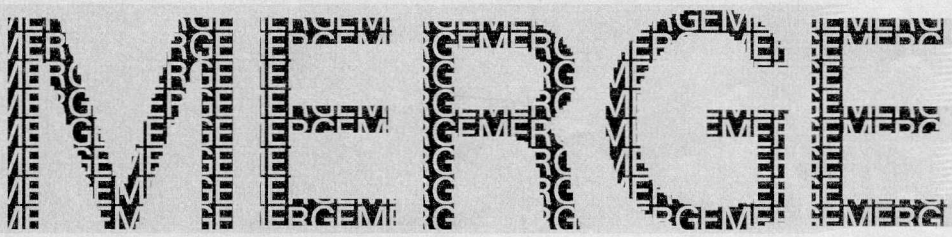
IBM Zurich Research Laboratory



A program to combine bitmaps
Device independent second version
Part of the Extended Digital Image Processing Facility

H. Thomas
Advanced Application Studies
Communications and Computer Science

February 1981

Introduction

The program MERGE combines text, line art, and images.

Modern text processing with variable fonts usually produces bitmaps for all-points-addressable printers instead of character strings for a fixed-character typewriter-like printer. On an all-points-addressable high-resolution printer other things besides text can be printed : images, line art, graphics.

After the text processing or the image processing - as the case may be - and before printing text, line art, and images are all prepared for the printer in the form of bitmaps of varying sizes. Sometimes special device-dependent or compressed forms of the bitmap are produced by the text processing system, but as they all must tell the printer in the end where to put black picture elements and where white, they can all be easily converted into the device independent format of a bitmap, which is more closely described in Ch. D'Heureuse's description of DIPF (Digital Image Processing Facility). Briefly a bitmap is a representation of an image to be printed on a black and white digital printer, where one line of picture elements is contained in one record and where the record is interpreted as a bit string with '1' meaning 'print a dot' and '0' for 'don't print'.

On this bitmap level, that is shared by text, line art, and images one can define all possible combinations, cut-and-paste jobs etc. in order to produce documents on the printer that consist of all these elements. The program MERGE is designed to do this cut-and-paste on bitmaps.

In summer 1980 the first version of MERGE was developed at the IBM Los Angeles Scientific Center. It had most of the features of the present second version and some more in addition. Its main drawback was, that it worked on a device dependent bitmap format and that it depended on the CMS environment. The new MERGE is device independent working on bitmaps as input and output files and it can be called from MVS. Some simplifications have been introduced. The FILLIN facility was taken out of MERGE as it is considered to be an independent image processing function that can be performed on images independent of the MERGE. It will be integrated in the DIPF package soon. The control file processing has been taken out of the program which now performs only one merge step per call. As a result of these simplifications MERGE uses less CPU time for a merge and it is simpler to use.

Still, as can be seen in the Chapter 'Definition of a merge' the input to this program is not very comfortable. It seems obvious, that this drawback can only be overcome in an interactive system, where the user can point at places on the page or on his images. Such an interactive input program could then call MERGE with the correct numerical values as input parameters. An implementation of such an interactive definition of a merge operation using the Graphics Attachment is suggested for making efficient use of MERGE

I wish to thank Ch. D'Heureuse and Dr. P.Stucki for their hints and discussions about how to build a stable device independent image processing environment.

Definition of a merge

In order to use the program MERGE it is necessary to consider, what a merge is and which parameters must be specified to define a merge.

Conceptually a merge consists of putting a rectangular input bitmap on top of another rectangular input bitmap and perform any boolean operation to produce the output bitmap. As it is desirable, that the user can "point" at the place where he wants to have the rectangles put and as this pointing is never very precise, the concept of a merge has been refined : the user indicates a target rectangle for each of the input rectangles and a nearest neighbour scale change is performed first for each input rectangle as a preparation step before the merge is performed.

