ROGER CULLIS

Within the BBC MICROCOMPUTER

A reference manual
for Assembly Language programmers

Iosco

ROGER CULLIS

Within the BBC MICROCOMPUTER

A reference manual for Assembly Language programmers

This book contains essential reference material for serious users of the BBC microcomputer. It includes descriptions and explanations of the principal ROM routines, memory maps, tables of RAM usage, ROM routine entry points, zero page locations and JMP/JSR and lookup reference origins. It covers OS 1.2, Basic 1, Basic 2, HiBasic, DFS 0.90, NFS 3.34, 6502 second processor OS 1.1 and DNFS and Econet TUBE communications.

Published by
Losco Limited
PO Box 4, Cranleigh, Surrey GU6 8BQ,
England

©1985 Losco Limited
ISBN 0 948203 00 5

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of the Publisher.

Whilst every effort has been made to ensure that the information in this book is accurate, neither the Publishers nor the author will accept any liability arising from or consequential upon any error or omission.

Typeset from word processor data
by Budget Typesetting 01-658 8754
Printed and bound in Great Britain by
Biddles Ltd, Guildford and King's Lynn

Copyright

This book contains a description of the operation of the BBC microcomputer and tables of where the controlling routines are to be found. It is intended to aid users to write programs which interact fully with the built-in software. The reader should, however, be aware that the source code and object code of ROM routines are the subject of copyright and may not be used without the copyright owner's permission. This means that the programmer cannot extract routines for his own software, although he may freely call them from programs running on the computer.

Within this book the terms Tube, Econet and Acorn are trade marks of Acorn Computers Limited and the term BBC is a trade mark of the British Broadcasting Corporation. All references to the BBC microcomputer refer to the computer produced by Acorn Computers Limited in association with the British Broadcasting Corporation's Computer Literacy Project.

Contents

Introduction

In order to write software which makes full use of the facilities provided by a computer, it is of vital importance to have an understanding of the computer's components and how they are controlled. The BBC microcomputer is a complex machine, which not only drives a wide variety of peripheral devices through built-in interfaces, but can also be connected to an increasing number of second processors to enhance its computing power. In many cases these resources can only be exploited by direct access to the utilities and routines which are an integral part of the languages and operating system provided in the machine's firmware.

The BBC microcomputer is built round the Mostek 6502 8-bit microprocessor. Although, in its standard form, the computer has 32K of ROM and either 16K (Model A) or 32K (Model B) of RAM, it was designed as an expandible system and is therefore primarily an input/output (I/O) device. It can service a variety of peripherals such as serial and parallel printers, local area networks, disc drives and speech synthesisors, but, in order to do this, blocks of memory have to be reserved for specific applications. For example, in order to provide buffers for transfer of data between the computer and the floppy disc controller, nearly 3K of read/write memory is required, adding Econet takes another 0.5K and speech synthesis, a further 1K. The 6502 microprocessor has a 16-bit address bus, which means that it can access 64K (2 ^16) banks of memory. Of this, 32K is ROM and memory-mapped I/O and 32K is RAM (16K on the Model A). RAM occupies the lower 32K addresses from 0000-7FFF (0-32,767 decimal) and ROM occupies two 16K blocks from 8000-BFFF and C000- FFFF. The lower ROM block is allocated to utilities, including the Basic language, the disc filing system, word processors such as View, or graphics extensions. Up to sixteen different 16K ROMs can be connected to this address space, although only one may be active at a given time. The machine operating system (MOS) or a control program ensures that the appropriate paged ROM is switched in when it is required. The upper ROM block (C000-FFFF) is occupied by the operating system which controls the running of the computer. Most of the memory space contains operating code, but part of the address allocation is devoted to memory-mapped input/output. If an instruction is written to one of these addresses, it is not stored, as it would be if it were RAM, but passed directly to the peripheral device to which the address is allocated.

The MOS, the paged ROMs and I/O devices all require read/write memory for storage. This memory may be set aside exclusively for this purpose, such as the soft-key buffer (0B00- 0BFF) which holds the strings called up by the user-definable keys, it may be shared amongst several chips, such as the paged ROMs' public workspace or the filing system Zero Page locations, or it may be allocated

dynamically during operation. An example of this latter is the Basic stack which acts as a temporary store and expands and contracts according to the needs of the program.

The screen buffer, which is where the computer stores what is currently displayed on the screen, occupies the highest region of RAM. Its size depends on the mode which is being displayed. Teletext mode (Mode 7) occupies only 1 K, whilst Mode 0 requires 20K.

Page 0 (addresses 0000-00FF) is used for fast-access dynamic storage and Page 1 (0100-01FF) is allocated to the 6502 stack. The Operating System takes Pages 2 and 3, whilst Pages 4 to 7 are given to the current language ROM or to interfacing routines if a second processor is active. Pages 8 to C (0800-0CFF) are devoted to Operating System buffers for input and output and for storage of soft key definitions and sound envelopes. Page D (0D00-0DFF) contains the non-maskable interrupt routines and parameters such as the current paged ROM status and extended vectors. Some Paged ROMs require additional workspace which is allocated on RESET. Public or static workspace is shared and extends from E000 in integral pages. Each Paged ROM has a specific requirement and the allocation is equal to the greatest of the individual needs. Above this, each Paged ROM may reserve further pages of RAM for its exclusive use. This is designated as private workspace and is used mainly for parameter storage whilst the ROM is dormant.

Computing power can be enhanced by the addition of a second processor. In this configuration, the host processor is dedicated to input/output operations. Communication between the two processors takes place via the Tube, a bidirectional, asynchronous bus, clocked at 2MHz.

The second processor has only a rudimentary operating system, since it is required solely to transfer parameters to and from the host processor and does not have to prepare data for the peripheral controller chips.

This book is arranged in a number of sections, each one dealing with a separate ROM and its associated routines. As far as possible, each section has been made self-contained in order to minimise cross-referencing. Although, there is a certain amount of duplication, it was felt that greater utility would result from this approach.

The data on each ROM follows a common layout. A title page summarises the principal features, including the memory map, and is followed by a chart of Zero Page usage. Next there is a description of the main routines within the ROM accompanied by a gazetteer of their entry points. A summary is then given of the areas of RAM used by the ROM routines. The section is concluded by reference tables giving the sources of subroutine calls, unconditional jumps and lookup references. In the case of the Basic ROMS, there is a common descriptive section, with the entry points for each ROM listed

at the beginning of each section. Except where specifically mentioned, all numeric information, including addresses, is in hexadecimal format.

ROM type :operating system

Binary version number:not applicable

Copyright offset :not applicable

Title string :OS 1.2

Assembly address :C000-FFFF

Relocation address :not applicable

Public workspace :not applicable

Private workspace :not applicable

OS 1.2 - Memory Map

C000-C4BF :VDU display setup parameters
C4C0-D93F :VDU routines
D940-D9CC :Page 2 setup - default values
D9CD-DC67 :RESET and BRK handling
DC68-DE8B :IRQ handling
DE8C-DEA8 :analogue-to-digital converter
DEA9-DEBA :ROM embedded message printing
DEBB-DF0B :character input
DF0C-E3A7 :* command handler and printer output
E3A8-E434 :user-defined key processing
E435-E582 :buffer housekeeping
E5B3-E656 :OSBYTE and OSWORD entry point vectors
E657-EB02 :miscellaneous OSBYTE and OSWORD routines
EB03-EB12 :sound processing
EE13-EED9 :speech processing
EEDA-F134 :keyboard input and housekeeping
F135-FBFF :CFS/RFS routines
FC00-FCFF :FRED - 1MHz Bus memory-mapped input/output
FD00-FDFF :JIM - 1MHz Bus memory expansion page
FE00-FEFF :SHEILA - MOS memory-mapped input/output
FF00-FFA5 :BRK and IRQ handling routines
FFA6-FFF9 :miscellaneous MOS call entry points
FFFA-FFFF :6502 RESET and interrupt vectors

OS 1.2 - Zero Page usage

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	*	*	.	.	.	.	.	.	.	.	.	.	.	.	.	.
10	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
20	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
30	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
40	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
50	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
60	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
70	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
80	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
90	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
A0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
B0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
C0	*	*	*	*	*	.	*	*	*	*	*	*	*	*	.	.
D0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
E0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
F0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

OS 1.2 - description

The Operating System ROM contains the routines to start up the computer when it is switched on or reset by pressing the break key. It also supervises the chips which control the peripheral devices and handles the flow of data to and from them.

The BBC microcomputer can have a variety of different filing systems - e.g. disc, Econet, Telesoft - for data storage and retrieval. Mostly, these are in the form of separate paged ROMs, but the two simplest - the ROM filing system (RFS), which can be used for data retrieval only, and the cassette filing system (CFS) which only implements a limited subset of the full filing system operations - are built into the MOS ROM.

Peripheral controller chips

The computer's ability to control a wide variety of peripheral devices stems from its incorporation of dedicated peripheral controller chips. For instance, disc storage is controlled by an Intel 8271 floppy disc controller chip, Econet by a Motorola 6854 advanced data link controller and the VDU by a Motorola 6845 cathode ray tube controller. These chips each have several registers which need to be initialised and, in addition, expect instructions in a predefined format. The lookup tables and control routines to meet these needs are stored in the operating system ROM or, in the case of filing systems, in a separate paged ROM.

6522 versatile interface adapter (VIA)

The main control of peripheral devices is performed by two 6522 versatile interface adapters. One of these controls internal functions such as sound and speech generation and keyboard access and the other deals with flow of data to printers and the user port. Each VIA has two 8-bit I/O ports (Port A and Port B) on which each pin can individually be selected as an input or an output, four status and control lines (two for each port), two 16-bit counter/timers which can be used to generate or count pulses, an 8-bit shift register to perform serial/parallel data conversion and interrupt generators to control the 6502 processor. Each VIA occupies sixteen addresses (FE40-FE4F for the MOS VIA and FE60-FE6F for the user VIA). Four registers control its operation. Data direction registers A and B (DDRA, DDRB) determine whether the pins on Ports A and B are configured as outputs or inputs. The peripheral control register (PCR) determines whether the rising edge or falling edge is recognised on the input status lines and how the other status lines operate and the auxiliary control register (ACR) determines whether the data ports are latched and how the timers and shift register operate.

SN76489AN sound generator

The sound generator comprises three programmable tone generators, a programmable noise generator, a clock scaler, individual attenuators for each generator and an audio mixer output buffer stage. A parallel 8-bit input port receives signals over the MOS VIA slow databus from the 6502 to control the audio output. Each tone generator consists of a frequency synthesis section and an attenuation section. The noise generator comprises a noise source and an attenuator. Ten bits of information are required to define the desired output frequency and four bits to select the attenuation. a frequency update requires a double byte transfer, whilst an attenuator update requires a single byte transfer. If no other control registers on the chip are accessed, a tone generator may be controlled by initially sending both bytes of frequency and attenuator

data, followed by the second byte of data only for succeeding values. This allows the six most significant bits to be quickly modified for frequency sweeps. The sound generator has eight internal registers to control the three tone generators and the noise source. During all data transfers, the first byte contains a three-bit field which determines the destination control register.

TMS 5220 speech generator

The speech processor has four registers, a control register which receives commands over the slow data bus, a 128-bit FIFO buffer to hold speech data when under 6502 processor control (usually from RAM), a serial-in, parallel-out data register to transfer data from the FIFO buffer or a speech data ROM and a status register. Access to the speech processor is by means of OSBYTE calls 9E (read) and 9F (write)

µPD7002 analogue-to-digital converter

The ADC chip has four analogue input channels and four internal registers. Selection of channel and resolution (8/12- bit) and start of conversion is determined by the control register (Sheila address FEC0). A status register (read only) is also located at this address. The result of a conversion is held in two data registers at addresses FEC1-FEC2.

6845 cathode ray tube controller

The 6845 cathode ray tube controller coordinates the flow of data from screen memory to character generator logic and thence on to the monitor. It possesses eighteen internal registers which establish operating parameters for the device. Registers R0 to R3 establish the horizontal format and timing, R4 to R9 control vertical format and timing, R10 and R11 hold the cursor start and stop scan lines, R12-R13 is the top of page address, R14-R15 is the cursor position, and R16-R17 holds the light pen position. The desired parameter register is selected by means of an address register AR. Access is achieved by writing the parameter register number to Sheila address FE00. A subsequent read or write to Sheila address FE01 then communicates directly with the parameter register.

Registers R0-R11 are set up on reset and the remainder are accessed dynamically. The CRTC provides the interlaced and non-interlaced modes and altered vertical position which are set by the OSBYTE 90 (*TV, *FX 144) command.

The cursor signal is generated by the CRTC whenever the screen memory address is equal to the address stored in the 6845 cursor position register. A blinking cursor is formed by not turning on the signal during some of the memory refresh cycles. If registers R10 and R11 contain the same value, the cursor will occupy a single scan line, whilst if they contain different values, a block cursor will be

formed. Bits 5 and 6 of the cursor start register determine if a blinking cursor is displayed.

C000-C4BF - VDU display setup parameters and routines
The first part of the MOS is devoted to lookup tables of parameters required to set up and run the visual display.

C000-C0BF - Character font lookup table
Text characters are displayed on an 8x8 matrix. The font for each character is stored (in ascending ASCII order) as an eight- byte block, each byte representing one line of the matrix. In each byte, 1-bits represent dots that are displayed. For example, the symbol '+' (ASCII code 2B), held in ROM at C058-C05F, is displayed as follows.

L/U table address	byte	binary equiv	visual display
C058	00	00000000	
C059	18	00011000	**
C05A	18	00011000	**
C05B	7E	01111110	******
C05C	18	00011000	**
C05D	18	00011000	**
C05E	00	00000000	
C05F	00	00000000	

C300
Unconditional jump to VDU initialisation routines

C303
BRK embedded messages

C31F-C332 - Byte mask lookup tables
Used in processing pixels for graphics display

C333-C353 - VDU entry point lo
C354-C374 - VDU entry point hi

The address of the entry point of the VDU routines, indexed by VDU code, is contained in two lookup tables. The hi byte is modified so that the number of associated parameters can be derived from it. The parameters are then assembled in a VDU queue.

C375-C3E6 - *640 and *40 multiplication tables
Lookup tables are provided for determination of the number of characters from the origin in text display modes.

C3E7-C3F6
Lookup tables for default text window size, indexed by screen mode.

C3F7-C447
Lookup tables for various display parameters. Several of these tables overlap one another and some serve a dual purpose.

C441-C444 - Sound pitch offset
If more than one channel is programmed to output the same frequency, it is not possible to distinguish by ear that a second or third channel is active. To avoid this effect, the pitch of channels 2 and 3 is offset slightly in the same way that piano strings are slightly detuned.

C447-C458
Screen buffer size and location lookup tables

C463-C468
Character position determination lookup tables

C469-C4A9
Initial register settings for 6845 cathode ray tube controller (CRTC) chip

C4AA-C4B5
Lookup tables for alternative entry points and branch loops in VDU processing routines

C4B6-C4B9
In Mode 7, ASCII codes for some characters are non-standard. This table provides the equivalents.

C4BA-C4BF
Soft character RAM allocation

C4C0-D93F - VDU routines
C4C0 - VDU command handler
Current VDU status is stored as a series of flag bits at Zero Page location D0. On input of a VDU command, permissible action is checked against VDU status and the entry point of the associated routine ascertained from a lookup table. The value of the status bit may be determined with OSBYTE 75. The significance of the bits is 0 - enable printer, 1 - disable scrolling, 2 - enable paged mode, 3 - enable software scrolling, 5 - enable text at graphics cursor, 6 - enable cursor editing and 7 - disable VDU. The command is processed and converted into instructions which are stored in

corresponding registers of the 6845 CRTC to modify the display. Parameters associated with a VDU command (e.g. VDU25,k,X,Y)) are stored as a VDU queue and processed in sequence.

Graphics display is controlled by manipulation of graphics coordinates. A region of low RAM is set aside as graphics coordinate workspace. Values such as the previous position of the graphics cursor are stored here. It is also used, for example, as a coordinate store in the 'plot and fill' routines.

Colour processing is controlled by a palette which stores the relationships between logical and physical colours. It is located in the video ULA chip.

D940-D9CC - Page 2 setup - default values
The RESET routine copies these values to Page 2

D9CD-DC67 - RESET and BRK handling routines
A microprocessor takes instructions one by one from the next address contained in its program counter register. When the computer is switched on the state of read/write memory is undefined, so it is necessary for the MOS ROM to contain routines which set up the contents of RAM and to initialise the peripheral controller chips. On RESET the 6502 microprocessor always transfers the address from locations FFFC and FFFD to its program counter and continues execution from that point, so the startup routines are stored in ROM at the vectored location.

Unless a disc or network filing system is installed, the BBC micro does not use non-maskable interrupts, so the first procedure is to transfer an RTI opcode to the starting address of the NMI routines (0D00). Subsequently, if the appropriate flag is set, the processor clears RAM to zero and successively checks the startup options (keyboard links), initialises the MOS variable store at the top of Page 2 (0290-02FF), copies default values of the OS vectors and remaining OS variables to the bottom of Page 2, initialises the CFS/RS432 asynchronous communications interface adapter (ACIA) chip and two versatile interface adapters (VIAs), checks for extended memory (JIM), initialises the sound generator chip and sound queues, switches off the cassette motor relay, resets the soft key buffers, checks for the presence of any paged ROMs and constructs a reference table of them in Page 2, checks for the presence of a speech processor and, if it finds one, initialises it, initialises the VDU system, sets the CFS interblock gap, checks the Tube for an active second processor and, if it finds one, initialises it, allocates public and private workspace for any paged ROMs, displays the greetings message ("**BBC Computer**"), checks memory size, sets up the keyboard and performs an autoboot, if required, or else initialises a language, copying it to the second processor where appropriate.

DC68-DE8B - IRQ handling routines

There are two types of interrupt request on the 6502 microprocessor - non-maskable interrupts (NMI) which, as the name implies, are dealt with immediately, and maskable interrupts (IRQ) which need not be processed until higher priority data processing has been completed. NMIs are reserved for filing systems and other operations which require instant and continuous processing. Maskable interrupt requests are further subdivided into high- and low-priority IRQs which are vectored by OS 1.2 respectively to DC54 and DE89.

Peripheral devices on the BBC microcomputer are interrupt-driven, that is to say, when they require servicing, they send an interrupt request to the 6502. When the processor receives an interrupt request, it calls the routine vectored by the addresses FFFE-FFFF. The entry point of the IRQ service call is at DC93. During the routine, the processor polls, in succession, the ACIA, the MOS VIA, the user VIA and the paged ROMs and then passes control to the low priority IRQ handler. If, at any point, the processor detects an interrupt request, it pauses to service that request before continuing to poll devices of lower priority in the chain.

Buffers are serviced by interrupts. When data are placed in a buffer, an interrupt request is generated. During the subsequent IRQ processing routine, these data are transferred from the buffer to the appropriate peripheral controller chip.

DE8C- DEA8 - Analogue-to-digital converter routines

ADC channel number is set up by OSBYTE call 11 which saves the specified (by X-parameter) channel as an Operating System variable at 024C and initialises the A/D converter. Actual A/D conversions are processed by IRQ handling routines at DE47.

DEA9-DEBA - ROM embedded message PRINTing routines

Located at intervals throughout the ROM are embedded messages such as "BBC microcomputer" and "RECORD then RETURN". These messages are in ASCII-encoded alphanumeric characters, followed by a terminator character "00". Error messages such as "Data?" or "Bad string" are preceded by a BRK opcode and a hexadecimal error code which may be used in error handling routines. The message processing routine sets up pointers to the location of the message and PRINTs it, character by character, by means of the MOS call OSASCI.

DEBB-DF0B - Character input routines

Nested routines transfer a character from Econet, an EXEC file, or the soft key or input buffers to the accumulator. A check is made for ESCAPE, which is processed if found.

DF0C-E3A7 - MOS '*' command handler and printer output routines

The MOS '*' command handler checks an input command against standard commands listed in a lookup table (DF10-DF88), recognising abbreviations that are terminated with a full stop (ASCII 2E). If a command is identified, it is processed at the entry point listed immediately after the command in the lookup table. Unrecognised commands are offered to the paged ROMs and then to the current filing system files. Initialisation is provided (GSINIT) for the operating system general string input routine (GSREAD).

E3A8-E434 - User-defined key routines

Up to sixteen soft key definitions may be stored in Page B buffers. The key strings are input by GSREAD and inserted in key- number order in the buffer. A pointer to the last character is maintained and, when a definition is changed, the positions of the remaining definitions are correspondingly adjusted.

E435-E5B2 - Buffer housekeeping routines

Lookup tables hold the start locations of the keyboard and RS432 input and RS432, printer, sound and speech output buffers in the form of a base address plus offset.

E5B3-EB02 - OSBYTE and OSWORD routines

MOS actions are initiated by operating system calls. Some of these, such as OSNEWL and OSRDRM, perform a single function, but others perform a number of operations, specified by input parameters. The most important of these are OSBYTE and OSWORD. Both of these have a call number, held in the accumulator. OSBYTE may have either one entry parameter, held in the X-register, or two, held in the X- and Y-registers. OSWORD has a control block, the start address of which is pointed by the X- and Y-registers. Entry points of OSBYTE and OSWORD calls are held in lookup tables stored at E5B3-E656 and the entry points of most of the corresponding routines are grouped about these addresses.

EB03-EE12 - Sound processing routines

The SOUND command has a number of parameters controlling amplitude or envelope, pitch, duration, synchronisation, buffer flushing and channel number. These are stored in a sound buffer according to channel (0-3), processed and then passed to the sound generator chip during the interrupt polling procedure.

In order to provide a distinguishing feature between notes of the same pitch on different channels a small offset is added to the pitch of the notes on channels 2 and 3 (E073).

Unrecognised channels are offered to the paged ROMs (E83A).

EE13-EED9 - Speech processing routines
If the speech system is implemented, it is initialised on RESET. When speech is being processed the ROM filing system is enabled. The characteristics of the RFS are similar to those of the cassette filing system, except that, obviously, no commands involving data saving will function.

In order to implement speech, bytes are read from a speech storage ROM (or PHROM) and passed nibble by nibble to the speech processor chip.

EEDA-F134 - Keyboard input and housekeeping routines
Access to the keyboard is over a slow eight-bit databus connected to Port B of the MOS VIA. Read or write operations are selected by the data direction register. The keyboard is an interrupt-driven device, feeding an input buffer 03E0-03FF. A two-key rollover is operated with the latest key pressed being stored at EC and the previous key at ED. Keys generate internal key numbers which are converted to ASCII codes by means of a series of lookup tables, indexed by the key number from a base address EFAB. Interposed between the conversion lookup tables are small routines requiring only a few bytes of code.

F135-FBFF - CFS/RFS routines
Nearly 3K of the operating system ROM is devoted to the cassette and ROM filing system routines. Due to the limited nature of these filing systems, not all of the operations or operating system calls relating to other filing systems can be fully implemented. For example, if a '*' command is not executed by the operating system or one of the sideways ROMs, it is passed to the current filing system which will normally attempt to '*RUN' the command. If no file is found within a default time limit, the filing system displays an error message. Since a cassette tape is open-ended, there is no guarantee that the file will be found within a reasonable time, so the CFS simply does not attempt to execute the operation.

FC00-FEFF - Memory mapped I/O
FC00-FCFF - FRED 1 MHz Bus memory-mapped I/O
FD00-FDFF - JIM 1 MHz Bus memory expansion page
FE00-FEFF - SHEILA MOS memory-mapped I/O

The 6502 microprocessor does not have any I/O capability. Part of its address space is therefore devoted to memory-mapped I/O. In the case of OS 1.2, Page FC (FRED) is allocated to peripherals and Page FD (JIM) to memory accessed over the 1 MHz bus. Part of this is reserved for Acorn's expansions, such as a hard disc controller or the teletext receiver and part is dedicated to user applications.

Page FE (SHEILA) is concerned with internal devices and is used

to service the peripheral controller chips built into the computer.

FF00-FFA5 - BRK and interrupt handling routines
After processing a BRK instruction or an interrupt, operation may need to be resumed at the previous point which may be within a sideways ROM. By rearranging the stack, and storing the identity of the ROM which is active, this can be achieved.

FFA6- FFF9 - Miscellaneous MOS call entry points
Defined entry points for MOS calls are provided to ensure compatibility with future upgrades. The locations of the routines themselves will change as they are rewritten.

FFFA-FFFF - 6502 vectors
On NMI, RESET, or IRQ, the 6502 transfers the values stored at FFFA-B, FFFC-D, or FFFE-F respectively to its program counter. The operating system ROM therefore stores the entry points of the corresponding routines at these addresses.

Low memory RAM usage
0200-0235 - MOS vectors

When the computer is switched on, the MOS sets up a number of pointers or vectors at the bottom of Page 2. These vectors are two-byte addresses of the entry point of routines which perform specific operations. For instance, UPTV, the user printer routine pointer is at 0222-0223. Thus, when the computer wishes to send a byte to a printer via a user-installed output routine, it sends it to the address stored at these two locations. (In conventional 6502 practice, the low byte is stored in the lower and the high byte in the higher address location.) Since the pointer locations are in read/write memory, they may easily changed. If a new filing system is initialised, it will change the values of certain of these vectors to point to its own routines. Alternative versions of the MOS require their own vector settings, since the layout of the ROM is different in each case.

0236- 02FF - MOS variables store and workspace
Parameters used by the operating system, such as the current setting of the system clock, or the VDU palette are stored in Page 2 above the MOS vectors. They are updated by the OS housekeeping routines and may be read or reset by user- or program-initiated calls.

0300-037F - VDU variables store and workspace
The bottom half of Page 3 is devoted to storage of parameters needed for the screen display.

0380-03DF - CFS/RFS variables store
The main part of the top of Page 3 is similarly allocated to the cassette and ROM filing systems

03E0-03FF - Keyboard input buffer

0400-06FF - TUBE communications routines
If no second processor is connected, Pages 4-7 are allocated to the current language ROM. For example, they are used by Basic for its variables tables and loop, function and subroutine parameter stores. If, however, a second processor is connected and active, the DNFS ROM installs the routines required for Tube communications in Pages 4-6. In a protocol similar to that used by the sideways ROMs, there is a language entry point at 0400 and a service entry point at 0403. The data transfer routines start at 0406.

0800-083F - Sound queues and workspace
In order to permit synchronisation of sound commands to different channels, the commands are transferred from the four sound buffers to queues in Page 8. The four queues are indexed by buffer number. They are serviced during the maskable interrupt processing routine. At the appropriate time, parameters are taken from the sound queues, processed and passed as instructions to the sound generator chip.

0840-0CFF - Buffers
The remainder of Pages 8-C are allocated to input and output buffers.

0D00-0D9E - NMI routines
High priority filing systems such as the disc and Econet use non-maskable interrupts. When these systems are initialised, they install NMI routines at the bottom of Page D, changing them according to prevailing circumstances. The CFS and RFS do not use NMIs, so, on RESET, the MOS install an RTI opcode (40) at address 0D00.

0D9F-0DFF - Paged ROM variables store
In order to permit the MOS vectors to point into a paged ROM, an extended vectors table is maintained at 0D9F-0DEF. This contains the value of each vector (lo then hi) followed by the corresponding ROM number.

Paged ROMs require space for storage such as filing system I/O buffers. This space may be shared with the other paged ROMs as public workspace or it may be reserved exclusively for an individual ROM as private workspace. Both are allocated on RESET, a table of private workspace addresses being maintained at 0DF0-0DFF.

C000-C4BF - VDU display setup parameters and routines

C000-C2FF :character font lookup table
C300 :initialise system
C303 :"BBC Computer" embedded message
C312 :"16k + BELL" embedded message
C317 :"32K + BELL" embedded message
C31F-C32E :4-colour MODE byte mask lookup table
C32F-C332 :16-colour MODE byte mask lookup table
C333-C353 :VDU entry point lo lookup table
C354-C374 :VDU entry point hi parameter lookup table
C375-C3B3 :*640 multiplication table (40-,80-column MODEs)
C3B5-C3E5 :*40 multiplication table (teletext MODE)
C3E7-C3EE :text window - bottom row lookup table
C3EF-C3F6 :text window - right-hand column lookup table
C3F7-C3FE :video ULA control register setting
C3FF-C406 :number of bytes per character for each display MODE
C407-C408 :mask table for 2-colour MODEs
C409-C40C :mask table for 4-colour MODEs
C40D-C414 :mask table for 2-colour MODEs, font flag mask table
C414-C41B :number of colours minus one for each MODE
C41B-C425 :GCOL PLOT options processing lookup table
C424-C425 :2-colour MODE colour parameter lookup table
C426-C429 :4-colour MODE colour parameter lookup table
C42A-C439 :16-colour MODE colour parameter lookup table
C43A-C441 :display MODE pixels/byte -1 lookup table
C440-C447 :screen display memory index lookup table
C441-C444 :sound pitch offset by channel lookup table
C44B-C44E :CRTC setup parameter
C44F-C453 :CRTC setup parameter
C447-C458 :VDU section control numbers
C459-C45D :MSB of memory occupied by screen buffer
C45E-C462 :MSB of first location occupied by screen buffer
C463-C465 :number of bytes/row
C466-C468 :row multiplication table pointer lo lookup table
C469-C46D :CRTC cursor end register setting lookup table
C46E-C479 :6845 registers 0-11 for MODEs 0-2
C47A-C485 :6845 registers 0-11 for MODE 3
C486-C491 :6845 registers 0-11 for MODEs 4-5
C492-C49D :6845 registers 0-11 for MODE 6
C49E-C4A9 :6845 registers 0-11 for MODE 7
C4AA-C4AD :VDU routine vector addresses
C4AE-C4B1 :VDU routine branch vector address lo
C4B2-C4B5 :VDU routine branch vector address hi
C4B6-C4B9 :teletext character conversion table
C4BA-C4BF :soft character RAM allocation

C4C0-D93F- VDU routines

C4C0 :check VDU command, if normal character, PRINT it

```
C4ED    :RUBOUT
C511    :VDU 00 - do nothing
C511    :VDU 06 - enable VDU drivers
C511    :VDU 27 - do nothing
C53B    :VDU 01 - send next character to printer only
C55E    :confirm input, output cursors not separated ....
C565    :prepare cursor character for display ....
C568    :interchange cursors, set CRTC address registers, toggle VDU screen
        :status
C588    :get graphics cursor printing status bit
C58D    :VDU 14 - paged mode on
C596    :VDU 02 - enable printer
C59B    :VDU 21 - disable VDU drivers or delete current line
C59D    :set VDU status or, if zero, reset default
C5A1    :VDU 03 - disable printer
C5A6    :VDU 15 - paged mode off
C5A8    :reset specified VDU status bits
C5AD    :VDU 04 - write text at text cursor
C5B9    :VDU 05 - write text at graphics cursor
C5C5    :VDU 08 - backspace one character
C621    :move graphics cursor one position
C65B    :VDU 11 - move cursor up one line
C664    :VDU 09 - advance text cursor if enabled
C684    :set text cursor to new line, scrolling if required
C6AC    :clear text line to background, calculate offset, set CRTC address
C6AF    :calculate offsets of character row, set text cursor address register
C6B4    :paged mode scrolling
C6F0    :VDU 10 - move cursor down one line (line feed)
C6FA    :VDU 28 - define text window
C735    :OSWORD 09 - read pixel value
C74B    :OSWORD 0B - read palette
C759    :VDU 12 - clear text area
C779    :VDU 30 - home text cursor to top left
C787    :VDU 31 - move text cursor to X,Y
C7A8    :interchange current text cursor with value from general graphics
        :coordinate workspace
C7AF    :VDU 13 - move cursor to start of current line
C7C0    :VDU 16 - clear graphics area
C7F9    :VDU 17 - define text colour
C7FD    :VDU 18 - define graphics colour
C839    :VDU 20 - restore default logical colour
C88F    :save colour index and define logical colour
C892    :VDU 19 - define logical colour
C89E    :set logical/physical colour relationship
C8E0    :OSWORD 0C - write palette
C8EB    :VDU 22 - select screen MODE
C8F1    :VDU 23 - re-program display character or write to CRTC
C938    :process unrecognised VDU command
C93F    :setup CRTC
```

```
C951    :set 6845 cursor register to previous value
C954    :set 6845 cursor parameters register = A-register
C958    :write parameter from VDU queue to specified CRTC register
C95E    :set specified CRTC register
C98C    :VDU 25 - PLOT k,X,Y
C994    :backspace one row, set CRTC screen start address registers
C9A4    :advance one row, set CRTC screen start address registers
C9B3    :set 6845 CRTC screen start address registers
C9BD    :VDU 26 - restore default windows
C9F6    :save 68465 text cursor position address, set corresponding 6845
        :registers
CA02    :set 6845 text cursor address registers
CA0E    :set 6845 CRTC registers indexed by Y-register
CA2B    :write two bytes to consecutive 6845 registers
CA39    :VDU 24 - define graphics window
CA88    :set text window width according to character size
CAA2    :VDU 29 - define graphics origin
CAAC    :VDU 127 - backspace and delete
CAD4    :add number of bytes in row to display address
CAE0    :zero PRINT line counter and scroll if enabled
CAE3    :scroll if enabled
CB14    :zero page PRINT line counter
CB1D    :initialise video display parameters and workspace
CB33    :set up video display parameters table
CBC1    :initialise screen display parameters
CBF3    :OSWORD 0A - read character definition
CC02    :character font definition indirection vector lookup table
CCF5    :process unrecognised VDU command
CCF8    :scroll back one row, check if wrap-round needed
CD07    :OSBYTE 14 - explode soft character RAM allocation
CD3F    :move text cursor to next line
CD6A    :process character for display
CD77    :recover A-register, processor status from stack
CD7A    :process cursor cHaracter for display
CDA4    :soft scroll one line
CDCE    :set scan line address, scroll back until counter zero
CDDA    :interchance current text cursor with value from coordinate workspace
CDDE    :interchange two Page 3 bytes indexed by X,Y
CDE2    :interchange old and current graphics cursors
CDE4    :interchange old graphics cursor with value from coordinate workspace
CDE6    :interchange four-byte block indexed by X,Y in Page 3
CDFF    :scroll text one line
CE2A    :set scan line address = source line address
CE38    :copy source line to scan line
CE5B    :determine text window height, if non-zero do CR
CE6E    :set text cursor X-coordinate to left-hand side of text window
CE73    :add character to scan line
CEAC    :clear text line to background colour
CEE8    :confirm cursor in text window, calculate character row offset
```

```
CF06    :calculate address of text cursor
CF63    :PRINT character at graphics cursor
CFA6    :home graphics cursor
CFAD    :set graphics cursor X-coordinate to left-hand column
CFB7    :set up character definition and display character
CFBF    :display character
CFDC    :insert teletext character in scan line, convert if required
D03E    :set up character definition pointers
D060    :process PLOT command
D0B3    :set Page 0 colour processing bytes
D0D9    :move graphics cursor to absolute position
D0DC    :copy current graphics cursor coordinates from VDU queue
D0DE    :copy four bytes from VDU queue
D0E3    :set colour of point at current graphics cursor
D0EB    :omit last point in line
D0F0    :set colour of current graphics point, save to graphics scan line
D0F3    :process colour and save to graphics scan line
D104    :process colour of current graphics point
D10D    :check current graphics cursor position with respect to to window
        :boundaries
D10F    :check position of indexed graphics coordinates with respect to
        :window boundaries
D128    :check current graphics point position with respect to window
        :boundaries
D149    :adjust coordinates according to display MODE
D14D    :adjust PLOT coordinate according to display options
D176    :adjust coordinate for relative/absolute PLOT, divide it by 2
D1AD    :divide PLOT coordinate by 2
D1B8    :calculate graphics cursor offset according to display MODE
D1D5    :adjust cursor coordinates, calculate cursor offset
D1E3    :use graphics origin to calculate cursor offset
D1ED    :execute PLOT command
D214    :set up branch loops, PLOT parameters
D2E3    :set scan byte if appropriate, adjust coordinates and scan position
D3D3    :decrement graphics cell top line address by one row, set Y=7
D3ED    :set graphics mask = left-hand colour mask, increment graphics cell
        :by one line
D3FD    :set graphics mask = right-hand colour mask, decrement graphics cell
        :by one line
D40D    :calculate graphics cursor relative to graphics window
D411    :calculate graphics coordinates relative to specified point
D418    :calculate graphics coordinate offset
D42C    :if necessary, set up working coordinates, evaluate difference,
D42C    :make positive, divide Y by 2
D459    :take modulus of working coordinate, divide Y coordinate by 2
D467    :change sign of working coordinate, if negative
D47C    :copy 8 Page 3 bytes to new location (indexed by X,Y)
D480    :copy two Page 3 bytes to coordinate workspace
D482    :copy two Page 3 bytes to new location (indexed by X,Y)
```

```
D486    :copy current graphics cursor to coordinate workspace
D488    :copy current graphics cursor to new Page 3 location (indexed by Y)
D58A    :copy four Page 3 bytes to new location (indexed by X,Y)
D49B    :form 2s complement of number in Y-,A-registers
D4AA    :if byte legal, process colour, else, discard it and move cursor
D4BF    :omit last point in inverting actions
D506    :PLOT with dotted line
D545    :move graphics cursor and calculate offset for MODE
D54B    :get current graphics byte mask, set colour, save byte to scan line
D574    :reset graphics byte and save to scan line
D592    :check X coordinate from VDU queue against graphics window boundary
D5AC    :process working X coordinate, transfer to VDU queue
D5CE    :OSWORD 0D - read old and current graphics position
D5D5    :transfer four bytes index by A-register to OS buffer
D5EA    :PLOT and fill triangle
D632    :if old graphics cursor greater than new PLOT point, interchange them
D636    :if source coordinate greater than destination coordinate,
        :interchange points
D647    :OSBYTE 86 - read text cursor position (POS and VPOS)
D658    :process coordinates and PLOT points til target reached
D6A2    :set and arrange working coordinates, ....
D6A6    :arrange working coordinates, process colour mask, save to scan line
D70E    :recover target coordinates from stack
D774    :derive coordinates for next point in triangular PLOT
D7AC    :add stored value to working coordinate, increment counter if
        :necessary
D7C2    :OSBYTE 87 - read character at text cursor position
D808    :derive character definition from displayed character
D839    :determine logical colour of specified point
D85D    :get coordinates from VDU queue, set up graphics line address, colour
        :mask
D85F    :set coordinate source index, set up line address, colour mask *
D864    :set up graphics line address
D8CE    :if enabled, PRINT at cursor position
D905    :if text cursor enabled, read character
D918    :reset CRTC cursor register, cursors together, enable screen, set
        :A-register to 0D
D923    :OSBYTE 84 - read HIMEM
D926    :OSBYTE 85 - read address of screen buffer for MODE
```

D940-D9CC - Page 2 setup - default values

```
D940-D976 :Page 2 vectors default settings
D976-D9CC :MOS variables default settings
```

D9CD-DC67 - RESET and BRK handling routines

```
D9CD    :RESET entry point
DBB8    :set up language
```

```
DBE7    :OSBYTE 8E - enter language ROM
DC0B    :switch ROM page, read byte, return to original ROM
DC16    :save ROM identity and switch paged ROM
DC1C    :maskable interrupt entry point
DC54    :BRKV - default BRK routine
```

DC68-DE8B-IRQ handling routines

```
DC68    :check buffer status, receive IRQ, set ACIA
DC7D    :transfer received serial data to input buffer
DC93    :IRQ1V - high priority IRQ routine entry point
DCA2    :poll 6850 ACIA for IRQ, service it if required
DCB3    :generate RS432 error event
DCBF    :output character from RS432 buffer
DCD6    :transmit serial data from output buffer
DCDE    :check for RS432 error
DCF3    :offer IRQ to paged ROMs, then low priority IRQ handler
DD06    :poll MOS VIA for IRQ, service it if required
DD13    :check for vertical sync IRQ, service it if required
DD47    :poll USR VIA for IRQ, service it if required
DD69    :check for further MOS VIA IRQs
DD6F    :process MOS VIA Timer 2 IRQ
DD79    :transfer data from speech buffer to processor
DDCA    :check MOS Timer 1 timeout IRQ, service it if required
DE47    :check for ADC IRQ, service it if required
DE6E    :set MOS VIA interrupt flag register
DE72    :check for key-pressed IRQ, service it if required
DE89    :IRQ2 - low priority IRQ handler routine
```

DE8C-DEA8-analogue-to-digital converter routines

```
DE8C    :OSBYTE 11 - force analogue-to-digital conversion
DE8F    :set up ADC channel
```

DEA9-DEBA-ROM embedded message printing routines

```
DEA9    :set BRK address = C300, PRINT embedded message
DEAB    :set BRK address, PRINT embedded message
DEB1    :PRINT embedded message
```

DEBB-DF0B-character input routines

```
DEBB    :set INKEY countdown timer, read character to A-register
DEC5    :RDCHV - read character from A-register
DED2    :read character from EXEC file
```

DF0C-E3A7 -MOS '*' command handler and printer output routines

DF0C-DF0F :copyright symbol lookup table
DF10-DF88 :command keyword and vector lookup table
DF89 :CLIV - interpret command line routine
DFB6 :process '*/' command
DFF2 :push command entry address on stack, then execute command
E004 :move pointers to next command vector
E009 :check command against commands from lookup table
E00D :transfer command line pointers to X-,Y-registers
E018 :*BASIC command
E031 :*RUN command
E031 :*CAT command
E039 :increment index (Y) and ....
E03A :flush MOS command buffer to non-blank, check for CR
E043 :flush line to non-blank, check for "," or CR
E045 :flush to non-blank, check for "," or CR
E04E :flush line to non-blank, convert ASCII to hexadecimal, clear carry
E07C :convert ASCII numeral input to hexadecimal
E07D :increment index (Y) and ....
E08A :flush input line to non-blank, clear carry
E08F :if hexadecimal numeral remove ASCII offset
E0A4 :WRCHV entry point - write character from A-register
E114 :if permitted, PRINT character
E11E :insert character into buffer and PRINT it
E13A :PRINT character via appropriate routine
E143 :parallel printer output
E168 :serial printer output
E170 :set buffer flag busy, if no space free, reset ACIA control register
E173 :if no buffer space free, reset ACIA control register
E17A :reset ACIA control register
E17C :OSBYTE 9C - read/write 6845 ACIA control register
E189 :write to ACIA control register, save setting to Page 2
E191 :PRINT character via Econet or user printer routine
E197 :OSBYTE 7B - inform OS, printer going dormant
E19B :if printer buffer occupied, output character
E1A2 :select printer buffer and output character
E1A4 :PRINT character via Econet or user printer routine
E1AD :flush specific buffer
E1AE :flush buffer
E1D1 :CNPV entry point - count or purge buffer
E1F8 :attempt to insert character into buffer
E20E :initialise parameter block 02EE,X-02FA,X
E21F :pack character into OSFILE control block
E227 :*KEY command
E23C :*LOAD command
E267 :"Bad address" error message - error code FC
E275 :OSBYTE 77 - close any SPOOL or EXEC files
E281 :*SPOOL command

```
E28F    :close SPOOL file
E2AD    :contruct OSFILE parameter block from OS command line buffer
E310    :"Bad command" error message - error code FE
E310    :USERV default entry point
E31D    :"Bad key" error message - error code FB
E33E    :*SAVE command
E342    :*FX command
E348    :*TAPE command
E348    :*TV command
E348    :*ROM command
E348    :*OPT command
E348    :*MOTOR command
E348    :*CODE command
```

E3A8-E434 - user-defined key routines

```
E3A8    :calculate length of key string
E3D1    :set up soft key definition
E3E9    :"Key in use" error message - error code FA
```

E435-E5B2 - buffer housekeeping routines

```
E435-E43D :buffer address hi lookup table
E43E-E446 :buffer address lo lookup table
E447-E44F :buffer start address offset
E450    :set up buffer pointers (X defines buffer)
E45B    :examine buffer status
E460    :OSBYTE 91 - get character from buffer
E464    :REMV entry point - remove byte from buffer (X= buffer number)
E494    :if event flag true, process event, clear carry
E4AF    :OSBYTE 8A - insert character into buffer
E4B0    :indirected insertion of character into buffer
E4B3    :INSBV entry point - insert byte into buffer
E4E3    :check if character alphabetic (carry clear)
E4F1    :insert character in keyboard buffer, check for ESCAPE
E4F3    :OSBYTE 99 - insert character into buffer, checking for ESCAPE
E577    :get character from Econet, soft key or input buffer as appropriate
E57E    :go to Econet control entry
E581    :get character from soft key or input buffer
```

E5B3-E656 - OSBYTE and OSWORD entry point vectors

```
E5B3    :OSBYTE calls 00-15 routines vector lookup table
E5DF    :OSBYTE 75-A0 routines vector lookup table
E637    :OSBYTEs A6-FF entry vector
E63B    :OSWORD vector lookup table
```

E657-EB02 - miscellaneous OSBYTE and OSWORD routines

```
E657    :OSBYTE 88 - execute code indirected via USERV (*CODE)
E659    :OSBYTE 0D - disable events
E673    :OSBYTE 7C - clear ESCAPE condition
E674    :OSBYTE 7D - set ESCAPE condition
E67F    :OSBYTE 88 - switch motor relay (*MOTOR)
E689    :OSBYTE 08 - set RS432 transmit rate
E68B    :OSBYTE 07 - set RS432 receive rate
E6A7    :set serial ULA control register
E6B0    :OSBYTE 09 - set flashing colour mark duration
E6B2    :OSBYTE 0A - set flashing colour space duration
E6D3    :OSBYTE 02 - select input stream
E6F9    :OSBYTE 0D - disable events
E6FA    :OSBYTE 0E - enable events
E706    :OSBYTE 10 - select ADC channels to be sampled
E713    :OSBYTE 81 - read key with time limit (INKEY)
E729    :OSBYTE 82 - read machine higher order address
E732    :check occupancy of input or free space of output buffer
E738    :clear overflow and count buffer contents
E73B    :set overflow and purge buffer
E741    :check buffer space used and compare with space free
E74F    :OSBYTE 80 - read ADC channel or get buffer status
E75F    :check fire buttons
E772    :BYTEV - OSBYTE routines entry point
E7BC    :merge in saved carry, clear overflow
E7C2    :process OSBYTE calls below 75
E7CA    :discard two bytes from stack, process unrecognised OSWORD call
E7DC    :check if free to PRINT message
E7EB    :WORDV - OSWORD routines entry point
E803    :OSWORD 05 - read byte of I/O processor memory
E80E    :write byte of I/O processor memory
E815    :set I/O processor address pointers
E821    :OSBYTE 00 - identify ROM
E823    :"OS 1.20" error message - error code F7
E82D    :OSWORD 07 - SOUND command
E86C    :OSBYTE 75 - read VDU status
E86F    :VDU 7 - sound BELL
E8AE    :OSWORD 08 - define an envelope
E8C9    :compare command line character with 16, get bits 0,1, increment
        :index
E8D1    :OSWORD 03 - read interval timer
E8D5    :OSWORD 01 - read system clock
E8E4    :OSWORD 04 - write interval timer
E8E8    :OSWORD 02 - write system clock
E902    :OSWORD 00 - read line from current input to memory
E93B    :delete previous input character
E946    :delete entire input line
E976    :OSBYTE 05 - select printer
```

```
E988        :OSBYTE 01 - user OSBYTE call (read/write location 0281)
E988        :OSBYTE 06 - set character ignored by printer
E98C        :OSBYTE 0C - set keyboard autorepeat rate
E995        :OSBYTE 0B - set keyboard autorepeat delay
E997        :OSBYTE 03 - select output stream
E997        :OSBYTE 04 - enable/disable cursor editing
E99C        :OSBYTEs A6-FF entry point
E9AD-E9B5   :baud rate setting command lookup table
E9C0        :OSBYTE A0 - read VDU variable
E9C8        :reset soft key buffer contents
E9D9        :OSBYTE 76 - set keyboard to match LED status
E9EA        :reset keyboard LEDs if carry clear, then test ESCAPE flag
E9F8        :write to MOS VIA register B
E9FF        :OSBYTE 9A - write to video ULA control register
EA00        :set new video ULA control parameter, set flash counter
EA10        :OSBYTE 9B - write to video ULA palette register
EA11        :save physical/logical colour conversion factor, set palette
EA1D        :initialise string input with first space, 2nd ' "'or CR terminator
EA1E        :GSINIT routines - initialise string input
EA2F        :GSREAD routines - parse input string, read character
EA8F        :"Bad string" error message - error code FD
EA9C        :check for specified keys pressed
EABF        :if alpha character, convert to control code
EAD2        :"/!BOOT" embedded message
EAD9        :perform vectored jump if Page 2 contains opcode
EAE3        :OSBYTE 90 - alter TV display parameters (*TV command)
EAF4        :OSBYTE 93 - write byte to FRED
EAF9        :OSBYTE 95 - write byte to JIM
EAFE        :OSBYTE 97 - write byte to SHEILA
```

EB03-EB12 - sound processing routines

```
EB03        :set completion flag, switch off sound
EB0A        :if sound output not disabled, write to sound generator
EB21        :write command to sound generator, preserving processor status
EB22        :write command to OS VIA Port A, blip sound generator
EB40        :sound channel address register parameter lookup table
EB44        :check buffer number, process next sound channel
EB47        :process sound interrupt
EB59        :process next channel in sound queue
EC59        :if buffer index not 4, process next sound channel
EC60        :initialise sound queues
EC6B        :flush current sound queue
ECA2        :initialise sound channel parameters
ECD0        :if no envelope specified, switch off sound
ED01        :update pitch commands to sound chip
ED95        :write command to sound generator via slow databus
ED98        :process next sound command from buffer
EDF7        :set duration parameter in sound queue
```

EDFB-EE06 :sound chip pitch lo order control parameter lookup table
EE07-EE12 :sound chip pitch hi order parameter lookup table

EE13-EED9 - speech processing routines

EE13 :set current filing system ROM/PHROM
EE18 :get byte from data ROM
EE3B :get address and set speech data ROM address pointer
EE40 :transfer two bytes from speech processor to ROM address pointers
EE51 :read RFS data ROM or speech processor
EE62 :if PHROM, read byte from speech processor
EE6D :OSBYTE 9E - read from speech processor
EE71 :load two nibbles to speech processor address pointer
EE7A :load a nibble to speech processor address register
EE7F :OSBYTE 9F - write to speech processor
EE84 :confirm speech system implemented, write command to speech processor
EEAD :set RFS data ROM address pointers
EEBB :set speech data ROM address register

EEDA-F134 - keyboard input and housekeeping routines

EEDA :if no key pressed, enable keyboard, reset shift lock, caps lock
EEEB :set shift or caps lock as appropriate
EF02 :KEYV entry point - controls keyboard access
EF13 :timer interrrupt entry point
EF16 :test shift and control keys
EF74 :set keyboard status, zero autorepeat timer, scan keyboard
EFAB :internal key number to ASCII lookup tables base address
EFC1 :check for ESCAPE, enter processed code in keyboard buffer
EFE9 :scan keyboard, save any key pressed
F00F :key-pressed interrupt entry point
F01F :set autorepeat countdown timer
F02A :write-disable keyboard - write/read keyboard over slow databus
F03B-F044 :internal key number to ASCII lookup table - keys 10-19
F045 :OSBYTE 78 - write current keys pressed
F055 :go to JIM paged entry vector
F04B-F054 :internal key number to ASCII lookup table - keys 20-29
F058 :perform indirected operation
F05B-F064 :internal key number to ASCII lookup table - keys 30-39
F065 :test if any bit set, go to KEYV
F068 :go to KEYV
F06B-F074 :internal key number to ASCII lookup table - keys 40-49
F075-F07A :MOS VIA settings for speech system
F07B-F084 :internal key number to ASCII lookup table - keys 50-59
F085 :OSBYTE 83 - read OSHWM
F08B-F094 :internal key number to ASCII lookup table - keys 60-69
F095 :set buffer number and flush it
F09B-F0A4 :internal key number to ASCII lookup table - keys 70-79

```
F0A5    :go to event handling routine
F0A8    :OSBYTE 0F - flush selected buffer class
F0AA    :flush all buffers
F0B4    :OSBYTE 15 - flush specific buffer
F0B9    :issue *HELP expansion service call, PRINT ROM identity
F0C1    :"OS 1.20" embedded message
F0CC    :clear carry and scan keyboard from 16 decimal
F0CD    :OSBYTE 7A - keyboard scan from 16 decimal
F0CF    :OSBYTE 79 - keyboard scan
F0D1    :scan keyboard (as OSBYTE 79)
F129    :write enable keyboard, blip interrupt enable
F12E    :write enable keyboard, transfer X- to A-register
```

