(19) (11) **NZ 323234 A**

(12) **PATENT DOCUMENT**

(40) Publication date **29/09/1999** (51)Int. Cl.: **G06K 15/00**
(21) Application number: **323234**
(22) Application date: **15/11/1996**
(85) National phase entry date: **1998-05-08**
**PCT/**
(86) PCT application number: **US1996/018252**
(87) PCT publication number: **WO1997/018530**

(30) Priorities:
**US 95558421 19951116**

(71) Applicant:
**DATASTRIP PRODUCTS INCORPORATED,**
**12110 Sunset Hills Road, US**

(72) Inventors:
**Gerety, Eugene P**

(54) Title:
**Aligning machine-readable codes to printer pixels**

(57) Abstract:
This printer processor (100) is one in which accuracy of a printed image is essential. The printer processor (100) receives exact pixel characteristics from a target printer (180), and instructs the target printer (180) to exactly reproduce the desired image after eliminating any distortions introduced by scaling or rotational factors.

New Zealand No 323234
International No. PCT/US96/18252

TO BE ENTERED AFTER
ACCEPTANCE AND PUBLICATION

Priority dates 16 11 1995,

Complete Specification Filed 15 11 1996

Classification (6) G06K15/00

Publication date 29 September 1999

Journal No 1444

NEW ZEALAND

PATENTS ACT 1953

# COMPLETE SPECIFICATION

Title of Invention

Aligning machine-readable codes to printer pixels

Name, address and nationality of
applicant(s) as in international
application form
DATASTRIP PRODUCTS INC , 12110 Sunset Hills Road, Reston, Virginia 22090,
United States of America

# ALIGNING MACHINE-READABLE CODES TO PRINTER PIXELS

## Field of the Invention

This invention relates to precise printing of one dimensional and two dimensional codes used for identification purposes and, more particularly, to a method for compensating and correcting for geometric distortions introduced by printers to ensure that the printed codes are accurate and machine readable.

## Background of the Invention

Identification codes, such as one dimensional codes (Bar codes), and two dimensional codes (Datastrip codes), are widely used for purposes of identifying items, products, and/or individuals, as the codes are read by intelligent machines. However, such machine readable codes must be printed very accurately in order to ensure that the code(s) are uniformly and accurately read by the intelligent machine(s). Even small errors in the printing process can result in a code that is unreadable by a particular machine, thus making the coding process unusable for its intended purpose.

For example, two dimensional codes (2D codes), such as utilized by Datastrip Products Inc., must generally be precisely aligned to a particular printer's pixel array when printing the code to avoid geometric distortion of the code bitmap. The geometric distortions result from small scale factors applied in the horizontal and vertical dimensions by some printer drivers and printing applications (such as Windows print drivers, and Windows graphics applications). Slight scaling (e.g., 99.5% to 100.5%) is not uncommon in the bitmap printed from Windows. The scale factors cause the code bitmap to be expanded or shrunk by replicating or deleting rows and/or columns of dots in the bitmap. This destroys low-level, localized, geometric characteristics of such codes, making them harder to read, less reliable and possibly rendering the codes unreadable.

A significant problem with printing bitmap images using commercially available software in device-independent operating environments such as Microsoft Windows (TM) is that slight scale and/or rotation factors are often applied to the printed output produced by this software. For most word processing and illustration purposes, these scale and/or rotation factors are unnoticeable. However, when attempting to print bitmap images of encoded data which require accurate geometric characteristics, the pixel shifts, and added or dropped rows of pixels distributed throughout the printed image as a result of slight scaling or rotation, can produce significant geometric distortions

An example of attempts to correct for printer scale and/or rotation factors in word processing applications is shown in U.S. Patent No 5,459,828, granted to Zack, et al on October 17, 1995. This patent teaches a method of producing a raster font from a contour font entailing the steps of deriving font metrics and character metrics of font characters in terms of arbitrary font units, scaling the font characters to a selected size and output resolution and altering the thickness of vertical and horizontal strokes of each character to a desired thickness. In essence, the thrust of the invention in the Zack et al patent is to modify the appearance of a particular character in order to make the character appear uniform to a reader.

In contrast, it is the object of the instant invention to very accurately print a particular machine readable code by correcting and compensating for distortions introduced by various printing machines , or to at least provide the public with a useful choice

Hereinafter, the term "interpretive printer" will be used to refer to a printer which executes a page description language within the printer itself, and the term "non-interpretive" printer will be used to describe a printer which merely prints an externally created image. An example of an interpretive printer is a Postscript (TM) printer (Postscript is a registered trademark of Adobe Systems, Inc.), such as the Apple Laserwriter, produced by Apple Computer, Inc., which executes a

Postscript language interpreter within the printer. Page descriptions sent to the printer are actually Postscript language programs which, when interpreted by the language interpreter in the printer, causes an image to be printed by the printer. An example of a non-interpretive printer is the Hewlett Packard Laserjet Series IV, produced by Hewlett Packard, Inc.

## Summary of the Invention

The present invention provides techniques for printing bitmap images of encoded data using standard, commercially available application software such as drawing programs or word processing programs, such that the bitmap images are exactly registered to a target printer's pixel grid independent of any rotation or scaling which may have been applied by the application software. Hereinafter, such bitmap images will be referred to as having been "pixel-registered" with respect to the target printer.

It is a feature of the invention that predetermined images to be printed are scaled to fit a predetermined number of pixels as identified by the particular printer being used.

It is a further feature of the invention that the resolution of the target printer is known before transmitting the image to the target printer, and compensation is applied to correct for printer induced scaling and rotation factors.

