EF9345 SEMI-GRAPHIC DISPLAY PROCESSOR GENERAL APPLICATION PRINCIPLES

AUGUSTIN GIADIN

ABSTRACT

Associated with a standard memory package, the EF9345 allows full implementation of a low-cost terminal display unit.

The aim of this Application Note is to aid the user in using the EF9345. Design considerations and programming of the circuit in the various operating modes will be discussed.

CONTENTS

MICROPROCESSOR INTERFACE

GENERAL PRINCIPLES

The EF9345 interfaces to a microprocessor by :

- an 8-bit address/data multiplexed bus AD(0:7)
- four control signals : AS (Address Strobe), DS (Data Strobe), R/W (Read/Write) and CS (Chip Select).

Each microprocessor access is made as follows :

- First the AS signal falling edge latches the DS, CS and AD(0:7) input. The EF9345 is selected only when CS is strobed low and AD(7:4) most significant bits of the address lines are strobed with the binary value 0010. The latched level of DS signal selects either the Intel mode (DS high) or the 6801 mode (DS low).
- During the second part of the access cycle, the AD(0:7) lines become the data bus. In the 6801 mode, data exchange is made while DS is high and the R/W signal specifies the data transfer direction (a write operation into the circuit is performed when R/W is low). In the Intel mode, DS is generally used as a RD (Read) signal and R/W as a WR (Write) signal.

So connecting the EF9345 to a multiplexed bus microprocessor is quite simple. Figures 1 and 2 show the interface with an EF6801 and an Intel type microprocessor (8085, 8051...).

Note : As the EF9345 is selected when the latched address binary value is 0010XXXX (or 2X in hexadecimal), the circuit takes 16 consecutive address locations in the microprocessor addressing space. These addresses correspond to 8 internal registers of the circuit, with each register selected by the three LSB of the address value (see programming description).

INTERFACE WITH A NON-MULTIPLEXED BUS MICROPROCESSOR

When the EF9345 is used with a non-multiplexed bus microprocessor such as EF6800, EF6809, Z80..., the microprocessor address and data lines must be generally multiplexed to pins AD(0:7). The address strobe and multiplexer command signals must also generated. Figure 3 shows an example of interfacing the EF9345 to an EF6800/6809 microprocessor, where address and data multiplexing is made with three-state buffers. The AS signal and the buffer enable signals are generated from the E signal with a few TTL-LS circuits. Figure 4 shows the associated timing diagram.

By using the principle described below, it is possible to realize the EF9345 interface with a non-multiplexed bus microprocessor without multiplexing the address and data lines. This principle allows reducing the number of TTL parts for the hardware interface implementation, but requires a few additional instructions when programming the circuit.

Figure 5 illustrates the principle for an EF6800/6809 application. The AD(0:7) pins are directly connected to the microprocessor data bus and the CS input is grounded. An address decoder provides two chip-select signal CS0 and CS1. Any microprocessor write operation to the address which generates CS0 low will result in an AS pulse while E is high and the data present on AD(0:7) are latched into the EF9345 as an "address". During an access to the address generating CS1 low, a DS pulse is generated while E is high and AD(0:7) act as a normal data bus, provided that the circuit has been previously selected.

So any micoprocessor access to the EF9345 is made in two steps :

- first the microprocessor must write at address CS0 a data whose binary value is 0010XXXX to select the circuit and to specify by XXXX what register is to be accessed,
- a normal data exchange (read or write operation) can then be made at address CS1 between the microprocessor and the EF9345 register selected during the first cycle.

Flowchart given in figure 6 shows how the microprocessor can read the status register RO.

This principle can be applied to any microprocessor type. Figure 7 shows an implementation example for interfacing with a Z80, where the AS pulse is generated during an I/O write operation at address A7 = 1, A6 = A5 = 0. Access to an EF9345 register is made by an I/O read or write at address A7 = 1, A6 = 1 and A5 = 0. As DS (CS1) is high when AS occurs, the EF9345 is here in the Intel mode.

Figure 1 : Interface with EF6801.

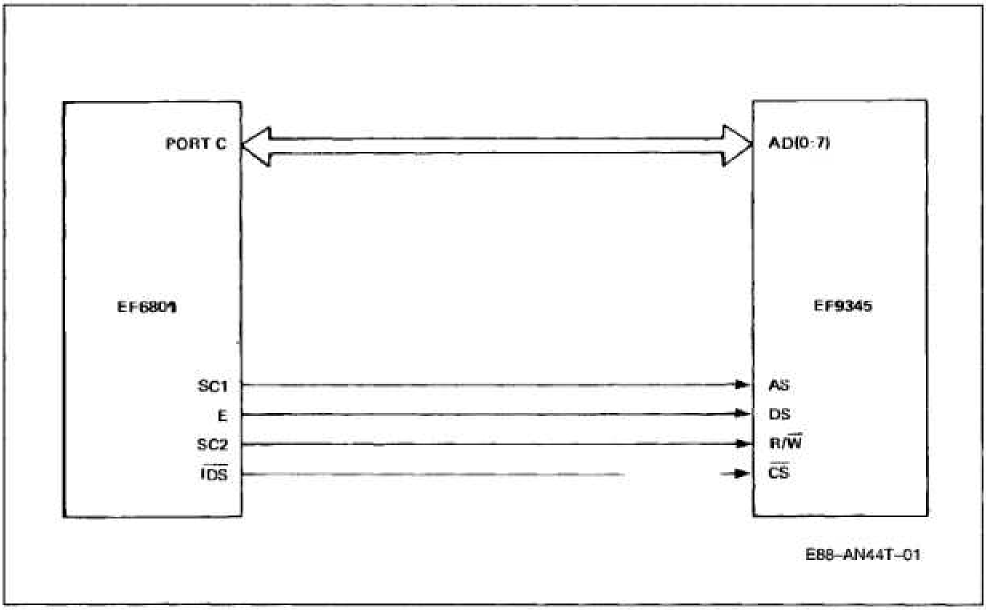


Figure 2 : Interface with a Multiplexed Bus Intel Type Microprocessor.

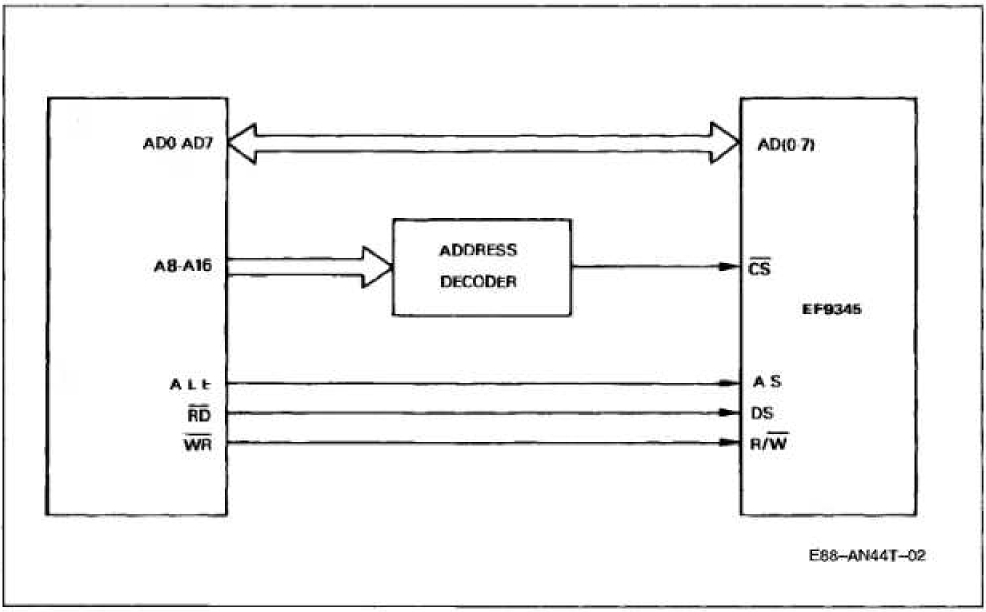
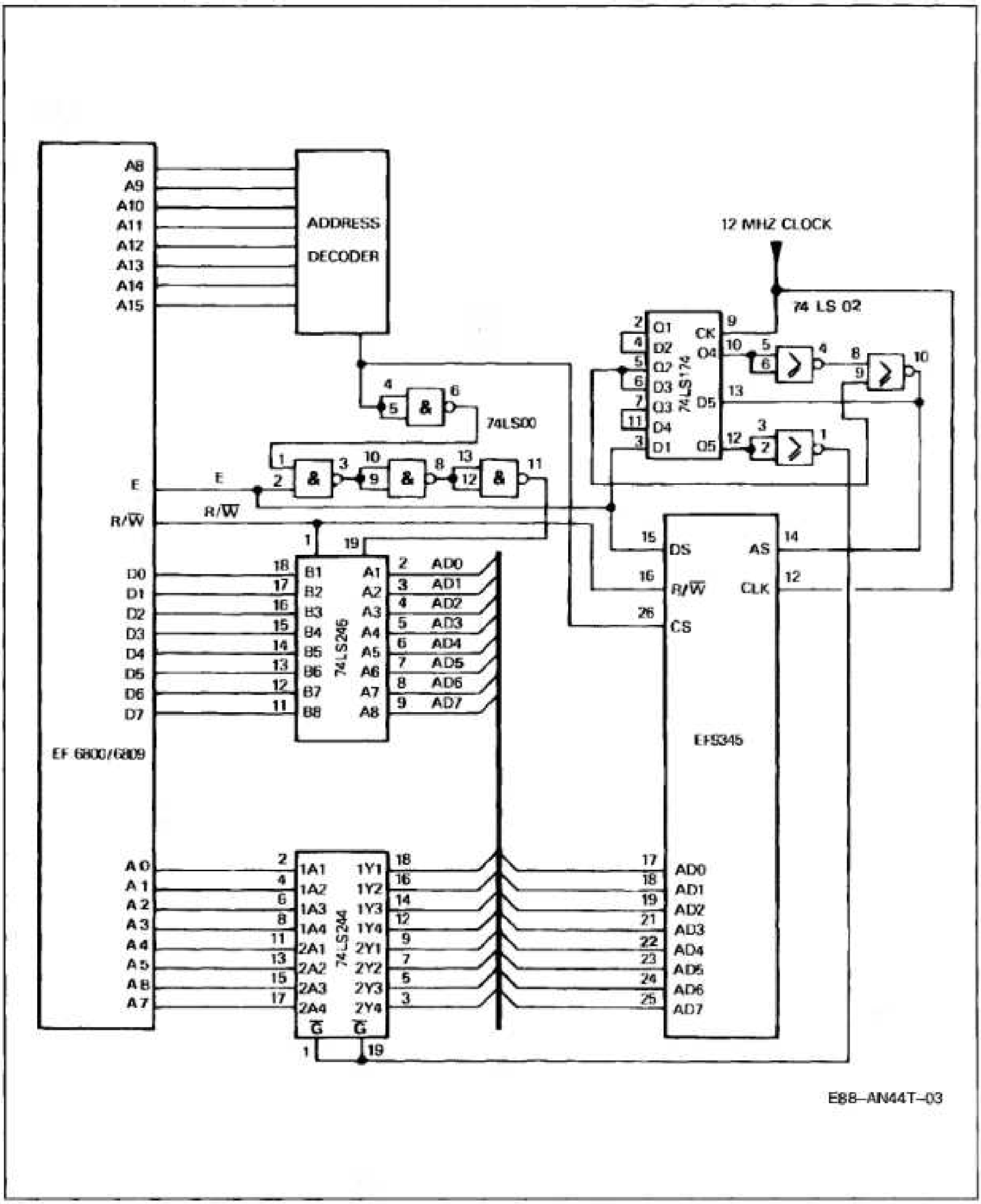


Figure 3 : Interface with EF6800/6809 by Multiplexing Address and Data Bus.



SGS-THOMSON MICROELECTRONICS

Figure 4 : Timing Diagram Associated with Figure 3.

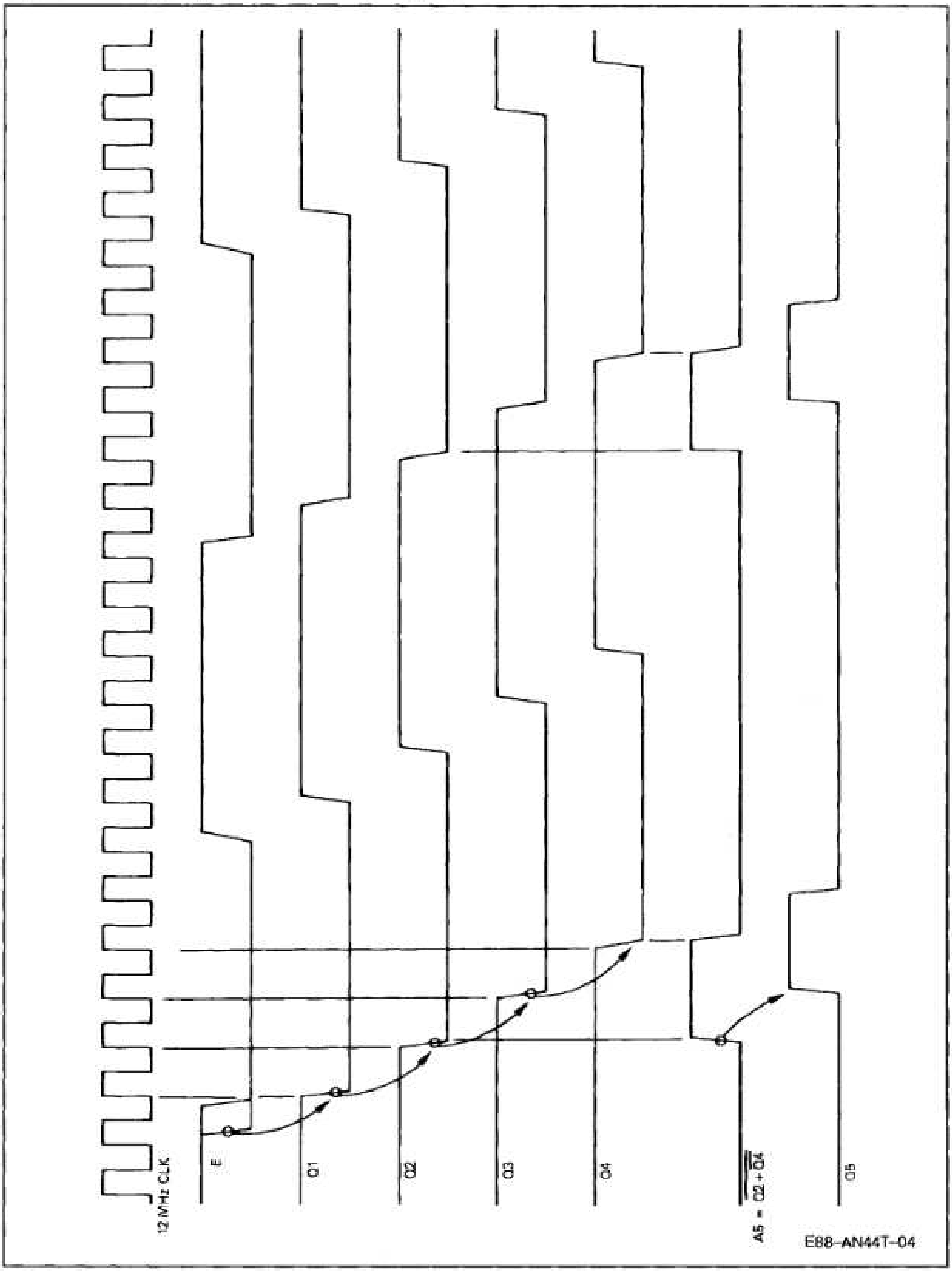


Figure 5 : Interface with EF6800/6809 without Multiplexing Address and Data Bus.

