WEST JAX 99ER NEWS

DEDICATED TO THE TI-99/4A

NOVEMBER 1986 #20

The West Jax 99'ers is a non-profit computer users group for the TI99 Home Computer. (Not affiliated in any way with Texas Instruments.) The club's mailing address is PO BOX 176, Orange Park, FL 32067

MEETINGS - Second Tues. of every month at the Webb Library, two lights west of Blanding on 103rd Street. Workshop time on the club computer begins at 6 PM, business meeting at 7 PM, more workshop time afterwards. Visitors are welcome to attend and find out more about the club activities and library. A second meeting of strictly workshop time is held on the Fourth Tuesday of the month beginning at 6PM.

OFFICERS -	President	Rick Felzien	(904) 772-9162
	Vice-Pres.	Steve Peacock	(904) 737-2859
	Treasurer	Thomas LeMay	(904) 282-5220
	Secretary	Ralph Glatli	(904) 757-3630
WEST JAX TIBBS BBS		Art Pugh(SysOp)	(904) 272-8067

For newsletter submissions and suggestions, contact Rick Felzien.

****
****

The Logo logician was remis this month and didn't get his article finished in time for this issue. He recently got a corcomp expansion system and has been converting all files over to DS/DD, Which all of those of you who have had to convert all your files know, is a not too small task and takes a few evenings to accomplish.

Richard Corder has written a nice article on his Super Cart which I call (Super Duper Cart). This sounds like a pretty nice setup. The Basic Assembler has furnished one of his fine articles also.

****
****

SUPER - SUPERCART
BY
RICHARD CORDER

Just when you thought you had all of the supercart articles you could use I have decided to give you one more. The part of this modification that makes this one unique is that I have used parts from several articles and combined them into one module. The parts are basicly the same as other carts except I have made modifcations to use sockets for all chips and an optional built in reset switch. I have included enough information to add a second ram chip and a socket to put in your Disk-manager chips or Ti-writer.

The start of all supercarts should begin with the reading of the article by JOHN CLULOW in micropendium. This article has been reprinted in many newsletters including our WEST JAX 99'ER NEWS JULY 1986. John explains all the basics used in construction of the module.

Lets start with preparing the board to accept all the componants and the sockets. Break the foil between the the two adjacent holes at F1 on the top of the board and the two holes at F3 on the bottom of the board. At the other end of the board where the reset resistor was removed locate the hole that would be number 16 if a grom chip were to be installed in the last set of holes. Isolate this hole by cutting the foil around it. Call this hole F2 and use it for the connection of D1,C2 and R3.

Remove the section of ground strip at the top of the boards where the old rom chip was removed. Also at this end of the board remove the circuit trace that connected to rom pin 7. This hole will become ram pin 9.

You will find there are many places on the board to make connections. If you trace a connection to another location on the board feel free to use it only double check it before you test the module.

If you wish to add a socket for a second grom chip locate grom pin 14 on the next set of grom chip holes. Isolate this solder pad by cutting the foil as in the top view of the pc board. Separate edge connector 30 by cutting the foil at the top of the connector. Remove enough of the foil to ensure there is no possibility of shorting the wire soldered to this edge connector later. This is the last foil cut on the top of the board.

On the reverce side locate the ground strip section at the top of the rom chip holes. Cut this foil and the section of foil at the top of edge connector pin 27 as shown in the bottom view of pc board. Double check the two holes next to rom pin 24 to ensure that only the one closest is connected.

Set your 28 pin socket in place and mark the location of the extra four pins at the top of the board. Drill or use some other means of putting these holes in the board. Do not worry if you make the holes a little oversized we will be connecting wire to the socket later.

We are now ready to install our components. Install the sockets in the board with the number 1 pin at the top of the board. Replace C1 in the holes outboard of the two it was removed from. Bend the leg on the flat side of the led 90 degrees and solder it into the hole next to F3 and the other leg to C1. On the bottom side of the board connect R1 (1K) to the leg of the led put through the hole. Connect the other end of R2 to any ground pad. If you are useing two ram chips connect two 1K resisters to F4.(this pad allso connects to F3) One will connect to pin 20 of the socket and the other to pin 20 on the upper ram chip.

Solder the banded end of D1 into the hole marked F2. You can make the other connections at F2 to this diode. The other end of D1 solders into the hole at the bottom right grom location. This would be grom pin 9 as shown in figure 1. Connect the positive lead of C2 to F2 and the other lead to ground. Solder the banded end of D2 to R3(1K) then connect the other end of R3 to F2. Connect the free end of D2 to the positive side of the battery holder or if you are not using a holder connect a short length of wire to the diode for use later.

On the bottom of the board connect a length of wire from F2 to socket pin 28. Connect a jumper to replace the section of ground foil we removed at the top of the board. Solder a wire from ram pin 9 to grom pin 11 or card edge connector 23. Connect a wire from ram pin 27 to card edge connector 32 on the top of the board. Solder a wire from ram pin 2 to card edge connector 24.

It is now time to make connections to our switches. Cut six lengths of wire approximately four inches long and one two inches in length. Starting with the double pole double throw switch with center off we connect one wire from the center leg of the switch to edge connector 29. The other center leg will connect to edge connector 34. Solder a wire from grom pin 14 on both sockets. Each of these wires will connect to the side of the switch that has been connected to edge connector 29. Connect the two inch long wire to the other side of the switch makeing sure you connect it to the same end of the switch as grom chip enable wire is connected for the socket you wish to use for your E/A chip.

