32K RAM EXPANSION BOARD FOR THE ATARI 400/800

USER MANUAL

Tara

Computer Products Inc.

TABLE OF CONTENTS

Please read this manual before attempting to install the Tara 32K RAM card in the Atari 400 or 800. Incorrect installation could cause permanent damage to both the RAM card and the computer.

Statler Building, 107 Delaware Ave.,
Suite 752, Buffalo, N.Y. 14202

2 Robert Speck Parkway, Suite 1540
Mississauga, Ontario L4Z 1H8

PRINTED IN CANADA

1

PREFACE

With the recent explosion of software for the Atari 400 and 800, first time users of these computers are finding themselves unable to take advantage of this software library. Recognizing this, Tara Computer Products has added to its family of memory expansion products an extremely cost-effective 32K RAM card for either the Atari 400 or 800. This card provides an Atari 400 with 32K in its single memory slot, and provides an Atari 800 with a full 48K memory map, with its original 16K complement. This card also provides the user with the quality construction, screen clarity and sharpness that one has come to expect from Tara.

Features:

- Ease of installation.
- Replaces existing 8K or 16K complement.
- Gold edge connectors for increased reliability.
- Allows for higher performance software and games to be run on Atari 400 or 800.
- Allows the 800 user to free up a slot normally taken up by inefficient 16K boards.
- Low cost per byte.
- Extensive manufacturing diagnostics and burn in.

This manual explains the installation, verification and usage of the Tara 32K RAM card, and some additional information on usage, from playing exciting 32K games to programming related information for those who tinker with the machine's bits.

2

INSTALLATION

This section takes you through the steps of installing the Tara RAM board in your Atari 400 or 800 computer. If you prefer, your dealer will be happy to install your Tara RAM board.

As you unpack your 32K RAM card check its contents against the items described on the packing list.

Fill out the pre-addressed warranty card and mail it in.

Leave RAM card in anti-stat package until ready for installation.

A Close Look

Let's examine the Tara RAM card for a moment. Handle the RAM card from its edges as you pull it out of the anti-stat bag and place it on a flat surface with the bag under it oriented the way you see it in fig. 1.1, and identify the major components of your RAM card, namely:

1 The Edge Connector. This connects the RAM card electrically to the computer and it is important not to touch the edge connector. This will prevent contamination of the contacts.

2 Dynamic RAM. These are the chips that make up the memory portion of the card. Each of these sixteen devices can store 16,384 x1 bits

of storage capability.

3 The rest of the logic on card provides data buffering and RAM select refresh logic.

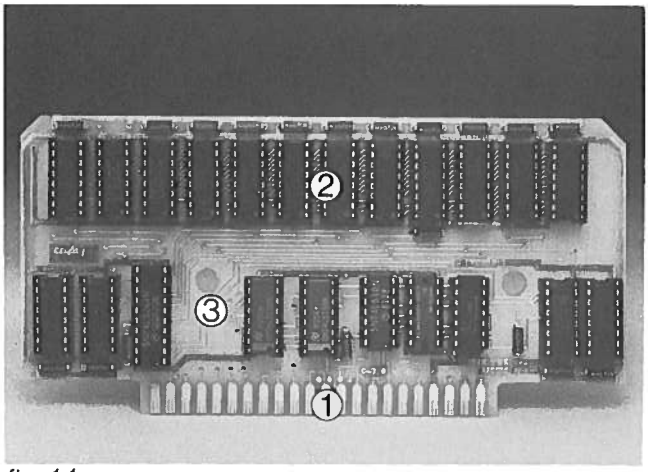


fig. 1.1

Preparing for Installation

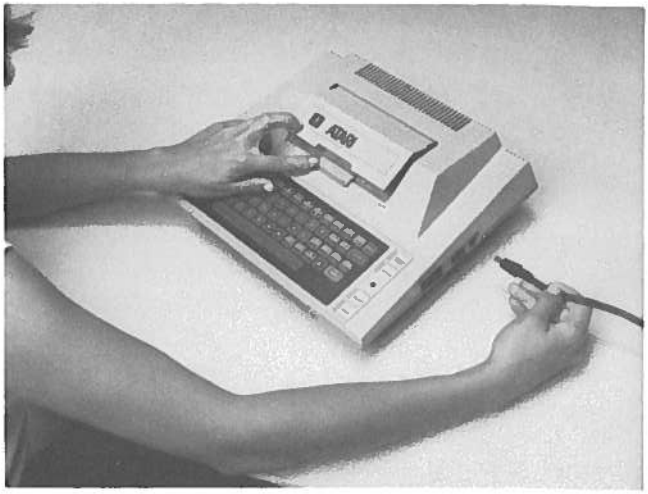
The only tool required is a medium Phillips screwdriver. Other than this, all that is required is a clear area where you can work. Please follow the step by step instructions, as they have been provided for quick and easy installation.

Please remember to keep related screws grouped and remember placement.