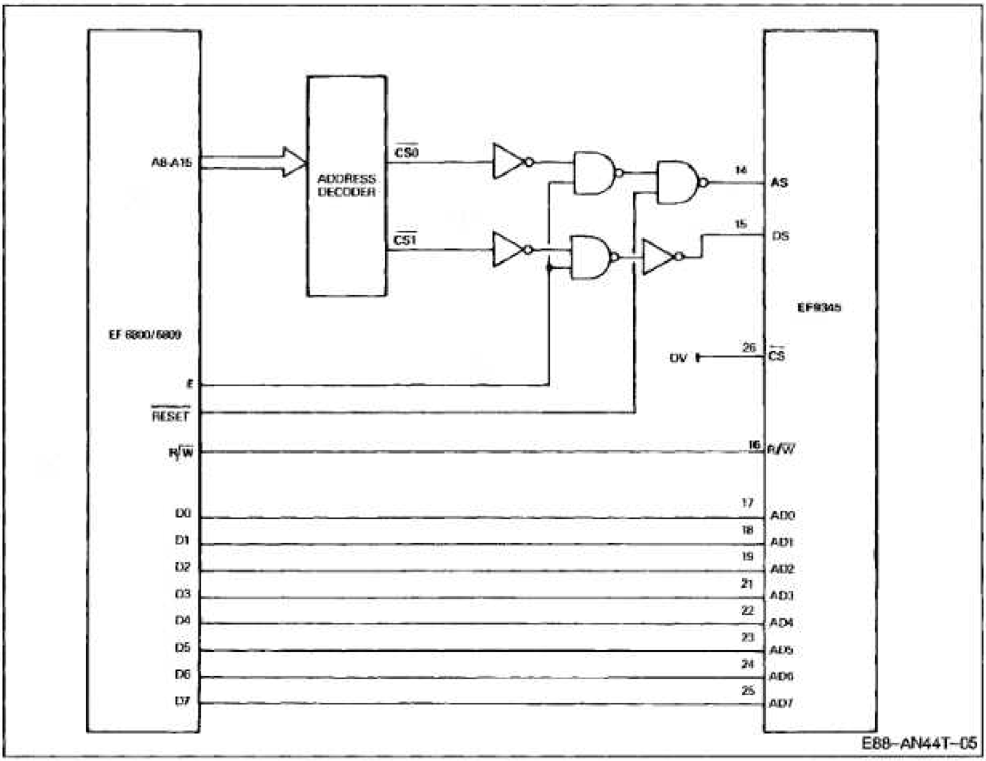


Figure 6 : Access to an EF9345 Register when Using the Non-Multiplexing scheme Interface.

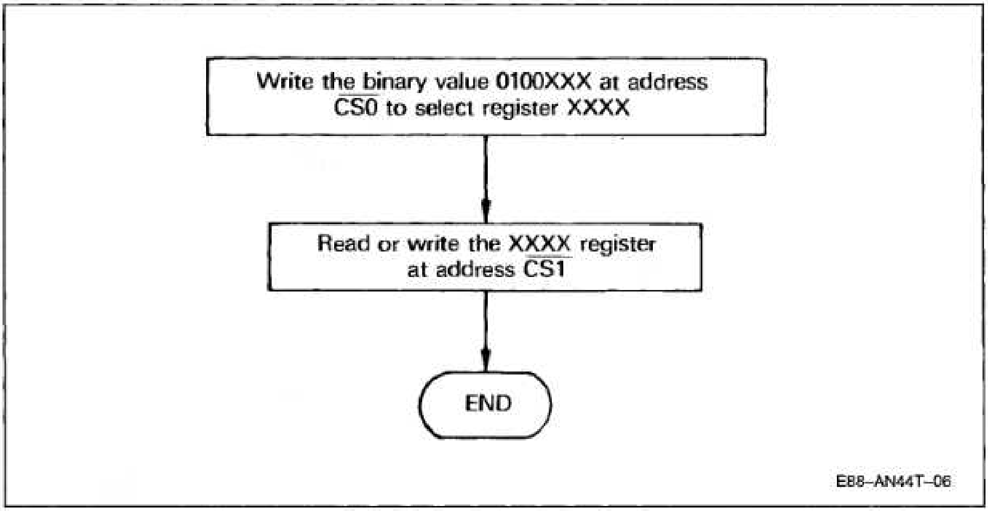
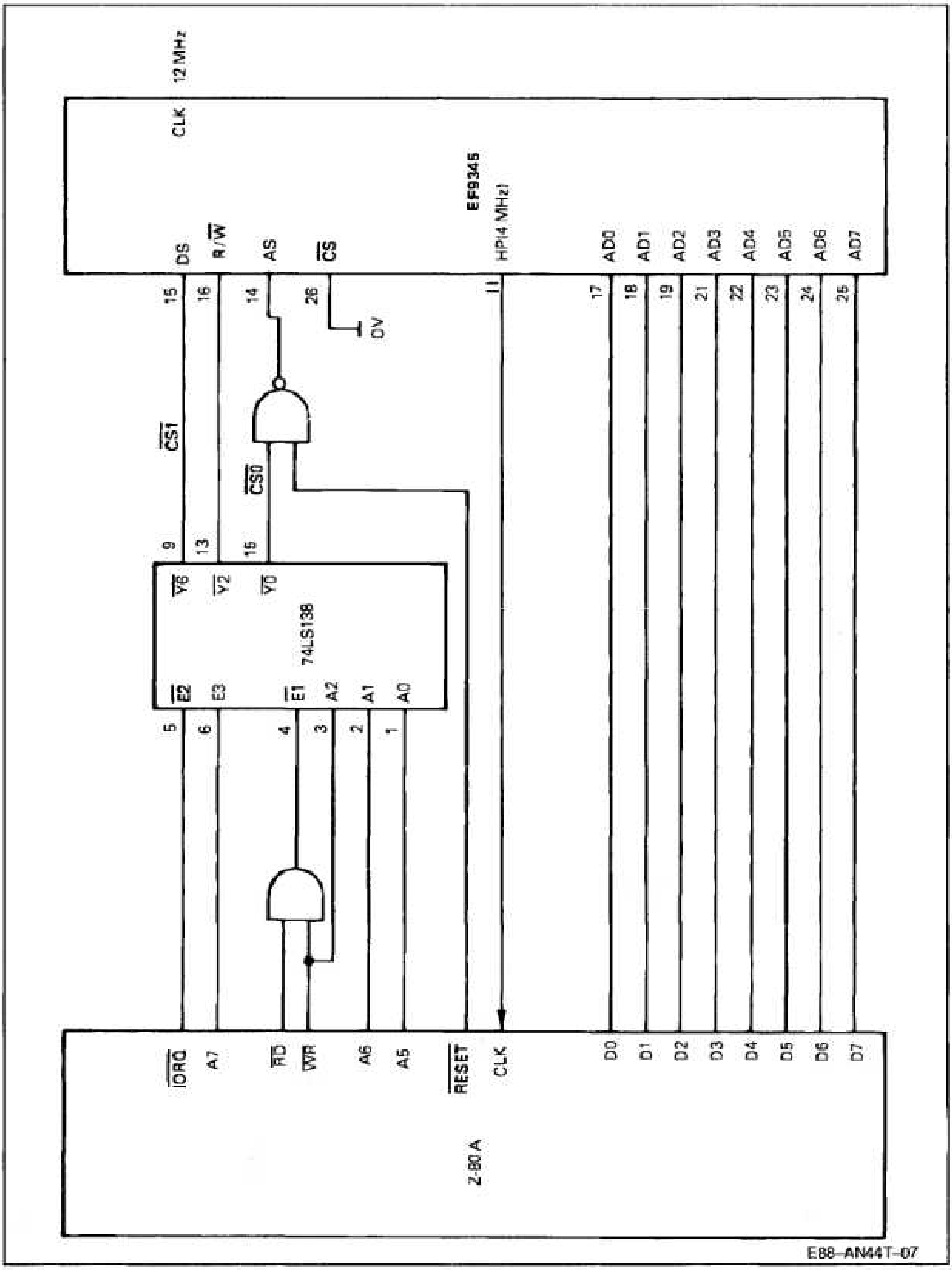

Write the binary value 0100XXX at address $\overline{CS0}$ to select register XXXX
Read or write the XXXX register at address $\overline{CS1}$
END
E88-AN44T-06

Figure 7 : EF9345 Interface with a Z-80 without Multiplexing Address and Data Bus.



MEMORY INTERFACE

The EF9345 can be used with a wide variety of standard memories and manages up to 16 kbytes of private memory.

The memory interfaces is made by :

an 8-bit address/data multiplexed bus ADM(0:7)

a 6-bit high order address bus AM(8:13)

three control signals : $\overline{OE}$ (Output Enable), $\overline{ASM}$ (Address Strobe Memory), $\overline{WE}$ (Write Enable).

During each memory cycle, the EF9345 outputs to ADM(0:7) low order address byte while $\overline{ASM}$ is high. The high order address bits are provided on AM(8:13) during the whole memory cycle. When $\overline{ASM}$ goes low, the ADM(0:7) lines become the memory data bus. For a read operation, the $\overline{OE}$ signal is active low to enable the memory output buffers. A write operation is made when $\overline{WE}$ is low.

INTERFACE WITH 2K*8 STATIC MEMORY

As the address lines are generally not latched by static RAMs, an external 8-bit latch (74LS373) must be used to store the low order address bits ADM(0:7) on the falling edge of $\overline{ASM}$ signal.

INTERFACE WITH 8K* 8 PSEUDO-STATIC RAM

The EF9345 can be directly connected to an 8K*8 pseudo-static RAM (NEC µPD 4168, INTEL 2187, INMOS 2630...). The $\overline{ASM}$ signal is fed to the $\overline{CE}$ input which latches the address lines. As the EF9345 performs DRAM refresh, the memory internal refresh circuitry is not use.

The schematic diagram of figure 8 gives a design example which allows interfacing the EF9345 to 2K*8 or 8K*8 memory. With static memory, the 8 jumpers of S8 are connected to provide the low order address lines from the 8-bit latch 74LS373. With pseudo-static memory, the 74LS373 is useless and the 8 jumpers of S7 are connected. Jumpers S1 to S6 are set in position 2 for 2K*8 RAMs, and in position 1 for 8K*8 RAMs.

INTERFACE WITH 16K*8 DRAM (see figure 9)

When using 16K*4 dynamic RAMs, the address provided by the EF9345 must be multiplexed to obtain the Row and Column address. $\overline{ASM}$ can be used directly as the $\overline{RAS}$ (Row Address Strobe) signal, but the CAS signal must be externally generated. Figure 9 shows an example of generating $\overline{CAS}$ and the multiplexer command signals from $\overline{ASM}$.

As previously, refresh operation is performed by the EF9345.

PROGRAMMING THE EF9345 - GENERAL PRINCIPLES

DIRECT ACCESS REGISTERS

As described in the microprocessor interface chapter, the EF9345 is accessed by the microprocessor at 16 consecutive locations from address XX20 to XX2F (hexadecimal), where XX is determined by the user's address decoding. These 16 addresses correspond to 8 internal registers RO to R7 (see figure 10). Each register can be accessed at two addresses : a lower address (bit 3 = 0) and an upper address (bit 3 = 1). For example, if the EF9345 is mapped in the microprocessor addressing space from F420 to F42F, register R1 can be read or written at both addresses F421 and F429.

However, a command present in register RO is executed only after an access to a register at an upper address. This scheme allows re-executing a same command by loading only one argument into an upper address register.

COMMAND EXECUTION

RO is a write command register and a read status register. A command present in RO is executed with the arguments in the other direct access registers after any access to a register at an upper address (from XX28 to XX2F).

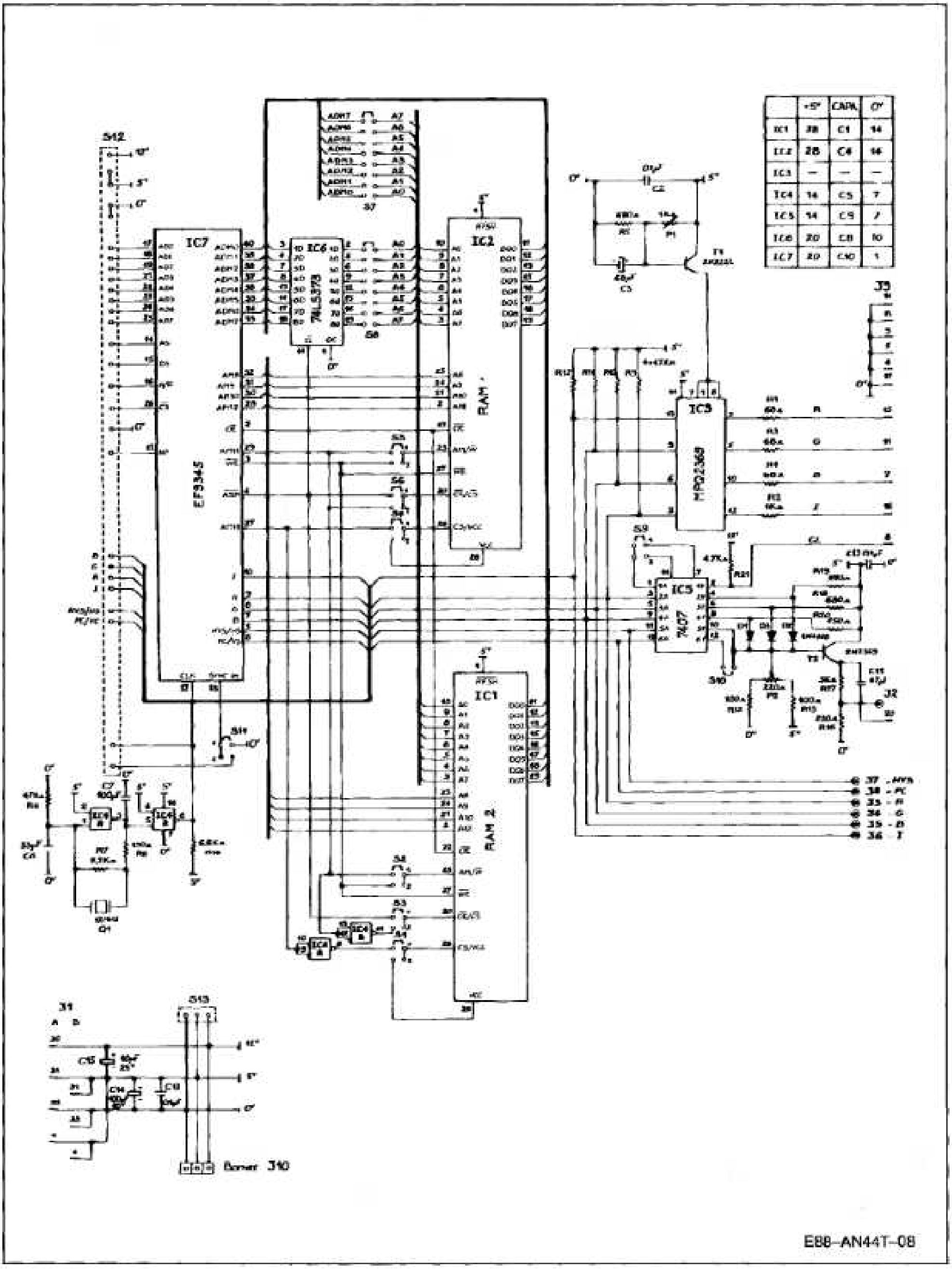
Before any access to a register, the Busy status in the Status register bit 7 must be tested to check a command is not currently executing. However, after power-up a NOP command should be executed without testing the Busy state to set the circuit into a determined state before further operation. A move command with no stop condition can also be aborted by executing a NOP command.

