done with the *showpage* operator.

This simple program demonstrates the essence of PostScript programming in the use of the stack and in the use of simple graphics commands. However, the language is much more

flexible and can be made more user friendly once a number of other features have become familiar.

Essential PostScript Features

The most important features of the language are the use of variables and in common with most other structured languages, the use of procedures. There are many other features which will be mentioned here but the ultimate source is the 764-page PostScript Language Reference Manual (ISBN 0-201-18127-4).

Variables allow the written program to be read, maintained and modified more easily as well as allowing shorter programs to be written. The way that variables are defined is to precede a name with a '/' character and to follow the name with a value and the *def* operator such as in */x 20 def*. The effect of this command is firstly to put */x* onto the stack then *20* onto the stack and when the interpreter sees the *def* operator it removes both *20* and */x* from the stack and stores the value *20* in */x* and keeps a record of */x* in a dictionary. The value can then be used by the program by referring to the name only, without the '/'.

For example, to draw a black box with the bottom left corner at (20, -100) and the top right corner at (45, 0) and to draw a medium grey box with the bottom left corner at (120, 50) and the top right corner at (250, 60) requires the following commands:-

```
/x 20 def                   %% define variables x, y, dx, dy and black
/y -100 def
/dx 25 def
/dy 100 def
/black 0 def
newpath                     %% start path construction
        x y moveto          %% establish current point
        0 dy rlineto        %% draw box using relative commands
        dx 0 rlineto
        0 dy neg rlineto    %% neg toggles the sign of the top number on stack
        dx neg 0 rlineto
closepath                   %% terminate the path
black setgray               %% select the shade of grey - black
fill                        %% fill the box with black

/x 120 def                  %% define variables x, y, dx, dy and grey
/y 50 def
/dx 130 def
/dy 10 def
/grey 0.5 def
newpath                     %% start path construction
        x y moveto          %% establish current point
        0 dy rlineto        %% draw box using relative commands
        dx 0 rlineto
        0 dy neg rlineto
```

```
        dx neg 0 rlineto
closepath                   %% terminate the path
grey setgray                %% select the shade of grey - black
fill                        %% fill the box with black

showpage                    %% print result
```

Using the variables it is then possible to make the program more modular. If the program is required to draw a number of boxes then a procedure can be written that has a common structure and only needs to be given different values for the box position and the colour of the box. A procedure is defined in much the same way as a simple variable but the value in this case is a set of PostScript instructions bracketed with *{ }*. For example to simply draw a box would require the following procedure:-

```
/drawbox
        {newpath                %% start path construction
                x y moveto      %% establish current point
                0 dy rlineto    %% draw box using relative commands
                dx 0 rlineto
                0 dy neg rlineto
                dx neg 0 rlineto
        closepath               %% terminate the path
        colour setgray          %% select the shade of grey - 0 black to 1 white
        fill                    %% fill the box with black
        } def
```

This is stored by the interpreter in a dictionary where the value of */drawbox* is the list of instructions. Using this procedure to draw both the black and grey boxes reduces the first program to a simple definition of the required variables, followed by an execution of the procedure. As before with the variables, access to the procedure is accomplished through the use of the name without the '/' character. In this case the interpreter looks up the instructions and executes them as though they had been directly written. So, apart from the actual procedure */drawbox*, the initial program reduces to

```
/x 20 def                   %% define variables x, y, dx, dy and colour
/y -100 def
/dx 25 def
/dy 100 def
/colour 0 def
drawbox                     %% draw the black box

/x 120 def                  %% re-define variables x, y, dx, dy and colour
/y 50 def
/dx 130 def
/dy 10 def
/colour 0.5 def
drawbox                     %% draw the grey box

showpage                    %% print result
```

