# EDITOR/ASSEMBLER

THE NEW EDITOR/ASSEMBLER PACKAGE EXTENDS THE FLEXIBILITY OF THE TI-99/4A BY ALLOWING USERS TO PROGRAM THE TI-99/4A ON THE MACHINE LEVEL VIA TMS9900 ASSEMBLY LANGUAGE. THIS GIVES USERS DIRECT ACCESS TO ALL SYSTEM FEATURES, SUCH AS SOUND, SPEECH, GRAPHICS, AND I/O, AS WELL AS PROVIDING THE HIGHEST SPEED POSSIBLE FROM THE COMPUTER'S 16-BIT MICROPROCESSOR.

ASSEMBLY-LANGUAGE PROGRAMS WRITTEN USING THE EDITOR/ASSEMBLER PACKAGE CAN EITHER BE RUN AS STAND-ALONE PROGRAMS OR LINKED INTO TI BASIC PROGRAMS AS SUBROUTINES. THE USER HANDLES THE LINKING PROCEDURE AND OTHER TASKS RELATED TO USING THE PACKAGE VIA SEVEN NEW BASIC STATEMENTS PROVIDED IN THE SOFTWARE -- CALL LOAD, CALL LINK, CALL INIT, CALL PEEK, CALL POKEV, CALL PEEKV, AND CALL CHARPAT.

ASSEMBLY-LANGUAGE SUBROUTINES WRITTEN USING THE EDITOR/ASSEMBLER CAN ALSO BE CALLED FROM EXTENDED BASIC. THE OUTPUT OF SUCH SUBROUTINES IS DIRECTLY USABLE BY THE EXTENDED BASIC PROGRAM, RATHER THAN REQUIRING CONVERSION AS WAS PREVIOUSLY THE CASE.

THE EDITOR/ASSEMBLER PACKAGE INCLUDES A SOLID STATE SOFTWARE (TM) COMMAND MODULE AND TWO FLOPPY DISKETTES, PLUS AN OWNER'S MANUAL WHICH PROVIDES EXTENSIVE DOCUMENTATION OF THE TI-99/4A'S SOFTWARE ARCHITECTURE. THE EDITOR AND ASSEMBLER ARE STORED ON ONE DISKETTE, WHILE THE COMMAND MODULE STORES THE SOFTWARE REQUIRED FOR LOADING THE EDITOR/ASSEMBLER AND EXECUTING TMS9900 ASSEMBLY-LANGUAGE PROGRAMS. THE DISKETTES ALSO CONTAIN THE SOURCE AND OBJECT CODE FOR AN INTERACTIVE ASSEMBLY-LANGUAGE DEBUGGER. IN ADDITION, THE SOURCE AND OBJECT CODE FOR THE GAME TOMBSTONE CITY ARE INCLUDED AS A SAMPE PROGRAM. USERS CAN THUS EXECUTE ASSEMBLY-LANGUAGE PROGRAMS WRITTEN BY OTHER TI-99/4A USERS, PROVIDED THE EDITOR/ASSEMBLER COMMAND MODULE IS PLUGGED INTO THE CONSOLE.

THE MINIMUM SYSTEM CONFIGURATION REQUIRED FOR USING THE EDITOR/ASSEMBLR PACKAGE CONSISTS OF A TI-99/4A CONSOLE, A MONITOR OR TELEVISION EQUIPPED WITH RF MODULATOR, THE TI MEMORY EXPANSION PERIPHERAL, AND THE TI DISK DRIVE AND CONTROLLER.

