ATARI ASSEMBLER EDITOR USER'S MANUAL ERRATA

The purpose of this errata compilation is to rectify the typographical and logical errors which have been discovered in the Assembler Editor User's Manual. The enclosed material should be studied thoroughly since many of the errors prevent example programs from working correctly, give information that is erroneous, or omit important information.

Following is a page-by-page accounting of all errors and omissions which have surfaced to date. In most cases, the line in error is printed, followed by the corrected version of the line (preceeded by a ➤) and any necessary explanatory notes.

INSIDE COVER

***** The codes listed here are BASIC error codes, not Assembler ones. Appendix 1 contains the Assembler error codes.

PAGE vii, CONTENTS

***** The "GETTING STARTED" section contains some erroneous page numbers. The section should look like this:

***** Also, the "USING THE EDITOR" section contains some page numbering typos. Correction:

PAGE viii, CONTENTS

***** In the "APPENDICES" for Appendix 9, the heading should read:

PAGE 11

***** Under "HOW TO WRITE OPERANDS", LABEL is misspelled:

Please refer to the description of the LABL= directive for an explanation of the definitions of lines 100 and 200.

➤ Please refer to the description of the LABEL= directive for an explanation of the definitions of lines 100 and 200.

PAGE 12

***** Using Indirect Indexed Operands will sometimes produce an Error 12 yet the source code appears to assemble correctly anyway! Use with caution; examine the object code to be certain.

PAGE 17

***** Under "REP" Command:

REP/OLD/NEW

➤ REP/OLD/NEW/

PAGE 18

***** In "Sample Program":

➤ Line numbers 20 thru 90 on sample form were omitted.

***** Near bottom of page, on screen listing:

50 IMY TALLY

➤ 50 INY TALLY

PAGE 19

***** Under "LIST" Command, insert a comma after (#C:):

Other possible devices are the printer (#P:), Program Recorder (#C:) and disk drive (#D1 through #D8: or #D:, which defaults to #D1:).

➤ Other possible devices are the printer (#P:), Program Recorder (#C:), and disk drive (#D1 through #D8: or #D:, which defaults to #D1:).

PAGE 20

***** In Line 30 of example program near middle of the page:

```
    ▼      ▼
30   REP LDX,ABSX,Y
```

➤
```
30 REP LDX ABSX,Y
```

```
    ▼    ▼ ▼
70   ABSX = $3744
80   XEQ = *+$60
    ▲   ▲ ▲
```

➤
```
70 ABSX=$3744
80 XEQ=*+$60
```

And further on down in the same example:

```
    ▼
70   ABSX=$3744
80   XEQ=*+$60
    ▲
```

➤
```
70 ABSX=$3744
80 XEQ=*+$60
```

PAGE 22

***** Near top of page, by "ENTER#C:":

Note that ENTER#C: clears the edit text▼buffer before retrieving the source program.

➤ Note that ENTER#C: clears the edit text buffer before retrieving the source program.

***** Near bottom of page, "LOAD" command:

```
         ▼         ▼
          device:
LOAD#
          filespec
```

➤ LOAD# $\left\{\begin{matrix}\text{device:}\\ \text{filespec}\end{matrix}\right\}$

PAGE 23

***** In the "LOAD#C:" section, it in fact is not possible to CLOAD object code from cassette. You must use the following routine to load your object code:

➤
```
100 TRAP 260
110 OPEN #3,4,0,"C:"
120 GET #3,X
130 GET #3,X
140 GET #3,X
150 GET #3,Y
160 ADSTART=256*Y+X     S
170 GET #3,X
180 GET #3,Y
190 ADEND=256*Y+X       E
200 ADCUR=ADSTART       C
210 GET #3,X
220 POKE ADCUR;X        C
230 ADCUR=ADCUR+1       C
240 IF ADCUR<= ADEND THEN GOTO 210          C <= E
250 GOTO 140
260 CLOSE #3
270 END
```

PAGE 25

***** ASM#[#D[n]:PROGNAME[.SRC]]...etc.

➤ ASM [#D[n]:PROGNAME[.SRC]]...etc.

