MICRO CHROMA 68 THE NEW "BUG" FROM MOTOROLA TVBUG®

MOS Microcomputer Systems Applications
Austin, Texas

Prepared By

Tim Ahrens
Jack Browne
Monitor written by John Dumas

The information contained in this application note allows the construction of a low cost development system. Object code (machine language) programs may be entered from the keyboard or loaded from audio cassette tapes and be debugged and developed. The programs may be dumped to cassette tape for permanent storage. Techniques for increasing system capabilities are included.

TABLE OF CONTENTS

TABLE OF CONTENTS (con't.)

(e.g. space bar). The modifier will be either two or four hexcharacters depending on the command mode. When entering hexidecimal data, i.e. memory examine function, the data entered is right-hand justified. For example: A desired address $015F is to be entered in the memory examine function. All that need be entered is $15F. If an error is entered ($15E) just type $015F following the previously entered three digits. The TVBUG firmware may be used to debug and evaluate a user program and to perform the following functions:

1) G - Go to Address "N" (User Program)
2) L - Load Kansas City Standard Tape (JBUG® Format)
3) P - Punch/dump Kansas City Standard (JBUG Format)
4) V - Verify Kansas City tape
5) M - Memory change
6) E - Examine a block of memory
7) Q - Quick load of Hex Data
8) F - Fill a block of memory
9) O - Offset calculation
10) R - Display contents of MPU registers
11) Z - Clear screen and initialize I/O
12) S - Set a breakpoint with address "N"
13) U - Unset breakpoint with address "N"
14) D - Delete all breakpoints

15) B - Print out all breakpoints
16) N - Trace the next instruction
17) C - Continue execution from the current location
18) T - Trace "N" instructions
19) !,",# - User defined functions

4.1.1 G - Go to User Program Function

This function allows the user to execute a USER program. To use this function type a "G", starting address, and return. The firmware will execute a USER program.

4.1.2 L - Load Tape Function (Kansas City Standard)

The function allows the user to load a Kansas City Standard formatted audio cassette tape. This includes tapes punched using Motorola's JBUG® and CRTBUG® monitor. To use this function:

1. Press Reset
2. Type "L". The firmware will CRLF and ask for an offset, (16 bits, Hexadecimal, with leading zeros assumed).
3. Enter the offset. The offset must be the difference between the existing start address and the desired start address; if none, type a space.
4. Type "return". Start the tape by pressing "play" on the cassette recorder. Insure that the recorder "ear" to P.C. board "ear" is connected.
5. After approximately 40 seconds of leader, the firmware will print a name if any, and a "B" for each 256 bytes and a "B" for the remainder, if any. If the data was not stored into memory correctly, the "B" is followed by the message, "MEMORY BAD" and the firmware will return to TVBUG program control.

4.1.3 P - Punch Tape Function (Kansas City Standard)

This function allows the user to store data from memory on audio cassette tape using the Kansas City Standard. To use this function:

1. Press Reset.
2. Type "P". The firmware will CRLF and ask for a beginning address.
3. Enter beginning address and type a space. The firmware will ask for a ending address.
4. Enter ending address. The firmware will CRLF and ask for a name.
5. Enter the name. The name may be up to 32 (31 + CR) characters long. If tape must be read by a JBUG monitor, do not use "B" or "G" in the name as these characters are interpreted by the JBUG firmware as control characters.
6. Connect the tape recorder "mike" to the P.C. board "in" (P3) and start recording.
7. Type return. The firmware will print 40 seconds of leader (F's) followed by an 80 (Start Char.), Name (ASCII Code), Byte count, Starting Address, and "42" (ASCII "B") followed by data. A short leader terminated with "42" (ASCII "B") will be printed for each 256 bytes.

4.1.4 V - Verify Tape (Kansas City Standard)

This function is used to verify a PUNCH or LOAD operation. To use this function:

1. Press Reset.
2. Enter a "V". The firmware will CRLF and ask for an offset.

3. Enter the offset. The offset must be the difference between the existing start address and the desired start address; if none, type a space.
4. Set up the tape recorder as shown in the load function.
5. The firmware will print file name, CRLF, and print a "B" for each 256 bytes. If the data on the tape and the contents of the memory do not agree, the firmware will print "MEMORY BAD" and return to TVBUG program control.

4.1.5 M - Memory Change Function

The function will examine a location in memory, change the contents if desired, and return the contents to memory in that order. To use the MEMORY CHANGE function:

1. Enter an "M".
2. Enter the address to be changed and press line feed. TVBUG firmware will CRLF and print the address followed by data.
3. Enter new data if desired. Line feed will then return data to memory and open the next location. Up arrow (↑) will return data to memory and open the previous location. To return to TVBUG control program, press the carriage return key.

```
TVBUG
M 0
0000 XX 00
0001 XX 00
TVBUG
```

4.1.6 E - Block Memory Examine Function

This function allows the user to display a block of memory on the screen. To use this function:

1. Enter an "E".
2. Enter the beginning address of the block to be examined and type a space. The firmware will ask for ending address.
3. Enter an address and type a space.
4. The firmware will CRLF and print the beginning address and contents of the first eight memory locations. Underneath the contents of each location is a period. If the data at that location is an ASCII character, the character will be printed under the data. Each time a space is entered, the next 8 locations will be printed until it reaches the ending address; at which time the firmware will return to the TVBUG control program.

```
TVBUG
E
BEG ADR?0  END ADR?F

0000   54  56  20  42  55  47  XX  XX
        T   V       B   U   G   .   .
0008   XX  XX  XX  XX  XX  XX  XX  XX
       .   .   .   .   .   .   .   .
TVBUG
```

4.1.7 Q - Quick Load Function

This function allows the user to enter blocks of hex data using the MEMORY EXAMINE function. To use this function:

1. Type "Q". The firmware will CRLF and ask for the beginning address.
2. Enter beginning address and type a space. The firmware will ask for the ending address.
3. Enter ending address and type return. The firmware will CRLF, print the beginning address and wait for data.
4. Enter hex data followed by a space. The firmware will CRLF on the 8th location, print the address and wait for data. When the ending data has been entered, the firmware will return to TVBUG control program.

Typical Display
```
TVBUG
Q
BEG ADR?0   END ADR?F

0000   XX  XX  XX  XX  XX  XX  XX  XX
0008   XX  XX  XX  XX  XX  XX  XX  XX
TVBUG
```

4.1.8 F - Memory Fill Function

This function allows the user to fill a block of memory with a character. To use this function:

1. Enter an "F". The firmware will CRLF and ask for the beginning address.
2. Enter the beginning address and type a space. The firmware will ask for an ending address.
3. Type a space. The firmware will CRLF and ask for a character.
4. Enter the desired character and type return. The firmware will write the character into each of the defined memory locations and return to TVBUG program control.

Typical display for MEMORY FILL function:
Note: Filling Stack RAM may result in loss of control as the MPU may execute an unimplemented opcode.

```
BEG ADR?   XXXX   END ADR?  XXXX
CHAR?   XX
TVBUG
```

4.1.9 O - Offset Calculation Function

This function allows the user to calculate 16-bit offsets. If the offset is outside the 8-bit "branch" limits, the firmware will print the offset followed by the message "TOO FAR". This function simplifies the calculation of offsets for branch instructions. To use this function:

1. Type an "O". This firmware will CRLF and ask for the beginning address.
2. Enter the address of the branch op code and type a space. The firmware will CRLF and ask for the ending address.
3. Enter the address of branch destination and type return. The firmware will CRLF, print the offset and return to TVBUG control program. Offsets will be printed as a 16-bit word. The least significant 8 bits will be the offset.

```
Positive Offset:
0
BEG ADR?0  END ADR?F
OFFSET = 000D
TVBUG

Negative Offset:
0
BEG ADR?F  END ADR?0
OFFSET = FFEF
TVBUG

Offset Outside of an 8-bit branch:
0
BEG ADR?0  END ADR? 82
OFFSET = 0080 TOO FAR!
TVBUG
```

4.1.10 R - Print contents MPU Registers

This function allows the user to examine the MPU registers by reading them from the stack. To use this function type "R". The firmware will place contents of the MPU registers onto the stack RAM and then place them on the screen in the following format:

```
   CC  B  A   X    P    S
   XX XX XX XXXX XXXX XXXX
```

Where:
CC = Condition Code register
B = B accumulator
A = A accumulator
X = Index register
P = Program counter
S = Stack pointer

4.1.11 Z - Clear Screen Function

To use this function type "Z". The firmware will fill the display memory block with a space character, clear the screen, initialize the system I/O ports, and return to TVBUG program control.

4.1.12 Breakpoints

There are 7 TVBUG commands dealing with breakpoints.
1) S - Set breakpoint with address "N".
2) U - Unset breakpoint with address "N".
3) D - Delete all breakpoints
4) N - Next instruction
5) T - Trace "N" instructions
6) C - Continue execution from current location
7) B - Print out all breakpoints

4.1.12.1 S - Set A Breakpoint with Address "N"

To set a breakpoint type an "S" followed by the address, then type return. The firmware will print the breakpoint address with up to 7 additional breakpoints

that might be set, and return to TVBUG program control.
Note: Breakpoint $0000 is illegal.

Typical Display for NEXT Instruction:

```
XX        XX   XX   XXXX   0030   XXXX

TVBUG

N
XX        XX   XX   XXXX   0032   XXXX
```

TVBUG Typical display for Setting Breakpoints:

```
TVBUG
S 10
0010
TVBUG
S 20
0010 0020
TVBUG
S 30
0010 0020 0030
TVBUG
```

4.1.12.2 U - Unset A Breakpoint with Address "N"

To unset a breakpoint type a "U" followed by the address, then return. The firmware will remove the breakpoint and return to TVBUG control program.

Typical Display for Unsetting Breakpoints:

```
TVBUG
U 10
TVBUG
U 20
TVBUG
```

4.1.12.3 D - Remove all Breakpoints

To remove all breakpoints, type a "D". The firmware will remove the breakpoints and return to TVBUG program control.

4.1.12.4 B - Print Out all Breakpoints

To examine breakpoints type a "B". The firmware will print all breakpoints and return to TVBUG control.

NOTE: The following commands assume that a program has been exeucted and halted at a Breakpoint.

4.1.12.5 N - Trace Next Instruction

This command allows the user to single step through a series of instructions. To use this command, type an "N". The firmware will execute the NEXT instruction and print the contents of the MPU registers. It will then return to TVBUG program control.

4.1.12.6 C - Continue

The Continue command is used to step the program from breakpoint to breakpoint. To use this command, type a "C". The firmware will execute the user program from the current location to the next breakpoint, and print out the contents of the stack.

Typical Display for CONTINUE Instruction:
(Breakpoints set at $0030, $0040, $0050)

```
XX  XX  XX  XXXX  0030  XXXX
TVBUG
C
XX  XX  XX  XXXX  0040  XXXX
TVBUG
C
XX  XX  XX  XXXX  0050  XXXX
TVBUG
```

4.1.13 User Defined Functions

TVBUG contains three user defined jumps that may be called from the keyboard and two user defined jumps called by the monitor. All jumps are initialized with a Reset. However, if the user wishes to prevent these vectors from being lost on Reset, the stack RAM may be hardware deselected from $F390, to $F39F inclusive. A small ROM, containing the permanent vectors, is patched over these locations (see listing in appendix B) as shown in Figure 4.2.

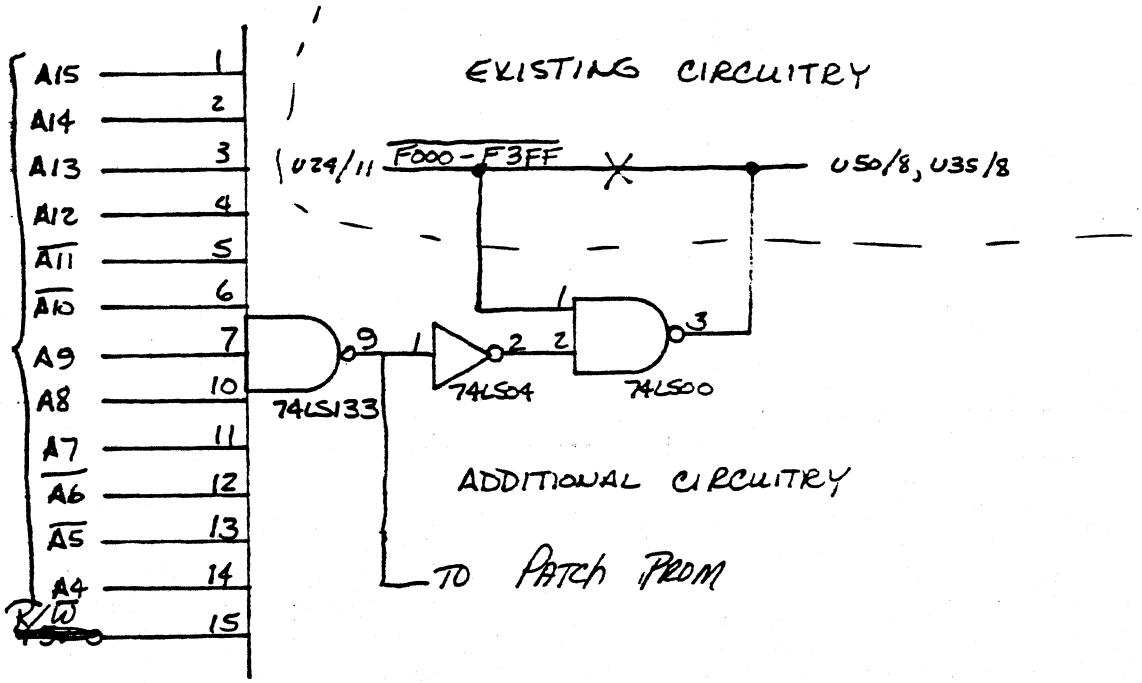


Figure 4.2 Hardware Deselect of Stack RAM from $F390 to $F39F

To use the keyboard jumps, type the appropriate character (!,",#). Firmware will then execute the program from the address stored in temporary RAM at the following locations:

CHAR.	INST. ($7E)	HIGH BYTE	LOW BYTE
!	$F396	$F397	$F398
"	$F399	$F39A	$F29B
#	$F39C	$F39D	$F39E

4.1.14 User Input Function

This function flowcharted in Figure 4.3 allows the user to insert a user routine into the monitor input loop. Each time the monitor goes around its input loop it checks the user input three byte vector. Since it is initialized to RTS, the monitor will ignore this vector until the user changes it. The three temporary RAM locations reserved for the user input vector are:

INST. (7E)	HIGH BYTE	LOW BYTE
$F390	$F391	$F392

To use this function, first write the user vector into stack.

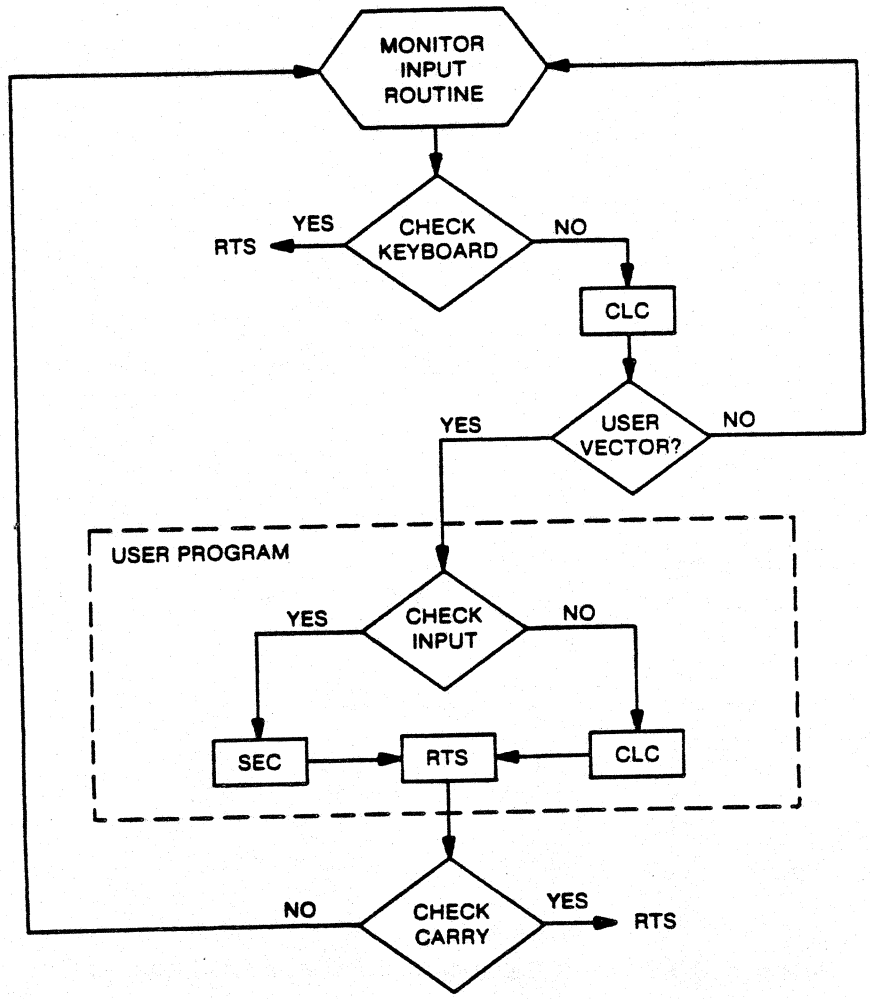


Figure 4.3 Flowchart for User Input Function

Example:
LDAA: User vector (High Byte)
STAA: $F391
LDAA: User vector (Low Byte)
STAA: $F392
LDAA: #$7E
STAA: $F390
Initialization complete

If this vector is entered with the keyboard, the jump instruction (7E) must be entered last.

The USER INPUT routine must set the carry bit if there was a user input. If there was no input it must clear the carry bit. All user I/O routines must end with RTS.

4.1.15 User Output Function

This function flowcharted in Figure 4.4 allows the user to insert a user output routine into the monitor output routine. Each time the monitor performs its OUTCH (output character) routine it checks the three temporary RAM locations reserved for the user output vector.
Since these locations are initialized to RTS, the monitor will ignore them until they are changed by the user.

INST. (7E)	HIGH BYTE	LOW BYTE
$F393	$F394	$F395

To use this function, write the jump vector into temporary RAM. If the vector is left in temporary RAM, I/O devices such as a printer or a modem may be controlled on the fly by changing the instruction location ($F393) from the jump ($7E) to an RTS ($39). All user I/O routines must end with RTS.

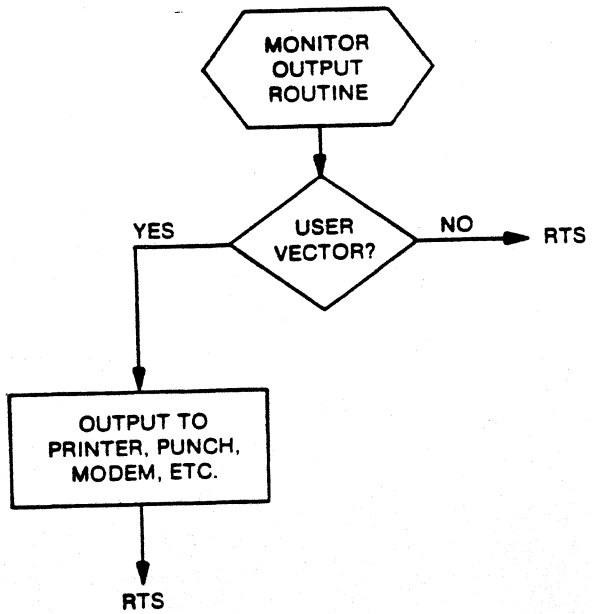


Figure 4.4 Flowchart for User Output Routine

4.2 Software Example

The following example program is suitable for gaining familiarity with the TVBUG monitor features. The program adds the five values in locations $10 through $14 using Accumulator A and stores the final result in location $15. The intermediate total is kept in Accumulator A; Accumulator B is used as a counter to count down the loop. The Index Register contains a "pointer" (i.e., X contains the address) of the next location to be added. The program, as follows, contains an error which will be used later to illustrate some of TVBUG's features.

In the following listing, the leftmost column contains the memory address where a byte (8 bits) of the program will be stored. The next column contains the machine language op-code and data for a particular microprocessor instruction. The next four columns contain the mnemonic respresentation of the program in assembler format.

```
*
*Add 5 numbers at locations 10-14
*Put answer in location 15
*

0020    8E     STRT     LDS     $FF      DEFINE STACK IN USER AREA
0021    00
0022    FF
0023    4F              CLRA             TOTAL = 0
0024    C6              LDAB    #4       INITIALIZE COUNTER
0025    04
0026    CE              LDX     #$10     POINT X TO LOCATION 10
0027    00
0028    10
0029    AB     LOOP     ADDA    0,X      ADD 1 LOCATION TO TOTAL
002A    00
002B    08              INX              POINT X TO NEXT LOCATION
002C    5A              DECB             DONE ALL 5 LOCATIONS?
002D    26              BNE     LOOP     BRANCH IF NOT.
002E    FA
002F    97              STAA    $15      SAVE ANSWER
0030    15
0031    3F              SWI              GO TO TVBUG
```

A detailed procedure for entering and debugging this program is shown in the following steps.

1. Start up and enter the program in RAM

   A. Turn power on. Push reset button on the card. TVBUG will respond as shown in Figure 4.1.
   B. Type M followed by 20CR. This displays the current contents of location $0020.
   C. Type 8E. This replaces the contents of $0020 with 8E which is the op-code for the first instruction, LDS.
   D. Type LF. This steps to the next location ($0021) and displays the contents.
   E. Type 00.

F. Type LF.
G. Type next byte of op-code or operand (FF in this case).
H. Repeat steps F and G for remaining instuctions.
I. Type CR to close the memory change function.

2. Verify that the program was entered correctly.

A. Type M 20 CR. Location 20 will be displayed.
B. Type LF. Next location will be displayed.
C. Repeat step B until done, visually verifying data entered in Step 1.
D. Type CR.

3. Enter Data in Locations 10-14

A. Same as 1 except type M 10 CR to start the sequence. Any data may be entered; however, for purposes of this example 01,02,03,04 and 05 should be entered.
B. Type CR

4. Verify Data

A. Repeat step 2 except type M 10 CR to begin the sequence. Verify that the memory contains the values 01,02,03,04 and 05 in sequencial order.

5. Run the Program

A. Type CR to insure no other option is active.
B. Type G 20. The program will run down to the "SWI" instruction at location 31 which will cause it to go to TVBUG and show the following display.

```
CC       B      A        X          P      S
DO      00     0A      0014        0032   00G8
TVBUG
```

6. Check the Answer

Type M 15 CR. (The answer is stored in location 15). Note that it says $0A (decimal 10). The correct answer is $0F or decimal 15; therefore, there is a problem in the program as originally defined. The next steps should help isolate the problem and correct it.

7. Breakpoint and Register Display

A. It might be helpful to see what the program was doing each time it went through the loop. Therefore, set a breakpoint at the beginning of the loop, location 0029. To do this type S 29 CR.
B. A breakpoint could also be set at location 002F to see the results. Type S 2F CR.
C. TVBUG must be told where to begin, so type G 20. TVBUG will run to the breakpoint and then display 0029 as the program counter. At this point the program is suspended just before location $29 and is in TVBUG. On detecting this breakpoint, TVBUG automatically displays the register contents.
D. Type C to return to the example program and resume executing. Since the breakpoint at location $0029 is in a loop it will again be the next breakpoint. At this point the register contents will be displayed again.

If this were done the A Register would appear to contain the partial sum and the B Register would be decremented. The X Register would be incremented by one.

E. Type C (Proceed). Once again the registers contents will be displayed.
F. Type C (Proceed). Same comment as D.
G. Type C (Proceed). Display will now show register contents as of breakpoint at $2F. The program has now successfully completed the loop 4 times and the A-Register contains the incorrect sum.

8. Correcting the Program

A. From above it is evident that although the program was supposed to add five numbers, the loop was executed only four times. Therefore, the LDAB #4 instruction at location 24 and 25 should have initialized B to 5.
B. Type D. Clear existing breakpoints.
C. Type M 25 CR. This display = 0025 04.
D. Type 05. The display = 0025 0405 enter 05. This will now permanently change the LDAB #4 instruction to a LDAB #5 instruction.
E. Type CR
F. Type G 20. Execute the program.
G. Type M15 Display = 0015 0F, the expected answer; the program is fixed.

9. Trace Through the Program

A. In order to execute a trace, the program must first be stopped at a breakpoint. To trace from the beginning do:
B. Type D. This clears the existing breakpoints.
C. Type S 20. This sets a breakpoint at the first instruction.
D. Type G 20 (go to user program). TVBUG will immediately get the breakpoint and stop before executing the instruction at 20.
E. Type N. The program will execute one instruction and display all register contents. To continue, type N.
F. To trace multiple instructions type T followed by the hexadecimal number of instruction to be traced. Register contents will be displayed after execution of each instruction.
G. All Breakpoints should be deleted by typing D CR before hitting Reset, or the program will be permanently altered.

10. Offset Calculation Including Register Modification

A. Assume the SWI instruction at location 31 is to be changed to a branch always (BRA) to location 20. This will cause the program to remain in an infinite loop (i.e., the program has no end and will run continuously unless interrupted by some outside stimuli). Type M31 to open the memory location. The display = 0031 3F.
B. The op-code for a BRA is a 20, so type 20 LF. The display = 0031 3F 20.
C. The second byte of the BRA instruction should be the two's complement negative offset to location 20. Type 0.
D. TVBUG will respond with "BEG ADR?". Type in the address, 31 CR, of the BRA op-code.
E. TVBUG will respond with "END ADR?". Type in the desired branch address, 20 C/R.
F. TVBUG will respond with "OFFSET=FFED".
G. Type M 32 CR.
H. Insert the branch offset by typing the last two hex digits, ED.

11. Executing and Aborting

A. Type G 20. The program will begin executing and the TVBUG cursor will disappear since the program now contains an infinite loop.
B. Hit the break switch. This interrupts the program, displays all registers, and returns control to TVBUG.
C. Type C. Program will again continue execution.
D. Repeat B and D as many times as you wish.
E. Reset may be used to halt the program, reinitialize the screen and the I/O.

12. Punch Program to Cassette

A. Rewind the cassette.
B. Press RESET.
C. Type P.
D. Type 20 CR for the begin address.
E. Type 32 CR for the end address.
F. Type in an optional title up to 31 characters and CR.
G. Turn on the cassette player in the Record mode.
H. Wait for the prompt and cursor to reappear (approximately 60 seconds).

13. Load Program from Cassette

A. Turn off power. This will cause the program in memory to be lost. Turn power back on.
B. RESET
C. Rewind cassette.
D. Start cassette in playback mode.
E. Type L. Wait for the TVBUG prompt and an offset (0) followed by a CR. Each 'B' represents 256 bytes of data being loaded. Test the program by any of the options described above.

14. Verify Program from Cassette

A. Push Reset button and get TVBUG prompt
B. Rewind cassette
C. Start cassette in playback mode.
D. Type V and an offset (0) followed by a V.
E. Each "B" represents 256 bytes of data being verified. The TVBUG prompt represents a complete verification of the cassette tape.

5. SYSTEM EXPANSION AND APPLICATIONS

The wire wrap area may be used to implement several applications and expand the system capabilities and usefulness.

5.1 Interface EXORciser Bus

As packaged, TVBUG can fullfill a multiple of applications, but does lack one thing, and this is the necessary components to expand. Uses of this expansion can be for additional memory, ROMS, peripherals and the like.

Looking at Figure 5.1, the address lines are connected to 8T97 three-state buffers which isolate the MPU bus from the edge connector/motherboard. Because the data bus is a bi-directional bus, provision has been made to provide two-way buffering. The driver enable signal for the 8T26's is provided by an 8 input NAND gate. This gate is necessary because certain addresses must be excluded while in the READ mode. These include the TVBUG ROM (F800-FFFF), Display RAM ($D000-E3FF), F400-F7FF (I/O), $F000-F3FF (Stack) and (E800-EFFF). This last 2K slot may be used for external routines in ROM and RAM and if desired could be placed on an external card. Delete the two connections to the gate if off-board operation is desired. If non-inverted data is desired, use 8T28's in place of 8T26's. This design assumes that all user RAM will be external.

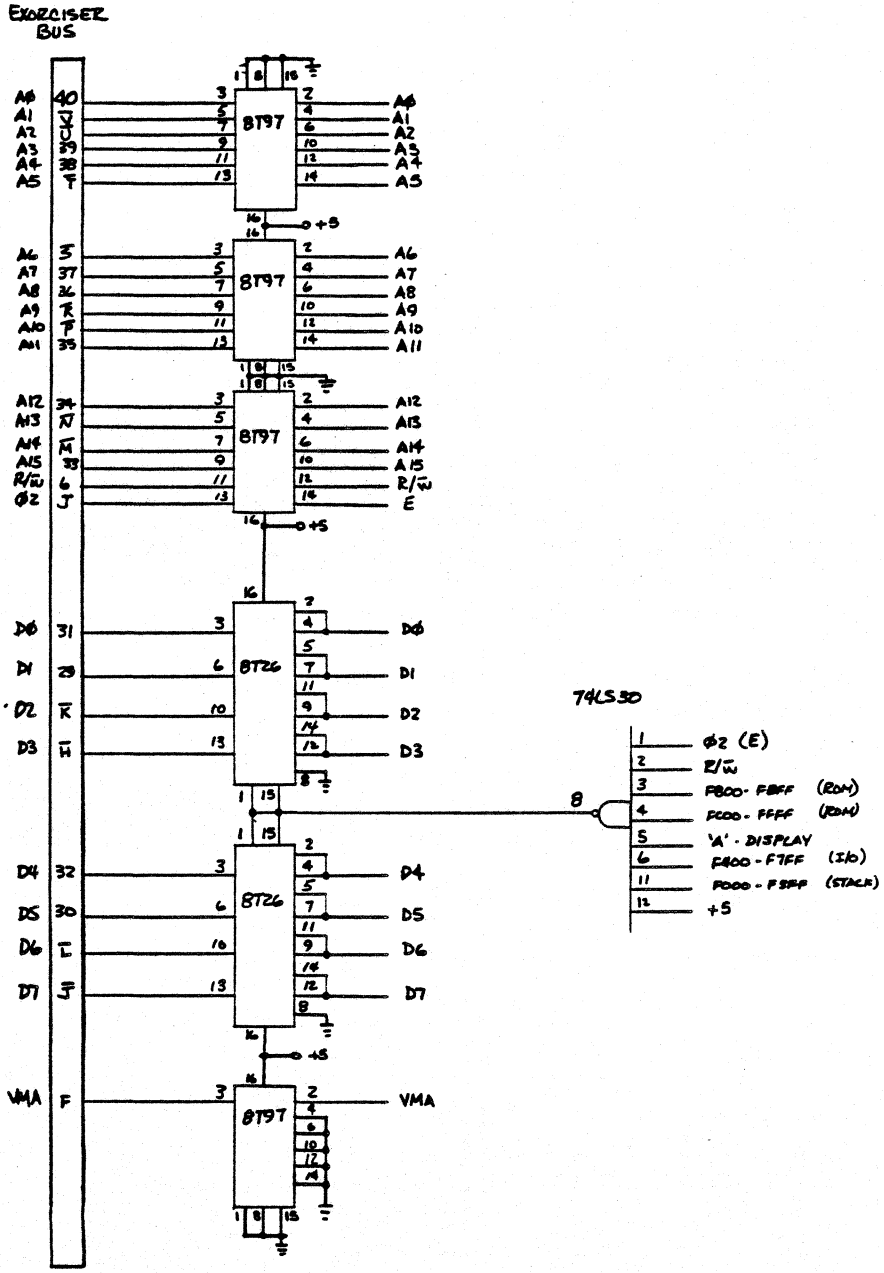


Figure 5.1 TVBUG Expansion to an Outside World

5.4 Use of MC1372

By configuring the MC1372 as a composite video generator, its output can be fed through coaxial cable to drive a remote MC1373 RF modulator. In some applications it is more advantageous to transmit a composite video signal down a line rather than an RF. See Figure 5.4.

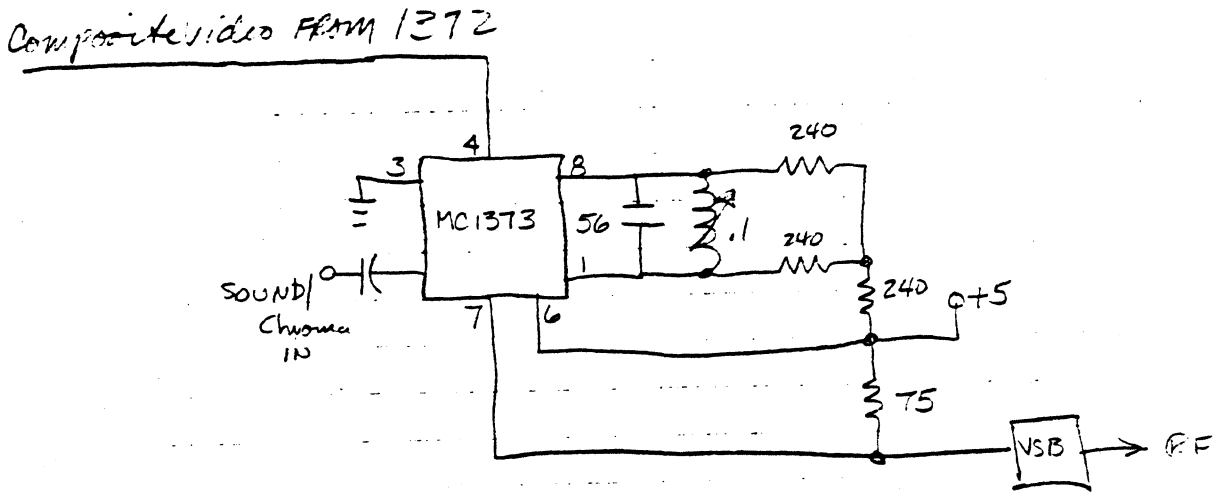
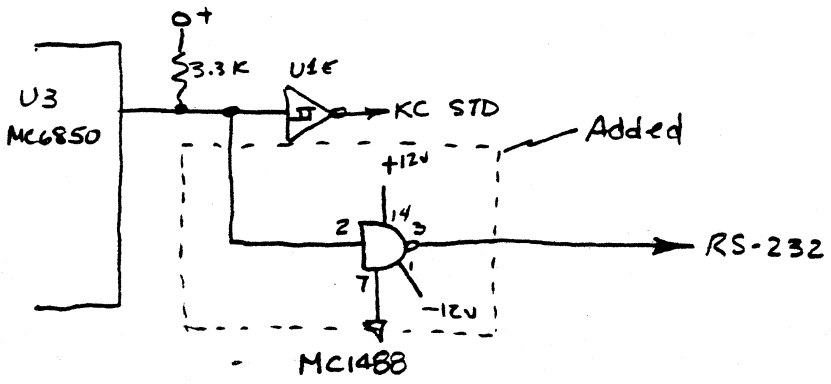


Figure 5.4 Remote RF with MC1372

5.5 RS-232 Drivers for Printer

To implement a "screen" printer for TVBUG, all that is required is to write driver software, (Appendix C) and hook up either RS-232 or TTL drivers for a printer. See Figure 5.5.



Note: See Appendix C for Software Drivers

Figure 5.5 Use of On-Board ACIA for Hard Copy (Printer)

5.5.1 Software for Printer

By using the user output routine provided in TVBUG, a character may be sent to a peripheral output device. Here's how it works: Everytime TVBUG performs an output character (OUTCH) routine, it checks three memory locations in the Stack RAM area. Since these routines are initialized to RTS, the monitor will ignore them until changed by the user. The program listing provided changes these locations to a JSR at the beginning location of the program. At this time, the OUTCH routine cycles through this additional subroutine and prints a character to whatever is connected to the on-board ACIA.

