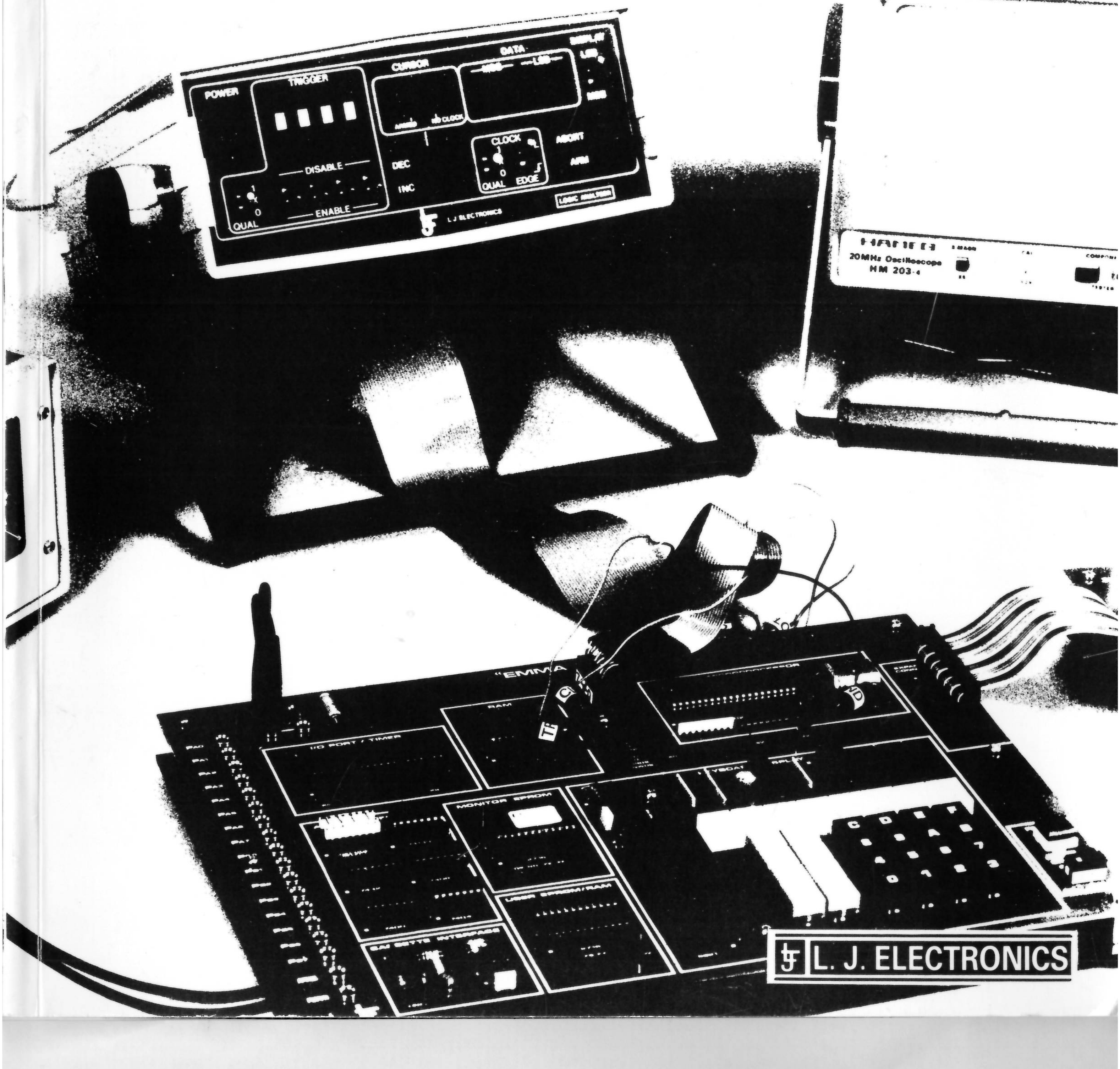
L.J. Technical Systems

EMMA II

Technical Manual



L.J. Electronics

EMMA II

Technical Manual

EMMA II Technical Manual

Contents

Part 1 Hardware

Part 2 Software

Appendices

EMMA II Technical Manual

Introduction

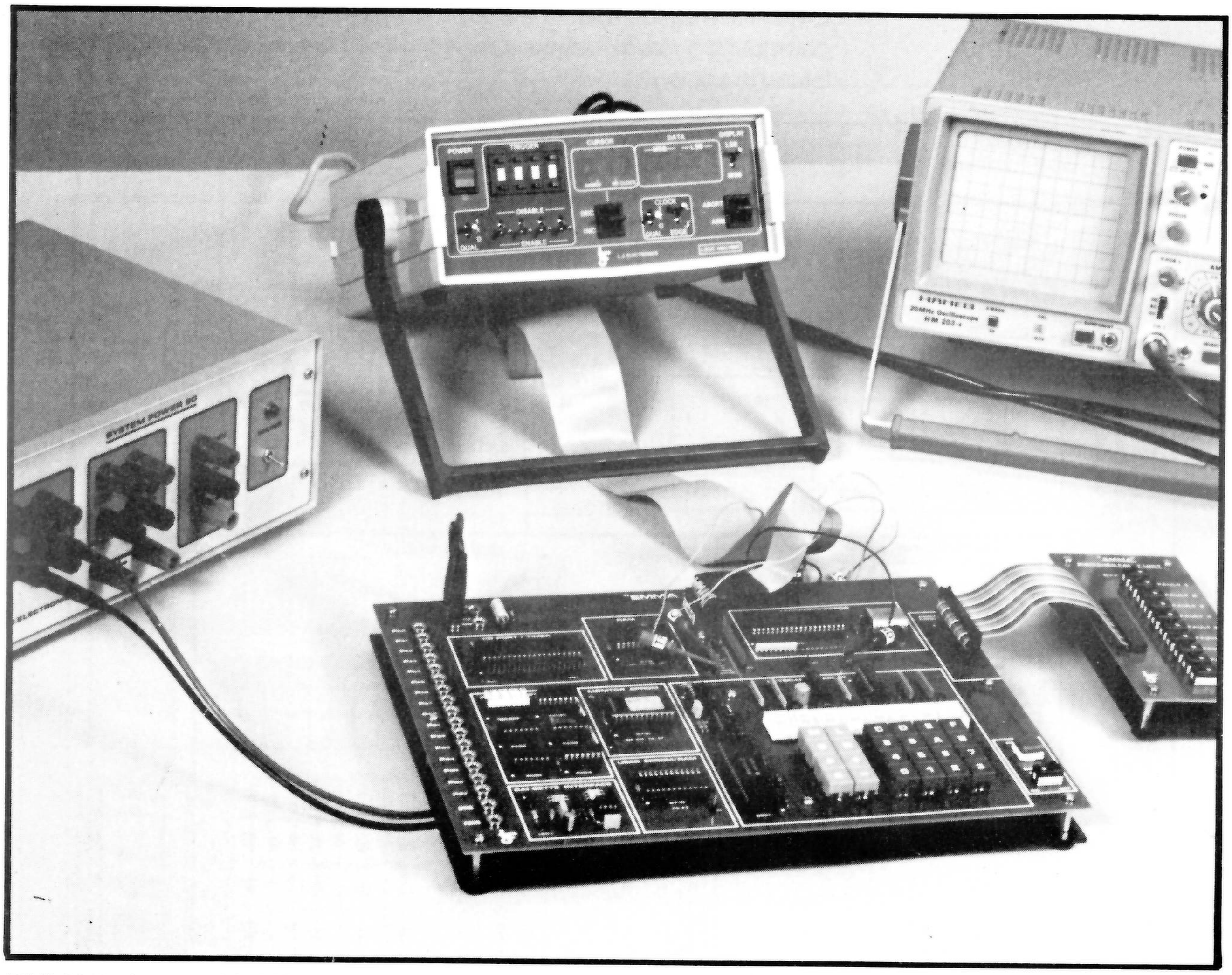
This manual is the technical support for the user manual. It supplies descriptions of chip functions and pin connections associated with EMMA II.

In order to best serve the user, this manual is written in two sections Part 1, and Part 2.
Part 1 is dedicated to hardware description, and
Part 2 is dedicated to software, a study of the 6502 instruction set and utilizing the support chips with respect to applying software.

Included in the appendices are a complete listing of the EMMA II monitor program contained in EPROM.