Solder a wire to each end of your single pole double throw switch then connect one of these to ram pin 20 on each ram chip.

The two inch long wire connected to the center leg of the DPDT switch connects to the center leg of the SPDT switch.

If you wish to add the reset switch to the module I would suggest the components on the right end of the board be installed on the bottom of the pc board. Connect one lead to edge connector 29 and one side of the switch. Connect the other side of the switch to the reset resistor (100 ohm) removed from the pc board. The other end of resistor is soldered to a short length of wire and it is then soldered to the solder pad connected to edge connector pin 1.

Insulate the top of the board with electrical tape in the area that the switches will be mounted. As an added precaution wrap the switches with tape allso. Depending on the size of your switches a good place to install them is on the top of the module near the center of the hump. If you find micro switches small enough they can be mounted through the front of the module. This location would be the ideal place if you can fit them in.

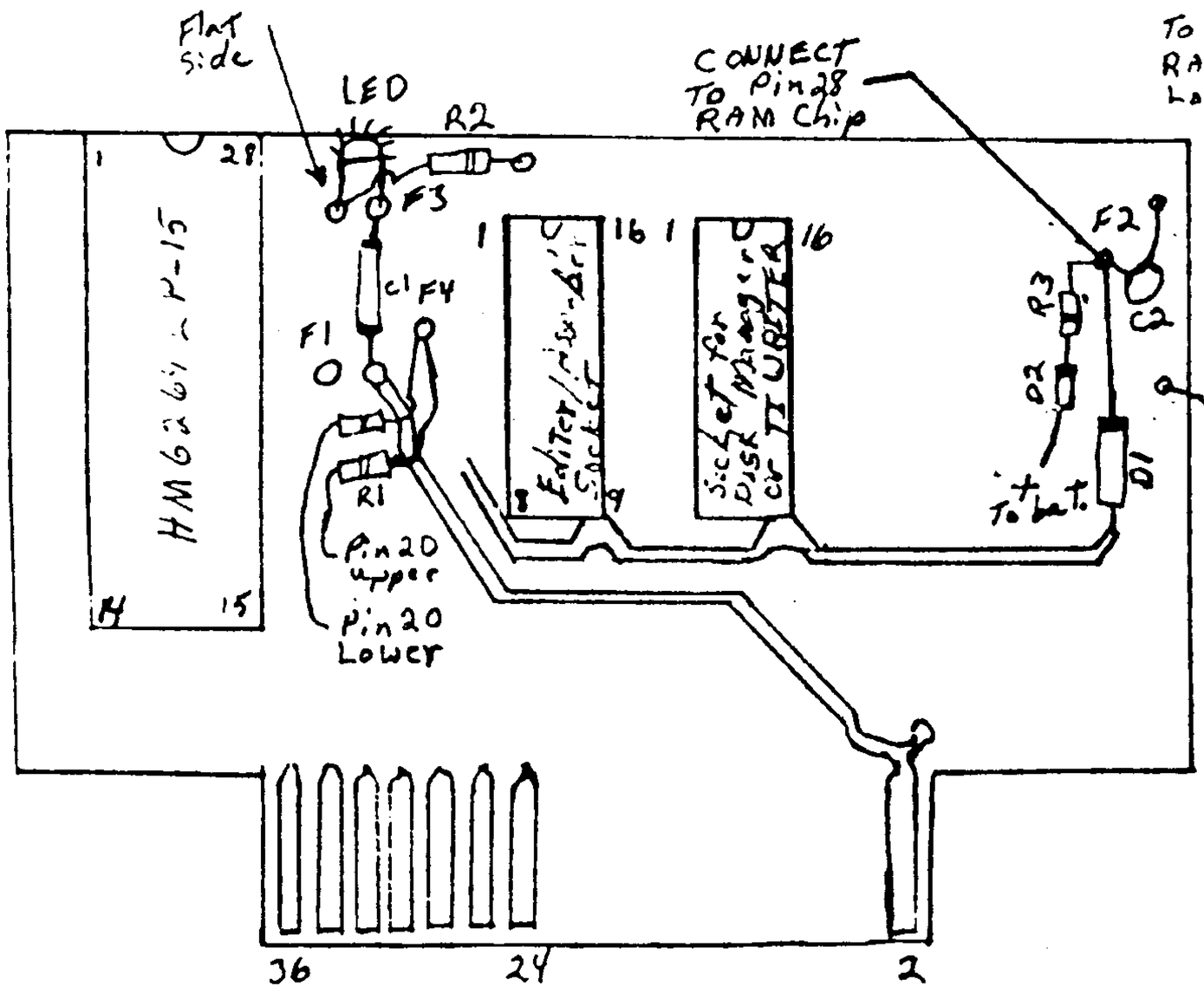
The two ram chips are soldered together one on top the other. All pins are connected except pin 20 on the upper chip it is bent out away from the lower chip.

It is now time to test your module and after double checking your board for solder bridges and proper placement or orientation of componants. DO NOT install the battery until you have tested the module and are satisfied all is correct. The easiest method of testing is to use CALL LOAD as in JOHN CLULOW'S article. Cut the case for clearance around the ram chip stack and for the led. I won't go into all the testing you could do to the module for this is very well covered by John and the other articles I combined to construct this module. I would like to give credit to Richard J. Bailey of the NH99ER USER GROUP , Jim McCulloch of Evanston, IL , and an article written by Dave Ratcliffe in which he is expanding on an article by Jim Ellis of Raleigh, North Carolina.