If it is known that a series of identical boxes are to be drawn, then it is possible to use a simple looping facility to change the value of variables and therefore the position of the box. This can be accomplished with the '*start finish stepsize { } for*' command. This command allows *start, finish* and *stepsize* to be real numbers and the *for* operator executes the set of instructions *{ }* for every value from *start* to *finish* inclusive increasing the initial value by *stepsize*. Each incremental value is pushed onto the stack and if the executed set of instructions do not remove them they will remain there until the program is terminated. To use those values placed on the stack requires them to be stored in a variable. If the required variable name is say *j*, and */j* is pushed onto the stack it will be seen that the value and the variable name are in the wrong order. As seen earlier the format for storing values requires the value to be put onto the stack *after* the name of the variable. Therefore a means is required to swap the two entries on top of the stack and this is achieved with the *exch* operator. Thus when using the for-loop construct it is important that the first instruction executed is */j exch def* as shown in the following example.

```
0 10 2
        {/j exch def
        . . .
        } for
```

The execution of this program segment would be to push 0, 10, 2 and the instructions *{ }* onto the stack, for the interpreter to see the *for* operator which effectively removes all the items from the stack except the value 0 (the start condition). Then the set of instruction *{ }* is executed and initially the value 0 is stored in the variable *j* due to the use of the *exch* operator. The rest of the instructions would be executed and may or may not leave other items on the stack but when the last instruction is finished the *for* operator increments the loop value (0) by the stepsize (2) and if it is not greater than the finish condition (10), pushes the result onto the stack (2). Thus the values that are stored in *j* in this example are sequentially 0, 2, 4, 6, 8, and 10.

So applying this to the problem of drawing a series of identically sized and coloured boxes, one can use the loop counter to change the x and y starting co-ordinates as follows:-

```
dx 20 def
dy 30 def
colour 0 def
0 10 2
        {/j exch def
        /x j 20 mul def     %% takes j and 20 mul(tiplys) them and puts result into x
        /y j 50 mul def
        drawbox
        } for
```

This routine would draw six black-filled boxes 20 by 30 points, with their bottom left corners at co-ordinates of (0, 0), (40, 100), (80, 200), (120, 300), (160, 400) and (200, 500).

This last example highlights the method of mathematical computation within PostScript. Rather than using the standard symbols +, -, x, and ÷, PostScript uses the operators *add*, *sub*, *mul* and *div*. In each of these operations two values are required on the stack for the mathematical operation to take place, for example if the stack contains the values 15 and 2 and *div* is executed, the interpreter will remove 2 and 15 and push 7.5 back on. This is obviously a real number computation. PostScript also has integer division using *idiv* and this would have taken 2 and 15 off the stack and pushed 7 back on instead.

The most commonly used maths operators, and their required syntax are listed here:-

stack	stack	operator	result	
num_1	num_2	add	sum	%% num_1 plus num_2
num_1	num_2	sub	difference	%% num_1 minus num_2
num_1	num_2	mul	product	%% num_1 times num_2
num_1	num_2	div	quotient	%% num_1 divided by num_2
int_1	int_2	idiv	quotient	%% integer divide
int_1	int_2	mod	remainder	%% int_1 mod int_2
	num_1	abs	num_2	%% absolute value of num_1
	num_1	round	num_2	%% round num_1 to nearest integer
	num	sqrt	real	%% square root of num
	-	rand	int	%% generate pseudo-random integer
	int	srand	-	%% set random number seed

The ability to use mathematical operations is fundamental to the usability of the language and allows one to further develop familiarity with the system by translating the scaling of drawings and text sizes from points to say millimetres. The scaling of PostScript graphics is controlled by the use of a matrix called the current transformation matrix (CTM). The is a six element array of the form *[a b c d t_x t_y]* which transforms a co-ordinate pair (x,y) into another co-ordinate pair (x',y') according to the linear equations:

$$x' = ax + cy + t_x \qquad y' = bx + dy + t_y$$

The default settings of the matrix parameters depends upon the particular laser printer and the selected page type and resolution on that printer. It is not necessary to know the actually values that these can take so long as the method for changing them is understood. As stated earlier the default co-ordinate system is to define each x or y unit as $^1/_{72}$ of an inch. To convert the co-ordinate units to millimetres simply requires the CTM to be scaled using the s_x s_y *scale* operator by setting both s_x and s_y to $^{72}/_{25.4}$. The following example shows the same command namely *1 0 moveto* under the application of various scale factors:

```
initmatrix          %% initialises the CTM to the default settings
1 0 moveto          %% moves 1 point along the x axis

72 72 scale         %% transforms the default CTM by 72 times in x, y axes
```