PAGE 28

***** Under "Title and Page Directives", a hyphen is missing:

We explain these directives together because they are intended to be used together to provide easily read information about the assembled program.

➤ We explain these directives together because they are intended to be used to provide easily-read information about the assembled program.

PAGE 29

***** The .TAB Directive doesn't seem to work as explained. TABs do occur but not precisely with the spacing you specified. There doesn't seem to be a simple method to correct this bad TABbing since the TABs which occur do not exhibit a pattern. C'est la vie.

PAGE 30

***** Under "BYTE Directive":

The rules for writing and evaluating an expression are given in Appendix D.

The rules for writing and evaluating an expression are given in Appendix 5.

PAGE 31

***** Under "LABEL=DIRECTIVE":

The rules for writing and evaluating an expression are given in Appendix 4.

The rules for writing and evaluating an expression are given in Appendix 5.

NOTE OF CAUTION:

Due to an assembler bug, forward referencing of labels does not always guarantee that label expressions will evaluate correctly.

For example, do not code:

```
10 COLREG=PVAL+$2000
20 PVAL=$1000
```

Instead:

```
10 PVAL=$1000
20 COLREG=PVAL+$2000
```

Again, forward referencing as in the first case may work, but be cautious and avoid it altogether.

PAGE 32

***** Near middle of page, the wrong integer is used:

The effect of the directive is to reserve 24 locations immediately after TABLE35.

The effect of the directive is to reserve 36 locations immediately after TABLE35.

***** Under the "IF Directive" section:

If the expression is not equal to zero, the IF directive has no effect on assembly.

If the expression is not equal to zero, all the code between lines 900 and 990 will not be assembled.

NOTE:

The spacing in the page 32 source code example is not entirely correct:

IF directives: at least 2 spaces between line number and the directive. Spacing on either side of the "@" is not critical.

IF label: 1 space between line number and the label which delimits the IF directive.

Op Code: at least 2 spaces between line number and the op code mnemonic.

Comment: 1 space between line number and semi-colon.

The corrected source code should read:

```
0100 ;CONDITIONAL ASSEMBLY EXAMPLE
0120 Z=0
0130  *=$5000
0140 LDA=$45
0150  .IF Z@ZNOTEQUAL0
0160  TAX;THIS CODE ASSEMBLED IFF Z=0
0170 ZNOTEQUAL0
0180  .IF Z-1@ZNOTEQUAL1
0190  ASL A ;THIS CODE ASSEMBLED IFF Z=1
0200 ZNOTEQUAL1
0210  INX;THIS CODE ALWAYS ASSEMBLED
```

PAGE 35

***** Under "DEBUG COMMANDS", Trace Operation:

Tmmmm Trace Operation

T or Tmmmm Trace Operation

PAGE 36

***** Under "D or Dmmmm Display Memory" section, the Example is incorrect. To wit:

Dmmmm,yyyy where yyyy is less than or equal to mmmm shows the contents of address mmmm.

Dmmmm,yyyy where yyyy is less than or equal to mmmm shows the contents from mmmm to yyyy, inclusive, with address "wraparound" occuring at address $FFFF.

Example:

D5000,0 [RETURN]

This will display address contents from $5000 to $0000.

D5000,100 [RETURN]

This will display address contents from $5000 to $0100 (address $0000 follows $FFFF here).

NOTE:

D5000,5000 will display only address $5000.

PAGE 37

***** Halfway down page under second Example:

```
500B 18 41 54 41 52 49 20 20
```

```
5008 18 41 54 41 52 49 20 20
```

PAGE 38

***** Under "Vmmmm Verify Memory":

Vmmmm yyyy,zzz compares memory yyyy to zzzz with memory starting at mmmm, and shows mismatches.