These and other aspects of the invention will become apparent from the following description used to illustrate a preferred embodiment of the invention in conjunction with the accompanying drawings in which:

## Brief Description of the Figures

Figures 1A-1D illustrate the effect of scaling on a bitmap in accordance with the invention;

Figure 2 is a block diagram of a computer-based system for printing pixel-registered bitmap images of encoded data on a printer according to the invention;

Figure 3 is a flow diagram showing processes and data flow for a method of printing pixel-registered bitmap images of encoded data on an interpretive printer according to the invention; and

Figure 4 is a flow diagram showing processes and data flow for a method of printing pixel-registered bitmap images of encoded data on a non-interpretive printer according to the invention.

Detailed Description of the Invention

Referring now to Figures 1A-1D, Figure 1A illustrates the bitmap for a particular two dimensional code, and Figure 1B illustrates how that code should appear when printed. As shown, the bitmap in Figure 1A consists of a plurality of 3x3 arrays, where an array of all "1" bits results in a black square being printed, while an array of all "0" bits results in a white square, i.e., no printing in this area.

In contrast, Figure 1C illustrates a bitmap after scaling by application software located in the CPU or in the printer. Note how the bitmap has been distorted, which results in the printed image shown in Figure 1D. These distortions can render the printer image unreadable by a machine, which is the problem solved by the instant invention.

Figure 2 shows a computer based system 100 for printing pixel-registered bitmap images, which includes a computer 140, application software 150 on the computer 140, image generation software 160 on the computer 140, print driver software 170 on the computer 140, and a printer 180. A user provides data for encoding 110 to be encoded into a bitmap image via the image generation software 160. Other user data 120 may be provided in combination with the data for encoding 110 for processing by the application software 150. Print driver software 150 controls

the flow and formatting of data generated by the image generation software 160, and the application software 160 to the printer 180. The user data 110, and other user data will usually reside on a storage device 130 connected to the computer 140.

Typically, a user will use the application software 160 (such as a word processor or drawing program) to process other user data 120. The result of this processing is a printed page. After processing of the data for encoding 110 by the image generation software, the user can combine the resultant image with the other user data 120, using an import facility of the application software. (Such import facilities are widely known to and understood by those of ordinary skill in the art, and will not be further elaborated upon herein). The combined output is then sent by the application software 150 via the print driver software 170 to the printer 180.

Figure 3 is a flow diagram 200, showing processes and data flow for a method of printing pixel-registered bitmap images on an interpretive printer. User data 210, provided by a user for encoding into a bitmap image, is encoded by an image generation process 230 according to a set of parameters (e.g., intended printer dot-per-inch rating, size and shape for image features, etc.). The image generation process 230 produces an image description 240 comprising a protective image shell 242, and an image portion 244. The protective image shell 242 is essentially a program in a page description language capable of being interpreted by an interpretive printer 280, for analyzing the printer's physical pixel array characteristics, and resizing and reorienting the image portion 244, as necessary to align exactly with an integer multiple of the printer's pixel resolution. That is, if the printer is a 300 dpi (dot per inch) printer, but has had a non-integer scale factor applied by an application program (e.g., 205%), then the protective image shell will detect the scaling and cause the image to be printed at the nearest integer scale factor (e.g., 200%), which does not shrink the image 244. The protective image shell 242 performs similarly with rotation factors, causing the image 244 to be

printed at only integer multiples of 90 degrees. Application software 260 processes additional user data 250, and imports the image description 240 for inclusion into printed output. Output from the application software 260 output is passed to a print driver 262 which formats it into a page description 270 for a printer. A duplicate image description 272, substantially identical to the image description (240 above) is incorporated into the page description 270. The page description 270 is essentially a program to be executed by a page description language interpreter resident in the interpretive printer 280 to produce printed output.

Figure 4 is a flow diagram 300 showing process and data flow for a method of printing pixel-registered bitmap images of encoded data on a non-interpretive printer 390. User data 310 is provided by a user for encoding into a bitmap image by image generation software 320. Additional user data 330 is provided by the user for processing by application software 340, such as a word processing program or a drawing program, to produce printed output on the non-interpretive printer 390. The image generation program 320 creates an image description 350 comprising a set of image parameters 350A (e.g., intended size of the image pixels, orientation, etc.), and an image bitmap 350B representing an encoding of the user data 310. The application software 340 treats the image generation program 320 as in import filter for incorporating the user data 310 in bitmap encoded form into its printed output. However, in this case, the image description 350 produced by the image generation program 320 (acting as an import filter for the application software 340), is passed to a print driver 370 by a scale and rotate process 360, which performs much the same function as described hereinabove with respect to Figure 2 for the protective image shell 242. The scale and rotate process 360 queries the print driver 370 about physical printer characteristics and about print scaling and rotation setting made by the application software 340, and temporarily bypasses the settings made by the application software 340. The scale and rotate process then picks the nearest integer multiple of scaling to the scale setting made by the application software

360, at which the image bitmap 350B aligns exactly with a pixel resolution for the printer 390, and picks the nearest multiple of 90 degrees rotation to the rotation setting made by the application software 340, and sends the image bitmap to the print driver 370 at that setting. After sending the image bitmap 350B to the print driver 370, the scale and rotation settings made by the application software are restored. The print driver 370 produces a page image 380 formatted for printing by the non-interpretive printer 390.

The methods shown and described hereinabove with respect to Figures 3 and 4, produce substantially identical end results. The primary difference between the methods, is that the method of Figure 3 performs image registration by appending a registration program to the image description for execution within an interpretive printer by a page description language interpreter, while the method of Figure 4 performs the registration process inside the computer (e.g., 140, Figure 2) prior to transmission to the printer.

