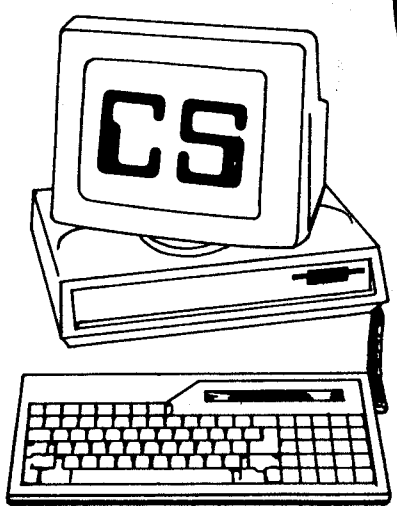
ULTIMON!

The on-board monitor/debugger



OWNERS MANUAL

A reference manual for

ULTIMON !

An in-built machine code Monitor/Debugger
capable of stopping a program

Last Revision Date: 20/01/86

For use on Atari
400, 800, XL and XE Computers

By John Lawson

This hardware is available only from
Computer Support (UK) Limited and approved dealers

TABLE OF CONTENTS

TABLE OF CONTENTS

PREFACE

400 and 800 ULTIMON! users should ignore any reference to XOS as the Operating System remains the same as before.

TRADEMARKS

Credit is herby given to the various trademarks that have been refered to throughout this manual.

ATARI, 400, 800, XL and XE are all trademarks of Atari Corporation.

OSS, MAC/65, ACTION, BASIC XE and BASIC XL are all trademarks of OSS Inc.

UTIMON! is a trademark of Computer Support (UK) Limited.

FITTING INSTRUCTIONS
XL & XE MODELS

Fitting the XOS is a relatively simple task. In most cases it simply involves unplugging an IC and replacing it with another. Some of the newer machines have got this IC soldered directly to the circuit board, but again, this should should be no problem providing you are proficient with a soldering iron.

1) Turn the computer upside down and remove the screws holding the case together.

2) Turn the case back up the right way and lift carefully, you will find on the XL model that the top and bottom are still joined by the keyboard cable.Unplug this from the lower part of the computer and the two halves should separate. On the XE, this cable is at the front.

3) now you should see that most of the circuit board is covered by a metal shield which will need to be removed.
This shield is usually held by screws or simple metal lugs which only need to be bent slightly to allow the cover to come off.

4) Now all is revealed and the next job is to locate the old Operating system chip and remove it!!.
On the XL machines In the upper right hand side of the circuit board you will find the required chip, about two inches to the right of the cartridge slot. It's easy to locate because it is the only 28 pin chip on the board. On the Atari XE the chip is located fairly centrally and slightly to the front of the circuit board.
Diagrams are included to help you locate the correct IC.

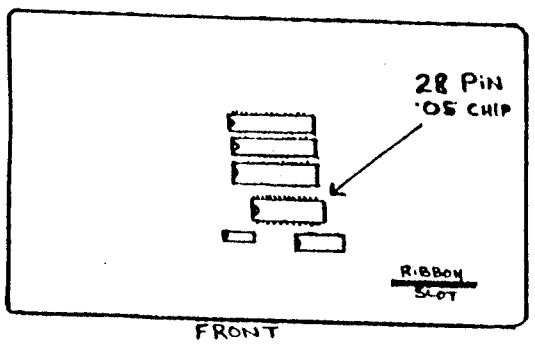
If you are lucky, you may find that it is in an IC socket and all that is required, is to swap the old chip for the new one, making sure you put it round the right way with the little indicator notch on one end of the IC facing the same way as it did on the chip that you have just taken out.

5) Unfortunately not many of us are this lucky and you may find that your OS chip is soldered directly to the circuit board, which means you are going to have to lift out the board and de-solder the pins. This is no job for the complete novice, so please get assistance if you're unsure of your capabilities.

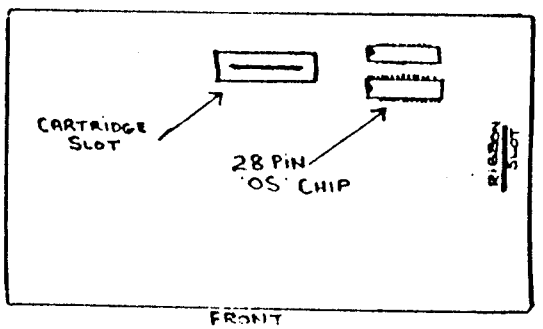
6) Included with the XOS chip is an IC socket, so you won't have to worry about ruining the XOS with heavy handed soldering. Solder the socket with its notch facing left, the same way as on the old chip.
<u>The chip is easily damaged by static electricity, before handling be sure to temporarily 'EARTH' yourself by touching a water pipe or some similarly earthed item</u>.
Once the socket is in place, plug in the XOS, making certain that it is fitted the right way round, with the cutout notch facing left (<u>Please note</u>, fitting the chip the wrong way round will destroy it). All that remains is to reassemble your computer and the job is completed.

If for some reason you do not feel that you would like to tackle the fitting. Computer Support will do the job for you providing you send your computer by Securicor or some other form of insured carrier. Cost of this service is £15 which includes return carriage.

ATARI XE. SHOWING POSITION OF 'OS' CHIP



ATARI XL. SHOWING POSITION OF 'OS' CHIP



HOW TO RUN ULTIMON!

The are several ways to enter ULTIMON!, the following will discribe them.

1) Whilst a program is running, [SHIFT]+[CONTROL]+[TAB], this method works in most cases depending on what the program is doing to the interrupts.

2) After Warmstart, hold [SHIFT]+[CONTROL]+[TAB] and press [RESET].

3) Hold [SHIFT]+[CONTROL]+[TAB] during powerup.

4) From BASIC command line, type BYE then hit [RETURN], this was used to run the self test program on the XL and XE (memo pad on 400 and 800)

5) From a BASIC program with X=USR(59300) (400/800 X=USR(49163).

