ADVANCED REFERENCE MANUAL for the BBC MASTER

The Advanced Reference Manual

For The BBC Master 128 Microcomputer

Published by Watford Electronics

Published in the United Kingdom by
Watford Electronics
Jessa House
250 Lower High Street
Watford
WD1 2AN
England

Telephone 0923 37774
Telex 8956095
Fax 01 950 8989

ISBN 0 948663 05 7

This book was computer typeset by Ian Bishop-Laggett,
Ideal Software Consultants, 11 Hathaway Close, Luton, Bedfordshire.

Re-mastered by dv8 in 2020
First revision, February 2021
For the latest revision go to: stardot.org.uk/forums/viewforum.php?f=42

Acknowledgements

Thanks to David Bell, Roger Cullis, Dave Futcher, Adrian Bishop-Laggett and all those people who made the publication of this manual possible.

CONTENTS

INTRODUCTION

This book is intended for peripheral hardware designers and software writers and expands the information given in Reference Manuals Parts 1 and 2.

It contains software and hardware reference material, with application guidelines which anyone who is attempting a major project for the first time will find particularly useful. The remaining chapters contain information on the Acorn-designed semi-custom chips and a number of detailed appendices highlight the differences between the Master 128 and other Acorn models including the Compact and the Electron.

It has been assumed that the reader has a good understanding of basic electronics and computer terminology.

1 THE MASTER SERIES' ARCHITECTURE

Introduction

The Master Series is based on and extends the architecture of the Acorn BBC Model B microcomputer. The heart of the computer is a comprehensive machine operating system (MOS) which controls and organises the communications between a central processing unit (CPU) and applications software, peripheral devices, such as video displays and printers, and filing systems which act as sources and stores for data. Language interpreters and compilers may be provided to convert high level languages into a format usable by the MOS. Alternatively, the applications may be in object code which runs directly on the CPU.

The simplest version of the computer (the Master 128) has a single processor which performs all of these executive functions. In other computers of the series, responsibility is split between a base processor which handles input/output (I/O) operations and a language processor, which performs the calculations and other data operations associated with the applications' tasks. In general, the language processor will be selected for its suitability for a particular application and will be different from the base processor.

Core Machine

All input/output (I/O) computing is performed by a 65C12 CPU with its principal ancillary components.

128 Kbyte of dynamic random access memory (DRAM)
Special expansion options allow a further expansion of 64 Kbyte. Dedicated hardware can be used to expand this almost indefinitely.

262 Kbyte of read-only memory (ROM)
Special expansion options allow a further expansion of approximately half a megabyte of ROM. Plug in cartridges are available which accept up to 256 Kbyte of ROM.

Internal I/O

Internal versatile interface adapter (VIA)
This services a 93-contact keyboard with two-key rollover, a three-channel sound generator with additional noise channel and a battery-backed real-time clock with fifty bytes of RAM.

External versatile interface adapter (VIA)
This services the parallel printer port and the user port.

Co-processors
These consist of an additional CPU with associated memory. They depend entirely on the main processor for all I/O operations.

External I/O

Video display
A 6845 CRT controller formats the output for RGB, composite video and PAL/NTSC connectors.

Analogue to Digital Converter
A four-channel A-D converter provides ten-bit binary conversions in 5ms. The absolute accuracy will depend on the conditions of use.

Tape Interface
Facilities to both save and retrieve data from audio cassettes.

Disc Interface
Facilities to both save and retrieve data from standard Shugart connected media. Filing systems data encoded in FM or MFM format.

Network Interface
Connection to ECONET is provided by a 68B54 advanced data link controller. This is fitted on a daughter board and may be an optional extra (standard on the ET machine).

1MHz Bus
Standard BBC computer 1 MHz bus.

External Second Processor
An external second processor may be connected. Selection of either internal co-processor or external second processor is performed by software. Only one second or co-processor can be active at a time.

Centronics Printer Port
Connection for a standard parallel printer.

User Port
The user port is an eight-bit bidirectional bus with two extra handshaking/serial lines. These are unbuffered.

RS423
A serial RS423 port. This is an enhanced version of the RS232C specification.

Audio Output

The output from the sound generator is amplified to a speaker and provided at a phono-style connector. Sound transfer to and from the modem.

Modem

Connection for a modem with both dial pulse and dual tone multi-frequency dialling.

Internal Input/Output

Slow peripherals

These are subsystems which are provided with data from port A of the system VIA. This data is stable until next programmed by the CPU.

Sound Generator

The sound generator is an SN7694A device, which generates three sound channels plus one pseudo random noise channel. The full description of it is found in the manufacturers data sheet. It receives a reference clock of 4MHz from central timing. The output can be connected by screened cable to the optional modem. This output is mixed on the modem board to generate dialling tones for DTMF exchanges where the modem hardware does not provide such tones itself.

Real time clock with RAM

A 146818 RTC and RAM chip is provided with battery-backed supply. The chip operation is described in the manufacturers data sheet. There are three AA size batteries which normally keep the RAM backed-up for at least a year (depending on how much the machine is NOT used).

The keyboard mounted battery is charged whilst the computer is running from the mains supply. An over-charge prevention circuit is provided with the following action:-

a) Upon switch on, charging current of about 30mA is applied.
b) After approximately 15 minutes the charging current falls to 1mA.
c) “Trickle” charging continues at 1mA for as long as mains power is applied.

The minimum charge burst is designed to provide battery back-up over a weekend after just a few minutes operation. A 10mf capacitor is connected across the clock chip supply connections to prevent loss of data in the event of accidental battery disconnection.

Configuration Status

Fifty bytes of CMOS RAM are available within the chip. Twenty of these are used by the operating and filing systems for initial configuration of the hardware. Of the remainder, ten are reserved for future use by ACORN, ten are for 'third party' use and the remainder are for the user.

Clock

The clock operates from a 32.768KHz crystal oscillator. A trimming capacitor is provided as is a test point with the buffered clock output. Year, month, day, hour, minute and second information is provided with automatic leap year (but not automatic leap century) correction. An alarm is also included within the chip, but there is no operating system support for this facility. An optional nIRQ connection can be made to the CPU from the clock chip, enabling the alarm to change program flow.

Operation of the clock chip in this manner involves direct manipulation of the chip control signals and should only be attempted by competent programmers. Acorn Computers are not responsible for incorrect programming by the user/software supplier.

If power is removed during an access to this chip, the chip select will become invalid, with the possibility of write accesses being corrupted. This is avoided by inverting the chip select with a transistor whose collector resistor is connected to the battery backed supply. As power fails to the main circuitry, the transistor base current reduces and the transistor switches off, deselecting the chip.

1MHz Internal I/O

Various devices operate at a 1MHz bus rate. Only one internal I/O component works at this speed - the system VIA.

System VIA

A 6522 allows several sources to create maskable interrupts. The sources are:-

a) CRTC vertical synchronisation.
b) A-D converter; end of conversion signal.
c) CRTC light pen strobe.
d) Keyboard key detect.

It also provides the slow data bus.
Port B on this device generates and reads a number of internal hardware strobes.

These are :-

Port B Data Strobe Active Level

Port B Data	Strobe	Active Level	
D7 D0			
D X X X X X X X	Clock Address	H	
X D X X X X X X	Clock chip enable	H	
X X D X X X X X	'Fire' button 2	Input	
X X X D X X X X	'Fire' button 1	Input	
X X X X D 0 0 0	Sound chip select	L	
X X X X D 0 0 1	Clock R/W	Q	H=read
X X X X D 0 1 0	Clock Data	H	
X X X X D 0 1 1	Keyboard enable	Q	H=auto-scan
X X X X D 1 0 0	C0 } Screen control	Q	
X X X X D 1 0 1	C1 } signals	Q	
X X X X D 1 1 0	Caps Lock indicator	L	
X X X X D 1 1 1	Shift Lock indicator	L	

Note: Q is the value of D after the port write operation is completed

2MHz Internal I/O

Only one internal I/O component operates at this clock rate, the internal second processor TUBE. Its data bus is connected directly to the CPU data bus. The second processor interface will only be specified as a hardware data transfer definition. In this way, the actual second processor used will not be constrained by this specification.

The interface is a parallel port providing the following data access signals:-

i) D0 to D7 A bi-directional bus to TTL levels.
ii) A0 to A2 A unidirectional bus to CMOS levels.

The following control and timing signals are provided:-

Host CPU phi2	CMOS levels
System Reset	TTL levels
Host CPU nIRQ	This must be an 'open collector' node with an active low TTL level
8MHz timing reference	TTL levels
TUBE chip select	CMOS levels
Read/Write	TTL levels

External Input/Output

1MHz External I/O

Analogue Port

This 15-way D-type connector provides access to an NEC mPD7002 four-channel, ten-bit analogue-to-digital converter. The sampled input is compared to a 1.8V reference derived from three small signal diodes in series.

A tracked link may be cut to deselect this reference. The user may then solder in a two-pin precision reference in the holes provided or supply an external reference. Any user supplied reference should have a maximum voltage of 2.5V.

An input voltage on any one of the four channels will be digitised when the A/D control register is so instructed. Conversions are in the range 0 to 1.8V.

The voltage reference is made available at the connector. Provision is made on the board for an additional high stability reference, if required. A link will have to be made for the additional reference to be used. Conversions take place in 5mS and the “end of conversion” pulse causes an IRQ to be generated by the system VIA.

Two “fire buttons” are provided for with the connections I0 and I1. These are connected to the system VIA and cause interrupts (as IRQ) to be generated.

Light Pen

A light pen may be connected to the signal LPSTB. This also causes the system VIA to generate an IRQ (if enabled). It also causes the 6845 CRTC to latch the address of the currently selected video data byte. This may not be the same as the displayed byte and some software correction may be necessary. Factors such as phosphor characteristics, light pen response and the angle at which the pen is used, may all affect the correction needed.

2MHz External I/O

Two peripheral devices operate at 2MHz. These are the external second processor connection and the ECONET connection.

External Second Processor

This interface has a buffered data bus via the Peripheral Bus Controller (PBC). The EXbus on this component provides for good data set up and hold times. Together with a limited degree of line matching, this ensures reliable high speed data transfer

with unspecified cable lengths. A maximum cable length of one metre is suggested to prevent noise problems.

The interface operates at 2Mhz. This means that if a 1MHz bus peripheral is also connected, then the address and data buses on this connector will appear to perform both 1 and 2MHz cycles.

The connections are:-

D0 to D7	Data Bus	CMOS levels
A0 to A7	Address Bus	TTL levels
IRQ	Interrupt Request	Open collector TTL levels
nTUBE	Parasite chip select	TTL levels
Supply		+5V
Ground		0V

2 CIRCUIT DESCRIPTION

This chapter should be read in conjunction with the circuit diagram at the rear of this manual.

The microprocessor used in the Master 128 is a 65SC12 running at either one or two megahertz clock rate. Most processing is done at 2MHz, including accesses to the Random Access Memory and Read-Only Memory. The processor slows down to 1MHz when addressing slow devices such as the 1MHz Extension Bus, the Analogue to Digital Converter and the Versatile Interface. A 16MHz crystal oscillator provides clock signals for the microprocessor in conjunction with divider circuitry on the video processor (VIDPROC) uncommitted logic array chip (IC42) which produces 8, 4, 2 and 1MHz signals.

Random Access Memory on the microcomputer is provided by four 4464 dynamic memory devices (ICs 17,18,23,26). Row-address and column address strobe signals for these RAMs are generated from the 8, 4 and 2MHz clock signals. These RAMs are cycled constantly at 4MHz. Two devices may have control of the RAM address lines, one is the 65SC12 microprocessor and the other is the 6845 Cathode Ray Tube Controller chip (IC22).

The CRTC generates the raster scan signals for the video display, together with the address for each memory-mapped byte of information in the RAMs which is required to refresh the display. An MSI CRTC multiplexer (IC31) switches control of the RAM address lines between the microprocessor and the CRTC.

The 65SC12 microprocessor is particularly suitable for this kind of application, because it runs from a constant clock, $\phi 2$, and so its requirements for memory access are predictable. Every 250ns, control of the RAM address lines is switched between the microprocessor and the CRTC. Thus, in a one microsecond period, the microprocessor has two RAM accesses and the CRTC has two RAM accesses. Because the CRTC generates a sequence of addresses in order to refresh the display, the row address lines of the RAMs are constantly cycled. Careful design of the addressing methods in each screen mode ensures that the dynamic RAMs are also refreshed by the sequential CRTC accesses.

Using this technique, two bytes of information are available per microsecond for refreshing the raster scanned video display. With each horizontal line having a period of 64ms, a 40ms active display area is usual. Thus, 640 bits of information per horizontal line are produced from the memory-mapped display. The video processor VIDPROC (IC42) is a custom Uncommitted Logic Array developed by Acorn. At the end of each CRTC 250ns access period, it latches the byte from the

RAM and, according to the display mode in operation, serialises the byte into a one-bit stream of eight bits or a two-bit stream of four bits etc. In this way, display modes varying from 640 pixels in 2 colours to 160 pixels in eight colours, which may be flashing, can be produced.

The video processor also contains a high speed block of static random access memory called a palette. This memory can be programmed to define the relationship between the logical colour produced by the RAM and the physical colour which will appear on the display. Thus, in a 640 pixel mode, the two colours to appear on the display need not be black and white, they may be, say, red and blue. The information in the RAM is unchanged by the palette; it is its interpretation into physical colours which changes.

Modes 0-6 in the microcomputer use software-generated characters, that is to say, the character font to be produced on the screen is held in the memory mapped display area of the RAM and graphics or characters may be held. This method of producing characters is expensive in memory, involving a minimum of eight kilobytes for the display memory.

Display Mode 7 is a Teletext mode implemented by an SAA5050 (IC32) Teletext character generator. IC15 latches the information coming from the RAM prior to the SAA5050. When using this mode, only 1K of RAM is devoted to the display memory and the characters are held within it as ASCII bytes. The SAA5050 then translates these bytes into a standard Teletext/Prestel format display.

The red, green and blue logic signals produced by the video processor are buffered by MSI CHROMA chip (IC40) and fed out together with a composite sync signal to the RGB connector. This output is suitable for feeding straight to the gun drives of RGB monitors. The red, green and blue lines are summed by binary weighted resistors to feed Q13 which produces a 1v composite video signal suitable for feeding to monochrome monitors, on which the different colours will appear as different shades of grey.

A modulator provides a UHF TV signal on channel 36, suitable for feeding to the aerial input of a domestic television receiver. Colour is derived from a PAL (phase alternating line) encoder circuit which modulates the colour information on to the colour subcarrier frequency. A 17.7345MHz oscillator circuit is divided by a ring counter (IC43) giving an output at the colour subcarrier frequency of 4.43361875MHz which is fed to IC40. This selects different phases of the ‘U’ and ‘V’ signals according to whether a red, green, blue, cyan, magenta, yellow or white colour is to be produced. These signals produce the colour subcarrier signal which is added to the monochrome output from Q11 by the buffer Q12. A reference colour burst is provided at the beginning of each line for the receiving television to interpret the colour information.

The PAL signal may be added to the 1v video connector by the insertion of a 470pF capacitor between the emitter of Q12 and the base of Q13.

Resistors R132-4 adjust the luminance balance of the colours.

Memory provision comprises four 4464 dynamic RAM chips (IC17, 18, 23, 26) which give 128 kilobytes of storage and a one megabit ROM (IC24) mapped as eight 16K blocks.

Input/output is under the control of an MSI I/O controller IC15. This is connected directly to the control lines of the executive chips responsible for peripheral access.

One 6522 VIA device (IC8) is devoted to internal system operation. Port B drives an addressable latch which is used to provide read and write strobe signals for the RTC/CMOS chip, the keyboard and the sound generator chip. Also coming from this latch (IC10) are control lines C0 and C1 which indicate the amount of RAM devoted to the display memory to be 16K, 8K, 10K or 20K. Pins 6 and 7 of the addressable latch drive the caps lock and shift lock LEDs on the keyboard.

The rest of Port B on the internal system VIA is used to input the two 'fire button' signals from the analogue to digital converter interface and to control a real-time clock/CMOS RAM chip. Each time the system VIA is written to, any changes on Port B which should affect the addressable latch are strobed into the latch by a flip flop which is triggered from the 1MHz clock signal. Port A of the system VIA (IC8) is a slow data bus which connects to the keyboard, the RTC/CMOS RAM chip and the sound generator.

IC12 is a four channel sound generator chip which may be programmed to give varying frequency and varying attenuation on each channel. An extra analogue input from the 1MHz extension bus is added to the sound generator signal and then filtered by a quad operational amplifier (IC9). IC13 provides audio power amplification to drive a speaker.

Two forms of serial interface are provided, one is an audio cassette at either 300 or 1200 baud and the other is RS423, over a whole range of baud rates. (RS423 is electrically compatible with RS232C in most applications.)

A 6850 asynchronous communications interface adaptor (IC45) is used to buffer and serialise or deserialise the data. A second ULA (SERPROC) is used in the serial interface (IC48). Contained within this ULA is a programmable baud rate generator, a cassette data/clock separator and switching to select either RS423 or cassette operations. IC40 divides the main board 16MHz clock by 13 and this signal is divided further within the serial interface ULA to produce the 1200 Hz cassette signal.

Automatic motor control of an audio cassette recorder is achieved by a small relay driven by a transistor from the serial interface ULA. The signal out of the cassette is buffered and the incoming signal is suitably filtered and shaped by a three stage amplifier. This is a quad operational amplifier (IC46). The RS423 data in and out signals and request-to-send and clear-to-send signals are interfaced by ICs 50 and 51 which translate between TTL and standard RS423/232 signal levels. This is one of the few sections of circuitry on the Microcomputer which requires an additional 5v supply to be present.

A four-channel analogue to digital converter facility is provided by a mPD7002 IC49. This device connects straight to the microcomputer's data bus and it is a dual slope converter with its voltage reference being provided by the three diodes, D17, D18 and D19.

Connection is made to the ECONET by a five way DIN connector mounted on the main circuit board. The interface electronics including the 68B54, line drivers, receivers and chatter disconnect components are mounted on a separate circuit board. This board has two connectors:-

a) A 5-way connector which has a one-to-one connection with the DIN connector.
b) A 15-way connector provides the CPU data bus together with address, timing reference, chip select and interrupt signals. The main pcb has two further address connections for future expansion.

A 6854 Advanced Data Link Controller circuit handles the Econet protocol. Data to be transmitted onto the network is fed from the ADLC to the line driver circuit which produces a differential signal drive to the Econet cables. Received data is detected and converted to a logic signal by one half of IC5 which is a dual compare circuit type LM319. The received data is then fed back to the data link controller circuit.

An Econet installation has a external master clock station which controls the timing for the network. This clock signal is transmitted around the network as a second differential line signal and it is used to clock the data in and out of the data link controller circuits. The network clock is also detected using one half of the LM319 comparator IC5 and the detected clock is then fed to both receive clock and transmit clock inputs on the 6854. In the presence of a network clock, the monostable circuit, IC2 is permanently triggered and this provides a data carrier detect signal for the data link controller chip. Once the network clock is removed, the monostable immediately drops out and the data carrier is no longer detected.

Econet is a broadcast network system on which a number of stations may attempt to transmit their data over the network at any given time. In this case, a collision can occur; the transmitting station detects the collision and backs off before attempting to try again to transmit over the network. Collision arbitration software is

included in the Econet system. Collisions on the network data lines result in the differential signal on the two data wires being reduced and this condition is detected by IC6 which is another dual comparator circuit.

When there is a good differential data signal on the network one output of IC6 or the other will be low, in which case the output of IC2 Pin 6 will be high, indicating no collision. When there are no collisions on the network, and the network clock is detected by the clock monostable, the data link controller is clear to send data over the network.

When there is a collision on the network both outputs of IC2 will go high and the clear to send condition will cease. Note that when the computer is not connected to the network a collision-like situation results, in which case again the data link controller will not get a clear to send condition.

Each Econet system requires termination at the two extreme ends of the network with network terminator boxes. It also requires an external network clock box. The network clock generates a 6MHz signal which is divided by two to produce 3MHz and other clock rates down to 75KHz. The setting of this clock signal depends on the length of the network, with the longer networks requiring a slower clock.

Up to 255 stations may be connected to each Econet with each station being identified by a unique station identification number. This station ID is programmed into the battery-backed CMOS RAM. The data link controller circuit produces interrupts which are fed to the central processor NMI line. These interrupts are enabled every time the station ID is read. Once in the data link controller interrupt service routine the DTR output of the ADLC goes low in order to remove the interrupt.

IC3 is a WD1770 or WD1772 floppy disc drive controller circuit which is used to interface to one or two single or double sided 5 or 8 inch floppy disc drives. Logic signals from the controller to the disc drive are buffered by IC1.

IC6 is a versatile interface adaptor. Port A is used to provide a centronics standard parallel printer interface, with the octal buffer IC5 being used to buffer the data lines. Port B is left uncommitted and is free for use by the user for input or output purposes.

The address and data lines A0-A7 and D0-D7, together with some page select lines are available as the 1MHz extension bus to which various peripheral devices, such as Teletext interface, may be connected. All accesses to this bus will be at 1MHz processor speed. The octal buffer IC7 and the Peripheral Bus Controller IC21 are used to interface these signals to the internal data address bus.

Selected address and data lines are available on the Tube connector which is used to connect second language processors into the system.

Keyboard

Ninety-three keys are provided, ninety-two of which are in a modified 8x13 matrix. A keyboard encoder, KBDENC (IC16) is used to scan the keyboard. During idle (free run) mode, pressing any key will cause an IRQ to be generated via the system 6522. A connection is provided from IC16 to a 6522 'CA' type connection. Hence the interrupts thus generated are controlled by the 6522 control register. Depression of either of the shift keys, or the control key does not generate an interrupt.

The power supply unit produces 5 volts at around 2 amps and -5 volts at around 50mA for use on the main circuit board. Auxiliary power for accessories is available on an external connector.

DETAILED CIRCUIT OPERATION

In this section, certain parts of the circuit will be described.

Pins 4, 5, 6, and 7 of the video processor (IC42) produce 1, 2, 4 and 8MHz clocks in phase. The 1MHz signal from the ULA is passed through a D-type flip flop (half of IC28) in order to produce the system 1MHz clock. A 2MHz signal of suitable phase is produced at the output of 74S00 (IC34) producing the normal 2MHz clock input to the microprocessor. Switching the CPU down to 1MHz from the 2MHz it is normally running at when accessing a slow peripheral, as required in the A, B and B+ models, doesn't happen in the Master. Instead, the Peripheral Bus Controller (IC21) takes care of this, dividing the bus into fast and slow sections.

At the appropriate time, as governed by the 2MHz clock, one of the 2MHz clock cycles is marked off by the D-type (half of IC28) and when this happens the D-type that remembered that a request had been made is cleared.

A 6MHz clock signal is required for the Teletext character generator (IC32). This signal is produced by knocking a reset flip flop (two quarters of IC39) backwards and forwards from 8MHz and 4MHz clock signals. The resulting flip flop output is then itself inverted according to the state of the 2MHz clock signal by an exclusive OR gate (of IC25). Glitches on this output are removed by R59 and C30 to produce the 6MHz clock signal at Pin B of IC25.

The dynamic RAMs are constantly cycled by a row address strobe signal which is produced by the CRTC multiplexer (IC31). The column address strobe is produced by part of two 74S00 (ICs 34 and 38). Two devices may have control of the RAM address lines: the 65C12 CPU (IC14) and the 6845 CRTC (IC22). Selecting which of these two devices control the DRAM address lines is done through the CRTC multiplexer (IC31) and address multiplexers (ICs 29, 30 and 33).

The video processor uncommitted logic array takes data bytes from the RAM at the rate of sixteen bits per microsecond and then serialises them according to the display mode required. The bit streams for serialisation are then fed through a block of high speed palette RAM which relates the logical colour from the serialiser to the physical colour to be produced on the display. The palette drive is 16x4 bits with the four bits representing red, green and blue drives, together with a flash bit. The data bus input to the video processor is also used to access the mode control register when the device is chip selected. In the Teletext display mode, RGB information is fed straight into the video processor from the SAA5050 for the cursor control to be added.

VDU throughput is much enhanced by the use of hardware scroll. A register in the CRTC is used to store the start of screen address in the screen memory. Thus, in order to scroll the screen, it is only necessary to increment this register by the number of characters per line and then write to the memory address where the last screen data was and where the new screen line data now needs to go.

The number of address lines from the CRTC used to address the screen memory has to be sufficient to cater for the biggest screen, which is 20 kilobytes, therefore, sufficient addresses to satisfy 32 kilobytes of screen memory are used. By the hardware scrolling technique the picture rolls around in 32 kilobytes. For example, with a scroll of eight kilobytes in a 20 kilobyte screen, the original start of screen for the 20 kilobyte mode was &3000. After the eight kilobyte scroll, the current start of screen address is &5000 with the end of the screen as viewed by the CRTC at &5000 plus 20 kilobytes, that is &A000.

The address &A000 is not physically in the RAM and it is therefore necessary to modify this address in order to move it to the original start of the screen. This is done by adding 12 kilobytes to get the required physical address. In this way, the physical memory addresses are kept within the required range. For the different screen modes we need to add different numbers as their start of screen addresses are different.

The following table shows this:-

Modes	Screen Size	Start of Screen Address	Number to be added
0,1,2	20K	&3000	12K
3	16K	&4000	16K
4,5	10K	&5800 (or &1800)	22K
6	8K	&6000 (or &2000)	24K

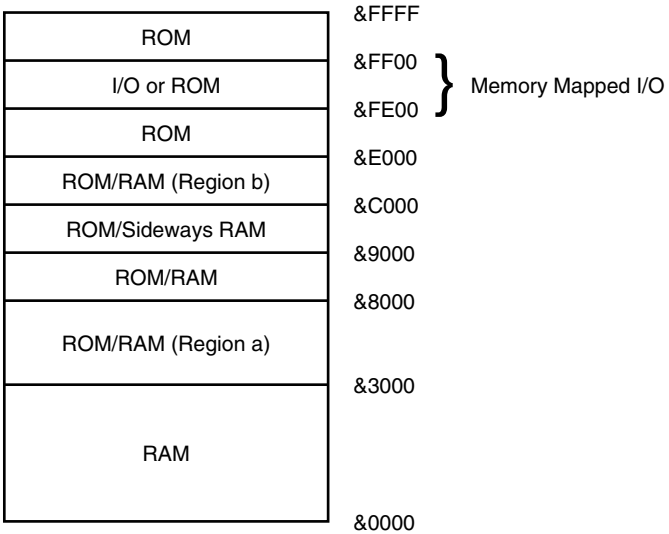
The number to be added to the start screen address in order to keep the hardware scrolling within the correct physical memory address range is defined by the control lines C0 and C1 from 74LS259 (IC10). This number is then computed with the result being added to the higher CRTC refresh address lines by the CTRC multiplexer (IC31).

3 MEMORY ORGANISATION

Operation of the RAM and ROM is controlled by the Memory Controller integrated circuit. The principal function of this device is to control the memory paging structure.

Memory Map

The 65C12 can directly address 64K locations. As over 1/2 Mbyte may be resident, a paging scheme is implemented.



Machine memory map

The current memory map is dictated by the contents of the two latches. ROM SELect and ACCess CONtrol located at &FE30 and &FE34 respectively. The contents of these two latches are:-

	d7	d6	d5	d4	d3	d2	d1	d0
(&FE30)	RAM	0	0	0	PM3	PM2	PM1	PM0
(&FE34)	IRR	TST	IFJ	ITU	Y	X	E	D

The contents of ROMSEL dictate the selection of memory which resides from &8000 to &BFFF.

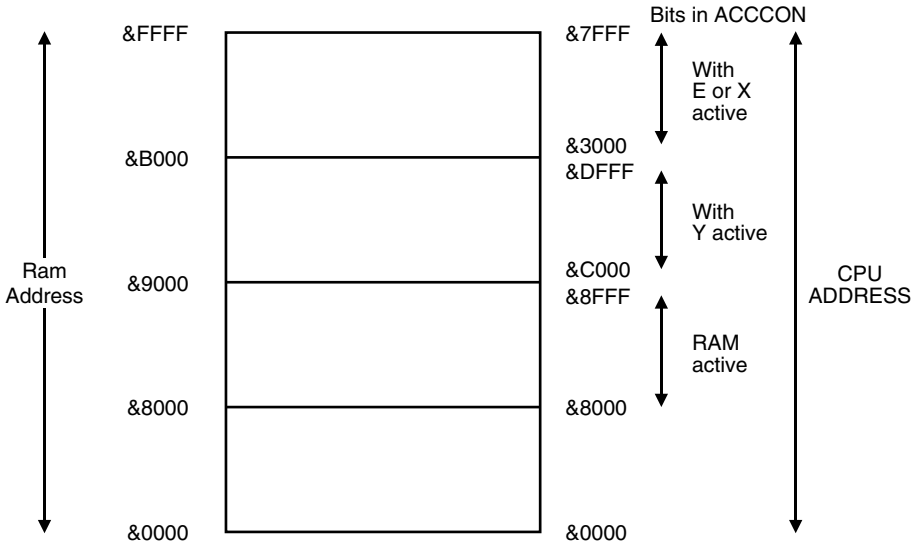
The contents of ACCCON principally dictate the activity of two regions of memory:

(a) &3000 to &7FFF
(b) &C000 to &DFFF

Random-Access Memory

RAM is functionally split up into two regions. The main region supports the language workspaces, buffers etc. and provides the bit-mapped screen. The second region provides four 16K “Sideways” RAM segments. These are link-selected into ROM locations 4,5,6 and 7. They may be deselected, reinstating the ROM sockets in blocks of 32 Kbytes.

Within the main 64 Kbyte region, the lower 32K is used within the &0000 to &7FFF region of the CPU memory map. The 64K of DRAM is distributed as follows:-

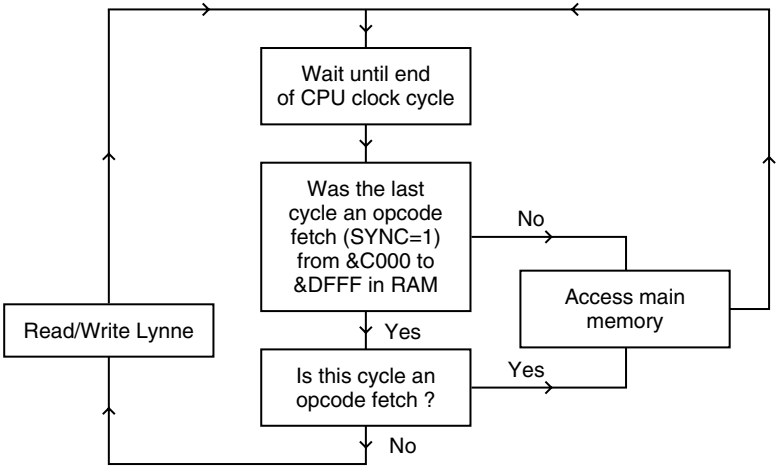


Summary of RAM memory map

The upper 32K is split up into three, self-contiguous regions. The largest portion of this is a 20Kbyte region designated LYNNE. This can be overlayed on the region (a) of main memory.

When bit D in ACCCON is set, the CRT controller will display the contents of LYNNE. When bit D is cleared, the region (a) of main memory will be displayed.

When bit E in ACCCON is set, if the address range is &3000 to &7FFF the CPU will read/write Lynne according to the flow shown below



This system allows for the screen bit map to be removed from the main CPU memory map of which it occupies a significant proportion. It will, however, only work if the screen is being accessed by opcodes from a known region - i.e. the MOS VDU drivers.

A mechanism is also provided to permit 'illegal' screen access. Bit X in ACCCON, when set, causes all accesses to region (a) to be re-directed to LYNNE. This occurs irrespective of the opcode address, hence considerable care must be exercised in its use. When cleared the memory map returns to its usual format.

In the same way that the BASIC variable HIMEM will always have the value &8000 when LYNNE is used, it is desirable for the variable PAGE to have the value &E00, irrespective of the current filing system. This is achieved by providing a filing system workspace. Bit Y in ACCCON when set, causes 8Kbyte of RAM, referred to as HAZEL, to be overlayed on the MOS VDU drivers, i.e. from &C000 to &DFFF. When this bit has been set, no calls may be made to the MOS for VDU operation. The code which performs this paging operation is responsible for resetting the Y bit, as no hardware is provided for this purpose.

The remaining bits in ACCCON are used to control various peripheral systems.

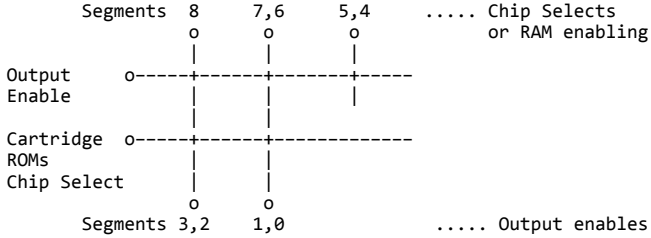
ITU, when set, enables the CPU to access the internal second processor rather than the external one.

IRR is InterRupt Request. When set, this bit causes an open drain output to pull the CPU NIRQ pin down to Vss.

ROMSEL

The contents of ROMSEL determine the paging of memory in the 16K region &8000 to &BFFF. One of sixteen 16Kbyte ROM memory segments may be selected. One additional 4Kbyte RAM segment may be selected from &8000 to &8FFF.

Eight of the segments are assumed to be in four 32Kbyte ROMs where the least significant bit of ROMSEL selects between the upper and lower segments. Seven of the segments exist together with a ROM which is active from &C000 to &FFFF within a 128Kbyte ROM. This ROM is connected via a separate data bus. The four 32Kbyte devices and one 16Kbyte device are connected in a matrixing scheme.

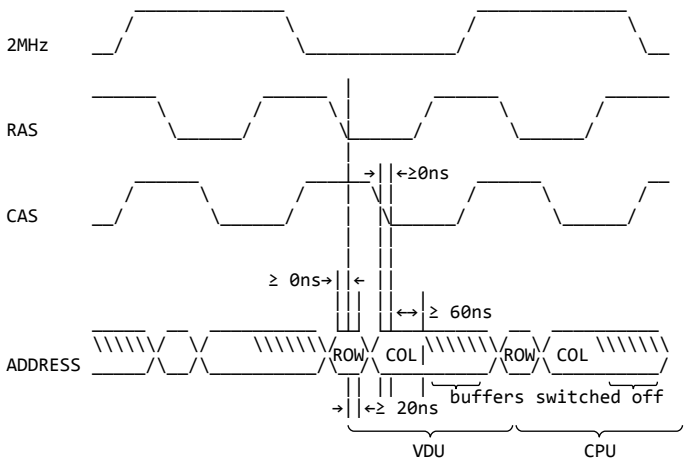


In this way, fewer connections to the controller logic are required to select a given ROM, although the power dissipation will be increased if all the ROMs in one column are inserted. A chip select will be driven low if an access to one of the segments (4 to 8) is required. If a cartridge ROM is required, then the Cartridge ROM chip select will be driven high. All chip selects are a decode of the CPU address most significant nibble. An output enable is turned active low during the CPU d2 period depending on which segment is required. The segment to be selected is determined by the binary number held within the least significant nibble of ROMSEL.

Overlaid RAM in ROM area

When the bit RAM is set in ROMSEL, accesses to the region &8000 to &8FFF are redirected from the currently selected ROM to a region of RAM referred to as ANDY. It is the responsibility of the code which set RAM to clear it after accessing ANDY. This is necessary to ensure correct operation of software in ROM.

A further 64 Kbyte of RAM is available as four pages of 16 Kbyte from &8000 to &BFFF. The ROM slots 4,5,6 and 7 are not active when this RAM is link-selected to be active.



DRAM timing

RAS is generated from 4M and 8M by the D-type IC28 pin 9. CAS for the main DRAMs is generated from 2M, inverted by a NAND in IC34 to give phi2 IN, gated with DRAMEN which enables the main RAM, and finally gated with 4M through another NAND in IC34.

4 SLOW DATA BUS

Several internal components need to work with access cycles slower than the CPU's normal 1 or 2 MHz rates. These are:

1) Keyboard
2) Sound Generator
3) Real Time Clock/RAM

Direct access of these devices is not recommended, as their operation may be subtly related to other functions, or be time-critical, or could cause malfunction if not performed correctly. The same functions may be provided by completely different hardware in earlier or subsequent products. For those who need direct access, rather than using the MOS, it is advisable to disable interrupts whilst accessing any of these devices because the MOS may change some of the settings whilst servicing an interrupt from another source.

Memory Locations

All these devices are accessed through the System VIA located at &FE40-9. The Slow Data Bus is connected to the 8-bit A port at &FE41. This is referred to as PA[0:7]. The B port at &FE40 is the control bus.

Slow Data Control Port (&FE40)

Writing the following values will have the indicated effect:

PB[7]	`DXXX XXXX`	- RTC/RAM Address strobe	: Active high
PB[6]	`XDXX XXXX`	- RTC/RAM Chip select	: Active high
PB[0:3]	`XXXX D111`	- Shift lock	: Active low
PB[0:3]	`XXXX D110`	- Caps lock	: Active low
PB[0:3]	`XXXX D101`	- Hardware Scroll 1 (HS1)	
PB[0:3]	`XXXX D100`	- Hardware Scroll 0 (HS0)	
PB[0:3]	`XXXX D011`	- Keyboard Enable (KBEN)	
PB[0:3]	`XXXX D010`	- RTC/RAM Data Strobe	: Active high
PB[0:3]	`XXXX D001`	- RTC/RAM Read Write	: High for Read
PB[0:3]	`XXXX D000`	- Sound Generator write	: Active low

`D` is set high or low as needed.
The hardware scroll bits HS[0:1] are used in VDU control.

Keyboard

The keyboard is accessed as a matrix of 8 rows by 13 columns. To access any particular key, it is necessary to assert KBEN and set the column and row addresses of that key on port A thus:

PA[3:0] (outputs) are the column address
PA[6:4] (outputs) are the row address
PA[7] (input) is the key output - active low if pressed.

An interrupt will be caused by CA2 via R13[0] (bit 0 of &FE4D) whenever a key is pressed.

Sound Generator

Within the MASTER 128, the sound generator chip is write-only. The write strobe must be asserted low for the data PA[0:7] to be written into it. Data must be stable during the 8ms in which the write strobe must be low.

Real-time clock/CMOS RAM

Fifty bytes of battery-backed CMOS RAM are available within the real-time clock chip. Twenty bytes are used to store the system configuration, ten are reserved for future use by Acorn, ten are reserved for use by third-party manufacturers and ten are available for use by the user. Extreme care should be taken in the direct control of this device to ensure integrity of the computer's configuration status. The MOS should be used for the normal reading/writing of the RAM. FX calls 161 and 162 (OSBYTES &A1,&A2) are used to access the RAM. OSWORDs 14 and 15 should be used to read/write the time.

CMOS RAM Allocation

Address (offset)	Function
0	Station Number
1	File server station number
2	File server bridge number
3	Printer server station number
4	Printer server bridge number
5	Default filing system/language
6-7	ROM frugal bits (set/cleared by *INSERT/*UNPLUG)
8	EDIT start-up settings
9	reserved for telecommunications applications
10	VDU Mode and *TV settings

11	ADFS start-up options and floppy drive parameters
12	Keyboard auto-repeat delay
13	Keyboard auto-repeat rate
14	Printer ignore character
15	Default printer type, serial baud rate, ignore status and TUBE select
16	Default serial data format, auto boot option, internal/external TUBE use, BELL amplitude
17	ANFS configuration control (on hard reset) bit 0 : Claim two static workspace pages bit 1 : Findlib bootstrap option bit 2 : Static workspace at &0B00/&0E00 bit 3 : User/Application bit 4 : User/Application bit 5 : User/Application bit 6 : Reserved for ANFS protection mechanisms bit 7 : Display version messages
18-19	Master Compact century, joystick and country
20-29	Reserved for future use by Acorn
30-45	For ROMs 0-15 (one per ROM)
46-49	Available for user applications

Note that the station number cannot be written to, and has to be accessed by code similar to that listed in the RTC alarm section.

Real Time Alarm Functions

The MOS does not provide control of the device's alarm facilities as these are only available on a daily basis, i.e. the alarm cannot be programmed to operate on a specific date. The alarm operates by generating an interrupt when the real time counters are equal to the alarm time registers.

The connection of the clock chip to the system interrupt line is via a shorting bar on Link 4. This would have to be fitted by the user. For the user willing to reserve some of the other battery-backed RAM for the target date, the following routine should be used to access the alarm and control registers. It is similar to those within the MOS and obeys the rules for reliable operation. It is in the style of BBC BASIC assembler.

```
pbq=&FE40                :REM Port B
paq=&FE41                :REM Port A
ddraq=&FE43              :REM Port B data direction register
                         :REM "1" = Output
                         :REM "0" = Input
```

```
EQUB &02:EQUB pbq        DS active
EQUB &82:EQUB pbq        Address strobe inactive
EQUB &FF:EQUB ddraq      Outputs
EQUB &0E:EQUB paq        slow bus address (see note 1)
EQUB &C2:EQUB pbq        chip select active
EQUB &42:EQUB pbq        Latch address
EQUB &41:EQUB pbq        Select write mode
EQUB &FF:EQUB ddraq      Outputs
EQUB &4A:EQUB pbq        Data strobe active
EQUB &00:EQUB paq        Write the data (see note 2)
EQUB &42:EQUB pbq        Data strobe inactive
EQUB &02:EQUB pbq        Chip select inactive
EQUB &00:EQUB ddraq      Inputs again
```

Note 1 This address should be made variable as it will be necessary to access one of a number of registers.

Note 2 Separate sequences may be necessary for read and write operations, depending on personal preferences.

RTCRAM Access Restrictions

The real-time clock section of the chip is updated from the real-time counters once every second. It is important that the user program does not try to access them at the same time as this will give erroneous results. There are three ways that the chip gives notice that it is in the process of updating the registers. These are documented in the manufacturers data sheet. Where possible it is recommended that an alternative approach be used which ensures user access. This is to set the SET (bit 7) flag in Register &B (the control register). It prevents the chip from updating the registers but does not affect the “counted” time. When the SET bit is reset, the registers will be reset to the current time approximately within the next second. Avoidance of this critical region, or the overriding of it, must be done whenever the real time or alarm registers are written.

The code should be assembled to operate in sideways RAM (ie in the region &8000 to &BFFF). The program is essentially in two parts:

a) To set the alarm time, an OSCLI command which will not conflict with any other in the machine, eg “*SETALARM hh:mm:ss” should be devised. This involves recognising Service Call &04 (Offer Command). The program should interpret the given time string as appropriate and load it into the alarm registers then re-enable the counter-register transfers and finally enable the alarm interrupt by setting the AIE (bit 5) flag in Register &B.

b) To respond to the alarm, the code should respond to Service Call &05 (Unknown Interrupt). The alarm flag - AF (bit 5) in Register &C should be examined to ascertain whether the alarm has occurred or not. If so, the appropriate action should be taken and the call should be claimed, otherwise the call should not be claimed. The interrupt will be cleared by reading register &C.

5 KEYBOARD CONTROLLER

Keyboard Operation

During free run mode, the keyboard column lines are continually scanned by incrementing a counter, decoding its outputs and pulling low a column line. Any key depressed will cause the interrupt to be generated. A signal, KeyBoard ENable is generated to stop free running mode. The counter contents are then loaded by CPU operation to determine on which row the key was pressed. The rows are then individually selected to determine which key was pressed. KBDENC is supplied with data from the slow data bus:-

PA0 to PA6 (slow bus connections):- PA0 to PA3 are the column select inputs and PA4 to PA6 are the row select inputs. PA7 is a three-state connection which is driven active low when a row/column combination describes a depressed key.

PA7 (row data bit output):- This 3-state output provides the ROW data signal to the host system. it is enabled by the nKBEN signal and its output is high if the row address set up on PA4-PA6 points to a row which is at logic low.

R0 to R7:- The keyboard row input connections are normally held high by internal pull-up resistors. If a key is depressed it will cause the appropriate row connection to be pulled low when its column is selected.

C0 to C14:- These open collector column driving outputs are sequentially taken active low in auto scan mode at a rate of 1MHz. In polled mode (nKBEN active low), the slow bus inputs PA0 to PA3 determine which output will be low. The selected column output is a direct decode of these inputs.