F135-FBFF - CFS/RFS routines

```
F135    :OSBYTE 8C - select CFS (*TAPE)
F135    :OSBYTE 8D - select ROM filing system (*ROM)
F140    :set OPT interblock gap to default
F14B    :initialise CFS and claim vectors
F168    :OSBYTE 8F - issue paged ROM service call
F18E    :CFS OSARGS routines - read/write whole file arguments
F1A3    :CFS OSFSC routine vector lookup table
F1B1    :CFS OSFSC routine - filing system control
F1C4    :load file
F1F6    :switch motor off and PRINT "Locked" error message
F1F9    :"Locked" error message
F246    :sound BELL and abort
F249    :PRINT newline if indicated by current block flag
F24D    :if CFS not executing operation, PRINT CR
F25A    :get filename from OS command line buffer
F27D    :CFS OSFILE routines - read or write a whole file
F290    :save load and execute addresses
F2CB    :save file to tape
F305    :CFS OSFSC call 2 - '*/' command
F305    :CFS OSFSC call 4 - *RUN command
F32B    :CFS OSFSC call 5 - *CAT command
F33B    :reset CFS status byte bit 3 (not cataloguing)
F33D    :mask A-register with CFS status byte and save it back
F342    :set CFS status byte bit 3 (EOF reached)
F344    :set specified (A-register) status byte bits
F348    :read data from CFS/RFS
F3CA    :CFS OSFIND routine - open or close a file
F3D4    :close files
F3D7    :close all open files
F3F2    :open a file for access
F46F    :save file status, restore registers
F471    :restore X-,Y-registers, copy file status to A-register
F478    :if enabled, write final block, close file
F496    :set up header, save block to tape
```

```
F4C9        :CFS OSBGET routines - read one byte from file
F523        :"EOF" error message - error code DF
F529        :CFS OSBPUT routines - write one byte to file
F588        :communicate with current filing system (CFS/RFS)
F54D        :CFS OSFSC call 0 - *OPT
F561        :*OPT 1,n
F568        :*OPT 2,n
F573        :*OPT 3,n
F581-F587   :*OPT setting bitwise lookup table
F588        :communicate with current filing system (CFS/RFS)
F61E        : CFS OSFSC call 1 - check EOF
F631        :search for file
F637        :search for specified block
F61E        :CFS OSFSC call 1 - check EOF
F644        :"Searching" embedded message
F674        :"File not found" error message - error code D6
F68B        :read from EXEC file
F68D        :close EXEC file if open, read from specified file
F6AC        :seek BGET file, read block
F6B4        :read block from BGET file
F723        :"Bad ROM" error message - error code D7
F77B        :read block header
F797        :get character from file, do cyclic redundancy check
F7B0        :perform cyclic redundancy check
F7D5        :reset flags
F7D7        :set flags
F7EC        :write block to tape
F875        :transfer byte to CFS, do cyclic redundancy check
F87B        :save checksum to tape, reset buffer flag
F882        :save byte to buffer, transfer to CFS, reset flag
F884        :when buffer/ACIA transfer done, reset flag, transfer buffer to
            :A-register
F892        :generate 5 second delay
F896        :generate delay set by interblock gap
F898        :generate 0.2 second delay
F8A9        :if not first or last block, ....
F8B6        :PRINT CR if indicated by current block flag, then ....
F8B9        :update block flag received, PRINT filename (+ address if required)
F8CD        :PRINT filename from CFS header block
F8EB        :pad filename out with trailing spaces
F915        :PRINT four spaces, execute address and load address
F927        :PRINT four bytes from CFS block header
F934        :prompt to start recording
F94A        :"RECORD then RETURN" embedded message
F96A        :increment current load address
F975        :PRINT a space, then ASCII equivalent of hexadecimal byte
F97A        :PRINT ASCII equivalent of hexadecimal byte
F983        :convert hexadecimal digit to ASCII and PRINT it
F991        :PRINT a space
```

```
F995    :confirm CFS not executing operation nor ESCAPE flag set
F9AD    :process ESCAPE call
F9AB    :"Escape" error message - error code 11
F9B4    :load file from tape
F9BA    :"CR Loading" embedded message
F9D9    :load data from CFS/rom and check validity
FA46    :if CFS not exececuting operation, PRINT embedded message
FA4A    :PRINT following embedded message
FA52    :PRINT next character
FA72    :check if header block filename matches sought filename
FA8E    :"Data?" error message - error code D8
FA99    :"File?" error message - error code DB
FAA4    :"Block?" error message - error code DA
FAC2    :"CR BELL rewind tape CR" embedded message
FAD6    :check loading progress, read another byte
FAE8    :sound BELL if enabled, switch motor off, reset serial interface
FAF2    :enable 2nd processor if appropriate, switch motor off, reset serial
        :interface
FAFC    :setup serial interface, initialise timeout counter, reset ACIA
FB0A    :save processor status and reset ACIA
FB1A    :set sequential block gap according to OPT, inter-block gap,
        :interrogate ESCAPE flag, reset timeout, ACIA
FB27    :set current message option, inter-block gap, interrogate
        :ESCAPE flag, reset timeout, ACIA
FB46    :reset ACIA
FB4A    :set ACIA control register
FB50    :switch motor on, enable CFS output, set baud rate
FB63    :set ACIA to CFS baud rate
FB69    :increment current block number
FB78    :zero buffer flag and checksum bytes
FB7C    :zero checksum bytes
FB81    :set (0030,X) as sought filename
FB8E    :enable CFS for read operation, switch motor on
FB90    :switch motor on
FB95    :switch cassette motor
FB9C    :confirm file is open
FBB1    :"Channel" error message - error code DE
FBBB    :read from 2nd processor if present
FBBD    :if 2nd processor file, set up parameter block, enable 2nd processor
        :if present
FBC7    :enable 2nd processor
FBD3    :if file for 2nd processor, test TUBE presence flag
FBE2    :set up CFS for write op
```

FC00-FCFF - FRED - 1MHz Bus memory-mapped I/O

```
FC00-FC0F :test hardware
FC10-FC13 :Teletext
FC14-FC1F :Prestel
```

FC20-FC27 :IEEE interface
FC28-FC2F :reserved for Acorn expansion
FC30-FC3F :Cambridge Ring interface
FC40-FC47 :Winchester disc interface
FC48-FC7F :reserved for Acorn expansion
FC80-FC8F :test hardware
FC90-FCBF :reserved for Acorn expansion
FCC0-FCFE :user applications
FCFF :paging register for JIM expansion memory

FD00-FDFF- JIM - 1 MHz Bus memory expansion page

FD00-FD7F :reserved for Acorn
FD80-FDFF :user applications

FE00-FEFF- SHEILA - MOS memory-mapped I/O

FE00 :6845 CRTC address register
FE01 :6845 CRTC register file
FE08 :6850 ACIA W-control register R-status register
FE09 :6850 ACIA W-transmit data, R-receive data
FE10 :serial ULA control register
FE18 :68B54 ADLC R - Econet station ID / disable interrupts
FE20 :W - video ULA control register R - enable ADLC interrupts
FE21 :video ULA palette register
FE30 :LS161 paged ROM identity
FE40 :MOS 6522 VIA ORB/IRB
FE41 :MOS 6522 VIA ORA/IRA
FE42 :MOS 6522 VIA data direction - register B
FE43 :MOS 6522 VIA data direction - register A
FE44 :MOS 6522 VIA T1C-L write latches, read counter
FE45 :MOS 6522 VIA T1C-H high order latches
FE46 :MOS 6522 VIA T1L-L low order latches
FE47 :MOS 6522 VIA T1L-H high order latches
FE48 :MOS 6522 VIA T2C-L W-latches, R-counter
FE49 :MOS 6522 VIA T2C-H T2 high order counter
FE4A :MOS 6522 VIA shift register
FE4B :MOS 6522 VIA auxiliary control register
FE4C :MOS 6522 VIA peripheral control register
FE4D :MOS 6522 VIA interrupt flag register
FE4E :MOS 6522 VIA interrupt enable register
FE4F :MOS 6522 VIA as register 1 except no handshake
FE60 :USR 6522 VIA ORB/IRB
FE61 :USR 6522 VIA ORA/IRA
FE62 :USR 6522 VIA data direction - register B
FE63 :USR 6522 VIA data direction - register A
FE64 :USR 6522 VIA T1C-L write latches, read counter
FE65 :USR 6522 VIA T1C-H high order latches
FE66 :USR 6522 VIA T1L-L low order latches

```
FE67    :USR 6522 VIA T1 L-H high order latches
FE68    :USR 6522 VIA T2C-L W-latches, R-counter
FE69    :USR 6522 VIA T2C-H T2 high order counter
FE6A    :USR 6522 VIA shift register
FE6B    :USR 6522 VIA auxiliary control register
FE6C    :USR 6522 VIA peripheral control register
FE6D    :USR 6522 VIA interrupt flag register
FE6E    :USR 6522 VIA interrupt enable register
FE6F    :USR 6522 VIA as register 1 except no handshake
FE80    :8721 FDC W-command register, R-status register
FE81    :8721 FDC W-parameter register R-result register
FE82    :8721 FDC reset register
FE83    :8721 FDC W-illegal R-illegal
FE84    :8721 FDC W-data R-data
FEA0    :68B54 ADLC W-control register 1 R-status register 1
FEA1    :68B54 ADLC W-control register 2/3, R-status register 2/3
FEA2    :68B54 ADLC W-Tx FIFO (frame continue) R-Rx FIFO
FEA3    :68B54 ADLC W-Tx FIFO (frame terminate) R-Rx FIFO
FEC0    :PD7002 ADC W-data latch A/D start, R-status
FEC1    :PD7002 ADC hi data byte
FEC2    :PD7002 ADC lo data byte
FEE0    :TUBE FIFO1 status register
FEE1    :TUBE FIFO1
FEE2    :TUBE FIFO2 status register
FEE3    :TUBE FIFO2
FEE4    :TUBE FIFO3 status register
FEE5    :TUBE FIFO3
FEE6    :TUBE FIFO4 status register
FEE7    :TUBE FIFO4
```

FF00-FFA5 - BRK and interrupt handling routines

```
FF36    :NETV default entry point
FF56    :on interrupt, process extended vector
FF89    :discard 3 bytes from bottom of stack, switch ROMs
```

FFA6-FFF9 - miscellaneous MOS call entry points

```
FFA6    :CFS OSGBPB - no function implemented, contains RTS opcode
FFA6    :VDUV default entry point - contains RTS opcode
FFA6    :IND1V default entry point - contains RTS opcode
FFA6    :IND2V default entry point - contains RTS opcode
FFA6    :IND3V default entry point - contains RTS opcode
FFA7    :OSBYTE 9D - fast BPUT
FFAA    :OSBYTE 92 - read byte from FRED
FFAE    :OSBYTE 94 - read byte from JIM
FFB2    :OSBYTE 96 - read byte from SHEILA
FFB6    :default vector table length and location lookup table
FFB9    :OSRDRM - read from paged ROM
```

```
FFBC    :VDU character output
FFBF    :OSEVEN generate an EVENT
FFC2    :GSINIT - initialise OS string input routine
FFC5    :GSREAD - read character from input string
FFC8    :NVRDCH - non-vectored character input
FFCB    :NVWRCH - non-vectored character output
FFCE    :OSFIND - open or close a file
FFD1    :OSGBPB - transfer block to or from file
FFD4    :OSBPUT - save a byte to file
FFD7    :OSBGET - read a byte from file
FFDA    :OSARGS - read or write file arguments
FFDD    :OSFILE - read or write whole file or attributes
FFE0    :OSRDCH - read character from input
FFE3    :OSASCI - output ASCII character
FFE7    :OSNEWL - start new line
FFEE    :OSWRCH - output character
FFF1    :OSWORD - execute OS routine utilising parameter block
FFF4    :OSBYTE - execute OS call
FFF7    :OSCLI - execute command input string
```

FFFA-FFFF-6502 vectors

```
FFFA    :(0D00) NMI vector
FFFC    :(D9CD) RESET vector
FFFE    :(DC1C) IRQ vector
```

0000-00FF - Zero Page

```
0000-0001 :RAM initialisation pointers
00B0-00B3 :current load address
00B4-00B5 :current block number
00B6-00B7 :next block number
00B8-00B9 :text pointers
00BA      :current block flag
00BB      :current OPTion
00BC      :CFS temporary store, file status
00BD      :character temporary storage buffer
00BE-00BF :cyclic redundancy check workspace
00C0      :filing system buffer flag
00C1      :checksum result
00C2      :length of filename, progress flag
00C3      :current file handle
00C4      :temporary store
00C5      :not used
00C6      :current baud rate setting
00C7      :current interblock flag
00C8      :OSFILE parameter block pointer lo
00C9      :OSFILE parameter block pointer hi
00CA      :current ACIA control register receive/transmit setting
00CB      :cyclic redundancy check bit counter
00CC      :file length counter lo
00CD      :file length counter hi
00CE      :not used
00CF      :not used
00D0      :VDU status - bit 0 - printer output enabled
          :             bit 1 - scrolling disabled
          :             bit 2 - paged scrolling enabled
          :             bit 3 - software scrolling selected
          :             bit 4 - not used
          :             bit 5 - printing at graphics cursor enabled
          :             bit 6 - cursor editing mode enabled
          :             bit 7 - VDU disabled
00D1      :byte mask for current graphics point
00D2-00D3 :text colour bytes to be ORed and EORed into memory
00D4-00D5 :graphics colour bytes to be ORed and EORed into memory
00D6-00D7 :address of top line of current graphics cell
00D8-00D9 :address of top scan line of current text character
00DA-00DF :temporary workspace
00E0-00E1 :CRTC row multiplication table pointer
00E2      :CFS status - bit 0 - input file open
          :             bit 1 - output file open
          :             bit 2 - not used
          :             bit 3 - current CATalogue status
          :             bit 4 - not used
          :             bit 5 - not used
```

```
            :              bit 6 - EOF reached
            :              bit 7 - EOF warning given
00E3        :CFS OPTions - bits 1,0 LOAD error actions
            :                (00 - ignore, 10 - retry, 01 - abort)
            :              bits 3,2 LOAD error messages
            :                (00 - none, 10 - short, 11 - long)
            :              bits 5,4 SAVE error actions
            :                (00 - ignore, 10 - retry, 01 - abort)
            :              bits 7,6 SAVE error messages
            :                (00 - none, 1. - short, 11 - long)
00E4-00E6   :general MOS workspace
00E7        :autorepeat countdown timer
00E8-00E9   :OSWORD 0 input buffer pointers
00EA        :RS432 timeout counter (0 - RS432 timed out, hi bit set - RS432 not
            :                       yet timed out, 1 - CFS active)
00EB        :CFS active flag
00EC        :internal key number of most-recently-pressed key
00ED        :internal key number of previously-pressed key
00EE        :RAM copy of FRED paging register
            :internal key number ignored in keyboard scan
00EF        :most recent OSBYTE/OSWORD A-register
00F0        :most recent OSBYTE/OSWORD X-register, stack pointer at last BRK
00F1        :most recent OSBYTE/OSWORD Y-register
00F2-00F3   :MOS command/filename text pointer
00F4        :current paged ROM
00F5        :current RFS ROM number
00F6-00F7   :paged ROM or PHROM address pointer
00F8-00F9   :not used
00FA-00FB   :general MOS workspace
00FC        :pre-interrupt A-register setting
00FD-00FE   :BRK return pointers
00FF        :ESCAPE flag
```

0200-0235 - MOS vectors

```
0200        :(E310) USERV - user vector
0202        :(DC54) BRKV - break vector
0204        :(DC93) IRQ1V - high priority interrupt vector
0206        :(DE89) IRQ2V - low priority interrupt vector
0208        :(DF89) CLIV - command line interpreter vector
020A        :(E772) BYTEV - OSBYTE indirection vector
020C        :(E7EB) WORDV - OSWORD indirection vector
020E        :(E0A4) WRCHV - write character vector
0210        :(DEC5) RDCHV - read character vector
0212        :(F27D) FILEV - file processing vector
0214        :(F18E) ARGSV - file arguments vector
0216        :(F4C9) BGETV - get byte vector
0218        :(F529) BPUTV - put byte vector
021A        :(FFA6) GBPBV - get block/put block vector
```

021C :(F3CA) FINDV - open/close file vector
021E :(F1B1) FSCV - filing system control vector
0220 :(FFA6) EVNTV - event handling routine vector
0222 :(F3CA) UPTV - user PRINT routine vector
0224 :(FF36) NETV - ECONET vector
0226 :(FFA6) VDUV - VDU processing vector
0228 :(EF02) KEYV - keyboard access vector
022A :(E4B3) INSBV - buffer insert character vector
022C :(E464) REMV - buffer remove character vector
022E :(E1D1) CNPV - count/purge buffer vector
0230 :(FFA6) IND1V - unallocated vector
0232 :(FFA6) IND2V - unallocated vector
0234 :(FFA6) IND3V - unallocated vector

0236-02FF MOS variables store and workspace

0236 :(90) -OSBYTE A6- MOS variables base address lo
0237 :(01) -OSBYTE A7- MOS variables base address hi
0238 :(9F) -OSBYTE A8- ROM pointer table address lo
0239 :(0D) -OSBYTE A9- ROM pointer table address hi
023A :(A1) -OSBYTE AA- ROM information table address lo
023B :(02) -OSBYTE AB- ROM information table address hi
023C :(2B) -OSBYTE AC- key translation table address lo
023D :(F0) -OSBYTE AD- key translation table address hi
023E :(00) -OSBYTE AE- VDU variables start address lo
023F :(03) -OSBYTE AF- VDU variables start address hi
0240 :(00) -OSBYTE B0- CFS timeout counter (vertical sync counter -
: decremented 50 times/sec)
0241 :(00) -OSBYTE B1- current input buffer number
0242 :(FF) -OSBYTE B2- keyboard interrupt processing flag
0243 :(00) -OSBYTE B3- primary OSHWM
0244 :(00) -OSBYTE B4- current OSHWM address hi
0245 :(01) -OSBYTE B5- RS432 mode
0246 :(00) -OSBYTE B6- character definition explosion switch
0247 :(00) -OSBYTE B7- filing system flag - 0=CFS, 2=ROM
0248 :(00) -OSBYTE B8- current video ULA control register setting
0249 :(00) -OSBYTE B9- current palette setting
024A :(00) -OSBYTE BA- number of ROM enabled before last BRK
024B :(FF) -OSBYTE BB- number of BASIC ROM
024C :(04) -OSBYTE BC- current ADC channel number
024D :(04) -OSBYTE BD- maximum ADC channel number
024E :(00) -OSBYTE BE- ADC conversion type (00 - default, 08 - 8-bit,
: 0C - 12-bit)
024F :(FF) -OSBYTE BF- RS432 busy flag (bit 7 reset = busy)
0250 :(56) -OSBYTE C0- current ACIA control register setting
0251 :(19) -OSBYTE C1- flash counter
0252 :(19) -OSBYTE C2- mark period count
0253 :(19) -OSBYTE C3- space period count
0254 :(32) -OSBYTE C4- keyboard autorepeat delay

```
0255    :(08) -OSBYTE C5- keyboard autorepeat rate
0256    :(00) -OSBYTE C6- *EXEC file handle (0 - not allocated)
0257    :(00) -OSBYTE C7- *SPOOL file handle (0 - not allocated)
0258    :(00) -OSBYTE C8- bit 0 - ESCAPE enable/disable
        :                 bit 1 - BRK normal/memory clear
0259    :(00) -OSBYTE C9- Econet keyboard disable flag
025A    :(20) -OSBYTE CA- keyboard status - bit 3 - 1=shift pressed,
        :                 bit 4 - 0=caps lock, bit 5 - 0=shift lock,
        :                 bit 6 - 1=control, bit 7 - 1=shift enabled
025B    :(09) -OSBYTE CB- buffer space left at buffer full signal
025C    :(00) -OSBYTE CC- RS432 input suppression flag
025D    :(00) -OSBYTE CD- cassette/RS432 selection flag (CFS-0, RS432-40)
025E    :(00) -OSBYTE CE- Econet OS call interception flag (bit 7)
025F    :(00) -OSBYTE CF- Econet OSRDCH interception flag (bit 7)
0260    :(00) -OSBYTE D0- Econet OSWRCH interception flag (bit 7)
0261    :(50) -OSBYTE D1- speech enable/disable flag (50/20)
0262    :(00) -OSBYTE D2- sound output enable flag
0263    :(03) -OSBYTE D3- BELL channel number
0264    :(90) -OSBYTE D4- BELL amplitude/ENVELOPE number
0265    :(64) -OSBYTE D5- BELL frequency
0266    :(06) -OSBYTE D6- BELL duration
0267    :(81) -OSBYTE D7- bit 7 set = ignore startup message,
        :                 bit 0 set = ignore RFS !BOOT error
0268    :(00) -OSBYTE D8- length of key string
0269    :(00) -OSBYTE D9- PRINT line counter
026A    :(00) -OSBYTE DA- number of items in VDU queue (2s complement of
        :                 number required)
026B    :(09) -OSBYTE DB- TAB key value
026C    :(1B) -OSBYTE DC- ESCAPE character
026D    :(01) -OSBYTE DD- input buffer code interpretation status (C0-CF)
026E    :(D0) -OSBYTE DE- input buffer code interpretation status (D0-DF)
026F    :(E0) -OSBYTE DF- input buffer code interpretation status (E0-EF)
0270    :(F0) -OSBYTE E0- input buffer code interpretation status (F0-FF)
0271    :(01) -OSBYTE E1- keyboard buffer code interpretation status (80-8F)
0272    :(80) -OSBYTE E2- input buffer code interpretation status (90-9F)
0273    :(90) -OSBYTE E3- input buffer code interpretation status (A0-AF)
0274    :(00) -OSBYTE E4- input buffer code interpretation status (B0-BF)
        :                 0 - ignore key, 1 - expand as normal key, 2-FF -
        :                 add to base for ASCII code, default on BRK - CA
0275    :(00) -OSBYTE E5- ESCAPE key status (0 - ESC, 1 - ASCII)
0276    :(00) -OSBYTE E6- ESCAPE action
0277    :(FF) -OSBYTE E7- user 6522 VIA IRQ bit mask
0278    :(FF) -OSBYTE E8- 6850 ACIA IRQ bit mask
0279    :(FF) -OSBYTE E9- MOS 6522 VIA IRQ bit mask
027A    :(00) -OSBYTE EA- TUBE presence flag
027B    :(00) -OSBYTE EB- speech processor presence flag
027C    :(00) -OSBYTE EC- character destination status
027D    :(00) -OSBYTE ED- cursor EDITING status
027E    :(00) -OSBYTE EE- not used
```

```
027F        :(00) -OSBYTE EF- not used
0280        :(00) -OSBYTE F0- country code
0281        :(00) -OSBYTE F1- user flag
0282        :(64) -OSBYTE F2- serial ULA control register setting
0283        :(05) -OSBYTE F3- current system clock store pointer
0284        :(FF) -OSBYTE F4- soft key status flag (0 = stable)
0285        :(01) -OSBYTE F5- printer destination
0286        :(0A) -OSBYTE F6- printer-ignore character
0287        :(00) -OSBYTE F7- BRK interception flag - (4C - JMP opcode)
0288        :(00) -OSBYTE F8- user BRK routine address lo
0289        :(00) -OSBYTE F9- user BRK routine address hi
028A        :(00) -OSBYTE FA- not used
028B        :(00) -OSBYTE FB- not used
028C        :(FF) -OSBYTE FC- current language ROM page
028D        :      -OSBYTE FD- last BREAK type (0 - soft, 1 - powerup, 2 - hard)
028E        :      -OSBYTE FE- available RAM (40 - 16K, 80 - 32K)
028F        :      -OSBYTE FF- startup options set by keyboard links
            :                  bits 0-2 screen MODE, bit 3 reset - reverse
            :                  SHIFT+BRK, bits 4,5 disc drive timing parameters
0290        :screen display vertical adjustment
0291        :interlace toggle flag
0292-0296   :system clock 1
0297-029B   :system clock 2
029C-02A0   :countdown interval timer buffer
02A1-02B0   :paged ROM type table
02B1-02B2   :INKEY countdown timer
02B3-02C7   :OSWORD 1 workspace
02B6-02B9   :lo bytes of most recent analogue conversion value
02BA-02BD   :hi bytes of most recent analogue conversion value
02BE        :analogue system flag (OSBYTE 80)
02B3        :maximum input line length (OSWORD 0)
02B4        :minimum input ASCII code (OSWORD 0)
02B5        :maximum input ASCII code (OSWORD 0)
02BF-02C8   :EVENT enable flags
02C9        :soft key expansion pointer
02CA        :first autorepeat count
02CB-02CD   :key-pressed processing workspace
02CE        :ENVELOPE processing software status
02CF-02D7   :buffer busy flags
02D8-02E0   :buffer start indices
02E1-02E9   :buffer end indices
02EA-02EB   :CFS currently resident block input size
02EC        :block flag of currently resident block
02ED        :last character, currently resident block, CFS input file
02EE-02FF   :OSFILE control block workspace
```

0300-037F-VDU variables store and workspace

```
0300-0301  current graphics window left-hand column (pixels)
```

```
0302-0303 :                    bottom row
0304-0305 :                    right-hand column
0306-0307 :                    top row
0308      :current text window left-hand column (characters from text origin)
0309      :                    bottom row
030A      :                    right-hand column
030B      :                    top row
030C-030F :current graphics origin (external coordinates)
0310-0313 :current graphics cursor (external coordinates)
0314-0317 :old graphics cursor (external coordinates)
0318-0319 :current text cursor (X,Y)
031A      :line in current graphics cell containing current graphics point
031B-0323 :VDU queue
031B-031E :graphics workspace
0324-0327 :current graphics cursor (internal coordinates)
0328-0349 :graphics coordinate workspace
0332-0333 :VDU branch routine vectors
034A-034B :text cursor position (6845 address registers)
034C-034D :text window width (bytes)
034E      :bottom of screen memory address hi
034F      :character size (bytes) for current MODE
0350-0351 :screen top left-hand corner 6845 address
0352-0353 :number of bytes/character row
0354      :screen memory size hi
0355      :current screen MODE
0356      :screen display type
0357      :foreground text colour
0358      :background text colour
0359      :foreground graphics colour
035A      :background graphics colour
035B      :foreground graphics PLOT mode
035C      :background graphics PLOT mode
035D-035E :VDU routine vector
035F      :6845 cursor start register old setting
0360      :number of logical colours -1 in current MODE
0361      :number of pixels/byte -1 in current MODE (0=text)
0362      :left colour mask
0363      :right colour mask
0364-0365 :text input cursor coordinates (X,Y)
0366      :MODE 7 cursor character
0367      :font flag
0368-037E :font location address hi
036F-037E :colour palette
```

0380-03DF - CFS/RFS variables store

```
0380-039C :BPUT file header block
039D      :BPUT buffer offset for next byte
039E      :BGET buffer offset for next byte
```

039F-03A6 :unallocated
03A7-03B1 :BGET filename
03B2-03D0 :most recent block header
03B2-03BD : filename
03BE-03C1 : load address
03C2-03C5 : execution address
03C6-03C7 : block number
03C8-03C9 : block length
03CA : block flag
03CB-03CE : RFS EOF +1
03CF-03D0 : checksum
03D1 :sequential block gap
03D2-03DC :sought filename
03DD-03DE :expected BGET file block number
03DF :copy of last read block flag

03E0-03FF- keyboard input buffer

0400- TUBE communications routines

0400 :TUBE language entry point
0403 :TUBE service entry point
0406 :TUBE data transfer entry point

0800-083F- sound queues and workspace

0800-0803 :unallocated
0804-0837 :sound queues
0804-0807 : sound queue occupancy flag
0808-080B : current amplitude, this channel
080C-080F : number of amplitude phases processed
0810-0813 : absolute pitch value
0814-0817 : number of pitch phases processed
0818-081B : number of steps to process
081C-081F : duration
0820-0823 : interval multiplier
0824-0827 : ENVELOPE number / autorepeat parameter
0828-082B : length of note interval left
082C-082F : sync hold parm
0830-0833 : sound chip current pitch setting
0834-0837 : pitch deviation
0838 :number of channels required for sync
0839 :current amplitude step
083A :current target amplitude
083B :number of channels on hold for sync
083C-083F :sound parameter calculation workspace
083D :lo order frequency parameter sent to sound generator
083E :hi order frequency parameter sent to sound generator

0840-0CFF - buffers

0840-084F : sound buffer channel #0
0850-085F : #1
0860-086F : #2
0870-087F : #3
0880-08BF : printer buffer
08C0-08FF : ENVELOPE store (#1-4)
0900-09BF : ENVELOPE store extension (#5-16)
09C0-09FF : speech buffer
0900-09BF : RS432 output buffer
0900-09FF : CFS output buffer
0A00-0AFF : RS432 input buffer
0A00-0AFF : CFS input buffer
0B00-0BFF : soft key buffer
0C00-0CFF : characters E0-FF - font store

0D00-0D9E - NMI routines

0D9F-0DFF - paged ROM variables store

0D9F-0DEF : paged ROM expanded vector set
0DF0-0DFF : paged ROM private workspace address hi
: bit 7 - filing system active flag

-7FFF - screen buffer

3000-7FFF : Modes 0-2
4000-7FFF : Mode 3
5800-7FFF : Modes 4-5
6000-7FFF : Mode 6
7C00-7FFF : Mode 7

```
JMP/JSR  :calling location
address  :address

(00B8)   :FA39 FA6F
(00FA)   :F058
(USERV)  :E659
(BRKV)   :DC51
(IRQ1V)  :DC24
(IRQ2V)  :DD03
(CLIV)   :OSCLI
(BYTEV)  :OSBYTE
(WORDV)  :OSWORD
(WRCHV)  :OSWRCH
(RDCHV)  :OSRDCH
(FILEV)  :OSFILE
(ARGSV)  :OSARGS
(BGETV)  :OSBGET
(BPUTV)  :OSBPUT
(GBPBV)  :OSGBPB
(FINDV)  :OSFIND
(FSCV)   :E031
(EVNTV)  :F0A5
(UPTV)   :E1AA
(NETV)   :E57E
(VDUV)   :C93C
(KEYV)   :F068
(INSV)   :E4B0
(REMV)   :E461
(CNPV)   :E73E
0287     :EAE0
(0332)   :D33F
(035D)   :CBF0 CCF5 D348 D367 D371 D37B
(03C2)   :F31F
0400     :DC08
0403     :E67C
0406     :FBCA FBD0
8000     :DC05
8003     :F17B
C300     :DB2A
C4C0     :D8FD E0C5 FFBC
C55E     :C4EA
C565     :C52F
C568     :C4C9 C526
C588     :C5C5 C65B C6F0 C759 C779 C787 C7AF C945 CAAF
C59D     :C723 D8F2 D902
C5A8     :C9ED CB9D D8F7 D91A
C5C5     :CAAC
C621     :C6F7
C664     :C4E7 D913
```

```
C6AC      :C61C
C6AF      :C607 C7BA CD56
C7A8      :C78C
C839      :CBBB
C88F      :C887
C892      :C868 C879
C89E      :C8E8
C938      :D0AB
C951      :C5B2 D91D
C954      :C5C0 D8E3
C95E      :CBB3
C994      :C614
C9A4      :C6A4
C9BD      :CBBE
C9F6      :F5EB C681 CBCC
CA02      :C574 C732
CA2B      :CA24 CBD1
CA39      :C9E8
CA81      :CA39
CA88      :C71E C9D0
CAD4      :C9AA CE0B CE22
CAE3      :C68B
CB14      :CAE0
CB1D      :C300
CB33      :C8EE
CBC1      :C764
CCF5      :C521 C52C C558
CCF8      :C99A CDB0 CDC6
CD07      :CB2A
CD3F      :C60B C69B
CD6A      :C4CC C529
CD77      :CD8C
CD7A      :C565 D8ED
CDA4      :C619
CDCE      :CDC3
CDDA      :CE5B CE6B
CDDE      :C56E C7AC D434
CDE2      :D0D9 D5D6 D615 D61C
CDE4      :D621
CDE6      :CA85 D503 D644
CDFF      :C6A9
CE2A      :CE1F
CE38      :CDCB CE27
CE5B      :CDA4 CDFF
CE6E      :C7B7 CEB0
CE73      :CDC0 CE1C
CEAC      :C6AC C76D
CEE8      :C72D C7A3
CF06      :C571 C6AF CDAD CE08 CEB3
```

```
CF63        :CAD1
CFA6        :C77E C7BD
CFAD        :C7B4
CFB7        :C4E4
CFBF        :CABF
CFDC        :CAC4
D03E        :C928 CAC9 CBF3 CFBC D7E3
D060        :C991
D0B3        :C7CB CF63 D080
D0D9        :D0D3 D545 D62F
D0DC        :D095
D0DE        :D612 D66D
D0E3        :CF77
D0EB        :D0CD D0D6
D0F0        :D571 D70B D720
D0F3        :D76E
D104        :D4F1 D537 D74D
D10D        :C623 C63A C6B8 C6CF
D10F        :D256 D265 D6C8 D6DF D85F
D128        :D115 D120
D149        :CA4F CA4A D83B
D14D        :D062
D176        :D154 D15E
D1AD        :D157 D16A D16D
D1B8        :C658 C6ED CAA9 CFB4 D548 D5D9
D1D5        :D1C1
D1E3        :D1DF
D1ED        :D0D0
D214        :D204
D2E3        :D3D0
D3D3        :D345 D381
D3ED        :D36E D38A D519
D3FD        :D378 D4D9
D40D        :D1ED
D411        :CA40 D602 D65D
D418        :D411
D42C        :D27E
D459        :D60D D668
D467        :D459
D47C        :C7D2 CA7E D5EE D62C
D480        :D247 D27B
D482        :CF8A CFAA CFB1 D6D5 D8EA
D486        :CF66
D488        :D1BA
D48A        :C72A CAA6 CFA3 D0E0 D23B D43B
D49B        :D2BD D450 D46F D5A2 D5BA
D4AA        :D4BF D506
D4BF        :D0B0
D506        :D0A5
```

```
D545      :D4BC
D54B      :D514 D53D
D574      :D4D4 D4F7
D592      :D4C8 D50B
D5AC      :D4FC F542
D5D5      :D5D0
D5EA      :D0A8
D632      :D5F1 F5FB
D636      :D5F8
D658      :D619 D625
D6A2      :D688
D6A6      :C7E5
D70E      :D771
D774      :D68D D692
D7AC      :D79E
D7C2      :D90D
D808      :D7DC
D839      :C741
D85D      :D0EB D4AA
D85F      :D0E5 D2FD D840
D864      :D2D9 D6DA
D8CE      :C572
D905      :E5A8
D918      :C4D5
DBB8      :DC65
DBE7      :E01E
DC0B      :FFB9
DC16      :DABD DBEB DC49
DCA2      :DE2E
DCF3      :DE7F
DD79      :DE28
DE6E      :DD44
DE8C      :E708
DE8F      :DE69
DEA9      :DB6E DB7F DB84
DEAB      :DBF2
DEB1      :DC56
DEBB      :D717
DEC5      :FFC8
E004      :E000
E009      :DFB7
E00D      :E02C
E031      :DF8F F14E
E039      :DFA0
E03A      :DFA9 DFFB E045 E04E E2AD E367 EA20 EA5A
E043      :E34F
E045      :E08A E330 E35B
E04E      :E327 E342 E354 E360
E07C      :E058
```

```
E07D      :E051 E08F
E08F      :E2B0 E2BB
E0A4      :FFCB
E114      :E0D3
E11E      :C542
E13A      :DD66 E135
E170      :E0ED
E173      :E543 E6E8
E17A      :DC7A
E189      :FB11
E19B      :DE3E
E1A2      :C596 C5A1 E194
E1A4      :E981
E1AD      :F098
E1AE      :E843
E1F8      :E0F3 E12B E864 E89C
E20E      :E24B E2B5
E21F      :E2B8
E267      :F1EA
E275      :F3D7
E2AD      :E260 E2C4 E2D6 E300 E309
E310      :F55E F939
E3A8      :E3D7 E598
E3D1      :E339 E38C
E450      :E46F E4CB
E45B      :DD7D E0E8 ECC5 ECDB ECF1
E460      :DCC1 DCCE DD8D DD92 DD97 E143 E539 ED9A EDAD EDB0 EDE5
          :EDEB
E494      :DCBC DD3F DDF7 DE60 E48C E4AB E4DB E50B FFBF
E4B0      :E1F9 E87C E8A5 E8A9
E4E3      :DFD2 EFB9 FA81
E4F1      :DB2F EFE6
E4F3      :DC8A
E577      :DEED
E57E      :E0B6 E1A7 E79D E96F
E674      :E510
E6A7      :DAB2
E738      :E744
E73B      :E1CB
E741      :DC71 DC8D E173
E7BC      :E7D3
E7DC      :F8C3 F942 FA46 FAE8
E815      :E803 E80B
E86F      :E534
E8C9      :E83B E848
E9C8      :DABA
E9D9      :CB0E CAE3 DB8B
E9EA      :E1FE E9DD
E9F8      :CB6D CB73
```

```
EA00      :CBA6 DD37
EA11      :C8CC
EA1D      :E254 F260
EA1E      :E378 FFC2
EA2F      :E257 E37B F265 FFC5
EA8F      :F274 F439
EA9C      :EFAD EFBE EFCE
EABF      :EA7F EFA6
EAD9      :DB32 DB88
EB03      :ECA2 ECD7 EDAA
EB0A      :EC04 EDC3
EB21      :ED85
EB22      :ED95
EB47      :DE13
EB59      :EC5D
EC59      :EB44
EC60      :DAAA
EC6B      :EB69 EB84
ECA2      :E1B8 EC63
ECD0      :ECFE
ED01      :EC56 EDF3
ED95      :ED13
ED98      :EC9F
EDF7      :EDB4
EE13      :F34E
EE18      :F351 F790
EE3B      :DDAF
EE51      :F58F F780 F7A0
EE62      :EE2C EE40
EE6D      :DD88 DE1F
EE71      :EEBF EED0
EE7A      :EE72 EED5
EE7F      :DB1B DDB5 DDB8 DDBD DDC2 EE66
EEAD      :F3A1
EEBB      :DDA6 EE27 EE3B
EEDA      :F00C
EEEB      :DA22 E203
EF74      :EF8E
EFC1      :EFB2
EFE9      :EF4D EF7B
F00F      :EF0B
F01F      :EF4A F009
F02A      :DA12 EF1D EF2B EF3B EFED F00F F0D4
F055      :DA7D
F058      :E7B9
F065      :DE3B DE78
F068      :E725 E9E4
F0A5      :E4A1
F0AA      :E66E
```

```
F0B4      :F0AE
F0CC      :EFFE F018
F0D1      :EF10
F129      :EFDE F0E3
F12E      :EEFD F129
F135      :DBA6
F137      :DBBB
F140      :DB35
F14B      :F13D
F168      :CD3C DB45 DB51 DB56 DB64 DB9C DC43 DCF5 E025 E277 E7CC
          :EE5A F0BB
F1C4      :F2A4 F30D
F1F6      :F6BA
F246      :F870
F249      :F6FC F8B6
F24D      :F78D FAD3
F25A      :F28B F305 F3F2
F33B      :F9A0
F33D      :F406 F493 F720
F342      :F42C F50E F513
F344      :F32D F421 F46A F4DC
F348      :F335 F651
F46F      :F54A
F471      :F3EF F520
F478      :F3DA F3EC
F496      :F48E F53F
F588      :DCB0 FAE2
F631      :F1CD F40D
F637      :F255 F702
F68B      :E27C FA31
F68D      :E66B
F6AC      :F4F9
F6B4      :F410
F77B      :F22D F359 F373 F6B1
F797      :F747 F759
F7B0      :F5E8 F60F F878
F7D5      :F21A F6D5
F7D7      :F3B2
F7EC      :F2F3 F4BD
F875      :F81C F826 F849
F87B      :F82E F852
F882      :F875 F87D
F884      :F7AD F811 F855 F858
F892      :F7FF F86D
F896      :F804
F898      :F860
F8A9      :FA06
F8B6      :F9FA
F8B9      :F864
```

```
F927        :F91F
F934        :F2BD F462
F96A        :F22A F2F8
F975        :F906
F97A        :F8F7 F90E F92C
F983        :F97F
F991        :F8EB F917 F922 F976
F995        :F739 F884 F89D F95D FADA FBEF
F9B4        :F21D F6D8
F9D9        :F3B7
FA46        :F24E F641 F947 F9B7
FA4A        :F0BE FABF
FA52        :FA65
FA72        :F381 F9CC
FAD6        :FA09 FAE5
FAE8        :F1E7 F246 FA36
FAF2        :F1F6 F465 F542 F6FF F9A3
FAFC        :F338
FB0A        :DA68
FB1A        :F409 F496 F4F5
FB27        :F1C6 F2BA F330 F45F
FB46        :F616 F85B FABB FB0B FB41 FBE7
FB4A        :F80E
FB50        :F5CC F736
FB63        :FBEC
FB69        :F222 F370 F6E3
FB78        :F5C3 F773 F80B
FB7C        :F744 F83B
FB81        :F4B1 F66A F6AE
FB8E        :F4B7 F93C
FB90        :F72F
FB95        :FAF9
FB9C        :F4CF F531 F625
FBB1        :F3E9
FBBB        :F215
FBBD        :F2C2 FAF4
FBC7        :F328
FBD3        :F5F3 F840 FBBD
FBE2        :F2C5 F4BA F93F
(FDFE)      :F055
FF51        :FF00 FF03 FF06 FF09 FF0C FF0F FF12 FF15 FF18 FF1B FF1E
            :FF21 FF24 FF2A FF2D FF30 FF33 FF36 FF39 FF3C FF3F FF42
            :FF45 FF48 FF4B FF4E
OSFIND      :DEE3 E28C E296 F698 F6A2
OSBPUT      :E108
OSBGET      :DED5
OSFILE      :E2AA
OSRDCH      :E924 F960
OSASCI      :DEB4 FA62
```

```
OSNEWL    :DBF7 DBFA DC5F DC62 E96C F967
OSWRCH    :E921 E94B F8CA F8DC F98E FAEF FFE9
OSBYTE    :E371 F9A8 FB99
OSCLI     :DBB0
```

key

```
USERV   = 0200
BRKV    = 0202
IRQ1V   = 0204
IRQ2V   = 0206
CLIV    = 0208
BYTEV   = 020A
WORDV   = 020C
WRCHV   = 020E
RDCHV   = 0210
FILEV   = 0212
ARGSV   = 0214

BGETV   = 0216
BPUTV   = 0218
GBPBV   = 021A
FINDV   = 021C
FSCV    = 021E
EVNTV   = 0220
UPTV    = 0222
NETV    = 0224
VDUV    = 0226
KEYV    = 0228
INSV    = 022A
REMV    = 022C
CNPV    = 022E
OSFIND  = FFCE
OSBPUT  = FFD4
OSBGET  = FFD7
OSFILE  = FFDD
OSRDCH  = FFE0
OSASCI  = FFDA
OSNEWL  = FFE7
OSWRCH  = FFEE
OSWORD  = FFF1
OSBYTE  = FFF4
OSCLI   = FFF7
```

L/U table address	:calling location address
8000	:DAC5
C31F	:CFF6 D008
C32F	:D02E
C333	:C4F0
C354	:F4F6
C3E7	:C705 C9D3
C3EF	:C710 C9CA CA73
C3F7	:CBA3
C3FF	:CB46
C406	:CB58 D8A0
C40D	:C8FF D04C
C414	:CB40
C41B	:D0BD
C41C	:D0B4
C41D	:D0B7
C420	:D0C0
C423	:C815
C43A	:CB4C
C43D	:ED73
C440	:CB64 D92A
C447	:D2A5
C44F	:CB6A
C44B	:CB70
C454	:CBDA
C459	:CB76
C45E	:CB7C D92D
C463	:CB92
C466	:CB89
C469	:CBAB
C46E	:CBB0
C4AA	:D220
C4AB	:D226
C4AE	:D290
C4B2	:D296
C4B6	:CFDE D7D0
C4B7	:CFE9 D7CB
C4BA	:CD29
D93F	:DA5B
D951	:F15B
DF0C	:DAC8 EE2F
DF0E	:DFE9
DF10	:DFC4 DFD7
DF11	:DFF7
DF12	:E004 E00A
E435	:E455
E43E	:E450

```
E447        :E1E7 E1EE E47B E4BC
E5B3        :E7A7
E5B4        :E7A2
E9AD        :E1B3 E696
EB3C        :EB1C ED0F ED80
EDFB        :ED34
EE07        :ED3A
EFAB        :EF91
F075        :EE8A
F077        :EE94
F079        :EEA3
F1A3        :F1BD
F1A4        :F1B9
F581        :F56D
FC00        :FFAA
FD00        :FFAE
FE00        :FFB2
FFB3        :F2CE
FFB7        :F2CB
```

BBC Basic (Basic 1, Basic 2, HiBasic) - description

To provide high level programming facilities, the BBC microcomputer has a BASIC interpreter which converts the high level instructions into machine code for processing by the Machine Operating System. There are several versions of BBC Basic. For 6502-processor-based machines, the code is stored in a paged ROM at 8000-BFFF. When the second processor is in operation, the RESET routines port the language across the Tube interface and load it into the second processor's memory at a relocation address which is specified in the ROM itself.

There are three 6502 versions of BBC Basic - Basic 1, Basic 2 and HiBasic - the last being coded to run at locations B800-F7FF in the second processor. The copyright messages (1981, 1982 and 1983 respectively) embedded in the code at the start of the ROMs provide the easiest way of identifying the respective version. The ROMs differ mainly by relocation of small routines (sometimes accompanied by slight variations in the coding) to reduce the number of opcodes required to achieve the desired macro- operation. Some routines have undergone major recoding using different algorithms, either to increase accuracy (with some of the higher order mathematical functions) or to provide additional facilities (separate ORG and relocation addresses with the built-in assembler).

BBC BASIC programs consist of a set of instructions which are processed serially. In the BBC microcomputer, they are loaded into user RAM at the address defined by PAGE. For orderly execution, these instructions have to be in predetermined format, which is then executed line by line in numerical order. A program line consists of an introductory four-byte block comprising a carriage return, line number HI byte, line number LO byte and an index byte giving the number of bytes in the line followed by the text of the line.

During execution, the program stores a list of the variables which it uses in a variables table which is normally placed at addresses immediately above the program.

RAM below the screen buffer is used by BASIC as a temporary store or stack, which grows and shrinks dynamically according to the demands of the program as it runs. For large programs, the bottom of the BASIC stack may need to grow into the area occupied by the variables table, resulting in the program crashing.

8000 8000 B800

Paged ROM protocol

It is possible with the BBC microcomputer to connect up to 16 alternative ROMs into the addresses 8000- BFFF by a process of

bank switching. These 'sideways' ROMs are switched in and out by the Operating System as they are required. In order to avoid conflicts, Acorn has laid down conventions or protocols which have to be followed. The Operating System requires the entry points to the boot procedures to be at defined addresses. To satisfy these needs, the initial locations contain unconditional jumps to the cold and warm boot routines and these are followed by the ROM title and copyright message.

801F 8023 B823
Warm boot procedures continued

8060 8071 B871
Keyword lookup table

BASIC stores its keywords in the form of characters (or tokens) with an ASCII code of 80 or greater (that is, with the HI bit set). For conversion between keyword and token for program entry and listing, BASIC provides a lookup table. This consists of a list of keywords in alphabetical order. Each keyword is followed by its token and a further hexadecimal byte which serves as a processing parameter. In general this parameter is zero, but in some instances it contains a byte which is used to control a series of conditional branches within the BASIC routines. The significance of the bits of the processing parameter is (bit 0) - tokenise only if next character is not alphanumeric, (bit 1) - switch to statement continuation mode, (bit 2) - switch to statement start mode, (bit 3) - treat following characters as FN/PROC name, (bit 4) - pack following line numbers, (bit 5) - do not tokenise subsequent statements on this line, (bit 6) - add 40 if token is at start of statement. Bit 7 is not used.

With Basic 2, the OPENIN command of Basic 1 which opens a file for random access is replaced by OPENUP. The token remains the same (AD). In Basic 2 and HiBasic OPENIN (with a new token 8E) is used to open a file for read access. COLOR is a legal keyword in HiBasic in addition to COLOUR, both performing the same function and having the same token.

835A 836D BB74
BASIC command addresses LO bytes

83CB 83DF BBE6
BASIC command addresses HI bytes

Certain commands, such as the Boolean operators OR AND and EOR are processed as they are encountered within more complex routines. Others (identified by a token of 8E or higher) have a defined entry point which is found by using the keyword token as an index into the two tables.

843B 8451 BC58

Assembler

BBC BASIC has a built-in assembler which permits the BASIC programmer to include machine language routines within his programs. This assembler starts with three lookup tables. The first two consist of packed 6502 mnemonics and are formed from the least significant five bits of its ASCII code. The assembler routines use the first two lookup tables to derive an index for the third table which contains the corresponding base object code. An example of the derivation for the BRK mnemonic (object code 00) is as follows

opcode	B	R	K
ASCII	42	52	4B
binary	0100 0010	0101 0010	0100 0111
LS 5 bits	0 0010	1 0010	0 0111
packed + leading zero	0000 1010	0100 0111	
HEX	0D	4B	

The first byte in the three lookup tables is 4B, 0D and 00, respectively.

The assembler routines follow the lookup tables. They make use of the fact that 6502 opcodes are grouped, with similar operations having the same base object code. Basic 2 and HiBasic have the facility that they can be assembled at a different address from the one at which the code will run.

8657 8696 BEB8

Error handling routines

If the computer is unable to follow the instructions it has been given, it stops operation and follows a standard error handling procedure which, unless it has been modified by the programmer, involves displaying an error message and performing a warm boot. Entry is initiated by error routines located at addresses throughout the ROM, usually in close proximity to the routines which most frequently give rise to that particular error. For example, the "RENUMBER space" error routine is adjacent to the RENUMBER command entry point. Errors, such as "Syntax error", which could arise for a wide variety of reasons, are processed via an unconditional jump located near the offending routine.

The general format for an error routine is

```
BRK                    ; break opcode
HEX 00                 ; single hexadecimal byte error code
ASC 'Error message'
HEX 00                 ; hexadecimal terminator byte (00)
```

Error codes are set out in the User Guide and have the value zero for untrappable errors.