6) From machine code, JMP $C00C or JMP BLKBDV ($E471).

XL/XE USERS NOTE. During powerup, if there is no drive 1 present and cassette boot or basic has not been enabled the system will default to ULTIMON!

SCREEN TABBING

A new and exiting feature has now been added to ULTIMON!, the ability to TAB between screens. After interrupting a program on the fly you may press [TAB] to toggle between the ULTIMON! screen and the program screen, on first impressions you'll proberly think the program is running again, it's not, only the screen display and vectored interrupts are, you can still command ULTIMON! without tabbing back to the ULTIMON! screen, just type the command and the ULTIMON! screen will appear but only until you press the [RETURN] for that command. Using this feature you may also cause a screen dump of any GRAPHICS MODE 0 screen by entering ULTIMON!, then press [TAB], now hold [CONTROL] whilst pressing [P] (ULTIMON! screen dump).

There is one difference between the XL and the XE versions of ULTIMON! and this affects screen 'tabbing'. The XL version uses the highest available RAM for it's screen display which can be either $BC40, $9C40 or $7C40 depending on what cartridge, (if any) is fitted, the display list occupys the bytes just below, therefore when you enter ULTIMON! its screen will draw there, if the program that you have stopped has it's screen there the ULTIMON! screen will over write it hence a screen tab will just display garbage or nothing. The XE version does not have these problems as it draws it screen in the highest block of the banked 64K, therefore stopping a program on an XE with XE ULTIMON! will not clear any part of the program at all.

XL OWNERS NOTE. We will be marketing an extra 256K battery backed ramdisk that will make your XL fully compatable with the XE you can therefore apply for an update XE ULTIMON! for the cost of £6.00, just return your chip along with the money.
Note. We will not accept back a chip for update if the label has been removed or tampered with. If any chip returned to us has broken legs it shall replaced and we will charge for supplying a new one.

COMMAND ENTRY

ULTIMON! uses spaces as parameter delimiters, any space between numbers will force the next number to be read as a new perameter. Spaces are not compressed therefore 2 spaces or more in a command will cause a syntax error.
There should not be any spaces after the last parameter as this will also cause a syntax error.

The command line is only 32 character spaces long, it will go no further.

All the input and output of ULTIMON! is in hexadecimal notation.

[RETURN] repeats the last command.

COMMAND DESCRIPTIONS

Throughout the descriptions of commands there is a 'Syntax' line and an 'Example' line followed by a discription of what the command does.

On the 'Syntax' line I have used the underline symbol as a 'space'.

A - Alter Memory Location Contents

Syntax: A_<START>_<BYTE>_[<BYTE>]...
Example: A 84FE 56 78 FF 00[RETURN]

The A command alters the contents of a memory location or a consecutive sequence of memory locations beginning at START, the maximum number of consecutive bytes on one command line is 9.

To use this command, first find the address where the alteration is to take place then type A then a space then the byte you would like to store there, followed by either a return or a sequence of space then byte (up to 9) and finally a return.

B - Breakpiont Set

Syntax: B_<ADDR>_<REG>
Example B 3E80 2[RETURN]

The B command allows the user to preset upto 6 breakpionts throughtout the code. To remove a breakpiont just enter zero as the address, followed by the breakpiont register number.
Note. All breakpoints should be cleared manually before leaving ULTIMON!, through the JP, J, GP, G, QD, QC or QR commands.

C - Compare two Blocks of Memory

Syntax: C_<FROM>_<TO>_<START>
Example: C 2000 4FFF 5000[RETURN]

The C command will compare one block of memory against another and then report the start of where any differences have accured.

To use this command, first establish the two blocks of memory that you wish compare then type C followed by a space now the starting address of block 1 then another space now the end address of the block 1 then another space and finally the start address of block 2 now return, the end address of block 2 is taken care of automatically.

If the compared blocks were the same then ULTIMON! will come back unchanged, if there were differences then the 'Display Memory' part of the screen will split down and display the two blocks (block 1 at the top), starting at

where the differences were.

D - Disassemble Memory

Syntax: D_<ADDR>
Example: D 9FAE[RETURN]
Immediate Key: [=] At any time will increment the disassembly by a logical line.

The D command disassembles the memory into standard MOS Technology (designers of 6502 chip) opcodes.

To use this command, establish the address that you would like to start disassembling from then type D followed by that address now a return.

The format of each disassembly line is as follows:

ADDRESS OPCODE BYTES ATASCII

ADDRESS = the address of the opcode, OPCODE = the disassembly interpretation of this byte, BYTES = the byte or bytes that form this particular opcode, ATASCII = the ATASCII interpretation of these bytes.

E - Examine Memory Contents

Syntax: E_<ADDR>
Example: E 4000[RETURN]
Immediate Keys: [=] at any time will increment the display one page at a time.
[-] at any time will decrement the display one page at a time.

The E command displays the contents of a 'half page' of memory beginning at the address specified, at any time you may increment or decrement the 'half page' display by pressing [=] for increment or [-] for decrement.

At the 5th line down you will see 'Ad' (meaning address) this is the start of the 'half page' display, the 'half page' display is 16 horizontal lines long, this is refered to during any display memory contents or disassembly.

To use this command, first establish the starting address of the area you wish to examine now type E followed by a space then the address and return.

The format of each display line is as follows:

AAAA 00 00 00 00 00 00 00 00 12345678

AAAA = the hexadecimal address of the first byte being displayed on this line, 00 = the hexadecimal contents of succesive memory locations beginning at location AAAA, and the 12345678 = the ATASCII character interpretation of the succesive memory locations.

F - Fill Memory

Syntax: F_<FROM>_<TO>_<BYTE>
Example: F 8000 8FFF 55[RETURN]

The F command will fill a block of memory.

Consider the following example:

F 2000 2FFF 55[RETURN]

this will fill from 2000 through to 2FFF with 55's.