A specific example of Postscript code for the protective image shell is set forth below. More particularly, the basic scheme of the following code is broken into two parts. The first part tests the printer's resolution and determines some basic parameters about the currently selected print coordinate system, determining whether or not there has been unacceptable rotation or shearing of the coordinate system. If there has, the first part "crashes" (stops), causing a second error printout part of the code to execute (via the "stopped" directive between the two portions). The error printout part prints an image resembling the shape and orientation that the image would have had if it had been printed in the current coordinate system (but does not print the actual strip), and dumps a list of relevant values determined by the first part.

The first part of the code uses a progress variable (Progress) to indicate where a failure has occurred. If the progress variable is equal to 6, then there is no error. If it

is less than 6, then its value indicates how far the first part of the code got before detecting a problem.

Explanatory notes are embedded in the code below in the form of Postscript comments, which begin with a percent (%) sign.

```
%    Start of first part of protective image shell.  Fails
%    to complete if a problem is detected.  Leaves a trace
%    value (Progress) behind to indicate how far through
%    the first process the problem occurred.

{

%    Set the progress indicator to zero.

/Progress 0 def

%    Define newline function to be used in printing
%    the datastrip image such that if the progress
%    variable is equal to 6, then the strip image is
%    data on the stack (i.e., strip image data) is
%    printed normally using a newline procedure (nl).
%    If not, then the data on the stack is popped off
%    the stack and ignored.

/NL {Progress 6 eq {nl} {pop} ifelse} def

%    Set a variable 'DPI' equal to the expected
%    printer DPI.  This value is inserted by the
%    program which appends this code to the image
%    data.

/DPI 300 def

%    Transform the point (1 inch, 0 inches) via the
%    printer's default matrix to determine the
%    printer's actual resolution in DPI.  (Note:  In
%    the default coordinate system is expressed in
```

```
%    "points" where 72 points is exactly equal to 1
%    inch.

72. 0. matrix defaultmatrix dtransform dup

%    Determine the actual printer DPI resolution by
%    examining the result of the foregoing
%    transformation.  Requires two tests, since the
%    printer can be in either portrait or landscape
%    mode.  Whichever test (x or y) yields a non-zero
%    value provides the result.

/ActualDPI exch abs def
     0. eq {/ActualDPI exch abs def} {pop} ifelse

%    Determines a useful value equivalent to 8 points
%    in the printer's native resolution.

/EightPoint 8. 72. div ActualDPI mul def

%    Initialize an error message to "*No Error*".

/StripErrMsg (*No Error*) def

%    Define functions for selecting the maximum or
%    minimum of two values on the Postscript stack.
/Max {
     /mxop2 exch def dup
     /mxop1 exch def
      myop2 gt myop2 exch {pop mxop1} if}def
/Min {
     /mnop2 exch def dup
     /mnop1 exch def
      mnop2 lt mnop2 exch {pop mnop1} if} def

%    Define a function for extracting the sign of a
%    numeric value on the Postscript stack.

/Sign {dup 0. ne {dup abs div} if cvi} def
```

```
%    Define a function for extracting a scale factor
%    by comparing the value on the Postscript stack to
%    the requested (expected) DPI rating.

/ScaleFactor
     {dup 0. ne
          {dup Sign exch DPI div 0.98 abs cvi 1 Max
          mul} {cvi}
     ifelse} def

%    Define an error processing function which copies
%    an error message on the Postscript stack into
%    "StripErrMsg" and executes the "stop" directive,
%    aborting the first part of the code.

/StripError {/StripErrMsg exch def stop} def

%    Function to test for a (0, 0) value on the
%    Postscript stack.

/ZeroCheck {0. eq exch 0. eq and} def

%    Function to replace a value with zero if it is
%    less than 1/50th of another value.

/Thresh {
     /Thop2 exch def dup Thop2 exch div abs 0.02 lt
     {0.} {Thop2} ifelse} def

%    Function which takes an (x, y) point and uses the
%    Thresh function to zero out either the x or y
%    portion if it is less than 1/50th of the other.


/Squelch {
     /Sqop2 exch def dup
     /Sqop1 exch def abs
      Sqop2 abs gt
```

```
{Sqop1 Sqop2 Thresh} {Sqop2 Sqop1 Thresh exch} ifelse}
def

%    Function which returns "true" if axis shearing
%    has occurred.

/ShearErrCheck {0. eq {} .eq} {0. ne} ifelse} def

%    Transform the point (0, 0) to determine the
%    current coordinate system's origin.

0 0 transform
     /udy0 exch cvi def
     /udx0 exc cvi def

%    determine response of the current coordinate
%    system to pure "x" offsets and pure "y" offsets,
%    using 1 inch (72 points) as the test value.

72. 0. dtransform
     /xyDPI exch def
     /xxDPI exch def
0.72. dtransform
     /yyDPI exch def
     /yxDPI exch def

%    No problems in initial setup.  Bump the progress
%    indicator.

/Progress 1 def

%    Test the coordinate system for zero width, zero
%    length, and collapsed axes.  Execute the error
%    procedure "StripError" if a problem is detected.

xxDPI xyDPI ZeroCheck {(Zero Width) StripError} if
xyDPI yyDPI ZeroCheck {(Zero Length) StripError} if
xxDPI yxDPI ZeroCheck {(XAxis Collapse) StripError} if
```

```
        xyDPI yyDPI ZeroCheck {(YAxis Collapse) StripError}
        if

        %    Got through the basic coordinate system testing.
        %    Bump the progress indicator.

/Progress 2 def

        %    Now "squelch" the coordinate system to remove any
        %    insignificant (i.e., less than 2%) rotation or
