Tansoft Two-Pass Assembler

Introduction

The Two Pass Assembler is an application that generates 6502 processor machine code. It is a more sophisticated version of the Translator application in XBUG. With the Translator a line of ‘source’ code is entered at the keyboard and the associated object (machine) code is generated and stored in memory. Once entered the source code is lost. The assembler keeps the source code and provides facilities for it to be reviewed, edited, dumped to and retrieved from tape. The other most significant difference is that the assembler uses labels (or symbols) to identify positions in the source code or memory that can be referred to with many op code instructions. The use of labels rather than actual locations means the code can be edited and/or re-located without the need to update the defined locations. The use of labels is demonstrated in the following simple clear screen subroutine:-

```
CLEAR       LDX   #$0
            LDA   #’
LOOP        STA   $200,X
            STA   $300,X
            INX
            BNE   LOOP
            RTS
```

CLEAR and LOOP are labels. To call this subroutine using an Assembler, you would code JSR CLEAR. The LOOP label, called by the branch instruction, is associated with the STA $200,X instruction.

There are 5 assembler instructions that are not used in the XBUG Translator. These are BYT, WOR, EQU, EPZ and ORG.

BYT and WOR are ways of defining data eg

```
DATA        BYT   $A,43,’C
```

Will generate hex 0A, decimal 43 (hex 28) and character C (hex 43) in locations DATA, DATA+1, DATA+2 (wherever that turns out to be).

WOR does the same for two byte constants. It generates the LOW byte first and the HIGH byte second so that:-

```
ADDR        WOR   $1234
```

Gives hex 34 in ADDR and hex 12 in ADDR+1.

EQU, EPZ and ORG are different in that they do not cause any object code to be generated. They are strictly directives to the assembler. Consider the following CLEAR screen subroutine:-

```
SCRTOP      EQU   $200
SCRBOT      EQU   $300
            ORG   $1400
CLEAR       LDX   #$0
            LDA   #’
LOOP        STA   SCRTOP,X
            STA   SCRBOT,X
            INX
            BNE   LOOP
            RTS
            ORG   *+256
            JSR   CLEAR
```

EQU stands for equate and tells the assembler to use address $200 whenever it encounters the label SCRTOP. Similarly, use $300 for SCRBOT. EPZ (Equate Page zero) does the same thing for page 0 addresses. If you use EQU and EPZ, it is recommended to place them at the start of the source code listing to aid legibility. EPZ must be declared before its label is used anyway (there is no other restrictions on the orders of things). Note that SCRBOT could have been substituted with SCRTOP+256 or SCRTOP+$100.

The directive ORG does not get translated into any code; what it does is to set the address at which the next instruction will be stored. The directive is used at the program start in the source code listing but can be used anywhere in the listing to locate successive code in defined locations.

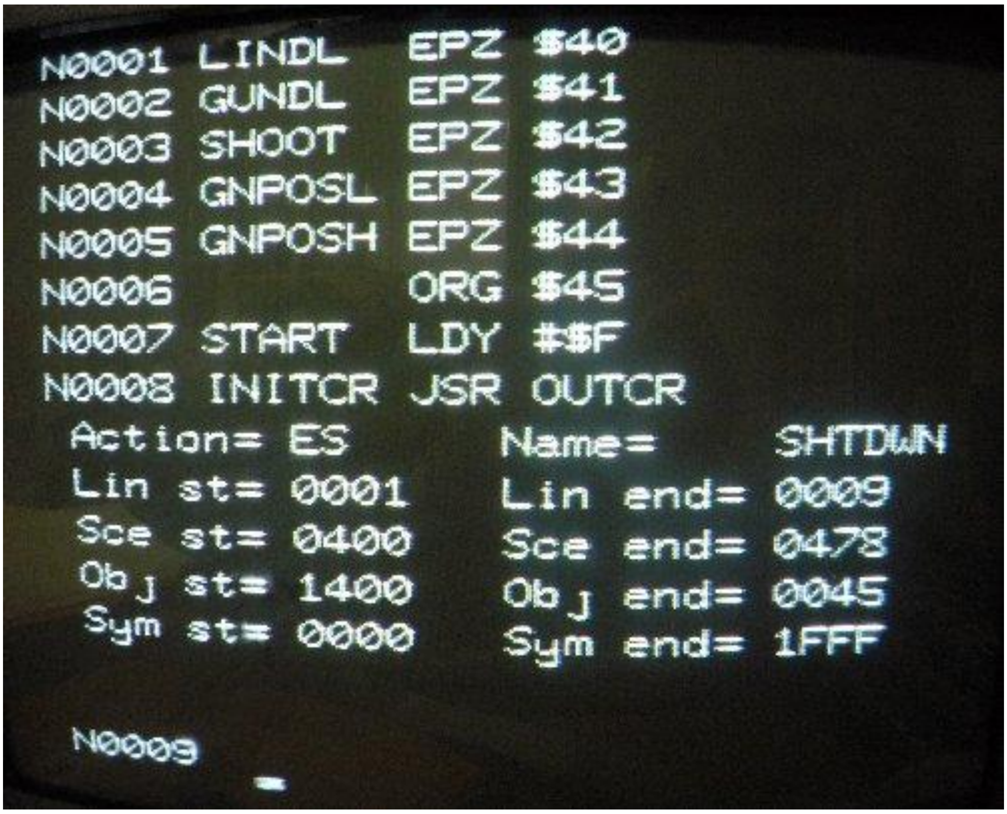
The * in the second ORG directive acts like a label and means ‘this address’. In the above example, the successive code will be placed 256 (decimal) bytes further on. * can be used with any instruction eg BEQ *+4 means branch on equal to this address plus 4 bytes.