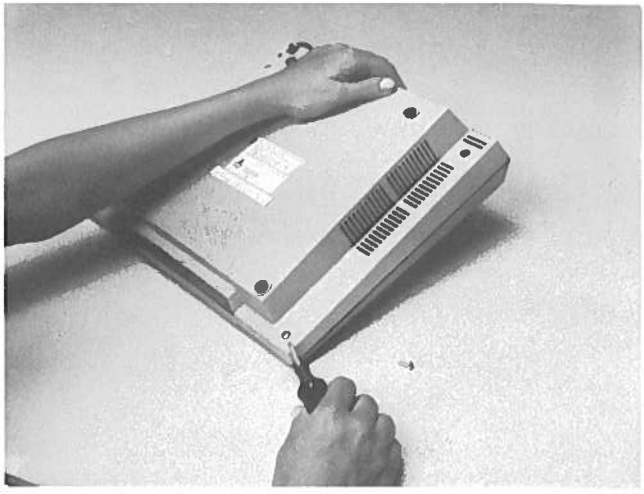
400 Installation:

Steps:

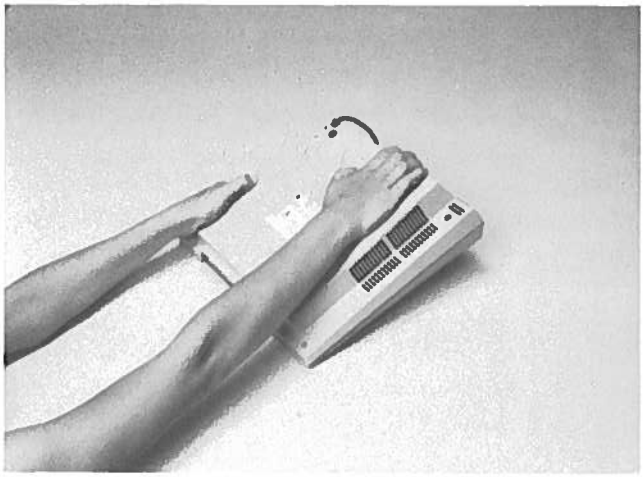
1. Remove all power to your computer.
2. Remove any cartridge that may be in your 400, close door and place the 400 bottom up toward you.



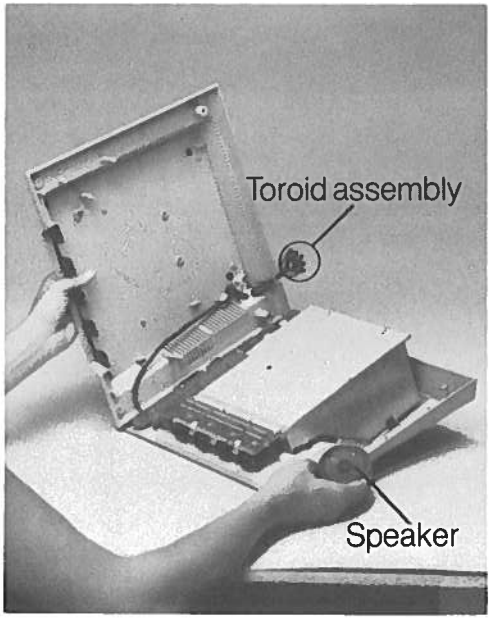
3. Remove the four corner screws.



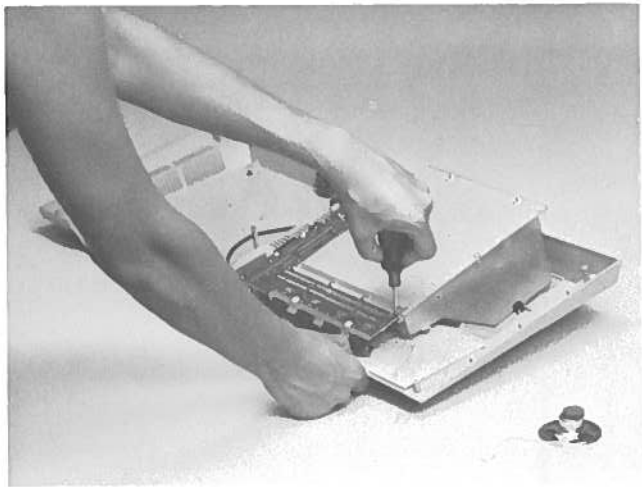
4. Remove the computer's bottom plastic case.



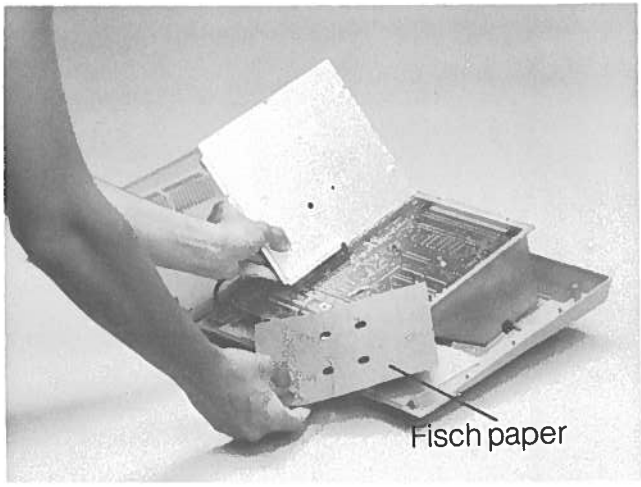
5. Fold to the left as shown. Please note placement of video cable and toroid for ease of installation. Remove speaker at this time and put it aside.



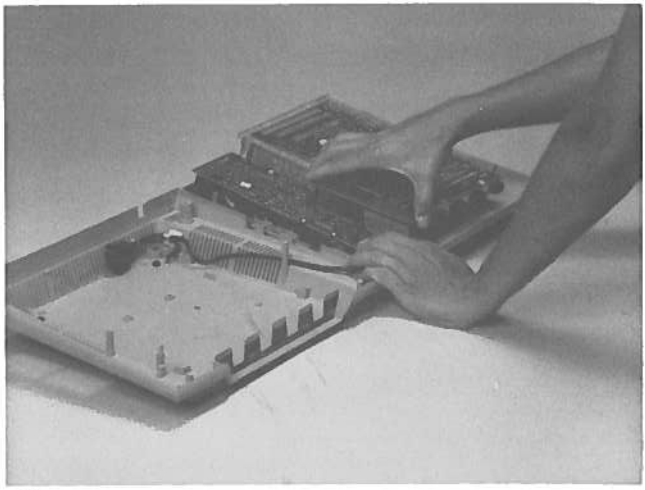
6. Remove the eight screws that hold on the computer's bottom shield plate. Keep these screws together.