INDIRECT ACCESS REGISTER (figure 11)

The EF9345 has 5 indirect access registers which define the various operating modes of the circuit : TGS, MAT, PAT, DOR, ROR. Each of these registers is assigned an index r and is indirectly accessed through register R1. Data is transfered between R1 and an indirect access register with the IND command, which specifies the transfer direction (bit R/W) and the register index r (bits 0 to 2).

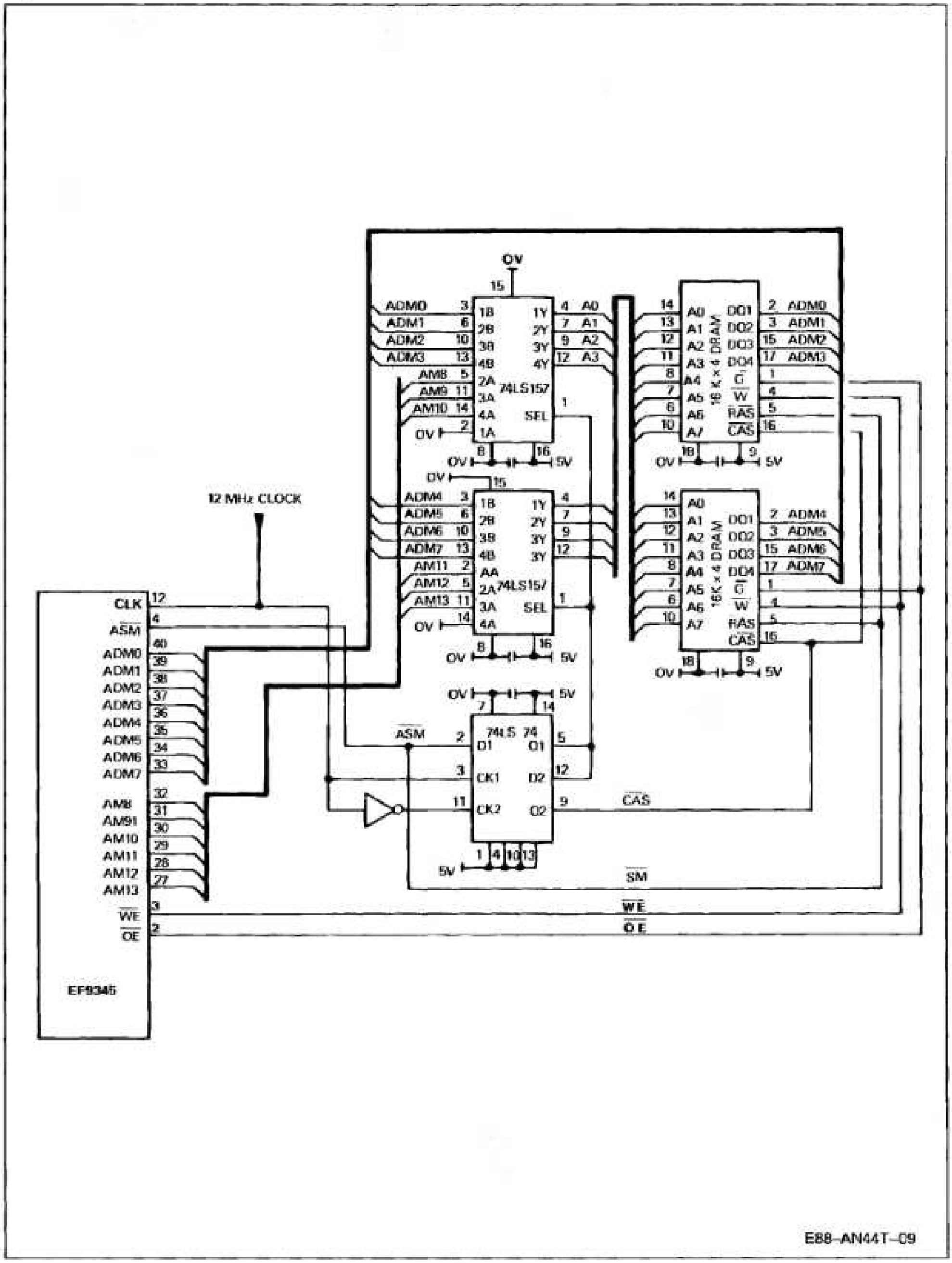
Flowchart of figure 12 gives an example of indirect access register loading.

Figure 8 : EF9345 Interface with 2K x 8 and 8K x 8 Memory.



	+5V	CAPA.	0V
IC1	28	C1	14
IC2	28	C4	14
IC3	—	—	—
IC4	14	C5	7
IC5	14	C9	7
IC6	20	C8	10
IC7	20	C10	1

Figure 9 : Interface with 16 x 4 Dram.



SGS-THOMSON MICROELECTRONICS

Figure 10 : Direct Access Registers.

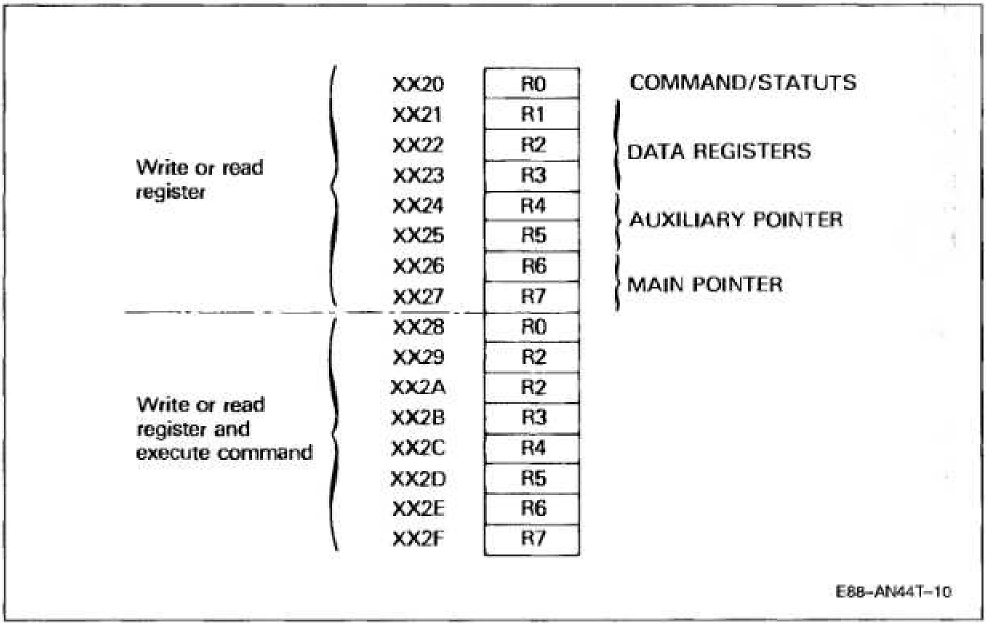


Figure 11 : Indirect Access Registers.

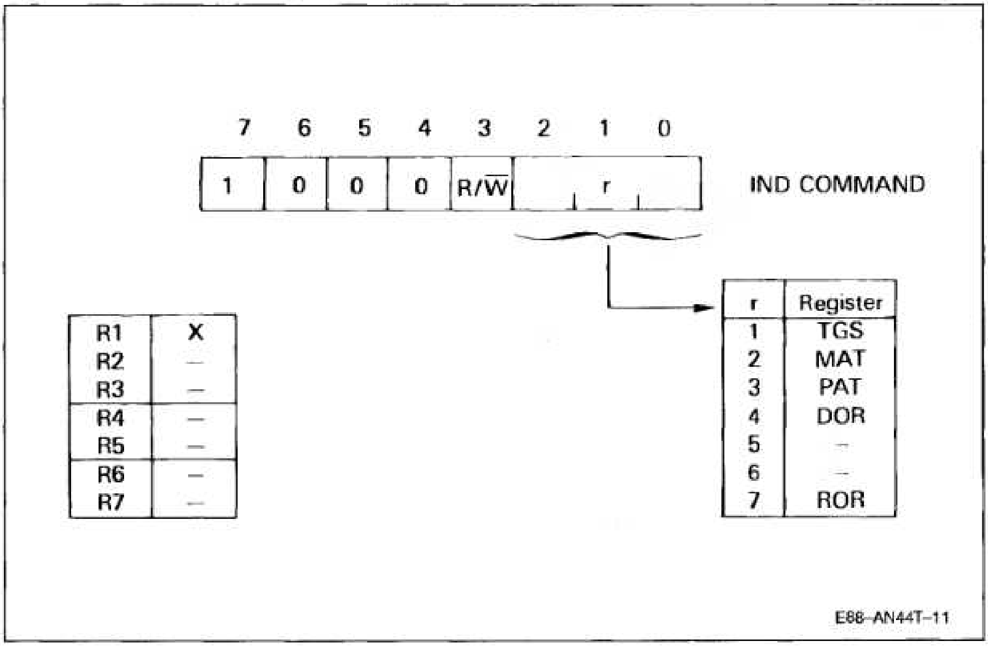
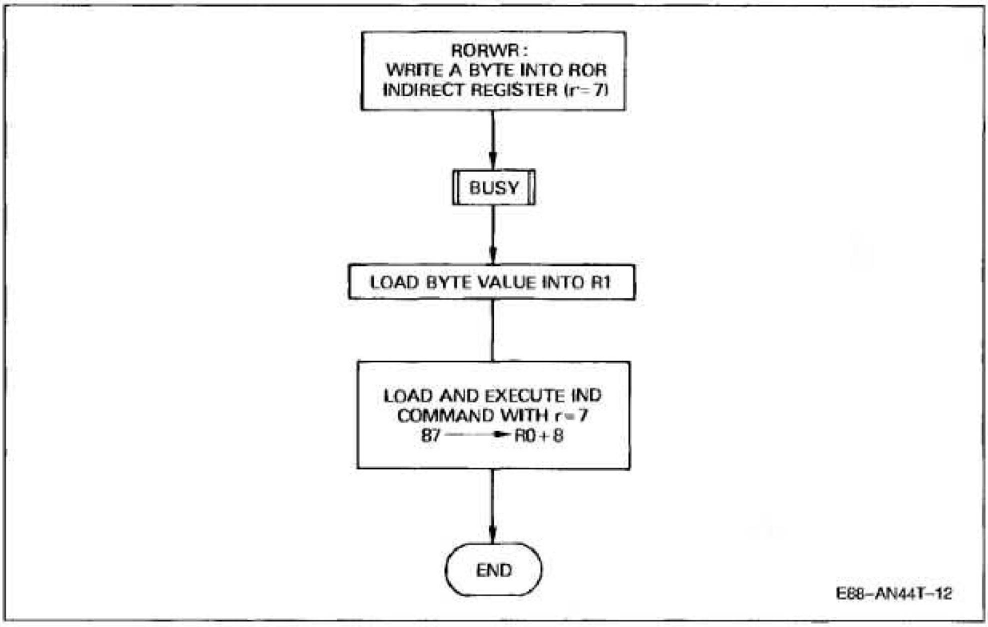


Figure 12 : Indirect Register Loading Example.



PROGRAMMING THE EF9345 IN 40 CHAR/ROW MODE

In the char/row mode, a page displayed by the EF9345 is made of 25 or 21 rows, each containing 40 character windows. A window is composed by 8 pixels and 10 lines.

Each window is associated with a character code in a page memory. One of three character code formats can be selected for a page :

- Fixed long codes (24 bits)
- Fixed short codes (16 bits)
- Variable codes (8/24 bits).

In this document, only fixed long code format will be discussed. With this format, each character window on the screen is associated with a 3 byte code, namely the C, B and A bytes. Interpretation of these bytes depends on the character type.

BICHROME CHARACTER CODE

For a bichrome character, the A byte defines :

- a background color
- a foreground color
- the negative (reverse video) attribute N
- the flash (blink) attribute F.

The B byte defines :

- a character set
- insert, double height, double width, and conceal attributes.

For bichrome characters, bits B (7:6) must differ from 11.

The C byte selects one of 128 characters in a character set. With the fixed long code format, bit C7 is don't care.