The on-board ACIA was used for the printer driver to save money, and to reduce the number of additional components. The present ACIA configuration will allow a character rate of 300 baud (from tape interface MC1455), but may be changed to allow any character rate when provided with the appropriate clock frequency (divide by 16). Because TVBUG only responds to carriage returns, a line-feed and 4 null characters are sent to the printer during a carriage return operation. This allows a printer to return fully to the left-hand most position before the continuation of printing.

It must be noted that printing will take place during punch and load operations, but will not provide the necessary CR's and null characters for proper printing. Although the use of an input device (serial or parallel) is not shown, its operation would be similar to that of the print routine. See User Input routine for further details.

5.6 S1-S9 Punch/Load Software

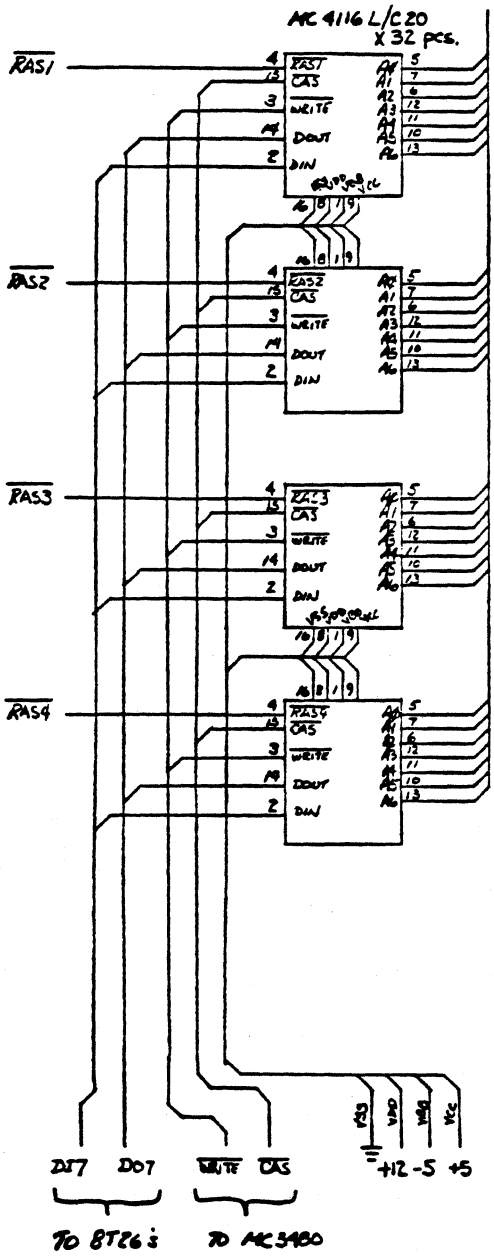
As purchased, TVBUG has the capability of loading and punching Kansas City Standard formatted tapes which utilize the MEK6800D2 binary style. Many styles of format have been used in the KC Standard, one of the most widely known is the format of MIKBUG®/MINIBUG®/EXBUG®, or the S1-S9 format. A program has been written which allows the TVBUG user to load and punch this type of tape format. See Appendix C. To use the S1-S9 system, type G $E803 CR. A prompt will ask whether to punch, load or verify. If during a load or verify operation a bad memory location is found, its address will be displayed. Each "S" displayed represents 19 characters dumped, loaded or verified.

5.7 Dynamic RAM Addition

Due to its high density and low relative cost, dynamic RAM has proven to be one of the most economical routes when designing systems which must have access to large areas of memory. TVBUG can be easily adapted to a full complement of user RAM 52K by the construction of the memory board shown in this application note.

5.7.1 4K or 16K Dynamic RAMs

4K or 16K Dynamic RAMs require a periodic "refreshing" to retain integrity of what is stored there. There are several methods of providing this refresh which include cycle stealing and transparent refresh. The board shown utilizes the transparent method and appears static to the processor. It appears static because all refresh is done during Ø1 or E low time. Fast RAMs must be used, because a refresh and access cycle must be performed in one MPU clock cycle. The system shown uses the MC3480-MC3242A combination of RAM controllers with delay lines providing the required "t" times. One shots could be utilized in place of these delay lines if they provide the correct amounts of delay. See MC3242A data sheet.



Note: Only one of eight columns are shown. All elements of Row 1 should be connected to RAS1, Row 2, to RAS2 etc. Only if not using 4K Dynamic RAMS.

Figure 5.6 Transparent Refresh Dynamic RAM Card

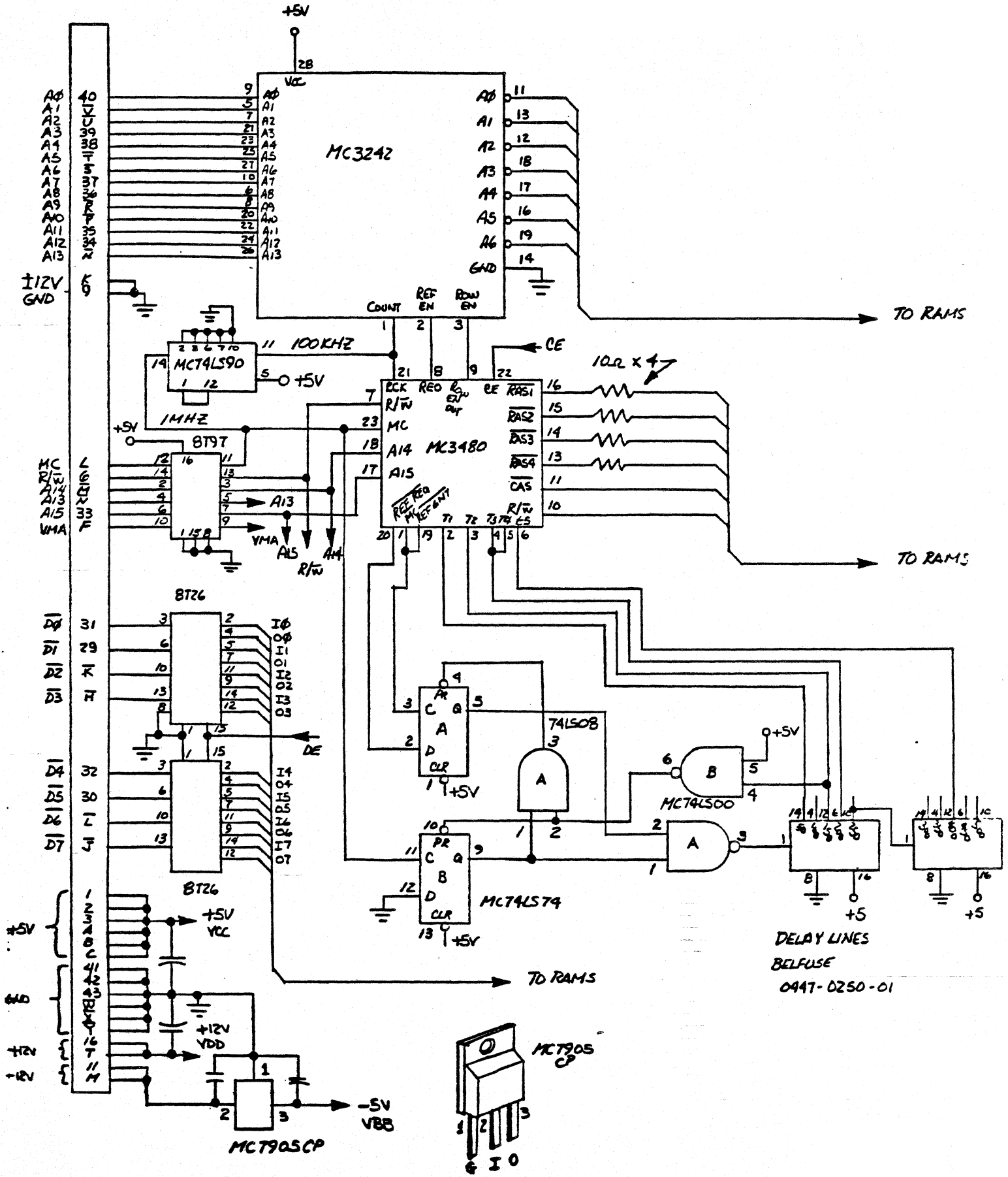


Figure 5.7 Refresh Logic

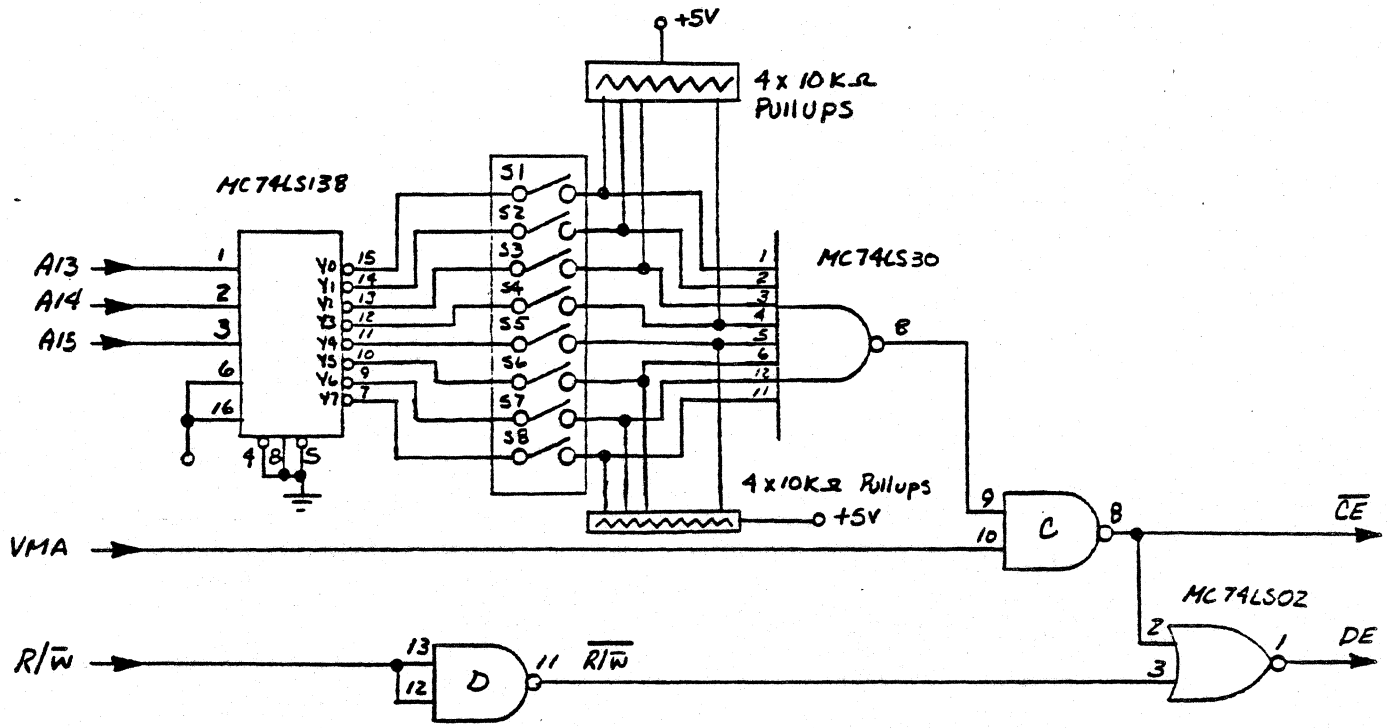
The actual schematic, and timing for the Transparent refresh method is shown in Figures 5.8-5.10. Memory clock (unstretched E) is fed into both MC3480 and SN74LS90. The LS90 is used to divide MC to give a 89 KHz pulse for the refresh clock. Although faster than the minimum 64 KHz required, this does nothing more than refresh at a higher rate. The other MC is fed into the MC3480 memory controller. Through the use of a flip flop and delay lines, the different timing requirements are provided for the MC3480. Although more expensive, the delay lines are far superior to using one shots whose external components can drift considerably with temperature variations. The MC3480 provides the RAM array with the necessary RAS and CAS signals, while the MC3242A provides the required addressing sequences, Chip enables come from the 74LS138-74LS30 combination and allow the user to have any or all 8K blocks of memory within the 6K memory map selected.

5.8 Graphic Mode Control

The Micro Chroma 68 printed circuit board has the capability of fully exploiting all VDG graphic modes. The mode select pins of the VDG are connected to the peripheral data port of the MC6846 RIOT. By writing the appropriate "word" to the port, any of the graphic modes may be selected. For example: to use the most dense (2 color) graphics mode, do the following:

```
M F441  80  00     Resets Port
  F442  00  FF     Sets as all outputs
  F443  --  39     Control word output
```

The word '39' is the control word for the most dense graphic mode. For other control words, see Figure 5.11. From this point on (initialization of the RIOT), any word placed in location $F443 will be written to the VDG. Remember that a Reset will reset the I/O port and thus require re-initilization.

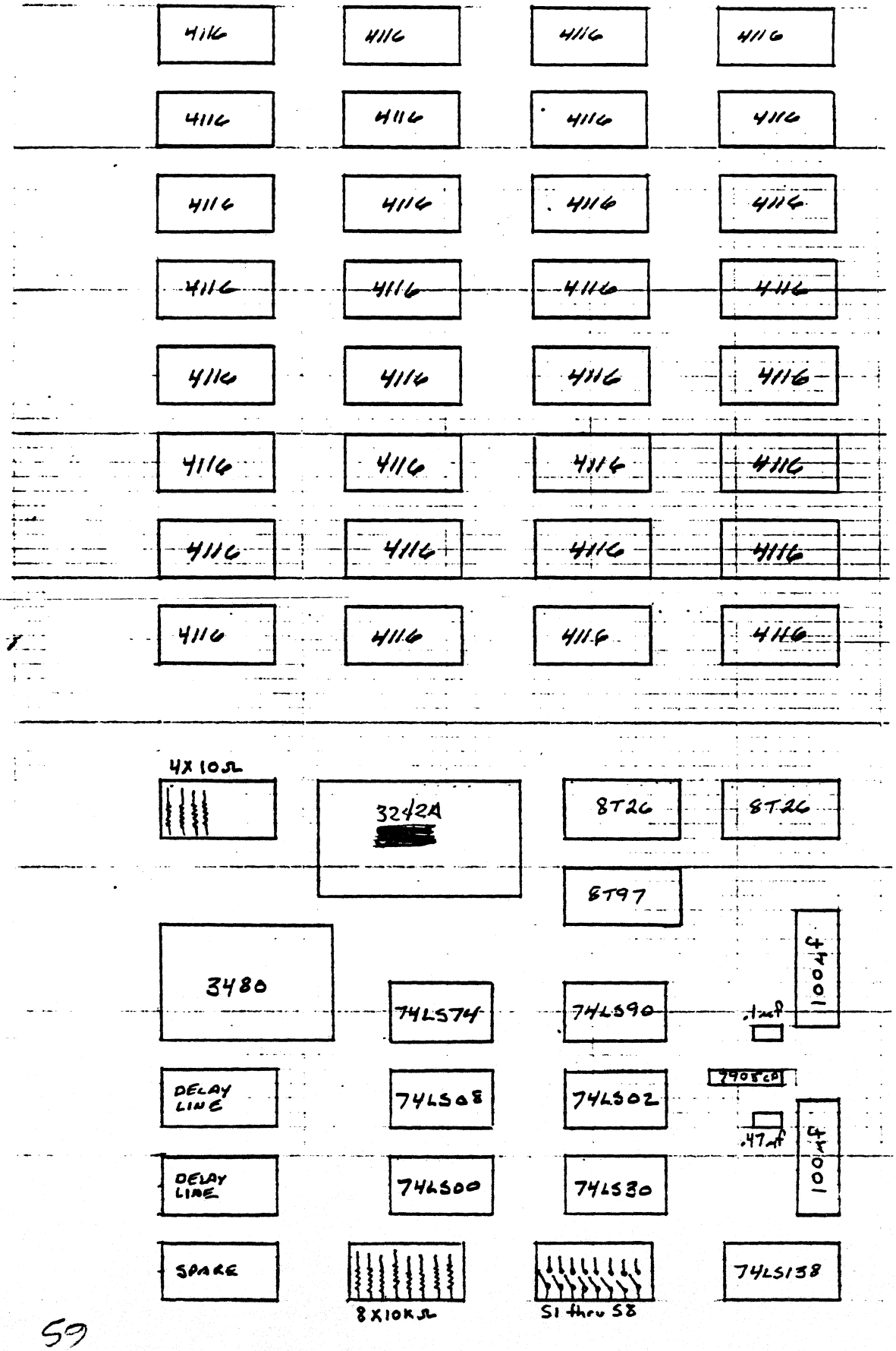


SWITCH 'ON'	BK ARRAY SELECTED
S1	0000 - 1FFF
S2	2000 - 3FFF
S3	4000 - 5FFF
S4	6000 - 7FFF
S5	8000 - 9FFF
S6	A000 - BFFF
S7	C000 - DFFF
S8	E000 - FFFF

NOTE: ANY OR ALL BANKS MAY BE SELECTED, AND MAY BE NON-CONTINUOUS.

Figure 5.8 Chip Select Logic

DYNAMIC RAM CARD



59

Figure 5.9 Suggested Layout

	DESITY	BACKGROUND	HEX VALUE TO 6846 I/O	MEMORY REQUIRED
Two Color Graphiics	256 x 192	GRN	$39	6K x 8
	256 x 192	BUFF	$38	6K x 8
Colors	128 x 192	GRN	$29	3K x 8
BUFF GREEN	128 x 192	BUFF	$28	3K x 8
	128 x 96	GRN	$19	2K x 8
	128 x 96	BUFF	$18	2K x 8
	128 x 64	GRN	$09	1K x 8
	128 x 64	BUFF	$08	1K x 8

Four Color Graphics	DENSITY	COLORS	HEX VALUE	MEMORY REQUIRED
	128 x 192	A	$31	6K x 8
	128 x 192	B	$30	6K x 8
	128 x 96	A	$21	3K x 8
COLORS	128 x 96	B	$20	3K x 8
	128 x 64	A	$11	2K x 8
A B	128 x 64	B	$10	2K x 8
Green Buff	64 x 64	A	$01	1K x 8
Yellow Cyan	64 x 64	B	$00	1K x 8
Blue Magenta				
Red Orange				

ALPHA/SEMI-GRAPHICS

	BACKGROUND	HEX VALUE	MEMORY REQUIRED
INTERNAL ROM	GRN	$47	512 x 8
	ORG	$46	512 x 8
EXTERNAL ROM	GRN	$43	512 x 8
	ORG	$42	512 x 8

ACTUAL CONNECTIONS
6846

P0	CSS
P1	A/G
P2	INT/EXT
P3	GM0
P4	GM1
P5	GM2
P6	INV
P7	N.C.

Figure 5.10 Graphic Mode Control

Appendix A

The Micro Chroma 68 Schematic, parts list, and assembly drawings and construction hints are included for reference.

Appendix B

The following software listing contains the monitor jump table and temporary RAM locations as well as other useful routines.

Note: Only the jump table and temporary RAM locations will be guaranteed on furture versions of TVBUG. Caution should be exercised when using any other routine.

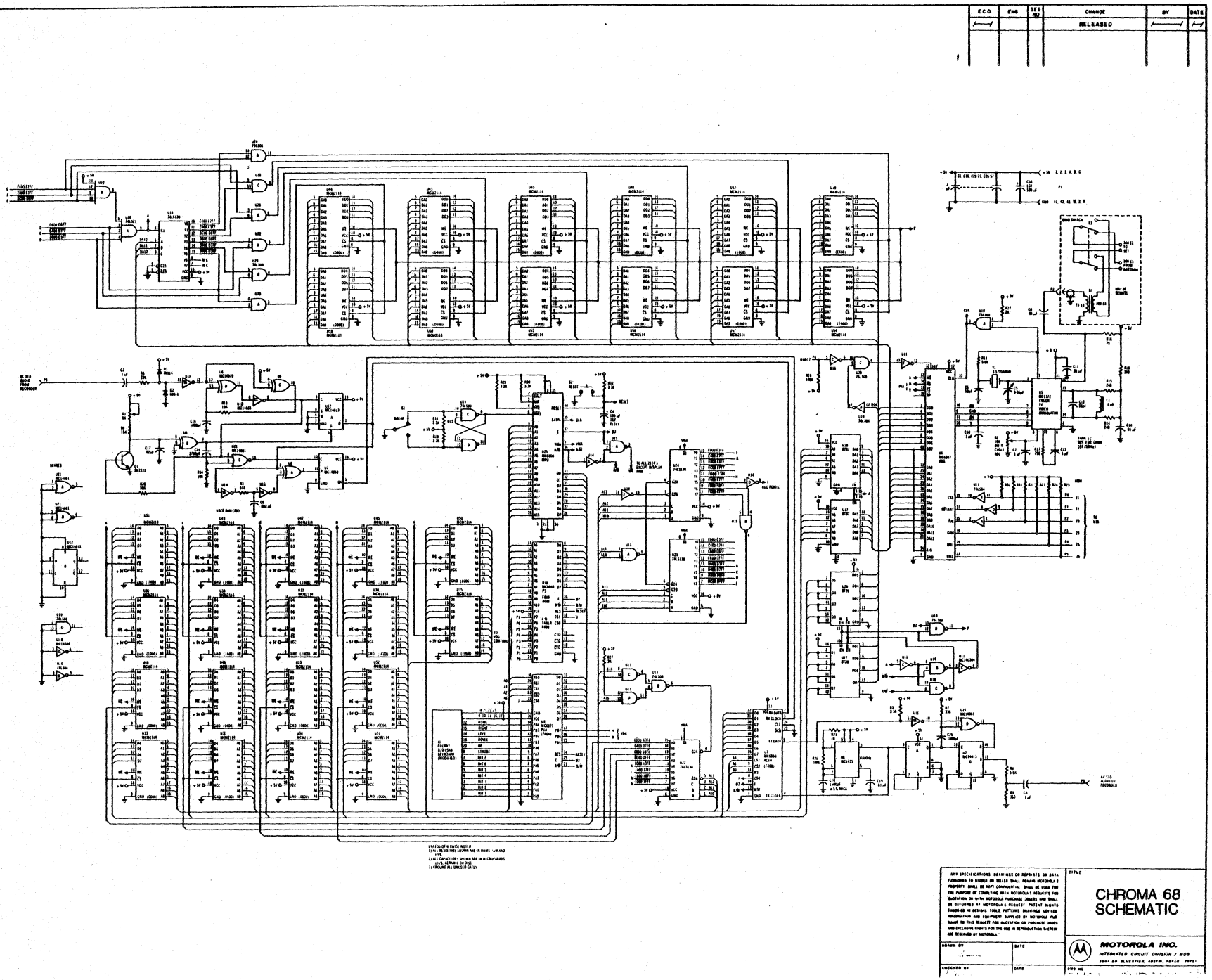
Appendix C

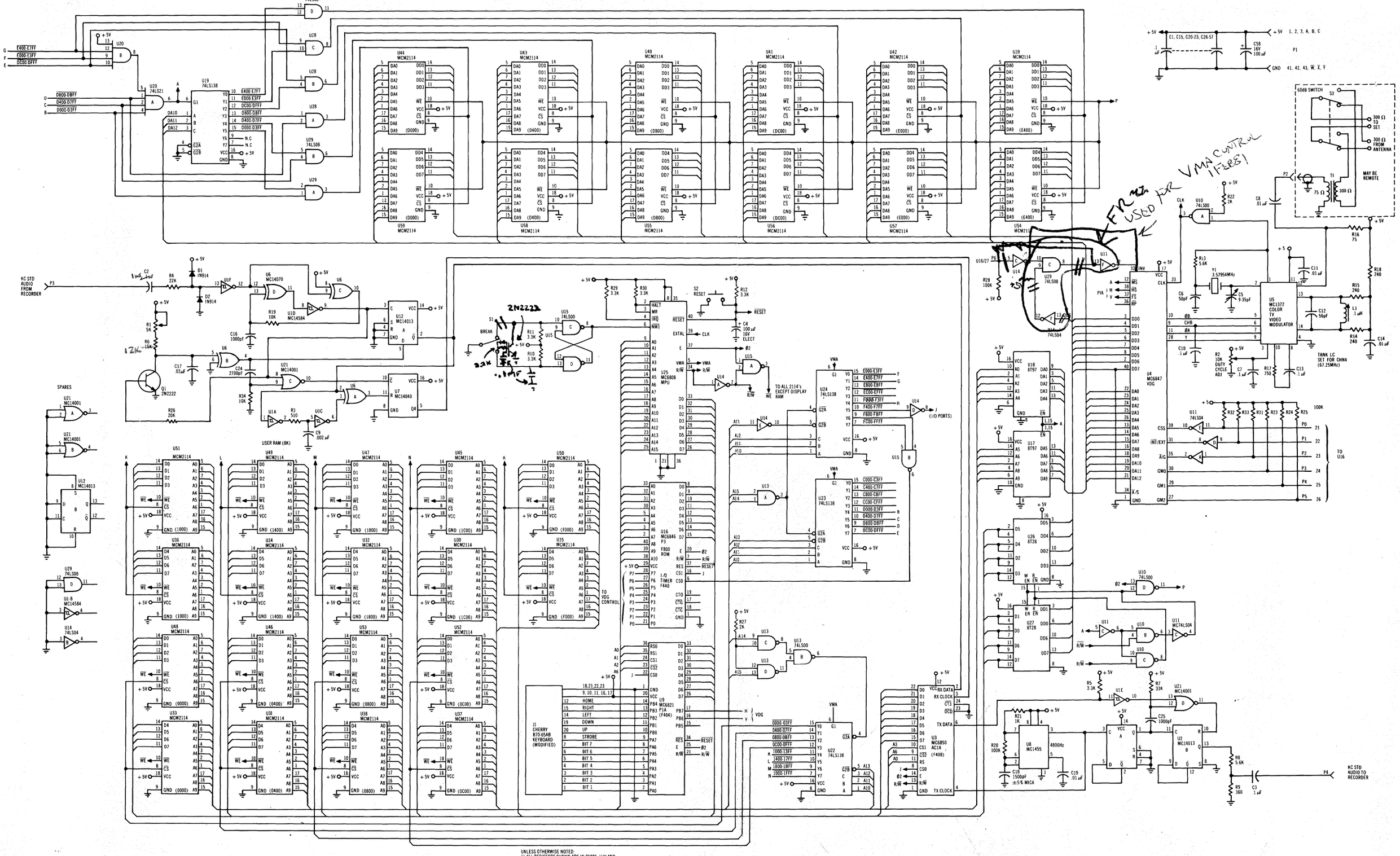
TVRTS is a listing of the RS-232 driver software and the S1-S9 Loader/Punch software.

Appendix D

The following are patches for TSC Software Packages. To covert to TVBUG, first load the original TSC Program. This can be accomplished by using the S1-S9 loader featured in Appendix C.

After loading, enter the following patches for each program. This can most easily be done by the memory Examine/Change function. Then punch on TVBUG tape.


ECO
ENG
SET NO.
CHANGE
RELEASED
BY
DATE
CHROMA 68
SCHEMATIC
MOTOROLA INC.
INTEGRATED CIRCUIT DIVISION / MOS
6501 ED BLUESTEIN, AUSTIN, TEXAS 78721
DRAWN BY
DATE
CHECKED BY
DATE



UNLESS OTHERWISE NOTED:
1) ALL RESISTORS SHOWN ARE IN OHMS, ¼W AND ±5%
2) ALL CAPACITORS SHOWN ARE IN MICROFARADS, ±10%, CERAMIC OR DISC
3) GROUND ALL UNUSED GATES

TITLE

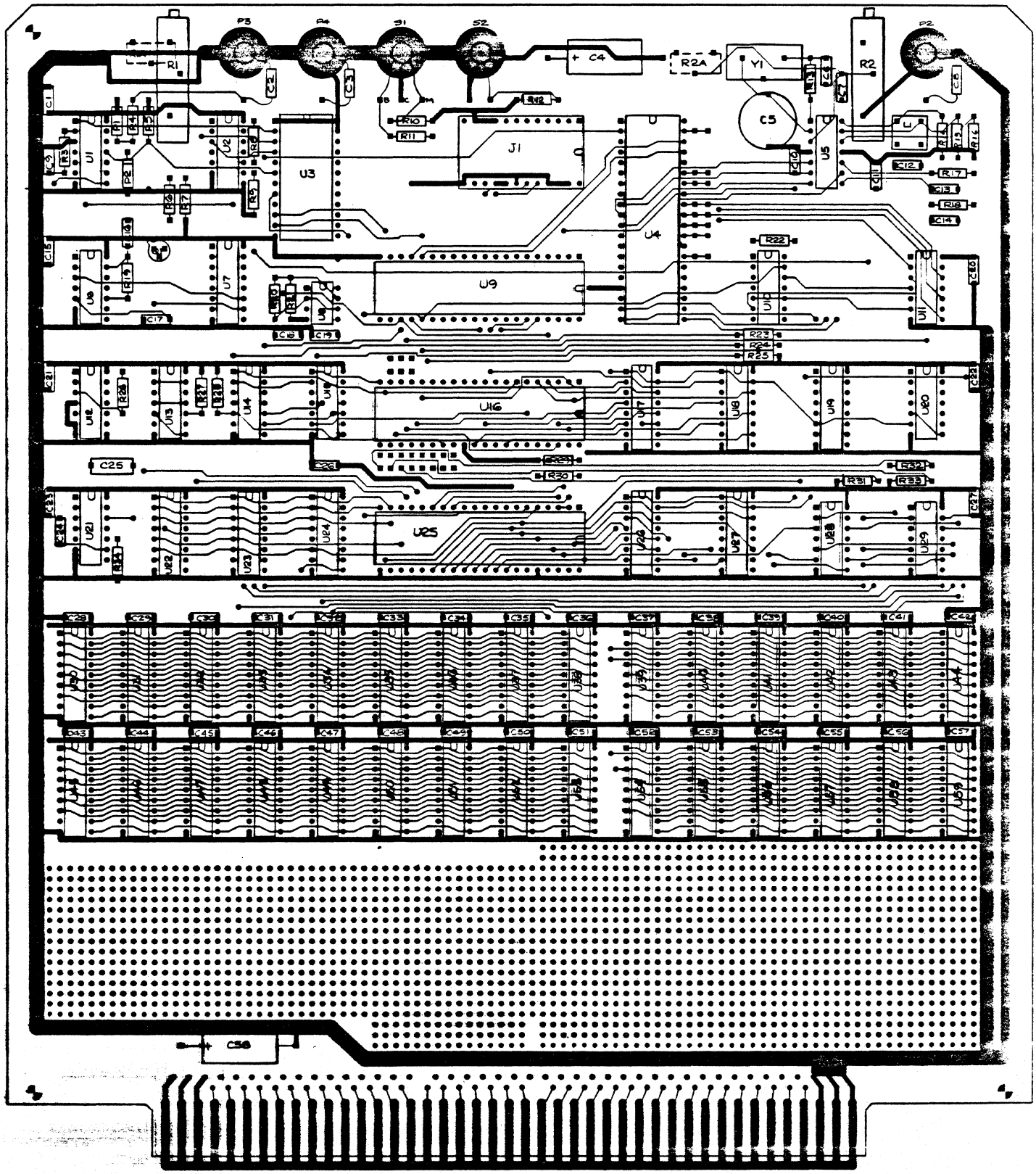
CHROMA 68 SCHEMATIC

MOTOROLA INC.
INTEGRATED CIRCUIT DIVISION / MOS
3501 ED BLUESTEIN, AUSTIN, TEXAS 78721

DRAWN BY	DATE
[illegible]	
CHECKED BY	DATE
J Ahrens	

DWG. NO.