87FD 887C C09E

Line editing routines

The outside world is decimal, but within the computer, numbers are processed in binary or hexadecimal format. To complicate matters, communication between the outside world and the computer is conducted in ASCII (American Standard Code for Information Interchange). Keywords are stored as tokens (single hexadecimal bytes), whilst numbers are either categorised as integers or as floating point variables, each having a different structure. The position of punctuation symbols has a special significance - the computer expects to find a closing bracket at the end of an array or a parenthesis at the end of a literal string.

8A3D 8AB6 C2D4

Program control commands

BASIC includes commands (OLD NEW END STOP) which condition the computer in different ways - to run a program, to receive new input and so on. These commands test line termination, initialise RAM and set the appropriate internal pointers.

In Basic 2 and HiBasic, the STOP command has been recoded as an untrappable error.

8AED 8B7D C398

Statement parsing routines

Page 0 has two sets of pointers (0B,0C index 0A and 19,1A index 1B) which mark the positions in memory from which the computer is currently retrieving instructions or data. These are the primary and secondary text buffers. Sometimes both sets of pointers mark the same address, whilst at other times they have different settings. A typical case is where the primary text pointers keep the place in the program whilst the secondary pointers indicate the position of a variable.

8BC3 8B73 C38E

*** command handler**

Whilst the BASIC ROM is active, it is sometimes necessary to pass instructions to the Operating System or other sideways ROMs. A simple routine traps these commands, which are identified by an asterisk at the start of the line. When the parsing routines identify this asterisk, control is passed to the Operating System command handling routine via the OSCLI entry vector.

8CC5 8D2B C549

PRINT formatting

As well as the TAB(and SPC commands, BASIC permits the programmer to predetermine the format for numerals (exponen-

tial, fixed number of decimal places, etc.) The parameters which control this processing are stored in the special four-byte system integer variable (@%) which is located at 0400-0403.

8E57 8EBD C6DB

Screen clearance

CLS and CLG are performed by placing the appropriate VDU number in the processor's accumulator and calling the Operating System OSWRCH routine to send the appropriate VDU command.

8E5E 8EC4 C6DF

Machine code calls

For machine code calls, BASIC sets up a parameter block in Page 6. A subroutine primes the processor registers from the least significant byte of the system integer variables A%, C%, X% and Y%. The related USR routines are tucked into a small block of memory at ABFB, ABD2 and E39D for the three Basics respectively.

8ECE 8F31 C74C

Program line editing

A series of editing routines perform the program manipulation for the DELETE, RENUMBER and AUTO commands. Program lines follow one another in numerical order, commencing at PAGE. The structure of a program line is

<CR>| Line# HI| Line# LO| Line length| Text of line

so, if a new line is inserted or an existing one deleted, the other lines have to be moved to accommodate the change. The RENUMBER command searches through the program, finding the line number references and changing them. Although, at the start of a line, the number is stored in hexadecimal form, this is not possible within the text of the line, since the BASIC interpreter would construe hexadecimal bytes with the bit 7 set as tokens. If, therefore, the line contains a reference to another line, for example, in a GOTO statement, the reference is stored in a packed format which has to be unpacked, processed and repacked.

9092 90DF C8FA

Allocation of memory space (DIM)

BASIC permits the storage of data in the form of multi-dimensional arrays. The space for this has to be reserved when the program is first loaded and does not change dynamically whilst the program is running. These routines calculate the amount of space which will be required to accommodate any given array.

The DIM command can also be used in a different syntax form

to reserve memory space for other purposes such as locating a machine language routine.

9212 925D CA78

Setting Basic parameters

The Operating System stores default values for parameters such as HIMEM, LOMEM and PAGE in specific locations in Page 0. The boundary values are altered by changing the values at these addresses.

9243 9295 CABO

TRACE

When running a program, a line number can be displayed as the corresponding line is executed. A flag is set to determine which course to follow. The TRACE ON and TRACE OFF commands set and reset the flag.

932F 937A CB95

Graphics and sound

Commands which control graphics and timing and sound are controlled by the Operating System. BASIC assembles the initial parameters into the required format and sends the data to the Operating System by standard OS calls (OSBYTE OSWORD and OSWRCH).

941B 945B CC76

Set up variables, functions and procedures

Numeric variables are stored either as integers which occupy four bytes of memory or as floating point variables which comprise a single-byte exponent and a four-byte mantissa. In both cases, the sign of the variable is determined by bit 7 of the most significant byte. Strings comprise up to FF consecutive ASCII characters. All variable types may be assembled into multidimensional arrays. Program modules may be incorporated into procedures or functions which can be called many times during the execution of a program.

In order to ascertain the current setting of a variable, the program refers to a variables table which is normally stored in the user program area of RAM immediately above the program itself. So that the program can locate any particular variable, apart from the system integers, which are stored in Page 4 locations from 0400-046B, all those having the same initial letter are stored in a single chain. Each variable holds the address of the next variable in the chain, whilst the starting address of the chain is stored in alphabetical sequence in Page 4. The format of the data stored in the variables tables is

NextVAR HI	NextVAR LO	Text of variable name
	00	variable descriptor

The first letter of the variable name is omitted, since all variables in a given chain will have the same initial. For a numeric variable, the descriptor will consist of four or five bytes (depending on whether it is an integer or floating point) containing the current value. For a string, the descriptor occupies four bytes containing the address in memory where the string is located (2 bytes), the number of bytes reserved for text (1 byte) and the current length of the string (1 byte). For arrays the descriptor is more complex. It contains a two-byte number for each element in the dimension, a single descriptor header byte (1 byte plus total of the number of dimension element bytes) as well as a descriptor in characteristic form for each element (integer, floating point or string) in the array. For functions or procedures, the descriptor consists of a two-byte address of the location of the FN/PROC in memory. For the final variable in a chain, a terminator character (00) is inserted in the next-variable address HI byte position.

955D **9595** **CDBO**

Indirection routines

Memory can be addressed directly by means of indirection routines which perform the equivalent of PEEK and POKE in other forms of BASIC. The routines can be used in two ways, either to insert data into memory or to ascertain what data are currently there. The entry point of the routine varies according to whether data is being written or read.

993D, 99B8 **996B,99B9** **D185, D1D3**

Hexadecimal/decimal lookup tables

Amongst the processes that can be accomplished more easily by lookup tables, is the decimal output of hexadecimal numbers. BASIC has lookup tables for various purposes distributed sporadically throughout the ROM.

9979 **99BE** **D1D8**

Arithmetic routines

As it includes higher functions, BASIC can use them recursively for other purposes. For example, natural logarithms are used in power calculation for the A operator.

9E81 **9E90** **D6A6**

Print formatting

The user can select different formats for screen output. Features such as the number of TAB spaces, the number of places after the decimal point and the numeric variable format are stored in the

four bytes of the system integer variable @%. A flag(15) is tested to determine whether output of numerals should be in decimal or hexadecimal form.

A0D2 A0E1 D8F7

Arithmetic routines

A large block of the ROM is taken up by routines which convert between integer and floating point format and perform multiplication and division operations by means of a series of additions, subtractions, sign changes, comparisons and shifts (a limitation imposed by the nature of the instruction set of the 6502 microprocessor).

A6C9 A6BE DED4

Higher order mathematical functions

It is impractical to have a built-in set of tables for logarithms, exponentials and trigonometric functions, since these would required far more memory than is available. BASIC therefore calculates the values as they are required, using a converging power series for this purpose. The power series for some functions converge more rapidly than those for others and fewer iterations are required. The coefficients for the power series are stored for each function as a consecutive table of floating point numerals. At the head of each lookup table is an iteration index giving the number of coefficients in that particular series. Located in the same region of memory are a number of other constants, such as PI, and 'e' and 57.298 (degree/radian conversion factor) which are used in calculations and can be called up by BASIC commands (DEG, RAD, PI etc.).

In performing the calculations, BASIC tests the size of the numbers and, where appropriate, makes approximations, such as equating small angles and their sines and tangents.

In Basic 2 and HiBasic, the SIN/COS calculation has been completely recoded to improve accuracy.

AB56 AB33 E349

Routines requiring MOS calls

Located at the end of the higher mathematical function routines is a group of routines (ADVAL POINT(POS VPOS) which utilise operating system calls OSBYTE and OSWORD. These calls are reached through standard locations FFF4 and FFF1,

ABCD ABA8 E377

Routines which involve a fixed constant multiplication

Routines which involve a fixed constant multiplication
Conversion between natural and base-10 logarithms, or degrees and radians, for example, involves the multiplication of a floating

point numeral, which may be entered at the keyboard or calculated in a program, by a constant value which is stored in ROM.

AC55 AC2F E3FA
Standard buffer commands

A number of commands (VAL INT ASC TRUE NOT) are effected simply by manipulation of data in the variable buffers. For example, the INT command performs a preliminary check of the variable type to confirm that it is a numeral and not a string, then checks whether it is a floating point variable, and, if it is, positions the binary point appropriately, and finally moves the variable from the floating point buffer to the integer buffer. For the NOT command, each byte in the integer buffer is complemented.

AE1B ADEC E5F2
Process statement

Parsing of the primary text buffer continues at this address by checking for the + and - operators and then carrying on to process variables or BASIC command tokens.

AEEF AECO E6BF
'Right-hand' commands

The indirection routines are divided into two groups, the first being used to set memory contents and a second, complementary group which polls the specified addresses and returns the values found. Several commands (TOP COUNT PAGE LOMEM etc.) are treated in a similar manner and, when their setting is interrogated, the value is transferred to the integer buffer.

AF39 AF0A E706
RaNDom number processing

Random number processing is based on a five-byte buffer (0D-11) in Page 0. The contents of the buffer are shuffled and then transferred to the integer buffer, with normalisation if required. The shuffling algorithm was changed fundamentally for HiBasic

AFDC AFAD E7B1
String handling commands

BASIC is endowed with a wealth of commands for manipulation of strings which are stored in Page 6 of the computer's memory. Substring commands (RIGHT$ LEFT$ MID$) are performed by means of a set of secondary text buffer pointers which are used to move the position of the string within the buffer. The current length of the string is an important parameter and this is stored temporarily in Page 0.

B12D B0FE E902
FuNctions and PROCedures
The FN and PROC commands provide a facility for introducing library subroutines into a BASIC program. Within a function or procedure it is possible to declare a variable as LOCAL and to give it a different value from that which it has elsewhere in the program. The way that this works is by using the processor stack as a temporary store for the global variables. The 6502 stack is saved on the BASIC stack at the start of the FN/PROC and restored when the main program is resumed.

B3F6 B3C5 EBC9
BRK and error handling routines
When the BASIC ROM is enabled, the Operating System BRKV at 0202,0203 points to the location of the routine which initialises the computer. When an error is encountered, the routine prints an error message which it obtains from ASCII-encoded bytes stored in the ROM immediately following the BRK opcode which caused the reset.

B461 B44C EC50
Sound commands
The SOUND and ENVELOPE commands are essentially Operating System commands. The corresponding BASIC routines just set up parameter blocks and prime the processor registers before passing control to the MOS.

B4DD B4A0 ECA4
Text processing for listing
Both LIST and LISTO share a common token, so BASIC looks for the terminal '0' before making the appropriate branch. The LISTO command sets a Page 0 flag which LIST checks to determine whether or not to insert leading spaces.

B681 B668 EE6B
FOR/NEXT loops
The parameters of active FOR/NEXT loops are stored in blocks of fifteen bytes in Page 5. Since up to ten loops may be nested, this requires 150 bytes in total. Parameters stored in each block include the address of the loop variable, the setting of STEP and the pointers to the next FOR/NEXT loop. As with arithmetic operations, BASIC tests to see if it is possible to process the loop using integer variables to speed operations.

B8B4 B888 F08B
GOSUB
GOSUBs may also be nested. Space is reserved in Page 5 for up to

twenty-six GOSUB return addresses.

B934 B915 F118
ON command handling

The ON command looks for a following ERROR token and, if it is found, branches to the error vector setting or resetting routines. If the token is not found, it seeks the GOTO or GOSUB tokens and proceeds accordingly.

B9ED B9CF F1D2
INPUT and INPUT#

BASIC uses the Operating System OSBGET call to process these commands. To enter a floating point variable, use is made of a temporary store in Page 4 between the system integer variables and the start of the normal variable address chains.

BB00 BAE6 F2E9
DATA handling commands

DATA are stored in the lines of the program at which they are entered. When these data are required by a DATA statement, BASIC sets pointers which are moved as each item is retrieved. The RESTORE command resets the pointers.

BBC5 BBA6 F3A9
REPEAT/UNTIL loops

Up to twenty REPEAT/UNTIL loops may be nested. The loop start addresses are stored in the remaining space on Page 5. When the space allocated for parameter storage in Page 5 is exceeded by the number of nested GOSUBs or FOR/NEXT or REPEAT/UNTIL loops, the corresponding error routine is triggered.

BC17 BBFC F3FF
Program line input

The PROMPT character > is generated by loading the corresponding ASCII code (3E) into the accumulator prior to transmission to the current output stream. Filters are set to reject control characters and to limit the maximum input line length to 238 characters. An auxiliary set of pointers is used to insert the line into the program.

BD38 BD20 F523
Program initialisation

Prior to RUNning a program, LOMEM and variables pointers are set to TOP and the variables chain initial pointers at the top half of Page 4 are all set to zero. Loop counters are also reset.

BD69 BD51 F554
BASIC stack handling
RAM below the screen buffer is used for dynamic storage during the running of a program. Before moving the stack pointers down, BASIC checks to see that there is sufficient memory and that it will not conflict with the top of the variables tables which moves up as the program proceeds.

BEDE BED2 F6D5
File handling routines
Filing systems such as the disc filing system, Econet and the Telesoft systems handle files by routines stored in other sideways ROMs. The cassette filing system routines are stored in the Operating System ROM. For file handling commands, BASIC sets up a parameter block and passes the command to the appropriate ROM by means of an Operating System call.

BBC Basic - RAM usage
The current addresses of PAGE, LOMEM and the other benchmarks, such as the bottom of the BASIC stack, are stored in predetermined locations in Page 0 (0000-00FF). Other Zero Page locations are used as flags or loop counters or serve as work areas for temporary storage during calculations. For example, 2A-2D serves as a buffer for computation using integer variables, whilst 2E-35 and 3B-42 are buffers for floating point processes.

Page 4 is used for variable storage and retrieval. The system integers (@%-Z%) are actually stored in 0400-046B and the upper half of the page is used for pointers to variable chains.

Parameters for nested loops and GOSUBs occupy the whole of Page 5 (0500-05FF). Page 6 (0600-06FF) serves alternately as a string buffer or a store for a CALL parameter block. The input buffer is located in Page 7 (0700-07FF). Since a page consists of 100 bytes, this sets an upper limit to the size of strings or input.

The BASIC program itself is loaded from PAGE upwards to TOP. (PAGE is normally set at the first free memory address after all Operating System and Paged ROM RAM requirements have been met.) When the program is running, it constructs a variables table from LOMEM (normally equal to TOP) upwards and a BASIC stack from HIMEM (normally equal to the next free address below the screen buffer) downwards. If BASIC programs are large, the BASIC stack and variables table may require more memory than is available between LOMEM and HIMEM, resulting in a fatal crash.

In the following table, the entry points of corresponding routines are listed. Identical coding (apart from location changes)

is indicated with "=" and minor changes with ":". Major changes in algorithms are shown as "*". The indication refers only to the immediate subroutine. A macro-routine may include several subroutines, some of which incorporate changes whilst others do not. It is therefore advisable to step through each macro to verify the particular circumstances of an individual case.

paged ROM protocol and initialisation

```
8000 * 8000 = B800
8009 : 8009 : B809
 NE    801F * B81F
801F : 8023 = B823
```

BASIC keyword lookup tables

```
806D : 8071 : B871
835A * 836D * BB74
83CB * 83DF * BBE6
```

assembler

```
843B : 8451 = BC58
8475 : 848B = BC92
84AE = 84C5 = BCCC
84E6 = 84FD = BD04
84ED = 8504 = BD0B
84F1 * 8508 : BD0F
8543 = 8581 = BDA0
85A1 = 85BA = BDDB
85CB = 85E1 = BE02
85D8 : 85F1 = BE12
85EE = 8607 = BE28
8607 = 8620 = BE41
8612 * 862B = BE4C
8634 = 8673 = BE94
8652 = 8691 : BEB2
8657 = 8696 = BEB8
8669 : 86A8 = BECA
8685 = 86C5 = BEE7
868C = 86CC = BEEE
8697 = 86D7 = BEF9
86BB = 86FB = BF1D
86CD = 870D = BF2F
86F7 : 8735 = BF57
86FA : 8738 = BF5A
875C = 879A = BFBC
87B2 = 87ED = C00F
87D8 = 8815 = C037
87E4 = 8821 = C043
87EA = 8827 = C049
87EF = 882C = C04E
87F2 = 882F = C051
87F5 = 8832 = C054
 NE    8867 = C089
```

line editing routines

```
87FD = 887C = C09E
8819 : 8897 = C0B9
885E : 88DA = C0F9
8871 = 88EC = C10B
8878 : 88F3 = C112
88AB = 8926 = C145
88BB = 8936 = C155
88C5 = 893D = C15C
88CA = 8942 = C161
88CC = 8944 = C163
 NE    894B = C16A
88D3 = 8951 = C170
88D7 : 8955 = C174
88D9 : 8957 = C176
88E3 : 8961 = C180
8901 = 897C = C19B
8913 = 898C = C1AB
892A = 89A3 = C1C2
893C = 89B5 = C1D4
8949 = 89C2 = C1E1
8959 = 89D2 = C1F1
8966 = 89DF = C1FE
8973 = 89EC = C20B
897F = 89F8 = C217
8985 = 89FE = C21D
899F = 8A18 = C237
89AC = 8A25 = C244
89BE = 8A37 = C256
89F9 = 8A72 = C291
8A13 = 8A8C = C2AB
8A1E = 8A97 = C2B6
8A35 = 8AAE : C2C1
8A29 = 8AA2 = C2C8
```

program control commands, statement parsing routines

```
8A3D = 8AB6 = C2D4
8A50 : 8AC8 = C2E6
8A59 * 8AD0 = C2EE
8A62 * 8AD3 = C2F1
8A7D = 8ADA = C2F5
8A80 : 8ADD = C2F8
8A96 : 8AF3 = C30E
8A99 : 8AF6 = C311
8AAB : 8B0B = C323
8AEA = 8B41 = C35C
8BA7 = 8B44 = C35F
```

8BAA = 8B47 = C362
8BBC = 8B59 = C374
8B60 = 8B60 = C37B
8BC3 = 8B73 = C38E
8AED = 8B7D = C398
8AF7 : 8B87 = C3A2
8B07 = 8B96 = C3B1
8B09 = 8B98 = C3B3
8B0C = 8B9B = C3B6
8B14 = 8BA3 = C3BE
8B22 = 8BB1 = C3CC
8B30 = 8BBF = C3DA
8B57 = 8BE4 = C3FF
8BA4 = 8C0B = C426
8B7E = 8C0E = C429
BBD0 : 8C1E = C439
8BD3 : 8C21 = C43C
8C18 = 8C84 = C49F
8CB2 = 8C97 = C4B2
8C36 = 8CA2 = C4BD
8C4B * 8CB7 = C4D2
8C5B = 8CC1 = C4DC
8C88 = 8CEE = C509
8C9D = 8D03 = C51E
8CC5 : 8D2B = C549
8CE6 = 8D4B = C569
8CF2 = 8D57 = C575
8CFF = 8D64 = C582
8D12 = 8D77 = C595

display control routines

8D33 = 8D9A = C5B8
8D3F = 8DA6 = C5C4
8C54 = 8DBB = C5D9
8D6B = 8DD2 = C5F0
8DA7 = 8E0E = C62C
8DBA = 8E21 = C63F
8DBD = 8E24 = C642
8DD9 : 8E40 : C65E
8DF2 = 8E58 : C676
8E01 = 8E67 = C682
8E04 = 8E6A = C685
8E0A = 8E70 = C68B
8E24 = 8E8A = C6A5
8E32 = 8E98 = C6B3
8E3E = 8EA4 = C6BF
8E57 = 8EBD = C6D8
8E5E = 8EC4 = C6DF
8E6C : 8ED2 = C6ED
8E7D : 8EE0 = C6FB
8EB8 = 8F1B = C736
8EBB = 8F1E = C739
8ECB = 8F2E = C749

program line numbering

8ECE = 8F31 = C74C
8F06 = 8F69 = C784
8F4A : 8F92 = C7AD
8F42 : 8F9A = C7B5
8F37 : 8FA3 = C7BE
8F78 = 8FD6 = C7F1
8F2F = 8FDF = C7FA
8FF0 : 9043 = C85B
901E : 9071 : C889
903C : 908D = C8A8
9052 = 909F = C8BA
908F = 90DC = C8F7

array dimensioning routines

9092 : 90DF = C8FA
90D5 = 9127 = C942
90DD = 912F = C94A
9119 = 916B = C986
916C = 91B7 = C9D2
91C0 = 920B = CA26
91CA = 9215 = CA30
91CD = 9218 = CA33
91D7 = 9222 = CA3D
91E6 = 9231 = CA4C
91EB = 9236 = CA51
920F = 925A = CA75

set BASIC parameters

9212 = 925D = CA78
9224 = 926F = CA8A
9239 = 9283 = CA9E
9326 : 928D = CAA8
9243 = 9295 = CAB0
9265 : 92B7 = CAD2
9272 = 92C0 = CADB
927B = 92C9 = CAE4
NE 92DA = CAF5
928C = 92DD = CAF8
9292 : 92E3 = CAFE

929C = 92EB = CB06
929F = 92EE = CB09
92A1 = 92F0 = CB0B
92A9 = 92F7 = CB12
92AC = 92FA = CB15
92AF = 92FD = CB18

PROCedure and LOCAL variable processing

92B6 = 9304 = CB1F
92CD = 931B = CB36
92D5 = 9323 = CB3E
92F3 = 9341 = CB5C
9305 = 9353 = CB6E
9310 = 9356 = CB71
931F = 9365 = CB80
9308 = 936B = CB86
9352 = 9372 = CB8D

BASIC VDU command handlers

932F = 937A = CB95
9346 = 938E = CBA9
935A : 939A = CBB5
9397 = 93DA = CBF5
93A1 = 93E4 = CBFF
93A5 = 93E8 = CC03
93AE = 93F1 = CC0C
93BA = 93FD = CC18
93EA = 942A = CC45
93EF = 942F = CC4A
9413 = 9453 = CC6E
9416 = 9456 = CC71

variable processing and setup routines

941B = 945B = CC76
9429 = 9469 = CC84
9439 = 9479 = CC94
945A = 949A = CCB5
9473 = 94B3 = CCCE
94A1 = 94E1 = CCFC
94AD = 94ED = CD08
94BC = 94FC = CD17
94D6 = 9516 = CD31
94F7 = 9531 = CD4C
94FF = 9539 = CD54
9507 = 9541 = CD5C
9519 = 9553 = CD6E
951F = 9559 = CD74
9521 = 955B = CD76
9545 : 957F = CD9A
9548 : 9582 = CD9D
955D : 9595 = CDB0
9571 = 95A5 = CDC0
9573 = 95A7 = CDC2
957C = 95B0 = CDCB
958B = 95BF = CDDA
9595 = 95C9 = CDE4
95A1 = 95D5 = CDF0
95A9 = 95DD = CDF8
95CB = 95FF = CE1A
95E7 = 961B = CE36
960D = 9641 = CE5C
9618 = 964C = CE67
962D = 9661 = CE7C
9647 = 967B = CE96
964B = 967F = CE9A
966B = 969F = CEBA
9672 = 96A6 = CEC1
967B = 96AF = CECA
9695 = 96C9 = CEE4
96A3 = 96D7 = CEF2
96AB = 96DF = CEFA
96C6 = 96F7 = CF12
977C = 97AD = CFC8
9789 = 97BA = CFD5
97A0 = 97D1 = CFEC
97AC = 97DD = CFF8
97AE = 97DF = CFFA
97BA = 97EB = D006
97D6 = 9807 = D022
97E2 = 9813 = D02E
97F0 = 9821 = D03C
9839 = 982A = D045
9848 = 9838 = D053
97FA = 9841 = D05C
9802 = 9849 = D064

statement parsing and IMMEDIATE mode handler

9805 = 984C = D067
980B = 9852 = D06D
9810 = 9857 = D072
9812 = 9859 = D074
981A = 9861 = D07C
9826 = 986D = D088
9830 = 9877 = D092

9834 = 987B = D096
9851 : 9880 = D09B
9861 = 9890 = D0AB
987D = 98AC = D0C7
988D = 98BC = D0D7
9890 = 98BF = D0DA
9893 = 98C2 = D0DD
98B2 = 98E1 = D0FC
98B4 = 98E3 = D0FE
98C2 = 98F1 = D10C
98D6 = 9905 = D120
98F1 = 991F = D13A
98F5 = 9923 = D13E
993D : 996B = D185
9942 = 9970 = D18A
99A6 = 99A7 = D1C1
99B8 = 99B9 = D1D3
9979 : 99BE = D1D8
99BD = 99E8 = D202
9A11 = 9A39 = D253
9828 = 9A50 = D26A
9A37 = 9A5F = D279
9A3A = 9A62 = D27C
9A6B = 9A93 = D2AD
9A72 = 9A9A = D2B4
9A75 = 9A9D = D2B7
9A76 = 9A9E = D2B8
9A8B = 9AB3 = D2CD
9ABF = 9AE7 = D301
9AF7 = 9B1D = D333
9B03 = 9B29 = D33F
9B14 = 9B3A = D350
9B2F = 9B55 = D36B
9B45 = 9B6B = D381
9B4C = 9B72 = D388
9B54 = 9B7A = D390
9B76 = 9B9C = D3B2
9B82 = 9BA8 = D3BE
9B9A = 9BC0 = D3D6
9BA5 = 9BD4 = D3EA
9BB9 = 9BDF = D3F5
9BD4 = 9BFA = D410

arithmetic routines

9BDD = 9C03 = D419
9BEF : 9C15 = D42B
9C1D = 9C42 = D458
9C29 = 9C4E = D464
9C36 = 9C5B = D471
9C52 = 9C77 = D48D
9C63 = 9C88 = D49E
9C66 = 9C8B = D4A1
9C76 = 9C9B = D4B1
9C7C = 9CA1 = D4B7
9C82 = 9CA7 = D4BD
9C90 = 9CB5 = D4CB
9D9D = 9CC2 = D4D8
9CBC = 9CE1 = D4F7
9CD5 = 9CFA = D510
9CE9 = 9D0E = D524
9CF8 = 9D1D = D533
9CFB = 9D20 = D536
9D07 = 9D2C = D542
9D14 = 9D39 = D54F
9D17 = 9D3C = D552
9D38 : 9D5C = D572
9D98 = 9DBB = D5D1
9D9A = 9DBD = D5D3
9DA8 = 9DCB = D5E1
9DAB = 9DCE = D5E4
9DAE = 9DD1 = D5E7
9DB1 = 9DD4 = D5EA
9DC2 = 9DE5 = D5FB
9DDE = 9E01 = D617
9DE7 = 9E0A = D620
9DFA = 9E1D = D633
9DFD = 9E20 = D636
9E12 : 9E35 = D64B
9E4D : 9E6F = D685
9E6A : 9E88 = D69E

PRINT formatting

9E81 = 9E90 = D6A6
9EA1 = 9EB0 = D6C6
9EB9 = 9EC8 = D6DE
9EC2 = 9ED1 = D6E7
9ED0 : 9EDF = D6F5
9EEA : 9EF9 = D70F
9F0E = 9F1D = D733
9F16 = 9F25 = D73B
9F22 = 9F31 = D747
9F8D = 9F9C = D7B2
9F91 = 9FA0 = D7B6
9FCC = 9FDB = D7F1
9FD9 = 9FE8 = D7FE
A006 = A015 = D82B

numeric input and floating point manipulation

A031 = A040 = D856
A043 = A052 = D868
A055 = A064 = D87A
A057 = A066 = D87C
A063 = A072 = D888
A06C = A07B = D891
A07C = A08B = D8A1
A08A = A099 = D8AF
A091 = A0A0 = D8B6
A099 = A0A8 = D8BE
A0D2 = A0E1 = D8F7
A102 = A111 = D927
A110 = A11F = D935
A12A = A139 = D94F
A131 = A140 = D956
A13C = A14B = D961
A149 = A158 = D96E
A161 = A170 = D986
A169 = A178 = D98E
A188 = A197 = D9AD
A196 = A1A5 = D9BB
A1CB = A1DA = D9F0
A1DE = A1ED = DA03
A1E5 = A1F4 = DA0A
A1EC = A1FB = DA11
A1F9 = A20B = DA1E
A1FC = A20B = DA21
A20F = A21E = DA34
A211 = A220 = DA36
A230 = A23F = DA55
A233 = A242 = DA58
A23E = A24D = DA63
A295 = A2A4 = DABA
A2AF = A2BE = DAD4
A2D7 = A2E6 = DAFC
A2DE = A2ED = DB03
A2F4 = A303 = DB19
A327 = A336 = DB4C
A33F = A34E = DB64
A36E = A37D = DB93
A372 = A381 = DB97
A376 = A385 = DB9B
A378 = A387 = DB9D
A37E = A38D = DBA3
A3A3 = A3B2 = DBC8
A3A6 = A3B5 = DBCB
A3F2 = A3E4 = DBFA
A3F5 = A3E7 = DBFD
A40C = A3FE = DC14
A41A = A40C = DC22
A44A = A43C = DC52
A460 = A450 = DC66
A463 : A453 = DC69
A47A = A46C = DC82
A49F = A486 = DC9C
A4BE = A4B0 = DCC6
A4C4 = A4B6 = DCCC
A4D5 = A4C7 = DCDD
A505 = A4D0 = DCE6
A4DE = A4D6 = DCEC
A4F0 = A4E8 = DCFE
NE A4FD = DD13
NE A500 = DD16
NE A505 = DD1B
A513 : A50B = DD21
A598 = A590 = DDA6
A5BF = A5B7 = DDCD
A5EE = A5E3 = DDF9
A611 = A606 = DE1C
A61E = A613 = DE29
A661 = A656 = DE6C
A664 = A659 = DE6F
A667 = A65C = DE72
A677 = A66C = DE82
A681 = A676 = DE8C
A687 = A67C = DE92
A691 = A686 = DE9C
A6A4 = A699 = DEAF
A6B0 = A6A5 = DEBB
A6BB = A6AD = DEC3
A6C6 = A6BB = DED1

higher order mathematical functions

A6C9 = A6BE = DED4
A6F2 = A6E7 = DEFD
A6FC = A6F1 = DF07
A715 = A70A = DF20
A7BE = A7A9 = DFBF
A7B4 : A7B4 = DFCA
A7C9 : A7B7 = DFCD
A7EF = A7E9 = DFFF
A7F3 = A7ED = E003
A7F7 = A7F1 = E007
A7FB = A7F5 = E00B
A804 = A7FE = E014

A807 = A801 = E017
A80E = A808 = E01E
A81A * A814 = E02A
A85B = A86E = E07F
A860 * A873 = E084
A889 = A897 = E0A8
A8C6 = A8D4 = E0E5
A8CC : A8DA = E0EB
A8DE : A8EA = E0FB
A8FE = A8FE = E10F
A907 : A907 = E118
A90A : A90A = E11B
NE A916 = E127
A91D : A91B = E12C
A929 : A927 = E138
A932 = A936 = E147
A956 = A95A = E16B
A98D * A98D = E19E
A998 * A998 = E1A9
A99A * A99E = E1AF
A9A1 * A9AA = E1BB
NE A9B1 = E1C2
A9BF : A9C3 = E1D4
A9DA : A9D3 = E1E4
AA45 = AA38 = E249
AA5C = AA48 = E259
AA60 = AA4C = E25D
AA69 = AA55 = E266
AA6D * AA59 = E26A
AA72 * AA5E = E26F
AA77 = AA63 = E274
AA7C = AA68 = E279
AA81 = AA6D = E27E
A856 = A869 = E283
AA86 * AA72 = E288
AAB4 = AA91 = E2AA
AAB7 = AA94 = E2A7
AACF = AAAC = E2C2
AAF4 = AAD1 = E2E7
AAFD = AADA = E2F0
AB07 = AAE4 = E2FA
AB0C = AAE9 = E2FF
AB35 = AB12 = E328
AB41 = AB25 = E33B

miscellaneous parameters determination

AB56 = AB33 = E349
ACF7 = ACD1 : E357

AB92 = AD6D : E368
AB9B = AB76 = E36E
ABCD = ABA8 : E377
ABD6 = ABB1 = E37E
ABE7 = ABC2 : E38F
ABF0 : ABCB = E396
ABFB = ABD2 = E39D
AC12 : AEB9 = E3B1
AC49 = AC23 = E3EB
AC0F = ABE6 = E3F7
AC55 = AC2F = E3FA
AC5A = AC34 = E3FF
AC99 = AC73 = E43C
AC9E = AC78 = E441
ACC4 = AC9E = E464
ACD3 = ACAD : E473
ACC1 = AC9B : E47D
ACDE = ACB8 = E480
ACEA = ACC4 : E48C
AEF9 = AECA : E499
ABAD = AB88 : E4A6
AB64 = AB41 : E4C0
AD08 : ACE2 = E4E9
AD8D = AD6A = E571
AD94 = AD71 = E578
AD99 = AD77 = E57E
ADA0 = AD7E = E585
ADB1 = AD8F = E596
ADB5 = AD93 = E59A
ADDC = ADAD = E5B4
ADF8 = ADC9 = E5D5
AE18 = ADE9 = E5EF
AE1B = ADEC = E5F2
AE31 : AE02 = E608
AE5F = AE30 = E636
AE69 = AE3A = E640
AE72 = AE43 = E649
AE90 = AE61 = E65B
AED9 = AEAA = E666
AE85 = AE56 = E670
AE9C = AE6D = E67B
AF0B = AEDC : E6B0
AF0B = AEDC : E6B0
AF13 = AEE4 : E6B7
AEFD = AECE = E6C5
AEEF = AEC0 : E6BF
AF00 = AED1 : E6C8
AF07 = AED8 : E6CF
AF19 = AEEA = E6D1

```
AF26 = AEF7 : E6DE
AF2B = AEFC : E6E2
AF32 = AF03 : E6E8
AFCE = AF9F : E6EE
AFD5 = AFA6 : E6F4
AEE3 = AEB4 = E6FA
```

RaNDom number processing

```
AF39 = AF0A = E706
AF53 = AF24 = E720
AF6E = AF3F = E73B
AF78 = AF49 : E745
AF85 = AF56 = E752
AF98 = AF69 : E765
AF9B = AF6C = E768
AFB6 = AF87 * E783
AFDC = AFAD = E7B1
```

string handling

```
AFE8 = AFB9 = E7BD
AFEE = AFBF = E7C3
AFF1 = AFC2 = E7C6
AFFB = AFCC = E7D0
B01D = AFEE = E7F2
B055 = B026 = E82A
B062 = B033 = E837
B065 = B036 = E83A
B068 = B039 = E83D
B0B2 = B083 = E887
B0C3 = B094 = E898
B0EE = B0BF = E8C3
B0F1 = B0C2 = E8C6
```

FN/PROC routines

```
B12D = B0FE = E902
B133 = B104 = E908
B141 = B112 = E916
B149 = B11A = E91E
B16B = B13C = E940
B1B9 = B18A = E98E
B1C4 = B195 = E999
B1C6 = B197 = E99B
B1D2 = B1A3 = E9A7
B1F7 = B1C8 = E9CC
B1F9 = B1CA = E9CE
B218 = B1E9 = E9ED
```

```
B223 = B1F4 = E9F8
B231 = B202 = EA06
B249 = B21A = EA1E
B255 = B226 = EA2A
B25F = B230 = EA34
B27C = B24D = EA51
B2BD = B28E = EA92
B2E4 = B2B5 = EAB9
B2ED = B2BE = EAC2
B2F9 = B2CA = EACE
B322 = B2F3 = EAF7
B332 = B303 = EB07
```

variable handling

```
B33C = B30D : EB11
B35B = B32C = EB30
B37E = B34F = EB53
B383 = B354 = EB58
B3B3 = B384 = EBB8
B3CC = B39D = EBA1
B3D6 : B3A7 = EBAB
B3EE = B3BD = EBC1
```

error and BRK handler

```
B3F6 = B3C5 = EBC9
B42A = B3F9 = EBFD
B433 * B402 = EC06
B443 * B433 = EC37
```

BASIC sound command processing

```
B461 : B44C = EC50
B49C : B472 = EC76
 NE    B48F = EC93
```

LISTing

```
B4CC = B4A0 = ECA4
B4DA = B4AE = ECB2
B4DD = B4B1 = ECB5
B4E0 = B4B4 = ECB8
B4E3 = B4B7 = ECBB
B4F2 = B4C6 = ECCA
B50C = B4E0 = ECE4
B515 = B4E9 = ECED
B53A = B50E = ED12
B54A = B51E = ED22
```

```
B54C = B520 = ED24
B562 = B536 = ED3A
 NE    B545 = ED49
 NE    B550 = ED54
B571 = B558 = ED5C
 NE    B562 = ED66
B57B = B565 = ED69
B58D = B577 = ED7B
 NE    B583 = ED86
B5A0 = B58A = ED8D
B5B5 = B59C = ED9F
B615 = B5FC = EDFF
B636 = B61D = EE20
B650 = B637 = EE3A
B672 = B659 = EE5C
```

FOR/NEXT loop handling

```
B681 = B668 = EE6B
B6A7 = B68E = EE91
B6AE = B695 = EE98
B6E0 = B6C7 = EECA
B6F0 = B6D7 = EEDA
B703 = B6EA = EEED
B75A = B741 = EF44
B76A = B751 = EF54
B77F = B766 = EF69
B7BA = B7A1 = EFA4
B7BD = B7A4 = EFA7
B7CA = B7B0 = EFB3
B7D8 = B7BD = EFC0
B84F = B7C4 = EFC7
B852 = B834 = F037
B852 = B837 = F03A
B86D = B84F = F052
```

GOSUB, GOTO, ON routines

```
B84B = B888 = F08B
B8B7 = B88B = F08E
B8A6 = B8A2 = F0A5
B8CE = B8AF = F0B2
B8D5 = B8B6 = F0B9
B8EB = B8CC = F0CF
B8F1 = B8D2 = F0D5
B8FC = B8DD = F0E0
B903 = B8E4 = F0E7
B911 = B8F2 = F0F5
B929 = B90A = F10D
B934 = B915 = F118
B9A9 = B98B = F18E
B9B8 = B99A = F19D
B9CD = B9AF = F1B2
B9D3 = B9B5 = F1B8
B9E2 = B9C4 = F1C7
B9E5 = B9C7 = F1CA
```

INPUT and DATA retrieval

```
B9ED = B9CF = F1D2
B9F8 = B9DA = F1DD
BA37 = BA19 = F21C
BA3D = BA1F = F222
BA49 = BA2B = F22E
BA5D = BA3F = F242
BA62 = BA44 = F247
BA78 : BA5A = F25D
BAEA = BAD0 = F2D3
BAF6 = BADC = F2DF
BB00 = BAE6 = F2E9
BB2F = BB15 = F318
BB39 = BB1F = F322
BB5A = BB40 = F343
BB6A = BB50 = F353
BBB6 = BB9C = F39F
```

REPEAT/UNTIL loop handling

```
BBC5 = BBA6 = F3A9
BBCC = BBB1 = F3B4
BBE8 = BBCD = F3D0
BBF1 = BBD6 = F3D9
BBFF = BBE4 = F3E7
```

BASIC program line input

```
BC17 = BBFC = F3FF
BC1D = BC02 = F405
BC42 = BC25 = F428
BC45 = BC28 = F42B
BC4A = BC2D = F430
BC7A = BC5D = F460
BC99 = BC7C = F47F
BC9E = BC81 = F484
BCAA : BC8D = F490
ADD2 = BCCC = F4CF
BCEE = BCD6 = F4D9
```

program housekeeping and control

```
BD29 = BD11 = F514
BD2C = BD14 = F517
BD38 : BD20 = F523
BD47 = BD2F = F532
BD52 = BD3A = F53D
BD69 = BD51 = F554
BD96 = BD7E = F581
BDA8 = BD90 = F593
BDAC = BD94 = F597
BDCA = BDB2 = F5B5
BDE3 = BDCB = F5CE
BDF4 = BDDC = F5DF
BDF9 = BDE1 = F5E4
BE02 = BDEA = F5ED
BE17 = BDFF = F602
BE23 = BE0B = F60E
BE25 = BE0D = F610
BE46 = BE2E = F631
BE59 = BE41 = F644
BE5C = BE44 = F647
BE6D = BE55 = F658
BE6E = BE56 = F659
BE7A = BE62 = F665
BE88 = BE6F = F672
BEAB = BE92 = F695
BEAC = BE93 = F696
BEB7 = BE9E = F6A1
BEBA = BEA1 = F6A4
BECB = BEB2 = F6B5
BED3 = BEBA = F6BD
 NE    BEC2 = F6C5
BEDB = BECF = F6D2
BEDE : BED2 = F6D5
BEE6 = BEDD = F6E0
BEF0 = BEE7 = F6EA
BEFA : BEF3 : F6F6
BF2D = BF24 = F727
BF33 = BF2A = F72D
```

data file access commands

```
BF39 = BF30 = F733
BF4F = BF46 = F749
BF50 = BF47 = F74A
BF61 = BF58 = F75B
BF78 = BF6F = F772
 NE    BF78 = F77B
BF81 = BF7C = F77F
BF85 = BF80 = F783
BF9B = BF96 = F799
BF9E = BF99 = F79C
BFAE = BFA9 = F7AC
BFBA = BFB5 = F7B8
BFCB : BFCF = F7C6
BFE6 : FBE4 = F7DB
 NE    BFC3 = F7F0
 NE    BFF6   NE
 NE     NE    F7FC
```

key

column 1 - Basic 1
column 2 - Basic 2
column 3 - HiBasic
NE - no direct equivalent
= - identical code
: - slight code changes
* - major code changes

ROM type	:0100 0000 - no service entry (+), :language entry, no second processor : relocation address
Binary version number	:00
Copyright offset	:0E
Title string	:BASIC (C) 1981 Acorn
Assembly address	:8000-BFFF
Relocation address	:none
Public workspace	:none
Private workspace	:none

(+) service entry is actually present

Basic 1 - Memory Map

8000-806C	:paged ROM protocol and initialisation
806D-843B	:BASIC keyword lookup tables
843C-87FC	:assembler
87FD-8A3C	:line editing routines
8A3D-8D32	:program control commands, statement parsing routines
8D33-8EC5	:display control routines
8EC6-8ECD	:machine code CALLs
8ECE-908E	:program line numbering
908F-9211	:array DIMensioning routines
9212-92B5	:setting BASIC parameters
92B6-932E	:PROCedure and LOCAL variable processing
932F-941A	:BASIC VDU command handler
941B-97AB	:variable processing and setup routines
97AC-9BDC	:statement parsing and IMMEDIATE command handler
9BDD-9E80	:arithmetic routines
9E81-A030	:PRINT formatting
A031-A6C8	:numeric input and floating point manipulation
A6C9-AB55	:higher order mathematical functions
AB56-AF32	:miscellaneous parameters determination
AF39-AFCD	:RaNDom number processing
AFDC-B1D9	:string handling
B1DA-B33B	:FN/PROC routines
B33C-B3F3	:variable manipulation

B3F6-B460 :error and BRK handler
B461-B4CB :BASIC sound command processing
B3CC-B680 :LISTing
B681-B8A5 :FOR/NEXT loop handling
B8A6-B9EC :GOSUB, GOTO, ON routines
B9ED-BBC4 :INPUT and DATA retrieval
BBC5-BC16 :REPEAT/UNTIL loop handling
BC17-BD28 :BASIC program line input
BD29-BF38 :program housekeeping and control
BF39-BFFF :data file access commands

Basic 1 - Zero Page usage

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
10	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
20	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
30	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
40	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	.
50	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
60	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
70	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
80	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
90	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
A0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
B0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
C0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
D0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
E0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
F0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	*
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

8000-806C - paged ROM protocol and initialisation

8000 :language entry
8003 :service entry
8009 :ROM title and copyright message
801F :ROM initialisation

B06D-843B - BASIC keyword lookup tables

806D-8359 :BASIC keyword and token table
835A-83CA :BASIC command routine address table (lo bytes)
83CB-843A :BASIC command routine address table (hi bytes)

843C-87FC - assembler

843C-8474 :assembler packed mnemonic lookup table 1
8475-84AD :assembler packed mnemonic lookup table 2
84AE-84E5 :assembler opcode lookup table
84E6 :reset OPT and quit assembler
84ED :assembler entry point
8585 :display address ....
8514 :display object code ....
851C :process next object code byte
852C :display source code from primary text buffer
853E :process next source code line
8543 :set index to source code length
856A :output ASCII equivalent of hexadecimal character + SPACE
8570 :output ASCII equivalent of hexadecimal character
857B :convert hexadecimal to ASCII and output character
8586 :store assembler symbol reference, process next character
85A1 :check for source code delimiting character
85CB :pack source code mnemonic
85D8 :process source code mnemonic
85EE :check for Boolean mnemonic
8607 :look up object code and transfer it to buffer
8612 :move next operation from assembly buffer to end of assembled code
8631 :reset length of object code
8634 :move source code address bytes to object code
8652 :check if error reporting required
8657 :"Out of range" error routine - error code 01
868C :"Byte" error routine - error code 02
86A1 :process next source code character
86BB :check indexed indirect instruction and process it
86CD :"Index" error routine - error code 03
86EF :add 8 to first byte in object code buffer, assemble 3 bytes
86FA :check third byte of object code, move object code from buffer
8728 :go to "Index" error routine
8764 :indexed indirect instruction
87BF :zero page instruction
87D8 :save OPT value, zero index
87E4 :evaluate expression, transfer variable to integer buffer
87EA :transfer secondary text buffer index to primary buffer index
87EF :add 16 to first byte in object code buffer
87F2 :add 8 to first byte in object code buffer

```
87F5    :add 4 to first byte in object code buffer
```

87FD-8A3C - line editing routines

```
87FD    :insert line number token, create space for line number instruction
8819    :convert decimal line number to hexadecimal, pack and insert
885E    :create space in program and insert line
8871    :insert text ....
8878    :set index=3, pack line number and insert
887A    :pack line number and insert
88AB    :check for operational or alphanumeric character.
88BB    :check for ASCII decimal numeral character
88C5    :check for decimal point or numeral
88CA    :get character from program and increment pointer
88CC    :increment pointers
88D3    :remove spaces from entire line, check for hexadecimal
88D9    :remove spaces from part of line, check for hexadecimal
88E9    :check for hexadecimal
88ED    :seek ASCII hexadecimal character
8901    :check for " : , . or *
8913    :check for : , . or *
892A    :check for .
8957    :flush buffer to first operational character
8959    :flush buffer to next operational character
8973    :test for presence of key word
8985    :search for key word in table
899F    :search keyword table for token
89AC    :move pointers on and check next keyword
89BE    :process token according to token processing parameter value
89F9    :is next character operator or alphanumeric?
8A13    :get non-space character from secondary text buffer
8A1E    :get non-space character from primary text buffer
8A29    :"Missing ," error routine - error code 05
8A35    :confirm next character is ","
```

8A3D-8D32 - program control commands, statement parsing routines

```
8A3D    :OLD command - token CB
8A40    :ROM service entry routines
8A50    :END command - token E0
8A59    :STOP command - token FA
8A62    :"CR STOP at line" embedded message
8A71    :PRINT ERL number
8A7D    :NEW command - token CA
8A80    :language entry routines
8A96    :reset BASIC pointers and initialise RAM
8A99    :initialise RAM settings (IMMEDIATE command handler)
8AAE    :RUN program
8AB6    :initialise page zero locations (BRK routine)
8AEA    :reset initial conditions (go to IMMEDIATE command handler)
8AED    :DEF command - token DD
8AED    :REM command - token F4
8AED    :DATA command - token DC
8B07    :decrement primary text buffer index, check line termination, process
        :buffer
```

```
8B09    :check line termination, process next statement from primary text
        :buffer
8B0C    :process next statement from primary text buffer
8B14    :check for BASIC command
8B22    :get token as index to BASIC command, execute it
8B30    :process text from primary text buffer
8B57    :LET command - token E9
8B6E    :confirm numeric variable and assign value
8B7E    :"Type mismatch" error routine - error code 06
8B8E    :check for FN * or [
8BA4    :go to "Syntax error" routine
ABA7    :go to assembler entry point
8BAA    :check for FN (token A4)
8BBC    :"No FN" error routine - error code 07
8BC3    :* command handler
8BD0    :pull integer from BASIC stack and ....
8BD3    :allocate variable
8C18    :store variable name length, point 2C,2D to next variable
8C2B    :store variable name
8C36    :save string from Page 6 buffer
8C4B    :"No room" error routine - error code 00
8C5B    :pull string off BASIC stack into buffer
8C88    :transfer string from BASIC stack to variables store
8C9D    :transfer variable from BASIC stack to variables store
8CC5    :PRINT# command - no token
8CE6    :PRINT integer variable to file
8CF2    :PRINT floating point variable to file
8CFF    :PRINT string variable to file
8D12    :recover channel number, save text buffer index, process next
        :statement
8D18    :process next line
8D1B    :decrement primary text buffer index, check line termination,
        :process next statement
```

8D33-8EC5 - display control routines

```
8D33    :PRINT command - token F1
8D3F    :tabulate according to formatting variable (@%) setting
8D5C    :check line termination
8D6B    :PRINT contents of buffer
8DA7    :PRINT string from Page 6 buffer
8DBD    :do Y tabulation
8DD9    :TAB( command - token 8A
8DF2    :SPC command - token 89
8DF2    :do X tabulation
8E01    :start new line, set COUNT=0 and primary=secondary text buffer
        :index, clear carry
8E04    :transfer secondary to primary buffer index, clear carry
8E0A    :check for formatting command (TAB SPC)
8E24    :seek formatting command, parenthetical string. PRINT it
8E32    :"Missing "" error routine - error code 09
8E3E    :output character, check for final "
8E57    :CLG command - token DA
8E5E    :CLS command - token DB
```

8EC6-8ECD - machine code CALLs

```
8E6C    :CALL command - token D6
8E7D    :process next CALL parameter
8E8B    :set up CALL parameter block in Page 6
8EA9    :check line termination, CALL machine code , process next statement
8EB8    :go to "No such variable" error routine
8EBB    :set carry flag, A-,X-,Y- registers CALL machine code subroutine
8ECB    :go to "Syntax error" routine
```

8ECE-908E - program line numbering

```
8ECE    :DELETE command - token C7
8F06    :push start line number to BASIC stack, STEP to stack
8F2A    :decrement primary text buffer index, test line termination
8F2F    :"Silly" error routine - error code 00
8F37    :RENUMBER command - token CC
8F78    :"RENUMBER space" error routine - error code 00
8F82    :increment line number, move program pointers to start of next line
8FAD    :set primary text buffer pointers to PAGE, process next statement
8FBA    :renumber any packed line number, prompt for input
8FDA    :reset BASIC pointers and initialise RAM
8FDD    :process program line
8FF0    :check number of next line
901E    :move program pointers to next line, increment RENUMBER space
        :pointers
902E    :"Failed at " error routine
9031    :"Failed at" error message
903C    :display line number, check next line
9052    :move program pointers to start of next line
905F    :AUTO command - token C6
```

