THE DATA HANDLER

Owner's Manual

THE
DATA HANDLER
OWNER'S MANUAL

The information in this manual is believed to be entirely reliable. However, no responsibility is assumed for inaccuracies. The material in this manual is for informational purposes only and is subject to change without notice.

Third Edition

TABLE OF CONTENTS

Chapter		Page
1	INTRODUCTION:	1-2
2	SPECIFICATIONS:	-3
3	SYSTEM INFORMATION:	-4
	SYSTEM SPEED:	
	CRYSTAL OPERATION:	
4	I/O PORT INFORMATION:	-5
5	PRE-ASSEMBLY NOTES:	6-8
6	ASSEMBLY INSTRUCTIONS:	9-13
7	SYSTEM CHECKOUT:	14-16
	SMOKE,HEAT,& ABNORMALITIES:	-14
	NORMAL CHECKOUT:	15-16
8	TROUBLESHOOTING:	-17
	TOUBLESHOOTING CHART:	18-21
9	MEMORY MAP:	-22
10	THEORY OF OPERATION:	23-26
	KEYBOARD:	-23
	FUNCTION GENERATOR:	-23
	ADDRESS/DATA ENTRY MODE:	-24
	SINGLE CYCLE & SINGLE INSTRUCTION:	-24
	CLEAR:	-25
	RUN/HALT:	-25
	BUS:	-25
	INITIALIZE:	-26

TABLE OF CONTENTS

Chapter		Page
11	OPERATING INSTRUCTIONS:	27-32
	TO DEPOSIT DATA:	-27
	TO EXAMINE DATA:	-29
	SINGLE INSTRUCTION:	-30
	SINGLE CYCLE:	
	INITIALIZE:	
12	SYSTEM EXPANSION:	33-34
13	PROGRAMMING:	35-46
	MEMORY 1 TEST PROGRAM:	39-40
	8 BIT SUBTRACTION:	-41
	8 BIT MULTIPLICATION:	-42
	ASCII TO HEX:	-43
	HEX TO ASCII:	-44
	0-99 NUMBER GUESSING GAME:	45-46
14	SOFTWARE/INSTRUCTION LISTING:	-47-
	INSTRUCTION LISTINGS:	
	COMPONENT LAYOUT DIAGRAM:	
	I.C. LOCATION/PARTS LIST:	
	DATA HANDLER BLOCK DIAGRAM:	
	WARRANTY:	
	SCHEMATICS:	

Welcome aboard, you are now one of thousands to take the big step into the fascinating world of microcomputing with the DATA HANDLER system.

By deciding to build your own computer you have an advantage over other computer hobbyist. You have saved yourself numerious dollars which can go into later expansion of your system, and after you have built this unique computer you will have a knowledge and understanding of microcomputer, and processors.

The DATA HANDLER is a complete microcomputer system on a single printed circuit board, (P.C.B.) designed around the Mos Technology 6502 Microprocessor.

This complete microcomputer system contains one thousand words (1K bytes) of random access memory (RAM) and one four bit parallel input port.

The DATA HANDLER contains a 26 keyboard switch hardware controlled front panel which will load data, examine data, perform single cycle and single instruction, initialize the system, run and halt the system all in hexidecimal (base 16) format.

This system contains a complete on board discrete L.E.D. hexidecimal address and data display which display the sixteen address bus lines and the eight data bus lines.

On the DATA HANDLER P.C.B. are three discrete L.E.D.'s which indicates whether the Data Handler front panel is in the address or data entry mide and whether it is halted or running.

The left rear portion of the P.C.B. contains two etched rows of fifty pads to facilitate the installation of one, one hundred pin ALTAIR/IMSAI type P.C.B. edge connector, or suitable connector to allow bus expansion. (Labeled S-100 on the P.C.B.).

These pads contain the tri-state buffered data, address, memory, I/O (input/output) signals, and timing signals used by the 8800 ALTAIR/IMSAI type peripheral boards. These bus lines are pin for pin compatable, and drive/receiver matched with the 8800 ALTAIR/IMSAI microcomputer peripherals.

These peripheral boards are the least expensive, most widely available, and offer the greatest variety in microcomputer peripherals available on the face of the earth.

The Data Handler is ideally suited as a computer instructing device due to its easy to understand and use qualities, yet retains the capabilities for gross system expansion by the advanced user.

Further, the DATA HANDLER is an excellent choice as an inexpensive tool for providing computer control for machinery, video displays, or electronic projects.

Note that the assembled Data Handler requires an external power source, +5 volts DC at 1.8 amps. However, a power supply capable of supplying 4 to 6 amps is recommended for future expansion.

Specifications

Uses the Mos Technology 6502 microprocessor.

Directly addresses 65K bytes of memory.

Contains 1K bytes of 500ns RAM on the P.C.B.(user selectable , address FC00-FFFF, or FE00-FFFF and 0000-01FF).

Direct memory access (DMA) circuitry and DMA acknowledge control lines.

Dual interrupt lines-one maskable, and one non-maskable.

Variable speed R/C clock, or optional crystal may be used for frequency stability to .2 mhz.

8800 ALTAIR/IMSAI identical tri-state address and data bus.

Uses single +5 volts supply @ 1.8 amps.

Discrete L.E.D.'s for address and data display.

Full function hardware controlled front panel.

Keyboard type data and control switches in hexi-decimal format.

OPTIONAL: High data rate (1200 baud) phase encoded cassette interface with on board dual I/O ports,(one 8 bit latching parallel input port with interrupt strobe, and one 8 bit latching parallel output port with a data flag).

System Information

System Speed: The maximum operating speed of the small or expanded DATA HANDLER system is determined by two factors. 1) C.P.U. speed, and 2) the memory access time.

The 6502 microprocessor is presently avaliable in models with operating speeds to 4 mhz (250ns cycle time). Due to the internal characteristics of the 6502, minimum memory access time (reading stable data from memory) is to be approximately half the cycle time. A system operating at 4 mhz needs a memory with a 125ns access time, 250ns for 2 mhz, 500ns for 1 mhz (supplied in the Data Handler complete kit), and so on. To double the normal data throughput of the Data Handler (operate at 2 mhz), a 6502A microprocessor should be installed, change R65 to a 1K ¼W, and change the memory I.C.'s (D1-D8) to 2102A-2.

For general system usage the kit supplied RC timing network is perfectly adequate. By setting the 50K ohm timing potentiometer (VR1) midway, a normal system clock speed of approximately 1 mhz will be in operation with a stability factor ± 20%. For time critical rock hard system stability greater than ± 20% a crystal may be incorporated into the processor timing.

Crystal Operation: If crystal frequency stability is desired for the Data Handler system a user supplied crystal may be installed in the two mounting holes on both sides of XL1 on the right rear portion of the P.C.B. This XL1 layout was designed to accomidate an AUGAT 8000-D crystal holder, however the crystal may be soldered directly on the P.C.B. then laid down onto the adjacent ground plane area, and the can (crystal case) soldered to the ground plane directly beneath it with a short piece of resistor lead or other wire.

I/O Port Information

The DATA HANDLER contains one four bit parallel input port. Access to this port is through the four pads between I.C. 4A & 5A, these are labelled as A,B, C,and D. When this input port is addressed by the Data Handler as location 7FFE, data present is displayed as the least signifigant bits- A is bit 0, b is Bit 1, C is bit 2, and D is bit 3.

These four bits are non latching and have an endless number of uses for status checking and device monitoring as well as receiving data from serial transmitting devices.

Notes:

DO NOT ASSEMBLE BEFORE READING COMPLETELY

PRE-ASSEMBLY NOTES:

(READ CAREFULLY)

1.) Assembling of the Data Handler P.C.B. consists of mounting components to the board. The parts location list (see index) contains lettering identifying where the components are to be mounted on the board along with their proper position and direction.

We recommend that you mount the components in groups, such as the integrated circuits, then the resistors, and then the capacitors. A step by step assembly procedure is provided for each major group of components.

2.) The Data Handler printed circuit board is industrial grade, double sided, with plated through holes. This means that in a number of locations, the foil paths on the top of the board are connected to the foil paths on the bottom through holes which have plating on the inside.

This plated through technique means you don't have to solder common foil areas on both sides. Soldering on the bottom side of the P.C.B. only normally will be adequate.

(NOTE: DO NOT USE ACID-CORE SOLDER)

3.) Due to the small foil around the printed circuit board holes and the small areas between the foil traces, you will have to use utmost soldering care to prevent solder bridges between adjacent foil areas. Use only a low-wattage soldering iron with a small tip. Do not use a soldering gun.

Keep the tip of your soldering iron clean (frequently wipe the tip on a damp clothe or solder sponge) and lightly coated with solder.