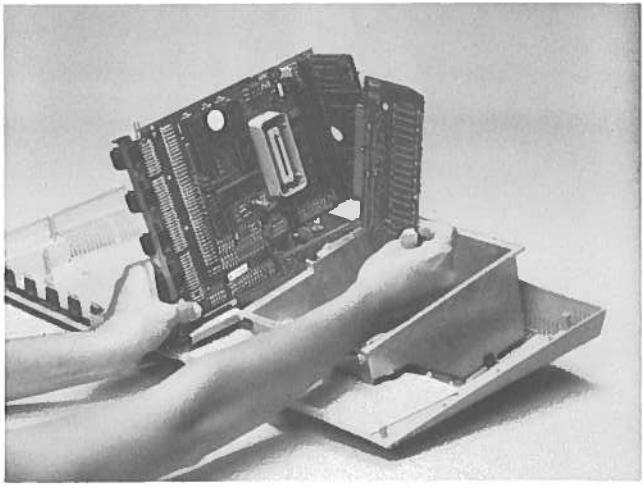
7. Remove plate, and remove fisch paper.



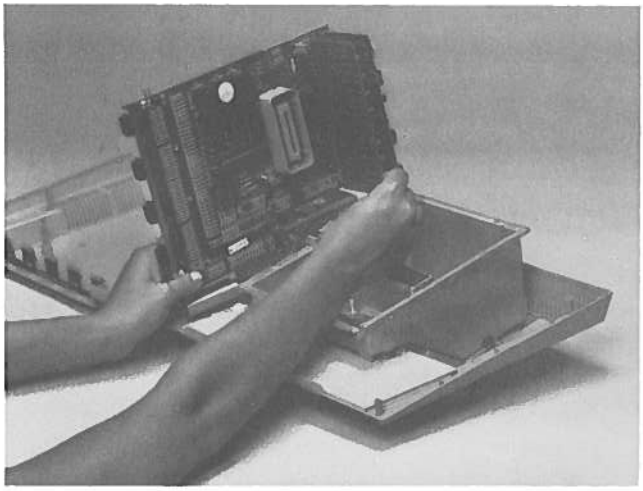
8. Grasp motherboard as shown and pry up gently but firmly to dislodge motherboard from connector. Pull motherboard out of shielded case, up and out to the left, taking care not to disconnect the keyboard ribbon assembly.



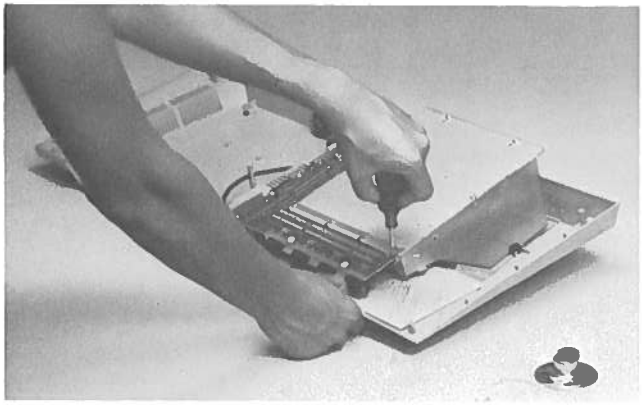
9. Hold motherboard as shown vertically and remove old 16K board from the computer's bus connector.



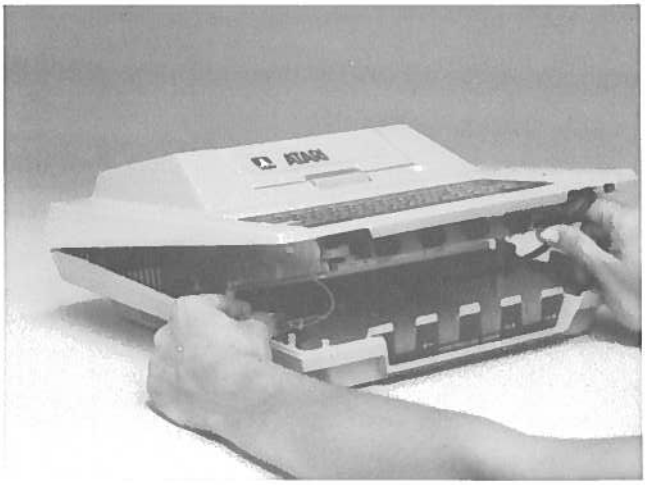
10. Install the Tara 32K RAM card as shown with solder side of the RAM card towards the keyboard of the computer.
11. Replace motherboard back into case as shown, and press firmly to re-connect the motherboard to the power connector.