A merge operation is defined by the

1) specification of the input and output files

2) specification of the input and target rectangles

3) specification of the boolean function for the merge

4) specification of the relative placement of the rectangles

1) specification of the input and output files

The only information, the program needs about the input and output files are their DDNAMES. It assumes that an outside procedure (a TSO-CLIST, a JCL-Procedure, or a CMS-EXEC) has already allocated the necessary files. The program uses the same convention for DDNAMES, as the Digital Image Processing Facility (DIPF), taking the first letter as an indication of the image file type ('B' for bitmap, 'Y' for bytemap, and 'W' for wordmap) and gives an error message, if one of the three files involved is not of the bitmap type.

The program MERGE does not need any other information about the input files, as it looks up additional information (organisation and logical record length) itself, using the subroutine FILEATT from the DIPF package.

As will be seen later on the concept of a merge has been further refined in order to include certain more artistic designing functions : pseudo input files and modified input files. This can be indicated in the file name too by starting it with certain nonalphabetical characters that tell the program how to interpret those rectangles. The user who wants to use these facilities is referred to the chapter "Extensions of the merge operation".

2) specification of the input and target rectangles

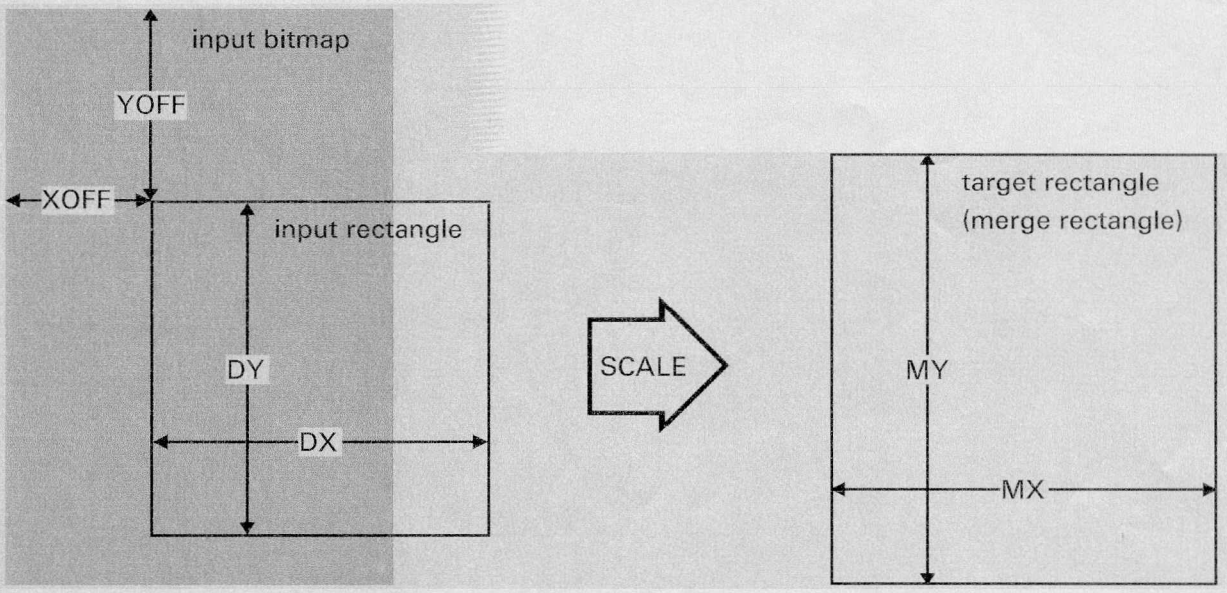
In order to specify the rectangles of the input bitmaps that are to be used in the merge one has to indicate their offsets and their dimensions in both the x-direction (horizontal from left to right) and the y-direction (vertical from top to bottom). In order to specify the target rectangles (also called merge rectangles in the sequel) only their sizes (dimensions) have to be given.

All dimensions given are in number of picture elements rather than in millimeters as the absolute dimensions of a bitmap are only defined relatively to an output device and the merge is defined independent of the output device. It is recommended to have a supervising program for special applications which allows the user to enter millimeters or inch and converts them to pel measurement before calling MERGE.