        %    shear factors.

        xxDPI xyDPI Squelch
             /xySql exch def
             /xxSql exch def
        yxDPI yyDPI Squelch
             /yySql exch def
             /yxSql exch def
        xxSql yx Sql Squelch
             /yxSql exch def
             /xxSql exch def
        xySql yySql Squelch
             /yySql exch def
             /xySql exch def

%    Got through the squelch processes.  Bump the progress
%    indicator.

/Progress 3 def

        %    Test for unacceptable coordinate system shear or
        %    rotation.  Execute the StripError procedure if a
        %    problem is found.

        xxSql xySql ShearErrCheck
             {(Shear/Rot. xx-xy) StripError} if
        yxSql yySql ShearErrCheck
             {(Shear/Rot. yx-yy) StripError} if
        xxSql yxSql ShearErrCheck
```

```
                {(Shear/Rot. xx-yx) StripError} if
            xySql yySql ShearErrCheck
                {(Shear/Rot. xy-yy) StripError} if

        %   Passed the shear and rotation tests.  Bump the
        %   progress indicator.

        /Progress 4 def

            %   extract scale factors from the "squelched" values

            /xxS xxSql ShearFactor def
            /xyS xySql ScaleFactor def
            /yxS yxSql ScaleFactor def
            /yyS yySql ScaleFactor def

        %   Got through the scale factor processing.  Bump the
        %   progress indicator.

        /Progress 5 def

            %   Define a coordinate system matrix in terms of
            %   integer scale factors for use when printing
            %   subsequent image data.

        /SSMatrix [xxS xyS yxS yyS udx0 udy0] def

            %   Convert from 72 dpi (user space) to device space,
            %   (i.e., the value "1" now refers to one pixel (or
            %   integer multiple thereof, depending upon the
            %   scale factors in SSMatrix.  By using this matrix
            %   for subsequent strip printing, the image data
            %   (which is defined in terms of printer pixels), is
            %   forced to be printed in perfect alignment with
            %   the printer's coordinate system at an integer
            %   scale factor.

                SSMatrix setmatrix
```

```
        %    Everything went OK.  Set the progress indicator
        %    to 6 and complete the first part of the code
        %    normally (i.e., without a "crash" or stop
        %    directive).

        /Progress 6 def

    }

    %    Test whether the first part of the code completed
    %    normally.  If not, execute the second part -- error
    %    processing.

    stopped {

        %    Restore and save the original coordinate system
        %    (and graphic environment).

        grestore
        gsave

        %    Create a horizontal scale factor based upon the
        %    width of the image data.  (The value for istrpwid
        %    is inserted by the program which appends this
        %    code to the image data).

        istrpwid 72 div 1 scale

        %    Create a pictorial representation of a strip
        %    outline in the current coordinate system, to show
        %    graphically how the strip would have been
        %    rotated, sheared, or scaled if it had been
        %    printed in this orientation.

        0.5 setlinewidth
            newpath
            0 0 moveto
            0 - 144 lineto
            72 - 144 lineto
```

```
        72 0 lineto
        closepath
        stroke

    %   Select Helvetica 8 point for printing error info

    /Helvetica findfont 8 scalefont setfont

        %   Finish off the pictorial strip
        %   representation.

        18 0 moveto
        36 0 rlineto
        0 -7 rlineto
        -36 0 rlineto
        closepath stroke

    %   Define text line height for message printing
    %   (9 points).

    /linestep -9 def

    %   Define left edge for text (5 points from left
    %   edge of strip).

    /leftedge 5 def

    %   Set initial position for message printing.

    leftedge linestep 2 mul moveto

    gsave

    %   Define a special "newline" function (SEnl) for
    %   "showing" a line of text on the stack, then
    %   stepping to the next line position.

    /SEnl {show grestore 0 linestep rmoveto gsave} def
```

```
        %    The fc lowing messages are printed
        %    uncona  ionally.

        (****ERROR*****) SEnl
        (ReqstdDPI=) show DPI 20 string cvs SEnl
        (ActualDPI=) show ActualDPI 20 string cvs SEnl
        (xxDPI=) show xxDPI 20 string cvs SEnl
        (xyDPI=) show xyDPI 20 string cvs SEnl
        (yxDPI=) show yxDPI 20 string cvs SEnl
        (yyDPI=) show yyDPI 20 string cvs SEnl

        %    The following messages are printed if the
        %    progress indicator is at least 3.

        Progress 3 ge {
             (xxSql=) show xxSql 20 string cvs SEnl
             (xySql=) show xySql 20 string cvs SEnl
             (yxSql=) show yxSql 20 string cvs SEnl
             (yySql=) show yySql 20 string cvs SEnl} if

        %    The following messages are printed if the
        %    progress indicator is at least 5.

        Progress 5 ge {
             (xxS=) show xxS 20 string cvs SEnl
             (xyS=) show xyS 20 string cvs SEnl
             (yxS=) show yxS 20 string cvs SEnl
             (yyS=) show yyS 20 string cvs SEnl} if

        %    Print the text description of what went wrong,
        %    followed by the value of the progress indicator.

        StripErrMag SEnl
        (Progress=) show Progress 20 string cvs show
        grestore

    }   if
```

It will be appreciated that changes and modifications are likely to occur to those skilled in the art, and it is intended in the appended claims to cover all those changes and modifications which fall within the spirit and scope of the present invention.