908F-9211 - array dimensioning routines

```
908F    :go to "DIM space" error routine
9092    :reserve memory space
90D5    :"Bad DIM" error routine - error code 0A
90DD    :DIM command - token DE
9119    :dimension array
916C    :dimension array element
91C0    :check for further array elements to dimension
91CD    :"DIM space" error routine - error code 0B
91D7    :increment integer buffer
91E6    :pull variable off BASIC stack to 3F-42
91EB    :multiply integer in 2A,2B by integer in 3F,40
```

9212-92B5 - setting BASIC parameters

```
9212    :HIMEM command - token D3
9224    :LOMEM command - token D2
9239    :PAGE command - token D0
9243    :TRACE command - token FC
9265    :set TRACE flag on, reset maximum TRACE line number
9272    :reset TRACE flag, process next statement
```

927B :TIME command - token D1
928C :evaluate expression with result in integer buffer
9292 :get variable type from program, confirm numeric, convert floating :point to integer
929C :return if positive, else convert floating point to integer
929F :get variable type and ....
92A1 :confirm numeric variable and convert floating point to integer
92A9 :go to "Type mismatch" error routine
92AC :confirm variable numeric, convert integer to floating point
92AF :check variable type and convert integer to floating point

92B6-932E - PROCedure and LOCAL variable processing

92B6 :PROC command - token F2
92CD :clear integer LOCAL variable
92D5 :LOCAL command - token EA
92F3 :check for another LOCAL variable
9305 :check line termination, process next statement from primary text :buffer
9308 :"Not LOCAL" error routine - error code 0C
9310 :ENDPROC command - token E1
931F :"No PROC" error routine - error code 0D
9326 :CLEAR command - token D8

932F-941A - BASIC VDU command handler

932F :GCOL command - token E6
9346 :COLOUR command - token FB
9352 :"Bad MODE" error routine - error code 19
935A :MODE command - token EB
9397 :output character from stack, then one from integer buffer, process :buffer contents
93A1 :MOVE command - token EC
93A5 :DRAW command - token DF
93AE :PLOT command - token F0
93EA :send second character from integer buffer to current output stream
93EF :VDU command - token EF
9413 :decrement primary text buffer index, check line termination, process :next statement
9416 :send first character from integer buffer to current output stream

941B-97AB - variable processing and setup routines

941B :search for FN/PROC
9429 :search for vriable pointed to by 37,38 - length of name in 39
9439 :check next variable name for identity
945A :search variables store for name
9473 :check identity of variable name, letter by letter
94A1 :increment pointers by value of index
94AD :set up FN/PROC references
94BC :set up new variable name
94D6 :add address to variables chain
94F2 :set up new variable name and clear space for variable
94F7 :clear space for variable

```
94FF    :increment TOP-OF-VARIABLES pointer and ....
9507    :check if room for another variable
9519    :go to "No room" error routine
951F    :set index (Y) =1 and ....
9521    :count alphanumeric characters in variable name
9545    :clear space and set up variable reference
9548    :set up variable reference (2A,2B)
955D    :check for indirection operators
956D    :? indirection routine
9571    :! indirection routine
957C    :$ indirection routine
958B    :"$ range" error routine
9595    :set secondary text buffer = primary text buffer, decrement primary
        :index
95A0    :increment index (Y) and ....
95A1    :set up variable reference
95A9    :process text from primary text buffer
95CB    :set up variable
95E7    :check next character of variable name
960D    :determine variable type and set up references
9618    :set up integer reference
9631    :check for indirection operators
9647    :process ? operator
964B    :process ! operator
966B    :recover variable type from stack, return with A = FF, carry clear
9672    :set up array references
967B    :set up string variable reference
9695    :set up string array reference
96A3    :"Array" error routine - error code 0E
96AB    :set up array references
96C6    :calculate space for array, adjust variables table pointers
9753    :calculate space for array elements
977C    :use array dimension to move variables table pointers on
9789    :check array subscript size
97A0    :"Subscript" error routine - error code 0F
```

97AC-9BDC - statement parsing and IMMEDIATE command handler

```
97AC    :increment primary text buffer index, then ...
97AE    :if line number token present, unpack number to 2A,2B, set carry
97BA    :unpack line number to 2A,2B
97E2    :check = then evaluate RHS of expression
97F0    :"Mistake" error routine - error code 04
97FA    :confirm next character is =
9802    :evaluate expression
9805    :transfer X- to A-register, check line termination
980B    :check line for correct termination
9810    :test line for correct termination
9826    :increment primary text buffer pointers and test ESCAPE flag
9830    :set primary text buffer index to 1, test ESCAPE flag
9834    :test ESCAPE flag
9839    :"Syntax error" routine - error code 10
9848    :"Escape" error routine - error code 11
9851    :move to next statement with TRACE if required
```

```
9861          :increment index (Y) and ....
9862          :check if TRACE required
987D          :add index to primary text buffer pointers, reset it to 1
988D          :initialise RAM settings
9890          :go to "Type mismatch" error routine
9893          :IF command - token E7
98B2          :THEN command - token 8C
98C2          :search for ELSE and execute it
98D6          :display TRACE marker if line number below set limit
98F1          :zero PRINT-FIELD-CHARACTERS setting and ....
98F5          :output decimal equivalent of number in 2A,2B with leading spaces if
              :required
993D-9941     :HEX to decimal conversion lookup table 1
9942          :search program for line number, carry set if not found
9979          :confirm numeric variables, load in integer buffer and 39-3C
99A6          :"Division by zero" error routine - error code 12
99B8-99BC     :HEX to decimal conversion lookup table 2
99BD          :division process
9A11          :unpack variable to floating point buffer #1 and compare with
              :floating point buffer #2
9A28          :save floating point buffer #1 to BASIC stack ....
9A37          :unpack variable to floating point buffer #2 and ....
9A3A          :compare floating point buffer #1 and floating point buffer #2 for
              :identity
9A6B          :test sign of floating point buffer #2 (carry set if negative), set
              :A=1
9A72          :go to "Type mismatch" error routine
9A75          :transfer variable type to A- register and ....
9A76          :compare identity of variables
9A8B          :compare identity of integer variables (Z-flag set)
9ABF          :compare identity of strings
9AF7          :equate primary and secondary text buffers, evaluate next statement
9B03          :evaluate expression (MAIN ENTRY POINT)
9B14          :OR command - token 84
9B2F          :EOR command - token 82
9B45          :convert variable to integer, save to BASIC stack, ....
9B4C          :process < > = OR AND, if present
9B54          :AND command - token 80
9B76          :check for < = or >, process it, if found
9B82          :process < = > operations
9B8F          :save Y-register to each byte of integer buffer
9B9A          :process < operation
9BAE          :process > = operations
9BB9          :process < > operations
9BC2          :process > operation
9BD4          :process > = operation
```

9BDD-9E80 - arithmetic routines

```
9BDD          :"String too long" error routine - error code 13
9BEF          :concatinate strings
9C1D          :check for + or - operator
9C29          :perform + operation
9C36          :add integer variables
```

```
9C52    :reset BASIC stack pointers, seek another operator
9C63    :go to "Type mismatch" error routine
9C66    :add floating point variables
9C82    :remove integer from BASIC stack and do floating point addition
9C90    :do subtraction
9C9D    :do integer subtraction
9CBC    :do floating point subtraction
9CD5    :convert integer to floating point and subtract
9CE9    :convert integer to floating point
9CF8    :convert integer to floating point and ....
9CFB    :do floating point multiplication
9D14    :go to "Type mismatch" error routine
9D17    :do multiplication
9D38    :integer multiplication
9D68    :integer division
9D9A    :transfer integer quotient or remainder to integer buffer
9DA8    :perform multiplication
9DAB    :save integer to BASIC stack ....
9DAE    :evaluate expression, check for multiplication or division operator
9DB1    :check for * / MOD or DIV
9DC2    :floating point division routine
9DDE    :DIV routine - token 81
9DE7    :MOD routine - token 83
9DFA    :save integer to BASIC stack ....
9DFD    :evaluate expression and get non-space in X-register
9E12    :process exponentiation
9E4D    :use natural logarithms to evaluate power
9E6A    :large number exponentiation
```

9E81-A030 - PRINT formatting

```
9E81    :assemble hexadecimal output for PRINT
9EA1    :convert hexadecimal to ASCII
9EEA    :assemble PRINT output string
9EB9    :assemble floating point output string
9ED0    :assemble formatted PRINT output string
9EEA    :assemble PRINT output string
9F0E    :go to PRINT using non-zero format parameter
9F16    :adjust position of decimal point for PRINTing
9F22    :divide floating point buffer #1 by ten, adjusting power in decimal
        :index store
9F8D    :PRINT with non-zero format parameter
9F91    :PRINT fixed number of decimal places
9FCC    :insert required number of zeros after decimal point
A006    :PRINT in exponential format
```

A031-A6C8- numeric input and floating point manipulation

```
A031    :transmit ASCII equivalent of decimal to string buffer
A043    :transfer ASCII equivalent of lo nibble to string buffer
A04B    :convert A-register to decimal ASCII, store in string buffer
A055    :add ASCII offset ....
A057    :store character in string buffer, incrementing pointers
```

```
A063    :save rounding byte, get sign of floating point buffer #1, set A=FF
A06C    :zero floating point buffer #1 mantissa and work area ....
A07C    :input first ASCII character. if numeric, enter in buffer
A08A    :input next ASCII character. If numeric, enter in buffer
A091    :process decimal point
A099    :enter ASCII numeral in floating point buffer #1
A0D2    :process exponentiation
A102    :exponentiation with negative index
A110    :exponentiation with index 0
A12A    :process -
A131    :process = or -, load next decimal digit if present
A13C    :process =
A149    :if numeral present, add it to ten times temporary store 4A
A161    :recover numeral from temporary store, clear carry
A169    :add corresponding byte of floating point work area to floating point
        :mantissa
A188    :multiply floating point buffer #1 by ten preserving A-register
A196    :multiply floating point buffer #1 mantissa by 4
A1CB    :get sign of floating point buffer #1
A1DE    :set sign, exponent and guard bytes of floating point buffer #1
A1E5    :multiply floating point buffer #1 by ten
A1F9    :add floating point buffer #2 mantissa to floating point buffer #1,
        :increment exponent if necessary
A1FE    :divide floating point buffer #1 mantissa by 2 and increment exponent
A20F    :copy floating point buffer #1 to floating point buffer #2
A230    :copy floating point buffer #1 to floating point buffer #2 and ....
A233    :divide floating point buffer #2 mantissa by 2
A23E    :divide floating point buffer #1 by ten
A295    :add A- to floating point buffer #1 mantissa incrementing exponent
        :if necessary
A2AF    :convert integer to floating point
A2D7    :transfer A-register to floating point buffer #1 sign, exponent and
        :guard bytes
A2DE    :if necessary, adjust sign and normalise floating point buffer #1
A2F4    :normalise floating point buffer #1
A327    :adjusting exponent, multiply floating point buffer #1 mantissa by 2
        :until bit 7 set
A33F    :unpack variable into floating point buffer #2
A36E    :pack floating point buffer #1 to memory at 0471
A372    :pack floating point buffer #1 to memory at 0476
A376    :pack floating point buffer #1 to memory at 046C
A37E    :pack floating point buffer #1 into memory
A3A3    :unpack variable at 046C to floating point buffer #1
A3A6    :unpack current variable to floating point buffer #1
A3D8    :round floating point buffer #1
A3F2    :position binary point, convert floating point to integer
A406    :copy floating point buffer #1 to floating point buffer #2, clear
        :floating point buffer #1 and ....
A40C    :shift binary point until exponent A= 0
```

```
A41A    :halve floating point buffer #1 mantissa, increment exponent,
        :repeating until exponent >A0
A44A    :divide floating point buffer #1 and floating point buffer #2
        :mantissa by 2, compensating exponent
A460    :go to "Too big" error routine
A463    :clear floating point buffer #2
A476    :check sign of floating point variable, complement mantissa if
        :negative
A47A    :change sign of floating point buffer #1 mantissa
A494    :change floating point buffer #1 exponent sign if negative, zero 4A
        :return with sign
A4BE    :change floating point buffer #1 mantissa sign and normalise buffer
A4C4    :increment mantissa of floating point buffer #1
A4D5    :increment floating point buffer #1 mantissa if negative, decrement
        :if positive
A4DE    :interchange variable in memory and floating point buffer #1
A4E4    :move floating point buffer #2 to floating point buffer #1
A4F0    :move mantissa from floating point buffer #2 to floating point
        :buffer #1
A505    :subtract variable from floating point buffer #1
A50B    :change sign of floating point buffer #1 ....
A50E    :align binary points, algebraicly sum floating point variables
A513    :align binary points, add floating point buffer #1 to floating point
        :buffer #2
A598    :take algebraic sum of mantissas
A5BF    :take smaller mantissa from larger
A5EE    :take floating point buffer #2 from floating point buffer #1
        :mantissa, difference in floating point buffer #1
A611    :if floating point buffer #1 positive, unpack variable to floating
        :point buffer #2 and multiply
A61E    :multiply floating point buffer #1 by floating point buffer #2,
        :product in floating point buffer #1
A661    :multiply floating point buffer #1 by current variable (4B,4C) ....
A664    :normalise floating point buffer #1 and round it off
A667    :round floating point buffer #1 if necessary
A677    :"Too big" error routine - error code 14
A681    :round floating point buffer #1 least significant byte
A687    :zero floating point buffer #1 rounding byte
A68B    :test for overflow
A691    :clear floating point buffer #1
A6A4    :set floating point buffer #1=- 1
A6B0    :invert floating point buffer #1 and change its sign
A6B8    :divide variable by floating point buffer #1, quotient in floating
        :point buffer #1
A6C6    :go to "Division by zero" error routine
```

A6C9-AB55 - higher order mathematical functions

```
A6C9    :TAN command - token B7
```

```
A6F2          :divide floating point buffer #1 by current variable (4B,4C)
A6FC          :divide floating point buffer #1 by floating point buffer #2
A715          :calculate mantissa
A7B4          :SQR command - token B6
A7B7          :check sign and calculate square root
A7BE          :"-ve root" error routine - error code 15
A7C9          :calculate square root of floating point numeral
A7EF          :set current variable pointer to 047B
A7F3          :set current variable pointer to 0471
A7F7          :set current variable pointer to 0476
A7FB          :set current variable pointer to 046C
A804          :LN command - token AA
A807          :check sign and calculate natural logarithm
A80E          :"Log range" error routine - error code 16
A81A          :calculate LN
A856          :constant (0.434 294 482) - for LN/LOG conversion
A85B          :constant (0.693 147 181) - used in LN calculation
A860-A888     :LN power series lookup table
A889          :power series calculation
A8C6          :ACS command - token 97
A8CC          :ASN command - token 98
A8DE          :calculate ASN for positive values
A8FE          :unpack PI/2 to floating point buffer #1
A907          :ATN command - token 99
A91D          :adjusting for fractional values, carry out ATN calculation
A929          :subtract angle from PI/2
A932          :carry out ATN calculation
A956-A988     :ATN power series
A989          :COS command - token 9B
A994          :SIN command - token B5
A99A          :calculate SIN
A9AB          :carry out SIN/COS calculation
A9DA          :adjust value of angle for SIN/COS calculation
AA45          :"Accuracy lost" error routine - error code 17
AA5C          :set current variable pointer to AA6D (-1.570 312 50)
AA60          :set current variale pointer to AA72 (-4.838 267 95 E-4)
AA69          :set current variable pointer to AA77 (PI/2)
AA6D-AA71     :constant (-1.570 312 50)
AA72-AA76     :constant (-4.838 267 95 E-4)
AA77-AA7B     :constant (-PI/2)
AA7C-AA80     :constant (1/57.295 779 6) - radian/degree conversion factor
AA81-AA85     :constant (57.295 779 5) - degree/radian conversion factor
AA86-AAB3     :SIN/COS power series
AAB4          :EXP command - token A1
AACF          :"Exp range" error routine - error code 18
AAF4          :multiply floating point buffer #1 by variable at 0476
AAFD          :carry out power series calculation with exponent power table
AB07-AB0B     :'e' (2.718 281 83)
AB0C-AB34     :EXP power series
```

```
AB35    :raise floating point variable to power specified in temporary power
        :store (4A)
```

AB56-AF52 - miscellaneous parameters determination

```
AB56    :ADVAL command - token 96
AB64    :POINT( command - token B0
AB92    :POS command - token B1
AB9B    :VPOS command - token BC
ABAD    :SGN command - token B4
ABCD    :LOG command - token AB
ABD6    :RAD command - token B2
ABDD    :set current variable pointer (4B,4C)=(Y,A). Multiply by floating
        :point buffer #1
ABE7    :DEG command - token 9D
ABF0    :PI command - token AF
ABFB    :USR command - token BA
AC0F    :go to "Type mismatch" error routine
AC12    :EVAL command - token A0
AC49    :restore secondary text buffer parameters, A=variable type
AC55    :VAL command - token BB
AC5A    :calculate VAL of string in buffer
AC99    :save variable type, reset secondary text buffer parameters,
        :A  variable type
AC9E    :INT command - token A8
ACC1    :go to "Type mismatch" error routine
ACC4    :ASC command - token 97
ACFA    :complement contents of integer buffer
ACD3    :INKEY command - token A6
ACDE    :EOF command - token C5
ACEA    :TRUE command - token C5
ACF7    :NOT command - token AC
AD08    :INSTR command - token A7
AD78    :zero integer buffer, set A40
8D8A    :go to "Type mismatch" error routine
AD8D    :ABS command - token 94
AD94    :make integer buffer positive
AD99    :take modulus of floating point buffer #1
ADA0    :change sign of floating point buffer #1. If negative, set A=FF
ADAE    :process -
ADB1    :check variable type and carry out subtraction if numeric
ADB5    :change sign of each location of integer buffer
ADCF    :reset BASIC pointers, process "LINE space" error routine
ADD2    :"LINE space" error routine - error code 00
ADDC    :transfer input line to Page 6 buffer, set A0
ADF8    :transfer string, if legal, to Page 6 string buffer
AE18    :go to "Missing " " error routine
AE1B    :check for -"+ and ....
AE31    :process next character
```

```
AE5F    :check OPT value, load object address to integer buffer
AE72    :"No such variable" error routine - error code 1A
AE85    :evaluate expression and check for )
AE90    :"Missing )" error routine - error code 1B
AE9C    :convert ASCII character to hexadecimal integer
AED9    :"Bad HEX" error routine - error code 1C
AEE3    :TIME command - token 91
AEEF    :PAGE command - token 90
AEF6    :go to "No such variable" error routine
AEF9    :FALSE command - token A3
AEFD    :go to "Type mismatch" error routine
AF00    :LEN command - token A9
AF07    :transfer A-register to least significant byte of integer buffer,
        :zero hi bytes
AF0B    :TOP command - token B8+50
AF19    :transfer A-, Y-register to integer buffer, zero hi bytes, A=40
AF26    :COUNT command - token 9C
AF2B    :LOMEM command - token 92
AF32    :HIMEM command - token 93
```

AF39-AFCD - RaNDom number processing

```
AF35    :process random number
AF53    :process RND(N)
AF6E    :process RND(-N)
AF78    :RND command - token B3
AF85    :move RND store to integer buffer
AF98    :process RND(1)
AF9B    :process RND(0)
AFB6    :shuffle numbers in RND store 32 times
AFB8    :shuffle numbers in RND store
AFCE    :ERL command - token 9E
AFD5    :ERR command - token 9F
```

AFDC-B1D9 - string handling

```
AFDC    :get character in X-register, with time limit, Y=0 success
AFE8    :GET command - token A5
AFEE    :GET$ command - token BE
AFF1    :save character to Page 6, set LENGTHOFSTRING 1, A=0
AFFB    :LEFT$( command - token C0
B01D    :RIGHT$( command - token C2
B055    :INKEY$ command - token BF
B062    :go to "Type mismatch" error routine
B065    :go to "Missing ," error routine
B068    :MID$( command - token C1
B0B2    :reposition substring within Page 6 buffer
B0C3    :STR$ command - token C3
B0EE    :go to "Type mismatch" error routine
```

```
B0F1        :STRING$(' command - token C4
B12A        :go to "String too long" error routine
```

B1DA-B33B - FN/PROC routines

```
B1D2        :restore primary text buffer pointers and go to ....
B133        :"No such FN/PROC" error routine - error code 1D
B141        :search for DEF in BASIC program
B149        :flush program to DEF token
B16B        :process DEF token
B1B9        :"Bad call" error routine - error code 1E
B1C4        :FN command - token A4
B1C6        :process FN/PROC
B1D2        :save 6502 stack to BASIC stack
B1F9        :set program pointers
B218        :point primary text buffer to next variable
B223        :check for LOCAL variables
B231        :set up LOCAL variable
B249        :discard LOCAL variables
B255        :recover primary text buffer parameters from stack
B25F        :pull processor stack off BASIC stack
B27C        :set up LOCAL variables in FN/PROC
B2BD        :process LOCAL variable arguments
B2E4        :reset processor, text buffer. issue "Arguments" error message
B2ED        :"Arguments" error routine - error code 1F
B2F9        :save global variables
```

B33C-B3F5 - variable manipulation

```
B33C        :store variable, save reference to BASIC stack
B35B        :unpack variable specified (type and location) by integer buffer into
            :integer floating point or string buffer according to variable type
B37E        :transfer variable address to A- Y-registers
B383        :unload floating point variable from variables table to floating
            :point buffer #1
B3B3        :unload string variable from variables table to Page 6
B3CC        :transfer string from variables table to Page 6
B3D6        :transfer string pointed by 2A,2BC
B3DC        :transfer string variable to Page 6 buffer
B3EE        :CHR$ command - token BD
```

B3F6-BB460 - error and BRK handler

```
B3F6        :reset ERRL pointer, set program pointer to PAGE
B42A        :check if primary text buffer pointers beyond program pointers
B433        :entry to BRK handling routine
B443        :default routine for ON ERROR
```

B461-B4CB - BASIC sound command processing

B461 :SOUND command - token D4
B49C :ENVELOPE command - token E2

B4CC-B680 - LISTing

B4CC :WIDTH command - token FE
B4DA :go to "Type mismatch" error routine
B4DD :confirm numeric variable, convert floating point to integer, save to :variables table
B4E0 :transfer variable from BASIC stack to variables table
B4E3 :assign value to numeric variable
B4F2 :transfer variable from integer buffer to variables table
B50C :confirm numeric variable, convert integer to floating point, save to :variables table
B515 :pack variable from floating point buffer #1 to variables table
B53A :output character in A-register, expanding if token
B54A :find keyword in table
B54C :check next keyword in table
B562 :expand keyword from token
B571 :output any character, updating COUNT
B57B :output SPACE, updating COUNT with new line if necessary
B57D :output character, updating COUNT, with new line if necessary
B587 :output character from stack updating COUNT
B58D :insert spaces according to LISTO setting
B5A0 :LISTO command - token C9+4F
B5B2 :go to "Type mismatch" error routine
B5B5 :LIST command - token C9
B615 :check if new line number is above finish line number
B636 :LIST line from program
B672 :unpack line number and PRINT it

B681-B8A5 - FOR/NEXT loop handling

B681 :check loop status and set flags
B6A7 :"No FOR" error routine - error code 20
B6AE :NEXT command - token ED
B6DF :"Can't match FOR" error routine - error code 21
B6F0 :check loop variable type and process it
B703 :process FOR/NEXT loop with integer variable
B75A :reset text buffer pointers, proceed with program
B76A :decrement FOR/NEXT loop counter
B77F :process FOR/NEXT loop with floating point variable
B7BD :"FOR variable" error routine - error code 22
B7CA :"Too many FORs" error routine - error code 23
B7D8 :"No TO" error routine - error code 24
B7DF :FOR command - token E3
B852 :store FOR/NEXT loop address and increment counter

```
B86A    :go to "Type mismatch" error routine
B86D    :save FOR/NEXT loop parameters to memory
```

B8A6-B9EC - GOSUB, GOTO, ON routines

```
B8A6    :"Too many GOSUBs" error routine - error code 25
B8B4    :GOSUB command - token E4
B8CE    :"No GOSUB" error routine - error code 26
B8D5    :RETURN command - token F8
B8EB    :GOTO command - token E5
B903    :test line termination, reset ERRV, process next statement
B911    :ON ERROR routine
B929    :"ON syntax" error routine - error code 27
B934    :ON command - token EE
B9A9    :"ON range" error routine - error code 28
B9B8    :confirm specified line number is present
B9CD    :search for given line number, carry set if not found
B9D3    :"No such line" error routine - error code 29
B9E2    :go to "Type mismatch" error routine
B9E5    :go to "Syntax error" ROUTINE
```

B9ED-BBC4 - INPUT and DATA retrieval

```
B9ED    :INPUT# command - token E8+23
BA37    :input numeric variable
BA3F    :input integer variable
BA49    :input floating point variable
BA5D    :discard input channel number and flag, process next statement
BA62    :INPUT command - token E8
BA78    :input next variable
BAEA    :derive numeric variable from ASCII input
BAF0    :pull variable from BASIC stack and input next variable
BAF6    :allocate string variable, input next variable
BB00    :RESTORE command - token F7
BB2F    :check for more DATA, if present, else process next statement
BB39    :READ command - token F3
BB6A    :seek next DATA item if present
BBB6    :"Out of DATA" error routine - error code 2A
```

BBC5-BC16 - REPEAT/UNTIL loop handling

```
BBC5    :"No REPEAT" error routine - error code 2B
BBCC    :UNTIL command - token FD
BBE8    :process next REPEAT/UNTIL loop
BBF1    :"Too many REPEATs" error routine - error code 2C
BBFF    :REPEAT command - token F5
```

BC17-BD28 - BASIC program line input

BC17 :read input line from current stream to Page 6
BC19 :input line from current stream to Page 6
BC1D :input line to Page 7 after PRINTing A-register as prompt
BC20 :input text line to Page 7 buffer
BC42 :start new line, zero COUNT
BC45 :zero COUNT
BC4A :search for specified line number, carry set if not found
BC99 :increment TOP setting, clear carry
BC9E :transfer next character to top of program
BCAA :insert BASIC line into program
BCE9 :set Y=1and go to ROM language entry routine
BCEE :make space and insert line

BD29-BF38 - program housekeeping and control

BD29 :RUN command - token F9
BD2C :reset BASIC pointer, BRKV and primary text buffer pointer
BD38 :reset BASIC pointers
BD47 :zero variables except system integers
BD52 :reset TOPOFVARIABLES pointer and loop counters
BD69 :pack floating point buffer #1 on to BASIC stack
BD96 :increment BASIC stack pointer, set current variable pointers (4B,4C)
BDA8 :push value of variable on to BASIC stack
BDAC :push integer buffer on to BASIC stack
BDCA :push string from Page 6 buffer to BASIC stack
BDE3 :transfer string from BASIC stack to Page 6
BDF4 :get character from BASIC stack, use it to increment stack pointer
BDF9 :increment BASIC stack pointer
BE02 :pull integer variable from BASIC stack
BE17 :increment BASIC stack pointers by 4
BE23 :pull variable off BASIC stack to 37-3A
BE25 :pull variable off BASIC stack to 00,X-03,X
BE46 :if enough memory, move BASIC stack pointer down
BE59 :go to "No room" error routine
BE5C :store integer buffer in 00,X to 00, X+3
BE6D :clear carry and increment line search pointer
BE6E :increment line search pointer
BE7A :load program, set TOP, check if program good
BE88 :relink BASIC program (set TOP =PAGE)
BE91 :check if program is good
BEAB :reset TOP to TOP+Y
BEAC :increment TOP
BEB7 :"Bad program" error routine
BECB :set up file name buffer
BED3 :terminate filename (string) in Page 6 with CR
BEDB :go to "Type mismatch" error routine
BEDE :get parameters for LOAD/SAVE

```
BEF0        :read machine higher order address
BEFA        :SAVE command - token CD
BF2D        :LOAD command - token C8
BF33        :CHAIN command - token D7
```

BF39-BFFF - data file access commands

```
BF39        :PTR command - token CF
BF4F        :EXT command - token A2
BF50        :PTR command - token 8F
BF61        :BPUT command - token D5
BF78        :BGET command - token 9A
BF81        :OPENOUT command - token AE
BF85        :OPENIN command - token AD
BF9B        :go to "Type mismatch" error routine
BF9E        :CLOSE command - token D9
BFAE        :get channel number for PRINT#, input number into A- Y-registers
BFBA        :get file number from primary text buffer
BFC8        :go to "Syntax error" ROUTINE
BFCB        :process error routine embedded in ROM
BFE3        :jump to address stored at 0037
BFE6        :REPORT command - token F6
```

Page 0

```
0000-0001 :LOMEM
0002-0003 :TOP-OF-VARIABLES pointer
0004-0005 :BASIC stack pointer
0006-0007 :HIMEM
0008-0009 :ERRL
000A      :primary text index
000B-000C :primary text pointer
000D-0011 :RND number store
0012-0013 :TOP
0014      :number of characters in PRINT field
0015      :HEX PRINT flag (bit 7 denotes status)
0015      :variables table index
0016-0017 :ERRV (default B443)
0018      :PAGE hi
0019-001A :secondary text pointer
001B      :secondary text index
001C-001D :DATA pointer
001E      :COUNT
001F      :LISTO option
0020      :TRACE flag (hi bit indicates status)
0021-0022 :maximum TRACE line number
0023      :WIDTH
0024      :REPEAT/UNTIL loop counter
0025      :GOSUBS counter
0026      :FOR/NEXT loop counter (fifteen times number of loops)
0027      :variable type evaluator parameter
0028      :OPT value for assembler
0029-002B :assembly buffer
002A-002D :integer buffer
002A      :integer buffer least significant byte
002A-002B :variable pointer
002A-002B :STEP
002C      :variable type
002C-002D :variables table pointer
002D      :integer buffer most significant and sign byte
002E-0035 :floating point buffer #1
002E      :          sign byte
002F      :          guard byte
0030      :          exponent
0031      :          mantissa most significant byte
0032      :          mantissa
0033      :          mantissa
0034      :          mantissa least significant byte
0035      :          rounding byte
0036      :length of string
0037-0038 :assembled code location
0037      :OSWORD control block - least significant byte of input line buffer
```

```
0037-0038 :RENUMBER pointer
0037-0038 :DIM variable name pointer
0037-0038 :program pointer
0037      :PRINT format parameter
0037-0042 :SOUND parameter
0038      :OSWORD control block - most significant byte of input line buffer
0038      :temporary store for sign during division calculation
0038      :number of decimal places in PRINT field
0039      :length of variable name
0039      :length of object code operation
0039-003A :secondary program pointer
0039-003A :BASIC keyword table pointer
0039-003A :DELETE range upper limit line number
0039-003C :integer variable temporary store
003A-003B :variable name pointer
003A      :string length used in string comparison
003A      :keyword table index
003B-0042 :floating point buffer #2
003B      :floating point buffer #2 sign byte
003B      :input buffer index
003B      :index temporary store
003B-003C :RENUMBER space pointer
003C      :floating point buffer #2 guard byte
003C      :text index temporary store
003C      :most significant byte of higher order address
003C-003D :next variable pointer
003D      :floating point buffer #2 exponent
003D-003E :assembler opcode lookup table index calculation workspace
003D-003E :line number temporary store
003D      :keyword token processing parameter
003D-003E :next line pointer
003E      :floating point buffer #2 mantissa most significant byte
003F      :floating point buffer #2 mantissa
003F      :variable length
003F      :array type
003F-0040 :buffer used in DIM calculation
003F      :LOCAL variables counter
003F      :line length
003F-0048 :LOAD/SAVE parameter
0040      :floating point buffer #2 mantissa
0041      :floating point buffer #2 mantissa least significant byte
0042      :floating point buffer #2 rounding byte
0043      :temporary store used in decimal numeral input
0043-0046 :work area used in tangent calculation
0044      :ENVELOPE number
0048      :index input temporary store
0048      :power series calculation iteration index store
0049      :decimal index
0049      :index working store used in exponent calculation
```

004A :temporary store for power used in exponent calculation
004A :temporary store used in decimal numeral input
004A :temporary angle store used in trigonometric calculation
004B-004C :current variable pointer
004B-004C :stack pointer
004D :channel number temporary store
004D-004E :power series pointer
004D :LOCAL variables arguments counter
004D :PRINT status flag
004E :PRINTFIELD parameter
004E :LOCAL variables counter
004E :input flag

Page 1

0100-01FF :6502 stack
0100-01FF :storage of LOCAL variables in FuNctions or PROCedures
0100- :error handling routine relocated to base of 6502 stack

Page 4

0400-046B :system integer variables @%-Z%
046C-047F :temporary storage for floating point variables during program execution
0480-04EC :start of variable address chains
04F6-0457 :start of PROC chain
04F8-04F9 :start of FN chain
04FA-04FF :unused

Page 5

0500-05A3 :FOR/NEXT loop parameter stores (up to ten blocks of fifteen bytes)
05A4-05B7 :REPEAT/UNTIL loop start address hi bytes (up to twenty)
05B8-05CB :REPEAT/UNTIL start address lo bytes
05CC-05E5 :GOSUB return address lo bytes (up to twenty-six)
05E6-05FF :GOSUB return address hi bytes

Page 6

0600-06FF :string buffer
0600-06FF :CALL parameter block

Page 7

0700-07FF :input buffer

user program area
PAGE-TOP :Basic program
LOMEM-VARTOP :variables table
VARTOP-STCKBASE :RENUMBER space
STCKBASE-HIMEM :Basic stack

```
JMP/JSR   :calling location
address   :address

(002A) :8EC8
(0037) :8B2D BFE3
(WRCHV):9418 B58A
801F   :8000
84ED   :8BA7
84F1   :8567
856A   :850C 8516
8570   :8507 856A
857B   :8575
85A1   :84FD
8612   :85F4 8700 8730 8761 87E1
861E   :862E
8685   :8742 878E
868C   :87D2
86CD   :8728 8774 87B2
86F7   :874A 87A0 87AF
86FA   :87D5
87E4   :8638 8685 869E 86D7 8714 8747 875C 876A 8793 87B5 87D8
       :9250 932F 9349 935D 93AE 9400 B461 B49C B4CC
87EA   :859E
87EF   :86AF 86E1 871E 8764 8767 87AC 87CB
87F2   :8682 86EF 87EF
87F5   :86F7 872B 87F2
87FD   :8861 89DC
8819   :8937
887A   :9017
88AB   :8952 895B 89CA 89FB B196
88BB   :88F2 892E
88C5   :893E
88CA   :B40A B419 B422 B427
88CC   :889E 88A1 88A4 88E3 88ED 8905 8943 8960 8A00
88D3   :9076 AC40
88D9   :88E6 894F 8AD3
88E3   :8970 8A10
893C   :8946
8959   :8963
897F   :89BB
89F9   :8A03
8A13   :8A35 8CCC 8DDC 8E7D 97FA AC76 AC81 AC8C ADDC AE31 B0C3 B2B6 B2CF
       :B805 B82E B884 BAD7 BB7A BB89 BFBA
8A1E   :84F1 85A3 867B 8697 86A1 86A8 86B2 86BF 86C6 86DA 86E4 870B 8717
       :8721 873B 8753 876D 8786 8799 87A3 87BB 87C2 8AE1 8D22 8D33 8D5C
       :8E24 8ED6 8F16 90DD 915E 91C0 92FB 93EF 9406 B16E B228 B2A8 B5D7
       :B5E8 B775 B911 B934 B9F8 BA62 BB08 BB2F
8A29   :8DBA AD29 B065
8A35   :9332 93B4 93C0 AB6A B0F7 B46E B4A6 BF65
```

```
8A40  :BCEB
8A80  :806A
8A96  :8A4D 8ADE 8F03 8FDA 908C BF30
8A99  :8561 8A56 8A7A 8AEA 8C58 988D B5AF B633 BEC8
8AAE  :BD35
8AB6  :B440
8AED  :8BCD B926
8AF7  :98D3
8B07  :8D1B 91C7 9302 9413 B7BA BB36
8B09  :8BA1 8D15 9305 B9EA BA5F
8B0C  :8B68 8B7B 8E69 8EB5 9221 9236 9240 9262 9289 92CA 932C 939E 93E7
      :B499 B4C9 B4D7 B8E8 B90E BB2C BBE5 BF2A BF4C BF75 BFAB BFFD
8B14  :84EA 8B04 98AF B23A B767 B867 B900 BC14
8B22  :AE3C
8B7E  :92A9 9890 9A72 9C63 9D14 AC0F ACC1 AD8A AEFD B062 B0EE B4DA B5B2
8B7E  :B86A B9E2 BEDB BF9B
8BD0  :8B68 BA31 BB57
8BD3  :B32F BAFA
8C5B  :9519 B24E BE59
8D54  :8D3C
8E04  :8DD6
8E0A  :8D77 8E27
8E24  :BA78 BA7D
8E32  :AE18
8EBB  :8EB1 ABFE
8E7D  :8EA6
8F06  :8F37 905F
8FBC  :904F
9052  :8F73 8FA8 901E
9092  :9116
90D5  :9169 920F
90DD  :91CA
91C0  :90D2
91CD  :908F
91D7  :8EF6 90A1 914A AF68
91E6  :915B
91EB  :9176 9705
928C  :8DD9 909E 9338 93C3 96D1 AB64 B076 B0F1 B471 B4A9 B810 B835
9292  :8DF2 957E 9576 965D AB56 ABFB ACF7 AFDC B3EE BFC1
929C  :9212 9224 9239 927B
929F  :8E6F 93BA B5A8 BBD2 BF40 BF6B
92A1  :87E7 8DC7 9139 928F 9654 9719 973E 997A 9989 9B18 9B33 9B46 9B55
      :9B5F AB70 AD32 AF3E B00C B02E B08D B940 B9C0
92AC  :9E19 A6C9 A7B4 A804 A8CC A907 A989 A994 AAB4 ABD6 ABE7
92AF  :9A31 9D04 9DC3 9DCF 9E13 B870 B88E
9397  :9343 934F
93BA  :93AB
9416  :8DD3 939B 93DF 9403
941B  :B210
```

```
9429 :911D 9626 9688 96AB
94AD :B1A0
94BC :8B46 9122 94F2 954F
94F7 :8B52 9127 9545 B1A5
94FF :B1B3
951F :90F9
9521 :B204
9548 :8586 8B57 9094 92DA B285 B7DF BA02 BA99 BB39
9595 :9548 B6AE
95A1 :8E86
95A9 :8B3A AE51
962D :9678
966B :9579
96AB :9675 969A
973B :96C3
9789 :96EB 9729 9749
97AE :8AD6 8ECE 8EDD 8F0B 8F1D 9243 98B4 B5C5 B5F1 B9B8
97BA :8FDD B672
97D6 :929C
97E2 :8B61 8B71 BF3D
97FA :8B43 B7E9
9802 :BF68
9805 :8BB9 B5A5 BBC5 BEE6
980B :8EAB 92C7 933B 93C6 B479 B4B1 B4CF BFA1
9810 :8550 8A3D 8A50 8A59 8A7D 8B09 8E57 8E5E 8EE2 8F2C 9253 9267 9274
     :931C 9326 934C 9360 9851 B5FC B8B7 B8D5 B8EE B903 BB21 BD29 BFE6
9812 :980D
981A :9808
9826 :84F8 8BC3 B19D B618 B91B BC05
9830 :98BC B764 B983
9834 :8EF3
9839 :85EB 8BA4 8ECB B6B9 B9E5 BFC8
9848 :BC3F
9851 :B852
9861 :8B01
9862 :8564
98B4 :B9B5
98D6 :9878 B8F5
98F1 :8A74 9047 98E5 B67B
98F5 :906B B636
9942 :B605 B9CD BC4A
9979 :99BD
99A6 :A6C6
99BD :9DDE 9DE7
9A37 :B7A8
9A3A :9A25
9A75 :9BA7 9BB0 9BBB 9BCB 9BD6
9A76 :9B89
9AF7 :87E4 8BB6 8E6C 9136 93A8 9893 B5A2 B93D B9BD BBCC BEDE
```

```
9B03 :8CD3 8D84 928C 93B7 9802 AC43 AD08 AD16 AE85 AFFB B01D B068 B2BD
     :B4DD B86D B88B
9B45 :9B14 9B2F
9B4C :9B03
9B76 :9B4C 9B5B
9BDD :B12A
9C1D :9A2B 9A7D 9AC2 9B76
9C52 :9CB9
9C76 :9C8D
9C7C :9CD2 9CE6
9D07 :9CF5
9D17 :9DA8
9D98 :9DE4
9D9A :9DF7
9DAB :9C2E 9C95
9DAE :9C1D 9C69 9CBF
9DB1 :9D11 9DA5
9DFA :9983 9D2E
9DFD :9BF2 9CFE 9DAE 9DC9
9E00 :9E34 9E67 9E7E
9E4D :9E73
9EC2 :9ECD 9F27
9ED0 :8D94 B0E8
9EEA :B0E2
9F8D :9F0E
A031 :9FD9
A043 :A019
A055 :A037 A051
A057 :9EB2 9EBF 9F0B 9FC2 9FC7 9FD0 9FE2 A008 A011 A026 A02D
A06C :AC86 AC91 AE59
A131 :A0D2
A13C :A12A
A169 :A1F9 A25B A65A
A188 :A040 A0B9
A1CB :9F00 A066 A0E1 A413 A49C A513 A611 A6B8 A6F2 A7B7 A807 A8CF A8F0
     :A90A ABA4 AD99 ADA0
A1E5 :9EC8 A0F9
A1F9 :A24C A275 A28D A5E8
A1FC :A2AB
A20F :9A1C A1F0 A230 A406 A6BD
A230 :A249 A24F
A233 :A1F3 A1F6 A252 A255 A258
A23E :9F22 9F5C A102
A295 :A3E2 A671
A2AF :92B3 9A19 9C73 9C8A 9CC9 9CDD 9CE9 9CF2 9CF8 9EFD AF53 B512
A2DE :A83F
A2F4 :A0F0 A2D4 A4C1 A664 AA09 AF62
A33F :9A37 9F65 A4DE A50E A616 A6F7 A9C5
A36E :9E4A 9E70 A7D7 AA0C
```

```
A372  :9E37 9E6A A8DE A938 A9BF AAE1
A376  :8CD6 9F2E A6B0 A7C9 A838 A88D A8E4 A9E0 AB42
A37E  :A4E1 A6D5 AA1B B87E B8A0
A3A3  :AA56 BA54
A3A6  :9A22 9E2A 9E42 9E50 A6C0 A8A4 A901 AA21 AAEC ABF3 B319
A3F2  :9299 92A5 989A 9E84 AF65 B4EF
A3F5  :ACBB
A40C  :A3F2 A49F A9E9 ACA8
A463  :A410 A81D A93B
A47A  :A4BE A4D5 A4DB AA06
A494  :9E22 AADB
A4BE  :A4B9
A4C4  :A4D8
A4D5  :ACB8
A4DE  :A6E0
A4F0  :A4A6
A505  :9CE3 A8EA AA37 AA3D
A50B  :9CCF A505 A92C
A50E  :9C79 A7E3 A850 A8BE AA18 AA2D B78D
A513  :9F6C A82C A946 A9CD
A611  :A661 AF5F
A661  :9D0A 9E59 9E62 A84A A8E1 AA12 AA27 AAF7 AB4F ABE1
A664  :A5E4 A60D A7B1 AFB0
A667  :A109 A5EB
A677  :A460
A687  :A674
A691  :9F4D 9F91 A2DF A409 A5BC A61B A6A4 AAC9
A6A4  :9E6D 9E79 9F11 A6B3 A8E7 A9B9 AB45 B881
A6B0  :A923 AB3D
A6B8  :9DD5 A7DD A8AA A8F8
A6F2  :A6EC A9E6
A7B7  :A8ED
A7EF  :A6D2 A6DD A6E9
A7F3  :9E5F A7E0 AA1E
A7F7  :9E56 A8F5 AAF4
A7FB  :9F62 A3A3 A7DA A84D A8A7 AA15 AA2A
A804  :ABCD
A807  :9E53
A889  :A835 A94D A9D4 AB01
A8CC  :A8C6
A8DE  :A8D8
A90A  :A8FB
A918  :A8DB
A91D  :A915
A929  :A8C9 A99E
A932  :A926
A99A  :A6DA A6E6 A991
A9AB  :A9A5
A9DA  :A6CC A98C A997
```

```
AA30 :AA42 AA59
AA5C :AA0F AA3A
AA60 :AA24 AA34
AA69 :A8FE A929 A9C2 A9E3 ABF0
AAB7 :9E5C
AAF4 :A950 A9D7
AAFD :AADE
AB35 :9E2F 9E47 AAF1
AB48 :AB52
AC49 :AC9B
AC5A :BAED
AC99 :AC89
ACEA :ABC2
AD94 :9980 9993 9D4D 9D5D
ADA0 :A508 A50B A9A8
ADB1 :AC96
ADB5 :9DA0 A2B9
ADCF :BCE6
ADDC :BAD4 BB52
AE10 :ADF5
AE1B :9292 92AC 9DFD ABAD AC12 AC55 AC9E ACC4 AD8D AF00 B0D2 BF88
AE31 :ADAE
AE72 :8EB8 AEF6
AE85 :8DC4 9716 973B AB6D AD2F AF3B B009 B02B B08A B0FA
AF07 :8F08 8F13 9130 92EB 96C7 AB8F AB98 ABA1 ABC7 ACD0 AD75 AF28 AFEB
     :B5C2 B82B BF7E BF98
AF19 :8596 9875 AB61 ACDB AE6F AEF3 AF2F AF36 AFD2 AFD9 B380
AF85 :9D9A B355
AF98 :AF59
AFB6 :AF80 AF98
AFCE :8A71
AFDC :ACD3 B055
AFF1 :B3F3
B141 :B215
B1C6 :92C4
B223 :B1B6
B231 :B339
B322 :B31C
B332 :B325
B33C :92DF B2A5
B35B :9651 AE56 B347
B383 :B77F
B3F6 :8A5C B433
B4DD :B7EC BB46
B4E0 :859B 8B78 90CB 92F0 BA57 BAF0
B4E3 :B322
B515 :B798
B53A :8538 B6A1 BFF7
B571 :8523 8526 8583 8DB0 8E3E 98E2 98EA 9936 B568 B599 B664 BAB9 BC1D
```

```
B57B :8520 856D 8D4E 8DA1 8DF9 98ED 992C B596
B58D :B63F B646 B64D
B650 :B67E
B6AE :B77C
B852 :B8A3
B8B7 :B999
B8F1 :98BF B986
B8FC :BBEE
B9B8 :B8B4 B8EB B97B BB19
B9CD :98B9
B9F8 :BA34 BA5A
BA78 :BAF3 BAFD
BB2F :BB67
BB5A :BB49
BB6A :BB40 BB4C
BC17 :BABC
BC1D :8AAB 9070
BC42 :853E 8A77 8D18 8DED 8E01 904A B584 B615 BFEE
BC45 :8E61 B578
BC4A :8EF0 BCAC
BC7A :BC96
BC9E :BC90 BC93
BCAA :8ADB 9079
BCE9 :8003
BD2C :BF36
BD38 :8A96 907C 9329 ADCF BD2C
BD47 :9233
BD52 :8ABD BD44
BD69 :9A16 9A28 9C66 9C87 9CBC 9CDA 9CEF 9CFB 9DC6 9E16 AF56
BD96 :9A1F 9A34 9C76 9CCC 9CE0 9D07 9DD2 9E27 9E4D 9E76 AF5C B316
BDA8 :B2C0 B34B
BDAC :858D 8B5E 8B6E 8E72 8ED3 8F0E 9068 909B 9133 92E6 93BD 96CE 9713
      :9A7A 9B49 9B58 9DAB 9DFA AB67 B0F4 B2C7 B358 B5C9 B5E1 B7E6 BA0F
      :BAEA BB43 BB4F
BDCA :9ABF 9BE5 AC20 AD13 AD2C B006 B028 B071
BDE3 :9C11 AD35 B00F B031 B090 B32C
BDF4 :8C85 9AEE AC46 AD5F
BDF9 :8CC2
BE02 :8BD0 8EAE 8EED 9065 9073 9A13 9C84 9CD7 9CEC 9D55 AB79 B0FF B2F9
      :B31F B5DE B602
BE17 :9B28 9B6F
BE23 :93D2 B24B B4E0
BE25 :8F3C 91E8 96DC 9724 9998
BE46 :B1CD BD6E BDB1 BDCF
BE5C :9D52 AF70 B30F B344
BE6D :BB1E
BE6E :BD1A
BE7A :BF2D BF33
BE88 :8A4A 8A53 8F3F B5FF BCE3 BF0D
```

```
BEAB    :BC99 BCCF
BEAC    :BEA4
BECB    :BEE3
BED3    :8C36 BF8D
BEDE    :BE7A BEFA
BEF0    :9363
BFAE    :8CC7 B9EF BF39 BF61 BF9E
BFBA    :ACDE BF55 BF78
BFCB    :8A5F 8C4B 902E BEB7
BFD3    :BFE0
OSFIND :BF95 BFA8
OSBPUT :8CDD 8CEA 8CF7 8D01 8D0A BF72
OSBGET :BA14 BA20 BA28 BA3F BA4B BF7B
OSARGS :BF49 BF5B
OSFILE :BE85 BF27
OSRDCH :AFE8 AFEE
OSASCI :BFDD
OSNEWL :BC42
OSWRCH :8DCC 8DD0 8E66 9340 9398 93CB 93CF 93D7 93DC 93E4 93EC B575
OSWRCH :BFEB
OSWORD :9286 AB88 AEE9 B496 B4C6 BC3A
OSBYTE :8021 802A 8AC2 937E AB5D AB94 AB9D ACE4 AFE5 BEF2
OSCLI   :8BCA
```

Basic 1 - lookup table reference origins

L/U table address	:calling location :address
82CB	:8B23
833C	:8B28
843B	:85DC
8474	:85E1
84AD	:8607
993D	:9902
99B8	:9908

Basic 2 - summary and description of contents

ROM type	:0110 0000 - no service entry, language :entry, second processor relocation :address provided
Binary version number	:01
Copyright offset	:0E
Title string	:BASIC (C) 1982 Acorn
Assembly address	:8000-BFFF
Relocation address	:8000-BFFF
Public workspace	:none
Private workspace	:none

Basic 2 - Memory Map

8000-8070	:paged ROM protocol and initialisation
8071-8450	:BASIC keyword lookup tables
8451-887B	:assembler
887C-8AB5	:line editing routines
8AB6-8D99	:program control commands, statement parsing routines
8D9A-8ED1	:display control routines
8ED2-8F30	:machine code CALLs
90DC-925C	:program line numbering
925D-9303	:array DIMensioning routines
9304-9379	:setting BASIC parameters
937A-945A	:PROCedure and LOCAL variable processing
945B-9840	:BASIC VDU command handler
9841-9C02	:variable processing and setup routines
9C03-9E8F	:statement parsing and IMMEDIATE command handler
9E90-9E8F	:arithmetic routines
9E90-A03F	:PRINT formatting
A040-A6BB	:numeric input and floating point manipulation
A6BE-AB32	:higher order mathematical functions
AB33-AF0B	:miscellaneous parameters determination
AF0A-AFAC	:RaNDom number processing
AFAD-B0FD	:string handling
B0FE-B30C	:FN/PROC routines

```
B30D-B3C4   :variable manipulation
B3C5-B44B   :error and BRK handler
B44C-B49F   :BASIC sound command processing
B4A0-B667   :LISTing
B668-B887   :FOR/NEXT loop handling
B888-B9CE   :GOSUB, GOTO, ON routines
B9CF-BBA5   :INPUT and DATA retrieval
BBA6-BBFB   :REPEAT/UNTIL loop handling
BBFC-BD10   :BASIC program line input
BD11-BF2F   :program housekeeping and control
BF30-BFF8   :data file access commands
```

