TANGERINE

TANGERINE
COMPUTER SYSTEMS LIMITED

FOREWORD.
CHAPTER 1 CASSETTE FIRMWARE.
CHAPTER 2 TRANSLATOR (LINE BY LINE ASSEMBLER).
CHAPTER 3 INSTRUCTION DIS-ASSEMBLER.

Copyright TANGERINE Computer Systems Ltd.
All Rights Reserved.

FOREWORD

XBUG is a firmware package containing cassette file handling routines, plus a line-by-line assembler (translator) and disassembler. The monitor TANBUG is directly linked to this package so that these facilities can be accessed directly via monitor commands. XBUG is supplied as a single 2 kilobyte 2716 EPROM. Installation is simple - merely plug the XBUG EPROM into socket G2 on TANEX, then cut LK1 on TANEX, and XBUG becomes part of the system monitor. The cassette handler and disassembler can be run from the hexadecimal keypad, however a full ASCII keyboard is required to run the line-by-line assembler.

WARNING: The translator and instruction disassembler should NOT be used in the locations Ø-5F or 1FØ-1FF as these programs will corrupt TANBUG's status locations and stack respectively.

CHAPTER 1

Cassette Firmware

This chapter should be read in conjunction with the TANEX manual, and replaces the section on dumping to and loading from cassette.

The software allows files to be dumped, verified with memory contents, and read back into memory. Files may be named with a filename consisting of up to 8 characters. Two tape speeds are available - a CUTS 3ØØ baud rate, and a TANGERINE format which approximates to 24ØØ baud. Several error checks are incorporated. The software is written such that the polarity of input data is irrelevant (your cassette recorder may invert the recorded signal).

CHOICE OF TAPE SPEED

Two tape speeds are available, a 3ØØ baud CUTS format, and an (approximate) 24ØØ baud format. The redundant information in the CUTS format is used to reduce the occurrence of tape errors should tape dropouts occur, and this format therefore gives more data protection. In general, use the CUTS rate if you are using low quality tape or cassette recorder, or of course if you are transferring data from another type of machine. Otherwise use the FAST rate.

The speed is selected by two TANBUG commands: F<CR> selects fast speed. C<CR> selects CUTS speed. Once selected, these speeds remain set until the system is powered down. They must be reset by the user on power-up. (On power-up, CUTS speed will always be selected).

SETTING UP THE CASSETTE RECORDER LEVELS

The following programs enable you to set up the recording levels for the two tape formats (they may be different for each). Automatic level recorders should also be checked to see that no errors are present.

a) Key-in the following program to record test data on cassette:-

```
1ØØ     A941       LDA #$41
1Ø2     48         PHA
1Ø3     2ØØØFØ     JSR $FØØØ
1Ø6     2Ø27FØ     JSR $FØ27
1Ø9     68         PLA
1ØA     48         PHA
1ØB     2Ø66FØ     JSR $FØ66
1ØE     68         PLA
1ØF     38         SEC
11Ø     69ØØ       ADC #$Ø
112     4CØAØ1     JMP $1ØA
```

b) Set CUTS speed: C<CR>.

c) Start the recorder in record mode.

d) Start the program by typing G1ØØ<CR>. Allow the recorder to record a few minutes of this test pattern.

e) Press RESET to escape.

Now repeat the procedure for fast speed.

To check the data:-

a) Key-in the following verification program.

```
115     2ØØØFØ     JSR $FØØØ
118     AØØØ       LDY #$Ø
11A     48         PHA
11B     2ØCBFØ     JSR $FØCB
11E     AA         TAX
11F     68         PLA
12Ø     9ØØ6       BCC $128
122     69ØØ       ADC #$Ø
124     C551       CMP $51
126     FØØ4       BEQ $12C
128     A942       LDA #$42
12A     ØØØ2       BNE $12E
12C     A947       LDA #$47
12E     99ØØØ2     STA $2ØØ,Y
131     C8         INY
132     8A         TXA
133     4C1AØ1     JMP $11A
```

b) Rewind the tape to the start.
c) Start the tape running.
d) Type G115<CR> to start the program.

