NOTE: ALL PROGRAM MODIFICATIONS SHOULD BE MADE TO A COPY OF MERLIN 64. DO NOT MAKE CHANGES TO THE MASTER DISKETTE.

CHANGING DISPLAY COLORS

In order to change the background, border or character colors in Merlin 64, enter the folowing:

```
LOAD "MERLIN",8
```

At the prompt, enter:

```
LIST 50
```

It should appear as follows:

```
50 DATA0,0,0,10,0,128,0,10,94,4,47,15,15,144
```

The last three data elements are referenced on page 74 of the Merlin 64 manual. These data elements (15,15 and 144) are the default values for border, background and character colors used by Merlin 64.

The following chart shows the values that can be substituted for the BORDER and BACKGROUND colors:

0 BLACK	4 PURPLE	8 ORANGE	12 MED. GREY
1 WHITE	5 GREEN	9 BROWN	13 LT. GREEN
2 RED	6 BLUE	10 LT. RED	14 LT. BLUE
3 CYAN	7 YELLOW	11 DARK GREY	15 LT. GREY

The following chart shows the values that can be substituted for the CHARAÇTER color:

144 BLACK	156 PURPLE	129 ORANGE	151 GREY 1
5 WHITE	30 GREEN	149 BROWN	153 LT. GREEN
28 RED	31 BLUE	150 LT. RED	154 LT. BLUE
159 CYAN	158 YELLOW	152 GREY 2	

Thus, if you wanted to have a black background and border with white characters, you would change Line 50 as follows:

```
50 DATA0,0,0,10,0,128,0,10,94,4,47,0,0,5
```

Color TV sets are limited in their ability to place certain colors next to each other on the same line. Therefore, some color combinations may produce blurred images. When you find the best combination for your system, save the BASIC program called MERLIN and Merlin 64 will then always boot with your color combination.

If you wish to have white characters on a black background when you are using the CHRGEN 80 utility, just put a $F0 in memory locations $A04 and $A05.

LOWER CASE ONLY ON PRINTOUTS

This is the result of the printer interface card (such as CARDCO) doing an ASCII conversion before printout. This may not surface when running a program in BASIC, but appears if the firmware on the card is incorrect. To solve this, enter:

```
LOAD "MERLIN",8
```

and add the following line to the program:

```
65 POKE 41191,97
```

Then list Line 60 which should appear as follows:

```
60 DATA8,16,0,0,128,60,7,80,128,0,0,97,14,20,31
```

Change the eleventh data element from 0 to 160. The new Line 60 should then appear as follows:

```
60 DATA8,16,0,0,128,60,7,80,128,0,160,97,14,20,31
```

After testing these changes to verify that they work with your interface card, save the BASIC program called MERLIN.

Note that these modifications only work with interface cards that do an ASCII conversion. Do not use these with straight throughput interfaces such as RS232 etc.

For more information, see the TECHNICAL INFORMATION section, pages 73 to 75.

USING PUT FILES WITH MERLIN 64

If your source file becomes too large to assemble in memory, you can use the PUT opcode. Just divide your program into sections and save each section as a separate text file by using the W (WRITE TEXT FILE) command. The PUT opcode loads these text files and "inserts" them in the "Master" source file at the location of the PUT opcode.

The "Master" source file can contain source code, equates, macros, and ALL of your PUT opcodes. You cannot define a macro from within a PUT file, and you cannot call the next PUT file from within the current PUT file. Thus, all macro definitions and all PUT opcodes must be in the "Master" source file.

The following three programs (2 PUT files and the Master) will demonstrate the proper method for using PUT files. First, write a text file called FILE1 by using the W command. This will be used as the first PUT file in the Master file.

```
1 FILE1    LDX  #0
2 LOOP1    LDA  STRING1,X
3          BEQ  FILE2
4          JSR  CHROUT
5          INX
6          BNE  LOOP1
7 STRING1  ASC  "THIS IS FILE1",00
```

Now write another text file (FILE2) with the W command. This will be the second PUT file used by the Master file.

```
1 FILE2    LDX  #0
2 LOOP2    LDA  STRING2,X
3          BEQ  DONE
4          JSR  CHROUT
5          INX
6          BNE  LOOP2
7 DONE     RTS
8 STRING2  ASC  "NOW ITS FILE2",00
```