4.) The Data handler kit contains (8) 16 pin I.C. sockets, for the 2102 memory, and (1) 40 pin I.C. socket for the 6502 microprocessor. Sockets may be used for all the I.C.'s. However, for system reliability sake they are not recommended.

5.) The side with the name Data Handler is the side on which all the components are to be loaded. (component side)

6.) On all I.C.'s facing the rear of the P.C.B., pin 1 is left rear pin (denoted by the clipped pad on the P.C.B.). The microprocessor (E7), and the option (E12, and E16) face to the left with pin 1 located on the front left row of pins (denoted by clipped pad on the P.C.B.).

7.) I.C. B7 is the only one mounted with pin number one to the right rear facing toward the keyboard switches.

Pre-Assembly Notes

8.) When installing the*L.E.D.'s be absolutely sure of their polarity before installation. (see fig.3) Test them first if necessary, but be sure. The plastic base of the L.E.D. contains a small notch or flat area, this denotes the cathode of the L.E.D. as the lead closest to the flat area.

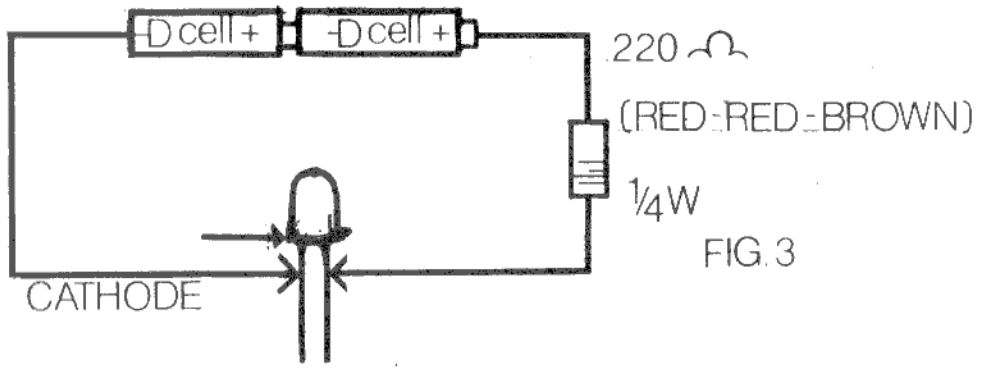


FIG. 3

9.) The cathode is to be connected to ground on all of the L.E.D.'s. On M4 thru M27 the cathodes face the front of the P.C.B., on M1, M2, and M3 they face toward the keyboard.

Caution

When inserting the static sensitive 6502 microprocessor into its socket, it is best done with one hand wrapped around a cold water pipe while the other hand gently inserts the I.C. into its socket. Avoid handling this I.C. until actual insertion time.

*(Light emitting diodes.)

Assembly Instruction

✓

1.() With the P.C.B. component side up (see pre-assembly note 5). Insert and solder pins 1 and 9 of the memory I.C. sockets (D1 thru D8) on the P.C.B..

2.() Insert the microprocessor socket and tack solder the corner pins (1 and 21) onto the P.C.B..

DO NOT INSTALL THE MEMORY I.C.'s OR THE MICRO-PROCESSOR IN THEIR SOCKETS YET.

3.() Refering to the component layout diagram, open one package of I.C.'s at a time, and tack solder the corner pins on all the I.C.'s except B7 onto the P.C.B..

4.() Install I.C. B7 according to pre-assembly note 7

5.() Check all I.C.'s and sockets for alignment, full insertion, bent under pins, and orientation (is B7 correct?).

6.() With all the I.C.'s being correct, go back and finish soldering all the I.C. pins. Make sure of good clean solder joints, and lack of solder bridge

7.() With all the I.C. sockets being correctly orientated, solder all the pins as in step 6.

8.() Again refering to your component layout diagram open the package of resistors and install them by groups in the following manner on the P.C.B.. Install the 4.7K ohm (yellow-violet-red) resistors in R1-R24 and solder in place.

9. () Insert R26-R51 (220 ohm, red-red-brown) and solder into place.

10.() Insert R54-R64 (1K ohm, brown-black-red) and solder into place.

11.() Again refering to the components layout diagram insert and solder the remaining resistors along with the 50K ohm potentiometer (VR1)

12.() With the resistors properly in place, mount the capacitors being careful to observe proper polarity on the electrolytics C20, and C21. (Plus on the P.C.B. shoud correspond with the plus end on the capacitor). (Refer to component layout).

13.() Double check the placement, polarity, alignment, etc. of the capacitors, and solder them into place. Trim (if needed) the excess lead length from these components from the underside of the board at this time.

14.() Insert the data L.E.D.'s M20 thru M27 (back row of 8 red L.E.D.'s) with the cathode (see pre-assembly notes). Toward the front edge of the P.C.B. (the long edge with numbers 1 thru 11 along it). Mount them all about 1/4" off surface of the P.C.B.. First solder one lead in place, then after making sure of their alignment and uniformit; solder the other lead.

15.() Install the row of 16 red address L.E.D.'s M4 thru M19 using the procedure previously described (step 14) for the data L.E.D.'s (cathodes toward the front edge of the P.C.B.). Also mount about 1/4" off the P.C.B. and solder into place.

16.() Install the red address L.E.D. M2 and red data L.E.D. M3 (located just to the left of the keyboard). Mount them flush with the P.C.B. with their cathodes towards the keyboard, and solder into place.

17.() Insert the yellow L.E.D. (M1) in its place directly below M2. Mount it with the cathode toward the keyboard and it also flush with the P.C.B., with alignment being correct solder it into place making sure to produce good solder joints.

18.() Install any user provided P.C.B. edge connector (Altair bus type), crystal, connectors for the I/O ports or what have you at this time and solder into place. Turn the P.C.B. over and ensure that all component leads are clipped and properly soldered.

19.() The P.C.B. without the switches installed, may be cleaned of any flux, dirt etc. at this time. At this point be careful not to let this sludge accumulate in the microprocessor or memory sockets. There are special chemicals available for cleaning P.C.B.'s, (freon TF, flux solvent, M.E.K. ETC.) but they should be used with proper ventilation. If the keyboard switches are already installed be extremely careful about keeping this flux remover off of them. It could result in destroying the switches. It is recommended that you remove the switches before the board is cleaned.

20.() Install the 26 keyboard switches one at a time soldering them from the bottom side when alignment, placement, and registration are verified with the component layout diagram.

21.() At this time insert the Data Handler P.C.B. into the top track of the Data Handler P.C.B. case.

22.() Connect and solder the wires from the user supplied power supply (plus 5 volts D.C. producing a minimum of 1.8 amps). The Data Handler requires only a plus 5 volts for power, however, consult the system expansion section (7) for other power configurations.

23.() Turn power to the Data Handler on at this time and quickly look around the P.C.B. and check for smoke, excessive heat or any abnormality. If any abnormality is found proceed immediately to smoke, heat, and abnormalities in the system checkout section.

24.() If a D.C. volt meter is available check for the plus 5 volts at pin 8 on the microprocessor socket and ground (0 volts) at pin 1. If a meter is not available wait an extra moment to ensure that an abnormality will not suddenly pop up later on.

25.() All steps being valid up to now, turn off the 5 volt supply to the Data Handler and install the memory I.C.'s into their sockets. Install the microprocessor into its socket (read how in the pre-assembly notes) being careful to use the correct pin number 1 for the registration.

26.() Ready to turn on the power? wait, once more check that the memory I.C.'s and the microprocessor are properly in place according to the component layout diagram.

27.() O.K. turn the power on and proceed to the system check out section.

Notes:

System Checkout

Smoke, Heat, & Abnormalities:

O.K., here you are with a problem, so lets be quick. None of the I.C.'s after 15 seconds (with power on), should be too hot to hold your finger on, so touch each one and remember which ones burned your finger. Turn the power off and check the hot (or smokey) I.C. for proper pin registration, alignment, solder blobs, a cut lead resting on the I.C. pins, etc. If I.C. B7 is hot chances are it's installed backwards. If all the I.C.'s are hot check the polarity of the power supply to the P.C.B., the polarity is either backwards (plus 5 volts is connected to gound on the P.C.B.), or the plus 5 volts is not actually plus 5 volts. If your power supply exhibited the smoke or abnormalities, chances are that there is a direct short on the P.C.B., this is easiest found with an ohm meter (look for solder blobs, lead touching under the P.C.B., etc).

Consult the troubleshooting section for further information.

Normal Checkout:

1.() With the HT (halt) button pressed, turn on the power supply to the Data Handler. The Data Handler will come up with a random address on the 16 address L.E.D.'s and random data on the 8 data L.E.D.'s. Either the addr or data L.E.D. (closest to the keyboard) will be on but not both, and the run L.E.D. (yellow L.E.D.) should not be on. This run L.E.D. indicates that the Data Handler is halted when not lit and running when it is on.