What Is Claimed Is

1 A printer processor implemented method for printing machine-readable code images on a selected printer having reproduction characteristics known and available, each machine-readable code image being defined by a code bitmap and having rectilinear areal geometric characteristics, the method including the steps of

introducing a code bitmap of a machine-readable code image to be printed into a printer output image bitmap by means of software which imposes a risk of inducing image bitmap distortion,

obtaining for the printer processor the known reproduction characteristics of said selected printer,

analyzing, by said printer processor, physical pixel array characteristics of said printer based on said known reproduction characteristics,

transforming the code bitmap of the machine-readable code image in the printer output image bitmap to resize and rotate the code image as required to align essentially exactly rectilinear areal geometric characteristics of the code image with integer multiples of said printer's physical pixel array characteristics, and

printing on the selected printer the code image defined by the transformed code bitmap in the printer output image bitmap

2 A method according to claim 1, further comprising the step of

after the step of analyzing physical pixel array characteristics of the printer and before printing the code image on the printer, transforming the code bitmap of the machine-readable code image in the printer output image bitmap to shear correct the code image as required to align essentially exactly rectilinear areal

geometric characteristics of the code image with integer multiples of said printer's physical pixel array characteristics

3 A printer processor implemented method for printing machine-readable code images on a selected printer having reproduction characteristics known and available, each machine-readable code image being defined by a code bitmap and having rectilinear areal geometric characteristics, the method including the steps of

introducing a code bitmap of a machine-readable code image to be printed into a printer output image bitmap by means of software which imposes a risk of inducing image bitmap distortion,

obtaining for the printer processor the known reproduction characteristics of said selected printer,

analyzing, by said printer processor, physical pixel array characteristics of said printer based on said known reproduction characteristics,

transforming the code bitmap of the machine-readable code image in the printer output image bitmap to resize and reposition the code image as required to align essentially exactly rectilinear areal geometric characteristics of the code image with integer multiples of said printer's physical pixel array characteristics, and

printing on the selected printer the code image defined by the transformed code bitmap in the printer output image bitmap

4 A method according to claim 3, further comprising the step of

after the step of analyzing physical pixel array characteristics of the printer and before printing the code image on the printer, transforming the code bitmap of

the machine-readable code image in the printer output image bitmap to rotate the code image as required to align essentially exactly rectilinear areal geometric characteristics of the code image with said printer's pixel array characteristics

5 A method according to claim 3, further comprising the step of

after the step of analyzing physical pixel array characteristics of the printer and before printing the code image on the printer, transforming the code bitmap of the machine-readable code image in the printer output image bitmap to shear correct the code image as required to align essentially exactly rectilinear areal geometric characteristics of the code image with integer multiples of said printer's physical pixel array characteristics

6 A printer processor implemented method for printing machine-readable code images on a selected printer having reproduction characteristics known and available, each machine-readable code image being defined by a code bitmap and having rectilinear areal geometric characteristics, the method including the steps of

introducing a code bitmap of a machine-readable code image to be printed into a printer output image bitmap by means of software which imposes a risk of inducing image bitmap distortion,

obtaining for the printer processor the known reproduction characteristics of said selected printer,

analyzing, by said printer processor, physical pixel array characteristics of said printer based on said known reproduction characteristics,

measuring a rotational difference between a pixel array defined by the code bitmap of the machine-readable code image in the printer output image bitmap and said printer's physical pixel array characteristics,

comparing the magnitude of said rotational difference with zero and with a predetermined maximum acceptable rotational difference value, and

(i) if the magnitude of the rotational difference is greater than zero and does not exceed said maximum acceptable rotational difference value, transforming the code bitmap of the machine-readable code image in the printer output image bitmap to essentially eliminate said rotational difference to align essentially exactly rectilinear areal geometric characteristics of the code image with said printer's physical pixel array characteristics, and

printing on the selected printer the code image defined by the transformed code bitmap in the printer output image bitmap, or

(ii) if the magnitude of said rotational difference exceeds said maximum acceptable rotational difference value, printing an error-indicating communication

7 A method according to claim 6 in which the error-indicating communication is an error-indicating graphics image

8 A printer processor implemented method for printing machine-readable code images on a selected printer having reproduction characteristics known and available, each machine-readable code image being defined by a code bitmap and having rectilinear areal geometric characteristics, the method including the steps of

introducing a code bitmap of a machine-readable code image to be printed into a printer output image bitmap by means of software which imposes a risk of inducing image bitmap distortion,

obtaining for the printer processor the known reproduction characteristics of said selected printer,

analyzing, by said printer processor, physical pixel array characteristics of said printer based on said known reproduction characteristics,

measuring shear distortion between the pixel array defined by the code bitmap of the machine-readable code image in the printer output image bitmap and said printer's physical pixel array characteristics to determine a shear error amount,

comparing the magnitude of said shear error amount with zero and with a predetermined maximum acceptable shear error amount value, and,

(i) if the magnitude of the shear error amount is greater than zero and does not exceed said maximum acceptable shear error amount value, transforming the code bitmap of the machine-readable code image in the printer output image bitmap to essentially eliminate said shear distortion to align essentially exactly rectilinear areal geometric characteristics of the code image with said printer's pixel array characteristics, and

printing on the selected printer the code image defined by the transformed code bitmap in the printer output image bitmap, or

(ii) if the magnitude of said shear error amount exceeds said maximum acceptable shear error amount value, printing an error-indicating communication

9 A method according to claim 8 in which the error-indicating communication is an error-indicating graphics image

10. A method as claimed in claim 1 substantially as herein described

11 A printed machine readable code image produced by the method of any one of claims 1 and 2 and 10

12 A method as claimed in claim 3 substantially as herein described

13 A printed machine readable code image produced by the method of any one of claims 3 to 5 and 12

14 A method as claimed in claim 6 substantially as herein described

15 A printed machine readable code image produced by the method of any one of claims 6 and 7 and 14

16 A method as claimed in claim 8 substantially as herein described

17 A printed machine readable code image produced by the method of any one of claims 8 and 9 and 16

18 A post script code substantially as herein described with reference to the example at pages 8 to 16 of the specification