FM - Format a Diskette

Syntax: FM
Example: FM[RETURN]

The FM command will format a diskette in the density that the drive may be selected to.

G - Go at Address

Syntax: G_<ADDR>
Example: G F11B[RETURN]

The G command will change the program counter to the address specified then go to it.

GP - Go at Program Counter

Syntax: GP
Example: G[RETURN]

The GP command will continue from where the program counter left off.

I<x> - Basic Interpreter OFF/ON XL/XE ONLY

Syntax: I<X>
Example: I<X>[RETURN]

The I0 command will disable the built in BASIC interpreter then move the screen up into the now available RAM.

The I1 command will move the screen down then enable the BASIC interpreter.

J - Jump to Subroutine at Address

Syntax: J_<ADDR>
Example: J F6A4[RETURN]

The J command will push the ULTIMON! go address to the stack then go to the subroutine address specified.

JP - Jump to Subroutine at PC

Syntax: JP
Example: JP[RETURN]

The JP command will push the ULTIMON! go address to the stack then go from the program counter.

L - Locate a String of Bytes

Syntax: L_<START>_<END>_<BYTE>_[<BYTE>]...
EXample: L 2000 8000 20 A4 F6 A9 FF[RETURN]

The L command will locate a sequental string of hexadecimal bytes upto 8 bytes long.

To use this command first establish the bytes that you wish to locate, now the address range then type L followed by a space then the starting address then the end address and lastly the byte or string of bytes, there must be a space between every byte and no spaces between the ending byte and the return.

M - Move a Block of Memory

Syntax: M_<from>_<to>_<start>
Example: M 3000 3FFF 6000[RETURN]

The M command will move a copy of a block of memory beginning at FROM and ending at TO to START

Consider the following example:

M 4000 5FFF 7000[RETURN]

this will move a copy of the block 4000 through to 5FFF into 7000 through to 7FFF.

Note. take care not to forward overlap a move, as this will destroy the overlapping part of your code, a backward overlap is okay.

Alter Processor Status Register

Syntax: P<byte>
Example: P24[RETURN]

The P command will change the contents of the ULTIMON! proccessor status register shaddow (this will be placed into the real proccessor status register on a 'execute from program counter'.
Note: No spaces are required.

P - Dump Screen to the Printer

Immediate: [CONTROL]+P

Printer dump may be called at any time without interfering with what you may be doing with the command line.
Note. This routine can also be called from outside ULTIMON!

Q - Quit out of ULTIMON!

Syntax: Q<X>
Example: Q<X>[RETURN]

The Q command will quit out from ULTIMON! to the particular media you require.

QD = quit out to disk based media
QC = quit out to cassete based media
QR = quit out to cartridge (ROM) based media

Note: if CAINI, CARTAD, CASINI, DOSINI or DOSVEC contains zero's then there is likely to be a system lockup depending on which media has been selected.

R - Read from Device

Syntax: R <SECNO> <BUFFER> <NOSECS>
Example: R 1 4000 2F[RETURN]

The R command will read a sector or string of sectors from the linked disk device, SECNO is the starting sector number, BUFFER is the starting address that the data is to be stored to, NOSECS is the number of sectors to be read.

S - Single Step

Syntax: S
Example: S[RETURN]

The S command will allow the user to single step through the code being displayed on the disassembly page. Holding return will continually single step until you let go.
Note. This command only works from the disassembly page.

S<X> - Supercartridge OFF/ON

Syntax: S<X>
Example: S<X>[RETURN]

The S0 command will disable any supercartridge that may be plugged in then move the screen up into the available RAM.

The S1 command will move the screen down then enable the supercartridge that may be plugged in.

Note. The only supercartridges in existance at the time of writing are MAC/65, ACTION, BASIC XE and BASIC XL.

T - Trace through Memory

Syntax: T
Example: T[RETURN]

The T command will allow the user to single step continually through the code being displayed on the disassembly page. Until the user presses the [ESC] key.

Note. This command only works from the disassembly page.

U - Display Update Mode

Syntax: U <addr>
Example: U 0000[RETURN]

The U command will display the memory contents exactly as the D command but will then continue sampleing the same area of memory until the [ESC] key is pressed, this is handy if you would like to see if a memory location or string of memory locations is being updated during vertical blank interrupts.

W - Write to Device

Syntax: W <SECNO> <BUFFER> <NOSECS>
Example: W 40 2000 80[RETURN]

The W command will write a sector or string of sectors from the linked device, SECNO is the starting sector number, BUFFER is the starting address that the data is to be read from, NOSECS is the number of sectors to be written.

X - One Byte Read

Syntax: X <ADDR>
Example: X D508[RETURN]

The X command is for a one byte read the address and result of that read is stored on the 1st and 2nd lines and is kept there until the next X command.
Note. This command is mainly of use to those electronics experts that wish to experiment with the address and read lines.

Pop Stack

Syntax: [CONTROL]+[-]
Example: [CONTROL]+[-][RETURN]

The command will remove the top two bytes (word) from the stack.

Usefull for removing a return address and then using the push command to replace it with an alternative one.
Note. no spaces required

Push Stack

Syntax: [CONTROL]+[=] <addr>
Example: [CONTROL]+[=] 725C[RETURN]

The push stack command will push two bytes (a word) to the stack, using this option you could push a return address for the next RTS.

Alter 6502 Registers

Syntax: A<byte> X<byte> Y<byte>
Examples: AFE[RETURN] X22[RETURN] Y5C[RETURN]

The A,X and Y commands will change the contents of the ULTIMON! register shaddows (these will be placed into the real registers on the JP, J, GP, G, QD, QC and QC commands. Note. only one register can be changed per command line, no spaces are required.

Link Device Type Select

Syntax: D[<D>][<C>]
Example: DC[RETURN]

The DD and DS commands will link the serial I/O to either Cassete or Disk. 128 byte or 256 byte I/O transfer can also be used on Cassette.

