# A Pocket Handbook for the COMMODORE 64






Peter Gerrard & Danny Doyle

# A Pocket Handbook for the Commodore 64

# A Pocket Handbook for the Commodore 64

Peter Gerrard
&
Danny Doyle



Duckworth

Second impression April 1984
First published in December 1983 by
Gerald Duckworth & Co. Ltd.
The Old Piano Factory
43 Gloucester Crescent, London NW1

© 1983 by Peter Gerrard

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 permission of the publisher.

ISBN 0 7156 1787 7

British Library Cataloguing in Publication Data
Gerrard, Peter
A pocket handbook for the Commodore 64.
1. Commodore 64 (Computer)
I. Title
001.64' QA76.8.C64

ISBN 0-7156-1787-7

Printed in Great Britain by
Redwood Burn Ltd., Trowbridge
and bound by Pegasus Bookbinding, Melksham

# Contents


| | |
|---|---|
| Preface | 6 |
| ASCII tables | 7 |
| Basic keywords | 9 |
| Basic error messages | 12 |
| Colour memory | 14 |
| Conversion tables | 15 |
| Disk commands | 19 |
| Disk error messages | 21 |
| Disk formats | 26 |
| Extramon listing | 35 |
| Flow charting | 40 |
| Hex/Dec convertor | 42 |
| Hyperbolic functions | 43 |
| Memory maps | 44 |
| Memory architecture | 51 |
| M/C instruction set | 54 |
| M/C mnemonics | 74 |
| Powers tables | 76 |
| Cartridge slot | 77 |
| Joystick slot | 78 |
| RS232 standards | 79 |
| Centronics standards | 86 |
| Other output | 87 |
| Screen memory | 89 |
| Sound chip registers | 90 |
| Musical notes values | 91 |
| Sprite memory diagram | 93 |
| Index | 95 |

# Preface

This book is a collection of relevant facts and figures about your Commodore 64 computer.

As well as including full memory maps, microprocessor instruction set, computer glossary, interface tables, disk formats, Basic commands, input/output tables, memory architecture, and much more, this conveniently sized book tells you, at a glance, any important fact you need to know when programming and using your computer.

We've tried to include as much relevant material as possible in as small a space as possible. As a result, the majority of this book is simply facts and figures: one day we might write a book with words in it!

We'd like to thank Jim Butterfield (without whom...) for getting the whole Commodore scene rolling in his own inimitable style, anyone who's given us any help, advice and information in compiling this book, the New Inn in Gawcott for providing excellent pints of Marston ales in times of stress, and finally our wives, Penny and Beryl, for putting up with us.

P.G. and D.D.

# ASCII tables

Standard ASCII characters (7-bit code)

| LSD | MSD | 0<br>000 | 1<br>001 | 2<br>010 | 3<br>011 | 4<br>100 | 5<br>101 | 6<br>110 | 7<br>111 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0000 | NUL | DLE | SP | 0 | @ | P | - | p |
| 1 | 0001 | SOH | DC1 | ! | 1 | A | Q | a | q |
| 2 | 0010 | STX | DC2 | " | 2 | B | R | b | r |
| 3 | 0011 | ETX | DC3 | £ | 3 | C | S | c | s |
| 4 | 0100 | EOT | DC4 | $ | 4 | D | T | d | t |
| 5 | 0101 | ENQ | NAK | % | 5 | E | U | e | u |
| 6 | 0110 | ACK | SYN | & | 6 | F | V | f | v |
| 7 | 0111 | BEL | ETB | ' | 7 | G | W | g | w |
| 8 | 1000 | BS | CAN | ( | 8 | H | X | h | x |
| 9 | 1001 | HT | EM | ) | 9 | I | Y | i | y |
| A | 1010 | LF | SUB | * | : | J | Z | j | z |
| B | 1011 | VT | ESC | + | ; | K | [ | k | |
| C | 1100 | FF | FS | , | < | L | | l | |
| D | 1101 | CR | GS | - | = | M | | m | |
| E | 1110 | SO | RS | . | > | N | | n | |
| F | 1111 | SI | US | / | ? | O | | o | DEL |

The ASCII symbols.

NUL - Null
SOH - Start of Heading
STX - Start of Text
ETX - End of Text
EOT - End of Transmission
ENQ - Enquiry
ACK - Acknowledge
BEL - Bell (audible alert)
BS - Backspace
HT - Horizontal Tabulation
LF - Line Feed
VT - Vertical Tabulation
FF - Form Feed
CR - Carriage Return
SO - Shift Out
SI - Shift In

DLE - Data Link Escape
DC - Device Control
NAK - Negative Acknowledge
SYN - Synchronous Idle
ETB - End of Transmission Block
CAN - Cancel
EM - End of Medium
SUB - Substitute
ESC - Escape
FS - File Separator
GS - Group Separator
RS - Record Separator
US - Unit Separator
SP - Space (Blank)
DEL - Delete

## Keyboard CNTL Sequences.

| | |
|---|---|
| NUL - CNTL 1 | DLE - CNTL P |
| SOH - CNTL A | DC1/2/3/4 - CNTL Q/R/S/T |
| STX - CNTL B | NAK - CNTL U |
| ETX - CNTL C | SYN - CNTL V |
| EOT - CNTL D | ETB - CNTL W |
| ENQ - CNTL E | CAN - CNTL X |
| ACK - CNTL F | EM - CNTL Y |
| BEL - CNTL G | SUB - CNTL Z |
| BS - CNTL H/BS | ESC - ESC |
| HT - CNTL I/TAB | FS - CNTL BACKSLASH |
| LF - CNTL J/LF | GS - CNTL ' |
| VT - CNTL K | RS - CNTL = |
| FF - CNTL L | US - CNTL - |
| CR - CNTL M/CR | SP - Space |
| SO - CNTL N | SI - CNTL O |

# Basic keywords

BASIC Commands

| Command | Format |
|---|---|
| ABS | ABS(<expression>) |
| AND | <expression> AND <expression> |
| ASC | ASC(<string>) |
| ATN | ATN(<number>) |
| CHR$ | CHR$(<number>) |
| CLOSE | CLOSE <file number> |
| CLR | CLR |
| CMD | CMD <file number> [,string] |
| CONT | CONT |
| COS | COS(<number>) |
| DATA | DATA <list of constants> |
| DEF FN | DEF FN<name>(<variable>) = <expression> |
| DIM | DIM <variable>(<subscripts>)[,<br><variable>(<subscripts>)...] |
| END | END |
| EXP | EXP(<number>) |
| FN | FN<number>(<number>) |
| FOR | FOR <variable> = <start> TO <limit> [STEP<br><increment> |
| FRE | FRE(<dummy>) |
| GET | GET <variable list> |
| GET£ | GET£ <file number>,<variable list> |
| GOSUB | GOSUB <line number> |
| GOTO | GOTO <line number> |
| IF | IF <expression> THEN <line number><br>IF <expression> GOTO <line number><br>IF <expression> THEN <statement> |
| INPUT | INPUT ["<prompt>"]; <variable list> |
| INPUT£ | INPUT£<file number>,<variable list> |
| INT | INT(<numeric>) |
| LEFT$ | LEFT$(<string>,<integer>) |
| LEN | LEN(<string>) |
| LET | LET <variable> = <expression> |
| LIST | LIST [[<first-line>]-[<last-line>]] |
| LOAD | LOAD ["<file-name>"][,<device>] |
| LOG | LOG(<numeric>) |
| MID$ | MID$(<string>,<numeric-1>[,<numeric-2>]) |
| NEW | NEW |
| NEXT | NEXT [<counter>][,<counter>]... |
| NOT | NOT <expression> |
| ON | ON <variable> GOTO/GOSUB <line-number><br>[,<line-number>]... |
| OPEN | OPEN <file-number>,[<device>][,<address>]<br>[,"<file-name>[,<type>][,<mode>]"] |
| OR | <operand> OR <operand> |
| PEEK | PEEK(<numeric>) |
| POKE | POKE <location>,<value> |

```
POS         POS(<dummy>)
PRINT       PRINT [<variable>][<,/]<variable>]...
PRINT£      PRINT£ <file-number> [,<variable> ]
                   [<,/;> <variable>]...
READ        READ <variable> [,<variable>]...
REM         REM [<remark>]
RESTORE     RESTORE
RETURN      RETURN
RIGHT$      RIGHT$(<string>,<numeric>)
RND         RND(<numeric>)
RUN         RUN [line number]
SAVE        SAVE ["<file-name>"][,<device-number>]
                 [,<address>]
SGN         SGN (<numeric>)
SIN         SIN(<numeric>)
SPC         SPC(<numeric>)
SQR         SQR(<NUMERIC>)
STATUS      ST
STOP        STOP
STR$        STR$(<numeric>)
SYS         SYS <memory location>
TAB         TAB(<numeric>)
TAN         TAN(<numeric>)
TIME        TI
TIME$       TI$
USR         USR(<numeric>)
VAL         VAL(<string>)
VERIFY      VERIFY
WAIT        WAIT <location>,<mask-1>[,<mask-2>]
```

---

[] indicates optional.
<> indicates mandatory.
() indicates brackets required.

## ARITHMETIC OPERATORS

| SYMBOL | EXAMPLE | PURPOSE |
|---|---|---|
| = | 10 X = Y | ASSIGNS VALUE TO VARIABLE. |
| / | 10 X = Y / Z | DIVISION. |
| * | 10 X = Y * Z | MULTIPLICATION. |
| + | 10 X = Y + Z | ADDITION. |
| - | 10 X = Y - Z | SUBTRACTION. |
| = | 10 IF X = Y THEN | TEST FOR EQUALITY. |
| <> | 10 IF X <> Y THEN | TEST FOR INEQUALITY. |
| < | 10 IF X < Y THEN | TEST FOR LESS THAN. |
| > | 10 IF X > Y THEN | TEST FOR GREATER THAN. |
| <= | 10 IF X <= Y THEN | TEST FOR LESS THAN OR EQUAL TO. |
| >= | 10 IF X >= Y THEN | TEST FOR GREATER THAN OR EQUAL TO |
| AND | 10 IF X AND Y THEN | LOGICAL AND. |
| OR | 10 IF X OR Y THEN | LOGICAL OR. |
| NOT | 10 IF NOT X THEN | LOGICAL NEGATION. |
| [] | 10 PRINT X[]Y | EXPONENTIATION. |

## ARITHMETIC FUNCTIONS

| FUNCTION | EXAMPLE | DESCRIPTION |
|---|---|---|
| ABS | 10 Z = ABS(X) | RETURNS MAGNITUDE OF ARGUMENT IRRESPECTIVE OF SIGN. |
| ATN | 10 Z = ATN(X) | RETURNS ARCTANGENT OF ARGUMENT. Z GIVEN I RADIANS. |
| COS | 10 Z = COS(X) | RETURNS COSINE OF ARGUMENT. X MUST BE IN RADIANS. |
| EXP | 10 Z = EXP(X) | RETURNS MATHEMATICAL CONSTANT 'E' RAISED TO POWER OF X. |
| INT | 10 Z = INT(X) | RETURNS INTEGER PART OF X. |
| LOG | 10 Z = LOG(X) | RETURNS NATURAL LOG OF ARGUMENT. X MUST BE GREATER OR EQUAL TO 0. |
| RND | 10 Z = RND(X) | RETURNS RANDOM NUMBER BETWEEN ZERO AND ONE. |
| SGN | 10 Z = SGN(X) | RETURNS MATHEMATICAL SIGN OF ARGUMENT. |
| SQR | 10 Z = SQR(X) | RETURNS SQUARE ROOT OF ARGUMENT. |
| TAN | 10 Z = TAN(X) | RETURNS TANGENT OF ARGUMENT. X MUST BE IN RADIANS. |

# Basic error messages

**BAD DATA** String data was received from an open file, but the program was expecting numeric data.

**BAD SUBSCRIPT** The program was trying to reference an element of an array whose number is outside of the range specified in the DIM statement.

**CAN'T CONTINUE** The CONT command will not work, either because the program was never RUN, there has been an error, or a line has been edited.

**DEVICE NOT PRESENT** The required I/O device was not available for an OPEN, CLOSE, CMD, PRINT#, INPUT#, or GET#.

**DIVISION BY ZERO** Division by zero is a mathematical oddity and not allowed.

**EXTRA IGNORED** Too many items of data were typed in response to an INPUT statement. Only the first few items were accepted.

**FILE NOT FOUND** If you were looking for a file on tape, and END-OF-TAPE marker was found. If you were looking on disk, no file with that name exists.

**FILE NOT OPEN** The file specified in a CLOSE, CMD, PRINT#, INPUT#, or GET#, must first be OPENed.

**FILE OPEN** An attempt was made to open a file using the number of an already open file.

**FORMULA TOO COMPLEX** The string expression being evaluated should be split into at least two parts for the system to work with.

**ILLEGAL DIRECT** The INPUT statement can only be used within a program, and not in direct mode.

**ILLEGAL QUANTITY** A number used as the argument of a function or statement is out of the allowable range.

**LOAD** There is a problem with the program on tape.

**NEXT WITHOUT FOR** This is caused by either incorrectly nesting loops or having a variable name in a NEXT statement that doesn't correspond with one in a FOR statement.

**NOT INPUT FILE** An attempt was made to INPUT or GET data from a file which was specified to be for output only.

**NOT OUTPUT FILE** An attempt was made to PRINT data to a file which was specified as input only.

**OUT OF DATA** A READ statement was executed but there is no data left unREAD in a DATA statement.

**OUT OF MEMORY** There is no more RAM available for program or variables. This may also occur when too many FOR loops have been nested, or when there are too many GOSUBs in effect.

**OVERFLOW** The result of a computation is larger than the largest number allowed, which is 1.70141884E+38.

**REDIM'D ARRAY** An array may only be DIMensioned once. If an array variable is used before that array is DIM'd, an automatic DIM operation is performed on that array setting the number of elements to ten, and any subsequent DIMs will cause this error.

**REDO FROM START** Character data was typed in during an INPUT statement when numeric data was expected. Just re-type the entry so that it is correct, and the program will continue by itself.

**RETURN WITHOUT GOSUB** A RETURN statement was encountered, and no GOSUB command has been issued.

**STRING TOO LONG** A string can contain up to 255 characters.

**?SYNTAX ERROR** A statement is unrecognizable by the Commodore 64. A missing or extra parenthesis, misspelled keywords, etc.

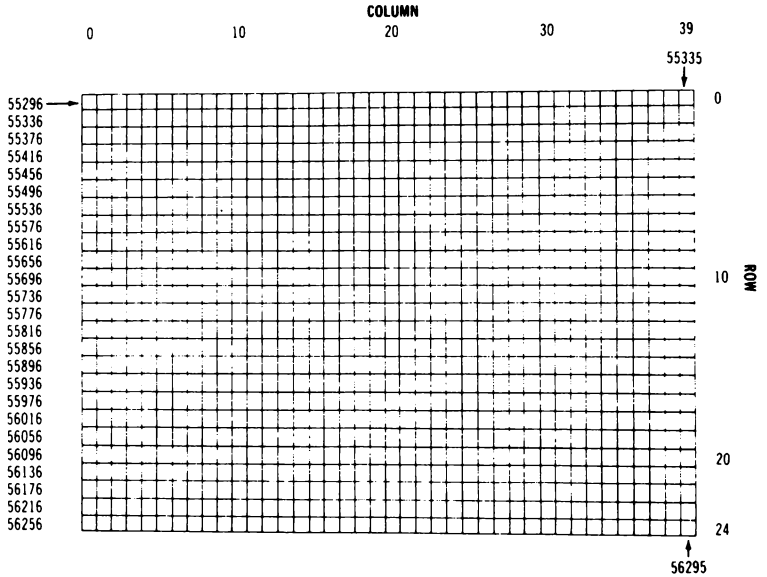
**TYPE MISMATCH** This error occurs when a number is used in place of a string, or vice-versa.

**UNDEF'D FUNCTION** A user defined function was referenced, but it has never been defined using the DEF FN statement.

**UNDEF'D STATEMENT** An attempt was made to GOTO or GOSUB or RUN a line number that doesn't exist.

**VERIFY** The program on tape or disk does not match the program currently in memory.

# Colour memory



# Conversion tables

Code Conversion Table

| Dec | Oct | Hex | 6502 Instruction | Poke 1 | Poke 2 | CHR$ | Asc | Token | Binary |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 000 | 00 | BRK | @ | @ | | NUL | | 0000 0000 |
| 1 | 001 | 01 | ORA (Ind,X) | A | a | | SOH | | 0000 0001 |
| 2 | 002 | 02 | - | B | b | | STX | | 0000 0010 |
| 3 | 003 | 03 | - | C | c | | ETX | | 0000 0011 |
| 4 | 004 | 04 | - | D | d | | EOT | | 0000 0100 |
| 5 | 005 | 05 | ORA 0-Page | E | e | | ENQ | | 0000 0101 |
| 6 | 006 | 06 | ASL 0-Page | F | f | | ACK | | 0000 0110 |
| 7 | 007 | 07 | - | G | g | | BEL | | 0000 0111 |
| 8 | 010 | 08 | PHP | H | h | | BS | | 0000 1000 |
| 9 | 011 | 09 | ORA Immediate | I | i | | HT | | 0000 1001 |
| 10 | 012 | 0A | ASL Accum. | J | j | | LF | | 0000 1010 |
| 11 | 013 | 0B | - | K | k | | VT | | 0000 1011 |
| 12 | 014 | 0C | - | L | l | | FF | | 0000 1100 |
| 13 | 015 | 0D | ORA Absolute | M | m | | CR | | 0000 1101 |
| 14 | 016 | 0E | ASL Absolute | N | n | | SO | | 0000 1110 |
| 15 | 017 | 0F | - | O | o | | SI | | 0000 1111 |
| 16 | 020 | 10 | BPL | P | p | | DLE | | 0001 0000 |
| 17 | 021 | 11 | ORA (Ind),Y | Q | q | CRSR DN | DC1 | | 0001 0001 |
| 18 | 022 | 12 | - | R | r | RVS ON | DC2 | | 0001 0010 |
| 19 | 023 | 13 | - | S | s | HOME | DC3 | | 0001 0011 |
| 20 | 024 | 14 | - | T | t | DEL | DC4 | | 0001 0100 |
| 21 | 025 | 15 | ORA 0-Page,X | U | u | | NAK | | 0001 0101 |
| 22 | 026 | 16 | ASL 0-Page,X | V | v | | SYN | | 0001 0110 |
| 23 | 027 | 17 | - | W | w | | ETB | | 0001 0111 |
| 24 | 030 | 18 | CLC | X | x | | CAN | | 0001 1000 |
| 25 | 031 | 19 | ORA Abs,Y | Y | y | | EM | | 0001 1001 |
| 26 | 032 | 1A | - | Z | z | | SUB | | 0001 1010 |
| 27 | 033 | 1B | - | [ | [ | | ESC | | 0001 1011 |
| 28 | 034 | 1C | - | | | | FS | | 0001 1100 |
| 29 | 035 | 1D | ORA Abs,X | ] | ] | | GS | | 0001 1101 |
| 30 | 036 | 1E | ASL Abs,X | | | | RS | | 0001 1110 |
| 31 | 037 | 1F | - | | | | US | | 0001 1111 |