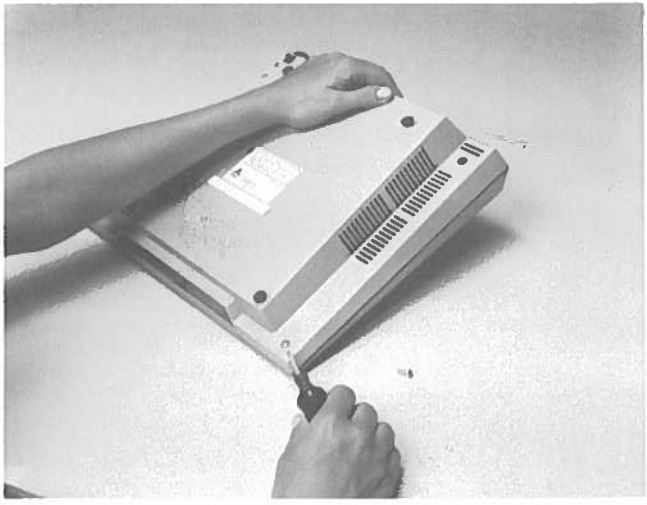
12. Replace fisch paper and bottom shield plate, and re-install the eight screws.
13. Replace bottom of computer being careful to align things together: i.e. the video cable toroid assembly.



14. Place machine in its upright position, lift left corner of case and relocate speaker as shown between plastic pins, then reconnect to speaker output.



15. Replace four bottom screws of computer.



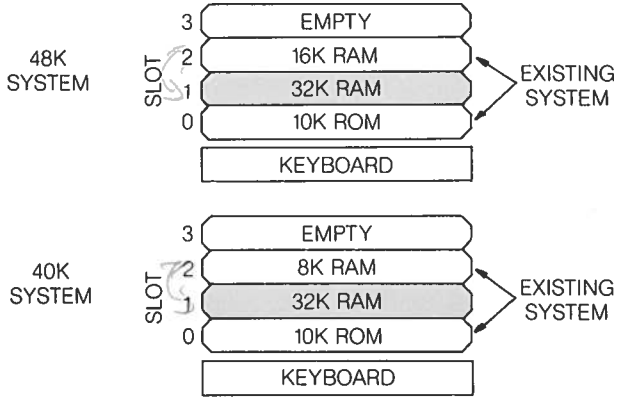
The Tara 32K RAM card is now installed in your Atari 400!

800 Installation

Installation of your 32K RAM card within your Atari 800 is somewhat easier as the I/O slots of the Atari 800 are readily accessible from the top of the computer by removing the computer's cover. The typical configuration consists of the original 8K or 16K RAM card and the Tara 32K RAM card in adjacent slots providing the Atari 800 with the full memory capacity of 48K.

Steps

1. Turn off power to your Atari 800 and unplug the AC cord as serious damage may result by installing the card with the power on!
2. Typical configurations are as follows:



3. Open cartridge access cover and locate rotating latches just to the left and to the right of the cartridge slots. Press release these latches so that they no longer press down on the metal ridge of the top cover. Next, pull the cover slightly toward the keyboard and up enabling

full access to the four slots as you pull the cover away.

4. Install the Tara 32K RAM card in the vacant slot 2. The component side of the card points to the rear of the computer, the solder side toward the keyboard. Check to ensure that the Tara RAM card is seated properly in the connector.
5. Replace the card case cover by noting the two metal tabs on the back of the cover that slip into two mating slots towards the rear of the 800. Press down on the back of the cover and slide it into the slots, then press the cover down on the front allowing for the latches to secure the cover back into position.
6. Reconnect power, switch on and check for proper operation.

Your Tara 32K RAM card is now installed in your Atari 800!

3

CHECKING IT OUT

Power Up and Initial Checkout

Re-connect power and install BASIC cartridge and wait for BASIC "ready" prompt. If all went well you should see the BASIC prompt. If not, power down and refer to troubleshooting section.

If a BASIC prompt awaits you type in the following:

Print FRE(Ø)

Computer should respond:

	Value
Atari 400 – 32K –	**29710**
Atari 800 – 32K –	**29710**
48K –	**37902**

which translates to about 29K free user space in BASIC on the Atari 400 or 800 (for those who feel short changed see section 4 for details).

If all the above initial tests are satisfactory then you have properly installed your 32K RAM.

Your newly installed Tara product has had extensive burn in and diagnostics to find any hard or soft errors at the factory. If you would like to test the system further refer to Appendix B.

Now that you have a 32K Atari 400 you might be asking yourself what this means and how you may take advantage of it. Well, to the casual Atari user the extended memory is transparent to your usage of the computer. That is, taking advantage of the RAM card is as simple as buying off-the-shelf software that can utilize your computer's extended RAM memory map. This means that you are no longer restricted to 16K programs that you buy or type in from your favorite magazine; you can now use any of the more powerful (memory hungry) software that is currently available for Atari 800 users. Boot any cassette or disk based product as you would normally load it and the software will load itself into your newly installed expanded RAM.

4

PROGRAMMING INFO

For BASIC Programmers

On your old 16K computer when a BASIC cartridge was in the 400 or 800 the "FRE" user space left after machine overhead was about 13K. This value is the memory available to you – that which is not already taken up by the operating system, the disk operating system, the display screen data, etc. . . . In other words, the amount of room that you have to program in. With the addition of the Tara 32K board you have now extended the amount of room you have to program in to about 29K, so feel free to go and find all those back issues of magazines and find all the 32K Atari BASIC programs and type them in. Simply said, for the Atari programmer this opens a larger work space for your advanced work or play.

Assembly programming

This section is for those who enjoy getting down to the heart of the matter, namely assembly language programmers. So for those whose immediate needs have been filled they need not read this section.