WARNING!!! As with any project of of this kind I nor the WEST JAX 99'ERS are responsible if you blow any chips or damage your computer in any way. PROCEED AT YOUR OWN RISK

*NOTE: I used a wirewrap socket for the second chip and raised it to the outside of the module. In this way I could change the grom chip without opening the module.

HM6264LP-15 CMOS RAM	(2)
TI GAME MODULE	(1)
1N914 SIGNAL DIODE	(2)
1K RESISTER	(4)
2.2 µf TANTALUM CAP.	(1)
RED LED	(1)
LITHIUM CELL	(1)
CELL HOLDER	(1)
16 PIN SOCKET	(2)
28 PIN SOCKET	(1)
DPDT SWITCH CENTER OFF	(1)
SPDT SWITCH	(1)
NC MOMENTARY CONT. SW	(1)



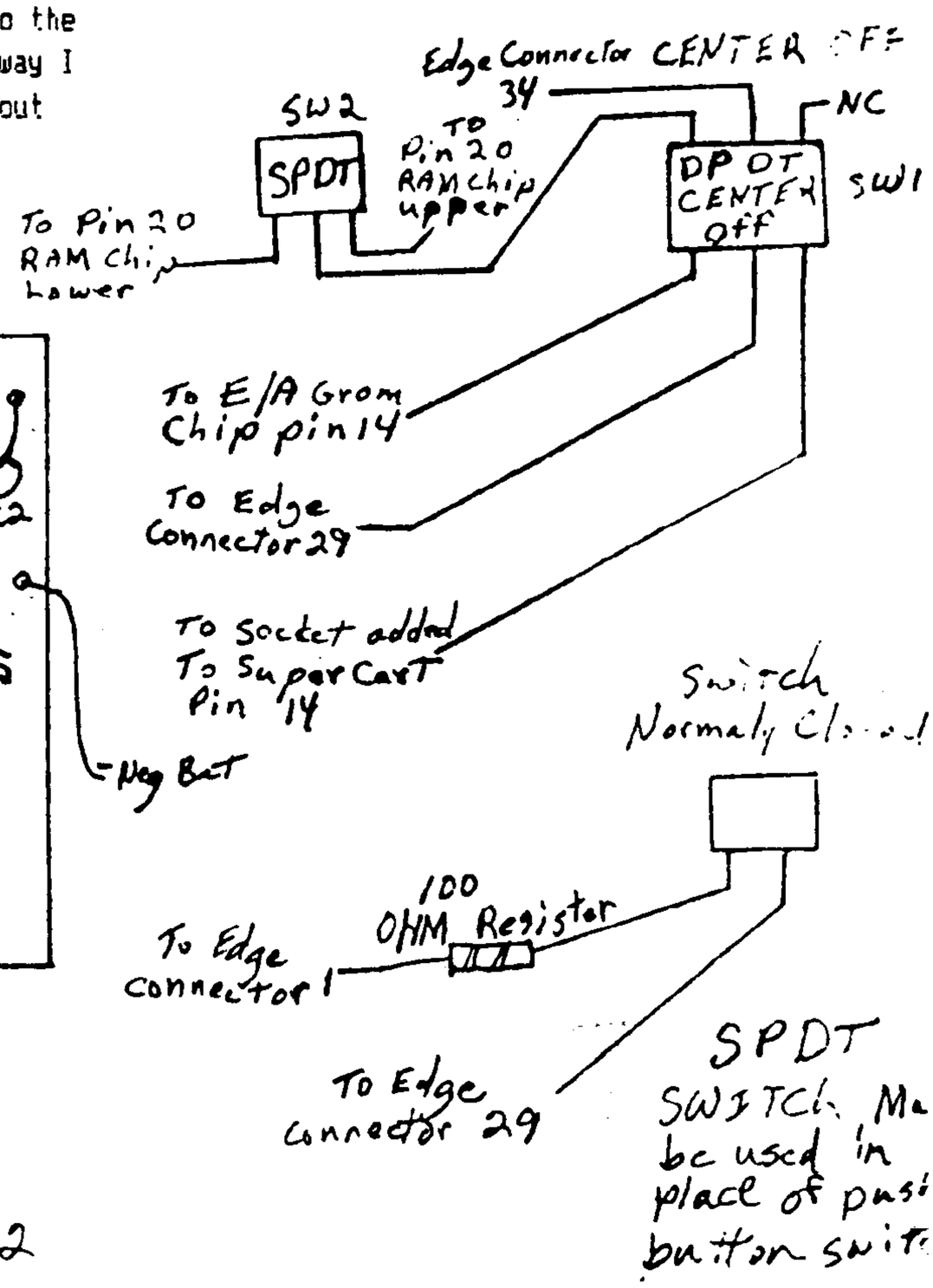
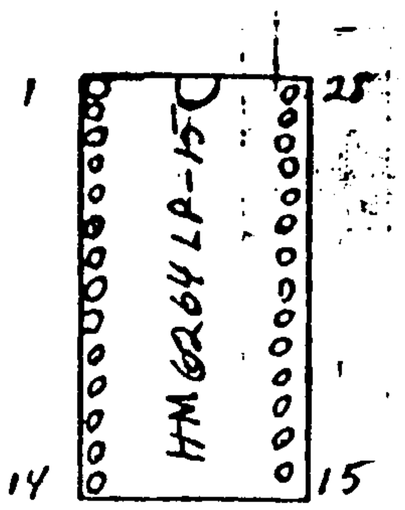
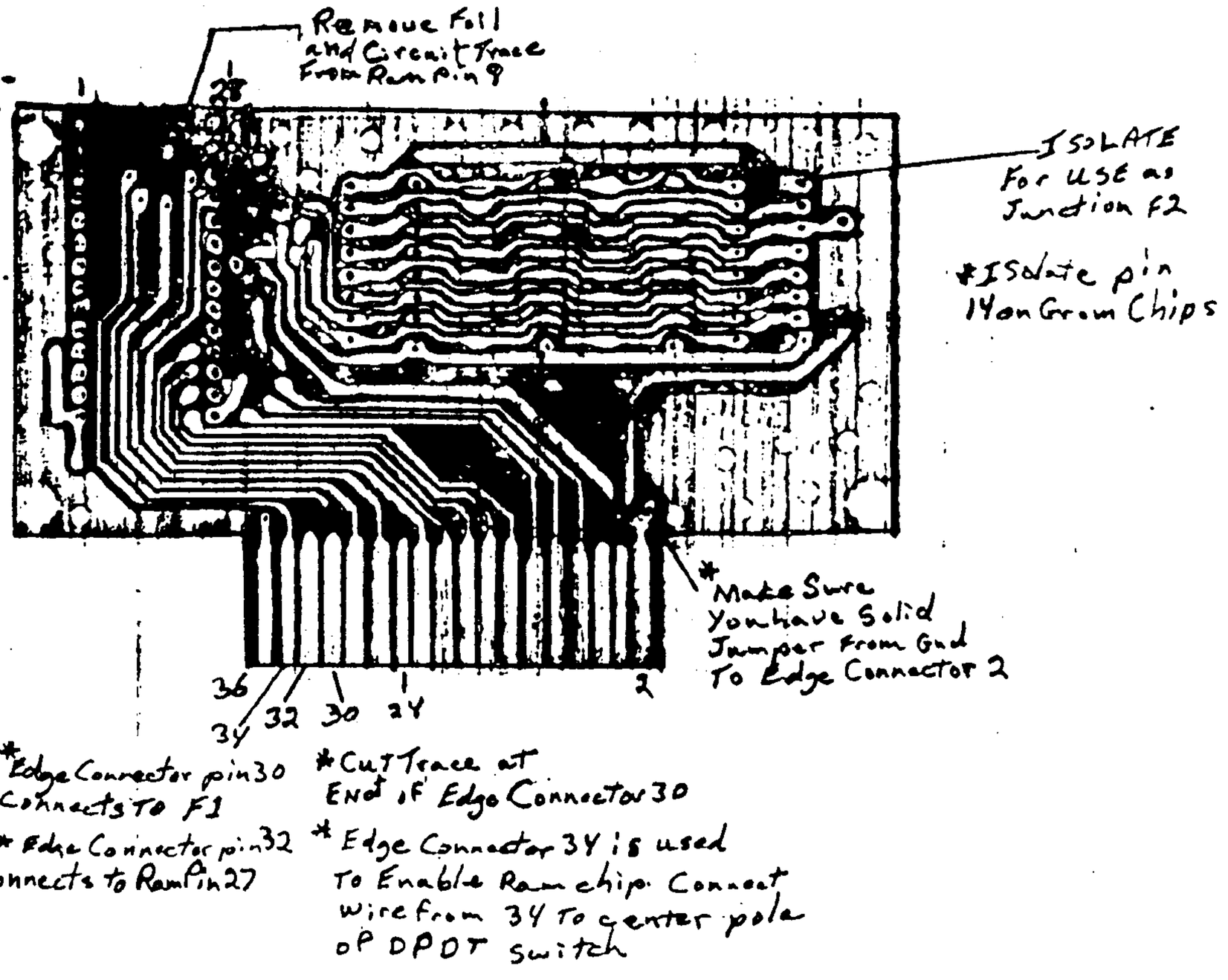