The abbreviations used for the dimensions of the input rectangles from here on are :

- XOFF1 : offset in the x-direction of the first input rectangle
- YOFF1 : offset in the y-direction of the first input rectangle
- DX1 : dimension in the x-direction of the first input rectangle
- DY1 : dimension in the y-direction of the first input rectangle
- MX1 : dimension in the x-direction of the first merge rectangle
- MY1 : dimension in the y-direction of the first merge rectangle
- XOFF2 : offset in the x-direction of the second input rectangle
- YOFF2 : offset in the y-direction of the second input rectangle
- DX2 : dimension in the x-direction of the second input rectangle
- DY2 : dimension in the y-direction of the second input rectangle
- MX2 : dimension in the x-direction of the second merge rectangle
- MY2 : dimension in the y-direction of the second merge rectangle

It should be noted that - contrary to the usual handling of windows, in DIPF for example - the window is not clipped, when it extends over the edges of the input bitmap but is filled with blanks. This implies that the offsets are allowed to be negative, whereas the dimensions have to be positive of course. This is a very powerful feature of MERGE. It allows to paste two images together without overlay by just defining enough empty space on one side of the first input rectangle and putting the second one there.



3) specification of the boolean function for the merge

As it was discovered, that almost all boolean functions are useful in certain situation in image processing all sixteen binary boolean functions are offered as merging options. The boolean function to be used is specified as in the PL/I-function BOOL as a character string of ones and zeros of length four. The meaning of this OPTION string can be taken from the following table :

first input bit	second input bit	output bit
0	0	first character of OPTION
0	1	second character of OPTION
1	0	third character of OPTION
1	1	fourth character of OPTION

Thus '0111' means 'OR', '0001' means 'AND', '0101' means always print what is on the second image suppressing the information of the first etc.

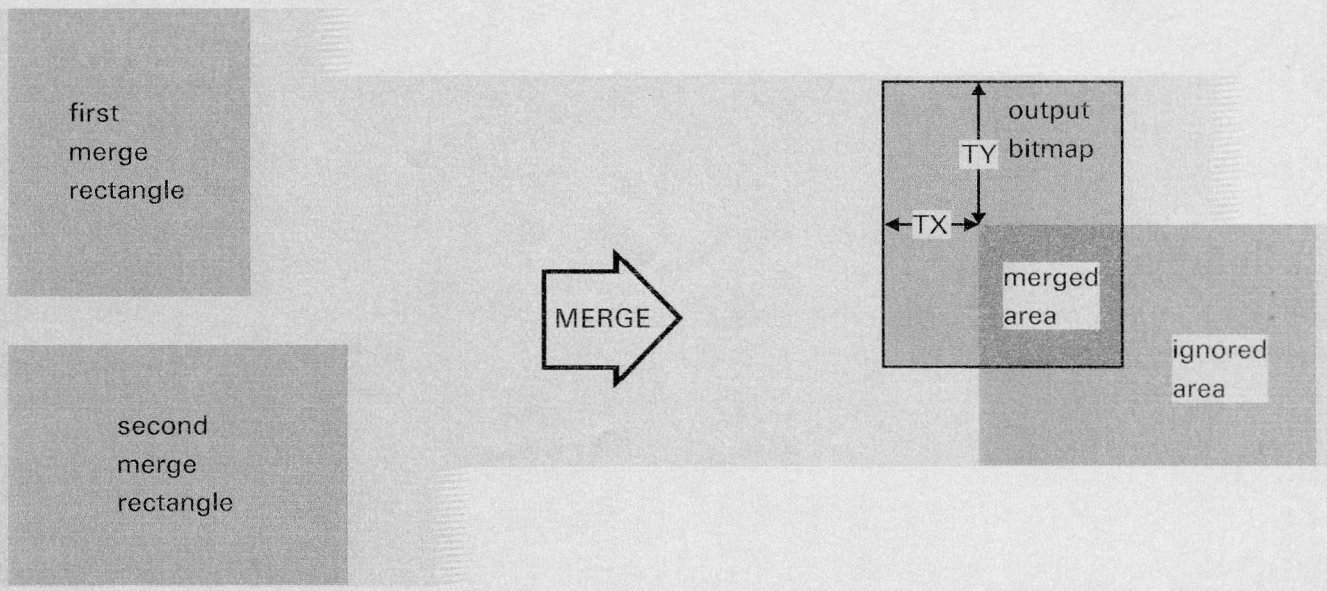
It should be noted, that the range of this boolean function is only the overlapping part of the two merge rectangles. Outside that, the first merge rectangle is produced without any changes.