Basic 2 - Zero Page usage

```
    0 1 2 3 4 5 6 7 8 9 A B C D E F
00  * * * * * * * * * * * * * * * *
10  * * * * * * * * * * * * * * * *
20  * * * * * * * * * * * * * * * *
30  * * * * * * * * * * * * * * * *
40  * * * * * * * * * * * * * * * .
50  . . . . . . . . . . . . . . . .
60  . . . . . . . . . . . . . . . .
70  . . . . . . . . . . . . . . . .
80  . . . . . . . . . . . . . . . .
90  . . . . . . . . . . . . . . . .
A0  . . . . . . . . . . . . . . . .
B0  . . . . . . . . . . . . . . . .
C0  . . . . . . . . . . . . . . . .
D0  . . . . . . . . . . . . . . . .
E0  . . . . . . . . . . . . . . . .
F0  . . . . . . . . . . . . . . . *
    0 1 2 3 4 5 6 7 8 9 A B C D E F
```

8000-8070 - paged ROM protocol and initialisation

8000 :language entry point
8009 :ROM title and copyright message
801F :relocation address
8023 :ROM initialisation routines

8071-8450- BASIC keyword lookup tables

8071-836C :BASIC keyword, token and processing parameter lookup table
836D-83DE :BASIC keyword routine entry point LO lookup table
83DF-8450 :BASIC keyword routine entry point HI lookup table

8451-887B - assembler

8451-848A :assembler packed mnemonic lookup table 1
848B-84C4 :assembler packed mnemonic lookup table 2
84C5-84FC :assembler opcode lookup table
84FD :reset OPT and quit assembler
8504 :go to assembler entry point
8508 :process next source code character
8556 :PRINT source code with spaces where appropriate
8581 :set index to source code line length
85BA :process next source code character
85E1 :pack source code mnemonic
85F1 :derive opcode lookup table index from packed mnemonic
8607 :check for Boolean mnemonic
8620 :look up object code and transfer to buffer
862B :assemble object code in buffer
8673 :move source code address bytes to object code
8691 :check if error reporting is required
8696 :"Out of range" error routine - error code 01
86A8 :assemble 2 bytes of object code
86C5 :check source code and assemble
86CC :"Byte" error routine - error code 02
86FB :check for indirect, indexed instruction and process it
870D :"Index" error routine - error code 03
8735 :add 4 to first object code byte and assemble
8738 :check third byte of object code and assemble
8764 :go to "Index" error routine
879A :assemble 3 bytes of object code
87ED :go to "Index" error routine
8815 :save OPT value, zero index
8821 :evaluate expression, convert floating point to integer
8827 :set primary = secondary text buffer index
882C :add sixteen to first byte in object code buffer
882F :add eight to first byte in object code buffer
8832 :add four to first byte in object code buffer
8867 :go to "Type mismatch" error routine

887C-8AB5 - line editing routines

```
887C    :insert line number token, create space for line number instruction
8897    :convert decimal line number to HEX, pack and insert
88DA    :create space in program and insert line number instruction
88EC    :move text
88F5    :pack line number and insert
8926    :check for operational (carry clear) or alphanumeric (carry set)
        :character
8936    :check for ASCII decimal numeral character
893D    :check for decimal point or numeric character
8942    :get character from program and increment pointers
8944    :increment program pointers
894B    :increment program pointer, get next character
8951    :reset pointers, strip spaces, check for HEX
8955    :remove spaces from entire line, check for HEX
8957    :flush line to non-space character
8961    :check for HEX
897C    :check for " : , . *
898C    :check for : , . *
89A3    :check for .
89B5    :flush to numeral or decimal point
89C2    :reset secondary program pointers
89D2    :flush to next operational character
89DF    :check for alphanumeric character
89EC    :test for presence of keyword
89F8    :check against keyword lookup table
89FE    :seek token in keyword table
8A18    :search keyword table for token
8A25    :move pointers on to next keyword
8A37    :process token according to value of token processing parameter
8A72    :check if next character is operational or alphanumeric
8A8C    :get non-space character from secondary text buffer
8A97    :get non-space character from primary text buffer
8AA2    :"Missing ," error routine - error code 05
8AAE    :confirm next character is ","
```

8AB6-8D99 - program control commands, statement parsing routines

```
8AB6    :OLD command - token CB
8AC8    :END command - token E0
8AD0    :STOP command - token FA
8AD3    :"STOP" error routine - error code 00
8ADA    :NEW command - token CA
8ADD    :language entry routines
8AF3    :reset BASIC pointers and initialise RAM
8AF6    :initialise input buffers, BRKV, PRINT prompt, execute instruction
        :(IMMEDIATE command handler)
8B0B    :execute next instruction
```

```
8B41    :go to IMMEDIATE command handler
8B44    :go to assembler entry point
8B59    :"No FN" error routine - error code 07
8B60    :check for = * FN or [
8B73    :* command handler
8B7D    :process next statement
8B7D    :DATA command - token DC
8B7D    :DEF command - token DD
8B7D    :REM command - token F4
8B87    :check for ELSE and process next instruction
8B96    :decrement primary text buffer index, check termination, process
        :next statement
8B98    :check line termination, process next statement from primary text
        :buffer
8B9B    :process next statement from primary buffer
8BA3    :evaluate next statement
8BB1    :process BASIC command, using token as lookup table index
8BBF    :process next statement from primary text buffer
8BE4    :LET command - token E9
8C0B    :go to "Syntax error" routine
8C0E    :"Type mismatch" error routine - error code 06
8C1E    :retrieve parameters from BASIC stack and allocate string variable
8C21    :allocate string variable
8C84    :transfer variable name length to variables table, next variable pointer
        :to 2C,2D
8C97    :transfer variable name from Page 6 to variables table
8CA2    :save string from Page 6 buffer
8CB7    :"No room" error routine - error code 00
8CC1    :pull string off BASIC stack into buffer
8CEE    :transfer string from BASIC stack to variables table
8D03    :transfer variable from BASIC stack to variables table
8D2B    :PRINT# routine
8D4B    :pass integer variable to file
8D57    :pass floating point variable to file
8D64    :pass string variable to file
8D77    :recover channel number, save primary text buffer index, process next
        :statement
```

8D9A-8ED1 - display control routines

```
8D9A    :PRINT command - token F1
8DA6    :TAB according to formatting variable (@%) setting
8DBB    :set PRINT field, HEX flag, check line termination, PRINT buffer
8DD2    :PRINT contents of buffer
8E0E    :PRINT string from Page 6 buffer
8E21    :go to "Missing ," error routine
8E24    :do Y TABulation
8E40    :TAB( command - token 8A
8E58    :SPC command - token 89
```

```
8E58    :do X TABulation
8E67    :start new line, zero COUNT, reset primary text buffer index
8E6A    :copy secondary to primary text buffer index, clear carry
8E70    :check for formatting command (TAB, SPC)
8E8A    :seek formatting command, parenthetical string, PRINT it
8E98    :"Missing "" error routine - error code 09
8EA4    :output character, check for "
8EBD    :CLG command - token DA
8EC4    :CLS command - token DB
```

8ED2-8F30 - machine code CALLs

```
8ED2    :CALL command - token D6
8EE0    :process next CALL parameter
8F1B    :go to "No such variable" error routine
8F1E    :set carry, A-,X-,Y-registers, call machine code code subroutine
8F2E    :go to "Syntax error" routine
```

8F31-90DB - program line numbering

```
8F31    :DELETE command - token C7
8F69    :push line number to BASIC stack and STEP to stack
8F92    :set RENUMBER pointers to TOP, program pointers to PAGE+1
8F9A    :set program pointers to PAGE+1
8FA3    :RENUMBER command - token CC
8FD6    :"RENUMBER space" error routine - error code 00
8FDF    :"Silly" error routine - error code 00
9043    :check number of next line
9071    :move program pointers to next line, increment RENUMBER space
          pointers
908D    :display line number and check next line
909F    :move program pointers to start of next line
90AC    :AUTO command - token C6
```

90DC-925C - array dimensioning routines

```
90DC    :go to "DIM space" error routine
90DF    :reserve memory space
9127    :"Bad DIM" error routine - error code 0A
912F    :DIM command - token DE
916B    :DIMension array
91B7    :DIMension array element
920B    :check for further array elements to DIMension
9215    :go to DIM command routine
9218    :"DIM space" error routine - error code 0B
9222    :increment integer buffer
9231    :multiply integer buffer by integer from BASIC stack
9236    :multiply integer in 2A,2B by integer in 3F,40
925A    :go to "Bad DIM" error routine
```

925D-9303 - setting BASIC parameters

```
925D    :HIMEM command - token D3
926F    :LOMEM command - token D2
9283    :PAGE command - token D0
928D    :CLEAR command - token D8
9295    :TRACE command - token FC
92B7    :set TRACE flag on, reset maximum TRACE line number
92C0    :reset TRACE flag, process next statement
92C9    :TIME command - token D1
92DA    :confirm next character is ",", evaluate expression, result
        :in integer buffer
92DD    :evaluate expression, get result in integer buffer
92E3    :get variable type from program, confirm numeric, convert floating
        :point to integer
92EB    :set secondary = primary text buffer, confirm numeric, convert
        :floating point to integer
92EE    :get variable type, confirm numeric, convert floating point to
        :integer
92F0    :confirm numeric variable and convert floating point to integer
92FA    :confirm numeric variable, convert integer to floating point
92FD    :confirm numeric variable, convert integer to floating point
```

9304-9379 - PROCedure and LOCAL variable handling

```
9304    :PROC command - token F2
931B    :clear integer LOCAL variable
9323    :LOCAL command - token EA
9341    :check for another LOCAL variable
9353    :check line termination, process next statement
9356    :ENDPROC command - token E1
9365    :"No PROC" error routine - error code 0D
936B    :"Not LOCAL" error routine - error code 0C
9372    :"Bad MODE" error routine - error code 19
```

937A-945A - BASIC VDU command handlers

```
937A    :GCOL command - token E6
938E    :COLOUR command - token FB
939A    :MODE command - token EB
93DA    :output character from 6502 stack, then from integer buffer, process
        :next statement
93E4    :MOVE command - token EC
93E8    :DRAW command - token DF
93F1    :PLOT command - token F0
93FD    :process parameter and execute VDU25 command
942A    :send second character from integer buffer to current output stream
942F    :VDU command - token EF
9453    :decrement primary text buffer index, check line termination, process
```

:next statement
9456 :send first character from integer buffer to current output stream

945B-9840 - variable processing and setup routines

945B :search for FN/PROC
9469 :search for variable pointed by 37,38
9479 :check variable name for identity
949A :search variables table for name
94B3 :check identity of variable name, letter by letter
94E1 :increment pointers by value of index
94ED :set up FN/PROC references
94FC :set up new variable name
9516 :add address to variables chain
9531 :clear space for variable
9539 :increment TOPOFVARS pointer, check if space for another variable
9541 :check if space for another variable
9553 :go to "No room" error routine
9559 :set Y=1, count alphanumeric characters in variable name
955B :count alphanumeric characters in variable name
957F :clear space and set up variable reference
9582 :set up variable references (address 2A,2B type 2C)
9595 :check for indirection operator
95A5 :! indirection routine
95A7 :? indirection routine
95B0 :$ indirection routine
95BF :"$ range" error routine - error code 08
95C9 :set secondary text buffer = primary text buffer, decrement primary :index
95D5 :strip spaces, set up variable reference
95DD :set up variable reference
95FF :set up array reference
961B :check next character from variable name
9641 :determine variable type and set up references
964C :set up integer reference
9661 :check for indirection operator
967B :process "?" operator
967F :process "!" operator
969F :recover variable type from stack, return with A- true, carry clear
96AF :set up string references
96C9 :set up string array references
96D7 :"Array" error routine - error code 0E
96DF :set up array references
96F7 :calculate space required for array, adjust variables table pointers
97AD :use array dimension to move pointers on
97BA :check array subscript size
97D1 :"Subscript" error routine - error code 0F
97DD :seek line number, if found, unpack to 2A,2B, set carry
97DF :if line number token present, unpack number to 2A,2B, carry set

```
97EB        :unpack line number to 2A,2B
9807        :set secondary = primary text buffer, check = , evaluate RHS of
            :expression
9813        :check for = , evaluate RHS of expression
9821        :"Mistake" error routine - error code 04
982A        :"Syntax error" routine - error code 10 (and token CE)
9838        :"Escape" routine - error code 11
```

9841-9C02 - statement parsing and IMMEDIATE mode handler

```
9841        :confirm next character is =
9849        :evaluate expression
984C        :transfer X- to A-register, check termination
9852        :evaluate integer expression, set secondary text buffer index, strip
            :spaces, test line termination
9857        :set primary text buffer index and test line for correct termination
9859        :strip spaces and test line for correct termination
9861        :check line termination, check buffer index
986D        :increment primary text pointers and test ESCAPE flag
9877        :set primary text buffer index = 1, test ESCAPE flag
987B        :test ESCAPE flag
9880        :move to next statement with TRACE if required
9890        :increment index and check if TRACE required
98AC        :use index to reset pointers, reset index to 1
98BC        :go to IMMEDIATE command handler
98BF        :go to "Type mismatch" error routine
98C2        :IF command - token E7
98E1        :increment primary text buffer index and process ....
98E3        :THEN command - token 8C
98F1        :search for ELSE, process it if found
9905        :display TRACE marker if line number below set limit
991F        :output decimal equivalent of number in 2A,2B
9923        :output decimal equivalent of current line #
996B-996F   :HEX to decimal conversion lookup table 1
9970        :search BASIC program for given line number, carry set if not found
99A7        :"Division by zero" error routine - error code 12
99B9-99BD   :HEX to decimal conversion lookup table 2
99BE        :confirm variable numeric, convert floating point to integer, store in
            :integer buffer, 39-3C
99E8        :perform division process
9A39        :unpack variable to floating point buffer #1 and compare with floating
            :point buffer #2
9A50        :save floating point buffer #1 to BASIC stack, unpack variable to floating
            :point buffer #2 compare with floating pooint buffer #1
9A5F        :unpack variable to floating point buffer #2 and compare with
            :floating point buffer #1
9A62        :check floating point buffer #1 and floating point buffer #2 for
            :identity
9A93        :test sign of floating point buffer #2
```

```
9A9A    :go to "Type mismatch" error routine
9A9D    :transfer X- to A-register, push variable to BASIC stack, check
        :for + or -
9A9E    :compare identity of variables
9AB3    :compare integer variable for identity
9AE7    :compare identity of strings
9B1D    :equate primary and secondary text buffers, evaluate next statement
9B29    :evaluate expression (main entry point)
9B3A    :OR command - token 84
9B55    :EOR command - token 82
9B6B    :process < > = OR or AND, if present
9B72    :check for < = > and AND
9B7A    :AND command - token 80
9B9C    :process < = > + - operators if present
9BA8    :process < = > operators
9BC0    :process < operator
9BD4    :process > = operator
9BDF    :process < > operators
9BE8    :process > operator
9BFA    :process > = operators
```

9C03-9E8F - arithmetic routines

```
9C03    :"String too long" error routine - error code 13
9C15    :concatinate strings
9C42    :check for + or - operator
9C4E    :process + operation
9C5B    :integer addition
9C77    :reset BASIC stack pointer, check for another operator
9C88    :go to "Type mismatch" error routine
9C9B    :perform floating point addition
9CA1    :transfer variable type to X-register, check for + or -
9CA7    :pull integer from BASIC stack and perform floating point addition
9CB5    :perform - operation
9CC2    :perform integer subtraction
9CE1    :subtract floating point variable
9CFA    :convert integer to floating point and perform floating point subtraction
9D0E    :convert integer to floating point
9D1D    :convert integer to floating point and perform floating point
        :multiplication
9D20    :perform floating point multiplication
9D2C    :multiply floating point buffer #1 by current variable, check for
        :further multiplication or division operation
9D39    :go to "Type mismatch" error routine
9D3C    :perform multiplication
9D5C    :integer multiplication
9DBB    :transfer quotient to integer buffer
9DBD    :transfer indexed part of quotient to integer buffer
9DCB    :go to multiplication routine
```

9DCE :push integer on BASIC stack, evaluate expression, check for :multiplication or division operator
9DD1 :evaluate expression, check for multiplication or division operator
9DD4 :check for multiplication or division operator
9DE5 :floating point division routine
9E01 :DIV routine
9E0A :MOD routine
9E1D :push integer on BASIC stack, evaluate expression, get character in :in X-register
9E20 :evaluate expression get non-space character in X-register
9E35 :process exponentiation
9E6F :use natural logs to evaluate power
9E88 :large number exponentiation

9E90-A03F - PRINT formatting

9E90 :assemble HEX output for PRINT
9EB0 :convert HEX to ASCII
9EC8 :assemble floating point output string
9ED1 :check if exponent is not 81 or more
9EDF :assemble formatted PRINT output string
9EF9 :assemble PRINT output string in general format
9F1D :go to non-zero formatting parameter PRINT routine
9F25 :adjust position of decimal point for PRINTing
9F31 :divide by ten, adjusting power in decimal index store
9F9C :PRINT with non-zero format parameter
9FA0 :PRINT fixed number of decimal places
9FDB :insert required number of decimal places after decimal point
9FE8 :save next digit to buffer
A015 :PRINT in exponential format

A040-A6BB - numeric input and floating point manipulation

A040 :transfer ASCII equivalent of decimal to string buffer
A052 :transfer ASCII equivalent of lo nibble to string buffer
A05A :convert character to ASCII, store in string buffer
A064 :add ASCII offset, store character in string buffer, increment :pointer
A066 :store character in Page 6 string buffer, increment pointers
A072 :save rounding byte, get sign of floating point buffer #2, set A = FF
A07B :initialise floating point buffer #1 mantissa and workspace
A08B :input ASCII character and check validity
A099 :input next ASCII character, enter in buffer if numeral
A0A0 :process decimal point
A0A8 :enter ASCII numeral into floating point buffer #1
A0E1 :process exponentiation
A111 :exponentiation with negative index
A11F :exponentiation with index = 0
A139 :process -

```
A140    :process + or - , load next decimal digit if present
A14B    :add number
A158    :increment index, add numeral to ten times value of temporary store
A170    :recover number from temporary store, clear carry
A178    :add floating point buffer #2 mantissa to floating point buffer #1,
        :increment exponent if necessary
A197    :multiply by ten, preserving A-register
A1A5    :multiply floating point buffer #1 mantissa by 4
A1DA    :get sign of floating point buffer #1
A1F4    :multiply floating point buffer #1 by ten
A1FB    :multiply floating point buffer #1 by 1.25 decimal
A208    :add corresponding byte of floating point buffer #2 mantissa to
        :floating point buffer #1, increment exponent if necessary
A20B    :divide floating point buffer #1 by 2 and increment exponent
A21E    :copy floating point buffer #1 to floating point buffer #2
A220    :set floating point buffer #2 sign byte, copy remaining bytes from
        :floating point buffer #1
A23F    :copy floating point buffer #1 to floating point buffer #2, divide
        :by 2
A242    :divide floating point buffer #1 mantissa by 2
A24D    :divide floating point buffer #1 by ten
A2A4    :add A- to floating point buffer #1, increment exponent if necessary
A2BE    :convert integer to floating point
A2E6    :set floating point buffer #1 sign, exponent and guard bytes
A2ED    :adjust floating point buffer #1 sign and normalise
A303    :normalise floating point buffer #1
A336    :shift floating point buffer #1 mantissa until all bytes are
        :significant
A34E    :unpack variable into floating point buffer #2
A37D    :pack floating point buffer #1 to memory at 0471
A381    :pack floating point buffer #1 to memory at 0476
A385    :pack variable from floating point buffer #1 to 046C
A387    :pack floating point buffer #1 to current variable location (4A,4B)
A38D    :pack floating point buffer #1 into memory
A3B2    :unpack variable at 046C to floating point buffer #1
A3B5    :unpack variable into floating point buffer #1
A3E4    :position binary point, convert floating point to integer
A3E7    :transfer contents of floating point buffer #1 mantissa to integer
        :buffer
A3FE    :shift binary point until exponent = A0
A40C    :halve floating point buffer #1 mantissa, compensating exponent until
        :it is = A0
A43C    :divide floating point buffers #1 and #2 mantissas by 2, compensating
        :exponent
A450    :go to "Too big" error routine
A453    :clear floating point buffer #2
A46C    :complement floating point buffer #1 mantissa
A486    :check sign of floating point buffer #1 exponent, adjust mantissa,
        :set index store 4A
```

A4B0 :complement floating point buffer #1 mantissa, normalise floating point buffer #1
A4B6 :increment mantissa of floating point buffer #1
A4C7 :reduce floating point buffer #1 mantissa magnitude by 1
A4D0 :subtract variable from floating point buffer #1
A4D6 :interchange variable in memory and floating point buffer #1
A4E8 :move mantissa from floating point buffer #2 to floating point buffer #1
A4FD :change sign of floating point buffer #1, unpack variable to floating point buffer #2, algebraicly sum
A500 :unpack variable to floating point buffer #2, algebraicly sum floating point variables
A505 :align binary points, add floating point buffer #1 to floating point buffer #2, round if necessary
A50B :align binary points, algebraicly sum floating point variables, round if necessary
A590 :take algebraic sum of mantissas
A5B7 :take smaller mantissa from larger
A5E3 :subtract floating point buffer #2 mantissa from floating point buffer :#1, result in floating point buffer #1
A606 :if floating point buffer #1 negative, unpack variable to floating point buffer #2 and multiply
A656 :multiply floating point buffer #1 by current variable
A659 :normalise floating point buffer #1 and round it up
A65C :round floating point buffer #1 if necessary
A66C :"Too big" error routine - error code 14
A676 :round floating point buffer #1 up
A67C :zero rounding byte, test overflow, clear floating point buffer #1
A686 :clear floating point buffer #1
A699 :set floating point buffer #1 = -1
A6A5 :invert floating point buffer #1 and change its sign
A6AD :divide variable by floating point buffer #1, quotient in floating point buffer #1
A6BB :go to "Division by zero" error routine

A6BE-AB32 - higher order mathematical functions

A6BE :TAN command - token B7
A6E7 :divide floating point buffer #1 by current variable (4B,4C)
A6F1 :divide floating point buffer #1 by floating point buffer #2
A70A :calculate mantissa
A7A9 :"-ve root" error routine - error code 15
A7B4 :SQR command - token B6
A7B7 :calculate square root of floating point numeral
A7E9 :set current variable pointer to 047B
A7ED :set current variable pointer to 0471
A7F1 :set current variable pointer to 0476
A7F5 :set current variable pointer to 046C
A7FE :LN command - token AA

```
A801          :check sign of variable and calculate LN
A808          :"Log range" error routine - error code 16
A814          :calculate LN
A869          :constant (0.434 294 482) used for LN/LOG conversion
A86E          :constant (0.693 147 181) used in LN calculation
A873-A896     :LN power series
A897          :carry out power series calculation
A8D4          :ACS command - token 95
A8DA          :ASN command - token 98
A8EA          :calculate ASN for positive values
A8FE          :unpack PI/2 to floating point buffer #1
A907          :ATN command - token 99
A90A          :calculate ATN of angle
A916          :reset floating point buffer #1 sign byte negative
A91B          :calculate ATN, inverting angle if > PI/4
A927          :subtract angle from PI/2
A936          :carry out ATN calculation
A95A-A98C     :ATN power series
A98D          :COS command - token 9B
A998          :SIN command - token B5
A99E          :carry out SIN/COS calculation, adjust angle and apply correction if
              :necessary
A9AA          :carry out SIN/COS calculation, with correction if necessary
A9B1          :apply correction to result
A9C3          :carry out SIN/COS calculation
A9D3          :adjust angle for SIN/COS calculation
AA38          :"Accuracy lost" error routine - error code 17
AA48          :set current variable pointer to AA59 (coarse -PI/2)
AA4C          :set current variable pointer to AA5E (correction factor)
AA55          :set current variable pointer to AA63 (PI/2)
AA59          :constant (-1.570 800 78)
AA5E          :constant (4.454 455 11 E-6)
AA63          :constant (PI/2 - 1.570 796 33)
AA68          :constant (1/57.295 779 6 - radian/degree conversion factor)
AA6D          :constant (57.295 779 5 - degree/radian conversion factor)
AA72-AA90     :SIN/COS power series
AA91          :EXP command - token A1
AA94          :calculate EXP of floating point variable
AAAC          :"Exp range" error routine - error code 18
AAD1          :multiply floating point buffer #1 by variable at 0476
AADA          :carry out power series calculation with EXP power series
AAE4          :'e' (2.718 281 83)
AAE9-AB11     :EXP power series
AB12          :raise floating point variable to power specified in temporary power
              :store (4A)
AB25          :multiply by current variable, adjusting index
```

AB33-AF0B - miscellaneous parameters determination

```
AB33    :ADVAL command - token 96
AB41    :POINT( command - token B0
AB6D    :POS command - token B1
AB76    :VPOS command - token BC
AB7F    :set integer buffer according to sign of floating point buffer #1
AB88    :SGN command - token B4
ABA8    :LOG command - token AB
ABB1    :RAD command - token B2
ABC2    :DEG command - token 9D
ABCB    :PI command - token AF
ABD2    :USR command - token BA
ABE6    :go to "Type mismatch" error routine
ABE9    :EVAL command - token A0
AC23    :restore secondary text buffer parameters, set A= variable type
AC2F    :VAL command - token BB
AC34    :calculate VAL of string in buffer
AC73    :save variable type, reset secondary text buffer parameters, A =
        :variable type
AC78    :INT command - token A8
AC9B    :go to "Type mismatch" error routine
AC9E    :ASC command - token 97
ACAD    :INKEY command - token A6
ACB8    :EOF command - token C5
ACC4    :TRUE command - token B9
ACC4    :set each byte of integer buffer true
ACD1    :NOT command - token AC
ACE2    :INSTR( command - token A7
AD67    :go to "Type mismatch" error routine
AD6A    :ABS command - token 94
AD71    :make integer buffer positive
AD77    :take modulus of integer buffer
AD7E    :change sign of floating point buffer #1, set A=FF
AD8F    :change sign of numeric variable
AD93    :change sign of integer variable, set A=40
ADAD    :transfer line from secondary text buffer to Page 6, set A=0
ADC9    :transfer string, if legal, to Page 6 buffer
ADE1    :save STRINGLENGTH, buffer index, set A=0
ADE9    :go to "Missing "" error routine
ADEC    :evaluate single part of expression - final
AE02    :process next character
AE30    :check OPT value and transfer assembly address to integer buffer
AE3A    :transfer assembly next address to integer buffer
AE43    :"No such variable" error routine - error code 1A
AE56    :evaluate expression and look for ")"
AE61    :"Missing )" error routine - error code 1B
AE6D    :convert ASCII character to HEX integer
AEAA    :"Bad HEX" error routine - error code 1C
```

```
AEB4    :TIME command - token 91
AEC0    :PAGE command - token 90
AECA    :FALSE command - token A3
AECE    :go to "Type mismatch" error routine
AED1    :LEN command - token A9
AED8    :transfer A- to integer buffer, clearing higher bytes
AEDC    :TO command - token B8
AEDC    :TOP command - token B8+50
AEEA    :transfer A-,Y-registers to integer buffer, zero hi byte, set A=40
AEF7    :COUNT command - token 9C
AEFC    :LOMEM command - token 92
AF03    :HIMEM command - token 93
```

AF0A-AFAC - RaNDom number processing

```
AF0A    :process RaNDom number
AF24    :RND(N) routine
AF3F    :RND(-N) routine
AF49    :RND command - token B3
AF56    :transfer 00,X-00,X+3 to integer buffer, set A=40
AF69    :RND(1) routine
AF6C    :RND(0) routine
AF87    :shuffle random number store 32 times
AF9F    :ERL command - token 9E
AFA6    :ERR command - token 9F
```

AFAD-B0FD - string handling

```
AFAD    :get character in X-register with time limit, Y=0 - success
AFB9    :GET command - token A5
AFBF    :GET$ command - token BE
AFC2    :save to Page 6, set string length=1, A=0
AFCC    :LEFT$( command - token C0
AFEE    :RIGHT$( command - token C2
B026    :INKEY$ command - token BF
B033    :go to "Type mismatch" error routine
B036    :go to "Missing ," error routine
B039    :MID$( command - token C1
B083    :reposition string within Page 6 buffer
B094    :STR$ command - token C3
B0BF    :go to "Type mismatch" error routine
B0C2    :STRING$( command - token C4
B0FB    :go to "String too long" error routine
```

B0FE-B30C - FN/PROC routines

```
B0FE    :restore primary text pointers and execute ....
B104    :"No such FN/PROC" error routine - error code 1D
B112    :search for DEF in BASIC program
```

B11A :flush program to DEF token
B13C :process DEF token
B18A :"Bad call" error routine - error code 1E
B195 :FN command - token A4
B197 :process FN/PROC
B1A3 :save 6502 stack to BASIC stack
B1C8 :increment index and set program pointers
B1CA :set program pointers
B1E9 :set primary text pointers to next variable
B1F4 :check for LOCAL variables
B202 :set up LOCAL variable
B21A :discard LOCAL variables
B230 :restore 6502 stack from BASIC stack
B24D :set up LOCAL variables in FN/PROC
B28E :process LOCAL variable arguments
B2B5 :reset stack pointer, secondary text pointer, execute ....
B2BE :"Arguments" error routine - error code 1F
B2CA :save global variables
B2F3 :save numeric LOCAL variable to stack
B303 :save LOCAL variable to stack

B30D-B3C4 - variable handling

B30D :store variable, save reference to BASIC stack
B32C :unpack variable into appropriate buffer
B34F :transfer variable address to A-,Y-registers
B354 :unload variable to floating point buffer #1
B384 :unload string variable from variables table to Page 6
B39D :transfer string variable from variables table to Page 6
B3A7 :transfer string pointed by 2A,2B to Page 6 buffer
B3BD :CHR$ command - token BD

B3C5-B44B - error and BRK handler

B3C5 :reset ERRL pointer, set program pointers to PAGE
B3F9 :check if primary text pointers are beyond program pointers
B402 :BRK routine
B433 :error REPORT embedded message

B44C-B49F - BASIC sound command processing

B44C :SOUND command - token D4
B472 :ENVELOPE command - token E2
B48F :set up control block, process OSWORD call

B4A0-B667 - LISTing

B4A0 :WIDTH command - token FE
B4AE :go to "Type mismatch" error routine

```
B4B1    :evaluate expression and pull variable from BASIC stack
B4B4    :pull variable from BASIC stack
B4B7    :assign value to numeric variable
B4C6    :transfer integer variable from integer buffer to variables table
B4E0    :confirm variable numeric, convert integer to floating point and ....
B4E9    :pack floating point variable from floating point buffer #1 to
        :variables table
B50E    :output character in A-register, decrunching if token
B51E    :zero index and find keyword in table
B520    :check next keyword in table
B536    :expand keyword from token
B545    :PRINT hexadecimal numeral
B550    :convert hexadecimal to ASCII and PRINT it
B558    :if character is CR, start new line, else, PRINT it, updating COUNT
B562    :PRINT hexadecimal numeral + SPACE
B565    :if there is room, PRINT a space
B577    :PRINT line with leading spaces according to LISTO setting
B58A    :LISTO command - token C9+4F
B59C    :LIST command - token C9
B5FC    :check if new line is above finish line
B61D    :LIST line from program
B637    :PRINT next character
B659    :unpack line number and PRINT it
```

B668-B887 - FOR/NEXT loop handling

```
B668    :check loop status and set flags
B68E    :"No FOR" error routine - error code 20
B695    :NEXT command - token ED
B6C7    :"Can't match FOR" error routine - error code 21
B6D7    :check loop variable type and process it accordingly
B6EA    :process FOR/NEXT loop with integer variable
B741    :reset primary text pointers, proceed with program
B751    :decrement FOR/NEXT loop counter
B766    :process FOR/NEXT loop with floating point variable
B7A1    :decrement primary text buffer index, check line termination, process
        :next statement
B7A4    :"FOR variable" error routine - error code 22
B7B0    :"Too many FORs" error routine - error code 23
B7BD    :"No TO" error routie - error code 24
B7C4    :FOR command - token E3
B837    :store FOR/NEXT loop address and increment counter
B84F    :save FOR/NEXT loop parameters to memory
```

B888-B9CE - GOSUB, GOTO, ON routines

```
B888    :GOSUB command - token E4
B88B    :process GOSUB routine
B8A2    :"Too many GOSUBs" error routine - error code 25
```

```
B8AF    :"No GOSUB" error routine - error code 26
B8B6    :RETURN command - token F8
B8CC    :GOTO command - token E5
B8D2    :process GOTO routine
B8DD    :reset primary text buffer pointers, process next statement
B8E4    :test line termination, reset ERRV, process next statement
B8F2    :ON ERROR routine
B90A    :"ON syntax" error routine - error code 27
B915    :ON command - token EE
B98B    :"ON range" error routine - error code 28
B99A    :check specified line number is present
B9AF    :seek given line number, carry clear if found
B9B5    :"No such line" error routine - error code 29
B9C4    :go to "Type mismatch" error routine
B9C7    :go to "Syntax error" routine
```

B9CF-BBA5 - INPUT and DATA retrieval

```
B9CF    :INPUT# command - token E8+23
B9DA    :get next character from primary text buffer
BA19    :INPUT numeric variable
BA1F    :INPUT integer variable
BA2B    :INPUT floating point variable
BA3F    :discard channel number and flag, process next statement
BA44    :INPUT command - token E8
BA5A    :INPUT next variable
BAD0    :derive variable from INPUT string
BADC    :allocate string variable, INPUT next variable
BAE6    :RESTORE command - token F7
BB15    :check for further DATA, else, process next statement
BB1F    :READ command - token F3
BB40    :move DATA pointers on, process further DATA or next statement
BB50    :seek next DATA item if present
BB9C    :"Out of DATA" error routine - error code 2A
```

BBA6-BBFB - REPEAT/UNTIL loop handling

```
BBA6    :"No REPEAT" error routine -error code 2B
BBB1    :UNTIL command - token FD
BBCD    :transfer address of next REPEAT/UNTIL loop to primary text pointers
BBD6    :"Too many REPEATs" error routine - error code 2C
BBE4    :REPEAT command - token F5
```

BBFC-BD10 - BASIC program line input

```
BBFC    :read INPUT line from current stream to Page 6
BC02    :PRINT character in A-register as PROMPT, start new line, zero COUNT
        :read line from current stream to Page 7
BC25    :start new line and zero COUNT
```

```
BC28    :zero COUNT
BC2D    :search for given line number, carry set if not found
BC5D    :increment index, pointers and examine next character
BC7C    :increment TOP setting, clear carry
BC81    :transfer next character to TOP
BC8D    :insert BASIC line into program
BCCC    :"LINE space" error routine - error code 00
BCD6    :make space and insert new line
```

BD11-BF2F - program housekeeping and control

```
BD11    :RUN command - token F9
BD14    :RUN program
BD20    :reset BASIC pointers
BD2F    :zero variables except system integers
BD3A    :reset variables table pointers and loop counters
BD51    :pack floating point variable on to BASIC stack
BD7E    :increment BASIC stack pointer, set current variable pointers (4B,4C)
BD90    :push value of variable on to BASIC stack
BD94    :push integer variable on to BASIC stack
BDB2    :push string variable to BASIC stack
BDCB    :transfer string from BASIC stack to Page 6
BDDC    :get STRINGLENGTH from BASIC stack, move pointers up
BDE1    :move BASIC stack pointers up
BDEA    :pull integer buffer from BASIC stack
BDFF    :increment BASIC stack pointers by 4
BE0B    :pull variable off BASIC stack to 37-3A
BE0D    :pull integer from BASIC stack to 00,X-00,X+3
BE2E    :if enough memory, move BASIC stack pointers down
BE41    :go to "No room" error routine
BE44    :transfer value from integer buffer to 00,X-00,X+3
BE55    :clear carry and increment line search/crunch pointer
BE56    :increment line search/crunch pointers
BE62    :LOAD program, set TOP and check if program is legal
BE6F    :relink BASIC program
BE92    :reset TOP to TOP + Y, set Y = 1
BE93    :add A- to TOP
BE9E    :PRINT "Bad program" and return to immediate mode
BEA1    :"Bad program" embedded message
BEB2    :set up filename buffer
BEBA    :terminate filename/string with CR
BEC2    :OSCLI command - token FF
BECF    :go to "Type mismatch" error routine
BED2    :get filename for LOAD/SAVE
BEDD    :set up LOAD/SAVE parameters
BEE7    :set machine higher order address pointers
BEF3    :SAVE command - token CD
BF24    :LOAD command - token C8
BF2A    :CHAIN command - token D7
```

BF30-BFF8 - data file access routines

```
BF30    :PTR command - token CF
BF46    :EXT command - token A2
BF47    :PTR command - token 8F
BF58    :BPUT command - token D5
BF6F    :BGET command - token 9A
BF78    :OPENIN command - token 8E
BF7C    :OPENOUT command - token AE
BF80    :OPENUP command - token AD
BF96    :go to "Type mismatch" error routine
BF99    :CLOSE command - token D9
BFA9    :input channel number into A-,Y-registers
BFB5    :get channel number from primary text buffer
BFC3    :"Missing #" error routine - error code 2D
BFCF    :process following message embedded in ROM
BFE4    :REPORT command - token F6
BFF9    :"Roger" embedded message
```

Page 0

```
0000-0001 :LOMEM
0002-0003 :TOPOFVARIABLES pointer
0004-0005 :BASIC stack pointer
0006-0007 :HIMEM
0008-0009 :ERRL
000A      :primary text index
000B-000C :primary text pointer
000D-0011 :RND number store
0012-0013 :TOP
0014      :number of characters in PRINT field
0015      :HEX PRINT flag (bit 7 denotes status)
0015      :variables table index
0016-0017 :ERRV (default B402)
0018      :PAGE hi
0019-001A :secondary text pointer
001B      :secondary text index
001C-001D :DATA pointer
001E      :COUNT
001F      :LISTO option
0020      :TRACE flag (hi bit indicates status)
0021-0022 :maximum TRACE line number
0023      :WIDTH
0024      :REPEAT/UNTIL loop counter
0025      :GOSUBS counter
0026      :FOR/NEXT loop counter (fifteen times number of loops)
0027      :variable type evaluator parameter
0028      :OPT value for assembler
0029-002B :assembly buffer
002A-002D :integer buffer
002A      :integer buffer least significant byte
002A-002B :variable pointer
002A-002B :STEP
002C      :variable type
002C-002D :variables table pointer
002D      :integer buffer most significant and sign byte
002E-0035 :floating point buffer #1
002E      :          sign byte
002F      :          guard byte
0030      :          exponent
0031      :          mantissa most significant byte
0032      :          mantissa
0033      :          mantissa
0034      :          mantissa least significant byte
0035      :          rounding byte
0036      :length of string
0037-0038 :assembled code location
0037      :OSWORD control block - least significant byte of input line buffer
```

0037-0038 :RENUMBER pointer
0037-0038 :DIM variable name pointer
0037-0038 :program pointer
0037 :PRINT format parameter
0037-0042 :SOUND parameter
0038 :OSWORD control block - most significant byte of input line buffer
0038 :temporary store for sign during division calculation
0038 :number of decimal places in PRINT field
0039 :length of variable name
0039 :length of object code operation
0039-003A :secondary program pointer
0039-003A :BASIC keyword table pointer
0039-003A :DELETE range upper limit line number
0039-003C :integer variable temporary store
003A-003B :variable name pointer
003A-003B :current ORG address
003A :string length used in string comparison
003A :keyword table index
003B-0042 :floating point buffer #2
003B :floating point buffer #2 sign byte
003B :input buffer index
003B :index temporary store
003B-003C :RENUMBER space pointer
003C :floating point buffer #2 guard byte
003C :text index temporary store
003C :most significant byte of higher order address
003C-003D :next variable pointer
003D :floating point buffer #2 exponent
003D-003E :assembler opcode lookup table index calculation workspace
003D-003E :line number temporary store
003D :keyword token processing parameter
003D-003E :next line pointer
003E :floating point buffer #2 mantissa most significant byte
003F :floating point buffer #2 mantissa
003F :variable length
003F :array type
003F-0040 :buffer used in DIM calculation
003F :LOCAL variables counter
003F :line length
003F-0048 :LOAD/SAVE parameter
0040 :floating point buffer #2 mantissa
0041 :floating point buffer #2 mantissa least significant byte
0042 :floating point buffer #2 rounding byte
0043 :temporary store used in decimal numeral input
0043-0046 :work area used in tangent calculation
0044 :ENVELOPE number
0048 :index input temporary store
0048 :power series calculation iteration index store
0049 :decimal index

0049 :index working store used in exponent calculation
004A :temporary store for power used in exponent calculation
004A :temporary store used in decimal numeral input
004A :temporary angle store used in trigonometric calculation
004B-004C :current variable pointer
004B-004C :stack pointer
004D :channel number temporary store
004D-004E :power series pointer
004D :LOCAL variables arguments counter
004D :PRINT status flag
004E :PRINTFIELD parameter
004E :LOCAL variables counter
004E :input flag

Page 1

0100-01FF :6502 stack
0100-01FF :storage of LOCAL variables in FuNctions or PROCedures
0100- :error handling routine relocated to base of 6502 stack

Page 4

0400-046B :system integer variables @%-Z%
046C-047F :temporary storage for floating point variables during program execution
0480-04EC :start of variable address chains
04F6-0457 :start of PROC chain
04F8-04F9 :start of FN chain
04FA-04FF :unused

Page 5

0500-05A3 :FOR/NEXT loop parameter stores (up to ten blocks of fifteen bytes)
05A4-05B7 :REPEAT/UNTIL loop start address hi bytes (up to twenty)
05B8-05CB :REPEAT/UNTIL start address lo bytes
05CC-05E5 :GOSUB return address lo bytes (up to twenty-six)
05E6-05FF :GOSUB return address hi bytes

Page 6

0600-06FF :string buffer
0600-06FF :CALL parameter block

Page 7

0700-07FF :input buffer

user program area

```
PAGE-TOP           :Basic program
LOMEM-VARTOP       :variables table
VARTOP-STCKBASE    :RENUMBER space
STCKBASE-HIMEM     :Basic stack
```

JMP/JSR :calling location
address :address

```
(002A) :8F2B
(0037) :8BBC BFE1
(WRCHV):9458 B574
8504   :8B44
8508   :85A2
8556   :8562
85BA   :8514
862B   :86AA 879C 881E 8864
86A8   :873C
86C5   :877D 87C9
86CC   :880D
870D   :8764 87AF 87ED
8735   :8785 87DB 87EA
8738   :8810
879A   :8732
8821   :8677 86C5 86DE 8717 8750 8782 8797 87A5 87CE 87F0 8815 885A
       :9188 92A1 9379 9390 939C 93F0 9440 B44C B472 B4A0
8827   :85B7 8875 9121
882C   :86EF 8721 875A 879F 87A2 87E7 8806
882F   :86C2 872F 882C
8832   :8735 8767 882F
887C   :88DD 8A55
8897   :89B0
88F5   :906A
8926   :89CB 89D4 8A43 8A74 B167
8936   :896D 89A7
893D   :89B7
8942   :B3D9 B3E8 B3F1 B3F6
8944   :8919 891C 891F 894B 8961 89BC 89D9 8A79
894B   :896A 8980 BFDC
8951   :90C3
8955   :AC1A
8957   :89C8 8B2A
8961   :89E9 8A89
89B5   :89BF
89D2   :89DC
89F8   :8A34
8A72   :8A7C
8A8C   :8AAE 8D32 8E43 8EE3 9841 AC50 AC5B AC66 ADAD AE02 B094 B287 B2A0
       :B7EA B813 B866 BABD BB60 BB6F BFB5
8A97   :8508 85BC 86BB 86D7 86E1 86E8 86F2 86FF 8706 871A 8724 8747 8753
       :875D 8776 878E 87A8 87C1 87D4 87DE 87F6 87FD 8B38 8D89 8D9A 8DC3
       :8E8A 8F39 8F79 912F 91A9 920B 9348 942F 9446 B13F B1F9 B279 B5BE
       :B5CF B75C B8F2 B915 B9DA BA44 BAEE BB15
8AA2   :8E21 AD03 B036
8AAE   :92D9 93F6 AB47 B0C8 BF5C
```

```
8ADD :806E
8AF3 :8B35 8F66 903A 90D9 BF27
8AF6 :859C 8B41 98BC B599 B61A BEAF
8B0B :BD1D
8B7D :B907
8B87 :9902
8B96 :8D80 9212 934F 9453 B7A1 BB1C
8B98 :8D7A 9352 B9CC BA41
8B9B :8BF8 8C08 8ECF 8F18 926B 9289 92B3 92D6 9317 93E0 9426 B49D B4AB
     :B8C9 B8EF BB12 BBCA BECC BF21 BF43 BF6C BFA6 BFF6
8BA3 :8501 98DE B20B B430 B74E B84C B8E1 BBF9
8BB1 :AE0D
8C0E :8867 92F6 98BF 9A9A 9C88 9D39 ABE6 AC9B AD67 AECE B033 B0BF B4AE
     :B9C4 BECF BF96
8C1E :8BF5 BA13 BB3D
8C21 :B300 BAE0
8CB7 :9553 BE41
8CC1 :B21F
8DBB :8DA3
8E6A :8E3D
8E70 :8DDE 8E8D
8E8A :BA5A BA5F
8E98 :ADE9
8EE0 :8F09
8F1E :8F14 ABD5
8F69 :8FA3 90AC
8F92 :8FAE 9040
8F9A :8FE7
909F :8FD1 9008 9071
90DF :9168
9127 :91B4 9259
912F :9215
920B :9124
9218 :90DC
9222 :8F59 90EE 9195 AF39
9231 :91A6
9236 :91C1 9736
92DA :937F 9402 B459 B47C
92DD :8E40 90EB 9702 AB41 B047 B0C2 B7F5 B81A
92E3 :8E58 95AA 95B2 9691 AB33 ABD2 ACD1 AFAD B3BD BFBC
92EB :925C 926E 9282 92C8
92EE :8ED5 93FC B592 BBB7 BF37 BF62
92F0 :8824 8E2E 92DF 9688 974A 976F 99BF 99CE 9B3E 9B59 9B6C 9B7B 9B85
     :AB4D AD0C AF0F AFDD AFFF B05E B921 B9A2
92FA :9E3C A6BE A7B4 A7FE A8DA A907 A98D A998 AA91 ABB1 ABC2
92FD :9A59 9D29 9DE6 9DF2 9E36 B852 B870
93DA :938A 9396
93FD :93ED
9456 :8E3A 93DD 941E 9443
```

```
945B :B1E1
9469 :916F 965A 96BC 96DF
94ED :B171
94FC :8BD5 9174 9589
9531 :8BDF 9179 957F B176
9539 :B184
9559 :914B
955B :B1D5
9582 :85A5 8BE4 90E1 9327 B256 B7C4 B9E4 BA7F BB1F
95C9 :9582 B695
95D5 :8EEC
95DD :8BC9 AE22
9661 :96AC
969F :95AD
96DF :96A9 96CE
97BA :971C 975A 977A
97DF :8B2D 8F31 8F40 8F6E 8F80 9294 98E3 B5AC B5D8 B99A
97EB :903D B659
9807 :92EA
9813 :8BEE 8BFE BF34
982A :8604 8855 8C0B 8F2E 86A0 B9C7
9838 :BC22
9841 :8BD2 B7CE
9849 :BF5F
984C :8B56 B58F BBB4 BEDA
9852 :8F0E 9314 9382 9405 B461 B484 B4A3 BF9C
9857 :8AB6 8AC8 8AD0 8ADA 8B98 8EBD 8EC4 8F45 8F8F 928C 92A4 92B8 92C1
     :9361 9393 939F 9880 B5E3 B88B B8B6 B8CF B8E4 BB07 BD11 BFE4
9859 :858C 9854
9861 :984F
986D :850F 8B73 B16E B5FF B8FC BBEA
9877 :98EB B74B B964
987B :8F56
9880 :B837
9890 :859F 8B91
98E3 :B997
9905 :98A7 B8D6
991F :9097 9914 B662
9923 :90B8 B61D
9970 :B5EC B9AF BC2D
99A7 :A6BB
99BE :9E01 9E0A
9A5F :B78F
9A62 :9A4D
9A9D :9BCD 9BD6 9BE1 9BF1 8BFC
9A9E :9BAF
9B1D :8821 886D 8B53 8ED2 93EA 98C2 B58C B91E B99F BBB1 BED2
9B29 :8D39 8DEB 92DC 93F9 9849 AC1D ACE2 ACF0 AE56 AFCC AFEE B039 B28E
     :B4B1 B84F B86D
```

```
9B6B :9B3A 9B55
9B72 :9B29
9B9C :9B72 9B81
9C03 :B0FB
9C42 :9A53 9AA5 9AEA 9B9C
9C77 :9CDE
9C9B :9CB2
9CA1 :9CF7 9D0B
9D2C :9D1A
9D3C :9DCB
9DBB :9E07
9DBD :9E1A
9DCE :9C53 9CBA
9DD1 :9C42 9C8E 9CE4
9DD4 :9D36 9DC8
9E1D :99C8 9D52
9E20 :9C18 9D23 9DD1 9DEC
9ED1 :9EDC 9F36
9EDF :8DFB B0B9
9EF9 :B0B3
9F9C :9F1D
A040 :9FE8
A052 :A02A
A064 :A046 A060
A066 :9EC1 9ECE 9F1A 9FD1 9FD6 9FDF 9FF1 A017 A020 A037 A03E
A07B :AC60 AC6B AE2A
A140 :A0E1
A14B :A139
A178 :A208 A64F
A197 :A04F A0C8
A1DA :9F0F A075 A0F0 A405 A48E A50B A606 A6AD A6E7 A7B7 A801 A8DD A8F0
     :A90A AB7F AD77 AD7E
A1F4 :9ED7 A108
A208 :A25B A26A A284 A29C A5E0
A20B :A2BA
A21E :9A44 A1FF A23F A3F8 A6B2
A23F :A258 A25E
A242 :A202 A205 A261 A264 A267
A24D :9F31 9F6B A111
A2A4 :A666
A2BE :9300 9A41 9C98 9CAF 9CEE 9D02 9D0E 9D17 9D1D 9F0C AF24 B4E6
A2ED :A852
A303 :A0FF A2E3 A4B3 A5DC A602 A659 AA0E AF33
A34E :9A5F 9F74 A4D6 A500 A60B A6EC A9DF
A37D :9E6C AA11
A381 :9E59 9E88 A8EA A93C A9C3 AABE
A385 :8D3C 9F3D A6A5 A7BE A84B A89B A9B1 A9D9 AB1F
A387 :A835
A38D :A4D9 A6CA A7CF A9B7 AA20 B860 B882
```

```
A3B2  :AA35 BA36
A3B5  :9A4A 9E4D 9E64 9E72 A6B5 A8B2 A901 AA26 AAC9 B2EA
A3E4  :92F3 98C9 9E93 AF36 B4C3
A3E7  :AC95
A3FE  :A3E4 A491 A9EE AC82
A453  :A402 A814 A93F
A46C  :A4B0 A4C7 A4CD AA0B
A486  :9E45 AAB8
A4B0  :A4AB
A4B6  :A4CA
A4C7  :AC92
A4D0  :9D08 A9BD
A4D6  :A6D5
A4E8  :A498
A4FD  :9CF4 A4D0
A500  :9C9E A7DD A848 A863 A8CC A92A A930 AA1D AA32 B774
A505  :A830 A94A A9E8
A50B  :AF7B A505
A606  :A656 AF30
A656  :9D2F 9E81 A842 A845 A85D A9B4 A9C6 AA17 AA2C AAD4 AB2C ABBC
A659  :A7A6 AF81
A65C  :A118 A508
A66C  :A450
A67C  :A669
A686  :9F5C 9FA0 A2EE A3FB A5B4 A610 A699 AAA6
A699  :9E8B 9F20 A6A8 A9BA AB22 B863
A6A5  :A921 AB1A
A6AD  :9DF8 A7D6 A8B8 A8F8
A6E7  :A6E1 A9EB
A7B7  :A9C0
A7E9  :A6C7 A6D2 A6DE A83F
A7ED  :9E7E A7CC AA23
A7F1  :A8F5 AAD1
A7F5  :9F71 A3B2 A860 A8B5 AA1A AA2F
A7FE  :A8A8
A801  :9E75
A897  :A83C A951 A9CD AADE
A8DA  :A8D4
A8EA  :A8E4
A8FE  :ABCB
A90A  :A8FB
A916  :A8E7
A91B  :A913
A927  :A8D7
A936  :A924
A99E  :A6CF A6DB A995
A9AA  :A9A4
A9B1  :A8ED
A9C3  :A9AE
```

```
A9D3  :A6C1 A990 A99B
AA48  :A927 AA14
AA4C  :A92D AA29
AA55  :A8FE A9DC
AA94  :9E7B
AAD1  :9E78 A954 A9D0
AADA  :AABB
AB12  :9E52 9E69 AACE
AB25  :AB2F
AC23  :AC75
AC34  :BAD3
AC73  :AC63
ACC4  :AB9D
AD71  :99C5 99D8 9D70 9D80
AD7E  :A4D3 A4FD A933 A9A7
AD8F  :AC70
AD93  :9DC3 A2C8
ADAD  :BABA BB38
ADE1  :ADC6
ADEC  :92E2 92F9 9E20 AB88 ABE9 AC2F AC78 AC9E AD6A AED1 B0A3 BF83
AE02  :AD8C
AE3A  :85AF
AE43  :8F1B AEC7
AE56  :8E2B 9747 976C AB4A AD09 AF0C AFDA AFFC B05B B0CB
AED8  :8F6B 8F76 9182 9338 96F8 AB6A AB73 AB7C ABA2 ACAA AD4F AEF9 AFBC
      :B5A9 B810 BF75 BF93
AEEA  :98A4 AB3E ACB5 AE40 AEC4 AF00 AF07 AFA3 AFAA B351
AF56  :9DBD B326
AF69  :AF2A
AF87  :AF51 AF69
AFAD  :ACAD B026
AFC2  :B3C2
B112  :B1E6
B197  :9311
B1F4  :B187
B202  :B30A
B2F3  :B2ED
B303  :B2F6
B30D  :932C B276
B32C  :9685 AE27 B318
B354  :B766
B3C5  :B402
B4B1  :B7D1 BB2C
B4B4  :85B4 8C05 911E 933D B139 BAD6
B4B7  :B2F3
B4E9  :B77F
B50E  :8571 B688 BFF0
B545  :8526 B562
B550  :B54A
```

```
B558 :855C 855F 8E17 8EA4 9911 9919 9964 B53C B583 B64B BA9F BC02
B562 :852B 854E
B565 :8544 8559 8DB5 8E08 8E5F 991C 995A B580
B577 :B626 B62D B634
B637 :B665
B695 :B763
B837 :B885
B88B :B97A
B8D2 :98EE B967
B8DD :BBD3
B99A :B888 B8CC B95C BAFF
B9AF :98E8
B9DA :BA16 BA3C
BA5A :BAD9 BAE3
BB15 :BB4D
BB40 :BB2F
BB50 :BB26 BB32
BBFC :BAA2
BC02 :8B08 90BD
BC25 :853F 857B 8D7D 8E53 8E67 909A B56E B5FC BFE7
BC28 :8EC7 93D6 B55F
BC2D :8F53 BC8F
BC5D :BC79
BC81 :BC73 BC76
BC8D :8B32 90C6
BD14 :BF2D
BD20 :8AF3 90C9 928F BCC9 BD14
BD2F :927D
BD3A :8B1A B41B BD2C
BD51 :9A3E 9A50 9C8B 9CAC 9CE1 9CFF 9D14 9D20 9DE9 9E39 AF27
BD7E :9A47 9A5C 9C9B 9CF1 9D05 9D2C 9DF5 9E4A 9E6F AF2D B2E7
BD90 :B291 B31C
BD94 :85AC 8BEB 8BFB 8ED8 8F36 8F71 90B5 90E8 9185 9333 93FF 96FF 9744
     :9AA2 9B6F 9B7E 9DCE 9E1D AB44 B0C5 B298 B329 B5B0 B5C8 B7CB B9F1
     :BAD0 BB29 BB35
BDB2 :9AE7 9C15 ABF7 ACED AD06 AFD7 AFF9 B042
BDCB :9C37 AD0F AFE0 B002 B061 B2FD
BDDC :8CEB 9B16 AC20 AD39 AD52
BDE1 :8D28
BDEA :8C1E 8F11 8F50 90B2 90C0 9A3B 9CA9 9CFC 9D11 9D78 AB56 B0D0 B2CA
     :B2F0 B5C5 B5E9
BDFF :9B4E 9B95
BE0B :9411 B21C B4B4
BE0D :8FA8 9232 970D 9755 99DD
BE2E :B19E BD56 BD99 BDB7
BE44 :885F 9D75 AF41 B2E0 B315
BE55 :BB04
BE56 :BD02
BE62 :BF24 BF2A
```

```
BE6F    :8AC3 8ACB 8FAB B5E6 BCC6 BEF3
BE92    :BC7C BCB2
BE93    :BE8B
BEB2    :BED7
BEBA    :8CA2 BF88
BED2    :BEC2 BEDD
BEDD    :BE62 BF0A
BEE7    :93A2
BFA9    :8D2D B9D1 BF30 BF58 BF99
BFB5    :ACB8 BF4C BF6F
BFCF    :9080 BE9E
OSFIND  :BF90 BFA3
OSBPUT  :8D43 8D4F 8D5C 8D66 8D6F BF69
OSBGET  :B9F6 BA02 BA0A BA21 BA2D BF72
OSARGS  :BF40 BF52
OSFILE  :BE6C FF1E
OSRDCH  :AFB9 AFBF
OSASCI  :BFD9
OSNEWL  :BC25
OSWRCH  :8E33 8E37 8ECC 9387 93DA 940A 940E 9416 941B 9423 942C B55C
OSWORD  :92D3 AB63 AEBA B49A BC1D
OSBYTE  :8025 802E 93BD AB3A AB6F AB78 ACBE AFB6 B423 B428 BEE9
OSCLI   :8B7A BEC9
```