Code Conversion Table

| Dec | Oct | Hex | 6502 Instruction | Poke U | Poke L | CHR$ U | CHR$ L | Std Asc | Token | Binary |
|---|---|---|---|---|---|---|---|---|---|---|
| 32 | 040 | 20 | JSR | SPC | SPC | SPC | SPC | SPC | | 0010 0000 |
| 33 | 041 | 21 | AND (Ind,X) | | | | | | | 0010 0001 |
| 34 | 042 | 22 | - | " | " | " | " | " | | 0010 0010 |
| 35 | 043 | 23 | - | £ | £ | £ | £ | £ | | 0010 0011 |
| 36 | 044 | 24 | BIT 0-Page | $ | $ | $ | $ | $ | | 0010 0100 |
| 37 | 045 | 25 | AND 0-Page | % | % | % | % | % | | 0010 0101 |
| 38 | 046 | 26 | ROL 0-Page | & | & | & | & | & | | 0010 0110 |
| 39 | 047 | 27 | - | ' | ' | ' | ' | ' | | 0010 0111 |
| 40 | 050 | 28 | PLP | ( | ( | ( | ( | ( | | 0010 1000 |
| 41 | 051 | 29 | AND Immediate | ) | ) | ) | ) | ) | | 0010 1001 |
| 42 | 052 | 2A | ROL Accum. | * | * | * | * | * | | 0010 1010 |
| 43 | 053 | 2B | - | + | + | + | + | + | | 0010 1011 |
| 44 | 054 | 2C | BIT Absolute | , | , | , | , | , | | 0010 1100 |
| 45 | 055 | 2D | AND Absolute | - | - | - | - | - | | 0010 1101 |
| 46 | 056 | 2E | ROL Absolute | . | . | . | . | . | | 0010 1110 |
| 47 | 057 | 2F | - | / | / | / | / | / | | 0010 1111 |
| 48 | 060 | 30 | BMI | 0 | 0 | 0 | 0 | 0 | | 0011 0000 |
| 49 | 061 | 31 | AND (Ind),Y | 1 | 1 | 1 | 1 | 1 | | 0011 0001 |
| 50 | 062 | 32 | - | 2 | 2 | 2 | 2 | 2 | | 0011 0010 |
| 51 | 063 | 33 | - | 3 | 3 | 3 | 3 | 3 | | 0011 0011 |
| 52 | 064 | 34 | - | 4 | 4 | 4 | 4 | 4 | | 0011 0100 |
| 53 | 065 | 35 | AND 0-Page,X | 5 | 5 | 5 | 5 | 5 | | 0011 0101 |
| 54 | 066 | 36 | ROL 0-Page,X | 6 | 6 | 6 | 6 | 6 | | 0011 0110 |
| 55 | 067 | 37 | - | 7 | 7 | 7 | 7 | 7 | | 0011 0111 |
| 56 | 070 | 38 | SEC | 8 | 8 | 8 | 8 | 8 | | 0011 1000 |
| 57 | 071 | 39 | AND Abs,Y | 9 | 9 | 9 | 9 | 9 | | 0011 1001 |
| 58 | 072 | 3A | - | : | : | : | : | : | | 0011 1010 |
| 59 | 073 | 3B | - | ; | ; | ; | ; | ; | | 0011 1011 |
| 60 | 074 | 3C | - | < | < | < | < | < | | 0011 1100 |
| 61 | 075 | 3D | AND Abs,X | = | = | = | = | = | | 0011 1101 |
| 62 | 076 | 3E | ROL Abs,X | > | > | > | > | > | | 0011 1110 |
| 63 | 077 | 3F | - | ? | ? | ? | ? | ? | | 0011 1111 |

Code Conversion Table

| Dec | Oct | Hex | 6502 Instruction | Poke U | Poke L | CHR$ U | CHR$ L | Std Asc | Token | Binary |
|---|---|---|---|---|---|---|---|---|---|---|
| 64 | 100 | 40 | RTI | | | @ | @ | @ | | 0100 0000 |
| 65 | 101 | 41 | EOR (Ind,X) | | A | A | a | A | | 0100 0001 |
| 66 | 102 | 42 | - | | B | B | b | B | | 0100 0010 |
| 67 | 103 | 43 | - | | C | C | c | C | | 0100 0011 |
| 68 | 104 | 44 | - | | D | D | d | D | | 0100 0100 |
| 69 | 105 | 45 | EOR 0-Page | | E | E | e | E | | 0100 0101 |
| 70 | 106 | 46 | LSR 0-Page | | F | F | f | F | | 0100 0110 |
| 71 | 107 | 47 | - | | G | G | g | G | | 0100 0111 |
| 72 | 110 | 48 | PHA | | H | H | h | H | | 0100 1000 |
| 73 | 111 | 49 | EOR Immediate | | I | I | i | I | | 0100 1001 |
| 74 | 112 | 4A | LSR Accum. | | J | J | j | J | | 0100 1010 |
| 75 | 113 | 4B | - | | K | K | k | K | | 0100 1011 |
| 76 | 114 | 4C | JMP Absolute | | L | L | l | L | | 0100 1100 |
| 77 | 115 | 4D | EOR Absolute | | M | M | m | M | | 0100 1101 |
| 78 | 116 | 4E | LSR Absolute | | N | N | n | N | | 0100 1110 |
| 79 | 117 | 4F | - | | O | O | o | O | | 0100 1111 |
| 80 | 120 | 50 | BVC | | P | P | p | P | | 0101 0000 |
| 81 | 121 | 51 | EOR (Ind),Y | | Q | Q | q | Q | | 0101 0001 |
| 82 | 122 | 52 | - | | R | R | r | R | | 0101 0010 |
| 83 | 123 | 53 | - | | S | S | s | S | | 0101 0011 |
| 84 | 124 | 54 | - | | T | T | t | T | | 0101 0100 |
| 85 | 125 | 55 | EOR 0-Page,X | | U | U | u | U | | 0101 0101 |
| 86 | 126 | 56 | LSR 0-Pag ,X | | V | V | v | V | | 0101 0110 |
| 87 | 127 | 57 | - | | W | W | w | W | | 0101 0111 |
| 88 | 130 | 58 | CLI | | X | X | x | X | | 0101 1000 |
| 89 | 131 | 59 | EOR Abs,Y | | Y | Y | y | Y | | 0101 1001 |
| 90 | 132 | 5A | - | | Z | Z | z | Z | | 0101 1010 |
| 91 | 133 | 5B | - | | | [ | [ | [ | | 0101 1011 |
| 92 | 134 | 5C | - | | | | | | | 0101 1100 |
| 93 | 135 | 5D | EOR Abs,X | | | ] | ] | ] | | 0101 1101 |
| 94 | 136 | 5E | LSR Abs,X | | | | | | | 0101 1110 |
| 95 | 137 | 5F | - | | | | | | | 0101 1111 |

Code Conversion Table

| Dec | Oct | Hex | 6502 Instruction | Poke 1 | Poke 2 | CHR$ | Asc | Token | Binary |
|---|---|---|---|---|---|---|---|---|---|
| 96 | 140 | 60 | RTS | | | | - | | 0110 0000 |
| 97 | 141 | 61 | ADC (Ind,X) | | | | a | | 0110 0001 |
| 98 | 142 | 62 | - | | | | b | | 0110 0010 |
| 99 | 143 | 63 | - | | | | c | | 0110 0011 |
| 100 | 144 | 64 | - | | | | d | | 0110 0100 |
| 101 | 145 | 65 | ADC 0-Page | | | | e | | 0110 0101 |
| 102 | 146 | 66 | ROR 0-page | | | | f | | 0110 0110 |
| 103 | 147 | 67 | - | | | | g | | 0110 0111 |
| 104 | 150 | 68 | PLA | | | | h | | 0110 1000 |
| 105 | 151 | 69 | ADC Immediate | | | | i | | 0110 1001 |
| 106 | 152 | 6A | ROR Accum. | | | | j | | 0110 1010 |
| 107 | 153 | 6B | - | | | | k | | 0110 1011 |
| 108 | 154 | 6C | JMP Indirect | | | | l | | 0110 1100 |
| 109 | 155 | 6D | ADC Absolute | | | | m | | 0110 1101 |
| 110 | 156 | 6E | ROR Absolute | | | | n | | 0110 1110 |
| 111 | 157 | 6F | - | | | | o | | 0110 1111 |
| 112 | 160 | 70 | BVS | | | | p | | 0111 0000 |
| 113 | 161 | 71 | ADC (Ind),Y | | | | q | | 0111 0001 |
| 114 | 162 | 72 | - | | | | r | | 0111 0010 |
| 115 | 163 | 73 | - | | | | s | | 0111 0011 |
| 116 | 164 | 74 | - | | | | t | | 0111 0100 |
| 117 | 165 | 75 | ADC 0-Page,X | | | | u | | 0111 0101 |
| 118 | 166 | 76 | ROR 0-Page,X | | | | v | | 0111 0110 |
| 119 | 167 | 77 | - | | | | w | | 0111 0111 |
| 120 | 170 | 78 | SEI | | | | x | | 0111 1000 |
| 121 | 171 | 79 | ADC Abs,Y | | | | y | | 0111 1001 |
| 122 | 172 | 7A | - | | | | z | | 0111 1010 |
| 123 | 173 | 7B | - | | | | ? | | 0111 1011 |
| 124 | 174 | 7C | - | | | | ? | | 0111 1100 |
| 125 | 175 | 7D | ADC Abs,X | | | | ? | | 0111 1101 |
| 126 | 176 | 7E | ROR Abs,X | | | | ? | | 0111 1110 |
| 127 | 177 | 7F | - | | | | DEL | | 0111 1111 |

Code Conversion Table

| Dec | Oct | Hex | 6502 Instruction | Poke 1 | Poke 2 | CHR$ | Asc | Token | Binary |
|---|---|---|---|---|---|---|---|---|---|
| 128 | 200 | 80 | - | | | | | END | 1000 0000 |
| 129 | 201 | 81 | STA (Ind,X) | | | | | FOR | 1000 0001 |
| 130 | 202 | 82 | - | | | | | NEXT | 1000 0010 |
| 131 | 203 | 83 | - | | | | | DATA | 1000 0011 |
| 132 | 204 | 84 | STY O-Page | | | | | INPUT£ | 1000 0100 |
| 133 | 205 | 85 | STA O-Page | | | | | INPUT | 1000 0101 |
| 134 | 206 | 86 | STX O-Page | | | | | DIM | 1000 0110 |
| 135 | 207 | 87 | - | | | | | READ | 1000 0111 |
| 136 | 210 | 88 | DEY | | | | | LET | 1000 1000 |
| 137 | 211 | 89 | - | | | | | GOTO | 1000 1001 |
| 138 | 212 | 8A | TXA | | | | | RUN | 1000 1010 |
| 139 | 213 | 8B | - | | | | | IF | 1000 1011 |
| 140 | 214 | 8C | STY Absolute | | | | | RESTORE | 1000 1100 |
| 141 | 215 | 8D | STA Absolute | | | | | GOSUB | 1000 1101 |
| 142 | 216 | 8E | STX Absolute | | | | | RETURN | 1000 1110 |
| 143 | 217 | 8F | - | | | | | REM | 1000 1111 |
| 144 | 220 | 90 | BCC | | | | | STOP | 1001 0000 |
| 145 | 221 | 91 | STA (Ind),Y | | | | | ON | 1001 0001 |
| 146 | 222 | 92 | - | | | | | WAIT | 1001 0010 |
| 147 | 223 | 93 | - | | | | | LOAD | 1001 0011 |
| 148 | 224 | 94 | STY O-Page,X | | | | | SAVE | 1001 0100 |
| 149 | 225 | 95 | STA O-Page,X | | | | | VERIFY | 1001 0101 |
| 150 | 226 | 96 | STX O-Page,Y | | | | | DEF | 1001 0110 |
| 151 | 227 | 97 | - | | | | | POKE | 1001 0111 |
| 152 | 230 | 98 | TYA | | | | | PRINT£ | 1001 1000 |
| 153 | 231 | 99 | STA Abs,Y | | | | | PRINT | 1001 1001 |
| 154 | 232 | 9A | TXS | | | | | CONT | 1001 1010 |
| 155 | 233 | 9B | - | | | | | LIST | 1001 1011 |
| 156 | 234 | 9C | - | | | | | CLR | 1001 1100 |
| 157 | 235 | 9D | STA Abs,X | | | | | CMD | 1001 1101 |
| 158 | 236 | 9E | - | | | | | SYS | 1001 1110 |
| 159 | 237 | 9F | - | | | | | OPEN | 1001 1111 |
| 160 | 240 | A0 | LDY Immediate | | | | | CLOSE | 1010 0000 |
| 161 | 241 | A1 | LDA (Ind,X) | | | | | GET | 1010 0001 |
| 162 | 242 | A2 | LDX Immediate | | | | | NEW | 1010 0010 |
| 163 | 243 | A3 | - | | | | | TAB( | 1010 0011 |
| 164 | 244 | A4 | LDY O-Page | | | | | TO | 1010 0100 |
| 165 | 245 | A5 | LDA O-Page | | | | | FN | 1010 0101 |
| 166 | 246 | A6 | LDX O-Page | | | | | SPC( | 1010 0110 |
| 167 | 247 | A7 | - | | | | | THEN | 1010 0111 |
| 168 | 250 | A8 | TAY | | | | | NOT | 1010 1000 |
| 169 | 251 | A9 | LDA Immediate | | | | | STEP | 1010 1001 |
| 170 | 252 | AA | TAX | | | | | + | 1010 1010 |
| 171 | 253 | AB | - | | | | | - | 1010 1011 |
| 172 | 254 | AC | LDY Absolute | | | | | * | 1010 1100 |
| 173 | 255 | AD | LDA Absolute | | | | | / | 1010 1101 |
| 174 | 256 | AE | LDX Absolute | | | | | | 1010 1110 |
| 175 | 257 | AF | - | | | | | AND | 1010 1111 |
| 176 | 260 | B0 | BCS | | | | | OR | 1011 0000 |
| 177 | 261 | B1 | LDA (Ind),Y | | | | | > | 1011 0001 |
| 178 | 262 | B2 | - | | | | | = | 1011 0010 |
| 179 | 263 | B3 | - | | | | | < | 1011 0011 |
| 180 | 264 | B4 | LDY O-Page,X | | | | | SGN | 1011 0100 |
| 181 | 265 | B5 | LDA O-Page,X | | | | | INT | 1011 0101 |
| 182 | 266 | B6 | LDX O-Page,Y | | | | | ABS | 1011 0110 |
| 183 | 267 | B7 | - | | | | | USR | 1011 0111 |
| 184 | 270 | B8 | CLV | | | | | FRE | 1011 1000 |
| 185 | 271 | B9 | LDA Abs,Y | | | | | POS | 1011 1001 |
| 186 | 272 | BA | TSX | | | | | SQR | 1011 1010 |
| 187 | 273 | BB | - | | | | | RND | 1011 1011 |
| 188 | 274 | BC | LDY Abs,X | | | | | LOG | 1011 1100 |
| 189 | 275 | BD | LDA Abs,X | | | | | EXP | 1011 1101 |
| 190 | 276 | BE | LDX Abs,Y | | | | | COS | 1011 1110 |
| 191 | 277 | BF | - | | | | | SIN | 1011 1111 |
| 192 | 300 | C0 | CPY Immediate | | | | | TAN | 1100 0000 |
| 193 | 301 | C1 | CMP (Ind,X) | | | | | ATN | 1100 0001 |
| 194 | 302 | C2 | - | | | | | PEEK | 1100 0010 |
| 195 | 303 | C3 | - | | | | | LEN | 1100 0011 |
| 196 | 304 | C4 | CPY O-Page | | | | | STR$ | 1100 0100 |
| 197 | 305 | C5 | CMP O-Page | | | | | VAL | 1100 0101 |
| 198 | 306 | C6 | DEC O-Page | | | | | ASC | 1100 0110 |
| 199 | 307 | C7 | - | | | | | CHR$ | 1100 0111 |
| 200 | 310 | C8 | INY | | | | | LEFT$ | 1100 1000 |
| 201 | 311 | C9 | CMP Immediate | | | | | RIGHT$ | 1100 1001 |
| 202 | 312 | CA | DEX | | | | | MID$ | 1100 1010 |
| 203 | 313 | CB | - | | | | | | 1100 1011 |

Code Conversion Table

| Dec | Oct | Hex | 6502 Instruction | Poke 1 | Poke 2 | CHR$ | Asc | Token | Binary |
|---|---|---|---|---|---|---|---|---|---|
| 204 | 314 | CC | CPY Absolute | | | | | | 1100 1100 |
| 205 | 315 | CD | CMP Absolute | | | | | | 1100 1101 |
| 206 | 316 | CE | DEC Absolute | | | | | | 1100 1110 |
| 207 | 317 | CF | - | | | | | | 1100 1111 |
| 208 | 320 | D0 | BNE | | | | | | 1101 0000 |
| 209 | 321 | D1 | CMP (Ind),Y | | | | | | 1101 0001 |
| 210 | 322 | D2 | - | | | | | | 1101 0010 |
| 211 | 323 | D3 | - | | | | | | 1101 0011 |
| 212 | 324 | D4 | - | | | | | | 1101 0100 |
| 213 | 325 | D5 | CMP 0-Page,X | | | | | | 1101 0101 |
| 214 | 326 | D6 | DEC 0-Page,X | | | | | | 1101 0110 |
| 215 | 327 | D7 | - | | | | | | 1101 0111 |
| 216 | 330 | D8 | CLD | | | | | | 1101 1000 |
| 217 | 331 | D9 | CMP Abs,Y | | | | | | 1101 1001 |
| 218 | 332 | DA | - | | | | | | 1101 1010 |
| 219 | 333 | DB | - | | | | | | 1101 1011 |
| 220 | 334 | DC | - | | | | | | 1101 1100 |
| 221 | 335 | DD | CMP Abs,X | | | | | | 1101 1101 |
| 222 | 336 | DE | DEC Abs,X | | | | | | 1101 1110 |
| 223 | 337 | DF | - | | | | | | 1101 1111 |
| 224 | 340 | E0 | CPX Immediate | | | | | | 1110 0000 |
| 225 | 341 | E1 | SBC (Ind,X) | | | | | | 1110 0001 |
| 226 | 342 | E2 | - | | | | | | 1110 0010 |
| 227 | 343 | E3 | - | | | | | | 1110 0011 |
| 228 | 344 | E4 | CPX 0-Page | | | | | | 1110 0100 |
| 229 | 345 | E5 | SBX 0-Page | | | | | | 1110 0101 |
| 230 | 346 | E6 | INC 0-Page | | | | | | 1110 0110 |
| 231 | 347 | E7 | - | | | | | | 1110 0111 |
| 232 | 350 | E8 | INX | | | | | | 1110 1000 |
| 233 | 351 | E9 | SBC Immediate | | | | | | 1110 1001 |
| 234 | 352 | EA | NOP | | | | | | 1110 1010 |
| 235 | 353 | EB | - | | | | | | 1110 1011 |
| 236 | 354 | EC | CPX Absolute | | | | | | 1110 1100 |
| 237 | 355 | ED | SBC Absolute | | | | | | 1110 1101 |
| 238 | 356 | EE | INC Absolute | | | | | | 1110 1110 |
| 239 | 357 | EF | - | | | | | | 1110 1111 |
| 240 | 360 | F0 | BEQ | | | | | | 1111 0000 |
| 241 | 361 | F1 | SBC (Ind),Y | | | | | | 1111 0001 |
| 242 | 362 | F2 | - | | | | | | 1111 0010 |
| 243 | 363 | F3 | - | | | | | | 1111 0011 |
| 244 | 364 | F4 | - | | | | | | 1111 0100 |
| 245 | 365 | F5 | SBC 0-Page,X | | | | | | 1111 0101 |
| 246 | 366 | F6 | INC 0-Page,X | | | | | | 1111 0110 |
| 247 | 367 | F7 | - | | | | | | 1111 0111 |
| 248 | 370 | F8 | SED | | | | | | 1111 1000 |
| 249 | 371 | F9 | SBC Abs,Y | | | | | | 1111 1001 |
| 250 | 372 | FA | - | | | | | | 1111 1010 |
| 251 | 373 | FB | - | | | | | | 1111 1011 |
| 252 | 374 | FC | - | | | | | | 1111 1100 |
| 253 | 375 | FD | SBC Abs,X | | | | | | 1111 1101 |
| 254 | 376 | FE | INC Abs,X | | | | | | 1111 1110 |
| 255 | 377 | FF | - | | | | | | 1111 1111 |

# Disk commands

DISKETTE COMMAND SUMMARY BASIC 4.0

APPEND £<file number>,"<name>":,D<x>;:ON U<y>;
Append data to the end of a sequential file.

BACKUP D<x> TO D<y> :ON U<z>;
Duplicate entire contents of one disk onto another.

CATALOG/DIRECTORY :D<x>;:ON U<y>;
Display disk directory on screen.

COLLECT :D<x>;:ON U<y>;
Release space allocated to improperly closed files.

CONCAT :D<x>,;"<name1>" TO :D<y>,;"name2>":ON U<z>;
Concatenate sequential files.

COPY :D<x>,;"<name1>" TO :D<y>,;"<name2>": ON U<z>;
Make a copy of a file within a disk unit.

DCLOSE :£<1>; :ON U<x>;
Close disk files.

DLOAD "<name>":,D<x>; :ON U<y>;
Load BASIC program file from disk.

DOPEN £<1>,"<name>":,L<y>;:,D<x>; :ON U<z>; :,W;
Declare a sequential or random access file for read or write.

DSAVE "<name>":,D<x>; :ON U<y>;
Save a BASIC program file to disk.

HEADER "<disk name>",D<x>:,I<zz>; :ON U<y>;
To format a blank disk or clear an old disk.

RECORD £<logical file>,<record>:,<byte>;
Used before GET£,INPUT£ or PRINT£ to position record pointer in a random access file.

RENAME :D<x>.;"<old name>" TO "<new name>" :ON U<y>]
Change the name of a disk file.

SCRATCH "<name>":,Dx; :ON U<y>;
Delete a disk file.

DISKETTE COMMAND SUMMARY BASIC 2.0

(APPEND) Not available.

(BACKUP) PRINT£15,"D<y> = x"

(CATALOG/ LOAD "$D<x>:?...name:*;:=P/S;;
DIRECTORY)

(COLLECT) PRINT£15,"VD<x>"

(CONCAT) PRINT£15,"CD<y>:<name3>=D<y>:<name2>,D<x>:
<name1>"

(COPY) PRINT£15,"CD<y>:<name2>=D<x>:<name1>"

(DCLOSE) CLOSE <file number>

(DLOAD) LOAD "D<x>:<filename>:*;"

(DOPEN) OPEN <file number>,;<device>;:,<address>;
:."<filename>:,<type>;:,<mode>;";

(DSAVE) SAVE :"<filename>";:,<device>;

(HEADER) PRINT£15,"ND<x>:filename,I<zz>"

(RECORD) Not available.