4) specification of the relative placement of the rectangles

In order to define the relative placement of the merge rectangles for the merge it is sufficient to indicate the offsets with which the second merge rectangle is to be placed on top of the first one. Note that the size of the first merge rectangle is the size of the output rectangle and that parts of the second merge rectangle extending over the edges of the first one are ignored.

The abbreviations used for the placement offsets of the merge rectangles from here on are :

- TX : translation offset in the x-direction
- TY : translation offset in the y-direction



Extensions of the merge

Some practical extensions to the elementary definition of a merge as given in the last chapter were added to the original concept, because they were found to be often needed and save diskspace, because the user invariably produces his own "all black image" and "all white image" to be used together with merge, when he doesn't have access to the special extensions of the definition of the merge described below.

The first extension is the introduction of so called pseudo files for special datasets. These are recognized by the program, because the DDNAME does not begin with an alphabetic character. The pseudo files recognised are :

- The all black file indicated by a "1" as the first character of the DDNAME
- The all white file indicated by a "0" as the first character of the DDNAME
- The grey file indicated by a "." as the first character of the DDNAME followed by numerals such, that the whole DDNAME indicates the fraction of grey to be printed.

These pseudo files are generated automatically by MERGE (using MECCA for halftoning). Obviously only MX and MY have a meaning for them describing the size to be produced. The other parameters are ignored, if given. A pseudo file can be given instead of the DDNAME of the first input file or instead of the DDNAME of the second input file. Understandably a pseudo file name can not replace the DDNAME of the output file.

The second extension is the introduction of the modification in the preparation of the merge rectangle. Instead of having the input rectangle scaled to fit the merge rectangle, one can also have it repeated in both dimensions as often as it will fit. This is indicated by prefixing the DDNAME with the repetition character '''. This solution was used, rather than integrating various textures into the program, as desired by users, as it seems impossible to define a small set of commonly used textures (like supercircle halftoning bitmaps) and thus the user can define his own textures in small bitmaps and thus also save considerable disk space, because only this nucleus is needed, to produce large bitmaps full of it. The parameters all have the same meaning as in the normal merge case. Only the change in size is attained by repetition instead of scale change. Also here it is clear that output DDNAMES cannot begin with '''.

The PL/I program MERGE

In this chapter the PL/I program MERGE is described, as it is installed at the IBM Research Laboratory Zurich at present. The description consists of the

1) general comments about the organiziation of the input/output

2) specification of the input parameter string as accepted by MERGE

3) specification of the default processing of MERGE

4) brief overview of the structure of the program MERGE

5) listing of the external subroutines used by MERGE

1) organization of the input/output

The input and output bitmaps are assumed to be disk datasets with fixed blocked record organization (RECFM=FB) and one image line per record. All these datasets must represent bitmaps, as decribed in the description of DIPF. The only additional input, the program MERGE uses, are the processing parameters descibed in the last two chapters. These have to be specified in the parameter string, when calling the program. All these input/output defaults were chosen so as to be compatible with the image programs in DIPF. The program MERGE can be installed in DIPF, should this be deemed desirable and in can be adapted for CMS in the same way, DIPF was adapted for CMS (see paragraph "external subroutines" and DIPF description for compatibility problems).