CA2:- Connected to the system VIA, this output will cause the VIA to generate an nIRQ. The line will be active low when an active key is detected.

nKBEN:- Generated by the system VIA, this line is taken active low to enable the row and column addresses to be determined by the Operating System.

MHz1:- Timing reference for the positive edge triggered counter and the reset generator circuit.

SWTI (switch input):- A transition from 5v to 0v or 0v to 5v on this input will cause an active low pulse of 200ms to be generated on pin22 (RSTO).

RSTO (reset output):- This open-drain output is triggered by a transition on the Switch Input pin SWTI and provides a logic low output pulse of at least 200ms. For example if SWTI is taken from 0v to 5v via a mechanical switch, the output will immediately fall to 0v, hold low for 200ms after switch bounce and then rise to 5v again.

VCC1 VCC2 (positive supply):- These pins must both be connected to the positive pole of a suitable power supply.

GND1, GND2 (ground):- These pins must both be connected to the power supply GND or RETURN line.

1	R0	VCC1	40
2	R6	MHZ1	39
3	R7	nKBEN	38
4	R2	PA4	37
5	R1	PA5	36
6	C11	PA6	35
7	C10	PA0	34
8	C12	PA1	33
9	C0	PA2	32
10	GND2	PA3	31
11	C2	VCC2	30
12	C9	PA7	29
13	C4	CA2	28
14	C5	R5	27
15	C6	R4	26
16	C8	R3	25
17	C7	C13	24
18	C3	C14	23
19	C1	RSTO	22
20	GND1	SWTI	21

KBDENC connections

The keyboard encoder scans the keyboard matrix, interrupting the CPU when a key is pressed. The MOS then puts the device in manual mode and scans the columns until it finds one where a key has been pressed. It then scans the rows until it finds one where a key has been pressed. It then goes on to check other columns and rows to find out if any other keys have been pressed. This continues at 10ms intervals (under the control of the system timer) until no keys are pressed, at which point the MOS switches the device back to automatic scanning. The operation of this circuit can be split into three modes.

Mode 1 - Free run

This is the state assumed during normal operating periods with no key pressed. The keyboard is constantly scanned, with no intervention from the CPU, until a key is pressed. A four-bit counter, clocked by a 1MHz signal drives a four-to-fifteen line decoder. This causes a logic low to ripple through C0 to C14. Should any key be pressed, the column in question will be connected to the relevant row, which will pull one of the inputs to the 7NAND gate low. As the other six inputs are all pulled high, the NAND output will go high and thus generate an interrupt signal on pin CA2.

Mode 2 - Column detection

The interrupt signal is registered in the host system which then takes a closer look at the keyboard. The Operating System keyboard scan routine is entered and individual addresses may be set up on PA0 to PA3. These are synchronously loaded into the counter while nKBEN is low, thus causing each keyboard column to be individually scanned. The interrupt CA2 may be examined after each counter load to see if the correct column has been reached. If this is so then the column address is held on the counter and stored for future reference, if not then the next address is loaded into the counter.

Mode 3 - Row detection

Having discovered and held the column address, the host may now set up addresses on PA4 to PA6. These are fed to an eight-way data selector and cause one of the eight rows to become available on the W output in an inverted state. Should the correct row be found, W will go high and the current address will be stored.

Keyboard Matrix

The keys are physically arranged as a QWERTY type keyboard with ten function keys, four cursor control keys and a nineteen-key numeric keypad.

	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
R7	ESC	f1	f2	f3	f5	f6	f8	f9	| \	RGT	4	5	2
R6	TAB	Z	spc	V	B	M	< ,	> .	? /	CPY	0	1	3
R5	SHIFT LOCK	S	C	G	H	N	L	+ ;	}]	DEL	#	*	,
R4	CAPS LOCK	A	X	F	Y	J	K	@	* :	RET	/	DEL	.
R3	! 1	" 2	D	R	& 6	U	O	P	{ [	UP	+	-	RET
R2	f0	W	E	T	' 7	I	) 9	0	£ _	DWN	8	9	
R1	Q	# 3	$ 4	% 5	f4	(8	f7	= -	~ ^	LFT	6	7	
R0	SHIFT	CTL											

INKEY NUMBERS

key	INKEY number		key	INKEY number	
f0	&DF	–33	ESCAPE	&8F	–113
f1	&8E	–114	TAB	&9F	–97
f2	&8D	–115	CAPS LOCK	&BF	–65
f3	&8C	–116	SHIFT LOCK	&AF	–81
f4	&EB	–21	CTRL	&FE	–2
f5	&8B	–117	SHIFT	&FF	–1
f6	&8A	–118	SPACE	&9D	–99
f7	&E9	–23	DELETE	&A6	–90
f8	&89	–119	RETURN	&B6	–74
f9	&88	–120	COPY	&96	–106

key	INKEY number		key	INKEY number	
A	&BE	–66	UP	&C6	–58
B	&9B	–101	LEFT	&E6	–26
C	&AD	–83	RIGHT	&86	–122
D	&CD	–51	DOWN	&D6	–42
E	&DD	–35	,	&99	–103
F	&BC	–68	–	&E8	–24
G	&AC	–84	.	&98	–104
H	&AB	–85	/	&97	–105
I	&DA	–38	[	&C7	–57
J	&BA	–70	\	&87	–121
K	&B9	–71	]	&A7	–89
L	&A9	–87	^	&E7	–25
M	&9A	–102	_	&D7	–41
N	&AA	–86	:	&B7	–73
O	&C9	–55	;	&A8	–88
P	&C8	–56	@	&B8	–72
Q	&EF	–17	*keypad* 0	&95	–107
R	&CC	–52	*keypad* 1	&94	–108
S	&AE	–82	*keypad* 2	&83	–125
T	&DC	–36	*keypad* 3	&93	–109
U	&CA	–54	*keypad* 4	&85	–123
V	&9C	–100	*keypad* 5	&84	–124
W	&DE	–34	*keypad* 6	&E5	–27
X	&BD	–67	*keypad* 7	&E4	–28
Y	&BB	–69	*keypad* 8	&D5	–43
Z	&9E	–98	*keypad* 9	&D4	–44
0	&D8	–40	*keypad* .	&B3	–77
1	&CF	–49	*keypad* ,	&A3	–93
2	&CE	–50	*keypad* #	&A5	–91
3	&EE	–18	*keypad* +	&C5	–59
4	&ED	–19	*keypad* –	&C4	–60
5	&EC	–20	*keypad* /	&B5	–75
6	&CB	–53	*keypad* *	&A4	–92
7	&DB	–37	*keypad* RETURN	&C3	–61
8	&EA	–22	*keypad* DELETE	&B4	–76
9	&D9	–39			

6 SCREEN DISPLAY

Screen Output

Three chips are primarily responsible for providing the screen output:-

a) Acorn VIDPROC ULA chip
b) 6845 Cathode Ray Tube Controller
c) Acorn CHROMA MSI video matrixing chip

The video processor takes a byte-wide data stream from memory, serialises it according to the screen mode in use, passes it through a palette to provide logical to physical colour transformation and on to the RGB outputs. From here the video data is buffered for connection to an RGB monitor and mixed for use with the composite video and colour television outputs.

High Resolution Modes

The 6845 generates a linear memory address sequence which increments every 0.5ms or 1ms, depending on the video bandwidth selected and video data format. The amount of memory reserved for screen use is also varied. The available options are

Video Data Formats

"Mode" Pixels/Byte	Format Bytes	Reserved Memory	
0	8	20K	
1	4	20K	
2	2	20K	
3	8	16K	
4	8	10K	
5	4	10K	
6	8	8K	
7	Teletext	1K	
128	8	20K	Reserved in LYNNE
129	4	20K	
130	2	20K	
131	8	20K	
132	8	20K	
133	4	20K	
134	8	20K	
135	Teletext	20K	

All modes except 7 and 135 display a bit-mapped image of the reserved memory. The 6845 may be re-programmed to display any arbitrary section of memory. If this is done, however, the hardware scrolling will not work correctly, as it assumes that the screen memory is in its usual location. The screen always ends at &7FFF and starts 1,8,10,16 or 20K below, depending on the selected mode. The selection of video bandwidth and data format is performed by programming the VIDPROC. The cursor size and position is also controllable by VIDPROC. Special measures have been taken to ensure correct cursor operation in the Teletext modes.

Teletext

The Teletext modes do not generate a bit mapped display, but a character cell one. The character/graphics ROM within a SAA5050 device generates RGB signals according to the desired character/graphics information within the reserved memory space. Each byte of memory is therefore just a definition of the character/graphics symbol required.

Other SAA505X devices may be used when different languages are required. Only 1 Kbyte of memory is needed for either of the Teletext modes, although 20K is reserved for it in mode 135. The MOS uses the spare 19K to speed up inter-filing system file transfers but the user may use this memory if no such transfers are to be done. VIDPROC has to be re-programmed to use the SAA5050 RGB outputs. The 6845 is still used to generate the cursor. As a delay of 2.75 ms will occur between reading a character from RAM and outputting the appropriate RGB signals, the 6845 has to be programmed accordingly. The “start” of screen signal is given a 1.5-byte time offset and the SAA5050 has a further one-byte time offset to restore the correct cursor/data phase.

VIDPROC has further adjustment which allows for the cursor to be adjusted to pixel accuracy.

Hardware Scroll

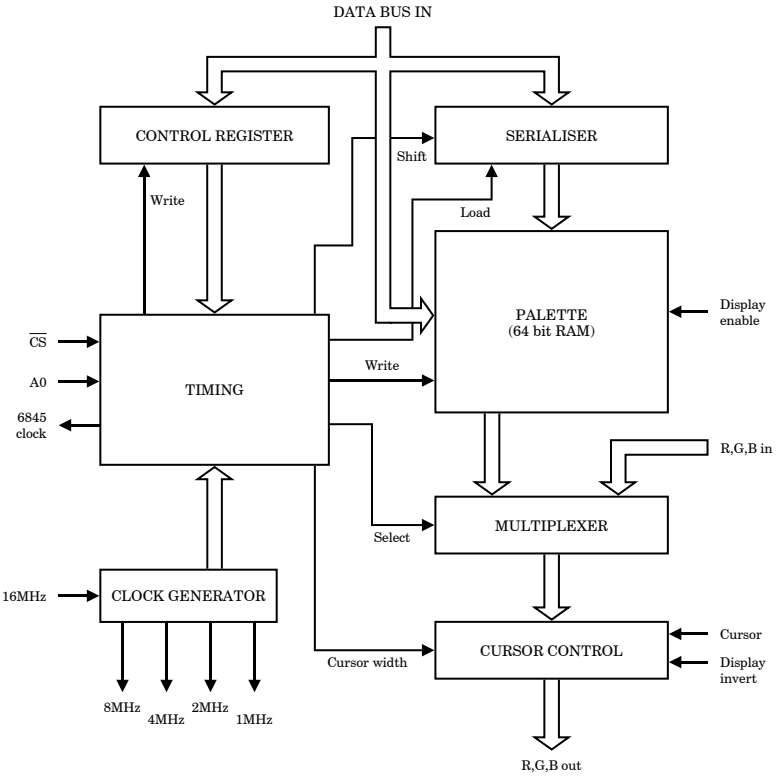
Scrolling may be achieved in any mode by re-programming the 6845 start of screen address to an integral number of video lines further down the memory map than the nominal start of screen. This causes the linear address generator to attempt to display an end of screen, which is out of the reserved video area. To overcome this effect, hardware scrolling is provided with a variable address wrap-around. When the address generator would otherwise attempt to access out-of-screen RAM, its addresses are modified to point to the gap between the original start of screen and scrolled start of screen. When this is done, only the end of screen needs to be written over in RAM. (If this is not done, the entire screen appears to “roll-over”). The amount of modification to be used is controlled by two nodes; C0 and C1.

Video Output

Three outputs are provided for displaying video data. These are:

a) PAL/NTSC encoded, UHF carrier. On channel 36 with 1.5mV into 75 ohm.
b) Composite video. This is a 1v peak-to-peak signal.
c) Digital Red-Green-Blue (RGB) - these are approximately 75 ohm outputs.

For use with NTSC, the modulator has to be changed from UM1233/E36 to a VHF equivalent. Provision is made for selection of either one of two channels with VHF. A Molex type link has to be inserted for this.



Block Diagram of the Video Processor

Control Registers

There are two control registers. The first contains miscellaneous control functions, the other dictates the contents of the palette.

Miscellaneous functions control register (write only)

bit #	function	parameters
0	select flash colour	0 - first colour 1 - second colour
1	teletext/high res	0 - palette output 1 - teletext output
2-3	bits/pixel	00 - 10 chars/line 16 colours 01 - 20 chars/line 16 colours 10 - 40 chars/line 4 colours 11 - 80 chars/line 2 colours
4	6845 clock rate	0 - 1MHz (Modes 4-7) 1 - 2MHz (Modes 0-3)
5-6	cursor width	00 - 1 byte 01 - not used 10 - 2 bytes (Modes 1,5, and 7) 11 - 4 bytes (Mode 2)
7	main cursor width	0 - small 1 - large

Notes

bit 0 is re-programmed by the MOS at intervals to cause physical flashing colour to alternate between its standard values and the (binary) logical complement.

bit 1 dictates whether the RGB signal supplied to the external buffers comes from the palette output or the Teletext character generator.

bits 5-6 The cursor is “on” for a number of byte-times, depending on the screen mode.

Palette Control Register (write only)

bits 0-3 - physical colour
bits 4-7 - logical colour

These are programmed together so that a certain physical colour is associated with a particular logical colour.

In two colour modes, bit 7 dictates the colour
- Eight locations must be programmed for each logical colour.

In four colour modes, bits 7 and 5 dictate the colour
- Four locations must be programmed for each logical colour.

In eight colour modes, Bits 7 to 4 dictate the colour
- One location must be programmed for each logical colour.

The principle is that the remaining locations must be set to the same value as the selected logical colour. If bits 7 and 5 in a four colour mode were “0,1” and physical colour “0,1,1,1” was to be written to this location, then “0,1,1,1” must be written to all logical colour locations obtained with the four combinations of bits 6 and 4 while 7 and 5 are held as “0,1”.

The Cathode Ray Tube Controller

The Cathode Ray Tube Controller (CRTC) is the heart of the microcomputer’s video display circuitry. Its primary function is to display all video data in the memory on a raster scan display device i.e. a television or a monitor.

The CRTC chip used in the Master Series of microcomputers has sixteen registers, which can all be accessed by the command VDU 23,0. The manufacturer’s data sheet gives the exact effect of the registers, and only the default values for each screen mode and the two control bits HS0 and HS1 in the slow bus control latch are listed here. The bits HS0 and HS1 affect the scrolling function by extending the maximum address in the display memory map, as seen by the CRTC. Note all the numbers are in Hexadecimal.

CRTC chip registers

6845 Registers	Screen MODEs 0(128)	1(129)	2(130)	3(131)	4(132)	5(133)	6(134)	7(135)	Note
R0	7F	7F	7F	7F	3F	3F	3F	3F	
R1	50	50	50	50	28	28	28	28	
R2	62	62	62	62	31	31	31	33	
R3(H sync)	08	08	08	08	04	04	04	04	
R3(V sync)	02	02	02	02	02	02	02	02	
R4	26	26	26	1E	26	26	1E	1E	
R5	00	00	00	02	00	00	02	02	**(1)**
R6	20	20	20	19	20	20	19	19	
R7	22	22	22	1B	22	22	1B	1B	
R8(Int'lace)	01	01	01	01	01	01	01	03	**(2)**
R8(Disp del)	00	00	00	00	00	00	00	01	
R8(Curs del)	00	00	00	00	00	00	00	02	
R9	07	07	07	09	07	07	09	12	
R10	67	67	67	67	67	67	67	72	
R11	08	08	08	09	08	08	09	13	
R12,13	3000	3000	3000	4000	5800	5800	6000	7C00	**(3)**
R14,15	xx	xx	xx	xx	xx	xx	xx	xx	**(4)**
R16,R17	----------Light Pen Position registers----------								
HS0,HS1	1,1	1,1	1,1	0,0	0,1	0,1	1,0	N/A	
Screen End Address	7FFF	7FFF	7FFF	7E7F	7FFF	7FFF	7F3F	7FE7	
Light pen offset	0606	0606	0606	0806	0B04	0B04	0C04	2808	**(5)**
Light pen cell mod.	1	2	4	1	1	2	1	1	**(5)**
Bytes per ** text line	280	280	280	280	140	140	140	28	**(6)**
Text Lines per screen	20	20	20	19	20	20	19	19	
VIDPROC ctrl register	9C	D8	F4	9C	88	C4	88	4B	**(7)**

Notes

1) These only apply if the screen position has not been modified by *CONFIGURE, or a subsequent *TV command.

2) These only apply if the interlace has been turned on by *CONFIGURE, or a subsequent *TV command.

3) These values are only valid before hardware scrolling has been used.

4) On reset, these registers are set to the screen start address, but the actual position will depend on how much screen output has been generated by languages, filing systems etc.

5) Light pens can be connected either to the Analogue Port at the rear of the machine, or to either of the Cartridge Sockets just behind the keyboard. A “low” pulse on any of these connections to the light pen strobe will cause the current scan position to be latched in the light pen position registers, R16 and R17. The accuracy of the measurement will depend on the sensitivity of the light pen. The figures given should be subtracted from the R16,R17 contents to yield the actual screen position, assuming ideal optical conditions. The adjustment arises out of the different screen start addresses. The final X,Y co-ordinates are:

X = ((R16,17 – Offset) DIV (characters per line))/Light Pen Cell Modifier
Y = (R16,17 – Offset) MOD (characters per line)

These offsets are only valid before hardware scrolling has been used. For this reason it is often advisable to restrict light pen use to text or graphics using graphics mode. The Light Pen Cell Modifiers are necessary as the 6845 is clocked at different clock speeds in different modes, so in a given time, the 6845 “sees” a different number of character cells from the one the viewer sees. The modifiers allow this to be taken into account.

6) Each character cell is eight bytes deep as the 6845 imposes this format on the memory map; so each entry in this line of the table is the number of character positions multiplied by eight. This figure can be used to establish the start and end address of any scan row, given the screen’s start address.

7) The VIDeo PROCessor (VIDPROC) control register’s least significant bit is changed in all modes except Mode 7 to cause the colours to flash.

CRTC Multiplexer

The CRTC Multiplexer converts the CRTC’s eighteen-bit address into two eight-bit addresses for the row and column parts of the DRAM’s video cycle. It also provides the hardware scroll logic to keep the addressed memory within the screen’s 20Kbyte boundaries.

Internal Timing

The device uses a slightly delayed version of the DRAMs' nRAS strobe to select between the row and column parts of the address.

Hardware Scroll

The hardware scroll address modification as described in the section on 6845 register values (MOS chapter) is performed by logic within this device. Some of the CRTC address lines are used in a non-standard way. The MA13 line is used as a "Bit-Mapped or Teletext" mode indicator and is used to modify the address scan accordingly.

Refresh Control

In the bit-mapped modes, the memory is scanned often enough to render explicit refresh unnecessary. In the Teletext modes, the addresses of non-displayed locations (as accessed in the 24ms per line when the display is inactive) are modified to produce sequential scanning and hence maintain the refresh.

Multiplexing

The address is output, one half at a time for each of the Row and Column addresses. One of four eight bit fields may be selected:

1) Bit mapped display - low order address
2) Bit mapped display - high order address
3) Teletext display - low order address
4) Teletext display - high order address

The VDU driver

The VDU Driver is extensively covered in Part 1 of the Reference Manual. However, by programming in machine code, the hardware may be accessed directly to give additional display modes, such as a 640×512 MODE. This is a two-colour mode which uses both the main and shadow screen memories to store alternate half-frames of an interlaced synchronisation and video picture. The method used is as follows:

1. Select MODE 0
2. Program the CRTC for interlaced sync. and video.
3. Set the EVNTV vector to point to your code.
4. Enable the vertical synchronisation event.
5. Use OSBYTE &70 (X=1) (*FX 112,1) to select the half-frame to be drawn.
6. Draw the half-frame.
7. Use OSBYTE &70 (X=2) (*FX 112,2) to select the second half-frame.
8. Draw the second half-frame.
9. Use OSBYTE &71 (X=1,X=2) (*FX 113,1 and *FX 113,2) to select alternate screens on alternate vertical synchronisation events.

The program will alternate the half-frames correctly but should provide the facility to reverse the display sequence as the hardware may present the two half-frames in the incorrect phase.

The display may be distorted if any software disables the vertical synchronisation event.

OSBYTE &75 (117) is used to read the VDU status byte, and puts its current value into the X register. The bits in the result have the following meanings.

VDU status -	bit 0	printer output enabled
	bit 1	scrolling disabled
	bit 2	paged software scrolling enabled
	bit 3	text window is currently defined this is set up by VDU 28 and cleared by VDU 26
	bit 4	shadow screen selected
	bit 5	printing at graphics cursor enabled
	bit 6	cursor editing mode enabled
	bit 7	VDU is disabled via VDU 21

7 THE USER PORT

The User Port provides the following facilities:

Eight-bit bi-directional data port with optional handshaking
Programmable pulse generator
Programmable frequency generator
Pulse counter
Synchronous/asynchronous SIPO/PISO shift register

It appears as a set of memory-mapped locations and is accessed using OSBYTEs &96,&97 (150,151). As the parallel printer port is controlled by the same 6522 versatile interface adapter (VIA) chip, care should be taken to avoid conflicts between the two applications. The 6522 registers that control the User Port are described here, bit-by-bit. D0 is the least significant bit, D7 is the most significant bit. The User 6522 VIA has a base address of &FE60

Timers

Two sixteen-bit counter/timers are provided. They are designated T1 and T2. Each consists of a sixteen-bit decrementing counter, one or two eight-bit latches and some control logic. The latches are used to store the values that will be loaded into their respective counters when a particular event occurs. The modes of operation are determined by the Auxiliary Control Register.

User VIA Address Mapping

Offset	Function
0	User Port Data Register
2	User Port Data Direction Register
4	T1 - Low Order Counter/Latch (R/W)
5	T1 - High Order Counter (R/W)
6	T1 - Low Order Latch (R/W)
7	T1 - High Order Latch (R/W)
8	T2 - Low Order Counter/Latch (R/W)
9	T2 - High Order Counter (R/W)
10	Shift Register
12	Peripheral Control Register
13	Interrupt Flag Register
14	Interrupt Enable Register

User Port Data Register

User Port access. Bit PB0 on the User Port corresponds to the data bit D0 whilst PB7 corresponds to D7. Control lines CB1 and CB2 can be programmed to behave as handshake lines. CB1 acts as Data Acknowledge. CB2 acts as Data Ready. For example, if the following connections are made between two Master Series computers (A and B)

Computer A		**Computer B**
PB[0:7]	to	PB[0:7]
CB1	to	CB2
CB2	to	CB1
Ground	to	Ground

when the interrupts are enabled, writing a byte to the User Port in A will cause an interrupt to be generated in B. When B reads the data from its User Port, A will be interrupted to indicate that the data has been taken. The data traffic will also work in the other direction.

The manufacturer's data sheet should be consulted for detailed timing information.

User Port Data Direction Register

Each bit in this register acts as a flag for the corresponding User Port bit. If set it will be an output, if clear an input.

Timer 1 Low Order Counter/Latch (R/W)

Read - the T1 low order counter is read and the T1 interrupt flag (in the Interrupt Flag Register) is cleared.
Write - the data written into this latch is transferred to the T1 low order counter after either the T1 high order counter is written to, or the T2 counter underflows through zero in the free-run mode.

Timer 1 High Order Counter (R/W)

Read - the T1 high order counter is read, but the T1 interrupt status is not affected.
Write - the data written into the latch is stored and transferred into the T2 High Order counter at the next system 1MHz high transition. T1 low order latch is transferred to T1 low order counter at the same time. This action effectively starts the counter and the T1 interrupt flag is cleared accordingly.

Timer 1 Low Order Latch (R/W)

Read - the value in the T1 low order latch is read. T1 interrupt status is not affected.
Write - equivalent to writing to Offset 4.

Timer 1 High Order Latch (R/W)

Read - the last value written is read back.
Write - the value written is stored, but is only transferred to the T1 high order counter when T1 underflows in free-run mode.

Timer 2 Low Order Counter/Latch (R/W)

Read - T2 low order counter is read and the T2 interrupt is cleared.
Write - the data written is stored in the T2 low order latch.

Timer 2 High Order Counter (R/W)

Read - T2 high order counter is read.
Write - the data is written directly into the T2 high order counter. This causes the value in the T2 low order latch to be transferred into the T2 low order counter and the T2 interrupt is cleared.

Shift Register

A multi-function register controlled by the Auxiliary Control Register at Offset 11. It is a left-shift, circulating register, i.e. data is shifted in from bit 0 towards bit 7 and when shifting out, has bit 7 connected to the input of bit 0. It has eight modes of operation which are in no way related to the screen modes.

Mode 0 - Static Shift Register

Read - the value shifted into the shift register is read.
Write - the shift register will contain the value written.
Shift - the data on CB2 will be shifted in on CB1 positive transitions.
Interrupts - the shift register interrupt is disabled.

Mode 1 - Data Shifted in by T2

Read - the value shifted into the shift register is read. Shifting will start.
Write - the shift register will contain the value written. Shifting will start.

Shift - data is shifted in on CB2

a) after a read/write operation with the SR interrupt clear.
b) after T2 times out following a read/write with SR interrupt SET. Shifting will occur for eight T2 time-outs.

Interrupts - the SR interrupt will occur after eight T2 time-outs.

Note: In this mode CB1 is clocked with the T2 time-out. This is to provide a clock for the external device providing the data. Data is shifted in on the CB1 negative edge, but is sampled (latched) on the CB1 positive edge. For this reason, the external device should be clocked on the CB1 negative edge. Shifting stops after the eighth shift.

Mode 2 - Data Shifted in by the system 1 MHz clock

This is similar to Mode 1 except that CB1 clock is the system 1MHz clock, divided by two.

Mode 3 - Data Shifted in by externally provided CB1 clock

This mode is used when data is provided by an asynchronous source from which a clock is derived.
Read - the value shifted into the shift register is read.
Write - the shift register will contain the value written.
Shift - data is shifted in on CB2 at the system 1MHz pulse after the CB1 positive transition.
Interrupts - the shift register interrupt is set after 8 data bits have been shifted in. It is reset at the next read/write of the shift register.

Note. Due to the shift-in timing, it is recommended that the incoming data rate should not exceed 250kHz, thereby allowing for the asynchronism between the transmitting and receiving units. The actual data rate is more likely to be limited by the speed with which the “register full” interrupt is serviced; the shift register keeps shifting whether or not it is serviced, so data may be lost if the user’s program does not respond in time.

Modes 4 and 5 - Data Shifted out by T2

Read - the current shift register value is read. Shifting will start.
Write - the shift register will contain the value written. Shifting will start.
Shift - data is shifted out on CB2

a) after a read/write operation with the SR interrupt clear.
b) after T2 times-out following a read/write with SR interrupt set. In Mode 4, shifting occurs at every T2 time-out. In Mode 5, shifting will occur for eight T2 time-outs and then stop until the interrupt is serviced and new data is loaded.

Interrupts - the SR interrupt will occur after eight T2 time-outs.

Note: In this mode CB1 is clocked with the T2 time-out. This is to provide a clock for the external device sampling the data. Data is shifted on the CB1 positive edge, but should be sampled by the external device on the CB1 negative edge. For this reason, the external device should be clocked on the CB1 negative edge. Shifting stops after the eighth shift in Mode 5 but is continuous in Mode 4.

Mode 6 - Data Shifted out by the system 1MHz clock

This is the shift out equivalent of Mode 2.

Mode 7 - Data Shifted out by externally provided CB1 clock

This is the shift out equivalent of Mode 3. The same restrictions to data rate apply.

Auxiliary Control Register (R/W)

Controls the shift register mode, Timer 1, Timer 2 and the Port A B latching. It is divided into three fields

(1) Port Latching

Bit 0 enables/disables latching of the Printer port. This bit must be maintained at all times.

Bit 1 enables/disables latching of the User Port. A logic 1 will enable latching. CB1 acts as a strobe to latch the data.

(2) Shift Register Control

Bits 4,3,2	Function
0 0 0	Mode 0
0 0 1	Mode 1
0 1 0	Mode 2
0 1 1	Mode 3
1 0 0	Mode 4
1 0 1	Mode 5
1 1 0	Mode 6
1 1 1	Mode 7

(3) Timer 2 Control

Bit 5 0 - interrupt when T2 decremented to zero
1 - decrement T2 with each pulse input to PB6. Interrupt when T2=0, then re-load and continue counting, so generating an interrupt stream. T2 high order counter must be written after every T2 interrupt to enable the next interrupt

(3) Timer 1 Control

Bits 6,7	Operation
0 0	After loading T1, it will generate a single interrupt after decrementing to zero.
0 1	After loading T1, it will generate a stream of interrupts; one whenever it counts down to zero.
1 0	As 00 but output a single pulse on PB7 as well as the interrupt.
1 1	As 01 but generate a stream of output pulses as well as the interrupts.

Note: When Timer 1 mode 11 is selected, PB7 will change polarity every time T1 counts down to zero. This means that it will output a waveform of frequency: PB7frequency=1/(<T1latches>*2)

Peripheral Control Register

The most significant nibble dictates the function of the CB1, CB2 control lines, whilst the least significant nibble controls CA1, CA2. The latter should not be touched as it may interfere with correct parallel printer operation. Whenever writing to this register, ensure that the least significant nibble is preserved.

CB1 Interrupt Control

Bit 4 0 - generate an interrupt on a CB1 negative edge.
1 - generate an interrupt on a CB1 positive edge.

CB2 Control

Bits 5,6,7	Operation
0 0 0	CB2 will generate an interrupt on its negative edge
0 0 1	CB2 as above, independent mode
0 1 0	CB2 will generate an interrupt on its positive edge
0 1 1	CB2 as above, independent mode
1 0 0	CB2 provides the “Data Ready” handshake output.
1 0 1	CB2 provides a single high-going pulse.
1 1 0	CB2 goes to a 0
1 1 1	CB2 goes to a 1

Independent Mode

Whilst reading the User Port Data Register would normally clear the interrupt request that transitions on CB2 have created, in the “independent modes” these interrupts have to be cleared by directly clearing the appropriate bits in the Interrupt Flag Register.
Note that the bits 0,1,2,3 perform a similar function for CA1 and CA2.

Interrupt Flag Register

The CPU has to be able to determine which function of the User Port is generating an interrupt. This register has a bit representing each of the functions that can do this. Even if an interrupt source has been disabled using the Interrupt Enable Register, it can still set its appropriate flag in this register. A set bit indicates that the function is trying to generate an interrupt.

Register bit	set when...	cleared when...
0	CA2 active edge occurs	Printer port is accessed
1	CA1 active edge occurs	Printer port is accessed
2	Shift Register completes 8 shifts	Shift Register is accessed
3	CB2 active edge occurs	User Port Data is accessed
4	CB1 active edge occurs	User Port Data is accessed
5	T2 times-out	Read T2 low order OR Write T2 high order
6	T1 times-out	Read T1 low order OR Write T1 high order
7	Any interrupt is set	All interrupts are clear

Note that bit 7 is designed to enable fast interrupt control. It is only necessary to test bit 7 to find out if any of the functions are generating an interrupt request. The CPU's BIT operation will cause its negative status bit to be set if bit 7 is set in this register.

Interrupt Enable Register

For each bit in the Interrupt Flag Register to cause an interrupt, the corresponding bit in the this register must be set.

Register bit	Enables the interrupt from
0	CA2
1	CA1
2	Shift Register
3	CB2
4	CB1
5	Timer 2
6	Timer 1
7	Global

If the Global bit is clear, then every set bit in the register disables the corresponding interrupt request. If it is set then every set bit in the register enables the corresponding interrupt request.
When this register is read, Bit 7 will be set and other bits will be as written.

Example of motor control

For example, to control a three axis machine which uses stepper motors, Timer 1 frequency generator output may be used to provide stepping pulses to motor phase sequence generators. Other PB lines can provide forward/backward control and move/hold controls. This means that all three motors can be rotating at once. The Timer 2 pulse counter can be used to count the number of pulses that have been applied to the motors. Every time a T2 interrupt is generated, those motors which are enabled will have their positions (as stored in memory) updated by the CPU. Limit switches on each axis can be connected to over-ride the 6522 outputs and logically ORed to generate an interrupt so that if any motor tries to go “off the end” the CPU will detect this and so prevent the occurrence of any damage. The PB lines can then be used as inputs to determine which motor has gone to its end stop.

Method

Assign the User Port pins :

a) CB1 will be the global alarm (overrun) input.
b) PB7 is the frequency generator output.
c) PB6 is the pulse counter.
d) PB5 is the Z axis enable/fault indicator.
e) PB4 is the Z axis direction control/fault indicator.
f) PB3 is the Y axis enable/fault indicator.
g) PB2 is the Y axis direction control/fault indicator.
h) PB1 is the X axis enable/fault indicator.
i) PB0 is the X axis direction control/fault indicator.

To run the motors:

PB7 must be a frequency output
PB6 must be a counter input
PB[0:5] must be outputs

Thus:

Location	Contents 7 0	Comments
&FE6C	0000oooo	CB1 negative interrupt
&FE6B	111ooooo	Set up the timer controls
&FE6E	1o1ooooo	Enable the T2 interrupt
&FE62	10111111	Enable the outputs
&FE60	XXDDDDDD	Operate the motors

o is the old contents D is the desired action

Timer 1 should be programmed with the value for the required operating frequency.

To find out which motor has overrun:

PB[0:5] should be inputs
PB7 should be switched off whilst the overrun is checked.

Thus:

Location	Contents 7 0	Comments
&FE6B	001ooooo	Switch off Timer 1
&FE62	10000000	Inputs to read the switches
&FE60	XXDDDDDD	Read the switches

o is the old contents D is the desired action

Operation can now be returned to “Running Mode”.

8 THE SERIAL PROCESSOR

The serial processor (SERPROC) is used in conjunction with the 6850 UART to provide the RS423 and cassette tape interfaces. It contains a baud rate generator, channel multiplexer and tone generator.

UART

The device responsible for providing most of the serial port functions is a 6850 UART. This has all the receive/transmit and data formatting/error checking that is necessary for both systems. It is fully described in the March 1983 edition of the Hitachi Microcomputer Databook.

SERPROC

The ACORN proprietary part, SERPROC is effectively a multiplexer and baud rate generator for the 6850. It also generates the phase-continuous transmission circuitry for use with the cassette interface.

Buffer Components

The RS423 transmit data and CTS lines are buffered by an AM26LS30 or equivalent. This provides a single ended transmission with slew rate limited output. RS423 receive data and RTS is buffered by a mA9637AC or equivalent. Both buffers are connected with single-ended input configurations.

Cassette data output from the SERPROC is buffered by a single, non-inverting operational amplifier with a simple single pole filter, a.c. coupling capacitor and current limiting output resistor.

Control Register Settings

Bit #	Function	Parameters msb
0-2	Transmit Baud Rate	000 : 19200 100 : 9600 010 : 4800 110 : 2400 001 : 1200 101 : 300 011 : 150 111 : 75
3-5	Receive Baud Rate	000 : 19200 100 : 9600 010 : 4800 110 : 2400 001 : 1200 101 : 300 011 : 150 111 : 75
6	Channel Select	0 : Select Tape 1 : Select RS423
7	Cassette Motor Relay	0 : Open 1 : Closed

Note. The Transmit and Receive baud rates both assume that the 6850 has its clock divider set to divide by 64.

Receive baud rate not used in cassette mode, but Bit 3 may control inversion of the Transmit data (VTI version of SERPROC)

9 THE PERIPHERAL BUS CONTROLLER

The peripheral bus controller buffers data between the 65C12 CPU (on the “CD” bus) and the internal peripherals on the “BD” bus, the external “1MHz Bus” and the external “Tube” interfaces (both on the “ED” bus). It also contains a timer to generate a long delay after power-up.

Internal Timing

All the necessary timing is synthesised from the system 8MHz and 1MHz signals.

Buffer Control

The selected buffer path is determined by the RDY and FIT signals, as described for the I/O Controller, together with the system R/W signal.

Timer

The timer is an eight-bit counter with an external oscillator, which is also used as the timer’s output. The oscillator output is used to charge/discharge a timing capacitor. The use of a charge time constant which is 1% of the discharge time constant causes the output (CHRG) to be low most of the time. When the input (TICK) crosses the threshold during an oscillation, the counter is incremented. When the terminal count is reached, the output is fixed high. The counter can only be reset by switching the power off. This timer was originally designed to support the boost charge of nickel-cadmium batteries for the Real Time Clock.

I/O Definition

Pin Name	No	I/O	Input Buffer Type	Output Buffer Type
TICK	4	I	CMOS SCHMITT	
NFIT	5	I	CMOS	
R/W	6	I	CMOS	
RDY	11	I	CMOS	
NPRST	1	I	TTL	-
DEN	2	I	TTL	-
M1	29	I	TTL	-
M8	31	I	TTL	-
CHRG	3	O	-	standard
BRNW	7	O	-	standard
EM1E	8	O	-	standard
ER/W	9	O	-	standard
ED7	12	I/O	TTL	standard + tristate
ED6	13	I/O	TTL	standard + tristate
ED5	14	I/O	TTL	standard + tristate
ED4	15	I/O	TTL	standard + tristate
ED3	16	I/O	TTL	standard + tristate
ED2	17	I/O	TTL	standard + tristate
ED1	18	I/O	TTL	standard + tristate
ED0	19	I/O	TTL	standard + tristate
CD7	28	I/O	TTL	standard + tristate
CD6	27	I/O	TTL	standard + tristate
CD5	26	I/O	TTL	standard + tristate
CD4	25	I/O	TTL	standard + tristate
CD3	24	I/O	TTL	standard + tristate
CD2	23	I/O	TTL	standard + tristate
CD1	22	I/O	TTL	standard + tristate
CD0	21	I/O	TTL	standard + tristate
BD7	40	I/O	TTL	standard + tristate
BD6	39	I/O	TTL	standard + tristate
BD5	38	I/O	TTL	standard + tristate
BD4	37	I/O	TTL	standard + tristate
BD3	36	I/O	TTL	standard + tristate
BD2	35	I/O	TTL	standard + tristate
BD1	34	I/O	TTL	standard + tristate
BD0	33	I/O	TTL	standard + tristate
VCC	30		Vcc connection	(low inductance)
GND1	10	Primary	GND connection	(low inductance)
GND2	32	Secondary	GND connection	(low inductance)
GND3	20	Secondary	GND connection	

AC Parametric Test Information - Timing Specifications

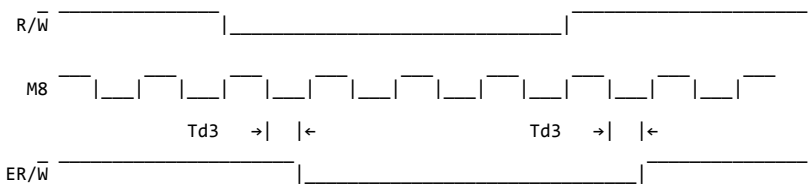
Timing Symbol	Point-to-Point Parametric-Specification measured at Vcc= Min Tamb= Max	Time (ns) Min	Max	Output I/face	Load Value
Tj1	M1 (LH/HL) jitter wrt M8 (HL)	-30	+4		
Td1	EM1E (LH/HL) from M8 (HL)	0	60	TTL	A
Td2	ER/W (LH/HL) from RNW (LH/HL)	0	80	TTL	A
Td3	ER/W (LH/HL) from M8 (HL)	0	70	TTL	A
Td4	BR/W (LH/HL) from R/W (LH/HL)	0	50	TTL	B
Td5	CD7..0 stable data from NFIT (HL)	0	85	TTL	C
Te2	BD7..0 (ZH/ZL) from M8 (LH)	0	90	TTL	B
Tz2	BD7..0 (HZ/LZ) from M8 (HL)	0	72	Z	B
Td6	B Bus , SA to SL data, from M8 (HL)	0	75	TTL	B
Td7	B Bus , SL to SA data, from M8 (LH)	0	90	TTL	B
Te3	ED7..0 (ZH/ZL) from NFIT (HL)	0	90	TTL	A
Tz3	ED7..0 (HZ/LZ) from M8 (HL)	0	105	Z	A
Tz4	ED7..0 (HZ/LZ) from NFIT (LH)	0	105	Z	A
Td8	CD7..0 (LH/HL) from BD7..0 (LH/HL)	0	70	TTL	C
Td9	CD7..0 (LH/HL) from ED7..0 (LH/HL)	0	70	TTL	C
Td10	ED7..0 (LH/HL) from CD7..0 (LH/HL)	0	60	TTL	A
Td11	BD7..0 (LH/HL) from CD7..0 (LH/HL)	0	70	TTL	B

Load circuit component values

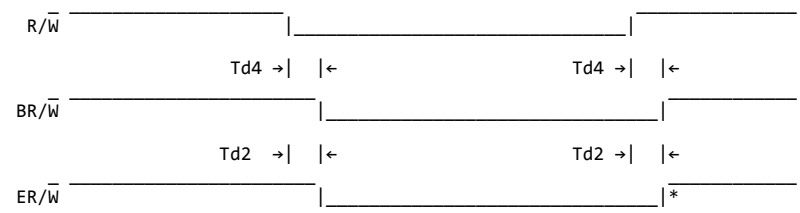
For details of load circuit see AC measurement definition

Load Value	C (pF)	R (ohms)
A	150	1000
B	100	1000
C	170	1000

R/W (latched)

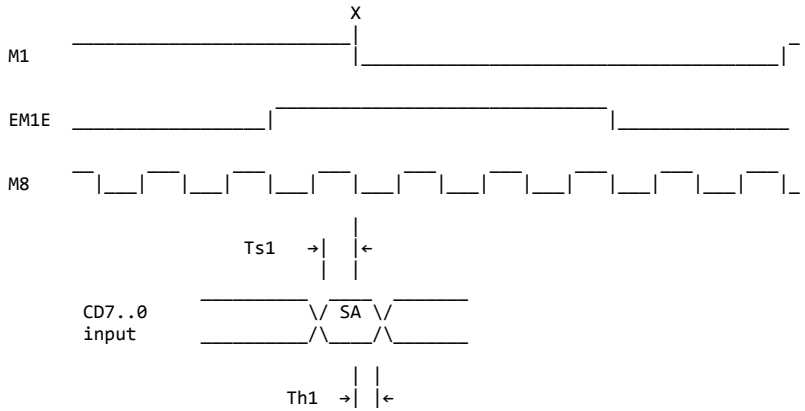


R/W (transparent)



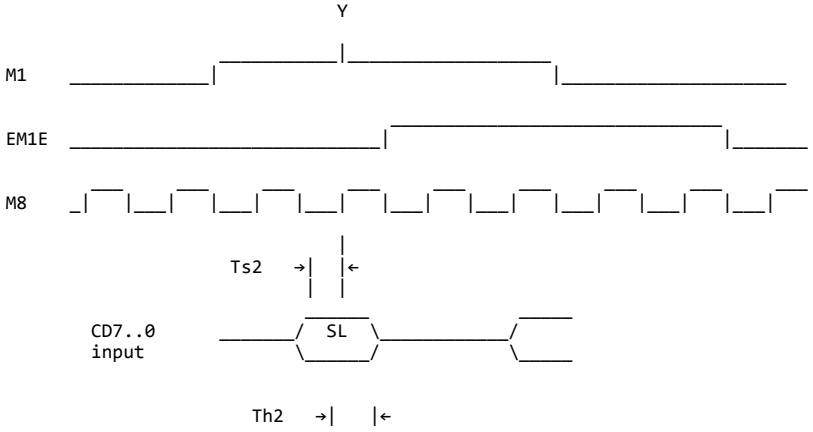
SA data latching point

The video data for the SA5050 Teletext Display device is time division multiplexed with the internal 1MHz peripheral data (as distinct from the external 1MHz Bus). This data is latched at the point X in the timing illustrated below.