RU614-CHROMA 68



COMPUROUTE W/O 02-7130 MOTOROLA
MICRO CHROMA 68
LAYER 1

ASSEMBLY DRAWING

MICROCHROMA 68 PARTS LIST

Capacitors

Quantity	Ref Des	Value
43	C1-C3, C7, C10, C13, C15, C20-23, C26-57	.1uF
2	C4, C58	100 uF @ 16V Electrolytic
1	C5	9-36pF Variable
1	C6	50pF
4	C8, C11, C14, C19	.01uF
1	C9	.002uF
1	C12	56pF
2	C16, C25	1000pF
1	C17	.02uF
1	C18	1500pF
1	C24	2700pF

Resistors (1/4W)

1	R1	5k Pot
1	R2	10k Pot
1	R3	510
1	R4	22k
6	R5, R10-12, R29, R30	3.3k
1	R6	15k
1	R7	33k
2	R8, R13	5.6k
1	R9	360
3	R14, R15, R18	240
1	R16	75
1	R17	750
2	R19, R34	10k
8	R20, R23-25, R28, R31-33	100k
1	R21	1k
2	R22, R27	2k
1	R26	20k

MICROCHROMA 68 PARTS LIST (CONTINUED)

Motorola Integrated Circuits

Quantity	Ref Des	P/N	Description
1	U1	MC14584B	CMOS Hex Schmitt Trigger
2	U2, U12	MC14013B	CMOS Dual D Flip-Flop
1	U3	MC6850*	NMOS Asynchronous Communications Interface Adapter (ACIA)
1	U4	MC6847*	NMOS Video Display Generator (VDG)
1	U5	MC1372*	Linear Color TV Video Modulator Circuit
1	U6	MC14070B	CMOS Quad Exclusive-OR Gate
1	U7	MC14040B	CMOS 12-Bit Binary Counter
1	U8	MC1455	Linear Timing Circuit
1	U9	MC6820/6821*	NMOS Parallel Interface Adapter (PIA)
3	U10, U13, U15	SN74LS00	TTL Quad 2-Input NAND Gate
2	U11, U14	SN74LS04	TTL Hex Inverter
1	U16	MC6846P3*	ROM, I/O, Timer (RIOT) w/TVBUG 1.2 Monitor
2	U17, U18	MC6883/8T97	Linear Hex Three-State Buffers
4	U19, U22-24	SN74LS138	TTL 3-to-8 Line Decoder
1	U20	SN74LS21	TTL Dual 4-Input AND Gate
1	U21	MC14001B	CMOS Quad 2-Input NOR Gate
1	U25	MC6808*	NMOS Microprocessor (MPU) with Clock
2	U26, U27	MC6889P/8T28	Linear Quad Bus Transceiver
2	U28, U29	SN74LS08	TTL Quad 2-Input AND Gate
30	U30, U59	MCM2114-45	NMOS 1K x 4 Static RAM

*Included in MicroChroma 68 Kits (Motorola P/N SCPROM02)

MICROCHROMA 68 KITS PARTS LIST (CONTINUED)

Miscellaneous Components

Quantity	Ref Des	Description
1	S1	Momentary SPDT (Break)
1	S2	Momentary SPST (Reset)
1	S3	DPDT Switch (60dB @ RF)
1	Y1	3.579545 MHz Crystal
1	Q1	2N2222A Transistor
1		Cherry "Pro" Keyboard (Cherry P/N B70-05AB) or equivalent with interface cable terminated with 24-pin header compatible with J1
1	L1	.1uH adjustable Inductor
1	T1	75 to 300 Matching Transformer
1	P2	Phono plug with compatible RF interconnect cable
1	P3, P4	Phono plugs with compatible audio interconnect cable
1		Vestigal Sideband Filter tuned to pass desired channel frequency
1	PCB*	MicroChroma 68 Printed Circuit Board (Motorola P/N SCPROM02PCB)
2	D1, D2	IN914 Serial Diode

* Included in MICROCHROMA 68 Kits (Motorola P/N SCPROM02PCB)

MicroChroma 68 Assembly Parts List

Capacitors

Ref Des	Value	Ref Des	Value
___ C1	.1uF	___ C30	.1uF
___ C2	.1uF	___ C31	.1uF
___ C3	.1uF	___ C32	.1uF
___ C4	100uF @ 16V Electrolytic	___ C33	.1uF
___ C5	9-35pF Variable	___ C34	.1uF
___ C6	50pF	___ C35	.1uF
___ C7	.1uF	___ C36	.1uF
___ C8	.01uF	___ C37	.1uF
___ C9	.002uF	___ C38	.1uF
___ C10	.1uF	___ C39	.1uF
___ C11	.01uF	___ C40	.1uF
___ C12	56pF	___ C41	.1uF
___ C13	.1uF	___ C42	.1uF
___ C14	.01uF	___ C43	.1uF
___ C15	.1uF	___ C44	.1uF
___ C16	1000pF	___ C45	.1uF
___ C17	.02uF	___ C46	.1uF
___ C18	1500pF	___ C47	.1uF
___ C19	.01uF	___ C48	.1uF
___ C20	.1uF	___ C49	.1uF
___ C21	.1uF	___ C50	.1uF
___ C22	.1uF	___ C51	.1uF
___ C23	.1uF	___ C52	.1uF
___ C24	2700pF	___ C53	.1uF
___ C25	1000pF	___ C54	.1uF
___ C26	.1uF	___ C55	.1uF
___ C27	.1uF	___ C56	.1uF
___ C28	.1uF	___ C57	.1uF
___ C29	.1uF	___ C58	100uF @ 16V Electrolytic

A-6

MicroChroma 68 Assembly Parts List (Continued)

Resistors (1/4W)

Ref Des	Value (Ohms)	Ref Des	Value (Ohms)
___ R1	5k Potentiometer*	___ R18	240
___ R2	10k Potentiometer*	___ R19	10k
___ R3	510	___ R20	100k
___ R4	22k	___ R21	1k
___ R5	3.3k	___ R22	2k
___ R6	15k	___ R23	100k
___ R7	33k	___ R24	100k
___ R8	5.6k	___ R25	100k
___ R9	360	___ R26	20k
___ R10	3.3k	___ R27	2k
___ R11	3.3k	___ R28	100k
___ R12	3.3k	___ R29	3.3k
___ R13	5.6k	___ R30	3.3k
___ R14	240	___ R31	100k
___ R15	240	___ R32	100k
___ R16	75	___ R33	100k
___ R17	750	___ R34	10k

*1 or 10 turn linear taper

Motorola Integrated Circuits

Ref Des	P/N	Description
___ U1	MC14584B	CMOS Hex Schmitt Trigger
___ U2	MC14013B	CMOS Dual D Flip-Flop
___ U3	MC6850**	NMOS Asynchronous Communications Interface Adaptor (ACIA)
___ U4	MC6847**	NMOS Video Display Generator (VDG)
___ U5	MC1372**	Linear Color TV Video Modulator Circuit
___ U6	MC14070B	CMOS Quad Exclusive-OR Gate
___ U7	MC14040B	CMOS 12-Bit Binary Counter
___ U8	MC1455	Linear Timing Circuit
___ U9	MC6820/MC6821**	NMOS Parallel Interface Adapter (PIA)

**Included in MicroChroma 68 Kit Motorola P/N SCPROMO2

Ref Des	P/N	Description
___ U10	SN74LS00	TTL Quad 2-Input NAND Gate
___ U11	SN74LS04	TTL Hex Inverter
___ U12	MC14013B	CMOS Dual D Flip-Flop
___ U13	SN74LS00	TTL Quad 2-Input NAND Gate
___ U14	SN74LS04	TTL Hex Inverter
___ U15	SN74LS00	TTL Quad 2-Input NAND Gate
___ U16	MC6846P3**	NMOS ROM, I/O, Timer (RIOT) with TVBUG 1.2 Monitor Program
___ U17	MC6887/MC8T97	Linear Hex Three-State Buffers
___ U18	MC6887/MC8T97	Linear Hex Three-State Buffers
___ U19	SN74LS138	TTL 3-to-8 Line Decoder
___ U20	SN74LS21	TTL Dual 4-Input AND Gate
___ U21	MC14001B	CMOS Quad 2-Input NOR Gate
___ U22	SN74LS138	TTL 3-to-8 Line Decoder
___ U23	SN74LS138	TTL 3-to-8 Line Decoder
___ U24	SN74LS138	TTL 3-to-8 Line Decoder
___ U25	MC6808**	NMOS Microprocessor (MPU) with Clock
___ U26	MC6889/MC8T28	Linear Quad Bus Transceiver
___ U27	MC6809/MC8T28	Linear Quad Bus Transceiver
___ U28	SN74LS08	TTL Quad 2-Input AND Gate
___ U29	SN74LS08	TTL Quad 2-Input AND Gate

MCM 2114-45 NMOS 1K X 4 Static RAMS

___ U30	___ U40	___ U50***
___ U31	___ U41	___ U51
___ U32	___ U42***	___ U52
___ U33***	___ U43	___ U53
___ U34	___ U44***	___ U54
___ U35***	___ U45	___ U55
___ U36	___ U46	___ U56
___ U37	___ U47	___ U57
___ U38	___ U48***	___ U58
___ U39	___ U49	___ U59***

*** These memories are required for the minimal system.

MicroChroma 68 Assembly Parts List (Continued)

Miscellaneous Components

Ref Des	Description
___ S1	Momentary SPDT (Break)
___ S2	Momentary SPST (Reset)
___ S3	DPDT Switch (60dB @ RF)
___ Y1	3.579545 MHz Crystal
___ Q1	2N2222A Transistor
___	Cherry "PRO" Keyboard (Cherry P/N B7Q-05AB) or equivalent with 24 pin header compatible with J1
___ L1	.1uH adjustable Inductor
___ T1	75 to 300 Matching Transformer
___ P2	Phono plug with compatible RF interconnect cable
___ P3	Phono plug with compatible audio interconnect cable
___ P4	Phono plug with compatible audio interconnect cable
___	Vestigal Sideband Filter tuned to pass desired channel frequency.
___ PCB **	MicroChroma 68 Printed Circuit Board (Motorola P/N SCPROM02PCB)
___ D1	IN914 Signal Diode
___ D2	IN914 Signal Diode