1 0 moveto	%% moves 1 *inch* (72 points) along the x axis
0.03937 0.03937 scale	%% transforms the modified CTM by $^{1}/_{25.4}$ in x, y axes
1 0 moveto	%% moves 1 *mm* (2.835 points) along the x axis

In principle these translations are completely transportable from one device to another, i.e. any program written to draw a shape a certain size on a Personal LaserWriter will essentially draw the same shape the same size on a QMS laser printer, a Linotronics laser printer or on any printer with a PostScript interpreter. For the purposes of writing masks for Random Phase Plates (RPP) however, this transportability is insufficient due to the way in which the PostScript graphics attempts to be transparent to the user. This is simply because very small structures are required in RPPs and the default CTM is already a transformation from the machine device space and there is the possibility that the sizes drawn on different machines will be different at the pixel level.

The machine device space on varying machines have various resolutions available which change the CTM settings. The Personal LaserWriter printer which is used to test RPP patterns has a maximum resolution of 300 dots per inch (DPI) but the acetate RPP masks are produced on a Linotronic 330 or 630 printer at a resolution of 2540 DPI. When writing programs this difference has to be taken into account and the approach taken is to force the printer to write at the pixel level to the nearest integer number of pixels for any size of structure to be printed. For example when testing a line structure with a width of say 300 μm, the LaserWriter would be set to draw the line 4 pixels wide which actually is 339 μm wide, whereas the Linotronic would draw a line 30 pixels wide and achieve the required spacing exactly. Further the exact position of the line drawn is important. The pixel level is essentially a digital representation of the required image. The image can be specified so as to start at any real valued (x, y) co-ordinate and this can result in adjacent pixels being unintentionally marked. This is shown in Figure 19 where the light shaded rectangle has been drawn with bottom left and top right co-ordinates respectively of (0.9, 0.6) and (3.0, 2.0) but the actual size of the more heavily shaded marked area is from (0,0) to (3.999, 2.999).

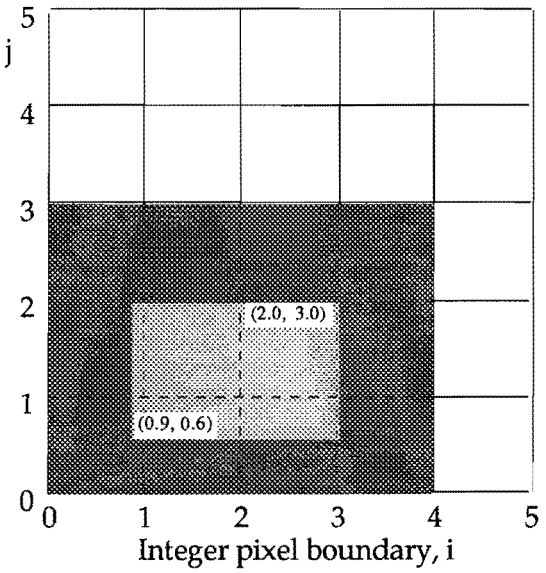


Figure 19. Demonstration of over-sized graphics due to real value CTM settings

The physical reason for this is that any real valued line or shape drawn through a co-ordinate (x, y) such that it intersects a

integer pixel (i, j) within the region of $i \leq x < i + 1$ and $j \leq y < j + 1$ then that pixel will be marked. The lower left co-ordinate (0.9, 0.6) is in the region of pixel (0, 0) and the upper right co-ordinate (3.0, 2.0) although on the boundary of pixel (3, 2), encompasses the region upto but not including the boundary (4, 3).

For this reason the standard practice has been to define all graphics movements to be referenced to the pixel boundaries but to also use mechanisms in the program to make such references still appear to in familiar units of millimetres for ease of operation.