**END OF CLAIMS**

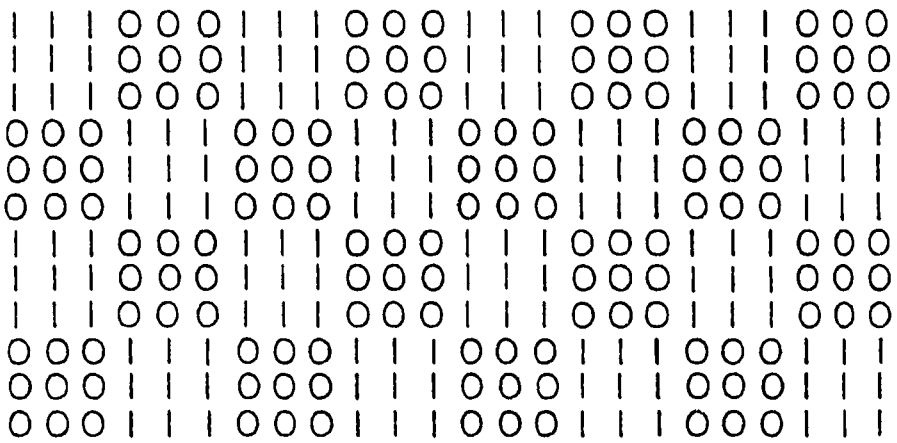

FIG. 1A

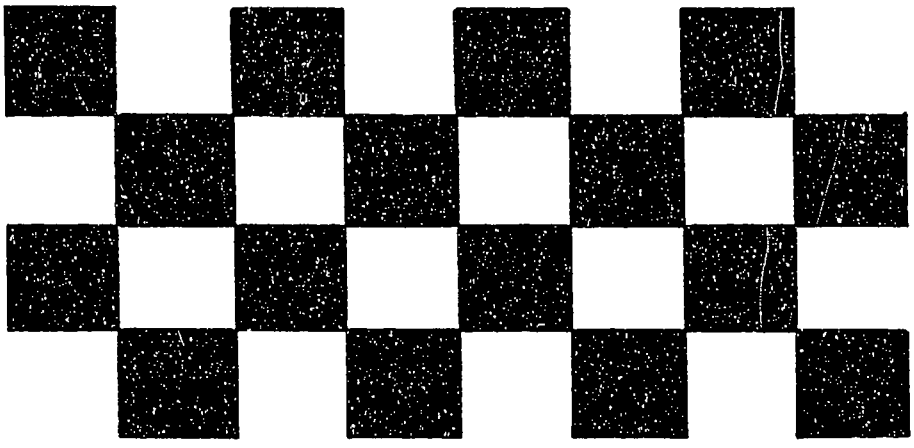

FIG. 1B

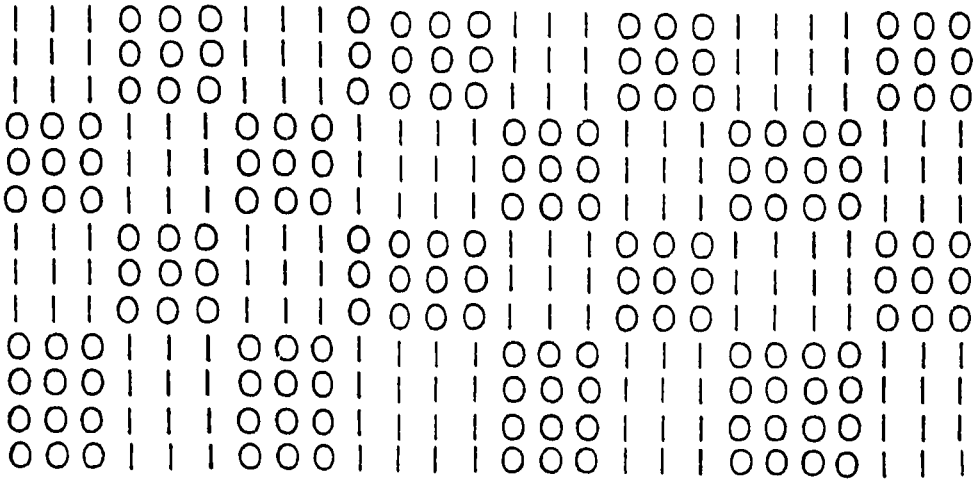

FIG.1C

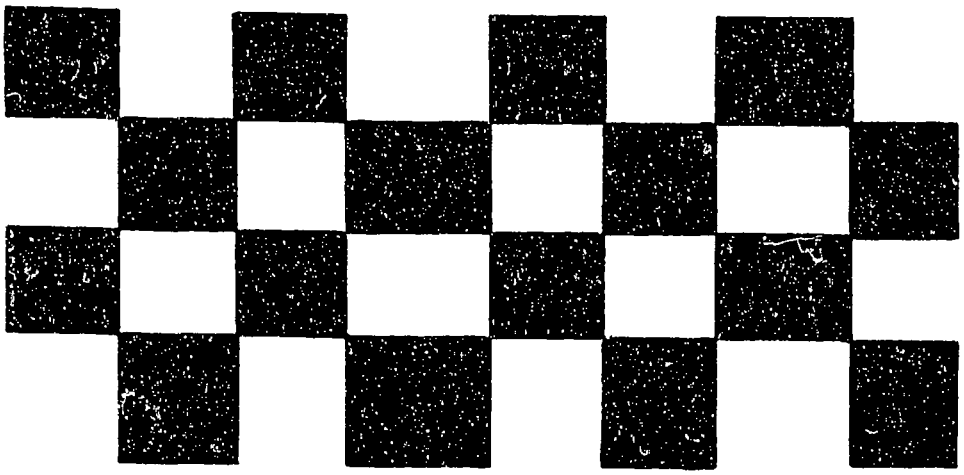

FIG. 1D

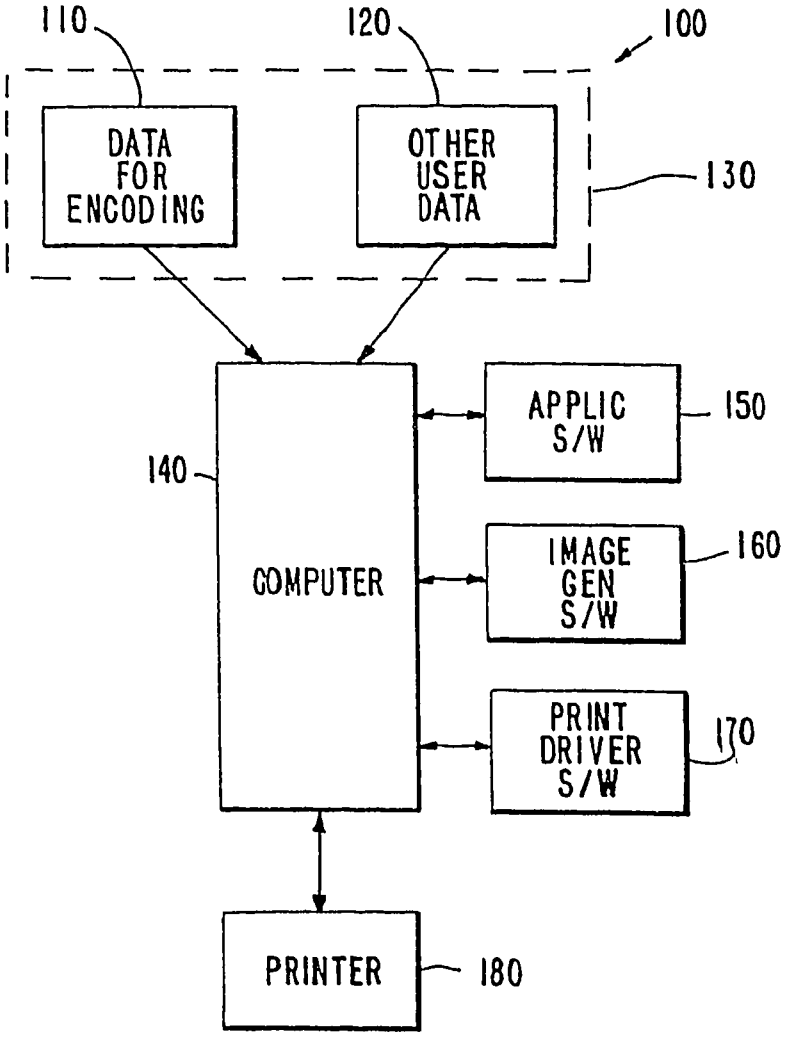
110
120
100
DATA FOR ENCODING
OTHER USER DATA
130