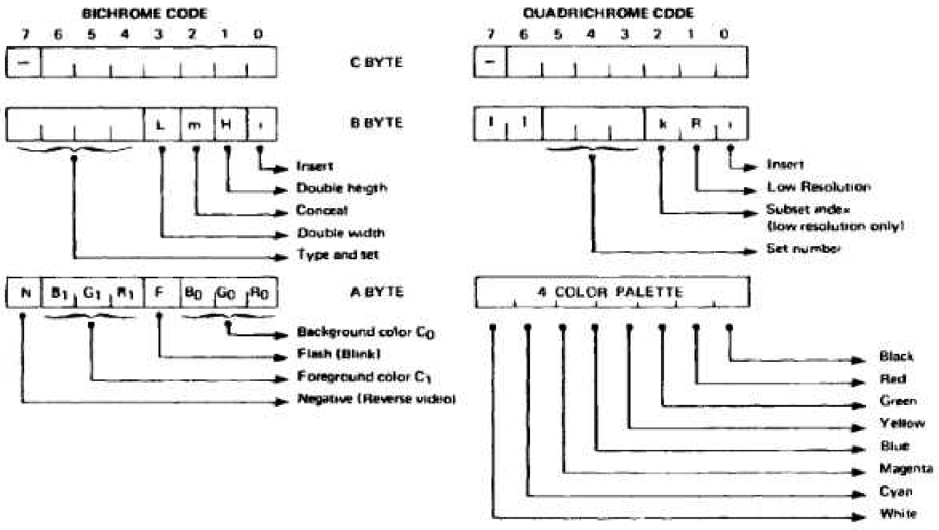
Example : to write a "B" with the following attributes :

- background color = blue
- foreground color = yellow
- flashing
- alphanumeric set G_0.

The hexadecimal values for the character code bytes are :

- C byte = 42
- B byte = 00
- A byte = 3C.

Figure 13 : 40 Char/Row Fixed Long Codes.



B	G	R	Color Value
0	0	0	Black
0	0	1	Red
0	1	0	Green
0	1	1	Yellow
1	0	0	Blue
1	0	1	Magenta
1	1	0	Cyan
1	1	1	White

E88-AN44T-13

QUADRICHROME CHARACTER CODE

Quadrichrome characters allow displaying up to 4 different colors in any 8 pixels by the 10 lines window, at the penalty of a halved horizontal resolution. By programming the R attribute in the character code B byte, the vertical resolution can be kept or halved.

For each quadrichrome character window, the A byte defines an ordered 4 color palette from 8 possible colors. Each bit is associated with a color which is selected when the corresponding bit is set. If more than 4 bits are set, higher ranking bits are ignored. When less than 4 bits are set, the color palette is implicitly completed with "white" value.

Example : A = 54 selects the red, yellow, blue and cyan colors.

A = 73 selects the black, red, blue and magenta colors. Bit 6 is set but ignored.

The character code B byte defines :
- a set number Q0 to Q7 by bits B (3:5)
- high or low resolution bit R. Bit R = 0 selects a high resolution quadrichrome and bit k is don't care.
  If R = 1, the character is a low resolution quadrichrome and k definies a subset index.
- bit i definies the character to be inserted or not.

The character code C byte selects one from 100 characters in a set. This byte can take values from 00 to 03 and from 20 to 7F (hexa).

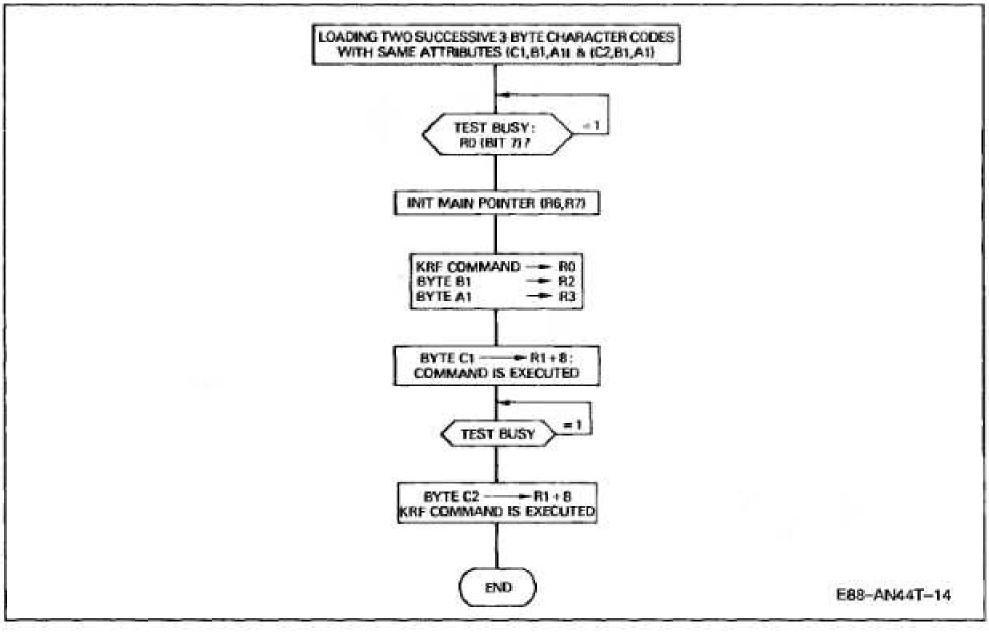
HANDLING LONG CHARACTER CODE

The KRF command allows an easy, X, Y random access or an X sequential access to the page memory. Data registers R1, R2 and R3 are used to transfer respectively the character code C, B and A bytes. The Main Pointer is used to address the page memory and specifies :
- a row number Y = (0 ; 8 to 31)
- a column position on a row X = (0 to 39)
- the first block number of the page memory Z (0:3).

Notes : 1. R6(6) is used by the Auxiliary Pointer
2. Order of bits Z0-Z1 are reversed in R7
3. When using pointer incrementation in KRF command (bit 0 = 1 in the command code), only the X part of R7 is incrementated modulo 40 after the command execution. No Y incrementation is made when X overflows from 39 to 00.
4. The cursor position one the screen is given by the Main Pointer.

A character code loading flowchart example is given in figure 14.

Figure 14 : Long Character Code Loading Example.


LOADING TWO SUCCESSIVE 3 BYTE CHARACTER CODES WITH SAME ATTRIBUTES (C1,B1,A1) & (C2,B1,A1)
TEST BUSY : R0 (BIT 7) ?
= 1
INIT MAIN POINTER (R6,R7)
KRF COMMAND → R0
BYTE B1 → R2
BYTE A1 → R3
BYTE C1 → R1 + 8 : COMMAND IS EXECUTED
TEST BUSY
= 1
BYTE C2 → R1 + 8 KRF COMMAND IS EXECUTED
END
E88-AN44T-14

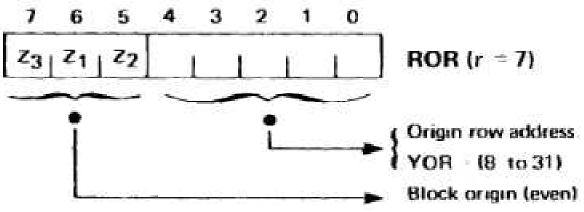
PAGE MEMORY SELECTION

In 40 char/row with the long code format, each character window on the screen is associated with 3 bytes in a page memory. As each displayed page contains up to 1000 windows (25 rows of 40 characters each), a page memory is made of three 1 Kbyte blocks. The first block holds the C bytes, the second one the B bytes and the last one the A bytes.

As the EF9345 can address up to 16 Kbytes of external memory, a page memory address must be selected by the user with the following requirements :

- the three blocks must be consecutive and lie in the same district, i.e. the two MSB Z3-Z2 of the block numbers must be the same
- the first block number must be even (Z0 = 0).

The base address of the page memory to be displayed on the screen, which is the first block number, is given in register ROR(5:7). As Z0 is implicity 0, it is not specified in ROR.



Example : with the displayed page memory starting from block number 4, Z3-Z2-Z1-Z0 = 0100 and ROR7-ROR6-ROR5 = 001.

Notes : 1. Order of bits Z1-Z2 is reversed in ROR.

2. Each page displayed by the EF9345 comprises a service row, which is always displayed on the stop of the screen, and 24 remaining rows. When accessing to the page memory, the service row number is Y = 0 and the remaining row number ranges from 08 to 31. Bits ROR(0:4) constitute the YOR origin register, which specifies the number of the first row displayed after the service row. By programming YOR from 8 to 31, the user can realise roll-up and roll-down operation.

USER DEFINED CHARACTER SET (UDS)

In 40 char/row mode, the User Defined Character Set (UDS) allows the user to define additional characters whose shapes can be dynamically loaded into the external character generator. The EF9345 can provide up to :

- 100 alphanumeric type UDS character (G'_0 set)
- 200 semi-graphic type UDS characters (G'_{1X} set)
- 800 quadrichrome UDS characters (Q_0 to Q_7 sets).

Alphanumeric and semi-graphic UDS are bichrome characters, with the difference that only alphanumerics can be underlined.

BICHROME UDS CHARACTERS

The shape of a bichrome character is defined in a 8 pixels by 10 lines dot matrix. Each line of the dot matrix is coded in the external character generator by an 8 bit value, or a slice byte. So a bichrome UDS character is defined by 10 slice bytes.

A slice byte value is obtained in the following way : on a line of the dot matrix, the dots defining the character shape are coded by a "1", the other dots by a "0". This eight bit result is then order reversed to obtain the value to be loaded into the external character generator. Figure 15 shows a slice coding example for a bichrome UDS character.

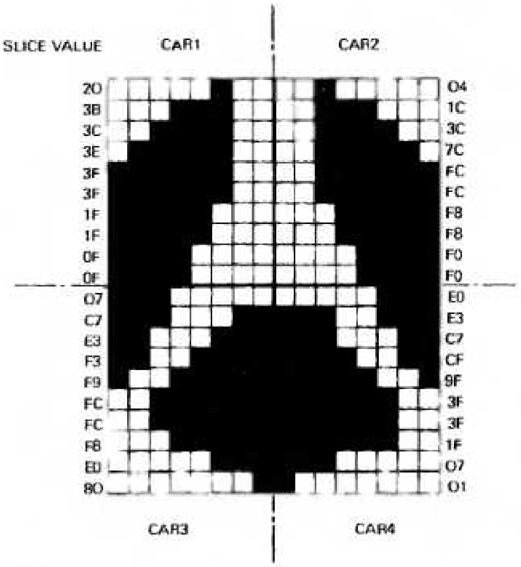
QUADRICHROME UDS CHARACTERS

An 8 pixels by 10 lines window displaying a quadrichrome character on the screen is composed by elementary "dots" whose size is :

- 2 pixels by 1 line for high resolution quadrichrome
- 2 pixels by 2 lines for low resolution quadrichrome.

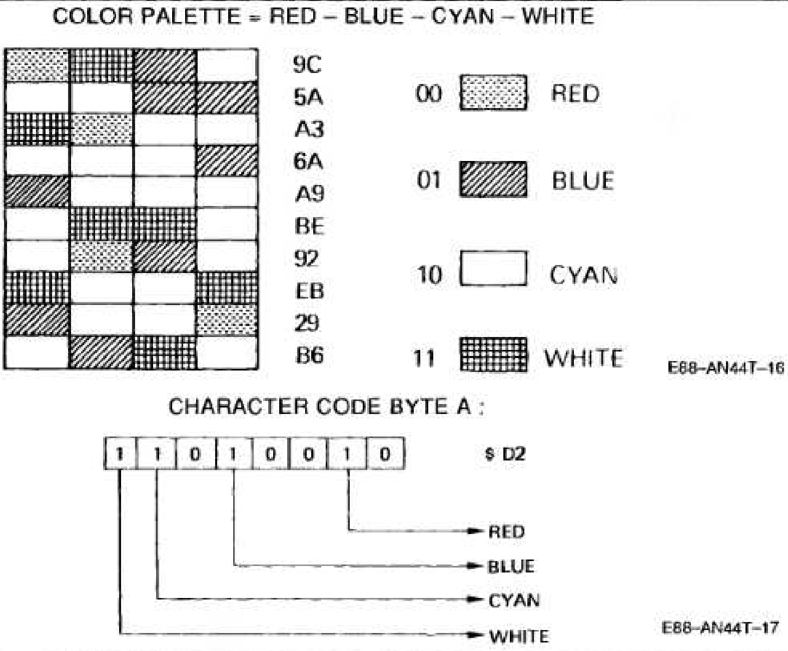
Each dot can take one of the 4 colors selected by the palette A byte of the character code associated to the window. So a quadrichrome character shape is defined by a 4 * 10 or 4 * 5 dot matrix, with each dot coded bit a two-bit value. Each line of the dot matrix is coded by a slice byte in the external character generator. A high resolution quadrichrome requires 10 slice bytes to be defined, and a low resolution quadrichrome 5 slice bytes.

Figure 15 : Bichrome UDS Slice Coding Example.



E88-AN44T-15

Figure 16 : Quadrichrome Slice Coding Example.



E88-AN44T-16

E88-AN44T-17

The 4 colors selected by the character code A byte are ordered. For example, if the A byte hexadecimal value is 5A, the 4 ordered colors are :
- Red with the binary rank 00
- Yellow with the binary rank 01
- Blue with the binary rank 10
- Cyan with the binary rank 11.

A slice byte is obtained by assigning to each dot the binary rank of its color, with the value for the right dots placed in the most significant position of the slice byte. Figure 16 shows a slice coding example for a quadrichrome character.

DOR REGISTER

During the display process, the base address for each UDS character generator is given in DOR register (see figure 17) :
- DOR(0:3) hold the number of the block which contains the alphanumeric UDS slices (G'_0).
- For semi-graphic UDS, the slice block number is given by DOR(4:6) and bit 4 of the character code B byte. So for UDS G'_{10} the slice block number is even (B4 = 0) and the following block contains slices for UDS G'_{11} (B4 =1).
- For each quadrichrome UDS (Q0 to Q7), the slice block number is given by DOR7 and bits B(5:3) of the character code, which select also the set.

ACCESS TO UDS SLICES IN MEMORY

A UDS slice address in memory is given by :
- a block number Z(0:3)
- the character code C byte : C(0:6)
- the slice number NT. For bichrome and high resolution quadrichrome, NT ranges from 0 to 9. For low resolution, quadrichrome, NT ranges from 0 to 9. For low resolution quadrichrome, NT ranges from 0 to 4 when K = 0 and from 5 to 9 when k = 1 (k is in bit 2 of character code B byte).

A UDS slice can be written into or read from the EF9345 private memory with the OCT command. This command uses register R1 for slice transfer and the Main or Auxiliary Pointer for slice addressing. As the Main Pointer generally points to the cursor position on the screen and is used for character code access, the Auxiliary Pointer should rather be used for slice access. Figure 18 shows how the Auxiliary Pointer value is obtained from the slice address :
- R4 holds bits C(2:6) of the character code and bit Z2 of the block number
- R5 holds bits C(0:1), the slice number NT and bits Z0-Z1
- Bit 6 of R6 holds bit Z3 of the block number.