2) input parameter string

The input parameter string is a varying length character string of maximally 256 characters. It must contain tho following parameters in the order indicated :
XOFF1 YOFF1 DX1 DY1 MX1 MY1 XOFF2 YOFF2 DX2 DY2 MX2 MY2 OPTION TX TY
where all parameters except OPTION are numbers with their meaning described in the two previous chapters and OPTION is a string of four characters containing zeros and ones and specifying the merge function to be used as is also previously described. It should be noted, that '*' can be entered instead of any of the values DX1, DY1, MX1, MY1, DX2, DY2, MX2, MY2 in order to use default processing. Further defaults can be introduced in CLISTS (see next chapter).

3) default processing

If the dimensions of an input rectangle are given as '*' the program checks how many columns and/or lines can be found in the input dataset and sets DX = INCOLUMNS - XOFF and DY = INLINES - YOFF. If pseudofiles are used MX and MY must be given in any case. Otherwise, when MX and/or MY are specified as '*' they are set equal to DX and DY. In any case the program warns the user when he has entered invalid parameters and it displays the parameters which it chose by default for processing.

4) program structure of MERGE

```
MERGE                                   main procedure

    INITIALIZE                          initializing procedure
                                        analyzes input parameters and file attributes

    PROCESS                             processing procedure

        PREPARATION OF THE FIRST MERGE RECTANGLE
                                        depending on processing parameters
                                        call one of the following

            BLACK                       procedure to produce an all black dataset

            WHITE                       procedure to produce an all white dataset

            TONE                        procedure to produce a toned dataset

            CUT                         procedure to cut a window from a bitmap

            TEXTUR                      procedure to produce a texture from a bitmap

            SCALE                       procedure to cut and scale a window from a bitmap

        PREPARATION OF THE SECOND MERGE RECTANGLE
                                        depending on processing parameters
                                        call one of the following

            BLACK                       procedure to produce an all black dataset

            WHITE                       procedure to produce an all white dataset

            TONE                        procedure to produce a toned dataset

            CUT                         procedure to cut a window from a bitmap

            TEXTUR                      procedure to produce a texture from a bitmap

            SCALE                       procedure to cut and scale a window from a bitmap

        PERFORM THE ACTUAL MERGE
                                        use OPTION, TX, TY for merging the
                                        merge rectangles

    TERMINATE                           procedure that writes a termination message
                                        and terminates execution
```

5) external subroutines used by MERGE

ENTRY NAME	TYPE	CALLED BY
FILEATT	PLI	internal subroutine INITIALIZE
GETPARM	PLI	internal subroutine INITIALIZE
MECCASM	ASM	internal subroutine TONE
DCBINFO	ASM	external subroutine FILEATT

Remarks :
All external subroutines used are part of the DIPF package.
The assembly subroutine DCBINFO is system dependent. For CMS adaption the subroutine FILESTA would have to be used and some minor modifications as for DIPF have to implemented.

Using MERGE under MVS

In this chapter the CLIST and JCL datasets used to execute MERGE under MVS are described and examples are given on how to use MERGE.

1) CLISTs and CNTL datasets for easy use of MERGE

The following CLIST is currently a member of the dataset PS.MASTER.CLIST and can be called by any user of MERGE :

```
PROC 3 INDSN1 INDSN2 OUTDSN XOFF1(0) YOFF1(0) DX1(*) DY1(*) MX1(*) +
MY1(*) XOFF2(0) YOFF2(0) DX2(*) DY2(*) MX2(*) MY2(*) OPTION(0111) +
TX(0) TY(0) USERID(&SYSUID) PROJ(4789) CLASS(X) MINUTES(1)
/*
CHOICE: WRITE
WRITENR ENTER "F" FOR FOREGROUND, "B" FOR BATCHPROCESSING :
READ ANSWER
IF &ANSWER ¬= F AND &ANSWER ¬= B THEN GOTO CHOICE
IF &ANSWER = F +
   THEN EXEC 'THO.C.CLIST(MERGEF)' '&INDSN1 &INDSN2 &OUTDSN +
      &XOFF1 &YOFF1 &DX1 &DY1 &MX1 &MY1 +
      &XOFF2 &YOFF2 &DX2 &DY2 &MX2 &MY2 +
      &OPTION &TX &TY'
   ELSE DO
      WRITE
      WRITE CURRENT BATCH PARAMETERS :
      WRITE USERID(&USERID) PROJ(&PROJ) CLASS(&CLASS) MINUTES(&MINUT
      WRITE
      WRITE PLEASE PRESS "ENTER" TO CONTINUE, "Q" FOR QUIT
      READ ANSWER
      IF &ANSWER ¬= Q +
         THEN EXEC 'THO.C.CLIST(MERGEB)' '&INDSN1 &INDSN2 &OUTDSN +
            &XOFF1 &YOFF1 &DX1 &DY1 &MX1 &MY1 +
            &XOFF2 &YOFF2 &DX2 &DY2 &MX2 &MY2 +
            &OPTION &TX &TY &USERID &PROJ &CLASS &MINUTES'
      END
```