Figure 1





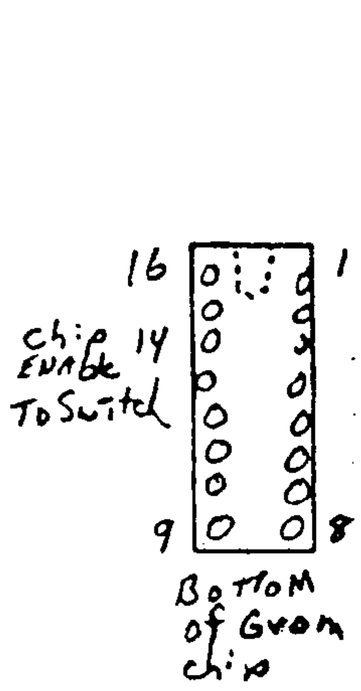
*Edge Connector pin 30 Connects TO F1

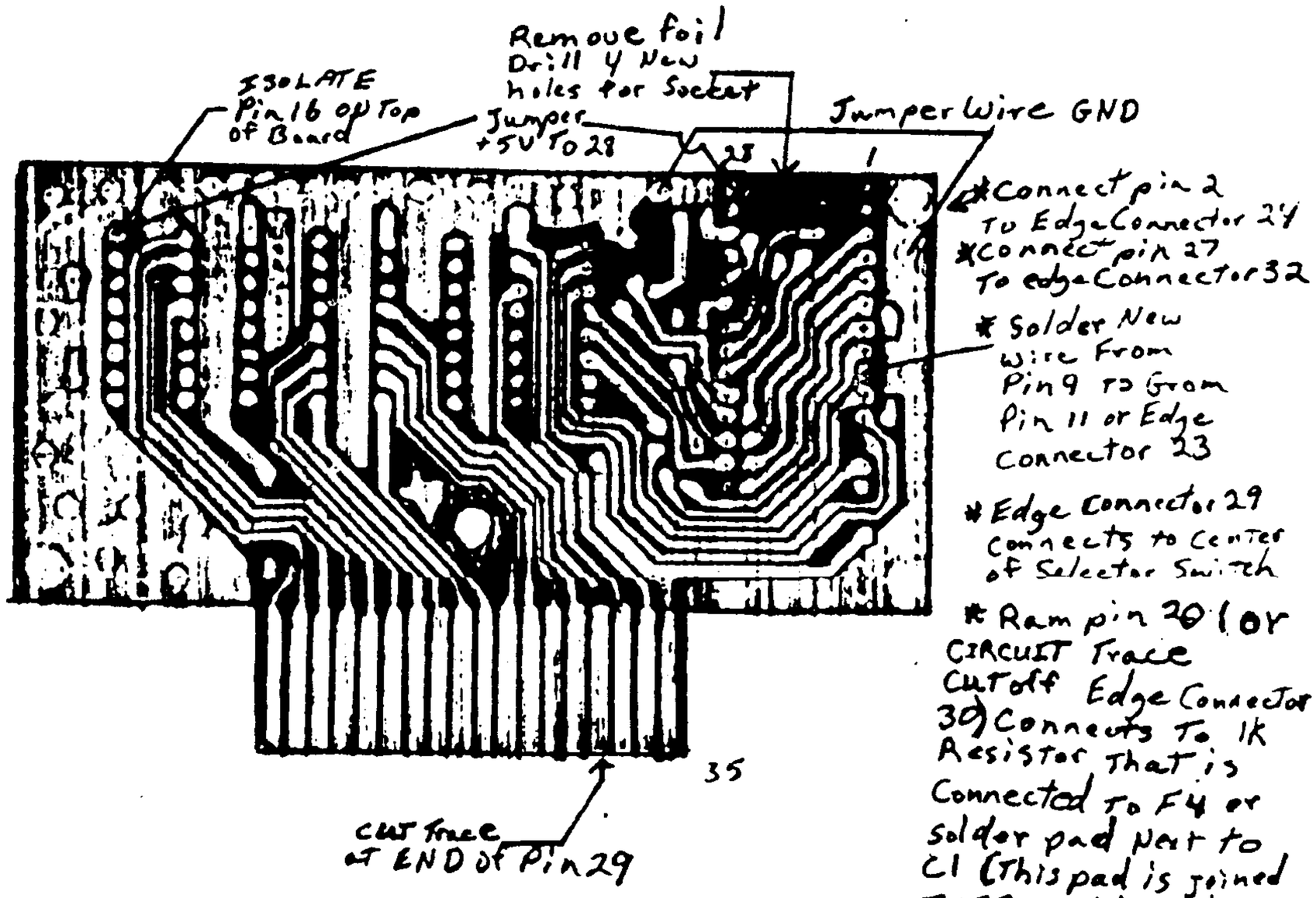
* Edge Connector pin 32 Connects to Ram Pin 27

*Cut Trace at END of Edge Connector 30

* Edge Connector 34 is used To Enable Ram chip. Connect wire from 34 To center pole of DPDT switch

Top View





*Connect pin 2 To Edge Connector 24

*Connect pin 27 To edge Connector 32

* Solder New wire From Pin 9 To Gram Pin 11 or Edge Connector 23

* Edge Connector 29 connects to Center of Selector Switch

* Ram pin 20 (or CIRCUIT Trace CUT off Edge Connector 30) Connects To 1K Resistor That is Connected To F4 or solder pad Next to C1 (This pad is joined To F3 on this side of The board)

Bottom View

T H E B A S I C A S S E M B L E R #1 By Steve Peacock

MULTICOLOR MODE

When I started to write this column for our newsletter , I stated that I would not present a game or utility. Only short ideas would be done, that you could use to create your own longer program. This month I am presenting a stand alone program. It did not start out that way, but I just kept adding refinements to it and a complete program, demonstrating multicolor mode, is the result.

In the multicolor mode the screen is made up of 48 rows of 64 columns. Each of these blocks are 4 pixels by 4 pixels. One of the best things is that each of the blocks can be any of the 16 colors that the TI computer can make. Also if you want to use sprites, you can. The color of every two boxes are described by one byte.

The first thing you must do, when using multicolor mode, is number the rows of the screen. In the first four rows the columns are numbered 0 to 31. The next four rows are numbered 32 to 63. This continues for a total of six groups.

ROWS	SCREEN POSITIONS					
1-4	0	1	2.....	29	30	31
5-8	32	33	34.....	61	62	63
9-12	64	65	66.....	93	94	95
13-16	96	97	98.....	125	126	127
17-20	128	129	130.....	157	158	159
21-24	160	161	162.....	189	190	191

In the multicolor mode, the Pattern Descriptor Table stores the color of the block, not the character pattern. This table starts at >0800 and ends at >0E00. If you wrote the color transparent to each of these addresses, the screen would clear.

To put the TI computer in the multicolor mode, write >E8 to VDP Write only register number one. This is done with the command -LI R0,>01E8-.

Now that the TI computer is in the multicolor mode and you have set up a screen numbering system, you must find the correct byte on the screen. If your starting row value was 12 and starting column value was 27, here is how you would calculate the correct byte:

```
Y=COLUMN (27), X=ROW (12)

A=Y/2              (A=13.5)
B=INT(Y/2)         (B=13)
C=A-B              (C=.5)
D=X/8              (D=1.5)
E=INT(X/8)         (E=1)
F=D-E              (.5)
G=>0800+E6+B*8+F    (G=2408.5)
```

This is the position in the Pattern Descriptor Table. Next you must determine if you need to change the left or right digit. If the value in F is zero, then change the left digit if not change the right digit.