2.() Press the AD (address) switch, then the DA (data) switch. Ensure that the associated L.E.D. changes with each particular keystroke.

3.() Press CL (clear), the 16 address and 8 data L.E.D.'s should go off.

4.() Press AD, now press the 1 key. The address 0 L.E.D. should come on and stay on. This should be the only row L.E.D. on. Press the 2 key, the 1 should have shifted left into the next set of four digits and was replaced by the 2 entry. Sequentially enter the rest of the 16 keys. Ensure that the proper key data is entered into the right most set of four L.E.D.'s each time a key is pressed, verify that this keyswitch data is only entered once per keystroke.

5.() Press DA and sequentially enter all the 16 individual keys as in the previous step. Ensure that they enter the correct data row L.E.D.'s and shift left with each single keystroke.

Normal Checkout:

6.() Enter all F's into both the address and data rows of L.E.D.'s. All the data and address (24 total) L.E.D.'s should now be on. This test indicates dead or weak L.E.D.'s.

7.() With the Data Handler halted (press HT), press AD, press the 1 key. The address 0 L.E.D. (right most L.E.D. in the row) should come on. Press (deposit). The address row of L.E.D.'s should increment by one (base 16) each and every time DP is pressed.

8.() Duplicate test 7, however, press the EX (examine) keyswitch in place of the DP keyswitch. The address row of L.E.D.'s should increment by one (base 16) and only one each time EX is pressed

9.() Jump to the operating instructions section (sec 11)and become familiar with the correct procedures for operating your Data Handler. The loop program at the end of section 6 will verify proper system operation.

Notes:

Before consulting the troubleshooting chart to solve a particular problem, do the following;

(A) Give your Data Handler a complete visual inspection and ensure that all the I.C.'s are properly orientated and that the ones in the sockets are inserted sufficiently without bent-under, or out of socket pins.

(B) Check for solder bridges, cut leads laying on or under the P.C.B., and any solder flakes that can possibly cause a short to the Data Handler.

(C) If steps A and B failed to cure the problem list all visible symptoms and proceed to the troubleshooting chart. Remember it is virtually impossible to troubleshoot a problem without proper knowledge of the symptoms.

Part numbers-Refer to I.C. locations (C3 is an I.C. location not capacitor C3 unless stated otherwise.

Troubleshooting Chart

Symptom	Probable Cause
1. No lights,(L.E.D.'s).	1. Major short on the P.C.B.. 2. Power supply dead. check the +5 volts. 3. L.E.D.'s installed backwards.
2. Lights (L.E.D.'s) dim.	1. Insufficient voltage from power supply. 2. Insufficient current from power supply, 1.5 amps min. 3. Tri-state bus buffers bucking one onother. Either the front panel buffers (C3-C6) should be on or the C.P.U. buffers (E3,E4,E5,D9, D10) but not both. 4. S-100 bus is shorted.
3. Run L.E.D. not on or won't go off.	1. Microprocessor clock not running. 2. B9 not switching on switch contact. 3. Run L.E.D. is dead.

Symptom	Probable Cause
4. Address and data L.E.D.'s both on or won't switch off.	1. One or both L.E.D.'s are dead or installed backwards. 2. A1 not switching. 3. B10 not driving the L.E.D.'s.
5. CL won't clear the address and data L.E.D.'s.	1. The Data Handler is not in the halted mode. 2. C11 not functioning properly.
6. Incorrect data entered from front panel.	1. One or more dead L.E.D. (do test 6) 2. D1-D8 not fully in sockets. 3. Front panel buffers (C3-C6) not providing drive or shorted. 4. B6 is dead check strobe line 2 of 6.
7. A keystroke enters more than one character.	1. Capacitor C14 is leaky. 2. Sticky keyswitch. 3. Function generator (A2, A3,& B1) functioning improperly.

Symptom	Probable Cause
8. Not able to enter any address or data digits.	1. A6,A7, debounce circuitry is dead. 2. B6 is dead. 3. A8-A-11 one or all indicate a false key closure. 4. Stuck keyswitch.
9. Unable to enter any data.	1. Check the orientation of B7 (see pre-assemb note 7). 2. Improper procedure (Read operating instruction).
10. EX or DP increment more than one 16.	1. A 7(A) runs too fast (increase R53). 2. Keyswitch sticky.
11. Data Handler looses running sequence.	1. Clock speed is too fast. 2. Memory I.C. failure. 3. Interrupt line from inputport continuously held low. 4. Memory I.C. in backwards.

Troubleshooting

Symptom	Probable Cause
12. C.P.U. won't run at all.	1. The 6502 (I.C.E7) installed backwards. 2. VR1 broken or not installed.

Notes:

Memory Map

The Data Handler contains one thousand words (1K byte) of random access memory (ram) on the P.C.B.. The memory select feature allows for four pages of memory (FC00-FFFF) in the very top of the address space by putting a wire jumper from hole "T" to hole "R", or two pages in the top (FE00-FFFF) and two pages in the bottom (0000-01FF) by placing the wire jumper from hole "S" to hole "T". The cassette interface and full use of the Data Handler require the two bottom pages of memory (jumper "S" to "T").

When additional ram memory is used in the Data Handler system, it is recommended that this memory be located begining at address 0000 to fully utilize the stack function of the processor.

The address locations 7F00 thru 7FFF are decoded on the Data Handler P.C.B. as the I/O device code area. (The area in memory designated for input and output devices). It is recommended that memory not be decoded to this address area.

Notes:

The following is a detailed theory of operation of the complete Data Handler and its associated circuitry. Refer to the schematic diagrams and the block diagram while reading this section.

Keyboard:

The complete keyboard control is accomplished with the 10 control keyswitches, and the 16 data/address entry keyswitches (3 of 6). The set of data/address keyswitches are hardware hex encoded by A8 thru A-11 (2 of 6) and "switch sensed" by B6 (2 of 6). I.C. B6 sends a "switch sensed" strobe line signal to the one shot keyswitch debounce circuitry made up of A7, and A6 (1 of 6) which then is decoded by D12 (1 of 6) as being a strobe pulse from one of the 16 possible key-switches.

Function Generator:

The strobe pulse from the debounce circuitry activates a four pulse function generator formed from B1, A2, and A3 (2 of 6) and is mode decoded (address mode or data mode) by A4 and the second half of A1 (2 of 6). These four address pulses or 2 data pulses then latch (B7 for data and B2-B5 for address) the hexadecimal encoded data presently being displayed thru A8-A-11 (2 of 6) by any-one of the 16 possible keyswitches.

Address/Data Entry Mode:

Two control keyswitches (3 of 6) control the entry mode of the front panel. The entry mode is selected by either the DA (data) or AD (address) switches. This switch information sets or resets the RS latch formed from the first half of A1 (2 of 6) and decodes the function generator pulses as being either address or data loading pulses. The state of this RS latch is displayed on L.E.D.'s M2 and M3 (2 of 6) and can be changed at anytime.

Single Cycle & Single Instruction:

The SC (single cycle) and SI (single instruction) control keyswitches are debounced by A7 and A6 (1 of 6) and decoded by half of I.C. D12 (1 of 6). The debounc keystroke pulse from the SI (single instruction) key-switch sets the D15 (1 of 6) flip-flop which in turn allows the RDY line on the 6502 microprocessor to go t the high state. This enables the Data Handler to run until the D15 (1 of 6) flip-flop receives a sync pulse (start of new instruction) from the 6502 at which time D15 (1 of 6) resets and thru D13 (1 of 6) halts syste operation. The SC (single cycle) keyswitch is debounc and wired to RDY in the same way as is the single in-struction, however it allows the Data Handler to run o from the start of a Ø cycle to the start of the next cycle in which D14 (1 of 6) is reset and system opera-tion is halted.

Theory Of Operation

Clear:

The CL (clear) switch accomplishes two functions when it is actuated.

(1) It resets the RS latch C11 (1 of 6) which gives control of the S-100 bus to the front panel.

(2) It clears the front panel data (B7) and address (B2-B5) latches.

The CL, as is INT, HT, RN, AD, and DA swithces are not debounced since they are primarily function circuitry of the Data Handler front panel.

Run/Halt:

The RN (run) and HT (halt) keyswitches enable us to ultimately control the operation of the Data Handler. These swithces either set or reset run latch B9 (1 of 6) which in turn will halt operation of the C.P.U.. The state of this run latch is displayed by L.E.D. M1 (1 of 6) which when lit indicates that the C.P.U. is running. Further more when the run mode this run latch (B9) disables the debounce circuitry to kill any spurious ocillations which might lead to system noise.

Bus:

The S-100 bus is a UNI-directional bus which is the bulk work of the Data Handler. It is driven/received with tri-state buffers by both the front panel and the 6502 microprocessor. This bus drives the discreet L.E.D. display and connects the I/O ports, memory, 6502, front panel, and edge connector pads together.