APPLIC S/W
150
140
COMPUTER
IMAGE GEN S/W
160
PRINT DRIVER S/W
170
PRINTER
180

FIG. 2

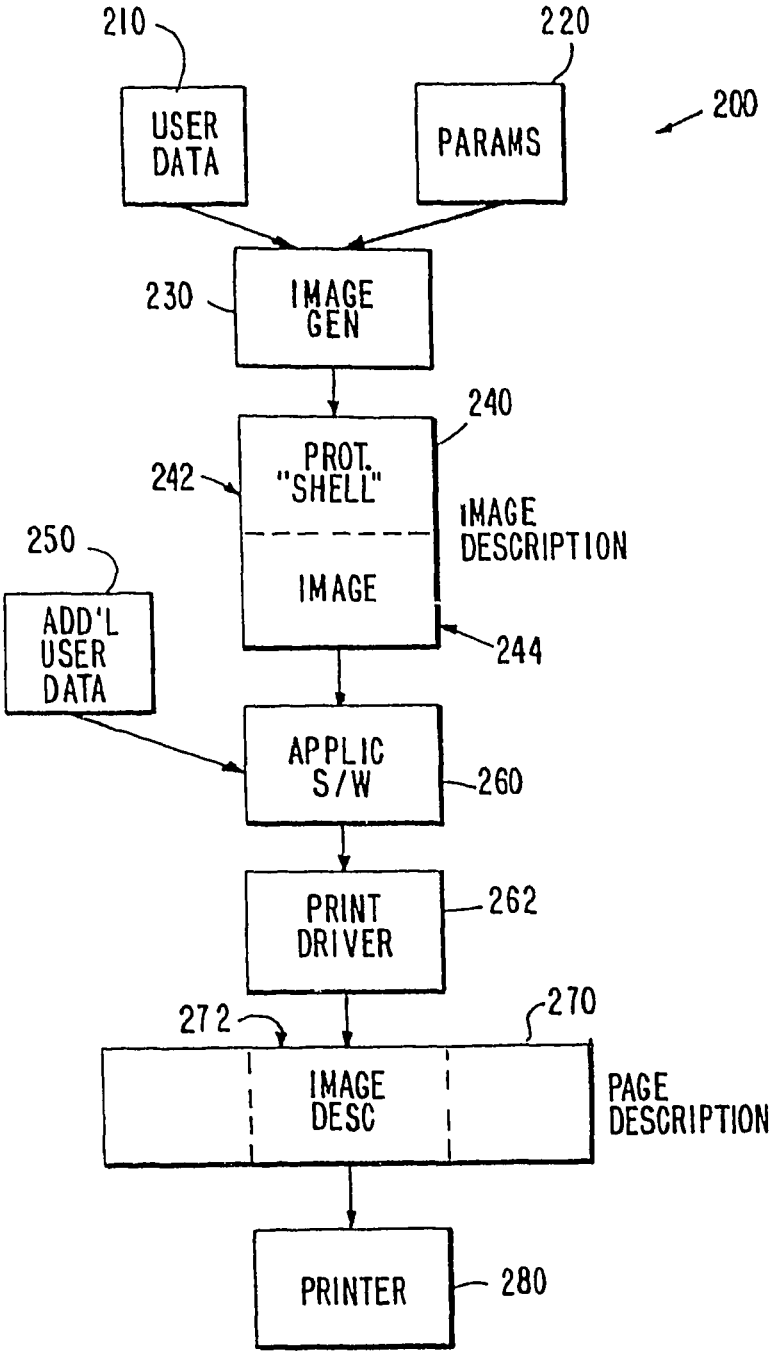
210
220
USER DATA
PARAMS
200
230
IMAGE GEN
240
PROT. "SHELL"
242
IMAGE DESCRIPTION
IMAGE
250
ADD'L USER DATA
244
APPLIC S/W
260
PRINT DRIVER
262
272
270
IMAGE DESC
PAGE DESCRIPTION
PRINTER
280