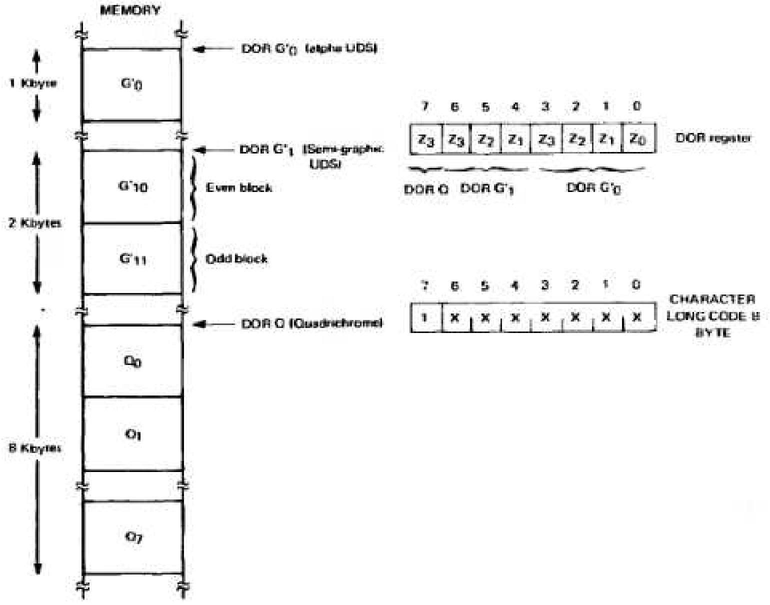
Figure 19 shows a flowchart example for loading 10 slices.

Note : As the slice number NT is not in the least significant bits of R5, executing the OCT command with pointer incrementation does not result in slice number incrementation.

SCREEN MAPPING WITH UDS CHARACTERS

In 40 char/row mode, the screen is made of 1000 windows. Each window can be assigned a UDS character to obtain a likely bit-mapped screen and to produce complex pictures. Up to 300 screen windows can be mapped with a 320 by 250 resolution and independant two color set in each window by bichrome characters. In the same way, quadrichrome characters allow mapping up to 800 (resp. 1600) windows with a 160 * 250 (resp. 160 * 125) resolution and with a selectable four color set for each window.

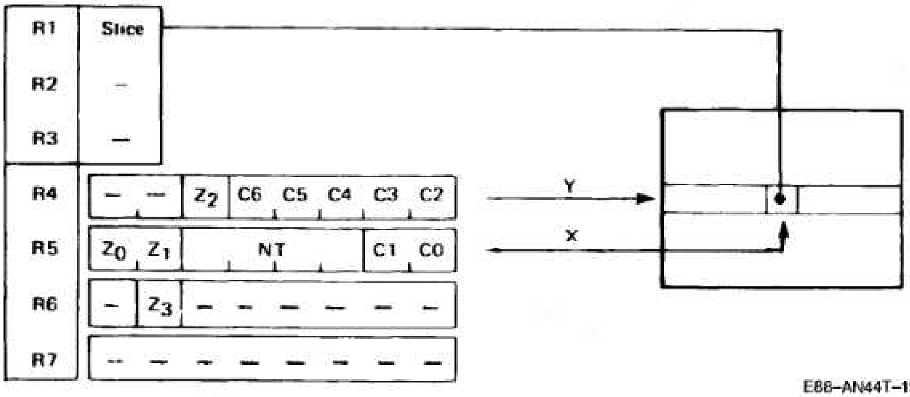
Figure 17 : UDS Fetch to Display.



UDS Set				Z Address			
#	B7	B6	B5	Z_3	Z_2	Z_1	Z_0
G'$_0$	1	0	0	DOR_3	DOR_2	DOR_1	DOR_0
G'$_{11}$	1	0	1	DOR_6	DOR_5	DOR_4	B4
Q_0–Q_7	1	1	X	DOR_7	B5	B3	B4

E88–AN44T–18

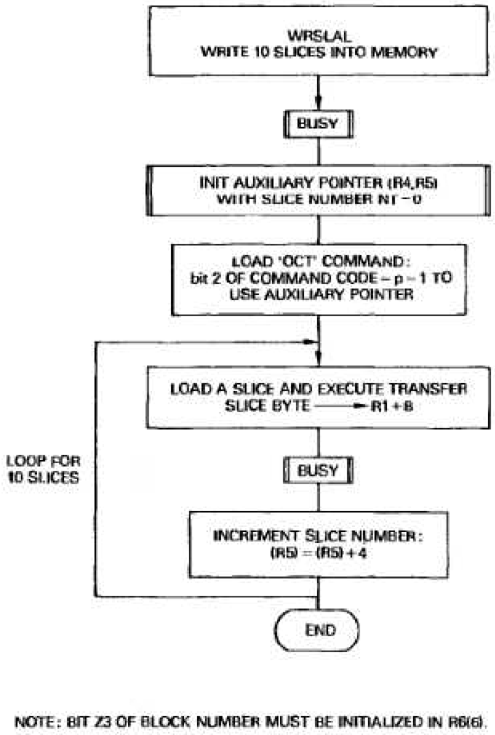
Figure 18 : Accessing a Character Slice in Memory Using Oct Command with Auxiliary Pointer.



E88–AN44T–19

SGS-THOMSON MICROELECTRONICS

Figure 19 : UDS Slice Loading Flowchart.


WRSLAL
WRITE 10 SLICES INTO MEMORY
BUSY
INIT AUXILIARY POINTER (R4,R5) WITH SLICE NUMBER NT = 0
LOAD 'OCT' COMMAND: bit 2 OF COMMAND CODE = p = 1 TO USE AUXILIARY POINTER
LOAD A SLICE AND EXECUTE TRANSFER SLICE BYTE → R1 + 8
LOOP FOR 10 SLICES
BUSY
INCREMENT SLICE NUMBER: (R5) = (R5) + 4
END
NOTE: BIT Z3 OF BLOCK NUMBER MUST BE INITIALIZED IN R6(6).

Note : Bit Z3 of block number must be initialized in R6(6).