Link Device Number Select

Syntax: N[<1>][<2>][<3>][<4>]
Example: N2[RETURN]

The N1, N2, N3 and N4 commands will select the linked drive number for disk I/O.
Note. This will have no affect on Cassette I/O.

Link Density Select

Syntax: :[<S>][<D>]
Example: :D[RETURN]

The :D and :S commands will toggle the device link to either single or double density (128 byte or 256 byte transfer).
ULTIMON! supports single, dual or double density disk I/O.

Single Density = DS (default)
Dual Density = DS (default)
Double Density = DD

Note. The first 3 sectors of a Double density disk are in Single Density.

A NOTE ON PIRACY

Generally what happens is that the person who is the cause of piracy is not aware of that, in most cases this person gives a copy to a friend/relative on the understanding that it will not be passed on from there, this understanding is soon forgotten and the end result of continually passing on the software/hardware could be hundreds of lost sales to the publisher. Piracy is generally on a large scale.

The persons that have endangered the very existance of this your favorite hobby are those that have spent all thier time cracking protection then they pass the unprotected pieces of software on, once is enough to kill the sales of any program. If the ubove mentioned persons spent their time writing programs and not cracking them, we would proberly see a better range of software available today and at cheaper prices, the trouble is their intelligence stops at the keyboard.

Back in the days when piracy was in its infancy there were dozens of new programs becoming available every week, due to piracy there is now proberly only 2 good programs appearing every month on average and these rearly pay the author a reasonable salary for the 3-6 months he/she may have worked to write it, imagine if you worked for that amount of time then earned only a months salary, you would not do it again!

Programmers can earn a reasonable salary if they move towards business computers, but surely we do not want that to happen do we?

PLEASE, BUY PROGRAMS DON'T STEAL THEM!

NOTES

NOTES

NOTES

The reference manual for

XOS!

A Revised Operating System
with most entry points compatible with ATARI Rev. B

Last Revision Date: 27/03/86

For use on ATARI
XL and XE Computers

By John Lawson

CONTENTS

COPYRIGHT NOTICE

TRADEMARKS

Credit is hereby given to the various trademarks that have been referred to throughout this manual.

ATARI, 400, 800, XL and XE are all trademarks of Atari Corporation.

MAC/65, ACTION, BASIC XE and BASIC XL are all trademarks of OSS Inc.

XOS! is a trademark of Computer Support (UK) Limited.

OVERVIEW

XOS! is an Operating System that uses the same entry points and vectors as Revision 'B' Atari Operating System.

Why do you need XOS?

A fair amount of existing software was designed for use on the old type 400/800 Atari systems, when the new XL/XE series machines were designed the Operating System was changed mainly for the self test, parallel port handler, international character set and warmstart routines, whilst adding these routines to the Operating System the programmer decided to alter the positions of the existing routines therefore any program that used the O.S. to perform some of its tasks would be jumping to the wrong place causing unpredictable results (usually a lockup).

Because XOS! is based around Revision 'B' Operating System most of the routines are in the same place the result is that the XOS! system is more compatible with existing software than the XL/XE Operating System.

ENHANCEMENTS

Powerup

During powerup, the [OPTION] key has to be held down to enable the BASIC INTERPRETER (the opposite effect to that of a normal XL/XE Operating System).

During powerup, the [SELECT] key has to be held down to enable the OSS supercartridge, if there's one plugged in that is. Once selected pressing [RESET] will not disable it.

Holding [CONTROL]+[TAB] down during powerup causes the system to ignore the initialisation and execution of a standard cartridge, this saves you removing the cartridge when you do not need it.
NOTE. Using this function will not give RAM where the cartridge may be.

Holding [2] key down will boot the system from drive 2, the default is drive 1.
NOTE. Single stage boot only, as all multi stage loaders look back to drive 1 to continue with the boot process.

Forced Warmstart

Pressing the [4] key whilst holding [CONTROL] down forces the system to do a warmstart regardless of the contents of COLDST ($0244) the result is like pressing reset.
Note. Forced warmstart will not re-boot a disk.

Anti-Coldstart

There are 3 powerup bytes in an XL/XE they normally contain three different numbers, when you press [RESET] the system examines these powerup bytes to see if they have been altered since powerup if they have the system will do a Coldstart otherwise there will be a Warmstart.

XOS! keeps replacing these numbers on vertical blank interrupts therefore the system can not coldstart through altering these locations.

Cursor Control

The XOS! cursor is considerably faster than the standard cursor, this makes editing text much quicker than normal.

ENHANCEMENTS

Character Set

The character set has been completely re-defined to allow a more readable display, all characters with the exception of the 'I' and some of the control characters are 7 pixels wide, rather than 6.

Faster Cassette Baud Rate

Cassette load/save now defaults to 820 baud. The Operating System automatically calculates the baud rate as a tape is loading, therefore any tape written by the XOS system at 820 baud should load on other Atari's with no problems, although it will be faster. XOS will also read a normal 600 baud tape with no problems.

Joysticks 3 & 4

All the Joystick, Paddle and Trigger registers that were used for sockets 3 & 4 on the 400/800 are in use with XOS. The registers for socket 1 echos its contents to the registers for socket 3, the same for that of 2 & 4 (this will hopefully ensure any programs that used sockets 3 & 4 will now work with two sockets)

SYSTEM INITIALISATION

Bootstrap

During a powerup XOS! looks to see if the [2] key is down, for a Boot from Drive 2, if the last key that you pressed before turning off was a [2], then you powerup again soon after XOS! will try to Boot from Drive 2 (the system does not completely power down until approx 2 minutes has elapsed), if this happens just hold the [SPACE] down during powerup.