To do this one needs to know the pixel size of the system being used and this can calculated using the PostScript operator *dx dy dtransform dx' dy'*. This operator takes the dx and dy distances and transforms them using the CTM into a corresponding distance in device space. Therefore using the standard unit size of 72 points per inch and transforming it results in modified distances in pixels per inch or DPI. Thus the short routine below calculates this and the corresponding pixel siz:-

```
/devicedpi 72 0 dtransform pop def
/deviceresolutionmm 25.4 devicedpi div def
```

The first command here pushes */devicedpi 72* and *0* onto the stack and *dtransform* removes *0* and *72* then replaces them with the modified values *dx'* and *dy'*. This last value is discarded with the use of the *pop* operator, with the final result that the value *dx'* is stored in the variable 'devicedpi' on execution of the *def* operator. The second command simply uses the devicedpi value and divides it into 25.4 in order to calculate the size of each pixel in millimetres, and it is this value which is stored in the variable *'deviceresolutionmm'*.

To achieve the required use of millimetres and pixel operation a procedure is required which converts from one to the other automatically. This can be done with the procedure definition *'mm'* which uses the calculated pixel size in millimetres and divides this into the number on top of the stack. This leaves a real number on the stack which is then made into the nearest integer with the *round* operator and this number is left on the stack.

```
/mm
        {deviceresolutionmm div round
        } def
```

In programming distances one simply uses the 'mm' routine as follows

```
/dx 10 mm def
```

This is easy to read as required and also converts the specified value to the nearest integer value of pixels. To make full and proper use of this method requires that the co-ordinates that one uses once transformed by the CTM are guaranteed to be situated on pixel boundaries. The

solution is to force the CTM to represent the pixel structure by setting it to [0 1 1 0 0 0]. This forces any co-ordinate to start on a pixel boundary and to allow more exact positioning of graphics to be made.

One final thing should be noted concerning the drawing of graphics: even with the CTM defined to unity scaling, and using the *'mm'* procedure it is still possible to draw oversized structures due to the pixel boundary problem discussed earlier. The best solution is to calculate the distances and when using them for drawing, subtract 1 (pixel) from the distance. For example if a square 400 μm on a side is required on a printer with only 300 DPI resolution the mm procedure will convert this to 5 pixels and if this value is used directly it will result in a structure actually 6 pixels on a side which is over 500 μm. On a printer with 2540 DPI resolution, mm will convert the distance to 40 pixels but again when drawing this will fill in 41 pixels and cause 410 μm squares to be drawn. Subtracting one pixel from the mm calculation results in squares of 430 μm and 400 μm for the different DPI resolutions which obviously is much closer to the initial requirements.

