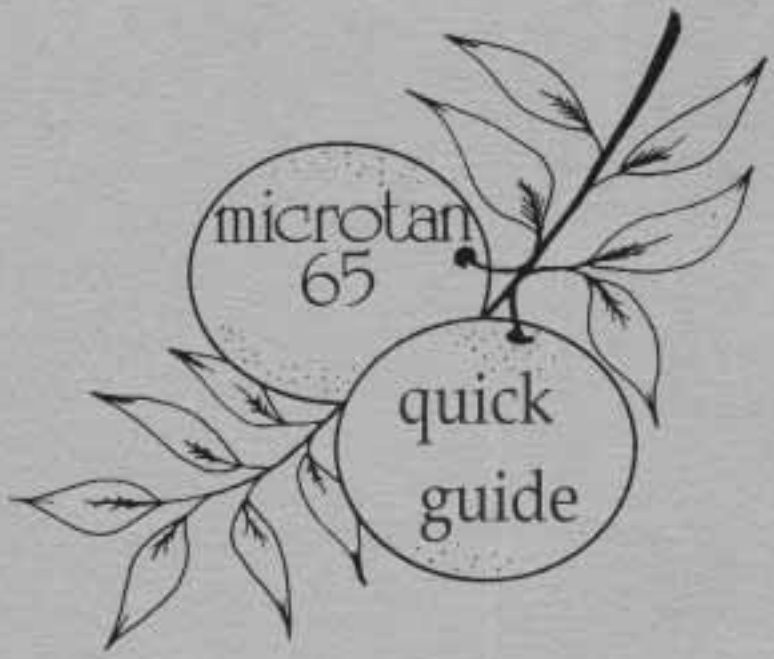
TANGERINE



TANGERINE
COMPUTER SYSTEMS LIMITED

Version 1.0 – 6th June 2000

Written by Geoff Macdonald

Memory Map

0000-003f	System variables
0040-00ff	Unused (except by BASIC)
0100-01ff	Processor Stack
0200-03ff	Display
0400-AFFF	RAM
BC00-BC01	1st AY8912 sound chip
BC02-BC03	2nd AY8912 sound chip
BC04	Space Invasion sound
BFC0-BFCF	1st 6522 VIA
BFD0-BFD3	Serial I/O (not implemented in emulator)
BFE0-BFEF	2nd 6522 VIA
BFF0	Read: Chunky graphics on Write: Reset keyboard interrupt flag
BFF1	Write: Start delayed NMI
BFF2	Write: Write hex. keypad column
BFF3	Read: Read ASCII keyboard last key/hex. keypad row Write: Chunky graphics off
C000-E7FF	BASIC interpreter ROM
F000-F7FF	XBUG ROM
F800-FFFF	TANBUG ROM

System Variables

0000	Used by breakpoints
0001	Last ASCII keyboard character
0002	Temporary character store
0003	Display index
0004-0006	Fast interrupt link
0007-0009	NMI link
000A-000B	Cursor index
000C	Zero if in user program
000D	nonzero if in single instruction
000E	Proceed counter
000F	Hex/ASCII keyboard
0010-0012	Slow interrupt link
0013-0014	Used by hexpack rountine
0015-0016	Psudo PC
0017	Psudo PSW
0018	Psudo SP
0019	Psudo IX
001A	Psudo IY
001B	Psude A
001C-001D	Temporary store
001E-001F	Copy store
0020-002F	Breakpoint addresses
0030-003F	Breakpoint code store

Display Format

The display is made up of 16 rows of 32 characters, in a contiguous block starting at address 0200. Characters are sequential, the first line beginning at address 0200, the next at 0220, and so on:

0200	0201	...	021E	021F
0220	0221	...	023E	023F
		...		
03C0	03C1	...	03DE	03DF
03E0	03E1	...	03FE	03FF

A single character cell may be either an ASCII character or a "chunky graphic" character. Reading from address BFF0 causes subsequent writes to the display to appear as chunky graphics. Writing to BFF3 causes subsequent characters to appear as ASCII characters. It is not possible to determine whether a particular character is being displayed as an ASCII character or a chunky character.

Character Sets

ASCII Characters

Characters 80-FF are repeats of characters 00-7f

Chunky Graphics

Chunky graphics characters are made of a 2x4 block. A pixel within the block is set if the corresponding bit of the character code is a "1". The pixels are arranged as follows:

0	1
2	3
4	5
6	7

00-0F

10-1F

20-2F

30-3F

40-4F

50-5F

60-6F

70-7F

80-8F

90-9F

A0-AF

B0-BF

C0-CF

D0-DF

E0-EF

F0-FF

Monitor Subroutines

FDFA **POLLKB**
Waits for the user to press a key, then returns. The ASCII code for the key pressed is stored at address 0001

FE73 **OUTPCR**
Outputs a CR to the display

FE75 **OPCHR**
Outputs the character in the accumulator to the display

FF0B **HEXPNT**
Outputs the accumulator to the display as a pair of hex digits

FF28 **HEXPCK**
Reads hex characters from the current cursor line and converts them into two 8-bit binary values stored in addresses 0013 and 0014. Set IY to the offset of the first character to convert (0=start of line). Conversion stops when a non-hex character is found. On exit, Z is clear if the terminating character was the cursor and V is set if there were one or more characters converted.