(RENAME) PRINT£15,"RD<x>:<name2>=<name1>"

(SCRATCH) PRINT£15,"SD<x>:<filename>:*;/*"

# Disk error messages

DOS ERROR MESSAGES

OK (00 00 00)

No errors encountered.

FILES SCRATCHED (00 £ Files 00)

Files scratched.

BLOCK HEADER NOT FOUND (20 T S)

Disk controller is unable to locate the header of the requested data block. Caused by an illegal sector number, or the header has been destroyed.

NO SYNC CHARACTER (21 T S)

Disk controller is unable to detect a sync mark on the desired track. Caused by misalignment of the read/write head or no diskette is present. Can also indicate a hardware failure.

DATA BLOCK NOT PRESENT (22 T S)

Disk controller has been requested to read or verify a data block which was not properly written. This error message occurs in conjunction with the BLOCK COMMANDS and indicates an illegal track and/or sector request.

CHECKSUM ERROR IN DATA BLOCK (23 T S)

This error message indicates that there is an error in one or more of the data bytes. The data has been read into the DOS memory, but the checksum over the data is in error. This message may also indicate grounding problems.

BYTE DECODING ERROR (24 T S)

The data or header has been read into the DOS memory, but a hardware error has been created due to an invalid bit pattern in the data byte. This message may also indicate grounding problems.

CHECKSUM ERROR IN HEADER (27 T S)

Disk controller has detected an error in the header of the requested data block. The block has not been read into the DOS memory. This message may also indicate grounding problems.

WRITE-VERIFY ERROR (25 T S)

This message is generated if the controller detects a mis-match between the written data and the data in the DOS memory.

WRITE PROTECT ON (26 T S)

This message is generated when the controller has been requested to write a data block while the write protect switch is depressed. Typically, this is caused by using a diskette with a write protect tab over the notch.

LONG DATA BLOCK (28 T S)

The controller attempts to detect the sync mark of the next header after writing a data block. If the sync mark does not appear within a pre-determined time, the error message is generated. The error message is caused by a bad diskette format (the data extends into the next block), or by a hardware failure.

DISK ID MIS-MATCH (29 T S)

This message is generated when the controller has been requested to access a diskette which has not been initialised. This error can also occur if a diskette has a bad header.

GENERAL SYNTAX (30 00 00)

The DOS cannot interpret the command sent to the command channel. Typically, this is caused by an illegal number of file names, or patterns are illegally used. For example, two file names may appear on the left side of the COPY command.

INVALID COMMAND (31 00 00)

The DOS does not recognise the command. The command must start in the first position.

LONG LINE (32 00 00)

The command sent is longer than forty characters.

INVALID FILE NAME (33 00 00)

Pattern matching is invalidly used in the OPEN, or SAVE command.

NO FILE GIVEN (34 00 00)

The file name was left out of the command, or the DOS does not recognise it as such. Typically, a quotation mark (") or colon (:) has been left out of the command.

INVALID DOS COMMAND (39 00 00)

An unrecognisable DOS command was received.

RECORD NOT PRESENT (50 00 00)

An INPUT£ or GET£ statement selected a record beyond the current end of file. This is an error if you are attempting to read a record; it is not necessarily an error if you are positioning to the end of the file in order to add new records to an old file.

OVERFLOW IN RECORD (51 T S)

A PRINT£ statement attempted to write more than the allowed number of characters to a relative file. The terminating carriage return is counted as one character when computing record length.

FILE TOO LARGE (52 T S)

The current record position will result in disk overflow on the next write-to-disk operation.

WRITE FILE OPEN (60 00 00)

This message is generated when a write file that has not been closed is being opened for reading.

FILE NOT OPEN (61 00 00)

This message is generated when a file is being accessed that has not been opened in the DOS. Sometimes, in this case, a message is not

generated; the request is simply ignored.

FILE NOT FOUND (62 00 00)

The requested file does not exist on the indicated drive.

FILE EXISTS (63 00 00)

The file name of the file being created already exists on the diskette.

FILE TYPE MIS-MATCH (64 00 00)

The file type does not match the file type in the directory entry for the requested file.

NO BLOCK (65 T S)

This message occurs in conjunction with the B-A command. It indicates that the block to be allocated has been previously allocated. The parameters indicate the next higher in number available track and sector. If the parameters are zero, then all blocks higher in number are in use.

ILLEGAL TRACK & SECTOR (66 T S)

An attempt has been made to access a sector that does not physically exist. The track and/or sector number specified is out of the allowed range for the current diskette. Unless you are using random access files, you should never see this error code.

ILLEGAL SYSTEM TRACK & SECTOR (67 T S)

When accessing program or data files, an attempt has been made to access a sector that is reserved for use by the DOS.

NO CHANNEL (70 00 00)

The requested channel is not available, or all channels are in use. A maximum of five sequential files may be opened at one time to the DOS. Direct access channels may have six open files.

DIR (71 00 00)

The BAM does not match the internal count. There is a problem in the BAM allocation or the BAM has

been over-written in DOS memory. To correct this problem re-initialise the diskette to restore the BAM in memory. Some active files may be terminated by the corrective action.

DISK FULL (72 00 00)

Either the blocks on the diskette are used, or the directory is at its limit (152 entries).

DOS MIS-MATCH (73 00 00)

Data written to a diskette using any one version of DOS may be read using any other version of DOS. However, you must write to a diskette using the same DOS version with which the diskette was initialised. Error 73 is reported if you attempt to write to a diskette using a different version of DOS from the one which created and initialised the diskette.

DRIVE NOT READY (74 00 00)

An attempt has been made to access the 8050 diskette unit with the selective drive.

STRUCTURE OF INDIVIDUAL DIRECTORY ENTRIES.

| BYTE | DEFINITION |
|---|---|
| 0 | File flag OR'ed with $80<br>0 = DELETED<br>1 = SEQ<br>2 = PRG<br>3 = REL |
| 1-2 | First data block track and sector. |
| 3-18 | Reserved for file name. |
| 19-20 | Track and sector of first side sector block (REL files only). |
| 21 | Record size for REL files. |
| 22-25 | Not used. |
| 26-27 | Track and sector of replacement file using '@' operations. |
| 28-29 | Number of blocks in file. |

STANDARD FILE FORMATS.

| BYTE | DEFINITION |
|---|---|
| 0-1 | Track and sector of next seqential block. |
| 2-255 | Reserved for file storage. |

STRUCTURE OF BAM ENTRY.

| BYTE | DEFINITION |
|---|---|
| 0 | Number of available sectors. |
| 1 | Bit map for sectors 00-07. |
| 2 | Bit map for sectors 08-15. |
| 3 | Bit map for sectors 16-23. |
| 4 | 8050 only: Bit map for sectors for 24-31. |

RELATIVE FILES ONLY.
Side sector blocks.

| BYTE | DEFINITION |
|---|---|
| 0-1 | Track and sector of next side sector block. |
| 2 | Side sector number. |
| 3 | Record length. |
| 4-5 | Track and sector of first side sector. |
| 6-7 | Track and sector of second side sector. |
| 8-9 | Track and sector of third side sector. |
| 10-11 | Track and sector of fourth side sector. |
| 12-13 | Track and sector of fifth side sector. |
| 14-15 | Track and sector of sixth side sector. |
| 16-255 | Track and sector pointers to data blocks. |

8050 BAM FORMAT.
Track 38, Sector 00.

| BYTE | CONTENTS | DEFINITION |
|---|---|---|
| 0-1 | 38,3 | Track and sector of second BAM block. |
| 2 | 67 | Indicates 8050. |
| 3 | 0 | NULL flag. |
| 4 | 1 | Lowest track number in this BAM block. |
| 5 | 51 | Highest track number plus 1 in this BAM block. |
| 6 | - | Number of unused blocks on track 1. |
| 7-10 | - | Bit map of available blocks on track 1. |
| 11-255 | - | 5 bytes each for BAM of tracks 2-50. |

8050 BAM FORMAT.
Track 38, Sector 03.

| BYTE | CONTENTS | DEFINITION |
|---|---|---|
| 0-1 | 39,1 | Track and sector of first directory block. |
| 2 | 67 | Indicates 8050. |
| 3 | 0 | NULL flag. |
| 4 | 51 | Lowest track number. |
| 5 | 78 | Highest track number plus 1. |
| 6 | - | Number of unused block on track 51. |
| 7-10 | - | Bit map of available blocks on track 51. |
| 11-140 | - | 5 bytes each for BAM of tracks 52-77. |
| 144-255 | - | Not used. |

COMMON DIRECTORY FORMAT
For 2040/3040/4040/1540/1541: Track 18, Sector 01.
For 8050: Track 39, Sector 01

| BYTE | DEFINITION |
|---|---|
| 0-1 | Track and sector of next directory block. |
| 2-31 | File entry £1. |
| 34-63 | File entry £2. |
| 66-95 | File entry £3. |
| 98-127 | File entry £4. |
| 130-159 | File entry £5. |
| 162-191 | File entry £6. |
| 194-223 | File entry £7. |
| 226-255 | File entry £8. |

2040/3040 BAM & HEADER.
Track 18, Sector 00.

| BYTE | CONTENTS | DEFINITION |
|---|---|---|
| 0-1 | 18,1 | Track and sector of first directory block. |
| 2 | 1 | Indicates DOS 1. |
| 3 | 0 | NULL flag. |
| 4-143 | - | The BAM bit map of available blocks for tracks 1-35. |
| 144-161 | - | Reserved for disk name. |
| 162-163 | - | Reserved for disk ID. |
| 164-170 | 160 | Shifted SPACES. |
| 171-255 | 0 | Not used. |

4040/1540/1541 BAM & HEADER.
Track 18, Sector 00

| BYTE | CONTENTS | DEFINITION |
|---|---|---|
| 0-1 | 18,1 | Track and sector of first directory block. |
| ?2 | 65 | Indicates 4040/1540/1541 |
| 3 | 0 | NULL byte. |
| 4-143 | - | BAM bit map of available blocks for tracks 1-35. |
| 144-161 | - | Reserved for disk name. |
| 162-163 | - | Reserved for disk ID. |
| 164 | 160 | Shifted SPACE. |
| 165-166 | 50,65 | Indicates DOS version (2A) and format. |
| 166-170 | 160 | Shifted SPACES. |
| 171-255 | 0 | Not used. |

8050 DIRECTORY HEADER.
Track 39, Sector 00.

| BYTE | CONTENTS | DEFINITION |
|---|---|---|
| 0-1 | 38,0 | Track and sector of first BAM block. |
| 2 | 67 | Indicates 8050. |
| 3 | 0 | NULL flag. |
| 4-5 | 0 | Not used. |
| 6-21 | - | Indicates disk name. |
| 22-23 | 160 | Shifted SPACES. |
| 24-25 | - | Indicates disk ID. |
| 26 | 160 | Shifted SPACE. |
| 27-28 | 50,67 | Indicates DOS version (2C) and format. |
| 29-32 | 160 | Shifted SPACES. |
| 33-255 | 0 | Not used. |