SL data latching point

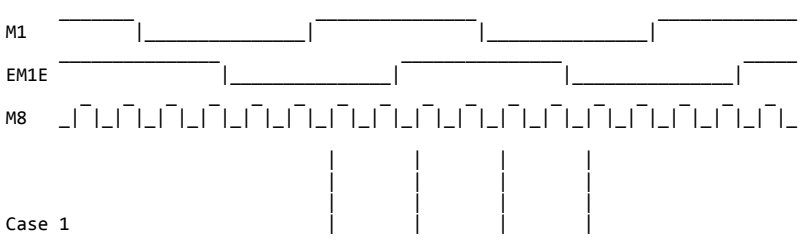
Data for 1MHz internal peripherals is latched at the point Y on the timing diagram below.



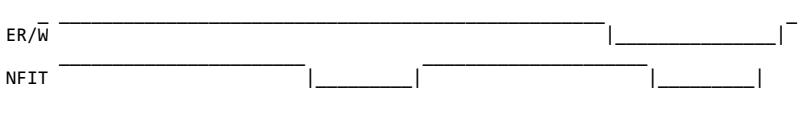
C Bus Drive Waveforms

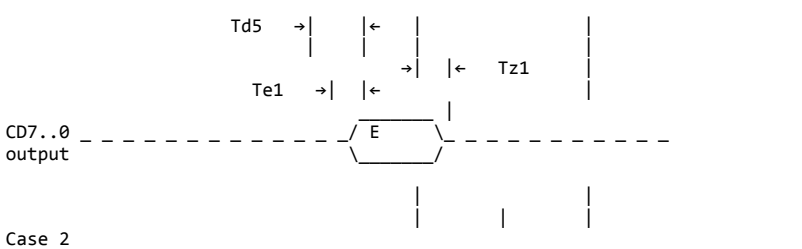
The peripheral bus controller drives the CPU data bus (the C Bus) on the following occasions:
a) Reading from internal peripherals
b) Reading from the external 1MHz Bus
c) Reading from the external Tube

Because these events may or may not be in phase with the CPU cycle, the PBC withholds the data until the correct time.

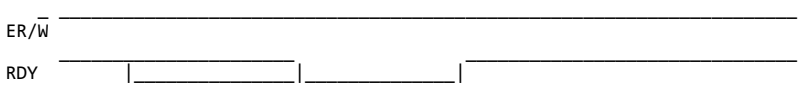


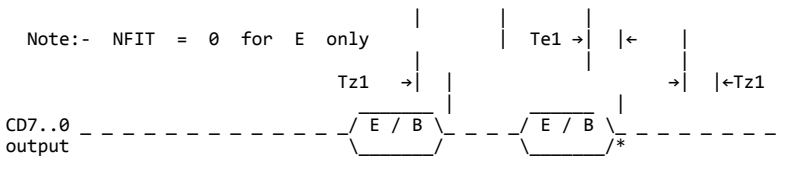
Reading from the 1MHz Bus or an internal 1MHz peripheral. EM1E is in phase.





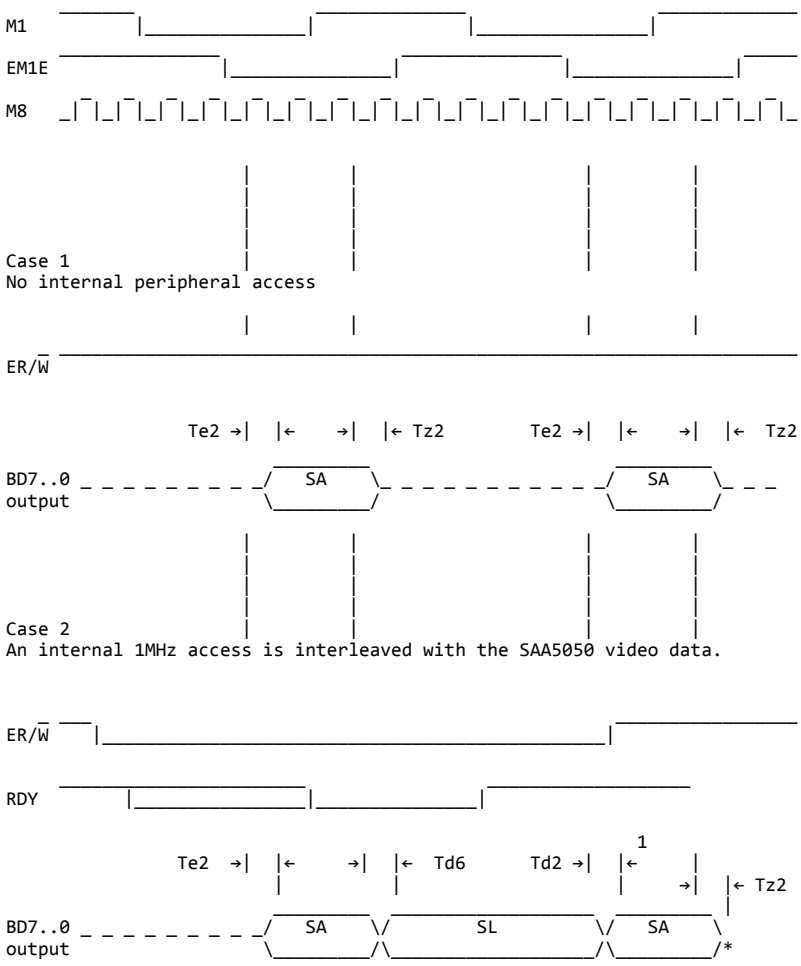
Reading from the 1MHz Bus or an internal 1MHz peripheral. EM1E is early.





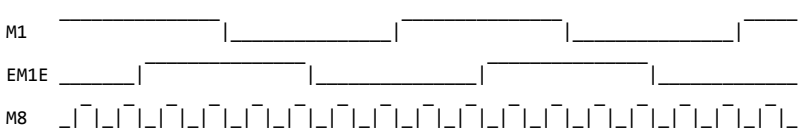
B Bus Drive Waveforms

The B Bus contains both the internal 1MHz peripheral data and the SAA5050 video data. This bus is used by the Modem connector, so it is important to observe the timing constraints.

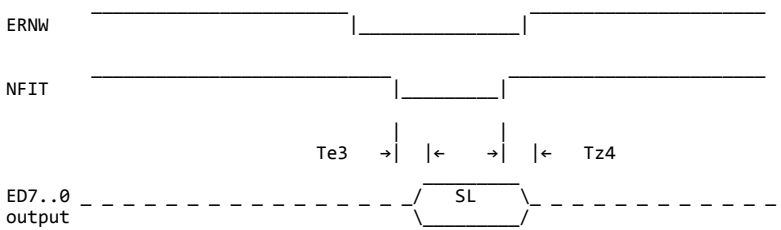


E bus drive waveforms

The E Bus operates at either 1MHz or 2MHz under the control of the CPU READY line, which it samples. This signal is driven by the I/O controller with a logic low to slow the CPU down to 1MHz when a slow access is made. The PBC extends its bus cycle time in much the same way as the CPU. In this way the 1MHz Bus and Tube connectors can be driven by the same buffer. It is important that 1MHz Bus peripherals using any significant length of ribbon cable (greater than 30cm) use 2k2 pull up/down resistors to minimise line reflections to the Tube.

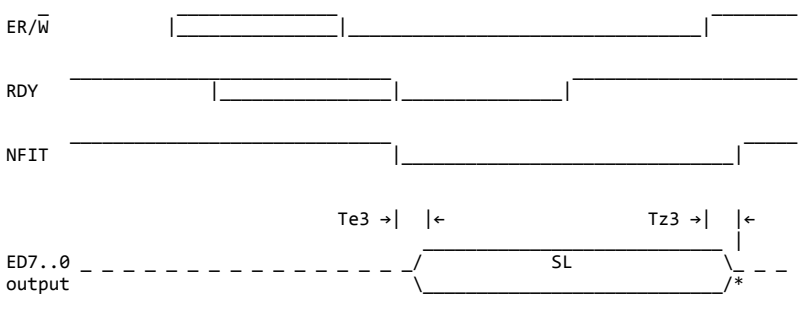


Case 1 - Writing to the Tube



Case 2 - Writing to the 1MHz Bus

Both of the two possible timing relationships are shown. The data has a nominal 250ns data setup time before the rising edge and a minimum hold time of 125ns after the falling edge of EM1E (measured at the PBC). The address set up is also shown. This is generated by a latch clocked at 4MHz and so presents a minimum address set up time of 250ns and a minimum address hold time of 250ns.



10 THE 1MHz BUS

This chapter describes the signals available on the 1MHz Bus, the circuitry required to utilise them, and the way in which they are connected to the Acorn Expansion Box. The expansion memory map is also defined. When interfacing designs to the 1MHz Bus, it is vital to ensure compatibility with Acorn standards, to prevent problems when using several pieces of equipment on the bus simultaneously.

The standards cover both hardware and software protocols. It is as important for the software to follow these guidelines as it is for the hardware, otherwise simultaneous operations of several peripherals may not be possible. The standards described allow up to 64K of paged address space to be accessed as well as 255 bytes of direct access ports.

Signal definitions

The following lines are available on the 1MHz Expansion Connector:

A0 - A7 The low eight address lines from the 6502, buffered by a 74LS373 (IC 7) permanently enabled.

D0 - D7 A bi-directional data bus connected to the CPU through IC 21, Peripheral Bus Controller. The direction of data is determined by the system Read-not-Write (R/W) line. The buffer is only enabled if nPGFC or nPGFD is low (see below).

Analogue in An input to the BBC Microcomputer audio circuitry. Input impedance is 9K. A signal of 3volts RMS will produce a saturated signal at the loudspeaker (full volume), though signals this large will cause distortion if the on-board sound or speech is used at the same time.

nRST Not Reset. This is an OUTPUT ONLY for the system reset line (active low). It may be used to initialise peripherals on power-up and when the “BREAK” key is pressed.

nPGFC & nPGFD “Not page FC” and “Not page FD”. Page select signals decoded from the top eight address bits of the system data bus. These signals are active low. Pages FC and FD (i.e. &FC00 to &FCFF and &FD00 to &FDFF) are the only pages available for general expansion. However, the

paging register described in Section 5 allows a much larger address space to be accessed.

nIRQ Not Interrupt Request (active low). The system IRQ line which is open collector (i.e. “wired-or”) and may be asserted by devices attached to the extension bus. The pull-up resistor on this line is 3K3. IRQ is level triggered and it is absolutely essential for correct operation of the machine that interrupts do not occur until the software is capable of dealing with them. Interrupts on the 1MHz bus should therefore be disabled on power-up and reset conditions. Significant use of interrupt service time may affect other machine functions. In particular, masking interrupts for more than 10mS will affect the real time clock.

nNMI Not Non-Maskable Interrupt (active low). The system NMI line which is open collector (i.e.“wired-or”) and may be asserted by devices attached to the extension bus. The pull-up resistor on this line is also 3K3. It should be remembered that NMI is negative-edge triggered and that both the disc and net chips on the main board use this line. Caution must be exercised to avoid masking other interrupts by holding the line low. Use of NMI facilities on the BBC machine requires an advanced knowledge of 6502 programming techniques and the Operating System Protocols.

1MHzE A system clock timing signal which is a 1MHz 50% duty-cycle square wave. During access to 1MHz peripherals and to the extension bus the processor clock (normally 2MHz) is stretched so that the trailing edges of 1MHzE and processor clock are coincident.

R/W The system Read-Not-Write signal which is derived from the CPU R/W signal through two 74LS04 inverters.

0V System 0V, i.e. GND wires, dispersed so as to interleave with asynchronous groups of signals in a flat ribbon cable.

Hardware requirements for 1MHz expansion bus peripherals

No power may be drawn from the BBC Microcomputer. Each peripheral should have its own integral power supply, although a separate power unit may be used.

Not more than one low-power Schottky TTL load may be presented to any bus line by each peripheral.

A 1MHz Bus feed-through connector should be provided. Connection to the BBC Microcomputer should be via 600mm of 34-way ribbon cable terminated with a 34-way IDC socket, and fitted with strain relief. Please note that copying the Teletext Adapter's layout is not possible, because this has been given the special status of the last box in the chain.

Optional bus termination should be provided on all bus lines except NRST, NNMI and NIRQ. The recommended termination is a 2K2 resistor to +5V and a 2K2 resistor to ground for each line.

Further requirements for equipment to be approved by Acorn Computers

Address space within page &FC must be allocated by the Research and Development Department of Acorn Computers Ltd.

The dimensions of any peripheral and its associated integral power supplies should allow it to be fitted into the BBC Microcomputer Expansion Box.

When housed in the Expansion Box, the equipment should meet BS415 Class 1 specifications for electrical safety.

Further details of the requirements and procedures for gaining approval should be obtained from Acorn. The information included here is for guidance only and is not intended to be a full specification for approval.

Derivation of valid Page signals

1MHz peripherals are clocked by a 1MHz 50% duty cycle square wave (chosen to allow chips such as the 6522 to use their timing elements reliably). The Master Series 65C12 normally operates with a 2MHz clock, but with a slow-down circuit which has the effect of stretching the “clock high” period immediately following the detection of a valid 1MHz peripheral address.

There are two problems as a result of this. First, addresses will change and may momentarily become 1MHz addresses while the 2MHz CPU clock is low, but while the 1MHzE signal is high. This could give rise to a spurious pulse on the chip select. Second, if the CPU deliberately addresses a 1MHz peripheral during the time that 1MHzE is high, the device will be addressed immediately, and then again when 1MHzE is next high: this is because the CPU clock will be held “high” by the stretching circuit until the next coincident falling edge of the 1MHz and 2MHz clocks. This double access is not usually a problem except when reading from or writing to a location twice has some additional effect: an example of this is an interrupt flag which is cleared by reading it.

These effects mean that the 1MHzE Bus cannot be used as a conventional “address valid” signal. However, addresses will always be valid on the rising edge of 1MHzE. If the chip select lines are latched by 1MHzE, the clean signal CNPGFC (or CNPGFD) will be generated.

Address space allocation

Page FC

Page FC is reserved for peripherals with small memory requirements. Only one peripheral will be allocated to each group of addresses. Further allocations must be agreed with the R & D department of Acorn Computers Ltd.

Initial allocations are:

FC00 to FC0F	Test Hardware
FC10 to FC13	Teletext
FC14 to FC1F	Prestel
FC20 to FC27	IEEE 488 Interface
FC28 to FC2F	Acorn Expansion: spare
FC30 to FC3F	Cambridge Ring Interface
FC40 to FC47	Winchester Disc Interface
FC48 to FC7F	Acorn Expansion: spare
FC80 to FC8F	Test Hardware

FC90 to FCBF	Acorn Expansion: spare
FCC0 to FCFE	User Applications
FCFF	Paging Register

Page FD

Page FD is used in conjunction with the paging register to provide a 64K address space, accessed one page at a time. Each BBC Expansion Box will have a paging register on the back plane, thus data will be latched simultaneously on every Expansion Box. Data latched into the paging register will provide the top eight address bits to the Eurocard back plane. These top address bits are referred to as the 'Extended Page Number'. Any peripheral designed to locate in page FD without using an expansion back plane must latch and decode the paging address information.

To make this facility as easy to use as possible, nPGFD (a hazard-free version of the signal available from PL12) will be connected to the back plane pin 24b, 'Not Valid Memory Address', and also OR-ed with the top four extended page address lines as a link selectable option to pin 31a 'BLKO'. (the other option on this pin will be nPGFC).

Extended pages &00 to &7F are reserved for Acorn use, pages &80 to &FF may be freely used by special applications. The paging register will be reset to &00 on power-up and BREAK.

Since the paging register is a write-only latch, location &00EE in the zero page of the BBC machine address map has been allocated as a RAM image of the register. Note that this location will remain in the I/O processor's memory map if a second processor is fitted.

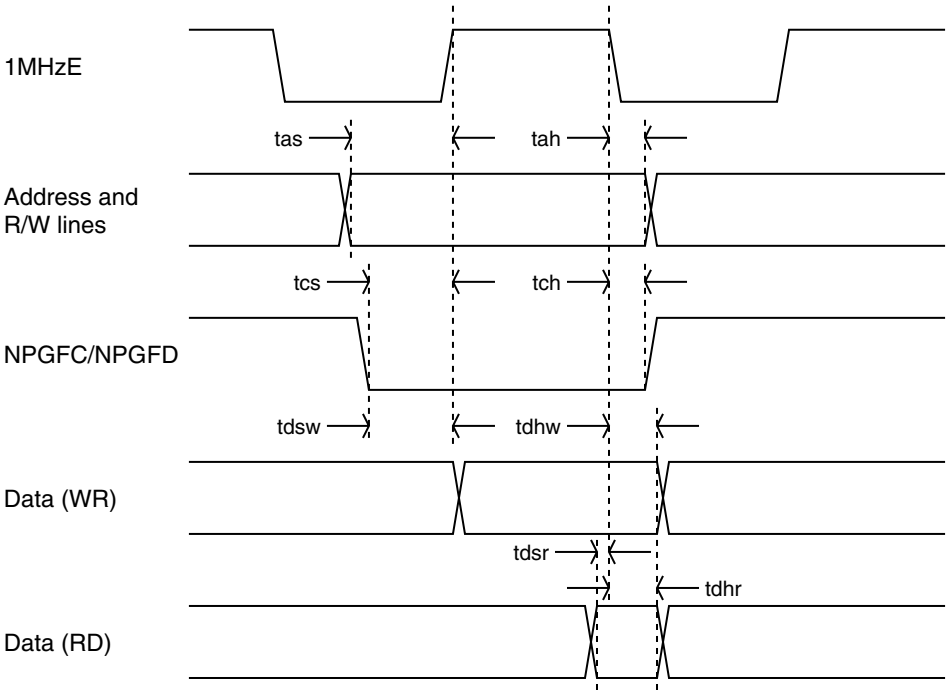
The importance of this image is that it allows interrupt routines to change the paging register and restore it again afterwards.

It is vital to change location &00EE BEFORE changing the paging register itself. If you don't, then an interrupt may occur before you change the RAM image and this will restore the paging register to the old value of &EE.

A suitable sequence is

```
LDA # new value
STA &EE
STA &FCFF
```

User routines should save the contents of &EE before changing the paging register and restore both &EE and &FCFF to this value before returning from the interrupt.

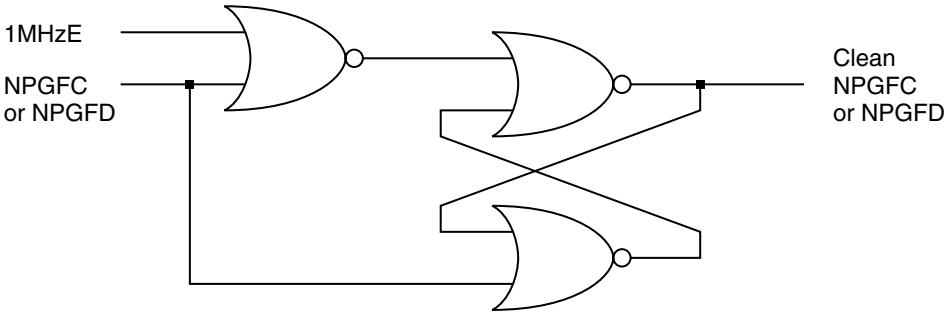


Timing requirements

Parameter	Symbol	Min.	Max.
Address Set-up time (& R/W Set-up time)	t as	300	1000
Address Hold Time (& R/W Hold Time)	t ah	30	
NPGFC & NPGFD Set-up Time	t cs	250	1000
NPGFC & NPGFD Hold Time	t ch	30	
Write Data Set-up Time	t dsw		150
Write Data Hold Time	t dhw	50	
Read Data Set-up Time	t dsr	200	
Read Data Hold Time	t dhr	30	

Note: The above timings are based on only one peripheral attached to the Expansion Bus. Heavy loading may slow the rise and fall times of 1MHzE with possible adverse effects on timings.

R-S flip-flop with gated input which allows ‘clean select’ to be set low only if 1MHzE is low. An alternative circuit using transparent flip-flops is shown on the circuit diagram for the Expansion Box back plane (Drawing 107,000)



11 THE MACHINE OPERATING SYSTEM

This section explains how to extend the MOS facilities of the microcomputer, such as the VDU driver and the TUBE interface. It includes a full address map (which has indicators showing where the MASTER 128 and the MASTER Econet Terminal differ from the earlier BBC machines), the vector allocations (which are given in full) and details on the use of vectors with interrupts and the Tube.

It may be helpful to refer to the chapter on the MOS in Part 1 of the Reference Manual for additional information.

Address space map

The address space map, which shows the address allocations and the areas of memory used by the computer, indicates to a programmer which areas of the memory are available for him to use. However, it does not show individual input/output allocations as they have already been documented in Part 1 of the Reference Manual.

Although this section explains how to use areas of memory which are normally reserved for specific purposes, Acorn does not condone the practice, as it may lead to software incompatibility when used on a machine other than the one on which it was written or if the configuration of the machine is changed.

Page 0

&0000-&008F: current language workspace - some languages e.g. BASIC, allow other programs to use areas of free memory,

&0090-&009F: ECONET private workspace - not available for any other use.

&00A0-&00A7: Non-Maskable Interrupts (NMI) workspace - may be used only after NMI has been claimed. The source of the NMI has a filing system number allocated to it (rather than a ROM number) and it must be able to service the calls &0B and &0C (which indicates that it is either in the “sideways” region &8000 to &BFFF, or that it can intercept OSBYTE &8F). NMIs should not change any locations unless they are specifically allowed to or unless it is their own workspace.

&00A8-&00AF: MOS scratch space. It is not necessary for this space to be preserved between MOS system calls and therefore may be used by other programs during this time. However, it is not recommended for general use because the integrity of the space will not be preserved across MOS calls.

&00B0-&00BF: filing system scratch space - like the MOS scratch space it is not preserved between system calls. During this time other programs may use it although this practice is not recommended because they will not be preserved across filing system calls. “Hidden” filing system calls e.g. those produced by OSWRCH if the command *SPOOL has been used also use this space.

&00C0-&00CF: current filing system workspace - under no circumstances must this area be used because it may be corrupted at any time

&00D0-&00FF: MOS workspace - not available for use by other programs. The VDU driver is fully explained in section E of Part 1 of the Reference Manual.
In previous BBC microcomputers this area contained various pointers and flags for I/O operations. This is not the case with the Master Series.

Pages 1 to &D

&0100-&01FF: processor stack and error messages buffer. The stack follows normal 6502 practice and works as a LIFO buffer at the top of the page. Error messages are stored temporarily at the bottom of the page.

&0200-&0235: vector addresses. For more details of this area please refer to the section on Extending the MOS.

&0236-&028F: main MOS variables - not recommended for any other purpose.

&0290-&02FF: MOS workspace - not available for other purposes.

&0300-&037F: VDU variables. It is only possible to us this area for graphics routines, more details on the use of these are available in sections D, E and F of the Reference Manual Part 1. In earlier BBC microcomputers some of the variables had different functions, details of which are given in the Appendices.

&0380-&03DF: Cassette Filing System workspace - available only if the CFS is not used.

&03E0-&03FF: keyboard input buffer - available only if the keyboard buffer has been replaced.

&0400-&07FF: language workspace - may be used if the current language allows (e.g. BASIC). It is also used for the relocation of the host communications routines with second processors.

&0800-&087F: sound workspace - its use is not recommended as this may cause the generation of spurious sounds.

&0880-&08BF: printer buffer - may be used for other purposes if printing is not required.

&08C0-&08FF: workspace for the sound envelopes 1 to 4 - available for other purposes if the envelopes are not used.

&0900-&09BF: RS423 output buffer, cassette output buffer for access to the first part of sequential files or workspace for sound envelopes 5 to 16 - otherwise available for other purposes.

&09C0-&09FF: Speech buffer or cassette output buffer for access to the second part of sequential files - available to users if not required for these purposes.

&0A00-&0AFF: RS423 input buffer or the cassette input buffer for access to sequential files - available for other uses if not required for these purposes.

&0B00-&0CFF: ECONET workspace - may not be used for any other purpose if at any time the computer will be connected to an ECONET system. In previous BBC microcomputers this area was used for the soft key buffer and the upper 32 characters of the exploded font. This means that previous routines for writing a soft key definition directly into the memory can no longer be used. Correct operation on the Master Series and on the earlier BBC machines can be achieved by using the OSCLI interface.

&0D00-&0D5F: NMI routine workspace. In order to make use of this area for other uses NMIs must be claimed (paged ROM service call &0C). The same restrictions apply to the use of this area as to &00A0-&00A7 which is described above. On earlier BBC microcomputers this region extended to &0D9E.

&0D60-&0D7F: ECONET workspace - it may be used for other purposes if the machine is not going to be connected to an ECONET system.

&0D80-&0D91: available for user programs.

&0D92-&0D9E: Reserved for a Trackerball or Mouse. It is necessary for these devices to have immediate access to non-paged memory in order to service the interrupts from their reference phase signals. This area has been reserved for fast updating of their counters.

&0D9F-&0DEF: extended vector address set, more details of which can be found in the section on extending the MOS.

&0DF0-&0DFF: paged ROM workspace. Usually one byte for each ROM is used for the high byte of the private workspace address. Some ROMs, such as the DNFS also use it to indicate that they are not active by resetting bit 7. The reason for the inactivity may be, for example, that essential hardware is not present or that a particular filing system is dormant.

Pages &E to &7F

The allocation of this area of the memory is variable. Some of the pages at the lower addresses may be used by the paged ROMs or by programs that raise the Operating System High Water Mark (OSHWM). Some pages at the higher addresses may be allocated to the screen, if it is not in shadow mode. The remaining memory is allocated to user memory, i.e. language workspace.

In the Master Series soft character definitions are held in RAM at &8900, whereas earlier BBC microcomputers stored them in RAM above &0E00, raising OSHWM.

Pages &80 to &BF

At any one time, one of sixteen images resides in the memory pages &80 to &BF. These images may be in ROM, RAM, or EPROM and include parts of the operating system, the sideways MOS ROM (ROM &F and the top 1.5k of the ROM &E).

The MOS makes the paged ROM code in the address range &8000 to &8FFF unavailable during graphics and soft-key calls by setting the high bit of the ROM select latch high. This swaps in 4k from a further 32k of RAM. Paged ROMs which need to use this area can do so by calling routines given in the VDU drivers specification section of Part 1 of the Reference Manual. Note great care must be taken when laying out these ROMs to avoid attempts to execute ROM code within the overlaid area.

Sideways ROM numbers 0,1,2 and 3 are allocated to the cartridges and a further “vertical” paging mechanism may be used with these. When using the “vertical” paging mechanism some 1Mbit and 512kbit EPROMs are arranged as sixteen and eight pages of 16k bits respectively. When these devices are plugged into the cartridge slots they will appear as a 16k byte image, but any one of the remaining seven (for the 1Mbit) or three (for the 512kbit) images may be obtained by writing to the EPROM with the vertical page number. This is a major departure from standard EPROMs and allows 512k bytes to be fitted into four EPROMs and yet only use 16k of the computer’s address space. This is illustrated below.

To insert the paged EPROM into the memory map of the computer the value of the EPROM is written to address &FE30. The required vertical image is then selected by writing to any location in the range &8000 to &BFFF. Note this selection is maintained even if through a hard break (e.g. CTRL-BREAK). The next access to these sideways EPROMs will be from the new image. On power-up the special EPROMs default to vertical page 0. To use this facility include a standard ROM header line for each vertical page. An example of a typical paged EPROM is the 27513, which is four pages of 16k bytes.

Pages &C0 to &DF and page &FF

The main MOS ROM resides in the areas &C0 to &DF and &FF. However, in the standard configuration pages &C0 to &DF of the MOS are not directly readable, because the filing system RAM is switched into this area. This part of the MOS contains the graphics routines and is enabled when needed. Another feature which should be noted is that access by instructions in the area &C0 to &DF to data in the locations &3000 to &7FFF are automatically mapped into either the main memory or the “shadow” screen memory depending on the current screen mode. The state of the memory map is determined by the ROM select latch at &FE30 and the memory access latch at &FE34. If these registers have been changed, then the memory map may not behave as described above.

Page &FC

Page &FC is mapped to either the external 1MHz Bus or the cartridges via the signal INFC (INternal FC). The cartridges will be accessed when bit IFJ is set in the register at &FE34. This page is intended to be used for memory mapped hardware.

Page &FD

This page is also mapped to the external 1MHz Bus or the cartridges by the signal INFD. This page will access the cartridges when the IFJ bit of the register &FE34 is set. The page &FD is intended to be used for accessing the remote memory. Note that location &FCFF is reserved as a paging register to allow up to 64k bytes to be accessed through this page.

The Second 32K of RAM

The second 32k of RAM does not occupy one contiguous block of addresses, but is allocated as follows:-

&3000-&7FFF: shadow screen memory - any part of it not required by the current screen mode is available for user programs. Access is gained by manipulating the memory map latch. However, note that the command *MOVE will use this area if one of the non-shadow modes or a shadow mode occupying less than 20k bytes, is being used.

&8000-&83FF: soft-key expansion buffer - not available for any other purpose.

&8400-&88FF: VDU workspace which can only be used for VDU routines that require large amounts of workspace, e.g. flood filling. Care must be taken to avoid conflicts between different routines of this sort. Commercial software should avoid using these areas.

&8900-&8FFF: character definitions.

&C000-&DBFF: paged ROM workspace. The ROMs use service calls to claim the area. This is a similar procedure to the one used to claim space above &E00. Static workspace in this area or above &E00 should only be used by filing systems although any ROM may have private workspace.

&DC00-&DCFF: MOS CLI buffer - this area is corrupted by all * commands, and its use for other programs is therefore not recommended.

&DD00-&DEFF: transient utility workspace and it is available for user written * commands and the *MOVE command.

&DF00-&DFFF: MOS workspace only. It may not be used for any other programs.

VDU Workspace

&00D0-&00D9: non-transient VDU variables and should not be used by any other program.

&00DA-&00E1: VDU scratch space and not available for other purposes.

&0300-&037F: VDU workspace. There are two forms of graphics co-ordinate, internal and external. The external graphics co-ordinate is the one used by the BASIC PLOT command. The internal graphics co-ordinate is derived from the external by taking into account the graphics origin and scaling so that it is measured in pixels, both horizontally and vertically. Graphics co-ordinates are stored in four bytes, with the low byte of the X co-ordinate first.

&8400-&87FF: VDU workspace in the shadow RAM used as scratch space for flood filling. If the flood fill is active, one of the values 0,1,2,3,4,5,6,7,8,9 or A will appear in the location &8601. Therefore any routines that need to use this space must have one or more values allocated to them by Acorn Services and Training Department. If a routine in the set changes any byte in the VDU workspace, it must leave one of its values in the location

&8601. If the workspace is assumed to contain any valid data, it must check that location &8601 contains a suitable value. If location &8601 does not contain a valid value then the routine must take the appropriate action.

VDU workspace allocations

&8400-&87FF: scratch space e.g. flood fill.
&8800-&882F: non-transient VDU variables.
&8830-&88BF: VDU scratch space.
&88C0-&88FF: reserved for future use by non-transient VDU variables.
&8900-&8FFF: current character definitions.

Earlier BBC Microcomputers and the Acorn Electron

&00D0-&00D9: VDU variables. These are not transient and should only be altered in keeping with their function.
&00DA-&00DF: VDU scratch space - it does not need to be preserved between VDU calls, and is not preserved across them.
&00E0-&00E1: non-transient VDU variables.
&0300-&0327: non-transient VDU variables.
&0328-&0349: With the exception of &338, which when in teletext mode is a non-transient variable, this area is a VDU scratch space.
&034A-&037F: non-transient variables.

Extending the MOS

There are occasions when the standard MOS facilities do not meet the requirements of a particular application e.g. when additional hardware has been included in the system. For such situations it is possible to extend or in some cases replace most of the MOS functions with user defined ones. It is possible to make extensions to both the time-dependent and the time-independent functions. It is recommended that users become familiar with the time-independent functions before changing the time-dependent functions which are more complex.

Time-Independent Functions

Time-independent functions may be invoked at any time. The main MOS functions are entered by calling a subroutine (JSR) at the appropriate entry point. (For example, OSWORD is entered at &FFF1.) The actual entry point for the start of the function is stored in a vector table. The routine is accessed by an indirect Jump (JMP) command located at the entry point. In the previous example of OSWORD,

the vector address is &20C and the MOS code at the OSWORD entry point is JMP (&20C). The vectors are stored as a lookup table in RAM at addresses &200-235. The table is initialised on RESET and by substituting vectors which point to user-supplied code it is possible to change the MOS functions.

Vectors in co-processors

Most of the MOS calls are available in the operating system of a co-processor. However, it should be borne in mind that although re-directing a vector in the co-processor will only affect the co-processor, re-directing a vector in the host will affect both the co-processor and the host. For example, intercepting the OSWRCH command with WRCHV in the host in order to change all lower case characters to upper case will change all the output from the host and the co-processor. However, if the intercept takes place in the co-processor then only the output from the current application will be changed, anything from the filing systems which operate only in the host will remain unchanged.

Vectors in Sideways ROM/RAM

Extended vectors may be used to point to sideways memory rather than a location in non-paged memory. This allows the user to specify the ROM (or RAM) slot number as well as the target address. The procedure is shown below.

a) Using OSBYTE 168, read the start of the extended vector space (<evs start>).

b) Starting at (<evs start> + 3*<vector>), place the following data into memory:
<entry point in ROM (least significant byte>.
<entry point in ROM (most significant byte)>.
<ROM slot number>.

c) the relevant vector is then changed to:
&FF00 + (<vector>-&0200)*3/2

The vector's location (<vector>) is selected from the table shown below. The number (<vector>-&0200)/2 is called the vector number.

MOS Function Vector Table

Function	Entry point	Vector name	Vector location
Main MOS Functions			
OSBYTE	&FFF4	BYTEV	&20A
OSWORD	&FFF1	WORDV	&20C
OSCLI	&FFF7	CLIV	&208
OSRDCH	&FFE0	RDCHV	&210
OSWRCH	&FFEE	WRCHV	&20E
OSEVEN	Via Vector	EVNTV	&220
Error (BRK) vector		BRKV	&202
User vector		USERV	&200
Input control			
keyboard operation		KEYV	&228
Output control			
unknown plot codes		VDUV	&226
user print vector		UPTV	&222
Buffer control			
buffer insert vector		INSV	&22A
buffer remove vector		REMV	&22C
buffer control		CNPV	&22E
Filing system functions			
OSFIND	&FFCE	FINDV	&21C
OSGPBP	&FFD1	GPBPV	&21A
OSBPUT	&FFD4	BPUTV	&218
OSBGET	&FFD7	BGETV	&216
OSARGS	&FFDA	ARGSV	&214
OSFILE	&FFDD	FILEV	&212
Filing system control		FSCV	&21E
ECONET vector		NETV	&224
Spare (indirect) vectors			
		IND1V	&230
		IND2V	&232
		IND3V	&234
Interrupt request vectors			
high priority devices		IRQ1V	&204
low priority devices		IRQ2V	&206

Notes

1) OSRDSC, OSWRSC, OSNEWL, OSASCI, GSINIT and GSREAD are not vectored because they have very specific functions, details of which are in the Reference Manuals Parts 1 and 2.

2) It is only possible to access functions without entry points by using vectors. User code must call the function indirectly by JMP (<location>), rather than directly by JMP <location>.
3) OSEVEN has been included in this section because although it is often used as a means of simulating real-time events its use is not restricted to this.
4) USERV has been included in the MOS sub-section because it is used to pass the unknown OSWORDs &E0 to &FF the user.
5) The time-dependent functions use the IRQ vectors and are included here for completeness.

Entry pointed vectors

The entry pointed vectors are used for most of the MOS routines. Part 1 of the Reference Manual fully describes the entry and exit conditions.

Vectors without MOS entry points

These are mainly user defined which means that MOS entry points cannot be defined.

EVNTV

System events may be simulated by using OSEVEN. OSEVEN is called with X being the event to which the routine is to be passed. A and Y are then transposed and X is preserved. The user's routine must preserve all the registers when passed on through EVNTV.

On entry Y corresponds to the event. The following table lists the values for Y and their corresponding events. The values for X and Y are event specific.

Event 0 - output buffer empty

X - buffer number — Y - unused.

0 keyboard
1 RS423 input
2 RS423 output
3 printer
4 sound channel 0
5 sound channel 1
6 sound channel 2
7 sound channel 3
8 speech

Event 1 - input buffer full
X - buffer number (as event 0) Y - overflow character

Event 2 - character entering buffer
X - unused Y - most-recent character

Event 3 - ADC conversion complete
X - unused Y - ADC channel measured

Event 4 - start of vertical sync. (retrace)
X - unused Y - unused
indicates a retrace has started

Event 5 - interval timer crossing zero
X - unused Y - unused
system VIA interval decremented to zero

Event 6 - ESCAPE has been pressed
X - unused Y - unused
Escape condition will not be generated or transmitted to parasite

Event 7 - RS423 error
X - 6850 status shifted right Y - char received

Event 8 - network event
X - lsb Y - msb of remotely requested procedure

Event 9 - user event
conditions are user-defined

Event &FE - network receive
This event is enabled by *FX52,150 ctrl blck # and disabled by *FX52,100. It is not affected by *FX13 and *FX14.

Note an escape condition will not be transmitted to the parasite when the ESCAPE key is pressed if an escape condition has not been generated by changing bits 6 and 7 in location &00FF.

BRK instruction

The instruction BRK is the software equivalent of the 65C12 processor to a hardware Interrupt ReQuest (IRQ). BRK fetches the next instruction from the address stored in &FFFE and &FFFF, which is the address of the IRQ routine in the MOS. The IRQ routine sets up the stack as described below and then via BRKV performs a JMP.

The BBC microcomputers use this mechanism to indicate an unrecoverable software fault and use the vector to implement error routines. For example, languages use the vector to point to their error handlers. The user routines pointed to by the BRKV command should exit via the old contents of the vector because the stack will have been modified.

The command ReTurn from Interrupt (RTI) should not be used as it may cause the program to jump into the stack where the error message might be.

On entry A, X, and Y will remain set up as they were before the BRK command.

An RTI instruction will be set to return the stack pointer to the location two bytes after the BRK command. RTI should only be used if special user code has been sent after the BRK instruction as opposed to the error structure described next.

The locations &00FD and &00FE are a pointer, placed by the MOS, to the location after the BRK.

The current stack pointer will be contained in location &00F0.

The slot number of the ROM that was active when the BRK instruction was issued can be read by OSBYTE &BA.

The following structure should be placed after the BRK.

```
                BRK
<pointer>       <error number>
                <first byte of error message>
                        "
                        "
                        "
                <last byte of error message>
                &00
```

The null is a recognised means of ending a message. The handler should interpret it accordingly.

BRK instruction in single processor systems

The Entry Structure is set up as shown above, but Service Call 6 (BRK) is performed before the vector indirection is performed so that the filing systems, or any other service ROM, can take the appropriate action.

BRK instruction in co-processor systems

If the BRK is executed in the host, the above structure is set up in the co-processor but terminated with an IRQ. This causes the Tube Operating System (TOS) to make a copy of the BRK and error string in its own memory. The BRK is then executed; it is treated as if the BRK had originated in the co-processor. If the BRK is originally executed in the co-processor, the error pointer is calculated as normal, interrupts are re-enabled and BRKV is used. Service Call 6 is not issued.

USERV

The USERV instructions cause program flow to be directed via USERV. This may be used for user-defined OSWORD calls. Entry to routines via the *CODE and *LINE commands is simplified by using the USERV vector.

Entry condition:

A=0	*CODE has been entered.
A=1	*LINE has been entered. For further information, refer to the Reference Manual part 1.
A=&E0-&FF	the indicated Unknown OSWORD has been called.

On Exit: A, X and Y should be the same as on entry and the user routine should end with an RTS instruction.

KEYV

The instruction KEYV is used to read the keyboard and it is this instruction that informs the MOS just how much work to do on the keyboard. The required operation is indicated by the status bits C (carry) and V (overflow). Normally these are set and serviced by the MOS. However, by redirecting this vector the user can invoke, supplement, or replace the normal MOS keyboard scanning, for example, to add an alternative keyboard. The vector can also be used to allow keyboard scanning when the interrupts have been switched off.

On entry:

If **C=0 and V=0** then the SHIFT and CTRL keys will be read, returning N=1 if CTRL is pressed and V=1 if SHIFT is pressed.
If **C=1 and V=0** the keyboard is scanned as described by OSBYTE &79.
If **C=0 and V=1** the key-pressed interrupt is serviced. This causes OSBYTE &78 to be performed which reads the character corresponding to the pressed key into memory.
If **C=1 and V=1** normal keyboard scanning will take place unless OSBYTE &C9 has been used to disable it. This entry is made once every 10ms until all key depressions have been removed. This processing includes SHIFT, SHIFT LOCK, CAPS LOCK and CTRL.

VDUV

A number of VDU control sequences are “unknown” to the MOS, this means that the MOS has no internal routines to which they correspond and therefore it passes control via VDUV to another code that may be able to deal with it. Those listed below are not the only unknown VDU codes but are merely those not previously assigned to other purposes. The Reference Manual Part 1 has a full list of the assignments.

VDU 23, <28 to 31>: This is used to provide up to 8 further parameters all of which must be supplied, even if they are zero.

VDU 25, <240 to 255>: These are the unknown graphics plot commands. If a VDU 25 command is made in a non-graphics area VDUV will be used.

VDU 25, <208 to 231>: Currently these are undefined.

All unknown VDU calls are indicated by the C (carry) flag.

On Entry:
C=0: Unknown PLOT (VDU 25) command. The parameters are stored in VDU variables 31 to 35 and can be read by OSBYTE &A0.

VDU variable	Contents
31	command number
32	co-ordinate X least significant byte
33	co-ordinate X most significant byte
34	co-ordinate Y least significant byte
35	co-ordinate Y most significant byte

The co-ordinates will already be scaled into internal pixel co-ordinates.

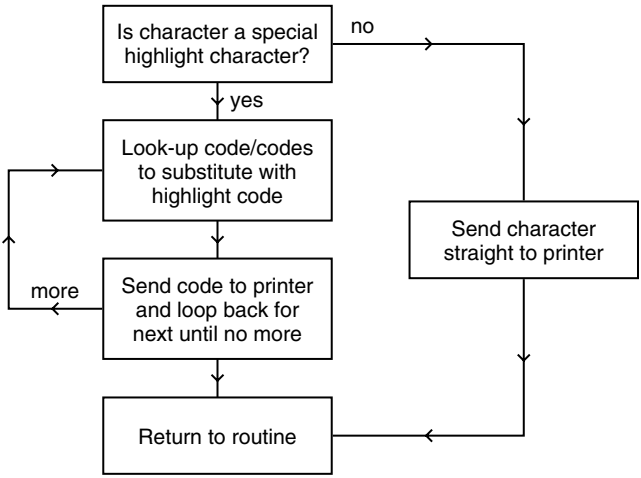
C=1: User defined ASCII command. A= the command number, the remaining eight variables are in VDU variables 27 to 35.

In non-graphics modes the parameters will be stored as in C=0 given above. The co-ordinates will not be scaled to internal co-ordinates in text modes because they have no meaning.

On Exit:
If the code is unknown to you the program flow should be returned via the OLD contents of VDUV, otherwise use an RTS to terminate the code.

UPTV

The User Print Vector (UPTV) is provided for user printer routines. There are two ways of enabling this vector, by using *FX5,3 or by default using the *CONFIGURE PRINT 3. UPTV is re-directed to point to user printer control routines. This facility is especially useful if the printer has special features which cannot be accessed by the standard printer drivers. Printers often have their more powerful features invoked by sending an <ESCAPE> character followed by a number of characters which specify the parameters to be used. The substitute printer driver can translate the special characters into the required command sequences. The following figure shows the flow of data under these circumstances.



Printer data flow

UPTV can be called when another printer driver is active, as shown below. If this is the case control should be returned via the old contents of UPTV rather than terminating with RTS.

In the following cases, when UPTV is used, it is the responsibility of the printer driver to manipulate the computer's parallel printer port directly or to output serial data via the RS423 stream. The A-register notifies the printer of the required operation and on exit the carry flag is used to indicate a result.