~~THE EDITOR/ASSEMBLER PACKAGE IS AVAILABLE IMMEDIATELY FOR $99.95.~~

EDITOR/ASSEMBLER
MANUAL CORRECTION
pp 342-344

The following program is a working version of the Graphics Example in section 21.7.1 of the Editor/Assembler manual. For instructions on how to run the program, see page 342 of the E/A manual.

```
 99/4 ASSEMBLER
VERSION 1.2                                                    PAGE 0001
  0001
  0002                      DEF     BUBBLE
  0003                      REF     VMBW,VMBR,VSBW
  0004                *
  0005 0000 3C7E    BBLE    DATA    >3C7E,>CFDF,>FFFF,>7E3C
       0002 CFDF
       0004 FFFF
       0006 7E3C
  0006 0008 F333    COLOR   DATA    >F333
  0007 000A   A0    BBL     BYTE    >A0
  0008 000B   A8    SPACE   BYTE    >A8
  0009 000C 01DA    LOC     DATA    >01DA,>020D,>0271,>02A5,>02D6,>02E1,>0000
       000E 020D
       0010 0271
       0012 02A5
       0014 02D6
       0016 02E1
       0018 0000
  0010 001A         MYREG   BSS     >20
  0011 003A         BUF1    BSS     >20
  0012 005A         BUF2    BSS     >20
  0013              *
  0014              *SET UP COLORS
  0015              *
  0016              BUBBLE
  0017 007A 02E0            LWPI    MYREG
       007C 001A'
  0018 007E 0200            LI      R0,>394         COLOR TABLE 20 AND 21.
       0080 0394
  0019 0082 0201            LI      R1,COLOR        LOAD COLORS >F3 AND >33.
       0084 0008'
  0020 0086 0202            LI      R2,2            TWO BYTES TO LOAD.
       0088 0002
  0021 008A 0420            BLWP    @VMBW           WRITE TO VDP RAM.
       008C 0000
  0022              *
  0023              *SET UP CHARACTER DEFINITION.
  0024              *
  0025 008E 0200            LI      R0,>D00         CHARACTER >A0 LOCATION.
       0090 0D00
  0026 0092 0201            LI      R1,BBLE         DEFINITION OF BUBBLE CHARACTER.
       0094 0000'
  0027 0096 0202            LI      R2,8            8 BYTES TO MOVE.
       0098 0008
  0028 009A 0420            BLWP    @VMBW
       009C 008C'
```

```
0029                    *
0030                    *CLEAR SCREEN
0031                    *
0032 009E 04C0          CLR   R0             START AT VDP RAM >0000.
0033 00A0 D060  LOOP1   MOVB  @SPACE,R1     MOVE SPACE CHARACTER.
     00A2 000B'
0034 00A4 0420          BLWP  @VSBW          MOVE ONE SPACE AT A TIME.
     00A6 0000
0035 00A8 0580          INC   R0             POINTS TO NEXT SCREEN LOCATION.
0036 00AA 0280          CI    R0,>300        OUT OF SCREEN?
     00AC 0300
0037 00AE 16F8          JNE   LOOP1          NO. GOTO LOOP1.
0038                    *
0039                    *PLACE BUBBLES ON THE SCREEN.
0040                    *
0041 00B0 D060          MOVB  @BBL,R1        LOAD CHARACTER CODE FOR BUBBLE.
     00B2 000A'
0042 00B4 0202          LI    R2,LOC         LOAD POINTER TO ADDRESS FOR BUBBLE.
     00B6 000C'
0043 00B8 C032  LOOP2   MOV   *R2+,R0        LOAD REAL ADDRESS.
0044 00BA C000          MOV   R0,R0          CHECK IF FINISHED LOADING.
0045 00BC 1303          JEQ   SCROLL         FINISHED. START SCROLLING SCREEN.
0046 00BE 0420          BLWP  @VSBW          WRITE BUBBLE ON THE SCREEN.
     00C0 00A6'
0047 00C2 10FA          JMP   LOOP2
0048                    *
0049                    *SCROLL SCREEN
0050                    *
0051 00C4 04C0  SCROLL  CLR   R0             BEGINNING OF SCREEN.
0052 00C6 0201          LI    R1,BUF1        CPU BUFFER1 ADDRESS.
     00C8 003A'
0053 00CA 0202          LI    R2,>20         NUMBER OF BYTES TO READ.
     00CC 0020
0054 00CE 0203          LI    R3,>20
     00D0 0020
0055 00D2 0420          BLWP  @VMBR          GET TOP LINE FROM SCREEN.
     00D4 0000
0056                    *
0057 00D6 0201          LI    R1,BUF2        CPU BUFFER2 ADDRESS.
     00D8 005A'
0058 00DA 0220  LOOP3   AI    R0,>20
     00DC 0020
0059 00DE 0280          CI    R0,>300        BOTTOM LINE OF SCREEN?
     00E0 0300
0060 00E2 1308          JEQ   PTROW1         YES. JMP TO PTROW1.
0061                    *
0062 00E4 0420          BLWP  @VMBR          READ NEXT LINE.
     00E6 00D4'
0063 00E8 6003          S     R3,R0          SUBTRACT HEX 20 FROM R0.
0064 00EA 0420          BLWP  @VMBW          WRITE TO UPPER LINE.
     00EC 009C'
0065 00EE 0220          AI    R0,>20         RETURN R0 TO NEXT LINE.
     00F0 0020
0066 00F2 10F3          JMP   LOOP3
0067                    *
```

```
0068 00F4 0201   PTROW1 LI     R1,BUF1       BUF1=TOP LINE.
     00F6 003A'
0069 00F8 0200           LI     R0,>2E0
     00FA 02E0
0070 00FC 0420           BLWP   @VMBW         TRANS TOP LINE TO BOTTOM.
     00FE 00EC'
0071                     *
0072 0100 10E1           JMP    SCROLL        KEEP SCROLLING.
0073                     *
0074                     END
0000 ERRORS
```