E88-AN44T-20

PROGRAMMING EXAMPLE IN 40 CHAR/ROW

```
PAGE  001  EF40    .SA:0

                             OPT    LLE=110
                      *
                      * EF9345 PROGRAMMING EXAMPLE IN 40 CHAR/ROW
                      * THIS PROGRAM IS WRITTEN IN 6809 ASSEMBLER LANGUAGE.
                      * AFTER INITIALIAZING THE EF9345 INDIRECT REGISTERS
                      * AND CLEARING THE SCREEN, THE THOMSON LOGO OF FIGURE 15
                      * AND THE UDS CHARACTER OF FIGURE 16 ARE DISPLAYED
                      * ON THE SCREEN.
                      *

                      * EF9345 REGISTER ADDRESS

            F420   A RO     EQU    $F420     COMMAND/STATUS REGISTER
            F421   A R1     EQU    RO+1      DATA REGISTERS
            F422   A R2     EQU    RO+2
            F423   A R3     EQU    RO+3
            F424   A R4     EQU    RO+4      AUXILIARY POINTER (Y)
            F425   A R5     EQU    RO+5      AUXILIARY POINTER (X)
            F426   A R6     EQU    RO+6      MAIN POINTER (Y)
            F427   A R7     EQU    RO+7      MAIN POINTER (X)

            F425   A XA     EQU    R5
            F424   A YA     EQU    R4
            F427   A XP     EQU    R7
            F426   A YP     EQU    R6

            4000   A STACK  EQU    $4000
            3F80   A STACKU EQU    STACK-128

A 1000                      ORG    $1000

            1000   A MAIN   EQU    *

A 1000 10CE 4000   A        LDS    #STACK    STACK INITIALIZATION
A 1004 CE   3F80   A        LDU    #STACKU

A 1007 86   91     A        LDA    #$91      LOAD AND EXECUTE A "NOP" COMMAND
A 1009 B7   F428   A        STA    RO+8      WITHOUT TESTING BUSY

                      *
                      * TGS REGISTER INITIALIZATION :
                      * TGS0 = 0 : 625 LINES (50 HZ)
                      * TGS1 = 0 : NOT INTERLACED
                      * TGS2 = 0 : HORIZONTAL RESYNC. DISABLED
                      * TGS3 = 0 : VERTICAL RESYNC. DISABLED
                      * TGS4 = 0 : HORIZONTAL SYNC. ON HVS/HS PIN AND
                      *            VERTICAL SYNC. ON PC/VS PIN
                      * TGS5 = 0 : SERVICE ROW Y = 0
                      * TGS(7:6) = 00 : 40 CHAR/ROW MODE, LONG CHAR CODE (3 BYTES)
                      *

A 100C BD   10DB   A        JSR    BUSY
A 100F 86   00     A        LDA    #$00      LOAD VALUE INTO R1
A 1011 B7   F421   A        STA    R1
A 1014 86   81     A        LDA    #$81      "IND" COMMAND TO LOAD TGS (r=1)
A 1016 B7   F428   A        STA    RO+8      LOAD AND EXECUTE COMMAND.
```

```
PAGE  002  EF40     .SA:0

00058                           *
00059                           * MAT REGISTER INITIALIZATION :
00060                           * MAT(2:0) = 100 : MARGIN COLOR = BLUE
00061                           * MAT3 = 1 : I SIGNAL IS HIGH DURING MARGIN PERIOD
00062                           * MAT(5:4) = 00 : FIXED COMPLEMENTED CURSOR
00063                           * MAT6 = 1 : CURSOR DISPLAY ENABLED
00064                           * MAT7 = 0 : NO ZOOM MODE
00065                           *

00067A 1019 BD    10DB     A          JSR    BUSY
00068A 101C 86    4C       A          LDA    #$4C      LOAD VALUE INTO R1
00069A 101E B7    F421     A          STA    R1
00070A 1021 86    82       A          LDA    #$82      "IND" COMMAND TO LOAD MAT (r=2)
00071A 1023 B7    F428     A          STA    R0+8      LOAD AND EXECUTE COMMAND.

00073                           *
00074                           * PAT REGISTER INITIALIZATION :
00075                           * PAT0 = 1 : SERVICE ROW ENABLED
00076                           * PAT1 = 1 : UPPER BULK ENABLED
00077                           * PAT2 = 1 : LOWER BULK ENABLED
00078                           * PAT3 = 1 : CONCEAL ENABLED
00079                           * PAT(5:4) = 11 : I SIGNAL IS HIGH DURING THE
00080                           *                 ACTIVE DISPLAYED AREA.
00081                           * PAT6 = 1 : FLASHING ENABLED
00082                           * PAT7 = 0 : 40 CHAR/ROW MODE, LONG CODE
00083                           *

00085A 1026 BD    10DB     A          JSR    BUSY
00086A 1029 86    7F       A          LDA    #$7F      LOAD VALUE INTO R1
00087A 102B B7    F421     A          STA    R1
00088A 102E 86    83       A          LDA    #$83      "IND" COMMAND TO LOAD PAT (r=3)
00089A 1030 B7    F428     A          STA    R0+8      LOAD AND EXECUTE COMMAND.

00091                           *
00092                           * DOR REGISTER INITIALIZATION :
00093                           * DOR(3:0) = 0011 : ALPHA UDS SLICES IN BLOCK 3
00094                           * DOR(6:4) =  001 : SEMIGRAPHIC UDS SLICES IN BLOCKS 2 AND 3
00095                           * DOR 1 = 0 : QUADRICHROME SLICES FROM BLOCK 0
00096                           *

00098A 1033 BD    10DB     A          JSR    BUSY
00099A 1036 86    13       A          LDA    #$13      LOAD VALUE INTO R1
00100A 1038 B7    F421     A          STA    R1
00101A 103B 86    84       A          LDA    #$84      "IND" COMMAND TO LOAD DOR (r=4)
00102A 103D B7    F428     A          STA    R0+8      LOAD AND EXECUTE COMMAND.
```

```
PAGE  003  EF40    .SA:0

00104                            *
00105                            * ROR REGISTER INITIALIZATION :
00106                            * ROR(4:0) = 01000 : ORIGIN ROW = 8
00107                            * ROR(7:5) =   000 : DISPLAYED PAGE MEMORY STARTS FROM BLOCK 0
00108                            *

00110A 1040 BD   10DB     A          JSR     BUSY
00111A 1043 86   08       A          LDA     #$08      LOAD VALUE INTO R1
00112A 1045 B7   F421     A          STA     R1
00113A 1048 86   87       A          LDA     #$87      "IND" COMMAND TO LOAD ROR (r=7)
00114A 104A B7   F428     A          STA     R0+8      LOAD AND EXECUTE COMMAND.

00116                            *
00117                            * CLEAR PAGE MEMORY WITH ALPHANUMERIC SPACES
00118                            * FOREGROUND AND BACKGROUND COLORS = BLACK
00119                            *
00120A 104D 86   20       A          LDA     #$20
00121A 104F 8E   0000     A          LDX     #$0000    CHAR CODE BYTES B & A
00122A 1052 BD   10E1     A          JSR     MPFILL

00124                            * STORE SLICES FOR THE 4 CHARACTERS OF THE THOMSON LOGO.
00125                            * CHARACTER CODE C BYTES ARE : $00,$01,$02,$03

00127A 1055 86   03       A          LDA     #$03      BLOCK NUMBER Z(3:0)
00128A 1057 C6   00       A          LDB     #$00      INITIAL CHAR CODE C BYTE
00129A 1059 ED   C3       A          STD     ,--U      SAVE ACC. A & B INTO U STACK
00130A 105B 8E   1167     A          LDX     #CAR1     SLICE BUFFER ADDRESS

00132A 105E EC   C4       A ET1      LDD     0,U       GET ARGUMENTS FOR WRSLAL
00133A 1060 C1   04       A          CMPB    #$04      SLICES LOADED FOR 4 CHAR ?
00134A 1062 27   07   106B           BEQ     ET2       YES, BRANCH
00135A 1064 BD   1149     A          JSR     WRSLAL    NO, LOAD TEN SLICES
00136A 1067 6C   41       A          INC     1,U       INCREMENT CHAR CODE C BYTE
00137A 1069 20   F3   105E           BRA     ET1

00139A 106B 33   42       A ET2      LEAU    2,U       UPDATE U POINTER

00141                            * WRITE THE 4 UDS CHAR CODES INTO PAGE MEMORY.
00142                            * BACKGROUND = BLACK, FOREGROUND = WHITE : A BYTE = $70

00144A 106D BD   10DB     A          JSR     BUSY
00145A 1070 86   01       A          LDA     #$01      LOAD "KRF" COMMAND WITH CURSOR INCREM.
00146A 1072 B7   F420     A          STA     R0        NO EXECUTION !

00148A 1075 86   26       A          LDA     #38       INIT MAIN POINTER TO COLUMN 38, ON THE FI
00149A 1077 B7   F427     A          STA     R7        ROW AFTER SERVICE ROW
00150A 107A 86   08       A          LDA     #8
00151A 107C B7   F426     A          STA     R6
```

```
PAGE  004  EF40    .SA:0

00153A 107F 86    80      A         LDA    #$80     STORE CHAR CODE B BYTE INTO R2
00154A 1081 B7    F422    A         STA    R2
00155A 1084 86    70      A         LDA    #$70     CHAR CODE A BYTE INTO R3
00156A 1086 B7    F423    A         STA    R3

00158A 1089 86    00      A         LDA    #$00     WRITE THE UPPER LEFT CHAR
00159A 108B B7    F429    A         STA    R1+8
00160A 108E BD    10DB    A         JSR    BUSY
00161A 1091 4C                      INCA            WRITE THE UPPER RIGHT CHAR
00162A 1092 B7    F429    A         STA    R1+8
00163A 1095 BD    10DB    A         JSR    BUSY

00165A 1098 86    26      A         LDA    #38      INIT MAIN POINTER FOR THE 2 LOWER CHAR
00166A 109A B7    F427    A         STA    R7
00167A 109D 86    09      A         LDA    #9
00168A 109F B7    F426    A         STA    R6       Y=9

00170A 10A2 86    02      A         LDA    #$02     WRITE THE 2 LOWER CHAR
00171A 10A4 B7    F429    A         STA    R1+8
00172A 10A7 BD    10DB    A         JSR    BUSY
00173A 10AA 4C                      INCA
00174A 10AB B7    F429    A         STA    R1+8

00176                     * LOAD THE 10 SLICES FOR THE QUADRICHROME CHARACTER

00178A 10AE 86    03      A         LDA    #$03     BLOCK NUMBER Z(3:0)
00179A 10B0 C6    4B      A         LDB    #$4B     CHAR CODE C BYTE
00180A 10B2 8E    118F    A         LDX    #QUADRI  SLICE BUFFER ADDRESS
00181A 10B5 BD    1149    A         JSR    WRSLAL

00183                     * WRITE THE QUADRICHROME CHAR CODE INTO PAGE MEMORY
00184                     * PALETTE = RED-BLUE-CYAN-WHITE : A BYTE = $D2
00185                     * QUADRICHROME SET Q3, HIGH RESOLUTION (R=0) : B BYTE = $D8
00186                     * C BYTE = $4B

00188A 10B8 BD    10DB    A         JSR    BUSY
00189A 10BB 86    14      A         LDA    #20      INIT MAIN POINTER : X=20
00190A 10BD B7    F427    A         STA    R7
00191A 10C0 86    14      A         LDA    #20      Y=20
00192A 10C2 B7    F426    A         STA    R6

00194A 10C5 86    01      A         LDA    #$01
00195A 10C7 B7    F420    A         STA    R0       LOAD "KRF" COMMAND
00196A 10CA 86    4B      A         LDA    #$4B     LOAD CHAR CODE C BYTE INTO R1
00197A 10CC B7    F421    A         STA    R1
00198A 10CF 86    D8      A         LDA    #$D8     CHAR CODE B BYTE INTO R2
00199A 10D1 B7    F422    A         STA    R2
00200A 10D4 86    D2      A         LDA    #$D2     CHAR CODE A BYTE INTO R3 AND
00201A 10D6 B7    F42B    A         STA    R3+8     EXECUTE TRANSFER COMMAND

00203A 10D9 20    FE      10D9 HERE BRA    HERE
```

```
PAGE  005  EF40    .SA:0

00205                          *
00206                          * BUSY : TEST BUSY STATE IN STATUS REGISTER BIT 7.
00207                          *

00209          10DB     A BUSY   EQU    *
00210A 10DB 7D F420     A        TST    R0
00211A 10DE 2B FB   10DB         BMI    BUSY      LOOP IF BIT 7 = 1
00212A 10E0 39                   RTS

00214                          *
00215                          * MPFILL : FILL THE 3-BLOCK PAGE MEMORY STARTING FROM BLOCK 0
00216                          *          WITH THE SAME LONG CHARACTER CODE
00217                          *          ENTRY : THE 1RST BLOCK IS FILLED WITH ACC. A CONTENTS
00218                          *                  THE 2ND BLOCK WITH X REG. (MSB) CONTENTS
00219                          *                  THE 3RD BLOCK WITH X REG. (LSB) CONTENTS.
00220                          *

00222          10E1     A MPFILL EQU    *

00224A 10E1 BD 10DB     A        JSR    BUSY      TEST BUSY STATUS
00225A 10E4 B7 F421     A        STA    R1        STORE CHAR CODE INTO R1,R2,R3
00226A 10E7 BF F422     A        STX    R2

00228A 10EA 4F                   CLRA             INIT MAIN POINTER TO THE BEGINNING
00229A 10EB B7 F426     A        STA    R6        OF THE SERVICE ROW : R6 = R7 = 0.
00230A 10EE B7 F427     A        STA    R7

00232A 10F1 86 05       A        LDA    #$05      LOAD AND EXECUTE "CLF" COMMAND
00233A 10F3 B7 F428     A        STA    R0+8

00235A 10F6 8E 07D0     A        LDX    #2000
00236A 10F9 30 1F       A FILL30 LEAX   -1,X      WAIT ABOUT 15 MILLISECONDS
00237A 10FB 26 FC   10F9         BNE    FILL30

00239A 10FD 86 91       A        LDA    #$91      EXECUTE A "NOP" COMMAND
00240A 10FF B7 F428     A        STA    R0+8      TO ABORT "CLF"

00242A 1102 39                   RTS
```

```
PAGE  006  EF40     .SA:0

00244                       *
00245                       * AXPNT : AUXILIARY POINTER SET SUBROUTINE
00246                       * ENTRY : ACC.A = 0-0-0-0-Z3-Z2-Z1-Z0
00247                       *         ACC.B = 0-C6-C5-C4-C3-C2-C1-C0, WHERE C(0:6)
00248                       *                IS BYTE C OF CHAR. CODE
00249                       * EXIT : R4 = YA = 0-0-Z2-C6-C5-C4-C3-C2
00250                       *        R5 = XA = Z0-Z1-0-0-0-0-C1-C0
00251                       *         R6(6)=YP(6)=Z3
00252                       * OPERATION : TEMPORARY STORAGE :
00253                       *             M(0,S) = Z0-Z1-0-0-0-0-0-0
00254                       *             M(1,S) - 0-0-Z2-0-0-0-0-0
00255                       *             M(2,S) = 0-0-0-0-Z3-Z2-Z1-Z0
00256                       *             M(3,S) = 0-C6-C5-C4-C3-C2-C1-C0
00257                       *
00258              1103   A AXPNT  EQU    *

00260A 1103 32     7C     A        LEAS   -4,S     RESERVE 4 BYTE TEMPORARY STORAGE
00261A 1105 ED     62     A        STD    2,S      SAVE ARGUMENT A & B.

00263A 1107 5F                     CLRB
00264A 1108 46                     RORA
00265A 1109 46                     RORA
00266A 110A 46                     RORA            CY=Z2,A7=Z1,A6=Z0.
00267A 110B 56                     RORB            B7=Z2
00268A 110C 49                     ROLA
00269A 110D 56                     RORB            B7=Z1,B6=Z2
00270A 110E 49                     ROLA
00271A 110F 56                     RORB            B7=Z0,B6=Z1,B5=Z2
00272A 1110 1F     98     A        TFR    B,A      DUPLICATE RESULT INTO ACC.A

00274A 1112 C4     20     A        ANDB   #$20     B = 0-0-Z2-0-0-0-0-0
00275A 1114 84     C0     A        ANDA   #$C0     A = Z0-Z1-0-0-0-0-0-0
00276A 1116 ED     E4     A        STD    0,S      SAVE A & B

00278A 1118 BD     10D8   A        JSR    BUSY
00279A 111B E6     63     A        LDB    3,S      RESTORE INITIAL ARGUMENT
00280A 111D C4     03     A        ANDB   #$03     KEEP ONLY THE 2 LSB
00281A 111F EA     E4     A        ORB    0,S      B = Z0-Z1-0-0-0-0-C1-C0
00282A 1121 F7     F425   A        STB    R5       STORE INTO R5=XA
00283A 1124 E6     63     A        LDB    3,S
00284A 1126 54                     LSRB
00285A 1127 54                     LSRB            B = 0-0-0-C6-C5-C4-C3-C2
00286A 1128 EA     61     A        ORB    1,S      B = 0-0-Z2-C6-C5-C4-C3-C2
00287A 112A F7     F424   A        STB    R4       STORE INTO R4 = YA
```

```
PAGE  007  EF40    .SA:0

00289A 112D A6   62      A        LDA    2,S       RESTORE Z3-Z0 ARGUMENT
00290A 112F 84   08      A        ANDA   #$08      TEST Z3
00291A 1131 27   0B   113E        BEQ    AXPNT5
00292A 1133 B6   F426    A        LDA    YP        Z3=1 : YP(6)=1.
00293A 1136 8A   40      A        ORA    #$40
00294A 1138 B7   F426    A        STA    YP

00296A 113B 32   64      A        LEAS   4,S       UPDATE STACK POINTER
00297A 113D 39                    RTS

00299A 113E B6   F426    A AXPNT5 LDA    YP        Z3=0 : YP(6)=0.
00300A 1141 84   BF      A        ANDA   #$BF
00301A 1143 B7   F426    A        STA    YP

00303A 1146 32   64      A        LEAS   4,S       UPDATE STACK POINTER
00304A 1148 39                    RTS
00305                     *
00306                     * WRSLAL : WRITE 10 UDS SLICES.
00307                     * ENTRY : ACC.A = 0-0-0-0-Z3-Z2-Z1-Z0, WHERE Z(3:0) IS
00308                     *         BASE ADDRESS FOR UDS SLICES.
00309                     *         ACC.B = 0-C6-C5-C4-C3-C2-C1-C0, WHERE C(0:6) IS
00310                     *         BYTE C OF CHAR CODE
00311                     *         X POINTS TO THE SLICE BUFFER.
00312                     * EXIT :  A & B DESTROYED
00313                     *         X = X + 10.
00314                     *
00315                     *         AUXILIARY POINTER IS USED : BIT 2 = p OF
00316                     *         "BYTE LOAD" COMMAND =1

00318A 1149 BD   1103    A WRSLAL JSR    AXPNT     SET AUXILIARY POINTER.

00320A 114C 86   34      A        LDA    #$34      "BYTE WRITE COMMAND "
00321A 114E B7   F420    A        STA    R0        STORE COMMAND WITHOUT EXEC.
00322A 1151 C6   0A      A        LDB    #10       INIT LOOP COUNTER FOR 10 SLICES.

00324A 1153 A6   80      A WRSLA1 LDA    0,X+      STORE A SLICE AND EXECUTE
00325A 1155 B7   F429    A        STA    R1+8      TRANSFER INTO MEMORY
00326A 1158 BD   10DB    A        JSR    BUSY

00328A 115B 86   04      A        LDA    #$04      INC. SLICE CNTER = R5(5:2)
00329A 115D BB   F425    A        ADDA   R5
00330A 1160 B7   F425    A        STA    R5

00332A 1163 5A                    DECB             DEC. LOOP COUNTER
00333A 1164 26   ED   1153        BNE    WRSLA1

00335A 1166 39                    RTS
```

```
PAGE  008  EF40     .SA:0

00337                        *
00338                        * SLICE VALUES FOR UDS CHARACTERS OF FIGURE 15
00339                        *
00340A 1167        20      A CAR1    FCB     $20,$38,$3C,$3E,$3F,$3F,$1F,$1F,$0F,$0F
00341A 1171        04      A CAR2    FCB     $04,$1C,$3C,$7C,$FC,$FC,$F8,$F8,$F0,$F0
00342A 117B        07      A CAR3    FCB     $07,$C7,$E3,$F3,$F9,$FC,$FC,$F8,$E0,$80
00343A 1185        E0      A CAR4    FCB     $E0,$E3,$C7,$CF,$9F,$3F,$3F,$1F,$07,$01
00344                        *
00345                        * SLICE VALUES FOR QUADRICHROME CHARACTER (FIGURE 16)
00346                        *
00347A 118F        9C      A QUADRI FCB     $9C,$5A,$A3,$6A,$A9,$BE,$92,$EB,$29,$B6

00349                                END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000
```

PROGRAMMING THE EF9345 IN 80 CHAR/ROW MODE

CHARACTER CODE (figures 20 and 21)

In 80 char/row mode, the screen is made of 25 or 21 rows of 80 characters.

Each character is displayed in a 6 pixels by 10 lines window, which is associated with a character code in a page memory.

For a page, one of two character code formats must be selected :

- Long codes (12 bits), which consist of a C byte and an attribute A nibble.
- Short codes (8 bits), which consist of only a C byte (see figure 20).

With short codes, the C byte selects one of the 128 internal alphanumeric characters (G_0 set), and characters are displayed without attributes.

Long code format provides an additional 1024 mosaic character set and four attributes : D (color select), N (negative), U (underline) and F (flash). For each character, the foreground/background colors and the insert attribute are selected by bits D and N from the values programmed in DOR and MAT registers.

PAGE MEMORY

With long character code format, a page memory consists of three 1 Kbyte blocks. The same rules as in 40 char/row mode apply to page memory selection. The first (resp. second) block holds the C bytes of the characters in even (resp. odd) position on the rows. Every two consecutive characters have their A nibble concatened to make a byte stored in the third block.

Short character codes are similarly packed in two consecutive blocks which hold only C bytes.

ACCESS TO CHARACTER CODE

KRL command performs long character code transfer between registers R1-R3 and the memory. R1 is used for C byte transfer and R3 for A nibble transfer. When loading a character code, the A nibble must be repeated in R3.

KRC command is similarly used for short character code access between R1 and the memory.

Both KRL and KRC commands use the Main Pointer (R6, R7) for memory addressing. With a page memory starting from block number Z(0:3), R6 holds the Y row number and Z3-Z2. As the character position on a row is given by X(0:5) and Z0, it must be transcoded to obtain the R7 value with Z0-Z1 in the most significant bits (see figure 22).

Figure 20 : 80 Char/Row Character Code.

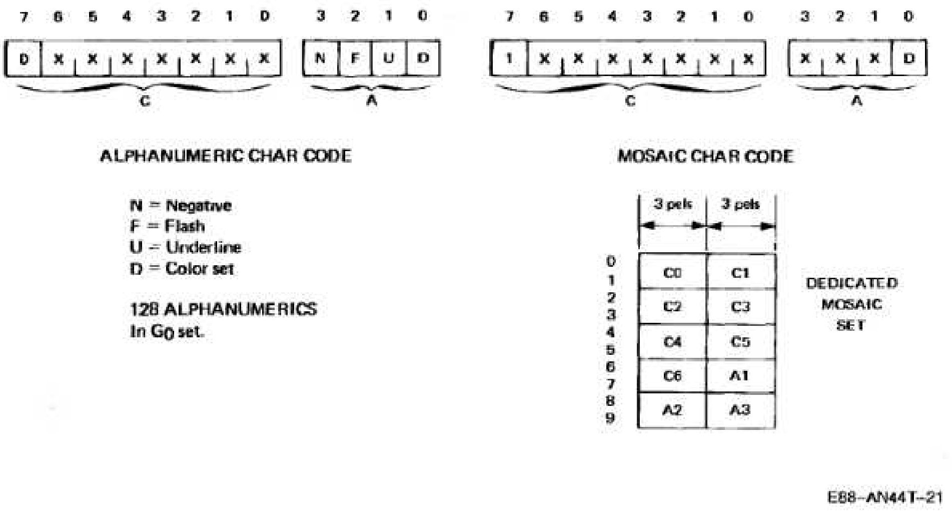
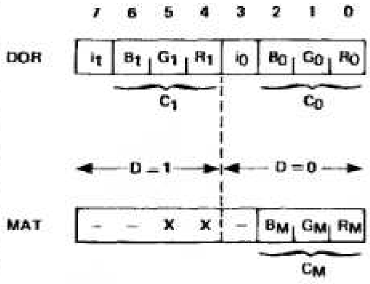


Figure 21 : Color Selection.