Example PostScript Program

Beyond the above mentioned features there are a number of standard statements that one needs to write in order to conform to the standard conventions of a Postscript program. These generally commence with a "%%" form and are demonstrated in the example program below which generates a Circular Random Phase Plate.

```
%!PS-Adobe-3.0
%%Title: (High resolution Circular RPP Generator.)
%%Creator: (Word 4 - Text Only.)
%%CreationDate: (Tuesday, 23rd June 1992)
%%For: (D. A. Pepler - Rutherford Appleton Laboratory)
%%Orientation: Landscape
%%BoundingBox: 0 0 842 595
%%DocumentNeededResources: font Helvetica
%%Requirements: resolution(2540,2540)
%%Pages: 1
%%EndComments

%%BeginDefaults
%%EndDefaults

%%BeginProlog

%**********PROCEDURES**********

/initialise
        {/black 0 def
         /white 1 def
         /pi 3.14159265359 def
         /str 80 string def
```

```
        /devicedpi 72 0 dtransform pop def
        /deviceresolutionmm 25.4 devicedpi div def

        /xpage 198.0 def                % A4 page in mm
        /ypage 290.0 def
        /xpage xpage deviceresolutionmm div round cvi def
        /ypage ypage deviceresolutionmm div round cvi def

        /maskradius 90.0 def                        % Mask size in mm
        /maskradius maskradius deviceresolutionmm div round cvi def
        } def

/resolutiongrid
        {/dotsizeminus dotsize 1 sub def
        /colour 1 def
        0 1 gridsize 1 sub
                {/y exch dotsize mul def
                /colour colour 1 sub abs def
                0 1 gridsize 1 sub
                        {/x exch dotsize mul def
                        colour setgray
                        newpath
                                xcord x add ycord y add moveto
                                0 dotsizeminus rlineto
                                dotsizeminus 0 rlineto
                                0 dotsizeminus neg rlineto
                        closepath
                        fill
                        /colour colour 1 sub abs def
                        } for
                } for
        } def

/resolutionchart
        {/xcord xpage 5 mm add def
        /ycord 0 def
        /dotsize 1 def
        /gridsize 8 def
        1 1 gridsize 1 sub
                {/dotsize dotsize 2 mul def
                } for

        resolutiongrid

        /ycord ycord gridsize dotsize mul add def
        /dotsize dotsize 2 idiv def
        resolutiongrid

        /xcord xcord gridsize dotsize mul add def
        /ycord ycord gridsize 2 idiv dotsize mul add def
        /dotsize dotsize 2 idiv def
        resolutiongrid

        /xcord xcord gridsize 2 idiv dotsize mul add def
        /ycord ycord gridsize 2 idiv dotsize mul sub def
        /dotsize dotsize 2 idiv def
        resolutiongrid
```

```
        /xcord xcord gridsize 2 idiv dotsize mul sub def
        /dotsize dotsize 2 idiv def
        resolutiongrid

        /ycord ycord gridsize dotsize mul add def
        /dotsize dotsize 2 idiv def
        resolutiongrid

        /xcord xcord gridsize dotsize mul add def
        /ycord ycord gridsize 2 idiv dotsize mul add def
        /dotsize dotsize 2 idiv def
        resolutiongrid

        /xcord xcord gridsize 2 idiv dotsize mul add def
        /ycord ycord gridsize 2 idiv dotsize mul sub def
        /dotsize dotsize 2 idiv def
        resolutiongrid
        } def

/mm
        {deviceresolutionmm div round
        } def

/showtitle
        {/Helvetica findfont 2 mm scalefont setfont
        5 mm 5 mm moveto
        white setgray
        (Round RPP;  Tuesday, 23rd June 1992. ) show
        elementsizemm str cvs show
        ( mm elements. Area fill : Width - 1. Resolution ) show
        devicedpi str cvs show
        ( dpi / ) show
        deviceresolutionmm str cvs show
        ( mm.) show
        ( Total ) show
        totalblack str cvs show
        ( B ) show
        totalwhite str cvs show
        ( W) show

        black setgray
        /Helvetica findfont 5.0 mm scalefont setfont
        200 mm 195 mm moveto
        (File : Adobe circle) show
        } def

/initpage
        {/mat [ 0 1 1 0 0.5 0.5 ] def
        mat setmatrix
        0 setlinewidth
        /Helvetica findfont 1.6 mm scalefont setfont
        black setgray
        newpath
                0 0 moveto
                0 ypage lineto
                xpage ypage lineto
                xpage 0 lineto
```

```
        closepath
        fill
        /centerx xpage 2 idiv def
        /centery radius 10 mm add def
        white setgray
        /totalblack 0 def
        /totalwhite 0 def
        } def

/rnd
        {rand 1073741824 idiv
        } def

/rnd360
        {rand 5965233 div
        } def

/oneblock
        {newpath
                centerx centery outerrad start last arc
                centerx centery innerrad last start arcn
        closepath
        fill
        } def

/ring
        {/totalringblack 0 def
        /totalringwhite 0 def
        /offset rnd360 def
        /inc 360.0 nosegs div def
        0 1 nosegs 1 sub
                {/n exch def
                rnd white eq
                        {/totalringwhite totalringwhite 1 add def
                        /start n inc mul offset add def
                        /last start inc add pixelindegrees sub def
                        oneblock}
                        {/totalringblack totalringblack 1 add def}
                        ifelse
                } for
        centerx radius add 1 mm add
        centery innerrad add moveto
        totalringblack str cvs show
        ( B ) show
        totalringwhite str cvs show
        ( W) show
        /totalblack totalblack totalringblack add def
        /totalwhite totalwhite totalringwhite add def
        } def

/rings
        {/annuli radius elementsize idiv 1 sub def
        0 1 annuli
                {/radnum exch def
                /innerrad radnum elementsize mul def
                /outerrad innerrad elementsize add 1 sub def
                /nosegs 2.0 radnum mul 1 add pi mul round cvi def
                /pixelindegrees 180 pi div outerrad div def
```

```
                ring
                } for
        } def

/doplate
        {/elementsize elementsizemm deviceresolutionmm div round cvi def
        elementsize 1 lt
                {/elementsize 1 def
                } if
        /radius maskradius elementsize div round cvi elementsize mul def
        /elementsizemm elementsize deviceresolutionmm mul def
        initpage
        0 srand
        rings
        showtitle
        resolutionchart
        showpage
        } bind def          % early binding should speed things up

%%EndProlog

%%BeginSetup
%%IncludeResource: font Helvetica
%%BeginFeature *PageSize A4
a4
%%EndFeature
initialise
%%EndSetup

%***** Main Program *****

%%Page: (1st) 1
/elementsizemm 4.0 def
doplate

%EOF
```