# Extramon listing

```
100 PRINT"TINY PEEKER/POKER"
110 X$="*":INPUTX$:IFX$="*"THENEND
120 GOSUB500
130 IF E GOTO280
140 A=V
150 IFJ>LEN(X$)GOTO300
160 FORI=0TO7
170 P=J:GOSUB550
180 C(I)=V
190 IF E GOTO 280
200 NEXTI
210 T=0
220 FORI=0TO7
230 POKE A+I,C(I)
240 T=T+C(I)
250 NEXT I
260 PRINT"CHECKSUM=";T
270 GOTO110
280 PRINTMID$(X$,1,J);"??":GOTO110
300 T=0
310 FORI=0TO7
320 V=PEEK(A+I)
330 T=T+V
340 V=V/16
350 PRINT " ";
360 FORJ=1TO2
370 V%=V
380 V=(V-V%)*16
390 IFV%>9THENV%=V%+7
400 PRINTCHR$(V%+48);
410 NEXT J
420 NEXT I
430 PRINT "/";T
440 GOTO110
500 P=1
510 L=4
520 GOTO600
550 P=J
560 L=2
600 E=0
610 V=0
620 FORJ=P TO LEN(X$)
630 X=ASC(MID$(X$,J))
640 IFX=32 THEN NEXT J
650 IFJ>LEN(X$)THEN790
660 P=J
670 FORJ=PTOLEN(X$)
680 X=ASC(MID$(X$,J))
```

```
690 IF X<>32 THEN NEXT J
700 IF J-P<>L THEN 790
710 FORK=PTOJ-1
720 X=ASC(MID$(X$,K))
730 IF X<58 THENX=X-48
740 IF X>64 THEN X=X-55
750 IF X<0 OR X>15 THEN790
760 V=V*16+X
770 NEXT K
780 RETURN
790 E=-1
800 RETURN
```

```
0800 00 1A 04 64 00 99 22 93
0808 12 1D 1D 1D 1D 53 55 50
0810 45 52 20 36 34 2D 4D 4F
0818 4E 00 31 04 6E 00 99 22
0820 11 20 20 20 20 20 20 20
0828 20 20 20 20 20 20 20 20
0830 00 4B 04 78 00 99 22 11
0838 20 2E 2E 4A 49 4D 20 42
0840 55 54 54 45 52 46 49 45
0848 4C 44 00 66 04 82 00 9E
0850 28 C2 28 34 33 29 AA 32
0858 35 36 AC C2 28 34 34 29
0860 AA 31 32 37 29 00 00 00
0868 AA AA AA AA AA AA AA AA
0870 AA AA AA AA AA AA AA AA
0878 AA AA AA AA AA AA AA AA

0880 A5 2D 85 22 A5 2E 85 23
0888 A5 37 85 24 A5 38 85 25
0890 A0 00 A5 22 D0 02 C6 23
0898 C6 22 B1 22 D0 3C A5 22
08A0 D0 02 C6 23 C6 22 B1 22
08A8 F0 21 85 26 A5 22 D0 02
08B0 C6 23 C6 22 B1 22 18 65
08B8 24 AA A5 26 65 25 48 A5
08C0 37 D0 02 C6 38 C6 37 68
08C8 91 37 8A 48 A5 37 D0 02
08D0 C6 38 C6 37 68 91 37 18
08D8 90 B6 C9 4F D0 ED A5 37
08E0 85 33 A5 38 85 34 6C 37
08E8 00 4F 4F 4F 4F AD E6 FF
08F0 00 8D 16 03 AD E7 FF 00
08F8 8D 17 03 A9 80 20 90 FF

0900 00 00 D8 68 8D 3E 02 68
0908 8D 3D 02 68 8D 3C 02 68
0910 8D 3B 02 68 AA 68 A8 38
0918 8A E9 02 8D 3A 02 98 E9
0920 00 00 8D 39 02 BA 8E 3F
0928 02 20 57 FD 00 A2 42 A9
0930 2A 20 57 FA 00 A9 52 D0
0938 34 E6 C1 D0 06 E6 C2 D0
0940 02 E6 26 60 20 CF FF C9
0948 0D D0 F8 68 68 A9 90 20
0950 D2 FF A9 00 00 85 26 A2
0958 0D A9 2E 20 57 FA 00 A9
0960 05 20 D2 FF 20 3E F8 00
0968 C9 2E F0 F9 C9 20 F0 F5
0970 A2 0E DD B7 FF 00 D0 0C
0978 8A 0A AA BD C7 FF 00 48

0980 BD C6 FF 00 48 60 CA 10
0988 EC 4C ED FA 00 A5 C1 8D
0990 3A 02 A5 C2 8D 39 02 60
0998 A9 08 85 1D A0 00 00 20
09A0 54 FD 00 B1 C1 20 48 FA
09A8 00 20 33 F8 00 C6 1D D0
09B0 F1 60 20 88 FA 00 90 0B
09B8 A2 00 00 81 C1 C1 C1 F0
09C0 03 4C ED FA 00 20 33 F8
09C8 00 C6 1D 60 A9 3B 85 C1
09D0 A9 02 85 C2 A9 05 60 98
09D8 48 20 57 FD 00 68 A2 2E
09E0 4C 57 FA 00 A9 90 20 D2
09E8 FF A2 00 00 BD EA FF 00
09F0 20 D2 FF E8 E0 16 D0 F5
09F8 A0 3B 20 C2 F8 00 AD 39

0A00 02 20 48 FA 00 AD 3A 02
0A08 20 48 FA 00 20 B7 F8 00
0A10 20 8D F8 00 F0 5C 20 3E
0A18 F8 00 20 79 FA 00 90 33
0A20 20 69 FA 00 20 3E F8 00
0A28 20 79 FA 00 90 28 20 69
0A30 FA 00 A9 90 20 D2 FF 20
0A38 E1 FF F0 3C A6 26 D0 38
0A40 A5 C3 C5 C1 A5 C4 E5 C2
0A48 90 2E A0 3A 20 C2 F8 00
0A50 20 41 FA 00 20 8B F8 00
0A58 F0 E0 4C ED FA 00 20 79
0A60 FA 00 90 03 20 80 F8 00
0A68 20 B7 F8 00 D0 07 20 79
0A70 FA 00 90 EB A9 00 85 1D
0A78 20 3E F8 00 20 A1 F8 00

0A80 D0 F8 4C 47 F8 00 20 CF
0A88 FF C9 0D F0 0C C9 20 D0
0A90 D1 20 79 FA 00 90 03 20
0A98 80 F8 00 A9 90 20 D2 FF
0AA0 AE 3F 02 9A 78 AD 39 02
0AA8 48 AD 3A 02 48 AD 3B 02
0AB0 48 AD 3C 02 AE 3D 02 AC
0AB8 3E 02 40 A9 90 20 D2 FF
0AC0 AE 3F 02 9A 6C 02 A0 A0
0AC8 01 84 BA 84 B9 88 84 B7
0AD0 84 90 84 93 A9 40 85 BB
0AD8 A9 02 85 BC 20 CF FF C9
0AE0 20 F0 F9 C9 0D F0 3B C9
0AE8 22 D0 14 20 CF FF C9 22
0AF0 F0 10 C9 0D F0 29 91 BB
0AF8 E6 B7 C8 C0 10 D0 EC 4C

0B00 ED FA 00 20 CF FF C9 0D
0B08 F0 16 C9 2C D0 DC 20 88
0B10 FA 00 29 0F F0 E9 C9 03
0B18 F0 E5 85 BA 20 CF FF C9
0B20 0D 60 6C 30 03 6C 32 03
0B28 20 96 F9 00 D0 D4 A9 90
0B30 20 D2 FF A9 00 00 20 EF
0B38 F9 00 A5 90 29 10 D0 C4
0B40 4C 47 F8 00 20 96 F9 00
0B48 C9 2C D0 BA 20 79 FA 00
0B50 20 69 FA 00 20 CF FF C9
0B58 2C D0 AD 20 79 FA 00 A5
0B60 C1 85 AE A5 C2 85 AF 20
0B68 69 FA 00 20 CF FF C9 0D
0B70 D0 98 A9 90 20 D2 FF 20
0B78 F2 F9 00 4C 47 F8 00 A5

0B80 C2 20 48 FA 00 A5 C1 48
0B88 4A 4A 4A 4A 20 60 FA 00
0B90 AA 68 29 0F 20 60 FA 00
0B98 48 8A 20 D2 FF 68 4C D2
0BA0 FF 09 30 C9 3A 90 02 69
0BA8 06 60 A2 02 B5 C0 48 B5
0BB0 C2 95 C0 68 95 C2 CA D0
0BB8 F3 60 20 88 FA 00 90 02
0BC0 85 C2 20 88 FA 00 90 02
0BC8 85 C1 60 A9 00 00 85 2A
0BD0 20 3E F8 00 C9 20 D0 09
0BD8 20 3E F8 00 C9 20 D0 0E
0BE0 18 60 20 AF FA 00 0A 0A
0BE8 0A 0A 85 2A 20 3E F8 00
0BF0 20 AF FA 00 05 2A 38 60
0BF8 C9 3A 90 02 69 06 29 0F

0C00 60 A2 02 2C A2 00 00 B4
0C08 C1 D0 08 B4 C2 D0 02 E6
0C10 26 D6 C2 D6 C1 60 20 3E
0C18 F8 00 C9 20 F0 F9 60 A9
0C20 00 00 8D 00 00 01 20 CC
0C28 FA 00 20 8F FA 00 20 7C
0C30 FA 00 90 09 60 20 3E F8
0C38 00 20 79 FA 00 B0 DE AE
0C40 3F 02 9A A9 90 20 D2 FF
0C48 A9 3F 20 D2 FF 4C 47 F8
0C50 00 20 54 FD 00 CA D0 FA
0C58 60 E6 C3 D0 02 E6 C4 60
0C60 A2 02 B5 C0 48 B5 27 95
0C68 C0 68 95 27 CA D0 F3 60
0C70 A5 C3 A4 C4 38 E9 02 B0
0C78 0F 88 90 0B A5 28 A4 29

0C80 4C 33 FB 00 A5 C3 A4 C4
0C88 38 E5 C1 85 1E 98 E5 C2
0C90 A8 05 1E 60 20 D4 FA 00
0C98 20 69 FA 00 20 E5 FA 00
0CA0 20 0C FB 00 20 E5 FA 00
0CA8 20 2F FB 00 20 69 FA 00
0CB0 90 15 A6 26 D0 64 20 28
0CB8 FB 00 90 5F A1 C1 81 C3
0CC0 20 05 FB 00 20 33 F8 00
0CC8 D0 EB 20 28 FB 00 18 A5
0CD0 1E 65 C3 85 C3 98 65 C4
0CD8 85 C4 20 0C FB 00 A6 26
0CE0 D0 3D A1 C1 81 C3 20 28
0CE8 FB 00 B0 34 20 B8 FA 00
0CF0 20 BB FA 00 4C 7D FB 00
0CF8 20 D4 FA 00 20 69 FA 00

0D00 20 E5 FA 00 20 69 FA 00
0D08 20 3E F8 00 20 88 FA 00
0D10 90 14 85 1D A6 26 D0 11
0D18 20 2F FB 00 90 0C A5 1D
0D20 81 C1 20 33 F8 00 D0 EE
0D28 4C ED FA 00 4C 47 F8 00
0D30 20 D4 FA 00 20 69 FA 00
0D38 20 E5 FA 00 20 69 FA 00
0D40 20 3E F8 00 A2 00 00 20
0D48 3E F8 00 C9 27 D0 14 20
0D50 3E F8 00 9D 10 02 E8 20
0D58 CF FF C9 0D F0 22 E0 20
0D60 D0 F1 F0 1C 8E 00 00 01
0D68 20 8F FA 00 90 C6 9D 10
0D70 02 E8 20 CF FF C9 0D F0
0D78 09 20 88 FA 00 90 B6 E0

0D80 20 D0 EC 86 1C A9 90 20
0D88 D2 FF 20 57 FD 00 A2 00
0D90 00 A0 00 00 B1 C1 DD 10
0D98 02 D0 0C C8 E8 E4 1C D0
0DA0 F3 20 41 FA 00 20 54 FD
0DA8 00 20 33 F8 00 A6 26 D0
0DB0 8D 20 2F FB 00 B0 DD 4C
0DB8 47 F8 00 20 D4 FA 00 85
0DC0 20 A5 C2 85 21 A2 00 00
0DC8 86 28 A9 93 20 D2 FF A9
0DD0 90 20 D2 FF A9 16 85 1D
0DD8 20 6A FC 00 20 CA FC 00
0DE0 85 C1 84 C2 C6 1D D0 F2
0DE8 A9 91 20 D2 FF 4C 47 F8
0DF0 00 A0 2C 20 C2 F8 00 20
0DF8 54 FD 00 20 41 FA 00 20
```

```
0E00 54 FD 00 A2 00 00 A1 C1   1000 00 BD 2A FF 00 20 B9 FE
0E08 20 D9 FC 00 48 20 1F FD   1008 00 D0 65 CA D0 D1 F6 0A
0E10 00 68 20 35 FD 00 A2 06   1010 20 B8 FE 00 D0 AB 20 B8
0E18 E0 03 D0 12 A4 1F F0 0E   1018 FE 00 D0 A6 A5 28 C5 1D
0E20 A5 2A C9 E8 B1 C1 B0 1C   1020 D0 A0 20 69 FA 00 A4 1F
0E28 20 C2 FC 00 88 D0 F2 06   1028 F0 28 A5 29 C9 9D D0 1A
0E30 2A 90 0E BD 2A FF 00 20   1030 20 1C FB 00 90 0A 98 D0
0E38 A5 FD 00 BD 30 FF 00 F0   1038 04 A5 1E 10 0A 4C ED FA
0E40 03 20 A5 FD 00 CA D0 D5   1040 00 C8 D0 FA A5 1E 10 F6
0E48 60 20 CD FC 00 AA E8 D0   1048 A4 1F D0 03 B9 C2 00 00
0E50 01 C8 98 20 C2 FC 00 8A   1050 91 C1 88 D0 F8 A5 26 91
0E58 86 1C 20 48 FA 00 A6 1C   1058 C1 20 CA FC 00 85 C1 84
0E60 60 A5 1F 38 A4 C2 AA 10   1060 C2 A9 90 20 D2 FF A0 41
0E68 01 88 65 C1 90 01 C8 60   1068 20 C2 F8 00 20 54 FD 00
0E70 A8 4A 90 0B 4A B0 17 C9   1070 20 41 FA 00 20 54 FD 00
0E78 22 F0 13 29 07 09 80 4A   1078 A9 05 20 D2 FF 4C B0 FD

0E80 AA BD D9 FE 00 B0 04 4A   1080 00 A8 20 BF FE 00 D0 11
0E88 4A 4A 4A 29 0F D0 04 A0   1088 9B F0 0E 86 1C A6 1D DD
0E90 80 A9 00 00 AA BD 1D FF   1090 10 02 08 E8 86 1D A6 1C
0E98 00 85 2A 29 03 85 1F 98   1098 28 60 C9 30 90 03 C9 47
0EA0 29 8F AA 98 A0 03 E0 8A   10A0 60 38 60 40 02 45 03 D0
0EA8 F0 0B 4A 90 08 4A 4A 09   10A8 0B 40 09 30 22 45 33 D0
0EB0 20 88 D0 FA C8 88 D0 F2   10B0 0B 40 09 40 02 45 33 D0
0EB8 60 B1 C1 20 C2 FC 00 A2   10B8 0B 40 09 40 02 45 B3 D0
0EC0 01 20 FE FA 00 C4 1F C8   10C0 0B 40 09 00 00 22 44 33
0EC8 90 F1 A2 03 C0 04 90 F2   10C8 D0 8C 44 00 00 11 22 44
0ED0 60 A8 B9 37 FF 00 85 28   10D0 33 D0 8C 44 9A 10 22 44
0ED8 B9 77 FF 00 85 29 A9 00   10D8 33 D0 0B 40 09 10 22 44
0EE0 00 A0 05 06 29 26 28 2A   10E0 33 D0 0B 40 09 62 13 78
0EE8 88 D0 F8 69 3F 20 D2 FF   10E8 A9 00 00 21 81 82 00 00
0EF0 CA D0 EC A9 20 2C A9 0D   10F0 00 00 59 4D 91 92 86 4A
0EF8 4C D2 FF 20 D4 FA 00 20   10F8 85 9D 2C 29 2C 23 28 24

0F00 69 FA 00 20 E5 FA 00 20   1100 59 00 00 58 24 24 00 00
0F08 69 FA 00 A2 00 00 86 28   1108 1C 8A 1C 23 5D 8B 1B A1
0F10 A9 90 20 D2 FF 20 57 FD   1110 9D 8A 1D 23 9D 8B 1D A1
0F18 00 20 72 FC 00 20 CA FC   1118 00 00 29 19 AE 69 A8 19
0F20 00 85 C1 84 C2 20 E1 FF   1120 23 24 53 1B 23 24 53 19
0F28 F0 05 20 2F FB 00 B0 E9   1128 A1 00 00 1A 5B 5B A5 69
0F30 4C 47 F8 00 20 D4 FA 00   1130 24 24 AE AE A8 AD 29 00
0F38 A9 03 85 1D 20 3E F8 00   1138 00 7C 00 00 15 9C 6D 9C
0F40 20 A1 F8 00 D0 F8 A5 20   1140 A5 69 29 53 84 13 34 11
0F48 85 C1 A5 21 85 C2 4C 46   1148 A5 69 23 A0 D8 62 5A 48
0F50 FC 00 C5 28 F0 03 20 D2   1150 26 62 94 88 54 44 C8 54
0F58 FF 60 20 D4 FA 00 20 69   1158 68 44 E8 94 00 00 B4 08
0F60 FA 00 8E 11 02 A2 03 20   1160 84 74 B4 28 6E 74 F4 CC
0F68 CC FA 00 48 CA D0 F9 A2   1168 4A 72 F2 A4 8A 00 00 AA
0F70 03 68 38 E9 3F A0 05 4A   1170 A2 A2 74 74 74 72 44 68
0F78 6E 11 02 6E 10 02 88 D0   1178 B2 32 B2 00 00 22 00 00

0F80 F6 CA D0 ED A2 02 20 CF   1180 1A 1A 26 26 72 72 88 C8
0F88 FF C9 0D F0 1E C9 20 F0   1188 C4 CA 26 48 44 44 A2 C8
0F90 F5 20 D0 FE 00 B0 0F 20   1190 3A 3B 52 4D 47 58 4C 53
0F98 9C FA 00 A4 C1 84 C2 85   1198 54 46 48 44 50 2C 41 42
0FA0 C1 A9 30 9D 10 02 E8 9D   11A0 F9 00 35 F9 00 CC F8 00
0FA8 10 02 E8 D0 DB 86 28 A2   11A8 F7 F8 00 56 F9 00 B9 F9
0FB0 00 00 86 26 F0 04 E6 26   11B0 00 F4 F9 00 0C FA 00 3E
0FB8 F0 75 A2 00 00 86 1D A5   11B8 FB 00 92 FB 00 C0 FB 00
0FC0 26 20 D9 FC 00 A6 2A 86   11C0 3B FC 00 5B FD 00 8A FD
0FC8 29 AA BC 37 FF 00 BD 77   11C8 00 AC FD 00 46 FB 00 FF
0FD0 FF 00 20 B9 FE 00 D0 E3   11D0 F7 00 ED F7 00 0D 20 20
0FD8 A2 06 E0 03 D0 19 A4 1F   11D8 20 50 43 20 20 53 52 20
0FE0 F0 15 A5 2A C9 E8 A9 30   11E0 41 43 20 58 52 20 59 52
0FE8 B0 21 20 BF FE 00 D0 CC   11E8 20 53 50 AA AA AA AA AA
0FF0 20 C1 FE 00 D0 C7 88 D0
0FF8 EB 06 2A 90 0B BC 30 FF
```

## Entering Extramon : Commodore 64 version

Use the program Tiny Peeker/Poker as follows.

Type POKE 8192,0:POKE44,32 (return)

Enter and run the peeker/poker program.

In response to the program prompts, type in the data as given in the following tables. When you've finished, type POKE44,8:POKE45,232:POKE46,17:CLR.

Save Extramon with a normal SAVE before attempting to run it.

Then type NEW. and use the following checksum program to enable you to identify and locate any errors:

```
100 REM EXTRAMON64 CHECKSUM PROGRAM
110 DATA10170,13676,15404,14997,15136,16221,16696
115 DATA12816,16228,14554
120 DATA14677,15039,14551,15104,15522,16414,15914
125 DATA8958,11945:S=204B
130 FORB=1TO19:READX:FORI=1TOS:N=PEEK(I):Y=Y+N
140 NEXTI:IFY<>XTHEN?"ERROR IN BLOCK "B:GOTO160
150 PRINT"BLOCK "B" CORRECT"
160 S=I:Y=0:NEXTB:REM CHECK LAST BLOCK BY HAND
```

This program must be run after the first set of POKEs, and before the second set.

If errors are found, type NEW (you won't lose Extramon!), re-load Tiny Peeker/Poker, enter block of memory again, type NEW, re-load checksum program, run it, if errors found type NEW and re-load Tiny Peeker/Poker, and so on until no errors remain. Then, issue last set of POKEs and SAVE Extramon!

## Extramon Instruction Set

This will be given in the form COMMAND, followed by the syntax.

1) Simple Assembler
```
.A 2000 LDA£12
```
start assembly at 2000 hex.

2) Disassembler
```
.D 2000
```
disassemble hex from 2000 onwards.

3) Printing Disassembler
```
.P 2000,2040
```
engage printer beforehand with OPEN4,4:CMD4.

4) Fill memory
```
.F 1000 1100 FF
```
fill memory from 1000 to 1100 hex with the byte FF.

5) Go run
```
.G 1000
```
go to hex 1000 and executre program there.

6) Hunt memory
```
.H C000 D000 'READ
```
look from C000 to D000 for the ASCII string READ.

7) Load
```
.L "FRED",08
```

8) Memory display
```
.M 0800 0820
```
display memory from hex 0800 to 0820.

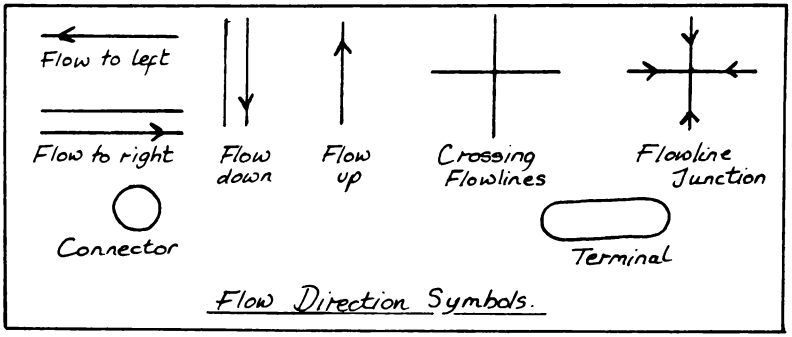
9) Register display
```
.R
```
displays register values when Extramon was entered.

10) Save
```
.S "0:FRED",08,0800,0820
```
save memory from hex 0800 to 0820 onto device 08 drive 1, and call that portion of memory FRED.

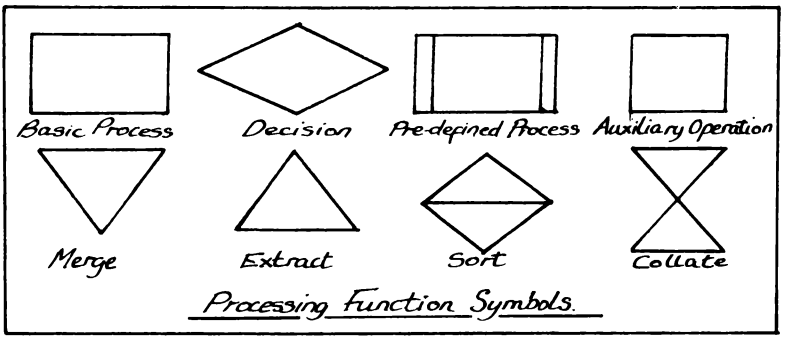
11) Transfer memory
.T 1000 1100 5000
transfer memory in the range hex 1000 to 1100 and start storing it at hex 5000 onwards.

12) Exit to Basic
.0
return to Basic ready mode. Perform a CLR before doing anything.

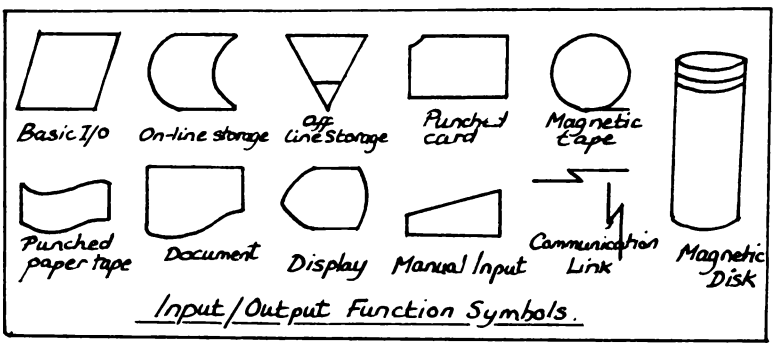
# Flow charting



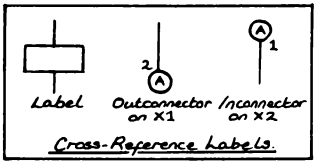
Flow Direction Symbols.



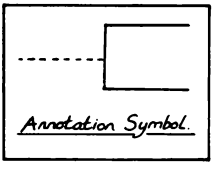
Processing Function Symbols.



Input/Output Function Symbols.



Cross-Reference Labels


Annotation Symbol.

# Hex/Dec convertor

Decimal & Hexadecimal Conversions

| HEXADECIMAL COLUMNS | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | | 5 | | 4 | | 3 | | 2 | | 1 | |
| HEX | DEC | HEX | DEC | HEX | DEC | HEX | DEC | HEX | DEC | HEX | DEC |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1,048,576 | 1 | 65,536 | 1 | 4,096 | 1 | 256 | 1 | 16 | 1 | 1 |
| 2 | 2,097,152 | 2 | 131,072 | 2 | 8,192 | 2 | 512 | 2 | 32 | 2 | 2 |
| 3 | 3,145,728 | 3 | 196,608 | 3 | 12,288 | 3 | 768 | 3 | 48 | 3 | 3 |
| 4 | 4,194,304 | 4 | 262,144 | 4 | 16,384 | 4 | 1,024 | 4 | 64 | 4 | 4 |
| 5 | 5,242,880 | 5 | 327,680 | 5 | 20,480 | 5 | 1,280 | 5 | 80 | 5 | 5 |
| 6 | 6,291,456 | 6 | 393,216 | 6 | 24,576 | 6 | 1,536 | 6 | 96 | 6 | 6 |
| 7 | 7,340,032 | 7 | 458,752 | 7 | 28,672 | 7 | 1,792 | 7 | 112 | 7 | 7 |
| 8 | 8,388,608 | 8 | 524,288 | 8 | 32,768 | 8 | 2,048 | 8 | 128 | 8 | 8 |
| 9 | 9,437,184 | 9 | 589,824 | 9 | 36,864 | 9 | 2,304 | 9 | 144 | 9 | 9 |
| A | 10,485,760 | A | 655,360 | A | 40,960 | A | 2,560 | A | 160 | A | 10 |
| B | 11,534,336 | B | 720,897 | B | 45,056 | B | 2,816 | B | 176 | B | 11 |
| C | 12,582,912 | C | 786,432 | C | 49,152 | C | 3,072 | C | 192 | C | 12 |
| D | 13,631,488 | D | 851,968 | D | 53,248 | D | 3,328 | D | 208 | D | 13 |
| E | 14,680,064 | E | 917,504 | E | 57,344 | E | 3,584 | E | 224 | E | 14 |
| F | 15,728,640 | F | 983,040 | F | 61,440 | F | 3,840 | F | 240 | F | 15 |

Notes.

To convert from hexadecimal to decimal, first find the corresponding column position for each hexadécimal digit. Make a note of the decimal equivalents, then add the noted values together to obtain the converted decimal value.

To convert from decimal to hexadecimal, find the largest decimal value in the table that will fit into the number to be converted. Next make a note of the hex equivalent and column position. Calculate the decimal remainder, and repeat the process on this and any subsequent remainders.