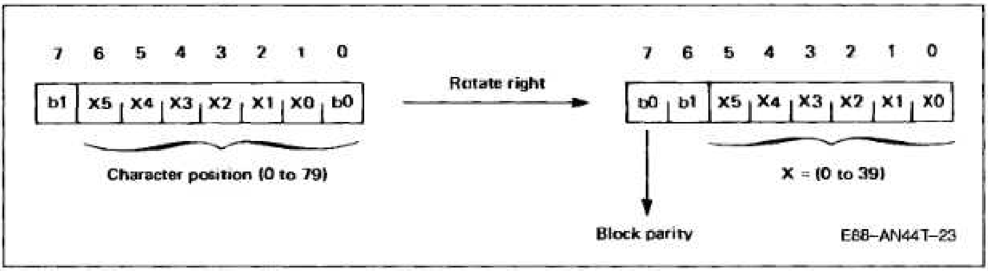

7 6 5 4 3 2 1 0
DOR
i_1 B_1 G_1 R_1 i_0 B_0 G_0 R_0
C_1
C_0
D = 1
D = 0
MAT
– – X X – B_M G_M R_M
C_M

D	N	BACKGND COLOR	FOREGND COLOR	i
0	0	C_M	C_0	i0
0	1	C_0	C_M	i0
1	0	C_M	C_1	i1
1	1	C_1	C_M	i1

The pixel shift frequency is f_{CLK} (12 MHz).

E88-AN44T-22

Figure 22 : Transcoding an Horizontal Screen Location into a R7 Pointer.



PROGRAMMING THE EF9345 IN 80 CHAR/ROW MODE

```
PAGE  001  EF80    .SA:0

00001                              OPT    LLE=110

00003                            *
00004                            * EF9345 PROGRAMMING EXAMPLE IN 80 CHAR/ROW
00005                            * THIS PROGRAM IS WRITTEN IN 6809 ASSEMBLER LANGUAGE.
00006                            * AFTER INDIRECT REGISTERS INITIALIZATION, TWO
00007                            * CHARACTER STRINGS ARE DISPLAYED AND A ROLL-UP
00008                            * OPERATION IS MADE.
00009                            *

00011                            * EF9345 REGISTER ADDRESS

00013              F420   A R0     EQU    $F420     COMMAND/STATUS REGISTER
00014              F421   A R1     EQU    R0+1      DATA REGISTERS
00015              F422   A R2     EQU    R0+2
00016              F423   A R3     EQU    R0+3
00017              F424   A R4     EQU    R0+4      AUXILIARY POINTER (Y)
00018              F425   A R5     EQU    R0+5      AUXILIARY POINTER (X)
00019              F426   A R6     EQU    R0+6      MAIN POINTER (Y)
00020              F427   A R7     EQU    R0+7      MAIN POINTER (X)

00022              F425   A XA     EQU    R5
00023              F424   A YA     EQU    R4
00024              F427   A XP     EQU    R7
00025              F426   A YP     EQU    R6

00027              4000   A STACK  EQU    $4000
00028              3F80   A STACKU EQU    STACK-128

00030A 1000                        ORG    $1000

00032              1000   A MAIN   EQU    *

00034A 1000 10CE 4000     A        LDS    #STACK    STACK INITIALIZATION
00035A 1004 CE   3F80     A        LDU    #STACKU

00037A 1007 86   91       A        LDA    #$91      LOAD AND EXECUTE A "NOP" COMMAND
00038A 1009 B7   F428     A        STA    R0+8      WITHOUT TESTING BUSY

00040                            *
00041                            * TGS REGISTER INITIALIZATION :
00042                            * TGS0 = 0 : 625 LINES (50 HZ)
00043                            * TGS1 = 0 : NOT INTERLACED
00044                            * TGS2 = 0 : HORIZONTAL RESYNC. DISABLED
00045                            * TGS3 = 0 : VERTICAL RESYNC. DISABLED
00046                            * TGS4 = 0 : HORIZONTAL SYNC. ON HVS/HS PIN AND
00047                            *            VERTICAL SYNC. ON PC/VS PIN
00048                            * TGS5 = 0 : SERVICE ROW Y = 0
00049                            * TGS(7:6) = 11 : 80 CHAR/ROW MODE, LONG CHAR CODE (12 BITS)
00050                            *

00052A 100C BD   1002     A        JSR    BUSY
00053A 100F 86   C0       A        LDA    #$C0      LOAD VALUE INTO R1
00054A 1011 B7   F421     A        STA    R1
00055A 1014 86   81       A        LDA    #$81      "IND" COMMAND TO LOAD TGS (r=1)
00056A 1016 B7   F428     A        STA    R0+8      LOAD AND EXECUTE COMMAND.
```

```
PAGE  002  EF80    .SA:0

00058                            *
00059                            * MAT REGISTER INITIALIZATION :
00060                            * MAT(2:0) = 100 : MARGIN COLOR = BLUE
00061                            * MAT3 = 1 : I SIGNAL IS HIGH DURING MARGIN PERIOD
00062                            * MAT(5:4) = 00 : FIXED COMPLEMENTED CURSOR
00063                            * MAT6 = 1 : CURSOR DISPLAY ENABLED
00064                            * MAT7 - 0 : NO ZOOM MODE
00065                            *

00067A 1019 BD    10D2     A          JSR     BUSY
00068A 101C 86    4C       A          LDA     #$4C      LOAD VALUE INTO R1
00069A 101E B7    F421     A          STA     R1
00070A 1021 86    82       A          LDA     #$82      "IND" COMMAND TO LOAD MAT (r=2)
00071A 1023 B7    F428     A          STA     R0+8      LOAD AND EXECUTE COMMAND.

00073                            *
00074                            * PAT REGISTER INITIALIZATION :
00075                            * PAT0 = 1 : SERVICE ROW ENABLED
00076                            * PAT1 = 1 : UPPER BULK ENABLED
00077                            * PAT2 = 1 : LOWER BULK ENABLED
00078                            * PAT3 = 1 : CONCEAL ENABLED
00079                            * PAT(5:4) = 11 : I SIGNAL IS HIGH DURING THE
00080                            *                 ACTIVE DISPLAYED AREA.
00081                            * PAT6 = 1 : FLASHING ENABLED
00082                            * PAT7 = 0 : 80 CHAR/ROW MODE, LONG CODE
00083                            *

00085A 1026 BD    10D2     A          JSR     BUSY
00086A 1029 86    7F       A          LDA     #$7F      LOAD VALUE INTO R1
00087A 102B B7    F421     A          STA     R1
00088A 102E 86    83       A          LDA     #$83      "IND" COMMAND TO LOAD PAT (r=3)
00089A 1030 B7    F428     A          STA     R0+8      LOAD AND EXECUTE COMMAND.

00091                            *
00092                            * DOR REGISTER INITIALIZATION :
00093                            * DOR(3:0) = 1111 : COLOR C0 = WHITE
00094                            * DOR(7:4) = 1000 : COLOR C1 = BLACK
00095                            * INSERT ATTRIBUTE i IS SET FOR ANY CHARACTER.
00096                            *

00098A 1033 BD    10D2     A          JSR     BUSY
00099A 1036 86    8F       A          LDA     #$8F      LOAD VALUE INTO R1
00100A 1038 B7    F421     A          STA     R1
00101A 103B 86    84       A          LDA     #$84      "IND" COMMAND TO LOAD DOR (r=4)
00102A 103D B7    F428     A          STA     R0+8      LOAD AND EXECUTE COMMAND.
```

```
PAGE  003  EF80     .SA:0

00104                         *
00105                         * ROR REGISTER INITIALIZATION :
00106                         * ROR(4:0) = 01000 : ORIGIN ROW = 8
00107                         * ROR(7:5) =   001 : DISPLAYED PAGE MEMORY STARTS FROM BLOCK 0
00108                         *

00110A 1040 BD   10D2    A          JSR    BUSY
00111A 1043 86   28      A          LDA    #$28     LOAD VALUE INTO R1
00112A 1045 B7   F421    A          STA    R1
00113A 1048 86   87      A          LDA    #$87     "IND" COMMAND TO LOAD ROR (r=7)
00114A 104A B7   F428    A          STA    R0+8     LOAD AND EXECUTE COMMAND.

00116                         *
00117                         * CLEAR PAGE MEMORY WITH ALPHANUMERIC SPACES
00118                         * BACKGROUND COLOR = CM (MARGIN COLOR)
00119                         *
00120A 104D 86   20      A          LDA    #$20     C BYTE FOR EVEN POSITION CHAR.
00121A 104F 8E   2000    A          LDX    #$2000   C BYTE FOR ODD POSITION AND A NIBBLES
00122A 1052 C6   04      A          LDB    #4       PAGE MEMORY FIRST BLOCK NUMBER
00123A 1054 BD   10D8    A          JSR    MPFILL

00125                         * WRITE "ABCD..." WITH FLASH AND NEGATIVE ATTRIBUTES
00126                         * ATTRIBUTE BITS (D,N)=01 :
00127                         *    BACKGROUND COLOR = C0 DEFINED IN DOR
00128                         *    FOREGROUND COLOR = CM (MARGIN COLOR)

00130A 1057 BD   10D2    A          JSR    BUSY
00131A 105A 86   51      A          LDA    #$51     LOAD "KRL" COMMAND WITH
00132A 105C B7   F420    A          STA    R0       CURSOR INCREMENTATION

00134A 105F 86   28      A          LDA    #$28     INIT MAIN POINTER (CURSOR)
00135A 1061 B7   F426    A          STA    R6
00136A 1064 86   00      A          LDA    #$00
00137A 1066 B7   F427    A          STA    R7

00139A 1069 86   CC      A          LDA    #$CC     LOAD ATTRIBUTE NIBBLE (REPEATED
00140A 106B B7   F423    A          STA    R3       INTO R3).

00142A 106E C6   0A      A          LDB    #10      LOOP COUNTER FOR 10 CHARACTERS
00143A 1070 86   41      A          LDA    #'A      FIRST CHAR CODE C BYTE

00145A 1072 B7   F429    A LOOP     STA    R1+8     STORE C.C. C BYTE AND EXEC COMMAND
00146A 1075 4C                      INCA            INCREMENT C BYTE
00147A 1076 BD   10D2    A          JSR    BUSY
00148A 1079 5A                      DECB            DEC LOOP COUNTER
00149A 107A 26   F6    1072         BNE    LOOP
```

```
PAGE  004  EF80    .SA:0

00151                          * WRITE "KLM...." WITH UNDERLINING
00152                          * (D,N) = (0,0) : BACKGROUND COLOR = CM
00153                          *                 FOREGROUND COLOR = CO

00155A 107C 86   2A      A        LDA    #$2A     INIT CURSOR
00156A 107E B7   F426    A        STA    R6
00157A 1081 86   00      A        LDA    #$00
00158A 1083 B7   F427    A        STA    R7

00160A 1086 86   22      A        LDA    #$22     ATTRIBUTE NIBBLE INTO R3
00161A 1088 B7   F423    A        STA    R3

00163A 108B C6   0A      A        LDB    #10
00164A 108D 86   4B      A        LDA    #'K

00166A 108F B7   F429    A LOOP1  STA    R1+8
00167A 1092 4C                    INCA
00168A 1093 BD   10D2    A        JSR    BUSY
00169A 1096 5A                    DECB
00170A 1097 26   F6   108F        BNE    LOOP1

00172                          * ROLL-UP OPERATION EXAMPLE

00174A 1099 BD   10D2    A        JSR    BUSY

00176A 109C 86   8F      A        LDA    #$8F     EXECUTE "IND" COMMAND TO READ ROR REGISTE
00177A 109E B7   F428    A        STA    R0+8

00179A 10A1 BD   10D2    A        JSR    BUSY     COMMAND EXECUTED?
00180A 10A4 B6   F421    A        LDA    R1       READ RESULT FROM R1

00182A 10A7 C6   87      A        LDB    #$87     STORE "IND" COMMAND FOR LOADING ROR
00183A 10A9 F7   F420    A        STB    R0

00185            10AC    A LOOP3  EQU    *

00187A 10AC B7   F429    A        STA    R1+8     STORE VALUE TO BE LOADED INTO ROR
00188A 10AF BD   10D2    A        JSR    BUSY
00189A 10B2 BD   10C6    A        JSR    WAIT     TEMPO
00190A 10B5 4C                    INCA
00191A 10B6 34   02      A        PSHS   A
00192A 10B8 84   1F      A        ANDA   #$1F     YOR = ROR(4:0) = 31 ?
00193A 10BA 81   1F      A        CMPA   #31
00194A 10BC 35   02      A        PULS   A
00195A 10BE 26   EC   10AC        BNE    LOOP3

00197A 10C0 84   E0      A        ANDA   #$E0     IF YOR=31, SET YOR=8
00198A 10C2 8B   08      A        ADDA   #8
00199A 10C4 20   E6   10AC        BRA    LOOP3

00201            10C6    A WAIT   EQU    *
00202A 10C6 34   10      A        PSHS   X
00203A 10C8 8E   FFFF    A WAIT1  LDX    #$FFFF
00204A 10CB 30   1F      A WAIT2  LEAX   -1,X
00205A 10CD 26   FC   10CB        BNE    WAIT2
00206A 10CF 35   10      A        PULS   X
00207A 10D1 39                    RTS
```