Entry
condition

A=0 the driver is entered this way once every 10ms, unless it has indicated that it is dormant, which is described below. The driver should ensure that the ACKnowledge line of the printer is active (high) and read a character from the buffer using OSBYTE &91. After any necessary translation the character is sent to the printer.
On exit the driver should declare itself dormant if the buffer is empty by using OSBYTE &7B followed by an RTS. This enables the printer driver to be changed if necessary and prevents the MOS from wasting time by sending 10ms calls to an empty buffer.

A=1 the driver has previously been dormant and one or more characters had been placed in the buffer. The reading and printing of the characters is as for A=0. On exit, the carry flag signals the buffer state to the MOS (C=1 shows that the buffer is empty).

A=2 ASCII code 2 (Ctrl-B) has been sent to the driver. Except in shared systems where it is used to claim a remote printer, the driver should be made to ignore this code.

A=3 ASCII code 3 (Ctrl-C) has been sent to the driver.

A=4 not used.

A=5 the printer type has been redefined using OSBYTE 5. The new printer driver number is in X.

FSCV

The vector FSCV provides access to a number of miscellaneous filing system functions. The required function is indicated by a reason code in the accumulator. Unless indicated, the registers are not defined and interrupts may be enabled during the call.

Entry
Condition

A=0 A *OPT command has been issued with X and Y as parameters.

A=1 check for End Of File (EOF) - file handle in X-register.
If on exit EOF is true X=&FF, otherwise X=0.

A=2 */<FILENAME> command has been issued. The filing system should try to *RUN the file named after the / symbol.

A=3 attempt to *RUN specified file. X and Y contain the lsb and msb respectively of the address of the ASCII string containing the name of the file. This call originates when a * command has been rejected by all ROMs. If the file cannot be *RUN, the message “BAD COMMAND” will be issued rather than “FILE NOT FOUND”.

A=4 X and Y point to the name of a file to be *RUN.
A=5 X and Y point to a string containing the parameters of a *CAT command that has just been issued.
A=6 another filing system is being invoked so *SPOOL and *EXEC files should be closed and other open files should be ensured.
A=7 the filing system is being interrogated to supply its range of file handles. On Exit X= the lowest handle, Y= the highest.
A=8 an OSCLI command has been issued. This call permits filing systems to ensure the integrity of their media.
A=9 a *EX command has been issued and the information is sent to the output stream.
A=10 *INFO command has been issued. The information is sent to the output stream.
A=11 *RUN a file via LIBFS.
A=12 *RENAME command has been issued.

INSV

The INSV Vector can be used to invoke a custom routine to insert characters into a specified buffer or to provide a much larger buffer.
On entry A=<character> and X=<buffer number>.
On exit C=1 if the buffer is full. (The MOS will abort or retry in response to this.)

REMV

The REMV vector may be used to invoke a routine to remove a character from the buffer or simply to examine the character.
On entry X= the buffer number, V=0 to remove the next character from the buffer or V=1 simply to examine the next character.
On exit C=1 if the buffer was empty, X is preserved, Y is the character to be removed, or A=the character that was examined.

CNPV

The CNPV vector points to a routine to count the number of characters in a buffer or to flush that buffer.
On entry X = the buffer number. To count the characters set V and C to 0 and to count the spaces set V to 0 and C to 1. To flush the buffer set V to 1.
On exit the values of V and C are preserved. If a count has been made, X= count least significant byte and Y= count most significant byte.

NETV

The NETV vector usually points to the routine which initialises the Advanced Network Filing System (ANFS) and thus permits the use of utilities like *VIEW and *REMOTE. The NETV facility can be used by user code for this purpose or to restrict the ECONET access to a particular part of the system by filtering out unwanted commands. On entry the function to be performed is contained in A.

Entry
condition

A=0-3,5 printer commands, same as for UPTV. The number for the ECONET printer driver is 4.

A=4 OSWRCH has been called.
On exit the character will be output if C=0, otherwise C=1.

A=6 OSRDCH has been called.
On exit the network should put the character into A.

A=7 OSBYTE has been called. The values of A, X, and Y are stored at &00EF to &00F1.
If on exit the call is passed to OSWORD then V=0, otherwise V=1.

A=8 a line has been read by OSWORD 0. ANFS can now take over OSRDCH.

INDirect Vectors

There are three indirect vectors available, these are IND1V, IND2V and IND3V. The indirect vectors are used to access sideways ROMs and the Terminal Emulator uses IND1V and IND2V.

Note on the entry points for these vectors
These routines are not provided with entry points, but the MOS versions of them terminate with an RTS. They should be called by:

```
                    JSR <callroutine>
                     "
                     "
                     "
                     "
.callroutine JMP (<vector>)
```

This performs a Jump to Subroutine and then an indirect Jump.

Time-dependent functions

In the previous section on time-independent functions some functions which might have been expected to be time-dependent were described. This was because software routines may be used to simulate tasks which are normally dictated by external events, a technique which is frequently used to develop real time software. Real time events usually occur at a high frequency compared with the time taken to run the service software and also they may occur fairly quickly in relation to other events.

Real time events are initiated by hardware, either internal or external, which passes an interrupt request (IRQ) to the CPU. An IRQ is generated by pulling the IRQ pin of the CPU low. As all devices are connected to this pin, the MOS has to interrogate them to determine which device was the source of the interrupt. When the source device has been identified the MOS will service it and perform a vectored subroutine call via EVNTV to pass on the information.

If the CPU cannot determine the source of the interrupt it offers it to each of the sideways ROMs or RAMs. In this way hardware which uses interrupts (for example, on the 1 MHz bus) may be accommodated. Whichever page the controller software is in, it will ultimately be notified of the interrupt.

The time this takes may result in data being lost. In order to alleviate this problem the computer can be set up to give the user the chance of identifying an interrupt before it is passed round the computer, or back to the MOS.

EVNTV

The entry parameters for EVNTV are detailed in the previous section. If any extra hardware has been added to the computer, it will generate an interrupt to cause the MOS to pass control via EVNTV with A=9, if it has not been able to determine the source of the interrupt itself. Note this only happens if the USER Event has been enabled with OSBYTE &E,9.

In order to process the IRQs quickly, it may be necessary to process them before they are passed round the sideways ROMs, or in some cases before the MOS services them. Two vectors IRQ1V and IRQ2V are provided for this purpose.

Function	**vector name**	**location**
To access the highest priority devices.	IRQ1V	&204
To pass the event round paged ROMs	IRQ2V	&206

All the user interrupt routines should be as short as possible, the recommended maximum is 0.5ms. This is particularly important when using IRQ1V because this services the interrupt before the MOS. As an example consider the operation of RS423 at 19,200 baud, which corresponds to one byte being transmitted every 416ms. As all interrupts would have to pass through user code pointed to by IRQ1V before the MOS could deal with them, a 2ms service routine would occupy the time for 4.8 bytes. This would lower the average speed to about 4000 baud. When the MOS is selected by IRQ1V (which is usually the case), it examines devices in the following order.

1. The 6850 ACIA which controls the RS423 interface and the cassette data.

2. The System Versatile Interface Adapter (VIA) which controls the vertical synchronisation, the interrupts, the light pen (if included in the system), the A/D converter, the system timer, the sound system, the keyboard and the real time clock.

3. The User VIA which controls the User Port and the Parallel Printer Port.

Note the manufacturers data sheets for these devices should be consulted for details of the interrupt status registers of these devices.

12 DUAL PROCESSOR SYSTEMS

Second processor architecture

To enhance the computing power of the BBC microcomputer, Acorn has adopted a two-processor architecture. The base, or host, processor performs most of the I/O routines, such as communicating with the keyboard and filing systems, whilst the language, or parasite, processor provides the raw computing power to perform applications.

The host processor is a 6502 in the Model B and a 65C12 in the B+ and Master Series. Acorn language processors range from the 8-bit 65C02 and Z80, through the 16/32-bit 80186 to the 32-bit 32016 and Acorn RISC Machine. Third-party manufacturers supply Z80, 6809 and 68000 systems. (The ARM second processor architecture is slightly different from that of the other language processors as it is provided with its own peripheral controller chips and communicates directly with the video and audio outputs. However, filing and other I/O operations still take place through the host.)

Each processor runs independently of the other and is provided with its own clock and memory chips. The two systems communicate with one another over a 2MHz asynchronous bus, known as the Tube, which is controlled at each end by a custom interface.

Since the language processor does not need to control complex peripherals directly, it can manage with only a rudimentary operating system. This MOS is required simply to initialise the system on RESET and to implement calls such as OSBYTE and OSWORD. The base processor then performs the required operations and returns the result to the language processor.

Not all the MOS calls are fully implemented. For example, filing system control is carried out by the base processor, so FSCV is not required and, in the Master Turbo for example, points to a “Bad” error routine; the default setting of EVNTV and the user-set vectors point to an RTS opcode. Whilst the operation is being carried out, the language processor can continue executing its application.

Operating system calls are implemented by transferring the call and its parameters to the base processor, which performs the desired operation and sends a response back via the Tube. To speed matters up, only the the minimum required number of

parameters is transferred. For instance, with OSBYTE calls 0-&7F, the Y-parameter is omitted. For those calls in which the carry status is a significant part of the result, it is transferred across the Tube by performing a shift operation in the source processor and a complementary shift operation to prime the carry flag in the destination processor.

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

Usually the language processor is provided with a clear block of contiguous read/write memory. Its boot operating system is in ROM which is mapped into the top of the processor's address range. On RESET the MOS is copied from ROM into RAM and awaits initialisation via the Tube. Processors such as the 80186, which run industry standard operating systems, have a ROM-based startup but load the remainder in from disc.

When functioning as an I/O processor, the base processor installs Tube communications routines in the regions of low memory which are normally allocated to the active language (addresses &0016-&005C in Page 0 and Pages 4-7). These communications routines have language and service entry points similar to those of paged ROMs and also a data entry point which is used once the Tube has been initialised.

If the second processor is added externally it is referred to as a "Second Processor" and one added internally as a "Co-Processor". Except where indicated, references to a co-processor apply equally to a second processor.

The Tube

The Tube provides the means for the language and I/O processors to communicate with each other. The Tube comprises a pair of proprietary chips coupled to the respective processors and communicating with one another over a 2MHz asynchronous bus.

The Tube chip is a semi-custom integrated circuit designed to overcome the problems of interfacing between processors running at different instruction and bus cycle rates. The language processors have different clock rates from that of the base processor and may also have incompatible instruction sets, which prevents the possibility of direct (synchronous) coupling between them. The Tube chip is therefore provided with the buffers and latches necessary to implement asynchronous coupling.

Tube Architecture

The Tube has two one-byte wide ports. One port is for the host and the other for the parasite. The ports provide access for the host and the parasite to a number of registers.

The Tube chip is located on the language processor circuit board and is connected to the host by a byte wide bus.

The Tube protocols allow the language processor to have full access to the filing system, the VDU Driver, the RS423 or any other I/O devices connected to the microcomputer.

The protocol is a set of software rules for passing data across the Tube chip. The Tube protocols are partly held in the MOS and partly in the language processor. Data is referred to as being passed “across the Tube”.

Tube Protocols

The protocols are sequences of read/write operations to the Tube chip that have to be performed in order to pass data between the host and parasite. Some sequences enable an application in the parasite to control the host, request data and transmit it to the outside world and are usually initiated by firmware routines in the parasite. These in turn will have been called by the applications program running in the language processor RAM.

Other sequences are used to pass events, errors and effect low-level block transfers; these are initiated by the host. There are sixteen different sequences, each designed for a specific task. Note that there are two calls which are only designed for use in the host to ensure compatibility with previous BBC Microcomputers. Three others are not intended to work “across the Tube” and are only mentioned here for completeness. The full list of sequence names and their purpose follows:

OSBYTE	Execute a MOS routine requiring up to a three byte argument.
OSWORD	Execute a MOS routine requiring a parameter block.
OSCLI	Interpret a *<text> command.
OSRDCH	Read a character from the input stream (e.g. RS423, keyboard).
OSRDSC	Read from the screen. (not available to language processors)
OSWRCH	Write a character to the output stream (e.g. RS423, screen).
OSNEWL	Write LF followed by CR to the output stream.
OSASCI	Write a character to the output stream, or LF followed by CR if the character is CR.
OSWRSC	Write to the screen. (not available to language processors)
OSFIND	Open or close a file for byte access.
OSFILE	Load or save a file.
OSARGS	Load or save data about a file (e.g. sequential pointer, extent).
OSGBPB	Load or save part of a file.
OSBPUT	Save a byte to a file.
OSBGET	Load a byte from a file.
OSEVEN	Generate an event. (not available to language processors)
GSINIT	Initialise GSREAD string. (not available to language processors)
GSREAD	Read a byte from a string. (not available to language processors)

The names are for reference only. The form of parameter(s) used by each sequence is listed in the Reference Manual, Part 1. Whatever microprocessor is used in the parasite, a given sequence with given parameters will always work in the same way.

In this text, “H➡P” indicates the passage of data from the host to the parasite and “P➡H” shows the passage of data from the parasite to the host.

Each protocol consists of read/write accesses to the Tube registers, conditional branching based on the register contents, and the copying of the contents into memory. The Tube chip appears, to both the host and the parasite, as a collection of memory or I/O mapped registers. There are four independent bi-directional communication paths, each of which consists of a one byte control register and a one byte data register (which may have a one-byte buffering). The roles of the respective registers are described below:

Operating System Usage

Registers R1STATUS, R1DATA, R2STATUS and R2DATA are mainly for MOS data and command transfer under polled or parasite IRQ operation.

Register 1 status (R1STATUS)

The status of R1DATA is indicated by this byte:

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	DA1	NF1	P	V	M	J	I	Q

When set to logic 1:

- DA1 - Data Available in data register 1
- NF1 - Data register 1 is Not Full
- P - Set parasite reset active low
- V - Enable two byte FIFO operation of R3DATA
- M - Enable parasite NMI from R3DATA
- J - Enable parasite IRQ from R4DATA
- I - Enable parasite IRQ from R1DATA
- Q - Enable host IRQ from R4DATA (Not Used)

Register 1 data (R1DATA)

H➡P
A 1-byte buffer is used by events in the host to generate IRQs to the parasite. Writing to this register will cause the parasite IRQ to be active low. It is also used to pass on the ESCAPE condition.

P➡H
This is a 24-byte FIFO buffer and carries the parameters for OSWRCH. Note that OSWRCH only uses a 10-byte parameter block, so a language processor can enter a full plot command without having to wait for the host to remove each byte in turn. Although the Tube chip circuitry is designed to be able to interrupt the host if the parasite writes to this register, this facility is not used on the host, which will normally poll R1STATUS until the data becomes available.

Register 2 status (R2STATUS)

The status of R2DATA is indicated by this read only byte.

Bit	7	6	5	4	3	2	1	0
	DA2	NF2	1	1	1	1	1	1

When set to logic 1:
DA2 - Data Available in data register 2
NF2 - Data register 2 is Not Full

Register 2 data (R2DATA)

Register 2 initiates MOS calls which may take a long time or must not interrupt host tasks.

H➡P
The host returns data as appropriate.

P➡H
The parasite requests the task and then passes data as appropriate.

Filing System Usage

Registers R3STATUS, R3DATA, R4STATUS and R4DATA are mainly used by filing systems for fast transfer under NMIs - may be used for high speed protocols by “claiming” the Tube (see section on the Host Protocols).

Register 3 status (R3STATUS)

The status of R3DATA is indicated by this read only byte.

Bit	7	6	5	4	3	2	1	0
	DA3	NF3	1	1	1	1	1	1

When set to logic 1:
DA3 - Data Available in R3DATA/Parasite NMI generated
NF3 - Data register 3 is Not Full

Register 3 data (R3DATA)

H➡P; P➡H
This is used for the fast data transfers. Note that the host can program it to operate in a two byte mode.

R3DATA and R3STATUS are used for the block transfers as a background task. For higher performance applications this register may interface to a DMA controller.

Register 4 status (R4STATUS)

The status of R4DATA is indicated by this read only byte.

Bit	7	6	5	4	3	2	1	0
	DA4	NF4	1	1	1	1	1	1

When set to logic 1:
DA4 - Data Available in R4DATA/Parasite IRQ generated
NF4 - Data register 4 is Not Full

Register 4 data (R4DATA)

H➡P
Writing to R4DATA sets the parasite IRQ. Reading R4DATA clears the IRQ.

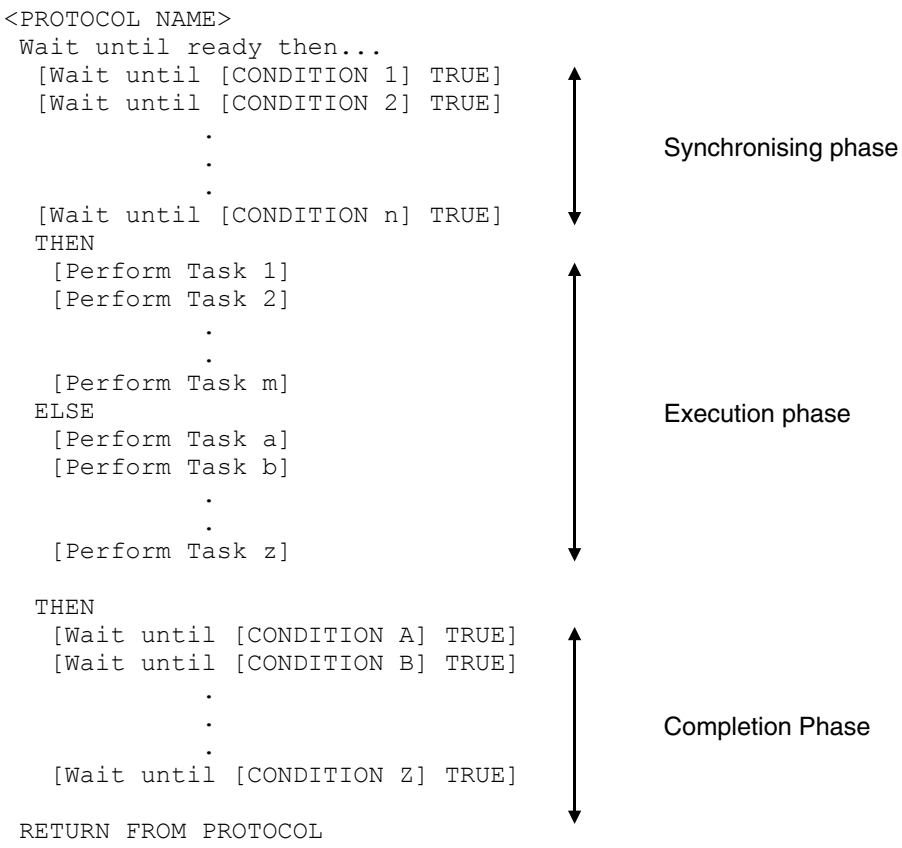
The Host interrupts the second processor by writing a byte describing the required action into R4DATA. The two machines then co-operate in passing data across register 4 until the job is done.

The register set is also used to initiate the passing of an error string from Host to Parasite. The Host interrupts the Parasite by writing an error code into R4DATA, the two machines then co-operate in passing the error string across R2DATA.

P➡H
R4DATA is used as a control channel to request block transfers through R3DATA.

Parasite Protocols

From the point of view of the language processor, the Tube protocols are presented in the following generalised form:



Vectors

Each Acorn-supplied second processor has a simple operating system which contains all of the routines necessary to implement the Tube communications protocols. This operating system is ROM-based and is copied across into RAM when the second processor is reset.

As the Master 128 65C12 and the Master Turbo 65C102 co-processor are opcode compatible, the entry points and vectors for a given OS call are the same in each. This also applies to the 6502 second processor.

Hardware Dependency

Host Hardware

Hardware dependent calls should not be redirected, as user code in the language processor cannot access the hardware (unless the user has set up a program in the host to intercept, say, a standard OSFILE call and turn it into a user-defined OSWORD).

Note that with the exception of the “1MHz Bus”, Cartridge Bus and User Port, Acorn does not support direct user control of hardware.

Parasite Hardware

The only hardware available to a program in the parasite is the CPU, memory and Tube. Redirecting, say, a VDU operation is of limited use. The exception to this is if the user is running the program in a specially constructed (external) second processor which has perhaps its own ultra-high resolution graphics circuitry, or a signal processing system to which the host does not have access.

Non-Interrupt protocols

OSWRCH

Wait until R1DATA not full, write character into R1DATA

OSRDCH

Wait until R2DATA not full, write RDCHNO (=&00) to R2DATA
Wait for data in R2DATA, top bit of R2DATA is 65C12 C-flag (validity bit)
Wait for data in R2DATA, R2DATA is 65C12 A register (character read).

OSCLI

Wait until R2DATA not full, write CLINO (=&02) to R2DATA
FOR all characters in the command string (including terminating <cr>)
DO [Wait until R2DATA not full, write character to R2DATA]
Wait for data in R2DATA and read it
IF this byte=&80 then code has been loaded into the language processor store as a result of the command and it should be entered at the address given by the last R4 protocol type 4 address. This means that another protocol has been invoked by this one and has already finished.

OSBYTE

IF osbyteno < &80 THEN
    Wait until R2DATA not full, write OSBYTNO (=&04) to R2DATA
    Wait until R2DATA not full, write parameter for 65C12-X to R2DATA
    Wait until R2DATA not full, write osbyte number to R2DATA
    Wait for data in R2DATA, read R2DATA which is 65C12-X register
ELSEIF osbyteno = &82 THEN
    result is machine high order address
ELSEIF osbyteno = &83 THEN
    result is low memory value
ELSEIF osbyteno = &84 THEN
    result is high memory value
ELSE
    Wait until R2DATA not full, write BYTENO (=&06) to R2DATA
    Wait until R2DATA not full, write parameter for 65C12-X to R2DATA
    Wait until R2DATA not full, write parameter for 65C12-Y to R2DATA
    Wait until R2DATA not full, write osbyteno to R2DATA
        IF osbyteno=&9D THEN RETURN from protocol (no reply)
        (Note: this is why OSBYTE &9D is faster than OSBPUT)
    Wait for data in R2DATA, bit 7 of byte read is from 65C12-C
    Wait for data in R2DATA, byte read is 65C12-Y
    Wait for data in R2DATA, byte read is 65C12-X

OSWORD

IF oswordno = &00 THEN (Note: Doing readline)
    Wait until R2DATA not full, write RDLNNO (=&0A) to R2DATA
    Wait until R2DATA not full, write upper bound char to R2DATA
    Wait until R2DATA not full, write lower bound char to R2DATA
    Wait until R2DATA not full, write length allowed to R2DATA
    Wait until R2DATA not full, write &07 to R2DATA
    Wait until R2DATA not full, write &00 to R2DATA
    Wait for data in R2DATA $ response
    IF response > &7F THEN RETURN from protocol (escape was pressed)
    Read a <cr> terminated string from R2DATA
ELSE
    Wait until R2DATA not full, write WORDNO (=&08) to R2DATA
    Wait until R2DATA not full, write oswordno to R2DATA
    Wait until R2DATA not full, write number of params to send to R2DATA
    Write parameter block to R2DATA, last byte first
    Wait until R2DATA not full, write number of parameters to receive to R2DATA
    Read bytes back from R2DATA into parameter block, last byte first

The number of parameters to send/receive is determined by:

IF oswordno < &15
THEN [Determine the number of parameters from following table:]

OSWORD number	Parameters to send	Parameters to receive
1 (&1)	0	5
2 (&2)	5	0
3 (&3)	0	5
4 (&4)	5	0
5 (&5)	2	5
6 (&6)	5	0
7 (&7)	8	0
8 (&8)	14	0
9 (&9)	4	5
10 (&A)	1	9
11 (&B)	1	5
12 (&C)	5	0
13 (&D)	0	8
14 (&E)	1	24
15 (&F)	32	0
16 (&10)	16	1
17 (&11)	13	13
18 (&12)	0	128
19 (&13)	8	8
20 (&14)	128	128

ELSEIF oswordno < &80 THEN
    Number of parameters to send=16
    Number of parameters to receive=16
ELSE
    Number of parameters determined in call specific manner (e.g. by embedding in transfer block)

OSBPUT

Wait until R2DATA not full, write BPUTNO (=&10) to R2DATA
Wait until R2DATA not full, Y to R2DATA (file handle)
Wait until R2DATA not full, A to R2DATA (byte to write)
Wait for data from R2DATA, discard it

OSBGET

Wait until R2DATA not full, write BGETNO (=&0E) to R2DATA
Wait until R2DATA not full, write file handle to R2DATA
Wait for data in R2DATA, top bit of byte is 65C12-C (validity bit)
Wait for data in R2DATA, read R2DATA which is byte read from file.

OSFIND

Wait until R2DATA not full, write FINDNO (=&12) to R2DATA
Wait until R2DATA not full, write type of open to R2DATA
IF type=0 THEN
    Wait until R2DATA not full, write file handle to R2DATA
    Wait for data in R2DATA, read result
ELSE
    Waiting for R2DATA not full, write filename to R2DATA (including <cr>)
    Wait for data in R2DATA, read handle from R2DATA

OSARGS

Wait until R2DATA not full, write ARGSNO (=&0C) to R2DATA
Wait until R2DATA not full, write file handle to R2DATA
Waiting for R2DATA not full, [write 4 bytes osarg-data to R2DATA] (most significant byte first)
Wait until R2DATA not full, write operation code to R2DATA
Wait for data in R2DATA, read fs type from R2DATA
Waiting for R2DATA, [read 4 bytes osarg-data from R2DATA] (msb first)

Note: osarg-data is the file sequential pointer or length depending on the type of OSARGS call.

OSFILE

Wait until R2DATA not full, write FILENO (=&14) to R2DATA
Waiting for R2DATA not full, [write 16-byte OSFILE control block to R2DATA] (last byte of block is written first)
Waiting for R2DATA not full, write filename to R2DATA (including <cr>)
Wait until R2DATA not full, write type of transfer to R2DATA (Any transfer is completed under interrupt using R3, R4)
Wait for data in R2DATA, read R2DATA AND &7F = Filing system type
Waiting for data in R2DATA, [read back 16-byte control block from R2DATA] (last byte of block is read first)

Note: The 16-byte control block has the format:

0	Load address
4	Execution address
8	Data start address or length *
12	End address or attributes *

* The contents of these fields depend on the call type e.g. catalogue information, file addresses. See the Reference Manual, Part 1.

OSGBPB

Wait until R2DATA not full, write GBPBNO (=&16) to R2DATA
Wait until R2DATA not full, [write 13-byte OSGBPB control block to R2DATA] (last byte of block is written first)
Wait until R2DATA not full, write type of transfer to R2DATA
Waiting for data in R2DATA, [read back 13-byte control block from R2DATA] (last byte of block is read first)
Wait for data in R2DATA, read R2DATA bit 7 is 65C12-C bit
Wait for data in R2DATA, read R2DATA which is 65C12-A register

Interrupt driven operations

In addition to these parasite-initiated activities the parasite is also required to respond to interrupts from registers 1, 3 and 4.

To determine the source of an interrupt it is important to follow the following order:

a) Check for register 4 interrupt
b) Check for register 1 interrupt

Register 1 interrupts

Register 1 interrupts occur only in the host-to-parasite direction. The interrupt service sequence is:

```
Read type byte from R1DATA
        IF type <0 THEN ; Escape flag update
                Replace the escape flag with bit 6 or type
                RETURN from servicing interrupt
        ELSE ; Event signal
                Interrupt-R1DATA-read 65C12 Y-event parameter
                Interrupt-R1DATA-read 65C12 X-event parameter
                Interrupt-R1DATA-read 65C12 A-event parameter
                ; Host machine will now continue processing
                ; any other actions to service event can be taken

        Where Interrupt-R1DATA-read is:
                UNTIL data-ready-in-R1 DO
                        [ IF data-ready-in-R4 THEN CALL R4-interrupt-service ]
                RETURN read R1DATA
```

Register 4 Interrupts

Read Type byte from R4DATA
IF TYPE <0 THEN ; HOST machine is reporting an error
    Wait for data in R2DATA, read and discard it
    Wait for data in R2DATA, Read error number from R2DATA
    Read a zero byte terminated string from R2DATA
ELSE [;Type is a command to initialise for Register 3 block transfer
    Wait for data in register 4, read claimer's identity from R4DATA (See Note 4)

CASE Type OF
- **0 -** Single byte transfer Parasite to Host.
  Read 4-byte base address for transfer from R4DATA msb first.
  Set NMI routine for this transfer type.
  Wait for & remove synchronising byte from R4DATA
- **1 -** Single byte transfer Host to Parasite.
  Read 4-byte base address for transfer from R4DATA msb first.
  Set NMI routine for this transfer type.
  Wait for & remove synchronising byte from R4DATA
- **2** - Double byte transfer Parasite to Host.
  Read 4-byte base address for transfer from R4DATA msb first.
  Set NMI routine for this transfer type.
  Wait for and remove synchronising byte from R4DATA
- **3** - Double byte transfer Host to Parasite.
  Read 4-byte base address for transfer from R4DATA msb first.
  Set NMI routine for this transfer type.
  Wait for & remove synchronising byte from R4DATA
- **4** - No transfer (Pass address Host to Parasite only).
  Read 4-byte address from R4DATA msb first.
  Wait for data in R4DATA, discard it.
- **5** - No transfer (Filing system release)
- **6** - 256-byte transfer Parasite to Host without interrupt.
  Read 4-byte base address for transfer from R4DATA msb first.
  Wait for data in Register 4, discard it.
  Transfer 256 bytes to Host via R3DATA.
  Write a byte into R4DATA ;To stop unwanted interrupts on Host
- **7 -** 256-byte transfer Host to Parasite without interrupt.
  Read 4-byte base address for transfer from R4DATA msb first.
  Wait for data in Register 4, discard it.
  Transfer 256 bytes from Host via R3DATA.

  ] RETURN ; From the interrupt

Notes:

1) Synchronising Bytes for types 0-3: As soon as the synchronising byte is removed, Register 3 transfer requests (NMIs) will start to occur. The data in a synchronising byte has no meaning; it is merely a handshake signal. When the interrupt occurs 1 or 2 bytes are transferred (depending on the current mode).

2) Filing System releases NMI ownership. A release (type 5) is a guarantee that no more Register 3 NMIs will occur for the current transfer.

3) Interrupt Service Time. The interrupts are caused by some external peripheral (e.g. discs or the ECONET) which cannot be slowed down, so the transfers must take place within the following times:

Type	Maximum allowed Time for NMI service routine	Maximum permissible service time from sync byte to first transfer NMI
0	24 mS per byte	24 mS
1	24 mS per byte	24 mS
2	26 mS per pair of bytes	24 mS
3	26 mS per pair of bytes	24 mS
6	10 mS per byte	19 mS
7	10 mS per byte	19 mS

4) Filing System claimer identities

When a filing system claims the R3/R4 resource in the host its identity is passed to the second processor as part of the R4 startup protocol. The identity codes, which are six-bit numbers, are not related to filing system or ROM slot numbers. They are arbitrary assignments made by ACORN.

Filing System	Claim identity used
Tape	0
DFS	1
NFS (Low Level)	2
NFS (Filing System)	3
ADFS	4
TFS (Telesoft Filing System)	5
Reserved for Acorn Use	6
VFS (Video filing system)	7
SRM (SRAM Utilities)	8
Z80 (For CP/M usage)	9

The Identity “&F” has been used by an independent manufacturer.

Startup protocol

The startup sequence for a language processor (e.g. when power is switched on, or Reset is pressed) is:
Use the OSWRCH mechanism to write out a startup message.
Send a zero byte to Host via R1DATA to terminate it.
Wait for data in R2DATA. ; during this wait a load may occur from the Host
; using R4/R3 block transfer protocols
IF byte=&80 THEN execute from the address given in the R4 type 4 transfer.

Notes:

1) The host operates the Tube by polling the registers, i.e. not by interrupts.
2) In all the transactions which may generate errors it is important to realise that if the error is reported by the BBC machine under interrupt (i.e. it was generated by a 65C12 BRK sequence), the protocol which generated the error is abandoned.

Register Addresses

The Tube can be put anywhere in the parasite memory map that is convenient to the language processor designer. In the 65C102 Co-processor and 6502 Second Processor, for example:

Register	Address in Parasite memory map
R1STATUS	&FEF8
R1DATA	&FEF9
R2STATUS	&FEFA
R2DATA	&FEFB
R3STATUS	&FEFC
R3DATA	&FEFD
R4STATUS	&FEFE
R4DATA	&FEFF

Tube protocols

Host Protocols

The host protocols obtain or distribute data which the parasite has requested or transmitted. Normally it is the MOS which responds, as the majority of OSBYTE and OSWORD calls are concerned with accessing hardware or flow control parameters stored in RAM. However, when data has to be passed quickly or in bulk, this is usually done by filing systems working under NMIs. The user has

access to the same facilities as filing systems (via Register sets 3 and 4) and can load a program into the host which may take advantage of these.

The procedure has five phases:

1) Check that the Tube is present
2) Claim the Tube
3) Initiate the data transfer
4) Transfer data
5) Release the Tube

Check for presence of the Tube

As a file intended for a parasite may be loaded into the host when the Tube is not present, it is a good practice to check for the presence of the Tube by calling OSBYTE 234 (&EA) with X=0, Y=255. On return, X=0 if the Tube is not present, otherwise X=255.

Claiming the Tube

For the user to gain control of the Tube permission is requested by calling the MOS Tube entry point at &0406 with a unique “reason code” in the 65C12 accumulator. The reason code is a six-bit number logically ORed with &C0, thus setting the top two bits. For example, (in BBC BASIC assembler):

```
reason%      LDA #(&C0 OR <unique identifier>)
```
The Accumulator now holds the reason code

The <unique identifiers> already in use are listed above. For third party software writers, they are allocated by Acorn Customers Services to prevent clashes with other proprietary software.

When the call returns, the CPU carry bit will indicate if the call was successful or not:

C=1 : The call was successful
C=0 : The call failed

If the call failed, this is because some other program had control of the Tube. The call should be repeated until successful:

```
reason% JSR &0406               ;Call Tube code
        BCC reason%             ;Try to claim the Tube
                                ;try again if failed
        RTS
```

Registers (A, X, Y as appropriate) should be saved as they may be corrupted on return.

Initiating data transfer

Once the Tube has been successfully claimed, a control block must be set up in the host indicating the address of the first byte in the target area in the parasite. This in its turn is pointed at by loading the CPU's X and Y registers with the high byte and low byte respectively of the control block's address:

n+3	Target address high byte
n+2	Target address high byte-1
n+1	Target address low byte+1 A
n	Target address low byte (Y*&100 + X)

When the control block is set up, the same entry point (&0406) is used to initiate data transfer. Once again a reason code in the accumulator is used, this time to indicate what action is required:

Reason code	Description	Delay (a)	Delay (b)
0	Multiple byte transfer P➡H	24ms	24ms/byte
1	Multiple byte transfer H➡P	0	24ms/byte
These transfer any number of bytes. Terminate by releasing the Tube or starting another protocol.			
2	Multiple pairs of bytes P➡H	26ms	26ms/pair
3	Multiple pairs of bytes H➡P	0	26ms/pair
These transfer an even number of bytes and are faster than the above two protocols as they use R3DATA in its two byte mode. Terminate by releasing the Tube or starting another protocol.			
4	Execute		
Execution starts at the address pointed to by the control block (see below). This option has an implied release of the Tube and does not return to the user's program.			
5	Reserved		
This option is used in handling MOS calls which are passed across the Tube.			
6	256-byte transfer P➡H	19ms	10ms/byte
7	256-transfer H➡P	0	10ms/byte
These will transfer exactly 256 bytes. Only after completion can the Tube be released or another protocol started.			

Note that the reason codes and functions are the same as the parasite side R4DATA transfers.

Transferring data

After the instruction has been passed to the system, the user program can start the transfer after the delay specified above. In the P➞H direction the delay (a) allows the parasite CPU to service the initiating NMI and “prepare itself” before the data starts. In the H➞P direction this will already have been done as it would have been the parasite which issued the call asking the host to fetch the data.

Once transfer has started, the delay (b) must be allowed between bytes (or pairs of bytes as indicated above) to allow sufficient time for the parasite R3DATA NMI code to complete. In the P➞H direction, the host must service each byte (or pair) within the indicated time.

Releasing the Tube

When the transfer is complete, the Tube must be released so that another program can use it. The procedure is to call the MOS Tube entry point, again with a reason code in the accumulator, this time using:

```
release% LDA #(&80 OR <unique identifier>)
         JSR &0406
         RTS
```

Once again, CPU registers must be saved as appropriate.

Register Locations

The Tube registers have the following locations in host memory map:

Register	Location
R1STATUS	&FEE0
R1DATA	&FEE1
R2STATUS	&FEE2
R2DATA	&FEE3
R3STATUS	&FEE4
R3DATA	&FEE5
R4STATUS	&FEE6
R4DATA	&FEE7

In practice, R3DATA is the register of prime interest as this is the data channel for the transfers described above, that is:

A H➞P transfer :

```
LDA data-source
STA R3DATA
```

A P➞H transfer :
```
LDA R3DATA
STA safe-place
```

Tube/filing system interface

Part 1 of the Reference Manual describes in some detail the format of the filing system interface (OSFILE, OSARGS etc.). The following information is intended to assist in the writing of filing systems which must be compatible with the Tube.

LOAD/SAVE addresses

It is necessary to indicate to a filing system whether a file's target address is in the host or the parasite address space. This is done by treating the address as a four-byte (32-bit) number where the two most significant bytes indicate the relevant side of the Tube:

&FFFF<0 to FFFE>	indicates the host memory WARNING When the Tube is active , its communications code is in &FFFF0400 to &FFFF07FF
&FFFFFFFF	indicates that the named program is to be EXECed
&FFFE<3000 to 7FFF>	indicates the "shadow" screen memory in the host This does not apply to CFS, TFS and RFS.
&JKLM<0 to FFFF>	indicates the parasite memory

This means that parasites can have memory from &00000000 to &BFFFFFFF. For a program in the parasite to set up a utility program (say, an interrupt handler), it should do either of the following:

Using OSWORD

1) Transfer a small routine to disable interrupts, then modify the interrupt vector and re-enable interrupts.

Using *RUN

1) Issue a *RUN <utility name> FFFF<host target address>.

In this case, the utility will be loaded and JuMPed into at the entry point stored on the filing media (e.g. disc). The utility should then:-

2) Modify the relevant vector itself to point to the "real" entry point and then do an RTS to cause the parasite protocol to be terminated.

Use of the Non-Maskable Interrupt

To avoid slowing the computer down with polling loops, programs which have to interface at high speed with the real world use interrupts. The MOS provides and maintains a flexible and powerful IRQ-based “event” structure. Any program, be it in RAM, sideways ROM or sideways RAM can couple to this structure by purpose-designed OSBYTE calls and vector redirection.

The penalty for this flexibility is the time it takes to let all interested parties know that an IRQ has happened. Usually this is not important. However, where a filing system is reading floppy discs, for example, there is insufficient time to call a routine in the MOS and then let it tell all the other systems until it eventually reaches the filing system. For this reason the Non-Maskable Interrupt (NMI) is used for critical data transfers.

To ensure that the NMI is serviced quickly enough, the MOS exercises no control over it. Not even a vector is used as its redirection would take 2ms. To distribute this valuable resource, the MOS maintains an arbitration system to ensure that only one program at a time is trying to use the NMI.

Claiming NMI workspace

(&0D00 to &0D5F and &00A0 to &00A7)

Even if an IRQ and NMI are made to the CPU at exactly the same time, the NMI will take priority. The CPU will JMP via an address stored at a fixed location in ROM to the start of a region in RAM which is reserved for use as NMI Workspace. When the computer is reset, this location is loaded with an RTI so that spurious interrupts will not cause the computer to “crash”. For a program to make use of NMIs, it must put a short routine into memory from &0D00. This should:

a) Do the minimum to ensure integrity of the previous routine (i.e. saving registers on the stack).
b) Service the interrupt as efficiently as possible.
c) Return

It is important that programs do not try to use the NMI workspace before the MOS has given permission for this. Otherwise it could interfere with another program (such as a filing system) which was already using NMIs.

The NMI workspace and hence NMIs are claimed as follows:

a) Issue a service request to claim the NMI (OSBYTE 143 (&8F); X=&0C).
b) When the service request comes round, any NMI owner should “switch off” its NMI usage. NMIs will be allocated to another program. This call must not be claimed, but passed on to the next sideways program. On return, the Y register should be saved as it will contain the identity of the previous owner. This call should only be issued if the current owner is the Network software or

none at all. If it is issued whilst ADFS (or DFS) is active, data or even directories may be lost.

When the NMIs are no longer needed, they should be released thus:

a) Issue a service request to release the NMI (OSBYTE 143, X=&0B, Y=<previous owner>)

NMIs should be released by synchronous systems (such as the disc interfaces) when a given task is complete. It will then be claimed by an asynchronous system (such as the Network) until such time as it is needed again by a synchronous one.

Hardware access to the NMI

The following interfaces have a connection to the NMI signal:

1) Disc interface
2) Econet adapter
3) 1MHz Bus
4) The Cartridges
5) The Modem Cavity

The disc and net interfaces are not directly connected to the CPU NMI-pin for the following reasons:

The Disc Interface

The WD1770 series disc controllers have two interrupting outputs. One indicates that a new byte has to be read from/written to the disc; the other indicates that the last command has been completed. Both of these signals have active high totem pole outputs whilst the system uses an open collector, active low system. The two interrupts are logically open collector NORed in the I/O controller.

Note that some machines are fitted with the WD1772 in place of the WD1770.

The Network Adapter

This uses a 68B54 Advanced Data Link Controller and will generate an interrupt for every data byte assembled from the ECONET. As net traffic may be generated by other users, it is desirable to prevent the 68B54 from generating NMIs when the ANFS is not the NMI owner. As the 68B54 does not have an interrupt mask, logic, again in the I/O controller, performs this function.

Suggestions of uses for NMIs other than the disc and net interfaces are:

Infra-red data transfer cartridge, for example, fibre optic
Compact Disc filing systems (CD-ROMs)
Video Disc filing systems, for example, BBC Domesday Project
High speed modems

13 THE Z80 SECOND PROCESSOR

Operating system calls

The operating system calls of the host processor can be accessed from the Z80 in a similar manner to the BBC Microcomputer itself. Operating system calls can be made via a jump table starting at address FFCEh. The entry point for each routine corresponds with the equivalent address on the 6502, e.g. the WRCH routine is entered at FFEEh. All operating system calls (apart from OSARGS - see below) take parameters in Z80 registers A, H and L corresponding to A, Y and X on the 6502. For all calls that use the carry flag on the 6502 this still applies on the Z80. For example:

```
LD    A,41h     ;Character to be written in A
CALL  0FFEEh    ;Call OSWRCH to write character
```

and

```
LD    A,5       ;*FX 5,2 => A set to 5
LD    L,2       ;L set to 2
CALL  0FFF4h    ;Call OSBYTE routine equivalent to *FX 5,2
```

Interception of any operating system call can be achieved by simply changing the address field of the relevant jump to point to the required user routine.

The new memory map is shown below

Address (Hex)	Purpose	
FFFE	INT vector reserved for the Z80 operating system	
FFFC	Event vector	
FFFA	BRK vector	
FFF7	OSCLI	- H,L point to command line
FFF4	OSBYTE	- A = OSBYTE number H,L are parameters
FFF1	OSWORD	- A = OSWORD number H,L point to control block
FFEE	OSWRCH	- A = character
FFE7	OSNEWL	- Write linefeed, carriage return to screen
FFE3	OSASCI	- Write character in A to screen plus line feed if A = 0Dh

FFE0	OSRDCH	- A = character
FFDD	OSFILE	- A = Operation type H,L point to control block
FFDA	OSARGS	- A = Operation type E = Handle H,L point to control block
FFD7	OSBGET	- A = Byte H = File handle
FFD4	OSBPUT	- A = Byte H = File handle
FFD1	OSGBPB	- A = Operation type H,L point to control block
FFCE	OSFIND	- A = Operation type H,L point to filename (A=0) H = file handle
FFC8	TERM	- A=0 Switch off terminal mode (default) A=1 Switch on terminal mode A=FFh Test terminal mode
FF82	Fault pointer	
FF80	Escape flag	- top bit set if escape condition exists

Faults and events

6502 Faults

When a fault is generated by the 6502 host processor the Z80 is interrupted and the fault number and string are passed across the Tube and placed in a fault buffer. The pointer at FF82h is then set to point to the fault number and the Z80 operating system indirects through the BRK vector at FFFAh.