Initialize:

The INT (initialize) keyswitch serves to accomplish two functions:

(1) it acts as an override to reset both the single instruction and single cycle circuitry.

(2) it pulls the reset line low on the 6502 microprocessor (4 of 6) which initiates the initialization procedure for the Data Handler (see the operating instructions for further information)

Notes:

To Deposit Data:

PRESS HT--Halt the Data Handler* (disregard if already halted).
PRESS CL--Clear the internal mode and the data holding registers.
PRESS AD--To switch the loading to the address lines (address L.E.D.'s).
KEY IN THE DESIRED ADDRESS TO BE MODIFIED.

PRESS DA--To switch the loading to the data lines (data L.E.D.'s).
KEY IN THE DESIRED DATA TO BE LOADED IN THE PRESENT DISPLAYED ADDRESS.

PRESS DP--This will deposit the data currently on the data lines into the address that is currently on the address lines.

After making the deposit into memory the Data Handler will automatically increment to the next address location, so that only the data needs to be altered and deposit pressed again for loading subsequent instructions.

NOTE* Due to the design of the 6502 mp, it is possible on power up for the Data Handler to enter an internal lock up mode that will not respond to halt. When this happens simply press INT while the HT switch is held down.

Operating Instructions

The Data Handler hardware controlled keyboard is composed of 26 keyboard switches which are divided into one group of ten, and one group of sixteen. The group of ten contains the function switches,

DP (DEPOSIT)
EX (EXAMINE)
CL (CLEAR)
AD (ADDRESS)
DA (DATA)
SI (SINGLE INSTRUCTION)
SC (SINGLE CYCLE)
HT (PROCESSOR HALT)
RN (PROCESSOR RUN)
INT (INITIALIZE)

The group of sixteen switches contains 0 thru f (hexidecimal) for data and address loading.

The Data Handler information display is comprised of 27 discrete L.E.D.'s (light emitting diodes), 26 red and one yellow. This display is divided into a row (bottom row) of 16 L.E.D.'s which represent in hex format the current address, and the top row of 8 represent the current data. The M.S.B. (most signifigant bit) of each row is the left most L.E.D..

In the group of three L.E.D.'s to the left of the keyboard, the top one labeled data indicates that the keyboard switches 0 thru F will be displayed on the data L.E.D.'s. Pressing the AD function switch will change the loading to the address L.E.D.'s. The bottom yellow L.E.D. indicaties that the DATA HANDLER is free running when it is on, and in the halted mode when it is off.

The Data Handler has a deposit lock out when in the examine mode to prevent inadvertant depositing of data. To switch from the examine mode to the deposit mode or vise versa the Data Handler must be exercised through one complete instruction (press SI once) before the desired mode can be selected.

To Examine Data

PRESS HT--Halt the Data Handler (disregard if already halted).
PRESS EX--Put the internal registers in the examine mode.
PRESS AD--To place the keyboard entry on the address lines.

Load the address to be examined onto the address lines, and the data L.E.D.'s will then display the data in that location of memory.

When in the examine mode, each time the EX function switch is pressed the next address and its data will be displayed.

Depositing and examining data will not affect the PC counter of the 6502 microprocessor, in that the Data Handler can be halted, data examined and or deposited without disturbing the existing running program sequence.

Single Instruction :

PRESS HT--Halt the Data Handler (disregard if already halted).
PRESS SI--The Data Handler will execute the next instruction and stop. Each time the SI switch is pressed the Data Handler will execute the next instruction and stop.

Single Cycle:

PRESS HT--Halt the Data Handler (disregard if already halted).
PRESS SC--The Data Handler will execute one cycle of the next instruction and stop.

Using the SI and SC modes to "walk" through a newly loaded or newly written program can save countless hours in the debuging of software, in that branches, jumps, all steps of the program are displayed and executed under full control of the operator.

Initialize:

When the INT switch is pressed the 6502 MP will begin the initialize process which consist of 6 cycles in which it will dump its present address on the stack and load its program counter (PC) from its initialize vector locations. The Data Handler will then begin executing at that memory location when run is pressed.

To demonstrate this let's load a small loop program.

Deposit the following data starting at address location FC00 by using the format discussed for depositing data.

ADDRESS	DATA	INSTRUCTION
FE00	EA	NOP
01	EA	NOP
02	4C	JMP
03	00	LOW ORDER ADDRESS
04	FE	HIGH ORDER ADDRESS

In the halt condition, to instruct the Data Handler to run the above program you would simply load the low order address of your program starting location into the first location of the initialize vector (FFFC), and the high order address into the second location of the initialize vector (FFFD), shown below.

FE00

HIGH ORDER ADDRESS | LOW ORDER ADDRESS

SO WE LOAD;

ADDRESS	DATA	
FFFC	00	(LOW ORDER ADDRESS)
FFFD	FE	(HIGH ORDER ADDRESS)

To get the Data Handler to begin execution of our loop program press INT, begin pressing SC and watch the Data Handler step through the initialize process (6 steps). It will step to the initialize vectors (FFFC, and FFFD) and arrive at its new starting address FE00 which should be displayed on the address display.

If your address is displaying 00FE you loaded your low and high order address backwards, if so go back and try again. If you arrive at the correct address (FE00), continue to single cycle through your loop. If you are convinced that it is looping properly you can discontinue "walking" through and let it free run by pressing RN. The Data Handler is now executing your loop program.

You will notice that the Data Handler continuously displays the exact data that we loaded in our loop program. Excellent! It's running our program exactly as we wrote it. We instructed it to perform two NOP's (EA, no operations) and then JMP (4C, jump to a new starting location) to begin execution of new data, which as we planned it turned out to be the same program we left.

Very cleaver is this program, this could be the basis for some sort of keyboard monitor or other program that looped while waiting for an interrupt from an external device (possibly connected onto our B input port).

The Data Handler address, data, and control lines exit the P.C.B. by way of two rows of fifty pads on the rear left portion of the Data Handler P.C.B..

These bus lines are pin for pin compatable, and driver/receiver matched with the 8800 ALTAIR/ IMSAI microcomputer peripherals.

Limited system expansion may be achieved with the addition of a 100 pin connector on the bus line pads on the left rear portion of the Data Handler P.C.B. and the installation of any particular 8800 ALTAIR/IMSAI type of peripheral board which fits the individual need.

Larger system expansion may be accomplished with the use of an ALTAIR type mother board or a suitable card cage set-up mounted next to the Data Handler P.C.B. with the bus lines connected to the mother board with ribbon cable.

Another excellent choice of system expansion would be the hard wire mounting of an Altair expander board directly perpindicular below the Data Handler P.C.B.. This would enable the use of more than one peripheral board.

The Data Handler may be used with a power supply greater than +5 volts D.C. when it is configured in such a manner as to incorporate the mounting of a +5 volt 5A regulator on the underside of the P.C.B.. The left rear part of the P.C.B. directly adjacent to the bus line pads, is laid out to accept a TO-3 78H05, or LM 323, (+5 volts @ amps) type regulator.

This regulator configuration is ideal when used with an Altair type (+8 volts) power supply to provide power for peripheral boards of the 8800 Altair/ IMSAI type. Note that these peripherals were designed to be powered by +8 volts with regulators dropping the voltage to +5 volts on each individual board. However, jumpers may be used in place of these regulators and a +5 volt power supply may be incorporated.

The Data Handler L.E.D. displays are driven directly from the address and data bus lines. When the system is expanded to a large degree it is suggested that a video type peripheral be added, or that tha L.E.D.'s be buffered, or seven segment displays added to prevent over loading of these bus lines.

Notes:

Programming

The writing of programs or programming as it's called by those with the ability, is nothing more than a logical plan for the computer to follow. This logical plan or program as we call it, consists of a sequence of instructions that the Data Handler understands and must obey precisely as it is written.

Without a program to run in, the Data Handler is lost, useless, and aimlessly searching. Our job (and privilege) is to take this mysterious little friend into our care and give it guidance and a direction in life (the kit builders are morally obligated since they brought it into the world).

The only problem we face is that at this stage of the game our Data Handler can't speak or understand a word of our language and we as newcomers (with little or no computer experience) don't exactly know how to communicate with him. This situation is comparable to being in Tia Juana Mexico (minus the flies, and tacos) for the first time.

We are in a foreign land that speaks a foreign language, and for those of you out there wondering, I'll tell you what this language is. It is called machine language or machine code (as it's called by the advanced programmers). Very cleaver the way these machines have a language named for them (yes! they are machines).

O.K. since our Data Handler does'nt have the ability to learn our language we had better learn to communicate with it in machine language. So, once again just like in Tia Juana we reach for our translation guide which in this case happens to be our instruction listings which can be found in section 14 of our owner's manual.