I have introduced two new commands this month. The command ANDI stands for AND Immediate. When this command is used to compare two words the bits that are set in both are set in the new word. For example if register 9 has the value of

>E318 (1110001100011000d) and you ANDI it with >2619 (0010011000011001d) the result will be:

```
   ANDI R9,>2619
    >E318  1110001100011000
    >2619  0010011000011001
```

Register 9 will now contain:

```
    >2218  0010001000011000
```

The other command A stands for Add. The command A R8,R9 will add the value in register 8 and the value in register 9 and put the answer in register the second register (9). If register 8 holds >E401 and register 9 holds >108C the command A R8,R9 results in >E401 plus >108C. The answer is put in register 9, so register 9 now holds >F48D. The value in register 8 is left unchanged.

Please note that the A command adds words (registers or a value at an address). The command AI will add a number to the register.

```
A--> A  R4,@>E09A
     A  R4,R5
AI-> AI R4,31
     AI R4,>20
```

This months Basic Assembler starts out by turning the screen black and going into text mode. This is done to set up the screen. Next the screen positions are numbered. Then the Pattern Descriptor Table is cleared to transparent. This loop is also used when you want to clear the screen from the program.

The main loop scans the keyboard for the following keys: W, E, R, S, D, F, Z, X, C, 1, 2, and 3. The letter keys are used to draw on the screen. The number keys are used like this: 1 to change screen color, 2 to change block color, and 3 to clear the screen.

The loop DRAW is the place that computes the location in the Pattern Descriptor Table. If you follow it statement by statement you will find it is the same as I have described above.