As can be seen from the listing above, this CLIST introduces additional defaults, to save the user writing superfluous writing time. The defaults can all be seen in the first PROC statement and the parameters should be compared with the definition of the parameters in previous chapters in order to understand the effect of using defaults. As can also be seen in the listing, the above CLIST offers the choice for foreground and background processing and then executes THO.C.CLIST(MERGEF) or THO.C.CLIST(MERGEB) correspondingly.

The following THO.C.CLIST(MERGEB) is executed, if batch processing was chosen :

```
PROC 22 INDSN1 INDSN2 OUTDSN XOFF1 YOFF1 DX1 DY1 MX1 MY1 +
                             XOFF2 YOFF2 DX2 DY2 MX2 MY2 +
                             OPTION TX TY +
                             USERID PROJ CLASS MINUTES
E 'THO.C.CNTL(MERGE)' CNTL
C 1 999999 /$USERID$/&USERID/ ALL
C 1 999999 /$PROJ$/&PROJ/ ALL
C 1 999999 /$CLASS$/&CLASS/ ALL
C 1 999999 /$MINUTES$/&MINUTES/ ALL
C 1 999999 /$INDSN1$/&INDSN1/ ALL
C 1 999999 /$INDSN2$/&INDSN2/ ALL
C 1 999999 /$OUTDSN$/&OUTDSN/ ALL
C 1 999999 /$XOFF1$/&XOFF1/ ALL
C 1 999999 /$YOFF1$/&YOFF1/ ALL
C 1 999999 /$DX1$/&DX1/ ALL
C 1 999999 /$DY1$/&DY1/ ALL
C 1 999999 /$MX1$/&MX1/ ALL
C 1 999999 /$MY1$/&MY1/ ALL
C 1 999999 /$XOFF2$/&XOFF2/ ALL
C 1 999999 /$YOFF2$/&YOFF2/ ALL
C 1 999999 /$DX2$/&DX2/ ALL
C 1 999999 /$DY2$/&DY2/ ALL
C 1 999999 /$MX2$/&MX2/ ALL
C 1 999999 /$MY2$/&MY2/ ALL
C 1 999999 /$OPTION$/&OPTION/ ALL
C 1 999999 /$TX$/&TX/ ALL
C 1 999999 /$TY$/&TY/ ALL
SUB
END N
```

Obviously this CLIST only edits THO.C.CNTL(MERGE) entering the processing parameters and submits it.

The following JCL (THO.C.CNTL(MERGE)) executes THO.C.CLIST(MERGEF) in the batch :

```
//$USERID$MRG JOB $PROJ$,$USERID$,MSGCLASS=$CLASS$,
//             NOTIFY=$USERID$,TIME=$MINUTES$
// EXEC PGM=IKJEFT01,DYNAMNBR=10
//SYSPROC   DD DSN=THO.C.CLIST,DISP=(SHR,KEEP)
//SYSPRINT  DD SYSOUT=$CLASS$
//SYSTSPRT  DD SYSOUT=$CLASS$
//SYSTSIN   DD *
   %MERGEF $INDSN1$ $INDSN2$ $OUTDSN$ +
           $XOFF1$ $YOFF1$ $DX1$ $DY1$ $MX1$ $MY1$ +
           $XOFF2$ $YOFF2$ $DX2$ $DY2$ $MX2$ $MY2$ +
           $OPTION$ $TX$ $TY$
/*
```

Thus whether batch or foreground processing was chosen, in the end THO.C.CLIST(MERGEF) is executed.