** Included in MicroChroma 68 Kit (Motorola P/N SCPROM02)

```
00001                    *
00002                         NAM    TVBUG
00003                         TTL    1.2 A VDG MONITOR FOR 6800,01,02,03,08 SYST
00004                    *    REV 1
00005                    *    COPYRIGHT (C) 1978 BY JOHN DUMAS
00006                    *    FOR MOTOROLA INC.
00007                    *
00008                    *    TVBUG (TM) MOTOROLA
00009                    *
00010                    *    AUSTIN, TEXAS
00011                    *    MICROCOMPUTER CAPITAL OF THE WORLD!
00012                    *
00013                    *    CURRENT REVISION DATE = NOV 20 1978
00014                    *
00015                    *  ALTHOUGH THE INFORMATION CONTAINED HEREIN,
00016                    *  AS WELL AS ANY INFORMATION PROVIDED RELATIVE
00017                    *  THERETO, HAS BEEN CAREFULLY REVIEWED AND IS
00018                    *  BELIEVED ACCURATE, MOTOROLA ASSUMES NO
00019                    *  LIABILITY ARISING OUT OF ITS APPLICATION OR
00020                    *  USE; NEITHER DOES IT CONVEY ANY LICENSE UNDER
00021                    *  ITS PATENT RIGHTS NOR THE RIGHTS OF OTHERS.
00022                    *
00023                    *    ----FOLLOWING ARE TVBUG COMMANDS----
00024                    *      EACH COMMAND IS 1 LETTER FOLLOWED
00025                    *      BY AN OPTIONAL MODIFIER (ADDRESS OR
00026                    *      DATA). MODIFIER IS ALWAYS HEX WITH
00027                    *      LEADING ZERO(S) ASSUMED.  MODIFIER
00028                    *      FIELD IS TERMINATED WITH A NON-
00029                    *      HEX ENTRY(I.E.SPACE BAR). MODIFIER
00030                    *      WILL BE EITHER 2 OR 4 HEX DEPENDING
00031                    *      UPON COMMAND MODE.
00032                    *
00033                    *    L  LOAD K.C. STANDARD TAPE (D2 FORMAT)
00034                    *    M  MEMORY CHANGE
00035                    *    P  PUNCH K.C. STANDARD TAPE (D2 FORMAT)
00036                    *    R  DISPLAY CONTENTS OF TARGET STACK
00037                    *         CC   B   A   X   P   S
00038                    *    B  PRINT OUT ALL BREAKPOINTS
00039                    *    F  FILL MEMORY BLOCK
00040                    *    C  CONTINUE EXECUTION FROM CURRENT LOCATION
00041                    *    N  NEXT INSTRUCTION TRACE
00042                    *    T  TRACE 'N' INSTRUCTIONS
00043                    *    G  GO TO LOCATION 'N'
00044                    *    D  DELETE ALL BREAKPOINTS
00045                    *    U  UNSET BREAKPOINT WITH ADDRESS 'N'
00046                    *    E  EXAMINE BLOCK OF MEMORY
00047                    *    Q  QUICK LOAD OF HEX DATA
00048                    *    O  OFFSET CALCULATION (BRANCH)
00049                    *    S  SET A BREAKPOINT WITH ADDRESS 'N'
00050                    *    V  VERIFY KC TAPE (D2 FORMAT)
00051                    *    Z  CLEAR TV SCREEN
00052                    *    !  USER FUNCTION #1  (SHIFT 1)
00053                    *    "  USER FUNCTION #2  (SHIFT 2)
00054                    *    #  USER FUNCTION #3  (SHIFT 3)
```

```
00056                    *
00057                           OPT    S,O,CREF
00058           003F   A SWI     EQU    $3F       SWI OP CODE
00059                    *
00060           D000   A VDGRAM EQU    $D000
00061                    *
00062                    * PIA FOR D2 HEX KEYBOARD & DISPLAY
00063                    * (USED IF D2 KIT IS RETROFITTED)
00064           F420   A KEYAD  EQU    $F420
00065           F421   A KEYAC  EQU    KEYAD+1
00066           F422   A KEYBD  EQU    KEYAD+2
00067           F423   A KEYBC  EQU    KEYAD+3
00068                    *
00069                    * PIA FOR VDG & ASCII KEYBRD
00070                    *
00071           F404   A PIAAD  EQU    $F404
00072           F405   A PIAAC  EQU    PIAAD+1
00073           F406   A PIABD  EQU    PIAAD+2
00074           F407   A PIABC  EQU    PIAAD+3
00075                    *
00076                    * ACIA FOR KC STANDARD TAPE INTERFACE
00077                    *
00078           F408   A ACIAS  EQU    $F408
00079           F409   A ACIAD  EQU    ACIAS+1
```

```
00081                    *
00082                    *
00083A F800                      ORG    $F800
00084          F800   A BASORG   EQU    *         BASE ORIGIN
00085                    *
00086                    * JUMP TABLE TO MONITOR
00087                    *
00088A F800 7E FEE1   A          JMP    INCH1     INPUT CHAR
00089A F803 7E FF0F   A          JMP    OUTCH1    OUTPUT CHAR
00090A F806 7E F995   A          JMP    PDATA1    OUTPUT STRING
00091A F809 7E F929   A          JMP    BADDR     INPUT HEX
00092A F80C 7E FFC9   A          JMP    SCROLL    UP 1 LINE
00093A F80F 7E F9B0   A          JMP    OUT4HS    OUTPUT 4 HEX+SPACE
00094A F812 7E F9B2   A          JMP    OUT2HS    OUTPUT 2 HEX+SPACE
00095A F815 7E FFA8   A          JMP    INIT      CLEAR SCREEN
00096A F818 7E F96C   A          JMP    GETADR    GET START & STOP ADR
00097A F81B 7E FF7F   A          JMP    SAVE      SAVE AREG 0,X
00098A F81E 7E FE86   A          JMP    SYNCLD    LOAD AREG 0,X
00099A F821 7E FB20   A          JMP    CONTRL    RESTART  POINT
00100                    *
00101                    *      I/O INTERRUPT SEQUENCE
00102                    *
00103A F824 FE F37A   A IO       LDX    IOV
00104A F827 6E 00     A          JMP    X
00105                    *
00106                    *      NMI SEQUENCE
00107                    *
00108A F829 FE F380   A POWDWN   LDX    NIO       GET NMI VECTOR
00109A F82C 6E 00     A          JMP    X
00110                    *
00111                    *      SWI INTERRUPT SEQUENCE
00112                    *
00113A F82E FE F384   A SFEI     LDX    SWI1
00114A F831 6E 00     A          JMP    X
```

```
00116                    *
00117                    * JUMP TABLE TO ROUTINES PERFORMING TVBUG FCTN'S
00118                    *
00119           F833   A FCTABL EQU    *
00120                    *
00121A F833     21     A        FCC    /!/
00122A F834     F396   A        FDB    USR1      GO USER #1
00123A F836     22     A        FCC    /"/
00124A F837     F399   A        FDB    USR2      GO USER #2
00125A F839     23     A        FCC    /#/
00126A F83A     F39C   A        FDB    USR3      GO USER #3
00127A F83C     42     A        FCC    /B/       "B" - PRINT ALL BREAKS
00128A F83D     FB52   A        FDB    PNTBRK
00129A F83F     43     A        FCC    /C/       "C" - CONTINUE
00130A F840     FB90   A        FDB    CONT
00131A F842     44     A        FCC    /D/       "D" - DELETE ALL BREAKS
00132A F843     FB47   A        FDB    DELBRK
00133A F845     47     A        FCC    /G/       "G" - GO TO ENTERED ADDRESS
00134A F846     FB61   A        FDB    GOTO
00135A F848     4C     A        FCC    /L/       "L" - LOAD
00136A F849     F9C0   A        FDB    LOAD
00137A F84B     4D     A        FCC    /M/       "M" - MEMORY CHANGE
00138A F84C     FA29   A        FDB    CHANGE
00139A F84E     4E     A        FCC    /N/       "N" - NEXT (TRACE 1 INSTR)
00140A F84F     FB73   A        FDB    NEXT
00141A F851     50     A        FCC    /P/       "P" - PUNCH
00142A F852     FC35   A        FDB    PUNCH
00143A F854     52     A        FCC    /R/       "R" - PRINT STACK
00144A F855     FB98   A        FDB    PSTAK1
00145A F857     54     A        FCC    /T/       "T" - TRACE N INSTRUCTIONS
00146A F858     FB8C   A        FDB    TRACE
00147A F85A     55     A        FCC    /U/       "U" - RESET A BREAKPOINT
00148A F85B     FB4D   A        FDB    RSTBRK
00149A F85D     53     A        FCC    /S/       "S" - SET A BREAKPOINT
00150A F85E     FB59   A        FDB    SETBRK
00151A F860     4F     A        FCC    /O/       OFFSET CALCULATION
00152A F861     FA9D   A        FDB    OFFSET
00153A F863     51     A        FCC    /Q/       QUICK LOAD
00154A F864     F8FE   A        FDB    FASTLD
00155A F866     45     A        FCC    /E/       EXAMINE BLOCK
00156A F867     F88F   A        FDB    DISPLY
00157A F869     46     A        FCC    /F/
00158A F86A     F8E3   A        FDB    FILL      FILL BLOCK
00159A F86C     56     A        FCC    /V/       VERIFY K.C. STANDARD TAPE
00160A F86D     F9B9   A        FDB    VERIFY
00161A F86F     5A     A        FCC    /Z/       CLEAR SCREEN
00162A F870     F923   A        FDB    ZSCR
00163           F872   A FCTBEN EQU    *
```

```
00165                    **********
00166                    *      INITIALIZATION/RESET CODE
00167                    * THIS DATA IS COPIED
00168                    * INTO RAM DURING START-UP INITALIZATION
00169                    *
00170           F872   A ADRSTR EQU    *
00171A F872     F36F   A        FDB    STACK     INIT FOR "SP"
00172A F874     FCE2   A        FDB    SWI1S     INIT FOR "SWI1"
00173A F876     FD0C   A        FDB    BRKINH    INIT FOR "SWI2"
00174                    *
00175A F878 20 03 F87D           BRA    BRG       "BRA" INST IS REPLACED BY
00176A F87A 7E FD4A   A          JMP    BRNOGO    COND BRA INST IN ROUT.
00177A F87D 7E FD4E   A BRG      JMP    BRGO      WHICH DETERMINES IF
00178                    *                       BRA IS GO/NOGO
00179A F880 39                   RTS
00180A F881 39                   RTS
00181A F882 39                   RTS
00182A F883 39                   RTS
00183A F884 39                   RTS
00184A F885 39                   RTS
00185A F886 7E FB20   A          JMP    CONTRL
00186A F889 7E FB20   A          JMP    CONTRL
00187A F88C 7E FB20   A          JMP    CONTRL
```

```
00189                    **********
00190                    * EXAMINE BLOCK OF MEMORY
00191                    * ACSII EQUIV PRINTED BENEATH
00192                    * HEX BYTE
00193                    *    FROM START ADR TO STOP ADR
00194                    * FORMAT :
00195                    * AAAA DD1 DD2.....DD8
00196                    *      AC1 AC2.....AC8
00197                    *
00198                    * AAAA=ADDRESS,DD=HEX DATA
00199                    * AC=ASCII CHAR (IF PRINTABLE)
00200                    **********
00201A F88F BD F96C  A DISPLY JSR    GETADR
00202                    *
00203A F892 BD F99C  A NEWLIN JSR    PCRLF
00204A F895 CE F37C  A        LDX    #BEGA     PRINT ADDRESS
00205A F898 BD F9B0  A        JSR    OUT4HS
00206                    *
00207A F89B FE F37C  A        LDX    BEGA      SET FOR 8/LINE
00208A F89E C6 08    A        LDAB   #8
00209                    *
00210A F8A0 BC F37E  A OUTDAT CPX    ENDA      MAIN LOOP
00211A F8A3 27 06 F8AB        BEQ    ASCII
00212A F8A5 BD F9B2  A        JSR    OUT2HS    PRINT HEX BYTE
00213A F8A8 5A                DECB
00214A F8A9 26 F5 F8A0        BNE    OUTDAT
00215                    *
00216                    * NOW DO ASCII
00217                    *
00218A F8AB BD F99C  A ASCII  JSR    PCRLF
00219A F8AE 8D 26 F8D6        BSR    OUT5S     SKIP ADDRESS
00220A F8B0 FE F37C  A        LDX    BEGA
00221A F8B3 C6 08    A        LDAB   #8
00222                    *
00223A F8B5 A6 00    A NEWCHR LDAA   0,X
00224A F8B7 81 1F    A        CMPA   #$1F
00225A F8B9 2F 04 F8BF        BLE    OUTPRD
00226A F8BB 81 60    A        CMPA   #$60
00227A F8BD 2D 02 F8C1        BLT    OUTASC
00228A F8BF 86 2E    A OUTPRD LDAA   #'.
00229A F8C1 8D 1D F8E0 OUTASC BSR    OUCH4
00230A F8C3 08                INX
00231A F8C4 BC F37E  A        CPX    ENDA
00232A F8C7 27 5D F926        BEQ    C3
00233                    *
00234A F8C9 8D 11 F8DC MORE   BSR    OUT2S
00235A F8CB 5A                DECB
00236A F8CC 26 E7 F8B5        BNE    NEWCHR
00237A F8CE FF F37C  A        STX    BEGA
00238A F8D1 BD FEE1  A        JSR    INCH1
00239A F8D4 20 BC F892        BRA    NEWLIN
```

```
00241                    *
0242A F8D6 8D 06 F8DE OUT5S  BSR    OUT1S
00243A F8D8 8D 04 F8DE       BSR    OUT1S
00244A F8DA 8D 02 F8DE       BSR    OUT1S
00245A F8DC 8D 00 F8DE OUT2S  BSR    OUT1S
00246A F8DE 86 20    A OUT1S  LDAA   #'
00247A F8E0 7E FF0F  A OUCH4  JMP    OUTCH1
```

```
00249                    *
00250                    **********
00251                    * FILL MEMORY BLOCK WITH KONSTANT
00252                    **********
00253                    *
00254A F8E3 BD F96C  A FILL   JSR    GETADR
00255A F8E6 CE FE63  A        LDX    #FILLMS
00256A F8E9 BD F995  A        JSR    PDATA1
00257A F8EC 8D 3B F929        BSR    BADDR
00258A F8EE B6 F3CF  A        LDAA   XLOW
00259A F8F1 FE F37C  A        LDX    BEGA
00260                    *
00261A F8F4 BC F37E  A FILL0  CPX    ENDA
00262A F8F7 27 2D F926        BEQ    C3
00263A F8F9 A7 00    A        STAA   0,X
00264A F8FB 08                INX
00265A F8FC 20 F6 F8F4        BRA    FILL0
```

```
00267                    *
00268                    * QUICK LOAD PRINTS ADDRESS ONCE
00269                    * PER LINE THEN ACCEPTS 8 HEX BYTES
00270                    * OF CODE & STORES IN MEMORY
00271                    *
00272A F8FE 8D 6C F96C FASTLD BSR    GETADR
00273                    *
00274A F900 BD F99C  A FAST0  JSR    PCRLF
00275A F903 CE F37C  A        LDX    #BEGA     PRINT ADDR
00276A F906 BD F9B0  A        JSR    OUT4HS
00277                    *
00278A F909 C6 08    A        LDAB   #8
00279                    *
00280A F90B 8D 1C F929 FAST1  BSR    BADDR
00281A F90D FE F37C  A        LDX    BEGA
00282A F910 B6 F3CF  A        LDAA   XLOW      GET BYTE &
00283A F913 A7 00    A        STAA   0,X       SAVE IT
00284A F915 08                INX
00285A F916 FF F37C  A        STX    BEGA
00286A F919 BC F37E  A        CPX    ENDA
00287A F91C 27 08 F926        BEQ    C3
00288A F91E 5A                DECB
00289A F91F 26 EA F90B        BNE    FAST1
00290A F921 20 DD F900        BRA    FAST0
00291                    *
00292                    *
00293                    *
00294A F923 BD FFA8  A ZSCR   JSR    INIT
00295A F926 7E FB20  A C3     JMP    CONTRL
```

```
00297                          **********
00298                          * BUILD ADDRESS
00299                          * A,B UNCHANGED
00300                          * INPUT HEX CHARS UNTIL NON-HEX
00301                          *    ENTERED THEN EXIT
00302                          * COUNT # OF CHAR ENTERED
00303                          *    SAVE COUNT IN CHRCNT
00304                          * STORE TERMINATION CHAR
00305                          *    IN TERMCH
00306                          * ON EXIT X = (XHI) = (TEMP2)
00307                          **********
00308A F929 7F F3C8   A BADDR  CLR    TEMP2
00309A F92C 7F F3C9   A        CLR    TEMP2+1
00310A F92F 37                 PSHB
00311A F930 36                 PSHA
00312A F931 86 FF     A        LDAA   #$FF      ZERO CHARACTER
00313A F933 B7 F3D2   A        STAA   CHRCNT    COUNTER
00314A F936 BD FEE1   A BAD0   JSR    INCH1     1 CHAR IN
00315A F939 7C F3D2   A        INC    CHRCNT    BUMP CHAR CNT
00316A F93C B7 F3D3   A        STAA   TERMCH    SAVE TERMINATION
00317                          *
00318                          * CHECK FOR VALID HEX
00319                          *
00320A F93F 80 30     A        SUBA   #$30
00321A F941 2B 20 F963         BMI    BAD2
00322A F943 81 09     A        CMPA   #9
00323A F945 2F 0A F951         BLE    BAD3
00324A F947 81 11     A        CMPA   #$11
00325A F949 2B 18 F963         BMI    BAD2
00326A F94B 81 16     A        CMPA   #$16
00327A F94D 2E 14 F963         BGT    BAD2
00328A F94F 80 07     A        SUBA   #7
00329                          *        GOOD HEX
00330A F951 48           BAD3  ASLA
00331A F952 48                 ASLA
00332A F953 48                 ASLA
00333A F954 48                 ASLA
00334A F955 C6 04     A        LDAB   #4
00335A F957 48           BAD1  ASLA
00336A F958 79 F3C9   A        ROL    TEMP2+1
00337A F95B 79 F3C8   A        ROL    TEMP2
00338A F95E 5A                 DECB
00339A F95F 26 F6 F957         BNE    BAD1
00340A F961 20 D3 F936         BRA    BAD0
00341                          *
00342A F963 FE F3C8   A BAD2   LDX    TEMP2
00343A F966 FF F3CE   A        STX    XHI
00344A F969 32                 PULA
00345A F96A 33                 PULB
00346A F96B 39                 RTS
```

```
00349                    ***********
00350                    * GET ADDRESS..SEND PROMPT
00351                    * FOR BEGIN & END ADDRESSES
00352                    * STORES IN BEGA & ENDA
00353                    * ON EXIT A=$D,B=0,X=(ENDA)
00354                    *
00355                    **********
00356A F96C CE FE27  A GETADR LDX    #MCL4
00357A F96F 8D 24 F995       BSR    PDATA1
00358A F971 8D B6 F929       BSR    BADDR
00359A F973 FF F37C  A       STX    BEGA
00360                    *
00361A F976 CE FE32  A       LDX    #MCL5
00362A F979 8D 1A F995       BSR    PDATA1
00363A F97B 8D AC F929       BSR    BADDR
00364A F97D 08               INX
00365A F97E FF F37E  A       STX    ENDA
00366A F981 39               RTS
```

```
00369                          **********
00370                          * OUTPUTS LEFT SIDE OF
00371                          * BYTE AS ASCII HEX CHAR
00372                          * ACCA IS KLOBBERED!
00373                          **********
00374A F982 44                 OUTHL  LSRA            OUT HEX LEFT BCD DIGIT
00375A F983 44                        LSRA
00376A F984 44                        LSRA
00377A F985 44                        LSRA
00378                          **********
00379                          * OUTPUTS RIGHT SIDE AS ASCII HEX
00380                          * KLOBBERS ACCA
00381                          **********


00384A F986 84 0F     A OUTHR  ANDA   #$F      OUT HEX RIGHT BCD DIGIT
00385A F988 8B 30     A        ADDA   #$30
00386A F98A 81 39     A        CMPA   #$39
00387A F98C 23 28 F9B6         BLS    OUTCH
00388A F98E 8B 07     A        ADDA   #$7
00389A F990 20 24 F9B6         BRA    OUTCH

00391                          *
00392                          *
00393                          *
00394                          **********
00395                          * PRINT DATA POINTED AT BY X-REG
00396                          * ON EXIT A=4,B=UNCHANGED, X POINTS AT EOT
00397                          **********
00398A F992 8D 22 F9B6 PDATA2  BSR    OUTCH
00399A F994 08                 INX
00400A F995 A6 00     A PDATA1 LDAA   X
00401A F997 81 04     A        CMPA   #4
00402A F999 26 F7 F992         BNE    PDATA2
00403A F99B 39                 RTS             STOP ON EOT
```

```
00405                    **********
00406                    * PRINT CR LF
00407                    * ACCA IS KLOBBERED!
00408                    **********
00409A F99C FF F3CE  A PCRLF  STX    XHI
00410A F99F 86 0D    A        LDAA   #$D
00411A F9A1 8D 13 F9B6        BSR    OUTCH
00412A F9A3 FE F3CE  A        LDX    XHI
00413A F9A6 39                RTS
00414
00415                    **********
00416                    * ON EXIT X=X+1
00417                    * BYTE IS PRINTED AS 2 HEX CHARACTERS
00418                    *
00419                    *
00420A F9A7 A6 00    A OUT2H  LDAA   0,X      OUTPUT 2 HEX CHAR
00421A F9A9 8D D7 F982        BSR    OUTHL    OUT LEFT HEX CHAR
00422A F9AB A6 00    A        LDAA   0,X      PICK UP BYTE AGAIN
00423A F9AD 08                INX
00424A F9AE 20 D6 F986        BRA    OUTHR    OUTPUT RIGHT HEX CHAR AND RTS

00426                    *
00427                    *
00428                    *
00429A F9B0 8D F5 F9A7 OUT4HS BSR    OUT2H    OUTPUT 4 HEX CHAR + SPACE
00430A F9B2 8D F3 F9A7 OUT2HS BSR    OUT2H    OUTPUT 2 HEX CHAR + SPACE
00431A F9B4 86 20    A OUTS   LDAA   #$20     SPACE
00432A F9B6 7E FF0F  A OUTCH  JMP    OUTCH1   (BSR & RTS)
00433                    *
```

```
00435                    **********
00436                    * VERIFY..SET VERIFY FLAG
00437                    * THEN GO TO LOAD
00438                    ************
00439A F9B9 86 01    A VERIFY LDAA   #1
00440A F9BB B7 F3D4  A        STAA   VFLAG
00441A F9BE 20 03 F9C3        BRA    LOAD0
00442                    *
00443                    *
00444                    **********
00445                    *LOAD MEMORY FROM KC STANDARD
00446                    * TAPE..
00447                    * BEFORE BLOCK OF DATA STARTS
00448                    *  ALL BYTES (EXCEPT RUBOUT)
00449                    *  ARE PRINTED AS ASCII CHARS
00450                    * THIS WILL DISPLAY FILE TITLE
00451                    *  IF ANY
00452                    * AFTER BLOCK BEGINS EACH
00453                    * BLOCK READ WILL PRINT A 'B' ON TV
00454                    **********
00455          F9C0  A LOAD   EQU    *
00456A F9C0 7F F3D4  A        CLR    VFLAG
00457                    *
00458A F9C3 CE FE51  A LOAD0  LDX    #OFSET
00459A F9C6 8D CD F995        BSR    PDATA1
00460A F9C8 BD F929  A        JSR    BADDR
00461A F9CB FF F3D5  A        STX    OFFADR
00462                    *
00463A F9CE 86 10    A        LDAA   #$10      DIV BY 1
00464A F9D0 B7 F408  A        STAA   ACIAS
00465A F9D3 16           BILD   TAB
00466A F9D4 8D 38 FA0E        BSR    KCIN
00467A F9D6 2B FB F9D3        BMI    BILD
00468A F9D8 8D DC F9B6        BSR    OUTCH
00469                    *
00470A F9DA C1 FF    A        CMPB   #$FF
00471A F9DC 26 F5 F9D3        BNE    BILD
00472A F9DE 81 42    A        CMPA   #'B       FIND BEGIN
00473A F9E0 27 06 F9E8        BEQ    RDBLCK
00474A F9E2 81 47    A        CMPA   #'G       FIND END
00475A F9E4 26 ED F9D3        BNE    BILD
00476A F9E6 20 7D FA65        BRA    C5
```

```
00478                    *
00479                    *
00480A F9E8 8D 24 FA0E RDBLCK BSR    KCIN
00481A F9EA 16                TAB              BYTE COUNT TO B
00482A F9EB 5C                INCB
00483A F9EC 8D 20 FA0E        BSR    KCIN      START ADDR
00484A F9EE B7 F37C  A        STAA   BEGA
00485A F9F1 8D 1B FA0E        BSR    KCIN
00486A F9F3 B7 F37D  A        STAA   BEGA+1
00487A F9F6 FE F37C  A        LDX    BEGA      TO X
00488A F9F9 8D 70 FA6B        BSR    ADDOFF
00489                    *
00490A F9FB 8D 11 FA0E STBLCK BSR    KCIN
00491A F9FD 7D F3D4  A        TST    VFLAG
00492A FA00 26 02 FA04        BNE    LOAD1
00493A FA02 A7 00    A        STAA   0,X
00494A FA04 A1 00    A LOAD1  CMPA   0,X
00495A FA06 26 58 FA60        BNE    LOAD19
00496A FA08 08                INX
00497A FA09 5A                DECB             BYTE COUNT -1
00498A FA0A 26 EF F9FB        BNE    STBLCK
00499A FA0C 20 C5 F9D3        BRA    BILD
```

```
00501                    *
00502                    * 1 CHAR IN FROM TAPE
00503                    *
00504A FA0E 8D 0A FA1A KCIN   BSR    CHKESC
00505A FA10 B6 F408  A        LDAA   ACIAS
00506A FA13 47                ASRA
00507A FA14 24 F8 FA0E        BCC    KCIN
00508A FA16 B6 F409  A        LDAA   ACIAD
00509A FA19 39                RTS
00510                    *
00511                    * CHECK FOR ESCAPE KEY
00512                    *
00513                    *
00514                    *
00515A FA1A 36           CHKESC PSHA
00516A FA1B B6 F404  A        LDAA   PIAAD
00517A FA1E 84 7F    A        ANDA   #$7F
00518A FA20 81 1B    A        CMPA   #$1B
00519A FA22 26 03 FA27        BNE    CHK1
00520A FA24 7E FAF3  A        JMP    ESC1
00521A FA27 32           CHK1   PULA
00522A FA28 39                RTS
00523                    *
```

```
00525                    *
00526                    * CHANGE MEMORY (M AAAA DD NN)
00527                    *
00528A FA29 BD F929  A CHANGE JSR    BADDR    BUILD ADDRESS
00529A FA2C BD F99C  A CHG0   JSR    PCRLF    START NEW LINE
00530A FA2F CE F3CE  A        LDX    #XHI     & PRINT ADDR
00531A FA32 BD F9B0  A        JSR    OUT4HS
00532                    *
00533A FA35 FE F3CE  A        LDX    XHI      PRINT OLD
00534A FA38 BD F9B2  A        JSR    OUT2HS   CONTENTS
00535A FA3B 09                DEX
00536                    *
00537A FA3C BD F929  A        JSR    BADDR
00538A FA3F FE F3CA  A        LDX    SAVEX    GOT NEW
00539                    *
00540A FA42 7D F3D2  A        TST    CHRCNT   NEW DATA??
00541A FA45 27 09 FA50        BEQ    SKPSTR   NO! SKIP LOAD
00542A FA47 B6 F3CF  A        LDAA   XLOW     NO PUT IN
00543A FA4A A7 00    A        STAA   0,X      NEW DATA &
00544A FA4C A1 00    A        CMPA   0,X      CHECK
00545A FA4E 26 10 FA60        BNE    LOAD19   BAD..QUIT
00546                    *
00547A FA50 B6 F3D3  A SKPSTR LDAA   TERMCH    GOOD ST
00548A FA53 08                INX              NEXT?
00549A FA54 81 0A    A        CMPA   #$0A      IF LF..YES
00550A FA56 27 D4 FA2C        BEQ    CHG0     NEXT ADDR
00551                    *
00552A FA58 81 5E    A        CMPA   #';       IF UP ARROW - DECREMENT ADDR
00553A FA5A 26 09 FA65        BNE    C5
00554A FA5C 09                DEX
00555A FA5D 09                DEX
00556A FA5E 20 CC FA2C        BRA    CHG0     LAST ADDR
00557A FA60 CE FE3C  A LOAD19 LDX    #NCHG
00558A FA63 8D 6F FAD4        BSR    PD2
00559A FA65 7E FB20  A C5     JMP    CONTRL
00560A FA68 7E FF0F  A OUTCH5 JMP    OUTCH1
```

```
00562                          **********
00563                          * ADD OFFSET TO VALUE IN X-REG
00564                          * A & B UNCHANGED
00565                          * OFFSET IS IN B,A
00566                          **********
00567A FA6B 36                 ADDOFF PSHA
00568A FA6C 37                        PSHB
00569A FA6D FF F3C8  A                STX    TEMP2
00570                          *
00571A FA70 F6 F3C8  A                LDAB   TEMP2
00572A FA73 B6 F3C9  A                LDAA   TEMP2+1
00573A FA76 BB F3D6  A                ADDA   OFFADR+1
00574A FA79 F9 F3D5  A                ADCB   OFFADR
00575A FA7C F7 F3C8  A                STAB   TEMP2
00576A FA7F B7 F3C9  A                STAA   TEMP2+1
00577                          *
00578A FA82 FE F3C8  A                LDX    TEMP2
00579A FA85 33                        PULB
00580A FA86 32                        PULA
00581A FA87 39                        RTS
```

```
00583                    *
00584                    * NMI ENTRY
00585                    *
00586A FA88 BF F382   A NMI   STS     SP
00587A FA8B BD F99C   A       JSR     PCRLF
00588A FA8E 86 42     A       LDAA    #'B
00589A FA90 8D D6 FA68        BSR     OUTCH5
00590A FA92 BD F9B4   A       JSR     OUTS
00591A FA95 86 02     A       LDAA    #2
00592A FA97 BD FBA6   A       JSR     BRKSUB
00593A FA9A 7E FB98   A       JMP     PSTAK1
```

```
00595                          *
00596                          * OFFSET CALCULATES BRANCH
00597                          *
00598A FA9D BD F96C  A OFFSET  JSR    GETADR
00599A FAA0 CE FE51  A         LDX    #OFSET
00600A FAA3 8D 2F FAD4         BSR    PD2        SEND MESSAGE
00601                          *
00602A FAA5 B6 F37F  A         LDAA   ENDA+1     CALCULATE
00603A FAA8 F6 F37E  A         LDAB   ENDA       OFFSET
00604A FAAB B0 F37D  A         SUBA   BEGA+1
00605A FAAE F2 F37C  A         SBCB   BEGA
00606                          *
00607A FAB1 80 03    A         SUBA   #3         AND ADJUST
00608A FAB3 C2 00    A         SBCB   #0
00609A FAB5 B7 F3C7  A         STAA   TEMP+1
00610A FAB8 F7 F3C6  A         STAB   TEMP
00611A FABB CE F3C6  A         LDX    #TEMP      PRINT ADDR
00612A FABE 36                 PSHA
00613A FABF BD F9B0  A         JSR    OUT4HS
00614                          *
00615                          * CHECK PROPER RANGE
00616                          *
00617A FAC2 32                 PULA
00618A FAC3 49                 ROLA
00619A FAC4 59                 ROLB
00620A FAC5 C1 00    A         CMPB   #0
00621A FAC7 27 9C FA65         BEQ    C5
00622A FAC9 C1 FF    A         CMPB   #$FF
00623A FACB 27 98 FA65         BEQ    C5
00624A FACD CE FE5A  A         LDX    #BADJP     TELL EM
00625A FAD0 8D 02 FAD4         BSR    PD2        IT'S BAD
00626A FAD2 20 91 FA65         BRA    C5
00627                          *
00628A FAD4 7E F995  A PD2     JMP    PDATA1
```

```
00630                    *
00631                    *
00632                    * CONSTANT INITIALIZATION
00633                    * S = POINTER TO ROM BYTES TO BE COPIED TO RAM
00634                    * X = POINTER TO RAM BYTES TO BE INITIALIZED
00635                    *
00636          FAD7  A START  EQU    *          ACTUAL CODE START
00637A FAD7 8E F871  A         LDS    #ADRSTR-1 START OF CONSTANT DATA
00638A FADA CE F382  A         LDX    #SP       START OF RAM AREA
00639                    *
00640A FADD 32           INILP1 PULA              GET NEXT CONSTANT BYTE
00641A FADE A7 00    A         STAA   0,X       INIT NEXT RAM BYTE
00642A FAE0 08                 INX              UPDATE POINTER
00643A FAE1 8C F3A0  A         CPX    #BRANEN   END OF CONSTANT RAM AREA?
00644A FAE4 26 F7 FADD         BNE    INILP1    NO, CONTINUE INITIALIZATION
00645                    *
00646                    * INITIALIZATION TO 0
00647                    * X HOLDS INDEX OF 1ST BYTE TO BE SET TO 0
00648                    *
00649A FAE6 6F 00    A INILP2  CLR    0,X       CLEAR NEXT BYTE OF RAM
00650A FAE8 08                 INX              UPDATE INDEX
00651A FAE9 8C F3DA  A         CPX    #ENDIN0   ANY MORE BYTES TO INIT?
00652A FAEC 26 F8 FAE6         BNE    INILP2    NO, CONTINUE CLEARING
00653                    *
00654                    * SET CC SO WHEN WE 'GO' TO USER PGM THE
00655                    *   INTERUPT MASK IS SET
00656                    *
00657A FAEE 86 D0    A         LDAA   #$D0
00658A FAF0 B7 F370  A         STAA   STACK+1   PUT IN STACK TO BE PULLED
```

```
00660                    *
00661                    * INITIALIZE KC TAPE ACIA
00662                    * & KEYBOARD PIA
00663                    *
00664A FAF3 BE F382   A ESC1   LDS    SP
00665A FAF6 86 3C     A        LDAA   #$3C
00666A FAF8 B7 F421   A        STAA   KEYAC     KILL HEX KEYBRD & TRACE
00667A FAFB B7 F423   A        STAA   KEYBC     FOR D2 KIT
00668                    *
00669A FAFE 86 03     A        LDAA   #3
00670A FB00 B7 F408   A        STAA   ACIAS     KILL KC TAPE
00671                    *
00672                    * CONFIGURE PIA FOR VDG & ASCII KBD
00673                    *
00674A FB03 7F F406   A        CLR    PIABD
00675A FB06 7F F404   A        CLR    PIAAD
00676A FB09 86 04     A        LDAA   #4
00677A FB0B B7 F405   A        STAA   PIAAC
00678A FB0E B7 F407   A        STAA   PIABC
00679                    *
00680A FB11 BD FFA8   A        JSR    INIT      CLEAR SCREEN
00681A FB14 CE FDDD   A        LDX    #MCL2     PRINT HEADER
00682A FB17 BD F995   A        JSR    PDATA1    PRINT DATA STRING
00683A FB1A CE FA88   A        LDX    #NMI      INIT PDN
00684A FB1D FF F380   A        STX    NIO
00685                    *
```

```
00687                    *
00688                    * MAIN COMMAND/CONTROL LOOP
00689                    *
00690           FB20   A CONTRL EQU    *
00691                    *
00692                    * RESTORE STACK POINTER REGISTER
00693                    *
00694A FB20 BE F382   A         LDS    SP        SP WAS INITIALIZED EARLIER
00695A FB23 CE FE49   A         LDX    #READY
00696A FB26 8D AC FAD4          BSR    PD2
00697                    *
00698                    *
00699                    *
00700A FB28 BD FEE1   A         JSR    INCH1     READ COMMAND CHARACTER
00701A FB2B 16                  TAB               SAVE CHARACTER IN B
00702A FB2C BD F9B4   A         JSR    OUTS      PRINT SPACE AFTER COMMAND
00703                    *
00704                    * B REGISTER HOLDS CHARACTER INPUT BY USER.
00705                    * USE JUMP TABLE TO GO TO APPROPRIATE ROUTINE.
00706                    *
00707A FB2F CE F833   A         LDX    #FCTABL   X:= ADDRESS OF JUMP TABLE
00708A FB32 E1 00     A NXTCHR  CMPB   0,X       DOES INPUT CHAR MATCH?
00709A FB34 27 0A FB40          BEQ    GOODCH    YES, GOTO APPROPRIATE ROUTINE
00710A FB36 08                  INX               ELSE, UPDATE INDEX INTO TABLE
00711A FB37 08                  INX
00712A FB38 08                  INX
00713A FB39 8C F872   A         CPX    #FCTBEN   END OF TABLE REACHED?
00714A FB3C 26 F4 FB32          BNE    NXTCHR    NO, TRY NEXT CHAR
00715A FB3E 20 E0 FB20          BRA    CONTRL    NO MATCH, REPROMPT USER
00716                    *
00717                    *
00718A FB40 EE 01     A GOODCH  LDX    1,X       GET ADDRESS FROM J.T.
00719A FB42 6E 00     A         JMP    0,X       GOTO APPROPRIATE ROUTINE
00720                    *
00721                    *

00723                    *
00724                    *
00725A FB44 7E F929   A BADDRJ  JMP    BADDR     GO BUILD ADDRESS
```

```
00727                    *
00728                    *
00729                    * RESET ALL BREAKPOINTS
00730                    *
00731A FB47 86 01    A DELBRK LDAA   #1        RESET BREAKS FLAG
00732A FB49 8D 5B FBA6 BSRBRK BSR    BRKSUB    BREAK HANDLING SUBR.
00733A FB4B 20 56 FBA3        BRA    CNTRL2    RETURN TO COMMAND LEVEL
00734                    *
00735                    * RESET 1 BREAKPOINT
00736                    *
00737A FB4D 8D F5 FB44 RSTBRK BSR    BADDRJ    PUTS USER ENTERED ADDRESS
00738                    *                          INTO XHI,XLOW
00739A FB4F 4F                CLRA             RESET 1 BREAK FLAG
00740A FB50 20 F7 FB49        BRA    BSRBRK    GO RESET 1
00741                    *
00742                    * PRINT OUT ALL NON-ZERO BREAK ADDRESSES
00743                    *
00744A FB52 BD F99C  A PNTBRK JSR    PCRLF     DO CR/LF
00745A FB55 86 02    A        LDAA   #2        PRINT BREAK ADDRESSES FLAGS
00746A FB57 20 F0 FB49        BRA    BSRBRK    GO PRINT
00747                    *
00748                    * SET ONE BREAK
00749                    *
00750A FB59 8D E9 FB44 SETBRK BSR    BADDRJ    GET USER ENTERED ADDRESS (XHI,XLOW
00751A FB5B 86 04    A        LDAA   #4        SET ONE BREAK FLAG
00752A FB5D 8D 47 FBA6        BSR    BRKSUB    GO SET IT
00753A FB5F 20 F1 FB52        BRA    PNTBRK    PRINT ALL BREAKPOINTS
```

```
00755                    *
00756                    * GO TO REQUESTED
00757                    *
00758A FB61 8D E1 FB44 GOTO   BSR    BADDRJ   GO GET ADDRESS FROM USER
00759                    *                     XHI,XLOW HOLD ADDRESS
00760A FB63 86 FF    A         LDAA   #$FF     FLAG FOR PUTTING IN BREAKS
00761A FB65 8D 3F FBA6         BSR    BRKSUB   GO PUT IN BREAKS
00762A FB67 30                 TSX
00763A FB68 B6 F3CE  A         LDAA   XHI      SAVE PCH ON STACK
00764A FB6B A7 05    A         STAA   5,X
00765A FB6D B6 F3CF  A         LDAA   XLOW     PSH PCL
00766A FB70 A7 06    A         STAA   6,X
00767A FB72 3B                 RTI            GO TO USER PRG
00768                    *
00769                    * SINGLE INSTRUCTION TRACE REQUESTED
00770                    *
00771A FB73 CE 0001  A NEXT   LDX    #1       # INSTRUCTIONS TO TRACE
00772A FB76 7F F3D9  A TRACE2 CLR    BRKTRC   CLEAR FLAG INDICATING TRACE
00773                    *                     IS DUE TO BREAK
00774A FB79 FF F3A3  A TRACE3 STX    NTRACE   SAVE # INST'S TO TRACE
00775A FB7C FE F382  A         LDX    SP       X : = STACK POINTER
00776A FB7F EE 06    A         LDX    6,X      X : = INSTR TO BE EXECUTED
00777A FB81 FF F3A0  A         STX    TRCADR   SAVE IN TRACE ADDRESS STORE
00778A FB84 A6 00    A         LDAA   0,X      GET INSTRUCTION TO BE TRACED
00779A FB86 B7 F3A2  A         STAA   TRCINS   SAVE IN INSTRUCTION STORE
00780A FB89 7E FD37  A         JMP    CONTRC   CONTINUE TRACE PART OF  PROGRAM
00781                    *
00782                    * MULTIPLE INSTRUCTION TRACE
00783                    *
00784A FB8C 8D B6 FB44 TRACE  BSR    BADDRJ   GET # OF INSTRUCTIONS TO TRACE
00785A FB8E 20 E6 FB76         BRA    TRACE2   GO TRACE'M
00786                    *
00787                    * CONTINUE EXECUTION
00788                    *
00789A FB90 7C F3D9  A CONT   INC    BRKTRC   TRACE 1 TO RESTORE SWI'S
00790A FB93 CE 0001  A         LDX    #1       ONE TRACE ONLY
00791A FB96 20 E1 FB79         BRA    TRACE3
00792                    *
00793                    *
00794                    *  R COMMAND
00795                    *
00796                    * PRINT STACK CONTENTS
00797                    *
00798A FB98 BD F99C  A PSTAK1 JSR    PCRLF    PRINT CR LF
00799A FB9B CE FE11  A         LDX    #MCL3    PRINT HEADER
00800A FB9E BD F995  A         JSR    PDATA1
00801A FBA1 8D 7B FC1E         BSR    PRINT    PRINT STACK
00802A FBA3 7E FB20  A CNTRL2 JMP    CONTRL   RETURN TO COMMAND LEVEL
```

```
00804                    **********
00805                    *
00806                    * BRKSUB
00807                    *
00808                    *
00809                    * THIS ROUTINE DOES A NUMBER OF OPERATIONS HAVING
00810                    * TO DO WITH BREAKPOINTS.
00811                    *
00812                    * THE A REGISTER DETERMINES FUNCTION PERFORMED:
00813                    *
00814                    * A = -1 =" BREAKS ARE PUT INTO USER'S CODE
00815                    * A =  0 =" THE BREAKPOINT WHOSE ADDRESS IS IN
00816                    *           XHI, XLOW IS PURGED;
00817                    *           ALL BREAKPOINTS ARE TEMPORARILY REMOVED
00818                    * A =  1 =" ALL BREAKPOINTS ARE PURGED
00819                    * A =  2 =" ALL BREAKPOINTS ARE PRINTED OUT
00820                    *           ALL BREAKPOINTS ARE TEMPORARILY REMOVED
00821                    * A =  3 =" ALL BREAKPOINTS ARE TEMPORARILY REMOVED
00822                    * A =  4 =" THE BREAK ADDRESS IN XHI, XLOW IS
00823                    *           PUT INTO THE FIRST ZERO BREAKPOINT
00824                    *           POSITION; ALL BREAKS ARE TEMPORARILY REMOVED
00825                    *
00826                    **********
00827                    *
00828          FBA6  A BRKSUB EQU    *
00829A FBA6 BF F3D0  A        STS    SSAVE    SAVE S SO WE CAN USE
00830A FBA9 B7 F3C5  A        STAA   ASAVE    A HOLDS THE FUNCTION #
00831                    *
00832A FBAC CE F3A5  A        LDX    #BRKADR  INIT X FOR LOOP THROUGH BREAKS
00833                    *
00834                    * START OF LOOP THROUGH BREAK ADDRESSES
00835                    *
00836A FBAF B6 F3C5  A BRKLP  LDAA   ASAVE    GET FUNCTION #
00837A FBB2 AE 00    A        LDS    0,X      S:=NEXT ADDRESS IN BRKPT LIST
00838A FBB4 27 2D FBE3        BEQ    LN       IF 0, THEN NOT A VALID BREAK
00839                    *
00840A FBB6 7D F3D8  A        TST    BRKSIN   ARE BREAKS IN USER'S CODE?
00841A FBB9 27 36 FBF1        BEQ    NOBRIN   BRANCH, IF NOT
00842                    *
00843                    * BREAKS ARE IN USER'S CODE
00844                    *
00845A FBBB 4D                TSTA            SHOULD BREAKS BE IN?
00846A FBBC 2B 21 FBDF        BMI    BKDONE   YES, RETURN TO CALLER
00847                    *
00848                    * BREAKS ARE TO BE TAKEN OUT OF USER'S
00849                    * CODE TEMPORARILY
00850                    *
00851A FBBE A6 10    A        LDAA   2*NBRBPT,X GET INSTR. BELONG-
00852                    *                          ING IN USER CODE
00853A FBC0 36                PSHA            PUT IT THERE
```

```
00855                    *
00856                    * OTHER ACTIONS TO BE PERFORMED EACH TIME THROUGH
00857                    * LOOP WHEN BREAK ADDRESS NOT EQUAL TO 0.
00858                    *
00859A FBC1 B6 F3C5  A BKCON1 LDAA   ASAVE    # OF B
00860A FBC4 27 37 FBFD        BEQ    FNDRPL   SEE IF BREAKPOINT NEEDS TO
00861                    *                          BE REPLACED
00862A FBC6 81 01    A        CMPA   #1       IS BRK ADDRESS TO BE RESET?
00863A FBC8 27 41 FC0B        BEQ    CLRBRK   YES, SET BRKADR TO 0
00864                    *
00865A FBCA 81 02    A        CMPA   #2       IS BRK ADDR TO BE PRINTED?
00866A FBCC 27 49 FC17        BEQ    PRNTBK   YES, GO PRINT ADDRESS
00867                    *
00868                    * UPDATE LOOP INDEX AND LOOP IF APPROPRIATE
00869                    *
00870A FBCE 08           BKCON2 INX              MAKE X POINT TO
00871A FBCF 08                  INX              NEXT BREAK ADDRESS
00872A FBD0 8C F3B5  A BKCON3 CPX    #BRKINS  ANY MORE BREAKS?
00873A FBD3 26 DA FBAF        BNE    BRKLP    YES,LOOP
00874                    *
00875                    * WRAP-UP PROCESSING AND EXIT
00876                    *
00877A FBD5 4F                CLRA             A = BREAKS IN FLAG
00878A FBD6 7D F3C5  A        TST    ASAVE    IS FUNCTION = -1?
00879A FBD9 2A 01 FBDC        BPL    BKPUT    NO, SO BRKSIN = 0
00880A FBDB 4C                INCA             FCTN = -1 =" BRKSIN:=1
00881A FBDC B7 F3D8  A BKPUT  STAA   BRKSIN   STORE APPROPRIATE FLAG
00882                    *
00883                    * RESTORE S-REG AND RETURN TO CALLER
00884                    *
00885A FBDF BE F3D0  A BKDONE LDS    SSAVE    RESTORE USER S-REG
00886A FBE2 39                RTS              RETURN
```

```
00888                    *
00889                    * MISCELLANEOUS ROUTINES FOR BRKSUB
00890                    *
00891                    * BREAKPOINT ADDRESS = 0 - IF FUNCTION = 4 THEN
00892                    * PUT BREAKPOINT ADDRESS IN CURRENT POSITION
00893                    * A HOLDS THE FUNCTION #, X HOLDS BREAKPOINT INDEX
00894                    *
00895A FBE3 81 04    A LN      CMPA    #4        IS FUNCTION = 4
00896A FBE5 26 E7 FBCE         BNE     BKCON2    IF NOT, THEN CONTINUE LOOP
00897                    *
00898A FBE7 BE F3CE  A         LDS     XHI       GET NEW BREAK ADDRESS
00899A FBEA AF 00    A         STS     0,X       PUT IN CURRENT POSITION
00900                    *
00901A FBEC 7A F3C5  A         DEC     ASAVE     DO NOT PLACE ADDRESS MORE
00902                    *                               THAN ONCE-CONT TO
00903                    *                               TAKE OUT BREAKPOINTSo
00904A FBEF 20 DD FBCE         BRA     BKCON2    CONTINUE LOOP
```

```
00906                          *
00907                          * BREAKS ARE NOT IN AND ADDRESS IS NON-ZERO.
00908                          * IF FUNCTION = -1 THEN SWI'S ARE TO BE PUT IN.
00909                          * A HOLDS FUNCTION NUMBER, S HOLDS ADDRESS
00910                          *
00911A FBF1 4D          NOBRIN TSTA
00912A FBF2 2A CD FBC1         BPL    BKCON1   NO,CONTINUE
00913                          *
00914A FBF4 34                 DES             MAKE ADDRESS POINT TO 1 LESS
00915A FBF5 32                 PULA            GET USER INSTRUCTION
00916A FBF6 A7 10    A         STAA   2*NBRBPT,X SAVE
00917A FBF8 86 3F    A         LDAA   #SWI     GET SWI OP CODE
00918A FBFA 36                 PSHA            REPLACE USER INSTRUCTION
00919A FBFB 20 D1 FBCE         BRA    BKCON2   CONTINUE LOOP
00920                          *
00921                          * FUNCTION=0, BRK ADDR NOT = 0, USER'S INSTR
00922                          * IS IN (NOT SWI).
00923                          * IF ADDRESS = XHI,XLO THEN SET ADDRESS = 0
00924                          *
00925A FBFD A6 00    A FNDRPL  LDAA   0,X      GET TOP BYTE OF ADDRESS
00926A FBFF B1 F3CE  A         CMPA   XHI      DO TOP BYTES COMPARE
00927A FC02 26 CA FBCE         BNE    BKCON2   NO,CONTINUE LOOP
00928A FC04 E6 01    A         LDAB   1,X      GET LOW BYTE OF ADDR
00929A FC06 F1 F3CF  A         CMPB   XLOW     SAME FOR LOW BYTES
00930A FC09 26 C3 FBCE         BNE    BKCON2
00931                          *
00932A FC0B 6F 00    A CLRBRK  CLR    0,X      CLEAR OUT BREAK
00933A FC0D 6F 01    A         CLR    1,X      ADDRESS FIELD
00934A FC0F 20 BD FBCE         BRA    BKCON2   CONTINUE LOOP
00935                          *
00936                          *
00937A FC11 7E F9B2  A OT2HS   JMP    OUT2HS
00938A FC14 7E F9B0  A OT4HS   JMP    OUT4HS
00939                          *
00940                          * PRINT OUT BREAK ADDRESS
00941                          * FUNCTION = 2, BREAK ADDRESS NOT = 0, X = ADDRESS INDEX
00942                          *
00943A FC17 BE F3D0  A PRNTBK  LDS    SSAVE
00944A FC1A 8D F8 FC14         BSR    OT4HS    OUTPUT ADDRESS AND SPACE
00945A FC1C 20 B2 FBD0         BRA    BKCON3   OUT4HS INCREMENTS X,
00946                          *                SO BYPAS 2 INX'S
```

```
00948                    *
00949                    * PRINT CONTENTS OF STACK
00950                    *
00951A FC1E BD F99C  A PRINT  JSR    PCRLF     PRINT CR LF
00952A FC21 FE F382  A        LDX    SP        PRINT OUT STACK
00953A FC24 08                INX
00954A FC25 8D EA FC11        BSR    OT2HS     CONDITION CODES
00955A FC27 8D E8 FC11        BSR    OT2HS     ACC-B
00956A FC29 8D E6 FC11        BSR    OT2HS     ACC-A
00957A FC2B 8D E7 FC14        BSR    OT4HS     X-REG
00958A FC2D 8D E5 FC14        BSR    OT4HS     P-COUNTER
00959A FC2F CE F382  A        LDX    #SP
00960A FC32 8D E0 FC14        BSR    OT4HS     STACK POINTER
00961A FC34 39                RTS
```

```
00963                    **********
00964                    *     PUNCH DUMP
00965                    *     PUNCH FROM BEGINING ADDRESS (BEGA) THRU ENDING
00966                    *     ADDRESS (ENDA)
00967                    **********
00968                    * FIRST GET START & END ADDRESS
00969                    *
00970A FC35 BD F96C   A PUNCH  JSR    GETADR
00971A FC38 09               DEX
00972A FC39 FF F37E   A       STX    ENDA
00973                    *
00974A FC3C CE FFED   A       LDX    #HEADMS  GET TITLE
00975A FC3F BD F995   A       JSR    PDATA1
00976A FC42 CE F3DA   A       LDX    #HEADBF
00977A FC45 BD FEE1   A PUN00 JSR    INCH1
00978A FC48 A7 00     A       STAA   0,X
00979A FC4A 08               INX
00980A FC4B 81 0D     A       CMPA   #$D
00981A FC4D 26 F6 FC45        BNE    PUN00
00982A FC4F 86 04     A       LDAA   #4
00983A FC51 A7 00     A       STAA   0,X
```

```
00985                    *
00986                    * NOW START PUNCH
00987                    *
00988A FC53 86 51     A          LDAA   #$51      8 BIT,2 STOP DIV 16
00989A FC55 B7 F408   A          STAA   ACIAS     RTS NOT HI
00990A FC58 CE 03FF   A          LDX    #$03FF
00991A FC5B 8D 7D FCDA           BSR    PNLDR
00992A FC5D 86 80     A          LDAA   #$80
00993A FC5F 8D 58 FCB9           BSR    KCOUT
00994A FC61 CE F3DA   A          LDX    #HEADBF
00995A FC64 8D 6D FCD3 PUND05    BSR    PUN
00996A FC66 81 0D     A          CMPA   #$D
00997A FC68 26 FA FC64           BNE    PUND05
00998A FC6A 86 FF     A          LDAA   #$FF
00999A FC6C 8D 4B FCB9           BSR    KCOUT
01000A FC6E F6 F37F   A PUND10   LDAB   ENDA+1
01001A FC71 F0 F37D   A          SUBB   BEGA+1
01002A FC74 B6 F37E   A          LDAA   ENDA
01003A FC77 B2 F37C   A          SBCA   BEGA
01004A FC7A 27 02 FC7E           BEQ    PUND25
01005A FC7C C6 FF     A          LDAB   #$FF
01006A FC7E 86 42     A PUND25   LDAA   #'B       PUNCH A B
01007A FC80 8D 37 FCB9           BSR    KCOUT
01008A FC82 37                   PSHB
01009A FC83 30                   TSX
01010A FC84 8D 4D FCD3           BSR    PUN
01011A FC86 32                   PULA              BYTE COUNT
01012A FC87 4C                   INCA
01013A FC88 B7 F3D2   A          STAA   CHRCNT
01014A FC8B CE F37C   A          LDX    #BEGA      PUNCH ADDRESS
01015A FC8E 8D 43 FCD3           BSR    PUN
01016A FC90 8D 41 FCD3           BSR    PUN
01017A FC92 FE F37C   A          LDX    BEGA       PUNCH DATA
```

```
01019A FC95 8D 3C FCD3 PUND30 BSR    PUN
01020A FC97 7A F3D2  A        DEC    CHRCNT
01021A FC9A 26 F9 FC95        BNE    PUND30
01022A FC9C FF F37C  A        STX    BEGA
01023A FC9F CE 0096  A        LDX    #$96
01024A FCA2 8D 36 FCDA        BSR    PNLDR    150 1'S
01025A FCA4 FE F37C  A        LDX    BEGA
01026A FCA7 09                DEX
01027A FCA8 BC F37E  A        CPX    ENDA
01028A FCAB 26 C1 FC6E        BNE    PUND10
01029A FCAD 86 47    A        LDAA   #'G
01030A FCAF 8D 08 FCB9        BSR    KCOUT    PUNCH END CHAR
01031A FCB1 CE 0019  A        LDX    #$19
01032A FCB4 8D 24 FCDA        BSR    PNLDR
01033A FCB6 7E FB20  A        JMP    CONTRL
01034                  *
01035                  *SUBROUTINE TO PUNCH A BYTE
01036                  * BYTE IS DISPLAYED ON TV AS 2 HEX
01037                  * NO REGISTER CHANGED
01038A FCB9 37         KCOUT  PSHB
01039A FCBA BD FA1A  A KC1    JSR    CHKESC
01040A FCBD F6 F408  A        LDAB   ACIAS    READY???
01041A FCC0 57                ASRB
01042A FCC1 57                ASRB
01043A FCC2 24 F6 FCBA        BCC    KC1
01044A FCC4 B7 F409  A        STAA   ACIAD
01045A FCC7 33                PULB
01046A FCC8 36                PSHA
01047A FCC9 BD F982  A        JSR    OUTHL    SEND BYTE( IN HEX)
01048A FCCC 32                PULA            TO VDG DISPLAY
01049A FCCD 36                PSHA
01050A FCCE BD F986  A        JSR    OUTHR
01051A FCD1 32                PULA
01052A FCD2 39                RTS
01053                  *
01054                  *SUB TO PUNCH 1 BYTE POINTED BY X REG
01055                  * X=X+1
01056                  * BYTE FETCH FROM MEMORY IS IN SYNC
01057                  * WITH TV RETRACE
01058A FCD3 BD FE86  A PUN    JSR    SYNCLD
01059A FCD6 8D E1 FCB9        BSR    KCOUT
01060A FCD8 08                INX
01061A FCD9 39                RTS
01062                  *
01063                  * PUNCH LEADER
01064                  *
01065A FCDA 86 FF    A PNLDR  LDAA   #$FF
01066A FCDC 8D DB FCB9        BSR    KCOUT
01067A FCDE 09                DEX
01068A FCDF 26 F9 FCDA        BNE    PNLDR
01069A FCE1 39                RTS
01070                  *
```

```
01072                          *
01073                          *
01074                          * SWI-1 SOFTWARE INTERRUPT LEVEL 1 PROCESSING
01075                          *
01076           FCE2  A SWI1S  EQU    *
01077A FCE2 BF F382   A        STS    SP          SAVE USER'S SP
01078                          *
01079A FCE5 86 03     A        LDAA   #3
01080A FCE7 BD FBA6   A        JSR    BRKSUB      GO TAKE OUT ALL BREAKS
01081                          *
01082                          * DECREMENT P-COUNTER
01083                          *
01084A FCEA 30                 TSX                X:=STACK POINTER - 1
01085A FCEB 6D 06     A        TST    6,X         IF LOWER BYTE = 0 =" BORROW
01086A FCED 26 02 FCF1         BNE    SWI1S1      BRANCH IF BORROW NOT REQ'D
01087A FCEF 6A 05     A        DEC    5,X         DECREMENT UPPER BYTE
01088A FCF1 6A 06     A SWI1S1 DEC    6,X         DECREMENT LOWER BYTE
01089                          *
01090                          * TEST FOR ADDRESS TRACE OR BREAK
01091                          *
01092A FCF3 EE 05     A        LDX    5,X         X:=P COUNTER
01093A FCF5 BC F3A0   A        CPX    TRCADR      IS SWI FOR TRACE?
01094A FCF8 27 18 FD12         BEQ    TRCINH      YES, GO TO TRACE INT HANDLER
01095                          *
01096A FCFA A6 00     A        LDAA   0,X         GET INSTRUCTION CAUSING SWI
01097A FCFC 81 3F     A        CMPA   #SWI        WAS IT REPLACED BY CALL BREAKOUT
01098A FCFE 26 0C FD0C         BNE    BRKINH      YES, SO MUST BE A BREAK
01099                          *
01100                          * USER SWI-TRANSFER THROUGH LEVEL 2 SWI
01101                          *
01102A FD00 30                 TSX                X:=STACK POINTER
01103A FD01 6C 06     A        INC    6,X         UPDATE LOW BYTE OF P-COUNTER
01104A FD03 26 02 FD07         BNE    INCNOV      BRANCH IF NO CARRY
01105A FD05 6C 05     A        INC    5,X         UPDATE HIGH BYTE IF NECESSARY
01106                          *
01107A FD07 FE F386   A INCNOV LDX    SWI2        X:=POINTER TO LEVEL 2 SWI HANDLER
01108A FD0A 6E 00     A        JMP    0,X         GO TO LEVEL 2 HANDLER
01109                          *
01110                          *
01111                          *
01112                          *
01113                          * BREAK INTERRUPT HANDLER
01114                          *
01115           FD0C  A BRKINH EQU    *
01116A FD0C BD FC1E   A        JSR    PRINT       STOP AND SHOW REGS TO USER
01117A FD0F 7E FB20   A CTRL   JMP    CONTRL      RETURN TO CONTROL LOOP
```

```
01119                          *
01120                          * TRACE INTERRUPT HANDLER
01121                          * P-COUNTER  HAS BEEN DECREMENTED TO POINT AT SWI
01122                          * TRCINS HOLDS OP CODE REPLACED BY SWI
01123                          * X HOLDS ADDRESS OF WHERE TRACE SWI IS
01124                          *
01125A FD12 B6 F3A2  A TRCINH LDAA   TRCINS   GET OP CODE OF TRACED INSTR
01126A FD15 A7 00    A        STAA   0,X      RESTORE TO USER'S CODE
01127                          *
01128A FD17 7D F3D9  A        TST    BRKTRC   IS PROCESSING TO BE
01129                          *                     IMMEDIATELY CONTINUED?
01130A FD1A 27 0F FD2B        BEQ    NBKTRC   BRANCH IF NOT
01131                          *
01132                          * PROCESSING IS TO "CONTINUE"
01133                          *
01134A FD1C 7F F3D9  A        CLR    BRKTRC   RESET CONTINUE FLAG
01135A FD1F 86 FF    A        LDAA   #$FF     FLAG TO SET BREAKS IN CODE
01136A FD21 BD FBA6  A        JSR    BRKSUB   PUT BREAKS IN
01137A FD24 7F F3A0  A        CLR    TRCADR   NO MORE TRACE, SO CLEAR ADDRESS
01138A FD27 7F F3A1  A        CLR    TRCADR+1
01139A FD2A 3B                RTI             CONTINUE
01140                          *
01141                          * TRACE IS DUE TO N OR T TRACE COMMANDS
01142                          *
01143A FD2B BD FC1E  A NBKTRC JSR    PRINT    PRINT STACK
01144A FD2E FE F3A3  A        LDX    NTRACE   GET # INSTRUCTIONS TO TRACE
01145A FD31 09                DEX             DECREMENT COUNT
01146A FD32 FF F3A3  A        STX    NTRACE   AND RESTORE
01147A FD35 27 D8 FD0F        BEQ    CTRL     BRANCH IF ALL TRACES DONE
01148                          *
01149                          * TRACE NOT DONE - TRACE NEXT INSTRUCTION
01150                          *
01151A FD37 B6 F3A2  A CONTRC LDAA   TRCINS   GET CURRENT INSTRUCTION
01152A FD3A B7 F388  A        STAA   BRINS    SAVE IN CASE IT'S A BRANCH
01153A FD3D 8D 70 FDAF        BSR    OPCBYT   GO GET # BYTES/TYPE
01154A FD3F 4D                TSTA            CHECK FOR BRANCH
01155A FD40 2A 35 FD77        BPL    CKOBRA   CHECK FOR OTHER THAN BRANCH
```

```
01157                          *
01158                          * RELATIVE BRANCH TYPE INSTRUCTION
01159                          * DETERMINE WHERE TO PUT SWI
01160                          * S- HOLDS POINTER TO USER STACK AFTER SWI
01161                          *
01162A FD42 32                        PULA            GET CONDITION CODE
01163A FD43 34                        DES             UPDATE STACK PTR AFTER PULL
01164A FD44 8A 10    A                ORAA    #%00010000 MAKE INT'S INHIBITED
01165A FD46 06                        TAP             RESTORE USER'S C. CODE REG
01166A FD47 7E F388  A                JMP     BRINS   GO SEE HOW RELATIVE BRANCH
01167                          *                              FARES
01168                          *
01169                          * BRANCH WAS NOGO - PUT SWI AT NEXT INSTRUCTION
01170                          *
01171A FD4A 86 02    A BRNOGO  LDAA    #2      A = # BYTES AFTER CURRENT INSTR
01172A FD4C 20 29 FD77                BRA     CKOBRA  GO PUT SWI APPROPRIATELY
01173                          *
01174                          * BRANCH WAS GO, PUT SWI AT ADDRESS BEING
01175                          * JUMPED TO
01176                          *
01177A FD4E FE F3A0  A BRGO     LDX     TRCADR  X : = TRACE ADDRESS
01178A FD51 A6 01    A                LDAA    1,X     GET BRANCH OFFSET
01179A FD53 08                        INX             OFFSET IS RELATIVE TO
01180A FD54 08                        INX             INSTR FOLLOWING BRANCH
01181A FD55 2B 12 FD69                BMI     BRGODC  BRANCH IF OFFSET NEGATIVE
01182A FD57 8D 16 FD6F BRG1     BSR     INCX    INCREMENT X BY AMOUNT IN
01183                          *                              A REG
01184A FD59 FF F3A0  A BRG2     STX     TRCADR  SAVE ADDRESS OF NEXT
01185                          *                              INSTR TO STOP ON
01186A FD5C A6 00    A                LDAA    0,X     GET INSTRUCTION TO BE REPLACED
01187A FD5E B7 F3A2  A                STAA    TRCINS  SAVE
01188A FD61 86 3F    A                LDAA    #SWI    GET SWI OP CODE
01189A FD63 A7 00    A                STAA    0,X     REPLACE INSTR WITH SWI
01190A FD65 BE F382  A                LDS     SP      GET ORIGINAL STACK POINTER
01191A FD68 3B                        RTI             TRACE ANOTHER INSTR
01192                          *
01193                          * X NEEDS TO BE DECREMENTED (OFFSET NEGATIVE)
01194                          *
01195A FD69 09               BRGODC   DEX             DECREMENT ADDRESS
01196A FD6A 4C                        INCA            INCREMENT COUNTER
01197A FD6B 26 FC FD69                BNE     BRGODC  IF COUNTER NOT 0, BRANCH
01198A FD6D 20 EA FD59                BRA     BRG2    IF DONE, GO RETURN TO USER PROG
01199                          *
01200                          * SUBROUTINE TO INCREMENT X BY CONTENTS OF A
01201                          *
01202A FD6F 4D               INCX     TSTA            IS A = 0?
01203A FD70 27 04 FD76                BEQ     INCXR   IF SO, INC DONE
01204A FD72 08               INXLP    INX             ELSE INCREMENT X
01205A FD73 4A                        DECA            DECREMENT COUNT
01206A FD74 26 FC FD72                BNE     INXLP   IF COUNT NOT YET 0, LOOP
01207A FD76 39               INCXR    RTS             RETURN FROM THIS SUBROUTINE
```

```
01209                    *
01210                    * INSTRUCTION TO BE TRACED IS NOT A BRANCH.
01211                    *
01212A FD77 FE F3A0  A CKOBRA LDX    TRCADR   X : = TRACE ADDRESS
01213A FD7A E6 00    A        LDAB   0,X      GET INSTR TO BE TRACED
01214A FD7C C1 6E    A        CMPB   #$6E     IS IT A JUMP, INDEXED?
01215A FD7E 27 1A FD9A        BEQ    JMPIDX   YES, GO SIMULATE JUMP IDXED
01216A FD80 C1 7E    A        CMPB   #$7E     JUMP, EXTENDED?
01217A FD82 27 1D FDA1        BEQ    JMPEXT
01218A FD84 C1 AD    A        CMPB   #$AD     JSR, INDEXED?
01219A FD86 27 12 FD9A        BEQ    JMPIDX   (JUMP IDXED IS SAME AS
01220                    *                             TRANSFER OF CONTROL)
01221A FD88 C1 BD    A        CMPB   #$BD     JSR, EXTENDED?
01222A FD8A 27 15 FDA1        BEQ    JMPEXT
01223A FD8C C1 3B    A        CMPB   #$3B     RTI?
01224A FD8E 27 15 FDA5        BEQ    RTISIM
01225A FD90 C1 39    A        CMPB   #$39     RTS?
01226A FD92 27 16 FDAA        BEQ    RTSSIM
01227A FD94 C1 8D    A        CMPB   #$8D     BSR?
01228A FD96 27 B6 FD4E        BEQ    BRGO     (BRANCH PROCESSING)
01229                    *
01230                    * NOT A BRANCH, JUMP, RTI, RTS
01231                    * A REGISTER HOLDS # BYTES IN INSTRUCTION
01232                    *
01233A FD98 20 BD FD57        BRA    BRG1     PUT IN NEW SWI AND
01234                    *                             TRACE NEXT INSTRUCTION
01235                    *
01236                    * JUMP, JSR INDEXED SIMULATION
01237                    *
01238A FD9A A6 01    A JMPIDX LDAA   1,X      A : = ADDRESS OFFSET
01239A FD9C 30                TSX
01240A FD9D EE 03    A        LDX    3,X      GET TARGET'S X REG
01241A FD9F 20 B6 FD57        BRA    BRG1     UPDATE X, TRACE NEXT INSTR
01242                    *
01243                    * JUMP, JSR EXTENDED
01244                    *
01245A FDA1 EE 01    A JMPEXT LDX    1,X      GET ADDRESS TO BE JUMPED TO
01246A FDA3 20 B4 FD59        BRA    BRG2     GO TRACE NEXT INSTR
01247                    *
01248                    * RTI ENCOUNTERED
01249                    *
01250A FDA5 30           RTISIM TSX
01251A FDA6 EE 0C    A        LDX    12,X     GET P-COUNTER FROM STACK
01252A FDA8 20 AF FD59        BRA    BRG2     GO TRACE NEXT INSTR.
01253                    *
01254                    * RTS ENCOUNTERED
01255                    *
01256A FDAA 30           RTSSIM TSX
01257A FDAB EE 07    A        LDX    7,X      GET RETURN P-REG FROM STACK
01258A FDAD 20 AA FD59        BRA    BRG2     GO TRACE NEXT INSTR
```

```
01260                    **************************
01261                    *
01262                    * OPBCYT
01263                    *
01264                    * THIS ROUTINE DETERMINES THE # OF BYTES
01265                    *  IN AN INSTRUCTION
01266                    * GIVEN ITS OP CODE.
01267                    *
01268                    * INPUT: A HOLDS THE OP CODE
01269                    *
01270                    * OUTPUT: X HOLDS INDEX OF TABLE ELEMENT
01271                    * B NOT RESTORED
01272                    * A HOLDS # BYTES IN INSTRUCTION
01273                    * EXCEPT FOR BRANCHES IN WHICH CASE A IS NEGATIVE
01274                    *
01275                    **************************
01276                    *
01277          FDAF   A OPCBYT EQU     *
01278A FDAF 16                TAB               B:= OP CODE
01279A FDB0 44                LSRA
01280A FDB1 44                LSRA
01281A FDB2 44                LSRA              PUT 4 UPPER BITS OF OP CODE INTO
01282A FDB3 44                LSRA              LOWER 4 BITS OF A
01283                    *
01284A FDB4 CE FDCD  A        LDX     #OPBTTB   X= ADDRESS OF TABLE
01285A FDB7 8D B6 FD6F        BSR     INCX      INC X TO POINT TO ENTRY
01286                    *
01287A FDB9 A6 00    A        LDAA    0,X       GET TABLE ENTRY
01288A FDBB 26 0F FDCC        BNE     OPBTRT    IF NOT 0 THEN NO FURTHER
01289                    *                           PROCESSING NEEDED
01290                    *
01291                    * IF TOP 4 BITS = 8 OR C, THEN THERE ARE TWO CLASSES
01292                    * OF INSTRUCTIONS: 2 BYTE INSTRUCTIONS AND
01293                    * CE, 8C AND 8E WHICH ARE 3 BYTE INSTRUCTIONS
01294                    *
01295A FDBD 86 02    A        LDAA    #2        # BYTES IN MOST OF 8# INSTRUCTIOI
01296A FDBF C1 8C    A        CMPB    #$8C      3 BYTE INSTRUCTION?
01297A FDC1 27 08 FDCB        BEQ     OPBT3     YES, UPDATE A
01298A FDC3 C1 CE    A        CMPB    #$CE      3 BYTE INSTR?
01299A FDC5 27 04 FDCB        BEQ     OPBT3     YES, UPDATE A
01300A FDC7 C1 8E    A        CMPB    #$8E      3 BYTE INSTRUCTION?
01301A FDC9 26 01 FDCC        BNE     OPBTRT    NO, RETURN
01302                    *
01303A FDCB 4C           OPBT3  INCA              # BYTES IN INSTRUCTION:=3
01304                    *
01305A FDCC 39           OPBTRT RTS               RETURN TO CALLER
```

```
01307                    *
01308                    * OP CODE TO NUMBER OF BYTES CONVERSION TABLE
01309                    *
01310                    *              # BYTES  TOP 4 BITS OF OPCODE
01311                    *              -------  --------------------
01312                    *
01313           FDCD   A OPBTTB EQU     *
01314A FDCD     01     A        FCB     1        0
01315A FDCE     01     A        FCB     1        1
01316A FDCF     82     A        FCB     2+%10000000 2 (MINUS=" BRANCHES)
01317A FDD0     01     A        FCB     1        3
01318A FDD1     01     A        FCB     1        4
01319A FDD2     01     A        FCB     1        5
01320A FDD3     02     A        FCB     2        6
01321A FDD4     03     A        FCB     3        7
01322A FDD5     00     A        FCB     0        8 # BYTES=2 EXCEPT 8C,8E
01323A FDD6     02     A        FCB     2        9
01324A FDD7     02     A        FCB     2        A
01325A FDD8     03     A        FCB     3        B
01326A FDD9     00     A        FCB     0        C # BYTES=2 EXCEPT CE
01327A FDDA     02     A        FCB     2        D
01328A FDDB     02     A        FCB     2        E
01329A FDDC     03     A        FCB     3        F
```

```
01331                    *
01332                    * CONSTANT DATA
01333                    *
01334A FDDD    56     A MCL2   FCC    /VDG DEBUG SYSTEM/
01335A FDED    3B     A        FCC    /;TV-BUG VER 1.2/
01336A FDFC    0D     A        FCB    $D
01337A FDFD    4D     A        FCC    /MOTOROLA-AUSTIN TEX/
01338A FE10    04     A        FCB    4
01339A FE11    43     A MCL3   FCC    /CC B  A   X    P    S/
01340A FE26    04     A        FCB    4
01341A FE27    0D     A MCL4   FCB    $D
01342A FE28    42     A        FCC    /BEG ADR? /
01343A FE31    04     A        FCB    4
01344A FE32    45     A MCL5   FCC    /END ADR? /
01345A FE3B    04     A        FCB    4
01346A FE3C    20     A NCHG   FCC    / MEMORY BAD!/
01347A FE48    04     A        FCB    4
01348A FE49    0D     A READY  FCB    $D
01349A FE4A    54     A        FCC    /TVBUG/
01350A FE4F    0D     A        FCB    $D,4
01351A FE51    0D     A OFSET  FCB    $D
01352A FE52    4F     A        FCC    /OFFSET=/
01353A FE59    04     A        FCB    4
01354A FE5A    54     A BADJP  FCC    /TOO FAR!/
01355A FE62    04     A        FCB    4
01356A FE63    0D     A FILLMS FCB    $D
01357A FE64    43     A        FCC    /CHAR?/
01358A FE69    04     A        FCB    4
01359                    *
```

```
01361                          **********
01362                          * A,B UNCHANGED, X=(NEXTBY)
01363                          * SAVE CHARACTER AT LOCATION
01364                          *   (NEXTBY)
01365                          * FLASH THE LOCATION (4 COLORS)
01366                          * REPLACE THE CHAR & RETURN
01367                          **********
01368A FE6A 37                BLINK  PSHB
01369A FE6B FE F3CC  A                 LDX    NEXTBY
01370A FE6E 8D 16 FE86                 BSR    SYNCLD
01371A FE70 36                         PSHA
01372A FE71 86 CF    A                 LDAA   #$CF     BLINK CURSOR
01373A FE73 C6 04    A                 LDAB   #4
01374A FE75 BD FF7F  A INC1    JSR    SAVE
01375A FE78 8B 10    A                 ADDA   #$10
01376A FE7A BD FF53  A                 JSR    WAITFS
01377A FE7D 5A                         DECB
01378A FE7E 26 F5 FE75                 BNE    INC1
01379                          *
01380                          * REPLACE CHARACTER
01381                          *
01382A FE80 32                         PULA
01383A FE81 BD FF7F  A                 JSR    SAVE
01384A FE84 33                         PULB
01385A FE85 39                         RTS
```

```
01387                    **********
01388                    * B,X UNCHANGED
01389                    * MASK INTERRUPT BEFORE LOAD
01390                    *   & UNMASK AFTER
01391                    * WAIT FOR LEADING EDGE
01392                    *   OF HORIZONTAL SYNC THEN
01393                    *   LOAD A FROM LOCATION 0,X
01394                    **********
01395A FE86 0F           SYNCLD SEI
01396A FE87 B6 F406  A  SLD1   LDAA   PIABD
01397A FE8A 2A FB FE87         BPL    SLD1
01398A FE8C B6 F406  A  SLD2   LDAA   PIABD
01399A FE8F 2B FB FE8C         BMI    SLD2
01400                    *
01401                    * NOW CAN LOAD
01402                    *
01403A FE91 A6 00    A          LDAA   0,X
01404A FE93 0E                  CLI
01405A FE94 39                  RTS
```

```
01407                    **********
01408                    * A,B UNCHANGED
01409                    * IF HOME KEY CLOSED
01410                    *    THEN ADR OF DISPLAY TO (NEXTBY)
01411                    * IF UP CLOSED
01412                    *    THEN 2'S COMP OF 32 TO A,B &
01413                    *         CALL UPDOWN
01414                    * IF DOWN KEY CLOSED
01415                    *    THEN 32 TO A,B & CALL UPDOWN
01416                    * IF RIGHT KEY CLOSED
01417                    *    THEN INCREMENT (NEXTBY) TO MARGIN
01418                    * IF LEFT KEY CLOSED
01419                    *    THEN DECREMENT (NEXTBY) TO MARGIN
01420                    **********
01421A FE95 37          CURSOR PSHB
01422A FE96 36                 PSHA
01423A FE97 F6 F406  A         LDAB   PIABD
01424                  *CHECK UP ARROW
01425A FE9A 56                 RORB
01426A FE9B 24 09 FEA6         BCC    CUR1
01427A FE9D 37                 PSHB
01428A FE9E 86 E0    A         LDAA   #$E0
01429A FEA0 C6 FF    A         LDAB   #$FF      -32
01430A FEA2 BD FF64  A         JSR    UPDOWN
01431A FEA5 33                 PULB
01432                  *CHECK DOWN ARROW
01433A FEA6 56         CUR1    RORB
01434A FEA7 24 08 FEB1         BCC    CUR2
01435A FEA9 37                 PSHB
01436A FEAA 86 20    A         LDAA   #$20
01437A FEAC 5F                 CLRB             +32
01438A FEAD BD FF64  A         JSR    UPDOWN
01439A FEB0 33                 PULB
01440                  *CHECK LEFT ARROW
01441A FEB1 56         CUR2    RORB
01442A FEB2 24 0E FEC2         BCC    CUR3
01443A FEB4 B6 F3CD  A         LDAA   NEXTBY+1
01444A FEB7 84 1F    A         ANDA   #$1F
01445A FEB9 27 07 FEC2         BEQ    CUR3
01446A FEBB FE F3CC  A         LDX    NEXTBY
01447A FEBE 09                 DEX
01448A FEBF FF F3CC  A         STX    NEXTBY
01449                  *CHECK RIGHT ARROW
01450A FEC2 56         CUR3    RORB
01451A FEC3 24 10 FED5         BCC    CUR4
01452A FEC5 B6 F3CD  A         LDAA   NEXTBY+1
01453A FEC8 8A E0    A         ORAA   #$E0
01454A FECA 81 FF    A         CMPA   #$FF
01455A FECC 27 07 FED5         BEQ    CUR4
01456A FECE FE F3CC  A         LDX    NEXTBY
01457A FED1 08                 INX
01458A FED2 FF F3CC  A         STX    NEXTBY
01459                  *CHECK HOME KEY
01460A FED5 56         CUR4    RORB
01461A FED6 24 03 FEDB         BCC    CUR5
01462A FED8 CE D000  A         LDX    #VDGRAM
01463A FEDB 32         CUR5    PULA
01464A FEDC 33                 PULB
```

```
01465A FEDD FF F3CC  A        STX    NEXTBY
01466A FEE0 39                RTS
```

```
01468                          *
01469                          * NOW CHECK KEYBOARD
01470                          * ON EXIT INPUT CHAR IN A-REG
01471                          * X,B UNCHANGED
01472A FEE1 FF F3CA  A INCH1    STX     SAVEX
01473A FEE4 37                 PSHB
01474A FEE5 8D 83 FE6A INCH0   BSR     BLINK    FLASH CURSOR
01475A FEE7 8D AC FE95         BSR     CURSOR
01476A FEE9 FE F3CA  A         LDX     SAVEX
01477                          *
01478                          *
01479A FEEC B6 F404  A         LDAA    PIAAD    LOOK FOR ANY KEY
01480A FEEF 2B 10 FF01         BMI     NONE
01481A FEF1 B1 F3D7  A         CMPA    FLAGK    SAME KEY
01482A FEF4 27 0E FF04         BEQ     SAME     FROM LAST TIME?
01483                          *
01484                          * CHECK FOR ESCAPE FUNCTION
01485                          *
01486A FEF6 81 1B    A         CMPA    #$1B
01487A FEF8 27 12 FF0C         BEQ     CNTLEV
01488                          *
01489A FEFA B7 F3D7  A         STAA    FLAGK
01490A FEFD 8D 10 FF0F         BSR     OUTCH1   ECHO CHAR
01491A FEFF 33                 PULB
01492A FF00 39                 RTS
01493                          *
01494                          * NO KEY CLOSED NOW GO TO USER ROUTINE
01495                          * IF USER HAS KEY DOWN
01496                          *    RETURN WITH CHAR IN A REG
01497                          *           WITH CARRY = 1
01498                          * ELSE RETURN WITH CARRY =0
01499                          *
01500                          *
01501A FF01 7F F3D7  A NONE    CLR     FLAGK
01502A FF04 0C        SAME     CLC
01503A FF05 BD F390  A         JSR     USRINP
01504A FF08 24 DB FEE5         BCC     INCH0
01505A FF0A 33                 PULB
01506A FF0B 39                 RTS
01507                          *
01508A FF0C 7E FB20  A CNTLEV  JMP     CONTRL
```

```
01510                          **********
01511                          * A,B,X UNCHANGED
01512                          * IGNORES LF
01513                          * CALLS "RETURN" IF C.R.
01514                          * CALLS "INIT" IF FORM FEED
01515                          * CALLS "SCROLL" IF BOTTOM
01516                          *    OF DISPLAY RAM IS EXCEEDED
01517                          * ELSE STORES CHAR
01518                          **********
01519A FF0F 37                 OUTCH1 PSHB
01520A FF10 36                        PSHA
01521A FF11 FF F3CA  A                STX    SAVEX
01522                          *
01523                          *
01524                          * CHECK IF END OF DISPLAY BUFF IF YES THEN SCROLL
01525                          *
01526A FF14 FE F3CC  A MAIN1   LDX    NEXTBY
01527A FF17 8C D200  A         CPX    #VDGRAM+512
01528A FF1A 26 03 FF1F         BNE    MAIN2
01529A FF1C BD FFC9  A         JSR    SCROLL
01530                          *
01531                          * CHECK FOR C.R. IF YES THEN FINISH LINE WITH BLANKS
01532                          * TRAP L.F.'S
01533                          *
01534A FF1F 81 0A    A MAIN2   CMPA   #$0A
01535A FF21 27 25 FF48         BEQ    MAIN6
01536A FF23 81 0D    A         CMPA   #$D
01537A FF25 26 06 FF2D         BNE    MAIN3
01538A FF27 8D 67 FF90         BSR    RETURN
01539A FF29 86 0A    A         LDAA   #$0A
01540A FF2B 20 E7 FF14         BRA    MAIN1
01541                          *
01542                          *
01543                          * CHECK FOR FORM FEED IF YES CLEAR SCREEN
01544                          *
01545A FF2D 81 0C    A MAIN3   CMPA   #$0C
01546A FF2F 26 04 FF35         BNE    MAIN4
01547A FF31 8D 75 FFA8         BSR    INIT
01548A FF33 20 1[illegible] FF48         BRA    ~~MAIN6~~ MAIN7
```

```
01550                    *
01551                    * CHECK FOR BACKSPACE IF YES MOVE POINTR BACK
01552                    * & STORE INVERTED BLANK
01553                    *
01554A FF35 81 08    A MAIN4  CMPA   #$08
01555A FF37 26 0C FF45        BNE    MAIN5
01556A FF39 8C D000  A        CPX    #VDGRAM
01557A FF3C 27 0A FF48        BEQ    MAIN6
01558A FF3E 09                DEX
01559A FF3F 86 60    A        LDAA   #$60
01560A FF41 8D 3C FF7F        BSR    SAVE
01561A FF43 20 03 FF48        BRA    MAIN6
01562                    *
01563                    * GET HERE TO SAVE THE BEGGAR..THEN HOME
01564                    *
01565          FF45  A MAIN5  EQU    *
01566A FF45 8D 38 FF7F        BSR    SAVE
01567A FF47 08                INX
01568A FF48 FF F3CC  A MAIN6  STX    NEXTBY
01569A FF4B FE F3CA  A MAIN7  LDX    SAVEX
01570A FF4E 32                PULA
01571A FF4F 33                PULB
01572                    *
01573                    * NOW GO TO USERS DISPLAY ROUTINE
01574                    *....NOTE USER MUST EXECUTE A RTS
01575                    * AS LAST INS
01576A FF50 7E F393  A        JMP    USROUT
01577                    *
```

```
01579                          **********
01580                          * A,B,X UNCHANGED
01581                          * WAIT FOR LEADING EDGE OF
01582                          *    FIELD SYNC(VERTICAL SYNC)
01583                          *    THEN EXIT
01584                          **********
01585           FF53  A WAITFS EQU    *
01586A FF53 37                  PSHB
01587A FF54 F6 F406  A WAIT2   LDAB   PIABD
01588A FF57 58                 ASLB
01589A FF58 58                 ASLB
01590A FF59 24 F9 FF54         BCC    WAIT2
01591                          * GET NEG EDGE
01592A FF5B F6 F406  A WAIT1   LDAB   PIABD
01593A FF5E 58                 ASLB
01594A FF5F 58                 ASLB
01595A FF60 25 F9 FF5B         BCS    WAIT1
01596A FF62 33                 PULB
01597A FF63 39                 RTS
```

```
01599                    **********
01600                    * ALL REGS ZAPPED
01601                    * ON ENTRY A,B CONTAIN OFFSET
01602                    *   TO BE ADDED TO (NEXTBY)
01603                    * OFFSET WILL BE ADDED ONLY
01604                    *   IF RESULT LOCATION WILL
01605                    *   BE IN ACTIVE AREA OF
01606                    *   DISPLAY RAM
01607                    * ON EXIT A,B CONTAIN RESULT
01608                    *   X = (NEXTBY) IF OK TO ADD
01609                    *   X = X(N-1) IF NOT OK TO ADD
01610                    **********
01611A FF64 BB F3CD  A UPDOWN ADDA   NEXTBY+1
01612A FF67 F9 F3CC  A        ADCB   NEXTBY
01613                    *
01614A FF6A B7 F3C7  A        STAA   TEMP+1
01615A FF6D F7 F3C6  A        STAB   TEMP
01616                    *
01617A FF70 C1 D0    A        CMPB   #VDGRAM/256
01618A FF72 2D 0A FF7E        BLT    UD0
01619A FF74 C1 D2    A        CMPB   #(VDGRAM+512)/256
01620A FF76 2C 06 FF7E        BGE    UD0
01621                    *
01622A FF78 FE F3C6  A        LDX    TEMP     OK TO ADJUST
01623A FF7B FF F3CC  A        STX    NEXTBY
01624                    *
01625A FF7E 39           UD0  RTS
```

```
01627                    *
01628                    *
01629                    * SAVE...COPIES BYTE
01630                    * WAITS FOR HORIZ SYNC THEN STORES IN
01631                    * DISPLAY RAM
01632                    * NO REG KLOBBERED
01633                    * ON ENTRY X REG POINTS TO LOC
01634                    * IN DISPLAY MEMORY TO STORE CHAR
01635                    * INTERRUPT IS MASKED BEFORE WRITE
01636                    * TO VDG MEMORY THEN UN MASKED
01637                    * (WE MUST WRITE DURING TV HORIZ RETRACE)
01638                    *
01639                    *
01640A FF7F 37          SAVE   PSHB
01641A FF80 0F                 SEI
01642                    *
01643A FF81 F6 F406  A SAVE0  LDAB   PIABD    WAIT FOR HS=1
01644A FF84 2A FB FF81         BPL    SAVE0
01645                    *
01646A FF86 F6 F406  A SAVE1  LDAB   PIABD    NEGATIVE EDGE
01647A FF89 2B FB FF86         BMI    SAVE1
01648                    *
01649                    * NOW CAN SAVE ( SCREEN BLANKED)
01650                    *
01651A FF8B A7 00    A         STAA   0,X
01652A FF8D 0E                 CLI
01653A FF8E 33                 PULB
01654A FF8F 39                 RTS
```

```
01656                    **********
01657                    * A,B UNCHANGED
01658                    * FILLS CURRENT LINE WITH
01659                    *   BLANKS FROM CURRENT
01660                    *   POSITION (NEXTBY) TO
01661                    *   END OF LINE
01662                    * POINTER (NEXTBY) SET TO
01663                    *   START OF NEXT LINE
01664                    * ON EXIT X = (NEXTBY)
01665                    **********
01666A FF90 FE F3CC  A RETURN LDX    NEXTBY
01667                    *
01668A FF93 36               PSHA
01669A FF94 86 60    A RET1   LDAA   #$60
01670A FF96 8D E7 FF7F       BSR    SAVE
01671                    *
01672                    * CONTINUE TO END OF LINE
01673                    *
01674A FF98 08               INX
01675A FF99 FF F3C6  A       STX    TEMP
01676A FF9C B6 F3C7  A       LDAA   TEMP+1
01677A FF9F 84 1F    A       ANDA   #$1F
01678A FFA1 26 F1 FF94       BNE    RET1
01679A FFA3 FF F3CC  A       STX    NEXTBY
01680A FFA6 32               PULA
01681A FFA7 39               RTS
01682
```

```
01684                          **********
01685                          * A,B UNCHANGED
01686                          * PIA (ASCII KEYBOARD & VDG)
01687                          *   IS CLEARED, SET FOR INPUT
01688                          *   & SET FOR DATA REG ACCESS
01689                          * ENTIRE SCREEN (512 BYTES) +
01690                          *   ADDITIONAL 512 BYTES ARE
01691                          *   FILLED WITH BLANKS
01692                          * ON EXIT X = (NEXTBY)
01693                          **********
01694A FFA8 7F F407  A INIT    CLR     PIABC
01695A FFAB 7F F406  A         CLR     PIABD
01696A FFAE 7F F3D7  A         CLR     FLAGK
01697                          *
01698A FFB1 36               A         PSHA
01699A FFB2 86 04    A         LDAA    #$4       POINT AT DATA REG
01700A FFB4 B7 F407  A         STAA    PIABC
01701                          *
01702                          *
01703                          * NOW BLANK SCREEN
01704                          *
01705A FFB7 CE D000  A         LDX     #VDGRAM
01706A FFBA 86 60    A         LDAA    #$60
01707A FFBC FF F3CC  A         STX     NEXTBY
01708                          *
01709A FFBF 8D BE FF7F INIT1   BSR     SAVE
01710A FFC1 08               A         INX
01711A FFC2 8C D400  A         CPX     #VDGRAM+1024
01712A FFC5 26 F8 FFBF         BNE     INIT1
01713A FFC7 32               A         PULA
01714A FFC8 39               A         RTS
```

```
01716                    **********
01717                    * A,B UNCHANGED
01718                    * MOVES EACH CHAR UP 1 LINE
01719                    *    (32 LOCATIONS)
01720                    * TOP LINE IS LOST
01721                    * BOTTOM LINE IS BLANK ON EXIT
01722                    * (NEXTBY) IS DECREMENTED BY 32
01723                    * ON EXIT X = (NEXTBY)
01724                    **********
01725A FFC9 36            SCROLL PSHA
01726                    *
01727A FFCA CE D000  A           LDX    #VDGRAM
01728                    *
01729A FFCD 37                  PSHB
01730A FFCE 0F                  SEI
01731A FFCF F6 F406  A  SCROL3  LDAB   PIABD
01732A FFD2 2A FB FFCF          BPL    SCROL3
01733                    *
01734A FFD4 F6 F406  A  SCROL4  LDAB   PIABD
01735A FFD7 2B FB FFD4          BMI    SCROL4
01736                    *
01737A FFD9 A6 20    A          LDAA   32,X
01738A FFDB 8D A2 FF7F          BSR    SAVE
01739A FFDD 08                  INX
01740A FFDE 8C D200  A          CPX    #VDGRAM+512
01741A FFE1 26 EC FFCF          BNE    SCROL3
01742A FFE3 CE D1E0  A          LDX    #VDGRAM+480
01743                    *
01744A FFE6 0E                  CLI
01745A FFE7 FF F3CC  A          STX    NEXTBY
01746A FFEA 33                  PULB
01747A FFEB 32                  PULA
01748A FFEC 39                  RTS
```

```
01750                    *
01751                    *
01752A FFED    0D     A HEADMS FCB    $D
01753A FFEE    4E     A        FCC    /NAME? /
01754A FFF4    04     A        FCB    4
```

```
01756                    *
01757                    *
01758                    *INTERRUPT VECTORS
01759                    *
01760A FFF8                       ORG    BASORG+$7F8
01761A FFF8    F824  A            FDB    IO
01762A FFFA    F82E  A            FDB    SFEI
01763A FFFC    F829  A            FDB    POWDWN
01764A FFFE    FAD7  A            FDB    START
```

```
01766                    *
01767                    * RAM SCRATCHPAD FOR TVBUG..+ STACK
01768                    *
01769                    *
01770A F000                       ORG    $F000
01771A F000    036F   A           RMB    879
01772A F36F    000B   A STACK     RMB    11
01773                    *
01774          0008   A NBRBPT    EQU    8          # OF BREAKPOINTS
01775                    *
01776                    *THE FOLLOWING ARE INITALIZED AT START
01777                    *
01778A F37A    0002   A IOV       RMB    2          I/O INTERRUPT POINTER
01779A F37C    0002   A BEGA      RMB    2          PRINT/PUNCH START LOC
01780A F37E    0002   A ENDA      RMB    2          PRINT PUNCH STOP LOC
01781A F380    0002   A NIO       RMB    2          NMI INTERRUPT POINTER
01782A F382    0002   A SP        RMB    2          USER STACK POINTER
01783A F384    0002   A SWI1      RMB    2          LEVEL 1 SWI VECTOR
01784A F386    0002   A SWI2      RMB    2          LEVEL 2 SWI VECTOR
01785A F388    0008   A BRINS     RMB    8          COND BRANCH STORAGE
01786                    *
01787                    * USER I/O VECTORS
01788                    * INITALIZED TO RTS
01789                    * USER MUST END HIS ROUTINES
01790                    * WITH RTS OR ALL IS LOST!!!!
01791                    *
01792                    *
01793A F390    0003   A USRINP     RMB    3          USER INPUT ROUTINE
01794A F393    0003   A USROUT     RMB    3          USER OUTPUT ROUTINE
01795                    *
01796A F396    0003   A USR1      RMB    3
01797A F399    0003   A USR2      RMB    3
01798A F39C    0004   A USR3      RMB    4
01799                    *
01800                    * A 16 BYTE PROM MAY BE PATCHED
01801                    * OVER THE ABOVE 16 LOCATIONS.
01802                    * USER VECTORS WILL THEN BE
01803                    * AVAILABLE AT POWER ON.
01804                    *
01805          F3A0   A BRANEN    EQU    *
```

```
01807                    *
01808                    * THE FOLLOWING ARE INITALIZED TO ZERO
01809                    *
01810                    *
01811A F3A0    0002   A TRCADR RMB    2          TRACE ADDRESS
01812A F3A2    0001   A TRCINS RMB    1          PP CODE REPLACED BY TRACE
01813A F3A3    0002   A NTRACE RMB    2          NO OF INS TO TRACE
01814                    *
01815A F3A5    0010   A BRKADR RMB    NBRBPT*2 BREAKPOINT TABLE
01816A F3B5    0010   A BRKINS RMB    NBRBPT*2 OP CODES FOR BREAK
01817                    *
01818A F3C5    0001   A ASAVE  RMB    1
01819A F3C6    0002   A TEMP   RMB    2
01820A F3C8    0002   A TEMP2  RMB    2
01821A F3CA    0002   A SAVEX  RMB    2
01822A F3CC    0002   A NEXTBY RMB    2
01823A F3CE    0001   A XHI    RMB    1
01824A F3CF    0001   A XLOW   RMB    1
01825A F3D0    0002   A SSAVE  RMB    2
01826A F3D2    0001   A CHRCNT RMB    1
01827A F3D3    0001   A TERMCH RMB    1
01828A F3D4    0001   A VFLAG  RMB    1
01829A F3D5    0002   A OFFADR RMB    2
01830A F3D7    0001   A FLAGK  RMB    1
01831A F3D8    0001   A BRKSIN RMB    1          1=BREAKS IN USER CODE
01832A F3D9    0001   A BRKTRC RMB    1          1=P-COUNTR IS AT BREAKPOINT
01833                    *              & USER WANTS TO CONTINUE---
01834                    *                     ONE TRACE WILL BE DONE
01835                    *              & BREAKS RESTORED
01836          F3DA    A ENDIN0 EQU    *
01837A F3DA    0020   A HEADBF RMB    32
```

```
01839                           END
TOTAL ERRORS 00000--00000


   F409 ACIAD  00079*00508 01044
   F408 ACIAS  00078*00079 00464 00505 00670 00989 01040
   FA6B ADDOFF 00488 00567*
   F872 ADRSTR 00170*00637
   F3C5 ASAVE  00830 00836 00859 00878 00901 01818*
   F8AB ASCII  00211 00218*
   F936 BAD0   00314*00340
   F957 BAD1   00335*00339
   F963 BAD2   00321 00325 00327 00342*
   F951 BAD3   00323 00330*
   F929 BADDR  00091 00257 00280 00308*00358 00363 00460 00528 00537 00725
   FB44 BADDRJ 00725*00737 00750 00758 00784
   FE5A BADJP  00624 01354*
   F800 BASORG 00084*01760
   F37C BEGA   00204 00207 00220 00237 00259 00275 00281 00285 00359 00484 004
               00487 00604 00605 01001 01003 01014 01017 01022 01025 01779*
   F9D3 BILD   00465*00467 00471 00475 00499
   FBC1 BKCON1 00859*00912
   FBCE BKCON2 00870*00896 00904 00919 00927 00930 00934
   FBD0 BKCON3 00872*00945
   FBDF BKDONE 00846 00885*
   FBDC BKPUT  00879 00881*
   FE6A BLINK  01368*01474
   F3A0 BRANEN 00643 01805*
   F87D BRG    00175 00177*
   FD57 BRG1   01182*01233 01241
   FD59 BRG2   01184*01198 01246 01252 01258
   FD4E BRGO   00177 01177*01228
   FD69 BRGODC 01181 01195*01197
   F388 BRINS  01152 01166 01785*
   F3A5 BRKADR 00832 01815*
   FD0C BRKINH 00173 01098 01115*
   F3B5 BRKINS 00872 01816*
   FBAF BRKLP  00836*00873
   F3D8 BRKSIN 00840 00881 01831*
   FBA6 BRKSUB 00592 00732 00752 00761 00828*01080 01136
   F3D9 BRKTRC 00772 00789 01128 01134 01832*
   FD4A BRNOGO 00176 01171*
   FB49 BSRBRK 00732*00740 00746
   F926 C3     00232 00262 00287 00295*
   FA65 C5     00476 00553 00559*00621 00623 00626
   FA29 CHANGE 00138 00528*
   FA2C CHG0   00529*00550 00556
   FA27 CHK1   00519 00521*
   FA1A CHKESC 00504 00515*01039
   F3D2 CHRCNT 00313 00315 00540 01013 01020 01826*
   FD77 CKOBRA 01155 01172 01212*
   FC0B CLRBRK 00863 00932*
   FF0C CNTLEV 01487 01508*
   FBA3 CNTRL2 00733 00802*
   FB90 CONT   00130 00789*
   FD37 CONTRC 00780 01151*
   FB20 CONTRL 00099 00185 00186 00187 00295 00559 00690*00715 00802 01033 011
```

```
FF2D MAIN3  01537 01545*
FF35 MAIN4  01546 01554*
FF45 MAIN5  01555 01565*
FF48 MAIN6  01535 01548 01557 01561 01568*
FDDD MCL2   00681 01334*
FE11 MCL3   00799 01339*
FE27 MCL4   00356 01341*
FE32 MCL5   00361 01344*
F8C9 MORE   00234*
FD2B NBKTRC 01130 01143*
0008 NBRBPT 00851 00916 01774*01815 01816
FE3C NCHG   00557 01346*
F8B5 NEWCHR 00223*00236
F892 NEWLIN 00203*00239
FB73 NEXT   00140 00771*
F3CC NEXTBY 01369 01443 01446 01448 01452 01456 01458 01465 01526 01568 016
            01612 01623 01666 01679 01707 01745 01822*
F380 NIO    00108 00684 01781*
FA88 NMI    00586*00683
FBF1 NOBRIN 00841 00911*
FF01 NONE   01480 01501*
F3A3 NTRACE 00774 01144 01146 01813*
FB32 NXTCHR 00708*00714
F3D5 OFFADR 00461 00573 00574 01829*
FA9D OFFSET 00152 00598*
FE51 OFSET  00458 00599 01351*
FDCB OPBT3  01297 01299 01303*
FDCC OPBTRT 01288 01301 01305*
FDCD OPBTTB 01284 01313*
FDAF OPCBYT 01153 01277*
FC11 OT2HS  00937*00954 00955 00956
FC14 OT4HS  00938*00944 00957 00958 00960
F8E0 OUCH4  00229 00247*
F8DE OUT1S  00242 00243 00244 00245 00246*
F9A7 OUT2H  00420*00429 00430
F9B2 OUT2HS 00094 00212 00430*00534 00937
F8DC OUT2S  00234 00245*
F9B0 OUT4HS 00093 00205 00276 00429*00531 00613 00938
F8D6 OUT5S  00219 00242*
F8C1 OUTASC 00227 00229*
F9B6 OUTCH  00387 00389 00398 00411 00432*00468
FF0F OUTCH1 00089 00247 00432 00560 01490 01519*
FA68 OUTCH5 00560*00589
F8A0 OUTDAT 00210*00214
F982 OUTHL  00374*00421 01047
F986 OUTHR  00384*00424 01050
F8BF OUTPRD 00225 00228*
F9B4 OUTS   00431*00590 00702
F99C PCRLF  00203 00218 00274 00409*00529 00587 00744 00798 00951
FAD4 PD2    00558 00600 00625 00628*00696
F995 PDATA1 00090 00256 00357 00362 00400*00459 00628 00682 00800 00975
F992 PDATA2 00398*00402
F405 PIAAC  00072*00677
F404 PIAAD  00071*00072 00073 00074 00516 00675 01479
F407 PIABC  00074*00678 01694 01700
F406 PIABD  00073*00674 01396 01398 01423 01587 01592 01643 01646 01695 01
            01734
FCDA PNLDR  00991 01024 01032 01065*01068
```

```
FB52 PNTBRK 00128 00744*00753
F829 POWDWN 00108*01763
FC1E PRINT  00801 00951*01116 01143
FC17 PRNTBK 00866 00943*
FB98 PSTAK1 00144 00593 00798*
FCD3 PUN    00995 01010 01015 01016 01019 01058*
FC45 PUN00  00977*00981
FC35 PUNCH  00142 00970*
FC64 PUND05 00995*00997
FC6E PUND10 01000*01028
FC7E PUND25 01004 01006*
FC95 PUND30 01019*01021
F9E8 RDBLCK 00473 00480*
FE49 READY  00695 01348*
FF94 RET1   01669*01678
FF90 RETURN 01538 01666*
FB4D RSTBRK 00148 00737*
FDA5 RTISIM 01224 01250*
FDAA RTSSIM 01226 01256*
FF04 SAME   01482 01502*
FF7F SAVE   00097 01374 01383 01560 01566 01640*01670 01709 01738
FF81 SAVE0  01643*01644
FF86 SAVE1  01646*01647
F3CA SAVEX  00538 01472 01476 01521 01569 01821*
FFCF SCROL3 01731*01732 01741
FFD4 SCROL4 01734*01735
FFC9 SCROLL 00092 01529 01725*
FB59 SETBRK 00150 00750*
F82E SFEI   00113*01762
FA50 SKPSTR 00541 00547*
FE87 SLD1   01396*01397
FE8C SLD2   01398*01399
F382 SP     00586 00638 00664 00694 00775 00952 00959 01077 01190 01782*
F3D0 SSAVE  00829 00885 00943 01825*
F36F STACK  00171 00658 01772*
FAD7 START  00636*01764
F9FB STBLCK 00490*00498
003F SWI    00058*00917 01097 01188
F384 SWI1   00113 01783*
FCE2 SWI1S  00172 01076*
FCF1 SWI1S1 01086 01088*
F386 SWI2   01107 01784*
FE86 SYNCLD 00098 01058 01370 01395*
F3C6 TEMP   00609 00610 00611 01614 01615 01622 01675 01676 01819*
F3C8 TEMP2  00308 00309 00336 00337 00342 00569 00571 00572 00575 00576 00
            01820*
F3D3 TERMCH 00316 00547 01827*
FB8C TRACE  00146 00784*
FB76 TRACE2 00772*00785
FB79 TRACE3 00774*00791
F3A0 TRCADR 00777 01093 01137 01138 01177 01184 01212 01811*
FD12 TRCINH 01094 01125*
F3A2 TRCINS 00779 01125 01151 01187 01812*
FF7E UD0    01618 01620 01625*
FF64 UPDOWN 01430 01438 01611*
F396 USR1   00122 01796*
F399 USR2   00124 01797*
F39C USR3   00126 01798*
```

```
F390 USRINP 01503 01793*
F393 USROUT 01576 01794*
D000 VDGRAM 00060*01462 01527 01556 01617 01619 01705 01711 01727 01740 01
F9B9 VERIFY 00160 00439*
F3D4 VFLAG  00440 00456 00491 01828*
FF5B WAIT1  01592*01595
FF54 WAIT2  01587*01590
FF53 WAITFS 01376 01585*
F3CE XHI    00343 00409 00412 00530 00533 00763 00898 00926 01823*
F3CF XLOW   00258 00282 00542 00765 00929 01824*
F923 ZSCR   00162 00294*
```

```
00001                           NAM    TVRTS
00002                           TTL    TVBUG ROUTINES
00003                   **********************************************************
00004                   *  THESE ROUTINES ARE POSITION INDEPENDENT AND MAY BE
00005                   *  LOCATED ANYWHERE IN MEMORY.
00006                   *  THEY INCLUDE THE FOLLOWING:
00007                   *  1.  PRINTER DRIVERS  -  START AT THE BASE ADDRESS
00008                   *
00009                   *     CONNECT AN APPROPRIATE RS-232 OR TTL DRIVER TO THE
00010                   *     OUTPUT OF THE ACIA (PIN 6), AND ATTACH TO
00011                   *     A PRINTER.  NOW TVBUG WILL PRINT A CHARACTER EACH
00012                   *     TIME A CHARACTER IS DISPLAYED ON THE SCREEN.
00013                   *     (JUST GREAT FOR WORD PROCESSING!)
00014                   *
00015                   *  2.  S1S9 LOADER/PUNCH  -  START AT THE BASE
00016                   *      ADDRESS +2
00017                   *
00018                   *     THIS PROGRAM WILL ALLOW TVBUG TO READ AND WRITE
00019                   *     PROGRAMS WHICH HAVE BEEN WRITTEN USING THE
00020                   *     MIKBUG, EXBUG, OR MINIBUG FORMAT.  (S1S9)
00021                   *     THIS INCLUDES SWTPC, AND TSC TAPES.
00022                   *
00023                   **********************************************************
00024
00025                   **********************************************************
00026                   *  ROM ROUTINES
00027                   **********************************************************
00028
00029           F800  A INCH   EQU    $F800
00030           F803  A OUTCH  EQU    $F803
00031           F806  A PDATA1 EQU    $F806
00032           F80F  A OUT4HS EQU    $F80F
00033           F812  A OUT2HS EQU    $F812
00034           F821  A CONTRL EQU    $F821
00035           F929  A BADDR1 EQU    $F929
00036           F96C  A GETADR EQU    $F96C
00037           F99C  A PCRLF  EQU    $F99C
00038           F9A7  A OUT2H  EQU    $F9A7
00039           FA6B  A ADDOFF EQU    $FA6B
00040           FE51  A OFSET  EQU    $FE51
00041                   **********************************************************
00042                   *  RAM TEMPORARIES
00043                   **********************************************************
00044           F300  A XHI    EQU    $F300
00045           F302  A MCONT  EQU    $F302
00046           F304  A TEMP   EQU    $F304
00047           F306  A BYTECT EQU    $F306
00048           F308  A TW     EQU    $F308
00049           F30A  A CKSM   EQU    $F30A
00050           F30C  A FIDH   EQU    $F30C
00051           F3CC  A NEXTBY EQU    $F3CC
00052           F3D4  A VFLAG  EQU    $F3D4
00053           F3D5  A OFFADR EQU    $F3D5
00054           F37C  A BEGA   EQU    $F37C
00055           F37E  A ENDA   EQU    $F37E
00056                   **********************************************************
00057                   *  PERIPHERIAL ADDRESSES
00058                   **********************************************************
```

```
00059           F408  A ACIAC  EQU    $F408
00060           F409  A ACIAD  EQU    $F409
00061
00062                   ************************************************************
00063                   *
00064A 0200                    ORG    $0200
00065                   *
00066A 0200 20 02 0204         BRA    ACIA
00067A 0202 20 4E 0252         BRA    S1S9
00068                   *
00069                   *
00070                   *
00071                   *  START OF ACIA DRIVERS
00072                   *
00073                   *
00074                   *
00075A 0204 86 03     A ACIA   LDAA   #$03
00076A 0206 B7 F408   A        STAA   ACIAC    MASTER RESET
00077A 0209 86 51     A        LDAA   #$51
00078A 020B B7 F408   A        STAA   ACIAC    8 BIT,NP,2STOPS,DIV16
00079A 020E 86 7E     A        LDAA   #$7E
00080A 0210 B7 F393   A        STAA   $F393
00081A 0213 8D 05 021A         BSR    SWAP
00082A 0215 7E F821   A        JMP    CONTRL   NOW BACK'N PRINT ALL
00083A 0218 20 15 022F START   BRA    SWICH
00084A 021A 30          SWAP   TSX
00085A 021B A6 00     A        LDAA   0,X
00086A 021D B7 F300   A        STAA   $F300
00087A 0220 A6 01     A        LDAA   1,X
00088A 0222 B7 F301   A        STAA   $F301
00089A 0225 FE F300   A        LDX    $F300
00090A 0228 08                 INX
00091A 0229 08                 INX
00092A 022A 08                 INX
00093A 022B FF F394   A        STX    $F394
00094A 022E 39                 RTS
00095A 022F 36          SWICH  PSHA             SAVE A
00096A 0230 8D 13 0245         BSR    TEST     GO PRINT CHAR IN A
00097A 0232 81 0D     A        CMPA   #$0D     CARRIAGE RETURN?
00098A 0234 26 0D 0243         BNE    END      NO!  GO GET NEXT CHAR
00099A 0236 86 0A     A        LDAA   #$0A     YES, ADD LF + 4 NULLS
00100A 0238 8D 0B 0245         BSR    TEST     PRINT LF
00101A 023A 4F                 CLRA            NULL
00102A 023B 8D 08 0245         BSR    TEST     1 NULL
00103A 023D 8D 06 0245         BSR    TEST     2 NULLS
00104A 023F 8D 04 0245         BSR    TEST     3 NULLS
00105A 0241 8D 02 0245         BSR    TEST     4 NULLS
00106A 0243 32          END    PULA
00107A 0244 39                 RTS
00108A 0245 37          TEST   PSHB             SAVE B
00109A 0246 F6 F408   A I1     LDAB   ACIAC    CHECK ACIA
00110A 0249 57                 ASRB
00111A 024A 57                 ASRB            READY?
00112A 024B 24 F9 0246         BCC    I1       NO
00113A 024D B7 F409   A        STAA   ACIAD    OUTPUT THE CHARACTER
00114A 0250 33                 PULB
00115A 0251 39                 RTS
```

```
00117                    *
00118                    *
00119                    *       S1 - S9 LOADER FUNCTION
00120                    *
00121                    *
00122A 0252 8D 77 02CB S1S9    BSR     SUB1      USE BSR TO TELL X WHERE TO PRINT
00123A 0254 20 21 0277         BRA     SWICH1
00124A 0256    50    A         FCC     /PUNCH (P), LOAD (L), VERIFY (V)?/
     A 0257    55    A
     A 0258    4E    A
     A 0259    43    A
     A 025A    48    A
     A 025B    20    A
     A 025C    28    A
     A 025D    50    A
     A 025E    29    A
     A 025F    2C    A
     A 0260    20    A
     A 0261    4C    A
     A 0262    4F    A
     A 0263    41    A
     A 0264    44    A
     A 0265    20    A
     A 0266    28    A
     A 0267    4C    A
     A 0268    29    A
     A 0269    2C    A
     A 026A    20    A
     A 026B    56    A
     A 026C    45    A
     A 026D    52    A
     A 026E    49    A
     A 026F    46    A
     A 0270    59    A
     A 0271    20    A
     A 0272    28    A
     A 0273    56    A
     A 0274    29    A
     A 0275    3F    A
00125A 0276    04    A         FCB     $04
00126A 0277 8D 1C 0295 SWICH1  BSR     PDATA3
00127A 0279 BD F800    A       JSR     INCH
00128A 027C 81 50      A       CMPA    #'P
00129A 027E 26 02 0282         BNE     KC1
00130A 0280 20 15 0297         BRA     S1S9P
00131A 0282 81 4C      A KC1   CMPA    #'L
00132A 0284 26 02 0288         BNE     KC2
00133A 0286 20 54 02DC         BRA     S1S9L2
00134A 0288 81 56      A KC2   CMPA    #'V
00135A 028A 26 02 028E         BNE     KC3
00136A 028C 20 50 02DE         BRA     S1S9V2
00137A 028E 81 1B      A KC3   CMPA    #$1B
00138A 0290 26 C0 0252         BNE     S1S9
00139A 0292 7E F821    A       JMP     CONTRL
00140                    *
00141A 0295 20 49 02E0 PDATA3  BRA     PDATA4
00142                    *
00143                    *
```

```
PAGE  004  S1S9    .SA:0  TVRTS  TVBUG ROUTINES

00144                          *
00145                          *      S1 - S9 PUNCH ROUTINE
00146A 0297 BD F96C  A S1S9P   JSR    GETADR    GO GET ADDRESSES
00147A 029A FE F37E  A         LDX    ENDA
00148A 029D 09                 DEX
00149A 029E FF F37E  A         STX    ENDA      CORRECTS FOR TVBUG
00150A 02A1 8D 3F 02E2         BSR    CLEAR1    CLEAR SCREEN
00151A 02A3 8D 3F 02E4         BSR    ACPI      INITIALIZE ACIA FOR PNCH
00152A 02A5 86 12    A         LDAA   #$12      TURN TTY PUNCH ON
00153A 02A7 8D 3D 02E6         BSR    DUMPO     OUTPUT CHAR
00154                          *
00155                          *  PUNCH LEADER - 25 NULLS
00156                          *
00157A 02A9 C6 25    A         LDAB   #$25      B=# NULLS TO PUNCH
00158A 02AB 4F         PNULL   CLRA             A=0  (NULL CHAR)
00159A 02AC 8D 38 02E6         BSR    DUMPO
00160A 02AE 5A                 DECB
00161A 02AF 26 FA 02AB         BNE    PNULL
00162A 02B1 FE F37C  A         LDX    BEGA
00163A 02B4 FF F308  A         STX    TW
00164A 02B7 B6 F37F  A PUN11   LDAA   ENDA+1
00165A 02BA B0 F309  A         SUBA   TW+1
00166A 02BD F6 F37E  A         LDAB   ENDA
00167A 02C0 F2 F308  A         SBCB   TW
00168A 02C3 26 23 02E8         BNE    PUN22
00169A 02C5 81 10    A         CMPA   #16
00170A 02C7 25 21 02EA         BCS    PUN23
00171A 02C9 20 1D 02E8         BRA    PUN22     GO AROUND THE NEXT STUFF
00172                          *
00173                          *    SOME MORE POSITION INDEPENDENCE STUFF
00174                          *
00175A 02CB 30         SUB1    TSX              PUT STACK POINTER INTO X-REG TOO!
00176A 02CC A6 00    A         LDAA   0,X       PUT FIRST BYTE OF LAST BSR ADDR
00177A 02CE B7 F300  A         STAA   $F300     INTO TEMP1
00178A 02D1 A6 01    A         LDAA   1,X       PUT SECOND BYTE IN
00179A 02D3 B7 F301  A         STAA   $F301     INTO TEMP2
00180A 02D6 FE F300  A         LDX    $F300     NOW GET STRING POINTER
00181A 02D9 08                 INX              CORRECT FOR BRA XXX
00182A 02DA 08                 INX              HERE TOO!
00183A 02DB 39                 RTS              GO BACK WHERE YA CAME FROM
00184                          *
00185A 02DC 20 61 033F S1S9L2  BRA    S1S9L1
00186A 02DE 20 61 0341 S1S9V2  BRA    S1S9V1
00187A 02E0 20 69 034B PDATA4  BRA    PDATA2    HELPSIES!
00188A 02E2 20 6D 0351 CLEAR1  BRA    CLEAR
00189A 02E4 20 57 033D ACPI    BRA    ACPIN
00190A 02E6 20 5B 0343 DUMPO   BRA    DUMP1
00191                          *
00192A 02E8 86 0F    A PUN22   LDAA   #15
00193A 02EA 8B 04    A PUN23   ADDA   #4
00194A 02EC B7 F302  A         STAA   MCONT     FRAME COUNT THIS RECORD
00195A 02EF 80 03    A         SUBA   #3
00196A 02F1 B7 F304  A         STAA   TEMP      BYTE COUNT THIS RECORD
00197                          *  PUNCH C/R, L/F, NULL, S, 1.
00198A 02F4 86 42    A         LDAA   #'B       PRINT 'B' FOR EACH RECORD
00199A 02F6 BD F803  A         JSR    OUTCH
00200A 02F9 8D D0 02CB         BSR    SUB1
00201A 02FB 20 09 0306         BRA    SWICH2
```

```
  202A 02FD    0D    A          FCB    $D,$A,0,0,0,0,'S,'1,4 PUNCH FORMAT
     A 02FE    0A    A
     A 02FF    00    A
     A 0300    00    A
     A 0301    00    A
     A 0302    00    A
     A 0303    53    A
     A 0304    31    A
     A 0305    04    A
00203A 0306 8D 3D 0345 SWICH2 BSR    PUNCHE   (LIKE PDATA, ONLY TO ACIA)
00204A 0308 5F                CLRB           CLEAR CHECKSUM
00205                      *  PUNCH FRAME COUNT
00206A 0309 CE F302  A        LDX    #MCONT
00207A 030C 8D 39 0347        BSR    PUNT2    PUNCH 2 HEX CHARACTERS
00208                      *  PUNCH ADDRESS
00209A 030E CE F308  A        LDX    #TW
00210A 0311 8D 34 0347        BSR    PUNT2
00211A 0313 8D 32 0347        BSR    PUNT2
00212                      *  PUNCH DATA
00213A 0315 FE F308  A        LDX    TW
00214A 0318 8D 2D 0347 PUN32  BSR    PUNT2    PUNCH ONE BYTE (2 FRAMES)
00215A 031A 7A F304  A        DEC    TEMP
00216A 031D 26 F9 0318        BNE    PUN32
00217A 031F FF F308  A        STX    TW
00218A 0322 53                COMB
00219A 0323 37                PSHB
00220A 0324 30                TSX
00221A 0325 8D 20 0347        BSR    PUNT2    PUNCH CHECKSUM
00222A 0327 33                PULB           RESTORE STACK
00223A 0328 FE F308  A        LDX    TW
00224A 032B 09                DEX
00225A 032C BC F37E  A        CPX    ENDA
00226A 032F 26 86 02B7        BNE    PUN11
00227A 0331 8D 98 02CB        BSR    SUB1
00228A 0333 20 03 0338        BRA    SWICH3
00229A 0335    53    A        FCC    /S9/
     A 0336    39    A
00230A 0337    04    A        FCB    $04
00231A 0338 8D 58 0392 SWICH3 BSR    PUNCHP   OUTPUT EOF
00232A 033A 7E F821  A        JMP    CONTRL   GO TO CONTROL
00233                      *
00234A 033D 20 1F 035E ACPIN  BRA    ACPINT
00235A 033F 20 6B 03AC S1S9L1 BRA    S1S9L
00236A 0341 20 5F 03A2 S1S9V1 BRA    S1S9V
00237A 0343 20 3D 0382 DUMP1  BRA    DUMP     THESE HELP MAKE IT POS. INDEP.
00238A 0345 20 4B 0392 PUNCHE BRA    PUNCHP
00239                      *
00240                      *  PUNCH 2 HEX CHARACTERS, UPDATE CHECKSUM
00241A 0347 EB 00    A PUNT2  ADDB   0,X      UPDATE CHECKSUM
00242A 0349 20 4E 0399        BRA    OUT2H1   OUTPUT TWO HEX CHAR & RTS
00243                      *
00244A 034B BD F99C  A PDATA2 JSR    PCRLF
00245A 034E 7E F806  A PDAT1P JMP    PDATA1
00246                      *
00247                      *  CLEAR SCREEN ROUTINE = ACCA UNCHANGED.
00248                      *
00249A 0351 36         CLEAR  PSHA
00250A 0352 86 0C    A        LDAA   #$0C     (FF)
```

```
00251A 0354 BD F803  A          JSR    OUTCH
00252A 0357 86 D0    A          LDAA   #$D0
00253A 0359 B7 F3CC  A          STAA   NEXTBY
00254A 035C 32                  PULA
00255A 035D 39                  RTS
00256                    *
00257                    *  ACIA INITIALIZE (PUNCH)
00258                    *
00259A 035E 86 03    A ACPINT   LDAA   #3        MASTER RESET
00260A 0360 B7 F408  A          STAA   ACIAC
00261A 0363 86 51    A          LDAA   #$51     8 BITS, NP, 2 STOPS,/16
00262A 0365 B7 F408  A          STAA   ACIAC
00263A 0368 39                  RTS
00264                    *
00265                    *  ACIA INITIALIZE  (LOAD & VERIFY)
00266                    *
00267A 0369 86 03    A ACLINT   LDAA   #3
00268A 036B B7 F408  A          STAA   ACIAC
00269A 036E 86 10    A          LDAA   #$10     DIVIDE BY 1
00270A 0370 B7 F408  A          STAA   ACIAC
00271A 0373 39                  RTS
00272                    *
00273                    *
00274A 0374 44           OUTHL  LSRA            OUT HEX LEFT BCD DIGIT
00275A 0375 44                  LSRA
00276A 0376 44                  LSRA
00277A 0377 44                  LSRA
00278A 0378 84 0F    A OUTHR    ANDA   #$F      OUTPUT HEX RIGHT BCD DIGIT
00279A 037A 8B 30    A          ADDA   #$30
00280A 037C 81 39    A          CMPA   #$39
00281A 037E 23 02 0382          BLS    DUMP
00282A 0380 8B 07    A          ADDA   #$7
00283A 0382 37           DUMP   PSHB
00284A 0383 F6 F408  A OUTC1    LDAB   ACIAC
00285A 0386 57                  ASRB
00286A 0387 57                  ASRB
00287A 0388 24 F9 0383          BCC    OUTC1    XMIT NOT READY
00288A 038A B7 F409  A          STAA   ACIAD
00289A 038D 33                  PULB
00290A 038E 39                  RTS
00291                    *
00292                    *
00293A 038F 8D F1 0382 PNCHP2   BSR    DUMP
00294A 0391 08                  INX
00295A 0392 A6 00    A PUNCHP   LDAA   0,X
00296A 0394 81 04    A          CMPA   #4
00297A 0396 26 F7 038F          BNE    PNCHP2
00298A 0398 39                  RTS              STOP ON EOT
00299                    *
00300                    *
00301A 0399 A6 00    A OUT2H1   LDAA   0,X       OUTPUT 2 HEX CHARACTERS
00302A 039B 8D D7 0374 OUT2HA   BSR    OUTHL    OUT LEFT HEX CHAR
00303A 039D A6 00    A          LDAA   0,X      PICK UP BYTE AGAIN
00304A 039F 08                  INX
00305A 03A0 20 D6 0378          BRA    OUTHR    OUTPUT RIGHT HEX CHARACTER
00306                    *
00307                    *
00308                    *     S1 - S9 VERIFY  SETS VERIFY FLAG
```

```
00309                          *              THEN GOES TO LOAD
00310                          *
00311                          *
00312A 03A2 86 01    A S1S9V   LDAA   #1
00313A 03A4 B7 F3D4  A         STAA   VFLAG
00314A 03A7 20 06 03AF         BRA    LOAD0
00315                          *
00316A 03A9 7E F821  A RESRT1  JMP    CONTRL
00317                          *
00318                          *
00319A 03AC 7F F3D4  A S1S9L   CLR    VFLAG
00320A 03AF CE FE51  A LOAD0   LDX    #OFSET   PRINT OFFSET QUESTIONS
00321A 03B2 8D 97 034B         BSR    PDATA2
00322A 03B4 BD F929  A         JSR    BADDR1   X = XHI = TEMP2
00323A 03B7 8D 98 0351         BSR    CLEAR
00324A 03B9 FF F3D5  A         STX    OFFADR
00325A 03BC 8D A0 035E         BSR    ACPINT   PUNCH RDR ON
00326A 03BE 86 11    A         LDAA   #$11     (DC1)
00327A 03C0 8D C0 0382         BSR    DUMP
00328A 03C2 8D A5 0369         BSR    ACLINT   INIT FOR INPUT
00329                          *
00330A 03C4 8D 55 041B LOAD3   BSR    LOADE
00331A 03C6 81 53    A         CMPA   #'S
00332A 03C8 26 FA 03C4         BNE    LOAD3    1ST CHAR NOT S
00333A 03CA BD F803  A         JSR    OUTCH    PRINT EACH S
00334A 03CD 8D 7B 044A         BSR    LOAD     READ CHAR
00335A 03CF 81 39    A         CMPA   #'9
00336A 03D1 27 D6 03A9         BEQ    RESRT1
00337A 03D3 81 31    A         CMPA   #'1
00338A 03D5 26 ED 03C4         BNE    LOAD3    2ND CHAR NOT 1
00339A 03D7 7F F30A  A         CLR    CKSM     ZERO CHECKSUM
00340A 03DA 8D 2D 0409         BSR    BYTE5    READ BYTE
00341A 03DC 80 02    A         SUBA   #2
00342A 03DE B7 F306  A         STAA   BYTECT
00343                          *
00344                          *  BUILD ADDRESS
00345                          *
00346A 03E1 8D 18 03FB         BSR    BADDR2
00347A 03E3 BD FA6B  A         JSR    ADDOFF   ADDS OFFSET TO X
00348                          *
00349                          *  STORE AND CHECK DATA
00350                          *
00351A 03E6 8D 21 0409 LOAD11  BSR    BYTE5
00352A 03E8 7A F306  A         DEC    BYTECT
00353A 03EB 27 47 0434         BEQ    LOAD15   ZERO BYTE COUNT
00354A 03ED 7D F3D4  A         TST    VFLAG
00355A 03F0 26 02 03F4         BNE    VERFON
00356A 03F2 A7 00    A         STAA   0,X      STORE DATA
00357A 03F4 A1 00    A VERFON  CMPA   0,X      CHECK DATA ENTRY FOR VERF
00358A 03F6 26 41 0439         BNE    LOAD19   DATA NOT STORED
00359A 03F8 08                 INX
00360A 03F9 20 EB 03E6         BRA    LOAD11
00361                          *
00362                          *  NEW BADDR2
00363                          *
00364                          *
00365A 03FB 8D 0C 0409 BADDR2  BSR    BYTE5
00366A 03FD B7 F300  A         STAA   XHI
```

```
00367A 0400 8D 07 0409          BSR     BYTE5
00368A 0402 B7 F301  A          STAA    XHI+1
00369A 0405 FE F300  A          LDX     XHI
00370A 0408 39                  RTS
00371                       *
00372                       *
00373                       *   INPUT BYTE FROM ACIA
00374                       *
00375                       *
00376A 0409 8D 12 041D BYTE5    BSR     INHEX3   GET HEX CHAR
00377A 040B 48                  ASLA
00378A 040C 48                  ASLA
00379A 040D 48                  ASLA
00380A 040E 48                  ASLA
00381A 040F 16                  TAB
00382A 0410 8D 0B 041D          BSR     INHEX3
00383A 0412 1B                  ABA
00384A 0413 16                  TAB
00385A 0414 FB F30A  A          ADDB    CKSM
00386A 0417 F7 F30A  A          STAB    CKSM
00387A 041A 39                  RTS
00388                       *
00389A 041B 20 2D 044A LOADE    BRA     LOAD
00390                       *
00391                       *
00392A 041D 8D 2B 044A INHEX3   BSR     LOAD
00393A 041F 20 00 0421          BRA     HEXID    CHECK + RTS
00394                       *
00395                       *   INHEX SUBROUTINE
00396                       *
00397A 0421 80 30    A HEXID    SUBA    #$30
00398A 0423 2B 47 046C          BMI     C1       NOT HEX
00399A 0425 81 09    A          CMPA    #$09
00400A 0427 2F 0A 0433          BLE     IN1HG
00401A 0429 81 11    A          CMPA    #$11
00402A 042B 2B 3F 046C          BMI     C1       NOT HEX
00403A 042D 81 16    A          CMPA    #$16
00404A 042F 2E 3B 046C          BGT     C1       NOT HEX
00405A 0431 80 07    A          SUBA    #7
00406A 0433 39           IN1HG  RTS
00407                       *
00408                       *   DOES CHECKDUM CHECK?
00409                       *
00410A 0434 7C F30A  A LOAD15   INC     CKSM
00411A 0437 27 8B 03C4          BEQ     LOAD3
00412A 0439 FF F30C  A LOAD19   STX     FIDH
00413A 043C 86 20    A          LDAA    #$20
00414A 043E BD F803  A          JSR     OUTCH
00415A 0441 CE F30C  A          LDX     #FIDH
00416A 0444 BD F80F  A          JSR     OUT4HS
00417A 0447 7E F821  A          JMP     CONTRL
00418                       *
00419A 044A B6 F408  A LOAD     LDAA    ACIAC
00420A 044D 47                  ASRA
00421A 044E 24 FA 044A          BCC     LOAD     RX NOT READY
00422A 0450 B6 F409  A          LDAA    ACIAD    INPUT CHAR
00423A 0453 84 7F    A          ANDA    #$7F     RESET PARITY
00424A 0455 81 7F    A          CMPA    #$7F
```