EMMA II Technical Manual

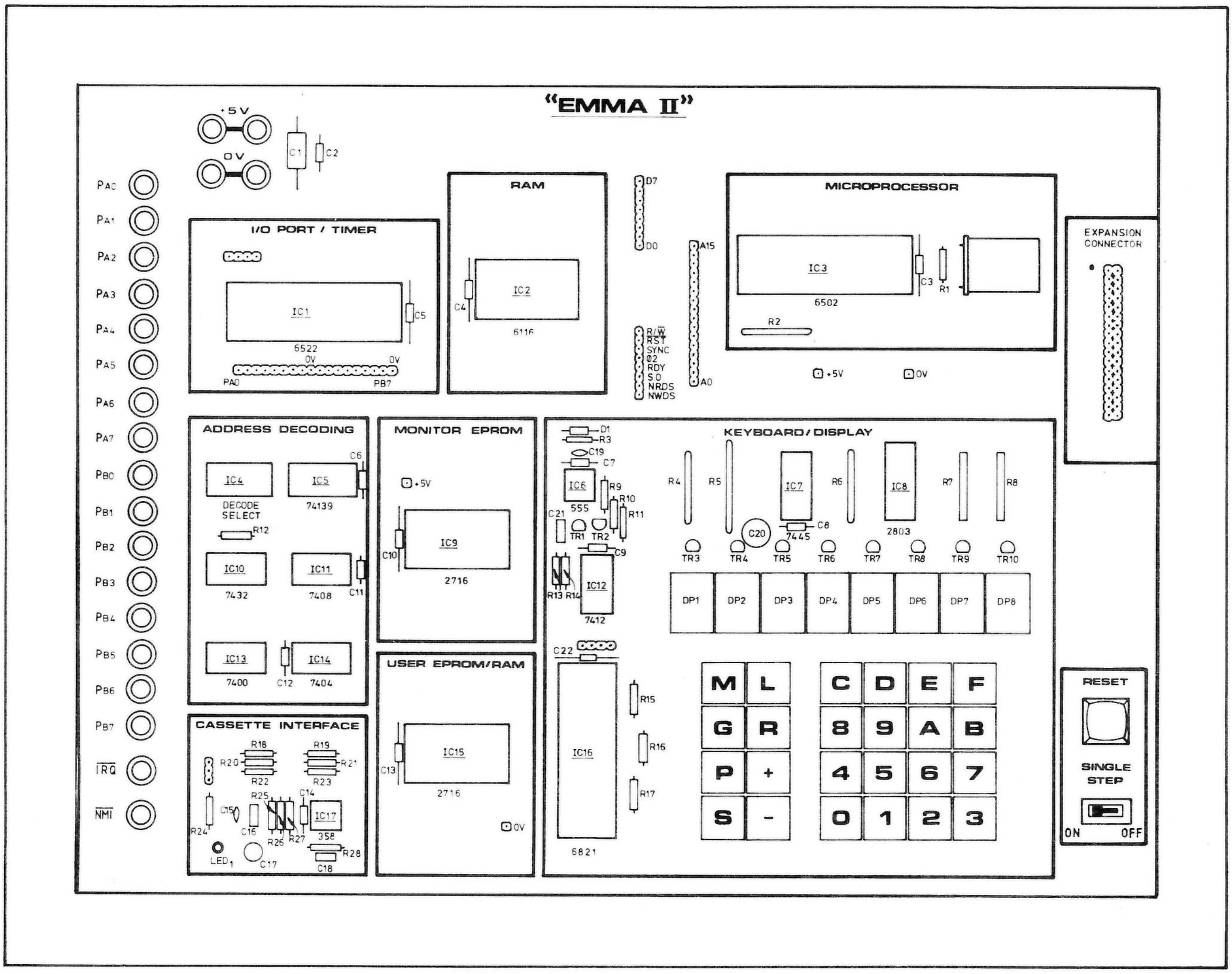
Part 1 EMMA II Hardware



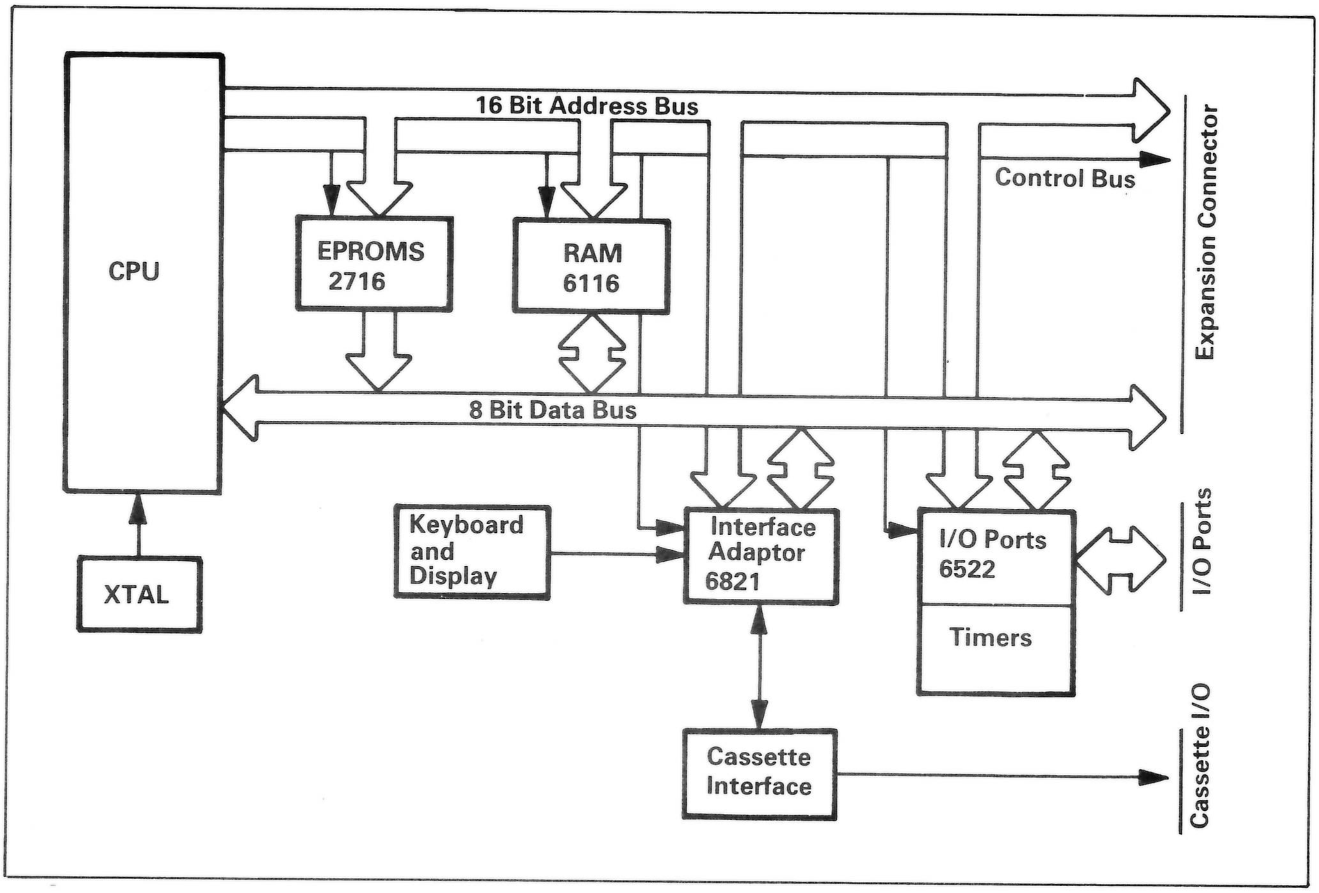
EMMA II shown with MS1A Switched Fault Unit, SA1 Logic Analyzer, PS4 Power Supply and HM203 Oscilloscope

Chapter 1 The EMMA II System

EMMA II is based upon the 6502 microprocessor. To form a computing system various support chips are needed. Layout of the board is shown below.



The figure below shows a block diagram of the EMMA II system. The diagram is in a simple form to display the major areas of the microprocessor system.



The basic microcomputer consists of three main sections:

1. Central processor unit
2. Memory section
3. Input/output interface section

These sections are connected via the bus systems:

i.e.
1. Data bus
2. Address bus
3. Control bus

Looking at each section in turn, a study will be made of its function in the EMMA II system.

- **CPU Section**
  The central processing unit or CPU is the 6502 microprocessor. The 6502 contains an Arithmetic Logic Unit, some Registers and Program Control Counter. The CPU sends out a memory address of the location of the first instruction to be executed. The instruction is read from memory and subsequently executed. After each operation, the Program Control Counter increments to the address of the next memory location, and the cycle continues.

- **Input/Output Interface Section**
  This section of EMMA II consists of:-
  6522 I/O Port Controller
  6821 Keyboard and Cassette Interface.
  These enable communication between the microprocessor and the outside world. The 6522 comprises two 8-bit I/O ports and two programmable timers.

- **Data Bus**
  The Data Bus is an 8-bit bi-directional path between the microprocessor and the memory and interface circuits. The data bus is tri-state and is capable of driving one standard TTL load and 130pF.

- **Address Bus**
  The CPU sends out on the address bus the address of a memory location which is to be read from or written into. The EMMA II system has a 16-bit address bus capable of addressing 64K locations.

- **Control Bus**
  The control bus carries signals such as READ/WRITE (R/$\overline{W}$),Ø2, $\overline{NMI}$, $\overline{IRQ}$. These signals are the controlling influence on the microcomputer. For example, to read a memeory location, the CPU first sends out the address on the address bus, then a memory read signal on the control line; the memory outputs the data from the address location to the data bus.