The addition of RAM is welcome in most cases to any assembler programmer and this case is no exception. The Atari computer is I/O memory mapped, so that means that some finite amount

of the 6502's address space has to be dedicated to I/O and operating system ROM's and in the case of the Atari 400 that region resides in the top end of memory from decimal 49152 to 65536. The region from zero to 32768 decimal is now where our new read–write memory locations reside. In the case of 32K installations our read–write memory map is not shared with the cartridge memory mapping. In the case of 48K installations the upper end of our RAM memory map from decimal 40960 to 49152 is shared with the 400 or 800 ROM slot. This is why when BASIC is run on the Atari there is only 38K user programmable memory left because with the BASIC cartridge in, the cartridge deselects that top bank of RAM. Atari 800 with both cartridges installed, sacrifices an even larger area, from 32768 to 49152 decimal. Needless to say with no cartridge installed the Atari has a user addressable area of the entire 48K RAM map minus zero page use and other system related needs of the RAM. (See Appendix D.)

5

APPENDIXES

Appendix A: Troubleshooting Hints

The installation of this pre-tested product should ensure functionality if installation instructions were carefully followed. In most cases problems can be as simple as plugging the RAM card in backwards. Recheck work and follow back through installation to check for missed step. Totally unfunctional systems can be traced to power or plugging the card in backwards (remember: solder side towards keyboard, component side towards back of computer). If problems persist return system to your Tara dealer for prompt remedial action. Otherwise contact us direct at Tara Computer Products for consultation.

Appendix B:
Further RAM Verification

The following Atari BASIC listings are provided for you to ensure the proper operation of the board. Type the listing exactly as shown into your computer and run it overnight as the total test time is quite lengthy. A total of six tests are performed and for those who are interested the BASIC listings are fully commented for a total understanding of the tests performed.

```
 10 REM  TARA Memory Test Program
 20 REM
 30 REM  by: Luke G. Matthys
 40 REM
 50 REM
 60 REM  This software is copyrighted
 70 REM  but permission is granted to
 80 REM  copy it for personal use.
 90 REM
100 REM  The purpose of this program is
110 REM  to test the available memory
120 REM  of the ATARI computer. Six
130 REM  tests will be performed. The
140 REM  tests range from simple
150 REM  constant value read-writes to
160 REM  intricate patterns designed to
170 REM  detect more subtle memory
180 REM  faults. Due to the BASIC
190 REM  environment limitations it is
200 REM  not possible to check all of
210 REM  the RAM in the system.
220 REM
230 PRINT  CHR$(125)
240 DIM  XT$(40)
```

```
250 DIM X2$(9)
260 X2$ = "ADDRESS"
270 DIM X3$(11)
280 X3$ = "EXPECTED"
290 DIM X4$(8)
300 X4$ = "FOUND"
310 XH = PEEK(741)+256*PEEK(742)-1
320 REM
330 REM The following DIM and DATA
340 REM statements are used to create
350 REM user defined machine code
360 REM functions. The DIM statement
370 REM defines a string which will
380 REM contain the code. The first
390 REM DATA statement provides the
400 REM number of bytes in the
410 REM function (and DATA statements
420 REM which follow). The DATA which
430 REM follows each of the length
440 REM bytes is the function's code.
450 REM Each function must pop the
460 REM argument count (1 byte) and
470 REM address (2 bytes, MSB followed
480 REM by LSB). The address is
490 REM operated on and return in
500 REM locations 212 (LSB) and 213
510 REM (MSB, zeroed in our case).
520 REM The function returns to BASIC
530 REM by executing an RTS opcode.
540 REM
550 REM
560 REM CMPL - complement
570 REM
580 DIM CMPL$(12)
590 DATA 12
600 DATA 104,104,104,73,255,133,212
610 DATA 169,0,133,213,96
620 REM
```

```
630 REM  LSB - least significant byte
640 REM
650 DIM  LSB$(12)
660 DATA  12
670 DATA  104,104,104,234,234,133,212
680 DATA  169,0,133,213,96
690 REM
700 REM  SWAP - swap nibbles of LSB
710 REM
720 REM  DIM  SWAP$(26)
730 DATA  26
740 DATA  104,104,169,0,133,213,104
750 DATA  170,41,15,10,10,10,10,
760 DATA  133,212,138,74,74,74,74,
770 DATA  5,212,133,212,96
780 REM
790 REM  CONST - constant value result
800 REM
810 DIM  CONST$(12)
820 DATA  12