Reset

XOS! uses a bank select shadow register this is called "BKFLG" (Memory Location $07) when the [RESET] keys is pressed, the contents of BKFLG are stored into PORTB (Memory Location $D301), this is the only time this shadow address is used. Also XOS! uses three powerup bytes these addresses are called "PWRBT1, PWRBT2, and PWRBT3" (Memory Location $033D-$033F) these are set, during a [RESET] to $5C, $93 and $25 respectively, they are kept that way on VBI's this is known as Anti-Coldstart as mentioned earlier.

RAM O.S. Default

If you find a need to have a TRANSLATOR or other RAM based O.S. in control, then using the special Reset routine mentioned above, you can set the system to default to your new O.S. on pressing [RESET]. The following describes how to do just that; once powered to the RAM based O.S. read the contents of PORTB, now store it to BKFLG, the only thing left to do is to store the three PWRBT numbers to their consecutive locations as mentioned above. If the Warmstart routine on your RAM based O.S. clears the PWRBT's then you must set them again after a System Reset, ideally all the above should be done by your O.S. automatically.

ADDRESSING MODES

The R6500 CPU family has 13 addressing modes. In the following discussion of these addressing modes, a bracketed expression follows the title of the mode. This expression is the term used in the Instruction Set Op Code Matrix table (later in this product description) to make it easier to identify the actual addressing mode used by the instruction.

ACCUMULATOR ADDRESSING [Accum]—This form of addressing is represented with a one byte instruction, implying an operation on the accumulator.

IMMEDIATE ADDRESSING [IMM]—In immediate addressing, the second byte of the instruction contains the operand, with no further memory addressing required.

ABSOLUTE ADDRESSING [Absolute]—In absolute addressing, the second byte of the instruction specifies the eight low order bits of the effective address while the third byte specifies the eight high order bits. Thus, the absolute addressing mode allows access to the entire 64K bytes of addressable memory.

ZERO PAGE ADDRESSING [ZP]—The zero page instructions allow for shorter code and execution times by fetching only the second byte of the instruction and assuming a zero high address byte. Careful use of the zero page can result in significant increase in code efficiency.

INDEXED ZERO PAGE ADDRESSING [ZP, X or Y]—(X, Y indexing)—This form of addressing is used with the index register and is referred to as "Zero Page, X" or "Zero Page, Y". The effective address is calculated by adding the second byte to the contents of the index register. Since this is a form of "Zero Page" addressing, the content of the second byte references a location in page zero. Additionally, due to the "Zero Page" addressing nature of this mode, no carry is added to the high order eight bits of memory and crossing of page boundaries does not occur.

INDEXED ABSOLUTE ADDRESSING [ABS, X or Y]—(X, Y indexing)—This form of addressing is used in conjunction with X and Y index register and is referred to as "Absolute, X" and "Absolute, Y". The effective address is formed by adding the contents of X or Y to the address contained in the second and third bytes of the instruction. This mode allows the index register to contain the index or count value and the instruction to contain the base address. This type of indexing allows any location referencing and the index to modify multiple fields, resulting in reduced coding and execution time.

IMPLIED ADDRESSING [Implied]—In the implied addressing mode, the address containing the operand is implicitly stated in the operation code of the instruction.

RELATIVE ADDRESSING [Relative]—Relative addressing is used only with branch instructions and establishes a destination for the conditional branch.

The second byte of the instruction becomes the operand which is an "Offset" added to the contents of the lower eight bits of the program counter when the counter is set at the next instruction. The range of the offset is −128 to +127 bytes from the next instruction.

INDEXED INDIRECT ADDRESSING [(IND, X)]—In indexed indirect addressing (referred to as (Indirect, X)), the second byte of the instruction is added to the contents of the X index register, discarding the carry. The result of this addition points to a memory location on page zero whose contents are the low order eight bits of the effective address. The next memory location in page zero contains the high order eight bits of the effective address. Both memory locations specifying the high and low order bytes of the effective address must be in page zero.

INDIRECT INDEXED ADDRESSING [(IND), Y]—In indirect indexed addressing (referred to as (Indirect), Y), the second byte of the instruction points to a memory location in page zero. The contents of this memory location are added to the contents of the Y index register, the result being the low order eight bits of the effective address. The carry from this addition is added to the contents of the next page zero memory location, the result being the high order eight bits of the effective address.

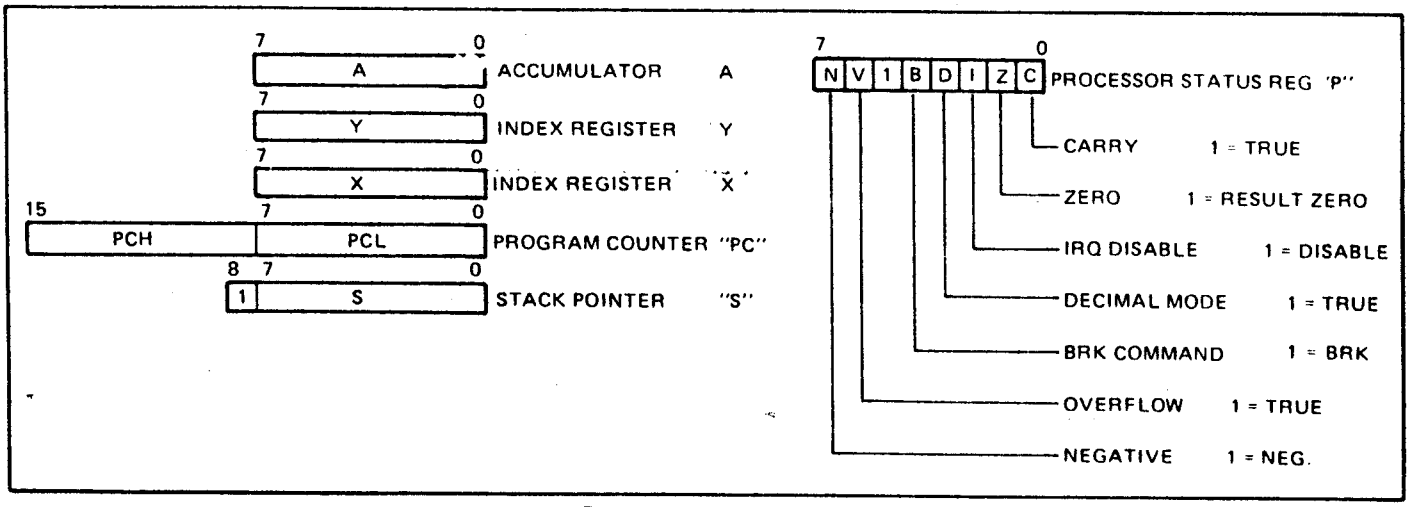
ABSOLUTE INDIRECT [Indirect]—The second byte of the instruction contains the low order eight bits of a memory location. The high order eight bits of that memory location are contained in the third byte of the instruction. The contents of the fully specified memory location are the low order byte of the effective address. The next memory location contains the high order byte of the effective address which is loaded into the sixteen bits of the program counter. (JMP (IND) only)

