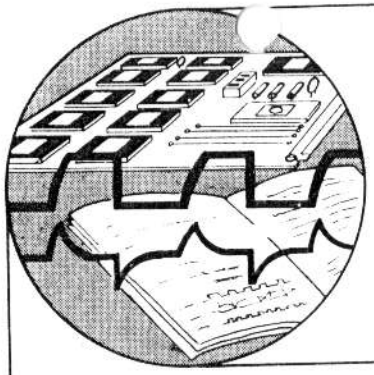
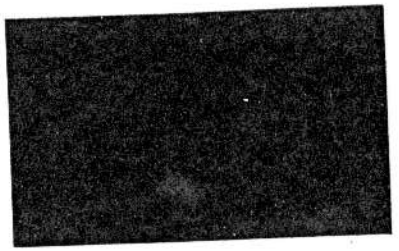


APPLICATION NOTE

MAY 1981

NA-002R2A



General application principles for EF9365 – EF9366

Philippe LAMBINET
Laboratoire d'applications

THOMSON-EFCIS
Integrated Circuits

INTRODUCTION

The aim of this document is to help EF9365 and EF9366 users in the implementation of their applications, either in black and white or in colour.

Will be successively described :

- An application example of EF9365,
- An application example of EF9366,
- A programmation example of EF9365,
- Some extensions to the preceding examples.

It is of course, required to know technical specifications of EF9365 and EF9366.

EF9365 APPLICATION

The following logical diagram and text describe an EF9365 application at its maximum definition (512 x 512 pixels) in black and white (one bit per pixel).

SCREEN MEMORY ORGANIZATION

Screen memory contains 262, 144 bits (512 x 512), therefore, 16 chips of 16 K bit dynamic RAM are used (4116 type), dispatched into two 8-chip-arrays.

The 7 address lines of memory devices are connected to the 7 DAD pins of EF9365.

These 7 lines bring out 14 multiplexed address bits.

Memory space addressable this way is 16 K bytes (one 8-chip array). Selection within both arrays is performed with output MSL3. Bit access within a byte is performed with the 3 output MSL0, MSL1, MSL2.

So actually, 18 address signals are obtained, which allow access to 2^{18} bit positions (262, 144).

GENERAL OPERATING PRINCIPLES

Microprocessor bus interface

The microprocessor bus runs at a 1 MHz frequency. The EF9365 takes 16 addresses inside the microprocessor peripheral space. So, 4 address bits are required as inputs A_0 , A_1 , A_2 , A_3 on EF9365.

The user may choose any address to set the EF9365 in the MPU space, by an appropriate decoding of the 12 MSB's of the address bus.

Chip validation is not directly made by a chip-select type pin. Input signal $\overline{E}$ actually validates EF9365 access. This signal is also the synchronizing clock input for all bus exchanges. $\overline{E}$ input is therefore the logical AND of address decoding and bus clock.

R/$\overline{W}$ signal (Read-Write) is provided directly by control bus and controls exchange direction through a bidirectional bus transceiver.

Screen memory addressing

Details about addressing signals provided by MSL0 to MSL3 and DAD0 to DAD6 outputs are described in the EF9365 specifications and supposed to be known.

Some general features and principles are pointed out in the following.

WRITE MODE ACCESS

Internal automata (characters and vector generators) need a bit level access to the screen memory. DAD's and MSL's give the address of the bit describing the state of the pixel pointed to by X and Y registers.

0 = pixel lighted, 1 = pixel off

Output signal $\overline{ALL}$ is then at a high level. It validates decoding of MSL0, MSL1 and MSL2.
The 4 MSL signals allow selection of one of 16 chips by enabling $\overline{RAS}$ and $\overline{CAS}$ strobing signals only for this chip.

DISPLAY AND REFRESH ACCESS

These accesses are collective ones ($\overline{ALL}$ is at a low level) that means 8 chips are selected at the same time ($\overline{RAS}$ and $\overline{CAS}$ are enabled for these 8 chips).

Difference between Display and Refresh is made by BLK output (0 = Display, 1 = Refresh).

During display period, BLK signal enables shift-register operation.

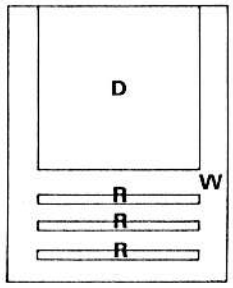
Loading of this shift-register is performed at the end of memory read cycle by the carry output from the 3-bit counter.

SHARING BETWEEN DIFFERENT CYCLES

During a 20 ms (50 Hz) period :

D = 64 cycles of 1.75 MHz clock x 256 lines
R = 64 cycles of 1.75 MHz clock x 12 lines
W = 20 ms - D - R

Time sharing obtained is W = 51 %
D = 46.8 %
R = 2.2 %



D : Display cycles
R : Refresh cycles
W : Write cycles

Refresh of entire dynamic RAM is performed in 256 accesses (128 for each array), that is 4 lines (64 accesses per line).

During display period, memory is entirely scanned and consequently refreshed.

During vertical blanking, EF9365 brings out 3 refresh periods (3 x 4 lines). That is largely enough to keep data valid (Vertical Blanking is only 3.6 ms long).

Video signal

Clock and strobe signals

All clock and strobe signals are provided by a crystal high-speed clock at 14 MHz frequency (HCK).

A 3-bit counter is used to address a bipolar PROM which brings out all 1.75 MHz signals : CK, $\overline{RAS}$, $\overline{CAS}$ and STR. STR signal is used to strobe $\overline{ALL}$, $\overline{DW}$, MSL0-3, BLK and DIN signals when they are all valid, and to keep them available during the whole memory cycle.