➤ Vmmmm yyyy,zzzz compares memory yyyy to zzzz with memory starting at mmmm, and shows mismatches.

***** First sentence:

The second command puts 34 and 87 in locations 700B and 700E respectively.

➤ The second command puts 31 and 87 in locations 700B and 700E respectively.

PAGE 40

***** Under "A Assemble One Instruction Into Memory" the spacing is incorrect:

```
5001<LDY $1234 [RETURN]
5001 AC3412            computer responds
<INY [RETURN]
5004 C8
```

➤
```
5001< LDY $1234 [RETURN]
5001 AC3412            computer responds
< INY [RETURN]
5004 C8
```

NOTE:

Always leave a space after the < when specifying a mnemonic to be assembled under this command.

***** Under "Gmmmm Go (Execute Program)":

➤ A BRK (op code = $00) instruction will also stop the "GO" command. So, even though the Debugger does not support an explicit breakpoint facility, you can simulate this facility with a little extra work.

For example, if you coded the following:

```
ADR       OBJ CODE     SOURCE

0604      AD6745       LDA $4567
0607      AC3412       LDY $1234
060A      C8           INY
 .          .            .
 .          .            .
 .          .            .
```

and wished to specify a breakpoint at 060A, you could:

1. Record the current contents of address 060A ("C8").
2. Use the "C" (Change) instruction to place a BRK ("00") at 060A.
3. Execute the "G" (Go Execute) command somewhere prior to the breakpoint.
4. The program will halt at 060A, displaying the current contents of all registers.
5. When you wish to continue, replace 060A previous contents ("C8") and perhaps place a breakpoint further along in memory. At any rate, restart execution with a G060A command to insure the INY at 060A gets executed.

***** Under "Tmmmm Trace Operation":

➤ TRACE stops when it encounters CPY. Merely restart the TRACE at the next instruction (the CPY having been executed prior to TRACE stopping).

➤ Also, typing T defaults to the address of the instruction immediately following the last instruction executed by a previous T or S command.

PAGE 47

***** Midway down page, bad spacing:

```
50 ▼HERE=*+5
```

➤ 50 HERE=*+5

***** Near bottom of page:

The asterisk also signifies multiplication (see Appendix ▼6).

➤ The asterisk also signifies multiplication (see Appendix 5).

PAGE 49

***** In explanatory section at bottom of page:

```
Z,PAGE X•Z PAGE,Y•ZERO PAGE INDEXED
ABS,X ABS,Y ABSOLUTE INDEXED
(IND)Y•INDIRECT INDEXED
```

```
Z,PAGE X•Z,PAGE Y•ZERO PAGE INDEXED
ABS,X•ABS,Y•ABSOLUTE INDEXED
(IND),Y•INDIRECT INDEXED
```

PAGE 51

***** Under "Examples":

```
600  LDA LABEL & $00FF
600  LDA #LABEL & $00FF

620  LDA LABEL/256
620  LDA #LABEL/256
```

PAGE 53

*****

```
.PAGE 'MESSAGE"
.PAGE "MESSAGE"
```

*****

```
.BYTE "AB...N"
.BYTE "A,B,...N"
```

***** .LABEL assembles following code, up to .LABEL, if and only if expression evaluates to zero.

LABEL assembles following code, up to LABEL, if and only if expression evaluates to zero.

PAGE 60

***** Under "Notes" section:

2. Except as shown, characters from 128-255 are reverse colors of 1 to 127.

2. Except as shown, characters from 128-255 are reverse video of 1 to 127.

4. To get ATASCII code, tell computer (direct mode) to PRINT ASC("__").

➤ 4. To get ATASCII code, tell computer (direct mode in BASIC) to PRINT ASC("__").

PAGE 64

***** 2nd full paragraph states that the programs in this section will work if typed in exactly as listed. They won't. There should be only 1 space between line numbers and labels and between line numbers and semi-colons which denote comments.