INSTRUCTION SET

The R6500 CPU has 56 instruction types which are enhanced by up to 13 addressing modes for each instruction. The accumulator, index registers, Program Counter, Stack Pointer and Processor Status Register are illustrated below.

Alphabetic Listing of Instruction Set

Mnemonic	Function	Mnemonic	Function
ADC	Add Memory to Accumulator with Carry	JMP	Jump to New Location
AND	"AND" Memory with Accumulator	JSR	Jump to New Location Saving Return Address
ASL	Shift Left One Bit (Memory or Accumulator)		
		LDA	Load Accumulator with Memory
BCC	Branch on Carry Clear	LDX	Load Index X with Memory
BCS	Branch on Carry Set	LDY	Load Index Y with Memory
BEQ	Branch on Result Zero	LSR	Shift One Bit Right (Memory or Accumulator)
BIT	Test Bits in Memory with Accumulator		
BMI	Branch on Result Minus	NOP	No Operation
BNE	Branch on Result not Zero		
BPL	Branch on Result Plus	ORA	"OR" Memory with Accumulator
BRK	Force Break		
BVC	Branch on Overflow Clear	PHA	Push Accumulator on Stack
BVS	Branch on Overflow Set	PHP	Push Processor Status on Stack
		PLA	Pull Accumulator from Stack
CLC	Clear Carry Flag	PLP	Pull Processor Status from Stack
CLD	Clear Decimal Mode		
CLI	Clear Interrupt Disable Bit	ROL	Rotate One Bit Left (Memory or Accumulator)
CLV	Clear Overflow Flag	ROR	Rotate One Bit Right (Memory or Accumulator)
CMP	Compare Memory and Accumulator	RTI	Return from Interrupt
CPX	Compare Memory and Index X	RTS	Return from Subroutine
CPY	Compare Memory and Index Y		
		SBC	Subtract Memory from Accumulator with Borrow
DEC	Decrement Memory by One	SEC	Set Carry Flag
DEX	Decrement Index X by One	SED	Set Decimal Mode
DEY	Decrement Index Y by One	SEI	Set Interrupt Disable Status
		STA	Store Accumulator in Memory
EOR	"Exclusive-OR" Memory with Accumulator	STX	Store Index X in Memory
		STY	Store Index Y in Memory
INC	Increment Memory by One		
INX	Increment Index X by One	TAX	Transfer Accumulator to Index X
INY	Increment Index Y by One	TAY	Transfer Accumulator to Index Y
		TSX	Transfer Stack Pointer to Index X
		TXA	Transfer Index X to Accumulator
		TXS	Transfer Index X to Stack Register
		TYA	Transfer Index Y to Accumulator



Programming Model

INSTRUCTION SET OP CODE MATRIX

The following matrix shows the Op Codes associated with the R6500 family of CPU devices. The matrix identifies the hexadecimal code, the mnemonic code, the addressing mode, the number of instruction bytes, and the number of machine cycles associated with each Op Code. Also, refer to the instruction set summary for additional information on these Op Codes.

