TI 9914A COLOR, GRAPHICS AND SPRITES

sp Write only registers

	0	1	2	3
0	Ø	4/16 K RAM 0 1	BASE ADDR SCREEN IMAGE TABLE = R2 * >400	BASE ADDRESS OF COLOR TABLE = R3 * >40
1	Ø	BLANK EN/DIS 0 1		
2	Ø	INTERRUPT EN/DIS 0 1		
3	Ø	MODE BIT 1 (TEXT)		
4	Ø	MODE BIT 2 (multicolor)		
5	Ø	Ø		
6	MODE BIT 3 (BIT MAP) VIDEO ENABLE/ DISABLE	SPRITE SIZE 0-std 1-double		
7		SPRITE MAGNIFY 0-un 1-mag		

BASE ADDR SCRN IMAGE TABLE (# >400)

most sig ↓ least sig

DEFAULTS:

	0	1	2	3
B	ØØ	EØ	ØØ	ØC
EB	ØØ	EØ	ØØ	2Ø
ASL	ØØ	ED	ØØ	OE

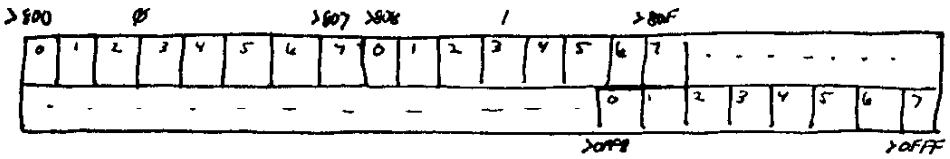
	4	5	6	7
	BASE ADDRESS PATTERN DESCRIPTOR TABLE = R4 * >800	BASE ADDRESS SPRITE ATTRIBUTE LIST = R5 * >80	BASE ADDRESS SPRITE DESCRIPTOR TABLE R6 * >800	COLOR CODE FOREGROUND/ TEXT MODE
				BACKGROUND COLOR ALL MODES

DEFAULTS:

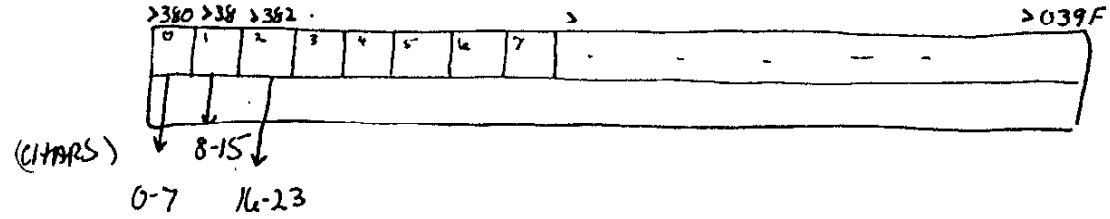
	4	5	6	7
B	ØØ	Ø6	ØØ	17
EB	ØØ	Ø6	ØØ	17
ASL	Ø1	Ø6	ØØ	F5

	GR	TEXT	multi	BIT map	
m1	Ø	1	Ø	Ø	
m2	Ø	Ø	1	Ø	
m3	Ø	Ø	Ø	1	

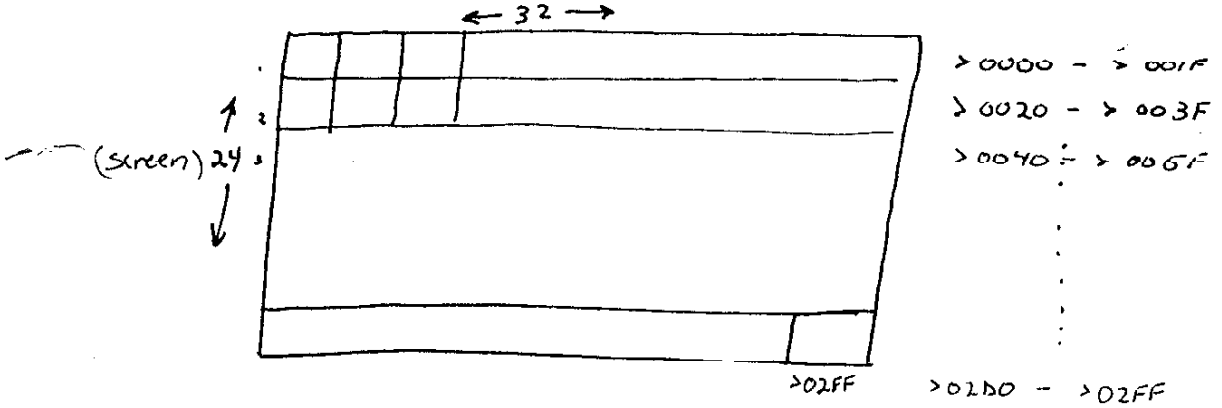
PATTERN DESCRIPTOR TABLE (>800) * each pattern takes 8 bytes of info