Now, in the top half of the VDU screen, the characters G (good data) or B (bad data) will be printed. When the actual signal starts, all G's should appear if the level is satisfactory. It may, on manual recording level recorders, be necessary to repeat the recording procedure at different levels to obtain optimum. RESET is necessary to exit from the test program.

DUMPING TO TAPE

The "D" command is used to dump an area of memory to tape. Its format is:

D<start address>,<end address>,<filename>

The filename nay be up to 8 characters long. Characters within it may be A-Z, Ø-9, . or /.

To dump a program onto tape proceed as follows:

a) Use the "D" command but do not type <CR> yet.
b) Start tape running in record mode.
c) Hit <CR>.
   The VDU will respond with the filename being dumped, with an added appendix of .A to distinguish this file as being an absolute file.
d) The cursor will disappear and the file will be dumped.
e) The cursor will reappear when the dump is complete - the program returns to TANBUG.
f) Stop the cassette.

Example: Type D4ØØ,41Ø,FILE1 <CR> . The display will appear as follows:

D4ØØ,41Ø,FILE1
FILE1 .A

The instruction dumped locations 4ØØ to 41Ø inclusive, and called the file FILE1.

A question mark error will be generated if the command format is illegal, if the filename contains an illegal character, or if it is more than 8 characters long.

EXAMINING A TAPE

The "E" command allows you to examine a tape to see that the file has been dumped correctly, and that it can be read back. This command searches the tape for the named file, then compares what it reads with the memory content.

To examine a tape:

a) Position the cassette on a piece of blank tape (i.e. a section with no recorded signal) somewhere before the file to be examined.

b) Type E, filename <CR>.
The VDU responds with the filename.

c) Start the tape in play mode.

d) When a file is encountered, the VDU will respond with the filename and dump start address.

e) If this filename is not identical with the one specified, go back to step d).

f) If a read error is encountered, the message F(n) is printed, indicating a filename error. The program then goes to step d).

g) If the filename is identical, then the comparison will be initiated.

h) If the comparison is correct, the program will return a cursor prompt and return to TANBUG.

i) If the comparison is incorrect three types of error may occur.

M(n) - memory error - contents of tape do not agree with contents of memory. (n) is the faulty location address.

P(n) - a parity error occurred when reading the data associated with location (n).

C(n) - a checksum error occurred at the end of the file - (n) indicates the file end address.

In all cases, after printing an error, the program continues to read data. Thus if only a few errors occur, they may be checked out by reference to their addresses.

Note that in FAST speed, a VDU scroll may induce a parity error due to the time taken for a scroll. It does mean, however, that there are more errors than acceptable and the file should be re-dumped until error-free.

Example

There are two files on a tape, FILE1 and FILE2. You wish to examine FILE2, but position the tape at the blank leader.

```
E,FILE2
FILE2 .A          ;prints your filename.
FILE1 .A  ØAØØ    ;prints first filename.
FILE2 .A  Ø4ØØ    ;prints second filename.

■                 ;comparison correct.
```

Or, with one memory error:

```
E,FILE2
FILE2 .A
FILE1 .A  ØAØØ
FILE2 .A  Ø4ØØ
MØ41Ø
■
```

Location Ø41Ø is at fault.

Tape directory

The "E" command may be used to obtain a complete listing of the tape contents, by rewinding the tape to the start and looking for a non-existent filename. Note that in this case it is necesssary to exit back to TANBUG at the tape end by using the RESET key since interrupts are disabled in the cassette software. This procedure may also be necessary if the examine procedure gets very badly out of step due to a large number of errors.