Starting the assembler

Entry to the assembler is at $C000. On entry, the message
START=C?
is displayed. Key C <CR> for a cold start or just <CR> for a warm start. For a cold start, the message
PRINTER?
is displayed.
If you are using TANBUG V2 or TUGBUG and have a printer connected, then initialise it using <CNTRL P> or <CNTRL V> and key <CR>. If the printer is subsequently reset, it will be necessary to key <CNTL P> twice to re-initialise it.
If however you wish to use your own routine, enter its hex address followed by <CR>. It will be called when the print option is set with each character to be printed in the accumulator. See Appendix B for example code.
The printer will remain inactive until the OP (Printer) option is exercised.
The next message displayed is
BLOCK GAP?
This enables the inter block gap used when dumping source to be altered. It will usually only be necessary to key <CR> to set the default value of $F which equates to about 0.7 second. Inter block gaps are inserted when dumping to tape to provide gaps for the processing of the data to take place when assembling direct from tape. Should programs with a very large number of labels be assembled from tape, it may be necessary to increase the length of the inter-block agap.

Assembler Screen

After initialising the assembler, the assembler screen is displayed. This is divided into three areas. The top half is where the code listing is displayed. Below this are the user parameters. These remain on the screen during all assembler operations and are explained below. The bottom three lines are reserved for messages and data entry.



```
N0001 LINDL  EPZ $40
N0002 GUNDL  EPZ $41
N0003 SHOOT  EPZ $42
N0004 GNPOSL EPZ $43
N0005 GNPOSH EPZ $44
N0006        ORG $45
N0007 START  LDY #$F
N0008 INITCR JSR OUTCR
  Action= ES        Name=     SHTDWN
  Lin st= 0001      Lin end= 0009
  Sce st= 0400      Sce end= 0478
  Obj st= 1400      Obj end= 0045
  Sym st= 0000      Sym end= 1FFF


  N0009
```

User parameters
Action: Entry point for the two-character user command
Name: Entry point for stored Filename (max 6 characters)
Lin st: All source code lines are numbered. This shows the Source Code Line start number. It is initialised at N0001 by the assembler but can be amended by the user to define the start line number for assembly or file dumping.
Lin end: Shows the Line Number following the end of the current source code. It is updated by the assembler editor and the Fetch File routines. It can be amended by the user to define the start line number for assembly or dumping.
Sce st: Shows the start (hex) address of the source code stored in the memory. The default set by the assembler is $400 but can be amended by the user
Sce end: Shows the end (hex) address + 1 of the source code stored in the memory. It is maintained by the assembler.
Obj st: Shows the start (hex) address for the storage of the object (assembled) code in memory. The default set by the assembler is $1400 but can be amended by the user before assembly commences.
Obj end: Shows the end (hex) address + 1 of the object code stored in the memory. It is maintained by the assembler.
Sym st: Shows the (hex) address of the start of the symbol (label) table stored in memory. It is maintained by the assembler
Sym end: Shows the end (hex) address for the storage of the assembler's symbol table. Note the table is built backwards through the memory beginning at this location. The default set by the assembler is $1FFF but can be amended by the user.