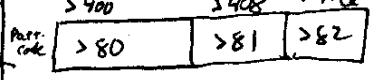
COLOR TABLE (>380)



SCREEN IMAGE TABLE (>0000)

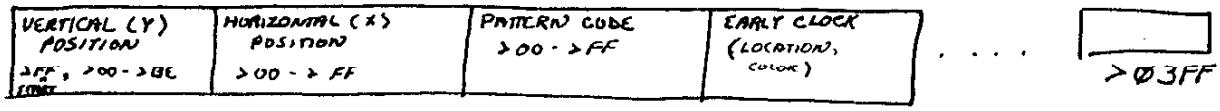


	START	END
SCREEN IMAGE TABLE	>0000	>02FF
SPRITE ATTRIBUTE LIST	>0300	>037F
COLOR TABLE	>0380	>039F
Free space	>03A0	>03FF
SPRITE DESCRIPTOR TABLE	>400	>77F
SPRITE MOTION TABLE	>780	>7FF
PATTERN DESCRIPTOR TABLE	>800	>0FFF



SPRITES

SPRITE ATTRIBUTE LIST (>0300) * @ 4 bytes



SPRITE DESCRIPTOR TABLE (>0000, but >0400 is usually used)

* see pattern descriptor table *

(special notes) >0000 for pattern code >00 (@ 8 bytes)
>0400 for pattern code >80

00 - STD
01 mag
10 - double
11 - double mag

SPRITE MOTION TABLE (>0780)
@ 4 bytes

Y motion	X motion	INTERRUPTS

* interrupts must be ENABLED
* >837A CPU RAM = # of sp. in motion
(2,4 mains = 5 (so 1,4,3,4,10 may move)

GRAPHICS MEMORY MAP

DESCRIPTION	START	END
SCREEN IMAGE TABLE	>0000	>02FF
SPRITE ATTRIBUTE LIST	>0300	>037F
COLOR TABLE	>0380	>039F
FREESPACE	>03A0	>03FF
SPRITE DESCRIPTOR TABLE	>0400	>07FF
SPRITE MOTION TABLE	>0780	>07FF
PATTERN DESCRIPTOR TABLE	>0800	>0FFF

To return to EA:

```
CLR  R0
MOVB R0, @>837C
LWPI GPLWS
LIMI 2
BL   @ >70
```

Side note: by altering CRU locations >8348[,9] returning to >70 will take you to the BASIC INTERPRETER w/o the cursor sprite!

witness [redacted]

Unprotecting EB tapes:

CPURAM >8345 (-31931)

Bit 7 (msBit) -

1 - protected

0 unprotected

On keyboard for 99/4: (getting STATUS byte test to work)

Must do a CLR @KEYBRD to get a key value
BL @SCANKY

from the very start, else STATUS says new key pressed on first code to KEY SCAN when, in actuality, no key has been pressed

---

CHAR SET LOCATIONS:

Grom >4A	lower case	>0874	7 bytes @ (>00 needed for 1st byte
Grom >4E	5 x 7	>06B4	7 bytes @ (>00 needed for first byte
Grom >4C	6 x 8	>04B4	8 bytes @

