

## How to download bit images

One or more bit images can be downloaded into RAM and nonvolatile memory (NV). Bit images in NV are retained when the printer is powered off, but those in RAM are cleared when the printer is turned off or initialized by using command **1B 40**. Therefore, the users need to download the bit images into RAM again whenever the printer is restarted.

**In the programming manual, please refer to command 1D 2A (download bit images into RAM) and 1C 71 (download bit images into NV) for detailed information on how to perform the download.** In both RAM bit image download and NV bit image download, a number must be specified for the image to be downloaded. Please read carefully the commands **1D 2A** and **1C 71** for the difference between RAM bit image download and NV bit image download.

### Bit image data processing by taking an example of BMP format bit images:

1. The data is arranged in column data type in the bit image download command. Refer to command **1D 2A** and **1C 71** for details.
2. BMP format bit image data is arranged in line data type. Normally, a monochromatic BMP image is made of sixty-two (62) bytes of BMP image attributes description (including bit image width and height etc.) and normal image data. The BMP image data is arranged in integral multiple of 4 bytes. For example, if the width is 34 dots, 8 bytes instead of 5 bytes are needed to store the data. Therefore, the total amount of bytes that a bit image occupies in the memory is “(Line width in bits+31)/32\*4 \* line height in bits”

### Below is an example written under VC++ environment (for reference only):

Note: A non-monochromatic BMP image must be transformed into a monochromatic image, otherwise the data processed shall not be correct. Because the printer needs vertical data in downloading bit image, both the height (in pixels) and width (in pixels) of BMP image shall be multiple of eight (8), otherwise the printer may not be able to handle the data correctly.

```
//*****//
//Function: AntiRotateBmp90D                                     //
// Utility: Transform bmp format bit image into printer processable data //
// Parameter: pBmpData--- Pointer to source data                  //
//      nPixelsOfWidth---bit image width (dots)                  //
//      nPixelsOfHeight---bit image height (dots)                //
//      pBmpDataRotated---Pointer to target data                 //
//Back value: 1:data transformation success 0: data error        //
//*****//
int AntiRotateBmp90D(
    char *pBmpData,
    const int nPixelsOfWidth,
    const int nPixelsOfHeight,
    char *pBmpDataRotated
```

```
)
{
    // Define process variables
    int nBytesOfWidth = 0, nBytesOfHeight = 0;
    int i=0, col=0, row=0, index = 0, colbyte = 0;
    char* midData;
    unsigned char tempdata = 0, colnum = 0, rownum = 0;
    unsigned char temp[8] = {0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01};

    // Comparing parameters
    if (pBmpData == NULL || pBmpDataRotated == NULL) return 0;
    if (nPixelsOfWidth <= 0 || nPixelsOfHeight <= 0) return 0;
    if ((nPixelsOfWidth % 8) != 0 || (nPixelsOfHeight % 8) != 0) return 0;

    // get real image dimension
    nBytesOfWidth = (nPixelsOfWidth+31)/32*4;
    nBytesOfHeight = nPixelsOfHeight / 8;

    // White/black reverse, the value 1 in BMP data is for white which is contrary to the
    printer definition.
    midData = (char*)malloc(nBytesOfWidth*nPixelsOfHeight+1);
    for(i=0;i<nBytesOfWidth*nPixelsOfHeight;i++){midData[i] = 0xff-pBmpData[i];}

    // Rotation. BMP data is arranged in line data type while data downloaded to printer
    are arranged in column data type.
    for (row = 0; row <nPixelsOfWidth; row++){
        for (colbyte = 0; colbyte < nBytesOfHeight; colbyte++){
            index = row * nBytesOfHeight + colbyte;
            pBmpDataRotated[index] = 0x00;
            for (col = 0; col < 8; col++){
                colnum = col % 8;
                rownum = row % 8;
                if(colnum >= rownum)
                    tempdata = temp[col] & (midData[(nPixelsOfHeight-1-colbyte*8-col)
* nBytesOfWidth + row / 8] >> (colnum-rownum));
                else
                    tempdata = temp[col] & (midData[(nPixelsOfHeight-1-colbyte*8-col)
* nBytesOfWidth + row / 8] << (rownum-colnum));
                pBmpDataRotated[index] |= tempdata;
            }
        }
    }

    // release interim buffers
    free(midData);
}
```

Download a bit image shown as below:

[illegible]

000001E000FC0000000001C00FFC000000000380FFFC00000000070FFFFC00000000  
07FFFFFC0000000007FFFFF0000000000FFFFF00000000000FFFFF000000000000FFF  
00000000000000FFC0000000000000F38000000000000007000000000000000E0000000  
00000000C000FC0000000001800FFE0000000003807FFE000000000303FFFE0000000  
0071FFFFE0000000007FFFFE000000000FFFFF9C000000000FFFF818000000000FF  
FC03807C000000FFE00700FE000000FF000E01FF00000070000C03FF80000000000C  
07FF8000000000000078780000000003F0F01C000000003FF8E01C00000000FFFD00  
C00000003FFFD800C00000007FFFF800C0000001FFFFF000C0000003FE07E000C00  
00007F001E001C000000F8001C001C000001E000380038000001C0003800780000038  
0007000F80000070000E003F00000070001E01FF00000060007FFFFE000000E003FFF  
FFC000000E03FFFFFFF8000000E3FFFFFFF0000000FFFFFFF0000000FFFFFFF0000000  
0000000FFFFE00000000001FFFC0000000000001FFC00000000000003FC00000000000  
03C000  
00  
00  
00  
00  
00

**1C 70 01 00** (Select the number of the bit image to be printed and Print the bit image in  
FLASH)