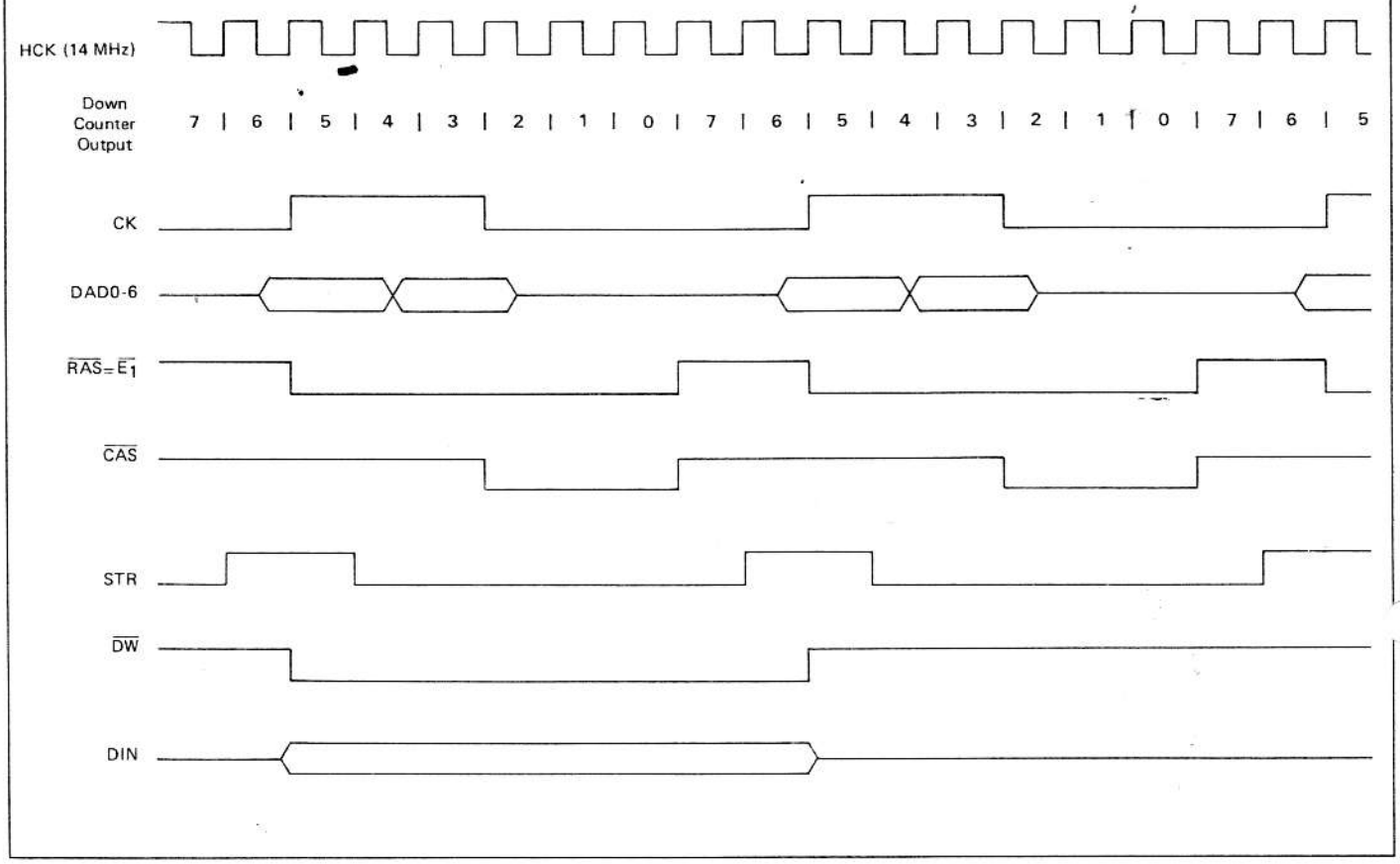
HCK is on an another hand used to shift data in the shift-register. Dot period in the video signal is then 71 ns (standard value).

The content of the 74S288 bipolar PROM is directly obtained from the timing diagram ; output signals from the PROM are latched by HCK and therefore take a one-clock-period delay in comparison with the address given by the 3-bit counter. This must be taken into account when programming the bipolar PROM.

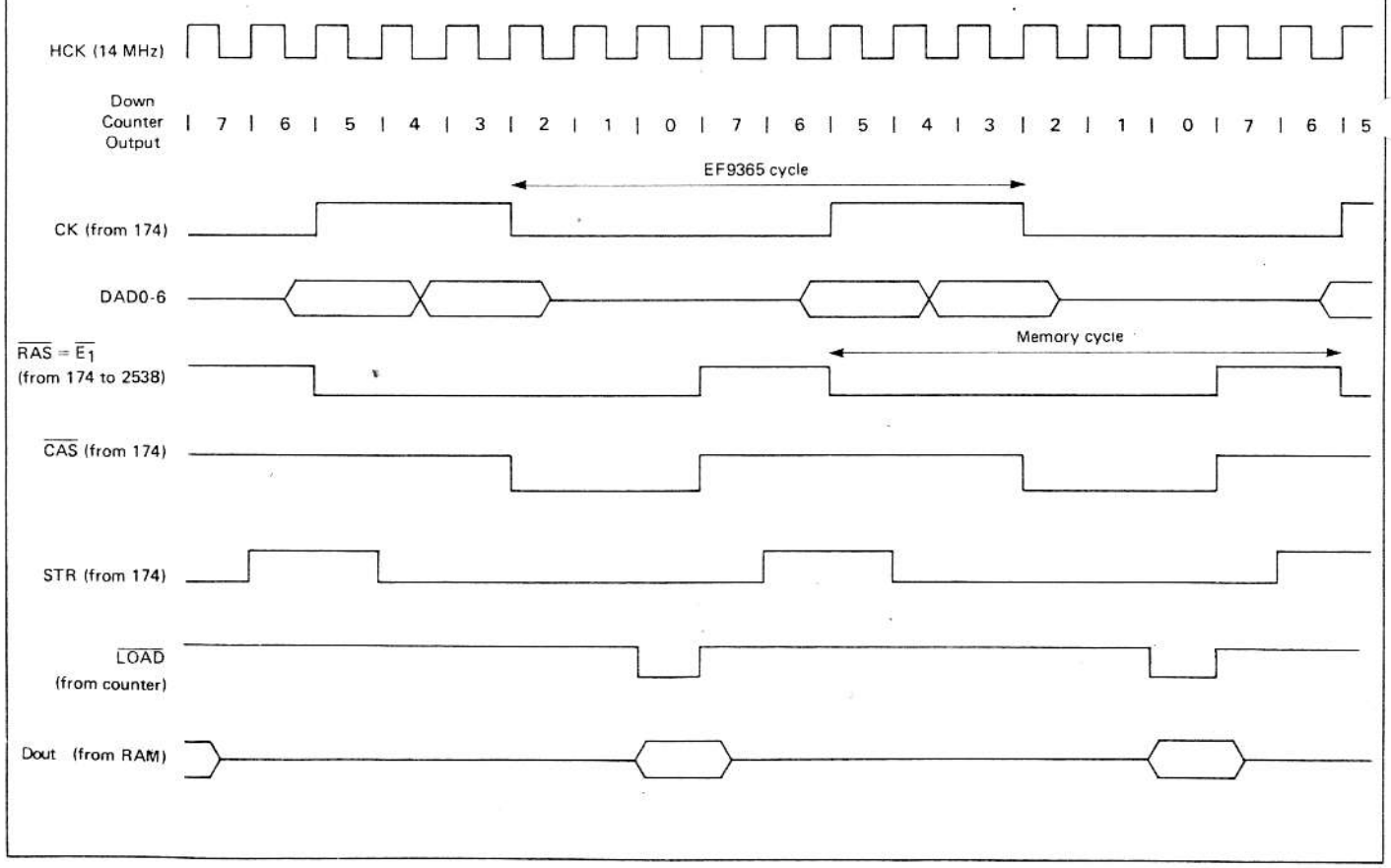
The PROM must then be programmed as follows :