DISK SECTOR-TO-SECTOR

```
99/4 ASSEMBLER
VERSION 1.2                                                    PAGE 0001
0001                        REF  DSRLNK,VMBW,VSBW
0002                        DEF  START,QUITEX,QUITPC
0003 0000 02E0  START       LWPI >8300
     0002 8300
0004 0004 0200              LI   R0,>302
     0006 0302
0005 0008 C100              MOV  R0,R4          SAVE
0006 000A C800              MOV  R0,@>8356
     000C 8356
0007 000E 0201              LI   R1,MSG
     0010 009A'
0008 0012 0202              LI   R2,2
     0014 0002
0009 0016 0420              BLWP @VMBW
     0018 0000
0010 001A 04E0              CLR  @>834A
     001C 834A
0011 001E 04E0              CLR  @>834E         BUFFER
     0020 834E
0012 0022 04E0              CLR  @>8350         SECTOR
     0024 8350
0013 0026 04C3              CLR  R3             SECTOR COUNT
0014 0028 06A0  LOOP        BL   @HEXOUT
     002A 0066'
0015 002C C820              MOV  @UOR,@>834C
     002E 0094'
     0030 834C
0016 0032 C804              MOV  R4,@>8356
     0034 8356
0017 0036 0420              BLWP @DSRLNK
     0038 0000
0018 003A 000A              DATA 10
0019 003C C803              MOV  R3,@>8350
     003E 8350
0020 0040 C820              MOV  @U1W,@>834C
     0042 0096'
     0044 834C
0021 0046 C804              MOV  R4,@>8356
     0048 8356
0022 004A 0420              BLWP @DSRLNK
     004C 0038'
0023 004E 000A              DATA 10
0024 0050 0583              INC  R3
0025 0052 C803              MOV  R3,@>8350
     0054 8350
0026 0056 0283              CI   R3,359 -360
     0058 0167
0027 005A 11E6              JLT  LOOP
0028 005C 2C40  *           XOP  R0,1
0029 005E 02E0              LWPI >83E0
     0060 83E0
0030 0062 0460              B    @>6A
     0064 006A
0031 0066 C160  HEXOUT      MOV  @>8350,R5
     0068 8350
0032 006A 0200              LI   R0,23*32+14
     006C 02EE
0033 006E C045  LOOP2       MOV  R5,R1
0034 0070 0A45              SLA  R5,4
```

```
0035 0072 0241          ANDI R1,>F000
     0074 F000
0036 0076 0941          SRL  R1,4
0037 0078 0281          CI   R1,>0900
     007A 0900
0038 007C 1202          JLE  NOTHEX
0039 007E 0221          AI   R1,>0700
     0080 0700
0040 0082 0221  NOTHEX  AI   R1,>3000
     0084 3000
0041 0086 0420          BLWP @VSBW
     0088 0000
0042 008A 0580          INC  R0
0043 008C 0280          CI   R0,23*32+14+4
     008E 02F2
0044 0090 16EE          JNE  LOOP2
0045 0092 045B          RT
0046              *
0047 0094 01FF  UOR     DATA >01FF
0048 0096 0200  U1W     DATA >0200
0049      0098' QUITEX  EQU  $
0050 0098 0000  QUITPC  DATA 0
0051 009A 0110  MSG     DATA >0110
0052 009C   44          TEXT 'DSK1.T     '
0053                    END
0000 ERRORS
```

```
        REF   DSRLN$,VMBW$,VSBW$
        REF   CLRSC$,VMBR$,VSBR$
        DEF   SCOPY,QUITEX,QUITPC
        DEF   SFIRST,SLOAD
SFIRST  EQU   $
SLOAD   EQU   $
SCOPY   LWPI  >8300          my WORKSPACE
        LI    R0,>302        VDP ADDRESS
        MOV   R0,R4          SAVE
        MOV   R0,@>8356      MOVE TO NAME LENGTH AREA
        LI    R1,MSG         LENGTH & CALL
        LI    R2,2           2 BYTES TO WRITE
        BLWP  @VMBW$         WRITE PAB
        CLR   @>834A         START AT SECT 0
        CLR   @>834E           BUFFER CLR, SO THE BUFFER IS IN THE SIT
        CLR   @>8350           SECTOR NUM
        CLR   R3               SECTOR COUNTER
LOOP    CLR   R0             SET TO ZERO
        LI    R1,256         BYTES TO READ/WRITE
        BLWP  @CLRSC$        CLEAR SCREEN
        BL    @HEXOUT        PRINT SECTOR # ON SCREEN
        MOV   @UOR,@>834C    UNIT ONE READ
        MOV   R4,@>8356      PAB NAME LENGTH POINTER
        BLWP  @DSRLN$        ] SECTOR READ
        DATA  10
        CLR   R0             ] READ THE FIRST BYTE
        BLWP  @VSBR$
        CB    R1,@NULL       ANYTHING THERE?
        JNE   WSEC           YES, SO WRITE THE SECTOR TO THE OTHER DISK
        LI    R1,BUF         BUFFER AREA
        LI    R2,256         AND LENGTH
        BLWP  @VMBR$         READ THE RECORD
CHECK   C     @NULL,*R1+     TEST ALL BYTES FOR YES
        JNE   WSEC           FOUND SOMETHING ELSE, SO WRITE IT
        CI    R1,BUFE        END OF BUFFER?
        JNE   CHECK          NO, SO KEEP LOOPING
        JMP   NOW            NOTHING FOUND, SO NEXT SECTOR
WSEC    MOV   R3,@>8350      SECTOR NUMBER
        MOV   @U1W,@>834C    WRITE THIS
        MOV   R4,@>8356      PAB NAME/LENGTH POINTER
        BLWP  @DSRLN$        ] WRITE THE SECTOR
        DATA  10
NOW     INC   R3             INCREMENT SECTOR COUNT
        MOV   R3,@>8350      MOVE TO INPUT PARMS
        CI    R3,360         LAST SECTOR ?
        JLT   LOOP           NO, SO KEEP GOING
*       XOP   R0,1           ------> jump to debugger
        LWPI  >83E0          ] RETURN TO E/A
        B     @>6A
HEXOUT  MOV   @>8350,R5      MOVE SECTOR # TO R5
        LI    R0,23*32+14    SCREEN ADDRESS
LOOP2   MOV   R5,R1
        SLA   R5,4
        ANDI  R1,>F000
        SRL   R1,4
        CI    R1,>0900
        JLE   NOTHEX
        AI    R1,>0700
NOTHEX  AI    R1,>3000
        BLWP  @VSBW$
        INC   R0
        CI    R0,23*32+14+4
        JNE   LOOP2
        RT
```