Z80 Faults

Faults can also be generated on the Z80 using the RST 38h instruction. All Z80-generated faults should adhere to the following convention:

```
RST  38h        Value FFh
Fault number    Fault string
Terminator      Value 00h
```

Events

When an event is detected by the 6502 operating system the event parameters A, Y and X are passed across the Tube to Z80 registers A, H and L respectively. The Z80 operating system then indirects through the event vector at FFFCh which is initialised to point to a Z80 RET instruction.

Escape processing

When the escape code is detected from the keyboard the top bit of the escape flag at FF80h is set. An escape condition should be detected by testing this bit and acknowledged by OSBYTE call 7Eh. The escape flag should be reset or set using OSBYTE calls 7Ch or 7Dh.

Interrupt handling

NMI Non-maskable interrupt

This interrupt is reserved for use by the Z80 operating system and cannot be intercepted by the user.

INT Interrupt request

When an INT is detected the Z80 operating system indirects through location FFFEh. All unrecognised interrupts are passed to a user INT routine at FFB0h in the jump table. The address field at FFB1h should be changed to point to the required user INT routine. This routine must preserve all registers and return using instructions:

EI	enable interrupts
RETI	return from maskable interrupt routine

Z80 Monitor

After turning on the Z80 and pressing BREAK the following display appears:-

```
Acorn TUBE Z80 64K n.nn

Acorn DFS

BASIC

*
```

where n.nn is the version number of the Z80 ROM. The * prompt indicates that the Z80 Monitor is running and at this stage all the standard * commands can be entered i.e *HELP, *FX4 etc. The Z80 Monitor will also recognise the following additional commands which allow memory to be examined, changed and small machine code programs to be entered directly and tested.

```
CPM
D <start address> <end address>
GO <address>
S <start address>
```

In these commands <address> refers to a hexadecimal address which can entered as 1 to 4 digits i.e. 3F can be entered as 3F, 03F or 003F. If more than 4 hex digits are entered the most significant digits will be truncated i.e. 12345 will be treated as 2345. If no address is specified the most recently specified address will be used instead. For all commands any leading spaces or asterisks and trailing spaces will be ignored.

CPM - allows the CP/M system to be loaded without resetting any previously entered * commands which would occur if CP/M was loaded using CTRL BREAK. i.e. typing

```
*KEY0 DIR|M
*KEY1 STAT *.*|M
*KEY2 ERA
*CPM
```

would allow the function keys to be defined before starting up CP/M (These key definitions would have been reset if CTRL BREAK had been used to load CP/M).

D (Dump) - gives a memory dump with character interpretation between the two specified addresses. At least one space is expected between the start and end addresses but no space is necessary before the first address. A dump can be terminated at any time by pressing ESCAPE.

GO - causes a jump to the specified address

S (Set) - allows memory to be entered and altered from the specified start address. No space is needed between the command and the address. The displayed memory location can be altered by entering valid hex digits which are shifted in from the right. The command can be terminated by entering any non hex character. To alter more than one location the UP and DOWN cursor keys can be used to increment or decrement the memory location.

Z80 OSWORD call

The Z80 provides an additional OSWORD call with A = 0FFh, to read or write blocks of I/O processor memory. On entry HL point to the following control block :-

HL + 0	Number of OSWORD parameters sent to I/O processor - 0Dh
HL + 1	Number of OSWORD parameters read from I/O processor - 01h
HL + 2	LSB of I/O processor address
HL + 3	•
HL + 4	•
HL + 5	MSB of I/O processor address
HL + 6	LSB of Z80 processor address
HL + 7	•
HL + 8	•
HL + 9	MSB of Z80 processor address
HL + A	LSB of number of bytes to read/write
HL + B	MSB of number of bytes to read/write
HL + C	Operation type - 0 to write to I/O processor 1 to read from I/O processor

The first two bytes are used by the Z80 operating system and must not be changed. If the I/O processor uses sixteen-bit addresses only the first two least significant bytes need to be specified.

For example, to read I/O processor screen memory (mode 0) into Z80 memory at 08000h

```
      LD      A,0FFh    ;OSWORD call 0FFh
      LD      HL,BLOCK  ;Set up HL to point to control block
      CALL    0FFF1h


BLOCK:DEFB    0Dh
      DEFB    01h
      DEFW    03000h    ;start of screen memory in I/O processor
      DEFW    0         ;set high word to zero
      DEFW    08000h    ;start of transfer address in Z80
      DEFW    0
      DEFW    05000h    ;size of screen memory (20K)
      DEFB    1         ;read operation
```

I/O Processor Memory Usage

The following areas of I/O processor memory are reserved and should not be corrupted by any user programs

2500h - 25FFh	Reserved for use by Z80 OS
2600h - 2FFFh	Reserved for use by CP/M
0070h - 0078h	Reserved for use by Z80 OS

Screen Control

There are three techniques that a CP/M application program can use to control the BBC Microcomputer's screen:

BBC Microcomputer Control Codes
Terminal Emulator Control Codes
GSX Functions

BBC Microcomputer Control Codes

All of the functions of the BBC Microcomputer normally accessed via the VDU command can be accessed through CP/M by sending an appropriate control code. These are explained in chapter 34 of the BBC Microcomputer User Guide and are summarised on page 378. For example, sending the sequence (as hexadecimal bytes) 1F 10 04 would position the cursor to cell x=16, y=4.

Terminal Emulator Control Codes

To allow existing CP/M applications to use basic terminal functions in a simple way, a terminal interface has been defined. This is by default disabled, but can be enabled, disabled or tested by assembler programs as follows:

To enable terminal mode

```
LD    A,1
CALL  0FFC8H
```

To disable terminal mode

```
LD    A,0
CALL  0FFC8H
```

To test terminal mode

```
LD    A,0FFH
CALL  FFC8H
```

In all cases, the state of the terminal mode prior to the call is returned in A:

A = 0 terminal mode was disabled
A = 1 terminal mode was enabled

An extra program is provided on the utilities disc: TERM.COM. This turns terminal mode on or off from the CCP.

TERM ON to enable terminal mode
TERM OFF to disable terminal mode

when the terminal mode is enabled, the following control codes and escape sequences can be used to control the screen. All numbers are hexadecimal.

07	Bleep
08	Move left one character
09	Move right one character
0A	Move down one line
0B	Move up one line
0C	Clear screen
0D	Carriage return
1B 3D YY XX	Position cursor to (XX-20, YY-20)
1B 3E ?…? 00	Send sequence of bytes X…X to the screen, where each byte X sent = ?-20
1B 3F	Clear to end of screen
1B 40	Clear to end of line
1E	Home cursor

Notes:
To send escape (1B) to the screen with the terminal emulator enabled, the sequence 1B 3E 3B 00 should be sent.

The clear to end of line and clear to end of screen functions are intended for 80-column screens of full dimensions, and will work in screen modes 0 and 3 only. They will reset the text window to the full screen.

GSX Functions

Refer to Digital Research Programmers' Manual

Character I/O under CP/M

Device assignments

The object of this implementation is to allow the user to redirect I/O either with the IOBYTE as on a normal CP/M system, or with OSBYTE calls as on a normal BBC machine. The following logical devices are present on a CP/M system.

CON: the user console
LST: the printer
RDR: the paper tape reader
PUN: the paper tape punch

These logical devices can be assigned to the following physical devices:

UC1: the user defined console device
CRT: the screen and keyboard
TTY: the RS423 serial lines
LPT: the printer
BAT: batch mode (input from RDR: and output to LST:)
PTR: paper tape reader - reassigned as the keyboard
PTP: paper tape punch - reassigned as the screen

CP/M also has the following physical devices which are all defined as null devices in this implementation:

UR1:
UR2:
UP1:
UP2:
UL1:

Null devices discard any output and return End-Of-File (1Ah) on input.

The default setting of IOBYTE has the following assignments:

CON: is UC1:
LST: is LPT:
RDR: is TTY:
PUN: is TTY:

The IOBYTE facility

The CP/M operating system allows the user to redirect the input and output of its logical devices to particular physical devices. As an example the CP/M system could be used with a remote terminal by assigning the physical device TTY: (the RS423 serial port) to the logical device CON: (the system console).

The use of the IOBYTE to reassign the physical devices is covered in the Digital Research CP/M Operating System Manual.

Care has been taken however to allow the user familiar with the BBC Micro to use OSBYTE calls to redirect input and output as required. This has been done by

providing the physical device UC1: which uses the normal BBC micro I/O streams. These can be altered as required. The default setting of the IOBYTE assigns the UC1: device to CON: so the system console behaves like a normal BBC micro.

The CP/M logical devices are as follows:-

CON: is the principal interactive console that communicates with the operator and is accessed through CP/M calls to the Console.

LST: is the principal listing device, usually a printer.

PUN: is the tape punching device - the name is a leftover from the days when computers used paper tape.

RDR: is the tape reading device. As with PUN: above it is inherited from the early versions of CP/M.

The Acorn CP/M system implements the following physical devices which are used in conjunction with the above logical devices:

UC1: is the normal BBC micro I/O channel. This allows the user familiar with the BBC to redirect I/O without using the CP/M IOBYTE. It also supports the terminal emulation facility described elsewhere.

CRT: provides direct access to the BBC screen and keyboard. Unlike the UC1: device, input and output cannot by redirected by OSBYTE calls. Input always comes from the keyboard and output always goes to the screen. It does not support the terminal emulation facility.

TTY: is the RS423 serial port. The default baud rate is 9600. It can be used for both input and output. Please note that the user should not disable the RS423 input if using the TTY: input device. The default setting is input enabled.

LPT: is the standard BBC micro printer device. This is a Centronics with no printer ignore character as a default but can be changed using OSBYTE calls. If a printer is not present then attempts to send characters to the printer will cause a message ‘Printer off line’ to appear. The user may then connect a printer and carry on. Alternatively if a printer is not available after a short time the message ‘SPACE starts Printer Sink’ appears and the user can press the SPACE bar to throw away the printer output. The printer sink is detailed in the BBC Micro Users Guide. Characters sent to the printer will continue to be ignored until the user selects another printer type with a *FX5 call. UC1: is the same as the LPT: device except no messages appear if the printer isn’t connected. The system will simply stop.

NUL: is a device which throws away all output and returns 1Ah on input, indicating End of File. This is present to prevent the system hanging if an unimplemented physical device is selected.

UR1:,UR2:,
UP1:,UP2: are all equivalent to the NUL: device.

The default value for the IOBYTE in the Acorn CP/M system is 83h. This assigns UC1: to CON:, LPT: to LST:, TTY: to RDR:, and TTY: to PUN:. It can be changed by applications programs or by the STAT command.

Device characteristics

The physical devices have the following characteristics:

UC1: the user-defined console device.

Default console device. It is also the fastest of the console devices since it uses the standard BBC input and output streams. These streams can be altered by using OSBYTE calls in the normal BBC manner, so the machine's I/O can be treated as that of a normal BBC machine, using the STAR command to avoid using the IOBYTE facility.

CRT: the screen and keyboard.

Input is taken from the keyboard and output is sent to the screen. The characteristics of this device cannot be changed with OSBYTE calls.

TTY: the RS423 serial lines.

The RS423 serial lines can be accessed using this device. The default baud rate setting for the TTY: is 9600 baud and may be changed by the appropriate OSBYTE call. No events are generated by ESCAPE characters, and non-ASCII codes cannot be programmed on the function keys. The normal BBC handshaking using CTS/RTS is implemented. The cassette driver should not be enabled nor the RS432 input disabled while using this device. Although a serial printer can be driven by this device, it bypasses the normal printer functions, such as setting an ignore character or handshaking . It is therefore better to use the LPT: device and set it to a serial printer with the appropriate OSBYTE call.

LPT: the printer.

Standard BBC printer driver. It can be changed to suit the particular printer in use. The default setting is the parallel printer, ignoring line feeds. It is recommended that when connecting a serial printer the IOBYTE be left unchanged and the LPT: device altered with the appropriate OSBYTE call. The alternative is to use the IOBYTE to select the TTY: driver but this does not carry out any of the standard printer functions.

BAT: batch mode (input from RDR: and output to LST:).

This device takes its input from the logical device RDR: This can in turn be assigned to any of the relevant physical devices. The output goes to the logical device LST: which can in turn be reassigned to the relevant physical device.

PTR: paper tape reader - reassigned as the keyboard.

In the absence of a paper tape reader this device is the keyboard.

PTP: paper tape punch - reassigned as the screen.

In the absence of a paper tape punch this device is the screen.

The System Patch Area

To allow temporary patches to be made to the Acorn CP/M system an area has been reserved in the BIOS. It starts immediately after the STARTUP entry point in the CP/M BIOS jump table and is 60h bytes long. This is EA33h to EA92h inclusive in the current Acorn CP/M system.

Patching should only be attempted by those familiar with the CP/M system.

There are two main types of patch. The first is to add special initialisation code. The instruction at EA33h is a RET. This location is called at cold start which allows a special subroutine to be inserted in place of the ROM. This could for example select a serial printer as default. The other use is to patch in temporary additions to the system. Certain application programs do so. Please note that patches of this sort may be overwritten by other programs. As a result they can only form temporary additions to the system and they should ‘tidy up’ on termination. i.e. any changes made to other parts of the operating system should be reversed after the patch has done its job.

14 THE 80186 SECOND PROCESSOR

Operating System Calls

The operating system calls of the Master 128 may be accessed from the 80186 coprocessor by using the 80186 software interrupts. 256 software interrupts are supported and each one has a corresponding four-byte vector in the first kilobyte of memory. Interrupts 040h - 04Ch are reserved for the thirteen MOS calls supported on the co-processor. All operating system calls take parameters in 80186 registers al,bh,bl corresponding to 6502 registers A,Y and X (except OSARGS - see below).

Address	Interrupt	Routine	Function
0100h	040h	OSFIND	Open or close a file
0104h	041h	OSGBPB	Read/Write part of a file
0108h	042h	OSBPUT	Write single byte to file
010Ch	043h	OSBGET	Read single byte from file
0110h	044h	OSARGS	Read/Write file data
0114h	045h	OSFILE	Read/Write a complete file
0118h	046h	OSRDCH	Read character from keyboard
011Ch	047h	OSASCI	Write character (plus LF)
0120h	048h	OSNEWL	Write CR,LF to screen
0124h	049h	OSWRCH	Write character to screen
0128h	04Ah	OSWORD	Various using control block
012Ch	04Bh	OSBYTE	Various using registers
0130h	04Ch	OSCLI	Interpret command line

MOS calls OSRDSC, OSWRSC, OSEVEN, GSINIT and GSREAD are not supported by the 80186 but OSWORD with al=0FAh provides the functions of OSRDSC and OSWRSC.

OSFIND

Opens a file for reading or writing

Entry parameters	al	operation type
	ds:bx	point to filename terminated by CR (al <> 0)
	ah	file handle (al = 0)
Exit parameters	al	file handle (0 = file could not be opened)
Flag status	undefined	
Preserved registers	ds, bx	

OSGBPB

Read/write block of bytes from/to specified open file

Entry parameters	al	operation type
	ds:bx	point to control block
Exit parameters	al = 0	operation attempted
	al	unchanged = not supported in this fs
Flag status	cf	clear = transfer completed ok
	cf	set = end of file reached before transfer completed
Preserved registers	ds, bx	

OSBPUT

Write single byte to specified open file

Entry parameters	al	byte to write to file
	bh	file handle
Exit parameters	none	
Flag status	undefined	
Preserved registers	all	

OSBGET

Read single byte from specified open file

Entry parameters	bh	file handle
Exit parameters	al	byte read from file
Flags status	cf	set if attempt made to read past end of file
Preserved registers	bx	

OSARGS

Read/write file attributes

Entry parameters	al	operation type
	ah	file handle
	ds:bx	points to 4-byte attribute block
Exit parameters	al	filing system number
	bx	points to 4-byte attribute block
Flags status	undefined	
Preserved registers	ds, bx	

OSFILE

Read/Write complete file or catalogue information

Entry parameters	al	operation type
	ds:bx	point to control block
Exit parameters	al or (bx)	dependent on operation
Flags status	undefined	
Preserved registers	all	

OSRDCH

Read a character from currently selected input stream

Entry parameters	none	
Exit parameters	al	character
Flags status	cf	clear = valid character read
		set = error condition, type in al
Preserved registers	ah, bx	

OSASCI

IF character <> CR do OSWRCH ELSE do OSNEWL

Entry parameters	al	character
Exit parameters	none	
Flags status	undefined	
Preserved registers	ah, bx	

OSNEWL

Write LF,CR to currently selected output stream

Entry parameters	none
Exit parameters	none
Flags status	undefined
Preserved registers	bx

OSWRCH

Write character to currently selected output stream

Entry parameters	al	character to write
Exit parameters	none	
Flags status	undefined	
Preserved registers	bx	

OSWORD

Various functions using control block

Entry parameters	al	OSWORD type
	ds:bx	points to control block
Exit parameters	(bx)	parameters returned in control block are call dependent
Flags status	undefined	
Preserved registers	all	

OSBYTE

Various functions using registers

Entry parameters	al	OSBYTE type
	bl	OSBYTE parameter
	bh	OSBYTE parameter (only if al > 07Fh)
Exit parameters	bl	return parameter
	bh	return parameter (only if al > 07Fh)
Flags status	cf	status call dependent
Preserved registers	al	

OSCLI

Send command to Command Line Interpreter

Entry parameters	ds:bx	point to command line
Exit parameters	none	
Flags status	undefined	
Preserved registers	all	

Error Handling by the 80186 Monitor

When an error is generated by the 65C12 host processor the error number and string are passed across the TUBE to the 80186 under interrupt. The error number and string are then placed in an error buffer on the 80186 and a pointer is initialised to point to the error number. The error string is terminated by a null byte (00h). The 80186 TUBE code then jumps to the error handler, prints out the error and returns control to the 80186 monitor (see below).

The locations of the error handler vector and error pointer are given below:

0000:05F4h error pointer - offset
0000:05F6h error pointer - segment

0000:05F8h error handler vector - offset
0000:05FAh error handler vector - segment

Error Handling by stand-alone languages or applications

The error handling provided by the 80186 monitor is not suitable for stand-alone languages (i.e. languages using only MOS functions and host filing systems - not DOS+ or Concurrent DOS) as control is returned to the monitor by the default error handler. When the language is started up it should initialise the error handler vector to point to its own error handler which can handle the error in an appropriate way and return control to a suitable point within the language.

An example is now given to illustrate a typical error handler. This assumes that the language is running at 0000:8000h. The example is written in Digital Research RASM86 assembler format.

```
cseg    0
org     08000h

osnewl  equ   048h
oswrch  equ   049h

error pointer offset   equ    .05F4h
error pointer segment  equ    .05F6h

error handler vec offset      equ     .05F8h
error handler vec segment     equ     .05FAh
```

```
-----------------------------

;initialise error handler vector to point to my error handler

      sub    ax,ax
      mov    ds,ax             ;set ds=0 to access system data

      mov    ax,offset my error handler
      mov    error handler vec offset,ax
      mov    ax,seg my error handler
      mov    error handler vec segment,ax

-----------------------------

my error handler:
;set up ds:si to point to error
      lds     si,dword ptr error pointer offset
      int     osnewl          ;new line
      inc     si              ;skip error number
      cld                     ;set forward direction
my error loop:
      lodsb                   ;get error string from buffer
      int     oswrch          ;and write it out
      test    al,al           ;end of string ?
      jnz     my error loop   ;no - get next character
      jmp     my command loop ;yes - jump to command loop
```

80186 Error Messages

Errors can also be generated by the 80186 using interrupt 04Fh and following it with the error number and error string terminated with a null byte. The error pointer will be initialised as for 65C12 errors and the error handler given by the error handler vector will be used.

The following example illustrates the use of 80186 errors. In this a test is being made for the presence of a file before attempting to load it. The file name is assumed to be in the current data segment.

```
;some interrupt numbers

error   equ    04Fh    ;the error interrupt number
osfind  equ    040h
```

```
;osfind parameters

open for input  equu    040h


;error numbers

not found error equu    06Dh

;some misc equates

cr     equ    13

--------------------------------

cseg

look for file:
      mov     al,open for input
      mov     bx,offset my file name
      int     osfind
      or      al,al          ;al=0 implies file not found
      jnz     load the file

;file not present - give error message

      int     error
      db      not found error, cannot find file, 0

;note no return after writing out error

;file loaded here if present

load the file:

--------------------------------

dseg

my file name:
      db      '$.myfile',cr

--------------------------------
```

Escape Processing

When an escape condition is detected by the 65C12 the top bit of the escape flag at 0000:05F2h on the 80186 is set under interrupt. An escape condition should be tested for by checking this escape flag. If an escape condition exists the escape must be acknowledged using OSBYTE with al=07Eh and optionally an 80186 error message can be generated. The escape flag should not be set or reset directly as the change will not be reflected on the host processor side. OSBYTE calls with al = 07Ch or 07Dh should be used to clear or set the escape condition respectively.

80186 Monitor

After enabling the co-processor and pressing BREAK the following display should appear:

```
Acorn TUBE 80186 512K

Acorn ADFS

BASIC

*
```

The * prompt indicates that the 80186 monitor has been entered and is waiting for commands to send to the Command Line Interpreter on the 80186 or the 65C12.

In addition to the standard MOS and filing system commands the 80186 recognises the following monitor commands:

name	function
D	memory dump in hex and ASCII
DOS	boot dos from hard disc if present else boot from floppy
F	fill memory with byte or word constant
GO	jump to specified address
MON	enter Monitor
S	set memory with hex or ASCII
SR	search memory for specified text string
TFER	transfer blocks of memory between 80186 and 65C12

Where <offset> is used below it refers to a hexadecimal offset address which can be entered as 1 to 4 digits - leading zeros can be omitted i.e. 7A can be entered as 7A , 07A or 007A. If more than 4 hex digits are entered the most significant digits will be truncated i.e. 12345 will be treated as 2345. Where is

used it refers to a 80186 segment address which can also be entered as 1 to 4 hex digits but must be followed immediately by a colon (:) to indicate that it is a segment value. In all relevant commands below if no segment address is specified the most recently specified value is used or 0 if no previous value has been specified. For all commands any leading spaces or asterisks or trailing spaces will be ignored. Items enclosed in <> brackets indicate parameters that the command uses, items also enclosed in () brackets indicate optional parameters that do not have to be specified. All commands can be entered in upper or lower case (or both).

D - memory dump

Syntax - *D (:) (<start offset>) (<end offset>)
Gives a memory dump between the specified addresses in hex and ASCII showing the addresses in segment:offset form. Characters outside the ASCII range 20h - 7Eh are shown as a full stop. All the parameters in the above commands are optional. If the segment address is omitted the last value will be used. If the start and end offsets are omitted the last end address + 010h is used as the start address and the last end address + 080h is used as the end address. If the end address is omitted the start address + 080h is used. The dump operation can be terminated at any time by pressing ESCAPE.

DOS - boot DOS

Syntax - *DOS
Allows DOS to be booted without CTRL BREAK i.e. from stand-alone languages or applications. DOS will be booted from the hard disc if present (and correctly initialised for use with DOS) else it will be booted from floppy.

F - fill memory with constant

Syntax - *F (:) <start offset> <end offset> <fill byte/word>
Fills memory with a constant value between the specified addresses. The constant used can be specified as a byte or word value. The end offset specified is the end address + 1 used by the fill command i.e.

```
*f 1000 1010 55
```

will fill bytes 1000h - 100Fh inclusive with the value 55h

```
*f 1000 1010 1234
```

will fill bytes 1000h - 100Fh inclusive with the word 1234h with the lsb written first. An end offset of 0 can be used to specify a fill operation to the last address in the specified segment

GO - jump to address

Syntax - *GO (:) <offset>
Transfers control to the specified address. Should be used with care as jumping to an address which does not contain any executable code could have undesirable consequences!

MON - enter 80186 monitor

Syntax - *MON
Allows the monitor to be re-entered from stand-alone languages or applications without pressing BREAK.

S - set memory contents

Syntax - *S (:) <start offset>
Allows memory contents to be examined and altered if required. A line of sixteen bytes of memory is displayed in hexadecimal and ASCII formats, initially with the cursor under the least significant digit of the first byte specified. Cursor movement and data entry is controlled using the following keys:

LEFT	move cursor left, if at far left display previous 16 bytes
RIGHT	move cursor right, if at far right display next 16 bytes
UP	display previous 16 bytes
DOWN	display next 16 bytes
SHIFT LEFT	move cursor to far left of current field
SHIFT RIGHT	move cursor to far right of current field
COPY	toggle between hex field and ASCII field

The display consists of two 16-byte fields - a hexadecimal display and an ASCII display. The COPY key is used to switch between the two. While the cursor is in the hex field, data is entered in hex digits, each digit being shifted in from the right. To advance to the next field the normal cursor keys are used. SHIFTed cursor keys are used to move to the far left or right of the current field. If the cursor is in the ASCII field, data is entered as ASCII bytes. The cursor is automatically advanced to the next field to allow text to be typed in directly. When text is entered at the far right of the field the next 16 bytes are automatically displayed to allow typing to continue over 16-byte boundaries.

The *S command is terminated by pressing ESCAPE

SR - search memory for string

Syntax - *SR (:) <start offset> <end offset> <“string”>
Search memory for specified text string reporting all occurrences in segment:offset form. The address given is of the first byte of the matching string. The search string must be enclosed in double quotes (") and can be up to 72 characters in length. (Maximum length for complete command line is 80 characters). The end offset specified is the end address + 1 of the search area so to allow the search to continue right up to the end of a segment. An end address of 0 can be specified i.e.

```
*sr 4000 0 "eric"
```

will search from 04000h up to 0FFFFh inclusive. The condition for a string to be found is that it must be completely contained within the search area, i.e. if string “eric” lives at 03FFDh then

```
*sr 0 4000 "eric"
```

will not report it but if the string “eric” lives at 03FFCh then the above search will find it. Any 8-bit character string can be sought using escape sequences to allow control codes and characters above 07Fh to be specified. (N.B. these are compatible with the MOS escape sequences). The | character is used to denote an escape sequence. The following table shows how all the characters are specified.

String	hex byte
“|@”	0
“|a” or “|A”	1
to	to
“|z” or “|Z”	1A
“|[”	1B
to	to
“|_”	1F
“ ”	20
to	to
“~”	7E

except for following two special cases

“ "" ”	22
“||”	7C
“|?”	7F
“|!<char>”	80-FF

where <char> is any of above 7 bit chars

Any escape arguments not recognised are reduced to the argument alone i.e. “|1” is reduced to “1” etc.

and any surplus redundant “|!” operators are ignored

i.e “|!!!|@” is reduced to “|!|@”

Any string not terminated by a " character or containing an odd number of " characters will be reported as bad strings i.e.

String	Error
"abc	No terminating "
"ab""	Single quote character in string
"ab""c"	As above

A Bad String error will also be generated if no argument is supplied for the escape character | or if a null string or equivalent is specified i.e.

String	Error
"a|"	No escape argument
""	Null string
"|!"	Reduced to null string hence as above

The search operation can be terminated at any time by pressing ESCAPE.

TFER - transfer blocks of memory between 80186 and 65C12

Syntax - *TFER <I/O addr> (:) <offset> <length> <r/w>
Allows fast block transfer of memory between 80186 and 65C12. The direction of transfer is specified by the final parameter which must be r or w: r indicates a read from 65C12 memory to 80186 memory, w indicates a write to 65C12 memory from 80186 memory. The transfer is implemented using OSWORD 0FAh (described below) and is optimised to use fast transfer types 6 and 7 (10ms/byte) where possible. If the transfer length is not a multiple of 256 bytes any remaining bytes are transferred using types 0 and 1 (24ms/byte).

80186 OSWORD call

The 80186 ROM implements an additional OSWORD call with al = 0FAh to allow efficient transfer of blocks of data between the 80186 and the host 65C12 processor. The OSWORD call is used by setting up the following control block which must be pointed to by ds:bx.

bx + 0	Number of parameters sent to I/O processor (0Dh or 0Eh)
bx + 1	Number of parameters read from I/O processor (01h)
bx + 2	LSB of I/O processor address
bx + 3	•
bx + 4	•
bx + 5	MSB of I/O processor address
bx + 6	LSB of 80186 offset address
bx + 7	MSB of 80186 offset address
bx + 8	LSB of 80186 segment address
bx + 9	MSB of 80186 segment address
bx + A	LSB of length of transfer
bx + B	MSB of length of transfer
bx + C	Operation type
bx + D	65C12 memory access control

The operation type specifies the type of transfer used as follows:

0	write to 65C12 (24 ms/byte)
1	read from 65C12 (24 ms/byte)
2	write to 65C12 (26 ms/pair of bytes)
3	read from 65C12 (26 ms/pair of bytes)
6	write to 65C12 (10 ms/byte 256 transfer)
7	read from 65C12 (10 ms/byte 256 transfer)

The memory access control byte allows access to the paged ROMs, paged RAM and shadow RAM in the host machine and is laid out as follows:

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	x	sm	m/s	c	pr3	pr2	pr1	pr0

where the bits have the following functions:

x	unused	
sm	if 3000h <= I/O address < 8000h, sm=1 = use screen memory regardless of state of *shadow (NB - this overrides bit 5 if a conflict arises)	
m/s	0 = use main screen memory if screen address specified 1 = use shadow screen memory if screen address specified	
c	if 8000h <= I/O address < 0000h	0 = use specified ROM number 1 = use currently selected ROM
pr3 - pr0	paged ROM number 0-15	

The memory access byte is only used if the first byte of the control block is set to 0Eh - it is ignored otherwise. Use of the memory access byte allows paged ROM software to be copied and therefore access may be restricted to system use. This,

however, would prevent access to the shadow RAM which is not used by the system and cannot be legally accessed by other means.

The following example of the call assumes that the control block has been set up correctly and is located in the first 64K segment.

A contiguous 36-Kbyte area of memory is being used as a buffer for data written from 2000:1000 in the 80186. The host buffer starts at 3000h and extends to BFFFh. 3000h - 7FFFh is specified as shadow screen memory and 8000h - BFFFh is specified as paged RAM in socket #5.

```
osword          equ     04Ah
transfer osword equ     0FAh

-------------------------------------------------------------

       sub    ax,ax                    ;point ds:bx at control
                                       ;block
       mov    ds,ax
       mov    bx,offset transfer block
       mov    al,transfer osword       ;set up osword type
       int    osword                   ;do the call

-------------------------------------------------------------

transfer block  db     0Eh
                db     01h
                dw     03000h,0         ;base address in 65C12
                dw     01000h,02000h    ;base address in 80186
                dw     09000h           ;length = 36k
                db     6                ;fast 256 byte transfer
                db     025h             ;use shadow, paged ram
```

15 DISC FILING SYSTEMS

Introduction

The Reference Manual, Part 1 gives much detailed information on the two disc filing systems within the system ROM. Extra information is provided here for those wishing to use their own filing systems, or needing a specification of ANFS.

The track format is described by detailing the data to be written to format a disc :

DFS

Description	Value to be written	Number of bytes	Comments
Post index gap (Gap 1)	**FF**	40	
	The bracketed field is repeated 16 times		
Sector ID field	**00**	6	Pll lock-up
	F5	3	(Note 1) 1
	FE	1	(Note 2) 1
	00 to 27 (40 track) OR **00 to 4F (80 track)**	1	track ID
	00 (on side 0) OR **01 (on side 1)**	1	head ID
	00 to 0F	1	sector number
	01	1	
	F7	1	(Note 3)
Sector ID/Data gap (Gap 2)	**4E**	10	
	00	4	Pll lock-up
	F5	3	(Note 1)
Data field	**FB/F8**	1	(Note 4)
	5A	256	Data
	F7	1	(Note 3)
Post data Gap 3	**FF**	10	
Runout (Gap 4)	**FF**	Until next index hole	

The first sector on every track is offset by 7 from the previous sector.

DFS supports both 40- and 80-track drives. It manages the sides of a disc separately. The WD1770 FDC is operated in its single density mode (FM) using ten 256-byte sectors per track with a data transfer rate of 1 byte every 32ms.

ADFS

Description	Value to be written	Number of bytes	Comments
Post index gap (Gap 1)	**4E**	60	
	The bracketed field is repeated 16 times		
Sector ID field	**00**	12	Pll lock-up
	F5	3	(Note 1) 1
	FE	1	(Note 2) 1
	00 to 27 (40 track)		
	OR	1	track ID
	00 to 4F (80 track)		
	00 (on side 0)		
	OR	1	head ID
	01 (on side 1)		
	00 to 0F	1	sector number
	01	1	
	F7	1	(Note 3)
Sector ID/Data gap (Gap 2)	**4E**	22	
	00	12	Pll lock-up
	F5	3	(Note 1)
Data field	**FB**	1	(Note 4)
	5A	256	Data
	F7	1	(Note 3)
Post data Gap 3	**4E**	43	
Runout (Gap 4)	**4E**	Until next index hole	

The first sector on every track is offset by 7 from the previous sector.

ADFS supports both 40- and 80-track drives. It manages both sides of a disc as a single volume. The WD1770 FDC is operated in its double density mode (MFM) using sixteen 256-byte sectors per track with a data transfer rate of 1 byte every 32ms.

Notes:
1) This causes three synchronisation bytes to be recorded.
2) This causes an ID data marker to be recorded.
3) This causes 2 CRC bytes to be recorded.
4) This causes a data marker to be recorded.

CP/M Disc Format

Acorn CP/M uses the following double-sided disc format:-

80 tracks / surface
10 sectors / track
256 bytes / sector

A double sided disc is regarded by CP/M as a single logical disc with 160 tracks numbered from 0 to 159. In order to obtain the best disc performance the following logical to physical track mapping is performed.

Logical CP/M track	**Physical track**
0 - 79	0 - 79 (top surface)
80 - 159	79 - 0 (bottom surface)

The first 3 tracks on the top surface are reserved for the CP/M system.

The CP/M directory starts at track 3 sector 0 and uses 4Kbytes to allow up to 128 directory entries per disc. This leaves 388 Kbytes per disc available for user programs and data.

Acorn CP/M uses deblocking to allow the physical disc sector size to be larger than the logical CP/M record size of 128 bytes. Although a 256-byte sector size is used the effective sector size is 512 bytes as all disc operations read or write 2 sectors at a time using an appropriate sector skew. The following table defines the logical record to physical sector relationship:

Logical CP/M record (128 bytes)	**Logical disc sector (512 bytes)**	**Physical disc sector (256 bytes)**	**Logical CP/M record (128 bytes)**	**Logical disc sector (512 bytes)**	**Physical disc sector (256 bytes)**
0	0	0	10	2	9
1	0	0	11	2	9
2	0	1	12	3	2
3	0	1	13	3	2
4	1	4	14	3	3
5	1	4	15	3	3
6	1	5	16	4	6
7	1	5	17	4	6
8	2	8	18	4	7
9	2	8	19	4	7

16 ADVANCED NETWORK FILING SYSTEM

This chapter covers the specification for the Master 128 and Econet Terminal implementations of ANFS. The specification includes differences between ANFS, for Master microcomputers and earlier BBC Model B machines. The computer is type five for machine peek operations.

Local buffering

ANFS will buffer data from all open files in RAM. This buffer is in the I/O processor in both single processor and second processor installations. This means that of all OSBGET and OSBPUT calls, only a small proportion (2%) will actually need to communicate over the network with the file server. The buffering code uses 256-byte buffers on a “dynamic” basis. There may be more than one buffer allocated per file channel. Up to sixteen channels may be active, which means that higher performance file servers can be implemented without any changes to the ANFS. The limit of sixteen also permits the user to maintain valid context and open files on more than one file server at once. Older file servers and the original NFS for use on earlier BBC Microcomputers can accommodate only eight channels.

When opening files, buffers are allocated dynamically. The number of buffers allocated is variable. If there were sixteen buffers and only a single file were opened, the first sixteen pages of the file would be read sequentially into the buffers. If these pages are then referenced, they are instantly available. When the seventeenth page is requested, the page that was referred to least recently is overwritten. This causes ANFS and the buffers to function as a cache. If two files are open then the buffers are shared according to the amount each file is used. If one file is used twice as much as the other then it is allocated about twice as many buffers.

The number of buffers is set to five initially. If there is space available then up to sixteen buffers can be allocated and this is very likely in most circumstances. The algorithm for marking buffers “Least Recently Used” (LRU) is as follows.

a) If the buffer has nothing in it then it will become LRU

b) If there are no empty buffers then the buffer adjacent to the buffer being accessed will be LRU if it is full. (This prevents the “leading edge” of a file being overwritten).

c) If multiple files are open and if one file is being heavily used, then it may have extra buffers allocated to it.

The translation of user handles to file server handles is more complex than in NFS, so the OSWORD call to set the context handles works in a slightly different way. If the user handle is not open or is not a directory then no change will occur to the handle being written. When this happens, rather than issuing a channel error, the reason code in the OSWORD block will be changed to zero to indicate failure. An OSARGS call is available to return the file server handle (and number) for a given user handle.

Operating System Commands

*HELP

The *HELP command shows the current version number and the station number identical to that displayed after BREAK. There are now two subcategories, “Net” for those commands which are part of the network filing system, and “Utils” for filing system independent utilities and commands.

There is a facility for large help texts (provided by users) to be accessed by the *HELP command. If the first argument given to the *HELP command is the word “on” then the remainder of the line will be used as a series of filenames. These filenames will be used to access Help texts stored on the file server. An example would be:

>*HELP on users and stations

This would attempt to *TYPE the files “users”, “and”, and “stations” on the screen. Obviously there should not be a file called “and”, but the files “stations” and “users” could contain useful information relevant to the user’s own installation. Provision of these help texts would be made by the network manager.

*CDIR <dir> (<number>)

Where the <number> is quoted in entries, a directory big enough to hold that many entries will be created. The <number> may range from 1 to 245. Any number outside this range will give an error. If the optional parameter is not given then the default of 19 will be used. This creates a directory of length &200.

*FLIP

The FLIP command will simply exchange the currently selected directory (CSD) and the currently selected library (CSL). This is a way of selecting the library as your

CSD and it is particularly useful when files which must be LOADed (via the OSFILE mechanism) are to be made public, and software must be able to access them easily. It is unwise to use the *DIR or *LIB commands in the “Flipped” state: the user should FLIP back first.

*FS (<station id.>)

The FS command will change the file server number. This allows a user to be logged on to two or more file servers at one time and to change between them. Any open files will be ensured to the current file server before the number is changed. If no argument is given, i.e. just *FS [RETURN] is typed, then the current file server number will be reported.

*I AM (<station id.>) <user id.> ((:[RETURN])<password>)

This command is essentially unchanged from NFS but now accepts [DELETE] and CTRL-U during the “invisible” part and will delete either the last character typed or the entire “invisible” part respectively. It is possible for this command to display the warning message “Data Lost” if data which had been written to a file was still buffered and was not able to be written to the file server. This could be caused by the file server having been restarted.

*LCAT (<dir>)

Catalogue the current library. This can also take an optional argument for a path from the currently selected library, e.g. *LCAT fonts [RETURN] will catalogue a directory called “fonts” in the currently selected library (CSL). This is useful because a command such as *FONTS.italics is looked up in the currently selected library.

*LEX (<dir>)

Examine the current library. This can take an optional argument, see *LCAT.

*PASS (:[RETURN]) <old password> <new password>

This accepts a “.” in the middle of the command like “*I AM ”, so that passwords may be hidden. It should be noted that although the “.” may appear anywhere in the line it would be most useful to have it before both the old and new passwords. Whilst the “invisible” part is being typed CTRL-U deletes the entire invisible part and [DELETE] deletes the last character.

*WIPE (<dir>)

*WIPE will offer each unlocked file or directory in the specified directory for deletion with a (Y/N/?) prompt. If “?” is typed the full object information will be printed followed by (Y/N). If “Y” is typed then the file or directory will be deleted. If anything else is typed the object will not be deleted.

Extra Utils star commands incorporated in the ROM

These commands may be issued when the ANFS is not the current filing system.

*POLLPS (<station id.>(,<ps type>))

The POLLPS command shows the currently selected printer server number and the currently selected printer type, for example,

```
>*POLLPS
Printer server is 235 "PRINT"
   235 is ready
```

Following the printer server number will be a list of all the printers on the network and the current status of each. The possible status conditions are :
“ready” which means the printer is ready for use or has timed out the current user
“busy” which means the station shown is currently using that printer
“jammed” which means that the printer server has characters in its buffer but has been unable to send characters out to its printer for some period of time. This would usually be caused by the printer being off-line or unable to accept characters for some reason.

It should be noted that *POLLPS does not alter the printer server number, the state of FX5, or the state of FX6.

If the command is followed by a station number then only the status of that station will be shown. No status will be shown if the station is not operating as a printer server. The command can also have a textual argument e.g. “DAISY”, “LINE”, “DRAFT”, or “LASER”. If this textual argument is supplied then only printers of that type will have their status listed. This allows users to examine only the status of printers in which they are interested.

*PROT (<prot type>) …

With no argument supplied this will protect against all operations. If any arguments are given then only those types of operations will be protected against. Note that this can have multiple arguments, e.g. *PROT PEEK POKE [RETURN].

*UNPROT (<prot type>) …

With no argument supplied this will unprotect all operations. If any arguments are given then only those types of operations will be unprotected. Note that this can have multiple arguments e.g. *UNPROT POKE JSR [RETURN].

*PS (<station id.> <ps type>)

The command *PS followed by a number behaves in the same way as the loaded transient command PS in NFS (and File servers), that is it sets the printer server station number to the one supplied. If a textual argument is given then the printer server will be set to the number of the first printer of that type to be found “ready”. If no printer is found to be ready then the printer server number will remain unchanged. If no argument at all is supplied then the printer server number will be set to the first “ready” printer. When a textual argument is supplied it will become the currently selected printer type, and this type will be used when *PS issued with no argument.

Note the power-up default type is “PRINT” to which all printer servers respond. The command *POLLPS with no argument also uses this currently selected type. If printing is taking place when the PS command is issued then the error message “Printer busy” will be issued and the printer server number will not change. Again it should be noted that this command does not affect the states of FX5 and FX6.

*WDUMP <filename> (<offset> (<address>))

This dumps the file in hex and ASCII in a format suitable for screen widths of 80 characters. The optional offset parameter is the number of bytes (in hex) to skip before starting to dump. The address parameter is the address (in hex) of the first byte in the file to be displayed. The default value for this is the file’s load address.

*CONFIGURE commands

The following commands are used as arguments to the *CONFIGURE command:

FS <station id.>

This sets the file server station number stored in CMOS RAM. This station number is used as the default at a hard BREAK.

PS <station id.>

This sets the printer server station number stored in CMOS RAM. This station number is used as the default at a hard BREAK.

SPACE / NOSPACE

This feature is enabled by “SPACE” and disabled by “NOSPACE”. Its purpose is to provide compatibility for those users who have networks consisting of BBC machines and Master Series machines. This is achieved by ensuring that “PAGE” in the I/O processor is at least &1000. This is required for transparent use of some network commands, e.g. *VIEW, *NOTIFY, etc.

*STATUS commands

The following commands are used as arguments to the *STATUS command:

FS

This displays the file server station number as stored in CMOS RAM. This station number is used as the default at a hard BREAK.

PS

This will display the printer server station number as stored in CMOS RAM. This station number is used as the default at a hard BREAK.

SPACE

This will display the state of this feature, either “No Space” or “Space”.

Extra *OPT commands

The setting of *OPT5 controls a level of bootstrapping only available on Master Series. This bootstrap is the *RUNning of a file called “FindLib”. If this utility is run it can be used to select a Master Series compatible library.

To enable operation with old software which has used the network workspace, there is a switch which controls the location of network private space. Normally this space is in pages &B and &C in the I/O processor. By setting *OPT6,1 then this space will be “claimed ” in the normal way. Note that this will increase the value of PAGE by &200. Setting *OPT6,0 will restore this claiming area to its correct location of &B and &C.

Both these commands are stored in CMOS RAM and must only be issued when ANFS is the current filing system.