FETCHING A FILE INTO MEMORY

To fetch (load) a file into memory, the "F" command is used as follows :

F,<FILENAME><CR>

For example, typing F,FILE1<CR> looks for FILE2 .A and loads it into memory. Operating procedures and errors are exactly as detailed in the section on the examine command. Should an "M" error occur, a hardware fault is indicated because the program loads the input data to the memory location and checks it immediately afterwards.

ADDITIONAL REMARKS

Do not mix tape speeds on a single cassette, otherwise filename errors will occur. Though this does not affect the operation of the search, it is not possible to list a filename directory with a single pass.

If one program uses several different areas of memory (e.g. one area for subroutines and another for main code) it is necessary either to dump the whole area in one file, thus encompassing some redundant locations, or to dump in two files.

ERROR MESSAGES

M(n) - when examining, memory location (n) contained a different value to that read from the tape.

\- when fetching, memory location (n) failed to be updated with the value read in (hardware error).

P(n) - a parity error occurred when reading the byte for location (n).

C(n) - a checksum error occurred during the tape read, (n) indicates the end of file address. Since a parity check is not infallible and will not detect two bits in error, a checksum is an additional data validity test. A checksum error will nearly always occur if a parity error occurs.
If a checksum error occurs, but no parity error,

the code must be listed and visually checked to determine where the error occurs.

F(n) - an error occurred when reading the filename (n) is meaningless.

If errors do occur, you should first try reading the tape again in case the error was due to mains borne noise. If, however, the same error persistently occurs on re-trys, the tape is likely to be in error.

READING CASSETTES DUMPED BY RAM-BASED CASSETTE SOFTWARE

XBUG software will not read files dumped by the RAM-based program described in the TANEX manual. However, you may recover these and dump them in the new absolute format by following the procedure below:

a) Key in the RAM-based cassette software (locations 5Ø to 145).

b) Dump locations 6Ø to 145 to cassette tape using the XBUG "D" command (locations 5Ø-6Ø are used by XBUG).

c) For each file to be converted use the XBUG "F" command to fetch the RAM-based software dumped in steps a) and b).

d) Enter locations 5Ø-5F by hand from the TANEX manual.

e) Read in the file in old format by following the instructions in the TANEX manual.

f) Dump the file using XBUG's "D" command.

Repeat steps (c-f) for each file.

CHAPTER 2

Translator

The translator program allows you to enter programs in 65Ø2 mnemonic assembly language. The translator verifies that the entered instruction is legal, and if so transfers it into machine code and puts it into memory. The user program counter is automatically incremented by the correct amount for the instruction entered.

To enter the translator, type "T", followed by the address at which code is to be entered, followed by a carriage return. XBUG prints the current program counter, and a special prompt "!" to indicate that a sub-program is being used.

Example

Type T4ØØ<CR>; the display will then be as follows:

```
T4ØØ
Ø4ØØ  DD      !
```

The exclamation mark indicates that XBUG is ready to accept input.

To enter code to be translated, type in the instruction followed by carriage return. If the instruction is legal, it is translated, the machine code equivalent displayed, and the program counter updated. If it is illegal a "?" is displayed, and the program counter remains unaltered. For example if the user inputs the INX instruction the display will appear as below:

```
T4ØØ
Ø4ØØ  E8      INX             h
Ø4Ø1  DD      !
```

On the completed line, the display format is:

(Program counter) (Opcode) (Mnemonic) (ASCII equivalent of opcode)

For example, user next types LDA $FE73<CR>.

```
T4ØØ
Ø4ØØ   E8       INX              h
Ø4Ø1   AD73FE   LDA $FE73         -s∿
Ø4Ø4   DD       !
```

This time, the entered instruction was a multiple byte instruction, so three bytes of machine code were generated, three bytes of ASCII equivalents printed, and the users program counter incremented by 3. Note that the translator does not allow labels to be used - all addresses must be entered as absolute values. A complete list of 65Ø2 instructions and the modes in which they are legal appears in the Mictotan 65 manual on pages 5-9 to 5-35. Instructions must be in the following format:

<OPCODE><CARRIAGE RETURN>

for implied instructions, and for all others:

<OPCODE><SPACE><OPERAND><CARRIAGE RETURN>

Where an operand contains a numeric value (that is, all modes except implied and accumulator), the numeric part of the operand must be either a hex value preceeded by a $ symbol, or a single ASCII value preceeded by a '. The hexadecimal operand must always contain at least one hexadecimal character, and never more than 4 (two for zero page, immediate, (Indirect,X) and (Indirect),Y . The following table shows the required operand format for each mode. H indicates a hexadecimal digit; A an ASCII digit.

Mode	Operand Format		
Immediate	#$HH	or	#'A
Zero Page	$HH	or	'A
Zero Page,X	$HH,X	or	'A,X
Zero Page,Y	$HH,Y	or	'A,Y
Absolute	$HHHH	or	'A
Absolute,X	$HHHH,X	or	'A,X
Absolute,Y	$HHHH,Y	or	'A,Y
Relative	$HHHH	or	'A
(Indirect,X)	($HH,X)	or	('A,X)
(Indirect),Y	($HH),Y	or	('A),Y

(Indirect) ($HHHH) or ('A)

Note that though four hexadecimal digits are shown, leading zeros need not be typed, for example typing JMP ($5)<CR> is legal, and equivalent to JMP ($ØØØ5)<CR>.

Examples of legal instructions:

```
LSR A          ;shift acc left
LSR $A         ;shift contents of location A left
LDA #'A        ;load acc with 41 Hex (ASCII A)
LDA $FF        ;load acc with contents of FF
STA $5Ø        ;2-byte zero page instruction
STA $55Ø       ;3-byte absolute instruction
LDA ($5Ø),Y    ;Indirect,Y
LDA $5         ;load acc with contents of location 5
INY            ;implied
BNE $A         ;jump to location A if test true
BNE $FØØØ      ;jump to location FØØØ if test true
```

Examples of illegal instructions:

```
LSR BB         ;no $ before Hex opcode
LDA #'AA       ;more than 1 ASCII argument
LDA #FF        ;no dollars in front of argument
LDA #$FFF      ;more than 2 characters in immediate mode
LDA ($5ØØ),Y   ;argument is not zero page
LDA $1ØØØ1     ;more than 4 characters in abs mode operand
INL A          ;illegal opcode
LDA A          ;illegal operand for this opcode
```

Relative mode instructions (that is, branches) must always contain a hexadecimal address as their argument. If the branch is within range, the correct offset is calculated and entered in memory, otherwise an error is displayed.

An error (?) is displayed if any of the following circumstances occur:

Illegal instruction/operand format.
Branch out of range.

Invalid opcode.
Invalid operand for opcode used.

Constants may also be entered directly into memory, for example to form tables. These constants may be entered in either Hex $HH or ASCII 'A format, and must be typed immediately after the ! prompt. For example, the user types $AA <CR> ; XBUG responds:

```
Ø41Ø   AA      $AA
Ø411   DD      !
```

User then types 'A<CR> and display becomes:

```
Ø41Ø   AA      $AA
Ø411   41      'A
Ø412   DD      !
```

An error print will occur if the format is illegal.

Comments may be entered by typing a ";" followed by the comment, and are ignored by the translator. For example if the user types ;THIS IS A COMMENT<CR> the display will be:

```
Ø412   DD      ;THIS IS A COMMENT
Ø412   DD      !
```

Comments are not allowed on the same line as instruction/opcode entry.

A memory error printout of the form M(address) is always displayed if you try and assemble a program into non-existent memory or into memory occupied by ROM. This message also occurs if RAM is faulty and the data is not written correctly.

Three further command in XBUG's translator allow you to manipulate the user program counter either to skip over locations which you do not wish to change, or to step backwards should you wish to correct an error.