PRINT SECT # ON SCREEN

```
*
UOR     DATA >01FF
U1W     DATA >0200
MSG     DATA >0110
QUITEX  EQU  $
QUITPC  EQU  $
NULL    DATA >E5E5
BUF     BSS  256
BUFE    EQU  $
        END
```

"INSERT LINE" for MW/MEMORY
... fills designated bytes with >ØØ

```
110 CALL CLEAR
120 CALL SCREEN(2)
130 FOR I=1 TO 8
140 CALL COLOR(I,16,2)
150 NEXT I
160 DIM HEXTBL(4)
170 HEXTBL(1)=4096
180 HEXTBL(2)=256
190 HEXTBL(3)=16
200 HEXTBL(4)=1
210 DV=0
220 REM  CALL CLEAR
230 INPUT "STARTING ADDRESS (4 DIGITS)?        >":HEX_START$
240 IF LEN(HEX_START$)>4 THEN 210
250 IF HEX_START$<"7D00" THEN 210
260 IF HEX_START$>"7FE0" THEN 210
270 L=LEN(HEX_START$)
280 FOR I=L TO 1 STEP -1
290 HD$=SEG$(HEX_START$,I,1)
300 IF ASC(HD$)>71 THEN 210
310 IF ASC(HD$)>64 THEN 320 ELSE 350
320 HEXDGT=ASC(HD$)
330 HEXDGT=HEXDGT-55
340 GOTO 360
350 HEXDGT=VAL(SEG$(HEX_START$,I,1))
360 DV=DV+(HEXDGT*HEXTBL(I))
370 NEXT I
390 PRINT ::
400 INPUT "# OF BYTES TO INSERT?     ":BYTES
410 IF BYTES=0 THEN 550
420 IF BYTES>6 THEN 400
430 START=DV
440 GAP=BYTES
450 SHIFTS=32735-GAP
460 REM  NOW DO THE SHIFTING
470 FOR I=SHIFTS TO START STEP -1
480 CALL PEEK(I,BYTE)
490 CALL LOAD(I+GAP,BYTE)
510 NEXT I
520 FOR I=START TO START+GAP-1
530 CALL LOAD(I,0)
540 NEXT I
550 END
```

KEYBOARD SCAN FOR 99/4 AND 99/4A

TO DIRECTLY SCAN THE KEYBOARDS OF THE 99/4 OR 99/4A, IT IS NECESSARY TO DO CRU I/O OPERATIONS. FIRST, THREE BITS ARE USED TO SELECT A PAIR OF COLUMN TO SCAN. (ONE ON THE LEFT SIDE AND ANOTHER ON THE RIGHT.) THEN EIGHT BITS ARE READ TO DETERMINE WHICH KEYS ARE DEPRESSED.