To update the User parameters, pressing <TAB> moves the cursor from the 'Action' data entry point on the screen to each of the user-definable parameters relevant to the user command (which has to be entered first). Keying <CR> will then action the command.

Assembler Commands

The assembler commands consist of the following:

Source Code Editing
Tape operations (dump, fetch and examine source code)
Code Assembly
Other miscellaneous commands (List Labels and Exit assembler)

For some commands, additional Options are available.

Source Code Editing

ES – Edit Source Code

To enter new source code, key ES as the Action. If you wish to alter the default source start address, <TAB> to the source start parameter (Lin st) . The ES command is executed when <CR> is keyed. A flashing cursor will appear at column 1 on the bottom line. This is where all editing action takes place. The format of the editing line on the bottom of the screen is as follows:

Column	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	- - - - - - - - -
Editor	Cmd	Line No				Sp	Label (optional)						Sp	Op Code			Sp	Operand	

The cursor will normally be aligned on column 7 ready for source code entry. <TAB> will rotationally move the cursor round columns 7, 14 and 18. <BS> (Backspace) will clear the bottom line and set the cursor at column 1 ready for a new editor command. <Cntrl L> and <Cntrl R> will move the cursor left and right respectively.

Source Code Formats

The label is an optional name consisting of up to 6 characters. The first character must be alphabetic, the remainder alphabetic or numeric.

The Op Code is any of the standard 6502 op codes mnemonics plus EPZ, EQU, ORG, BYT and WOR. For standard op codes, the operand may take one of the following formats:

Address	- page 0, absolute or relative
Address,X	- page 0, absolute or relative
Address,Y	- page 0, absolute or relative
(Address),Y	- indirect Y
(Address,X)	- indirect X
#Constant	- immediately
@	- accumulator (replaces A as used in XBUG)
Null	- for other single byte ops
(Address)	- for JMP indirect

Address can consist of:
a) a label or *, either with (optionally) + or – constant
b) a constant (giving a fixed address)

Constant can consist of:

a) $hhhh	- where hhhh specifies up to 4 hex characters
b) 'c	- where c is a single ASCII character
c) nnnnn	- where nnnnn specifies up to 5 decimal numbers

Below is the complete list of the editor commands.

N <CR>	Enter new code from line 1 (or as set by the user 'Lin st' parameter).
N n <CR>	Enter new code from line number n
A n <CR>	Amend line number n
I n <CR>	Insert new code before line number n
D n <CR>	Delete line number n
S n <CR>	Show from line number n
<BS>	Clear bottom screen line and set cursor at column 1
<LF>	Exit from the editor
<TAB>	Tab cursor to columns 7, 14 and 18
<CNTRL L>	Move cursor left
<CNTRL R>	Move cursor right

Key N <CR> to enter new source code. N0001 appears where 0001 is the start line number and is automatically incremented as each line is inserted.

To enter new code at any valid line number, key N n <CR>. If this equals the end line number, then new code is added to the end. If less than the end line number, then new code will overwrite any existing code.

Commands A, I or D will show the existing line of code for the given line number as a checking aid. If <CR> is keyed at this point (with the cursor at column 6), no changes are made. To amend or insert a new line of code, key over the displayed line and key <CR>. It will be entered (if valid) as long as the cursor is past column 6. If deleting the displayed line, again move the cursor past column 6 before keying <CR>

Whenever an insertion or deletion is made, the source code lines are renumbered. So if working from a printed listing, consider working back from the end of the program so that earlier lines are not renumbered.

To display lines of code already entered, use the editor's S command (Show). Key <BS> to clear the bottom line ready for a new editor command followed by S1 for example (ie Show from line 1) then <CR>. The first 8 lines of code will be displayed on the top half of the screen. The 'S' line number on the bottom line will be incremented automatically so that subsequent lines may be displayed by simply keying <CR>.

To exit the editor and return the cursor to the user 'Action' command entry point, key <LF> (Linefeed).

Tape Operations

File facilities are provided for the dumping, examining and reading of source code. Object code should be dumped and read using the usual XBUG file handling routines. Source code should always be dumped prior to running an assembled program in case the program runs amok and corrupts the source. The source code is compacted in store and on tape by the removal of redundant spaces. The default speed for recording is fast mode. This can be set to slow (CUTS) speed or reverted back to fast speed by selecting Option S (see below).