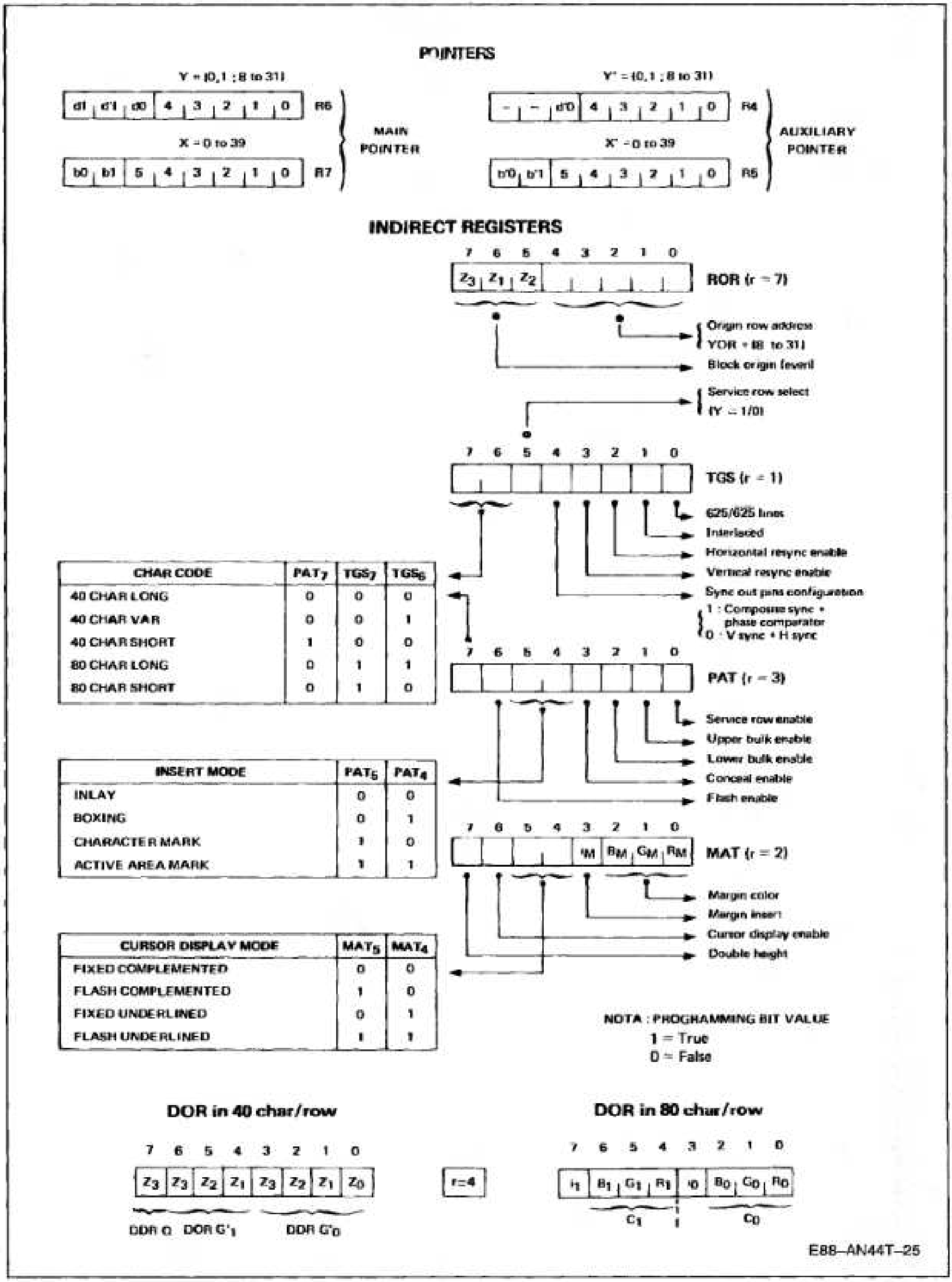
```
PAGE  005  EF80     .SA:0

00209                          *
00210                          * BUSY : TEST BUSY IN STATUS REGISTER R0(7)
00211                          *

00213              10D2      A BUSY   EQU    *
00214A 10D2 7D     F420      A        TST    R0
00215A 10D5 2B     FB    10D2         BMI    BUSY       LOOP IF BIT 7 = 1
00216A 10D7 39                        RTS

00218                          *
00219                          * MPFILL : FILL THE 3-BLOCK PAGE MEMORY STARTING FROM BLOCK 0
00220                          *          WITH THE SAME LONG CHARACTER CODE
00221                          *          ENTRY : THE 1RST BLOCK IS FILLED WITH ACC. A CONTENTS
00222                          *                  THE 2ND BLOCK WITH X REG. (MSB) CONTENTS
00223                          *                  THE 3RD BLOCK WITH X REG. (LSB) CONTENTS.
00224                          *

00226              10D8      A MPFILL EQU    *

00228A 10D8 BD     10D2      A        JSR    BUSY       TEST BUSY STATUS
00229A 10DB B7     F421      A        STA    R1         STORE CHAR CODE INTO R1,R2,R3
00230A 10DE BF     F422      A        STX    R2

00232A 10E1 4F                        CLRA              INIT MAIN POINTER TO THE BEGINNING
00233A 10E2 B7     F426      A        STA    R6         OF THE SERVICE ROW : R6 = R7 = 0.
00234A 10E5 B7     F427      A        STA    R7

00236A 10E8 86     05        A        LDA    #$05       LOAD AND EXECUTE "CLF" COMMAND
00237A 10EA B7     F428      A        STA    R0+8

00239A 10ED 8E     07D0      A        LDX    #2000
00240A 10F0 30     1F        A FILL30 LEAX   -1,X       WAIT ABOUT 15 MILLISECONDS
00241A 10F2 26     FC    10F0         BNE    FILL30

00243A 10F4 86     91        A        LDA    #$91       EXECUTE A "NOP" COMMAND
00244A 10F6 B7     F428      A        STA    R0+8       TO ABORT "CLF"

00246A 10F9 39                        RTS

00248                                 END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000
```

COMMAND TABLE

Type	Memo	Code				Parameter				Status				Arguments							Execution Time (1)	
		7	6	5	4	3	2	1	0	AI	LX_m	LX_1	$R1_7$	R1	R2	R3	R4	R5	R6	R7	Write	Read
Indirect	IND	1	0	0	0	$R/\overline{W}$	–	r	–	0	0	0	0	D	–	–	–	–	MP		2	3.5
40 Characters – 24 Bits	KRF	0	0	0	0	$R/\overline{W}$	0	0	1	X	X	0	0	C	B	A	–	–	MP		4	7.5
40 Characters – 16 Bits	KRG	0	0	0	0	$R/\overline{W}$	0	1	1	X	X	0	0	A'	B'	W	–	–	MP		5.5	7.5
80 Characters – 8 Bits	KRC	0	1	0	0	$R/\overline{W}$	0	0	1	X	X	0	0	C	–	–	–	–	MP		9	9.5
80 Characters – 12 Bits	KRL	0	1	0	1	$R/\overline{W}$	0	0	1	X	X	0	0	C	–	A	–	–	MP		12.5	11.5
40 Characters Variable	KRV	0	0	1	0	$R/\overline{W}$	0	0	1	X	X	X	X	C	B	A	–	XF	MP		(2) 3 + 3 + j	3.5 + 6 • j
Expansion	EXP	0	1	1	0	0	0	0	0	X	0	X	0	C	B	A	PW	XF	MP		(3) < 247	–
Compression	CMP	0	1	1	1	0	0	0	0	X	0	X	0	C	B	A	PW	XF	MP		(3) < 402	–
Expanded Characters	KRE	0	0	0	1	$R/\overline{W}$	0	0	1	X	X	0	0	C	B	A	PW	–	MP		4	7.5
Byte	OCT	0	0	1	1	$R/\overline{W}$	p	0	1	X	X	X	0	D	–	–	AP		MP		4	4.5
Move Buffer	MVB	1	1	0	1	s	$\overline{s}$	$\overline{a}$	a	0	0	0	0	W	–	–	AP		MP		(2) 2 + 4.n	–
Move Double Buffer	MVD	1	1	1	0	s	$\overline{s}$	$\overline{a}$	a	0	0	0	0	W	–	–	AP		MP		(2) 2 + 6.n	–
Move Triple Buffer	MVT	1	1	1	1	s	$\overline{s}$	$\overline{a}$	a	0	0	0	0	W	–	–	AP		MP		(2) 2 + 12.n	–
Clear Page (4) – 24 Bits	CLF	0	0	0	0	0	1	0	1	X	X	0	0	C	B	A	–	–	MP		< 4700 (1 K code)	–
Clear Page (4) – 16 Bits	CLG	0	0	0	0	0	1	1	1	X	X	0	0	A'	B'	W	–	–	MP		< 5800 (1 K code)	–
Vertical Sync Mask Set	VSM	1	0	0	1	1	0	0	1	0	0	0	0	–	–	–	–	–	–	–	1	–
Vertical Sync Mask Reset	VRM	1	0	0	1	0	1	0	1	–	–	–	–	–	–	–	–	–	–	–	1	–
Increment Y	INY	1	0	1	1	0	0	0	0	0	0	0	0	–	–	–	–	–	Y	–	TBD	–
No Operation	NOP	1	0	0	1	0	0	0	1	–	–	–	–	–	–	–	–	–	–	–	1	–

p	:	Pointer Select 1 : Auxiliary Pointer 0 : Main Pointer.	–	:	Not Affected
			W	:	Used as Working Register
			PW (ZW, YW)	:	Working Buffer
s, $\overline{s}$	:	Source Destination 0 1 : Source = MP ; Destination = AP 1 0 : Source = AP ; Destination = mP	X	:	Set or Reset
			XF	:	X File
			I	:	Pointer incrementation
$\overline{a}$, a	:	Stop Condition 0 1 : Stop at End of Buffer 1 0 : No Stop	D	:	Data
			MP	:	Main Pointer
			AP	:	Auxiliary Pointer.
r	:	Indirect Register Number			

(1) Unit : 12 clock periods (= 1 µs) without possible suspension.
(2) n : total number of words ≤ 40 ; j = 1 for long codes, j = 0 for short codes.
(3) Worst case (20 long codes + 20 short codes).
(4) These commands repeat KRF or KRG with Y incrementation when X overflows. When the last position is reached in a row, Y is incremented and the process starts again on the next row. These commands stop only with abort.

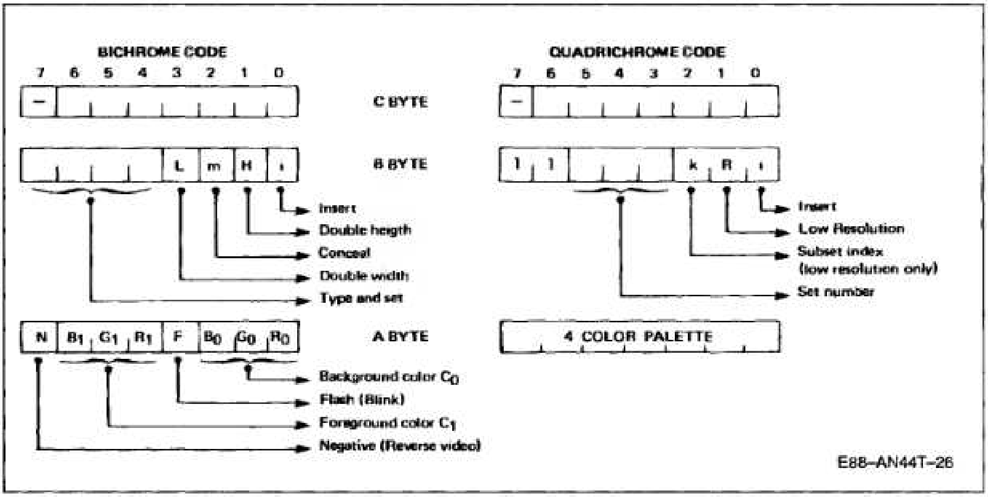


CHAR CODE	PAT_7	TGS_7	TGS_6
40 CHAR LONG	0	0	0
40 CHAR VAR	0	0	1
40 CHAR SHORT	1	0	0
80 CHAR LONG	0	1	1
80 CHAR SHORT	0	1	0

INSERT MODE	PAT_5	PAT_4
INLAY	0	0
BOXING	0	1
CHARACTER MARK	1	0
ACTIVE AREA MARK	1	1

CURSOR DISPLAY MODE	MAT_5	MAT_4
FIXED COMPLEMENTED	0	0
FLASH COMPLEMENTED	1	0
FIXED UNDERLINED	0	1
FLASH UNDERLINED	1	1

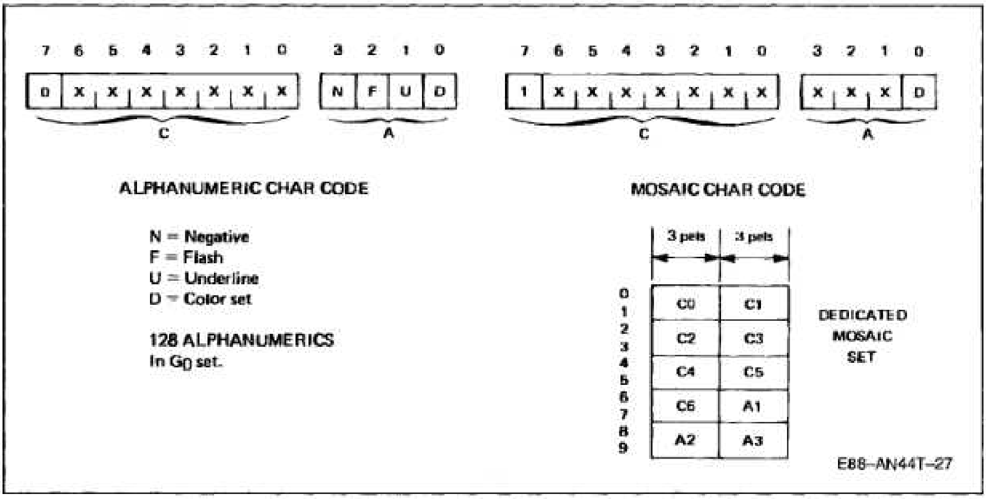
40 Char/Row Fixed Long Codes



Type and Set Code : B (4 : 7)				Number of Character Per Set	Set Name	Set Type		Cell Location
7	6	5	4	C (0 : 6)				
0	0	1 1	0 1	128 Standard Mosaics 32 Strokes	G_{10} G_{11}	SEMI-GR.	B I C H R O M E	ON-CHIP ROM
	0	0	U N D E R L I N E	128 Alphanumerics	G0	ALPHA		
	1	0 1		Accentued Lower Case Alpha	G20 G21			
1	0	1		100 Alpha UDS	G'0			EXTERNAL MEMORY
	0	1 1	1 0	100 Semi-graphic UDS 100 Semi-graphic UDS	G'10 G'11	SEMI-GR.		
	1	X	X	8 Sets of 100 Quadrichrome Character	Q0 to Q7	QUADRICHROME		

Nota : Programming bit value
1 = True
0 = False.

80 Char/Row Character Code



COLOR SELECTION

D	N	BACKGND COLOR	FOREGND COLOR	i
0	0	C_M	C_0	i0
0	1	C_0	C_M	i0
1	0	C_M	C_1	i1
1	1	C_1	C_M	i1

(C_0, C_1, i0, i1) : defined in DOR
C_M : margin color defined in MAT