Address				Data					This data is output from the 74LS174 during cycle :
C	B	A	Hexa	CK D4	$\overline{CAS}$ D3	$\overline{RAS}$ D2	STR D1	Hexa	
1	1	1	$7	0	1	1	1	$7	6
1	1	0	$6	1	1	0	1	$D	5
1	0	1	$5	1	1	0	0	$C	4
1	0	0	$4	1	1	0	0	$C	3
0	1	1	$3	0	0	0	0	$0	2
0	1	0	$2	0	0	0	0	$0	1
0	0	1	$1	0	0	0	0	$0	0
0	0	0	$0	0	1	1	0	$6	7

TIMING DIAGRAM FOR WRITE MEMORY CYCLES EF9365 - EF9366 ($R/\overline{W} = 0$) ($\overline{ALL} = 1$)



TIMING DIAGRAM FOR READ MEMORY CYCLES EF9365 - EF9366 ($R/\overline{W} = 1$) ($\overline{ALL} = 0$)



EF9366 APPLICATION

The following logical diagram and text describe an EF9366 application in color (3 memory arrays).

SCREEN MEMORY ORGANIZATION

Screen memory is divided into 3 arrays of 8 - 16 K bit-chips (4116 type).

Each array is the binary equivalent of display for each basic color, Red, Green or Blue.
By combination of possible values of each bit on these three arrays, 8 colors are possible for each dot of the screen.

Input Channels			Resulting Color
R	G	B	
0	0	0	WHITE
0	0	1	YELLOW
0	1	0	MAGENTA
0	1	1	RED
1	0	0	CYAN
1	0	1	GREEN
1	1	0	BLUE
1	1	1	BLACK

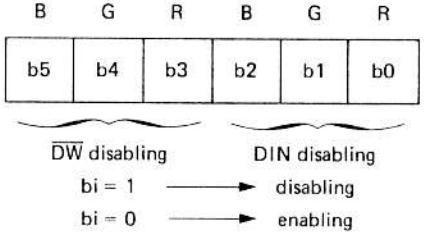
GENERAL OPERATING PRINCIPLES

Microprocessor bus interface

This interface is exactly the same as the EF9365's one. The only difference between this application and the EF9365 application is the color selection register connected to the MPU bus.

This register takes one address within the MPU space. Its function is to enable or disable writing in each memory array.

The MPU programs this register as explained bellow :



These bits are latched and only change during an updating of the register.

It is then possible to disable one or several arrays during one or several writing cycles.

Screen memory addressing

WRITE MODE ACCESS

The only difference between EF9366 and EF9365 use is the no-use of the MSL3 signal in EF9366 applications.

DISPLAY AND REFRESH ACCESSES

Idem EF9365.

SCANNING

Scanning is non-interlaced. All frames are identical and are 312 lines long.

Input frequency is no more 1.75 MHz, but is :

$$f_{ck} = 1.75 \times \frac{624}{625} = 1.7472 \text{ MHz}$$

(If an exact 50 Hz frequency is needed).

Video signals

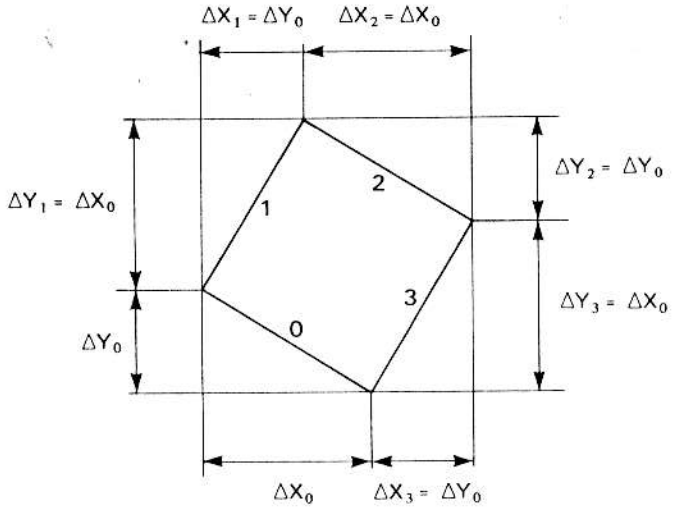
Video signals corresponding to the three R, G, B channels are output from 3 shift-registers.
Composite synchro is given by the EF9366.

Clock signals

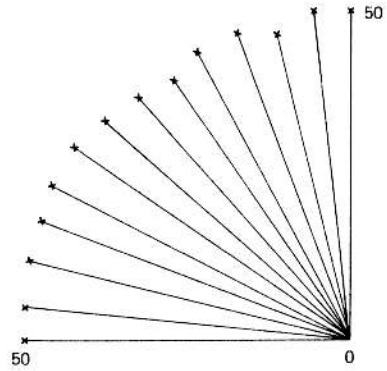
All these signals derive from the high-speed clock HCK. Generation of all clocks and strobing signals is performed with exactly the same method as previously indicated.

ANIMATION PRINCIPLES

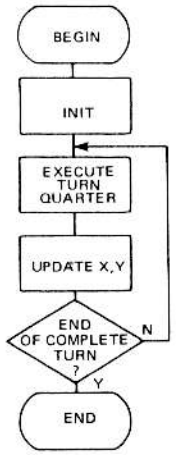
A graphic animation is a succession of writing and erasing operations, in order to simulate a motion.
A sufficient number of different positions of the object must be determined to give a good impression of continuity.
Given example is a square rotation. A square is easily described by one of its sides, that is for the EF9365 by one dot position (starting point) and projections ΔX and ΔY of one of its side.