The * = command allows you to set the program counter to any desired location. For example, to change the program counter

to 5ØØ the user must type * = $5ØØ<CR> and the resulting display will be:

```
Ø412  DD        *=$5ØØ
Ø5ØØ  FF        !
```

indicating that the translator is ready to accept input of location 5ØØ.

The ↑ (up-arrow) single key command decrements the program counter by 1 each time it is depressed. If you have made a mistake and wish to correct it, you can easily step back to the required location.

The LF (line feed) single key command increments the program counter by 1 each time it is depressed. For example, suppose the following code has been entered:

```
Ø41Ø   A9FF     LDA #$FF
Ø412   855Ø     STA $5Ø
Ø414   DD       !
```

and you wish to change the LDA #$FF instruction to AND #$FE, hit the ↑ key 4 times, XBUG will display:

```
Ø41Ø   A9FF     LDA #$FF
Ø412   855Ø     STA $5Ø
Ø414   DD
Ø413   5Ø
Ø412   85
Ø411   FF
Ø41Ø   A9       !
```

Now type in the new instruction and XBUG will display:

```
Ø412   85
Ø411   FF
Ø41Ø   29FE     AND #$FE
Ø412   85       !
```

Now hit LF twice to increment the program counter back to

location Ø414.

The ESC key, when depressed, causes an exit from XBUG's translator, and returns to TANBUG.

Example Program

The following program runs through the ASCII character set, displaying each character on the screen, and resides at memory location ØØØØ. First, enter the translator by typing TØØØ<CR>. Enter the following lines of code:

```
LDA #$D
PHA
JSR $FE75
PLA
SEC
ADC #$Ø
LDX #$FF
LDY #$FF
DEY
BNE $ØØE
DEX
BNE $ØØE
BEQ $ØØ2
```

The complete program with its addresses and assembled code is now displayed on the screen. Exit from the translator by typing ESC. Start the program by typing GØØØ<CR>. The ASCII character set will now gradually be built up on the screen.

CHAPTER 3

Dis-Assembler

The instruction dis-assembler performs the converse function to that performed by the translator, that is it reads machine code locations stored in memory and displays them as mnemonic instructions and opcodes.

To enter the Instruction Dis-assembler, type:

I<ADDRESS><CR>

For example, I4ØØ <CR> dis-assembles the program given as an example in chapter 2. When the VDU screen is full, the dis-assembler gives you a ! prompt. You may respond to this with either <CR>, <LF> or <ESC>. Escape exits from XBUG and returns to TANBUG. Carriage return displays the next VDU page of data and pauses with a prompt at the end. Line feed displays the next VDU page of data and inhibits the pause at the end of the page, so the dis-assembler continues to operate until the BREAK or RESET key is depressed.

There are some circumstances where the dis-assembler encounters stored numbers not corresponding to a legal opcode, for example in RAM which contains random data generated on power-up, where programs contain data tables, or where the dis-assembler has been instructed to begin from the operand part of an instruction rather than the opcode part. In this case a single byte is printed, followed by a ? to indicate an illegal opcode. Sometimes, data bytes may be translated as legal opcodes, thus dis-assembling the first few bytes of a program incorrectly. For example, the example program previously discussed dis-assembles satisfactority if the command I4ØØ is used. However, if I4Ø1 is used, the data at location 4Ø1 is treated as an opcode. Because this is in fact the opcode for ORA, the program dis-assembles as:

```
Ø4Ø1   ØD482Ø   ORA $2Ø48
Ø4Ø4   75FE     ADC $ØØFE,X
Ø4Ø6   68       PLA
```

Thus location 4Ø6 is reached before the dis-assembler gets in step and interprets the code correctly. Care should always be taken to start the dis-assembler at a known instruction address.

TANGERINE
COMPUTER SYSTEMS LIMITED

Forehill Works Forehill Ely Cambs England