Files are output as a series of 256 (max) byte blocks. Each file is preceded by a label block containing the name of the file, as taken from the user 'Name' parameter, and ends with an identifier block containing a string of >>>>>>>>>>>. In order to preserve user RAM space, the file routines use $200-$2FF as a buffer. Progress may therefore be observed in the top half of the screen.

DF – Dump File
The source code identified by the source start address parameter is dumped from the line start number to line end number. This will normally be the complete program but you may alter the line start and/or the line end numbers and only dump a portion of the code. There is nothing to stop you having two or more sets of source code at different RAM locations (if you are careful).

EF – Examine File
This command should be used immediately after dumping to check the recording. A tick will appear on the message line if the file is successfully found. Read or compare failures will cause message code R to be displayed on the message line. The file should then be re-examined or re-dumped. Escape from the read routines is possible by keying Control A but only if a block is currently being read (ie a file header block is not being searched for).

FF - Fetch File
This reads the file identified by the name parameter into store at the address given by the source code given by the source start parameter. The line start (=1) and line end parameters are updated. Escape is possible using Control A as above. If there are read errors, R is displayed on the message line. If failure persists, then the file is partly recoverable because only the block(s) in error will be missing. To fully recover, it will be necessary to identify the missing lines of source code and re-key them.

Code Assembly

Assembly may be performed from source code held in store or, if space is limited, from tape (dumped via the DF command). The following commands are used:

A1 – Assemble Pass 1 from store
A2 – Assemble Pass 2 from store
F1 – Assemble Pass 1 from tape
F2 – Assemble Pass 2 from tape

Before assembling from tape, it is advisable to first assemble from store as a means of fully vetting the code. Although as much vetting as possible is done by the editor during code entry, certain errors cannot be detected at this stage (eg labels not declared). If there is insufficient space in RAM to hold the object while vetting, then output of object code can be suppressed using the N option. Errors during assembly will cause assembly to terminate with the offending line displayed on the bottom line. The error should be corrected and the program re-assembled. A list of error codes is given in the appendix.

Before assembly the following user parameters must be correctly set:-

a) The source code start, line start and line end parameters although these are automatically set by the editor and Fetch File (FF) command.

b) The object start address. Assembly code will be placed here unless overridden by an ORG instruction.

c) The symbol table end address. This specifies where the assembler is to build its symbol table containing the labels and their associated addresses. This table is built BACKWARDS through store.

To access these parameters on the screen, type in the command A1 (but do not press <CR>) and use <TAB> to move around the individual parameters.

After assembly (including suppressed object code runs), the end address + 1 of the object code and the start address – 1 of the symbol table will be updated. You may therefore see if sufficient space is available to hold the object. The assembler makes no other demands on RAM space above $400.

To list the labels, use the List Label (LL) command. This will display the labels and their addresses. Options L (List) and P (Print) are also available – see below.

Options

Various options are available during assembly. Each is toggled on and off by the successive use of the appropriate command. Active options are displayed on the screen below the message line. Note that options L (List) and P (Print) are not available when assembling from tape because of the time delay imposed by keying and printing respectively.

OL – List
During Pass 2 assembly, source and object code is generated on the bottom line of the screen and scrolled to the top half of the screen. The list option causes this process to halt every 8 lines pending the keying of <LF> so that code can be examined. <CR> will switch off the option and allow the assembly to complete uninterrupted. Note that because of the limited screen width, the source appears on one line and the object on the next. The cursor character is used to generate CRs

for printing where source and object appear ion the same line. This option has the same effect when listing labels.

OM – Multi-part
This allows the consecutive assembly of several sections of source code. Its effect is to inhibit the resetting of the object assembly address on assembly. It also inhibits the clearing of the symbol table prior to Pass 1. Suppose there are 3 files of source code on tape which constitute one program. To assemble these as one object program, perform the following:
a) F1 on the first file as normal
b) Set option M and F1 on the second and third files
c) Clear option M, rewind the tape and F2 on the first file
d) Set option M and F2 on the second and third files
Assembly is now complete.

OP – Print
Causes the source and object to be printed using the Print routine set up during the cold start. This will either be the new TANBUG V2 routine or one supplied by you. It is called with each character to be printed stored in the accumulator. <CR> ($0D) indicates a new line. It is necessary for user routines to save both X and Y registers. It also causes the label list to be printed for the LL (List Label) command.