PAGE 009 S1S9 .SA:0 TVRTS TVBUG ROUTINES

```
00425A 0457 27 F1 044A          BEQ     LOAD      RUBOUT - DEL
00426A 0459 39                  RTS
00427                   *
00428                   *   PRINT LINE WITH A PRECEEDING CR/LF
00429                   *    X POINTS TO STRING.  STRING MUST
00430                   *    TERMINATE WITH A $04 CHARACTER.
00431                   *
00432                   *   INPUT BYTE  (TWO FRAMES FROM KEYBOARD)
00433                   *
00434A 045A 8D 13 046F BYTE     BSR     INHEX     GET HEX CHARACTER
00435A 045C 48                  ASLA
00436A 045D 48                  ASLA
00437A 045E 48                  ASLA
00438A 045F 48                  ASLA
00439A 0460 16                  TAB
00440A 0461 8D 0C 046F          BSR     INHEX
00441A 0463 1B                  ABA
00442A 0464 16                  TAB
00443A 0465 FB F30A  A          ADDB    CKSM
00444A 0468 F7 F30A  A          STAB    CKSM
00445A 046B 39                  RTS
00446                   *
00447A 046C 7E F821  A C1       JMP     CONTRL
00448                   *
00449A 046F BD F800  A INHEX    JSR     INCH
00450A 0472 20 AD 0421          BRA     HEXID     CHECK + RTS
00451                   *
00452                           END
TOTAL ERRORS 00000--00000
```

```
0204 ACIA    00066 00075*
F408 ACIAC   00059*00076 00078 00109 00260 00262 00268 00270 00284 00419
F409 ACIAD   00060*00113 00288 00422
0369 ACLINT  00267*00328
02E4 ACPI    00151 00189*
033D ACPIN   00189 00234*
035E ACPINT  00234 00259*00325
FA6B ADDOFF  00039*00347
F929 BADDR1  00035*00322
03FB BADDR2  00346 00365*
F37C BEGA    00054*00162
045A BYTE    00434*
0409 BYTE5   00340 00351 00365 00367 00376*
F306 BYTECT  00047*00342 00352
046C C1      00398 00402 00404 00447*
F30A CKSM    00049*00339 00385 00386 00410 00443 00444
0351 CLEAR   00188 00249*00323
02E2 CLEAR1  00150 00188*
F821 CONTRL  00034*00082 00139 00232 00316 00417 00447
0382 DUMP    00237 00281 00283*00293 00327
0343 DUMP1   00190 00237*
02E6 DUMPO   00153 00159 00190*
0243 END     00098 00106*
F37E ENDA    00055*00147 00149 00164 00166 00225
F30C FIDH    00050*00412 00415
F96C GETADR  00036*00146
```