In the following THO.C.CLIST(MERGEF) the type processing of the datasets should be noted. It imitates the default processing of the Digital Image Processing Facility (DIPF) :

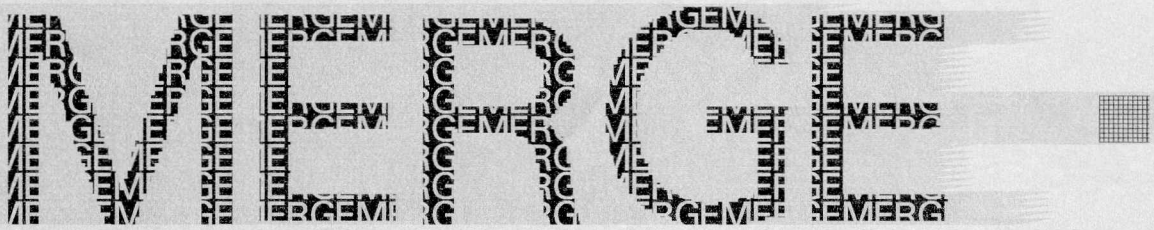
```
PROC 18 INDSN1 INDSN2 OUTDSN XOFF1 YOFF1 DX1 DY1 MX1 MY1 +
                             XOFF2 YOFF2 DX2 DY2 MX2 MY2 +
                             OPTION TX TY
/*
CONTROL NOFLUSH NOMSG
FREE F(INP1,INP2,OUTPUT,WORK1,WORK2)
CONTROL FLUSH MSG
/*
IF &SUBSTR(1:1,&INDSN1) ¬= &STR(0) AND +
   &SUBSTR(1:1,&INDSN1) ¬= &STR(1) AND +
   &SUBSTR(1:1,&INDSN1) ¬= &STR(.) THEN DO
   IF &STR(&SUBSTR(1:1,&INDSN1)) = &STR(") THEN DO
      SET INDSN1 = &SUBSTR(2:&LENGTH(&INDSN1),&INDSN1)
      SET INFILE1 = &STR(B"INP1)
      END
   ELSE SET INFILE1 = &STR(BINP1)
   IF &SUBSTR(1:1,&INDSN1) ¬= &STR(') THEN DO
      SET L = &LENGTH(&INDSN1)
      IF &L < 5 +
         THEN SET INDSN1 = &INDSN1..BIT
         ELSE IF &SUBSTR(&L-3:&L,&INDSN1) ¬= .BIT THEN +
            SET INDSN1 = &INDSN1..BIT
      END
   ALLOC F(INP1) DA(&INDSN1) SHR
   END
ELSE SET INFILE1 = &STR(*)&INDSN1
/*
IF &SUBSTR(1:1,&INDSN2) ¬= &STR(0) AND  +
   &SUBSTR(1:1,&INDSN2) ¬= &STR(1) AND  +
   &SUBSTR(1:1,&INDSN2) ¬= &STR(.) THEN DO
   IF &SUBSTR(1:1,&INDSN2) = &STR(") THEN DO
      SET INDSN2 = &SUBSTR(2:&LENGTH(&INDSN2),&INDSN2)
      SET INFILE2 = &STR(B"INP2)
      END
   ELSE SET INFILE2 = &STR(BINP2)
   IF &SUBSTR(1:1,&INDSN2) ¬= &STR(') THEN DO
      SET &L = &LENGTH(&INDSN2)
      IF &L < 5 +
         THEN SET &INDSN2 = &INDSN2..BIT
         ELSE IF &SUBSTR(&L-3:&L,&INDSN2) ¬= .BIT THEN +
            SET &INDSN2 = &INDSN2..BIT
      END
   ALLOC F(INP2) DA(&INDSN2) SHR
   END
ELSE SET INFILE2 = &STR(*)&INDSN2
/*
```

```
IF &STR(&SUBSTR(1:1,&OUTDSN)) ¬= &STR(') THEN DO
   SET &L = &LENGTH(&OUTDSN)
   IF &L < 5 +
      THEN SET &OUTDSN = &OUTDSN..BIT
      ELSE IF &SUBSTR(&L-3:&L,&OUTDSN) ¬= .BIT THEN +
         SET &OUTDSN = &OUTDSN..BIT
   END
CONTROL NOFLUSH NOMSG
ERROR DO
   ERROR OFF
   CONTROL FLUSH MSG
   ALLOC DA(&OUTDSN) SP(1,5) CYLINDERS NEW RELEASE
   END
ALLOC F(OUTPUT) DA(&OUTDSN) SHR
ERROR OFF
CONTROL FLUSH MSG
/*
ALLOC F(WORK1) NEW SP(1,5) CYLINDERS RELEASE
ALLOC F(WORK2) NEW SP(1,5) CYLINDERS RELEASE
/*
SET PARMS = &STR(&INFILE1),&STR(&INFILE2),BOUTPUT,WORK1,WORK2
SET PARMS = &STR(&PARMS),&STR(&XOFF1),&STR(&YOFF1)
SET PARMS = &STR(&PARMS),&STR(&DX1),&STR(&DY1),&STR(&MX1),&STR(&MY1)
SET PARMS = &STR(&PARMS),&STR(&XOFF2),&STR(&YOFF2)
SET PARMS = &STR(&PARMS),&STR(&DX2),&STR(&DY2),&STR(&MX2),&STR(&MY2)
SET PARMS = &STR(&PARMS),&STR(&OPTION),&STR(&TX),&STR(&TY)
/*
CALL 'THO.C.LOAD(MERGE)' '/&PARMS'
/*
FREE F(INP1,INP2,OUTPUT,WORK1,WORK2)
END
```