830 DATA  104,104,104,169,255,133,212
840 DATA  169,0,133,213,96
850 REM
860 REM  RIPPLE - 1 shifted left by
870 REM  ( x mod 7 ) bits
880 REM
890 DIM  RIPPLE$(24)
900 DATA  24
910 DATA  104,104,104,41,7,170
921 DATA  232,169,1
930 DATA  202,240,3,10,144,250,234,234
940 DATA  133,212,169,0,133,213,96
950 REM
960 REM  The following block of code
970 REM  uses the DATA declared above
980 REM  to initialize the function
990 REM  strings. For the sake of
1000 REM  brevity an array of pointers
```

```
1010 REM  is initialized and used
1020 REM  within the main initializing
1030 REM  loop.
1040 REM
1050 DIM  FN(4)
1060 FN(0) = ADR(CMPL$)
1070 FN(1) = ADR(LSB$)
1080 FN(2) = ADR(SWAP$)
1090 FN(3) = ADR(CONST$)
1100 FN(4) = ADR(RIPPLE$)
1110 FOR  I = 0 TO 4
1120 READ  L
1130 FOR J = FN(1) TO FN(1)+L-1
1140 READ  N
1150 POKE  J,N
1160 NEXT  J
1170 NEXT  I
1180 REM
1190 REM  Commence testing...
1200 REM  The test subroutine requires
1210 REM  the title string (XT$), the
1220 REM  high address (XH$) and the
1230 REM  function pointer (XF).
1240 REM
1250 REM  TEST #1 - constant values
1260 REM  note that the specific value
1270 REM  to be used must be poked into
1280 REM  the function code at offset
1290 REM  4.
1300 REM
1310 XT$ = "TEST #1: CONSTANT $FF"
1320 XF = ADR(CONST$)
1330 POKE  XF+4,255
1340 GOSUB  2320
1350 XT$ = "          CONSTANT $00"
1360 POKE  XF+4,0
1370 GOSUB  2320
1380 XT$ = "          CONSTANT $AA"
```

```
1390 POKE XF+4,170
1400 GOSUB 2320
1410 XT$ = "          CONSTANT $55"
1420 POKE XF+4,85
1430 GOSUB 2320
1440 REM
1450 REM TEST #2 - address LSB
1460 REM Note that LSB contains 2 NOP
1470 REM opcodes which are altered to
1480 REM create a NOT LSB function
1490 REM
1500 XT$ = "TEST #2: ADDRESS L.S.B."
1510 XF = ADR(LSB$)
1520 POKE XF+3,234
1530 POKE XF+4,234
1540 GOSUB 2320
1550 REM
1560 REM TEST #3 - complement LSB
1570 REM LSB complements the result if
1580 REM 2 NOPs are changed to EOR $FF
1590 REM
1600 XT$ = "TEST #3: ADDRESS L.S.B. CO
     MPLEMENT
1610 POKE XF+3,73
1620 POKE XF+4,255
1630 GOSUB 2320
1640 REM
1650 REM 1660 REM TEST #4 - nibble swap
1660 REM
1670 XT$ = "TEST #4: ADDRESS L.S.B. NIB
     BLE SWAP"
1680 XF$ = ADR(SWAP$)
1690 GOSUB 2320
1700 REM
1710 REM TEST #5 - rippling ones
1720 REM Note that the ripple can also be
1730 REM altered to complement the
1740 REM result.
```

```
1750 REM
1760 XT$ = "TEST #5: RIPPLING ONES"
1770 XF = ADR(RIPPLE$)
1780 POKE XF+15,234
1790 POKEXF+16,234
1800 GOSUB 2320
1810 REM
1820 REM TEST #6 - rippling ones
1830 REM complement. Note the NOP,
     NOP
1840 REM change to EOR $FF.
1850 REM
1860 XT$ = "TEST #6: RIPPLING ONES
     COMPLEMENT
1870 POKE XF+15,73
1880 POKE XF+16,255
1890 GOSUB 2320
1900 END
1910 REM
1920 REM This subroutine acts as a
1930 REM mainline for the individual
1940 REM memory tests. The title of
1950 REM the test is printed and then
1960 REM the available memory low
1970 REM limit is adjusted. Note that
1980 REM this limit is adjusted to
1990 REM allow for dynamic memory
2000 REM allocation. The first FOR
2010 REM loop serves to initialize the
2020 REM memory to specific values.
2030 REM The calculation of these
2040 REM values is carried out by
2050 REM executing the appropriate
2060 REM machine code function pointed
2070 REM to by XF. The value derived
2080 REM is a function of the address.
2090 REM The loop also contains a
2100 REM "bells & whistles" statement
```

```
2110 REM to indicate that the program
2120 REM is still alive and well.
2130 REM This is achieved by calling
2140 REM SETCOLOR to change the back-
2150 REM ground color of the screen.
2160 REM Every 16 pokes the hue will
2170 REM change (SWAP will switch
2180 REM the address LSB's nibbles and
2190 REM this value will be normalized
2200 REM MOD 16). You may find this
2210 REM nauseating and may remove it
2220 REM at will - it may even speed
2230 REM up the test. The second FOR
2240 REM loop serves to verify the
2250 REM contents of memory. The
2260 REM actual contents are compared
2270 REM to the expected contents and
2280 REM discrepancies are reported.
2290 REM Here again a life sign is
2300 REM provided.
2310 REM
2320 PRINT XT$
2330 XL=XH-FRE(0)+50
2340 FOR I=XL TO XH:POKE I,USR(XF,I):
     SETCOLOR 2,USR(ADR(SWAP$),I),0:
     NEXT I
2350 FOR I=XL TO XH:IF PEEK(I)<>USR
     (XF,I) THEN PRINT X2$;I;X3$;
     USR(XF,I);X4$;PEEK(I)
2360 SETCOLOR 2, USR(ADR(SWAP$),I),0:
     NEXT I
2370 PRINT "          COMPLETE"
2380 RETURN
```