# Hyperbolic functions

| FUNCTION | BASIC EQUIVALENT |
|---|---|
| SECANT | SEC(X)= 1/COS(X) |
| COSECANT | CSC(X)= 1/SIN(X) |
| COTANGENT | COT(X)= 1/TAN(X) |
| INVERSE SINE | ARCSIN(X)=ATN(X/SQR(−X*X+1)) |
| INVERSE COSINE | ARCCOS(X)=−ATN(X/SQR (−X*X +1)) +π/2 |
| INVERSE SECANT | ARCSEC(X)=ATN(X/SQR(X*X−1)) |
| INVERSE COSECANT | ARCCSC(X)=ATN(X/SQR(X*X−1)) +(SGN(X)−1*π/2 |
| INVERSE COTANGENT | ARCOT(X)=ATN(X)+π/2 |
| HYPERBOLIC SINE | SINH(X)=(EXP(X)−EXP(−X))/2 |
| HYPERBOLIC COSINE | COSH(X)=(EXP(X)+EXP(−X))/2 |
| HYPERBOLIC TANGENT | TANH(X)=EXP(−X)/(EXP(x)+EXP (−X))*2+1 |
| HYPERBOLIC SECANT | SECH(X)=2/(EXP(X)+EXP(−X)) |
| HYPERBOLIC COSECANT | CSCH(X)=2/(EXP(X)−EXP(−X)) |
| HYPERBOLIC COTANGENT | COTH(X)=EXP(−X)/(EXP(X) −EXP(−X))*2+1 |
| INVERSE HYPERBOLIC SINE | ARCSINH(X)=LOG(X+SQR(X*X+1)) |
| INVERSE HYPERBOLIC COSINE | ARCCOSH(X)=LOG(X+SQR(X*X−1)) |
| INVERSE HYPERBOLIC TANGENT | ARCTANH(X)=LOG((1+X)/(1−X))/2 |
| INVERSE HYPERBOLIC SECANT | ARCSECH(X)=LOG((SQR (−X*X+1)+1/X) |
| INVERSE HYPERBOLIC COSECANT | ARCCSCH(X)=LOG((SGN(X)*SQR (X*X+1/x) |
| INVERSE HYPERBOLIC COTAN-GENT | ARCCOTH(X)=LOG((X+1)/(x−1))/2 |

# Memory maps

| | | | |
|---|---|---|---|
| 0000 | | 0 | Chip directional register |
| 0001 | | 1 | Chip I/O; memory & tape control |
| 0003 | -0004 | 3-4 | Float-Fixed vector |
| 0005 | -0006 | 5-6 | Fixed-Float vector |
| 0007 | | 7 | Search character |
| 0008 | | 8 | Scan-quotes flag |
| 0009 | | 9 | TAB column save |
| 000A | | 10 | 0 = LOAD, 1 = VERIFY |
| 000B | | 11 | Input buffer pointer/# subscrpt |
| 000C | | 12 | Default DIM flag |
| 000D | | 13 | Type: FF = string, 00 = numeric |
| 000E | | 14 | Type: 80 = integer, 00 = floating point |
| 000F | | 15 | DATA scan/LIST quote/memry flag |
| 0010 | | 16 | Subscript/FNx flag |
| 0011 | | 17 | 0 = INPUT;$40 = GET;$98 = READ |
| 0012 | | 18 | ATN sign/Comparison eval flag |
| 0013 | | 19 | Current I/O prompt flag |
| 0014 | -0015 | 20-21 | Integer value |
| 0016 | | 22 | Pointer: temporary string stack |
| 0017 | -0018 | 23-24 | Last temp string vector |
| 0019 | -0021 | 25-33 | Stack for temporary strings |
| 0022 | -0025 | 34-37 | Utility pointer area |
| 0026 | -002A | 38-42 | Product area for multiplication |
| 002B | -002C | 43-44 | Pointer: Start-of-Basic |
| 002D | -002E | 45-46 | Pointer: Start-of-Variables |
| 002F | -0030 | 47-48 | Pointer: Start-of-Arrays |
| 0031 | -0032 | 49-50 | Pointer: End-of-Arrays |
| 0033 | -0034 | 51-52 | Pointer: String-storage(moving down) |
| 0035 | -0036 | 53-54 | Utility string pointer |
| 0037 | -0038 | 55-56 | Pointer: Limit-of-memory |
| 0039 | -003A | 57-58 | Current Basic line number |
| 003B | -003C | 59-60 | Previous Basic line number |
| 003D | -003E | 61-62 | Pointer: Basic statement for CONT |
| 003F | -0040 | 63-64 | Current DATA line number |
| 0041 | -0042 | 65-66 | Current DATA address |
| 0043 | -0044 | 67-68 | Input vector |
| 0045 | -0046 | 69-70 | Current variable name |
| 0047 | -0048 | 71-72 | Current variable address |
| 0049 | -004A | 73-74 | Variable pointer for FOR/NEXT |
| 004B | -004C | 75-76 | Y-save; op-save; Basic pointer save |
| 004D | | 77 | Comparison symbol accumulator |
| 004E | -0053 | 78-83 | Misc work area, pointers, etc |
| 0054 | -0056 | 84-86 | Jump vector for functions |
| 0057 | -0060 | 87-96 | Misc numeric work area |
| 0061 | | 97 | Accum#1: Exponent |
| 0062 | -0065 | 98-101 | Accum#1: Mantissa |
| 0066 | | 102 | Accum#1: Sign |
| 0067 | | 103 | Series evaluation constant pointer |
| 0068 | | 104 | Accum#1 hi-order (overflow) |
| 0069 | -006E | 105-110 | Accum#2: Exponent, etc. |
| 006F | | 111 | Sign comparison, Acc#1 vs #2 |

| | | | |
|---|---|---|---|
| 0070 | | 112 | Accum#1 lo-order (rounding) |
| 0071 | -0072 | 113-114 | Cassette buff len/Series pointer |
| 0073 | -008A | 115-138 | CHRGET subroutine; get Basic char |
| 007A | -007B | 122-123 | Basic pointer (within subrtn) |
| 008B | -008F | 139-143 | RND seed value |
| 0090 | | 144 | Status word ST |
| 0091 | | 145 | Keyswitch PIA: STOP and RVS flags |
| 0092 | | 146 | Timing constant for tape |
| 0093 | | 147 | Load = 0, Verify = 1 |
| 0094 | | 148 | Serial output: deferred char flag |
| 0095 | | 149 | Serial deferred character |
| 0096 | | 150 | Tape EOT received |
| 0097 | | 151 | Register save |
| 0098 | | 152 | How many open files |
| 0099 | | 153 | Input device, normally 0 |
| 009A | | 154 | Output CMD device, normally 3 |
| 009B | | 155 | Tape character parity |
| 009C | | 156 | Byte-received flag |
| 009D | | 157 | Direct = $80/RUN = 0 output control |
| 009E | | 158 | Tp Pass 1 error log/char buffer |
| 009F | | 159 | Tp Pass 2 err log corrected |
| 00A0 | -00A2 | 160-162 | Jiffy Clock HML |
| 00A3 | | 163 | Serial bit count/EOI flag |
| 00A4 | | 164 | Cycle count |
| 00A5 | | 165 | Countdown,tape write/bit count |
| 00A6 | | 166 | Tape buffer pointer |
| 00A7 | | 167 | Tp Wrt ldr count/Rd pass/inbit |
| 00A8 | | 168 | Tp Wrt new byte/Rd error/inbit cnt |
| 00A9 | | 169 | Wrt start bit/Rd bit err/stbit |
| 00AA | | 170 | Tp Scan;Cnt;Ld;End/byte assy |
| 00AB | | 171 | Wr lead length/Rd checksum/parity |
| 00AC | -00AD | 172-173 | Pointer: tape bufr, scrolling |
| 00AE | -00AF | 174-175 | Tape end adds/End of program |
| 00B0 | -00B1 | 176-177 | Tape timing constants |
| 00B2 | -00B3 | 178-179 | Pntr: start of tape buffer |
| 00B4 | | 180 | 1 = Tp timer enabled; bit count |
| 00B5 | | 181 | Tp EOT/RS232 next bit to send |
| 00B6 | | 182 | Read character error/outbyte buf |
| 00B7 | | 183 | # characters in file name |
| 00B8 | | 184 | Current logical file |
| 00B9 | | 185 | Current secndy address |
| 00BA | | 186 | Current device |
| 00BB | -00BC | 187-188 | Pointer to file name |
| 00BD | | 189 | Wr shift word/Rd input char |
| 00BE | | 190 | # blocks remaining to Wr/Rd |
| 00BF | | 191 | Serial word buffer |
| 00C0 | | 192 | Tape motor interlock |
| 00C1 | -00C2 | 193-194 | I/O start address |
| 00C3 | -00C4 | 195-196 | Kernel setup pointer |
| 00C5 | | 197 | Last key pressed |
| 00C6 | | 198 | # chars in keybd buffer |
| 00C7 | | 199 | Screen reverse flag |
| 00C8 | | 200 | End-of-line for input pointer |
| 00C9 | -00CA | 201-202 | Input cursor log (row, column) |
| 00CB | | 203 | Which key: 64 if no key |

| | | | |
|---|---|---|---|
| 00CC | | 204 | 0 = flash cursor |
| 00CD | | 205 | Cursor timing countdown |
| 00CE | | 206 | Character under cursor |
| 00CF | | 207 | Cursor in blink phase |
| 00D0 | | 208 | Input from screen/from keyboard |
| 00D1 | -00D2 | 209-210 | Pointer to screen line |
| 00D3 | | 211 | Position of cursor on above line |
| 00D4 | | 212 | 0 = direct cursor, else programmed |
| 00D5 | | 213 | Current screen line length |
| 00D6 | | 214 | Row where curosr lives |
| 00D7 | | 215 | Last inkey/checksum/buffer |
| 00D8 | | 216 | # of INSERTs outstanding |
| 00D9 | -00F2 | 217-242 | Screen line link table |
| 00F3 | -00F4 | 243-244 | Screen color pointer |
| 00F5 | -00F6 | 245-246 | Keyboard pointer |
| 00F7 | -00F8 | 247-248 | RS-232 Rcv pntr |
| 00F9 | -00FA | 249-250 | RS-232 Tx pntr |
| 00FF | -010A | 256-266 | Floating to ASCII work area |
| 0100 | -013E | 256-318 | Tape error log |
| 0100 | -01FF | 256-511 | Processor stack area |
| 0200 | -0258 | 512-600 | Basic input buffer |
| 0259 | -0262 | 601-610 | Logical file table |
| 0263 | -026C | 611-620 | Device # table |
| 026D | -0276 | 621-630 | Sec Adds table |
| 0277 | -0280 | 631-640 | Keybd buffer |
| 0281 | -0282 | 641-642 | Start of Basic Memory |
| 0283 | -0284 | 643-644 | Top of Basic Memory |
| 0285 | | 645 | Serial bus timeout flag |
| 0286 | | 646 | Current color code |
| 0287 | | 647 | Color under cursor |
| 0288 | | 648 | Screen memory page |
| 0289 | | 649 | Max size of keybd buffer |
| 028A | | 650 | Repeat all keys |
| 028B | | 651 | Repeat speed counter |
| 028C | | 652 | Repeat delay counter |
| 028D | | 653 | Keyboard Shift/Control flag |
| 028E | | 654 | Last shift pattern |
| 028F | -0290 | 655-656 | Keyboard table setup pointer |
| 0291 | | 657 | Keyboard shift mode |
| 0292 | | 658 | 0 = scroll enable |
| 0293 | | 659 | RS-232 control reg |
| 0294 | | 660 | RS-232 command reg |
| 0295 | -0296 | 661-662 | Bit timing |
| 0297 | | 663 | RS-232 status |
| 0298 | | 664 | # bits to send |
| 0299 | -029A | 665 | RS-232 speed/code |
| 029B | | 667 | RS232 receive pointer |
| 029C | | 668 | RS232 input pointer |
| 029D | | 669 | RS232 transmit pointer |
| 029E | | 670 | RS232 output pointer |
| 029F | -02A0 | 671-672 | IRQ save during tape I/O |
| 02A1 | | 673 | CIA 2 (NMI) Interrupt Control |
| 02A2 | | 674 | CIA 1 Timer A control log |
| 02A3 | | 675 | CIA 1 Interrupt Log |
| 02A4 | | 676 | CIA 1 Timer A enabled flag |

| | | | | |
|---|---|---|---|---|
| 02A5 | | 677 | Screen row marker | |
| 02C0 | -02FE | 704-766 | (Sprite 11) | |
| 0300 | -0301 | 768-769 | Error message link | |
| 0302 | -0303 | 770-771 | Basic warm start link | |
| 0304 | -0305 | 772-773 | Crunch Basic tokens link | |
| 0306 | -0307 | 774-775 | Print tokens link | |
| 0308 | -0309 | 776-777 | Start new Basic code link | |
| 030A | -030B | 778-779 | Get arithmetic element link | |
| 030C | | 780 | SYS A-reg save | |
| 030D | | 781 | SYS X-reg save | |
| 030E | | 782 | SYS Y-reg save | |
| 030F | | 783 | SYS status reg save | |
| 0310 | -0312 | 784-785 | USR function jump | (B248) |
| 0314 | -0315 | 788-789 | Hardware interrupt vector | (EA31) |
| 0316 | -0317 | 790-791 | Break interrupt vector | (FE66) |
| 0318 | -0319 | 792-793 | NMI interrupt vector | (FE47) |
| 031A | -031B | 794-795 | OPEN vector | (F34A) |
| 031C | -031D | 796-797 | CLOSE vector | (F291) |
| 031E | -031F | 798-799 | Set-input vector | (F20E) |
| 0320 | -0321 | 800-801 | Set-output vector | (F250) |
| 0322 | -0323 | 802-803 | Restore I/O vector | (F333) |
| 0324 | -0325 | 804-805 | INPUT vector | (F157) |
| 0326 | -0327 | 806-807 | Output vector | (F1CA) |
| 0328 | -0329 | 808-809 | Test-STOP vector | (F6ED) |
| 032A | -032B | 810-811 | GET vector | (F13E) |
| 032C | -032D | 812-813 | Abort I/O vector | (F32F) |
| 032E | -032F | 814-815 | Warm start vector | (FE66) |
| 0330 | -0331 | 816-817 | LOAD link | (F4A5) |
| 0332 | -0333 | 818-819 | SAVE link | (F5ED) |
| 033C | -03FB | 828-1019 | Cassette buffer | |
| 0340 | -037E | 832-894 | (Sprite 13) | |
| 0380 | -03BE | 896-958 | (Sprite 14) | |
| 03C0 | -03FE | 960-1022 | (Sprite 15) | |
| 0400 | -07FF | 1024-2047 | Screen memory | |
| 0800 | -9FFF | 2048-40959 | Basic RAM memory | |
| 8000 | -9FFF | 32768-40959 | Alternate: ROM plug-in area | |
| A000 | -BFFF | 40960-49151 | ROM: Basic | |
| A000 | -BFFF | 49060-59151 | Alternate: RAM | |
| C000 | -CFFF | 49152-53247 | RAM memory, including alternate | |
| D000 | -D02E | 53248-53294 | Video Chip (6566) | |
| D400 | -D41C | 54272-54300 | Sound Chip (6581 SID) | |
| D800 | -DBFF | 55296-56319 | Color nybble memory | |
| DC00 | -DC0F | 56320-56335 | Interface chip 1, IRQ (6526 CIA) | |
| DD00 | -DD0F | 56576-56591 | Interface chip 2, NMI (6526 CIA) | |
| D000 | -DFFF | 53248-53294 | Alternate: Character set | |
| E000 | -FFFF | 57344-65535 | ROM: Operating System | |
| E000 | -FFFF | 57344-65535 | Alternate: RAM | |
| FF81 | -FFF5 | 65409-65525 | Jump Table, Including: | |
| FFC6 | | | - Set Input channel | |
| FFC9 | | | - Set Output channel | |
| FFCC | | | - Restore default I/O channels | |
| FFCF | | | - INPUT | |
| FFD2 | | | - PRINT | |
| FFE1 | | | - Test Stop key | |
| FFE4 | | | - GET | |

A000; ROM control vectors
A00C; Keyword action vectors
A052; Function vectors
A080; Operator vectors
A09E; Keywords
A19E; Error messages
A328; Error message vectors
A365; Misc messages
A38A; Scan stack for FOR/GOSUB
A3B8; Move memory
A3FB; Check stack depth
A408; Check memory space
A435; 'out of memory'
A437; Error routine
A469; BREAK entry
A474; 'ready.'
A480; Ready for Basic
A49C; Handle new line
A533; Re-chain lines
A560; Receive input line
A579; Crunch tokens
A613; Find Basic line
A642; Perform [NEW]
A65E; Perform [CLR]
A68E; Back up text pointer
A69C; Perform [LIST]
A742; Perform [FOR]
A7ED; Execute statement
A81D; Perform [RESTORE]
A82C; Break
A82F; Perform [STOP]
A831; Perform [END]
A857; Perform [CONT]
A871; Perform [RUN]
A883; Perform [GOSUB]
A8A0; Perform [GOTO]
A8D2; Perform [RETURN]
A8F8; Perform [DATA]
A906; Scan for next statement
A928; Perform [IF]
A93B; Perform [REM]
A94B; Perform [ON]
A96B; Get fixed point number
A9A5; Perform [LET]
AA80; Perform [PRINT#]
AA86; Perform [CMD]
AAA0; Perform [PRINT]
AB1E; Print string from (y.a)
AB3B; Print format character
AB4D; Bad input routine
AB7B; Perform [GET]
ABA5; Perform [INPUT#]
ABBF; Perform [INPUT]
ABF9; Prompt & input
AC06; Perform [READ]
ACFC; Input error messages
AD1E; Perform [NEXT]
AD78; Type match check
AD9E; Evaluate expression
AEA8; Constant - pi
AEF1; Evaluate within brackets
AEF7; ')'
AEFF; comma..
AF08; Syntax error
AF14; Check range
AF28; Search for variable
AFA7; Setup FN reference
AFE6; Perform [OR]
AFE9; Perform [AND]
B016; Compare
B081; Perform [DIM]
B08B; Locate variable
B113; Check alphabetic
B11D; Create variable
B194; Array pointer subrtine
B1A5; Value 32768
B1B2; Float-fixed
B1D1; Set up array
B245; 'bad subscript'
B248; 'illegal quantity'
B34C; Compute array size
B37D; Perform [FRE]
B391; Fix-float
B39E; Perform [POS]
B3A6; Check direct
B3B3; Perform [DEF]
B3E1; Check fn syntax
B3F4; Perform [FN]
B465; Perform [STR$]
B475; Calculate string vector
B487; Set up string
B4F4; Make room for string
B526; Garbage collection
B5BD; Check salvageability
B606; Collect string
B63D; Concatenate
B67A; Build string to memory
B6A3; Discard unwanted string
B6DB; Clean descriptor stack
B6EC; Perform [CHR$]
B700; Perform [LEFT$]
B72C; Perform [RIGHT$]
B737; Perform [MID$]
B761; Pull string parameters
B77C; Perform [LEN]
B782; Exit string-mode
B78B; Perform [ASC]
B79B; Input byte paramter
B7AD; Perform [VAL]
B7EB; Parameters for POKE/WAIT
B7F7; Float-fixed
B80D; Perform [PEEK]
B824; Perform [POKE]
B82D; Perform [WAIT]

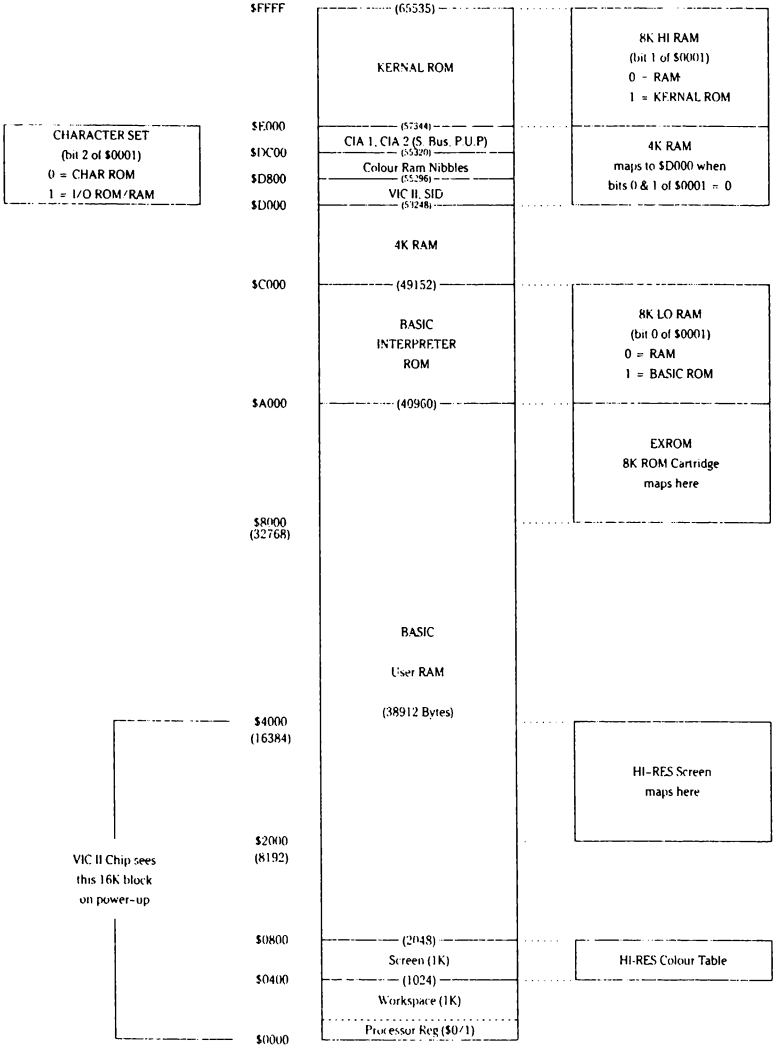
B849; Add 0.5
B850; Subtract-from
B853; Perform [subtract]
B86A; Perform [add]
B947; Complement FAC*1
B97E; 'overflow'
B983; Multiply by zero byte
B9EA; Perform [LOG]
BA2B; Perform [multiply]
BA59; Multiply-a-bit
BA8C; Memory to FAC*2
BAB7; Adjust FAC*1/*2
BAD4; Underflow/overflow
BAE2; Multiply by 10
BAF9; + 10 in floating pt
BAFE; Divide by 10
BB12; Perform [divide]
BBA2; Memory to FAC*1
BBC7; FAC*1 to memory
BBFC; FAC*2 to FAC*1
BC0C; FAC*1 to FAC*2
BC1B; Round FAC*1
BC2B; Get sign
BC39; Perform [SGN]
BC58; Perform [ABS]
BC5B; Compare FAC*1 to mem
BC9B; Float-fixed
BCCC; Perform [int]
BCF3; String to FAC
BD7E; Get ascii digit
BDC2; Print 'IN..'
BDCD; Print line number
BDDD; Float to ascii
BF16; Decimal constants
BF3A; TI constants
BF71; Perform [SQR]
BF7B; Perform [power]
BFB4; Perform [negative]
BFED; Perform [EXP]
E043; Series eval 1
E059; Series eval 2
E097; Perform [RND]
E0f9; ?? breakpoints ??
E12A; Perform [SYS]
E156; Perform [SAVE]
E165; Perform [VERIFY]
E168; Perform [LOAD]
E1BE; Perform [OPEN]
E1C7; Perform [CLOSE]
E1D4; Parameters for LOAD/SAVE
E206; Check default parameters
E20E; Check for comma
E219; Parameters for open/close
E264; Perform [COS]
E26B; Perform [SIN]
E2B4; Perform [TAN]
E30E; Perform [ATN]
E37B; Warm restart