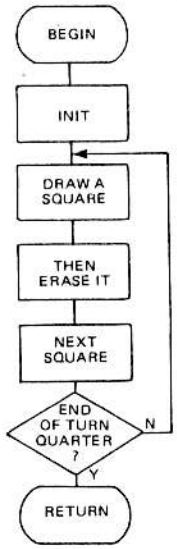
This diagram shows that projections of all sides are available from projections of only one side.
Successive values of ΔX and ΔY during the rotation are stored in a table. A quarter of a turn is executed that way ; three following quarters are executed in the same manner.



Successive positions of the first side of the square.
A table contains 14 values giving DELTAX and DELTAY.



PROGRAM STRUCTURE



TURN QUARTER SUBROUTINE

```
PAGE  001  EXAMP   .SA:0

00001                  ********************************************************
00002                  * EF9365 APPLICATION EXAMPLE: A SIMPLE MOTION          *
00003                  ********************************************************
00004                  *
00005                  * THIS PROGRAM SHOWS A VERY SIMPLE MOTION :
00006                  * A SQUARE ROTATION.
00007                  * PROGRAMMING GENERAL PRINCIPLES USED HERE SHOULD BE
00008                  * FOLLOWED FOR MORE COMPLEX APPLICATIONS
00009                  *    -SYSTEMATIC TESTING OF STATUS BIT 2
00010                  *    -SUBROUTINE STRUCTURE
00011                  *    -ERASE-WRITE CYCLES ANIMATION
00012                  *    -WAIT TIME BETWEEN WRITING AND ERASING
00013                  *    -ERASING ONLY DURING VERTICAL BLANKING
00014                  *
00015                  *
00016                  *  THIS PROGRAM IS STRUCTURED INTO SEVERAL SUBROUTINES
00017                  *     BY INCREASING GENERALITY
00018                  *   -EXCHANGE OF DELTAX AND DELTAY PARAMETERS
00019                  *   -SQUARE DRAWING OF KNOWN COORDINATES
00020                  *   -SQUARE DRAWING IN WRITE MODE
00021                  *   -SQUARE DRAWING IN ERASE MODE
00022                  *   -TURN QUARTER MOTION
00023                  *   -COMPLETE TURN MOTION
00024                  *   -INITIALIZATION AND CALLING
00025                  *
00026                  *
00027                  ************************************************************
00028                  *
00029                  * EF9365 REGISTERS ADDRESSES
00030                  *
00031          FDD0  A STATUS EQU    $FDD0
00032          FDD0  A CMD    EQU    $FDD0
00033          FDD1  A CTRL1  EQU    $FDD1
00034          FDD2  A CTRL2  EQU    $FDD2
00035          FDD3  A CSIZE  EQU    $FDD3
00036          FDD5  A DELTAX EQU    $FDD5
00037          FDD7  A DELTAY EQU    $FDD7
00038          FDD8  A MSBX   EQU    $FDD8
00039          FDD9  A LSBX   EQU    $FDD9
00040          FDDA  A MSBY   EQU    $FDDA
00041          FDDB  A LSBY   EQU    $FDDB
00042                *
00043                *
```



```
PAGE  002  EXAMP   .SA:0

00045                  *
00046A 2000                   ORG    $2000
00047                  *
00048                  *
00049                  * ROLLING SQUARE COORDINATES:
00050                  * SUCCESSIVE DELTAX AND DELTAY VALUES DURING A TURN QUARTER.
00051                  *
00052A 2000    5000  A COORD  FDB    $5000,$5008,$4E14,$4C1C,$4824
00053A 200A    422C  A        FDB    $422C,$3C38,$363C,$3040,$2A44
00054A 2014    2248  A        FDB    $2248,$1A4C,$0E50,$0050
00055A 201C    FF    A        FCB    $FF
00056                  *
00057                  * POINTER VARIABLE ON A COORDINATES WORD
00058                  *
00059A 201D    0002  A POINTO RMB    2
00060                  *
00061                  * TESTING MACROS FOR STATUS BITS 1 AND 2
00062                  *
00063                  * TESTS IF GDP IS READY FOR A NEW COMMAND
00064                  READY  MACR
00065                       LDAA STATUS
00066                       BITA  #$04
00067                       BEQ   *-5
00068                       ENDM
00069                  *
00070                  * TESTS VERTICAL BLANKING
00071                  BLANK  MACR
00072                        LDAA STATUS
00073                        BITA #$02
00074                        BEQ *-5
00075                        ENDM
00076                  *
00077                  * 24 BITS (ON A AND X) TEMPO MACRO
00078                  TEMP   MACR
00079                        LDAA #\0
00080                        LDX #\1
00081                        JSR  TEMPO
00082                        ENDM
00083                  *
00084                  * 24 BITS TEMPO SUBROUTINE
00085                  *
00086          201F  A TEMPO  EQU    *
00087A 201F 36           LOOP1  PSHA            SAVE A
00088A 2020 4A           LOOP2  DECA            DECREMENT A
00089A 2021 26 FD 2020          BNE    LOOP2    WHEN A=0
00090A 2023 09                  DEX             DECREMENT X
00091A 2024 27 03 2029          BEQ    EXIT     A IS TAKEN AGAIN FOR NEXT LOOP
00092A 2026 32                  PULA            AS LONG AS X IS NOT =0
00093A 2027 20 F6 201F          BRA    LOOP1
00094A 2029 32           EXIT   PULA            STACK RESTORED
00095A 202A 39                  RTS             END OF TEMPO
00096                  *
```

```
PAGE  003  EXAMP   .SA:0

00098                    *
00099                    * DELTAX AND DELTAY EXCHANGE SUBROUTINE
00100                    *  4 SIDES OF A SQUARE HAVE EQUAL OR EXCHANGED PROJECTIONS
00101                    *  ON BOTH AXIS
00102                    *  TO DRAW A SIDE FROM THE PRECEEDING ONE,YOU ONLY HAVE
00103                    *  TO EXCHANGE DELTAX AND DELTAY
00104                    * THIS SUBROUTINE IS CALLED BETWEEN TWO SIDE WRITING
00105                    *
00106                    *
00107           202B   A INTERV EQU    *
00108A 202B B6 FDD5   A         LDAA   DELTAX
00109A 202E F6 FDD7   A         LDAB   DELTAY
00110A 2031 B7 FDD7   A         STAA   DELTAY   DELTAX GOES IN DELTAY
00111A 2034 F7 FDD5   A         STAB   DELTAX   DELTAY GOES IN DELTAX
00112A 2037 39                  RTS
00113                    *
00114                    *
00115                    * SQUARE DRAWING SUBROUTINE
00116                    * THIS SUBROUTINE ACTUALLY DRAWS THE SQUARE
00117                    * ALL PARAMETERS MUST BE INITIALIZED BEFORE:
00118                    *    DELTAX AND DELTAY MUST CONTAIN THE CORRECT VALUE
00119                    *    THE SAME FOR X AND Y REGISTERS
00120                    * FOR A LEFT TO RIGHT MOTION SUCCESSIVE VECTORS ARE
00121           0013   A SIDE1  EQU    $13
00122           0011   A SIDE2  EQU    $11
00123           0015   A SIDE3  EQU    $15
00124           0017   A SIDE4  EQU    $17
00125                    *
00126           2038   A SQUARE EQU    *
00127A 2038 86 13     A         LDAA   #SIDE1