One last thing. I have put in a label 'RELAY'. This is used because the program must jump more than >100 bytes (256d). All the 'RELAY' is used for is a double jump.

```
100 REM  PROGRAM BA16B==>Basic Assembler #16 Basic Version
110 REM  MULTICOLOR MODE
120 REM  (C)1986 S. PEACOCK
130 PRINT "NO BASIC COUNTERPART FOR    MULTICOLOR MODE."
140 PRINT
150 PRINT "USING EXTENDED BASIC YOU CANSEE WHAT MULTICOLOR MODE IS LIKE. IN COMM
AND MODE TYPE:"
160 PRINT "CALL LOAD(-31788,232). THEN PRESS ANY KEY. TO EXIT TYPE CALL LOAD(-31
788,224) (YOU"
170 PRINT "WILL NOT SEE ANYTHING BUT   COLORED BOXES). THEN PRESS  ENTER."
180 END
```

```
*****************************************************************************
*
*PROGRAM BA16A==>Basic Assembler #16 Assembly Version
*MULTICOLOR MODE
*(C)1986 S. PEACOCK
*
*****************************************************************************
        REF   VSBW,VWTR,KSCAN,VSBR
        DEF   START
START   CLR   @>8374          *SETUP KEY SCAN
        LI    R0,>0711        *MAKE SCREEN BLACK ON BLACK
        BLWP  @VWTR           *BY WRITING >11 TO WRITE ONLY REG 7
        LI    R0,>01F0        *PUT IN 40 COLUMN MODE BY
        BLWP  @VWTR           *WRITING >F0 TO WRITE ONLY REG 1
*************************START OF SCREEN INITIALIZATION
        CLR   R0              *SCREEN POSITION IS THE START
        LI    R7,6            *THERE WILL BE 6 GROUPS OF 128 BYTES
        CLR   R5              *REG 5 WILL HOLD VALUE TO BE WRITTEN
LP1     LI    R3,4            *THERE ARE 4 LINES IN EACH GROUP
LP2     LI    R4,32           *THERE ARE 32 CHARACTERS ON EACH LINE
        MOVB  R5,R1           *THERE IS MORE TO BE WRITTEN
LP3     BLWP  @VSBW           *BRANCH TO VSBW
        INC   R0              *SCREEN POSITION IS INCREASED BY ONE
        AI    R1,>0100        *ADD ONE TO THE VALUE TO BE WRITTEN
        DEC   R4              *DECREASE THE BYTES TO BE WRITTEN TO THE LINE
        JNE   LP3             *IF NOT AT THE END OF THE LINE STAY IN LOOP LP3
        DEC   R3              *DECREASE THE NUMBER OF LINES IN THE GROUP
        JNE   LP2             *IF ANY LINES ARE LEFT, STAY IN LOOP LP2
        AI    R5,>2000        *START THE NEXT GROUP 32d LARGER
        DEC   R7              *DECREASE THE NUMBER OF GROUPS
        JNE   LP1             *IF GROUPS LEFT STAY IN LOOP LP1
************************CLEAR ALL BOXES TO TRANSPARENT
CL      LI    R0,>0800        *START OF PATERN DESCRIPTOR TABLE
        CLR   R1              *>00 IS TRANSPARENT
LP4     BLWP  @VSBW           *BRANCH TO VSBW
        INC   R0              *INCREASE POSITION TO BE WRITTEN
        CI    R0,>0E00        *>0E00 IS THE END OF THE PATTERN DESCRIPTOR TABLE
        JNE   LP4             *IF NOT REACHED STAY IN LOOP LP4
        LI    R0,>01E8        *>E8 TO WRITE ONLY REGISTER 1 SETS MULTICOLOR MODE
        BLWP  @VWTR           *BRANCH TO VWTR
        SWPB  R0              *SWAP BYTES IN REG 0 =>E801
        MOVB  R0,@>83D4       *MOVE LEFT BYTE OF REG 0 (E8) TO >83D4
        LI    R3,32           *COLUMN STARTING POSITION
        LI    R4,24           *ROW STARTING POSITION
        LI    R5,>0001        *STARTING SCREEN COLOR (BLACK)
        LI    R14,>9000       *STARTING BLOCK COLOR (LIGHT RED)
*************************
*       MAIN LOOP       *
*************************
LL      LIMI  2               *FCTN QUIT
        LIMI  0               *WILL WORK
        LI    R13,>2000       *DELAY LOOP TIME BETWEEN PRINTING A BLOCK TO SCREEN
        DEC   R13             *INCREASE OR DECREASE AS YOU LIKE
        JNE   $-2             *
        BLWP  @KSCAN          *KEYBOARD SCAN
        CLR   R1              *CLEAR REG 1, WILL HOLD THE ASCII OF THE KEY PRESSED
        MOV   @>8375,R1       *PUT THE ASCII OF KEY PRESSED INTO RIGHT BYTE OF REG
        CI    R1,83           *COMPARE IT TO 83 (S KEY)
```

```
        JEQ   LEFT            *IF EQUAL JUMP TO LEFT
        CI    R1,68           *COMPARE IT TO 68 (D KEY)
        JEQ   RIGHT           *IF EQUAL JUMP TO RIGHT
        CI    R1,69           *COMPARE IT TO 69 (E KEY)
        JEQ   UP              *IF EQUAL JUMP TO UP
        CI    R1,88           *COMPARE IT TO 88 (X KEY)
        JEQ   DOWN            *IF EQUAL JUMP TO DOWN
        CI    R1,49           *COMPARE IT TO 49 (1 KEY)
        JEQ   SC              *IF EQUAL JUMP TO SC (SCREEN COLOR)
        CI    R1,50           *COMPARE IT TO 50 (2 KEY)
        JEQ   BC              *IF EQUAL JUMP TO BC (BLOCK COLOR)
        CI    R1,51           *COMPARE IT TO 51 (3 KEY)
        JEQ   CL              *IF EQUAL JUMP TO CL (CLEAR SCREEN)
        CI    R1,82           *COMPARE IT TO 82 (R KEY)
        JEQ   UPRIT           *IF EQUAL JUMP TO UPRIT
        CI    R1,87           *COMPARE IT TO 87 (W KEY)
        JEQ   UPLEF           *IF EQUAL JUMP TO UPLEF
        CI    R1,90           *COMPARE IT TO 90 (Z KEY)
        JEQ   DNLEF           *IF EQUAL JUMP TO DNLEF
        CI    R1,67           *COMPARE IT TO 67 (C KEY)
        JEQ   DNRIT           *IF EQUAL JUMP TO DNRIT
        JMP   LL              *IF ANY OTHER KEY PRESSED, STAY IN MAIN LOOP
**************************
LEFT    DEC   R3              *MOVE PRINT POSITION 1 TO LEFT
        CI    R3,-1           *CHECK TO SEE IF OUT OF BOUNDS
        JNE   DRAW            *IF NOT, JUMP TO DRAW
        CLR   R3              *IF OUT OF BOUNDS, RESET
        JMP   DRAW            *
**************************
RIGHT   INC   R3              *MOVE PRINT POSITION 1 TO RIGHT
        CI    R3,64           *CHECK TO SEE IF OUT OF BOUNDS
        JLT   DRAW            *IF NOT, JUMP TO DRAW
        LI    R3,63           *IF OUT OF BOUNDS, RESET
        JMP   DRAW            *
**************************
UP      DEC   R4              *MOVE PRINT POSITION 1 UP
        CI    R4,-1           *CHECK TO SEE IF OUT OF BOUNDS