2) Some examples

Using the above CLISTS and the following two input bitmaps

MRGLOGO.BIT (2000*480) GRID.BIT (100*100)

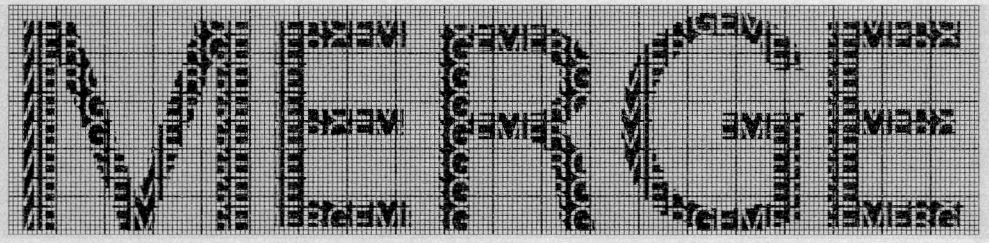


the command

%MERGE MRGLOGO "GRID MRGLOG1 DX2(100) DY2(100) MX2(2000) MY2(480)

results in

MRGLOG1.BIT(2000*480)



It should be noted, that the percent sign (%) must necessarily precede the command MERGE, if the users file SYSPROC is allocated to the dataset PS.MASTER.CLIST, as the command will otherwise be confused with the TSO command MERGE.

All the images in this text were merged onto the pages using PXDECODE and PXENCODE of the Extended Digital Image Processing Facility to transform sweep files into bitmap format and back. MERGE itself naturally treats all bitmaps equally (text and images).

As a last example for MERGE showing the scaling facility and an unusual boolean function as option consider the following command :
%MERGE TEMP1 TEMP2 TEMP3 DX1(190) DY1(40) MX1(1300) MY1(800) DX2(2030) DY2(2030) MX2(1300) MY2(800) OPTION(0110)

MERGE