00128A 203A B7 FDD0   A         STAA   CMD       DRAWS FIRST VECTOR
00129A 203D                     READY  365       READY ?
00130A 2044 BD 202B   A         JSR    INTERV
00131A 2047 86 11     A         LDAA   #SIDE2
00132A 2049 B7 FDD0   A         STAA   CMD       DRAWS SECOND VECTOR
00133A 204C                     READY  365       READY ?
00134A 2053 BD 202B   A         JSR    INTERV
00135A 2056 86 15     A         LDAA   #SIDE3
00136A 2058 B7 FDD0   A         STAA   CMD       DRAWS THIRD VECTOR
00137A 205B                     READY  365       READY ?
00138A 2062 BD 202B   A         JSR    INTERV
00139A 2065 86 17     A         LDAA   #SIDE4
00140A 2067 B7 FDD0   A         STAA   CMD       DRAWS LAST VECTOR
00141A 206A                     READY  365       READY ?
00142A 2071 BD 202B   A         JSR    INTERV    INITIAL PARAMETERS RESTORED
00143A 2074 39                  RTS
```



```
PAGE  004  EXAMP   .SA:0

00145                    *
00146                    *  WRITE MODE SQUARE DRAWING  SUBROUTINE
00147                    * THIS SUBROUTINE IS USED TO MAKE SURE YOU ARE IN WRITE MODE
00148                    * BEFORE YOU CALL SQUARE SUBROUTINE
00149                    *      SAME INPUT PARAMETERS AS SQUARE SUBROUTINE
00150                    *      OUTPUT IS DONE IN WRITE MODE
00151                    *
00152                    *
00153                    *
00154           2075   A WRIT   EQU    *
00155A 2075                     READY  EF9365    READY?
00156A 207C 7F FDD0   A         CLR    CMD       SELECT WRITE MODE
00157A 207F                     READY  COMMAND   EXECUTED ?
00158A 2086 BD 2038   A         JSR    SQUARE    DRAWS THE SQUARE
00159A 2089                     READY  END       OF DRAWING ?
00160A 2090 39                  RTS
00161                    *
00162                    * SQUARE ERASING SUBROUTINE
00163                    * THIS PROGRAM IS USED TO ERASE A SQUARE USING SQUARE
00164                    * SUBROUTINE .TO DO SO,GDP IS PROGRAMMED IN ERASING MODE
00165                    * BEFORE CALLING SQUARE SBR.
00166                    *      SAME INPUT PARAMETERS AS SQUARE SBR
00167                    *      OUTPUT IS DONE IN WRITE MODE
00168                    *
00169           2091   A ERASE  EQU    *
00170A 2091                     READY  EF9365    READY ?
00171A 2098 86 01     A         LDAA   #$01      SELECT ERASING MODE
00172A 209A B7 FDD0   A         STAA   CMD
00173A 209D                     READY  ?
00174A 20A4                     BLANK
00175A 20AB BD 2038   A         JSR    SQUARE    ERASE THE SQUARE
00176A 20AE                     READY  ERASING   END ?
00177A 20B5 7F FDD0   A         CLR    CMD       SELECT WRITE MODE AGAIN
00178A 20B8                     READY  ?
00179A 20BF 39                  RTS
00180                    *
00181                    *
```

```
PAGE  005  EXAMP   .SA:0

00183                    *
00184                    * TURN QUARTER EXECUTING SUBROUTINE
00185                    * THIS SUBROUTINE CALLS ONE AFTER THE OTHER WRIT AND ERASE
00186                    * SUBROUTINES.COORDINATES TABLE IS SCANNED IN ORDER TO DRAW
00187                    * DIFFERENT SQUARES,GIVING A MOTION FEELING.
00188                    * INPUT PARAMETERS ARE:
00189                    *       X AND Y POINT TO THE RIGHT HAND LOWER SQUARE CORNER
00190                    *       DELTAX=SIDE LENGTH,DELTAY =0
00191                    * OUTPUT PARAMETERS ARE:
00192                    *       X AND Y POINT TO THE LEFT HAND LOWER SQUARE CORNER
00193                    *       DELTAX=0,DELTAY=SIDE LENGTH
00194                    *
00195           20C0  A QTURN  EQU    *
00196A 20C0 BD 2075  A         JSR    WRIT      SQUARE WRITING ACCORDING TO ITS COORD.
00197A 20C3                    TEMP   $0A,$0200 SQUARE IS LET ON SCREEN SOME TIME
00198A 20CB BD 2091  A         JSR    ERASE     BEFORE ERASING IT
00199A 20CE FE 201D  A         LDX    POINTO    COORDINATES POINTER UPDATING
00200A 20D1 08                 INX              IN ORDER TO POINT NEXT SQUARE COORD.
00201A 20D2 08                 INX              AND THEN GET NEW VALUES FOR
00202A 20D3 FF 201D  A         STX    POINTO    DELTAX AND DELTAY.
00203A 20D6 A6 00    A         LDAA   0,X       DELTAX VALUE
00204A 20D8 81 FF    A         CMPA   #$FF      IS TABLE OVER ?
00205A 20DA 27 0A 20E6         BEQ    BYE
00206A 20DC B7 FDD5  A         STAA   DELTAX    GOOD VALUE IN DELTAX
00207A 20DF E6 01    A         LDAB   1,X       DELTAY VALUE
00208A 20E1 F7 FDD7  A         STAB   DELTAY    GOOD VALUE IN DELTAY.
00209A 20E4 20 DA 20C0         BRA    QTURN     LOOP UNTIL END OF TURN QUARTER
00210A 20E6 39           BYE   RTS
00211                    *
```



```
PAGE  006  EXAMP   .SA:0

00213                    *
00214                    *  COMPLETE TURN EXECUTING SUBROUTINE
00215                    *  THIS SUBROUTINE CALLS 4 TIMES QTURN TO EXECUTE 4 QUARTERS
00216                    *  INPUT PARAMETERS ARE:
00217                    *  X AND Y MUST POINT THE RIGHT HAND LOWER CORNER OF THE
00218                    *  STARTING SQUARE
00219                    *  OUTPUT PARAMETERS ARE:
00220                    *  X AND Y INCREMENTED BY 4 TIMES THE SIDE LENGTH
00221                    *
00222           20E7  A TURN   EQU    *
00223A 20E7 CE 2000  A         LDX    #COORD    TABLE POINTER  INITIALIZATION
00224A 20EA FF 201D  A         STX    POINTO    AT THE TABLE BEGINNING
00225A 20ED A6 00    A         LDAA   0,X       INITIALIZATION OF
00226A 20EF B7 FDD5  A         STAA   DELTAX    DELTAX
00227A 20F2 E6 01    A         LDAB   1,X       AND
00228A 20F4 F7 FDD7  A         STAB   DELTAY    DELTAY
00229A 20F7 BD 20C0  A         JSR    QTURN     TURN QUARTER EXECUTING
00230A 20FA B6 FDD9  A         LDAA   LSBX      NEXT ROTATION AXIS COMPUTING
00231A 20FD BB 2000  A         ADDA   COORD     AND TEST
00232A 2100 24 03 2105         BCC    NOCARR    TO DETERMINE
00233A 2102 7C FDD8  A         INC    MSBX      IF COMPLETE TURN
00234A 2105 81 C5    A NOCARR  CMPA   #$C5       IS OVER(85+50+50+50+50=1C5)
00235A 2107 27 05 210E         BEQ    OUTPUT    THEN BACK TO DRIVER
00236A 2109 B7 FDD9  A         STAA   LSBX      IF NOT OVER,EXECUTE ANOTHER QUARTER
00237A 210C 20 D9 20E7         BRA    TURN
00238           210E  A OUTPUT EQU    *         WHEN TURN IS OVER
00239A 210E BD 2075  A         JSR    WRIT      SQUARE IS LET A MOMENT
00240A 2111                    TEMP   $0A,$1000 ON SCREEN
00241A 2119 39                 RTS
00242                    *
```