Appendix II

Random Number Generation

Within the PostScript language there is a pseudo-random number generator which can produce all integers from 0 to 2147483647 inclusive (2^{31}-1) in a random sequence before repeating over again. This range is great enough for all designs of random phase plates to date. Even the smallest structures used over the largest beam size (~50 micron squares, ~150 mm beam) have needed 'only' 9 x 10^6 calls to the random number generator.

Over the length of the generator the frequency of any number being produced is unity with no missing numbers so dividing the number by 1073741824 (2^{30}) will produce a random sequence of 0's and 1's (which are representations of the 0 or π phase in the phase plate). Over the whole range of the random number generator the ratio of 0's to 1's is 50:50 which is required for a good phase plate. However, a phase plate may have only a few hundred or even a few tens of calls to the pseudo random generator and the ratio is not likely to be exactly 50:50 or even consistent from one run to the next unless careful initialisation of the sequencer is provided each time a program is run.

Normally when a program is run by the PostScript interpreter the first value (or seed value) for the random sequencer is set by the turn-on time of the printer and thus the sequence that follows although random is different each time. To obtain an identical sequence every time the program is run the seed value is defined before making any calls on the generator by using the *srand* operator. Thus */srand 12 def* sets the initial value to 12 and defines the sequence that follows. However, arbitrarily setting the seed value generates sequences that may not produce a 50:50 ratio over a short sequence. For example if a plate has only 10 elements it is quite legitimate for a random sequence to have eight 0's and only two 1's giving a ratio of 80:20.

For normal phase plates either a single call or two calls on the generator per element are required. For example the simplest random phase plate just requires the phase of each cell to be random and so it uses *all* calls on the generator for this purpose. A PZP plate uses one call to determine the phase and another call to randomise the focal length. If the phase is the first call, the focal length adjustment the second and the next phase the third call etc. then all of the *odd* calls are used to achieve 50:50 ratio in the phase. If the phase is the second call, fourth, sixth calls etc. then all of the *even* calls are used to achieve the 50:50 ratio on the phase.

To overcome this aspect of design, the random number generator was called with many different seed values and a table generated indicating the appropriate seed value to use to obtain

as close to a 50:50 ratio as possible for a phase plate containing a certain number of elements and with the phase defined by *all* calls or either the *odd* or *even* calls to the generator.

Part of this data is shown in Table 2. The columns 'all', 'odd' and 'even' are generated using calls on the generator and display the number of calls needed to obtain a 50:50 phase ratio using all calls, the first, third fifth etc. and the second, fourth sixth etc. respectively. For example if a random phase plate requiring a single call to the generator (per element) is made with only 14 elements then the seed value could be 1 as the 'all' column contains the value 14. If a plate with 2 calls is made and has 14 elements then the seed value could also be 1 if the phase is determined by the odd calls or it could be 28 instead as both the odd and even columns contain the value 14.