ON – No Object
This suppresses the storage of object code. It has no other effect.

OR – ROM assembly.
This allows programs to be assembled for subsequent EPROM or PROM programming. In this mode ORG will determine the object address for assembly purposes whilst the object start parameter determines the object's storage address. At the end of the assembly, the object start parameter will be updated to show the END address of the stored object whilst the object end parameter will show the assembled end address.

OS – Slow
Toggles Slow (CUTS) and Fast mode for tape recording or reading.

Miscellaneous Commands

LL – List Labels
To list the labels, use the LL (List Label) command. This will display the labels and their addresses. Options L (List) and P (Print) are also available with this command.

EX – Exit
To exit the assembler and return to the monitor (TANBUG), use the EX command.

Warm Starting
A warm start allows entry to the assembler with user parameters as previously set. However, this depends on certain locations not corrupted in the meantime. These locations are:
£40 to $ 43
$62 to $73 : start and end line numbers. Source, object and symbol table start and end addresses
$FF

Additional Notes

Interrupts are disabled when in the assembler.
Certain extra Op codes (MDF, MEM and MAC) are accepted by the assembler. This is to facilitate the eventual inclusion of macro facilities. They will be treated as WOR. Labels including the characters [\ ] are also allowed but should not be used as they will have special significance to the macro assembler.

Appendix A – Error Codes

General

✓	File on tape found (not an error condition)
C	Invalid assembler command
G	Start line number greater than end line number
N	Line number does not exist
R	Read or compare failure when reading from tape

Editor

A	Argument (line number) to command required
E	Edit command invalid
N	Line number does not exist

Assembler

A	Addressing error (eg label not declared)
C	Constant is invalid
D	Double byte operand wanted
I	Invalid operand
K	Label not allowed in label field
L	Label wanted in label field
O	Op code is invalid
R	Reconciliation failure (probably caused by failure to pre-declare a page 0 label or a symbol table corruption)
S	Single byte operand wanted
	Offset too large for branch instruction
	A page 0 address calculated (less than $100) for EQU
	A two byte address calculated (greater than $FF) for EPZ
	Label not previously declared
X	Duplicate labels

Appendix B – Printer Routines

```
0001   INIT        LDA   #$FF       1400   A9  FF        ; Initialise 6522 registers
0002               STA   $BFE2      1402   8D  E2  BF    ; set ports to output
0003               LDA   #$A0       1405   A9  A0
0004               STA   $BFEC      1407   8D  EC  BF    ; set pulse output mode
0005               LDA   #$7F       140A   A9  7F
0006               STA   $BFEE      140C   8D  EE  BF    ; disable all interrupts
0007               RTS              1407   60            ; return


0008               CMP   #$0D       1410   C9  0D        ; check carriage return?
0009               BNE   PRCHAR     1412   D0  08        ; no – skip to print character
0010               STA   $BFE0      1414   8D  E0  BF    ; yes – process it
0011               JSR   WAIT       1417   20  IF  14    ; wait printer until ready
0012               LDA   #$0A       141A   A9  0A        ; load Line Feed character
0013   PRCHAR      STA   $BFE0      141C   8D  E0  BF    ; send character to printer
0014   WAIT        LDA   $BFED      141F   AD  ED  BF    ; wait routine
0015               AND   #$10       1422   29  10        ; check control line 1 flag
0016               BEQ   WAIT       1424   F0  F9        ; printer not finished
0017               RTS              1426   60            ; return
```

Data Direction Register (DDRA / DDRB)
Set all peripheral pins to output

Peripheral Control Register (PCR)
Set Control line 1 for negative active edge
Set Pulse output mode

Interrupt Enable Register (IER)
Disable all interrupts

Interrupt Flag Register (IFR)
Check Control Line 1 Flag

I/O Socket	A1	B1	C1	D1
6522	A2	A2	B2	B2
DDRA / DDRB	$BFC3	$BFC2	$BFE3	$BFE2
Initialisation value	#$FF	#$FF	#$FF	#$FF
PCR	$BFCC	$BFCC	$BFEC	$BFEC
Initialisation value	#$0A	#$A0	#$0A	#$A0
IER	$BFCE	$BFCE	$BFEE	$BFEE
Initialisation value	#$7F	#$7F	#$7F	#$7F
IFR	$BFCD	$BFCD	$BFED	$BFED
Control line 1 flag	#$02	#$10	#$02	#$10