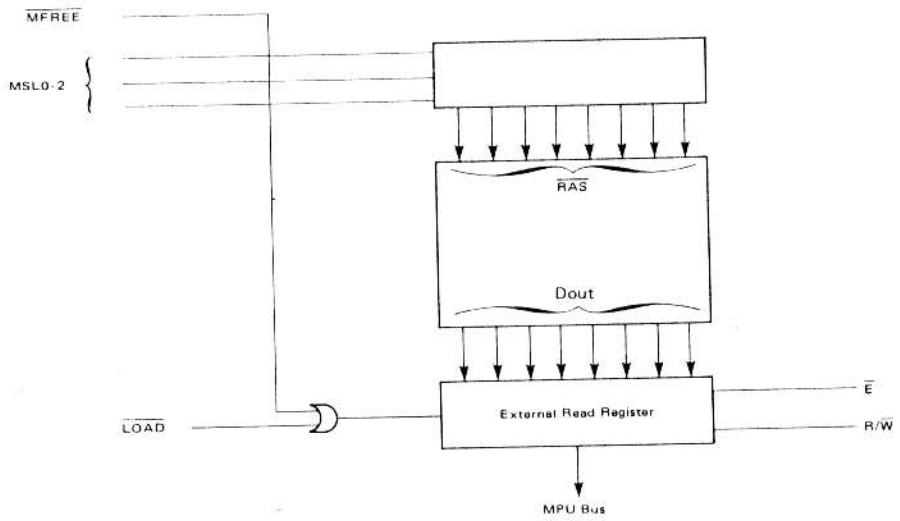
```
PAGE  007  EXAMP    .SA:0

00244                    *
00245                    * DRIVER
00246                    * THIS PROGRAM INITIATES ALL VARIABLES
00247                    * INITIAL SQUARE POSITION
00248                    * PLOTTING TYPE
00249                    * APPROPRIATE OPERATING MODE:WRITE MODE ,PEN DOWN...
00250                    *
00251            211A  A DRIVER EQU     *
00252A 211A 8E 3000   A         LDS     #$3000    STACK INITIALIZATION
00253A 211D 86 03     A         LDAA    #$03      WRITE MODE
00254A 211F B7 FDD1   A         STAA    CTRL1     PEN DOWN
00255A 2122 86 06     A         LDAA    #$06      EF9365 INITIALIZATION
00256A 2124 B7 FDD0   A         STAA    CMD       AND SCREEN ERASING
00257A 2127                     READY   DONE      ?
00258A 212E CE 0085   A         LDX     #$0085
00259A 2131 FF FDD8   A         STX     MSBX      STARTING POINT: X AND Y
00260A 2134 CE 00A0   A         LDX     #$00A0    COORDINATES
00261A 2137 FF FDDA   A         STX     MSBY
00262A 213A BD 20E7   A         JSR     TURN      COMPLETE TURN
00263A 213D 7C FDD2   A         INC     CTRL2     PLOTTING TYPE CHANGE
00264A 2140 20 D8 211A          BRA     DRIVER    THIS PROGRAM NEVER STOPS
00265            211A  A        END     DRIVER
TOTAL ERRORS 00000--00000
```


EXTERNAL ACCESS (READ MODE)

EF9365 and EF9366 allow access to the screen memory from the MPU data bus, with the $0F_{16}$ command.

EF9365 or EF9366 receiving this command, waits for the next write cycle and then it addresses memory at the location pointed to by X and Y registers.

During this cycle $\overline{MFREE}$ signal is at a low level and therefore can be used to enable loading of a register. Then, the MPU will be able to read this register. This cycle is a bit access cycle.



EXTERNAL ACCESS (WRITE MODE)

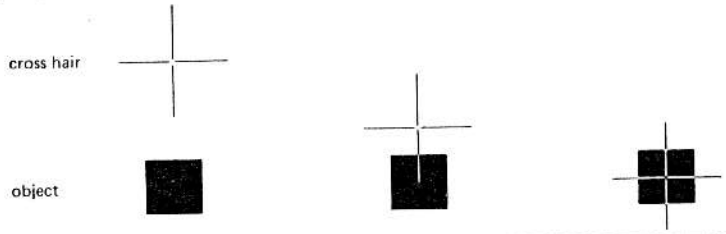
The principle is the same as in the Read mode, but it is necessary to control externally $\overline{WRITE}$ signal applied to memories in order to have a write cycle.