Printing

After a *FX5,4 has been issued, a VDU2 or CTRL-B will cause the status of the current printer server to be examined. If this status is “jammed” or “busy” then an error (BRK) will be generated to that effect and the VDU2 will be cancelled. If any type of error occurs during transmission or reception from the printer server then an error will be generated but the VDU2 will not be cancelled. For more information see the section on errors. Issuing a VDU3 whilst not printing will have no effect. Issuing a VDU3 whilst printing will send to the printer any data which is in the local printer buffer. This will work even if there is no data to send. This is most useful since this act of transmitting to the printer server will reset its timeout. If the [BREAK] key is pressed whilst printing is taking place then all characters in the printer buffer will be sent and the printing will be terminated: a VDU3 will be simulated.

Extra interfaces

An OSWORD call is available for reading and writing the default printer server type. This <type> is a six character ASCII string. If <type> is shorter than six characters, it is padded to six with spaces. Other extensions will permit the reading of the handle associated with the last error, if there was one. There are extensions to read how many characters are in the network printer buffer, to determine the local network number, and to translate external station addresses to local ones.

Enhancements to the filing system interface

Write only files

File servers support the notion of files which are write only with public access. These files have access strings such as “WR/W” or “LWR/W” to support simple “mail” schemes with some privacy. Write only files should be opened for update and will give an error if an attempt is made to read data (either with OSBGET or OSGBPB). It should be noted that there is no buffering of write only files.

Note that this function requires file servers of the following version numbers or above:

“Level 2 Version 1.05” or “Version IV.05”.

OSFILE

The interface is “create”, and behaves as “save” except that no data is transferred. This means that large files can be created without the necessity of transferring large amounts of data. This is useful where the creation of a file of &10000 bytes would otherwise have required a “save”, e.g. *SAVE FILENAME 0 10000 [RETURN]. This would “crash” the computer because the process of saving would read some “read sensitive” locations. These include data transfer registers of some I/O devices, eg, Tube, ADLC, and FDC. Direct access to this function is provided by the MOS command *CREATE. Note that this function requires file servers of the following version numbers or above:

“Level 2 Version 1.02” or “Version IV.03”.

OSARGS

All calls to “ensure” files save the relevant buffers of an open file to the file server, and then close the file. OSARGS with A=0 returns the same filing system number as NFS 3.40 and NFS 3.60.

A new function, expressed in BASIC as EXT#= is implemented. This can either increase or decrease the length of a file. Note that this function requires file servers of the following version numbers or above:

“Level 2 Version 1.02” or “Version IV.03”.

The amount of disc space allocated to a file can now be read. This value is greater than or equal to the current extent. An interface has also been provided for Z80-based software (CP/M): the file server handle for a particular file. Users are advised NOT to use this handle since its use could result in lost data.

Error messages

Both the “**`Not listening`**” and “**`No reply`**” error messages have the station number added to them to become: “**`Station nnn not listening`**” and “**`No reply from station nnn`**”. The “**`Channel`**” error is followed by the channel that was found to be in error, e.g. “**`Channel 99`**”. If any sort of error occurs during a “random access” operation (OSBGET, OSBPUT, OSARGS, OSFILE, and OSGBPB) then information regarding the channel that had the error will be appended to the error message. For example, if the file server was very busy a message such as “**`Station 254 not listening on channel 32`**” might be caused. It should be noted that the channel associated with the error may not have been the channel on which the operation was attempted. If an OSBPUT is attempted then a buffer may be required. If this means that the previous contents of the buffer need to be written back to the file server then the error may occur on the channel associated with that buffer.

The fatal error caused by the OSWORD call in BBC Model B computers now produces the message "**Remoted**" and may have channel information added if the error occurred during a "random access" operation.

There are eleven more errors than those produced by the BBC Model B:

"**No.**" occurs when an attempt is made to *RUN a file with a load address of &FFFFFFxx and an execute address which is not &FFFFFFFF.

The expanded printer interface now has the extra errors "**Printer busy**" , "**Printer jammed**", and "**Station not present**".

There is an error "**Syntax**" for commands which are recognised but have the wrong syntax (missing parameters).

The ANFS maintains a checksum on its private workspace and this is checked before any random access operation and the new error "**Bad net sum**" may be generated. If a file is opened on a particular file server then it can be accessed only when using that file server. If an attempt is made to use the file whilst logged on to another file server then the error "**Channel nnn not on this file server**" will be issued.

The "**Bad hex**" and "**Bad number**" errors occur if a number was expected and non-hexadecimal or numeric characters were encountered.

The errors "**Bad station number**" and "**Bad net number**" are issued from, for instance, the *I AM command.

"**Bad parameter**" will be caused if a numeric argument is out of range. "**Write only**" is the error from attempting to read a write only file.

"**No more FCBs**" will be given if an attempt is made to open more than 16 objects at one time.

A User Root Directory (URD) reference point

Any object reference that starts with "&" is assumed to refer to a pathway from the URD not the CSD. This is compatible with the ADFS since ADFS interprets "&" as "$" and makes access to files in non-local parts of the user's directory structure easier.

Compatibility with DFS based software

If an object reference starts with ":0.", i.e. an explicit disc reference then it is translated to "&.0.". This means that disc software which references specific files can be made to work under the ANFS by creating directories "0" and "1" in the user's root directory, but prevents the use of single character titles for discs.

Additional library functionality

To be compatible with the ADFS, ANFS adopts the convention that any file which is *RUN <filename>, */<filename>, or *<filename> will be treated as *EXEC <filename> rather than *RUN <filename> if it has an exec address of &FFFFFFFF. This means that EXEC files for commonly used sequences can be stored in the library. There is also a "User Library" which is searched after the CSD and before the CSL. This user library can therefore be used to override normal library functions as well as to extend the user's personal library. The main advantage is that users who need non-standard libraries no longer need to duplicate sections of the main library. This "User Library" is "&.Library", so the users should create a directory called "Library" in their root directory to take advantage of this facility.

Time and Date

The time and the date maintained by the file server have always been readable by the user, via the OSWORD call. There now exists a second OSWORD call which reads the same information in two new (standard) formats. One format returns the information in BCD rather than packed binary, the other returns it as a string. Note that *TIME will return the time and date from the computer's battery-backed clock. OSWORD has to be re-vectored to enable ANFS to reply to this call.

I/O processor address space

The I/O processor is normally accessed using addresses between &FFFF0000 and &FFFFFFFF. The ANFS will use addresses between &FFFF0000 and &FFFFFFFF to refer to the user's RAM, and addresses between &FFFE0000 and &FFFEFFFF to refer to the memory which is the current screen. Note that if a screen is saved using, say, &FFFE3000 to &FFFE7FFF then this will reload correctly on a BBC microcomputer or an Electron microcomputer fitted with ANFS.

Automatic Bootstrapping

During a log-on the normal course is to establish the user's startup option and act on it. Any action which otherwise may have taken place will be suppressed if the

[CONTROL] key is held depressed during the boot sequence. Note that this suppression does not occur if the log-on is a result of a [SHIFT] [BREAK].

Re-tries

The defaults for the number of re-tries for several operations are now “user adjustable”. The number of transmit retries is adjustable from 1 to 255. If zero is used then transmit will try “for ever”. Since some operations are normally inESCapable after 255 tries, the operation becomes ESCapable. The default for transmit is 255. The operation *I AM <user id.> and VDU2 have a prior “machine peek” to determine the existence of the destination station. For this type of transmit the default is ten. When waiting for a reception the receive block is checked some number of times and if the reception has not occurred then the “No reply” error is issued. The default number of check operations is 40. These values are accessed via the OSWORD call.

File server / Bridge net number translation

To support the use of bridges there is a call to translate a net number given by a remote station to one relative to the current station. This is via OSWORD. This means that *VIEW, *REMOTE, and *NOTIFY will work for all cases in multi-net configurations. The same OSWORD also returns the local net number. If this call should fail for some reason the error will be indicated in the data returned to the user.

If a full station number, including network number is given when using the commands *I AM, *FS, and *PS then the network number is compared to the local net number. If the net number given is the same as the local net number then the network number will be treated as zero. This allows the use of global numbering.

Detection of wrong versions and ANFS

Since there is more than one version of the ANFS (e.g. for BBC Model B), a method has been devised to prevent the incorrect version from running in any machine. This is done by checking the version of the operating system. If the wrong operating system is detected the ANFS ROM prints “Bad ROM nn” and then the machine will start up in the normal way and will completely ignore the bad ROM.
It may be necessary for software to detect that the network software is ANFS rather than NFS, if the software is intended to work on BBC Model B microcomputers. To do this the following is recommended:

In BBC BASIC

```
DEF FNIsThisANFS
LOCAL A%, X%, Y%, Value%
DIM X% 3:Y%=X% DIV 256:A%=19
X%?0=15           :REM Read re-try count for ANFS, Read error
                   number for NFS
CALL OSWORD
Value%=X%?2       :REM This byte written by ANFS but not by NFS
X%?2=X%?2 EOR 255 :REM Invert all bits
CALL OSWORD       :REM If NFS is present X%?2 will remain
                   inverted
=(Value% = X%?2)
```

In 6502 assembler (MASM format)

```
IsANFS ROUT ; Returns "EQ" for ANFS "NE" for NFS
   LDXIM :LSB: Block
   LDYIM :MSB: Block
   LDAIM 15
   STA   Block
   LDAIM 19
   JSR   OSWORD
   LDA   Block + 2
   PHA
   EORIM 255
   STA   Block + 2
   LDAIM 19
   JSR   OSWORD
   PLA
   CMP   Block + 2
   RTS
```

Entry of hexadecimal numbers

Where a decimal number is expected, a hexadecimal number may be entered if it is preceded by the “&” character. This is the same format as BASIC uses to enter hexadecimal. Note that where an entry in hexadecimal is expected the “&” character should not be used.

Events on reception

Event number 254 is the Econet receive event. This event is enabled with *FX52,150 and disabled with *FX52,100. When enabled an event will be generated by the completion of a successful reception.

17 TERMINAL EMULATOR

The Terminal Emulator operates both as a language and a service facility. There are two key functions which the language has to ask the service code to perform - actuating its own buffer control and enabling XON/XOFF flow control as appropriate. OSBYTE 96 is used for this purpose.

OSBYTE 96,x

x=&40: Turn off RFC (Receive Flow Control)
x=&41: Turn on RFC
x=&62: Turn off TFC (Transmit Flow Control)
x=&63: Turn on TFC
x=&81: Set up intercept on vectors INSV and REMV
x=&80: Remove intercept.

Terminal File Transfer

Transferring a file either into or out of the MASTER 128 in Terminal mode is most elegantly done with an APC (Application Program Control) sequence to trap OSRDCH or OSWRCH and use OSGBPB to transfer the local buffer accordingly. In many cases however, a more trivial solution will suffice for the transmission of text files.

1) Type *SPOOL Dump-File [RETURN] in Terminal LOCAL mode.
2) Select Terminal LINE mode.
3) Type *TYPE My-File [RETURN] This will cause the file to be transmitted to the remote computer. Only if that computer echoes it back, will it appear on the screen.
4) Select Terminal LOCAL mode when the remote computer indicates completion (it must print a prompt of some description on the screen on detecting the appropriate End-Of-File delimiter).
5) Type *SPOOL [RETURN].

The file will now have been transferred into the other computer. It is its responsibility to ensure that the data has been saved as appropriate. Note that non-text files can be transmitted but the only trivial solution to this is to use *PRINT, rather than *TYPE in the above sequence. This will cause control characters and “top bit set” characters to be encoded in GSREAD format which will need to be translated in the remote machine, which may not always be practicable. There is a special problem with the backslash. In some files this is encoded as character &60 and is translated by the Terminal Emulator to &BB. A received character &BB is not converted back to &60.

18 THE EDITOR

Buffer Transfer

The Editor can be invoked by any language which can implement the following protocol. This enables it to be used as a general language editor, freeing code space for the language itself.

From the language to Editor

Editor puts text in the memory from &E00 up to &7FFF, assuming a shadow screen mode is in use. Text before the cursor is located from &E00 upwards, and that after the cursor, from &7FFF downwards. The language should leave its program at the high end of this space, i.e. without any split, as the editor will, on entry, put the cursor at the first character of the buffer. It should then call the buffer with a command line of:

```
*EDIT <start pointer> <end pointer> [RETURN]
```

The pointers must be in page zero. <start pointer> points to the first location in the buffer. <end pointer> points to the first location after the last character in the buffer. All addresses must be in hexadecimal.

From Editor to the language

The Editor will leave a pointer to the start of text, in location &0000. Text is terminated by a NULL (&00).

Using “Return To Language”, the language will be invoked by:

```
*<language name> @[RETURN]
```

This method is dictated by BASIC which has no service entry point.

19 THE VIEW AND VIEWSHEET FORMATS

The VIEW Word Processor represents text as 7-bit ASCII codes, using certain codes for special control functions. This chapter describes the codes and the overall structure of text both in the computer's memory and in disc files.

Reserved Characters and File Format

VIEW normally only permits the use of characters with ASCII codes &20-&7F (32-127) as text. This allows codes &80 (128) to &FF (255) to be used for control functions, although not all of these are used.

In the Master Compact version of VIEW, only the codes &80 to &86 are used by VIEW, the remaining values, &87 to &FF are available for use as printable characters. The 8-bit function key codes are recognised by the presence of a null character provided by the MOS.

VIEW formatting characters

&09	TAB	- As used in tabulation
&0B	Left Margin TAB	- To set the left margin
&0D	CARRIAGE RETURN	- End of line
&1A	Soft space character	- Inserted between words for formatting
&1C	Highlight (one)	- (Underscore indication)
&1D	Highlight (two)	- (Emboldening indication)
&80	Stored command	- The first character of a stored command
&81	Ruler	- The character preceding a ruler
&80	Printer highlight zero	
&86	Printer highlight six	

TABs are inserted into the text when the TAB key is pressed.

Left Margin Tabs are inserted into the text by VIEW when the left margin is active. In stored files they separate text in the left margin, e.g. stored commands, from the main body of the text. Where there is no left margin text, the line will begin with a Left Margin Tab.

Carriage returns are at the end of each line. The maximum line length is 132 characters excluding the carriage return.

Soft space characters are added between words to justify a line.

Highlight codes (one) and (two) are placed in the text, corresponding to the appropriate Highlight function key. They are translated into printer highlights when the text is printed.

The **stored command** lead-in character only appears as the first character of a line and is immediately followed by the two ASCII characters of the command. Parameters follow the command directly. Spaces appearing on the screen between the command and the parameters are placed there by VIEW.

Similarly, the **ruler** lead-in character will only appear as the first character on a line.

The syntax for a ruler is

<Ruler lead-in character>“..”<Ruler as ASCII text><Carriage Return>

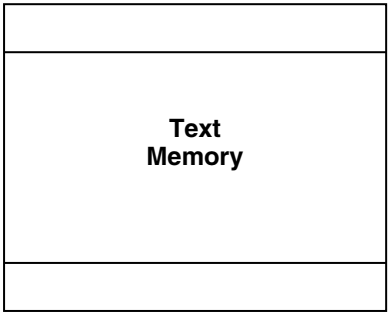
The printer highlight characters do not appear in the body of the text or in the stored file. They are only sent to a printer for control purposes.

Memory Format

Text is stored as a contiguous block. Unlike EDIT, there is no split at the cursor. Text starts with a <CARRIAGE RETURN> and ends with a <NULL>. The NULL must be preceded by a <CARRIAGE RETURN> which will be the end of the last line.

There are three pointers to locate the start, finish and validity of the text.

1) A two byte pointer at &000B indicates the Start of Text.
2) A two byte pointer at &000D points to the NULL after the text.
3) A two byte pointer at &000F points to the End of Text



&0D - Beginning
- Start of Text

&0D - End of Text
&00 - End

VIEW uses a memory consistency check of the following:
1) &AA stored at &000A
2) &AA stored at (&001F)
3) &0D stored at &05CE
4) &0D stored at (&000B)-1
5) &0D,&00 stored at (&000D)

Number Registers

The number registers are stored as two bytes each. The registers are only available and valid during printing and the printer driver can access these if required. The location of each number register is

&798 + 2*n (where n=0 to 25 for registers A to Z).

VIEWSHEET

The data representation of VIEWSHEET is based on dynamic allocation and as such is less straight forward to interface with. Anyone wishing to produce a product to interface with VIEWSHEET should in the first instance contact Acorn Customer Services.

APPENDIX 1 FUNCTIONAL DIFFERENCES BETWEEN MODEL B+ AND MODEL B

Operating system New Series 2 for the 64K machine (Current version is 2.00)

Memory map

Additional 32K of RAM positioned as follows:

Shadow Screen (20K)	&3000 - &7FFF (Sideways)
Paged RAM (12K)	&8000 - &AFFF (Sideways)

The use of shadow screen memory for screen display releases memory in the main memory area for program use. Modes 128 through 135 are shadow screen equivalents of Modes 0 through 7. The 12K of Paged RAM should not be used for applications that may need to be compatible with future Acorn products.

Shadow Screen

*SHADOW (0 or NIL parameter) selects shadow mode regardless of mode number. *SHADOW 1 selects non-shadow mode as the default state, but Modes 128 to 135, when selected, cause entry into a temporary Shadow state. BREAK preserves (or sets queued) Mode/Shadow option. CTRL + BREAK sets default state (non-shadow).

OSBYTE changes

OSBYTE/FX 0 - Read/Display MOS version

(with BRK if X=0)
If X<>0, then value returned in X is:

X = 0	- OS 1.00 (BBC Model A/B or Electron)
X = 1	- OS 1.20 (BBC Model A/B or USA Version)
X = 2	- OS 2.00 (BBC Model B+)

To display the full MOS version number, use OSBYTE with A=0 & X=0 or use *FX0 a BRK instruction precedes the displayed value.
(See also OSBYTE 129 below)

OSBYTE/FX 114 (&72) - set shadow mode state

On entry, X=0 selects shadow, X=1 selects non-shadow.
On exit, X contains previous state, A is preserved, Y and C are undefined.
(This FX call is identical to *SHADOW, and is implemented at the next Mode change or soft BREAK.)

OSBYTE 117 (&75) - Read VDU status

Bit 4 is now used in this call to define actual (not pending) shadow state. Bit 4 is set for shadow state.

OSBYTE 129 (&81) - Read display MOS version

(Special case of INKEY)
On entry, X=0 & Y=255. The value returned in X is:

X = 0 - BBC Model A/B version 0.1
X = 1 - Electron
X = 250 - ABC
X = 251 - BBC Model B+ version 2.00
X = 252 - Reserved
X = 253 - Reserved
X = 254 - BBC Micro USA version
X = 255 - BBC Model A/B version 1.0 or 1.2

Note that BASIC INKEY (-256) performs the same function except that -1 is returned for BBC Model A/B version 1.0 or 1.2 not 255.

OSBYTE 132 (&84) - Read bottom of display RAM address

This returns &8000 in X and Y if Shadow is in operation (not pending).

OSBYTE 133 (&85) - Read bottom of display

RAM address for a specified mode. This returns &8000 in X and Y if shadow is in operation (or pending). It responds similarly if a mode number >127 is in operation (or pending).

OSBYTE 135 (&87) - Read character at cursor text position

This call has not changed at all and can be used to read the currently set Mode. Note that, as before, only the lower three Mode bits are returned hence it cannot be used to determine whether Modes 128-135 are set.

OSBYTE 239 (&EF) - Read/Write shadow mode state

Read/Write location &27F which contains shadow mode flag. This may be a pending status.
To read, X=0, Y=255.
On exit X=0 for shadow, or X=1 for non-shadow.
To write, X=New value, Y=0.
On exit X=Previous value, and Y=contents of &280.
N.B. MOS 1.2 returns X=0 for its only (non-shadow) mode. An application program must therefore find out which operating system is fitted (i.e. whether it is a 32K or 64K machine) if it is to use this call.

OSWRSC (&FFB3) - Write screen

Writes byte in A to the screen. The display location should be set up in &D6 (LSB) and &D7 (MSB).
On entry, Y contains the address offset. This offset feature may not be valid in future machines.
On exit, A, X and Y are preserved. C is undefined.
N.B. Sideways RAM in the range &8000-&AFFF is not written to with this call. Shadow RAM, if selected, will be written to, regardless of screen mode selected, down to &3000. Below &3000, main memory is written to. This call does not work across the TUBE.

OSRDSC (&FFB9) - Read Screen/ROM

(Renamed call - was OSRDRM)
Reads byte from screen into A. The display location should be in &F6 (LSB) and &F7 (MSB). If the address is below &8000, then the entry value of Y is irrelevant (this is the new OSRDSC use). If the address is &8000 or above, then the ROM number in Y on entry is read (this is the original OSRDRM use).

i.e. Address	> &7FFF	Reads ROM always.
	> &2FFF but < &8000	Reads screen selected by shadow/mode command.
	< &3000	Reads main RAM.

This call does not work across the TUBE.

OSWORD changes

OSWORD A=&5 - Read I/O Processor memory

The additional 12K of memory between &8000 and &AFFF can be accessed by ROM ID's 128-255 (i.e. with the top bit set) and hence will not receive service calls. This area can not contain sideways ROMs because the MOS does not switch it in as part of the ROM handling routines. Bytes may be read from the RAM with the top of the memory block set to &FFFExxxx.

The 4K of memory between &B000 and &BFFF is read from the ROM with the equivalent ROM ID Modulo 16.

e.g. Select Paged ROM 135 (128+7):
Reads RAM between &8000 and &AFFF
Reads ROM 7 between &B000 and &BFFF

Note: This OSWORD call cannot be used to read shadow screen with memory block set below &8000.

OSWORD A=&6 - Write I/O Processor memory

Bytes may be written to RAM between &8000 and &AFFF by setting the top of the memory block to &FFFExxxx. User VDU Driver machine code should be placed between &A000 and &AFFF to allow diversion to the appropriate screen (i.e. writing to &3000-&7FFF by machine code in &A000-&AFFF will automatically divert to shadow display RAM if shadow is active).

Note: This OSWORD call cannot be used to write to shadow screen with memory block set below &8000.

Hardware control locations

&FE30 - Paged ROM/RAM select
Bits 0 to 3 select appropriate ROM number.
Bit 7 set selects paged RAM.
There is a RAM copy of &FE30 at &F4.

Sideways ROM layout and selection

The ROM sockets are arranged as two rows of three sockets near the top left of the PCB. ROM selection numbers are as follows:

8,9	10,11	MOS + BASIC 14,15 (opt 0,1)
2,3	4,5	6,7

BASIC is normally at ROM numbers 14 and 15. It can be set optionally to 0 and 1. Link setting is:

S13 North - BASIC is ROM 0 and 1
S13 South - BASIC is ROM 14 and 15

ROM numbers 12 and 13 are not decoded. As in the Model B, high ROM position numbers have priority at reset.

For 16K ROMs, links S9,11,12,15,18,19 for ICs 35,44,57,62,68,71 respectively are set West. For 32K ROMs set the appropriate link East. Priority for 32K ROM is same as 16K for ‘lower’ 16K, ‘top’ 16K has one priority lower.

Disk Filing System

When fitted, the Disk Filing System (DFS) is largely compatible with that used in the Model B. The Floppy Disk Controller chip, however, is normally a Western Digital 1770 in the Model B+ rather than the Intel 8271 fitted to the Model B. When the 1770 is used, the screen display shows “Acorn 1770 DFS”. This DFS has a few additional commands, including Format and Verify utilities in the ROM. When creating disk software protection routines, software writers should not make any assumptions as to the FDC hardware fitted in the micro. Disks that have in the past accessed the 8271 directly through memory-mapped I/O for protection will probably not work with the 1770 DFS.

Second Processors

The 1770 DFS ROM contains the TUBE code necessary when a second processor is fitted. If a second processor is used with a non-disc Model B+, then either a 1770 DFS ROM, or the DNFS ROM supplied with the second processor will need to be fitted to provide the TUBE code.

Compatibility with Aries B32 and similar Model B “Add-Ons”

Aries B32 and similar memory extensions typically run, by default, in an equivalent mode to SHADOW on. A good degree of compatibility can be obtained by the use of “Mode 1xx” (i.e. Top Bit set) in an application program where the extra memory is required. Aries B32 (i.e. the Model B base machine) will ignore the top bit of the Mode command, but will typically already be in a SHADOW state. The Model B+ will respond and enter the SHADOW state.

Where compatibility is required across the range of products, the use of the OS calls which determine the memory currently available (i.e. HIMEM position), or the memory which will be available in a certain Mode, are the preferred route. Calls to the OS to determine if a Model B+ is the environment will, of course, not provide the complete story if a Model B is being used with an Aries B32 or similar extension, as the Operating System will not know of their existence.

APPENDIX TWO FUNCTIONAL DIFFERENCES BETWEEN MASTER 128 AND MODELS B AND B+

Operating System version

New Series 3 OS for the Master 128/512/Sc/Turbo
New Series 4 OS for the Master Econet Terminal.

Operating System changes

As version 1.2 with following extensions:

Shadow Screen - Use of MODE 128 to 135 instead of MODE 0 to 7 results in the use of a section of memory for the screen separate from the main area of RAM. This allows more room for user programs.

Real-time clock - the status of a Real-time clock can be called from the MOS. The OSWORD calls &0E and &0F Read & Write the clock in BCD or text formats.

Default command line interpreter if no language present or *GO used.

Configure CMOS RAM commands - the state of the CMOS RAM can be set for both reserved and unreserved bits. Reserved Bits include:

MODE <0-7,128-135>	Start-up mode
FS <0-255[.0-255]>	File-Server station number
PS <0-255[.0-255]>	Printer-Server station number Transient Command - Econet station ID (Must be set by Net Manager) FS & PS status bits operational only when ANFS fitted
LANG <ROM>	Start-up language ROM number
FILE <ROM>	Start-up filing system ROM number
TV [<Dec>[,<Dec>]	*TV position & Interlace state
DELAY <0-255>	Auto-repeat delay
REPEAT <0-255>	Auto-repeat rate
PRINT <0-255>	Default printer (*FX5 type)

IGNORE <0-255>	Print ignore character (no param=no ignore)
EXTUBE/INTUBE	Tube selected (Internal/External) (Looks for internal Tube if no external fitted)
NOTUBE/TUBE	Tube ON/OFF (TUBE defaults to I/O Proc if no co-processor fitted)
BAUD <1-8>	Serial baud rate (Both ways)
DATA <0-7>	Serial data format
BOOT/NOBOOT	Boot status - (Reverses BRK and SHIFT + BRK action)
SCROLL/NOSCROLL	Scroll state (on/off)
FDRIVE <0-3>	Floppy-Drive params (speed etc) (Use low number for highest step rate 0 has MFM pre-compensation 1 is same speed but no compensation)
FLOPPY/HARD	Floppy or Winchester start-up (with HARD, floppy is selected if no Winchester fitted)
DIR/NODIR	ADFS or FADFS as default (FADFS does not mount disc automatically)
LOUD/QUIET	Bell character volume
CAPS/NOCAPS/SHCAPS	Caps lock on/off

Configuration is set using *CONFIGURE <param> or read using *STATUS <param> or just *STATUS to read all.

Soft characters fully exploded
Improved soft-char structure with more buffer area.

New resident Operating System commands:

***APPEND**	functions as BUILD but appends to end of file.
***BUILD**	as DFS but control codes input by | mechanism.
***CLOSE**	closes open files on current filing system.
***CONFIGURE**	to set start-up options.
***CREATE**	creates empty file using *SAVE params.
***DELETE**	as DFS delete.
***DUMP**	<start in file> <start address on output>.
***EX**	examine files in specified directory.
***EXEC**	take input from file rather than keyboard.
***GO**	to enter address in language or I/O processor.
***GOIO**	to enter program at address in I/O processor.
***IGNORE**	as *FX6.
***INFO**	as *EX, but for single files or Wildcard use.
***INSERT**	inserts ROM number n into ROM map from reset.
***LIST**	list file in GSREAD format (*TYPE with line nos.)
***LIBFS**	define FS where LIB is from current FS.
***MOVE**	copies files from one place to another including between filing systems.
***PRINT**	as TYPE but gets/processes vdu codes.
***REMOVE**	as DELETE but no error message if not found.

***ROMS**	lists ROM names, sockets, versions & UNPLUG/PLUG.
***SHADOW**	with 0 or no parameter, gives shadow on next mode. with 1, drops shadow on next mode.
***SHOW**	displays soft-key contents.
***SHUT**	close open files on all filing systems.
***SRDATA**	reserve Sideways RAM for data use.
***SREAD**	copy Sideways RAM to main RAM.
***SRROM**	reserve Sideways RAM for direct addressing.
***SRWRITE**	copy main RAM to Sideways RAM.
***SPOOL**	direct screen output to named file.
***SPOOLON**	functions as SPOOL but appends to end of file.
***STATUS**	lists status of start-up options.
***TIME**	displays time from Real-time clock.
***TYPE**	list file with control codes displayed as |@ etc.
***UNPLUG**	removes ROM number n from ROM map from reset.

Numeric keypad - options to set base for range of values returned and set whether SHIFT/CTRL etc affect codes.

Cassette filing system upgraded to do OSGBPB (calls 1 & 3) in those cases where it is to write to or read from the current pointer.
OSFSC extended for CFS to provide call 7 so that it returns legal file handle range.
CFS will perform *EX (but not *INFO).
CFS can be reselected by *FX143,18,<n> where n=1 (1200 baud), n=2 (300 baud), n=3 (ROM filing system).

Multifiling system capability
Name of filing system can be prefixed to the file name itself.
Files may be opened and kept open across several filing systems.
Filing names currently defined include CFS (or TAPE), ROM, DFS, ADFS, NET.

Extended graphics facilities:
Extended Fill commands using checkerboard pattern/stipple.
Mark-to-space ratio of dotted line.
New triangle algorithm for correct plotting under all conditions.
Parallelogram - Solid fill.
Rectangle - Solid fill.
Circle - Outline.
Circle - Solid fill.
Arc - Line fill.
Arc - Solid fill (Pie shape).
Arc - Solid fill (Chord segment).
Ellipse - Outline.
Ellipse - Solid fill.
Fill enclosing shape outline (flood-fill).

Enhanced speed of all fill operations.
Move/copy rectangle.
Extended horizontal line fill.

Extension text:
Improved character draw speed (especially VDU5).
Clear block of text window.
Output character with no cursor move (dead key).
Advance left/right after drawing character.
Pending scroll to allow bottom right character to be drawn.
Cursor options held across mode changes.
Direct window scroll - left/right/up/down.
Extra 128 standard characters in ASCII range 128 to 255.

OSBYTE calls

&0 (0) Enter with X<>0, Returns X=3 for Master 128
Entering with X=0, (or *FX0) displays MOS version

&14 (20) parameters for *FX20 are now ignored and *FX20 resets standard exploded font. Software writers must add parameters as required for Model B & B+

&16 (22) Increment ROM polling semaphore. Used to request MOS polling with service call 21 every 10mS (polling with semaphore non-zero)

&17 (23) Decrement ROM polling semaphore. Used to stop MOS polling with service call 21 every 10mS

&6B (107) Switch Internal/External 1MHz Bus
*FX107,0 - Select external bus (default)
*FX107,1 - Select internal (cartridge) bus

&6C (108) Switch Main/Shadow, memory into main map (&3000-&7FFF)
*FX108,0 - Switch Main memory into main map area (immediate)
*FX108,1 - Switch Shadow memory into main map area (immediate)

&6D (109) Make temporary filing system permanent

&70 (112) Write to Main/Shadow memory
*FX112,0 - Write to memory specified by mode change
*FX112,1 - Write to main memory (immediate)
*FX112,2 - Write to shadow memory (immediate)

&71 (113) Display Main/Shadow memory
*FX113,0 - Display memory specified by mode change
*FX113,1 - Display main memory (immediate)
*FX113,2 - Display shadow memory (immediate)

&72 (114) Write to/Display Main/Shadow memory (*SHADOW)
*FX114,0 - Use shadow memory at next mode change
*FX114,n - Use mode defined at next mode change (where n is 1-255)

&81 (129) Now extended to return new MOS version
Enter with X=0 & Y=255. Returns X=253 for OS 3.x, X=247 for OS 4.x, X=245 for Master Compact OS 5.x
&84 (132) Read top of user RAM (was display RAM start)
&85 (133) Read top of user RAM for a given mode (was display RAM start for a given mode)
&A1 (161) Read CMOS RAM
Enter with X=n, where n is the RAM location number (30-49)
Returns with result in Y
Use *STATUS for locations 0-29
&A2 (162) Write CMOS RAM
Enter with X=n, where n is new RAM location number (30-49)
*FX162 can be used. Use *CONFIGURE for locations 1-29.
Location 0 is protected
&A4 (164) Check processor type
&A5 (165) Read output cursor position
&B3 (179) Read/Write ROM polling semaphore (was Read/Write OSHWM)
A=179, X=n, Y=0 reads semaphore into X and sets state to n. Setting state directly with this call will interfere with OSBYTE 22 & 23 use.
A=179, X=0, Y=255 reads semaphore into X
&B6 (182) Read NOIGNORE state (was Read font explosion)
&EE (238) Changes numeric pad base
*FX238,<base> with base from 0-255 will alter key characters to their ASCII value -48+<base>
&FA (250) Read memory area used for writing to
&FB (251) Read memory area used for reading from
&FE (254) Controls effect of SHIFT on numeric pad
*FX254,0 - makes SHIFT have effect
*FX254,<1-255> - deletes effect of SHIFT
(Note: This call returned RAM size in Model B & B+)

OSWORD calls

&E (14) Read CMOS clock
&F (15) Write to CMOS clock

New Service calls to Sideways ROMs

&15 (21) Polling interrupt. Made 100 times per sec. if OSBYTE 22 issued
&18 (24) Interactive HELP. Made by MOS when it executes a *HELP command, after service call 9. MOS offers CLI text following a *HELP to a ROM participating in the interactive help system

&21 (33) Offer Static Workspace in Hidden RAM. Call is made on a Reset. Workspace starts at &C000 in Hidden RAM and can only be used by a Filing System, and only one at a time. Workspace has an upper limit of &DBFF. Call analogous to &01, but uses hidden RAM

&22 (34) Offer Dynamic Workspace in Hidden RAM. ROMs should ideally ignore Call &02, which takes workspace in main memory

&23 (35) Tells ROMs location of top of Static Workspace in Hidden RAM

&24 (36) Dynamic Workspace requirements. ROMs should indicate how much memory they will each claim through Call 34. Y contains current bottom of dynamic allocation and should be decremented by required number of pages

&25 (37) Inform MOS of filing system name and info
(See Reference Manual 1 for detailed information on this call)

&26 (38) Close all files. Issued at a Reset. Filing systems should select themselves, close open files and then de-select. Used by *SHUT

&27 (39) Reset has occurred. Call made after hard reset. Mainly for Econet Filing system so that it can claim NMIs. This call is now required since the MOS no longer offers workspace on a soft BREAK. A Sideways ROM should therefore re-initialise itself

&28 (40) Unknown CONFIGURE option. Used to extend range of commands. A Sideways ROM having a claim on CMOS RAM may use this command to update its configuration information

&29 (41) Unknown STATUS option. Used to provide extra commands. See &28.

&2A (42) ROM-based language starting up. This enables languages, such as the TERMINAL, to remove their interception of buffering functions etc. prior to the next language taking control

VDU commands

VDU18,m,c - Define graphics colour

m = 0 to 4	same as MOS 1.2
m = 5	Leave screen colour unchanged
For each of n = 1,2,3,4 (ecf pattern numbers):	
m = 16n	Overwrite the colour on the screen
m = 16n + 1	OR the colour of the screen
m = 16n + 2	AND the colour of the screen
m = 16n + 3	EOR the colour of the screen
m = 16n + 4	Invert the colour of the screen
m = 16n + 5	Leave screen colour unchanged

VDU22,m - Select screen mode

m = 0 to 7	As MOS 1.2
m = 128 to 135	covers shadow screen modes

VDU23,0,r,v,0,0,0,0,0,0 - Control 6845 CRTC directly

As MOS 1.2 i.e. writes value of v to 6845 register r

VDU23,1,n,0,0,0,0,0,0,0 - Turn cursor on/off

As MOS 1.2 (but with additions of n=2 & n=3):

n = 0 Stops cursor appearing
n = 1 Cursor appears on screen (Default case)
n = 2 Cursor is steady
n = 3 Cursor flashes at approx 1.5 times/sec (Default)

Flash rate is doubled in cursor edit mode

VDU23,2-5,a,b,c,d,e,f,g,h - Set ecf pattern

ecf patterns can be set to pixel groups of 8*8, 4*8 or 8*8 if mode has 2, 4 or 16 colours respectively. VDU23,2 to VDU23,5 sets patterns 1 to 4 respectively.

Integers a to h define pattern rows from top to bottom. If the integer is derived from stuvwxyz in binary, then: For 2 colour mode, logical colours from left to right are: s, t, u, v, w, x, y, z

For 4 colour mode, logical colours from left to right are: sw, tx, uy, vz

For 16 colour mode, logical colours from left to right are: suwy, tvxz

VDU23,6,n,0,0,0,0,0,0,0 - Set dotted lines pattern

n = &FF Solid line
n = &AA Dotted line as in MOS 1.2 (Default, reset on mode change)
n = &EE Dashed line (dot-dot-dot-space repeated)
n = &E4 Dash-dotted line (dot-dot-dot-spc-spc-dot-spc-spc repeated)

VDU23,7,m,d,z,0,0,0,0,0 - Scroll window directly

Allows text window or arbitrary rectangle to be scrolled without cursor movement:

m = 0 Scroll text window
m = 1 Scroll entire screen
d = 0 Scroll right
d = 1 Scroll left
d = 2 Scroll down
d = 3 Scroll up
d = 4 Scroll in positive X direction (defined by VDU23,16, etc.)
d = 5 Scroll in negative X direction (defined by VDU23,16, etc.)
d = 6 Scroll in positive Y direction (defined by VDU23,16, etc.)
d = 7 Scroll in negative Y direction (defined by VDU23,16, etc.)
z = 0 Scroll by 1 character cell
z = 1 Scroll by 1 character cell vertically, 1 byte horizontally

(i.e. 8 Pixels in 2-colour modes, 4 in 4-colour modes, 2 in 16-colour modes, and 1 character in mode 7). This is the minimum distance that can be scrolled to enable a hardware scroll if the full screen is scrolled.

VDU23,8,t1,t2,x1,y1,x2,y2,0,0 - Clear block of text window

This causes a block of the text window to be cleared to the text background colour. The parameters indicate where the two ends of the block (i.e. string start & string finish) are, with t1,x1 and y1 relating to the start of the block and t2,x2 and y2 to the end of the block. In each case, it indicates a base position (ti), to which (xi,yi) is added to get the true position. The character position at the start of the block is generally included in the clear, but that at the end is not.

ti = 0 Base position is “top left of window”
ti = 1 Base position is “top of cursor column”
ti = 2 Base position is “off top right of window”
ti = 4 Base position is “left end of cursor line”
ti = 5 Base position is cursor position
ti = 6 Base position is “off right end of cursor line”
ti = 8 Base position is “bottom left of window”
ti = 9 Base position is “bottom of cursor column”
ti = 10 Base position is “off bottom right of window”

Other values of ti have undefined effects.

(The quotes are to indicate that all of these positions are calculated taking the cursor movement control set by VDU23,16 into account e.g. after VDU23,16,2,0,0,0,0,0,0,0, “left” above right etc.)

The results of this function are undefined if the absolute values of the coordinates of the two ends go outside the range -128 to 127. This is best avoided by not using values of xi and yi outside the range -128 to 47. Should the end point of the block lie before the start point, no clearing will be done.

VDU23,9,n,0,0,0,0,0,0,0 - Set 1st flash time

Same spec as *FX9 in MOS 1.2

VDU23,10,n,0,0,0,0,0,0,0 - Set 2nd flash time

Same spec as *FX10 in MOS 1.2

VDU23,11,0,0,0,0,0,0,0,0 - Set default ecf patterns

Mode	Pattern	Colour	VDU23,2-5 type definition
0	1	Dark grey	&CC,&00,&CC,&00,&CC,&00,&CC,&00
	2	Grey	&CC,&33,&CC,&33,&CC,&33,&CC,&33
	3	Light grey	&FF,&33,&FF,&33,&FF,&33,&FF,&33
	4	Hatching	&03,&0C,&30,&C0,&03,&0C,&30,&C0
1,5	1	Red-orange	&A5,&0F,&A5,&0F,&A5,&0F,&A5,&0F
	2	Orange	&A5,&5A,&A5,&5A,&A5,&5A,&A5,&5A
	3	Yellow-orange	&F0,&5A,&F0,&5A,&F0,&5A,&F0,&5A
	4	Cream	&F5,&FA,&F5,&FA,&F5,&FA,&F5,&FA

```
2       1       Orange          &0B,&07,&0B,&07,&0B,&07,&0B,&07
        2       Pink            &23,&13,&23,&13,&23,&13,&23,&13
        3       Yellow-green    &0E,&0D,&0E,&0D,&0E,&0D,&0E,&0D
        4       Cream           &1F,&2F,&1F,&2F,&1F,&2F,&1F,&2F
4       1       Dark grey       &AA,&00,&AA,&00,&AA,&00,&AA,&00
        2       Grey            &AA,&55,&AA,&55,&AA,&55,&AA,&55
        3       Light grey      &FF,&55,&FF,&55,&FF,&55,&FF,&55
        4       Hatching        &11,&22,&44,&88,&11,&22,&44,&88
```

Mode 0 patterns are different from 4 to avoid TV effects.

VDU23,12-15,a,b,c,d,e,f,g,h - Set simple ecf pattern

This sets a simple 2*4 (or double for mode 0) pattern. Patterns 1 to 4 are set by VDU23,12 to VDU23,15 respectively. The logical colours from left to right are:

Top row - a,b
next row - c,d
next row - e,f
last row - g,h

Mode 0 has double pixels to avoid TV patterning.

VDU23,16,x,y,0,0,0,0,0 - Cursor movement control

Allows control of cursor after a character has been printed. This control sequence replaces the current flag byte as follows:

((current byte) AND y) EOR x

If the byte flag is abcdefgh in binary, then:

a = 0 Normal
a = 1 Undefined
b = 0 In VDU 5 mode, cursor movement outside a window causes special actions i.e. Carriage returns generated
b = 1 In VDU 5 mode, cursor movement outside a window does not cause special actions
c = 0 Cursor moves in positive direction. d & h define action if cursor moves outside a window
c = 1 Cursor does not move
d = 0 If Y movement would go outside a window, window is scrolled in VDU 4 mode; in VDU 5 mode it moves to opposite edge of the window.
d = 1 As above but cursor always moves to opposite edge
efg=000 Text X direction is right, Y direction is down
efg=001 Text X direction is left, Y direction is down
efg=010 Text X direction is right, Y direction is up
efg=011 Text X direction is left, Y direction is up
efg=100 Text X direction is down, Y direction is right
efg=101 Text X direction is down, Y direction is left
efg=110 Text X direction is up, Y direction is right
efg=111 Text X direction is up, Y direction is left

h = 0 If movement would go outside a window, cursor moves to negative edge and one step in positive Y direction. If this goes outside a window, d defines behaviour. This is '80' column mode

h = 1 If movement would go outside a window, a 'pending cursor movement' is generated. It is released before next character is printed (or another control code). This is '81' column mode

VDU23,17-26,a,b,c,d,e,f,g,h - Unassigned (but reserved)

VDU23,27,a,b,c,d,e,f,g,h - Acornsoft sprites

VDU23,28-31,a,b,c,d,e,f,g,h - Unassigned (for user application programs)

Reserved for use by application programs. Results in a call to the unknown Plot codes vector &226,&227. Call can be recognised as follows:
C = 1 on entry to the vector.
A contains the VDU23 code (i.e. the first number following 23).
All of the sequence except the 23 can be found in ascending order starting at the location: (Start of VDU variables) + &1B, i.e. at &31B in MOS version 1.2

VDU23,32-255,a,b,c,d,e,f,g,h - Define character

Spec as MOS 1.2

VDU24,ll,lh,bl,bh,rl,rh,tl,th - Set graphics window

Spec as MOS 1.2

VDU25,p,xl,xh,yl,yh - Plot

VDU25,0-63 - Plot line

Spec as MOS 1.2, but some improvements

VDU25,64-71 - Plot point

Same as MOS 1.2

VDU25,72-79 - Horizontal line fill

Spec as MOS 1.2

VDU25,80-87 - Plot triangle

Spec as MOS 1.2

VDU25,88-95 - Horizontal line fill

Spec as MOS 1.2

VDU25,96-103 - Plot rectangle

Plots a filled axis aligned rectangle with opposite corners at the current graphics cursor and the new point.