***** ASM,,#C: works for short programs but long ones get timeout errors when loaded. ASM opens the file to tape, then assembles, then writes to tape. This takes too long.

➤ To fix this problem, assemble in memory and then SAVE to cassette.

PAGE 65

***** The first paragraph states to CLOAD your object file from cassette. It can't be done. See Page 23 errata for a BASIC program to accomplish the loading from cassette.

PAGE 67

***** The nifty subroutine is in error:

```
25010?J+5;"E$(";A;",",B,")=";CHR$(34);
25020 FOR I=A TO B:?"ESC ESC";CHR$(PEEK)I+C));:NEXT I
```

➤
```
25010?J+5;"E$(";A;",";B;")=";CHR$(34);
25020 FOR I=A TO B:?"ESC ESC";CHR$(PEEK(I+C));:NEXT I
```

***** Midway down page:

To make this line part of your BASIC prgram simply move the cursor up to the line and press [RETURN].

➤ To make this line part of your BASIC program simply move the cursor up to the line and press [RETURN].

PAGE 68

***** Regarding the following example programs (pages 68-74), keep in mind that there should be only one space between the line number and label and one space between the line number and a semi-colon which denotes a comment statement. With this as a warning, the listing of individual bad spacing errors in the 4 example programs is not necessary (the list would be too lengthy anyway).

PAGE 69

***** Invert lines 0250 and 0260.

Also:

```
                   ▼
    0420    CMP #$BO
 ➤  0420    CMP #$B0
                  ▼
    0470    LDA #$OE
 ➤  0470    LDA #$0E
```

PAGE 70

```
       ▼
*****          ;ROUTINE SPLAY
   ➤   30      ;ROUTINE SPLAY
                    ▼
*****  0180 COLORO = $02C4 LOCATION OF COLOR REGISTERS
   ➤   0180 COLOR0 = $02C4 LOCATION OF COLOR REGISTERS
```

PAGE 73

```
                ▼    ▼
*****  0400    LOOP 1    STA   (POINTA),Y
   ➤   0400 LOOP1    STA   (POINTA),Y
```

PAGE 74

***** Top of page:

```
                 ▼
    0628    8D102    0640  STA  VDSLST+1
 ➤  0628    8D0102    0640  STA  VDSLST+1
```

*****   Lines 0910 to 0970 need revision:

```
065D   8D0AD0   0910    STA COLBAK   BLACKEN ALL REGISTERS
0660   8D16D0   0920    STA COLPF0
0663   8D17D0   0930    STA COLPF1
0669   E6CF     0940    STA COLPF2
066B   68       0950    INC DECK     NEXT DECK
066C   40       0960    PLA          RESTORE ACCUMULATOR
                0970    RTI          DONE

065D   8D1AD0   0910    STA COLBAK   BLACKEN ALL REGISTERS
0660   8D16D0   0920    STA COLPF0
0663   8D17D0   0930    STA COLPF1
0666   8D18D0   0940    STA COLPF2
0669   E6CF     0950    INC DECK     NEXT DECK
066B   68       0960    PLA          RESTORE ACCUMULATOR
066C   40       0970    RTI          DONE
```

PAGE 75

*****   SAVE #C: xxxx,yyyy... etc.
SAVE #C:<xxxx,yyyy... etc.

*****   LOAD #C: does not work. Please refer to Page 23 Errata for a BASIC program to load object code from cassette.

*****   Three more Editor commands should be described in this section:

LOMEM	Bumps the Edit Text buffer (your source program) upward in memory. See page 7.
SIZE	Gives memory map buffer addresses for Edit Text Buffer and user RAM. See page 6.
DOS	Switches to DOS menu, destroying current assembler RAM program in the process. No page reference in this manual.

PAGE 77

```
*****    10            *=600
         10       *=$600

         70            END
         70       .END
```

If further errors or omissions are discovered, please inform ATARI by filling in and mailing the pre-addressed form which is provided with the Assembler Editor User's Manual.