Monitor Commands

All commands and data must be typed in upper case. If you type anything incorrectly, TANBUG will display a "?" at the end of the line.

M - Memory examine/modify

Displays the content of a specified memory location and allows you to change it.
Command format:

M<ADDRESS>

Where **ADDRESS** is the address if the memory to display/change.
The current content is displayed after the address. If you want to change it, type the new value. Pressing *ENTER* will store the new value (if there is one) and exit. Pressing *Ctrl-ENTER* (this was a single key on the Microtan keyboard - *LF*) stores the new value and opens up the next location. *ESC* stores the new value and opens up the previous location.

L - List memory

Displays the contents of a section of memory.
Command format:

L<ADDRESS>,<NUMBER OF LINES>

Where **ADDRESS** is the first address to be displayed and **NUMBER OF LINES** is the number of eight bytes lines to display.
Each line displayed comprises the address of the first byte on the line followed by eight bytes of data.

G - Go

Starts execution of a program.
Command firmst:

G<ADDRESS>

Where **ADDRESS** is the address of the program start. The program will execute until either a *BRK* instruction is executed, or the Microtan is reset.

R - CPU register display/modify

Memory locations 0015-001B are used to hold the contents of the CPU registers. The CPU registers are loaded from these locations when you execute a program with the *G* command, and are stored there when a *BRK* instruction is executed, prior to the system returning to TANBUG. The *R* command simply performs a *M0015* command to allow you to display and modify the CPU registers.

0015	Program Counter (PC) low byte
0016	Program Counter (PC) high byte
0017	Processor status word (PSW)
0018	Stack Pointer (SP)
0019	Index X (IX)
001A	Index Y (IY)
001B	Accumulator (A)

S - Enable single instruction mode

When single instruction mode is enabled, your program will execute one instruction at a time. The CPU registers will be displayed after each instruction.

N - Normal mode (disable single instruction mode)

This mode is also automatically set when the CPU is reset.

P - Proceed

Executes the next instruction. If you follow the *P* command with a number, that number of instructions will be executed.

B - Set/clear breakpoint

Command format:

B<ADDRESS>,<BREAKPOINT NUMBER>

When **ADDRESS** is the address at which to set the breakpoint. **BREAKPOINT NUMBER** is from 0 to 7 and is the ID number of the breakpoint. To clear a breakpoint, set its address to zero. Used on its own, the *B* command will clear all breakpoints.

NOTE: This command works by replacing the instructions at the breakpoint addresses with *BRK* instructions when you execute your program. When your program hits a *BRK* and returns to TANBUG, all the breakpoint *BRK* instructions are replaced by their original values. So:

1. A breakpoint should only be set at the op-code part of your instruction.
2. If breakpoints are set and the CPU is reset, the breakpoints will be left as *BRK* instructions.
3. Setting more than one breakpoint at the same address will cause a *BRK* instruction to be left at that address.
4. if your program is self-modifying and it changes an instruction

where a breakpoint has been set, the breakpoint will not occur and the original value restored if the program exits because of a *BRK*.