This instruction listing gives us a brief explanation of what the different 55 instructions do. For a very complete explanation purchase and read a copy of the Mos Technology 6502 programming manual).

Take a moment to read and understand these Different instructions. It is with this instruction set that you'll command your Data Handler. If anyone of these seem unclear as to end result or location, try it and watch (through the single cycle procedure) exactly where the Data Handler steps to or loads data from. Alright- now that we feel that we have a relatively good understanding of the instruction set lets continue on.

Programs as I stated before are nothing more than a sequential series of instructions from our instruction set. This sequential series can begin anywhere in the address space that we have memory available to store it, so the Data Handler will always increments to the next higher address for its next instruction.

An example of this would be if we started the Data Handler in our loop program (see under operating instructions). This program starts at address FE00, so using the procedure described in the operating instructions let's load this program and secondly load our program address into the initialization address locations (FFFC for the lower half of the location and FFFD for the upper half) Press INT and press SI twice- this will put the Data Handler at the start of our loop program (FE00).

Our first instruction instructs the Data Handler to perform a NOP, so we have instructed it in its own language by loading that memory location with EA. Anything short of exactly that data will not give your computer the proper instruction.

You will notice from the instruction set write up, that under No. bytes it has a 1, that means it needs only one byte of data (one memory location) to give it this instruction. After performing this instruction (no operation) the Data Handler will immediately go to the next instruction in our program. As I mentioned before it will increment the address and perform the next instruction which as it turns out happens to be another NOP (at address location FE01).

When the Data Handler increments to FE02 it is instructed to perform a jump (4C), this instruction is a 3 No. bytes instruction and the computer is going to jump to and begin program execution at the address specified by the two following bytes of data stored in the two following locations (FE03 and FE04).

If you did not store the proper data don't expect the Data Handler to jump to the address you wanted instead you can be certain that it will jump to and begin program execution at the address location specified by the data in the two following bytes.

Load and examine the running sequence of some of the simpler sample programs in this software section of the manual and with a little bit of practice programming will become relatively easy.

Notes:

MEMORY 1 TEST PROGRAM

This program is designed to test individual 256 word blocks of memory, this is written to reside in the top block of memory ($FF00_{16}$-$FFFF_{16}$) which for obvious reasons should not be tested without relocation of the memory test program 1.

```
FFFC-00
FFFD-FF

FF00-AD,10,FF          LDA ABS.          PROGRAM
  03-8D,16,FF          STA ABS.          FORMATTING
  06-AD,0F,FF          LDA ABS.
  09-8D,15,FF          STA ABS.
  0C-A9,XX             LDA 1MM.          TESTING
  0E-8D,XX,BLOCK       STA ABS.          LOOP
  11-EA                NOP
  12-EA                NOP
  13-EA                NOP
  14-CD,XX,XX          CMP ABS.
  17-D0,13             BNE               TEST FOR EQUA
  19-EE,0D,FF          INC ABS.          INC. DATA
  1C-C9,00             CMP 1MM.          TEST OVER FLO
  1E-F0,03             BEQ
  20-4C,0C,FF          JMP ABS.
  23-EE,0F,FF          INC ABS.          INC. ADDRESS
  26-EE,15,FF          INC ABS.
  29-4C,0C,FF          JMP ABS.
  2C-8D,33,FF          STA ABS           DUMP BAD DATA
  2F-EA                NOP
  30-4C,2F,FF          JMP ABS.          BAD LOOP
  33-XX
```

MEMORY 1 TEST PROGRAM

Load the memory block address to be tested into memory location $FF10_{16}$ (EXAMPLE: FC_{16} will test FC00 to FCFF). Press INT then press RN. Under normal running conditions A5,A6, and A7 will slowly increment off and on. If a data error is encountered in the test, the Data Handler will enter the bad loop which is easily seen as A6 and A7 continuously off.

For the following data examine these locations.

FF0D-CORRECT DATA
FF33-BAD DATA
FF0F-LO ADR. OF BAD DATA
FF10-HIGH ADR. OF BAD DATA

To execute the above program press INT and then press RN.

X= DON'T CARE.

8 BIT SUBTRACTION

This program will perform an 8 Bit subtraction and store the result if negative in memory location 0002, or if the result is a positive value in the following location(0003).

LOAD THE FOLLOWING DATA

```
0000-MINUEND
0001-SUBTRAHEND
0002-00
0003-00

FF00-D8       START    CLD          initialize the C.P.U.
FF01-38                SEC           internal registers
FF02-A5 00             LDA Z        load the minuend
FF04-E5 01             SBC Z        sub. the subtrahend
FF06-10 08             BPL   PS
FF08-E9 00             SBC I
FF0A-49 FF             EOR I        invert neg. result
FF0C-85 02             STA Z 02     store neg. result
FF0E-D0 02             BNE   DONE LOOP
FF10-85 03    PS:      STA Z        store pos. result
FF12-EA  DONE LOOP:    NOP          done
FF13-4C 12 FF          JMP   DONE LOOP

FFFC-00
FFFD-FF
```

8 BIT MULTIPLICATION

This program will multiply two Hexidecimal (base 16) encoded numbers and store the lower half of the result in memory location 0002, and the upper half in memory location 0003.

LOAD THE FOLLOWING DATA

```
00-MULTIPLICAND
01-MULTIPLIER
02-00
03-00
```

```
00-18       START    CLC         initialize the C.P.U
01-D8                CLD          internal registers
02-A5 00             LDA Z       load multiplicand
04-A6 01             LDX Z       load multiplier
06-CA       01:      DEX
07-F0 08             BEQ  02     branch to store
09-65 00             ADC Z       multiplicand
0B-90 F9             BCC  01
0D-E6 03             INC Z       upper half result
0F-D0 F5             BNE  01
11-85 02    02:      STA Z       lower half result
13-4C 16 FE          JMP  DONE LOOP   done
16-EA    DONE LOOP:  NOP
17-4C 16 FE          JMP         jump on self
```

```
FC-00
FD-FE
```

First press INT, then press run, press HT and then press EX.

The answer will be in address locations 0002, and 0003 respectively.

ASCII TO HEX

This ASCII to hexidecimal conversion program will assume at its start that the ASCII number to be converted is present in the accumulator, and at its end the Hex equivilent is in its place in the accumulator. A ?-3F (question mark) indicates a non-equivilent character.

ASCII	HEX		
30	= 00	35 = 05	41 = 0A
31	= 01	36 = 06	42 = 0B
32	= 02	37 = 07	43 = 0C
33	= 03	38 = 08	44 = 0D
34	= 04	39 = 09	45 = 0E
			46 = 0F

```
FF00-D8       START    CLD
FF01-B8                CLV          initialize the C.P.U.
FF02-38                SEC           internal registers
FF03-E9 30             SBC I        subtract 30
FF05-30 0F             BMI    01    non-hex (less 30)
FF07-C9 0A             CMP I
FF09-30 0A             BMI    02    hex numeric value
FF0B-E9 07             SBC I
FF0D-C9 0A             CMP I
FF0F-30 05             BMI    01    non-hex
FF11-C9 10             CMP I
FF13-10 01             BPL    01    non-hex (greater 46)
FF15-60       02:      RTS          return from sub.
FF16-A9 3F    01:      LDA I        load ? (3F)
FF18-60                RTS          return from sub.
```

HEX TO ASCII

This Hexidecimal to ASCII conversion program assumes at its start that the Hex character to be converted is present in the accumulator, and at its end the ASCII equivilent is in the accumulator. This program is the exact opposite of the ASCII to hex conversion program on the opposite page.

```
FE00-38       START    SEC          set carry bit
FE01-C9 0A             CMP I
FE03-10 03             BPL   01     hex numeric
FE05-69 30    02:      ADC I        add 30
FE07-60                RTS          return from sub.
FE08-69 06    01:      ADC I        add 06
FE0A-D0 F9             BNE   02     all branch
```


0 to 99 Number Guessing Game

The 0 to 99 number guessing game is a simple game program that takes a few minutes to load and play. The Data Handler randomly generates a number which the player tries to guess by loading his guess number (0-99) into memory location FC10. The average number of tries to correctly guess the randomly generated number is six. Try it and see if you can beat the average number of guesses.

```
FFFC-00
FFFD-FE

FE00-F8              SED
  01-A9,0E           LDA 1MM  ---- STORE NEW INT LOC.
  03-8D,FC,FF        STA ABS  ----
  06-69,01           ADC 1MM  ----
  08-AA              TAX      ----
  09-4C,06,FC        JMP      ---- RANDOM NUMBER GEN.
  0C-B8              CLV           CLEAR OV FLAG
  0D-18              CLC           CLEAR CARRY FLAG
  0E-8A              TXA
  0F-E9,GUESS        SBC 1MM       CALCULATE DIFFERENCE
  11-F0,2D           BEQ           BRANCH TO CORRECT
  13-10,0B           BPL           BRANCH TO LOW
  15-30,69           BM1           BRANCH TO HIGH

FE20-EA              NOP  ---------- LOW GUESS
  21-4C,20,FE        NOP  ----

FE40-A2,00           LDA 1MM  ------ CORRECT GUESS
  42-8D,FC,FF        STA ABS  ------
  45-4C,45,FE        JMP

FE80-EA              NOP  ---------- HIGH GUESS
  81-4C,80,FE        JMP  ----
```

Load the preceding program data into the associated memory locations (using the procedure described in the Dat Handler manual) press INT and RN. The Data Handler will enter a portion of the program which generates a random decimal number. Press HT (halt the Data Handler), press C (to switch control to the front panel), press ADR, now ent the memory address location (FC10) in which we will place our guess number.