Basic 2 - lookup table reference origins

L/U table address	:calling location :address
82CB	:8B23
833C	:8B28
843B	:85DC
8474	:85E1
84AD	:8607
993D	:9902
99B8	:9908

ROM type	:0110 0000 - no service entry, language :entry, second processor relocation :address provided
Binary version number	:03
Copyright offset	:0E
Title string	:BASIC (C) 1983 Acorn
Assembly address	:8000-BFFF
Relocation address	:B800-F7FF
Public workspace	:none
Private workspace	:none

HiBasic -Memory Map

B800-B870:paged ROM protocol and initialisation
B871-BC57:BASIC keyword lookup tables
BC58-C088:assembler
C09E-C2D3:line editing routines
C2D4-C5B7:program control commands, statement parsing routines
C5B8-C6EC:display control routines
C6ED-C74C:machine code CALLs
C74C-C8F6:program line numbering
C8F7CA773:array DIMensioning routines
CA78-CAF4:setting BASIC parameters
CB1F-CBA8:PROCedure and LOCAL variable processing
CBA9-CC75:BASIC VDU command handler
CC76-D066:variable processing and setup routines
D067-D418:statement parsing and IMMEDIATE command handler
D419-D69D:arithmetic routines
D6A6-D855:PRINT formatting
D856-DED3:numeric input and floating point manipulation
DED4-E348:higher order mathematical functions
E349-E705:miscellaneous parameters determination
E706-E7B0:RaNDom number processing
E7B1-E901:string handling
E902-EB10:FN/PROC routines

EB11-EC05:variable manipulation
EBC9-EC4F:error and BRK handler
EC50-ECA3:BASIC sound command processing
ECA4-EE6A:LISTing
EE6B-F08A:FOR/NEXT loop handling
F08B-F1D1:GOSUB, GOTO, ON routines
F1D2-F3A8:INPUT and DATA retrieval
F3A9-F3FE:REPEAT/UNTIL loop handling
F3FF-F513:BASIC program line input
F514-F732:program housekeeping and control
F733-F7F0:data file access commands

HiBasic - Zero Page usage

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
10	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
20	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
30	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
40	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	.
50	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
60	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
70	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
80	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
90	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
A0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
B0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
C0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
D0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
E0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
F0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	*
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

B800-B870 - paged ROM protocol and initialisation

```
B800        :language entry point
B809        :ROM title string and copyright message
B81F        :relocation address
B823        :ROM initialisation
```

B871-BC57 - BASIC keyword lookup tables

```
B871-BB73 :BASIC keyword, token and processing parm lookup table
BB74-BBE5 :BASIC routine entry point address lo lookup table
BBE6-BC57 :BASIC routine entry point address hi lookup table
```

BC58-C088 - assembler

```
BC58-BC91 :assembler packed mnemonic lookup table 1
BC92-BCBB :assembler packed mnemonic lookup table 2
BCCC-BD03 :assembler base opcode lookup table
BD04        :reset OPT and quit assembler
BD0B        :assembler entry point
BD0F        :process next source code character
BDA0        :set index to source code line length
BDDB        :check source code delimiting character
BE02        :pack source code mnemonic
BE12        :derive object code lookup table index from packed mnemonic
BE28        :check for Boolean mnemonic
BE41        :look up opcode and transfer to buffer
BE4C        :move object code from buffer to end of asembled code
BE94        :move source code address bytes to object code
BEB2        :check if error reporting required
BEB8        :"Out of range" error routine - error code 01
BECA        :assemble two bytes of object code
BEE7        :check source code and assemble
BEEE        :"Byte" error routine - error code 02
BEF9        :process STA (Indirect,X)
BF1D        :check for indirect indexed instruction and process it
BF2F        :"Index" error routine - error code 03
BF57        :add four to first object code byte, move code from buffer
BF5A        :check third byte of object code, move code from buffer
BFBC        :assemble 3 bytes from object code buffer
C00F        :go to "Index" error routine
C037        :save OPT value, zero index
C043        :evaluate expression, convert floating point to integer
C049        :set primary = secondary text buffer index
C04E        :add sixteen to first byte in object code buffer
C051        :add eight to first byte in object code buffer
C054        :add four to first byte in object code buffer
C089        :go to "Type mismatch" error routine
```

C09E-C2D3 - line editing routines

```
C09E    :insert line number token, make space for line number instruction
C0B9    :convert decimal line number to HEX, pack and insert
C0F9    :create space in program and insert line number instruction
C10B    :move text
C112    :pack line number and insert
C145    :check for operational (carry clear) or alphanumeric character (carry
        :set)
C155    :check for ASCII decimal character
C15C    :seek decimal point or ASCII decimal numeral
C161    :get character from program and increment pointers
C163    :increment program pointers
C16A    :increment program pointers, get next character
C170    :set secondary text buffer pointers, strip spaces, check for HEX
C174    :set secondary text pointer, strip spaces and check for HEX
C176    :strip spaces and check for HEX
C180    :check for HEX
C19B    :check for " : , . *
C1AB    :check for : , . or *
C1C2    :check for .
C1D4    :flush to numeral or decimal point
C1E1    :reset secondary program pointers
C1F1    :flush to next operational character
C1FE    :check for alphanumeric character
C20B    :test for presence of keyword
C217    :compare character against keyword lookup table character
C21D    :seek token in keyword lookup table
C237    :search keyword table for token
C244    :move pointers on to next keyword
C256    :process token according to value of token processing parameter
C291    :check if next character is operational or alphanumeric
C2AB    :get next non-space character from secondary text buffer
C2B6    :get next non-space character from primary text buffer
C2C1    :confirm next character is ","
C2C8    :"Missing ," error routine - error code 05
```

C2D4-C5B7 - program control commands, statement parsing routines

```
C2D4    :OLD command - token CB
C2E6    :END command - token E0
C2EE    :STOP command - token FA
C2F1    :"STOP" error routine - error code 00
C2F5    :NEW command - token CA
C2F8    :language entry routines
C30E    :reset BASIC pointers and initialise RAM
C311    :initialise input buffers, BRKV, PRINT prompt, execute instruction
        :(IMMEDIATE instruction handler)
C323    :execute next instruction
```

```
C35C    :go to IMMEDIATE command handler
C35F    :go to assembler entry point
C362    :check for FN
C374    :"No FN" error routine - error code 07
C37B    :check for = * or [
C38E    :* command handler
C398    :process next instruction from primary text buffer
C398    :DATA command - token DC
C398    :DEF command - token DD
C398    :REM command - token F4
C3A2    :check for ELSE and process next instruction
C3B1    :decrement primary text buffer index, test line termination, process
        :next statement
C3B3    :test line termination and process next instruction
C3B6    :process next instruction
C3BE    :check for BASIC command token
C3CC    :get BASIC command entry point from lookup table using token as index
C3DA    :process next statement from primary text buffer
C3FF    :LET command - token E9
C426    :go to "Syntax error" routine
C429    :"Type mismatch" error routine - error code 06
C439    :retrieve parameters from BASIC stack and allocate string variable
C43C    :allocate string variable
C49F    :transfer variable name length to variables table, next variable
        :pointer to 2C,2D
C4B2    :transfer variable name from Page 6 to variables table
C4BD    :save string from Page 6 buffer
C4D2    :"No room" error routine - error code 06
C4DC    :pull string off BASIC stack into buffer
C509    :transfer string from BASIC stack to variables table
C51E    :transfer variable from BASIC stack to variables table
C549    :PRINT# routine
C569    :pass integer variable to file
C575    :pass floating point variable to file
C582    :pass string variable to file
C595    :recover channel number, save primary text buffer index, process next
        :statement
```

C5B8-C6EC - display control routines

```
C5B8    :PRINT command - token F1
C5C4    :TAB according to formatting variable (@%) setting
C5D9    :set PRINT field, HEX flag, check line termination, PRINT buffer
C5F0    :PRINT contents of buffer
C62C    :PRINT string from Page 6 buffer
C63F    :go to "Missing ," error routine
C642    :do Y TABulation
C65E    :TAB( command - token 8A
C676    :SPC( command - token 89
```

```
C676    :do X TABulation
C682    :start new line, zero COUNT, reset primary text buffer index
C685    :copy secondary to primary text buffer index
C68B    check for formatting command
C6A5    :seek formatting command, parenthetical string, PRINT it
C6B3    :"Missing "" error routine - error code 09
C6BF    :output character, check for "
C6D8    :CLG command - token DA
C6DF    :CLS command - token DB
```

C6ED-C74C - machine code CALLs

```
C6ED    :CALL command - token D6
C6FB    :process next CALL parameter
C736    :go to "No such variable" routine
C739    :set carry flag, A-,X-,Y-registers, execute machine code routine
C749    :go to "Syntax error" routine
```

C74C-C8F6 - program line numbering

```
C74C    :DELETE command - token C7
C784    :push start line number, STEP size on to BASIC stack
C7AD    :set RENUMBER pointers to TOP and program pointers to PAGE + 1
C7B5    :set program pointers to PAGE+1
C7BE    :RENUMBER command - token CC
C7F1    :"RENUMBER space" error routine - error code 00
C7FA    :"Silly" error routine - error code 00
C85B    :check number of next line
C889    :move pointers to next line, increment RENUMBER space pointers
C8A8    :display line number and check next line
C8BA    :move program pointers to start of next line
C8C7    :AUTO command - token C6
```

C8F7-CA77 - array dimensioning routines

```
C8F7    :go to "DIM space" error routine
C8FA    :reserve memory space
C942    :"Bad DIM" error routine - error code 0A
C94A    :DIM command - token DE
C986    :DIMension array
C9D2    :DIMension array element
CA26    :check for further array elements to DIMension
CA30    :go to DIM command routine
CA33    :"DIM space" error routine - error code 0B
CA3D    :increment integer buffer
CA4C    :multiply integer in 3F-42 by integer buffer
CA51    :multiply integer from workspace, by integer buffer
CA75    :go to "Bad DIM" error routine
```

CA78-CAF4 - setting BASIC parameters

```
CA78    :HIMEM command - token D3
CA8A    :LOMEM command - token D2
CA9E    :PAGE command - token D0
CAA8    :CLEAR command - token D8
CAB0    :TRACE command - token FC
CAD2    :set TRACE flag on, reset maximum TRACE line number
CADB    :reset TRACE flag, process next statement
CAE4    :TIME command - token D1
CAF5    :confirm next character is ",", evaluate expression, result in
        :integer buffer
CAF8    :evaluate expression, get result in integer buffer
CAFE    :get variable type from program, confirm numeric, convert floating
        :point to integer
CB06    :set secondary = primary text buffer, confirm numeric, convert
        :floating point to integer
CB09    :get variable type, confirm numeric, convert floating point to
        :integer
CB0B    :confirm variable numeric, convert floating point to integer
CB12    :go to "Type mismatch" error routine
CB15    :evaluate variable, confirm numeric, convert integer to floating point
CB18    :confirm numeric and convert integer to floating point
```

CB1F-CBA8 - PROCedure and LOCAL variable processing

```
CB1F    :PROC command - token F2
CB36    :clear integer LOCAL variable
CB3E    :LOCAL command - token EA
CB5C    :check for another LOCAL variable
CB6E    :check line termination, process next statement
CB71    :ENDPROC command - token E1
CB80    :"No PROC" error routine - error code -0D
CB86    :"Not LOCAL" error routine - error code 0C
CB8D    :"Bad MODE" error routine - error code 19
```

CBA9-CC75 - BASIC VDU command handlers

```
CB95    :GCOL command - token E6
CBA9    :COLOR command - token FB
CBA9    :COLOUR command - token FB
CBB5    :MODE command - token EB
CBF5    :output character from stack, then one from integer buffer, process
        :next statement
CBFF    :MOVE command - token EC
CC03    :DRAW command - token DF
CC0C    :PLOT command - token F0
CC18    :process parameters and execute VDU25 command
CC45    :send second character from integer buffer to current output stream
```

```
CC4A    :VDU command - token EF
CC6E    :decrement primary text buffer index, test line termination, process
        :next statement
CC71    :send first character from integer buffer to current output stream
```

CC76-D066 - variable processing and setup routines

```
CC76    :search for FN/PROC
CC84    :search for variable pointed by 37,38
CC94    :check variable name for identity
CCB5    :search variables table for name
CCCE    :check identity of variable name, letter by letter
CCFC    :increment pointers by value of index
CD08    :set up FN/PROC references
CD17    :save new variable name to variables table
CD31    :add address to variables chain
CD4C    :clear space for variable
CD54    :move variables table pointer, check if space for another variable
CD5C    :check if space for another variable
CD6E    :go to "No room" error routine
CD74    :count alphanumeric characters in variable name
CD76    :check next character of variable name
CD9A    :clear space and set up variable reference
CD9D    :set up variable references (address 2A,2B type 2C)
CDB0    :check for indirection operator
CDC0    :"!" indirection routine
CDC2    :"?" indirection routine
CDCB    :"$" indirection routine
CDDA    :"$ range" error routine - error code 08
CDE4    :set secondary text buffer index = primary, decrement primary index
CDF0    :strip spaces and set up variable reference
CDF8    :set up variable reference
CE1A    :set up array reference
CE36    :check next character from variable name
CE5C    :determine variable type and set up references
CE67    :set up integer reference
CE7C    :check for indirection operator
CE96    :process "?" operator
CE9A    :process "!" operator
CEBA    :recover variable type, set A=FF, carry clear
CEC1    :set up array references
CECA    :set up string references
CEE4    :set up string array references
CEF2    :"Array" error routine - error code 0E
CEFA    :set up array references
CF12    :calculate space required for array, adjust variables table pointers
CFC8    :use array dimension to move pointers on
CFD5    :check array subscript size
CFEC    :"Subscript" error routine - error code 0F
```

```
CFF8        :seek line number token, if found, unpack number to 2A,2B, set carry
CFFA        :if line number token present, unpack number to 2A,2B, set carry
D006        :unpack line number to 2A,2B
D022        :equate primary and secondary text buffers, evaluate RHS of
            :expression
D02E        :check for = , then evaluate RHS of expression
D03C        :"Mistake" error routine - error code 04
D045        :"Syntax error" routine - error code 10 (also command token CE entry
            :point)
D053        :"Escape" routine - error code 11
D05C        :confirm next character is =
D064        :evaluate expression, transfer X- to A- register, check line
            :termination
```

D067-D418 - statement parsing and IMMEDIATE mode handler

```
D067        :transfer X- to A-register, check line termination
D06D        :set secondary text buffer index, strip spaces, check line
            :termination
D072        :set primary text buffer index, strip spaces, check line termination
D074        :strip spaces and check line termination
D07C        :check line for correct termination, increment text buffer index
D088        :increment primary text buffer index, test ESCAPE flag
D092        :set buffer index = 1, test ESCAPE flag
D096        :test ESCAPE flag
D09B        :move to next statement with TRACE if required
D0AB        :increment index and check if TRACE required
D0C7        :use index to advance pointers, reset index to 1
D0D7        :go to IMMEDIATE command handler
D0DA        :go to "Type mismatch" error routine
D0DD        :IF command - token E7
D0FC        :increment primary text buffer index and ....
D0FE        :process THEN command - token 8C
D10C        :search for ELSE and process it if found
D120        :display TRACE marker if line number below set limit
D13A        :output decimal equivalent of number in 2A,2B
D13E        :output decimal equivalent of number in 2A,2B with leading spaces
D185-D187   :HEX to decimal conversion lookup table 1
D18A        :search BASIC program for line number, carry set, Y=2 if not found
D1C1        :"Division by zero" error routine - error code 12
D1D3-D1D7   :HEX to decimal conversion lookup table 2
D1D8        :confirm numeric variable, convert to integer, store in integer
            :buffer, 39-3C
D202        :perform division process
D279        :unpack variable to floating point buffer #1, compare with floating
            :point buffer #2
D26A        :save floating point buffer #1 to BASIC stack, unpack variable to floating
            :point buffer #2, compare with floating point buffer #1
D2C9        :unpack variable to floating point buffer #2, compare with floating
```

```
                :point buffer #1
D27C            :check floating point buffer #2 and floating point buffer #1 for
                :identity
D2AD            :test sign of floating point buffer #2
D2B4            :to to "Type mismatch" error routine
D2B7            :transfer variable type to A-register, check identity of variables
D2B8            :check identity of variables
D2CD            :compare integer variables for identity
D301            :compare strings for identity
D333            :equate primary and secondary text buffers, evaluate next statement
D33F            :evaluate next statement (MAIN ENTRY POINT)
D350            :OR command - token 84
D36B            :EOR command - token 82
D381            :process < > = OR or AND, if present
D388            :check for < > = and AND
D390            :AND command - token 80
D3B2            :process < = > + - operators, if present
D3BE            :process < = > operators
D3D6            :process < operator
D3EA            :process > = operators
D3F5            :process < > operators
D410            :process > = operators
```

D419-D69D - arithmetic routines

```
D419            :"String too long" error routine - error code 13
D42B            :concatinate strings
D458            :check for + or - operator
D464            :process + operation
D471            :integer addition
D48D            :reset BASIC stack pointer, check for another operator
D49E            :go to "Type mismatch" error routine
D4A1            :check for multiplication or division operator, do floating point
                :addition
D4B1            :perform floating point addition
D4B7            :recover variable type, check for + or -
D4BD            :pull integer from BASIC stack and perform floating point addition
D4CB            :perform - operation
D4D8            :perform integer subtraction
D4F7            :subtract floating point variable
D510            :convert integer to floating point and perform floating point subtraction
D524            :convert integer to floating point
D533            :convert integer to floating point and perform floating point
                :multiplication
D536            :perform floating point multiplication
D542            :multiply floating point buffer #1 by current variable, check for
                :further multiplication or division operator
D54F            :go to "Type mismatch" error routine
D552            :perform multiplication
```

D572 :integer multiplication
D5D1 :transfer quotient to integer buffer
D5D3 :transfer indexed part of quotient to integer buffer
D5E1 :go to multiplication routine
D5E4 :push integer on BASIC stack, evaluate expression, check for :multiplication or division operator
D5E7 :evaluate expression, check for multiplication or division operator
D5EA :check for multiplication or division operator
D5FB :floating point division routine
D617 :DIV routine
D620 :MOD routine
D633 :push integer on to BASIC stack, evaluate expression, get character :in X-register
D636 :evaluate expression and get non-space character in X-register
D64B :process exponentiation
D685 :use natural logs to evaluate power
D69E :large number exponentiation

D6A6-D855 - PRINT formatting

D6A6 :assemble HEX output for PRINT
D6C6 :convert HEX to ASCII
D6DE :assemble floating point output string
D6E7 :check if exponent is 81 or more
D6F5 :assemble formatted PRINT output string
D70F :assemble PRINT output string
D733 :go to non-zero formatting parameter PRINT routine
D73B :adjust position of decimal point for PRINTing
D747 :divide by ten, adjusting power in decimal index store
D7B2 :PRINT with non-zero format parameter
D7B6 :PRINT fixed number of decimal places
D7F1 :insert required number of zeros after decimal point
D7FE :save next digit to buffer

D82B :PRINT in exponential format

D856-DED3 - numeric input and floating point manipulation

D856 :transfer ASCII equivalent of decimal to string buffer
D868 :transfer ASCII equivalent of lo nibble to string buffer
D87A :add ASCII offset, store character in Page 6 buffer, increment :pointers
D87C :store character in Page 6 buffer, increment pointers
D888 :save rounding byte, get sign of floating point buffer #1, set A=FF
D891 :initialise floating point buffer #1 mantissa and workspace
D8A1 :input ASCII character and check validity
D8AF :input next ASCII character, if numeral, enter in buffer
D8B6 :process decimal point
D8BE :enter ASCII numeral into floating point buffer #1
D8F7 :process exponentiation

```
D927    :process exponentiation with negative index
D935    :exponentiation with index = 0
D94F    :process -
D956    :process + or - or load next decimal digit, if present
D961    :process +
D96E    :increment index, add numeral to ten times temporary store (4A)
D986    :recover number from temporary store, clear carry
D98E    :add corresponding byte of floating point buffer #2 to floating point
        :buffer #1
D9AD    :multiply mantissa by ten, preserving A-register
D9BB    :multiply floating point buffer #1 mantissa by 4
D9F0    :get sign of floating point buffer #1
DA03    :get sign, exponent and guard bytes of floating point buffer #1
DA0A    :multiply floating point buffer #1 by ten
DA11    :multiply floating point buffer #1 by 1.25 decimal
DA1E    :add corresponding byte of floating point buffer #2 mantissa to
        :floating point buffer #1, increment exponent if necessary
DA21    :divide floating point buffer #1 mantissa by 2 and increment exponent
DA34    :copy floating point buffer #1 to floating point buffer #2
DA36    :set floating point buffer #2 sign byte, copy remaining bytes from
        :floating point buffer #1
DA55    :copy floating point buffer #1 to floating point buffer #2 and divide
        :floating point buffer #2 mantissa by 2
DA58    :divide floating point buffer #2 mantissa by 2
DA63    :divide floating point buffer #1 by ten
DABA    :add A-register to floating point buffer #1
DAD4    :convert integer to floating point
DAFC    :set floating point buffer #1 sign, exponent and guard bytes
DB03    :adjust sign and normalise
DB19    :normalise floating point buffer #1
DB4C    :shift floating point buffer #1 mantissa until all bytes are
        :significant
DB64    :unpack variable to floating point buffer #2
DB93    :pack floating point buffer #1 to memory at 0471
DB97    :pack floating point buffer #1 to memory at 0476
DB9B    :pack floating point buffer #1 to memory at 046C
DB9D    :pack floating point buffer #1 to current variable location (4A,4B)
DBA3    :pack floating point buffer #1 to memory
DBCB    :unpack variable at 046C to floating point buffer #1
DBFA    :position binary point, convert floating point to integer
DBFD    :transfer floating point buffer #1 mantissa to integer buffer
DC14    :shift binary point until exponent = A0
DC22    :halve floating point buffer #1 mantissa, compensating exponent until
        :it is = A0
DC52    :divide floating point buffers #1 and #2 mantissas by 2, compensating
        :exponent
DC66    :go to "Too big" error routine
DC69    :clear floating point buffer #2
DC82    :complement floating point buffer #1 mantissa
```

```
DC9C    :check floating point buffer #1 exponent sign, if positive, zero 4A,
        :return with sign
DCC6    :change sign of floating point buffer #1 mantissa, normalise floating
        :point buffer #1
DCCC    :increment mantissa of floating point buffer #1
DCDD    :increment mantissa of floating point buffer #1 if positive,
        :decrement if negative
DCE6    :subtract variable from floating point buffer #1
DCEC    :interchange variable in memory and floating point buffer #1
DCFE    :transfer mantissa from floating point buffer #2 to floating point
        :buffer #1
DD13    :change sign of floating point buffer #1, unpack variable to floating
        :point buffer #2, algebraicly sum them
DD16    :unpack variable to floating point buffer #2, algebraicly add to
        :to floating point buffer #1
DD1B    :align binary points, add floating point variables, rounding if
        :necessary
DD21    :align binary points and algebraicly sum variables
DDA6    :take algebraic sum of mantissas
DDCD    :take smaller mantissa from larger
DDF9    :subtract floating point buffer #2 mantissa from floating point
        :buffer #1, result in floating point buffer #1
DE1C    :if floating point buffer #1 positive, unpack variable to floating
        :point buffer #2 and multiply
DE29    :multiply floating point buffer #1 by floating point buffer #2, result
        :floating point buffer #1
DE6C    :multiply floating point buffer #1 by current variable
DE6F    :normalise floating point buffer #1 and round it if necessary
DE72    :round floating point buffer #1 if necessary
DE82    :"Too big" error routine - error code 14
DE8C    :round floating point buffer #1 up ....
DE92    :zero rounding byte, test for overflow, clear floating point
        :buffer #1
DE9C    :clear floating point buffer #1
DEAF    :set floating point buffer #1 = -1
DEBB    :invert floating point buffer #1 and change its sign
DEC3    :divide variable pointed by 4A,4B by floating point buffer #1
DED1    :go to "Division by zero" error routine
```

DED4-E348 - higher order mathematical functions

```
DED4    :TAN command - token B7
DEFD    :divide floating point buffer #1 by current variable
DF07    :divide floating point buffer #1 by floating point buffer #2
DF20    :calculate mantissa
DFBF    :"-ve root" error routine - error code 15
DFCA    :SQR command - token B6
DFCD    :calculate square root of floating point numeral
DFFF    :set current variable pointer to 047B
```

```
E003        :set current variable pointer to 0471
E007        :set current variable pointer to 0476
E00B        :set current variable pointer to 046C
E014        :LN command - token AA
E017        :check sign of variable and calculate LN
E01E        :"Log range" error routine
E02A        :calculate LN
E07F        :constant (0.693 147 181) used in LN calculation
E084-E0A7   :LN power series (7 iterations)
E0A8        :carry out power series calculation
E0E5        :ACS command - token 95
E0EB        :ASN command - token 98
E0FB        :calculate ASN for positive values
E10F        :unpack PI/2 to floating point buffer #1
E118        :ATN command - token 99
E11B        :calculate ATN of angle
E127        :reset floating point buffer #2 sign byte negative
E12C        :calculate ATN, inverting angle if > PI/4
E138        :subtract angle from PI/2
E147        :calculate ATN
E16B-E19D   :ATN power series (8 iterations)
E19E        :COS command - token 9B
E1A9        :SIN command - token B5
E1AF        :carry out SIN/COS calculation, adjusting angle and applying
            :correction, if necessary
E1BB        :carry out SIN/COS calculation, with corrections, if necessary
E1C2        :apply correction to result
E1D4        :carry out SIN/COS calculation
E1E4        :adjust value of angle for SIN/COS calculation
E249        :"Accuracy lost" error routine - error code 17
E259        :set current variable pointer to E26A (coarse -PI/2)
E25D        :set current variable pointer to E26F (correction factor)
E266        :set current variable pointer to E274 (PI/2)
E26A        :constant (coarse -PI/2 = -1.570 800 78)
E26F        :constant (coarse -PI/2 correction factor = 4.454 455 11 E-6)
E274        :constant (PI/2 = 1.570 796 33)
E279        :radian/degree conversion factor (1/57.295 779 6)
E27E        :degree/radian conversion factor (57.297 779 5)
E283-E1A6   :SIN/COS power series (6 iterations)
E2AA        :calculate EXP of floating point variable
E2A7        :EXP command - token A1
E2C2        :"Exp range" error routine - error code 18
E2E7        :multiply floating point buffer #1 by variable at 0476
E2F0        :carry out power series calculation with EXP power series
E2FA        :'e' (2.718 281 83)
E2FF-E327   :EXP power series (8 iterations)
E328        :raise floating point variable to power specified in 4A
E33B        :multiply by current variable, adjusting index
```

E349-E705 - miscellaneous parameters determination

```
E349    :ADVAL command - token 96
E357    :NOT command - token AC
E368    :POS command - token B1
E36E    :VPOS command - token BC
E377    :LOG command - token AB
E37E    :RAD command - token B2
E38F    :DEG command - token 9D
E396    :PI command - token AF
E39D    :USR command - token BA
E3B1    :EVAL command - token A0
E3EB    :restore text parameters, set A = variable type
E3F7    :go to "Type mismatch" error routine
E3FA    :VAL command - token BB
E3FF    :calculate VAL of string in buffer
E43C    :restore secondary text buffer, set A = variable type
E441    :INT command - token A8
E464    :ASC command - token 97
E473    :INKEY command - token A6
E47D    :go to "Type mismatch" error routine
E480    :EOF command - token C5
E48C    :TRUE command - token B9
E499    :FALSE command - token A3
E4A6    :SGN command - token B4
E4C0    :POINT( command - token B0
E4E9    :INSTR( command - token A7
E571    :ABS command - token 94
E578    :make integer buffer positive
E57E    :take modulus of floating point buffer #1
E585    :change sign of floating point buffer #1, set A = FF
E596    :confirm variable numeric, change its sign
E59A    :change sign of each location of integer buffer, set A=40
E5B4    :transfer input line to Page 6 buffer, set A=0
E5D5    :transfer string, if legal, to Page 6 buffer
E5EF    :go to "Missing "" error routine
E5F2    :parse expression
E608    :process next character
E636    :check OPT value and transfer asssembly address to integer buffer
E640    :transfer assembly next address to integer buffer
E649    :"No such variable" error routine - error code 1A
E65B    :"Missing )" error routine - error code 1B
E666    :"Bad HEX" error routine - error code 1C
E670    :evaluate expression and check for ")"
E67B    :convert ASCII character to HEX integer
E6B0    :TO command - token B8
E6B0    :TOP command - token B8+50
E6B7    :TOP routine
E6C5    :go to "Type mismatch" error routine
```

```
E6BF    :PAGE command - token 90
E6C8    :LEN command - token A9
E6CF    :zero index (Y), ....
E6D1    :transfer A-register to LSB of integer buffer, clearing higher bytes
E6DE    :COUNT command - token 9C
E6E2    :LOMEM command - token 92
E6E8    :HIMEM command - token 93
E6EE    :ERL command - token 9E
E6F4    :ERR command - token 9F
E6FA    :TIME command - token 91
```

E706-E7B0 - RaNDom number processing

```
E720    :RND(N) routine
E73B    :RND(-N) routine
E745    :RND command - token B3
E752    :copy 00,X-00,X+3 to integer buffer, set A=40
E765    :RND(1) routine
E768    :RND(0) routine
E783    :shuffle random number store
```

E7B1-E901 - string handling routines

```
E7B1    :get character within time limit
E7BD    :GET command - token A5
E7C3    :GET$ command - token BE
E7C6    :save character to Page 6 buffer
E7D0    :LEFT$( command - token C0
E7F2    :RIGHT$( command - token C2
E82A    :INKEY$ command - token BF
E837    :go to "Type mismatch" error routine
E83A    :go to "Missing ," error routine
E83D    :MID$( command - token C1
E887    :reposition string within Page 6 buffer
E898    :STR$ command - token C3
E8C3    :go to "Type mismatch" error routine
E8C6    :STRING$( command - token C4
```

E902-EB10 - FN/PROC routines

```
E902    :restore primary text pointers and execute ....
E908    :"No such FN/PROC" error routine - error code 1D
E916    :search for DEF in program
E916    :search for DEF in BASIC program
E91E    :flush program to DEF token
E940    :process DEF token
E98E    :"Bad call" error routine - error code 1E
E999    :FN command - token A4
E99B    :process FN/PROC
```

```
E9A7    :save 6502 stack to BASIC stack
E9CC    :increment index and set program pointers
E9CE    :set program pointers
E9ED    :set primary text pointers to next variable
E9F8    :check for LOCAL variables
EA06    :set up LOCAL variable
EA1E    :discard LOCAL variables
EA2A    :recover primary text pointers from BASIC stack
EA34    :restore 6502 stack from BASIC stack
EA51    :set up LOCAL variable in FN/PROC
EA92    :process LOCAL variable arguments
EAB9    :reset stack pointer, secondary text pointer, execute ....
EAC2    :"Arguments" error routine - error code 1F
EACE    :save global variables
EAF7    :save numeric LOCAL variables to stack
EB07    :save LOCAL variables to stack
```

EB11-EC05 - variable handling

```
EB11    :store variable and save references to stack
EB30    :unpack specified variable to appropriate buffer
EB53    :transfer variable address to A-,Y-registers
EB58    :unload floating point variable from variables table to floating
        :point buffer #1
EBB8    :unload string variable from variables table to Page 6
EBA1    :transfer string from variables table to Page 6
EBAB    :transfer string pointed by 2A,2B to Page 6
EBC1    :CHR$ command - token BD
```

EBC9-EC4F - error and BRK handler

```
EBC9    :reset ERL pointer, set program pointers to PAGE
EBFD    :check if primary text pointers are beyond program pointers
EC06    :default BRK routine
EC37    :error REPORT embedded message
```

EC50-ECA3 - BASIC sound command processing

```
EC50    :SOUND command - token D4
EC76    :ENVELOPE command - token E2
EC93    :set up control block, process OSWORD call
```

ECA4-EE6A - LISTing

```
ECA4    :WIDTH command - token FE
ECB2    :go to "Type mismatch" error routine
ECB5    :confirm variable numeric, confirm floating point to integer, save to
        :variables table
ECB8    :pull variable from BASIC stack and ....
```

```
ECBB    :assign value to numeric variable
ECCA    :transfer integer from buffer to variables table
ECE4    :confirm variable is numeric, convert integer to floating point, ....
ECED    :pack variable from floating point buffer #1 to variables table
ED12    :output character in A-register, expanding if a token
ED22    :find keyword in table
ED24    :check next keyword in table
ED3A    :expand keyword from token
ED49    :PRINT hexadecimal numeral
ED54    :convert HEX numeral to ASCII and PRINT it
ED5C    :if CR, start new line, else PRINT character updating COUNT
ED66    :PRINT HEX numeral + SPACE
ED69    :if there is room, PRINT a space
ED7B    :PRINT line with leading spaces in accordance with LISTO setting
ED86    :PRINT X spaces
ED8D    :LISTO command - token C9+4F
ED9F    :LIST command - token C9
EDFF    :check if new line is above finish line number
EE20    :LIST line from program
EE3A    :PRINT next character
EE5C    :unpack line number and PRINT it
```

EE6B-F08A - FOR/NEXT loop handling

```
EE6B    :check loop status and set flags
EE91    :"No FOR" error routine - error code 20
EE98    :NEXT command - token ED
EECA    :"Can't match FOR" error routine - error code 21
EEDA    :check loop variable and process it accordingly
EEED    :process FOR/NEXT loop with integer variable
EF44    :reset primary text pointers and proceed with program
EF54    :decrement FOR/NEXT loop counter
EF69    :process FOR/NEXT loop with floating point variable
EFA4    :process next statement
EFA7    :"FOR variable" error routine - error code 22
EFB3    :"Too many FORs" error routine - error code 23
EFC0    :"No TO" error routine - error code 24
EFC7    :FOR command - token E3
F037    :store FOR/NEXT loop address in Page 5, increment loop counter
F03A    :increment FOR/NEXT loop counter
F052    :save FOR/NEXT loop parameters to memory
```

F08B-F1D1 - GOSUB, GOTO, ON routines

```
F08B    :GOSUB command - token E4
F08E    :process GOSUB routine
F0A5    :"Too many GOSUBs" error routine - error code 25
F0B2    :"No GOSUB" error routine - error code 26
F0B9    :RETURN command - token F8
```

```
F0CF    :GOTO command - token E5
F0D5    :process GOTO routine
F0E0    :reset primary text buffer pointers, process next statement
F0E7    :test line termination, reset ERRV, process next statement
F0F5    :ON ERROR routine
F10D    :"ON syntax" error routine - error code 27
F118    :ON command - token EE
F18E    :"ON range" error routine - error code 28
F19D    :check for specified line number
F1B2    :seek given line number, carry clear if found
F1B8    :"No such line" error routine - error code 29
F1C7    :go to "Type mismatch" error routine
F1CA    :go to "Syntax error" routine
```

F1D2-F3A8 - INPUT and DATA retrieval

```
F1D2    :INPUT# routine
F1DD    :get next character from primary text buffer
F21C    :INPUT numeric variable
F222    :INPUT integer variable
F22E    :INPUT floating point variable
F242    :discard channel number and flag, process next statement
F247    :INPUT command - token E8
F25D    :INPUT next variable
F2D3    :derive variable from INPUT string
F2DF    :allocate string variable, INPUT next variable
F2E9    :RESTORE command - token F7
F318    :check for more DATA, then process next operation
F322    :READ command - token F3
F343    :move DATA pointers on, process further DATA or next statement
F353    :seek next DATA item if present
F39F    :"Out of DATA" error routine - error code 2A
```

F3A9-F3FE - REPEAT/UNTIL loop handling

```
F3A9    :"No REPEAT" error routine - error code 2B
F3B4    :UNTIL command - token FD
F3D0    :transfer address of next REPEAT/UNTIL loop to primary text pointers
F3D9    :"Too many REPEATs" error routine - error code 2C
F3E7    :REPEAT command - token F5
```

F3FF-F513 - BASIC program line input

```
F3FF    :read line from input stream to Page 6 buffer
F405    :PRINT prompt, then input line from current stream to P7
F428    :start new line and zero COUNT
F42B    :zero COUNT
F430    :seek given line number, set carry if not found
F460    :increment index, pointers, examine next character
```

```
F47F    :increment TOP setting, clear carry
F484    :transfer next character to TOP
F490    :insert BASIC line into program
F4CF    :"LINE space" error routine - error code 00
F4D9    :make space and insert new line
```

F514-F732 - program housekeeping and control

```
F514    :RUN command - token F9
F517    :RUN program
F523    :reset BASIC pointers
F532    :zero variables, except system integers
F53D    :reset TOPOFVARIABLES pointer and loop counters
F554    :pack floating point variable on to BASIC stack
F581    :increment BASIC stack pointer by 5
F593    :push value of variable on to BASIC stack
F597    :push integer variable on to BASIC stack
F5B5    :push string variable from Page 6 to BASIC stack
F5CE    :transfer string from BASIC stack to Page 6 buffer
F5DF    :get STRINGLENGTH from BASIC stack, move pointers up
F5E4    :move BASIC stack pointers up
F5ED    :pull integer from BASIC stack
F602    :increment BASIC stack pointers by 4
F60E    :pull integer from BASIC stack to 37-3A
F610    :pull integer from BASIC stack to 00,X-00,X+3
F631    :if enough memory, move BASIC stack pointers down
F644    :go to "No room" error routine
F647    :transfer integer buffer to 00,X-00,X+3
F658    :clear carry and increment line number search/crunch pointers
F659    :increment line number search/crunch pointers
F665    :load program, set TOP and check if program is legal
F672    :set TOP = PAGE
F695    :reset TOP = TOP + Y
F696    :add A-register to TOP
F6A1    :PRINT "Bad program" and go to IMMEDIATE mode
F6A4    :"Bad program" embedded message
F6B5    :set up filename buffer
F6BD    :terminate filename string with CR
F6C5    :OSCLI command - token FF
F6D2    :go to "Type mismatch" error routine
F6D5    :get filename for LOAD/SAVE
F6E0    :set up LOAD/SAVE parameters
F6EA    :set machine higher order address pointers
F6F6    :SAVE command - token CD
F727    :LOAD command - token C8
F72D    :CHAIN command - token D7
```

F733-F7F0 - data file access commands

```
F733    :PTR command - token CF
F749    :EXT command - token A2
F74A    :PTR command - token 8F
F75B    :BPUT command - token D5
F772    :BGET command - token 9A
F77B    :OPENIN command - token 8E
F77F    :OPENOUT command - token AE
F783    :OPENUP command - token AD
F799    :go to "Type mismatch" error routine
F79C    :CLOSE command - token D9
F7AC    :input channel number to A-,Y-registers
F7B8    :get channel number from primary text buffer
F7C6    :process message embedded in ROM
F7DB    :REPORT command - token F6
F7F0    :"Missing #" error routine - error code 2D
F7FC    :"3.1" embedded message
```

Page 0

```
0000-0001 :LOMEM
0002-0003 :TOPOFVARIABLES pointer
0004-0005 :BASIC stack pointer
0006-0007 :HIMEM
0008-0009 :ERRL
000A      :primary text index
000B-000C :primary text pointer
000D-0011 :RND number store
0012-0013 :TOP
0014      :number of characters in PRINT field
0015      :HEX PRINT flag (bit 7 denotes status)
0015      :variables table index
0016-0017 :ERRV (default B402)
0018      :PAGE hi
0019-001A :secondary text pointer
001B      :secondary text index
001C-001D :DATA pointer
001E      :COUNT
001F      :LISTO option
0020      :TRACE flag (hi bit indicates status)
0021-0022 :maximum TRACE line number
0023      :WIDTH
0024      :REPEAT/UNTIL loop counter
0025      :GOSUBS counter
0026      :FOR/NEXT loop counter (fifteen times number of loops)
0027      :variable type evaluator parameter
0028      :OPT value for assembler
0029-002B :assembly buffer
002A-002D :integer buffer
002A      :integer buffer least significant byte
002A-002B :variable pointer
002A-002B :STEP
002C      :variable type
002C-002D :variables table pointer
002D      :integer buffer most significant and sign byte
002E-0035 :floating point buffer #1
002E      : sign byte
002F      : guard byte
0030      : exponent
0031      : mantissa most significant byte
0032      : mantissa
0033      : mantissa
0034      : mantissa least significant byte
0035      : rounding byte
0036      :length of string
0037-0038 :assembled code location
0037      :OSWORD control block - least significant byte of input line buffer
```

0037-0038 :RENUMBER pointer
0037-0038 :DIM variable name pointer
0037-0038 :program pointer
0037 :PRINT format parameter
0037-0042 :SOUND parameter
0038 :OSWORD control block - most significant byte of input line buffer
0038 :temporary store for sign during division calculation
0038 :number of decimal places in PRINT field
0039 :length of variable name
0039 :length of object code operation
0039-003A :secondary program pointer
0039-003A :BASIC keyword table pointer
0039-003A :DELETE range upper limit line number
0039-003C :integer variable temporary store
003A-003B :variable name pointer
003A-003B :current ORG address
003A :string length used in string comparison
003A :keyword table index
003B-0042 :floating point buffer #2
003B :floating point buffer #2 sign byte
003B :input buffer index
003B :index temporary store
003B-003C :RENUMBER space pointer
003C :floating point buffer #2 guard byte
003C :text index temporary store
003C :most significant byte of higher order address
003C-003D :next variable pointer
003D :floating point buffer #2 exponent
003D-003E :assembler opcode lookup table index calculation workspace
003D-003E :line number temporary store
003D :keyword token processing parameter
003D-003E :next line pointer
003E :floating point buffer #2 mantissa most significant byte
003F :floating point buffer #2 mantissa
003F :variable length
003F :array type
003F-0040 :buffer used in DIM calculation
003F :LOCAL variables counter
003F :line length
003F-0048 :LOAD/SAVE parameter
0040 :floating point buffer #2 mantissa
0041 :floating point buffer #2 mantissa least significant byte
0042 :floating point buffer #2 rounding byte
0043 :temporary store used in decimal numeral input
0043-0046 :work area used in tangent calculation
0044 :ENVELOPE number
0048 :index input temporary store
0048 :power series calculation iteration index store
0049 :decimal index

0049 :index working store used in exponent calculation
004A :temporary store for power used in exponent calculation
004A :temporary store used in decimal numeral input
004A :temporary angle store used in trigonometric calculation
004B-004C :current variable pointer
004B-004C :stack pointer
004D :channel number temporary store
004D-004E :power series pointer
004D :LOCAL variables arguments counter
004D :PRINT status flag
004E :PRINTFIELD parameter
004E :LOCAL variables counter
004E :input flag

Page 1

0100-01FF :6502 stack
0100-01FF :storage of LOCAL variables in FuNctions or PROCedures
0100- :error handling routine relocated to base of 6502 stack

Page 4

0400-046B :system integer variables @%-Z%
046C-047F :temporary storage for floating point variables during program execution
0480-04EC :start of variable address chains
04F6-0457 :start of PROC chain
04F8-04F9 :start of FN chain
04FA-04FF :unused

Page 5

0500-05A3 :FOR/NEXT loop parameter stores (up po ten blocks of fifteen bytes)
05A4-05B7 :REPEAT/UNTIL loop start address hi bytes (up to twenty)
05B8-05CB :REPEAT/UNTIL start address lo bytes
05CC-05E5 :GOSUB return address lo bytes (up to twenty-six)
05E6-05FF :GOSUB return address hi bytes

Page 6

0600-06FF :string buffer
0600-06FF :CALL parameter block

Page 7

0700-07FF :input buffer

user program area

```
PAGE-TOP          :Basic program
LOMEM-VARTOP      :variables table
VARTOP-STCKBASE   :RENUMBER space
STCKBASE- HIMEM   :Basic stack
```