Appendix C:
Some Useful Peeks & Pokes

Memory Configuration

14,15 Display Screen Lower Limit (APPMHI)
These locations contain the highest location available for program lines and variables. Memory above that is used for the screen display.

88,89 Screen Memory Address (SAVMSC)
These addresses contain the lowest address of the screen memory. The value at that address is displayed at the upper left-hand corner of the screen.

106 Top of RAM Address
(Most Significant Byte) (RAMTOP)
This location contains a value 16 times the number of 4K RAM blocks present. PEEK (740)/4 gives the number of 1K blocks present.

741,742 Free Memory High Address (MEMTOP)
At any time, PEEK(741) + 256 * PEEK(742) – 1 is the highest memory location in the free memory area. The value changes when power is turned on, SYSTEM RESET occurs, or a channel is opened to the display.

743,744 Free Memory Low Address (MEMLO)
This location contains the address of the first location in the free memory region. The value changes when power is turned on or SYSTEM RESET occurs.

Display Screen

77 Attract Mode On/Off (ATRACT)
Setting this location to 0 disables attract mode on the display screen. This happens automatically whenever a key on the keyboard is pressed. Setting this location to 254 enables attract mode. This happens automatically after nine minutes without a key being pressed.

82 Left Margin of Text Area (LMARGN)

Specifies the column of the graphics mode 0 left margin. PEEK (82) will be between 0 and 39, 0 being the left edge of the screen. The default is 2.

83 Right Margin of Text Area (RMARGN)

Specifies the column of the graphics mode 0 right margin. PEEK (83) will be between 0 and 39, 39 being the right edge of the screen. The default is 39.

84 Current Row Cursor Position (ROWCRS)

Specifies the row where the next read or write to the main screen will occur. PEEK (85) will be at least 0; its highest value depends on the graphics mode.

85,86 Current Column Cursor Position (COLCRS)

Specifies the row where the next read or write to the main screen will occur. PEEK (84) will be at least 0; its highest value depends on the graphics mode. Location 86 will always be 0 in graphics modes 0 through 7.

87 Display Mode (DINDEX)

This location contains the current screen mode.

90 Starting Graphics Cursor Row (OLDROW)

This location determines the starting row for the DRAWTO and XIO 18 (graphics FILL) statements.

93 Cursor Character Save/Restore (OLDCHR)

This location contains the character that is underneath the visible text cursor. The value is used to restore the hidden character when the cursor moves.

91,92 Starting Graphics Cursor Column (OLDCOL)

This location determines the starting column for the DRAWTO and XIO 18 (graphics FILL) statements.

94,95 Cursor Memory Address (OLDADR)

This location contains the memory address of the current visible text cursor. The value is used in conjunction with OLDCHR (location 93) to restore the original character hidden by the cursor when the cursor moves.

96 Ending Graphics Cursor Row (NEWROW)
This location determines the ending row for the DRAWTO and XIO 18 (graphics FILL) statements.

97,98 Ending Graphics Cursor Column (NEWCOL)
This location determines the ending column for the DRAWTO and XIO 18 (graphics FILL) statements.

201 Display Screen Tab Interval (PTABW)
Specifies the number of columns between each tab stop. The first tab will be at column number PEEK (201). The default is 10.

656 Text Cursor Row Position (TXTROW)
Specifies the row where the next read or write to the split-screen text window will occur. PEEK (656) will be between 0 and 3, 0 being the top of the split-screen text window.

657,658 Text Cursor Column Position (TXTCOL)
Specifies the column position where the next read or write to the split-screen text window will occur. PEEK (657) will be between 0 and 39, 0 being the first column of the split-screen text window. Location 658 is always 0 unless you change it.

675-680 Display Screen Tab Stop Map (TABMAP)
The tab stops are retained in a 15-byte (120-bit) map. Each bit corresponds to a column on a logical line. If the bit is on, a tab stop is set in that column. Whenever you open the display screen (device S: or E:), each byte of this map is assigned the value 1, thereby providing default tab stops at columns 7, 15, 23, and so on.

752 Cursor Inhibit (CRSINH)
When this location has a value of 0, the display screen cursor will be visible. When the value is nonzero, the cursor will be invisible. Cursor visibility does not change until the next time the cursor moves.

755 Character and Cursor Control (CHACT)
This location normally has a value of 2. Other

values can make the cursor opaque or invisible and can make all characters display upside-down.

756 Character Address Base (CHBAS)

This variable determines which character set will be used in screen modes 1 and 2. A value of 224 provides the capital letters and number set; a value of 226 provides the lower-case letters and graphics character set.

765 Fill Data (FILDAT)

This location contains the data value for the region to be filled by an XIO 18 command.

766 Display Control Characters (DSPFLG)

When this location is 0, the ATASCII codes 27-31, 123-127, 187-191, and 251-255 perform their normal display screen control functions. When this location is nonzero, these ATASCII codes generate characters on the display screen.

659 Split-Screen Text Window Screen Mode (TINDEX)

This location contains the current split-screen mode.

669,661 Split-Screen Memory Address (TXTMSC)

These locations contain the lowest address of the split-screen memory. The value of that address is displayed at the upper left-hand corner of the split-screen text window.

665-667 Split-Screen Cursor Data

These locations contain the split-screen equivalents of OLDCHR (location 93) and OLDADR (locations 94 and 95).

763 Last ATASCII Character or Plot Point (ATACHR)

This location contains the ATASCII code for the character most recently written or read, or the value of the graphics point last displayed. The value at this location is used to determine the line color when a DRAW or XIO 18 (FILL) is performed.

54273 Character Control Register (CHACTL)

Same as location 755 (CHACT)

Display Lists

512,513 Display List Interrupt Vector (VDSLST)
These locations store the address of the instructions that will be executed in the event of a display list interrupt.

559 DMA Control Register (SDMCTL)
This location enables or disables direct memory access. The default value is 22, which enables DMA for fetching display list instructions and for retrieving normal playfield display data. A value of 0 disables DMA.

560,561 Display List Address (SDLST)
This location stores the address of the active display list.

54286 Non-maskable Interrupt Enable (NMIEN)
This location enables or disables the display list interrupt and the vertical blank interrupt. A value of 0 disables the display list, 128 disables the vertical blank and enables the display list, and 192 enables both.

Player-Missile Graphics

623 Player/Playfield Priorities (GPRIOR)
This location determines what color will display when players overlap playfield objects. A value of 1 gives all players priority over the playfield. A value of 2 gives players 0 and 1 priority over all playfield registers, and priority over players 2 and 3 as well. A value of 4 gives the playfield priority over players. A value of 8 gives playfield color registers 0 and 1 priority over all players and priority over playfield registers 2 and 3.

704-707 Player-Missile Color Registers (COLPM0-COLPM3)
Each of these locations determines the color of a player and its associated missile.