The Master source is assembled and saved using the S (SAVE SOURCE) command, followed by the O (SAVE OBJECT) command.

```
1 * MASTER SOURCE FILE *
2          ORG  $8000
3 CHROUT   EQU $FFD2
4          PUT "FILE1"
5          PUT "FILE2"
```

When the Master file is assembled, it will look like this:

```
                    1   * MASTER SOURCE FILE *
                    2           ORG  $8000
                    3   CHROUT  EQU  $FFD2
                    4           PUT  FILE1"
8000: A2 00        >1   FILE1   LDX  #0
8002: BD 0D 80     >2   LOOP1   LDA  STRING1,X
8005: F0 14        >3           BEQ  FILE2
8007: 20 D2 FF     >4           JSR  CHROUT
800A: E8           >5           INX
800B: D0 F5        >6           BNE  LOOP1
800D: D4 C8 C9     >7   STRING1 ASC  "THIS IS FILE1",00
8010: D3 A0 C9 D3 A0 C6 C9 CC
8018: C5 B1 00
                    5           PUT  "FILE2"
801B: A2 00        >1   FILE2   LDX  #0
801D: BD 28 80     >2   LOOP2   LDA  STRING2,X
8020: F0 06        >3           BEQ  DONE
8022: 20 D2 FF     >4           JSR  CHROUT
8025: E8           >5           INX
8026: D0 F5        >6           BNE  LOOP2
8028: 60           >7   DONE    RTS
8029: CE CF D7     >8   STRING2 ASC  "NOW ITS FILE2",00
802B: A0 C9 D4 D3 A0 C6 C9 CC
8033: C5 B2 00

--End assembly, 55 bytes, Errors: 0
```

USING THE DSK COMMAND WITH MERLIN 64

This causes Merlin 64 to assemble the file directly to disk. It is used at the very start of a source file before any code is generated. It will assemble all code to disk, using the specified filename, until it encounters another DSK or reaches the end of code. Since DSK saves the object code to

disk, you do not have to use the O (SAVE OBJECT) command.

```
                    1   * DSK COMMAND FOR MERLIN 64 *
                    2             DSK  "FILE ONE" ;ASSEM TO DISK
                    3             ORG  $8000
                    4   CHROUT    EQU  $FFD2
                    5   SCNKEY    EQU  $FF9F
                    6   GETIN     EQU  $FFE4
8000: A2 00         7             LDX  #0
8002: BD 1D 80      8   LOOP1     LDA  STRING1,X
8005: F0 06         9             BNE  SCAN1
8007: 20 D2 FF     10             JSR  CHROUT
800A: E8           11             INX
800B: D0 F5        12             BNE  LOOP1
800D: 20 9F FF     13   SCAN1     JSR  SCNKEY
8010: 20 E4 FF     14             JSR  GETIN
8013: F0 F8        15             BEQ  SCAN1
8015: C9 0D        16             CMP  #$0D        ;RETURN PRESSED?
8017: F0 03        17             BEQ  DONE1       ;IF Y WE'RE DONE
8019: 4C 0D 80     18             JMP  SCAN1       ;IF N GO BACK
801C: 60           19   DONE1     RTS
801D: D4 C8 C9     20   STRING1   ASC  "THIS IS FILE ONE",00
8020: D3 A0 C9 D3 A0 C6 C9 CC
8028: C5 A0 CF CE C5 00
                   21             DSK  "FILE TWO" ;ASSEM TO DISK
802E: A2 00        22             LDX  #0
8030: BD 4B 80     23   LOOP2     LDA  STRING2,X
8033: F0 06        24             BNE  SCAN2
8035: 20 D2 FF     25             JSR  CHROUT
8038: E8           26             INX
8039: D0 F5        27             BNE  LOOP2
803B: 20 9F FF     28   SCAN2     JSR  SCNKEY
803E: 20 E4 FF     29             JSR  GETIN
8041: F0 F8        30             BEQ  SCAN2
8043: C9 0D        31             CMP  #$0D        ;RETURN PRESSED?
8045: F0 03        32             BEQ  DONE2       ;IF Y WE'RE DONE
8047: 4C 3B 80     33             JMP  SCAN2       ;IF N GO BACK
804A: 60           34   DONE2     RTS
804B: CE CF D7     35   STRING2   ASC  "NOW ITS FILE TWO",00
804E: A0 C9 D4 D3 A0 C6 C9 CC
8056: C5 A0 D4 D7 CF 00

--End assembly, 92 bytes, Errors: 0
```