VDU25,104-111 - Horizontal line fill

Similar to VDU25,72-79…, with the difference that the word "nonbackground" should be replaced by "foreground"

VDU25,112-119 - Plot parallelogram

Plots a filled parallelogram with vertices at the old graphics cursor, the current graphics cursor, the new point, and at (new point)-(current graphics cursor)+(old graphics cursor) in cyclic order. The fourth point is calculated in terms of internal pixel co-ordinates to ensure that the sides are parallel.

VDU25,120-127 - Horizontal line fill

Similar to VDU25,88-95…, with the difference that the word "background" should be replaced by "non-foreground"

VDU25,128-143 - Flood fill

This flood fills the screen starting from the new point and continuing until non-background (plot codes 128-135) or foreground (plot codes 136-143) pixels are found. These sequences make use of soft-key 11-15 buffers (they will reset soft keys to empty strings and will fail to do anything if these soft keys are being expanded. Sequences may fail if the area to be filled is too complicated, the colour being used to fill can itself be filled or an escape occurs

VDU25,144-159 - Plot circle

Plots a circle outline (plot codes 144-151) or a filled circle (plot codes 152-159) with its centre at the current graphics cursor and the new point on its boundary.

VDU25,160-183 - Plot circular arc

Plots a circular arc (plot codes 160-167) the filled chord segment between a circular arc and the chord joining its end points (plot codes 168-175) or the filled pie sector between a circular arc and the two radii joining its end points to the centre of the circle (plot codes 176-183). In all three cases, the centre of the circle is at the old graphics cursor, the first endpoint of the arc is at the current graphics cursor the second endpoint of the arc is on the circle and in the same direction from the centre of the circle as the new point is, and the circular arc is taken to be the arc going clockwise from the first end point to the second one.

VDU25,184-191 - Move/copy rectangle

Causes the axis-aligned rectangle with opposite corners at the old and current graphics cursors to be moved (plot codes 185,189) or copied (plot codes 186,187,190,191) so that its new bottom left hand point is at the new point (plot codes 184 and 188 simply move the graphics cursor to the new point, like other plot codes which are 0 MOD 4).
Any part of the source rectangle which lies outside the current graphics window is assumed to contain the current graphics background colour for the purposes of the copy or move. The difference between copying and moving is that moving sets any part of the source rectangle which lies outside the destination rectangle to background, whereas copying leaves such parts of the source rectangle unchanged

VDU25,192-207 - Plot ellipse

Plots an ellipse outline (plot codes 192-199) or a filled ellipse (plot codes 200-207). The centre of the ellipse is at the old graphics cursor.

VDU25,208-231 - Unassigned

Not reserved for application programs

VDU25,232-239,xl,xh,yl,yh - Acornsoft sprites

VDU25,240-255 - User program calls

Reserved for application programs. Will result in a call to the unknown plot codes vector (&226,&227). Call recognised by:
C=0 on entry
Computer is in a graphics mode (can test location (start of VDU variables) + &61, i.e. &361 on MOS 1.2 This contains (number of pixels/byte)-1 (i.e. 1,3 or 7) in graphics modes, and 0 in nongraphics modes).
A contains the VDU25 code (i.e. the first number following the 25). The coordinates can be found in ascending order starting at the location (start of VDU variables) + &20 i.e. &320 on MOS 1.2

VDU26 - Restore default windows

Spec as MOS 1.2

VDU27 - Null

Spec as MOS 1.2

VDU28,lx,by,rx,ty - Define text window

Spec as MOS 1.2

VDU29,xl,xh,yl,yh - Define graphics origin

Spec as MOS 1.2

VDU30 - Home cursor
Spec as MOS 1.2

VDU31,x,y - Tab cursor
Spec as MOS 1.2

VDU32-126 - Print a character
Spec as MOS 1.2

VDU127 - Backspace and delete
Spec as MOS 1.2

VDU128-255 - Print a character
Prints characters from the extended character set in a similar manner to VDU32-126

EDITOR

Text file editing with ability to go to defined line numbers.
Includes a formatter to provide reasonable presentation on printed documents.
Can display on-screen help info. State of this HELP info is kept in CMOS RAM.
Cursor keys used to move cursor around screen, with screen scrolling up or down as necessary.

Shift mode	Action
NONE + up/down arrow	cursor moves one line
NONE + left/right "	" " character
SHIFT + up/down arrow	" " screen
SHIFT + left/right "	" " word
CTRL + up/down arrow	" " top/bottom doc
CTRL + left/right "	" " start/end line

ESCAPE safely abandons most operations.
Function key operations:

f0 - Goto specified line number
SHIFT f0 - Toggles between invisible display of carriage returns (default) and small reverse video 'M's, thus easily controlling trailing spaces.
f1 - Access to OSCLI e.g. <f1>CAT for Disc Catalogue.
SHIFT f1 - Toggles entry mode between insert (default) and overtype. State is displayed at bottom left of screen.
f2 - Load text from named file.
SHIFT f2 - Insert text from named file at the cursor position.
f3 - Save text to named file.
SHIFT f3 - Remove top and bottom scroll margins.

f4 - Enter interactive find and replace option. A prompt of FIND/REPLACE will appear at bottom of the screen. The text until the next '/' or RETURN is the string to be searched for.
Some characters represent powerful operators:

- $ - Carriage return.
- . - ANY character except carriage return.
- # - A digit 0 to 9.
- @ - An alphabetic (a to z, A to Z) or digit.
- ~ - 'NOT' the following character
- * - Any number of the following character.
- ^ - As many as possible of the following character.
- \ - An escape to allow special characters to be introduced.
- - - Signifies a range e.g. a-z matches a lower case character.
- [ ] - Is a set of options for this character e.g. [0123] would allow 0,1,2, or 3.
- | - Control characters as MOS e.g. |J is CTRL + J.

If the string is ended in a RETURN i.e. no /, then the string will be found (if possible) and a prompt of R(eplace), C(ontinue) or ESCAPE will be displayed. C moves on to the next occurrence, R replaces and moves on, ESCAPE stops altogether. The text after the second '/' is the string which may replace the found string. If there is no / then you may type in the text and move on when R is used. There are special characters here:

- $ - Again represents carriage return.
- \ - As find string.
- | - As find string.
- & - Represents the found string.
- %n - (n is 0 to 9) represents the nth wild section of the found string.

SHIFT f4 - Return to language, text being replaced into language. Clear text buffer.

f5 - Global replace. Syntax exactly as find string. Displays the number of found objects after working. Will operate either over whole file or from mark to cursor.

SHIFT f5 - Set HELP display mode.

f6 - Mark position. When present, the character that the cursor is on will be replaced by an inverted 1 or 2 and the number of marks displayed at the bottom left of the screen will be increased. Marked positions can be used to delete blocks of text, copy and move them, and perform restricted searches. Up to two marked positions may be set; an attempt to set a third will produce an error message. No editing may be done while marks are set.

SHIFT f6 - Remove all marks.

f7 - Copy text delimited by two marks to where the cursor is now. The marks are not cleared so the operation can be repeated. The cursor must not be in the marked area.

SHIFT f7 - Move text delimited by two marks to where the cursor is now. The marks are cleared. The cursor must not be in the marked area.

f8 - Print out text

SHIFT f8 - Delete text between single marked position and current cursor position. An error message will occur if there is not exactly one mark present.

f9 - Get old text back. Works if SHIFT f9 has just been pressed or EDITOR has been broken out of and re-entered.

SHIFT f9 - Delete all text.

Miscellaneous changes

Sideways ROM headers which have been “illegal” previously on Models B & B+, but accepted by the MOS, may not be accepted by this MOS version.

The screen Paged Mode algorithm has been improved in this MOS, resulting in slight differences in the number of displayed lines in some modes compared with Model B & B+.

In the Econet Terminal, the User VIA chip (6522) is not normally fitted. Application software intended for Econet use should not use the VIA timer.

The RAM latch at &FE34 has some changes and additions to the use of each bit.

The machine start-up mode & language are now easily reconfigured. Software should always have an auto-boot file which sets the language and screen mode, without assuming a default state. (The machine may NOT be in Mode 7 or in BASIC at disk Boot-up).

Software should not use “abbreviation” due to the risk of clashes with other fitted ROMs with similar *commands. In particular, *D. no longer selects the DFS, but selects *DUMP.

Software should NEVER assume the position of PAGE. If it needs to know it should ask the operating system where OSHWM is.

To provide conversion compatibility across DFS & ADFS, note the following:

*DRIVE ADFS does not support this. Use *DIR :X.Y, which is a format common to both filing systems.

*DIR ADFS does not support moves directly from one directory to another, arbitrarily across the structure. Use the full pathname where relevant in DFS to support ADFS.

Programs should never define soft characters by directly loading into the soft-char definition area (&C00-&CFF).

Memory map changes

The Shadow screen in the Model B+ from &3000 - &7FFF (Sideways) is present in the Master 128 machine. The 12K of additional Paged RAM in Model B+ is implemented differently in the Master 128 machine and is reserved for operating and filing system use as “Private RAM” (see below).

The minimum machine configuration is 128K, where 64K is allocated to sideways RAM (4 by 16K pages) in a similar way to the 128K Model B+. Use of internally fitted ROMs in two of the three sockets will require changes to the link settings which will remove access to some of the sideways RAM. (They are ROM locations 4,5,6 & 7).

Memory map below OSHWM

with changes from Model B/B+ indicated

Page zero - &00-&FF

&00-&8F Language workspace. BASIC allows &70-&8F for the user.
&90-&9F ECONET workspace. Do not use.
&A0-&A7 NMI workspace. It must be claimed before use.
The owner must
1) have a Filing System number allocated to it. Note the error in the Advanced User Guide (Bray, Dickens & Holmes) on page 323. The NMI ID passed around by Service Calls &0B and &0C are Filing System numbers not ROM numbers.
2) be able to process ROM Service Calls &0B and &0C i.e. they must be ROMs or intercept OSBYTE &8F.
&A8-&AF MOS scratch space.
&B0-&BF Filing System scratch space. Watch out for “hidden” filing system calls i.e. those produced by OSWRCH if *SPOOL used.
&C0-&CF Current Filing System PRIVATE workspace.
&D0-&FF MOS workspace only

Page one - &100-&1FF

Machine stack and error message buffer.

Page two - &200-&2FF

&200-&235	Vectors.
&236-&28F	Main MOS variables.
&290-&2FF	MOS workspace (for MOS only).

Page three - &300-&3FF

&300-&37F	VDU variables (for use by graphics routines only).

In the Model B, B+ and this machine, Page three is used for VDU workspace. Most variables are the same, with the following exceptions:

	Model B & B+	**Master**
&359	Foreground graphics colour	Plotting foreground/background
&35A	Background graphics colour	Current graphics plot mode
&366	Mode 7 cursor	Cursor control flags (VDU 23,16)
&367	Exploded font flag	Dotted line pattern (VDU 23,6)
&368	Exploded font location bytes	Current dotted line state
&369		Plot colour (0-solid,Not 0-pattern)
&36A		F/grnd col. (0-solid,Not 0-pattern)
&36B		B/grnd col. (0-solid,Not 0-pattern)
&36C		Col. 81 flag (top bit set pending)
&36D		Foreground graphics colour
&36E		Background graphics colour
&380-&3DF	Cassette Filing System workspace (do not use).	
&3E0-&3FF	Keyboard input buffer (do not use unless replacing).	

Page four, five, six and seven - &400-7FF

Language workspace. Do not use unless you are the current language or the current language has allowed you to use it.

Page eight - &800-&8FF

&800-&87F	Sound workspace. DO NOT USE (unless you want strange noises!).
&880-&8BF	Printer buffer. Useable if, and only if, no printing needed.
&8C0-&8FF	Sound workspace (envelopes 1-4). Useable if, and only if, envelopes not needed.

Page nine - &900-&9FF

&900-&9BF	RS423 O/P buffer, or cassette O/P buffer (1st part), or sound workspace (envelopes 5-16). Do not use for anything else, as this area could be re-allocated.
&9C0-&9FF	Speech buffer, or cassette O/P buffer (2nd part). Do not use for anything else, as this area could be re-allocated.

Page ten - &A00-&AFF

RS 423 I/P buffer, or cassette I/P buffer. Do not use for anything else, as this area could be re-allocated.

Page eleven - &B00-&BFF

ECONET workspace. Do not use.

In the Model B & B+ this is used for the soft key buffer. In this machine, the soft key buffer resides in the Private 12K RAM and should not be accessed by the user. Programs directly accessing Page eleven on the Model B & B+ for soft-key purposes will be incompatible with this machine

Note: Initial Page states are:

Model B, B+: &B00-&BFF - &10

This machine: &B00-&BFF - &00

Page twelve - &C00-&CFF

ECONET workspace. Do not use.

In the Model B & B+ this is used for user defined characters in the range ASCII 224-255. In this machine, all characters up to ASCII 255 are defined with a standard font. This definition resides in the Private RAM area and should not be accessed by the user (the user can, however, re-define all of them as before with VDU23). Programs directly accessing Page twelve on the Model B & B+ for user defined characters will be incompatible with this machine.

Note: Initial Page states are:

Model B, B+: &C00 - &0D &C01 -&CFF - &00

This machine: &C00-&CFF - &00

Page thirteen - &D00-&DFF

&D00-&D5F	NMI routine and workspace. NMI's must be claimed to use this area.
&D60-&D9E	ECONET private workspace. Do not use.
&D9F-&DEF	Expanded vector set.
&DF0-&DFF	Paged ROM workspace, one byte per ROM.

12K Private RAM

&8000-&83FF	Soft-key buffer. Do not use directly.
&8400-&87FF	VDU workspace. Reserved for routines that need a large area, such as flood-fill. Do not use directly.
&8800-&88FF	VDU variables and workspace. Do not use directly.
&8900-&8FFF	Character definitions. Do not use directly.
&C000-&DCFF	Paged ROM workspace, claimed via a Service Call.
&DD00-&DFFF	MOS workspace. Do not use.

PCB Link settings

Fitted links

Link 1	Audio to 1 Mhz bus. Normally set for input. Can be changed to audio output.
Link 4	Allocated for advanced software use. Enables real-time clock Alarm facility to be used. Not normally fitted.
Link 12	Composite Sync output to Cartridge. Normally set to B (East). Set to A (West) for Composite Sync out to Cartridge.
Link 18	Set East for ROM socket IC 41 active. Normally set West for sideways RAM active.
Link 19	Set East for ROM socket IC 37 active. Normally set West for sideways RAM active.
Link 21	Not normally fitted. Links Light-pen strobe into the intercartridge link pin 10. Can be used for Genlock sync via LPSTB.
Link 60/61	Normally AB only linked. Link CD in addition for 8MHz output to cartridges. Link DB only (not AC) for external clock input to computer e.g. Genlock etc.

Non-fitted link positions

Link 2	Not used.
Link 5	Invert Sync output.
Link 7	Invert Video.
Link 9	IC 24 pin 22 connected to GND when 1/2 Mbit MOS fitted as in Econet Terminal machine.
Link 10	Used to select Channel 3 or 4 on VHF modulators.
Link 11	Chroma on video output (issue 2 board only).
Link 13	Change Link when fitting a different reference diode on the A/D input. This diode should be fitted at position PR1.
Link 14	Used for Divide-by-13 circuit when chroma MSI chip not fitted.
Link 15	Select PAL or NTSC TV system encoding.
Link 16	Not used.

Cartridge sockets

The cartridge socket specification is a superset of the Electron Plus-1 specification. All Electron Cartridges should work in this machine, but may run faster. The converse may not be true unless a specification sub-set is used.

APPENDIX THREE FUNCTIONAL DIFFERENCES BETWEEN MASTER 128 AND MASTER COMPACT

Hardware

Interfaces present on the Master 128 which are deleted or changed

Cassette	Connector and internal hardware deleted.
Tube	Connector and internal hardware deleted.
1MHz bus	Connector deleted.
User I/O port	Connector deleted. The internal 6522 User VIA connections to the original 20-pin connector are split as follows: Joystick/mouse - PB0-PB4 + 2 control bits. Expansion port - PB5-PB7
Disc	25-pin D-type socket. Note that there is no hardware support for a third drive.
Printer	24-pin Delta-ribbon socket.
RS 423	Now optional and RS 232 specification. The upgrade consists of plugging in four ICs (5, 9, 13 and 14).
A to D	Connector and internal hardware deleted.
Audio (external)	Connector and internal hardware deleted.
Composite Video	Monochrome only, cannot be colour.
TV	Connector and internal hardware deleted.
Cartridge sockets	Connectors deleted. Potential capability through the expansion port.
Internal modem	Internal connector deleted.
Aux power out	Connector deleted. No PSU in computer case.

Interfaces added:

Joystick/Mouse	Suitable for one digital joystick (Atari compatible) or mouse with suitable pinning. A Trackerball can also be used.
+5V DC	Power input to the computer.

Expansion Port This interface is similar, but not identical to a Master 128 Cartridge socket. It can support Sideways ROM's 0 & 1 when link PL11 is set North. A 2MHz bus is provided by this port as in the Master 128. The port must be used with care as lines are not necessarily buffered. Only a limited amount of +5V power is available, and demand should be kept below 200mA total for this connector, the RGB connector and the Joystick/Mouse port.

Functions

In addition to the functional changes implied by the interface changes mentioned above, also note:

Real-time clock Deleted.

CMOS RAM Deleted. Function replaced by an EEPROM device which does not need battery back-up. This device is socketed and has a maximum number of 1000 write cycles per location.

Sideways ROMs In addition to the system ROM, there are four 28 pin sockets. Three take 16K ROM's (ICs 23, 17, 29 - ROM number 2, 3, 8 resp.) and one takes a 16K or 32K (IC 38, ROM numbers 0 & 1). The latter socket must be enabled by setting link PL11 South. It is normally set for the "external" ROM(s) to be active for test purposes. Note that "Paged" EPROM's such as the 27513 and 27011 cannot be used.

System ROM Link PL12 is set North for a 64K ROM and South for a 128K ROM.

Links (misc)

Inverse Video	PL9 (not fitted) is normally tracked East.
Inverse Sync	PL10 (not fitted) is normally tracked East
Sound volume	VR1 (10K) may be fitted.

Expansion Port pinout

SOLDER SIDE			COMPONENT SIDE		
PIN	**M-COMPACT**	**M-128**	**PIN**	**M-COMPACT**	**M-128**
1a	SCREEN (0V)	+5V	1b	SCREEN (0V)	+5V
2a	+5V	AT13	2b	+5V	A10
3a	AT13	(neg)RST	3b	A10	CD3
4a	(neg)RST	AA15	4b	CD3	A11
5a	AA15	A8	5b	A11	A9
6a	A8	A13	6b	A9	CD7
7a	A13	A12	7b	CD7	CD6
8a	A12	PHI 2 out	8b	CD6	CD5
9a	PHI 2 out	-5V	9b	CD5	CD4
10a	N/C	(neg) CSYNC/O	10b	CD4	LPSTB
11a	N/C	BR/(neg)W	11b	LPSTB	BA7
12a	BR/(neg)W	(neg)NMI	12b	BA7	BA6
13a	(neg)NMI	(neg)IRQ	13b	BA6	BA5
14a	(neg)IRQ	(neg)INFC	14b	BA5	BA4
15a	(neg)INFC	(neg)INFD	15b	BA4	BA3
16a	(neg)INFD	AA14	16b	BA3	BA2
17a	AA14	(neg)8/16MHz	17b	BA2	BA1
18a	(neg)8MHz	CRTC(neg)RST	18b	BA1	BA0
19a	0V	ANOUT	19b	BA0	CD0
20a	PB7 USER	GND	20b	CD0	CD2
21a	PB6 USER	SPEECH	21b	CD2	CD1
22a	PB5 USER	0V	22b	CD1	0V
POLARISATION SLOT			**POLARISATION SLOT**		
24a	0V		24b	0V	
25a	SCREEN (0V)		25b	SCREEN (0V)	

Firmware

ADFS

***DRIVE** has been added to the ADFS to assist with compatibility in file conversions from DFS. *DRIVE n is equivalent to *DIR :n
As ADFS only has two drives, if n<4 it is forced to drive 4 or 5. If n>5 it is rejected. (*DRIVE should not be used in new applications.)

***COPY/*COMPACT/*BACKUP** use shadow RAM if available, and will not corrupt user workspace. If shadow RAM is not available, the utilities will first consider using unclaimed Filing system RAM, and then finally will force Mode135. The commands force *FX112,0 to avoid overwriting their own buffer.

***COMPACT** no longer takes parameters and ADFS will issue an error message to remind the user that the memory specified will not be used.

***FORMAT/*VERIFY/*BACKUP** are contained within the ROM.

***FORMAT** takes parameters <drv> <siz> where <drv> is the drive number (0 or 1, 4 or 5), and <siz> is S,M or L for 40-track, 80-track single-sided and 80-track double-sided respectively. 40-track is provided for use only where a 5.25" single-sided 40-track drive is fitted. The user must ensure that the syntax chosen is suitable for the drive type being used. The use of *FORMAT does not corrupt user workspace i.e. it uses 2 pages of utility workspace at &DD00. Sector skew is now 4 (it was 9 in the Master 128). This results in slightly faster disk performance with the 3.5" drive fitted as standard.

OSGBPB calls 6 & 7 return a zero byte after the CSD name or library name to be compatible with the ownership byte returned by the Net Filing System.

CLOSE#0 no longer produces “Channel on Channel 57” when following an EXEC sequence.

Modifications have been made to the floppy driver software in ROM which results in a noticeable speed-up in disk operation compared with the Master 128.

***CONFIGURE FDRIVE** now uses write pre-compensation on all four parameter values. This is applied to tracks 32-79 and 112-160. The four FDRIVE step rates are

0 - 6mS
1 - 12mS
2 - 2mS
3 - 3mS

The 40 track limitation which caused OSWORD &72 (*LOAD/SAVE etc) to generate an error when an attempt was made to read the last track of a 40-track disk has been removed.

The **TUBE and Winchester** support code has been removed to provide space for the utilities.

A **head settle** has been added to cover the situation when doing a *BACKUP between two 5.25" drives and the head is on the right track, the other drive has just been used and the motor is still on. A disk error 48 might otherwise be issued.

***BYE** now closes all files when in a “No directory” state.

***RENAME** wildcards are always rejected.

MOS

The Operating System is effectively compatible with that of the Master 128. All of the extended graphics features are available as for the Master 128.

The **Real-time clock** is not present, and calls to this will return a year of “1999” i.e. “Fri,31 Dec 1999.23:59:59”.

The **Configuration** system is similar for *CONFIGURE and *STATUS, but the latter lists in alpha order. References to Tube/Notube/Extube/Intube have been deleted and new keywords for the joystick have been added as follows:

SWITCHED	makes stick default to switched mode (0/&7FFF/&FFFF). Currently affects bit &20 of default *FX190 value.
PROPORTIONAL	makes stick give values in the range 0 thru &FFFF. Currently affects bit &20 of default *FX190 value.
STICK <decimal>	makes stick have speed <decimal>. Currently affects bits &1F of default *FX190 value. The default takes effect after power-up, CTRL + BREAK or BREAK.

An **EEPROM** is used instead of the Master 128 CMOS RAM. This is normally 128 bytes, but a 256-byte version may be fitted later.

OSBYTE call with A = 161, X = 255 yields the following:

Y=0	indicates no EEPROM present.
Y=&7F	128 byte EEPROM present.
Y=&FF	256 byte EEPROM present.

Writes to EEPROM address 128 using *FX162 will be ignored. A read from 128 is allowed.

The A to D port is not present, and hence analogue joysticks cannot be used. The new **digital Joystick/Mouse** port is introduced, and this is a sub-set of the previous User Port connections. The User Port is no longer present as such. The connections for this 9 way D-type connector are:

Joystick D-type pins:	**6522 connections:**
1 Up (-ve true)	(PB3)
2 Down (-ve true)	(PB2)
3 Left (-ve true)	(PB1)
4 Right (-ve true)	(PB4)
5 No joystick connection	(CB1)
6 Fire	(PB0)
7 +5V	
8 0V	
9 No joystick connection	(CB2)

On power-up, CB1 and CB2 interrupts are enabled. A sideways ROM that can process these interrupts must be present if a mouse or tracker-ball are fitted. When such an interrupt is confirmed, the sideways ROM can set the top bit of OSBYTE 190's X parameter to disable MOS processing of ADVAL values, then every clock tick, service call &2C is offered sideways. In the Y register is an offset from &0200 to the following workspace:

+0 ADVAL lo-byte	(ADVAL hi-byte from OSBYTE var 188)
+1 Xlo-coord	(x-coord returned as ADVAL1)
+2 Xhi-coord	
+3 Ylo-coord	(y-coord returned as ADVAL2)
+4 Yhi-coord	
+5 spare	
+6	

OSBYTE 188 and **189** have their normal meanings.

If the top bit of **OSBYTE 190's** X parameter is set, the MOS will not update ADVAL values from the digital joystick or cursor keys. This is designed only for external ROM's wishing to control ADVAL values e.g. mouse/trackerball software. Note that by just setting the top bit of this option, the old value may conveniently be restored by simply resetting the top bit.

***FX190,64** This option enters a key into the keyboard buffer according to bits set in ADVAL0 (lo-byte). The character "typed" is as follows (in order of priority):

bit 7	&80 (right)	cursor right
bit 6	&40 (up)	cursor up
bit 5	&20 (down)	cursor down
bit 4	&10 (left)	cursor left
bit 3	&08	delete key
bit 2	&04	return key
bit 1	&02	copy key
bit 0	&01 (fire)	copy key

The characters are typed with (almost) the same effect as typing them at the keyboard (i.e. within a centisecond or two). Auto-repeat is supported. In this mode ADVAL1 and ADVAL2 will not reflect the state of the "joystick" position. Bits &08, &04 and &02 are never set by the digital joystick, but may be set if a mouse/trackerball is supported.

***FX190,32** affects the digital joystick and ADVAL. It is designed for games that used the analogue joystick as switches and has the following effect:

ADVAL1	Xleft `&FFFF`	Xcentre `&7FFF`	Xright `&0000`
ADVAL2	Ydown `&0000`	Ycentre `&7FFF`	Yup `&FFFF`

***FX190,1 (or,2,3,4,5,6,7)** - an experimental feature by which the speed of the analogue simulation of the joystick may be adjusted.

*FX190,1 make left/right & up/down sweeps slow.
(*FX190,2 or ,3,4,5,6 are progressively faster).
*FX190,7 make left/right & up/down sweeps fast.

“Standard” settings *FX190,0, *FX190,8 and *FX190,12 use the speed selected by *FX190,3

***FX4,3**
The *FX4,3 option makes the cursor keys have joystick-like effects:

left cursor	moves joystick left
right cursor	moves joystick right
up cursor	moves joystick up
down cursor	moves joystick down
copy key	makes joystick fire

The state of the real joystick and cursor keys (in this mode) are read together. This has the primary advantage that either the real joystick or the cursor keys may be used to affect ADVAL values. When this option is selected, pressing a cursor key does not enter a code into the keyboard buffer. If a value is “poked” into the keyboard buffer, RDCH will assume the code to represent a soft key (rather like *FX4,2). If a mouse or trackerball is connected, this option has no effect (the mouse/trackerball takes priority). (See BASIC section below for ADVAL implications).

Some TUBE code has been removed, but:

The TUBE flag is accessible from **OSBYTE 234** remains and indicates NOTUBE. Service call &FE remains.

The command ***X** (which controls an external Tube splitter) has been removed.

SRAM utils and **Ellipse** code are now within the MOS ROM area. A bug with long, thin ellipses has been fixed. A facility to load an SRAM image and update the MOS ROM type table has been added. An “I” should be added to the *SRLOAD command.

***BUILD/*APPEND** now allow top bit set characters to be input.

The keyboard layout has been changed as follows:

The “@” character has been moved to the “Shifted-0” position as for the Electron. SHIFT+0 gives “@” (&40), and CTRL+0 gives NUL (&00).

The key position previously used for the "@" character is now used for "CODE" input and is marked with two squares (set vertically). The use of CTRL+SHIFT+CODE preceding any ONE key stroke will cause that key stroke character to be entered with the top-bit set. Top-bit set characters must not be used within file names.

The first call of **JSR BREAK** in the MOS to allow break indirection has been changed to preserve ROMID.

INKEY-256 now returns 245 (&F5).

***FX16,0** suppresses ADVAL support as usual, and reduces interrupt processing overhead accordingly. The default number of channels has been altered to 2.

Key interpretations set by ***FX 221 thru 228** have been extended. Options 0 and 1 of *FX 22x remain as before. The meaning of value 2 has been changed. It used to mean "use 2 as a base". It now means "return a code representing the key preceded by a NUL". For example:

After ***FX225,2** set NUL &80 means f0
After ***FX225,2** set NUL &89 means f9
After ***FX226,2** set NUL &91 means SHIFT + f1
After ***FX227,2** set NUL &A3 means CTRL + f3
After ***FX228,2** set NUL &B6 means SHIFT + CTRL + f6

Values other than 0,1 and 2 remain unchanged.
Note that when a NUL is entered at the keyboard it is supplied as NUL NUL.

This extension of the *FX calls enables applications (such as the revised version of VIEW in the machine) to continue using function keys extensively but also handle characters with the top-bit set.

Operation of the keyboard is transparent to the MOS RDCH routine. For example, pressing CTRL+0 on the keyboard results in a NUL being returned to RDCH. Similarly entering a top-bit bit code results in the return of that single byte through RDCH.

Keyboard buffer input and output changes (the use of NUL in the two operations is two completely separate uses and they should not be confused)

Keyboard buffer input rules (via *FX138 etc.):
normal codes &01 thru &7F are entered using 1 byte (as normal).
special keys (e.g. function & cursor keys) are entered using 1 byte (as normal).
extended printable codes (&80 thru &FF) are entered as 2 bytes i.e. NUL followed by extended (top-bit set) code.

A NUL must be entered using 2 bytes NUL NUL.

Note that when CTRL+0 is entered into the keyboard, the MOS automatically supplies NUL NUL to the keyboard buffer.

RDCH automatically converts codes leaving the buffer as follows:
normal codes &01 thru &7F are remove as 1 byte (as normal).
special keys have their usual special effects e.g. key expansion (as normal).

Extended printable codes are returned in a SINGLE byte. a NUL NUL is returned as a single NUL (as normal).

This means that legal calls continue to work as before, except NULs poked into the keyboard buffer may have strange effects.

When the new *FX22x,2 is in operation
special keys may expand to 2 bytes (NUL followed by &80 thru &BF).
NUL is returned as 2 bytes (NUL NUL).

Previously the VDU drivers made calls to the **user printer vector** and the **extension vector**, without allowing for the possibility that these may page-in the FSRAM. This has been corrected.

***ROMS** now indicates whether a slot is ROM or RAM.

***TAPE and *MOTOR** commands are supported, but have no effect. “-CFS-” and “-TAPE-” are not supported.

To provide for additional fonts in the future (e.g. the ISO-font), an additional parameter value 8 has been added to ***FX25** specifically to select the Master 128 font i.e:

*FX25,8 forces the Master (Series) font.
*FX25,0 continues to reselect the default font.

***TIME** will attempt to get the time from the Network if the ANFS is in use. The “day” will be filled with three SPACE characters.

A fix to prevent spurious 1770 NMIs has been added.

***BUILD/*APPEND** now allow 8-bit characters to be entered.

***SHOW** without a parameter now displays all soft keys.

BASIC

The version of BASIC fitted is IV, with improvements to accuracy and speed of transcendental functions.

ADVAL is implemented via the digital joystick port as follows:

ADVAL returns:
Hi-byte of 16-bit value - last channel to convert now totally bogus but provided for compatibility.
Lo-byte of 16-bit value - bits (msb to lsb) are:
PB4 PB3 PB2 PB1 0 0 0 FIRE

ADVAL1 ADVAL3 ADVAL5 return:
x coordinate in range &0000 thru &FFFF

ADVAL2 ADVAL4 ADVAL6 return:
y coordinate in range &0000 thru &FFFF

The x and y coordinates are supported in an Acorn-compatible fashion i.e:

Left - x value increases
Right - x value decreases
Up - y value increases
Down - y value decreases
ADC events are still supported.

TIME$ returns the dummy time “Fri,31 Dec 1999.23:59:59”, unless ANFS is present and active, in which case an attempt will be made to get the time from the Network.

*BASIC uses *FX142 to change language.

APPENDIX FOUR FUNCTIONAL DIFFERENCES BETWEEN NFS AND ANFS

The Advanced Network Filing System contains the features of the NFS with the following additions:

Local file buffering. All open files will be buffered in RAM in the I/O Processor. All uses of OSBGET and OSBPUT calls will be significantly faster. Up to 16 buffers are allocated dynamically. If only one file is open, for example, sixteen pages of data are buffered on that file.

New machine entry points for automatic retries of packets.

File-server extensions.

Extra commands in ROM

ANFS:

***CDIR <Dir> (<Number>)**	creates directory. Number can be between 1 & 245 entries, with 19 as default.
***FS (<stn. id.>)**	changes file server number. Enables user to be logged on to more than one FS.
***FLIP**	exchanges CSD and CSL. Useful when files which are to be LOADed, via OSFILE, are to be made public.
***HELP**	has sublevels ANFS & UTILS.
***LCAT (<Dir>)**	catalogue the current library, or pathname
***LEX (<Dir>)**	examine the current library, or pathname
***WIPE (<Dir>)**	deletes files or directories also.
***I AM (<stn.id.>) <user id> ((:<CR>)<password>)**	now accepts [DELETE] and [CTRL-U] during invisible input.
***PASS (:<CR>)<old password><new password>**	Now accepts a "." within the line to allow invisible passwords.

UTILS:

These now work when ANFS is the currently selected FS or not:

***POLLPS (<stn. id>:<ps type>)**	shows CSPS number and type. Also lists all network printers and their state.
***PROT (<prot type>)…**	protects against some/all ops
***PS (<stn. id.>:<ps type>)**	selects PS.
***UNPROT (<prot type>)…**	converse of PROTECT.

***WDUMP <filename> (<offset> (<address>))**

CONFIGURE:

***FS <stn. id.>**	selects File Server.
***PS <stn. id.>**	selects Printer Server.
***SPACE <number>**	moves OSHWM to use old FS utilities.
***NOSPACE <number>**	converse of SPACE.

STATUS:

***FS**	returns File Server station id.
***PS**	returns Printer Sever station id.
***SPACE**	returns “Space” or “NoSpace”.

Extra filing system interfaces:

***OPT5**	additional bootstrap by *RUNning ‘FindLib’. For FS Library compatible with this machine. CMOS RAM bit.
***OPT6 - OPT6,1**	claims &200 space instead of using &B and &C. OPT6,0 reverts to normal CMOS RAM bit.
OSFILE	with A = &7 on entry, a file is created. This behaves similarly to ‘save’ (A = &0) but no data is transferred.
OSARGS A=&FF	now functions and ensures all open files to the file server. A call with Y=0 and A=2 returns 0 to differentiate from NFS 3.nn.
OSARGS A=3	performs BASIC “EXT#channel=value”
OSARGS A=4	returns space allocated to file.
OSARGS A=&80	returns variety of file/FS info.
OSWORD A=&0E	read the time.
OSWORD A=&10	extended for zero length transmissions.
OSWORD A=&13	returns fault indication.

Extended error messages:

'**Not listening**' & '**No Reply**' now have the station number added.
'**Won't**' (&93,147) - occurs when trying to *RUN a file with load address of &FFFFFFFF or execute address which isn't &FFFFFFFF.
'**Bad parameter**' (&94,148)
'**Station <stn. id.> not present**' (&A4,164)
'**Printer busy**' (&A6,166)
'**Printer jammed**' (&A7,167)
'**Bad net sum**' (&AA,170)
'**Bad rename**' (&B0,176)
'**Outside file**' (&B7,183)
'**Write only**' (&D4,212)
'**No more FCBs**' (&C0,192)
'**Bad station number**' (&D0,208)
'**Bad net number**' (&D1,209)
'**Remoted**' (&0)
'**No!**' (&93,147) - *RUN file with load &FFFFFFxx
'**Syntax**' (&DC,220) - Recognised command but wrong syntax
'**Net channel**', '**On channel**', '**Not on this file server**' (&DE,222)
'**Bad number**' (F0,240)
'**Bad hex**' (F1,241)
'**Bad address**' (FC,252)
'**Bad string**' (FD,253)
Fatal error caused by OSWORD A=&14 function code 2 now produces '**Fatal error**'

Additional library functionality:

Any file with exec address of &FFFFFFFF that is *RUN, */<filename> or *<filename> will be *EXECd.

Screen saving. To differentiate between 'standard' and shadow screen modes, the following is used:

&FFFExxxx - Current screen RAM.
&FFFFxxxx - User RAM or non-shadow mode screens.

Auto action during Logon can now be stopped using CTRL key.

Number of "retries" is now adjustable.

'Help' extensions so that help text can be held on FS disk.

A new Master Series Level 2/3 File Server Utilities Disk is available to optimise the use of a Master Series machine on an Econet Network. (Ordering Code ADJ25)

APPENDIX FIVE CHANGES INTRODUCED IN BASIC 4

Changes from Basic 2 and Basic 3

Provides some formatting of assembly listings
`COLOR` is accepted as an alternative to `COLOUR`
`SAVE A$+B$` works correctly
Use of `!` & `?` as formal parameters works correctly
A US version listing `COLOR` instead of `COLOUR` is available

Additional changes

The version number in ROM is 4.

Incorporates all the 65C02 (65C12) instructions in Assembler:
`DEC A` may be represented as `DEA`
`INC A` may be represented as `INA` for compatibility with MASM
`STZ` may be represented as `CLR`
`ASL ALFRED` or similar is accepted i.e. the 'A' indication of accumulator addressing mode for `ASL`, `LSR`, `ROL`, `ROR`, `DEC`, `INC` no longer affects symbol recognition.
X,Y or A in the assembler may be in lower case. `EQUB`, `EQUW` & `EQUD` may also be in lower case.

Trailing spaces will always be stripped from lines entered into the interpreter.
Leading spaces will be stripped from lines entered into the interpreter when a non-zero `LISTO` is set. The assumption is that there will be a formatted listing on screen when cursor editing is used when `LISTO` is non-zero.
`LISTO` indents loops correctly.

Cross-reference/Search output is available from `LIST`. Lines will be `LIST`ed `IF` the specified string is present e.g.:

```
LIST IF DEF
LIST 10,1000 IF =
LIST ,2000 IF A%
```

It is not possible to search for `TIME=90`, for example, as a statement - it will only be checked for as a Boolean expression; `PTR#`, `HIMEM`, `LOMEM` are similarly affected.

RENUMBER or LIST will not be affected by &8D in comments or strings. In addition, LIST will not be confused by coloured comments.

A statement to update an open file's extent 'EXT#chan=length' has been added. This uses OSARGS and so will not work until there are suitable filing systems.

A display real-time clock pseudo variable TIME$ has been introduced. It fetches a fixed 24-byte string from the operating system in response to PRINT TIME$ (or similar). The string looks like Wed,31 Dec 1900.23:59:59. Assigning TIME$="fred" merely passes the string directly to the operating system with the length in the first byte.

AUTO no longer outputs a space after the line number.

General recursion is now allowed in 'FOR' loops e.g.:

```
DEF FNQ FORJ=1T010:P.J;:N.:=10
FORI=FNQ-9 TO FNQ STEP FNQ/10
```

now works. In previous versions only the first FNQ or FNQ's without the FOR loop would work.

A new command EDIT which has identical syntax to LIST (even the IF section) can be used to create an in-core text file of the current program (or section of it e.g. EDIT 10,100). It then issues the command "*EDIT hh,hh" where the hex addresses are addresses in zero page of the start of the in-memory text and the address in zero page of the end of the in-memory text plus one. LISTO 0 is set before conversion begins. If there is not enough space to convert the entire file, the error message 'No room' will be given together with a line number which shows how far through the program it had reached. At this stage either CLEAR or a different EDIT command should be used. ESCAPE will behave similarly, stopping the conversion to text.

The use of the "l" character at the end of VDU parameters can be used to insert the correct number of remaining zeros.

APPENDIX SIX
PCB SELECTION AND TEST POINTS

The printed circuit board is provided with a number of points which may be used to select different hardware configurations or to extract test signals.

Master 128

LK1 PCB track, made A: 1MHz Bus Audio Input/Output - two-position link.
In the A position the 1MHz Bus signal is an input to the computer's audio mixer.
In the B position the 1MHz Bus signal is an output from the computer's audio circuit (Minimum load 1Kohm).
This link is a permanent track in the A position. The track must be cut before a wire link is used to make the B position.

LK2 PCB track, made: Cartridge -5V decoupler - one-position link.
In some instances, particular cartridge hardware may need a -5V supply that is decoupled from the main computer -5V load. To do this R9 needs to be fitted and LK2 which is a track on the circuit board should be cut.

LK3 : Not present.

LK4 plug, not made: Clock chip IRQ - one-position link.
The 6818 clock/RAM chip has a daily alarm function built in. When the alarm is triggered, the CPU is interrupted via its IRQ line. Fitting a shunt over LK4 connects the CPU IRQ line to the clock line. This function is not supported by the operating system as this feature may not be present in future versions of the circuit board. Consequently the clock chip must be directly operated by the application software.

LK5 PCB track, made East: CSYNC polarity - two-position link.
The polarity of the composite synchronisation signal is determined by this link. It is supplied as a track on the PCB causing negative synchronisation polarity. This track must be broken and a piece of wire used to make the other side of the link for positive synchronisation.

LK6(0) and LK6(1) plug, made A B: Main Clock Select - multi-function link.
This group of 4 pins can take either one or two shorting plugs as follows:

Link between A and B - The computer main 16MHz reference is provided by on-board circuitry. This is normally how computers are shipped.
Link between B and D - The computer main 16MHz reference must be provided from pin A17 on either of the cartridge connectors. Note that in this case a clock source MUST be provided or the dynamic memories could be destroyed.
Link between C and D - The cartridges are clocked by the 8MHz signal from the computer. This is a synchronous signal with the 2MHz (ϕ2) signal, also supplied to the cartridges. Note that the link between A and B must also be fitted.

LK7 PCB track, made East: Video polarity - two-position link.
The polarity of the video RGB signals is determined by this link. It is supplied as a track on the bottom of the PCB causing true polarity. This track must be broken and a piece of wire used to make the link West for negative polarity.

LK8 : Not present.

LK9 : Not present.

LK10 fitted for NTSC only: Channel Select - two position link.
When used with NTSC VHF televisions, the modulator enables one of two channels to be selected. Note that the computer as supplied for use in the UK is fitted with a UHF modulator so LK10 is not fitted.

LK11 : Present for Issue 2 only: Chroma on Video out.

LK12 Plug, made B (East): CSYNC/Cartridge Machine Detect - two-position link.
Position A - This connection to the computer CSYNC line is provided for GENLOCK purposes.
Position B - Certain hardware cartridges may need to detect whether they are plugged into a Master Series computer or an Acorn Electron. Master computers are shipped with this link in the B position causing a logic LOW to appear on pin A10 of the cartridges. The Electron has no connection to this pin.

LK13 PCB track, made West: A to D converter reference select - two-position link.
As shipped, this link is a track on the bottom of the PCB causing the A-to-D converter reference voltage input to be 1.8V.
If the LK13 track is cut then the voltage reference must be applied between analogue ground and Vref on the external connector.
If the LK13 track is cut and LK13 made East with a wire link, a precision reference can be fitted in the position PR1 shown on the circuit diagram.

LK14 PCB track, made: Serial data clock reference - one-position link.
As shipped, this link is a track on the PCB connecting the CHROMA chip 1.23MHz output to the Serial Processor. This link is provided for production purposes and should not be modified.