MSD \ LSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	MSD
0	BRK Implied 1 7	ORA (IND, X) 2 6				ORA ZP 2 3	ASL ZP 2 5	RMB0 ZP 2 5	PHP Implied 1 3	ORA IMM 2 2	ASL Accum 1 2			ORA ABS 3 4	ASL ABS 3 6	BBR0 ZP 3 5**	0
1	BPL Relative 2 2**	ORA (IND), Y 2 5*				ORA ZP, X 2 4	ASL ZP, X 2 6	RMB1 ZP 2 5	CLC Implied 1 2	ORA ABS, Y 3 4*				ORA ABS, X 3 4*	ASL ABS, X 3 7	BBR1 ZP 3 5**	1
2	JSR Absolute 3 6	AND (IND, X) 2 6			BIT ZP 2 3	AND ZP 2 3	ROL ZP 2 5	RMB2 ZP 2 5	PLP Implied 1 4	AND IMM 2 2	ROL Accum 1 2		BIT ABS 3 4	AND ABS 3 4	ROL ABS 3 6	BBR2 ZP 3 5**	2
3	BMI Relative 2 2**	AND (IND, Y) 2 5*				AND ZP, X 2 4	ROL ZP, X 2 6	RMB3 ZP 2 5	SEC Implied 1 2	AND ABS, Y 3 4*				AND ABS, X 3 4*	ROL ABS, X 3 7	BBR3 ZP 3 5**	3
4	RTI Implied 1 6	EOR (IND, X) 2 6				EOR ZP 2 3	LSR ZP 2 5	RMB4 ZP 2 5	PHA Implied 1 3	EOR IMM 2 2	LSR Accum 1 2		JMP ABS 3 3	EOR ABS 3 4	LSR ABS 3 6	BBR4 ZP 3 5**	4
5	BVC Relative 2 2**	EOR (IND), Y 2 5*				EOR ZP, X 2 4	LSR ZP, X 2 6	RMB5 ZP 2 5	CLI Implied 1 2	EOR ABS, Y 3 4*				EOR ABS, X 3 4*	LSR ABS, X 3 7	BBR5 ZP 3 5**	5
6	RTS Implied 1 6	ADC (IND, X) 2 6				ADC ZP 2 3	ROR ZP 2 5	RMB6 ZP 2 5	PLA Implied 1 4	ADC IMM 2 2	ROR Accum 1 2		JMP Indirect 3 5	ADC ABS 3 4	ROR ABS 3 6	BBR6 ZP 3 5**	6
7	BVS Relative 2 2**	ADC (IND, Y) 2 5*				ADC ZP, X 2 4	ROR ZP, X 2 6	RMB7 ZP 2 5	SEI Implied 1 2	ADC ABS, Y 3 4*				ADC ABS, X 3 4*	ROR ABS, X 3 7	BBR7 ZP 3 5**	7
8		STA (IND, X) 2 6			STY ZP 2 3	STA ZP 2 3	STX ZP 2 3	SMB0 ZP 2 5	DEY Implied 1 2		TXA Implied 1 2		STY ABS 3 4	STA ABS 3 4	STX ABS 3 4	BBS0 ZP 3 5**	8
9	BCC Relative 2 2**	STA (IND, Y) 2 6			STY ZP, X 2 4	STA ZP, X 2 4	STX ZP, Y 2 4	SMB1 ZP 2 5	TYA Implied 1 2	STA ABS, Y 3 5	TXS Implied 1 2			STA ABS, X 3 5		BBS1 ZP 3 5**	9
A	LDY IMM 2 2	LDA (IND, X) 2 6	LDX IMM 2 2		LDY ZP 2 3	LDA ZP 2 3	LDX ZP 2 3	SMB2 ZP 2 5	TAY Implied 1 2	LDA IMM 2 2	TAX Implied 1 2		LDY ABS 3 4	LDA ABS 3 4	LDX ABS 3 4	BBS2 ZP 3 5**	A
B	BCS Relative 2 2**	LDA (IND), Y 2 5*			LDY ZP, X 2 4	LDA ZP, X 2 4	LDX ZP, Y 2 4	SMB3 ZP 2 5	CLV Implied 1 2	LDA ABS, Y 3 4*	TSX Implied 1 2		LDY ABS, X 3 4*	LDA ABS, X 3 4*	LDX ABS, Y 3 4*	BBS3 ZP 3 5**	B
C	CPY IMM 2 2	CMP (IND, X) 2 6			CPY ZP 2 3	CMP ZP 2 3	DEC ZP 2 5	SMB4 ZP 2 5	INY Implied 1 2	CMP IMM 2 2	DEX Implied 1 2		CPY ABS 3 4	CMP ABS 3 4	DEC ABS 3 6	BBS4 ZP 3 5**	C
D	BNE Relative 2 2**	CMP (IND), Y 2 5*				CMP ZP, X 2 4	DEC ZP, X 2 6	SMB5 ZP 2 5	CLD Implied 1 2	CMP ABS, Y 3 4*				CMP ABS, X 3 4*	DEC ABS, X 3 7	BBS5 ZP 3 5**	D
E	CPX IMM 2 2	SBC (IND, X) 2 6			CPX ZP 2 3	SBC ZP 2 3	INC ZP 2 5	SMB6 ZP 2 5	INX Implied 1 2	SBC IMM 2 2	NOP Implied 1 2		CPX ABS 3 4	SBC ABS 3 4	INC ABS 3 6	BBS6 ZP 3 5**	E
F	BEQ Relative 2 2**	SBC (IND), Y 2 5*				SBC ZP, X 2 4	INC ZP, X 2 6	SMB7 ZP 2 5	SED Implied 1 2	SBC ABS, Y 3 4*				SBC ABS, X 3 4*	INC ABS, X 3 7	BBS7 ZP 3 5**	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

0
0 | BRK / Implied / 1 7 | —OP Code
—Addressing Mode
—Instruction Bytes; Machine Cycles

*Add 1 to N if page boundary is crossed.
**Add 1 to N if branch occurs to same page; add 2 to N if branch occurs to different page.