With FC10 on the address lights press DA, now enter your guess onto the data lights (the program will accept only decimal number between 0 and 99 for the guess). With your guess number on the data lights press DP. Press INT and RN, or to observe the Data Handler calculating the results of your guess press INT, and then the Data Handler may be walked through the program (by repeatedly pressing SC or SI).

The Data Handler will ultimately end up in one of thre loops, the low guess (your guess was low) is most easily seen with the address light A5 on continuously, the high guess (your guess was high) is most easily seen with the address light A7 on continuously. The correct guess (your guess was 100% absolutely correct) is most easily seen with the address light A6 on continuously.

If you correctly guessed the computers random number in 5 or less tries you have done it above the average. If not try again.

To restart the game press INT. This vectors the Data Handler back into the random number generator to await your first guess.

CONSULT THE MOS TECHNOLOGY 6502 PROGRAMMING MANUAL FOR USE OF THE FOLLOWING INSTRUCTION LIST AND FURTHER INFORMATION.

INSTRUCTION LIST
ALPHABETIC BY MNEMONIC
WITH OP CODES, EXECUTION CYCLES
AND MEMORY REQUIREMENTS.

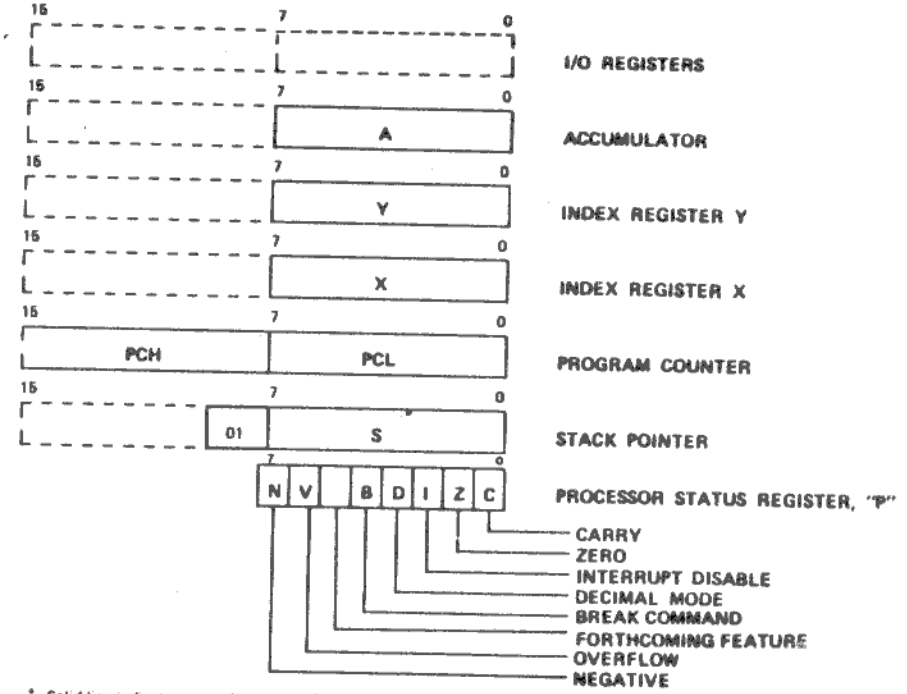
INSTRUCTION ADDRESSING MODES AND RELATED EXECUTION TIMES (in clock cycles)

	Accumulator	Immediate	Zero Page	Zero Page, X	Zero Page, Y	Absolute	Absolute, X	Absolute, Y	Implied	Relative	(Indirect, X)	(Indirect), Y	Absolute Indirect
ADC		2	3	4		4	4*	4*			6	5*	
AND		2	3	4		4	4*	4*			6	5*	
ASL	2		5	6		6	7						
BCC										2**			
BCS										2**			
BEQ										2**			
BIT			3			4							
BMI										2**			
BNE										2**			
BPL										2**			
BRK													
BVC										2**			
BVS										2**			
CLC													
CLD													
CLI													
CLV													
CMP		2	3	4		4	4*	4*			6	5*	
CPX		2	3			4							
CPY		2	3			4							
DEC			5	6		6	7						
DEX									2				
DEY									2				
EOR		2	3	4		4	4*	4*			6	5*	
INC			5	6		6	7						
INX									2				
INY									2				
JMP						3							5

	Accumulator	Immediate	Zero Page	Zero Page, X	Zero Page, Y	Absolute	Absolute, X	Absolute, Y	Implied	Relative	(Indirect, X)	(Indirect),Y	Absolute Indirect
JSR						6							
LDA		2	3	4		4	4*	4*			6	5*	
LDX		2	3		4	4		4*					
LDY		2	3	4		4	4*						
LSR	2		5	6		6	7						
NOP									2				
ORA		2	3	4		4	4*	4*			6	5*	
PHA									3				
PHP									3				
PLA									4				
PLP									4				
ROL	2		5	6		6	7						
RTI									6				
RTS									6				
SBC		2	3	4		4	4*	4*			6	5*	
SEC									2				
SED									2				
SEI									2				
STA			3	4		4	5	5			6	6	
STX*			3		4	4							
STY**			3	4		4							
TAX									2				
TAY									2				
TSX									2				
TXA									2				
TXS									2				
TYA									2				

* Add one cycle if indexing across page boundary.
** Add one cycle if branch is taken. Add one additional if branching operation crosses page boundary

PROGRAMMING MODEL MCS650X



* Solid line indicates currently available features
Dashed line indicates forthcoming members of family

ADC

Add memory to accumulator with carry

ADC

Operation: A + M + C → A, C

N Z C I D V
✓ ✓ ✓ – – ✓

(Ref: 2.2.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	ADC # Oper	69	2	2
Zero Page	ADC Oper	65	2	3
Zero Page, X	ADC Oper, X	75	2	4
Absolute	ADC Oper	6D	3	4
Absolute, X	ADC Oper, X	70	3	4*
Absolute, Y	ADC Oper, Y	79	3	4*
(Indirect, X)	ADC (Oper, X)	61	2	6
(Indirect), Y	ADC (Oper), Y	71	2	5*

* Add 1 if page boundary is crossed.

AND

"AND" memory with accumulator

AND

Logical AND to the accumulator

Operation: A ∧ M → A

N Z C I D V
✓ ✓ – – – –

(Ref: 2.2.3.0)

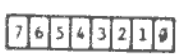
Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	AND # Oper	29	2	2
Zero Page	AND Oper	25	2	3
Zero Page, X	AND Oper, X	35	2	4
Absolute	AND Oper	2D	3	4
Absolute, X	AND Oper, X	3D	3	4*
Absolute, Y	AND Oper, Y	39	3	4*
(Indirect, X)	AND (Oper, X)	21	2	6
(Indirect), Y	AND (Oper),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

* Add 1 if page boundary is crossed.

ASL

ASL Shift Left One Bit (Memory or Accumulator)

ASL

Operation: C ←  ← 0

N Z C I D V
✓ ✓ ✓ – – –

(Ref: 10.2)

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

BCC

BCC Branch on Carry Clear

BCC

Operation: Branch on C = 0

N Z C I D V
– – – – – –

(Ref: 4.1.1.3)

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

* Add 1 if branch occurs to same page.

* Add 2 if branch occurs to different page.

BCS

BCS *Branch on carry set*

BCS

Operation: Branch on C = 1

N Z C I D V
– – – – – –

(Ref: 4.1.1.4)

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

* Add 1 if branch occurs to same page.
* Add 2 if branch occurs to next page.

BEQ

BEQ *Branch on result zero*

BEQ

Operation: Branch on Z = 1

N Z C I D V
– – – – – –

(Ref: 4.1.1.5)

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

* Add 1 if branch occurs to same page.
* Add 2 if branch occurs to next page.