PAGE 010 S1S9 .SA:0 TVRTS TVBUG ROUTINES

```
0421 HEXID  00393 00397*00450
0246 I1     00109*00112
0433 IN1HG  00400 00406*
F800 INCH   00029*00127 00449
046F INHEX  00434 00440 00449*
041D INHEX3 00376 00382 00392*
0282 KC1    00129 00131*
0288 KC2    00132 00134*
028E KC3    00135 00137*
044A LOAD   00334 00389 00392 00419*00421 00425
03AF LOAD0  00314 00320*
03E6 LOAD11 00351*00360
0434 LOAD15 00353 00410*
0439 LOAD19 00358 00412*
03C4 LOAD3  00330*00332 00338 00411
041B LOADE  00330 00389*
F302 MCONT  00045*00194 00206
F3CC NEXTBY 00051*00253
F3D5 OFFADR 00053*00324
FE51 OFSET  00040*00320
F9A7 OUT2H  00038*
0399 OUT2H1 00242 00301*
039B OUT2HA 00302*
F812 OUT2HS 00033*
F80F OUT4HS 00032*00416
0383 OUTC1  00284*00287
F803 OUTCH  00030*00199 00251 00333 00414
0374 OUTHL  00274*00302
0378 OUTHR  00278*00305
F99C PCRLF  00037*00244
034E PDAT1P 00245*
F806 PDATA1 00031*00245
034B PDATA2 00187 00244*00321
0295 PDATA3 00126 00141*
02E0 PDATA4 00141 00187*
038F PNCHP2 00293*00297
02AB PNULL  00158*00161
02B7 PUN11  00164*00226
02E8 PUN22  00168 00171 00192*
02EA PUN23  00170 00193*
0318 PUN32  00214*00216
0345 PUNCHE 00203 00238*
0392 PUNCHP 00231 00238 00295*
0347 PUNT2  00207 00210 00211 00214 00221 00241*
03A9 RESRT1 00316*00336
0252 S1S9   00067 00122*00138
03AC S1S9L  00235 00319*
033F S1S9L1 00185 00235*
02DC S1S9L2 00133 00185*
0297 S1S9P  00130 00146*
03A2 S1S9V  00236 00312*
0341 S1S9V1 00186 00236*
02DE S1S9V2 00136 00186*
0218 START  00083*
02CB SUB1   00122 00175*00200 00227
021A SWAP   00081 00084*
022F SWICH  00083 00095*
0277 SWICH1 00123 00126*
```

```
0306 SWICH2 00201 00203*
0338 SWICH3 00228 00231*
F304 TEMP   00046*00196 00215
0245 TEST   00096 00100 00102 00103 00104 00105 00108*
F308 TW     00048*00163 00165 00167 00209 00213 00217 00223
03F4 VERFON 00355 00357*
F3D4 VFLAG  00052*00313 00319 00354
F300 XHI    00044*00366 00368 00369
```