LK15 PCB track, made West: PAL/NTSC select - two-position link.
As shipped in the UK, this link is a track on the bottom of the PCB causing the CHROMA chip to encode colour information on to the television output in PAL format. If the track is cut and a wire link used to make the other side of the link, then colour information will be encoded in NTSC. In general, televisions within the UK can only accept the PAL format.

LK16 wire link, not fitted:
Chrominance information luma trap bypass - one-position link.
This link is not normally fitted. It is provided for those applications where filtering of the luminance information from the chrominance part of the television signal is not required.

LK17 : Not present.

LK18 plug, made West: Paged ROM/RAM Select - two-position link.
When fitted in the West position, this link causes 16Kbyte of RAM to appear in each of the “sideways” memory “slots” 4 and 5.
When fitted in the East position, a 32Kbyte ROM occupying slots 4 and 5 may be plugged into socket labelled IC41.

LK19 plug, made West: Paged ROM/RAM Select - two-position link.
When fitted in the West position, this link causes 16Kbyte of RAM to appear in each of the “sideways” memory “slots” 6 and 7.
When fitted in the East position, a 32Kbyte ROM occupying slots 6 and 7 may be plugged into socket labelled IC37.

LK20 : Not present.

LK21 plug, not made: Light Pen Strobe to cartridge.
This link is not normally made, so position B10 on the cartridges is merely a connection from one to the other. When the shunt is fitted. the CRTC Light Pen Strobe input is connected to B10. This is to facilitate GENLOCK and an alternative LPSTB connection to the rear analogue connector.

Master Compact

TP1 - MAX232 -ve output.
If the serial interface is fitted, the voltage on this pin should be between -10v and -5v. A figure of -9v is quite typical.

TP2 - MAX232 +ve output.
If the serial interface is fitted, the voltage on this pin should be between 5v and 10v. A figure of 9v is quite typical.

Test points TP1 and TP2 are positioned close to IC5 (North of the PCB).

TP3 - connected to the CPU NMI pin.
This should be generally at 5v while running, making excursions to 0v only when disc and Econet are being used.

TP4 - connected to the CPU IRQ pin.
Check that this is not stuck either high or low when free running.

TP5 - connected to the CPU SYNC pin.
This is asserted during an op-code fetch by the processor, and is used by ACCCON to ensure that the correct memory area is accessed at this time. If this is continuously high or low, then the processor has completely stalled.

TP6 - This is connected to the processor READ/WRITE line.
This should change between 0v and 5v frequently (but not necessarily regularly!)

Test points TP3 to TP6 are situated South of the CPU IC28 (65C12) to the southeast of the PCB.

PL7 - Not fitted
allows the light pen strobe (LPTSTB) to be connected to the CRTC IC.

PL9 - pcb track made north
If set North, the video output will be normal, if set South the video output will be inverted. If change is required, cut circuit board track, and either use tinned copper wire, or fit three pins, and select the required position using a mini shunt.

PL10 - pcb track made east
If set East, the RGB CSYNC signal will be inverted. If set West, it will be non-inverted. This is necessary for certain monitors. If change is required, cut circuit board track, and either use tinned copper wire, or fit three pins, and select the required position using a mini shunt.

PL11 - plug made north
If set North, 32k ROM space banks 0 and 1 are assigned to the edge connector. If set South, 32k ROM space banks 0 and 1 are assigned to IC38.

PL12 - plug made north
If set North, allows system ROM containing 64k bytes of code. If set South, allows ROM containing 128k bytes. Factory position is currently NORTH, but may change to SOUTH in future production.

Circuit board modifications necessary for fitting optional components.

VR1
If a volume control is required for the loudspeaker, a preset potentiometer VR1 may be fitted. If this modification is done, first cut the circuit board track joining two pins of VR1.

FS1
A fuse (FS1) may be fitted if required, first cut the track under FS1 on the PCB.

L1/L2
If further filtering (L1 and L2) is used, the tracks under L1 and L2 on the main PCB must be cut.

APPENDIX SEVEN
THE MASTER 128
CARTRIDGE INTERFACE

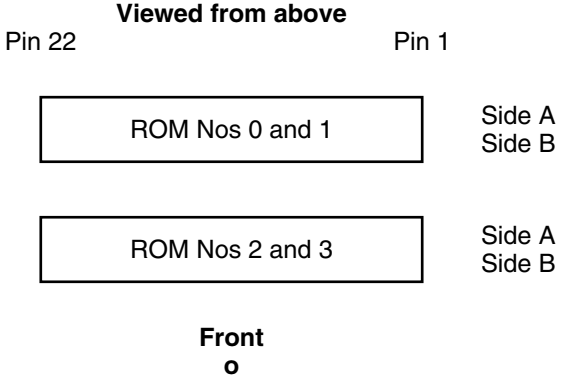
The Master Series Cartridge Interface is an enhancement of that of the Electron Plus 1. The connections and any differences are noted below.

Abbreviations used in this Appendix are as follows:

A/L	Active low.
O/C	Open Collector output.
CMOS	Complementary Metal Oxide Semiconductor.
CPU	Central Processor Unit i.e. the microprocessor.
TTL	Transistor-Transistor Logic.
&	A hexadecimal number follows.
n	As a signal prefix means Active low output (A/L).
PCB	Printed Circuit Board.
NMOS	N-channel Metal Oxide Semiconductor

Cartridge Orientation

The cartridge pinning in the Master Series machine is arranged as follows:



Components are normally mounted on to Side A of the PCB within the cartridge.

Pinout

Pins are described viewed from “within” the cartridge i.e. an “input” is an input to the cartridge. An “output” is an output to the computer.

Side A

1 **+5V - logic power supply**
150mA max in a Master with co-processor fitted and with disc drives.
50mA max in an Electron Plus 1.

2 **nOE - Output Enable** **Input from A/L CMOS level.**
low during phi2 period of system clock. It is intended to switch on the output buffers of cartridge memory devices. It is not guaranteed low at other times.

3 **nRST - System Reset** **Input from A/L CMOS level.**
low during a system reset. It is not synchronised to any clock.

4 **CSRW - chip select - Read/Write** **Input from CMOS level.**
Master
Changes function according to the memory region that the CPU is addressing. During accesses to &FC00 thru &FEFF it is equivalent to the CPU Read/Write line during phi2. For all other accesses, it is an Active High chip select for memory devices. It is not guaranteed low at other times.
Electron
CPU Read/Write line.

5 **A8 - Address line 8** **Input from TTL level.**

6 **A13 - Address line 13** **Input from TTL level.**

7 **A12 - Address line 12** **Input from TTL level.**

8 **phi2 - CPU clock** **Input from CMOS levels.**
computer’s phi2 output.

9 **-5V - Negative supply voltage.**
20mA max. This -5V may not be available on all Acorn Cartridge Interfaces. To ensure compatibility, negative voltages should be generated within the Cartridge if required.

10 **CSYNC/MADET**
Master
There are two functions dependent upon link 12 in the computer:
E/nB - the default function. It enables Cartridges to know which machine they are plugged into. It is connected to 0V in the Master, (and unconnected in the Electron). Link 12 is set to position B.
CSYNC - Composite Sync. **Input from TTL levels.**
System Vertical & Horizontal sync is made available for Genlock use. Set Link 12 to position A.
Electron
Unconnected

11 RnW/READY

Master

R/W - Data Direction control Input from TTL levels.

System data buffer direction control. If low, cartridges are being written to; if high and selected, they may drive the bus during phi2.

Electron

READY - CPU wait state control O/C A/L output.

When driven low, this line will cause the CPU to extend its cycle until READY is released. Only works with CMOS CPUs and only on READ cycles.

12 nNMI - Non-maskable Interrupt O/C A/L output.

Connected to system NMI line.

13 nIRQ - Interrupt Request O/C A/L output.

Connected to the system IRQ line.

14 nINFC - Internal Page &FC Input from TTL levels.

A/L. Memory Active decode input.

Master

When bit IFJ in the Master ACCON register (via &FE34) is set, all accesses to &FC00 thru &FCFF will cause this select to become active.

Electron

Not applicable.

15 nINFD - Internal Page &FD Input from TTL levels.

A/L. Memory Active decoded input.

Master

When bit IFJ in the Master ACCCON register (via &FE34) is set, all accesses to &FD00 thru &FDFF will cause this select to become active.

Electron

Not applicable.

16 ROMQA - Memory paging select Input from TTL levels.

This is the least significant bit of the ROM select latch located at &FE30 in the Master, and &FE05 in the Electron.

17 Clock Input/Output TTL levels.

Master

Links 6(0) and 6(1) on the computer select one of two functions:

a) 16Mhz output to computer (Link DB only).

b) 8Mhz Input to cartridge (Link CD in addition to AB).

The user should ensure that the links are set correctly, and that there is proper termination. Normally only AB is linked in the computer.

Electron

16MHz Input.

18 nROMSTB/nCRTCRST TTL levels.

Master

nCRTCRST is an Active Low Output signal of the system CRTC reset input. It is provided for Genlock use.

Electron
nROMSTB is an Active Low Input which selects &FC73. It is intended to be used as a Paging Register.

19 ADOUT - System audio Output.
Filtered output of the sum of all audio inputs to the computer. No significant load should be taken from this pin.

20 AGND - Audio Ground.
The zero volt return for ADOUT. It should be used instead of system 0V to minimise audio noise.

21 ADIN - System audio input.
Master
An input to the computer's audio circuitry. It presents an impedance of at least 1K ohm. Only one cartridge using this input should be connected to the computer at one time.
Electron
This is a connection from one cartridge to the other.

22 0V - Zero volts.
System earth return for digital signals.

Side B

1 +5V - Logic power supply
150mA max in a Master with Co-processor fitted and with disc drives.
10mA max in an Electron Plus 1.

2	**A10 - Address line 10**	**Input from TTL levels.**
3	**D3 - Data bus line 3**	**Input/Output TTL levels.**
4	**A11 - Address line 11**	**Input from TTL levels.**
5	**A9 - Address line 9**	**Input from TTL levels.**
6	**D7 - Data bus line 7**	**Input/Output TTL levels.**
7	**D6 - Data bus line 6**	**Input/Output TTL levels.**
8	**D5 - Data bus line 5**	**Input/Output TTL levels.**
9	**D4 - Data bus line 4**	**Input/Output TTL levels.**

10 nOE2/LPSTB - O/P Enable/Light Pen Strobe Input from TTL levels.
Master
With link 21 removed in the computer, this pin provides a connection between the two cartridges. With the link in place, the pin forms a connection to a pull-up resistor in the computer to +5V. The connection is also made to the CRTC Light-Pen Strobe and interrupt structure.
Electron
This provides an additional A/L enable for ROMs in the Electron. This corresponds to ROM position 13 and responds quickly to Service Calls. It is low during the A/L portion of phi2. It is not guaranteed high at other times.

11 BA7 - Buffered address line 7 Input from TTL levels.

Master

This line holds addresses valid for 125nS after phi2 goes low.

Electron

This is not buffered nor held valid for an extended period.

12 BA6 - Buffered address line 6 Input from TTL levels.

See pin 11.

13 BA5 - Buffered address line 5 Input from TTL levels.

See pin 11.

14 BA4 - Buffered address line 4 Input from TTL levels.

See pin 11.

15 BA3 - Buffered address line 3 Input from TTL levels.

See pin 11.

16 BA2 - Buffered address line 2 Input from TTL levels.

See pin 11.

17 BA1 - Buffered address line 1 Input from TTL levels.

See pin 11.

18 BA0 - Buffered address line 0 Input from TTL levels.

See pin 11.

19 D0 - Data bus line 0 Input/Output TTL levels.

20 D2 - Data bus line 2 Input/Output TTL levels.

21 D1 - Data bus line 1 Input/Output TTL levels.

22 0V - Zero volts

Digital signal Earth return.

APPENDIX EIGHT
65C12 INSTRUCTION SET

This appendix lists each 65C12 instruction on a separate page along with details of the status flags affected and a brief description.

A number of new mnemonics which do not exist on the 6502 are provided on the 65C12 which also has one new addressing mode called “(indirect zero page)”. This is similar to “(indirect,X)” and “(indirect),Y” but does not require the X or Y registers to be set to zero.

The new 65C12 mnemonics are:

BRA	Branch always
CLR	Clear memory (also STZ)
DEA	Decrement accumulator
INA	Increment accumulator
PHX	Push X register onto stack
PHY	Push Y register onto stack
PLX	Pull X register from stack
PLY	Pull Y register from stack
STZ	Clear memory (also CLR)
TRB	Test and reset bits
TSB	Test and set bits

The Rockwell R65C02, which is normally fitted within the 6502 and Turbo Second processors, has two instructions which do not exist on the 65C12 and which have to be assembled by hand.

BBR	Branch on bit reset
BBS	Branch on bit set

In the tables listing the various op.codes the time taken to execute each instruction is given as a number of cycles. Each cycle represents:

0.5µ5 on a BBC model B
0.33µs on a Master or 6502 Second Processor
0.25µs on a Master Turbo Co-processor.

ADC

7	Status Register						0
N	V		B	D	I	Z	C
✓	✓	-	-	-	-	✓	✓

Add to Accumulator with Carry

Operation

A,C = A + M + C

Description

Adds the contents of a memory location to the accumulator. If the carry flag is set then 1 is also added. If the result overflows then the carry flag will be set, allowing multiple byte addition.

Op.Code	Addressing Mode	Assembly Lang.		No.Bytes	No.Cycles
&69	Immediate	ADC	#dd	2	2
&65	Zero Page	ADC	aa	2	3
&75	Zero Page,X	ADC	aa,X	2	4
&72	(Indirect Zero Page)	ADC	(aa)	2	5
&6D	Absolute	ADC	aaaa	3	4
&7D	Absolute,X	ADC	aaaa,X	3	4*
&79	Absolute,Y	ADC	aaaa,Y	3	4*
&61	(Indirect,X)	ADC	(aa,X)	2	6
&71	(Indirect),Y	ADC	(aa),Y	2	5*

* Add 1 cycle if page boundary crossed
** Add 1 cycle for all addressing modes if in decimal mode

AND

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	-

AND Memory with Accumulator

Operation

A = A AND M

Description

A logical AND is performed between the accumulator and a memory location. The result is then left in the accumulator.

Op.Code	Addressing Mode	Assembly Lang.		No.Bytes	No.Cycles
&29	Immediate	AND	#dd	2	2
&25	Zero Page	AND	aa	2	3
&35	Zero Page,X	AND	aa,X	2	4
&32	(Indirect Zero Page)	AND	(aa)	2	5
&2D	Absolute	AND	aaaa	3	4
&3D	Absolute,X	AND	aaaa,X	3	4*
&39	Absolute,Y	AND	aaaa,Y	3	4*
&21	(Indirect,X)	AND	(aa,X)	2	6
&31	(Indirect),Y	AND	(aa),Y	2	5*

* Add 1 cycle if page boundary crossed

ASL

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	✓

Accumulator Shift Left

Operation

$C = M_7$, $M = M * 2$

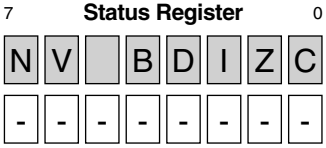
Description

Shifts the contents of a memory location or the accumulator one bit to the left. This operation effectively multiplies by two and leaves any overflow in the carry flag.

Op.Code	Addressing Mode	Assembly Lang.		No.Bytes	No.Cycles
&0A	Accumulator	ASL	A	1	2
&06	Zero Page	ASL	aa	2	5
&16	Zero Page,X	ASL	aa,X	2	6
&0E	Absolute	ASL	aaaa	3	6
&1E	Absolute,X	ASL	aaaa,X	3	6*

* Add 1 cycle if page boundary crossed
Always takes 7 cycles on 6502A

BBR



Branch on Bit Reset

Operation

Branch if bit = 0

Description

BBR is not normally available but does exist on the Rockwell R65C02 which is usually fitted within the Master Turbo and 6502 Second Processors. If a bit in a zero page location is clear a branch will occur.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&0F	Zero page, bit 0	BBR0 aa	3	5*
&1F	Zero page, bit 1	BBR1 aa	3	5*
&2F	Zero page, bit 2	BBR2 aa	3	5*
&3F	Zero page, bit 3	BBR3 aa	3	5*
&4F	Zero page, bit 4	BBR4 aa	3	5*
&5F	Zero page, bit 5	BBR5 aa	3	5*
&6F	Zero page, bit 6	BBR6 aa	3	5*
&7F	Zero page, bit 7	BBR7 aa	3	5*

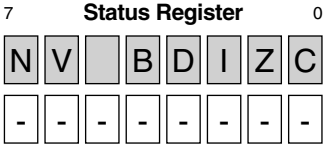
* Add 1 cycle if branch crossed a page boundary

This instruction is not available in the BASIC assembler and will have to be inserted using EQUB.

Example: Branch if bit 5 of zero page location &70 is 0 (reset)

```
EQUB &5F      \ BBR op.code for bit 5
EQUB &70      \ zero page &70
EQUB &09      \ branch forward 9 bytes
```

BBS



Branch on Bit Set

Operation

Branch if bit = 1

Description

BBS is not normally available but does exist on the Rockwell R65C02 which is usually fitted within the Master Turbo and 6502 Second Processors. If a bit in a zero page location is set a branch will occur.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&8F	Zero page, bit 0	BBS0 aa	3	5*
&9F	Zero page, bit 1	BBS1 aa	3	5*
&AF	Zero page, bit 2	BBS2 aa	3	5*
&BF	Zero page, bit 3	BBS3 aa	3	5*
&CF	Zero page, bit 4	BBS4 aa	3	5*
&DF	Zero page, bit 5	BBS5 aa	3	5*
&EF	Zero page, bit 6	BBS6 aa	3	5*
&FF	Zero page, bit 7	BBS7 aa	3	5*

* Add 1 cycle if branch crossed a page boundary

This instruction is not available in the BASIC assembler and will have to be inserted using EQUB.

Example: Branch if bit 5 of zero page location &70 is 1 (set)

```
EQUB &DF      \ BBS op.code for bit 5
EQUB &70      \ zero page &70
EQUB &09      \ branch forward 9 bytes
```

BCC

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	-	-

Branch on Carry Clear

Operation

Branch if Carry flag = 0

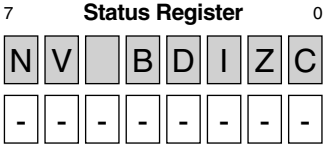
Description

If the carry flag is clear this instruction performs a relative jump forwards or backwards a specific number of bytes from the next instruction. This relative figure is a two's complement signed number which can span up to 127 bytes forward, or 128 bytes backward.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&90	Relative	BCC aa	2	2*

* Add 1 cycle if branch occurs or
Add 2 cycles if branch crossed a page boundary

BCS



Branch on Carry Set

Operation

Branch if Carry flag = 1

Description

If the carry flag is set this instruction performs a relative jump forwards or backwards a specific number of bytes from the next instruction. This relative figure is a two's complement signed number which can span up to 127 bytes forward, or 128 bytes backward.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&B0	Relative	BCS aa	2	2*

* Add 1 cycle if branch occurs or
Add 2 cycles if branch crossed a page boundary

BEQ

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	-	-

Branch on Result Equal to Zero

Operation

Branch if Zero flag = 1

Description

If the zero flag is set this instruction performs a relative jump forwards or backwards a specific number of bytes from the next instruction. This relative figure is a two's complement signed number which can span up to 127 bytes forward, or 128 bytes backward.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&F0	Relative	BEQ aa	2	2*

* Add 1 cycle if branch occurs or
Add 2 cycles if branch crossed a page boundary

BIT

7	Status Register						0
N	V		B	D	I	Z	C
M_7	M_6	-	-	-	-	✓	-

Test Bits in Memory with Accumulator

Operation

A AND M, $N = M_7$, $V = M_6$

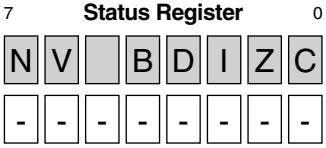
Description

This instruction is used to test whether various bits are set in a memory location by performing an AND instruction. It does not however effect either the accumulator of the memory location, but just sets the status flags. Also bits 7 and 6 are transferred to the N and V flags respectively.

Op.Code	Addressing Mode	Assembly Lang.		No.Bytes	No.Cycles
&89*	Immediate*	BIT	#dd	2	2
&24	Zero Page	BIT	aa	2	3
&34*	Zero Page,X*	BIT	aa,X	2	4
&2C	Absolute	BIT	aaaa	3	4
&3C*	Absolute,X*	BIT	aaaa,X	3	4

* New op. codes for 65C12 only, which is fitted in the Master.

BMI



Branch on Result Minus

Operation

Branch if Negative flag = 1

Description

If the negative flag is set this instruction performs a relative jump forwards or backwards a specific number of bytes from the next instruction. This relative figure is a two's complement signed number which can span up to 127 bytes forward, or 128 bytes backward.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&30	Relative	BMI aa	2	2*

* Add 1 cycle if branch occurs or
Add 2 cycles if branch crossed a page boundary

BNE

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	-	-

Branch on Result Not Equal to Zero

Operation

Branch if Zero flag = 0

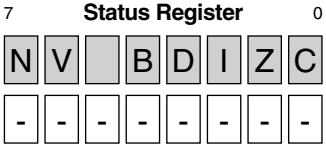
Description

If the zero flag is clear this instruction performs a relative jump forwards or backwards a specific number of bytes from the next instruction. This relative figure is a two's complement signed number which can span up to 127 bytes forward, or 128 bytes backward.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&D0	Relative	BNE aa	2	2*

* Add 1 cycle if branch occurs or
Add 2 cycles if branch crossed a page boundary

BPL



Branch on Result Plus

Operation

Branch if Negative flag = 0

Description

If the negative flag is clear this instruction performs a relative jump forwards or backwards a specific number of bytes from the next instruction. This relative figure is a two's complement signed number which can span up to 127 bytes forward, or 128 bytes backward.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&10	Relative	BPL aa	2	2*

* Add 1 cycle if branch occurs or
Add 2 cycles if branch crossed a page boundary

BRA

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	-	-

Branch Always

Operation

Branch Always

Description

This instruction always performs a relative jump forwards or backwards a specific number of bytes from the next instruction. This relative figure is a two's complement signed number which can span up to 127 bytes forward, or 128 bytes backward.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&80	Relative	BRA aa	2	3*

* Add 1 cycle if branch crossed a page boundary

BRK

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	1	0	1	-	-

Force Break

Operation

Push PC+2 and P on stack, PC=(&FFFE)

Description

This instruction forces a break which causes the program counter to be pushed onto the stack along with the status register. The program counter is then set to the contents of &FFFE/F. The BRK instruction is usually used for errors.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&00	Implied	BRK	1	7

BVC

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	-	-

Branch on Overflow Clear

Operation

Branch if Overflow flag = 0

Description

If the overflow flag is clear this instruction performs a relative jump forwards or backwards a specific number of bytes from the next instruction. This relative figure is a two's complement signed number which can span up to 127 bytes forward, or 128 bytes backward.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&50	Relative	BVC aa	2	2*

* Add 1 cycle if branch occurs or
Add 2 cycles if branch crossed a page boundary

BVS

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	-	-

Branch on Overflow Set

Operation

Branch if Overflow flag = 1

Description

If the overflow flag is set this instruction performs a relative jump forwards or backwards a specific number of bytes from the next instruction. This relative figure is a two's complement signed number which can span up to 127 bytes forward, or 128 bytes backward.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&70	Relative	BVC aa	2	2*

* Add 1 cycle if branch occurs or
Add 2 cycles if branch crossed a page boundary

CLC

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	-	0

Clear Carry Flag

Operation

Carry flag = 0

Description

This instruction clears the carry flag and is mainly used to prepare for ADC or SBC.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&18	Implied	CLC	1	2

CLD

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	0	-	-	-

Clear Decimal Mode

Operation

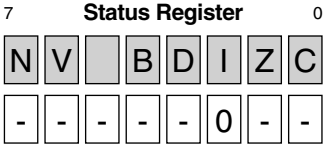
Decimal flag = 0

Description

This instruction switches the 65C12 back to normal binary arithmetic mode.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&D8	Implied	CLD	1	2

CLI



Clear Interrupt Disable Bit

Operation

Interrupt flag = 0

Description

When maskable interrupts have been disabled using SEI, this instruction re-enables them.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&58	Implied	CLI	1	2

CLR

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	-	-

Clear Memory

Operation

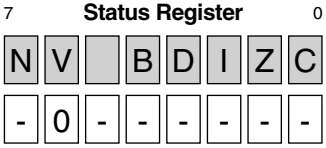
M = 0

Description

CLR clears a byte of memory by storing zero at the specified location. STZ is an alternative mnemonic.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&64	Zero Page	CLR aa	2	3
&74	Zero Page,X	CLR aa,X	2	4
&9C	Absolute	CLR aaaa	3	4
&9E	Absolute,X	CLR aaaa,X	3	5

CLV



Clear Overflow Flag

Operation

Overflow flag = 0

Description

This instruction clears the overflow flag.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&B8	Implied	CLV	1	2

CMP

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	✓

Compare Memory and Accumulator

Operation

A - M

Description

CMP subtracts the contents of a memory location from the accumulator and sets the status flags without actually affecting the contents of the accumulator. See table below for results of compare.

Op.Code	Addressing Mode	Assembly Lang.		No.Bytes	No.Cycles
&C9	Immediate	CMP	#dd	2	2
&C5	Zero Page	CMP	aa	2	3
&D5	Zero Page,X	CMP	aa,X	2	4
&D2	(Indirect Zero Page)	CMP	(aa)	2	5
&CD	Absolute	CMP	aaaa	3	4
&DD	Absolute,X	CMP	aaaa,X	3	4*
&D9	Absolute,Y	CMP	aaaa,Y	3	4*
&C1	(Indirect,X)	CMP	(aa,X)	2	6
&D1	(Indirect),Y	CMP	(aa),Y	2	5*

* Add 1 cycle if page boundary crossed

After a CMP instruction the following conditions will apply:

A < M	N = 1*	Z = 0	C = 0
A = M	N = 0	Z = 1	C = 1
A > M	N = 0*	Z = 0	C = 1

* Only valid for “two’s complement” compare

CPX

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	✓

Compare Memory and X Register

Operation

X - M

Description

CPX subtracts the contents of a memory location from the X register and sets the status flags without actually affecting the contents of the X register. See table below for results of compare.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&E0	Immediate	CPX #dd	2	2
&E4	Zero Page	CPX aa	2	3
&EC	Absolute	CPX aaaa	3	4

After a CPX instruction the following conditions will apply:

X < M	N = 1*	Z = 0	C = 0
X = M	N = 0	Z = 1	C = 1
X > M	N = 0*	Z = 0	C = 1

* Only valid for “two’s complement” compare

CPY

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	✓

Compare Memory and Y Register

Operation

Y - M

Description

CPY subtracts the contents of a memory location from the Y register and sets the status flags without actually affecting the contents of the Y register. See table below for results of compare.

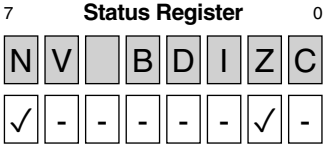
Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&C0	Immediate	CPY #dd	2	2
&C4	Zero Page	CPY aa	2	3
&CC	Absolute	CPY aaaa	3	4

After a CPY instruction the following conditions will apply:

Y < M	N = 1*	Z = 0	C = 0
Y = M	N = 0	Z = 1	C = 1
Y > M	N = 0*	Z = 0	C = 1

* Only valid for “two’s complement” compare

DEC / DEA



Decrement Memory by One

Operation

M = M - 1

Description

This instruction subtracts one from a memory location and sets the appropriate status flags. The additional addressing mode on the 65C12 allows the accumulator to be decremented by using DEC A or just DEA.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&3A*	Accumulator*	DEC A (DEA)	1	2
&C6	Zero Page	DEC aa	2	5
&D6	Zero Page,X	DEC aa,X	2	6
&CE	Absolute	DEC aaaa	3	6
&DE	Absolute,X	DEC aaaa,X	3	7

* New op. code for 65C12 only, which is fitted in the Master.

DEX

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	-

Decrement X Register by One

Operation

X = X - 1

Description

This instruction subtracts one from the X register and sets the appropriate status flags.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&CA	Implied	DEX	1	2

DEY

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	-

Decrement Y Register by One

Operation

Y = Y - 1

Description

This instruction subtracts one from the Y register and sets the appropriate status flags.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&88	Implied	DEY	1	2

EOR

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	-

Exclusive-OR Memory with Accumulator

Operation

A = A EOR M

Description

This instruction performs an Exlusive OR between the accumulator and a memory location leaving the result in the accumulator.

Op.Code	Addressing Mode	Assembly Lang.		No.Bytes	No.Cycles
&49	Immediate	EOR	#dd	2	2
&45	Zero Page	EOR	aa	2	3
&55	Zero Page,X	EOR	aa,X	2	4
&52	(Indirect Zero Page)	EOR	(aa)	2	5
&4D	Absolute	EOR	aaaa	3	4
&5D	Absolute,X	EOR	aaaa,X	3	4*
&59	Absolute,Y	EOR	aaaa,Y	3	4*
&41	(Indirect,X)	EOR	(aa,X)	2	6
&51	(Indirect),Y	EOR	(aa),Y	2	5*

* Add 1 cycle if page boundary crossed

INC / INA

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	-

Increment Memory by One

Operation

M = M + 1

Description

This instruction adds one to a memory location and sets the appropriate status flags. The additional addressing mode on the 65C12 allows the accumulator to be incremented by using INC A or just INA.

Op.Code	Addressing Mode	Assembly Lang.		No.Bytes	No.Cycles
&1A*	Accumulator*	INC	A (INA)	1	2
&E6	Zero Page	INC	aa	2	5
&F6	Zero Page,X	INC	aa,X	2	6
&EE	Absolute	INC	aaaa	3	6
&FE	Absolute,X	INC	aaaa,X	3	7

* New op. code for 65C12 only, which is fitted in the Master.

INX

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	-

Increment X Register by One

Operation

X = X + 1

Description

This instruction adds one to the X register and sets the appropriate status flags.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&E8	Implied	INX	1	2

INY

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	-

Increment Y Register by One

Operation

Y = Y + 1

Description

This instruction adds one to the Y register and sets the appropriate status flags.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&C8	Implied	INY	1	2

JMP

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	-	-

Jump to new location

Operation

PC = new location

Description

This instruction jumps to the specified location by loading the new address into the program counter.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&4C	Absolute	JMP aaaa	3	3
&6C	(Indirect)	JMP (aaaa)	3	6
&7C*	(Indirect,X)*	JMP (aa,X)	3	6

* New op. code for 65C12 only, which is fitted in the Master.

JSR

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	-	-

Jump to Subroutine

Operation

Push PC+2 on stack, PC = new location

Description

This instruction is similar to JMP but first pushes the current program counter plus 2 onto the stack. When a RTS instruction is encountered the program counter is then reset using the location that was previously stored on the stack.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&20	Absolute	JSR aaaa	3	6

LDA

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	-

Load Accumulator with Memory

Operation

A = M

Description

This instruction loads the accumulator with the contents of a specified byte of memory.

Op.Code	Addressing Mode	Assembly Lang.		No.Bytes	No.Cycles
&A9	Immediate	LDA	#dd	2	2
&A5	Zero Page	LDA	aa	2	3
&B5	Zero Page,X	LDA	aa,X	2	4
&B2	(Indirect Zero Page)	LDA	(aa)	2	5
&AD	Absolute	LDA	aaaa	3	4
&BD	Absolute,X	LDA	aaaa,X	3	4*
&B9	Absolute,Y	LDA	aaaa,Y	3	4*
&A1	(Indirect,X)	LDA	(aa,X)	2	6
&B1	(Indirect),Y	LDA	(aa),Y	2	5*

* Add 1 cycle if page boundary crossed

LDX

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	-

Load X Register with Memory

Operation

X = M

Description

This instruction loads the X register with the contents of a specified byte of memory.

Op.Code	Addressing Mode	Assembly Lang.		No.Bytes	No.Cycles
&A2	Immediate	LDX	#dd	2	2
&A6	Zero Page	LDX	aa	2	3
&B6	Zero Page,Y	LDX	aa,Y	2	4
&AE	Absolute	LDX	aaaa	3	4
&BE	Absolute,Y	LDX	aaaa,Y	3	4*

* Add 1 cycle if page boundary crossed

LDY

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	-

Load Y Register with Memory

Operation

Y = M

Description

This instruction loads the Y register with the contents of a specified byte of memory.

Op.Code	Addressing Mode	Assembly Lang.		No.Bytes	No.Cycles
&A0	Immediate	LDY	#dd	2	2
&A4	Zero Page	LDY	aa	2	3
&B4	Zero Page,X	LDY	aa,X	2	4
&AC	Absolute	LDY	aaaa	3	4
&BC	Absolute,X	LDY	aaaa,X	3	4*

* Add 1 cycle if page boundary crossed

LSR

7	Status Register						0
N	V		B	D	I	Z	C
0	-	-	-	-	-	✓	✓

Logical Shift Right

Operation

$C = M_0$, $M = M / 2$

Description

Shift the contents of a memory location or the accumulator one bit to the right. This operation effectively divides by two and leaves any remainder in the carry flag.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&4A	Accumulator	LSR A	1	2
&46	Zero Page	LSR aa	2	5
&56	Zero Page,X	LSR aa,X	2	6
&4E	Absolute	LSR aaaa	3	6
&5E	Absolute,X	LSR aaaa,X	3	6*

* Add 1 cycle if page boundary crossed

NOP

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	-	-

No Operation

Operation

No operation

Description

This is an instruction which has no effect other than to use up a memory location and take 2 cycles. It may be used to reserve space or to replace redundant code without having to re-assemble.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&EA	Implied	NOP	1	2

ORA

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	-

OR Memory with Accumulator

Operation

A = A OR M

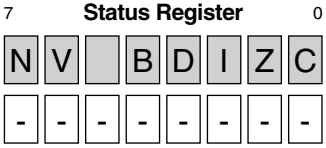
Description

A logical OR is performed between the accumulator and a memory location. The result is then left in the accumulator.

Op.Code	Addressing Mode	Assembly Lang.		No.Bytes	No.Cycles
&09	Immediate	ORA	#dd	2	2
&05	Zero Page	ORA	aa	2	3
&15	Zero Page,X	ORA	aa,X	2	4
&12	(Indirect Zero Page)	ORA	(aa)	2	5
&0D	Absolute	ORA	aaaa	3	4
&1D	Absolute,X	ORA	aaaa,X	3	4*
&19	Absolute,Y	ORA	aaaa,Y	3	4*
&01	(Indirect,X)	ORA	(aa,X)	2	6
&11	(indirect),Y	ORA	(aa),Y	2	5*

* Add 1 cycle if page boundary crossed

PHA



Push Accumulator onto Stack

Operation

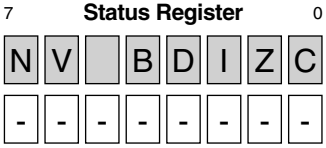
Push Accumulator

Description

This instruction pushes the contents of the accumulator onto the stack.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&48	Implied	PHA	1	3

PHP



Push Processor Status onto Stack

Operation

Push Status register (P)

Description

This instruction pushes the contents of the status register onto the stack.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&08	Implied	PHP	1	3

PHX

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	-	-

Push X Register onto Stack

Operation

Push X register

Description

This instruction pushes the contents of the X register onto the stack.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&DA	Implied	PHX	1	3

PHY

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	-	-

Push Y Register onto Stack

Operation

Push Y register

Description

This instruction pushes the contents of the Y register onto the stack.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&5A	Implied	PHY	1	3

PLA

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	-

Pull Accumulator from Stack

Operation

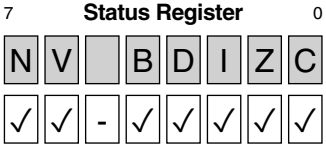
Pull Accumulator

Description

This instruction pulls a value from the stack into the accumulator.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&68	Implied	PLA	1	4

PLP



Pull Processor Status from Stack

Operation

Pull Status register (P)

Description

This instruction pulls a value from the stack into the status register.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&28	Implied	PLP	1	4

PLX

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	-

Pull X Register from Stack

Operation

Pull X register

Description

This instruction pulls a value from the stack into the X register.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&FA	Implied	PLX	1	4

PLY

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	-

Pull Y Register from Stack

Operation

Pull Y register

Description

This instruction pulls a value from the stack into the Y register.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&7A	Implied	PLY	1	4

ROL

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	✓

Rotate Left

Operation

$C = M_7$, $M = M * 2$, $M_0 = C$

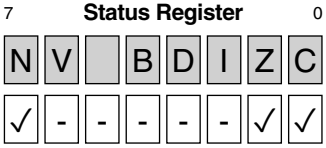
Description

Rotate the contents of a memory location or the accumulator one bit to the left.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&2A	Accumulator	ROL A	1	2
&26	Zero Page	ROL aa	2	5
&36	Zero Page,X	ROL aa,X	2	6
&2E	Absolute	ROL aaaa	3	6
&3E	Absolute,X	ROL aaaa,X	3	6*

* Add 1 cycle if page boundary crossed

ROR



Rotate Right

Operation

$C = M_0$, $M = M / 2$, $M_7 = C$

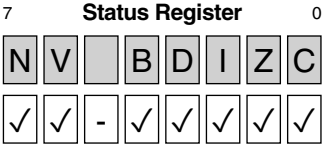
Description

Rotate the contents of a memory location or the accumulator one bit to the right.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&6A	Accumulator	ROR A	1	2
&66	Zero Page	ROR aa	2	5
&76	Zero Page,X	ROR aa,X	2	6
&6E	Absolute	ROR aaaa	3	6
&7E	Absolute,X	ROR aaaa,X	3	6*

* Add 1 cycle if page boundary crossed

RTI



Return from Interrupt

Operation

Pull Status register (P), pull Program counter (PC)

Description

This instruction pulls both P and PC from the stack on return from an interrupt.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&40	Implied	RTI	1	6

RTS

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	-	-

Return from Subroutine

Operation

Pull Program counter (PC)

Description

This instruction is used in conjunction with JSR to terminate a subroutine. RTS pulls into the program counter the values pushed by JSR from the stack. Execution is then resumed just after the original JSR.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&60	Implied	RTS	1	6

SBC

7	Status Register						0
N	V		B	D	I	Z	C
✓	✓	-	-	-	-	✓	✓

Subtract from Accumulator with Carry

Operation

A,C = A - M - (1 - C)

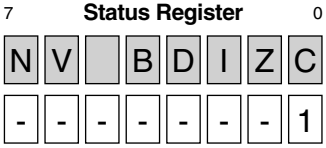
Description

This subtracts the contents of a memory location from the accumulator. The carry flag is used as a borrow and is usually set before a subtraction. When the carry flag is clear 1 is also taken away, thus allowing multiple byte subtraction.

Op.Code	Addressing Mode	Assembly Lang.		No.Bytes	No.Cycles
&E9	Immediate	SBC	#dd	2	2
&E5	Zero Page	SBC	aa	2	3
&F5	Zero Page,X	SBC	aa,X	2	4
&F2	(Indirect Zero Page)	SBC	(aa)	2	5
&ED	Absolute	SBC	aaaa	3	4
&FD	Absolute,X	SBC	aaaa,X	3	4*
&F9	Absolute,Y	SBC	aaaa,Y	3	4*
&E1	(Indirect,X)	SBC	(aa,X)	2	6
&F1	(Indirect),Y	SBC	(aa),Y	2	5*

* Add 1 cycle if page boundary crossed

SEC



Set Carry Flag

Operation

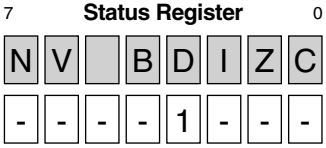
Carry flag = 1

Description

This instruction sets the carry flag and is mainly used to prepare for SBC or ADC.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&38	Implied	SEC	1	2

SED



Set Decimal Mode

Operation

Decimal flag = 0

Description

This instruction switches the 65C12 to binary coded decimal arithmetic mode.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&F8	Implied	SED	1	2

Note: The 65C12 takes one cycle more than the 6502 to perform decimal mode arithmetic.

SEI

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	1	-	-

Set Interrupt Disable Status

Operation

Interrupt flag = 1

Description

This instruction disables maskable interrupts by setting the interrupt flag. While set all normal MOS functions will be suspended until a CLI is performed.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&78	Implied	SEI	1	2

STA

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	-	-

Store Accumulator in Memory

Operation

M = A

Description

This instruction stores the accumulator's contents to a specified memory location.

Op.Code	Addressing Mode	Assembly Lang.		No.Bytes	No.Cycles
&85	Zero Page	STA	aa	2	3
&95	Zero Page,X	STA	aa,X	2	4
&92	(Indirect Zero Page)	STA	(aa)	2	5
&8D	Absolute	STA	aaaa	3	4
&9D	Absolute,X	STA	aaaa,X	3	5
&99	Absolute,Y	STA	aaaa,Y	3	5
&81	(Indirect,X)	STA	(aa,X)	2	6
&91	(Indirect),Y	STA	(aa),Y	2	6

STX

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	-	-

Store X Register in Memory

Operation

M = X

Description

This instruction stores the X register's contents in a specified memory location.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&86	Zero Page	STX aa	2	3
&96	Zero Page,Y	STX aa,Y	2	4
&8E	Absolute	STX aaaa	3	4

STY

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	-	-

Store Y Register in Memory

Operation

M = Y

Description

This instruction stores the Y register's contents in a specified memory location.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&84	Zero Page	STY aa	2	3
&94	Zero Page,X	STY aa,X	2	4
&8C	Absolute	STY aaaa	3	4

STZ

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	-	-

Clear Memory

Operation

M = 0

Description

STZ clears a byte of memory by storing zero at the specified location. CLR is an alternative mnemonic.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&64	Zero Page	STZ aa	2	3
&74	Zero Page,X	STZ aa,X	2	4
&9C	Absolute	STZ aaaa	3	4
&9E	Absolute,X	STZ aaaa,X	3	5

TAX

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	-

Transfer Accumulator to X Register

Operation

X = A

Description

This instruction copies the contents of the accumulator to the X register.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&AA	Implied	TAX	1	2

TAY

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	-

Transfer Accumulator to Y Register

Operation

Y = A

Description

This instruction copies the contents of the accumulator to the Y register.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&A8	Implied	TAY	1	2

TRB

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	✓	-

Test and Reset Bits

Operation

M = (A EOR &FF) AND M

Description

This instruction ANDs the complement of the accumulator with the specified memory location and stores the result in that location. The Z flag is set if A AND M = 0.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&14	Zero Page	TRB aa	2	5
&1C	Absolute	TRB aaaa	3	6

TSB

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	✓	-

Test and Set Bits

Operation

M = A OR M

Description

This instruction ORs the accumulator with the specified memory location and then stores the result in that location. The Z flag is set if A AND M = 0.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&04	Zero Page	TSB aa	2	5
&0C	Absolute	TSB aaaa	3	6

TSX

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	-

Transfer Stack Pointer to X Register

Operation

X = Stack pointer (S)

Description

This instruction copies the contents of the stack pointer to the X register.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&BA	Implied	TSX	1	2

TXA

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	-

Transfer X Register to Accumulator

Operation

A = X

Description

This instruction copies the contents of the X register to the accumulator.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&8A	Implied	TXA	1	2

TXS

7	Status Register						0
N	V		B	D	I	Z	C
-	-	-	-	-	-	-	-

Transfer X Register to Stack Pointer

Operation

Stack pointer (S) = X

Description

This instruction copies the contents of the X register to the stack pointer.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&9A	Implied	TSX	1	2

TYA

7	Status Register						0
N	V		B	D	I	Z	C
✓	-	-	-	-	-	✓	-

Transfer Y Register to Accumulator

Operation

A = Y

Description

This instruction copies the contents of the X register to the accumulator.

Op.Code	Addressing Mode	Assembly Lang.	No.Bytes	No.Cycles
&98	Implied	TYA	1	2

INDEX

Jessa House, 250 High Street, Watford, WD1 2AN, England
Tel: Watford (0923) 37774, Telex: 8956095 WATFRD, Fax: 01 950 8989