TO SCAN THE INNER TWO COLUMNS ON THE 99/4:

10 ms

```
LI   R12,>24     CRU BASE FOR OUTPUT STROBE
LI   R1,>400     CODE FOR INSIDE TWO COLUMNS
LDCR R1,3        OUTPUT 3 LS BITS OF R1 MS BYTE
LI   R12,6       CRU BASE FOR INPUT SCAN
STCR R4,8        READ 8 BITS INTO R4 MS BYTE
```

R4'S MOST SIGNIFICANT BYTE NOW CONTAINS 1'S AND 0'S FOR KEY SWITCHES.
1'S MEAN KEY UP, 0'S MEAN KEY DOWN.
OTHER COLUMNS MAY BE SCANNED BY PUTTING OTHER VALUES IN R1.

99/4 KEYBOARD MATRIX

R1 VALUE	KEYS RETURNED (MS BIT ... LS BIT)								
	80	40	20	10	08	04	02	01	
>0000	1	Q	space	shift	0	P	L	enter	
>0100	2	W	A	Z	9	O	K	.	
>0200	3	E	S	X	8	I	J	M	
>0300	4	R	D	C	7	U	H	N	
>0400	5	T	F	V	6	Y	G	B	
>0500				up	down	right	left	fire	(joystick 1)
>0600				up	down	right	left	fire	(joystick 2)

99/4A KEYBOARD MATRIX

R1 VALUE	KEYS RETURNED								
>0000		ctrl	shift	fctn		enter	space	=	
>0100	X	W	S	2	9	O	L	.	
>0200	C	E	D	3	8	I	K	,	
>0300	V	R	F	4	7	U	J	M	
>0400	B	T	G	5	6	Y	H	N	
>0500	Z	Q	A	1	0	P	;	/	
>0600				up	down	right	left	fire	(joystick 1)
>0700				up	down	right	left	fire	(joystick 2)

THE 99/4A ALPHA LOCK KEY MAY BE TESTED WITH:

```
CLR  R12
SBZ  21        power the alpha lock line
SRC  R12,14    kill some time
TB   7         see if switch is closed
JEQ  NOALPH    if not closed
...
SBO  21        turn off alpha lock line before leaving
```

NOTE: THE 99/4 KEYBOARD WILL OBVIOUSLY NEVER BE REDEFINED, BUT THE 99/4A KEYBOARD IS SUBJECT TO CHANGE WITHOUT NOTICE DURING PRODUCT LIFE.

BREAKING OBJECT MODULES THAT ARE TOO LARGE

1. LINK OBJECT MODULES W/ XLE

   control file:

   FORMAT ASCII
   TASK DEMON (or any name)
   PROG >2000
   INCL "filename1" (no quotes)
   PROG >A000
   INCL "filename2" (no quotes)
   END

2. RESULT is a linked object file
3. EDIT the linked file
   a) FIND THE 9 tag to the 2nd prog + SPLIT file
   b) ADD a >F to last byte of 1st file
   c) Put a : on next line
   d) SAVE this file out
4. EDIT the 2nd part
   a) ADD a >ØØØØ XXXXXXXX to the first line and place a >F over the >7F for checksum (XX... is a new name)
   b) SAVE this file out (may check last line for comments ... delete + place >F on last nybble)

5. 2 object files are now generated

CHANGING 'LOAD + GO' OBJECT FILES TO 'LOAD + RUN' FILES

1. Insert Disk. - LOAD AND RUN "DSK#.DISK0", PGM NAME= "START"
2. Choose 2. Search
   2a. get last sector #
3. Choose 1. Sector Edit
   3a. Sector # = last sector # - 1
4. Press Fctn 2 (ASCII)
5. Change 1 in UL corner to 'B'
6. Press Fctn 1 (HEX)
7. Find '31A000' in UL corner 31A01L
8. Change 'A000' to 0000
9. Press fctn 'REDO' and SAVE sector
10. Find DEF in REF/DEF table (>3F00 - >3F50)