= 129

```
0200 20 02 20 4E 86 03 B7 F4 08 86 51 B7 F4 08 86 7E   . N..7...Q7....
0210 B7 F3 93 8D 05 7E F8 21 20 15 30 A6 00 B7 F3 00   7.....! .0&.7..
0220 A6 01 B7 F3 01 FE F3 00 08 08 08 FF F3 94 39 36   &.7...........96
0230 8D 13 81 0D 26 0D 86 0A 8D 0B 4F 8D 08 8D 06 8D   ....&.....O.....
0240 04 8D 02 32 39 37 F6 F4 08 57 57 24 F9 B7 F4 09   ...297...WW$.7..
0250 33 39 8D 77 20 21 50 55 4E 43 48 20 28 50 29 2C   39.. !PUNCH (P),
0260 20 4C 4F 41 44 20 28 4C 29 2C 20 56 45 52 49 46    LOAD (L), VERIF
0270 59 20 28 56 29 3F 04 8D 1C BD F8 00 81 50 26 02   Y (V)?...=...P&.
0280 20 15 81 4C 26 02 20 54 81 56 26 02 20 50 81 1B    ..L&. T.V&. P..
0290 26 C0 7E F8 21 20 49 BD F9 6C FE F3 7E 09 FF F3   &@..! I=........
02A0 7E 8D 3F 8D 3F 86 12 8D 3D C6 25 4F 8D 38 5A 26   ..?.?...=F%O.8Z&
02B0 FA FE F3 7C FF F3 08 B6 F3 7F B0 F3 09 F6 F3 7E   .......6..0.....
02C0 F2 F3 08 26 23 81 10 25 21 20 1D 30 A6 00 B7 F3   ...&#..%! .0&.7.
02D0 00 A6 01 B7 F3 01 FE F3 00 08 08 39 20 61 20 61   .&.7.......9 . .
02E0 20 69 20 6D 20 57 20 5B 86 0F 8B 04 B7 F3 02 80    . . W [....7...
02F0 03 B7 F3 04 86 42 BD F8 03 8D D0 20 09 0D 0A 00   .7...B=...P ....

0300 00 00 00 53 31 04 8D 3D 5F CE F3 02 8D 39 CE F3   ...S1..=_N...9N.
0310 08 8D 34 8D 32 FE F3 08 8D 2D 7A F3 04 26 F9 FF   ..4.2....-...&..
0320 F3 08 53 37 30 8D 20 33 FE F3 08 09 BC F3 7E 26   ..S70. 3....<..&
0330 86 8D 98 20 03 53 39 04 8D 58 7E F8 21 20 1F 20   ... .S9..X..! .
0340 6B 20 5F 20 3D 20 4B EB 00 20 4E BD F9 9C 7E F8   . _ = K.. N=....
0350 06 36 86 0C BD F8 03 86 D0 B7 F3 CC 32 39 86 03   .6..=...P7.L29..
0360 B7 F4 08 86 51 B7 F4 08 39 86 03 B7 F4 08 86 10   7...Q7..9..7....
0370 B7 F4 08 39 44 44 44 44 84 0F 8B 30 81 39 23 02   7..9DDDD...0.9#.
0380 8B 07 37 F6 F4 08 57 57 24 F9 B7 F4 09 33 39 8D   ..7...WW$.7..39.
0390 F1 08 A6 00 81 04 26 F7 39 A6 00 8D D7 A6 00 08   ..&...&.9&..W&..
03A0 20 D6 86 01 B7 F3 D4 20 06 7E F8 21 7F F3 D4 CE    V..7.T ...!..TN
03B0 FE 51 8D 97 BD F9 29 8D 98 FF F3 D5 8D A0 86 11   .Q..=.)....U. ..
03C0 8D C0 8D A5 8D 55 81 53 26 FA BD F8 03 8D 7B 81   .@.%.U.S&.=.....
03D0 39 27 D6 81 31 26 ED 7F F3 0A 8D 2D 80 02 B7 F3   9'V.1&.....-..7.
03E0 06 8D 18 BD FA 6B 8D 21 7A F3 06 27 47 7D F3 D4   ...=...!...'G..T
03F0 26 02 A7 00 A1 00 26 41 08 20 EB 8D 0C B7 F3 00   &.'.!.&A. ...7..

0400 8D 07 B7 F3 01 FE F3 00 39 8D 12 48 48 48 48 16   ..7.....9..HHHH.
0410 8D 0B 1B 16 FB F3 0A F7 F3 0A 39 20 2D 8D 2B 20   ..........9 -.+
0420 00 80 30 2B 47 81 09 2F 0A 81 11 2B 3F 81 16 2E   ..0+G../...+?...
0430 3B 80 07 39 7C F3 0A 27 8B FF F3 0C 86 20 BD F8   ;..9...'..... =.
0440 03 CE F3 0C BD F8 0F 7E F8 21 B6 F4 08 47 24 FA   .N..=....!6..G$.
0450 B6 F4 09 84 7F 81 7F 27 F1 39 8D 13 48 48 48 48   6......'.9..HHHH
0460 16 8D 0C 1B 16 FB F3 0A F7 F3 0A 39 7E F8 21 BD   ...........9..!=
0470 F8 00 20 AD 00 00 00 00 E6 06 A7 06 E7 04 E6 03   .. -......'.....
```

```
S11B02002002204E8603B7F4088651B7F408867EB7F3938D057EF82122
S11B02182015 30A600B7F300A601B7F301FEF300080808FFF3943936C5
S11B02308D13810D260D860A8D0B4F8D088D068D048D02323937F6F406
S11B024808575724F9B7F40933398D77202150554E4348202850292CF7
S11B0260204C4F414420284C292C2056455249465920285629 3F048DCD
S11B02781CBDF800815026022015814C26022054815626022050811BF7
S11B029026C07EF8212049BDF96CFEF37E09FFF37E8D3F8D3F86128DA5
S11B02A83DC6254F8D385A26FAFEF37CFFF308B6F37FB0F309F6F37EE2
S11B02C0F2F308262381102521201D30A600B7F300A601B7F301FEF315
S11B02D800080839206120612069206D2057205B860F8B04B7F3028067
S11B02F003B7F3048642BDF8038DD020090D0A00000000 5331048D3DD2
S11B03085FCEF3028D39CEF3088D348D32FEF3088D2D7AF30426F9FF66
S11B0320F3085337308D2033FEF30809BCF37E26868D98200353390479
S11B03388D587EF821201F206B205F203D204BEB00204EBDF99C7EF8FB
S11B03500636860CBDF80386D0B7F3CC32398603B7F4088651B7F4080E
S11B0368398603B7F4088610B7F40839444444444840F8B308139230245
S11B03808B0737F6F408575724F9B7F40933398DF108A600810426F7ED
S11B039839A6008DD7A6000820D68601B7F3D420067EF8217FF3D4CE8C
S11B03B0FE518D97BDF9298D98FFF3D58DA086118DC08DA58D558153FA
S11B03C826FABDF8038D7B813927D6813126ED7FF30A8D2D8002B7F35B
S11B03E0068D18BDFA6B8D217AF30627477DF3D42602A700A10026418A
S11B03F80820EB8D0CB7F3008D07B7F301FEF300398D12484848481655
S11B04108D0B1B16FBF30AF7F30A39202D8D2B200080302B4781092FE2
S11B04280A81112B3F81162E3B8007397CF30A278BFFF30C8620BDF86E
S11B044003CEF30CBDF80F7EF821B6F4084724FAB6F409847F817F2781
S11B0458F1398D1348484848168D0C1B16FBF30AF7F30A397EF821BD45
S1 70470F80020ADBF
S9030000FC
```

Appendix D (con't)

1) TSC SPACE VOYAGE

```
00E9    7E  F806    PDATA
00EC    7E  F986    OUTHR
00EF    7E  F982    OUTHL
00F2    7E  F9B4    OUTS
00F5    7E  F803    OUTCH
00F8    7E  F800    INCH
00FB    7E  F04A    RANDOM

0100    8E  F36F    STACK

04DC    20  20  53  44  54  45  3A  20  04
0DA0    20  20  5B  48  4C  44  3A  20  04
0DB6    0D  04
0E95    43  4B  2D  53  48  49  45  4C  44
        53  20  48  4F  4C  44  49  4E  47  04
0F5C    0D  04
```

2) TSC DEBUG PACKAGE

```
4106    7E  F800    INCH
4109    7E  F803    OUTCH
410C    7E  F821    MONITOR
410F        F404    KEYBOARD P/A

42A1    2B  0A      BMI PCRLF2      CHECKS FOR
42A3    A6  00      LDAA 0,X        CHAR. FROM
42A5    01          NOP             KEYBOARD

42BE    2B  FC      BMI WAITR1
42C0    01          NOP             CHECK FOR CHAR.
42C1    A6  00      LDAA 0,X

42C5    B1  4112    CMPA DEL        CNTRL C?

5999    0D  04
```

NOTE: An "ESC" Character will stop the display. A control 'C' will restart it. This is the only difference from standard operation. It is necessary due to the operation of the keyboard scan routine.

3) TSC DISASSEMBLER

```
1900    8E  F36F    STACK
190C    7E  F803    OUTCH
190F    7E  F800    INCH

197A    0D  04
```

4) TSC KLINGON CAPTURE

```
002F      7E   F806      PDATA
0032      7E   F803      OUTCH
0035      7E   F800      INCH
0038      7E   F04A      RANDOM
003B      7E   F821      CONTROL
003E      8E   F36F      STACK
03B2      0D   0A   0A   0A   0A   0A
03D5
040F
045A
047D
0499
04D1                                  (Replace all NULLS
051A                                   with LF's, which
055E                                   TVBUG will not
058B                                   Respond to).
05C7
0605
0672
0694
06B1
06E4
0724
074C
0765
0765
079F      0D   0A   0A   0A   0A   0A
```

5) TSC RANDOM NUMBER GENERATOR

Use Quick Load Function to enter. Start Location F04A. Change all 'A0' to 'F0' and seed F070,1,2,3,4 with Non-Zero Numbers.

6) TSC BATTLESHIP

```
0100      8E   F36F      STACK
0107      7E   F803      OUTCH
010A      7E   F800      INCH
010D      7E   F821      MONITOR

066E      0D   04

0732      0A   0A   0A   0A   0A   0A
```

7) TSC STOCKMARKET

```
0102      7E   F803      OUTCH
0105      7E   F800      INCH
0108      7E   F806      PDATA
```

```
010B    7E   F982      OUTCH
040E    7E   F986      OUTHR
0111    7E   F821      CONTROL
0114    7E   F04A      RANDOM
0117    8E   F36F      STACK

04C0    0D   0D   0A   0A   0A   0A   0A
04ED    0D   0A   0A   0A   0A   0A
058A    0D   0A   0A   0A   0A   0A
05B7    0D   0A   0A   0A   0A   0A
05E6    0D   0A   0A   0A   0A   0A
0621    0D   04
```

8) TSC HANGMAN

```
0102    7E   F803      OUTCH
0105    7E   F800      INCH
0108    7E   F806      PDATA
010B    7E   F9B4      OUTS
010E    7E   F821      CONTROL
0111    7E   F04A      RANDOM
0114    8E   F36F      STACK

021F    0D   04
0245    0D   0A   0A   0A   0A   0A
```

9) TSC ACEY- DUCEY

```
0034    7E   F04A      RANDOM
0037    7E   F806      PDATA
003A    7E   F803      OUTCH
003D    7E   F800      INCH
0040    7E   F982      OUTHL
0043    7E   F986      OUTHR
0046    8E   F36F      STACK

0223    0D   04

022A    0D

0241    0D

02AF    0D   0A   0A   0A   0A   0A

02FD    0D   0A   0A   0A   0A   0A
```

10) TSC 'CRAPS'

```
0022    7E   F04A     RANDOM
0025    7E   F806     PDATA
0028    7E   F803     OUTCH
002B    7E   F800     INCH
002E    7E   F982     OUTHL
0031    7E   F986     OUTHR
0034    7E   F821     CONTROL

0044    8E   F36F     STACK

0238    0D   04
0243    0D
0252    0D
025A    0D
0269    0D
0289    0D
02A1    0D   0A  0A  0A  0A  0A
032E    0D
```

11) TSC MASTERMIND

```
0042    7E   F800     INCH
0045    7E   F803     OUTCH
0048    7E   F806     PDATA
004B    7E   F9B4     OUTS
004E    7E   F982     OUTHL
0051    7E   F986     OUTHR
0054    7E   F821     CONTROL
0057    7E   F04A     RANDOM
005A    8E   F36F     STACK

0173    0D   0A  0A  0A  0A  0A
018A
01AB
01C1
01CF
01E7
0206    0D   0A  0A  0A  0A  0A
```

12) TSC CARD SHUFFLE AND DEAL

```
0061    BD   F04A      RANDOM
0100    8E   F36F      STACK
0115    BD   F803      OUTCH
011A    BD   F803      OUTCH
011F    BD   F803      OUTCH
012E    BD   F806      PDATA
0137    BD   F806      PDATA

013B    0D   04

0141    0D   0D   0D   04

014B    BD   F803      OUTCH
```

13) TSC NUMBER GUESS 1

```
0020    8E   F36F      STACK
0028    BD   F806      PDATA
002B    BD   F04A      RANDOM
0039    BD   F806      PDATA
003C    BD   F800      INCH
004A    BD   F806      PDATA
0058    BD   F806      PDATA
0060    BD   F806      PDATA
0068    BD   F806      PDATA
0071    BD   F806      PDATA
0076    BD   F986      OUTHR
007C    BD   F806      PDATA
007F    BD   F800      INCH
0086    BD   F821      CONTROL
0089    0D   0A   0A   0A
00A6    0D   0A   0A   0A
00BA    0D   0A   0A   0A
00D3    0D   0A   0A   0A
00DF    0D   0A   0A   0A
00EC    0D   0A   0A   0A
00F9    0D   0D   0A   0A
011A    0D   0A   0A   0A
012C    0D   0A   0A   0A
```

14) TSC NUMBER GUESS II

```
0042    7E   F803     OUTCH
0045    7E   F800     INCH
0048    7E   F806     PDATA
004B    7E   F982     OUTHL
004E    7E   F986     OUTHR
0051    7E   F821     CONTROL
0054    7E   F04A     RANDOM
0057    8E   F36F     STACK

0125    0D   04
0155    0D   0A  0A  0A  0A  0A
016B
0183
01ED    0D   0A  0A  0A  0A  0A
```

15) TSC HURKLE

```
0022    7E   F04A     RANDOM
0025    7E   F806     PDATA
0028    7E   F800     INCH
002B    7E   F803     OUTCH
002E    7E   F821     CONTROL

0038    8E   F36F     STACK

013C    0D   04
015D    0D
01A3    0D
```

16) TSC ROVER

```
0022    7E   F04A     RANDOM
0025    7E   F806     PDATA
0028    7E   F800     INCH
002B    7E   F803     OUTCH
002E    7E   F821     CONTROL
0049    8E   F36F

01B8    0D   04
01C3    0D
01D2    0D
022D    0D
```

17) TSC SWITCH

```
0102    7E   F800     INCH
0105    7E   F806     PDATA
0108    7E   F982     OUTHL
010B    7E   F986     OUTHR
010E    7E   F9B4     OUTS
0111    7E   F821     CONTROL
0114    7E   F04A     RANDOM

0215    0D   04
023D    0D   0D   0A   0A   0A   0A
```

18) TSC CHOM

```
0042    7E   F803     OUTCH
0045    7E   F800     INCH
0048    7E   F806     PDATA
004B    7E   F9B4     OUTS
004E    7E   F821     CONTROL
0051    8E   F36F     STACK

0178    0D   04
01A2    0D   04
01B8    0D   0D   0A   0A   0A   0A
```

19) TSC 10K BASIC

```
0106    7E   F821     CONTROL (EXIT)
0109    7E   F800     INCH
010F    7E   F803     OUTCH
0112    7E   F000     TINCH  EXTERNAL ROUTINES
0115    7E   F020     TOUCH

0042         F408
0888    0D   04

01F8    BD   01   C7   20   05
0252    BD   01   C7   20   06
0496    01   01   01
01C7    B6   F4   04   81   03   27   0A   39
```

A clear screen may be accomplished by the following 2 statements in a program.

```
Print CHR $(12)
Poke HEX("F3CC"), HEX("DØ")
```

TOUCH
TINCH

```
C8A3 F000   37 C6 51 F7 F4 08 F6 F4 08 57 57 24 F9 B7 F4 09 33 39
C8B5 F020   86 03 B7 F4 08 86 10 B7 F408 B6 F408 47 24 FA B6 F4 09
            39
```

These Routines at $F000 are necessary for cassette Save and Load operation with TVBUG. They may be placed any where in memory. The Lower Stack area is most convenient.

External Plot Routings may be added by using the USR Function.

20) TSC RELOCATOR

```
        01F0      7E   F020      KCIN

        021F      7E   F803      OUTCH
        0222      7E   F800      INCH
        0225      7E   F821      CONTROL

        0242      8D   AC

        02AA      BD   F020      KCIN

        02B1      BD   F020      KCIN

        05B1      0A   0A   0A   0A   04
        05B6      0A   0A   0A   0A   04
        05BB      0D   04
```

Note: Punch From 1F0 - 6B3 to include Lower jump

```
F020 86  10  B7  F4  08  B6  F4  08  47  24  FA  B6  F4  09   39
```

Note: This Relocator will Load tape programs only from the S1-S9 format. If TVBUG-type tapes are to used, First Load the tape, then use the relocator to move the program. Also note that the Offset may be used for TVBUG tapes, but data Blocks will not be preserved.

21) TSC TEXT EDITOR

```
        0206      7E   F800      INCH
        0209      7E   F803      OUTCH
        020C      7E   F020      TINCH
        020F      7E   F000      TOUCH

        0458      0D   04

        098B      7E   F821      CONTROL
        0D92      0D   0A   0A   0A   0A   0A   0A

TOUCH   F000 37   C6   51   F7   F4   08   F6   F4   08
             57   57   24   F9   B7   F4   09   33   39

TINCH   F020 86   03   B7   F4   08   86   10   B7   F4
             08   B6   F408 47   24   FA   B6   F4   09   39
```

22) TSC MNEMONIC ASSEMBLER

```
0300      8E   F36F      STACK

031B      7E   F821      CONTROL

0320      7E   E803      OUTCH
0323      7E   E000      TOUCH

07CF      0D   04
11D1      0D   04
156F      0D   0A   0A   0A   0A   0A

F000      37 C6  51  F7  F408  F6  F408  57  57  24  F9  B7  F4  09  33
          39
```

23) TSC TEXT PROCESSOR

```
0203      7E   F803      OUTCH
0206      7E   F800      INCH
0209      7E   F821      CONTROL

1543      0D   04

1471      B6  F404  2A  01  39          NEW BREAK ROUTINE
          B6  F404  81  03  26  F8       FOR TVBUG
          CE  1592  7E  0C  D8  01       (SEE BELOW)
```

This break routine shold be used with TVBUG instead of the current one.

```
0471      B6  F404  TSTBRK    LDAA  PIA
0474      2A  01              BPL   TSTBR4        CHARACTER?
0476      39        TSTBR2    RTS
0477      B6  F404  TSTBR4    LDAA  PIA
047A      81  03              CMPA  #$03         CNTRL 'C'?
047C      26  F8              BNE   TSTBR2
047E      CE  1592            LDX   #BRKSTR      POINT TO STRING
0481      7E  0CD8            JMP   STOP1        OUTPUT IT
0484      01                  NOP
```

MICRO CHROMA 68
THE NEW "BUG"
FROM MOTOROLA
TVBUG 1.2®

NMOS Microcomputer Systems Applications
Austin, Texas

Prepared By
Tim Ahrens
Dave Williamson
Software by John Dumas

MOTOROLA INC.

Introduction:

In a continuing effort to provide support for the users of MC6800's and MC6800 Family Peripherals, Motorola has introduced products which will interface directly with a standard color television receiver.

Introducing Micro Chroma 68

This package contains the major kit components necessary to make a standard, unmodified color television receiver into a monitor for a highly sophisticated computer system.

1) MC6808 Microprocessor with Clock Generator (MPU)
2) MC6846P3 RIOT with TVBUG
3) MC6821 Peripheral Interface Adapter (PIA)
4) MC6850 Asynchronous Serial Interface Adapter (ACIA)
5) MC6847 Video Display Generator (VDG)
6) MC1372 Video RF Modulator
7) Data sheets on all parts
8) Hardware/software descriptions
9) Complete system schematic
10) Complete parts list

I. VDG SYSTEM HARDWARE

The TVBUG system as shown in Figure 1, is configured for a minimum hardware application. This system includes the MC6808 microprocessor, the MC6846 Timer, I/O, ROM with TVBUG 1.2, keyboard interface chip, the MC6821 PIA, MC6850 ACIA for Serial I/O, the MC6847 Video Display Generator, and an MC1372 Video Radio Frequency (RF) modulator. In addition to this, the schematic shows 1K bytes of RAM for display, the STACK RAM area, and the user RAM. A Kansas City Standard 300 baud cassette tape interface is also shown for ease of program storage.

BASIC OPERATION

The new MC6808 (microprocessor with clock) is used as the microprocessor and is coupled with the MC6846P3 (ROM, I/O, and timer) for a permanent monitor system. The MC6846P3 (TVBUG) has been programmed to "talk" to the MC6847 VDG display RAM as an output port, and a standard ASCII keyboard is connected to the MC6821 for data input. The actual hook-up of the MPU is standard to almost any dedicated system with a few exceptions. The MCM2114's cannot have the R/$\overline{W}$ pulse applied to them during the normal MC680X write cycle, as any data which is on the bus will be written to the RAM. To solve this problem, the combination of $\phi 2$ with R/$\overline{W}$ will provide a delayed R/W pulse within the MCM2114 specifications.

All address decoding for 8K bytes of RAM has been done on board by use of the 74LS138 three to eight line decoder. With the decoding scheme shown, up to 8K bytes of user RAM can be placed on the bus without further decoding, and decoding for every 1K block of memory from D000 to FFFF has also been provided. This latter decoding will provide the user with an option of including up to an additional 5K of bytes of display RAM to utilize the full 256 X 192 element graphic mode of the MC6847. One note of caution though, no additional decoding (i.e. 74LS138's) can be supported by the higher order address bus lines without using the traditional 8T26-28 bus extenders. An alternative to this problem is to do a different decode with the present 138's, and use MCM6641's (4K X 1) RAM chips. This would mean that additional RAM could be added, but only in 4K blocks. The use of MCM2114's in this project was selected because of the MCM2114's ability to be added in 1K blocks, thereby reducing the initial cost of the project.

The "Break" key on the board through the use of cross coupled NAND gates provide the NMI (Non Maskable Interrupt) input with a debounced key closure. Within the TVBUG Monitor, the NMI vector causes the current contents of the registers within the MPU to be displayed on the TV screen.

The RESET is a normally open switch which does not need to be debounced due to the RC time constant of the 3.3K Ω and 100 μF capacitor, and the Schmitt trigger input on the MC6808.

The MC6821 PIA is configured to receive key closures from a standard ASCII keyboard. The negative going strobe must be at least a duration of 100 ms due to the type of polling used to scan for a keyboard input. Perhaps a more conventional way to poll the keyboard would have been through the use of the PIA's IRQ line; but by dedicating the IRQ vector to a keyboard input routine, the capability of using this line for a user function would have been lost.

Although the ACIA in this system is dedicated to a Kansas City Standard tape interface unit, the inputs and outputs could be switched to provide the user with a different use for the serial I/O port.

In the transmit mode, the clock for the ACIA, and tone for the Kansas City Standard, is provided by an MC1455. This 4800 Hz is fed into the transmit circuitry, and when combined with the TX (transmit) data, provides the traditional 2400 Hz for a Mark (1) and 1200 Hz for Space (0) required by the Kansas City Standard. This output data is not the S1-S9 data format as used by some other systems, but a binary format which is both faster, and compatible with the many Motorola MEK6800D2 kits now in use. The speaker audio is brought onto the board and fed into the input of an MC14584. This hex Schmitt trigger provides about 0.5 volts of hysteresis when operated at 5 volts, and does an excellent job of squaring up the incoming audio tones. Following this squaring of the tones, the data is processed and fed into the RX (receive) data pin of the ACIA. Connected to this is a leading edge detector which is used to reset the binary counter which feeds the RX clock signal. By doing the clock recovery in this manner, the binary counter is assured to be in synchronization with the data and provide the proper clock pulse to the ACIA at the correct time.

The cassette interface provided in this documentation has proven to be a very effective and reliable method for generating and recovering data from audio tapes, while using only six CMOS packages.

THE VDG AND ASSOCIATED CIRCUITRY

The MC6847 VDG operates like a microprocessor. It accesses its own RAM by putting out the specific addresses on the bus. Because of this, the VDG address bus must be separate from the MPU's address bus, but both must have the ability to address the common display RAM. By using the 8T97 tri-state buffers on the address lines, wherever the "A" signal goes low, signifying that the MPU is talking to the display RAM, the VDG is turned off. In addition, the bidirectional data bus buffers must have the same type of control. The other inverters and NAND gates (U19-A, B and U16-A, B, C) provide the display MCM2114's with the correct R/$\overline{W}$ signal. The VDG directly generates the chrominance and luminance analog (R-Y, B-Y, Y) output levels to be fed to the MC1372 Radio Frequency (RF) modulator.

The new MC1372 modulator contains a chroma subcarrier oscillator (3.579 MHz to be used as a system clock), and all other circuitry necessary to support direct RF modulation to any standard color television set. As shown in the schematic, the MC1372 clock is buffered and fed directly to the MC6847 and MC6808 EXTAL inputs where it is used as a system clock. This provides an internal system clock of approximately 895 kHz ($\phi 2$).

The Tank L-C values shown (56 pF and .1μH) will provide a center frequency of 67.25 MHz (channel 4). The component values required for channel 3

operations are 75 pF and .1μH. This provides a center frequency of 63 MHz. The 75 Ω output may be fed directly to the set, or through a 75 to 300 Ω matching transformer available wherever televisions or electronic components are sold. Either method MUST include a 60 dB switch to minimize RF radiation problems (TVI). This assures that the antenna is disconnected while using TVBUG.

CONSTRUCTION HINTS

There are several devices on the schematic which require adjustment, these include L1, R10, C4, and R6. The following is a "tune-up" procedure for these.

The value of L1 determines the frequency of operation, in other words, which channel you are on. This coil can be bought as a .1μH tuneable coil, or fabricated on almost any type of coil form which uses a ferrite slug for tuning. The prototypes were made on forms 1/8 inch in diameter with 3 1/2 turns #30 gauge wire wrap wire. After the coil has been made and placed in the circuit, turn the television on to the desired channel (3 or 4), and turn the slug until a picture is presented on the screen. If all that is seen is "confetti" on the set, the coil could possibly be resonating at the wrong frequency. Change the channels until a picture is seen. Turning the slug down into the coil lowers the frequency (and channel), and unscrewing it produces the opposite effect. If the picture is seen at a channel which is not desired, adding more turns (one or two) will lower the frequency.

The duty cycle adjustment varies the actual duty cycle of the 3.58 MHz clock. This is an optional part, as the MC1372 is internally set for a 50% duty cycle. Adjustment should be made while observing a picture, and tuning the pot until the best picture is obtained.

Y1's frequency adjustment trimmer, C4, should be set for 3.57954 MHz with a frequency counter, or adjusted until the television set responds with the correct colors.

The audio cassette standard has an adjustment, R10, which should be tuned for best data at pin 2 of the ACIA. This adjusts a time constant, and some chopping of the waveform will be noticed until this adjustment has been made.

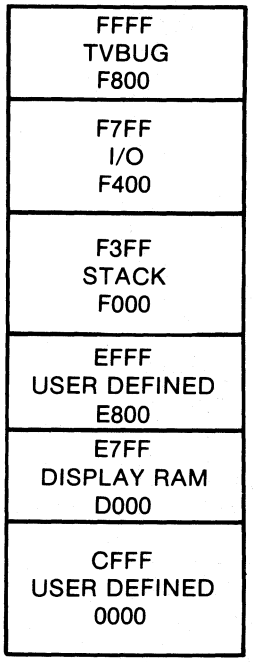
Care should be taken in parts placement, as there are several areas where there are low level analog signals present. Fast digital signals do a good job of interfacing with these, so keep as much space as possible between them. Also be sure to adequately bypass as shown on the schematic.

Conclusions: This minimum configuration provides the user with a software and low density (64 X 32 element) color graphics development system. By expanding this system with a minimum number of components and enough support RAM (both display and user RAM), a full color graphics system can be implemented for any type of home computer, game, or low cost color graphics system.

II. TVBUG 1.2 SOFTWARE

The TVBUG software monitor is provided in the MC6846 combo chip along with an eight-bit parallel interface and a programmable timer. It is designed to implement the MC6847 and MC1372 in a MC6800/01/02/03/08 based system. Micro Chroma 68 provides the user with an operating/debug system, parallel interface to an ASCII keyboard, a Kansas City Standard for an audio cassette interface, and an interface to an unmodified color TV receiver.

HARDWARE CONFIGURATION MEMORY MAP



NOTE: All addresses are in hexidecimal.

SOFTWARE OPERATION

The operating procedure for each command is given in the following paragraphs. All of these commands assume that the system is under TVBUG firmware control and the last data displayed on the screen was "TVBUG" followed by a carriage return-line feed (CRLF), and a blinking cursor.

RESET FUNCTION

The RESET function is used when power is first applied, anytime TVBUG firmware loses program control, and before a PUNCH, LOAD or VERIFY command is used. To use this function:

Press the reset switch. TVBUG firmware will gain program control and display "TVBUG" followed by CRLF and blinking cursor. The TVBUG monitor is ready for an input.

TVBUG COMMANDS

The following are TVBUG commands. Each command is one letter followed by an optional modifier (address or data). The modifier is always hexadecimal with leading zeros assumed. The modifying field is terminated with a non-hex entry (e.g. space bar). The modifier will be

either two or four hex digits depending on the command mode. The TVBUG firmware may be used to debug and evaluate a user program and to perform the following functions:

1) G — GO TO ADDRESS "N" (USER PROGRAM)
2) L — LOAD KANSAS CITY STANDARD TAPE (J-BUG FORMAT)
3) P — PUNCH DUMP KANSAS CITY STANDARD (J-BUG FORMAT)
4) V — VERIFY KANSAS CITY TAPE
5) M — MEMORY CHANGE
6) E — EXAMINE A BLOCK OF MEMORY
7) Q — QUICK LOAD OF HEX DATA
8) F — FILL A BLOCK OF MEMORY
9) O — OFFSET CALCULATION
10) R — DISPLAY CONTENTS OF MPU REGISTERS
11) Z — CLEAR SCREEN AND INITIALIZE I/O
12) S — SET A BREAKPOINT WITH ADDRESS "N"
13) U — UNSET BREAKPOINT WITH ADDRESS "N"
14) D — DELETE ALL BREAKPOINTS
15) B — PRINT OUT ALL BREAKPOINTS
16) N — TRACE THE NEXT INSTRUCTION
17) C — CONTINUE EXECUTION FROM THE CURRENT LOCATION
18) T — TRACE "N" INSTRUCITONS
19) !, ", # — USER DEFINED FUNCTIONS

G — Go To User Program Function

This function allows the user to exeucte a USER program. To use this function type a "G", starting address, and return. The firmware will execute a USER program.

L — Load Function

This function allows the user to load a Kansas City Standard formatted audio cassette tape. This includes tapes punched using Motorola's J-Bug monitor. To use this function:

1. Press Reset
2. Type "L". The firmware will CRLF and ask for an offset, (16 bits, Hexadecimal, with leading zeros assumed).
3. Type "return". Start the tape by pressing "play" on the cassette recorder. Insure that the recorder "ear" to P.C. board "ear" is connected.
4. After approximately 40 seconds of leader, the firmware will print a name if any, and a "B" for each 256 bytes and a "B" for the remainder, if any. If the data was not stored into memory correctly, the "B" is followed by the message, "MEMORY BAD" and the firmware will return to TVBUG program control.

P — Punch Dump Function

This function allows the user to store data from memory on audio cassette tape using the Kansas City Standard. To use this function:

1. Press Reset.
2. Type "P". The firmware will CRLF and ask for a beginning address.
3. Enter beginning address and type a space. The firmware will ask for a ending address.
4. Enter ending address. The firmware will CRLF and ask for a name.
5. Enter the name. The name may be up to thirty-two (31 + CR) characters long. If tape must be read by a J-Bug monitor, do not use "B" or "G" in the name as these characters are interpreted by the J-Bug firmware as control characters.
6. Connect the tape recorder to the P.C. board "mike" to "mike" and start recording.
7. Type return. The firmware will print leader (F's), followed by an 80 (Start Char.), Name (ASCII Code), Byte count, Starting Address, and "42" (ASCII "B") followed by data. A short leader terminated with "42" (ASCII "B") will be printed for each 256 bytes.

V — Verify Kansas City Tape

This function is used to verify a PUNCH or LOAD operation. To use this function:

1. Press Reset.
2. Enter a "V". The firmware will CRLF and ask for an offset.
3. Enter the offset. The offset must be the difference between the existing start address and the desired start address; if none, type a space.
4. Set up the tape recorder as shown in the load function.
5. The firmware will print file name, CRLF, and print a "B" for each 256 bytes. If the data on the tape and the contents of the memory do not agree, the firmware will print "MEMORY BAD" and return to TVBUG program control.

M — Memory Change Function

This function will examine a location in memory, change the contents if desired, and return the contents to memory in that order. To use the MEMORY CHANGE function:

1. Enter an "M".
2. Enter the address to be changed and press line feed. TVBUG firmware will CRLF and print the address followed by data.
3. Enter new data if desired. Line feed will then return data to memory and open the next location. Up arrow (^) will return data to memory and open location minus one. To return to TVBUG control program, press the return key.

```
TVBUG
M O
0000 XX 00
0001 XX 00
TVBUG
```

E — Block Memory Examine Function

This function allows the user to display a block of memory on the screen. To use this function:

1. Enter an "E".
2. Enter the beginning address of the block to be examined and type a space. The firmware will ask for ending address.
3. Enter an address and type a space.
4. The firmware will CRLF and print the beginning address and contents of the first eight memory locations. Underneath the contents of each location is a period. If the data at that location is an ASCII character, the character will be printed underneath the data. Each time a space is entered, the next eight locations will be

printed until it reaches the ending address; at which time the firmware will return to the TVBUG control program.

```
TVBUG
E
BEG ADR? 0     END ADR? F


0000    54  56  20  42  55  47  XX  XX
        T   V       B   U   G   .   .
0008    XX  XX  XX  XX  XX  XX  XX  XX
TVBUG   .   .   .   .   .   .   .   .
```

Q — Quick Load Function

This function allows the user to enter blocks of hex data using the MEMORY EXAMINE function. To use this function:

1. Type "Q". The firmware will CRLF and ask for the beginning address.
2. Enter beginning address and type a space. The firmware will ask for the ending address.
3. Enter ending address and type return. The firmware will CRLF, print the beginning address and wait for data.
4. Enter hex data followed by a space. The firmware will CRLF on the eighth location, print the address and wait for data. When the ending data has been entered, the firmware will return to TVBUG control program.

Typical Display:

```
TVBUG
Q
BEG ADR? 0     END ADR? F

0000    XX  XX  XX  XX  XX  XX  XX  XX
0008    XX  XX  XX  XX  XX  XX  XX  XX
TVBUG
```

F — Memory Fill Function

This function allows the user to fill a block of memory with a character. To use this function:

1. Enter an "F". The firmware will CRLF and ask for the beginning address.
2. Enter the beginning address and type a space. The firmware will ask for an ending address.
3. Type a space. The firmware will CRLF and ask for a character.
4. Enter the desired character and type return. The firmware will write the character into each of the defined memory locations and return to TVBUG program control.