E394; Initialize
E3A2; CHRGET for zero page
E3BF; Initialize Basic
E447; Vectors for $300
E453; Initialize vectors
E45F; Power-up message
E500; Get I/O address
E505; Get screen size
E50A; Put/get row/column
E518; Initialize I/O
E544; Clear screen
E566; Home cursor
E56C; Set screen pointers
E5A0; Set I/O defaults
E5B4; Input from keyboard
E632; Input from screen
E684; Quote test
E691; Setup screen print
E6B6; Advance cursor
E6ED; Retreat cursor
E701; Back into previous line
E716; Output to screen
E87C; Go to next line
E891; Perform <return>
E8A1; Check line decrement
E8B3; Check line increment
E8CB; Set color code
E8DA; Color code table
E8EA; Scroll screen
E965; Open space on screen
E9C8; Move a screen line
E9E0; Synchronize color transfer
E9F0; Set start-of-line
E9FF; Clear screen line
EA13; Print to screen
EA24; Synchronize color pointer
EA31; Interrupt - clock etc
EA87; Read keyboard
EB79; Keyboard select vectors
EB81; Keyboard 1 - unshifted
EBC2; Keyboard 2 - shifted
EC03; Keyboard 3 - 'comm'
EC44; Graphics/text contrl
EC4F; Set graphics/text mode
EC78; Keyboard 4
ECB9; Video chip setup
ECE7; Shift/run equivalent
ECF0; Screen ln address low
ED09; Send 'talk'
ED0C; Send 'listen'
ED40; Send to serial bus
EDB2; Serial timeout
EDB9; Send listen SA
EDBE; Clear ATN
EDC7; Send talk SA
EDCC; Wait for clock
EDDD; Send serial deferred
EDEF; Send 'untalk'

EDFE; Send 'unlisten'
EE13; Receive from serial bus
EE85; Serial clock on
EE8E; Serial clock off
EE97; Serial output '1'
EEA0; Serial output '0'
EEA9; Get serial in & clock
EEB3; Delay 1 ms
EEBB; RS-232 send
EF06; Send new RS-232 byte
EF2E; No-DSR error
EF31; No-CTS error
EF3B; Disable timer
EF4A; Compute bit count
EF59; RS232 receive
EF7E; Setup to receive
EFC5; Receive parity error
EFCA; Receive overflow
EFCD; Receive break
EFD0; Framing error
EFE1; Submit to RS232
F00D; No-DSR error
F017; Send to RS232 buffer
F04D; Input from RS232
F086; Get from RS232
F0A4; Check serial bus idle
F0BD; Messages
F12B; Print if direct
F13E; Get..
F14E; ..from RS232
F157; Input
F199; Get.. tape/serial/rs232
F1CA; Output..
F1DD; ..to tape
F20E; Set input device
F250; Set output device
F291; Close file
F30F; Find file
F31F; Set file values
F32F; Abort all files
F333; Restore default I/O
F34A; Do file open
F3D5; Send SA
F409; Open RS232
F49E; Load program
F5AF; 'searching'
F5C1; Print filename
F5D2; 'loading/verifying'
F5DD; Save program
F68F; Print 'saving'
F69B; Bump clock
F6BC; Log PIA key reading
F6DD; Get time
F6E4; Set time
F6ED; Check stop key
F6FB; Output error messages
F72D; Find any tape headr
F76A; Write tape header
F7D0; Get buffer address
F7D7; Set buffer start/end pointers
F7EA; Find specific header
F80D; Bump tape pointer
F817; 'press play..'
F82E; Check tape status
F838; 'press record..'
F841; Initiate tape read
F864; Initiate tape write
F875; Common tape code
F8D0; Check tape stop
F8E2; Set read timing
F92C; Read tape bits
FA60; Store tape chars
FB8E; Reset pointer
FB97; New character setup
FBA6; Send transition to tape
FBC8; Write data to tape
FBCD; IRQ entry point
FC57; Write tape leader
FC93; Restore normal IRQ
FCB8; Set IRQ vector
FCCA; Kill tape motor
FCD1; Check r/w pointer
FCDB; Bump r/w pointer
FCE2; Power reset entry
FD02; Check 8-rom
FD10; 8-rom mask
FD15; Kernal reset
FD1A; Kernal move
FD30; Vectors
FD50; Initialize system constnts
FD9B; IRQ vectors
FDA3; Initialize I/O
FDDD; Enable timer
FDF9; Save filename data
FE00; Save file details
FE07; Get status
FE18; Flag status
FE1C; Set status
FE21; Set timeout
FE25; Read/set top of memory
FE27; Read top of memory
FE2D; Set top of memory
FE34; Read/set bottom of memory
FE43; NMI entry
FE66; Warm start
FEB6; Reset IRQ & exit
FEBC; Interrupt exit
FEC2; RS-232 timing table
FED6; NMI RS-232 in
FF07; NMI RS-232 out
FF43; Fake IRQ
FF48; IRQ entry
FF81; Jumbo jump table
FFFA; Hardware vectors

# Memory architecture

## Commodore-64 Architecture Map



## Processor I/O Port (6510)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $0000 | IN | IN | OUT | IN | OUT | OUT | OUT | OUT | DDR 0 |
| $0001 | | | Tape Motor | Tape Sense | Tape Write | D-ROM Switch | EF RAM Switch | AB RAM Switch | PR 1 |

## SID (6581)

| Voice 1 | Voice 2 | Voice 3 | | Voice 1 | Voice 2 | Voice 3 |
|---|---|---|---|---|---|---|
| $D400 | $D407 | $D40E | Frequency L | 54272 | 54279 | 54286 |
| $D401 | $D408 | $D40F | Frequency H | 54273 | 54280 | 54287 |
| $D402 | $D409 | $D410 | Pulse Width L | 54274 | 54281 | 54288 |
| $D403 | $D40A | $D411 | 0 0 0 0 H | 54275 | 54282 | 54289 |
| $D404 | $D40B | $D412 | Voice Type: NSE PUL SAW TRI Key | 54276 | 54283 | 54290 |
| $D405 | $D40C | $D413 | Attack Time 2ms - 8ms / Decay Time 6ms - 24 sec | 54277 | 54284 | 54291 |
| $D406 | $D40D | $D414 | Sustain Level / Release Time 6ms - 24 sec | 54278 | 54285 | 54292 |

Voices (write only)

| | | |
|---|---|---|
| $D415 | 0 0 0 0 0 L | 54293 |
| $D416 | Filter Frequency H | 54294 |
| $D417 | Resonance / Filter Voices: Ext V3 V2 V1 | 54295 |
| $D418 | Passband: V3 off HI BP LO / Master Volume | 54296 |

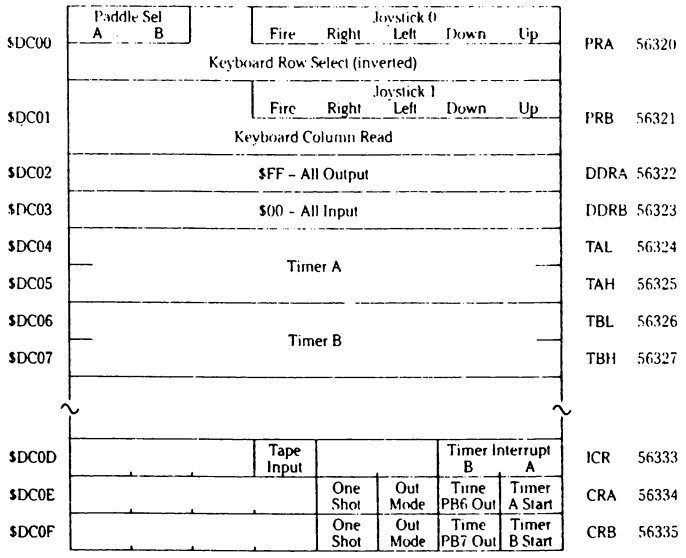
Filter & Volume (write only)

| | | |
|---|---|---|
| $D419 | Paddle X (A/D #1) | 54297 |
| $D41A | Paddle Y (A/D #2) | 54298 |
| $D41B | Noise 3 (random) | 54299 |
| $D41C | Envelope 3 | 54300 |

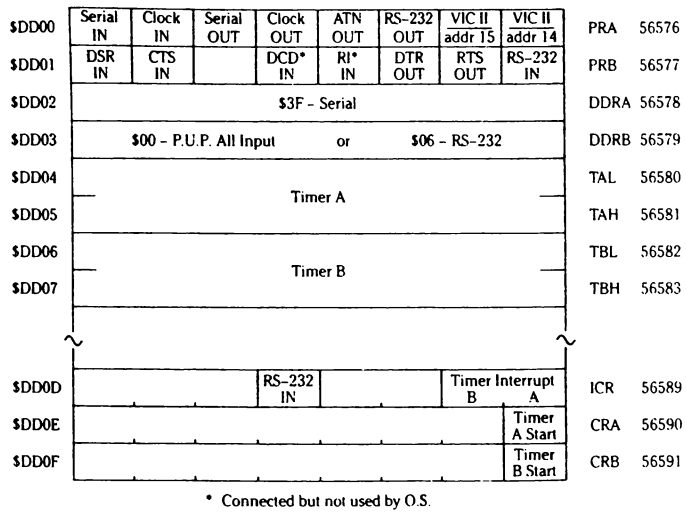
Sense (read only)

Note: Special Voice Features
(TEST, RING MOD, SYNC)
are omitted from the above diagram.

## CIA 1 (IRQ) (6526)




## CIA 2 (NMI) (6526)




* Connected but not used by O.S.

# M/C instruction set

The following notation applies to this summary:

| | |
|---|---|
| A | Accumulator |
| X, Y | Index registers |
| M | Memory |
| P | Processor status register |
| S | Stack Pointer |
| √ | Change |
| – | No change |
| + | Add |
| ∧ | Logical AND |
| − | Subtract |
| V | Logical Exclusive-OR |
| →,← | Transfer to |
| ∀ | Logical (inclusive) OR |
| PC | Program counter |
| PCH | Program counter high |
| PCL | Program counter low |
| #dd | 8-bit immediate data value (2 hexadecimal digits) |
| aa | 8-bit zero page address (2 hexadecimal digits) |
| aaaa | 16-bit absolute address (4 hexadecimal digits) |
| ↑ | Transfer from stack (Pull) |
| ↓ | Transfer onto stack (Push) |

# ADC

## *Add to Accumulator with Carry*

Operation: A + M + C → A, C

N Z C I D V
√ √ √ – – √

| Addressing Mode | Assembly Language Form | | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|---|
| Immediate | ADC | #dd | 69 | 2 | 2 |
| Zero Page | ADC | aa | 65 | 2 | 3 |
| Zero Page, X | ADC | aa,X | 75 | 2 | 4 |
| Absolute | ADC | aaaa | 6D | 3 | 4 |
| Absolute, X | ADC | aaaa,X | 7D | 3 | 4* |
| Absolute, Y | ADC | aaaa,Y | 79 | 3 | 4* |
| (Indirect, X) | ADC | (aa,X) | 61 | 2 | 6 |
| (Indirect), Y | ADC | (aa),Y | 71 | 2 | 5* |

*Add 1 if page boundary is crossed.

# AND

## *AND Memory with Accumulator*

Logical AND to the accumulator
Operation: A ∧ M → A

N Z C I D V
√ √ – – – –

| Addressing Mode | Assembly Language Form | | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|---|
| Immediate | AND | #dd | 29 | 2 | 2 |
| Zero Page | AND | aa | 25 | 2 | 3 |
| Zero Page, X | AND | aa,X | 35 | 2 | 4 |
| Absolute | AND | aaaa | 2D | 3 | 4 |
| Absolute, X | AND | aaaa,X | 3D | 3 | 4* |
| Absolute, Y | AND | aaaa,Y | 39 | 3 | 4* |
| (Indirect, X) | AND | (aa,X) | 21 | 2 | 6 |
| (Indirect), Y | AND | (aa),Y | 31 | 2 | 5* |

*Add 1 if page boundary is crossed.

# ASL

## *Accumulator Shift Left*

Operation: C ← [7|6|5|4|3|2|1|0] ← 0

N Z C I D V
√ √ √ – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Accumulator | ASL A | 0A | 1 | 2 |
| Zero Page | ASL aa | 06 | 2 | 5 |
| Zero Page, X | ASL aa,X | 16 | 2 | 6 |
| Absolute | ASL aaaa | 0E | 3 | 6 |
| Absolute, X | ASL aaaa,X | 1E | 3 | 7 |

# BCC

## *Branch on Carry Clear*

Operation: Branch on C = 0

N Z C I D V
– – – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Relative | BCC aa | 90 | 2 | 2* |

*Add 1 if branch occurs to same page.
Add 2 if branch occurs to different page.
Note: AIM 65 will accept an absolute address as the operand (instruction format BCC aaaa), and convert it to a relative address.

# BCS

## *Branch on Carry Set*

Operation: Branch on C = 1

N Z C I D V
– – – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Relative | BCS aa | B0 | 2 | 2* |

*Add 1 if branch occurs to same page.
Add 2 if branch occurs to next page.
Note: AIM 65 will accept an absolute address as the operand (instruction format BCS aaaa), and convert it to a relative address.

## BEQ

### *Branch on Result Equal to Zero*

Operation: Branch on Z = 1

N Z C I D V
– – – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Relative | BEQ aa | F0 | 2 | 2* |

*Add 1 if branch occurs to same page.
Add 2 if branch occurs to next page.
Note: AIM 65 will accept an absolute address as the operand (instruction format BEQ aaaa), and convert it to a relative address.

## BIT

### *Test Bits in Memory with Accumulator*

Operation: A M, $M_7 \rightarrow N$, $M_6 \rightarrow V$
Bit 6 and 7 are transferred to the Status Register. If the result of A M is zero then Z = 1, otherwise Z = 0

N Z C I D V
$M_7$ √ – – – $M_6$

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Zero Page | BIT aa | 24 | 2 | 3 |
| Absolute | BIT aaaa | 2C | 3 | 4 |

## BMI

### *Branch on Result Minus*

Operation: Branch on N = 1

N Z C I D V
– – – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Relative | BMI aa | 30 | 2 | 2* |

*Add 1 if branch occurs to same page.
Add 2 if branch occurs to different page.
Note: AIM 65 will accept an absolute address as the operand (instruction format BMI aaaa), and convert it to a relative address.

## BNE

*Branch on Result Not Equal to Zero*

Operation: Branch on Z = 0

N Z C I D V
– – – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Relative | BNE aa | D0 | 2 | 2* |

*Add 1 if branch occurs to same page.
Add 2 if branch occurs to different page.
Note: AIM 65 will accept an absolute address as the operand (instruction format BNE aaaa), and convert it to a relative address.

## BPL

*Branch on Result Plus*

Operation: Branch on N = 0

N Z C I D V
– – – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Relative | BPL aa | 10 | 2 | 2* |

*Add 1 if branch occurs to same page.
Add 2 if branch occurs to different page.
Note: AIM 65 will accept an absolute address as the operand (instruction format BPL aaaa), and convert it to a relative address.

## BRK

*Force Break*

Operation: Forced Interrupt PC + 2 ↓ P ↓

B N Z C I D V
1 – – – 1 – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | BRK | 00 | 1 | 7 |

## BVC

*Branch on Overflow Clear*

Operation: Branch on V = 0

N Z C I D V
– – – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Relative | BVC aa | 50 | 2 | 2* |

*Add 1 if branch occurs to same page.
Add 2 if branch occurs to different page.
Note: AIM 65 will accept an absolute address as the operand (instruction format BVC aaaa), and convert it to a relative address.

## BVS

*Branch on Overflow Set*

Operation: Branch on V = 1

N Z C I D V
– – – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Relative | BVS aa | 70 | 2 | 2* |

*Add 1 if branch occurs to same page.
Add 2 if branch occurs to different page.
Note: AIM 65 will accept an absolute address as the operand (instruction format BVS aaaa), and convert it to a relative address.

## CLC

*Clear Carry Flag*

Operation: 0 → C

N Z C I D V
– – 0 – – –

| Addressing Mode | Assembly Language Form | OP CODE | No Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | CLC | 18 | 1 | 2 |

## CLD

*Clear Decimal Mode*

Operation: 0 → D

N Z C I D V
– – – – 0 –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | CLD | D8 | 1 | 2 |

## CLI

*Clear Interrupt Disable Bit*

Operation: 0 → I

N Z C I D V
– – – 0 – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | CLI | 58 | 1 | 2 |

## CLV

*Clear Overflow Flag*

Operation: 0 → V

N Z C I D V
– – – – – 0

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | CLV | B8 | 1 | 2 |

## CMP

*Compare Memory and Accumulator*

Operation: A — M

N Z C I D V
√ √ √ — — —

| Addressing Mode | Assembly Language Form | | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|---|
| Immediate | CMP | #dd | C9 | 2 | 2 |
| Zero Page | CMP | aa | C5 | 2 | 3 |
| Zero Page, X | CMP | aa,X | D5 | 2 | 4 |
| Absolute | CMP | aaaa | CD | 3 | 4 |
| Absolute, X | CMP | aaaa,X | DD | 3 | 4* |
| Absolute, Y | CMP | aaaa,Y | D9 | 3 | 4* |
| (Indirect, X) | CMP | (aa,X) | C1 | 2 | 6 |
| (Indirect), Y | CMP | (aa),Y | D1 | 2 | 5* |

*Add 1 if page boundary is crossed.

## CPX

*Compare Memory and Index X*

Operation: X — M

N Z C I D V
√ √ √ — — —

| Addressing Mode | Assembly Language Form | | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|---|
| Immediate | CPX | #dd | E0 | 2 | 2 |
| Zero Page | CPX | aa | E4 | 2 | 3 |
| Absolute | CPX | aaaa | EC | 3 | 4 |

## CPY

*Compare Memory and Index Y*

Operation: Y — M

N Z C I D V
√ √ √ — — —

| Addressing Mode | Assembly Language Form | | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|---|
| Immediate | CPY | #dd | C0 | 2 | 2 |
| Zero Page | CPY | aa | C4 | 2 | 3 |
| Absolute | CPY | aaaa | CC | 3 | 4 |

## DEC

*Decrement Memory by One*

Operation: M − 1 → M

N Z C I D V
√ √ − − − −

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Zero Page | DEC aa | C6 | 2 | 5 |
| Zero Page, X | DEC aa,X | D6 | 2 | 6 |
| Absolute | DEC aaaa | CE | 3 | 6 |
| Absolute, X | DEC aaaa,X | DE | 3 | 7 |

## DEX

*Decrement Index X by One*

Operation: X − 1 → X

N Z C I D V
√ √ − − − −

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | DEX | CA | 1 | 2 |

## DEY

*Decrement Index Y by One*

Operation: Y − 1 → Y

N Z C I D V
√ √ − − − −

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | DEY | 88 | 1 | 2 |

## EOR

*Exclusive-OR Memory with Accumulator*

Operation: A V M → A

N Z C I D V
√ √ – – – –

| Addressing Mode | Assembly Language Form | | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|---|
| Immediate | EOR | #dd | 49 | 2 | 2 |
| Zero Page | EOR | aa | 45 | 2 | 3 |
| Zero Page, X | EOR | aa,X | 55 | 2 | 4 |
| Absolute | EOR | aaaa | 4D | 3 | 4 |
| Absolute, X | EOR | aaaa,X | 5D | 3 | 4* |
| Absolute, Y | EOR | aaaa,Y | 59 | 3 | 4* |
| (Indirect, X) | EOR | (aa,X) | 41 | 2 | 6 |
| (Indirect), Y | EOR | (aa),Y | 51 | 2 | 5* |

*Add 1 if page boundary is crossed.