BIT

BIT *Test bits in memory with accumulator*

BIT

Operation: A ∧ M, M_7 → N, M_6 → V

Bit 6 and 7 are transferred to the status register. If the result of A∧M is zero then Z = 1, otherwise Z = 0

N Z C I D V
M_7 ✓ – – – M_6

(Ref: 4.2.1.1)

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

BMI

BMI *Branch on result minus*

BMI

Operation: Branch on N = 1

N Z C I D V
– – – – – –

(Ref: 4.1.1.1)

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

* Add 1 if branch occurs to same page.
* Add 2 if branch occurs to different page.

BNE

BNE *Branch on result not zero*

BNE

Operation: Branch on Z = 0

N Z C I D V
– – – – – –

(Ref: 4.1.1.6)

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

* Add 1 if branch occurs to same page.
* Add 2 if branch occurs to different page.

BPL

BPL *Branch on result plus*

BPL

Operation: Branch on N = 0

N Z C I D V
– – – – – –

(Ref: 4.1.1.2)

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

* Add 1 if branch occurs to same page.
* Add 2 if branch occurs to different page.

BRK

BRK *Force Break*

Operation: Forced Interrupt PC + P +

N Z C I D V
- - - 1 - -

(Ref: 9.11)

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

1. A BRK command cannot be masked by setting I.

BVC

BVC *Branch on overflow clear*

Operation: Branch on V = 0

N Z C I D V
- - - - - -

(Ref: 4.1.1.8)

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

* Add 1 if branch occurs to same page.
* Add 2 if branch occurs to different page.

BVS

BVS *Branch on overflow set*

Operation: Branch on V = 1

N Z C I D V
- - - - - -

(Ref: 4.1.1.7)

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

* Add 1 if branch occurs to same page.
* Add 2 if branch occurs to different page.

CLC

CLC *Clear carry flag*

Operation: 0 → C

N Z C I D V
- - 0 - - -

(Ref: 3.0.2)

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

CLD

CLD *Clear decimal mode*

Operation: 0 → D

N Z C I D V
- - - - 0 -

(Ref: 3.3.2)

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

CLI

CLI *Clear interrupt disable bit*

Operation: 0 → I

N Z C I D V
- - - 0 - -

(Ref: 3.2.2)

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

CLV

CLV *Clear overflow flag*

Operation: Ø → V

N Z C I D V
– – – – – Ø

(Ref: 3.6.1)

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

CMP

CMP *Compare memory and accumulator*

Operation: A - M

N Z C I D V
✓ ✓ ✓ – – –

(Ref: 4.2.1)

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

* Add 1 if page boundary is crossed.

CPX

CPX *Compare Memory and Index X*

Operation: X - M

N Z C I D V
✓ ✓ ✓ – – –

(Ref: 7.8)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	CPX #Oper	E0	2	2
Zero Page	CPX Oper	E4	2	3
Absolute	CPX Oper	EC	3	4

CPY

CPY *Compare memory and index Y*

Operation: Y - M

N Z C I D V
✓ ✓ ✓ – – –

(Ref: 7.9)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	CPY #Oper	C0	2	2
Zero Page	CPY Oper	C4	2	3
Absolute	CPY Oper	CC	3	4

DEC

DEC *Decrement memory by one*

Operation: M - 1 → M

N Z C I D V
✓ ✓ – – – –

(Ref: 10.7)

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

DEX

DEX *Decrement index X by one*

Operation: X - 1 → X

N Z C I D V
✓ ✓ – – – –

(Ref: 7.6)

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

DEY

DEY *Decrement index Y by one*

Operation: Y - 1 → Y

N Z C I D V
✓ ✓ – – – –

(Ref: 7.7)

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

EOR

EOR *"Exclusive-Or" memory with accumulator*

Operation: A ∀ M → A

N Z C I D V
✓ ✓ – – – –

(Ref: 2.2.3.2)

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

* Add 1 if page boundary is crossed.

INC

INC *Increment memory by one*

Operation: M + 1 → M

N Z C I D V
✓ ✓ – – – –

(Ref: 10.6)

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

INX

INX *Increment Index X by one*

Operation: X + 1 → X

N Z C I D V
✓ ✓ – – – –

(Ref: 7.4)

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

INY

INY *Increment Index Y by one*

Operation: Y + 1 → Y

N Z C I D V
✓ ✓ – – – –

(Ref: 7.5)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	INY	C8	1	2

JMP

JMP *Jump to new location*

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

N Z C I D V
– – – – – –

(Ref: 4.0.2)

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

JSR

JSR *Jump to new location saving return address*

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

N Z C I D V
– – – – – –

(Ref: 8.1)

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

LDA

LDA *Load accumulator with memory*

Operation: M → A

N Z C I D V
✓ ✓ – – – –

(Ref: 2.1.1)

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

* Add 1 if page boundary is crossed.

LDX

LDX *Load index X with memory*

Operation: M → X

N Z C I D V
✓ ✓ – – – –

(Ref: 7.0)

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

* Add 1 when page boundary is crossed.

LDY

LDY *Load index Y with memory*

Operation: M → Y

N Z C I D V
✓ ✓ – – – –

(Ref: 7.1)

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

* Add 1 when page boundary is crossed.

LSR

LSR *Shift right one bit (memory or accumulator)*

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

N Z C I D V
0 ✓ ✓ – – –

(Ref: 10.1)

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

NOP

NOP *No operation*

Operation: No Operation (2 cycles)

N Z C I D V
– – – – – –

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

ORA

ORA *"OR" memory with accumulator*

Operation: A ∨ M → A

N Z C I D V
✓ ✓ – – – –

(Ref: 2.2.3.1)

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

* Add 1 on page crossing

PHA

PHA *Push accumulator on stack*

Operation: A ↓

N Z C I D V
– – – – – –

(Ref: 8.5)

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

PHP

PHP *Push processor status on stack*

Operation: P↓

N Z C I D V
– – – – – –

(Ref: 8.11)

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

PLA

PLA *Pull accumulator from stack*

Operation: A ↑

N Z C I D V
✓ ✓ – – – –

(Ref: 8.6)

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

PLP

PLP *Pull processor status from stack*

Operation: P ↑

N Z C I D V
From Stack

(Ref: 8.12)

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

ROL

ROL *Rotate one bit left (memory or accumulator)*

Operation: M or A — 7 6 5 4 3 2 1 0 ← C ←

N Z C I D V
✓ ✓ ✓ – – –

(Ref: 10.3)

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

RTI

RTI *Return from interrupt*

Operation: P↑ PC↑

N Z C I D V

From Stack

(Ref: 9.6)

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

RTS

RTS *Return from subroutine*

Operation: PC↑, PC + 1→ PC

N Z C I D V

(Ref: 8.2)

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

SBC

SBC *Subtract memory from accumulator with borrow*

Operation: A - M - $\overline{C}$ → A

Note: $\overline{C}$ = Borrow (Ref: 2.2.2)

N Z C I D V

✓ ✓ ✓ – – ✓

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

* Add 1 when page boundary is crossed.

SEC

SEC *Set carry flag*

Operation: 1 → C

N Z C I D V

(Ref: 3.0.1)

– – 1 – – –

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

SED

SED *Set decimal mode*

Operation: 1 → D

N Z C I D V

(Ref: 3.3.1)

– – – – 1 –

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

SEI

SEI *Set interrupt disable status*

Operation: 1 → I

N Z C I D V

(Ref: 3.2.1)

– – – 1 – –

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

STA

STA *Store accumulator in memory*

Operation: A → M

N Z C I D V
– – – – – –

(Ref: 2.1.2)

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

STX

STX *Store index X in memory*

Operation: X → M

N Z C I D V
– – – – – –

(Ref: 7.2)

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

STY

STY Store index Y in memory

Oper tion: Y → M

N Z C I D V
– – – – – –

(Ref: 7.3)

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

TAX

TAX *Transfer accumulator to index X*

Operation: A → X

N Z C I D V
✓ ✓ – – – –

(Ref: 7.11)

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

TAY

TAY *Transfer accumulator to index Y*

Operation: A → Y

N Z C I D V
✓ ✓ – – – –

(Ref: 7.13)

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

TYA

TYA *Transfer index Y to accumulator*

Operation: Y → A

N Z C I D V
✓ ✓ – – – –

(Ref: 7.14)

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

TSX

TSX *Transfer stack pointer to index X* **TSX**

Operation: S → X

N Z C I D V
✓ ✓ – – – –

(Ref: 8.9)

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

TXA

TXA *Transfer index X to accumulator* **TXA**

Operation: X → A

N Z C I D V
✓ ✓ – – – –

(Ref: 7.12)

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

TXS

TXS *Transfer index X to stack pointer* **TXS**

Operation: X → S

N Z C I D V
– – – – – –

(Ref: 8.8)