Table 2. Number of random number calls to obtain 50:50 ratio for certain seed values

Seed 1			Seed 2			Seed 10			Seed 28		
Odd	Even	All	Odd	Even	All	Odd	Even	All	Odd	Even	All
2	6	10	2	2	4	2	2	4	14	4	2
4	8	12	16	38	66	6	4	52	16	6	4
6	12	14	18	40	68	8	24	54	18	12	6
8	28	16	20	50	70	16	26	64	32	14	20
10	30	18	30	52	72	18	64	98	34	16	22
12	32	24	32	58	88	20	66	100	36	46	26
14	48	164	62	62	120	26	70	102	40	48	28
16	50	166	64	76	122	28	72	120	46	488	32
18	58	202	66	86	124	30	74	124	54	516	42
88	108	340	68	88	152	34	84	134	60	518	74
90	110	528	70	90	182	68	86	168	62	520	76
92	112	530	76	112	186	84	94	174	64	522	92
96	114	552	78	114	188	88	96	218	66	534	94
242	124	554	92	126	268	92	98	224	68	538	4678
244	126	556	96	128	270	2488	476	226	70	540	4680
248	128	558	98	130	304	2502	478	240	2060	542	4698
250	138	698	106	132	340	2506	482	242	2070	548	4712

This deliberate modification of the random sequence to 'fix' the uniformity may seem as though there is a problem with the generator. It should be noted that we are considering relatively small numbers of calls and the only guarantee of a 50:50 ratio is after the generator has completed the full $2^{31} - 1$ calls. It should also be noted that any attempt to make the sequence 'more random' or 'more uniform' by say dividing the generated number by any prime number will almost certainly make the resultant sequence far *less* random.

Appendix III

Interference microscope techniques

RPP / PZP's require the phase shift imposed on the beam to be as close to π radians as possible. Deviation from this value will be detrimental to performance. A simple method of measuring the thickness of the deposited resist is by uses an interference microscope. The interference microscope views the light reflected from the etched surface of the plate and produces fringes positioned according to the height of the plate surface. Light incident on the phase step is reflected as shown in Figure 20.

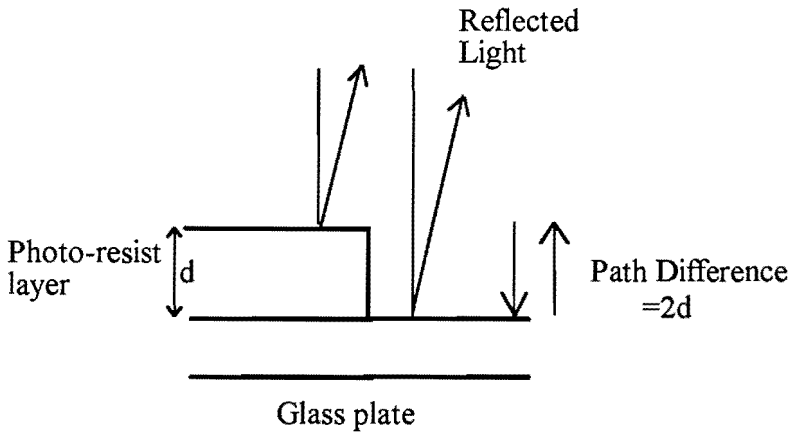


Figure 20. Reflection of light from the phase step

The relative path difference is given by $2d = m\lambda$; where m is the fringe shift and λ is the operational wavelength of the microscope.

The measurement procedure is as follows:

1) Place the plate to be measured on the microscope with one of its edges beneath the lens, and with the phase plate coating upwards. While gently moving the plate from side to side adjust the focus of the microscope until a definite edge can be seen. The sideways movement will make it easier to spot the edge of the plate when trying to focus. Slowly move the plate sideways until a series of interference fringes can be seen. Using the tilting controls adjust the position of the plate until the fringes appear either horizontal or vertical. Using both the sideways movement and focus try to find an edge running across the fringes as shown in Figure 21. For the most accurate results this should be as perpendicular to the fringes as possible.