O - Calculate branch offset

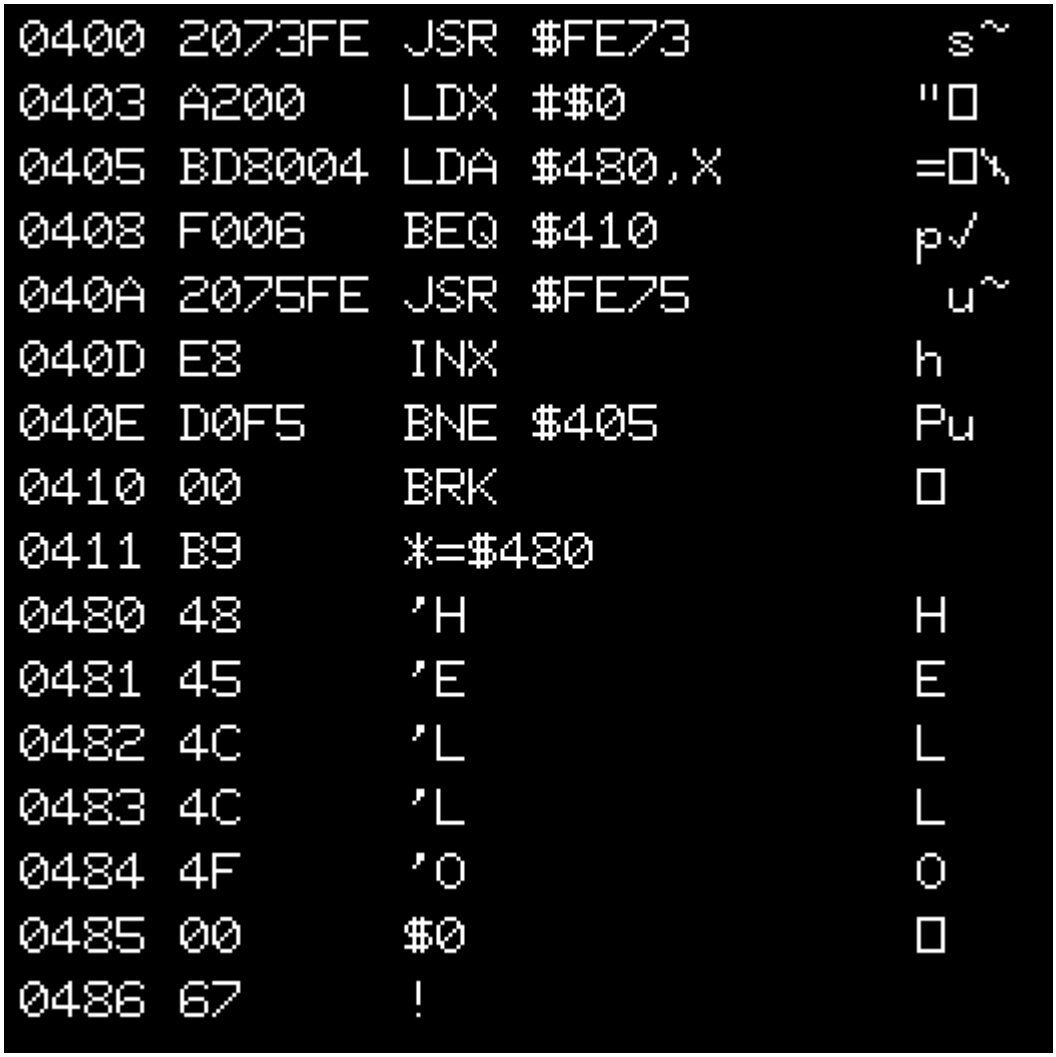
Calculates the offset required for a branch instruction
Command format:
O<BRANCH OPCODE ADDRESS>,<BRANCH DESTINATION ADDRESS>
Where **BRANCH OPCODE ADDRESS** is the address of the opcode of the branch instruction and **BRANCH DESTINATION ADDRESS** is the address where the branch is to jump to.

C - Copy memory

Copies a block of memory.
Command format:
C<SOURCE START ADDRESS>,<SOURCE END ADDRESS>,<DESTINATION START ADDRESS>
Where **SOURCE START ADDRESS** is the start address of the source block, **SOURCE END ADDRESS** is the end address of the source bock (this address is included in the copy) and **DESTINATION START ADDRESS** is the start address of the destination.
Note that this command always copies from the start to the end and, so if the destination start address is within the source block, the block will be corrupted.

T - Translate assembler to machine code

Begin using the single line assembler.
Command format:
T<ADDRESS>
Where **ADDRESS** is the address at which to begin assembling. The display will show the address followed by the byte currently stored at this address and the input cursor (which has changed to an exclamation mark). You may now enter a line of 6502 assembler, followed by *ENTER*.
Each assembler line consists of a three letter mnemonic and, if there is an operand, a space followed by the operand. All letters must be in upper case, and hexadecimal values must be preceded by a dollar "$". You can enter a single character as operand data by preceding the character with an apostrophe **'.** Labels cannot be used. The immediate operator is a "#".
When you have entered a valid line of assembler, the machine code will be shown after the address, and the ASCII equivalent on the right. The address will automatically increment.
If you enter an invalid line, a question mark will be shown and the address will not change.
Pressing caret "^", will cause the address to decrement by one and *Ctrl-ENTER* causes the address to increment by one.
You may change the address by entering ***=$<ADDRESS>**
Data may be directly entered by typing **$<HEX BYTE>** or **'<CHARACTER>**
When you have finished, press *ESC*.
An example is shown below:



I - Interpret (disassemble) machine code as assembler

Disassemble a section of memory.
Command format:

I<ADDRESS>

Where **ADDRESS** is the address at which to begin disassembling.
The display will show fifteen lines of disassembly and stop. You may new press:

ENTER - display the next fifteen lines
ESC - return to TANBUG
Ctrl-ENTER - display continuous disassembly until the Microtan is reset.

BAS - Start BASIC interpreter

This starts the BASIC interpreter. You can also type *GE2ED*, which does the same thing.