External-write register is programmed by the MPU. $0F_{16}$ command initiates operations and $\overline{MFREE}$ signal enables register contents to be applied to memory inputs.

READ-MODIFY-WRITE

Application example

If a cross-hair is to be moved over a picture, it is imperative not to destroy this picture. If a single memory array is provided, the only mean to draw this cross-hair is to use the read-modify-write mode.

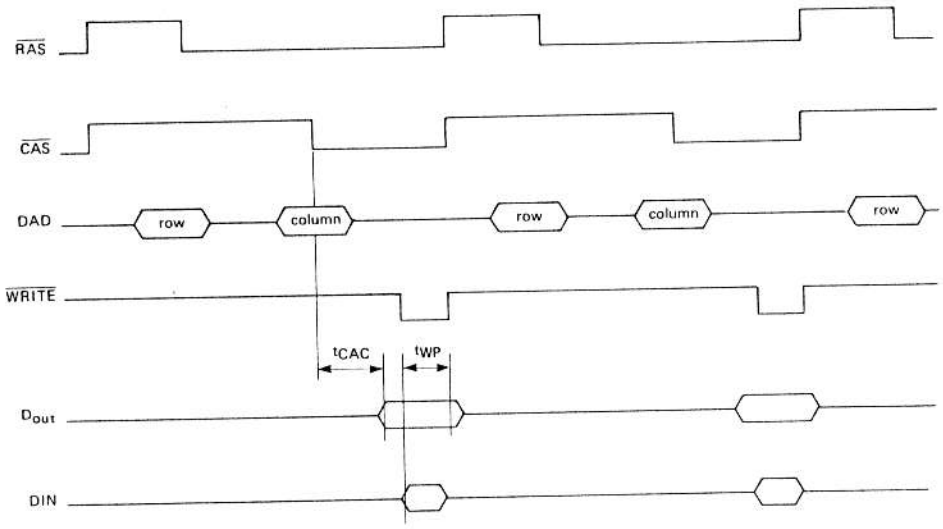


Dots under the cross-hair are complemented. A dot previously lighted is switched off by the cross-hair and a dot previously switched off is lighted.

Erasing of the cross-hair can only be performed by re-drawing it at the same place ; then, the object remains unchanged.

Implementation principle

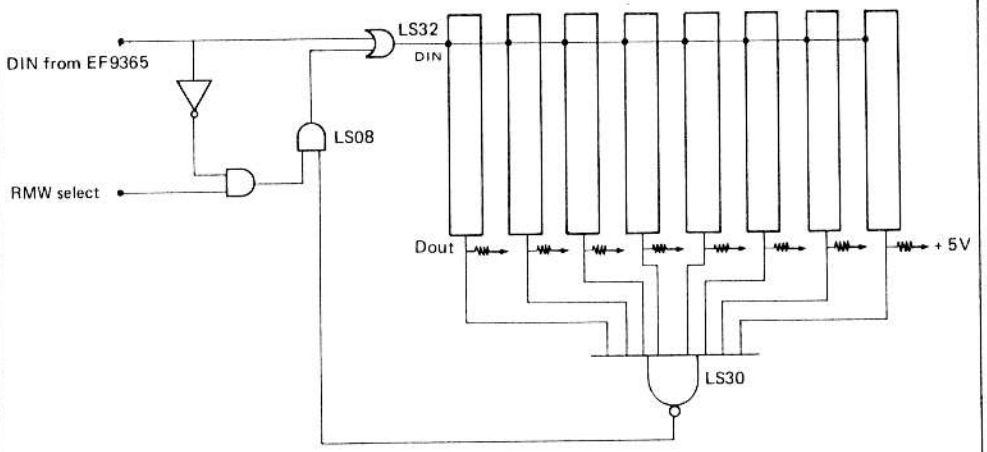
Read-modify-write cycle of dynamic RAM's is used.



First part of the cycle is in Read mode. Data is available on D_{out} pin t_{CAC} after the falling edge of $\overline{CAS}$. Data input must be available on DIN pin when $\overline{WRITE}$ is falling. Timing restrictions compel using dynamic RAMs with 150 ns access time.

Diagram shows that at least 42 ns are available to get the output, modify it and build the DIN input. This input must be valid during at least 45 ns.

Logic diagram



RMW select is programmed by the microprocessor in a one-bit register according to the chosen operating mode .

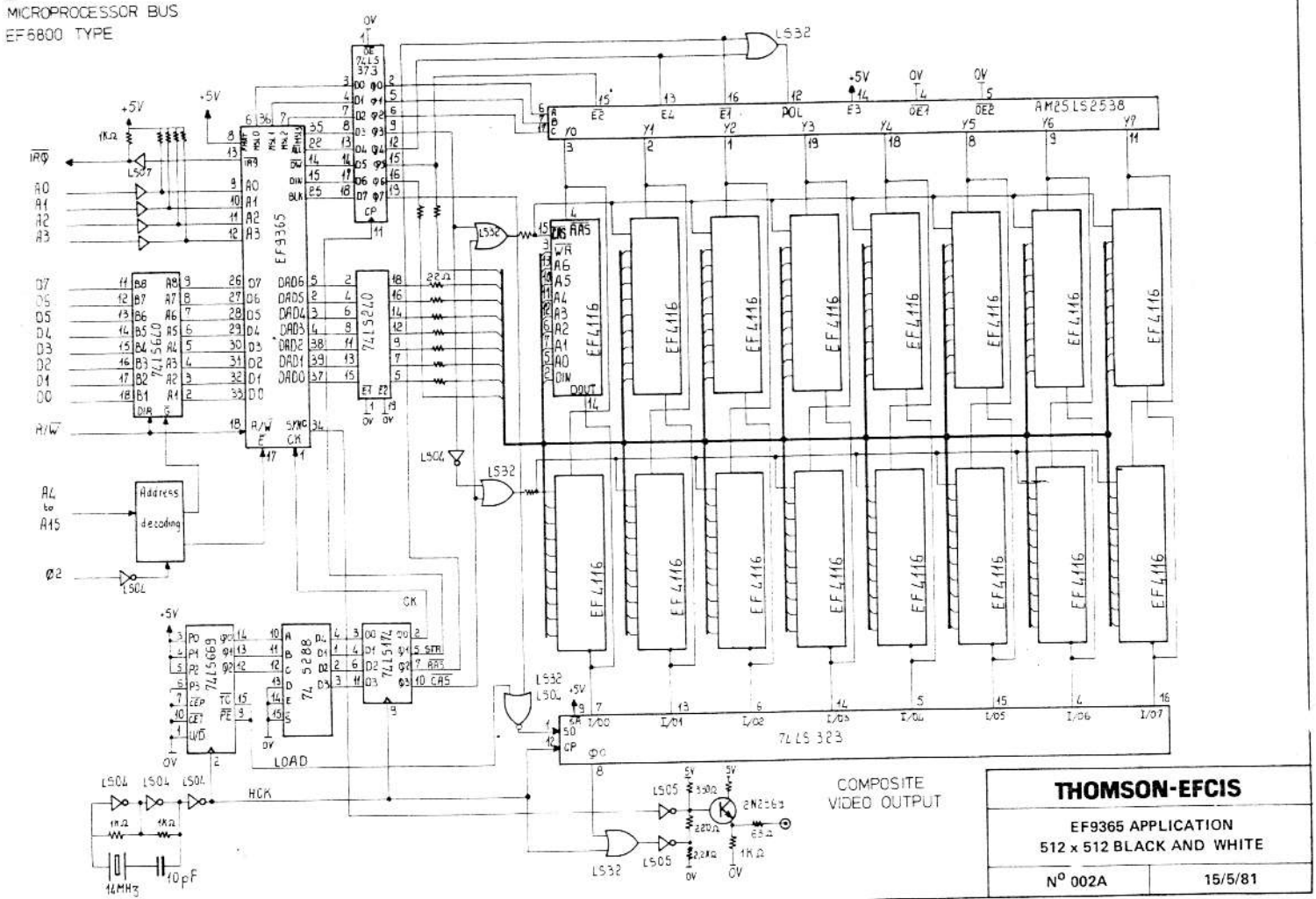
RMW = 1	Read-modify-write mode
RMW = 0	Normal write mode