## INC

*Increment Memory by One*

Operation: M + 1 → M

N Z C I D V
√ √ – – – –

| Addressing Mode | Assembly Language Form | | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|---|
| Zero Page | INC | aa | E6 | 2 | 5 |
| Zero Page, X | INC | aa,X | F6 | 2 | 6 |
| Absolute | INC | aaaa | EE | 3 | 6 |
| Absolute, X | INC | aaaa,X | FE | 3 | 7 |

## INX

*Increment Index X by One*

Operation: X + 1 → X

N Z C I D V
√ √ – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | INX | E8 | 1 | 2 |

## INY

*Increment Index Y by One*

Operation: Y + 1 → Y

N Z C I D V
✓ ✓ – – – –

| Addressing Mode | Assembly Language Form | OP Code | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | INY | C8 | 1 | 2 |

## JMP

*Jump*

Operation: (PC + 1) → PCL
(PC + 2) → PCH

N Z C I D V
– – – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Absolute | JMP aaaa | 4C | 3 | 3 |
| Indirect | JMP (aaaa) | 6C | 3 | 5 |

## JSR

*Jump to Subroutine*

Operation: PC + 2↓, (PC + 1) → PCL
(PC + 2) → PCH

N Z C I D V
– – – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Absolute | JSR aaaa | 20 | 3 | 6 |

# LDA

## *Load Accumulator with Memory*

Operation: M → A

N Z C I D V
✓ ✓ – – – –

| Addressing Mode | Assembly Language Form | | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|---|
| Immediate | LDA | #dd | A9 | 2 | 2 |
| Zero Page | LDA | aa | A5 | 2 | 3 |
| Zero Page, X | LDA | aa,X | B5 | 2 | 4 |
| Absolute | LDA | aaaa | AD | 3 | 4 |
| Absolute, X | LDA | aaaa,X | BD | 3 | 4* |
| Absolute, Y | LDA | aaaa,Y | B9 | 3 | 4* |
| (Indirect, X) | LDA | (aa,X) | A1 | 2 | 6 |
| (Indirect), Y | LDA | (aa),Y | B1 | 2 | 5* |

*Add 1 if page boundary is crossed.

# LDX

## *Load Index X with Memory*

Operation: M → X

N Z C I D V
✓ ✓ – – – –

| Addressing Mode | Assembly Language Form | | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|---|
| Immediate | LDX | #dd | A2 | 2 | 2 |
| Zero Page | LDX | aa | A6 | 2 | 3 |
| Zero Page, Y | LDX | aa,Y | B6 | 2 | 4 |
| Absolute | LDX | aaaa | AE | 3 | 4 |
| Absolute, Y | LDX | aaaa,Y | BE | 3 | 4* |

*Add 1 when page boundary is crossed.

## LDY

*Load Index Y with Memory*

Operation: M → Y

N Z C I D V
√ √ – – – –

| Addressing Mode | Assembly Language Form | | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|---|
| Immediate | LDY | #dd | A0 | 2 | 2 |
| Zero Page | LDY | aa | A4 | 2 | 3 |
| Zero Page, X | LDY | aea,X | B4 | 2 | 4 |
| Absolute | LDY | aaaa | AC | 3 | 4 |
| Absolute, X | LDY | aaaa,X | BC | 3 | 4* |

*Add 1 when page boundary is crossed.

## LSR

*Local Shift Right*

Operation: 0 → | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | → C

N Z C I D V
0 √ √ – – –

| Addressing Mode | Assembly Language Form | | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|---|
| Accumulator | LSR | A | 4A | 1 | 2 |
| Zero Page | LSR | aa | 46 | 2 | 5 |
| Zero Page, X | LSR | aa,X | 56 | 2 | 6 |
| Absolute | LSR | aaaa | 4E | 3 | 6 |
| Absolute, X | LSR | aaaa,X | 5E | 3 | 7 |

## NOP

*No Operation*

Operation: No Operation (2 cycles)

N Z C I D V
– – – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | NOP | EA | 1 | 2 |

## ORA

*OR Memory with Accumulator*

Operation: A V M → A

N Z C I D V
√ √ – – – –

| Addressing Mode | Assembly Language Form | | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|---|
| Immediate | ORA | #dd | 09 | 2 | 2 |
| Zero Page | ORA | aa | 05 | 2 | 3 |
| Zero Page, X | ORA | aa,X | 15 | 2 | 4 |
| Absolute | ORA | aaaa | 0D | 3 | 4 |
| Absolute, X | ORA | aaaa,X | 1D | 3 | 4* |
| Absolute, Y | ORA | aaaa,Y | 19 | 3 | 4* |
| (Indirect, X) | ORA | (aa,X) | 01 | 2 | 6 |
| (Indirect), Y | ORA | (aa),Y | 11 | 2 | 5* |

*Add 1 on page crossing.

## PHA

*Push Accumulator on Stack*

Operation: A ↓

N Z C I D V
– – – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | PHA | 48 | 1 | 3 |

## PHP

*Push Processor Status on Stack*

Operation: P ↓

N Z C I D V
– – – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | PHP | 08 | 1 | 3 |

## PLA

*Pull Accumulator from Stack*

Operation: A ↑

N Z C I D V
✓ ✓ – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | PLA | 68 | 1 | 4 |

## PLP

*Pull Processor Status from Stack*

Operation: P ↑

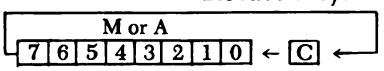
N Z C I D V
From Stack

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | PLP | 28 | 1 | 4 |

## ROL

*Rotate Left*

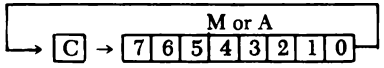
Operation:




N Z C I D V
✓ ✓ ✓ – – –

| Addressing Mode | Assembly Language Form | | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|---|
| Accumulator | ROL | A | 2A | 1 | 2 |
| Zero Page | ROL | aa | 26 | 2 | 5 |
| Zero Page, X | ROL | aa,X | 36 | 2 | 6 |
| Absolute | ROL | aaaa | 2E | 3 | 6 |
| Absolute, X | ROL | aaaa,X | 3E | 3 | 7 |

## *Rotate Right*

Operation: 


N Z C I D V
✓ ✓ ✓ – – –

| Addressing Mode | Assembly Language Form | | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|---|
| Accumulator | ROR | A | 6A | 1 | 2 |
| Zero Page | ROR | aa | 66 | 2 | 5 |
| Zero Page, X | ROR | aa,X | 76 | 2 | 6 |
| Absolute | ROR | aaaa | 6E | 3 | 6 |
| Absolute, X | ROR | aaaa,X | 7E | 3 | 7 |

# RTI

## *Return from Interrupt*

Operation: P↑ PC↑

N Z C I D V
From Stack

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | RTI | 40 | 1 | 6 |

# RTS

## *Return from Subroutine*

Operation: PC↑, PC + 1 → PC

N Z C I D V
– – – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | RTS | 60 | 1 | 6 |

## SBC

*Subtract from Accumulator with Carry*

Operation: $A - M - \overline{C} \rightarrow A$

Note: $\overline{C}$ = Borrow

N Z C I D V
√ √ √ – – √

| Addressing Mode | Assembly Language Form | | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|---|
| Immediate | SBC | #dd | E9 | 2 | 2 |
| Zero Page | SBC | aa | E5 | 2 | 3 |
| Zero Page, X | SBC | aa,X | F5 | 2 | 4 |
| Absolute | SBC | aaaa | ED | 3 | 4 |
| Absolute, X | SBC | aaaa,X | FD | 3 | 4* |
| Absolute, Y | SBC | aaaa,Y | F9 | 3 | 4* |
| (Indirect, X) | SBC | (aa,X) | E1 | 2 | 6 |
| (Indirect), Y | SBC | (aa),Y | F1 | 2 | 5* |

*Add 1 when page boundary is crossed.

## SEC

*Set Carry Flag*

Operation: 1 → C

N Z C I D V
– – 1 – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | SEC | 38 | 1 | 2 |

## SED

*Set Decimal Mode*

Operation: 1 → D

N Z C I D V
– – – – 1 –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | SED | F8 | 1 | 2 |

### *Set Interrupt Disable Status*

Operation: 1 → I

N Z C I D V
– – – 1 – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | SEI | 78 | 1 | 2 |

## STA

### *Store Accumulator in Memory*

Operation: A → M

N Z C I D V
– – – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Zero Page | STA aa | 85 | 2 | 3 |
| Zero Page, X | STA aa,X | 95 | 2 | 4 |
| Absolute | STA aaaa | 8D | 3 | 4 |
| Absolute, X | STA aaaa,X | 9D | 3 | 5 |
| Absolute, Y | STA aaaa,Y | 99 | 3 | 5 |
| (Indirect, X) | STA (aa,X) | 81 | 2 | 6 |
| (Indirect), Y | STA (aa),Y | 91 | 2 | 6 |

## STX

### *Store Index X in Memory*

Operation: X → M

N Z C I D V
– – – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Zero Page | STX aa | 86 | 2 | 3 |
| Zero Page, Y | STX aa,Y | 96 | 2 | 4 |
| Absolute | STX aaaa | 8E | 3 | 4 |

## STY

*Store Index Y in Memory*

Operation: Y → M

N Z C I D V
– – – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Zero Page | STY aa | 84 | 2 | 3 |
| Zero Page, X | STY aa,X | 94 | 2 | 4 |
| Absolute | STY aaaa | 8C | 3 | 4 |

## TAX

*Transfer Accumulator to Index X*

Operation: A → X

N Z C I D V
✓ ✓ – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | TAX | AA | 1 | 2 |

## TAY

*Transfer Accumulator to Index Y*

Operation: A → Y

N Z C I D V
✓ ✓ – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | TAY | A8 | 1 | 2 |

## TSX

*Transfer Stack Pointer to Index X*

Operation: S → X

N Z C I D V
✓ ✓ – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | TSX | BA | 1 | 2 |

## TXA

*Transfer Index X to Accumulator*

Operation: X → A

N Z C I D V
✓ ✓ – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | TXA | 8A | 1 | 2 |

## TXS

*Transfer Index X to Stack Pointer*

Operation: X → S

N Z C I D V
– – – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | TXS | 9A | 1 | 2 |

## TYA

*Transfer Index Y to Accumulator*

Operation: Y → A

N Z C I D V
✓ ✓ – – – –

| Addressing Mode | Assembly Language Form | OP CODE | No. Bytes | No. Cycles |
|---|---|---|---|---|
| Implied | TYA | 98 | 1 | 2 |

# M/C mnemonics

6502 INSTRUCTION REPERTOIRE

| MNEMONIC | FUNCTION |
|---|---|
| ADC | Add memory to accumulator with carry. |
| AND | AND memory with acuumulator. |
| ASL | Shift left 1 bit. |
| BCC | Branch on carry clear. |
| BCS | Branch on carry set. |
| BEQ | Branch on 0. |
| BIT | Test bits in memory with accumulator. |
| BMI | Branch on result negative. |
| BNE | Branch on result <> 0. |
| BPL | Branch on result positive. |
| BRK | Force break. |
| BVC | Branch on overflow clear. |
| BVS | Branch on overflow set. |
| CLC | Clear carry flag. |
| CLD | Clear decimal mode. |
| CLI | Clear interrupt disable. |
| CLV | Clear overflow flag. |
| CMP | Compare memory with accumulator. |
| CPX | Compare memory with X register. |
| CPY | Compare memory with Y register. |
| DEC | Decrement memory. |
| DEX | Decrement X register. |
| DEY | decrement Y register. |
| EOR | Exclusive OR memory with accumulator. |
| INC | Increment memory. |
| INX | Increment X register. |
| INY | Increment Y register. |
| JMP | Jump to specified location. |
| JSR | Jump to subroutine. |
| LDA | Load accumulator. |
| LDX | Load X register. |
| LDY | Load Y register. |
| LSR | Shift right 1 bit. |
| NOP | No operation. |
| ORA | OR memory with accumulator. |
| PHA | Push accumulator onto stack. |
| PHP | Push processor status onto stack. |
| PLA | Pull accumulator from stack. |
| PLP | Pull processor status from stack. |

| | |
|---|---|
| ROL | Rotate left 1 bit. |
| ROR | Rotate right 1 bit. |
| RTI | Return from interrupt. |
| RTS | Return from subroutine. |
| SBC | Subtract memory with borrow from accumulator. |
| SEC | Set carry flag. |
| SED | Set decimal mode. |
| SEI | Set interrupt disable. |
| STA | Store accumulator. |
| STX | Store X register. |
| STY | Store Y register. |
| TAX | Transfer accumulator to X register. |
| TAY | Transfer accumulator to Y register. |
| TSX | Transfer stack pointer to X register. |
| TXA | Transfer X register to accumulator. |
| TXS | Transfer X register to stack pointer. |
| TYA | Transfer Y register to accumulator. |

# Powers tables

Powers of 2

| N<br>2 | N |
|---|---|
| 256 | 8 |
| 512 | 9 |
| 1,024 | 10 |
| 2,048 | 11 |
| 4,096 | 12 |
| 8,192 | 13 |
| 16,384 | 14 |
| 32,768 | 15 |
| 65,536 | 16 |
| 131,072 | 17 |
| 262,144 | 18 |
| 524,288 | 19 |
| 1,048,576 | 20 |
| 2,097,152 | 21 |
| 4,194,304 | 22 |
| 8,388,608 | 23 |
| 16,777,216 | 24 |

Powers of 16

| N<br>16 | N |
|---|---|
| 1 | 0 |
| 16 | 1 |
| 256 | 2 |
| 4,096 | 3 |
| 65,536 | 4 |
| 1,048,576 | 5 |
| 16,777,216 | 6 |
| 268,435,456 | 7 |
| 4,294,967,296 | 8 |
| 68,719,476,736 | 9 |
| 1,099,511,627,776 | 10 |
| 17,592,186,044,416 | 11 |
| 281,474,976,710,656 | 12 |
| 4,503,599,627,370,496 | 13 |
| 72,057,594,037,927,936 | 14 |
| 1,152,921,504,606,846,976 | 15 |

# Cartridge slot

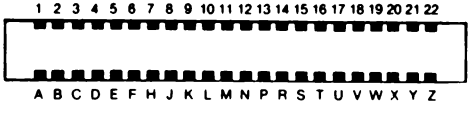
## Cartridge Expansion Slot

| Pin | Type |
|---|---|
| 22 | GND |
| 21 | CD0 |
| 20 | CD1 |
| 19 | CD2 |
| 18 | CD3 |
| 17 | CD4 |
| 16 | CD5 |
| 15 | CD6 |
| 14 | CD7 |
| 13 | $\overline{\text{DMA}}$ |
| 12 | BA |

| Pin | Type |
|---|---|
| 11 | $\overline{\text{ROML}}$ |
| 10 | I/O2 |
| 9 | $\overline{\text{EXROM}}$ |
| 8 | $\overline{\text{GAME}}$ |
| 7 | I/O1 |
| 6 | Dot Clock |
| 5 | $\overline{\text{CR/W}}$ |
| 4 | $\overline{\text{IRQ}}$ |
| 3 | + 5V |
| 2 | + 5V |
| 1 | GND |

| Pin | Type |
|---|---|
| Z | GND |
| Y | CA0 |
| X | CA1 |
| W | CA2 |
| V | CA3 |
| U | CA4 |
| T | CA5 |
| S | CA6 |
| R | CA7 |
| P | CA8 |
| N | CA9 |

| Pin | Type |
|---|---|
| M | CA10 |
| L | CA11 |
| K | CA12 |
| J | CA13 |
| H | CA14 |
| F | CA15 |
| E | S02 |
| D | $\overline{\text{NMI}}$ |
| C | $\overline{\text{RESET}}$ |
| B | $\overline{\text{ROMH}}$ |
| A | GND |

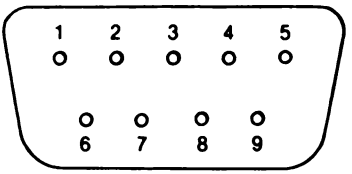


# Joystick slot

1) Game I/O
2) Cartridge Slot
3) Audio/Video
4) Serial i/O (Disk/Printer)
5) Modulator Output
6) Cassette
7) User Port

**Control Port 1**

| Pin | Type | Note |
|---|---|---|
| 1 | JOYA0 | |
| 2 | JOYA1 | |
| 3 | JOYA2 | |
| 4 | JOYA3 | |
| 5 | POT AY | |
| 6 | BUTTON A/LP | |
| 7 | +5V | MAX. 100mA |
| 8 | GND | |
| 9 | POT AX | |




**Control Port 2**

| Pin | Type | Note |
|---|---|---|
| 1 | JOYB0 | |
| 2 | JOYB1 | |
| 3 | JOYB2 | |
| 4 | JOYB3 | |
| 5 | POT BY | |
| 6 | BUTTON B | |
| 7 | +5V | MAX. 100mA |
| 8 | GND | |
| 9 | POT BX | |

# RS232 standards

EIA RS232-C (CCITT V24)

Notes
---

Transmission is serial (asynchronous).
MARK = binary 1 = OFF = -3 to -25 volts.
SPACE = binary 0 = ON = +3 to +25 volts.
25-pin "D" type connector.
Data Control Equipment (DCE) has female connector.
Data Terminal Equipment (DTE) has male connector.
Open circuit drive voltage cannot exceed 25 volts.
Terminator resistance 3-7K ohms.
50 foot maximum DCE, DTE separation.
2500 pico farad max conductor capacitance.

| PIN | NAME | DIRECTION | CIRCUIT CCITT | EIA | FUNCTION |
|---|---|---|---|---|---|
| 01 | FG | - | 101 | AA | Frame Ground. |
| 02 | TD | To DCE | 103 | BA | Transmitted Data. |
| 03 | RD | To DTE | 104 | BB | Received Data. |
| 04 | RTS | To DCE | 105 | CA | Request To Send. |
| 05 | CTS | To DTE | 106 | CB | Clear To Send. |
| 06 | DSR | To DTE | 107 | CC | Data Set Ready. |
| 07 | SG | - | 102 | AB | Signal Ground. |
| 08 | DCD | To DTE | 109 | CF | Data Carrier Detect. |
| 09 | | To DTE | | | Positive DC Test Voltage. |
| 10 | | To DTE | | | Negative DC Test Voltage. |
| 11 | QM | To DTE | Bell 208A | | Equaliser Mode. |
| 12 | (S)DCD | To DTE | 122 | SCF | Secondary Data Carrier Detect. |
| 13 | (S)CTS | To DTE | 121 | SCB | Secondary Clear To Send. |
| 14 | (S)TD | To DCE | 118 | SBA | Secondary Transmitted Data. |
| | NS | To DCE | Bell 208A | | New Synch. |
| 15 | TC | To DTE | 114 | DB | Transmitter Clock. |
| 16 | (S)RD | To DTE | 119 | SBB | Secondary Received Data. |
| | DCT | To DTE | Bell 208A | | Divided Clock Transmitter. |
| 17 | RC | To DTE | 115 | DD | Receiver Clock. |
| 18 | DCR | To DTE | Bell 208A | | Divided Clock Receiver. |
| 19 | (S)RTS | To DCE | 120 | SCA | Secondary Request to Send. |
| 20 | DTR | To DCE | 108.2 | CD | Data Terminal Ready. |
| 21 | SQ | To DTE | 110 | CG | Signal Quality Detect. |
| 22 | RI | To DTE | 125 | CE | Ring Indicator. |
| 23 | | To DCE | 111 | CH | Data Rate Selector. |
| | | To DTE | 112 | CI | Data Rate Selector. |
| 24 | TC | To DCE | 113 | DA | EXT Transmitter Clock. |
| 25 | | To DCE | Bell 113B | | Busy. |

Circuit 102 - Signal Ground or Common Return

This conductor establishes the signal common return for interchange circuits.

Circuit 103 - Transmitted Data

The data signals originated by the DTE, to be transmitted via the data channel to one or more remote data stations, are transferred on this circuit to DCE.

Circuit 104 - Received Data

The data signals generated by the DCE, in response to data channel line signals received from a remote data station, are transferred on this circuit to the DTE.

Circuit 105 - Request to Send

Controls the data channel transmit function of the DCE.

Circuit 106 - Ready for Sending

Indicates whether the DCE is conditioned to transmit data on the data channel.

Circuit 107 - Data Set Ready

Indicates whether the DCE is ready to operate.

Circuit 108/1 - Connect Data Set to Line

Controls switching of the signal-conversion or similar equipment to or from the line.

Circuit 108/2 - Data Terminal Ready

Controls switching of the signal-conversion or similar equipment to or from the line.

Circuit 109 - Carrier Detect

Indicates whether the received data channel line signal is within appropriate limits, as specified by the relevant recommendation for DCE.

Circuit 110 - Data Signal Quality Detector

Indicates whether there is a reasonable probability of an

error in the data received on the data channel.

Circuit 111 - Data Signalling Rate Selector

Used to select one or two data signalling rates of a dual-rate synchronous DCE, or to select one of the two ranges of data signalling rates of a dual-range synchronous DCE.

Circuit 112 - Data Signalling Rate Selector

Used to select one of the two data signalling rates or ranges of rates in the DTE to coincide with the data signalling rate or range of rates in use in a dual-rate synchronous or dual-range asynchronous DCE.

Circuit 113 - Transmitter Signal Element Timing

Provides the DCE with signal element timing information.

Circuit 114 - Transmitter Signal Element Timing

Provides the DTE with signal element timing information.

Circuit 115 - Receiver Signal Element Timing

Provides the DTE with signal element timing information.

Circuit 116 - Select Standby

Used to select the normal or standby facilities such as signal convertors and communication channels.