53248-53251 Player Horizontal Position Registers (HPOSP0-HPOSP3)
Each of these locations determines the horizontal position of one player. Values range between 0 (the

left edge of the screen) and 277 (the right edge of the screen).

53256-53259 Player Width Registers (SIZEP0-SIZEP3)

Each location changes the magnification factor used to display one player. A value of 0 or 2 displays a player at normal width, 1 displays twice normal width, and 3 displays quadruple width.

53260 Missile Width Register (SIZEM)

This location controls the magnification of all four missiles. A value of 0 or 2 displays missiles at normal width, 1 displays twice normal width, and 3 displays quadruple width.

53277 Graphics Control Register (GRACTL)

Along with location 559 (DMACTL), this location controls DMA for player-missile graphics. A value of 2 enables player DMA only, a value of 1 enables missile DMA only, and a value of 3 enables both.

54279,54280 Player-Missile Base Register (PMBASE)

These locations contain the starting address of the player-missile definition table.

Cassette Buffer

61 Cassette Buffer Pointer (BPTR)

This location contains a pointer to the next location to be used in the cassette buffer. The value may be anything from 0 to the value in BLIM (location 650). If BPTR = BLIM, then the buffer is full if writing or empty if reading.

63 Cassette End-of-File Flag (FEOF)

This location is used by the cassette handler to indicate whether an end-of-file has been detected. If the value of this location is 0, an end-of-file has not yet been detected; if the value is not 0, it has been detected.

64 Beep Count (FREQ)

This location contains the number of beeps requested by the cassette handler.

649 Cassette Read/Write Mode Flag (WMODE)
This location specifies whether the current cassette operation is read (value = 0) or write (value = 128).

650 Cassette Buffer Size (BLIM)
This location contains the number of active data bytes in the cassette buffer. BLIM will have a value from 0 to 128.

1021-1151 Cassette Buffer (CASBUF)
These locations are a buffer used by the cassette handler to read data from and write data to the program recorder.

Keyboard

17 BREAK Key Flag (BRKKEY)
A 0 in this location indicates that the BREAK key has been pressed.

694 Inverse Video Keystrokes (INVFLG)
When this location is 0, keystrokes generate ATASCII codes for normal video characters. If the value is nonzero, keystrokes generate ATASCII codes for inverse video characters.

702 Shift/Control Lock Flag (SHFLOK)
Meaningful values for this location are 0 (normal mode – no locks in effect), 64 (caps lock), and 128 (control lock).

764 Keyboard Character (CH)
This location reports the value of the most recently pressed key, or the value 255, which indicates that no key has been pressed.

767 Start/Stop Display Screen (SSFLAG)
When this location is 0, screen output is not stopped. If the value is 255, output to the screen is stopped. The value is complemented by pressing CTRL-1.

53279 CONSOLE Switch Port (CONSOL)
This location has two uses. PEEK (53279) tells whether a special function key is pressed. To ensure an accurate reading, do a POKE 53279,8 before doing a PEEK(53279).

POKE 53279,0 extends the cone of the built-in speaker. POKE 53279,8 retracts it. Alternate the two statements repeatedly to produce a series of clicks from the speaker. The operating system effectively does an automatic POKE 53279,8 every 1/60 second.

Sound Control

65 Input/Output Noise Control (SOUNDR)

This location is normally nonzero. In that case, noise is audible over the television audio circuit during disk or cassette read and write operations. If this location is 0, the noise is inhibited.

Printer

29 Printer Buffer Pointer (PBPNT)

This location specifies the current position in the computer's printer buffer. The value ranges from 0 to PBUFSZ (location 30).

30 Printer Buffer Size (PBUFSZ)

This location specifies the size of the computer's printer buffer. The value is 40 for normal mode or 29 for sideways mode.

960-999 Printer Buffer (PRNBUF)

The printer handler collects output from LPRINT statements to the printer in the computer's printer buffer, sending it out when an EOL occurs, or when the buffer is full.

Free Area

1536-1663 Conditionally Available

These locations are normally free for machine language programs, display lists, and so forth. However, whenever the INPUT statement retrieves more than 128 characters, it uses these locations to hold the characters in excess of 128.

1664-1791 Unconditionally Available

These locations are always free for machine language programs, display lists, and so forth.

BASIC Program Control

186,187 Stop Line Number (STOPLN)

These locations report the line number in which a BASIC program halts because of a STOP or TRAP statement, an error, or use of the BREAK key.

195 Error Number (ERRSAV)

If an error occurs, its number is placed in this location. Appendix A translates error numbers to messages.

251 Radians or Degrees (RADFLG or DEGFLG)

If the value of this location is 0, trigonometric functions calculate in terms of radians, if 6, in terms of degrees.

564,565 Light Pen Position (LPENH and LPENV)

Location 564 reports the horizontal position of a light pen. Location 565 reports the vertical position. These are not the same as the actual screen row and column numbers. There are 228 horizontal positions (each is called *color clock*). The leftmost horizontal position is 67. Each time you move the light pen one position to the right, the value in location 564 increases by 1. After the value reaches 255, it resets to 0 and resumes counting by 1 from there. The rightmost horizontal position is 7. There are 96 vertical positions, numbered from 16 at the top of the screen to 111 at the bottom.

Interrupt Control

53744 IRQ Interrupt Status/Enable (IRQST/IRQEN)

This location reports interrupt status via PEEK, or enables interrupts via POKE. Each bit corresponds to a different interrupt (see Table G-3). With PEEK, a 0 bit means the corresponding interrupt is present and a 1 bit means it is not present. With POKE, a 0 bit disables the corresponding interrupt and a 1 bit enables it.

Appendix D:
Atari 400/800 Memory Map