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

ØØ - BRK
Ø1 - ORA - (Indirect,X)
Ø2 - Future Expansion
Ø3 - Future Expansion
Ø4 - Future Expansion
Ø5 - ORA - Zero Page
Ø6 - ASL - Zero Page
Ø7 - Future Expansion
Ø8 - PHP
Ø9 - ORA - Immediate
ØA - ASL - Accumulator
ØB - Future Expansion
ØC - Future Expansion
ØD - ORA - Absolute
ØE - ASL - Absolute
ØF - Future Expansion
1Ø - BPL
11 - ORA - (Indirect),Y
12 - Future Expansion
13 - Future Expansion
14 - Future Expansion
15 - ORA - Zero Page,X
16 - ASL - Zero Page,X
17 - Future Expansion
18 - CLC
19 - ORA - Absolute,Y
1A - Future Expansion
1B - Future Expansion
1C - Future Expansion
1D - ORA - Absolute,X
1E - ASL - Absolute,X
1F - Future Expansion

2Ø - JSR
21 - AND - (Indirect,X)
22 - Future Expansion
23 - Future Expansion
24 - BIT - Zero Page
25 - AND - Zero Page
26 - ROL - Zero Page
27 - Future Expansion
28 - PLP
29 - AND - Immediate
2A - ROL - Accumulator
2B - Future Expansion
2C - BIT - Absolute
2D - AND - Absolute
2E - ROL - Absolute
2F - Future Expansion
3Ø - BMI
31 - AND - (Indirect),Y
32 - Future Expansion
33 - Future Expansion
34 - Future Expansion
35 - AND - Zero Page,X
36 - ROL - Zero Page,X
37 - Future Expansion
38 - SEC
39 - AND - Absolute,Y
3A - Future Expansion
3B - Future Expansion
3C - Future Expansion
3D - AND - Absolute,X
3E - ROL - Absolute,X
3F - Future Expansion

4Ø - RTI
41 - EOR - (Indirect,X)
42 - Future Expansion
43 - Future Expansion
44 - Future Expansion
45 - EOR - Zero Page
46 - LSR - Zero Page
47 - Future Expansion
48 - PHA
49 - EOR - Immediate
4A - LSR - Accumulator
4B - Future Expansion
4C - JMP - Absolute
4D - EOR - Absolute
4E - LSR - Absolute
4F - Future Expansion
5Ø - BVC
51 - EOR - (Indirect),Y
52 - Future Expansion
53 - Future Expansion
54 - Future Expansion
55 - EOR - Zero Page,X
56 - LSR - Zero Page,X
57 - Future Expansion
58 - CLI
59 - EOR - Absolute,Y
5A - Future Expansion
5B - Future Expansion
5C - Future Expansion
5D - EOR - Absolute,X
5E - LSR - Absolute,X
5F - Future Expansion

60 - RTS
61 - ADC - (Indirect,X)
62 - Future Expansion
63 - Future Expansion
64 - Future Expansion
65 - ADC - Zero Page
66 - Future Expansion
67 - Future Expansion
68 - PLA
69 - ADC - Immediate
6A - Future Expansion
6B - Future Expansion
6C - JMP - Indirect
6D - ADC - Absolute
6E - Future Expansion
6F - Future Expansion
70 - BVS
71 - ADC - (Indirect),Y
72 - Future Expansion
73 - Future Expansion
74 - Future Expansion
75 - ADC - Zero Page,X
76 - Future Expansion
77 - Future Expansion
78 - SEI
79 - ADC - Absolute,Y
7A - Future Expansion
7B - Future Expansion
7C - Future Expansion
7D - ADC - Absolute,X
7E - Future Expansion
7F - Future Expansion

80 - Future Expansion
81 - STA - (Indirect,X)
82 - Future Expansion
83 - Future Expansion
84 - STY - Zero Page
85 - STA - Zero Page
86 - STX - Zero Page
87 - Future Expansion
88 - DEY
89 - Future Expansion
8A - TXA
8B - Future Expansion
8C - STY - Absolute
8D - STA - Absolute
8E - STX - Absolute
8F - Future Expansion
90 - BCC
91 - STA - (Indirect),Y
92 - Future Expansion
93 - Future Expansion
94 - STY - Zero Page,X
95 - STA - Zero Page,X
96 - STX - Zero Page,Y
97 - Future Expansion
98 - TYA
99 - STA - Absolute,Y
9A - TXS
9B - Future Expansion
9C - Future Expansion
9D - STA - Absolute,X
9E - Future Expansion
9F - Future Expansion

A0 - LDY - Immediate
A1 - LDA - (Indirect,X)
A2 - LDX - Immediate
A3 - Future Expansion
A4 - LDY - Zero Page
A5 - LDA - Zero Page
A6 - LDX - Zero Page
A7 - Future Expansion
A8 - TAY
A9 - LDA - Immediate
AA - TAX
AB - Future Expansion
AC - LDY - Absolute
AD - LDA - Absolute
AE - LDX - Absolute
AF - Future Expansion
B0 - BCS
B1 - LDA - (Indirect),Y
B2 - Future Expansion
B3 - Future Expansion
B4 - LDY - Zero Page,X
B5 - LDA - Zero Page,X
B6 - LDX - Zero Page,Y
B7 - Future Expansion
B8 - CLV
B9 - LDA - Absolute,Y
BA - TSX
BB - Future Expansion
BC - LDY - Absolute,X
BD - LDA - Absolute,X
BE - LDX - Absolute,Y
BF - Future Expansion

C0 - CPY - Immediate
C1 - CMP - (Indirect,X)
C2 - Future Expansion
C3 - Future Expansion
C4 - CPY - Zero Page
C5 - CMP - Zero Page
C6 - DEC - Zero Page
C7 - Future Expansion
C8 - INY
C9 - CMP - Immediate
CA - DEX
CB - Future Expansion
CC - CPY - Absolute
CD - CMP - Absolute
CE - DEC - Absolute
CF - Future Expansion
D0 - BNE
D1 - CMP - (Indirect),Y
D2 - Future Expansion
D3 - Future Expansion
D4 - Future Expansion
D5 - CMP - Zero Page,X
D6 - DEC - Zero Page,X
D7 - Future Expansion
D8 - CLD
D9 - CMP - Absolute,Y
DA - Future Expansion
DB - Future Expansion
DC - Future Expansion
DD - CMP - Absolute,X
DE - DEC - Absolute,X
DF - Future Expansion

E0 - CPX - Immediate
E1 - SBC - (Indirect,X)
E2 - Future Expansion
E3 - Future Expansion
E4 - CPX - Zero Page
E5 - SBC - Zero Page
E6 - INC - Zero Page
E7 - Future Expansion
E8 - INX
E9 - SBC - Immediate
EA - NOP
EB - Future Expansion
EC - CPX - Absolute
ED - SBC - Absolute
EE - INC - Absolute
EF - Future Expansion
F0 - BEQ
F1 - SBC - (Indirect),Y
F2 - Future Expansion
F3 - Future Expansion
F4 - Future Expansion
F5 - SBC - Zero Page,X
F6 - INC - Zero Page,X
F7 - Future Expansion
F8 - SED
F9 - SBC - Absolute,Y
FA - Future Expansion
FB - Future Expansion
FC - Future Expansion
FD - SBC - Absolute,X
FE - INC - Absolute,X
FF - Future Expansion

The following notation applies to this summary:

A	Accumulator
X, Y	Index Registers
M	Memory
P	Processor Status Register
S	Stack Register
✓	Change
—	No Change
+	Add
∧	Logical AND
-	Subtract
∀	Logical Exclusive Or
↑	Transfer from Stack
↑	Transfer to Stack
→	Transfer to
←	Transfer to
V	Logical OR
PC	Program Counter
PCH	Program Counter High
PCL	Program Counter Low
OPER	OPERAND
#	IMMEDIATE ADDRESSING MODE

Note: At the top of each table is located in parentheses a reference number (Ref: XX) which directs the user to that Section in the MCS6500 Microcomputer Family Programming Manual in which the instruction is defined and discussed.

Warranty

Kits:

Western Data Systems hereby warrants all Data Handler kits that, 1) during the first thirty (30) days of ownership all components supplied by W.D.S. to be free from defects in materials and workmanship, and 2) in the event of component failure or malfunction said components will be repaired or replaced free of charge at the descretion of W.D.S. when returned to the factory postage paid.

Any returned Data Handler kits which have been fully owner assembled and require factory repair due to normal use and service within the first thirty (30) days of ownership, will be repaired to operating order for a nominal shipping and handling charge of $10.00.

Assembled Units:

During the first three (3) months of ownership, any factory assembled units, that we determine, under conditions of normal use and service, fail to perform according to our published specification, will be replaced or repaired, at no charge to you at the W.D.S. factory.

The above warranty applies to the original purchaser only and does not cover damage by use of acid-core solder, incorrect assembly, misuse, fire, floods or acts of God. If W.D.S. or a representative there of determines that your Data Handler system needs repair due to said conditions, a charge greater than the nominal amount may be required.