Circuit 117 - Standby Indicator

Indicates whether the DCE is conditioned in its standby mode with the pre-determined facilities replaced by their reserves.

Circuit 118 - Transmitted Backward Channel Data

Equivalent to circuit 103, except that it is used for data received on the backward channel.

Circuit 120 - Transmit Backward Channel Line Signal

Equivalent to circuit 105, except that it is used to control the backward channel transmit function of the DCE.

Circuit 121 - Backward Channel Ready

Equivalent to circuit 106, except that it is used to

indicate whether the DCE is conditioned to transmit data on the backward channel.

Circuit 122 - Supervisory Carrier Detect

Equivalent to circuit 109, except that it is used to indicate whether the received backward channel line signal is within appropriate limits.

Circuit 123 - Backward Channel Signal Quality Detector

Equivalent to circuit 110, except that it is used to indicate the signal quality of the received backward channel line signal.

Circuit 124 - Select Frequency Groups

Used to select the desired frequency groups available on the DCE.

Circuit 125 - Calling Indicator

Indicates whether a calling signal is being received by the DCE.

Circuit 126 - Select Transmit Frequency

Used to select the required transmit frequency of the DCE.

Circuit 127 - Select Receive Frequency

Used to select the required receive frequency of the DCE.

Circuit 128 - Receiver Signal Element Timing

Provides the DCE with signal element timing information.

Circuit 129 - Request to Receive

Used to control the receive function of the DCE.

Circuit 130 - Transmit Backward Tone

Controls the transmission of a backward channel tone.

Circuit 131 - Received Character Timing

Provides the DTE with character timing information.

Circuit 132 - Return to Non-Data Mode

Used to restore the non-data mode provided with the DCE, without releasing the line connection to the remote station.

Circuit 133 - Ready for Receiving

Controls the transfer of data on circuit 104, indicating whether the DTE is capable of accepting a given amount of data, specified in the appropriate recommendation for intermediate equipment, for example, error control equipment.

Circuit 134 - Received Data Present

Used to separate information messages from supervisory messages, transferred on circuit 104.

Circuit 191 - Transmitted Voice Answer

Signals generated by a voice answer unit in the DTE are transferred on this circuit to the DCE.

Circuit 192 - Received Voice Answer

Received voice signals, generated by a voice answering unit at the remote data terminal, are transferred on this circuit to the DTE.

## Other CCITT "V" Interfaces

V10

Electrical characteristics for unbalanced double-current interchange circuits for general use with integrated circuit equipment in the field of data communications.

V11

Electrical characteristics for balanced double-current interchange circuits for general use with integrated circuit equipment in the field of data communications.

V15

Use of acoustic coupling for data transmission.

V16

Medical analogue data transmission modems.

V19

Modems for parallel data transmission using telephone signalling frequencies.

V20
---

Parallel data transmission modems standardised for universal use in the general switch telephone network.

V21
---

200-baud modem standardised for use in the general switched telephone network.

V22
---

Defines the procedures and standards for 1200 baud full duplex communications over the public switched network.

V23
---

600/1200-baud modem standardised for use in the general switched telephone network.

V24
---

List of definitions for interchange circuits between data terminal equipment and data circuit terminating equipment.

V25
---

Automatic calling and/or answering equipment on the general swi'=hed telephone network, including disabling of echo-suppressors on manually established calls.

V26
---

2400 bits per second modem standardised for use on 4-wire leased telephone-type circuits.

V26 (alternative)
-----------------

2400/1200 bits per second modem standardised for use in the general switched telephone network.

V27
---

4800 bits per second modems with manual equaliser standardised for use on leased telephone-type circuits.

V27 (alternative 1)
-------------------

4800 bits per second modems with automatic equaliser

standardised for use on leased telephone-type circuits.

V27 (alternative 2)

4800/2400 bits per second modems standardised for use in the general switched telephone network.

V28

Electrical characteristics for unbalanced double-current interchange circuits.

V29

9600 bits per second modems standardised for use in leased telephone circuits.

V31

Electrical characteristics for single current interchange circuits controlled by contact closure.

V35

Data transmission at 48 kilobits per second using 60-108 KHz group band circuits.

V36

Modems for synchronous transmission using 60-108 KHz group band circuits.

# Centronics standards

CENTRONICS PARALLEL INTERFACE

Notes
---

Busy is set if:

1) Data is being received.
2) Printer is printing.
3) Printer is offline.
4) An error condition is present.

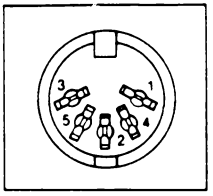
On pins 02-09 a high level represents binary ONE, a low level represents binary ZERO. All printable characters ( i.e. codes having a ONE in DATA 6 or DATA 7) are stored in the printer buffer. Control characters (i.e. codes ZERO in both DATA 6 and DATA 7) are used to specify special control functions. These codes are not stored in the buffer except when they specify a print command and are preceded by at least one printable character in that line.

| PIN | CODE | FUNCTION |
|---|---|---|
| 01 | STROBE | Read Data Pulse. |
| 02 | DATA 1 | Data lines. |
| 03 | DATA 2 | ditto. |
| 04 | DATA 3 | ditto. |
| 05 | DATA 4 | ditto. |
| 06 | DATA 5 | ditto. |
| 07 | DATA 6 | ditto. |
| 08 | DATA 7 | ditto. |
| 09 | DATA 8 | ditto. |
| 10 | ACKNLG | Data Received and Ready for More. |
| 11 | BUSY | Not Ready for Data. |
| 12 | PE | SET high when Out-of-Paper. |
| 13 | +5V | |
| 14 | AUTO FEED | Switch Set gives extra line-feed. |
| 15 | NC | No Connection. |
| 16 | GND LOGIC | Logic Ground. |
| 17 | GND CASE | Chassis Ground. |
| 18 | NC | No Connection. |
| 19-30 | GND | Signal Grounds. |
| 31 | INT | Reset and Buffer Clear. |
| 32 | ERROR | See Notes on BUSY. |
| 33 | GND | Signal Ground. |
| 34 | NC | No Connection. |
| 35 | +5V | |
| 36 | SLCT IN | Optional DC1/DC3. |

# Other output

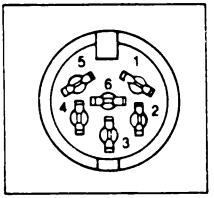
### Audio/Video

| Pin | Type | Note |
|---|---|---|
| 1 | LUMINANCE | |
| 2 | GND | |
| 3 | AUDIO OUT | |
| 4 | VIDEO OUT | |
| 5 | AUDIO IN | |



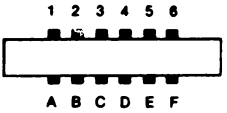

### Serial I/O

| Pin | Type |
|---|---|
| 1 | SERIAL $\overline{\text{SRQIN}}$ |
| 2 | GND |
| 3 | SERIAL ATN IN/OUT |
| 4 | SERIAL CLK IN/OUT |
| 5 | SERIAL DATA IN/OUT |
| 6 | $\overline{\text{RESET}}$ |




### Cassette

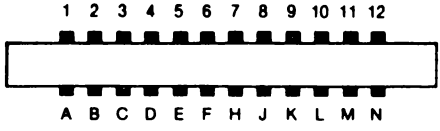
| Pin | Type |
|---|---|
| A-1 | GND |
| B-2 | +5V |
| C-3 | CASSETTE MOTOR |
| D-4 | CASSETTE READ |
| E-5 | CASSETTE WRITE |
| F-6 | CASSETTE SENSE |



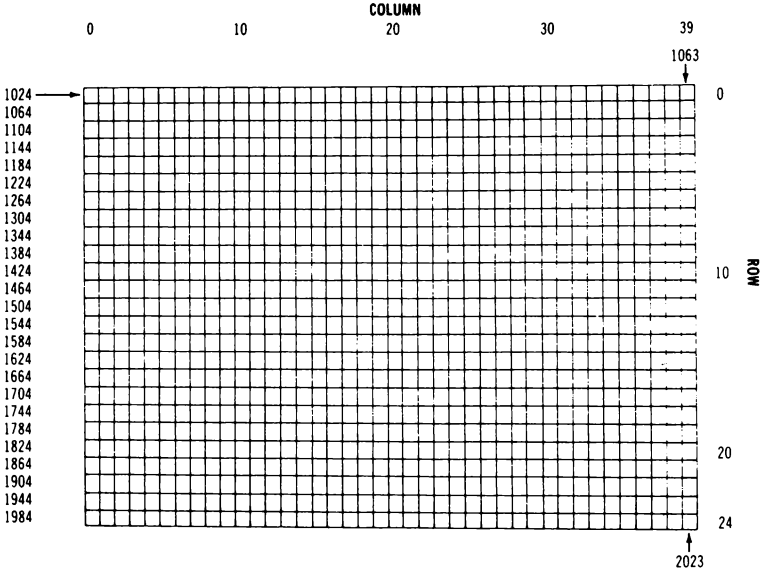
## User I/O

| Pin | Type | Note |
|---|---|---|
| 1 | GND | |
| 2 | +5V | MAX. 100 mA |
| 3 | $\overline{\text{RESET}}$ | |
| 4 | CNT1 | |
| 5 | SP1 | |
| 6 | CNT2 | |
| 7 | SP2 | |
| 8 | $\overline{\text{PC2}}$ | |
| 9 | SER. ATN IN | |
| 10 | 9 VAC | MAX. 100 mA |
| 11 | 9 VAC | MAX. 100 mA |
| 12 | GND | |

| Pin | Type | Note |
|---|---|---|
| A | $\overline{\text{GND}}$ | |
| B | FLAG2 | |
| C | PB0 | |
| D | PB1 | |
| E | PB2 | |
| F | PB3 | |
| H | PB4 | |
| J | PB5 | |
| K | PB6 | |
| L | PB7 | |
| M | PA2 | |
| N | GND | |



# Screen memory



# Sound chip registers

| Byte | Description |
|---|---|
| 00 | Low Frequency value of note for voice 1 |
| 01 | High Frequency value of note for voice 1 |
| 02 | Low Pulse Rate for voice 1 |
| 03 | High Pulse Rate for voice 1 |
| 04 | Waveform for voice 1 |
| 05 | Attack/Decay for voice 1 |
| 06 | Sustain/Relese for voice 1 |
| 07 | Low Frequency value of note for voice 2 |
| 08 | High Frequency value of note for voice 2 |
| 09 | Low Pulse Rate for voice 2 |
| 10 | High Pulse Rate for voice 2 |
| 11 | Waveform for voice 2 |
| 12 | Attack/Decay for voice 2 |
| 13 | Sustain/Release for voice 2 |
| 14 | Low Frequency value of note for voice 3 |
| 15 | High Frequency value of note for voice 3 |
| 16 | Low Pulse Rate for voice 3 |
| 17 | High Pulse Rate for voice 3 |
| 18 | Waveform for voice 3 |
| 19 | Attack/Decay for voice 3 |
| 20 | Sustain/Release for voice 3 |
| 21 | High Frequency Cut-Off |
| 22 | Low Frequency Cut-Off |
| 23 | Turn on filtering |
| 24 | Set volume for all three voices Plus select filter type |
| 25 | Access To Output of envelope generator of voice 3 |
| 27 | Digitised output from voice 3 |
| 28 | Digitised output from envelope generator 3 |

# Musical notes values

| Note | Note–Octave | Hi Freq | Low Freq |
|---|---|---|---|
| 0 | C–0 | 1 | 18 |
| 1 | C#–0 | 1 | 35 |
| 2 | D–0 | 1 | 52 |
| 3 | D#–0 | 1 | 70 |
| 4 | E–0 | 1 | 90 |
| 5 | F–0 | 1 | 110 |
| 6 | F#–0 | 1 | 132 |
| 7 | G–0 | 1 | 155 |
| 8 | G#–0 | 1 | 179 |
| 9 | A–0 | 1 | 205 |
| 10 | A#–0 | 1 | 233 |
| 11 | B–0 | 2 | 6 |
| 12 | C–1 | 2 | 37 |
| 13 | C#–1 | 2 | 69 |
| 14 | D–1 | 2 | 104 |
| 15 | D#–1 | 2 | 140 |
| 16 | E–1 | 2 | 179 |
| 17 | F–1 | 2 | 220 |
| 18 | F#–1 | 3 | 8 |
| 19 | G–1 | 3 | 54 |
| 20 | G#–1 | 3 | 103 |
| 21 | A–1 | 3 | 155 |
| 22 | A#–1 | 3 | 210 |
| 23 | B–1 | 4 | 12 |
| 24 | C–2 | 4 | 73 |
| 25 | C#–2 | 4 | 139 |
| 26 | D–2 | 4 | 208 |
| 27 | D#–2 | 5 | 25 |
| 28 | E–2 | 5 | 103 |
| 29 | F–2 | 5 | 185 |
| 30 | F#–2 | 6 | 16 |
| 31 | G–2 | 6 | 108 |
| 32 | G#–2 | 6 | 206 |
| 33 | A–2 | 7 | 53 |
| 34 | A#–2 | 7 | 163 |
| 35 | B–2 | 8 | 23 |
| 36 | C–3 | 8 | 147 |
| 37 | C#–3 | 9 | 21 |
| 38 | D–3 | 9 | 159 |
| 39 | D#–3 | 10 | 60 |
| 40 | E–3 | 10 | 205 |
| 41 | F–3 | 11 | 114 |
| 42 | F#–3 | 12 | 32 |
| 43 | G–3 | 12 | 216 |

| Note | Note–Octave | Hi Freq | Low Freq |
|---|---|---|---|
| 44 | G#–3 | 13 | 156 |
| 45 | A–3 | 14 | 107 |
| 46 | A#–3 | 15 | 70 |
| 47 | B–3 | 16 | 47 |
| 48 | C–4 | 17 | 37 |
| 49 | C#–4 | 18 | 42 |
| 50 | D–4 | 19 | 63 |
| 51 | D#–4 | 20 | 100 |
| 52 | E–4 | 21 | 154 |
| 53 | F–4 | 22 | 227 |
| 54 | F#–4 | 24 | 63 |
| 55 | G–4 | 25 | 177 |
| 56 | G#–4 | 27 | 56 |
| 57 | A–4 | 28 | 214 |
| 58 | A#–4 | 30 | 141 |
| 59 | B–4 | 32 | 94 |
| 60 | C–5 | 34 | 75 |
| 61 | C#–5 | 36 | 85 |
| 62 | D–5 | 38 | 126 |
| 63 | D#–5 | 40 | 200 |
| 64 | E–5 | 43 | 52 |
| 65 | F–5 | 45 | 198 |
| 66 | F#–5 | 48 | 127 |
| 67 | G–5 | 51 | 97 |
| 68 | G#–5 | 54 | 111 |
| 69 | A–5 | 57 | 172 |
| 70 | A#–5 | 61 | 126 |
| 71 | B–5 | 64 | 188 |
| 72 | C–6 | 68 | 149 |
| 73 | C#–6 | 72 | 169 |
| 74 | D–6 | 76 | 252 |
| 75 | D#–6 | 81 | 161 |
| 76 | E–6 | 86 | 105 |
| 77 | F6 | 91 | 140 |
| 78 | F#–6 | 96 | 254 |
| 79 | G–6 | 102 | 194 |
| 80 | G#–6 | 108 | 223 |
| 81 | A–6 | 115 | 88 |
| 82 | A#–6 | 122 | 52 |
| 83 | B–6 | 129 | 120 |
| 84 | C–7 | 137 | 43 |
| 85 | C#–7 | 145 | 83 |
| 86 | D–7 | 153 | 247 |
| 87 | D#–7 | 163 | 31 |
| 88 | E–7 | 172 | 210 |
| 89 | F–7 | 183 | 25 |
| 90 | F#–7 | 193 | 252 |
| 91 | G–7 | 205 | 133 |
| 92 | G#–7 | 217 | 189 |
| 93 | A–7 | 230 | 176 |
| 94 | A#–7 | 244 | 103 |

# Sprite memory diagram

| Address (53248 + ) | Description |
|---|---|
| 00 | X position of sprite 0 |
| 01 | Y position of sprite 0 |
| 02-15 | Ditto for sprites 1 through 7 |
| 16 | Most Significant Bit of X position |
| 17 | Most Significant Bit of Y position |
| 18 | Raster Register |
| 19 | X position of Light Pen |
| 20 | Y position of Light Pen |
| 21 | Turn Sprite On |
| 23 | Expand Sprite in Y direction |
| 24 | Memory Pointers |
| 25 | Interrupt Register |
| 26 | Enable Interrupt |
| 27 | Sprite Data Priority |
| 28 | Multi-colour Sprites! |
| 29 | Expand Sprite in X direction |
| 30 | Sprite to Sprite collision |
| 31 | Sprite Data collision |
| 32 | Exterior colour |
| 33 | Background Colour 0 |
| 34 | ,, ,, 1 |
| 35 | ,, ,, 2 |
| 36 | ,, ,, 3 |
| 37 | Sprite multi-colour 0 |
| 38 | Sprite multi-colour 1 |
| 39 | Colour for sprite 0 |
| 40-46 | Ditto for sprites 1 through 7 |

# DUCKWORTH HOME COMPUTING

a new series

All books written by Pete Gerrard, former editor of *Commodore Computing International*, author of two top-selling adventure games for the Commodore 64 and a regular contributor to *Personal Computer News, Which Micro? and Software Review.*

**USING THE COMMODORE 64**

A complete look at the latest home computer from Commodore Business Machines. Starting with a refresher course in Basic Programming, it moves on through machine code, before considering in great detail sprites, graphics and sound. A section on peripherals, and then the heart of the book: an in-depth look at the chips that make it work, including the 6581 Sound Interface Device and the 6566 Video Controller Chip, as well as the heart of the computer, the 6510. The comprehensive appendices cover the full Basic and Machine Code Instruction sets, as well as several useful reference tables, and a complete machine code assembler/disassembler listing.

*Available now £9.95*

**EXPLORING ADVENTURES ON THE COMMODORE 64**

The complete guide to computer adventure games: playing, writing and solving them. Starting with an introduction to adventures, and their early history, it takes you gently through the basic programming necessary on the 64 before you can start writing your own games. Inputting of information, room mapping, movement, vocabulary, and everything required to write an adventure game are explored in full detail. Then follow a number of adventure scenarios, and finally three complete listings, written specially for the 64. The three games listed in this book are available on one cassette at £7.95.

*Available now £6.95*

Other titles in the series include *The Beginner's Guide to Computers & Computing, Sprites & Sound on the 64, 12 Simple Electronic Projects for the VIC, Will You Still Love Me When I'm 64, Advanced Basic & Machine Code Programming on the VIC, Advanced Basic & Machine Code Programming on the 64, Exploring Adventures on the VIC,* as well as *Pocket Handbooks for the VIC, 64, Dragon, Spectrum and BBC Model B.* Write in for a descriptive leaflet.



**DUCKWORTH**

The Old Piano Factory, 43 Gloucester Crescent, London NW1 7DY
Tel: 01-485 3484

# Index

Arithmetic operators: 11
Arithmetic functions: 11
ASCII tables: 7
BAM formats: 29, 30, 32, 33
Basic commands: 9, 10
Cartridge slot: 77
CCITT V24 definitions: 80-83
CCITT V interfaces: 83-85
Centronics standards: 86
Code conversion tables: 15-18
Colour memory: 14
Dec/hex convertors: 42
Directory format: 31, 34
Directory structure: 26
Disk commands Basic 4: 19
Disk commands Basic 2: 20
Disk file formats: 27, 28
DOS error messages: 21-25
Error messages: 12, 13
Extramon listing: 35-39
Flow charting: 40, 41
Hyperbolic functions: 43
Input/output: 87, 88
Joystick slot: 78
Keyboard sequences: 8
Machine Code instruction set: 54-73
Machine Code mnemonics: 74, 75
Memory architecture: 51-53
Memory map: 46, 47
Musical note values: 91, 92
Page zero memory map: 44, 45, 46
Powers table: 76
ROM memory map: 48-50
RS232 standards: 79
Screen memory: 89
Sound chip registers: 90
Sprite memory diagram: 93

# Duckworth Home Computing

**A POCKET HANDBOOK FOR THE COMMODORE 64**
**by Peter Gerrard and Danny Doyle**

This book contains all the vital information you will need when using your 64. There are sections on: ASCII tables – Basic keywords – Basic error messages – Colour memory – Conversion tables – Disk commands – Disk error messages – Disk formats – Extramon listing – Flow charting – Hex/Dec convertor – Hyperbolic functions – Memory maps – Memory architecture – M/C instruction set – M/C mnemonics – Powers tables – Cartridge slot – Joystick slot – RS232 standards – Centronics standards – Other output – Screen memory – Sound chip registers – Musical notes values – Sprite memory diagram. In short, everything you need to know about your machine.

ISBN 0-7156-1787-7

9 780715 617878

**Duckworth**
The Old Piano Factory
43 Gloucester Crescent, London NW1

0684
NR00238
73,75



This was brought to you

from the archives of

http://retro-commodore.eu