Read-modify-write is performed only if RMW = 1 and DIN = 0 (Pen down). If RMW = 1 and DIN = 1 (Erasing mode) DIN is preserved and Erasing mode is not disturbed. If RMW = 0, normal write mode of EF9365 remains.

LIGHT PEN OPERATIONS

Light-pen use is very easy with EF9365. LPCK input takes the rising edge provided by the light-pen when it grabs the beam as it passes in front of it. Then XLP and YLP registers keep the coordinates of the light-pen.

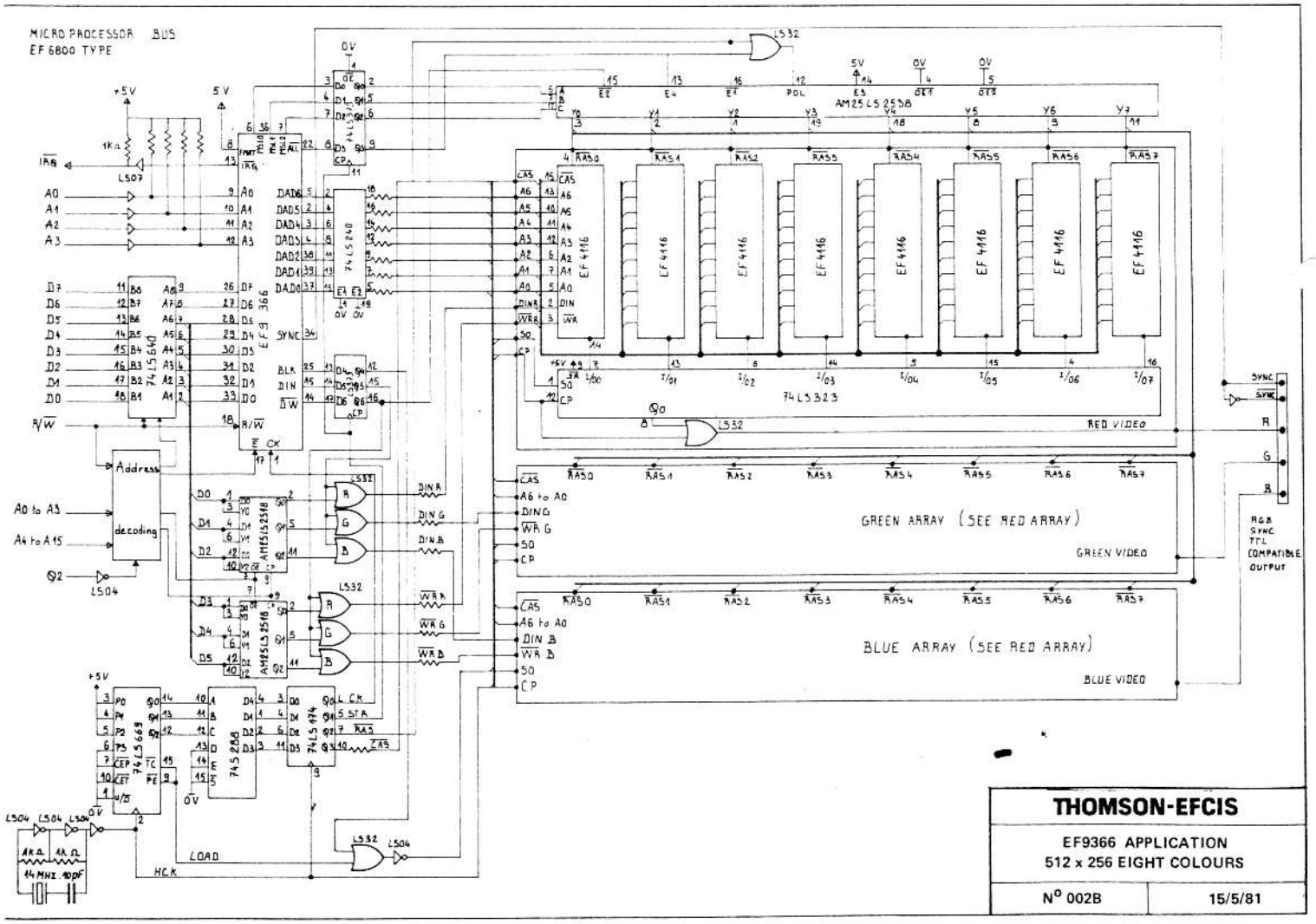
Easiest method consists in connecting $\overline{WHITE}$ output to $\overline{CLEAR}$ input of the shift-register.
08_{16} command initiates a white frame, that the light-pen detects. (cf. specifications).



THOMSON-EFCIS

EF9365 APPLICATION
512 x 512 BLACK AND WHITE

N° 002A	15/5/81



THOMSON-EFCIS

EF9366 APPLICATION
512 x 256 EIGHT COLOURS

N° 002B	15/5/81