INSTRUCTION SET SUMMARY

INSTRUCTIONS		IMMEDIATE			ABSOLUTE			ZERO PAGE			ACCUM			IMPLIED			(IND, X)			(IND), Y			Z PAGE, X			ABS, X			ABS, Y			RELATIVE			INDIRECT			Z PAGE, Y			PROCESSOR STATUS CODES 7 6 5 4 3 2 1 0	
MNEMONIC	OPERATION	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	N V • • • • Z C	MNEMONIC
ADC	A + M + C → A (4) (1)	69	2	2	6D	4	3	65	3	2							61	6	2	71	5	2	75	4	2	7D	4	3	79	4	3										N V • • • • Z C	ADC
AND	A∧M → A (1)	29	2	2	2D	4	3	25	3	2							21	6	2	31	5	2	35	4	2	3D	4	3	39	4	3										N • • • • • Z •	AND
ASL	C ← [ ] ← 0				0E	6	3	06	5	2	0A	2	1										16	6	2	1E	7	3													N • • • • • Z C	ASL
BCC	BRANCH ON C = 0 (2)																															90	2	2								• • • • • • • •	BCC
BCS	BRANCH ON C = 1 (2)																															B0	2	2								• • • • • • • •	BCS
BEQ	BRANCH ON Z = 1 (2)																															F0	2	2								• • ● • • • •	BEQ
BIT	A∧M				2C	4	3	24	3	2																																M M • • • • Z •	BIT
BMI	BRANCH ON N = 1 (2)																															30	2	2								• • • • • • • •	BMI
BNE	BRANCH ON Z = 0 (2)																															D0	2	2								• • • • • • • •	BNE
BPL	BRANCH ON N = 0 (2)																															10	2	2								• • • • • • • •	BPL
BRK	BREAK													00	7	1																									• • • 1 • • •	BRK
BVC	BRANCH ON V = 0 (2)																															50	2	2								• • • • • • • •	BVC
BVS	BRANCH ON V = 1 (2)																															70	2	2								• • • • • • • •	BVS
CLC	0 → C													18	2	1																									• • • • • • • 0	CLC
CLD	0 → D													D8	2	1																									• • • • 0 • • •	CLD
CLI	0 → I													58	2	1																									• • • • • 0 • •	CLI
CLV	0 → V													B8	2	1																									• 0 • • • • • •	CLV
CMP	A - M	C9	2	2	CD	4	3	C5	3	2							C1	6	2	D1	5	2	D5	4	2	DD	4	3	D9	4	3										N • • • • • Z C	CMP
CPX	X - M	E0	2	2	EC	4	3	E4	3	2																															N • • • • • Z C	CPX
CPY	Y - M	C0	2	2	CC	4	3	C4	3	2																															N • • • • • Z C	CPY
DEC	M - 1 → M				CE	6	3	C6	5	2													D6	6	2	DE	7	3													N • • • • • Z •	DEC
DEX	X - 1 → X													CA	2	1																									N • • • • • Z •	DEX
DEY	Y - 1 → Y													88	2	1																									N • • • • • Z •	DEY
EOR	A ⊻ M → A (1)	49	2	2	4D	4	3	45	3	2							41	6	2	51	5	2	55	4	2	5D	4	3	59	4	3										N • • • • • Z •	EOR
INC	M + 1 → M				EE	6	3	E6	5	2													F6	6	2	FE	7	3													N • • • • • Z •	INC
INX	X + 1 → X													E8	2	1																									N • • • • • Z •	INX
INY	Y + 1 → Y													C8	2	1																									N • • • • • Z •	INY
JMP	JUMP TO NEW LOC				4C	3	3																												6C	5	3				• • • • • • • •	JMP
JSR	JUMP SUB				20	6	3																																		• • • • • • • •	JSR
LDA	M → A (1)	A9	2	2	AD	4	3	A5	3	2							A1	6	2	B1	5	2	B5	4	2	BD	4	3	B9	4	3										N • • • • • Z •	LDA
LDX	M → X (1)	A2	2	2	AE	4	3	A6	3	2																			BE	4	3							B6	4	2	N • • • • • Z •	LDX
LDY	M → Y (1)	A0	2	2	AC	4	3	A4	3	2													B4	4	2	BC	4	3													N • • • • • Z •	LDY
LSR	0 → [ ] → C				4E	6	3	46	5	2	4A	2	1										56	6	2	5E	7	3													0 • • • • • Z C	LSR
NOP	NO OPERATION													EA	2	1																									• • • • • • • •	NOP
ORA	A ⊻ M → A	09	2	2	0D	4	3	05	3	2							01	6	2	11	5	2	15	4	2	1D	4	3	19	4	3										N • • • • • Z •	ORA
PHA	A → Ms S - 1 → S													48	3	1																									• • • • • • • •	PHA
PHP	P → Ms S - 1 → S													08	3	1																									• • • • • • • •	PHP
PLA	S + 1 → S Ms → A													68	4	1																									N • • • • • Z •	PLA
PLP	S + 1 → S Ms → P													28	4	1																									(RESTORED)	PLP
ROL	[ ] ← [C] ←				2E	6	3	26	5	2	2A	2	1										36	6	2	3E	7	3													N • • • • • Z C	ROL
ROR	→ [C] → [ ]				6E	6	3	66	5	2	6A	2	1										76	6	2	7E	7	3													N • • • • • Z C	ROR
RTI	RTRN INT													40	6	1																									(RESTORED)	RTI
RTS	RTRN SUB													60	6	1																									• • • • • • • •	RTS
SBC	A - M - C̄ → A (1)	E9	2	2	ED	4	3	E5	3	2							E1	6	2	F1	5	2	F5	4	2	FD	4	3	F9	4	3										N V • • • • Z C	SBC
SEC	1 → C													38	2	1																									• • • • • • • 1	SEC
SED	1 → D													F8	2	1																									• • • • 1 • • •	SED
SEI	1 → I													78	2	1																									• • • • • 1 • •	SEI
STA	A → M				8D	4	3	85	3	2							91	6	2	91	6	2	95	4	2	9D	5	3	99	5	3										• • • • • • • •	STA
STX	X → M				8E	4	3	86	3	2																														96	4	2	• • • • • • • •	STX
STY	Y → M				8C	4	3	84	3	2													94	4	2																	• • • • • • • •	STY
TAX	A → X													AA	2	1																									N • • • • • Z •	TAX
TAY	A → Y													A8	2	1																									N • • • • • Z •	TAY
TSX	S → X													BA	2	1																									N • • • • • Z •	TSX
TXA	X → A													8A	2	1																									N • • • • • Z •	TXA
TXS	X → S													9A	2	1																									• • • • • • • •	TXS
TYA	Y → A													98	2	1																									N • • • • • Z •	TYA

(1) ADD 1 TO N IF PAGE BOUNDARY IS CROSSED
(2) ADD 1 TO N IF BRANCH OCCURS TO SAME PAGE
ADD 2 TO N IF BRANCH OCCURS TO DIFFERENT PAGE
(3) CARRY NOT BORROW
(4) IF IN DECIMAL MODE Z FLAG IS INVALID
ACCUMULATOR MUST BE CHECKED FOR ZERO RESULT

X INDEX X
Y INDEX Y
A ACCUMULATOR
M MEMORY PER EFFECTIVE ADDRESS
MS MEMORY PER STACK POINTER
+ ADD
- SUBTRACT
∧ AND
∨ OR
⊻ EXCLUSIVE OR
M7 MEMORY BIT 7
M6 MEMORY BIT 6
/ NO CYCLES
NO BYTES