JMP/JSR :calling location
address :address

```
(002A)    :C746
(0037)    :C3D7 F7D8
(WRCHV)   :CC73 ED78
BD0B      :C35F
BD0F      :BDC1
BDDB      :BD1B
BE4C      :BECC BFBE C040 C086
BECA      :BF5E
BEE7      :BF9F BFEB
BEEE      :C02F
BF2F      :BF86 BFD1 C00F
BF57      :BFA7 BFFD C00C
BF5A      :C032
BFBC      :BF54
C043      :BE98 BEE7 BF00 BF39 BF72 BFA4 BFB9 BFC7 BFF0 C012 C037
          :C07C C9A3 CABD CB95 CBAC CBB8 CC0C CC5B EC50 EC76 ECA4
C049      :BDD6 C097 C93C
C04E      :BF11 BF43 BF7C BFC1 BFC4 C009 C028
C051      :BEE4 BF51 C04E
C054      :BF57 BF89 C051
C09E      :C0FC C274
C0B9      :C1CF
C114      :C882
C145      :C1EA C1F3 C262 C293 E96B
C155      :C0C2 C18C C1C6
C15C      :C1D6
C161      :EBDD EBEC EBF5 EBFA
C163      :C138 C13B C13E C16A C180 C1DB C1F8 C298
C16A      :C189 C19F F7D3
C170      :C8DE
C174      :E3E2
C176      :C1E7 C345
C180      :C208 C2A8
C1D4      :C1DE
C1F1      :C1FB
C217      :C253
C291      :C29B
C2AB      :C2C1 C550 C661 C6FE D05C E419 E424 E42F E5B4 E608 E898
          :EA8B EAA4 EFED F016 F069 F2C0 F363 F372 F7B8
C2B6      :BD0F BDDD BEDD BEF9 BF03 BF0A BF14 BF21 BF28 BF3C BF46
          :BF69 BF75 BF7F BF98 BFB0 BFCA BFE3 BFF6 C000 C018 C01F
          :C353 C5A7 C5B8 C5E1 C6AF C754 C794 C94A C9C4 CA26 CB64
          :CC4A CC61 E943 E9FD EA7D EDC1 EDD2 EF5F F0F5 F118 F1DD
          :F247 F2F1 F318
C2C1      :CAF5 CC12 E4C6 E8CC F75F
C2C8      :C63F E50A E83A
```

```
C2F8      :B86E
C30E      :C350 C781 C8F4 F72A
C311      :BDBB C35C D0D7 ED9C EE1D F6B2
C326      :F520
C398      :F10A
C3A2      :D11D
C3B1      :C59E CA2D CB6B CC6E EFA4 F31F
C3B3      :C598 CB6E F1CF F244
C3B6      :C413 C423 C6EA C733 CA87 CAA5 CACF CAF2 CB33 CBFC CC42 ECA1
          :ECAF F0CC F0F2 F315 F3CD F6CF F724 F746 F76F F7A9 F7ED
C3BE      :BD08 D0F9 EA0F EC34 EF51 F04F F0E4 F3FC
C3CC      :E613
C429      :C089 CB12 D0DA D2B4 D49E D54F E3F7 E47D E56E E6C5 E837
          :E8C3 ECB2 F1C7 F6D2 F799
C439      :C410 F216 F340
C43C      :EB04 F2E3
C4D2      :CD6E F644
C4DC      :EA23
C5D9      :C5C1
C685      :C65B
C68B      :C5FC C6A8
C6A5      :F25D F262
C6B3      :E5EF
C6FB      :C724
C739      :C72F E3A0
C784      :C7BE C8C7
C7AD      :C7C9 C858
C7B5      :C802
C8BA      :C7EC C823 C88A
C8FA      :C983
C942      :C9CF CA75
C94A      :CA30
CA26      :C93F
CA33      :C8F7
CA3D      :C774 C909 C9B0 E735
CA4C      :C9C1
CA51      :C9DC CF51
CAF5      :CB9B CC1E EC5D EC80
CAF8      :C65E C906 CF1D E4C0 E84B E8C6 EFF8 F01D
CAFE      :C676 CDC5 CDCD CEAC E349 E357 E39D E7B1 EBC1 F7BF
CB06      :CA78 CA8A CA9E CAE4
CB09      :C6F0 CC18 ED95 F3BA F73A F765
CB0B      :C046 C64C CAFB CEA3 CF63 CF8A D1D9 D1E8 D354 D36F D382
          :D391 D39B E4CC E513 E70B E7E1 E803 E862 F124 F1A5
CB15      :D652 DED4 DFCA E014 E0EB E118 E19E E1A9 E2A7 E37E E38F
CB18      :D273 D53F D5FC D608 D64C F055 F073
CBF5      :CBA6 CBB2
CC18      :CC09
CC71      :C658 CBF9 CC3A CC5E
```

```
CC76      :E9E5
CC84      :C98A CE75 CED7 CEFA
CD08      :E975
CD17      :C3F0 C98F CDA4
CD4C      :C3FA C994 CD9A E97A
CD54      :E988
CD74      :C966
CD76      :E9D9
CD9D      :BDC4 C3FF C8FC CB43 EA5A EFC7 F1E7 F282 F322
CDE4      :CD9D EE98
CDF0      :C707
CDF8      :C3E4 E628
CE7C      :CEC7
CEBA      :CDC8
CEFA      :CEC4 CEE9
CFD5      :CF37 CF75 CF95
CFFA      :C348 C74C C75B C789 C79B CAB0 D0FE EDAF EDDB F19D
D006      :C855 EE5C
D022      :CB06
D02E      :C409 C419 F737
D045      :BE25 C077 C426 C749 EEA3 F1CA
D053      :F425
D05C      :C3ED EFD1
D064      :F762
D067      :C371 ED92 F3B7 F6DD
D06D      :C729 CB30 CB9E CC21 EC65 EC88 ECA7 F79F
D072      :C2D4 C2E6 C2EE C2F5 C3B3 C6D8 C6DF C760 C7AA CAA8 CAC0
          :CAD4 CADD CB7D CBAF CBBB D09B EDE6 F08E F0B9 F0D2 F0E7
          :F30A F514 F7DB
D074      :BDAB D06F
D07C      :D06A
D088      :BD16 C38E E972 EE02 F0FF F3ED
D092      :D106 EF4E F167
D096      :C771
D09B      :F03A
D0AB      :BDBE C3AC
D0FE      :F19A
D120      :D0C2 F0D9
D13A      :C8B2 D12F EE65
D13E      :C8D3 EE20
D18A      :EDEF F1B2 F430
D1C1      :DED1
D1D8      :D617 D620
D279      :EF92
D27C      :D267
D2B7      :D3E3 D3EC D3F7 F407 D412
D2B8      :D3C5
D333      :C043 C08F C36E C6ED CC06 D0DD ED8F F121 F1A2 F3B4 F6D5
D33F      :C557 C609 CAF8 CC15 D064 E3E5 E4E9 E4F7 E670 E7D0 E7F2
```

```
         :E83D EA92 ECB5 F052 F070
D381     :D350 D36B
D388     :D33F
D3B2     :D388 D397
D419     :E8FF
D458     :D26D D2BF D304 D3B2
D48D     :D4F4
D4B1     :D4C8
D4B7     :D50D D521
D542     :D530
D552     :D5E1
D5D1     :D61D
D5D3     :D630
D5E4     :D469 D4D0
D5E7     :D458 D4A4 D4FA
D5EA     :D54C D5DE
D633     :D1E2 D568
D636     :D42E D539 D5E7 D602
D6E7     :D6F2 D74C
D6F5     :C619 E8BD
D70F     :E8B7
D7B2     :D733
D856     :D7FE
D868     :D83E
D87A     :D85C D876
D87C     :D6D7 D6E4 D730 D7E7 D7EC D7F5 D807 D82D D836 D84B D852
D891     :E429 E434 E630
D956     :D8F7
D961     :D94F
D98E     :DA1E DE65
D9AD     :D865 D8DE
D9F0     :D725 D88B D906 DC1B DCA4 DD21 DE1C DEC3 DEFD DFCD E017 E0EE
         :E101 E11B E49D E57E E585
DA0A     :D6ED D91E
DA1E     :DA71 DA80 DA9A DAB2 DDF6
DA21     :DAD0
DA34     :D25E DA15 DA55 DC0E DEC8
DA55     :DA6E DA74
DA58     :DA18 DA1B DA77 DA7A DA7D
DA63     :D747 D781 D927
DABA     :DE7C
DAD4     :CB1C D25B D4AE D4C5 D504 D518 D524 D52D D533 D722 E720
         :ECEA
DB03     :E068
DB19     :D915 DAF9 DCC9 DDF2 DE18 DE6F E21F E72F
DB64     :D279 D78A DCEC DD16 DE21 DF02 E1F0
DB93     :D682 E222
DB97     :D66F D69E E0FB E14D E1D4 E2D4
DB9B     :C55A D753 DEBB DFD4 E061 E0AC E1C2 E1EA E335
```

```
DB9D      :E04B
DBA3      :DCEF DEE0 DFE5 E1C8 E231 F063 F085
DBCB      :D264 D663 D67A D688 DECB E0C3 E112 E237 E246 E2DF EAEE
          :F239
DBFA      :CB0F D0E4 D6A9 E732 ECC7
DBFD      :E45E
DC14      :DBFA DCA7 E1FF E44B
DC69      :DC18 E02A E150
DC82      :DCC6 DCDD DCE3 E21C
DC9C      :D65B E2CE
DCC6      :DCC1
DCCC      :DCE0
DCDD      :E45B
DCE6      :D51E E1CE
DCEC      :DEEB
DCFE      :DCAE
DD13      :D50A DCE6
DD16      :D4B4 DFF3 E05E E079 E0DD E13B E141 E22E E243 EF77
DD1B      :E046 E15B E1F9
DD21      :D791 DD1B
DE1C      :DE6C E72C
DE6C      :D545 D697 E058 E05B E073 E1C5 E1D7 E228 E23D E2EA E342
          :E389
DE6F      :DFBC E77D
DE72      :D92E DD1E
DE82      :DC66
DE92      :DE7F
DE9C      :D772 D7B6 DB04 DC11 DDCA DE26 DEAF E2BC
DEAF      :D6A1 D736 DEBE E1CB E338 F066
DEBB      :E132 E330
DEC3      :D60E DFEC E0C9 E109
DEFD      :DEF7 E1FC
DFCD      :E1D1
DFFF      :DEDD DEE8 DEF4 E055
E003      :D694 DFE2 E234
E007      :E106 E2E7
E00B      :D787 DBC8 E076 E0C6 E22B E240
E014      :E377
E017      :D68B
E0A8      :E052 E162 E1DE E2F4
E0EB      :E0E5
E0FB      :E0F5
E10F      :E396
E11B      :E10C
E127      :E0F8
E12C      :E124
E138      :E0E8
E147      :E135
E1AF      :DEE5 DEF1 E1A6
```

```
E1BB      :E1B5
E1C2      :E0FE
E1D4      :E1BF
E1E4      :DED7 E1A1 E1AC
E259      :E138 E225
E25D      :E13E E23A
E266      :E10F E1ED
E2AA      :D691
E2E7      :D68E E165 E1E1
E2F0      :E2D1
E328      :D668 D67F E2E4
E33B      :E345
E36E      :E368
E3EB      :E43E
E3FF      :F2D6
E43C      :E42C
E499      :E67B
E578      :D1DF D1F2 D586 D596
E585      :DCE9 DD13 E144 E1B8
E596      :E439
E59A      :D5D9 DADE
E5B4      :F2BD F33B
E5F2      :CAFE CB15 D636 E3B1 E3FA E441 E464 E4A6 E571 E6C8 E8A7
          :F786
E608      :E593
E640      :BDCE
E649      :C736
E670      :C649 CF62 CF87 E4C9 E510 E708 E7DE E800 E85F E8CF
E6CF      :C786 C791 C99D CB54 CF13 E374 E470 E4BD E556 E7C0 EDAC
          :F013 F778 F796
E6D1      :D0BF E354 E47A E646 EB55
E752      :D5D3 EB2A
E765      :E726
E783      :E74D E765
E7B1      :E473 E82A
E7C6      :EBC6
E916      :E9EA
E99B      :CA2D
E9F8      :E98B
EA06      :EB0E
EAF7      :EAF1
EB07      :EAFA
EB11      :CB48 EA7A
EB30      :CEA0 E62D EB1C
EB58      :EF69
EBC9      :EC06
ECB5      :EFD4 F32F
ECB8      :BDD3 C420 C939 CB59 F23C F2D9
ECBB      :EAF7
```

```
ECED    :EF82
ED12    :BD6E BD90 EE8B F7E7
ED49    :BD2D ED66
ED54    :ED4E
ED5C    :C635 C6BF D12C D134 D17E ED40 EE4E F2A2 F405
ED66    :BD32 BD52
ED69    :C5D3 C626 D137 ED86
ED7B    :EE29 EE30 EE37
ED86    :BD4B BD61 BD7D C67D D175
EE3A    :EE68
EE98    :EF66
F03A    :F088
F08E    :F17D
F0D5    :D109 F16A
F0E0    :F3D6
F19D    :F08B F0CF F15F F302
F1B2    :D103
F1DD    :F219 F23F
F25D    :F2DC F2E6
F318    :F350
F343    :F332
F353    :F329 F335
F3FF    :F2A5
F405    :C323 C8D8
F428    :BD46 BD9A C59B C671 C682 C8B5 ED72 EDFF F7DE
F42B    :C6E2 CBF2 ED63
F430    :C76E F492
F460    :F47C
F484    :F476 F479
F490    :C34D C8E1
F517    :F730
F523    :C30E C8E4 CAAB F4CC F517
F532    :CA99
F53D    :C335 EC1F F52F
F554    :D258 D26A D4A1 D4C2 D4F7 D515 D52A D536 D5FF D64F E723
F581    :D261 D276 D4B1 D507 D51B D542 D60B D660 D685 E729 EAEB
F593    :EA95 EB20
F597    :BDCB C406 C416 C6F3 C751 C78C C8D0 C903 C9A0 CB4F CC1B
        :CF1A CF5F D2BC D385 D394 D5E4 D633 E4C3 E8C9 EA9C EB2D
        :EDB3 EDCB EFCE F1F4 F2D3 F32C F338
F5B5    :D301 D42B E3BF E4F4 E50D E7DB E7FD E846
F5CE    :D44D E516 E7E4 E806 E865 EB01
F5DF    :C506 D32C E3E8 E540 E559
F5E4    :C546
F5ED    :C439 C72C C76B C8CD C8DB D255 D4BF D512 D527 D58E E4D4
        :E8D4 EACE EAF4 EDC8 EDEC
F602    :D364 D3AB
F60E    :CC2D EA20 ECB8
F610    :C7C3 CA4E CF28 CF70 D1F7
```

```
F631        :E9A2 F559 F59C F5BA
F647        :C081 D58B EAE4 EB19
F658        :F307
F659        :F505
F665        :F727 F72D
F672        :C2E1 C2E9 C7C6 EDE9 F4C9 F6F6
F695        :F47F F4B5
F696        :F68E
F6B5        :F6DA
F6BD        :C4BD F78B
F6D5        :F6C5 F6E0
F6E0        :F665 F6F9
F6EA        :CBBE
F7AC        :C54B F1D4 F733 F75B F79C
F7B8        :E480 F74F F772
F7C6        :C89B F6A1
OSFIND      :F793 F7A6
OSBPUT      :C561 C56D C57A C584 C58D F76C
OSBGET      :F1F9 F205 F20D F224 F230 F775
OSARGS      :F743 F755
OSFILE      :F66F F721
OSRDCH      :E7BD E7C3
OSASCI      :F7D0
OSNEWL      :F428
OSWRCH      :C651 C655 C6E7 CBA3 CBF6 CC26 CC2A CC32 CC37 CC3F
              CC47 ED60
OSWORD      :CAEF E4E0 E700 EC9E F420
OSBYTE      :B825 B82E CBD9 E350 E370 E486 E7BA EC27 EC2C F6EC
OSCLI       :C395 F6CC
```

L/U table address	calling location address
BAE6	C3CD
BB58	C3D2
BC57	BE16
BC91	BE1B
BCCB	BE41
D185	D14B
D1D3	D151

DFS 0.90 - summary and description of contents

ROM type	:1000 0010 (service ROM)
Binary version number	:5A
Copyright offset	:11
Title string	:DFS 0.90
Assembly address	:8000-9FFF
Relocation address	:none
Public workspace	:0E00-1700
Private workspace	:2 pages

DFS 0.90 - Memory Map

8000-8014	:paged ROM protocol
8015-8AB3	:file handling and cataloguing
8AB4-8CE1	:floppy disc controller access routines
8CE2-8D11	:floppy disc controller initialisation parameter :lookup tables
8D12-8DDC	:NMI routines (relocated to Page D)
8DDD-9905	:DFS OS filing system calls
9906-9948	:miscellaneous OSBYTE routines
9949-99C5	:vector and entry point lookup tables
99C6-9ABF	:*DFS and *HELP routines
9AC0-9E86	:disc housekeeping routines
9E87-9FFF	:file construction and listing commands

DFS 0.90 - Zero Page usage

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
10	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
20	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
30	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
40	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
50	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
60	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
70	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
80	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
90	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
A0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
B0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
C0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
D0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
E0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	*
F0	*	*	*	*	*	.	.	.	.	.	.	.	.	.	.	*
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

8271 floppy disc controller (FDC)

Control of the disc filing system rests with an Intel 8271 floppy disc controller chip. This supports either two single or one dual floppy drive using soft-sectored discs. It is a high level controller which relieves the CPU of many of the tasks associated with implementing a floppy disc interface. The 8271 has four registers associated with data transfer (command, parameter, result and status) and a reset register which permits it to be reset under program control by means of a write operation to FE82.

Operation commences with a command phase in which the MOS writes a command and parameters into the 8271 command

and parameter registers. An execution phase in which the FDC carries out the commands then follows. Subsequently, the FDC signals that execution has finished and the MOS reads one or more registers to determine the outcome of the operation.

The data transfer registers are selected by means of read and write operations to FE80 and FE81. The MOS loads an eight-bit command into the command register; bits 6 and 7 select the surface/drive number and bits 0-5 constitute the command opcode. Up to five eight-bit parameters may accompany a command; these are loaded serially into the parameter register. The outcome of the FDC command execution (such as good/bad completion) is signalled in the result register. Bits 0,1 indicate completion code, bits 2,3 completion type and bit 5 the discovery of deleted data. Bits 0, 6 and 7 are not used. The status register indicates the current state of the FDC. The status byte has the following format - bits 0,1 not used, bit 2 non-DMA request - an interrupt is generated with each data byte read from or written to the disc, bit 3 interrupt request - which is cleared by reading the result register, bit 4 - result buffer full, bit 5 - parameter buffer full, bit 6 - command register full, bit 7 - command busy.

Data transfers between memory and the 8271 are carried out by read and write operations at Sheila address FE84.

8000-8014 - paged ROM protocol

Information necessary for the MOS to interface with DFS 0.90 follows standard Acorn format.

8018 - error message handling

DFS 0.90 uses the base of the stack to process error routines. ASCII-encoded error messages are PRINTed via the MOS OSASCI call.

80DA-8AB3 - file handling and cataloguing

The filing system is non-hierarchical, with the result that there is a limit (31 decimal) on the number of files that can be stored. Each file is identified by a one-character directory plus a seven-character filename. There are four possible drive numbers, corresponding to the two surfaces of a dual-disc drive.

When the DFS is active, catalogue information is stored in Pages E and F. The cataloguing routines are concerned with the input, verification and display of filenames and parameters, and with DFS commands such as *WIPE and *LIB and the MOS OSFSC calls which perform housekeeping functions on the catalogue and files.

841E - OSFSC call 05 (*CAT command)
The CATalogue command reads the relevant data from the catalogue track (00) on the disc into Pages E and F and then PRINTs it on the screen in predefined format.

85B8-866B - command lookup table
DFS command in ASCII format, followed by the entry point of the corresponding routines are stored as a lookup table.

8AB4 - update disc write-cycle counter
As a means of identification, a write-cycle counter is stored in the catalogue track of each disc. Every time a write operation is performed, this counter is updated.

8AB4-8CE1 - FDC access routines
The 8271 floppy disc controller handles all low-level access between the DFS and the disc. The access routines are provided to present data and commands to the FDC in a suitable format and to interpret and present the results. If a second processor is active, data are transferred by way of the TUBE, otherwise a suitable buffer is set up in I/O processor memory.

8D12-8DDC - NMI routines
Transfer of data to and from the FDC is by means of non-maskable interrupt routines installed by the DFS at 0D00. Routines are provided for reading from and writing to host and second processor memory. The parameters of these NMI routines are stored in four lookup tables at 9994-99AA. Two of the tables have to be customised by the insertion of the page number of the DFS chip. The position at which this has to be inserted is stored in the lookup table at 99A9-99AA.

8DDD-9905 - paged ROM service calls
In order to accommodate a number of ROMs in the memory space 8000-BFFF, the MOS provides a number of service calls to which the ROMs must respond. These calls set up specific requirements (such as claiming the NMI space, or reserving a sufficiently large amount of public workspace) or make appropriate responses such as performing an autoboot.

94FF-957A - DFS-specific OSWORD calls
The operating system uses the OSWORD call as a general entry point to routines which handle a block of data, the routine being selected by the initial value stored in the accumulator. Many OSWORD routines are stored in the MOS ROM, but if the entry parameters are not recognised, the call is offered in succession to the paged ROMs. Calls 7D (return current disc access counter

setting), 7E (return number of sectors on disc) and 7F (read/write a sector from/to disc) are specific to the disc filing system.

99C6-9ABF - *DFS and *HELP routines
Routines and ASCII messages to provide user guidance. A lookup table of file attributes for use as a prompt when a DFS command is used with the wrong syntax.

9AC0-9E86 - disc housekeeping routines
DFS 0.90 may operate with a single or dual drive configuration. It therefore maintains flags to indicate source or destination disc, source or destination drive and issues prompts for disc changes at the appropriate time during copying operations.

The user program area is used as a buffer and programs may be corrupted whilst copying is taking place.

9E87-9FFF - file construction and listing commands.

RAM usage
Page 0
DFS 0.90 uses a block of addresses from 00A0-00CF for filing system functions. It also accesses a number of MOS zero page locations when it carries out operating system calls.

Page 1
Error routines are relocated to the bottom of Page 1 for execution.

Page D
NMI routines for transfer of data between the FDC and processor memory are relocated to Page D. A table of extended vectors is installed when the DFS is enabled and a table of private workspace pointers is set up on RESET.

Pages E and F
Pages E and F hold current catalogue information read from track 00.

Pages 10 and 11
When DFS is enabled, it copies blocks of operating parameters from its private workspace to pages 10 and 11 (thus permitting direct rather than indirect addressing). On shutdown, it performs the reverse operation.

Pages 12-16
When DFS is active, it uses pages 12-16 as read/write buffers for file maintenance.

8000-8014:paged ROM protocol

8003 :service entry point

8015-8AB3 - file handling and cataloguing

8015 :go to file system control vector entry
8018 :PRINT "Disk " ....
8022 :PRINT "Bad " ....
802B :PRINT "File " ....
8033 :process error message
805B :set error message parameter, BRK code, process message following
8065 :PRINT embedded message following
8074 :increment text pointers, test next character
8083 :restore index (Y), A-register, continue program
808A :reset error message parameter, perfrm warm start
809A :PRINT "."
809C :test error message parameter, PRINT character if required
80B8 :save A-register to bottom of stack
80C2 :PRINT ASCII equivalent of HEX byte
80CA :convert lo nibble to ASCII and PRINT it
80DA :move characters from workspace to current file buffer and PRINT
80E2 :transfer character to public workspace
80EA :transfer 2 characters from current workspace to current file buffer
80ED :transfer character from current workspace to current file buffer
80F4 :fill filename store with spaces
80FE :input filename
8106 :search default directory/drive for command/file
8109 :set command/filename pointers and ....
811B :read filename from input string
818E :"Bad filename" error message - error code CC
81A1 :check if disc has been changed
81AF :"Disk changed" error message - error code C8
81C0 :PRINT directory and filename
81F4 :insert Y spaces, preserving A-register
81FB :shift A-register bits 6 and 7 to 0,1, mask out remaining bits
81FD :shift A-register bits 4 and 5 to 0,1, mask out remaining bits
81FF :shift A-register bits 2 and 3 to 0,1, mask out remaining bits
8204 :shift A-register right 5 times
8205 :shift A-register right 4 times
820A :shift A-register left 5 times
820B :shift A-register left 4 times
8210 :increment index (Y) 8 times
8211 :increment index (Y) 7 times
8214 :increment index (Y) 4 times
8219 :decrement index (Y) 8 times
8222 :set I/O pointers to load address, evaluate sectors required for file
825E :set wildcard store to ASCII "#"
8262 :reset wildcard store to FF

```
8268        :input filename and seek in catalogue
826E        :search default directory/drive and catalogue for file
8271        :search catalogue for file, error message if not found
8276        :"File not found" error message - error code D6
8284        :*INFO command
8296        :search catalogue for filename, carry set if found
829D        :check filename against catalogue entry
82CD        :decrement index to next catalogue entry
82D1        :delete file from catalogue
82D4        :shuffle catalogue to fill vacated place
82EE        :check for alphabetic character
82FC        :PRINT filename + parameters according to OPT 0 setting
8301        :PRINT filename + file parameters
8335        :PRINT 3 file parameter characters + SPACE
8347        :rearrange stack and read catalogue from track 00
834D        :set current directory and drive to default
8358        :initialise string input and set drive number
835D        :input drive number and check if legal
8374        :"Bad drive" error message - error code CD
837E        :calculate file parameters and save to current workspace
83A7        :extract hi order bits from composite byte
83BB        :check for second processor reference, save address hi byte to
            :current workspace
83CF        :transfer 2 characters from file parameter buffer to current
            :workspace
83D2        :transfer character from file parameter buffer to current workspace
83DA        :increment current text pointers
83E1        :rearrange stack, set return address to 8404, reset processor
            :registers
8404        :recover register from stack
840A        :recover registers, leaving parameter on stack
8411        :rearrange stack, set return address to 840A, reset stack pointers
841E        :OSFSC 05 - *CAT command
843D        :PRINT " ("
8442        :PRINT disc write access counter
8448        :PRINT ") <CR> Drive "
8453        :PRINT current drive number + thirteen spaces
845D        :PRINT "Option "
8467        :PRINT boot option (number)
8470        :PRINT " ("
8467        :PRINT boot option (text)
8484        :PRINT ") <CR> Directory :"
8494        :PRINT directory and drive number
84A8        :PRINT "Library :"
84B4        :PRINT library
84C6        :PRINT catalogue entries
84F3        :increment file counter and test if file locked
84F6        :test if file is locked
856F-857E   :boot option embedded messages
```

```
857F        :set up file length and start sector and ....
859E        :check if room on disc for file
85B8-866B   :command name and vector lookup table
866C        :OSFSC 03 - *RUN file whose name is pointed by X,Y
8671        :DFS control entry
86B8        :set OS command/filename pointers, zero index (Y)
86BF        :initialise OS general string input routine
86C3        :*WIPE command
86D4        :' : ' embedded message
86DB        :input "Y/N?" response, if "Y" delete file
86E0        :check filename against catalogue entry
86E9        :delete file from catalogue
86FE        :*DELETE command
8710        :*DESTROY command
872C        :"Delete Y/N?" embedded message
8767        :"Deleted" embedded message
8775        :*DRIVE command
877E        :wait until floppy disc controller ready, then set drive number
8786        :OSFILE 00 - save a block of memory as a file
878F        :write block to disc
8794        :OSFILE FF - load named file
879D        :set up parameters, PRINT filename, read block from disc
87C6        :read block from disc
87D4        :OSFSC 04 - *RUN command
87EE        :OSFSC 02 - */ command file
87FC        :"Bad command" error message - error code FE
880A        :move pointers on, continue execution in indicated processor
8841        :set current file pointers to OS command pointers
884E        :*DIR command
8852        :*LIB command
8860        :input directory and drive settings
888B        :"Bad directory" error message - error code CE
8899        :save directory and set drive
88A3        :*TITLE command
88AC        :initialise title store
88C3        :update disc write cycle counter, save catalogue and warm start
88C6        :save title character to Page 0E
88D2        :*ACCESS command
88EC        :set access bit, process further file, save catalogue
890E        :"Bad attribute" error message - error code CF
891C        :OSFSC 00 - test OPT value, set it if legal
8928        :"Bad option" error message - error code CB
8933        :set OPT 0
893E        :set OPT 4
8958        :"Disk full" error message - error code C6
8961        :if file exists, delete it, set up SAVE parameters
899D        :set start sector, check catalogue space availability
89B4        :PRINT error message, else, add file to catalogue
89BC        :if room for file, insert entry in catalogue
```

```
89DC        :transfer filename to catalogue filename buffer
89E7        :transfer file parameters to catalogue parameters buffer
89F5        :increment number of catalogue entries
8A04        :"Catalogue full" error message - error code BE
8A17        :calculate start sector lo
8A39        :*ENABLE command
8A3F        :check if file for second processor, set load address
8A56        :check if file for second processor, set exec address
8A6D        :*RENAME command
8A99        :"File exists" error message - error code C4
8AAB        :transfer current filename to catalogue buffer
```

8AB4-8CE1 floppy disc controller access routines

```
8AB4        :increment disc write cycle counter, save catalogue
8AD8        :set up directory and drive ....
8ADE        :check if drive ready ....
8AEB        :read catalogue from track 00
8B04        :seek track 00
8B21        :analyse reason for disc access failure
8B28        :"Disk read only" error message - error code C9
8B41        :"Disk fault" error message - error code C7
8B4C        :PRINT error number "at" track/sector
8B50        :" at " embedded message
8B74        :"Drive fault" error message - error code C5
8B85        :go to PRINT error number "at" track/sector routine
8B88        :transfer command and parameters to floppy disc controller
8B8E        :transfer next parameter from lookup table
8B9E        :select drive and switch motor on
8BBD        :read drive status
8BCD-8BD0   :drive lookup table
8BD1        :point hi order address bytes to I/O processor memory
8BDC        :save call, transfer file across TUBE, if necessary
8BE6        :if second processor address given, transfer data via TUBE
8C02        :transfer file across TUBE if indicated
8C11        :initialise write second processor, pass data via TUBE
8C16        :set up floppy disc controller to read data (ten attempts)
8C1F        :initialise read second processor, get data via TUBE
8C24        :set up floppy disc controller to write data (ten attempts)
8C35        :set ATTEMPTS counter to ten
8C3A        :read or write to floppy disc controller
8C5F        :test ATTEMPTS counter
8C78        :set up disc access parameters ....
8CB0        :when disc controller ready, write parameter
8CBD        :*ENABLE drive and write command to floppy disc controller
8CC6        :when disc controller ready, transfer 6502 A-register to 8271 command
            :register
8CCF        :when disc controller ready, transfer 8271 result register to 6502
            :A-register
```

8CD6 :wait until disc controller is ready

8CE2-8D11 floppy disc controller initialisation parameter lookup tables

8CE2-8CE7 :8271 register setting lookup table - initialisation parameters 1
8CE8-8CED :8271 register setting lookup table - initialisation parameters 2
8CEE-8CF3 :8271 register setting lookup table - initialisation parameters 3
8CF4-8CFB :8271 register setting lookup table - initialisation parameters 4
8CFA-8CFF :8271 register setting lookup table - load bad tracks side 0
8D00-8D05 :8271 register setting lookup table - load bad tracks side 1
8D06-8D09 :8271 register setting lookup table - specify non-DMA access mode
8D0A-8D0C :8271 register setting lookup table - seek track 00
8D0D-8D11 :8271 register setting lookup table - verify data

8D12-8DDC NMI routines (relocated to Page D)

8D12-8D86 :NMI routine 0
8D32-8D86 :NMI routine 1
8D5D :NMI routine 2
8D87-8DA1 :NMI routine 6
8DA2-8DBC :NMI routine 4
8DBD-8DCC :NMI routine 5
8DCD-8DDC :NMI routine 3

8DDD-9905 DFS OS filing system calls

8DDD :OSFSC 07 - list range of DFS file handles (11-15)
8DE2 :OSFSC 06 - shut down filing system
8DED :close each open file and save it to disc
8DFA :close file for byte access, save file to disc
8E05 :if channel open, save file to disc
8E55 :set directory/drive - check if disc has been changed
8E58 :check if disc has been changed
8E83 :set current directory and drive
8E90 :go to "Disk changed" error routine
8E93 :DFS OSFIND routines
8E9D :open a file
8EB8 :delete file
8ECB :open file for random access
8EEA :"Too many files open" error message - error code C0
8F02 :"File open" error message - error code C2
8F2F :derive file handle from channel number
8F99 :save filename index and try next channel
8F9E :open file
8FDD :try next channel
8FF2 :OSFIND FF, Y= 0 - update files to disc
8FFB :save file to disc
9007 :DFS OSARGS routines
901C :OSARGS calls with Y= 0

```
902E    :transfer sequential pointer to Page 0 control block
9051    :confirm specified channel is open
9076    :derive channel index from file handle in X-register
907B    :derive channel index from file handle in Y-register
907D    :if file handle legal, convert to channel index
908D    :OSFCS 01 - check EOF (result in X-register)
90A5    :evaluate index from file handle, confirm channel open
90AD    :"Channel" error message - error code DE
90B9    :"EOF" error message - error code DF
90C1    :DFS OSBGET routines
9121    :add sequential pointer to set new start sector, write enable
913A    :write-enable file
913C    :set flag bits according to key, clear carry
9141    :write-disable file
9143    :reset flag bits according to mask, clear carry
914B    :confirm file is read-enabled
9153    :save channel buffer to disc
9190    :go to "Channel" error routine
9193    :go to "File locked" error routine?
9196    :"File read only" error message - error code C1
91A4    :write byte to current channel
91AA    :DFS OSBPUT routines
91B0    :put byte to file
9211    :"Can't extend" error message - error code BF
92A7    :OSARGS 01, Y>0 - write sequential pointer of file
92B6    :confirm sequential pointer is within file, else extend file
92F8    :compare sequential pointer with file length
9310    :compare open file parameter with OSARGS control block parameter
9328    :"Acorn DFS" embedded message
9338    :*DISK command
        :*DISK command
9358    :set up ROM pointers
93B2    :initialise DFS
9427    :"File not found" error message
943C    :set up !BOOT command
9455    :DFS entry routine
9459    :paged ROM service call 01 - establish public workspace requirements
9464    :paged ROM service call 02 - set up private workspace
9482    :paged ROM service call 03 - autoboot
949F    :paged ROM service call 04 - check for unrecognised command
94AB    :paged ROM service call 09 - check for *HELP command expansion call
94BF    :paged ROM service call 0A - claim public workspace
94EE    :paged ROM service call 08 - unrecognised OSWORD call
94FF    :OSWORD 7F - read/write a sector from/to disc
954B    :check for OSWORD 7D
9559    :OSWORD 7D - return current disc access counter setting
9562    :OSWORD 7E - return number of sectors on disc
957B    :DFS OSFILE routines
```

```
95AA  :DFS OSFSC routines
95BD  :OSARGS 02, Y=0 - return with Page 0 command line address
95DO  :DFS OSGBPB routines
962A  :process OSGBPB call
9683  :OSGBPB 08 - read filenames from current directory
9695  :output filenames from catalogue
96CF  :OSGBPB 05 - get disc title and startup option
96DA  :transfer disc title
97OO  :OSGBPB 06 - read current selected directory and drive
9714  :OSGBPB 07 - read library directory and drive
9728  :set Zero Page pointers (B8,B9) to address pointer
9737  :increment address pointer 1061-4
973C  :increment 1060,X-1063,X
9748  :complement 1065,X-1068,X
9756  :set up Page 0 pointers to control block
9761  :set call = 1, write data to TUBE or I/O, increment pointer
976A  :write data to TUBE or I/O buffer, increment pointer
977D  :OSGBPB 01 - put bytes using new sequential pointer
9785  :read data from TUBE or I/O buffer
9798  :OSGBPB 08 - reset ENABLE flag
97A1  :OSFILE 05 - read files catalogue data
97A4  :evaluate file parameters, transfer to current workspace, set A=1
97AA  :OSFILE 06 - delete named file
97B5  :OSFILE 01 - write parameter block data to catalogue
97CO  :OSFILE 02 - write load address to catalogue
97C8  :OSFILE 03 - write exec address to catalogue
97DO  :OSFILE 04 - write file attributes to catalogue
97D6  :transfer file attributes to buffer, save catalogue data to disc
97D9  :save catalogue data to disc, set A=1
97DF  :transfer load address from current workspace to catalogue buffer
97FB  :transfer exec address from current workspace to catalogue buffer
981E  :transfer attributes from current workspace to catalogue buffer
9837  :seek file in default directory/drive, if found check access
983C  :check if file is locked
9841  :"File locked" error routine - error code C3
984C  :if file unlocked and closed, open it
984F  :if file closed, open it
985A  :if file in default directory/drive, set input buffer to load address
9864  :search default directory/drive for file, if found ....
986E  :use load address to set current workspace pointers
9879  :determine free RAM space
988F  :claim public workspace, set ownership flag true
989E  :set pointers to private workspace
98AC  :claim NMI space and set up interrupt routines
98C3  :transfer interrupt handling routines to Page OD
98E7  :when disc controller ready ....
98EA  :claim NMI space
```

9906-9948 miscellaneous OSBYTE routines

```
9906    :rearrange stack, flush input buffer
9909    :OSBYTE 0F - flush input buffer
9911    :OSBYTE C7 - read/write *SPOOL file handle
9917    :select output stream determined by input A-register
9918    :OSBYTE 03 - select output stream determinded by input X-register
991C    :OSBYTE EC - read/write character destination status
9920    :OSBYTE C7 - read/write *SPOOL file handle
9924    :OSBYTE EA - read flag indicating TUBE presence
9928    :OSBYTE A8 - read address of ROM pointer table
992C    :OSBYTE 8F - issue paged ROM service request
9930    :OSBYTE FF - read/write startup options
9934    :set Y=FF and call OSBYTE
9939    :"L.!BOOT" embedded message
9941    :"E.!BOOT" embedded message
```

9949-99C5 vector and entry point lookup tables

```
9949-9956 :Page 2 vector lookup table
9957-996B :DFS OS call extended vectors installed at 0D9F
996C-9974 :OSFSC routine address lo lookup table
9975-9980 :OSFSC routine address hi lookup table
997E-9980 :OSARGS routine address lo lookup table
9981-9983 :OSARGS routine address hi lookup table
9984-998B :OSFILE routine continuation address lo lookup table
998C-9993 :OSFILE routine continuation address hi lookup table
9994-999A :NMI routine address lo lookup table
999B-99A1 :NMI routine address hi lookup table
99A2-99A8 :NMI routine length (bytes) lookup table
99A9-99AA :current paged ROM position in NMI routine lookup table
99AB-99B3 :OSGBPB routines address lo lookup table
99B4-99BC :OSGBPB routines address hi lookup table
99BD-99C5 :OSGBPB routine processing parameter lookup table
```

99C6-9ABF :*DFS and *HELP routines

```
99C6    :*DFS command
99CB    :*HELP command
99CC    :"DFS 0.90" embedded message
99D9    :display commands from lookup table
99EE    :*UTILS command
99F5    :user input command
9A01    :initialise OS general string input routine, check for null string
9A06    :"Syntax" error message - error code DC
9A13    :display command + attributes and warm start
9A19    :set command lookup table index and ....
9A1B    :display command + attributes from lookup table
9A34    :process command attribute parameter
```

9A4E :display command attribute
9A5B-9ABF :command attribute lookup table

9ACO-9E86 disc housekeeping routines

9ACO :*COMPACT command
9AC3 :PRINT "compacting drive"
9AD7 :print drive number ....
9AFB :check if more files to compact
9B04 :"Disk compacted" embedded message
9B2C :"free sectors" embedded message
9B3D :OSARGS 00 - return with DFS code (4) in A-register
9B40 :compact next file
9B97 :compact current file
9BB5 :PRINT filename + parameters, check if more to compact
9BBD :check if command is ENABLEd
9BC2 :"Not enabled" error message - error code BD
9BD2 :set up source and destination drives
9C02 :"Copying from drive" embedded message
9C1E :"to drive" embedded message
9C38 :check if single or double drive, prompt for disc
9C43 :if twin drive system, prompt for destination disc
9C53 :"Insert" embedded message
9C62 :"source" embedded message
9C6D :"destination" embedded message
9C7C :"disk and hit a key" embedded message
9C93 :flush input buffer, get a character
9C9E :input 'Y/N?' response
9CB8 :PRINT "Disk full" error message
9CBB :*BACKUP command
9D27 :*COPY command
9DB7 :save buffer contents to disc
9DF5 :interchange data between current file and public workspace
9E06 :transfer block by block from source to destination disc

9E87-9FFF file construction and listing commands

9E87 :*TYPE command
9E8E :*LIST command
9EA4 :read character from file and process it
9ECB :close file
9ED0 :*DUMP command
9F10 :"**" embedded message
9F48 :*BUILD command
9F52 :PRINT line number and save line to file
9F92 :acknowledge detection of ESCAPE
9F9A :PRINT CR, preserving A-register
9FA2 :PRINT line number
9FB7 :PRINT byte, with leading space if required

```
9FC1    :PRINT nibble or, if zero, SPACE
9FCB    :PRINT two spaces, preserving A-register
9FCE    :PRINT a space, preserving A-register
9FD7    :prepare stack, zero line number counter, point X,Y to filename
```

Page 0

```
00A0        :previous NMI space holder
00A1        :8271 FDC command temporary store
00A2        :ATTEMPTS counter
00A3-00A5   :loading counter
00A6-00A7   :I/O pointer
00A8-00A9   :line number (for *LIST, *DUMP)
00A8        :source/destination flag
00A9        :single/double drive copy flag
00AA        :access status flag (bit 7)
            :source/destination drive flag (bits 0,1)
00AB        :file counter
            :catalogue index
            :PRINT line number flag
00AC-00AD   :input buffer pointer
00AE-00AF   :text pointer
00B0-00B1   :current workspace pointer
00B2        :filename character counter
00B3        :current channel index
            :temporary store
00B4-00B5   :OSGBPB control block pointer
00B4        :current file status
            :OSFIND parameter
            :startup option
00B6-00B7   :OS control block pointer
00B6        :temporary store
00B8-00B9   :NMI routine pointer
00B8        :lookup table index
00BA        :current track
00BB        :current sector
00BC-00BD   :current filename pointer
            :data file buffer pointer
00BE-00BF   :load address
00C0-00C1   :exec address
00C2-00C3   :file length
00C4-00C5   :start sector
00C6-00CC   :filename
00C7-00CD   :filename input buffer
00CA-00CB   :next available sector
00CC        :index of current file in catalogue
00CE        :current directory
00CF        :current drive
00EF        :most recent OSBYTE/OSWORD call number
00F0-00F1   :OS call X,Y parms
00F2-00F3   :OS command name pointer
00F4        :current paged ROM pointer
00FF        :ESCAPE flag
```

Page 1

0100-01FF :6502 stack
0100- :error handling routine relocated to base of 6502 stack

Page D

0D00-0D74 :NMI routines
0D9D-0DB8 :extended vector table
0DF0-0DFF :paged ROM private workspace pointer hi

Page E

0E00-0E07 :disc title (first 8 bytes)
0E08-0E0E :first filename
0E0F :first file directory
0E10-0E16 :second filename
0E17 :second file directory
:repeated for remainder of page (31 files)

Page F

0F00-0F04 :disc title (last 5 bytes)
0F05 :number of catalogue entries (*8)
0F06 :bits 0,1 - number of sectors on disc (2 hi order bits)
0F06 :bits 4,5 - !BOOT startup option
0F07 :number of sectors on disc (8 lo order bits)
0F08-0F09 :first file LOAD address (lo and middle order bits)
0F0A-0F0B :first file EXEC address (lo and middle order bits)
0F0C-0F0D :first file length (lo and middle order bits)
0F0E :bits 0,1 - first file start sector (hi order bits)
:bits 2,3 - first file LOAD address (hi order bits)
:bits 4,5 - first file length (hi order bits)
:bits 6,7 - first file EXEC address (hi order bits)
0F0F :first file start sector (8 lo order bits)
0F10-0F11 :second file LOAD address (lo and middle order bits)
0F12-0F13 :second file EXEC address (lo and middle order bits)
0F14-0F15 :second file length (lo and middle order bits)
0F16 :bits 0,1 - second file start sector (hi order bits)
:bits 2,3 - second file LOAD address (hi order bits)
:bits 4,5 - second file length (hi order bits)
:bits 6,7 - second file EXEC address (hi order bits)
0F17 :second file start sector (8 lo order bits)
:repeated for remainder of page (31 files)

Pages 10-11 (public workspace - used only when DFS active)

1000-11FF:DFS parameter store, when active (relocated from private workspace)

Pages 12-16 (public workspace - used only when DFS active)

1200-16FF:5 one-page read/write access buffers

JMP/JSR :calling location
address :address

```
(00AE)    :8087
(00C0)    :883E
0100      :8097
(USERV)   :0D3C 0D5C (relocated from 8D6E)
(FSCV)    :8015
0406      :883B 8C0B 9623
0D38      :0D14 (relocated from 8D26)
0D3E      :0D1D 0D2F 0D4F (relocated from 8D2F, 8D61)
(10D8)    :962A
8015      :9343
8018      :8958 8B28 8B41
8022      :818E 8374 87FE 888B 890E 8928
802B      :8276 8A99 8F02 9196 9841
8033      :81AF 8A04 8B74 8EEA 90AD 90B9 9211 9A06 9BC2
805B      :8018 8022 802B
8065      :843D 8448 845D 8470 8484 84A8 86D4 872C 8767 8B50 9328
          :9427 99CC 9AC3 9B04 9B2C 9C02 9C1E 9C53 9C62 9C6D 9C7C
           9F10
8074      :8080
808A      :8B71 9A16
809A      :81D3 849A 84BA
809C      :807D 80D5 81D0 81DD 81EE 8435 847D 84A0 84C0 8B5E 9A21
          :9A3E 9A54 9CAF 9F9D 9FD1
80C2      :832F 833A 8445 8B4D 8B59 8B6E 9B29 9EEA 9EEF 9F02
80CA      :80C6 8329 8455 846D 8497 84B7 9ADD 9B25 9C1B 9C2E 9FC6
80DA      :950E 9593
80E2      :80DF
80EA      :80DA 9590
80ED      :80EA
80F4      :8101 8109
80FE      :8268 88D8 8A78 8A91 941E 9D35
8106      :826E 87DD 8961 8EA9 9864
8109      :87F6
811B      :8179
81A1      :86E9 8748
81AF      :8E90
81C0      :8304 8569 86D1 8721
81F4      :845A 84A5 855E
81FB      :8A5D
81FD      :8235 8582 8F5B 8F7D 9B6B 9D6D 9DDB
81FF      :8A46
8204      :8F33 8F8C 905D
8205      :80C3 846A 8BC7 9411 951C 96F5 9A31 9FB9
820A      :9087
820B      :8947 8E39
8210      :82A0 82A8 84DF 84F3 8532 89F8 8AA6 96AD
```

```
8211      :8341 8525
8214      :92AD
8219      :86F5 8759 89B6 9AFD
8222      :87C9
825E      :8284 86C3 8713 88D2 9D27
8262      :86FE 8A6D 957F
8268      :828A 86C9 8704 8719
826E      :8794
8271      :826B
8276      :88E9 8A82 9D46 9EA1
8296      :8271 874B 87E3 87F9 88E4 8964 8A7D 8A94 8E64 8EAC 9421
          :9867 9DC1 9DC8
829D      :8290 86E3 8727 875F 88FC 9DB1
82D1      :86EC 870A 8753 8969 97B0 9DCD
82EE      :82B4 86A9 96A0
82FC      :8707 87C3 88F9 89F0 9B42
8301      :828D 9BB7 9D4B
8335      :8319 831C 831F
8347      :81A7 88A9 9552 9AEB 9D9F 9DC5
834D      :80FE 8106 88A6 8940 8AD8 954F 9683 96CF
8358      :8ADB 9AC0
835D      :8778 887F 9BDA 9BE5
8374      :8148
837E      :8311 878C 879A 97A4 97AD
83BB      :83AC
83CF      :8391
83D2      :83CF
83DA      :8042 8074
83E1      :809C 81A1 81C0 8301 8347 837E 891C 8DE2 8E97 900B 902E
          :91A4 91AA 92A7 9484 949F 94AB 94C1 95D4 9737 97DF 97FB
          :981E 984F 98AC 98EA 98F2 9906 9A34 9C38 9C43
8404      :840E
8411      :8E9D 901C 90C1 94EE 957B
84E4      :856C
84F6      :84E6 8513
857F      :89B9
859E      :89AE
8671      :94A4 99EB
86B8      :841E 866C 87D4 941B
86BF      :8116 8358 867D 8864 88DD 8A70 8A8C 99F5 9A01 9BD2 9BE0
           9D2D
86E0      :86FB
877E      :813C 8355 836F 87F3 886B 8873 8E8D 9505 9D3E 9D9C
878F      :9179 9E5E
879D      :880A
87C6      :9189 9E38
8841      :87D7
8860      :8854
88C3      :97D9
```

```
88C6      :88B0 88BC
8958      :9CB8
8961      :8786 8EC8
899D      :8987 9DE0
89BC      :89B1
8A17      :89D9
8A3F      :87B0 9DD3
8A56      :87BE 9DD6
8AB4      :86EF 870D 8764 88C3 8955 89FE 8E47 9234 9BAF 9D24
8AD8      :8421
8ADE      :8296 9686 96D2
8AEB      :834A 8943 9407 9CD5 9CF4 9D14
8B04      :8ABF 8AEE
8B21      :8C63
8B4C      :8B85
8B88      :8ABC 8B19 93F7 93FE
8B8E      :8B99
8B9E      :8AEB 9154 9508 9E35 9E5B
8BBD      :8ADE 8BB4
8BD1      :915F 9E06
8BDC      :8C73
8BE6      :8C13 8C21 9521
8C02      :8BE3
8C11      :87C6
8C16      :8AF1
8C1F      :878F
8C24      :8AD3
8C35      :8AC2 8C3F
8C3A      :8D7B
8C78      :87CC 8AF9 8CFC
8CB0      :8B68 8B96 8BAB 8BB1 8C8E 8C93 953A
8CBD      :8B63 8B8B 8BBF 8C89 9518
8CC6      :8BA6
8CCF      :8ACE 8B1C 8B6B 8BC2 8C3A 9541
8CD6      :877E 8CCF 98E7
8DE2      :8DEA
8DED      :8FF6
8DFA      :8E9A 8FFF
8E05      :8DF5 920E 9AE5
8E55      :8E1D
8E58      :91C7
8E83      :8E55 90E5 91BB
8F02      :9857
8F99      :8EE0
8F9E      :8ED2 9852
8FDD      :8F9B
8FF2      :94E0
902E      :9644
9051      :8E06 90A8
```

```
9076      :9202
907B      :8DFD 90A5
907D      :9078
90A5      :9031 9092 90C4 91AD 92AA
90AD      :8E02 9190
90C1      :9765
9121      :9186 9250
913A      :92F5
913C      :90D6 926D 9294
9141      :911B 9279
9143      :9181 92D4 93A5
914B      :8E4D 90E8 9242
9153      :90EC 9256
91A4      :92BD
91AA      :9780
91B0      :91A7
92A7      :9019 9641
92B6      :92C0
92F8      :9096 90C8 91C2 924B 928D
9310      :92B0 92B6
9320      :9498
9455      :8003
962A      :9659
9728      :9775 9790
9737      :9772 977A 978D 9795
973C      :9660 9665 9671
9748      :9651 966C
9756      :95FA 9676
9761      :9700 970B 9714 971F
976A      :96B4 96BC 96D7 96DF 96EA 96F8 96FD 9708 9711 971C 9725
9785      :977D
97DF      :97B8 97C3
97FB      :97BB 97CB
981E      :97D6
9837      :97AA 97B5 97C0 97C8
983C      :8ECF 984C
9841      :9193
984C      :82D1 8A85
984F      :88EC 97D3
985A      :97A1 97D0
9864      :9837 985A
986E      :8789 8797
9879      :9AE8 9BFF
988F      :9389 93B9
989E      :937A 94C4 9894
98AC      :8B06 8C18 8C26 9527
98E7      :87D1 8B01 9237 9BB2
98EA      :8B21 9548
9906      :9C93 9C9E
```

```
9911        :9209
9917        :80AC
9918        :80B5
991C        :80A5
9920        :91FF
9924        :93BC
9928        :9351
992C        :9377 9891 98FD
9930        :93E7
9934        :98B9
99CB        :94BA
9A01        :8287 86C6 8701 8716 8775 88A3 88D5
9A06        :8A75 9BD7 9D32 9FE9
9A19        :99DE 9A13
9A1B        :9A24
9A34        :9A2E
9A4E        :9A57
9AFB        :9BBA
9BB5        :9B94
9BBD        :8710 9CBB
9BD2        :9CBE 9D2A
9C38        :9CCD 9D38 9D96 9E32
9C43        :9CEC 9DBA 9E58
9C9E        :86DB 8740
9DB7        :9E44
9DF5        :9DB7 9DEB
9E06        :9BAC 9D09 9D93
9EA4        :9EC4
9ECB        :9F97
9F52        :9F8F
9F9A        :8332 84C3 84F0 854E 8551 86E0 8724 8745 99E1 9AE0 9C31
            :9C9B 9EC8 9F35
9FA2        :9EB1 9F52
9FB7        :9FB2
9FC1        :9FBD
9FCB        :81CB 81E4 8566 99DB
9FCE        :81F1 81F4 8316 8344 9EB4 9EF2 9F05 9F55 9FCB
9FD7        :9E87 9E8E 9ED0 9F48
GSINIT      :86C0
GSREAD      :811B 812E 813F 8153 816C 817C 8189 835D 8876 8882 889B
            :88B7 8905 99FA
OSFIND      :9E97 9ECD 9ED5 9F4D
OSBPUT      :9F7C 9F8C
OSBGET      :9EA4 9EFB
OSRDCH      :9C96 9CA1
OSASCI      :80B2 9EB8 9F2E
OSWORD      :9F6E
OSBYTE      :8DE7 9489 9495 987B 9883 9936 9F94
OSCLI       :9452
```

L/U table address	calling location address
856F	847A
85B8	8681 868C 8699 86AF 9A1C 9A2B
85B9	86B3
8BCD	8BAE
8CE2	8B88 8B8F
993C	935C 9362
9949	9348
996C	95B5
9975	95B1
997E	9029
9981	9025
9984	95A5
998C	95A1
9994	98C5
999B	9ACA
99A2	98CF
99A9	98E0
9A5B	9A43
9A5B	9A4F
99AB	95E4
99B4	95EA
99BD	95F0

NFS 3.34 - summary and description of contents

ROM type	:1000 0010 (service entry point, no :language entry+)
Binary version number	:03
Copyright offset	:0C
Title string	:NET
Assembly address	:8000-9FFF
Relocation address	:N/A
Public workspace	:0E00-1000
Private workspace	:2 pages

+ a language entry routine is, in fact, provided

NFS 3.34 - Memory Map

8000-8014	:paged ROM protocol
8015-81E5	:entry routines and lookup tables
81E6-83AF	:NFS startup and shutdown
83A2-8BD5	:Econet OS filing system calls
8BD6-8BF1	:Econet command and entry point lookup table
8BF2-8DAE	:catalogue routines
8DAF-9006	:Econet-specific OSBYTE and OSWORD calls
9007-90FB	:error processing
90FC-9306	:language entry calls
9307-974B	:TUBE BRK and communications routines (for relocation)
9663-96F5	:ADLC setup
95F6-9FCA	:network handshaking routines
9FCB-9FD8	:NMI routines (for relocation)

NFS 3.34 - Zero Page usage

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
10	*	*	*	*	*	*	+	+	+	+	+	+	+	+	+	+
20	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
30	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
40	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
50	+	+	+	+	+	+	+	*	*	*	*	+	+	+	+	+
60	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
70	+	+	+	+	+	+	.	.	.	.	.	.	.	.	.	.
80	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
90	.	.	.	.	.	.	.	.	.	.	*	*	*	*	*	*
A0	*	*	*	*	*	*	*	*	.	.	.	.	.	.	.	.
B0	*	*	*	*	*	*	*	*	*	.	.	*	*	*	*	*
C0	*	*	.	.	*	.	.	.	*	*	*	*	.	*	*	*
D0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
E0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	*
F0	*	*	.	.	*	.	*	*	.	.	.	.	.	.	.	*
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

+ only when second processor is active

Econet is a bus-based local area network (LAN) which is implemented on the BBC microcomputer as a filing system. NFS 3.34 is in the form of an 8K paged ROM occupying address space 8000-9FFF. Each Econet station is connected to a twin-conductor data line. Synchronisation is provided by a system clock, the maximum rate of which is determined by the physical length of the network bus.