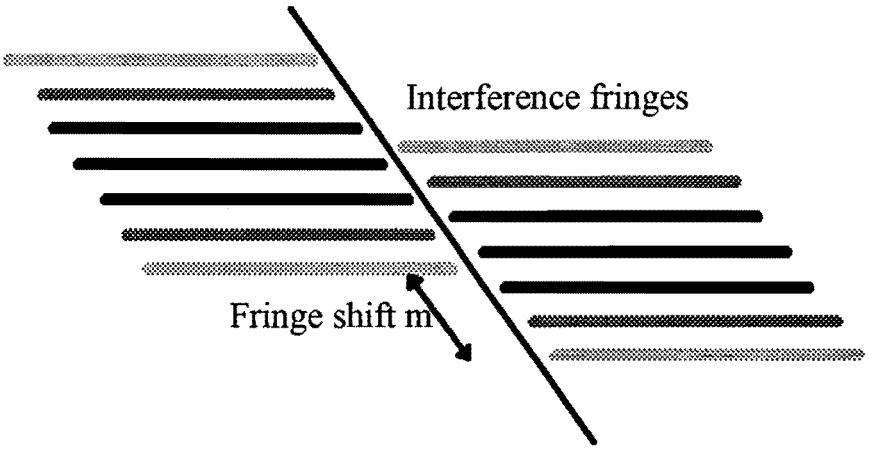


Figure 21. Generated fringe shift of interference microscope

2) Use Polaroid 559 colour film in the microscope's camera and set the exposure for ISO 80. Check the focus of the image using the cameras viewfinder and adjust as necessary. Carefully pull out the metal plate from the film cartridge until the blue line can be seen, check the image once more and start the exposure (the length of the exposure will be set automatically by the camera system). Once exposed, slide the metal plate back to cover the film. Remove the film from the camera and wait for 30 seconds for the image to be developed. Peel off the backing paper and ensure that the photograph has a clear image. Check the viewfinder once more and if nothing has moved place a calibrated green interference filter (526 nm) into the microscope's light source. Pull out the metal plate again and expose another photograph.

3) From the first photograph count the number of fringes shifted at an edge. This initial estimate will make it easier to determine the exact value. On the second photograph measure the distance between a large number of fringes (assuming good resolution). Using this value calculate a single fringe separation.

4) Carefully cut along the edge of the fringes on the second photograph. Slide the two pieces along each other until the fringes are reconnected and mark a point on the edge of one of the pieces to represent the distance moved. Measure this distance and divide by the single fringe separation calculated in 3, this will give a value for m. Put this into the equation (6) along with the wavelength of the interference filter to calculate the depth of the resist.

Appendix IV

Current suppliers

Equipment

Chrome coated quartz plates Quartz plates Glass plates	Hoya Europe, 54/58, Uxbridge Road, Ealing, London. W5 2TL
Fume cupboard Filters	Astecair Environmental Systems Ltd., 31, Lynx Crescent, Weston Industrial Estate, Weston Super-Mare, Avon. BS24 9DJ
Oven	Jencons (Scientific) Ltd., Cherrycourt Way Industrial Estate, Stanbridge Road, Leighton Buzzard, Bedfordshire. LU7 8UA
Acetate mask printing	HBS Design Associates Ltd. Suite 1 Phoenix Brewery, Bartholomew Street, Newbury, Berkshire. RG14 5QA
Light box	RS Components, PO Box 99, Corby, Northamptonshire. NN17 9RS

Mercury lamps Power supplies	L.O.T.-Oriel Ltd., 1, Mole Business Park, Leatherhead, Surrey. KT22 7AU

Chemicals

Visible photo-resist (S1813 SP-15) Universal Developer	Shipley Chemicals Ltd., Herald Way, Coventry. CV3 2RQ
UV photo-resist (P10) Spectrum Positive Resist Thinner Methyl Isobutyl Keytone (MIBK)* Isopropanol Alcohol (IPA)* Acetone*	Micro-Image Technology Ltd., Amber Business Centre, Riddings, Alfreton, Derbyshire. DE55 4DA

*Available from RAL central stores