        JNE   DRAW            *IF NOT, JUMP TO DRAW
        CLR   R4              *IF OUT OF BOUNDS, RESET
        JMP   DRAW            *
**************************
DOWN    INC   R4              *MOVE PRINT POSITION 1 DOWN
        CI    R4,48           *CHECK TO SEE IF OUT OF BOUNDS
        JLT   DRAW            *IF NOT, JUMP TO DRAW
        LI    R4,47           *IF OUT OF BOUNDS, RESET
        JMP   DRAW            *
**************************
UPRIT   DEC   R4              *MOVE PRINT POSITION UP 1
        CI    R4,-1           *CHECK UP FOR OUT OF BOUNDS
        JNE   N1              *IF NOT, CHECK RIGHT
        CLR   R4              *IF IS, RESET UP
N1      INC   R3              *MOVE PRINT POSITION 1 TO RIGHT
        CI    R3,64           *CHECK RIGHT FOR OUT OF BOUNDS
        JLT   DRAW            *IF NOT, JUMP TO DRAW
        LI    R3,63           *IF IS, RESET RIGHT
        JMP   DRAW            *
**************************
UPLEF   DEC   R4              *MOVE PRINT POSITION UP 1
```

```
         CI    R4,-1           *CHECK UP FOR OUT OF BOUNDS
         JNE   N2              *IF NOT, CHECK LEFT
         CLR   R4              *IF IS,RESET UP
N2       DEC   R3              *MOVE PRINT POSITION 1 TO LEFT
         CI    R3,-1           *CHECK LEFT FOR OUT OF BOUNDS
         JNE   DRAW            *IF NOT, JUMP TO DRAW
         CLR   R3              *IF IS, RESET LEFT
         JMP   DRAW            *
**************************
DNRIT    INC   R4              *MOVE PRINT POSITION DOWN 1
         CI    R4,48           *CHECK DOWN FOR OUT OF BOUNDS
         JLT   N3              *IF NOT CHECK RIGHT
         LI    R4,47           *IF IS, RESET DOWN
N3       INC   R3              *MOVE PRINT POSITION 1 TO RIGHT
         CI    R3,64           *CHECK RIGHT FOR OUT OF BOUNDS
         JLT   DRAW            *IF NOT, JUMP TO DRAW
         LI    R3,63           *IF IS, RESET RIGHT
         JMP   DRAW            *
**************************
DNLEF    INC   R4              *MOVE PRINT POSITION DOWN 1
         CI    R4,48           *CHECK DOWN FOR OUT OF BOUNDS
         JLT   N4              *IF NOT, CHECK LEFT
         LI    R4,47           *IF IS, RESET LEFT
N4       DEC   R3              *MOVE PRINT POSITION 1 TO LEFT
         CI    R3,-1           *CHECK LEFT FOR OUT OF BOUNDS
         JNE   DRAW            *IF NOT, JUMP TO DRAW
         CLR   R3              *IF IS, RESET LEFT
         JMP   DRAW            *
**************************
RELAY    JMP   LL              *A RELAY BECAUSE OF 'OUT OF RANGE'
**************************
**************************SCREEN COLOR
SC       CI    R5,>000F        *COMPARE TO >F (LAST COLOR)
         JNE   $+4             *
         CLR   R5              *IF >F THEN RESET TO 0
         INC   R5              *IF NOT >F THE INCREASE BY ONE
         MOV   R5,R0           *MOVE REG 5 INTO REG 0
         AI    R0,>0700        *ADD >0700 TO REG 0
         BLWP  @VWTR           *WRITE THE NEW SCREEN COLOR
         LI    R13,>2000       *DELAY LOOP
         DEC   R13             *
         JNE   $-2             *
         JMP   RELAY           *
**************************
**************************BLOCK COLOR
BC       AI    R14,>1000       *ADD ONE TO CURRENT BLOCK COLOR
         LI    R13,>2000       *DELAY LOOP .
         DEC   R13             *
         JNE   $-2             *
**************************
**************************DRAW
DRAW     LI    R15,2           *PUT 2 IN REG 15
         MOV   R3,R7           *MOVE COLUMN VALUE INTO REG 7
         CLR   R6              *CLEAR REG 6
         DIV   R15,R6          *DIVIDE REG 6 BY 2. QUOTIENT IS PUT IN REG 6
**************************WITH REMAINDER PUT IN REG 7
         MOV   R4,R9           *MOVE ROW VALUE INTO REG 9
         CLR   R8              *CLEAR REG 8
```

```
        LI    R15,8           *PUT 8 INTO REG 15
        DIV   R15,R8          *DIVIDE REG 8 BY 15. QUOTIENT IS PUT IN REG 8
**************************WITH REMAINDER PUT IN REG 9
        SLA   R6,3            *ShiftLeftArithmetic SAME AS MULTIPLYING BY 8
        SLA   R8,8            *ShiftLeftArithmetic SAME AS MULTIPLYING BY 256
        A     R6,R8           *ADD REG 6 AND REG 8, ANSWER PUT IN REG 8
        A     R8,R9           *ADD REG 8 AND REG 9, ANSWER PUT IN REG 9
        AI    R9,>0800        *ADD REG 9 TO >0800. ()0800 IS START OF PATERN
**************************DESCRIPTOR TABLE)
        MOV   R9,R0           *MOVE REG 9 INTO REG 0
        CLR   R1              *CLEAR REG 1. THE VALUE READ FROM THE PATTERN
**************************DESCRIPTOR TABLE WILL BE PUT INTO REG 0
        BLWP  @VSBR           *READ THE BYTE
        MOV   R1,R10          *STORE THE BYTE IN REG 1, IN REG 10
        CI    R7,0            *CHECK REMAINDER OF FIRST DIVISION TO SEE IF
**************************LEFT OR RIGHT BYTE IS TO BE CHANGED
        JEQ   LD              *IF ZERO THE JUMP TO LEFT DIGIT
        ANDI  R10,>F000       *RIGHT DIGIT TO BE CHANGED SO CLEAR THE RIGHT
**************************RIGHT DIGIT, BUT DO NOT CHANGE LEFT DIGIT
        SRL   R14,4           *ShiftRightLogical SHIFT ALL DIGITS FOUR PLACES
**************************TO THE RIGHT
        AB    R14,R10         *AddBytes
        SLA   R14,4           *SHIFT BACK FOUR PLACES TO LEFT
AL      MOV   R10,R1          *MOVE THE NEW COLOR INTO REG 1 SO IT CAN BE
**************************WRITTEN TO THE PATTERN DESCRIPTOR TABLE
        BLWP  @VSBW           *WRITE IT
        JMP   RELAY           *JUMP TO RELAY (ELSE OUT OF RANGE)
LD      ANDI  R10,>0F00       *CHANGE THE LEFT COLOR DIGIT, RIGHT UNCHANGED
        AB    R14,R10         *WRITE THE NEW COLOR TO REG 10. DO NOT SHIFT.
**************************IS IN CORRECT PLACE
        JMP   AL              *LEFT DIGIT UPDATED, JUMP BACK AND WRITE IT
        END
```

WEST JAX 99'ers
P.O. BOX 176
ORANGE PARK, FL. 32067