The heart of NFS 3.34 is the Motorola 6854 advanced data link controller (ADLC) chip. The ADLC transmits and receives data

in the form of a frame comprising a start flag, an address field, a control field, an optional data field, a frame check sequence field and a closing flag. In Econet, the address field contains a two-byte address of the receiver station followed by a two-byte transmitter station address. One byte of each address is the station ID and the other, the network ID.

The ADLC has four control registers, two status registers, three transmit registers and three receive registers, the transmit and receive registers being configured as FIFO buffers. The 6845 occupies addresses FEA0-FEA3. A read to FE18 indicates station ID. Locations FE18 and FE20 are also used by NFS 3.34 for interrupt switching.

8000-8014 - paged ROM protocol
As a paged ROM, NFS 3.34 follows the Acorn paged ROM protocol. However, despite the indication in the ROM type that it is not a language, there is a language entry point.

8014-801C - stack return address
A number of routines perform selective branching by pushing addresses obtained from these lookup tables on to the stack and then using them as return addresses for the continuation routine. The block of addresses is selected by a lookup index offset stored in the initiating routine. The lookup table includes addresses for paged ROM service calls, language entry calls, OSFSC calls and Econet-specific OSBYTE calls.

81E6-83A1 - NFS startup and closedown
On startup the station ID is PRINTed, a check is made for the presence of the network clock, MOS OS vectors and extended vectors are installed and the filing system vectors and static workspace claimed.

83A2-8BD5 - NFS MOS filing system calls
Econet provides a full implementation of the MOS filing system calls. In particular, files possess a complete range of attributes which control their access by designating them as publicly or privately owned.

8BD6-8BF1 - Econet command and entry point lookup table
Many of the commands associated with Econet are specific either to the file server or to the printer server. In the former case, the corresponding routines are loaded in from disc with the file server software and in the latter case, they are stored in a separate paged ROM. The universal commands and their entry points are listed in this lookup table.

8DAF-9006 - Econet-specific OSBYTE and OSWORD calls

NFS 3.34 provides a block of OSBYTE (32-35) and OSWORD (10- 14) calls to deal with identification of various network stations, and the preparation and decoding of data so that it is in a suitable format for processing by the ADLC. The OSWORD calls are trapped when they are offered by the MOS through a paged ROM service call.

9307-974B - TUBE communications

When a second processor is active, NFS installs routines in Page 0 and Pages 4-6 to handle communications across the TUBE and to provide the necessary interfacing to process MOS calls. These routines are stored in the ROM and relocated when required.

9663-96F5 - ADLC setup

On RESET the ADLC registers required initialisation from the stored default values. Econet communication employs non-maskable interrupts to control the 6502 processor, so the NMI space has to be claimed and the routines installed in Page D.

95F6-9FCA - network handshaking

Econet communication is relatively slow, involving elaborate handshaking procedures. The space allocated in Page D for NMI routines is not large enough to accommodate these procedures, so only the entry and exit routines are installed there. The entry routine ends with an unconditional jump to a continuation routine which, in due course, returns the user to the exit routine. As the handshaking procedure progresses, different continuation routine address are placed at the end of the entry routine.

0016-0060 - TUBE BRK handling routines

```
0016          :TUBE BRK handling routine
0036          :save parameters, execute call from TUBE
003A          :read byte from second processor
0057-005A     :second processor relocation address
005B          :go to TUBE language entry routine
005E          :go to TUBE service entry routine
```

0400- 06FF - TUBE communication routines

```
0400          :TUBE language entry point
0403          :TUBE service entry point
0406          :access TUBE
040E          :claim TUBE for filing system N
0414          :set TUBE ownership flag
0419          :release TUBE from filing system N
0423          :save TUBE access status
0426          :set up TUBE data transfer
045D          :claim TUBE, signal via FIFO2, read from second processor
0464          :when FIFO4 ready, flag FIFO3 twice, send Y by FIFO1
0473          :TUBE language entry routine
04CC          :copy code to second processor memory, then execute it
04E0          :set pointers to relocation address, access TUBE
04E7          :signal TUBE and execute TUBE call
04EF          :load parameters, read byte + carry across TUBE FIFO2
04F7          :when ready, read byte from TUBE FIFO2
0500          :access vectored OS call (after lo byte reset)
0500-051B     :TUBE OS call vector table
0500          :OSRDCH
0502          :OSCLI
0504          :OSBYTES 0-7F (with X-parameter only)
0506          :OSBYTES 80+ (with X- and Y-parameters)
0508          :OSWORD calls >0
050A          :OSWORD 0
050C          :load parameters, read call and carry status from TUBE
050E          :save parameters, recover call from stack
0510          :OSARGS
0512          :OSBGET
0514          :OSBPUT
0516          :OSFIND
0518          :OSFILE
051A          :OSGBPB
051C          :TUBE OSWRCH interface
0522          :if ready execute TUBE call, else, PRINT from second processor
0543          :TUBE OSBGET interface
0550          :TUBE OSRDCH interface
055F          :signal second processor and transfer byte via TUBE
0569          :TUBE OSFIND interface
```

```
0580        :TUBE OSFIND 0 interface
058C        :TUBE OSARGS interface
05B1        :read string from TUBE to Page 7 buffer
05CB        :read from second processor
05CD        :write byte to TUBE FIFO2 and read from second processor
0626        :TUBE OSBYTE 0-7F interface
063B        :TUBE OSBYTE 80+ interface
065D        :TUBE OSWORD >0 interface
06A3        :TUBE OSWORD 0 interface
06BB        :write string from Page 7 buffer to TUBE
06D0        :when ready, write byte to TUBE FIFO2
06D9        :when ready, write byte to TUBE FIFO4
06E2        :write FF to TUBE FIFO1
06E8        :TUBE EVENT handling routine
06F7        :when ready, write byte to TUBE FIFO1
```

0D00-0D1F - NMI routines

```
0D00        :NMI routine entry point
0D04        :save Y-register, switch ROM, continue NMI routine
0D07        :NFS ROM page number
0D0C        :NMI continuation address lo
0D0D        :NMI continuation address hi
0D0E        :reset continuation address, switch ROM, restore registers, enable
            :interrupts
0D14        :switch ROM, restore registers, enable interrupts
```

8000-8014 - paged ROM protocol

```
8000        :language entry
8003        :service entry
8010        :"ROFF" embedded message
8014-8020   :error message offset lookup table
8021-8044   :stack return address lo
8045-8068   :stack return address hi
8069        :paged ROM service call 07 - process unrecognised OSBYTE call
8079        :*INFO command
808C        :ECONET OSFSC entry point
8099        :language entry routines
809F        :set up continuation address from lookup table
80AF        :service entry routines
80B5        :paged ROM service call FE - TUBE post initialisation
80C8        :paged ROM service call FF - TUBE main initialisation
80F9        :relocate TUBE communications code to Pages 4-6
8111        :up relocate TUBE BRK handling routines to Page 0
811F        :check for low value service calls
8123        :paged ROM service call 12 - initialise filing system Y
812B        :paged ROM service calls 00-0C
8145        :paged ROM service calls 00,06,0A (RTS)
```

```
8146        :test if keyboard enabled, if so, disable it and ....
815C        :set current ECONET call interception status
8172        :paged ROM service call 04 - unrecognised command
8179        :*ROFF command
8184        :*NET command - initialise ECONET filing system
819B        :check OS command against lookup table
81B2        :flush OS command to non-space, check for <CR>
81BC        :paged ROM service call 09 - *HELP expansion
81CC        :cause OS to issue filing system shutdown call
81D1        :paged ROM service call 03 - autoboot
```

81E6-83A1 - NFS startup and shutdown

```
81E6        :PRINT station ID
8205        :"No Clock" embedded message
8212        :"<CR> <CR>" embedded message
8217        :install NFS vectors, claim vectors, public workspace
822E        :claim vectors and public workspace, boot system
8245        :"I .BOOT" embedded message
824D-825A   :ECONET filing system calls vector lookup table
825B-826E   :extended vector lookup table
826F        :paged ROM service call 01 - reserve absolute workspace
8276-8277   :error handling routine address lookup table
8278        :paged ROM service call 02 - reserve private workspace
8297        :install default NFS parameters
82C3        :set station ID and initialise NFS
82D1        :set ROM table pointers, NETV
82E5        :set up extended vector table
82FD        :OSFSC 06 - shut down NFS
830E        :set up filing system workspace in Page 0, clear carry
8310        :set up filing system workspace in Page 0
831C        :initialise filing system Page 0 workspace
8334-833F   :Page 0 filing system workspace default parameters
8349        :*BYE command
```

83A2-8BD5 - ECONET OS filing system calls

```
83A2        :ECONET OSBPUT entry point
83A3        :put byte to NFS
8424        :"No reply" error routine
8428        :read error code, relocate error routine, process it
842A        :relocate error message, process error routine
8444        :"*SP." embedded message
8448        :"Out of data" error routine
844A        :process error routine
847A        :test ESCAPE flag
8485        :ECONET OSBGET entry point
84A4        :increment index (Y) 5 times
84A5        :increment index (Y) 4 times
```

```
84AA        :decrement index (Y) 4 times
84AB        :decrement index (Y) 3 times
84AF-8507   :error message and error code lookup table
8508        :save A-register, set up parameter block pointers (BB-BF)
853B        :PRINT the following embedded message
8555        :flush to non-space, check for "A"
8560        :evaluate user ID
8588        :update user environment each channel
8589        :clear carry and derive channel number from file handle
858A        :derive channel number from file handle
85A5        :derive index from channel number
85AF        :convert HEX numeral to decimal and PRINT it
85BC        :calculate decimal digit and PRINT it
85DA        :OSFSC 07 - list ECONET file handles (20-27)
85EB        :convert HEX byte to ASCII and PRINT it
85F6        :convert HEX digit to ASCII and PRINT it
8640        :PRINT a space
8690        :pull 3 bytes from stack
8694        :ECONET OSFILE entry point
86CB        :OSFILE FF - load named file
8741        :OSFILE 00-06
8746        :OSFILE 00 - SAVE memory block as file
881F        :OSFSC 01 - check for EOF
8844        :OSFILE 01-06
8859        :OSFILE 02,03 - write load/exec address to catalogue
886E        :OSFILE 01 - transfer data from parameter block to catalogue
888A        :OSFILE 06 - delete named file
8899        :OSFILE 04 - write file attributes
88A3        :OSFILE 05 - read catalogue data
8949        :ECONET OSFIND entry point
8DF8        :OSARGS 00,02 - read sequential pointer and length of file
8912        :OSARGS 01 - write sequential pointer
892C        :restore A-,X-,Y-registers from temporary store
8936        :OSARGS 00 (Y=0)
893D        :OSARGS 01 (Y=0) - return command line address
8941        :open file for read access
8957        :open file for write or read/write access
8988        :close all files
898F        :close specified file
89A1        :OSFSC 01 - *OPT command
89B6        :"Bad option" error routine
89BB        :*OPT 4 - set autoboot option
89EA        :ECONET OSGBPB entry point
8AAD        :OSGBPB 04-08
8AFB        :read data from appropriate processor, open file for read access
8AFF        :read data from host processor
8903        :OSGBPB 00-03
8810        :read data from second processor
8B8A        :claim TUBE for filing system 3 (RFS)
```

```
8B92        :continue BOOT routine
```

8BD6-8BF1 - ECONET command and command entry point lookup table

```
8BD6        :(8079) - 'I.'
8BDA        :(8D06) - 'I AM'
8BE0        :(8BF2) - 'EX '
8BE5        :(8BF2) - 'EX'
8BEA        :(8349) - 'BYE'
8BF0        :(8079)
```

8BF2-8DAE - catalogue routines

```
8BF2        :*EX command
8BFD        :OSFSC 05 - *CAT
8C20        :PRINT catalogue display
8CE7-8CF6   :"L.!BOOT <CR> E. !BOOT <CR>" embedded message
8D02-8D05   :startup option message address lo lookup table
8D06        :*I AM command
8D3A-8D4B   :startup option embedded messages
8D4B-8D4E   :startup message offset
8D4F        :PRINT ten characters from catalogue
8D51        :PRINT Y characters from catalogue
8D5C        :PRINT X spaces
8D63        :zero index (X) ....
8D65        :zero index (Y) ....
8D67        :copy filename to catalogue
8273        :zero index (X) and ....
8D75        :PRINT legal characters from catalogue
8D84        :set pointers to execution address and start program
```

8DAF-9006 - ECONET-specific OSBYTE ans OSWORD calls

```
8DAF        :OSBYTE 32 - poll TRANSMIT block
8DC9        :OSBYTE 33 - poll RECEIVE block
8DDF        :OSBYTE 34 - delete RECEIVE block
8DF2        :OSBYTE 35 - sever remote connection
8DF7        :paged ROM service call 08 - process unrecognised OSWORD call
8E18-8E1C   :ECONET specific OSWORD routine address lo lookup table
8E1D-8E21   :ECONET specific OSWORD routine address hi lookup table
8E22        :read/write parameters from/to control block
8E33        :OSWORD 10 - call TRANSMIT
8E53        :OSWORD 12 - read remote arguments
8E7B        :OSWORD 13 - read/set miscellaneous NET information
8E83        :OSWORD 13 (0-3) - read/write file/printer server
8E89        :OSWORD 13 (2,3) - read/write printer server number
8E9B        :OSWORD 13 (4,5) - read/write protection mask
8EAF        :OSWORD 13 (8) - read local station ID
8EC6        :OSWORD 13 (7) - write user environment
```

```
8EDB        :OSWORD 13 (A) - read error number
8EE3        :OSWORD 13 (6) - read user environment
8EF0        :OSWORD 11 - open/read RECEIVE block
8F72        :OSWORD 14 - call file server/remote machine
```

9007-90FB - error processing

```
9007        :set up error processing routine
9019        :restore registers and status
9020        :push return address on to stack and go there to continue
902B-9033   :error routine continuation address lo
9034-903C   :error routine continuation address hi
90B5        :check against value from lookup table
```

90FC-9306 - language entry calls

```
90FC        :language entry call 1
912A        :language entry call 3
913A        :language entry call 4
914A        :language entry call 0
9291        :language entry call 2
92D6        :set protection mask
```

9307-974B - TUBE BRK and communications routines

```
9304-9306   :OSBYTE call number lookup table
9348-934B   :default second processor relocation address
934C-974B   :TUBE communications source code (to relocate to Pages 4-6)
```

9663-96F5 - ADLC set up

```
9663        :go to NFS initialise routine
966F        :initialise NFS
9681        :install NMI routines, save NFS variables, enable ADLC interrupt
969D        :paged ROM service call 0C - claim NMI space
96B4        :paged ROM service call 0B - release NMI space
96CD        :relocate NMI routines to Page D
96DC        :initialise ADLC
96EB        :set ADLC control registers
96EB        :reset Tx, enable Rx interrupts, flush ADLC
```

95F6-9FCA - network handshaking routines

```
96F6        :NMI continuation routine 00
9715        :NMI continuation routine 01
9737        :return to standby or quit
9744        :go to return to standby
9747        :NMI continuation routine 02
982D        :NMI continuation routine 03 - reset Tx, Rx interrupt enable
```

```
9839    :NMI continuation routine 04 - if address present, check against
        :station ID
984F    :NMI continuation routine 05
9865    :NMI continuation routine 06
98F7    :NMI continuation routine 07
99BB    :NMI continuation routine 08
9A40    :return to standby state
9A43    :reset NMI continuation address to default, quit interrupt
9B2F    :NMI continuation routine 09
9B52    :process unrecognised interrupt
9BA4    :initialise network error EVENT
9D4C    :NMI continuation routine 0A
9EA4    :if frame not complete, transmit 2 bytes
```

9FCB-9FD8 - NMI routines (relocated to 0D00)

Page 0

```
0000-0001 :
0010-0015 :
0016-0060 :TUBE BRK handling routines
0057-005A :second processor relocation address
009C-00A6 :
00B0-00B8 :
00BB-00C1 :
00C3-00CF :
00EF      :last OSBYTE/OSWORD call number
00F0-00F1 :last OSBYTE/OSWORD entry parameters
00F2-00F3 :OS command pointer
00F4      :current paged ROM
00F6-00F7 :OS workspace pointer
```

Pages 4-6

```
0400-06FF :TUBE communication routines
```

Page D

```
0D00-0D0D :NMI entry routines
0D0E-0D1F :NMI exit routines
0D20-0DFF :NFS parameter and flag store
0D22      :station ID number
0D52      :TUBE active flag
```

Pages E-F

```
0E00-0FFF :NFS parameter store
0E00      :file server ID
0E01      :file server network ID
0E05      :autoboot option
0E06      :messages option
0E09      :error number
0F09-0F0C :execution address
0F12      :owner flag
0F13      :directory cycle #
0F16-0F1F :current directory
0F20-0F29 :current library
```

JMP/JSR :calling location
address :address

```
0036      :04EC 053A
003A      :05AE 05D5 0623 0638 06A0 06CD
(00B0)    :8552
(00C4)    :8418
0100      :8441 9137
(FSCV)    :81CE
0406      :04BC 04E4 8B1D 8B2F 8B8C 8DA9 99F0 9A3D 9F98 9FA0
0414      :810E
0473      :0400
04E0      :04C3
04F7      :04F3 0543 0547 0550 0569 0580 058C 0592 059B 05B5 05DA
          :05EB 0604 060C 0626 062A 063B 063F 0643 065D 06A5
(0500)    :0054
0522      :0532
055F      :0558
05B1      :056F 05C5 05E2
05CB      :04AB 054D 0589
05CD      :0461 0566 057D 06B8
06D0      :0020 0026 002C 04E9 051F 0562 0579 05A1 05A8 05F3 05FA
          :0616 061D 0650 0684 06BF 06C5
06D9      :0018 042A 0432 0448
06E2      :0403
06F7      :06EB 06EF 06F3
0D0E      :9712 9734 9836 984C 9862 9880 9887 998F 99B2 9A47 9D91
          :9DAF 9DC5 9DDB 9E28 9E46 9E4D 9E8B 9EA1 9EE6 9EFC 9F12
0D14      :976E 98CB 992D 9D6F 9E7A 9EDA
(0D58)    :9BA1
(0F09)    :8DAC
8000      :9BB8
8079      :8D1C
8099      :8000
809F      :8139
80AF      :8003
8146      :8DF2
815C      :911B
819B      :8174 817F
81CC      :8184 81D1
822E      :818A
82D1      :8222
82E5      :8229
830E      :8381
8310      :880E
831C      :8310 8370 83B9
8340      :8A3B
8346      :86D7 8780
8350      :807D 8834 88AA 88FC 8923 8996 89C0 8A9A 8B50 8C18 8C4F 8CC8
```

```
8351      :8892 896C
836A      :8AF3
8380      :8790 8A77
83A3      :8486
8428      :868D
842A      :89B8
8448      :8730 8FBB 927D
844A      :8386 881A
847A      :845E 8674
84A4      :8760
84A5      :8A4D
84AA      :874F
84AB      :885C 8A55
8508      :808C 8694 88E1 8949 89EA 8B92
8513      :886E 8899
851D      :879B 88B6
853B      :81BC 81E6 8205 8212 8C20 8C2A 8C38 8C43 8CFD 8C6E 8C81
          :8C95 8CA2
8555      :8C0C 8D06
8560      :8D0D 8D13
8588      :83ED 8A05 8ECC
8589      :83AC 8822 88EC
858A      :894D
85A5      :8980 8EE6
85AF      :81FB 8C27
85BC      :85B2 85B7
85CE      :8716 87C9
85DF      :8496 883D 8A81
85E4      :849D 8929 8975 899C 8A84
85EB      :8639 8C6B
85F6      :85F0
8600      :8703 8783
862A      :8624
8635      :8621 8621
8640      :8617 8D5C
8644      :837C 8809
864C      :9001 905B 925B
864E      :8FA4
8650      :8FE6
86D0      :8D90
8716      :8706 8A70
87AD      :86E0 86EB
87BA      :86E3 86E8
87C8      :8789 8A6B
8844      :8743
892C      :86C8 87AA 89F5 8CFF
8945      :8A95 8B32
89CA      :8A89
89CF      :8B79
```

```
89D1      :89CC
89D2      :8A8F 8B85
89E3      :89DD
8AAD      :8A00
8AFB      :8B6E
8B92      :8242
8B8A      :8B10 8DA0
8D4F      :8C1D 8C8D 8C9F
8D51      :8C55
8D5C      :8C30 8C5A 8C92
8D63      :8079 86CB 888A
8D65      :8779 8883 88A5 8964
8D67      :8C13 8CC3
8D75      :8CD2
8DB7      :82B6 8DCB 8DE1 8F01 8F1A
8E22      :8E46 8E62 8E96 8F31
8F48      :8E4C 9004
8F57      :8F8A
8F68      :8F5D
9020      :9016
904B      :8159 9092 90F3
90B5      :906D 9076
9159      :8FE2 9FF5 90F8
9162      :911E
9166      :922B
91EC      :91E0 91F6
9217      :91E7 920B
9248      :839B 83D3
92D6      :8E65 9102 9134 9151
92DD      :92CE
92EE      :92EB
92F0      :92E3
9660      :8667 8E49
9663      :82C9
966F      :9663
9681      :96CA
969D      :9666
96B4      :9669
96CD      :9681
96DC      :9672 973E 9894
96EB      :9A40
96F6      :0D0B
9797      :9819
97FB      :9AC5
981C      :9ABA
9870      :9F2F
988A      :9797 9930 99B8 9A8C
995E      :98F4
9974      :982A 9B0F
```

```
9A19        :9A04 9A56
9A34        :9897 9B4F 9E93 9F48
9A40        :9744
9A43        :9728 9741
9A4A        :97F8
9A59        :9794
9AF9        :9AD8
9B12        :99C0
9B4F        :9A7D
9B52        :966C
9BDD        :9BAF 9BBB
9BE4        :9660
9C84        :9BF9
9C88        :9C81
9D45        :9D08 9D13
9E3B        :99B5
9EDD        :9DA8
9F39        :9963 9D9E
9F3F        :9891 9D85 9DE0 9E98
9F5B        :980E 9AEE 9D05 9D10 9D42
OSEVEN      :9BAC
GSINIT      :86A5
GSREAD      :86A8
NVRDCH      :0042 052F 055B
OSFIND      :0573 0586
OSGBPB      :0612
OSBPUT      :054A
OSBGET      :0554
OSARGS      :059E
OSFILE      :05EE
OSASCI      :854D 85CB 8611 8C7B 8D54 8D7E
OSNEWL      :8627
OSWORD      :067F 06B1 92B0
OSBYTE      :04B4 062D 0646 80C1 8154 8164 81D6 81E3 8232 8237 8291
            :82D7 830B 834B 83E9 8480 898A 9127 9156 91DA 9EFB 967B
OSCLI       :05C8 83F8 8D37
```

L/U table address	calling location address
8000	:048B
8001	:0490
8002	:0495
8003	:049A
8004	:049D
8008	:81A3 81AC
8014	:842D
8020	:80A8
8044	:80A4
824D	:8219
8240	:82E5 82EB
8310	:8F78
8334	:831F
84AF	:8435
8530	:852A
8BD6	:8B9B 8BA8
8BD7	:8BB9
8D02	:8D32
8D3A	:8C76
8D4B	:8C73
8E18	:8E06
8E1D	:8E02
8E79	:8E8D
902B	:9024
9034	:9020
90BE	:90B5
918E	:9167
9304	:92F6
9307	:8113
934C	:80F9
944C	:80FF
954C	:8105
9A0E	:9A87
9A16	:9A83
9B0E	:9B8C
9B13	:9B88
9C53	:9CCF
9C5B	:9CCB
9ECA	:9CC5
9ED2	:9CBF
9FCA	:96D1

6502 second processor OS 1.1 - summary and description of contents

ROM type :operating system

Binary version number :N/A

Copyright offset :N/A

Title string :Acorn TUBE 6502 64K

Assembly address :F800-FFFF

Relocation address :F800-FFFF

Public workspace :N/A

Private workspace :N/A

6502 second processor OS 1.1 - Memory Map

F800-F95C :initialisation routines
F95D-FCE4 :MOS call handler
FCE5-FD64 :IRQ, EVENT and ESCAPE handlers for base processor
FD65-FE94 :NMI routines
FEF8-FEFF :memory-mapped I/O - TUBE FIFO buffers
FF80-FFB7 :default values of MOS vectors
FFB9-FFF9 :MOS call entry points
FFFA-FFFF :6502 vectors

6502 second processor OS 1.1 - Zero Page usage

```
    0 1 2 3 4 5 6 7 8 9 A B C D E F
00  . . . . . . . . . . . . . . . .
10  . . . . . . . . . . . . . . . .
20  . . . . . . . . . . . . . . . .
30  . . . . . . . . . . . . . . . .
40  . . . . . . . . . . . . . . . .
50  . . . . . . . . . . . . . . . .
60  . . . . . . . . . . . . . . . .
70  . . . . . . . . . . . . . . . .
80  . . . . . . . . . . . . . . . .
90  . . . . . . . . . . . . . . . .
A0  . . . . . . . . . . . . . . . .
B0  . . . . . . . . . . . . . . . .
C0  . . . . . . . . . . . . . . . .
D0  . . . . . . . . . . . . . . . .
E0  . . . . . . . . . . . . . . * *
F0  * * * * * * * * * * * * * * * *
    0 1 2 3 4 5 6 7 8 9 A B C D E F
```

Each peripheral device attached to the BBC microcomputer makes some demands on the available memory. This may be as little as 32 (decimal) bytes for the keyboard buffer or as much as 20 Kbytes for a high resolution mode screen store. With the addition of each paged ROM, a page or two of RAM is required for private workspace. The incremental effect of all of these requirements is that there is a limited amount of space available for user programs.

Access to peripherals takes time. Controller chips have to be initialised; data have to be presented in a defined format. Programs with a large amount of I/O therefore run slowly.

The solution devised by the designers of the BBC microcomputer to overcome these limitations, was to separate the data processing and I/O functions and to provide separate processors for each task. The basic computer provides the I/O facilities and a

second processor is devoted to pure computing tasks.

Communication between the two processors takes place through the TUBE, a bidirectional, asynchronous data bus clocked at 2 MHz. The TUBE actually consists of four bidirectional FIFO buffers located at FEE0-FEE7 in the I/O processor and FEF8-FEFF in the 6502 second processor. Each buffer is accessed at two locations - a control/status register and a data transfer register. Bit 6 of the status register indicates whether there is space available for a write operation to the corresponding data transfer register and bit 7 indicates whether there are data present in the FIFO buffer waiting to be read.

6502 second processor OS 1.1

Since all of the peripherals' control and housekeeping functions are performed by the base processor, the 6502 second processor possesses only a very rudimentary operating system, which is required only for initialisation on RESET and to handle communications with the TUBE. (Although the second processor is, in fact a 65C02, the operating system does not make use of any of the additional opcodes of its enhanced instruction set.)

The 6502 second processor is provided with a full 64K complement of RAM. On RESET, the first action is to copy the operating system from ROM and load it into RAM at F800-FFFF. MOS call entry points are the same as for the base processor and there is a corresponding set of indirection vectors at 0200-0235. Not all the operating system calls are fully implemented. For example, filing system control is carried out by the base processor, so FSCV is not required and points to a "Bad" error message; the default setting of EVNTV, and the user set vectors point to an RTS opcode. Operating system calls are implemented by transferring the call and its parameters to the base processor which performs the desired operation and sends a response back via the TUBE. To speed matters up, only the minimum required number of parameters is transferred. For instance, with OSBYTE calls 0-7F, the X-parameter only accompanies the call. For those calls in which the carry status is a significant part of the result, it is transferred across the TUBE by performing a shift operation in the source processor and a complementary shift operation to prime the carry flag in the destination processor.

Data transfers are achieved by generating interrupts in the second processor. Different routines are provided for different operations, the appropriate one being selected by resetting the 6502 NMI vector (which is feasible, since after RESET, all read operations are directed to RAM).

Operating system usage of Page 0, in general, differs from that of the base processor MOS although 00FC-00FF have the same function.

F800-F95C - initialisation routines

```
F800        :RESET entry point
F859        :default error routine (relocated to 0100) - enter IMMEDIATE mode
            :test FIFO1, PRINT * as prompt, await input
F860        :PRINT "Acorn TUBE 6502 64K", ....
F863        :"<LF> Acorn TUBE 6502 64K <LF><LF><CR>" embedded message
F88D        :PRINT * as prompt, and await input (IMMEDIATE mode)
F8A7        :ESCAPE routine
F8AC        :"Escape" error message - error code 11
F8B5        :investigate ROM type
F909        :"This is not a language" error routine - error code 00
F92C        :"I cannot run this code" error routine - error code 00
F945        :default BRK routine
```

F95D-FCE4 - MOS call handler

```
F95D-F961 :OSWORD 0 parameter block
F962        :OSWRCH routine - write character to FIFO1
F96C        :OSRDCH routine - read character and status register via FIFO2
F971        :read two characters from FIFO2, recovering carry status
F975        :when ready, read character from FIFO2
F97E        :flush command line to non-space character
F97F        :get non-space character from command line
F986        :calculate execution address pointers from ASCII input
F9A4        :shift input character into workspace
F9B2        :set up pointers, transfer filename to FIFO2
F9B6        :transfer string to FIFO2
F9CA        :OSCLI routine
F9CF        :check for *GO, *HELP
FA17        :process *HELP
FA1A        :"<LF><CR> 6502 TUBE 1.10 <LF><CR>" embedded message
FA2D        :check for filing system release - FIFO2 call 02
FA3E        :confirm *GO command and execute it
FA47        :*GO command
FA5C        :check ROM type
FA71        :OSBYTE 8E - enter language ROM
FA73        :OSBYTE routine
FA77        :OSBYTE 0-7F (X-parameter only transferred) - FIFO2 call 04
FA9C        :OSBYTE 80+ entry point
FAA8        :OSBYTE 85+ (X- and Y-parameters transferred) - FIFO2 call 06
FAF0        :OSBYTE 84 - read HIMEM
FAF5        :OSBYTE 85 - read bottom of display RAM for MODE (returns 8000)
FAFA        :OSBYTE 82 - read machine higher order address (returns 0000)
FAFF        :OSWORD 1+ - FIFO2 call 08
FB2D        :transfer OSWORD parameters to base processor
FB59        :read OSWORD return parameters across TUBE
FB77        :OSWORD 0 (read line from input) - FIFO2 call 0A
FBCC        :OSARGS - FIFO2 call 0C
```

```
FC0C        :OSFIND - FIFO2 call 12
FC1A        :close file
FC24        :open file
FC2A        :OSBGET - FIFO2 call 0E
FC36        :OSBPUT - FIFO2 call 10
FC4A        :when FIFO2 ready, write character to it
FC53        :OSFILE - FIFO2 call 14
FC8E        :OSGBPB - FIFO2 call 16
FCB7        :"Bad" error routine - error code 00
            :       IRQ2 default setting
            :       OSRDRM default setting
            :       OSVDU default setting
            :       OSEVEN default setting
            :       GSINIT default setting
            :       GSREAD default setting
            :       FSCV default setting
            :       UPTV default setting
            :       NETV default setting
            :       VDUV default setting
            :       KEYV default setting
            :       INSV default setting
            :       REMV default setting
            :        CNPV default setting
FCBC-FCD0   :OSWORD input parameter counter lookup table
FCD0-FCE4   :OSWORD return parameter counter lookup table
```

FCE5-FD64 - IRQ EVENT and ESCAPE handlers for base processor

```
FCE5        :test for BRK, process it, else process IRQ1
FCF0        :process trans-TUBE IRQ and EVENTs, then IRQ2
FCFD        :set up Page 0 BRK pointers, execute BRK routine
FD18        :if ESCAPE flag clear, process EVENT
FD36        :go to EVENT processing routine
FD39        :reset ESCAPE flag
FD3F        :IRQ1 - read interrupt routine via FIFO2
FD4D        :set up trans-TUBE BRK routine in Page 2
```

FD65-FE94 - NMI routines

```
FD65        :read index across TUBE, set up NMI routine
FDCF        :transfer page via FIFO3
FDEC        :read page from FIFO3
FE00        :NMI routine 0 - transfer byte from memory to FIFO3
FE11        :NMI routine 1 - transfer byte from FIFO3 to memory
FE22        :NMI routine 2 - write 2 bytes to FIFO3, preserving A-,Y-registers
FE41        :NMI routine 3 - read 2 bytes from FIFO3, preserving A-,Y-registers
FE60- FE67  :byte transfer base address lo lookup table
FE68-FE6F   :byte transfer base address hi lookup table
FE70-FE77   :NMI routine address lo lookup table
```

```
FE78-FE7F :NMI routine address hi lookup table
FE80      :read data from FIFO1, checking for NMI clearance signal on FIFO4
FE98      :set up pointers and display message following
FEA0      :PRINT next character if hi bit not set
FEB3      :NMI routines 4-7 - write to FIFO3
```

FEF8-FEFF - memory-mapped I/O - TUBE FIFO buffers

```
FEF8      :FIFO1 status register
FEF9      :FIFO1 data register
FEFA      :FIFO2 status register
FEFB      :FIFO2 data register
FEFC      :FIFO3 status register
FEFD      :FIFO3 data register
FEFE      :FIFO4 status register
FEFF      :FIFO4 data register
```

FF80-FFB7 - default values of MOS vectors

```
FF80      :(FCB7) USERV
FF82      :(F945) BRKV
FF84      :(FCF0) IRQ1V
FF86      :(FCB7) IRQ2V ("Bad" error routine)
FF88      :(F9CA) CLIV
FF8A      :(FA73) BYTEV
FF8C      :(FAFF) WORDV
FF8E      :(F962) WRCHV
FF90      :(F96C) RDCHV
FF92      :(FC53) FILEV
FF94      :(FBCC) ARGSV
FF96      :(FC2A) BGETV
FF98      :(FC36) BPUTV
FF9A      :(FC8E) GBPBV
FF9C      :(FC0C) FINDV
FF9E      :(FCB7) FSCV ("Bad" error routine)
FFA0      :(F97D) EVNTV (RTS)
FFA2      :(FCB7) UPTV ("Bad" error routine)
FFA4      :(FCB7) NETV ("Bad" error routine)
FFA6      :(FCB7) VDUV ("Bad" error routine)
FFA8      :(FCB7) KEYV ("Bad" error routine)
FFAA      :(FCB7) INSV ("Bad" error routine)
FFAC      :(FCB7) REMV ("Bad" error routine)
FFAE      :(FCB7) CNPV ("Bad" error routine)
FFB0      :(F97D) IND1V (RTS)
FFB2      :(F97D) IND2V (RTS)
FFB4      :(F97D) IND3V (RTS)
FFB6      :(8036)
```

FFB9-FFF9 - MOS call entry points

```
FFB9        :OSRDRM
FFBC        :OSDU
FFBF        :OSEVEN
FFC2        :GSINIT
FFC5        :GSREAD
FFC8        :NVRDCH
FFCB        :NVWRCH
FFCE        :OSFIND
FFD1        :OSGBPB
FFD4        :OSBPUT
FFD7        :OSBGET
FFDA        :OSARGS
FFDD        :OSFILE
FFE0        :OSRDCH
FFE3        :OSASCI
FFE7        :OSNEWL
FFEE        :OSWRCH
FFF1        :OSWORD
FFF4        :OSBYTE
FFF7        :OSCLI
```

FFFA-FFFF - 6502 vectors

```
FFFA        :(FE00) NMI vector - dynamically reset
FFFC        :(F800) RESET vector
FFFE        :(00E5) IRQ vector
```

Page 0

002A-002D :OSARGS parameter block (00,X-03,X)
00EE-00EF :current language entry point
00F0-00F1 :execution address workspace
00F2-00F3 :HIMEM
00F4-004F :byte transfer pointers
00F6-00F7 :MOS text pointers
00F8-00F9 :general purpose pointers
:MOS ROM relocation pointers
:OSCLI command line pointers
:filename pointers
:OSWORD parameter block pointers
00FA-00FB :OSFILE parameter block pointers
:text pointers
00FC :pre-interrupt A-register setting
00FD-00FE :last BRK pointers
00FF :ESCAPE flag

Page 1

0100-01FF :6502 stack
0100- :error routine handler

Page 2

0200-0235 :MOS vectors
0236-02FF :trans-TUBE data input buffer

JMP/JSR address : calling location address

```
(00F2)     :F8FC
(00FA)     :FEB0
0100       :F856
(BRKV)     :FD15
(IRQ1V)    :FCED
(IRQ2V)    :FCFA
(CLIV)     :FFF7
(BYTEV)    :FFF4
(WORDV)    :FFF1
(WRCHV)    :FFEE
(RDCHV)    :FFE0
(FILEV)    :FFDD
(ARGSV)    :FFDA
(BGETV)    :FFD7
(BPUTV)    :FFD4
(GBPBV)    :FFD1
(FINDV)    :FFCE
(EVENTV)   :FD36
0236       :FD62
F88D       :F85D F8A4 F95A
F8B5       :FA62
F962       :FFCB
F96C       :FFC8
F971       :FC33 FCB4
F975       :F886 F971 FA35 FBA3 FBF2 FBF6 FBFB FC00 FC05 FC1F FC27
           :FC45 FC78 FC7E FCA8 FD53 FD5A
F97E       :FA44
F97F       :F9D1 FA4A
F986       :FA47
F9B2       :FC24 FC71
F9B6       :FA32
FB2D       :FB21
FB59       :FB4D
FC4A       :F96E FA2F FB79 FB8F FBCF FBDC FBE1 FBE6 FBEB FBEF FC0F
           :FC13 FC1C FC2C FC30 FC39 FC3D FC41 FC5A FC61 FC75 FC95
           :FC9C FCA3
FCB7       :FFB9 FFBC FFBF FFC2 FFC5
FD36       :FD2C
FE80       :FD21 FD25 FD29 FE91
FE98       :F860 FA17
FEA0       :FEAD
OSCLI      :F8A1
OSBYTE     :F8A9
OSWRCH     :F88F F951 FEAA FFE9
OSWORD     :F898
OSNEWL     :F948 F957
```

L/U :calling location
address :address

```
FF00        :F802
FF80        :F80D
FDFF        :F819
F859        :F83B
FCBC        :FB24
FCD0        :FB50
FE70        :FD6D
FE78        :FD73
FE60        :FD79
FE68        :FD7E
FFFF        :FDD6
```

Base processor communications routines

When the second processor is active, the base processor has to provide communications routines to receive and process MOS calls and to transfer languages from ROM or the current filing system across the TUBE. These routines are provided either by NFS 3.34 or the DNFS version of Econet and are relocated to Pages 4- 6. A block of Page 0 locations is set up to hold the routines which control the second processor interrupt handlers.

The communications routines function in a complementary manner to the corresponding routines in the second processor operating system. They receive the calls and parameters over the TUBE and pass them to the base processor Operating System which handles them in the same way as if they had actually originated in the base processor. The result is returned to the TUBE interface.

Like paged ROMs, the communications routines have language and service entry points, at 0400 and 0403 respectively. The base processor uses Page 7 and the second processor the upper part of Page 2 (0236-02FF) as a data transfer buffer.

Page 0

```
0016          :set up second processor interrupt, transfer data block via FIFO2
0032          :initialise stack, set up pointers and execute vectored routine
0036          :set up pointers and execute vectored routine
0053-0056     :second processor relocation address
0057          :go to language entry point
005A          :go to service entry point
```

Pages 4-6

```
0400          :language entry point
0403          :service entry point
0406          :TUBE data transfer routines
0414          : release TUBE from filing system
0421          :set current TUBE ownership and access flags
0428          : claim TUBE for filing system
0435          : set up data transfer across TUBE
0484          :TUBE language entry
04CB          :set up relocation address pointers, go to TUBE data transfer routine
04D2          :set up second processor relocation address pointers
0500-0517     :OS call entry point lookup table
0500          :(0537) - OSRDCH
0502          :(0596) - OSCLI
0504          :(05F2) - OSBYTE 0-7F (with X-parameter only)
0506          :(0607) - OSBYTE 80+ (with X- and Y-parameters)
0508          :(0627) - OSWORD (calls >0)
050A          :(0668) - OSWORD 0
050C          :(055E) - OSARGS
050E          :(052D) - OSBGET
0510          :(0520) - OSBPUT
0512          :(0542) - OSFIND
0514          :(05A9) - OSFILE
0516          :(05D1) - OSGBPB
0518-051F     :TUBE FIFO1 call lookup table
0520          :OSBPUT
052D          :OSBGET
0537          :OSRDCH
053A          :write shifted byte to FIFO2, set up pointers, execute vectored
              :routine
0542          :OSFIND
055E          :OSARGS
0582          :read string from FIFO2 to Page 7
0596          :OSCLI
059C          :write 7F to FIFO2, set up pointers, execute vectored routine
059E          :set up pointers and execute vectored routine
05A9          :OSFILE
05D1          :OSGBPB
05F2          :OSBYTE 0-7F (with transfer of X-parameter only)
```

JMP/JSR address	calling location address
0032	0477
0036	057F 05A6 0604 0665 0692
0406	049A 04CF
0414	0471
0484	0400
04CB	04A4
04D2	049F 04C6
053A	0534 05EF
0582	0548 0596 05B3
059C	0489 052A 055B
059E	053F 054F 067B
0695	0474 053B 0572 0579 05C2 05C9 05E8 061A 0684 068A
069E	0418 041D 043B 0443 0448 0463
06A7	0403
06BC	06B0 06B4 06B8
06C5	0520 0524 052D 0542 0552 055E 0564 056C 0586 05AB 05BC 05D3 05DB 05F2 05F6 0607 060B 060F 0627 066A
9B6E	06DE
967A	06D1
9F57	06D4
9698	06D7
OSWORD	0649 0676
OSBYTE	0492 05F9 0612
OSBPUT	0527
OSBGET	0531
OSGBPB	05E0
OSFILE	05BF
OSFIND	054C 0558
OSRDCH	0537
OSCLI	0599
OSARGS	056F

```
0607        :OSBYTE 80+ (with transfer of X-, and Y- parameters)
0627        :OSWORD 1+
0668        :OSWORD 0 - read input line
0695        :when ready, write byte to FIFO2
069E        :when ready, write byte to FIFO3
06A7        :TUBE service entry
06AD        :transfer Y-,X- and A-registers via FIFO1
06BC        :when ready, write byte to FIFO1
06C5        :when ready, read byte from FIFO2
```

Page 0

```
0016        :set up second processor interrupt, transfer data block via FIFO2
0036        :get call from TUBE and execute it
0057-005A   :second processor reloation address
005B        :go to language entry routine
005E        :go to service entry routine
```

Pages 4-6

```
0400        :language entry point
0403        :service entry point
0406        :access TUBE
040E        :release TUBE from filing system
0419        :claim TUBE for filing system
0423        :save current TUBE access status
0426        :set up data transfer
045D        :set TUBE ownership flag, signal via FIFO2, read from second
            :processor
0464        :when FIFO4 ready, flag FIFO3 twice, write to FIFO1
0473        :TUBE language entry routine
04CC        :copy code to second processor memory, then execute it
04E0        :set execution address pointers, transfer data across TUBE
04E7        :signal TUBE and execute TUBE call
04EF        :read load parameters, read byte and carry status across TUBE

04F3        :read carry status and byte across TUBE
0500-051A   :OS call entry points lookup table
0500        :(055B) - OSRDCH
0502        :(05C5) - OSCLI
0504        :(0626) - OSBYTE 0-7F (with X-paramater only)
0506        :(063B) - OSBYTE 80+ (with X- and Y- parameters)
0508        :(065D) - OSWORD 1+
050A        :(06A3) - OSWORD 0
050C        :(04EF) - load parameters, read call and carry status across TUBE
050E        :(053D) - save parameters, recover call from stack
0510        :(058C) - OSARGS
0512        :(0550) - OSBGET
0514        :(0543) - OSBPUT
0516        :(0569) - OSFIND
0518        :(05D8) - OSFILE
051A        :(0602) - OSGBPB
051C        :OSWRCH interface routine
0522        :if ready, execute TUBE call, else PRINT message from second
            :processor
0535        :signal second processor and execute TUBE call
053D        :recover parameters and call
0543        :OSBPUT interface routines
0550        :OSBGET interface routines
```

```
055F    :OSRDCH interface routines - signal second processor via FIFO2 and
        :read from TUBE
0569    :OSFIND interface routines
0580    :OSFIND 0
058C    :OSARGS interface routines
05B1    :read string from TUBE to Page 7 buffer
05C5    :OSCLI interface routine
05CD    :write parameter to FIFO2 and read byte from second processor
05D8    :OSFILE interface routines
0602    :OSGBPB interface routines
0626    :OSBYTE 0-7F interface routines
063B    :OSBYTE 80+ interface routines
065D    :OSWORD 1+ interface routines
06A3    :OSWORD 0 interface routines
06BB    :write string from Page 7 to FIFO2
06D0    :when ready, write byte to FIFO2
06D9    :when ready, write byte to FIFO4
06E2    :write FF to FIFO1
06E8    :EVENT handler
06F7    :when ready, write byte to FIFO1
```

JMP/JSR :calling location
address :address

```
0036        :04EC 053A
003A        :05AE 05D5 0623 0638 06A0 06CD
0406        :04BC 04E4
0473        :0400
04E0        :04C3
04F7        :04F3 0543 0547 0550 0569 0580 058C 0592 059B 05B5 05DA 05EB
            :0604 060C 0626 062A 063B 063F 0643 065D 06A5
0522        :0532
055F        :0558
05B1        :056F 05C5 05E2
05CB        :04AB 054D 0589
05CD        :0461 0566 057D 06B8
06D0        :04E9 051F 0562 0579 05A1 05A8 05F3 05FA 0616 061D 0650 0684
            :06BF 06C5
06D9        :042A 0432 0448
06E2        :0403
06F7        :06EB 06EF 06F3
NVRDCH      :055B
NVWRCH      :052F
OSFIND      :0573 0586
OSGBPB      :0612
OSBPUT      :054A
OSBGET      :0554
OSARGS      :059E
OSFILE      :05EE
OSWORD      :067F 06B1
OSBYTE      :04B4 062D 0646
OSCLI       :05C8
```

Within the
BBC
MICROCOMPUTER

Name _____

Address _____

_____ Post code _____

With a fast moving field it is inevitable that there will be developments leading to the introduction of changes in the manufactured product. In order to maintain the maximum utility of this book, it is the intention of the Publisher to bring out further editions from time to time. Owners of previous editions may purchase an updated version at a privilege price by completing and returning this registration card.

It will be helpful to the author to receive any suggestions for alterations or additions to be included in future editions.

I would like to see the following alterations or additions to future editions:-

Please
affix
stamp

Losco Limited,
P.O. Box 4,
Cranleigh,
Surrey, GU6 8BQ.

Within the BBC MICROCOMPUTER

A reference manual for Assembly Language programmers

This book contains essential reference material for serious users of the BBC microcomputer. It includes descriptions and explanations of the principal ROM routines, memory maps, tables of RAM usage, ROM routine entry points, zero page locations and JMP/JSR and lookup reference origins. It covers OS 1.2, Basic 1, Basic 2, HiBasic, DFS 0.90, NFS 3.34, 6502 second processor OS 1.1 and DNFS and Econet TUBE communications.

ISBN 0 948203 00 5

Losco Limited
PO Box 4, Cranleigh, Surrey GU6 8BQ,
England