FIG.3

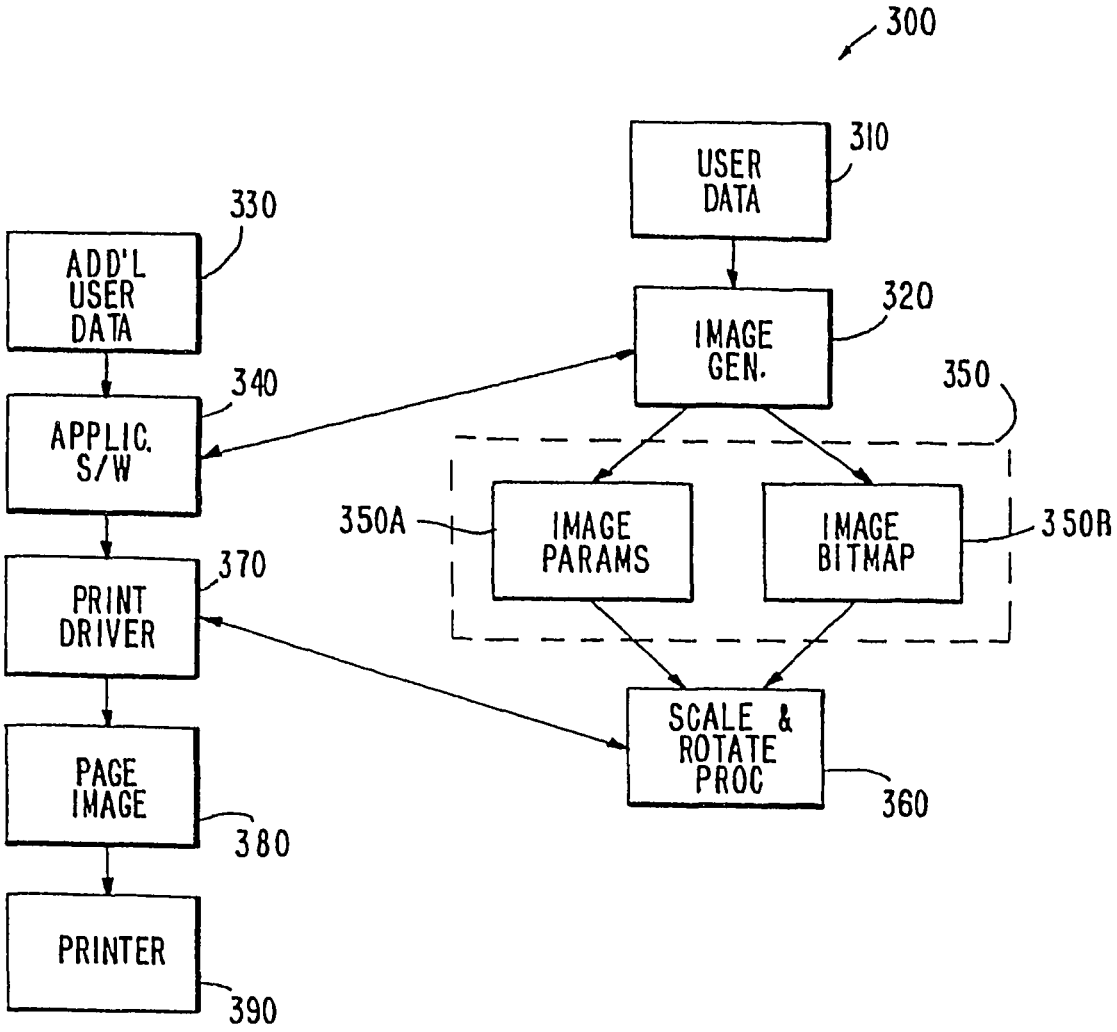
300
USER DATA
310
ADD'L USER DATA
330
IMAGE GEN.
320
APPLIC. S/W
340
350
350A
IMAGE PARAMS
IMAGE BITMAP
350B
PRINT DRIVER
370
SCALE & ROTATE PROC
360
PAGE IMAGE
380
PRINTER
390

FIG. 4

END