Typical display for MEMORY FILL function:

```
TVBUG
F


 BEG  ADD  XXXX   END   ADD   XXXX
CHAR? XX
TVBUG
```

O — Offset Calculation Function

This function allows the user to calculate sixteen bit offsets. If the offset is outside the eight bit "branch" limits, the firmware will print the offset followed by the message "TOO FAR". This function simplifies the calculation of offsets for branch instructions. To use this function:

1. Type an "O". This firmware will CRLF and ask for the beginning address.
2. Enter the address of the branch op code and type a space. The firmware will CRLF and ask for the ending address.
3. Enter the address of branch destination and type return. The firmware will CRLF, print the offset and return to TVBUG control program. Offsets will be printed as a sixteen bit word. The least significant eight bits will be the offset.

1) Positive Offset:

```
O
BEG ADR? 0 END ADR? F
OFFSET = 000D
TVBUG
```

2) Negative Offset:

```
O
BEG ADR? F END ADR? 0
OFFSET = FFEF
TVBUG
```

3) Offset Outside of an eight bit branch:

```
O
BEG ADR? 0 END ADR? 82
OFFSET = 0080 TOO FAR!
TVBUG
```

R — Print contents MPU Registers

This function allows the user to examine the MPU registers by reading them from the stack. To use this function type "R". The firmware will place contents of the MPU registers onto the stack RAM and then place them on the screen in the following format:

```
      CC    B     A     X     P     S
 XX   XX    XX   XXXX  XXXX  XXXX
```

Where:

CC = condition code register
B = B accumulator
A = A accumulator
X = index register
P = program counter
S = stack pointer

Z — Clear Screen Function

To use this function type "Z". The firmware will fill the display memory block with a space character, clear the screen, initialize the system I/O ports, and return to TVBUG program control.

BREAKPOINTS

There are seven TVBUG commands dealing with breakpoints.

1. S — SET BREAKPOINT WITH ADDRESS "N"

2. U — UNSET BREAKPOINT WITH ADDRESS "N"
3. D — DELETE ALL BREAKPOINTS
4. N — NEXT INSTRUCTION
5. T — TRACE "N" INSTRUCTIONS
6. C — CONTINUE EXECUTION FROM CURRENT LOCATION
7. B — PRINT OUT ALL BREAKPOINTS

S — Set A Breakpoint With Address "N"

To set a breakpoint type an "S" followed by the address, then type return. The firmware will print the breakpoint address with up to seven additional breakpoints that might be set, and return to TVBUG program control.

NOTE: Breakpoint 0000 is illegal.

Typical display for Setting Breakpoints:

```
TVBUG
S 10
0010
TVBUG
S20
0010 0020
TVBUG
S 30
0010 0020 0030
TVBUG
```

U — Unset A Breakpoint With Address "N"

To unset a breakpoint type a "U" followed by the address, then return. The firmware will remove the breakpoint and return to TVBUG control program.

Typical Display for Unsetting Breakpoints:

```
TVBUG
U 10
TVBUG
U 20
TVBUG
```

D — Remove All Breakpoints

To remove all breakpoints, type a "D". The firmware will remove the breakpoints and return to TVBUG program control.

B — Print Out All Breakpoints

To examine breakpoints type a "B". The firmware will print all breakpoints and return to TVBUG control.

NOTE: The following commands assume that a program has been executed and halted at a breakpoint.

N — Trace The Next Instruction

This command allows the user to single step through a series of instructions. To use this command, type an "N". The firmware will execute the NEXT instruction and print the contents of the MPU Registers. It will then return to TVBUG program control.

Typical Display for NEXT Instruction:

```
XX  XX  XX  XXXX  0030  XXXX
TVBUG
N
XX  XX  XX  XXXX  0032  XXXX
TVBUG
```

C — Continue

The CONTINUE command is used to step the program from breakpoint to breakpoint. To use this command, type a "C". The firmware will execute the user progam from the current location to the next breakpoint, and print out the contents of the stack.

Typical Display for CONTINUE Instruction:
(Breakpoints set at 0030, 0040, 0050)

```
XX  XX  XX  XXXX  0030  XXXX  TVBUG
C
XX  XX  XX  XXXX  0040  XXXX  TVBUG
C
XX  XX  XX  XXXX  0050  XXXX  TVBUG
```

T — Trace "N" Instructions

This command works exactly like the NEXT command, except that after entering "T" the firmware will ask for the number of instructions to be traced. Enter the hexadecimal number and the firmware will print out contents of target stack at each instruction. Then it will return to TVBUG program control.

Typical Display for TRACE Instructions:

```
TVBUG
XX  XX  XX  XXXX  0030  XXXX
T3
XX  XX  XX  XXXX  0033  XXXX
XX  XX  XX  XXXX  0036  XXXX
XX  XX  XX  XXXX  0039  XXXX
```

USER DEFINED FUNCTIONS

TVBUG contains three user defined jumps that may be called from the keyboard and two user defined jumps called by the monitor. All jumps are initialized with a RESET. However, if the user wishes to prevent these vectors from being lost on RESET, the stack RAM may be hardware deselected from F390 to F29F inclusive and a small ROM, containing the permanent vectors, patched over these locations (see listing).

To use the keyboard jumps, type the appropriate character (!, ", #). Firmware will then execute the program from the address stored in temporary RAM at the following locations:

CHAR.	INST. (7E)	HIGH BYTE	LOW BYTE
!	F396	F397	F398
"	F399	F39A	F29B
#	F39C	F39D	F39E

USER INPUT FUNCTION

This function allows the user to insert a user routine into the monitor input loop. Each time the monitor goes around its input loop it checks the user input three byte vector. Since it is initialized to RTS, the monitor will ignore this vector until the user changes it. The three temporary RAM locations reserved for the user input vector are:

INST. (7E)	HIGH BYTE	LOW BYTE
F390	F391	F392

To use this function, first write user vector into stack.

Example:

```
LDAA: User vector (High Byte)
STAA: $F391
LDAA: User vector (Low Byte)
STAA: $F392
LDAA: #$7E
STAA: $F390
:     Initialization complete
```

If this vector is entered with the keyboard, the jump instruction (7E) must be entered last.

The USER INPUT routine must set the carry bit if there was a user input. If there was no input it must clear the carry bit. All user I/O routines must end with RTS.

USER OUTPUT FUNCTION

This function allows the user to insert a user output routine into the monitor output routine. Each time the monitor performs its OUTCH (output character) routine it checks the three temporary RAM locations reserved for the user output vector.

Since these locations are initialized to RTS, the monitor will ignore them until they are changed by the user.

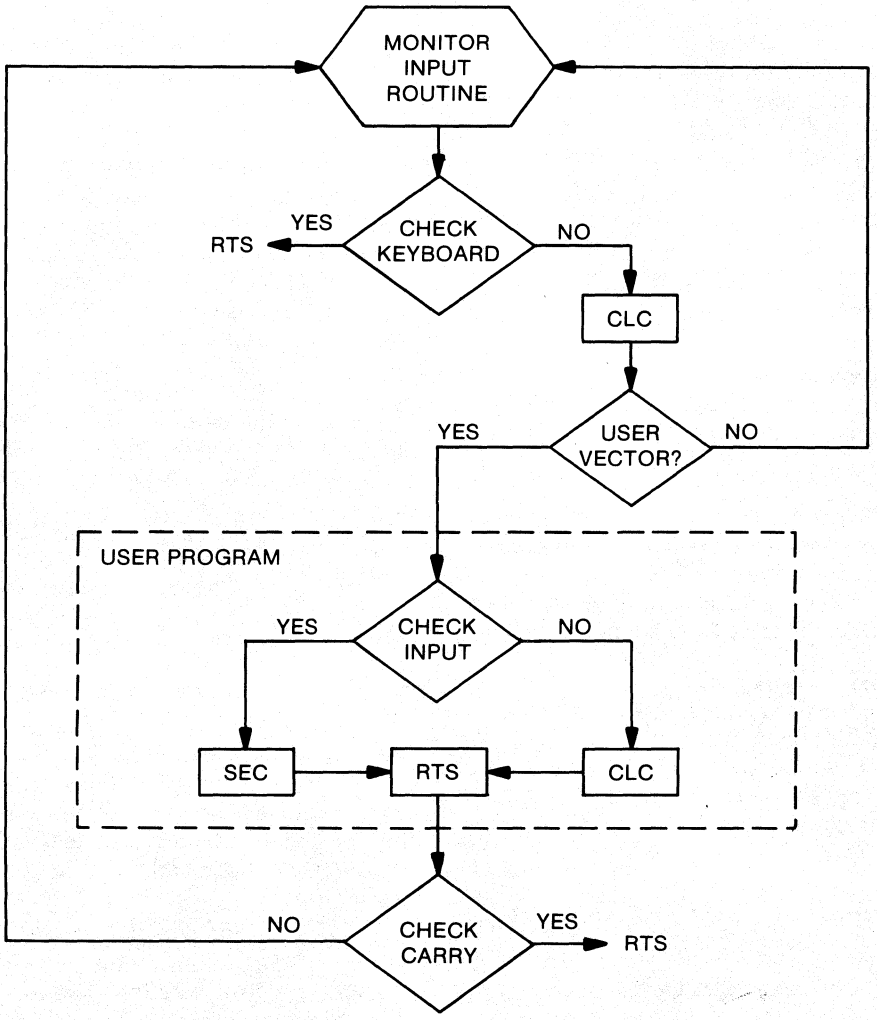
INST. (7E)	HIGH BYTE	LOW BYTE
F393	F394	F395

To use this function, write the jump vector into temporary RAM. If the vector is left in temporary RAM, I/O devices such as a printer or modem may be controlled on the fly by changing the instruction location (F393) from the jump (7E) to RTS (39). All user I/O routines must end with RTS.

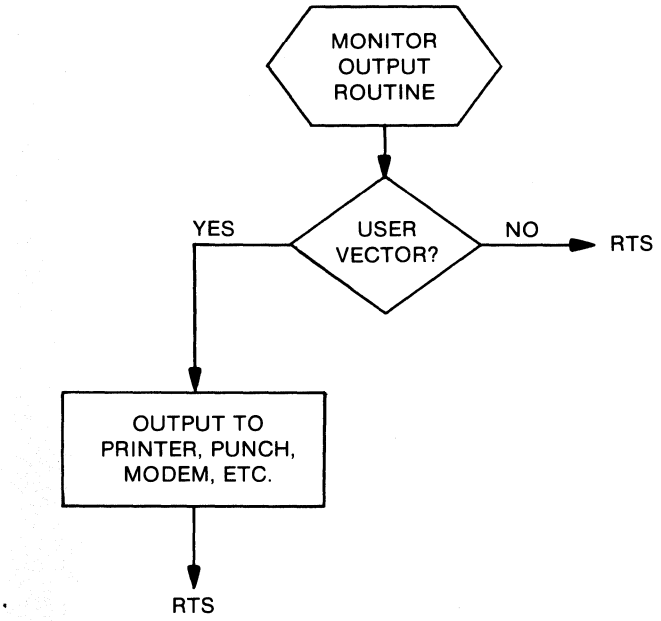
The following software listing contains the monitor jump table and temporary RAM locations as well as other useful routines.

Only the jump table and temporary RAM locations will be guaranteed on future versions of TVBUG. Caution should be exercised when using any other routines.

FLOW CHART FOR USER INPUT FUNCTION



FLOW CHART FOR USER OUTPUT ROUTINE



```
 NAM    TVBUG
 TTL    A VDG MONITOR FOR 6800,01,02,03,08
REV 1
COPYRIGHT (C) 1978 BY JOHN DUMAS
FOR MOTOROLA INC.
6 APR 78

TVBUG (TM) MOTOROLA

AUSTIN, TEXAS
MICROCOMPUTER CAPITAL OF THE WORLD!

CURRENT REVISION DATE = SEPT 08 1978

-----FOLLOWING ARE TVBUG COMMANDS-----
  EACH COMMAND IS 1 LETTER FOLLOWED
  BY AN OPTIONAL MODIFIER (ADDRESS OR
  DATA). MODIFIER IS ALWAYS HEX WITH
  LEADING ZERO(S) ASSUMED.  MODIFIER
  FIELD IS TERMINATED WITH A NON-
  HEX ENTRY(I.E.SPACE BAR). MODIFIER
  WILL BE EITHER 2 OR 4 HEX DEPENDING
  UPON COMMAND MODE.

L  LOAD K.C. STANDARD TAPE (D2 FORMAT)
M  MEMORY CHANGE
P  PUNCH K.C. STANDARD TAPE (D2 FORMAT)
R  DISPLAY CONTENTS OF TARGET STACK
     CC   B   A   X   P   S
B  PRINT OUT ALL BREAKPOINTS
F  FILL MEMORY BLOCK
C  CONTINUE EXECUTION FROM CURRENT LOCATION
N  NEXT INSTRUCTION TRACE
T  TRACE 'N' INSTRUCTIONS
G  GO TO LOCATION 'N'
D  DELETE ALL BREAKPOINTS
U  UNSET BREAKPOINT WITH ADDRESS 'N'
E  EXAMINE BLOCK OF MEMORY
Q  QUICK LOAD OF HEX DATA
O  OFFSET CALCULATION (BRANCH)
S  SET A BREAKPOINT WITH ADDRESS 'N'
V  VERIFY KC TAPE (D2 FORMAT)
Z  CLEAR TV SCREEN
!  USER FUNCTION #1
"  USER FUNCTION #2
#  USER FUNCTION #3
```

JUMP TABLE TO MONITOR

```
F800 7E FEE1  A        JMP     INCHI    INPUT CHAR
F803 7E FF0F  A        JMP     OUTCHI   OUTPUT CHAR
F806 7E F995  A        JMP     PDATAI   OUTPUT STRING
F809 7E F929  A        JMP     BADDR    INPUT HEX
F80C 7E FFC9  A        JMP     SCROLL   UP 1 LINE
F80F 7E F980  A        JMP     OUT4HS   OUTPUT 4 HEX+SPACE
F812 7E F982  A        JMP     OUT2HS   OUTPUT 2 HEX+SPACE
F815 7E FFA8  A        JMP     INIT     CLEAR SCREEN
F818 7E F96C  A        JMP     GETADR   GET START & STOP ADR
F81B 7E FF7F  A        JMP     SAVE     SAVE AREG 0,X
F81E 7E FE86  A        JMP     SYNCLD   LOAD AREG 0,X
F821 7E FB20  A        JMP     CONTRL   RESTART  POINT
```

INTERRUPT VECTORS

```
FFF8                           ORG     BASORG+*7F8
FFF8    F824  A                FDB     TO
FFFA    F82E  A                FDB     SFEI
FFFC    F829  A                FDB     POWDWN
FFFE    FAD7  A                FDB     START
```

THE FOLLOWING ARE INITALIZED AT START

```
F37A    0002  A IOV     RMB     2       I/O INTERRUPT POINTER
F37C    0002  A BEGA    RMB     2       PRINT/PUNCH START LOC
F37E    0002  A ENDA    RMB     2       PRINT PUNCH STOP LOC
F380    0002  A NIO     RMB     2       NMI INTERRUPT POINTER
F382    0002  A SP      RMB     2       USER STACK POINTER
F384    0002  A SWI     RMB     2       LEVEL 1 SWI VECTOR
F386    0002  A SWI2    RMB     2       LEVEL 2 SWI VECTOR
F388    0008  A BRINS   RMB     8       COND BRANCH STORAGE
```

USER I/O VECTORS
INITALIZED TO RTS
USER MUST END HIS ROUTINES
WITH RTS OR ALL IS LOST!!!!

```
F390    0003  A USRINP RMB     3       USER INPUT ROUTINE
F393    0003  A USROUT RMB     3       USER OUTPUT ROUTINE
                *
F396    0003  A USR1    RMB     3
F399    0003  A USR2    RMB     3
F39C    0004  A USR3    RMB     4
                *
        F3A0  A BRANEN  EQU     *
```

```
F3C5    0001  A ASAVE  RMB   1
F3C6    0002  A TEMP   RMB   2
F3C8    0002  A TEMP2  RMB   2
F3CA    0002  A SAVEX  RMB   2
F3CC    0002  A NEXTBY RMB   2
F3CE    0001  A XHI    RMB   1
F3CF    0001  A XLOW   RMB   1
F3D0    0002  A SSAVE  RMB   2
F3D2    0001  A CHRCNT RMB   1
F3D3    0001  A TERMCH RMB   1
F3D4    0001  A VFLAG  RMB   1
F3D5    0002  A OFFADR RMB   2
F3D7    0001  A FLAGK  RMB   1
```

```
        PRINT DATA POINTED AT BY X-REG
        ON EXIT A=4,B=UNCHANGED,X=X+1

F992 8D 22 F9B6 PDATA2 BSR     OUTCH
F994 08                INX
F995 A6 00     A PDATA1 LDAA    X
F997 81 04     A        CMPA    #4
F999 26 F7 F992         BNE     PDATA2
F99B 39                 RTS                  STOP ON EOT
```

```
        PRINT CR LF
        NO REG CHANGED

F99C FF F3CE   A PCRLF  STX     XHI
F99F 86 0D     A        LDAA    #$D
F9A1 8D 13 F9B6         BSR     OUTCH
F9A3 FE F3CE   A        LDX     XHI
F9A6 39                 RTS

                    ON EXIT X=X+1
```

```
SAVE...COPIES BYTE
WAITS FOR HORIZ SYNC THEN STORES IN
DISPLAY RAM
NO REG KLOBBERED
ON ENTRY X REG POINTS TO LOC
IN DISPLAY MEMORY TO STORE CHAR
INTERRUPT IS MASKED BEFORE WRITE
TO VDG MEMORY THEN UN MASKED
(WE MUST WRITE DURING TV HORIZ SYNC)
```

```
FF7F 37          SAVE   PSHB
FF80 0F                 SEI

FF81 F6 F406  A SAVE0  LDAB   PIABD     WAIT FOR HS=1
FF84 2A FB FF81        BPL    SAVE0

FF86 F6 F406  A SAVE1  LDAB   PIABD     NEGATIVE EDGE
FF89 2B FB FF86        BMI    SAVE1

                    NOW CAN SAVE ( SCREEN BLANKED)

FF8B A7 00    A        STAA   0,X
FF8D 9E                CLI
FF8E 33                PULB
FF8F 39                RTS

                  A,B,X UNCHANGED
                  IGNORES LF
                  CALLS "RETURN" IF C.R.
                  CALLS "INIT" IF FORM FEED
                  CALLS "SCROLL" IF BOTTOM
                     OF DISPLAY RAM IS EXCEEDED
                  ELSE STORES CHAR

FF0F 37          OUTCH1 PSHB
FF10 36                 PSHA
FF11 FF F3CA  A         STX    SAVEX

                  CHECK IF END OF DISPLAY BUFF IF YES THEN SCROLL

FF14 FE F3CC  A MAIN1  LDX    NEXTBY
FF17 8C D200  A        CPX    #VDGRAM+512
FF1A 26 03 FF1F        BNE    MAIN2
FF1C BD FFC9  A        JSR    SCROLL

                  CHECK FOR C.R. IF YES THEN FINISH LINE WITH BLA
                  TRAP L.F.'S

FF1F 81 0A    A MAIN2  CMPA   #$0A
FF21 27 25 FF48        BEQ    MAIN6
FF23 81 0D    A        CMPA   #$D
FF25 26 06 FF2D        BNE    MAIN3
FF27 8D 67 FF90        BSR    RETURN
FF29 86 0A    A        LDAA   #$0A
FF2B 20 E7 FF14        BRA    MAIN1

                  CHECK FOR FORM FEED IF YES CLEAR SCREEN

FF2D 81 0C    A MAIN3  CMPA   #$0C
FF2F 26 04 FF35        BNE    MAIN4
FF31 8D 75 FFA8        BSR    INIT
FF33 20 13 FF48        BRA    MAIN6
```

```
                    CHECK FOR BACK SPACE IF YES  MOVE POINTR BACK
                    & STORE INVERTED BLANK

FF35 81 08    A MAIN4  CMPA   #$08
FF37 26 0C FF45        BNE    MAIN5
FF39 8C D000  A        CPX    #VDGRAM
FF3C 27 0A FF48        BEQ    MAIN6
FF3E 09                DEX
FF3F 86 60    A        LDAA   #$60
FF41 8D 3C FF7F        BSR    SAVE
FF43 20 03 FF48        BRA    MAIN6

                    GET HERE TO SAVE THE BEGGAR..THEN HOME
                    IF TEXT DISPLAY BLACK CHAR ON GREEN BACKGROUND
                    ELSE STORE AS IS

       FF45  A MAIN5  EQU
FF45 8D 38 FF7F        BSR    SAVE
FF47 08                INX
FF48 FF F3CC  A MAIN6  STX    NEXTBY
FF4B FE F3CA  A        LDX    SAVEX
FF4E 32                PULA
FF4F 33                PULB

                    NOW GO TO USERS DISPLAY ROUTINE
                    ....NOTE USER MUST EXECUTE A RTS
                    AS LAST INS
FF50 7E F393  A        JMP    USROUT



                         NOW CHECK KEYBOARD
                         ON EXIT INPUT CHAR IN A-REG

     FEE1 FF F3CA  A INCH1  STX    SAVEX
     FEE4 37                PSHB
     FEE5 8D 83 FE6A INCH0  BSR    BLINK     FLASH CURSOR
     FEE7 8D AC FE95        BSR    CURSOR
     FEE9 FE F3CA  A        LDX    SAVEX


     FEEC B6 F404  A        LDAA   PIAAD     LOOK FOR ANY KEY
     FEEF 2B 10 FF01        BMI    NONE
     FEF1 B1 F3D7  A        CMPA   FLAGK     SAME KEY
     FEF4 27 0E FF04        BEQ    SAME      FROM LAST TIME?

                         CHECK FOR ESCAPE FUNCTION

     FEF6 81 1B    A        CMPA   #$1B
     FEF8 27 12 FF0C        BEQ    CNTLEV

     FEFA B7 F3D7  A        STAA   FLAGK
     FEFD 8D 10 FF0F        BSR    OUTCH1    ECHO CHAR
     FEFF 33                PULB
     FF00 39                RTS
```

```
                        NO KEY CLOSED NOW GO TO USER ROUTINE
                        IF USER HAS KEY DOWN
                           RETURN WITH CHAR IN A REG
                                 WITH CARRY = 1
                        ELSE RETURN WITH CARRY =0


FF01 7F F3D7   A NONE    CLR     FLAGK
FF04 0C          SAVC    CLC
FF05 BD F390   A         JSR     USRINP
FF08 24 DB FEC5          BCC     INCHG
FF0A 33                  PULS
FF0B 39                  RTS
                    *
FF0C 7E FB20   A CNTLEV  JMP     CONTRL
```

TV BUG PARTS LIST

Integrated Circuits: LSI, MSI, SSI

QTY	PART #	FUNCTION	MANUFACTURER	DESIGNATION
1	MC6847P	VDG	MOTOROLA	U3
1	MC6808P	MPU		U1
1	MC6821P	PIA		U2
1	MC6846P3	TV BUG-RI/OT		U4
1	MC6850P	ACIA		U15
6	MCM2114	RAM		U9-U14
1	MC1372P	RF MODULATOR		U20
2	MC6887P	8T97		U5, U6
2	MC6889P	8T28		U7, U8
3	MC74LS138P	3 TO 8 DECODER		U28, U29, U30
3	MC74LS00P	QUAD NAND		U17, U18, U19
1	MC74LS04P	HEX INVERTER		U16
1	MC1455P	OSCILLATOR		U27

Integrated Circuits: CMOS

2	MC14013BP	DUAL D F-F	MOTOROLA	U25, U26
1	MC14040BP	12 BIT COUNTER		U24
1	MC14584BP	SCHMIDT HEX. INV.		U22
1	MC14070BP	EXOR		U21
1	MC14001BP	NOR		U23

Resistors (1/4W, ± 5% composition):

QTY	VALUE: (Ω)	DESIGNATION
3	240	R1, R2, R3
2	5.6K	R7, R14
1	75	R4
1	22K	R9
1	370	R15
1	750	R5
2	10K	R8, R13
1	2K	R19, R26
2	20K	R11, R12
6	3.3K	R18, R21, R22, R20, R23, R24
1	1K	R17
1	100K	R16
1	510	R25

Adjustable Resistors (1 Turn, Carbon Film, I10%)

1	10K	R6
1	20K	R10

Capacitors (in microfaroads, 25WVDC ± 10%, Ceramic):

QTY	VALUE	DESIGNATION
12	0.01	C1, C2, C9, C12, C13, C14, C15, C16, C17, C18, C19, C20
2	100 (electrolytic) 16V	C15, C21
1	50pF	C5
5	0.1	C6, C8, C11, C22, C23
1	.001	C7
1	.002	C14
1	2700pF	C10
1	1500pF	C13
1	56pF	C3

Misc. Components

QTY	COMPONENT	DESIGNATION
1	Variable Capacitor 9 to 35pF	C4
1	Crystal 3.57954 MHz	Y1
1	Transistor 2N2222	Q1
1	Keyboard-Cherry B70-05AB	
2	Switch Momentary B8600	S1, S2
1	Switch, DPDT, 60dB separation @ RF	S3
1	Transformer, 75 to 300	T1
2	Diodes, IN 914	D1, D2

MICROCHROMA 68 PARTS LIST

Capacitors

Quantity	Ref Des	Value
43	C1-C3, C7, C10, C13, C15, C20-23,C26-57	.1uF
2	C4, C58	100 uF @ 16V Electrolytic
1	C5	9-36pF Variable
1	C6	50pF
4	C8, C11, C14, C19	.01uF
1	C9	.002uF
1	C12	56pF
2	C16, C25	1000pF
1	C17	.02uF
1	C18	1500pF
1	C24	2700pF

Resistors (1/4W)

1	R1	5k Pot
1	R2	10k Pot
1	R3	510
1	R4	22k
6	R5, R10-12, R29, R30	3.3k
1	R6	15k
1	R7	33k
2	R8, R13	5.6k
1	R9	360
3	R14, R15, R18	240
1	R16	75
1	R17	750
2	R19, R34	10k
8	R20, R23-25, R28, R31-33	100k
1	R21	1k
2	R22, R27	2k
1	R26	20k

MICROCHROMA 68 PARTS LIST (CONTINUED)

Motorola Integrated Circuits

Quantity	Ref Des	P/N	Description
1	U1	MC14584B	CMOS Hex Schmitt Trigger
2	U2, U12	MC14013B	CMOS Dual D Flip-Flop
1	U3	MC6850*	NMOS Asynchronous Communications Interface Adapter (ACIA)
1	U4	MC6847*	NMOS Video Display Generator (VDG)
1	U5	MC1372*	Linear Color TV Video Modulator Circuit
1	U6	MC14070B	CMOS Quad Exclusive-OR Gate
1	U7	MC14040B	CMOS 12-Bit Binary Counter
1	U8	MC1455	Linear Timing Circuit
1	U9	MC6820/6821*	NMOS Parallel Interface Adapter (PIA)
3	U10, U13, U15	SN74LS00	TTL Quad 2-Input NAND Gate
2	U11, U14	SN74LS04	TTL Hex Inverter
1	U16	MC6846P3*	ROM, I/O, Timer (RIOT) w/TVBUG 1.2 Monitor
2	U17, U18	MC6887/8T97	Linear Hex Three-State Buffers
4	U19, U22-24	SN74LS138	TTL 3-to-8 Line Decoder
1	U20	SN74LS21	TTL Dual 4-Input AND Gate
1	U21	MC14001B	CMOS Quad 2-Input NOR Gate
1	U25	MC6808*	NMOS Microprocessor (MPU) with Clock
2	U26, U27	MC6889P/8T28	Linear Quad Bus Transceiver
2	U28, U29	SN74LS08	TTL Quad 2-Input AND Gate
30	U30, U59	MCM2114-45	NMOS 1K x 4 Static RAM

*Included in MicroChroma 68 Kits (Motorola P/N SCPROM02)

MICROCHROMA 68 KITS PARTS LIST (CONTINUED)

Miscellaneous Components

Quantity	Ref Des	Description
1	S1	Momentary SPDT (Break)
1	S2	Momentary SPST (Reset)
1	S3	DPDT Switch (60dB @ RF)
1	Y1	3.579545 MHz Crystal
1	Q1	2N2222A Transistor
1		Cherry "Pro" Keyboard (Cherry P/N B70-05AB) or equivalent with interface cable terminated with 24-pin header compatible with J1
1	L1	.1uH adjustable Inductor
1	T1	75 to 300 Matching Transformer
1	P2	Phono plug with compatible RF interconnect cable
1	P3, P4	Phono plugs with compatible audio interconnect cable
1		Vestigal Sideband Filter tuned to pass desired channel frequency
1	PCB*	MicroChroma 68 Printed Circuit Board (Motorola P/N SCPROM02PCB)
2	D1, D2	IN914 Serial Diode

* Included in MICROCHROMA 68 Kits (Motorola P/N SCPROM02PCB)

MicroChroma 68 Assembly Parts List

Capacitors

Ref Des	Value	Ref Des	Value
___ C1	.1uF	___ C30	.1uF
___ C2	.1uF	___ C31	.1uF
___ C3	.1uF	___ C32	.1uF
___ C4	100uF @ 16V Electrolytic	___ C33	.1uF
___ C5	9-35pF Variable	___ C34	.1uF
___ C6	50pF	___ C35	.1uF
___ C7	.1uF	___ C36	.1uF
___ C8	.01uF	___ C37	.1uF
___ C9	.002uF	___ C38	.1uF
___ C10	.1uF	___ C39	.1uF
___ C11	.01uF	___ C40	.1uF
___ C12	56pF	___ C41	.1uF
___ C13	.1uF	___ C42	.1uF
___ C14	.01uF	___ C43	.1uF
___ C15	.1uF	___ C44	.1uF
___ C16	1000pF	___ C45	.1uF
___ C17	.02uF	___ C46	.1uF
___ C18	1500pF	___ C47	.1uF
___ C19	.01uF	___ C48	.1uF
___ C20	.1uF	___ C49	.1uF
___ C21	.1uF	___ C50	.1uF
___ C22	.1uF	___ C51	.1uF
___ C23	.1uF	___ C52	.1uF
___ C24	2700pF	___ C53	.1uF
___ C25	1000pF	___ C54	.1uF
___ C26	.1uF	___ C55	.1uF
___ C27	.1uF	___ C56	.1uF
___ C28	.1uF	___ C57	.1uF
___ C29	.1uF	___ C58	100uF @ 16V Electrolytic

MicroChroma 68 Assembly Parts List (Continued)

Resistors (1/4W)

Ref Des	Value (Ohms)	Ref Des	Value (Ohms)
___ R1	5k Potentiometer*	___ R18	240
___ R2	10k Potentiometer*	___ R19	10k
___ R3	510	___ R20	100k
___ R4	22k	___ R21	1k
___ R5	3.3k	___ R22	2k
___ R6	15k	___ R23	100k
___ R7	33k	___ R24	100k
___ R8	5.6k	___ R25	100k
___ R9	360	___ R26	20k
___ R10	3.3k	___ R27	2k
___ R11	3.3k	___ R28	100k
___ R12	3.3k	___ R29	3.3k
___ R13	5.6k	___ R30	3.3k
___ R14	240	___ R31	100k
___ R15	240	___ R32	100k
___ R16	75	___ R33	100k
___ R17	750	___ R34	10k

*1 or 10 turn linear taper

Motorola Integrated Circuits

Ref Des	P/N	Description
___ U1	MC14584B	CMOS Hex Schmitt Trigger
___ U2	MC14013B	CMOS Dual D Flip-Flop
___ U3	MC6850**	NMOS Asynchronous Communications Interface Adaptor (ACIA)
___ U4	MC6847**	NMOS Video Display Generator (VDG)
___ U5	MC1372**	Linear Color TV Video Modulator Circuit
___ U6	MC14070B	CMOS Quad Exclusive-OR Gate
___ U7	MC14040B	CMOS 12-Bit Binary Counter
___ U8	MC1455	Linear Timing Circuit
___ U9	MC6820/MC6821**	NMOS Parallel Interface Adapter (PIA)

**Included in MicroChroma 68 Kit Motorola P/N SCPROM02

MicroChroma 68 Assembly Parts List (Continued)

Ref Des	P/N	Description
___ U10	SN74LS00	TTL Quad 2-Input NAND Gate
___ U11	SN74LS04	TTL Hex Inverter
___ U12	MC14013B	CMOS Dual D Flip-Flop
___ U13	SN74LS00	TTL Quad 2-Input NAND Gate
___ U14	SN74LS04	TTL Hex Inverter
___ U15	SN74LS00	TTL Quad 2-Input NAND Gate
___ U16	MC6846P3**	NMOS ROM, I/O, Timer (RIOT) with TVBUG 1.2 Monitor Program
___ U17	MC6887/MC8T97	Linear Hex Three-State Buffers
___ U18	MC6887/MC8T97	Linear Hex Three-State Buffers
___ U19	SN74LS138	TTL 3-to-8 Line Decoder
___ U20	SN74LS21	TTL Dual 4-Input AND Gate
___ U21	MC14001B	CMOS Quad 2-Input NOR Gate
___ U22	SN74LS138	TTL 3-to-8 Line Decoder
___ U23	SN74LS138	TTL 3-to-8 Line Decoder
___ U24	SN74LS138	TTL 3-to-8 Line Decoder
___ U25	MC6808**	NMOS Microprocessor (MPU) with Clock
___ U26	MC6889/MC8T28	Linear Quad Bus Transceiver
___ U27	MC6809/MC8T28	Linear Quad Bus Transceiver
___ U28	SN74LS08	TTL Quad 2-Input AND Gate
___ U29	SN74LS08	TTL Quad 2-Input AND Gate

MCM 2114-45 NMOS 1K X 4 Static RAMS

___ U30	___ U40	___ U50***
___ U31	___ U41	___ U51
___ U32	___ U42***	___ U52
___ U33***	___ U43	___ U53
___ U34	___ U44***	___ U54
___ U35***	___ U45	___ U55
___ U36	___ U46	___ U56
___ U37	___ U47	___ U57
___ U38	___ U48***	___ U58
___ U39	___ U49	___ U59

*** These memories are required for the minimal system.

MicroChroma 68 Assembly Parts List (Continued)

Miscellaneous Components

Ref Des	Description
___ S1	Momentary SPDT (Break)
___ S2	Momentary SPST (Reset)
___ S3	DPDT Switch (60dB @ RF)
___ Y1	3.579545 MHz Crystal
___ Q1	2N2222A Transistor
___	Cherry "PRO" Keyboard (Cherry P/N B70-05AB) or equivalent with 24 pin header compatible with J1
___ L1	.1uH adjustable Inductor
___ T1	75 to 300 Matching Transformer
___ P2	Phone plug with compatible RF interconnect cable
___ P3	Phono plug with compatible audio interconnect cable
___ P4	Phono plug with compatible audio interconnect cable
___	Vestigal Sideband Filter tuned to pass desired channel frequency.
___ PCB **	MicroChroma 68 Printed Circuit Board (Motorola P/N SCPROMO2PCB)
___ D1	IN914 Signal Diode
___ D2	IN914 Signal Diode

** Included in MicroChroma 68 Kit (Motorola P/N SCPROMO2)

DEAR MICROCHROMA 68 PURCHASER:

A USERS MANUAL CONTAINING THE SOURCE LISTING FOR TVBUG, AND USEFUL APPLICATIONS HARDWARE AND SOFTWARE WILL BE AVAILABLE SHORTLY. TO ENSURE THAT YOU RECEIVE YOUR COPY AS SOON AS IT IS AVAILABLE, SEND IN THE BOTTOM HALF OF THIS SHEET. IF YOU EXPERIENCE ANY PROBLEM OR WISH TO SHARE OTHER USEFUL INFORMATION, CALL THE "MICROCHROMA HOTLINE" (512) 928-6878.

TO: Motorola Microcomponent Applications
MS F2605 Attn: MicroChroma 68
3501 Ed Bluestein Blvd.
Austin, Texas 78721

SIRS: PLEASE SEND THE TVBUG APPLICATIONS MANUAL TO;

NAME_____

ADDRESS_____

CIRY_____ STATE_____ ZIP CODE_____

MOTOROLA INC. *Integrated Circuits Division*

3501 ED BLUSTEIN BLVD., AUSTIN, TEXAS 78721