# Editor/Assembler Manual Corrections

| Page | Section | Description |
|---|---|---|
| 42 | 3.1.3.1 | In the first paragraph, last sentence, change "least" to "most". |
| 92 | 6.8 | In the second line of the example, change "value of ADDR" to "value in ADDR". |
| 103 | 6.14.2 | In the example, change "MOV *11,1" to "MOV *11+,1". |
| 127 | 7.20.1 | In next to last line, change ">2220" to ">C220". |
| 168 | 10.5 | In the example, change ">2A41" to "@>2A41" and change "Register 3" to Register 2". |
| 262 | 16.2.4 | Add the following note.<br>NOTE: Some devices modify the GROM read address. RS232 and TP are known offenders. If your program accesses these devices, it should save the current GROM address (see section 16.5.2) before the I/O operation and restore it (see section 16.5.1) afterwards. Otherwise the program will not be able to return to the Editor/Assembler or to Basic, or perform a BLWP @GPLLNK properly. |
| 289 | 15.2.1 | Change line 130 in the Basic program to CALL LOAD("DSK1.BSCSUP","DSK2.STRINGO"). This assumes that the source file on the next page has been entered using the editor, and saved as DSK2.STRING, and that the assembler has been run, using DSK2.STRING for a source file and producing DSK2.STRINGO as an object file. |
| 328 | 21.1 | The default for VDP Register 7 is >07 in the BASICs. |
| 335 | 21.5 | In the second paragraph, change ">00 or >04" to ">03 or >07". In the next to last paragraph, change ">00 or >04" to ">7F or >FF". |
| 415 | 24.4.8 | The second instance of GRMRD should read "GRMRA EQU >9802". |
| 416 | 24.4.8 | The second line should read "NUMREF EQU >200C". |
| 420 | 24.5 | Add the following note.<br>NOTE: A program to be saved using the SAVE utility should not have an entry point defined on the END statement. If you want to save the Tombstone City game in memory image format, you must first change the last line from "END START" to "END", and then reassemble the program. Otherwise the game starts to run as soon as it is loaded and you do not get a chance to execute the SAVE utility. |
| 465 | Index | VDP write-only Registers--add p. 267. |

Application notes:

The BSS directive is used to start a block. Blocks are used to set up areas of code that you wish to have loaded into specific memory locations; for example, to set up a reference table. The AORG directive must precede the BSS directive.

### 14.1.5 Block Ending with Symbol--BES

Syntax definition:

[<label>] b BES b <wd-exp> b [<comment>]

Example:

| | | | |
|---|---|---|---|
| BUFF2 | BES | >10 | Reserves a 16-byte buffer. If the Location Counter contains >100 when the Assembler processes this directive, BUFF2 is assigned the value >110. |

Definition

Advances the Location Counter according to the value of the well-defined expression in the operand field. If a label is included, the directive assigns the new Location Counter value to the symbol in the label field. The BES directive marks the end of a block started with the BSS directive.

### 14.1.6 Word Boundary--EVEN

Syntax definition:

[<label>] b EVEN b [<comment>]

Example:

| | | |
|---|---|---|
| WRF1 | EVEN | Assigns the Location Counter address to label WRF1 and ensures that the Location Counter contains a word boundary address. |

Definition:

Places the Location Counter on the next word boundary (even) byte address. If the Location Counter is already on a word boundary, the Location Counter is not

altered. If a label is used, the value in the Location Counter is assigned to the label before processing the directive. The operand field is not used.

Application notes:

The EVEN directive ensures that the program is at an even word boundary when a statement that consists of only a label is preceded by a TEXT or BYTE directive and is followed by a DATA directive or a machine instruction. In this case, the label does not have the same value as a label in the following instruction unless the TEXT or BYTE directive left the Location Counter on an even (word) location.

Using an EVEN directive before or after a machine instruction or a DATA directive is redundant since the Assembler automatically advances the Location Counter to an even address when it processes a machine instruction or a DATA directive.

### 14.1.7 Program Segment--PSEG

Syntax definition:

[<label>] b PSEG b [<comment>]

Example:

```
LABEL   PSEG
```

Definition:

Places a value in the Location Counter and defines successive locations as program-relocatable. If a label is used, it is assigned the value that the directive places in the Location Counter. The value placed in the Location Counter as a result of this directive is zero if no program-relocatable code has been previously assembled. Otherwise, it is the maximum value the Location Counter has attained as a result of the assembly of any preceding block of program-relocatable code.

Application notes:

The PSEG directive only repeats the default mode. If you are using another loader that also accepts the CSEG, CEND, DSEG, and DEND directives, when the PSEG directive is useful.