● Memory Section

The memory section in EMMA II comprises of RAM and EPROM. The memory allocations for EMMA II are shown below:

0000 Reserved EMMA/VISA Monitor; used in conjunction with
001F devices such as the Eprom Programmer, Matrix Printer,etc.

0020 Available as user RAM
00FF

0100 Designated by the 6502 as the system stack
01FF

0200 Available as user RAM
03FF

0900 Decoded for use by the EMMA II input/output ports.
0AFF

0C00 User RAM
0FFF

0EFC-0EFF Interrupt vectors.
0E80-0EFF In use when VISA connected

D800 EMMA II Monitor; cassette routines, useful
DDFF sub-routines

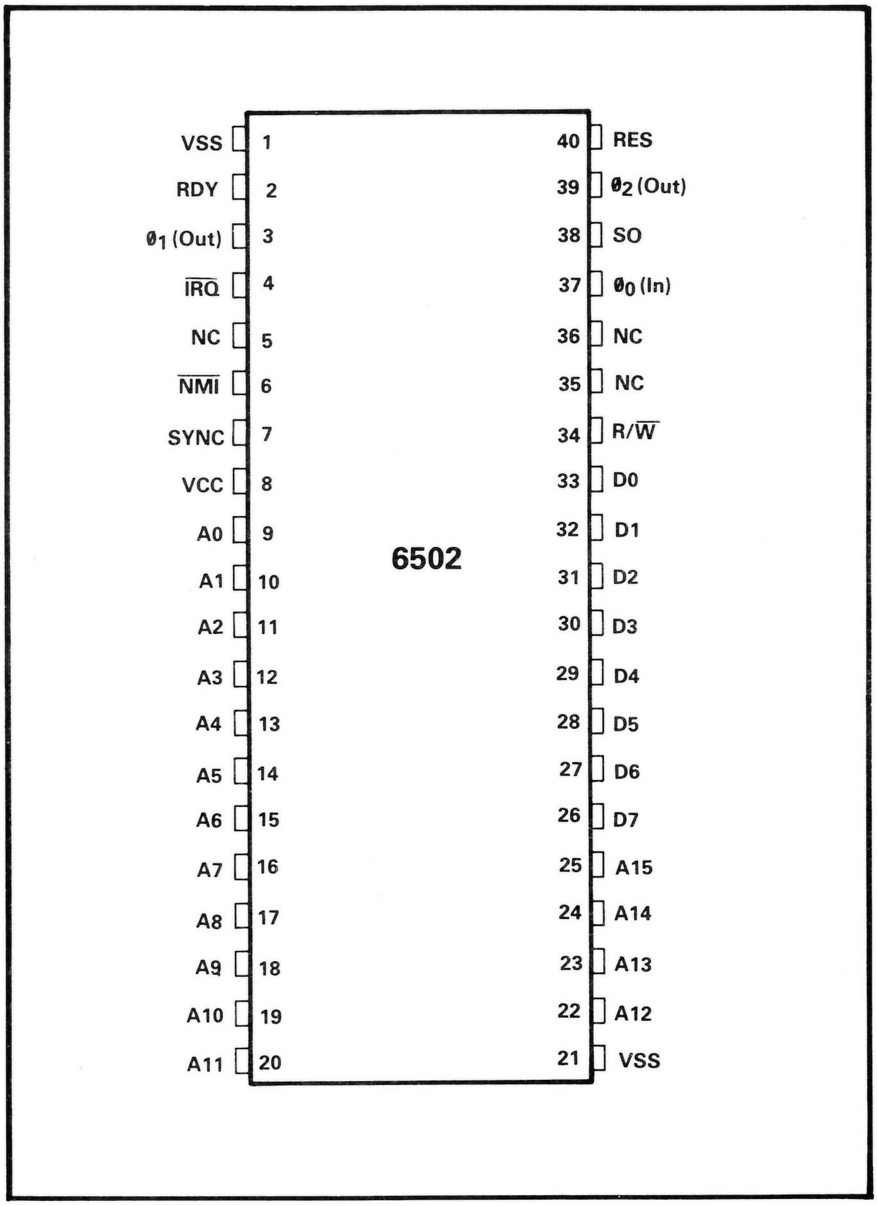
F000 Available for user EPROM expansion
F7FF

FE00 EMMA II Monitor program
FFFF

The RAM and EPROM are explained in more detail further in this manual.

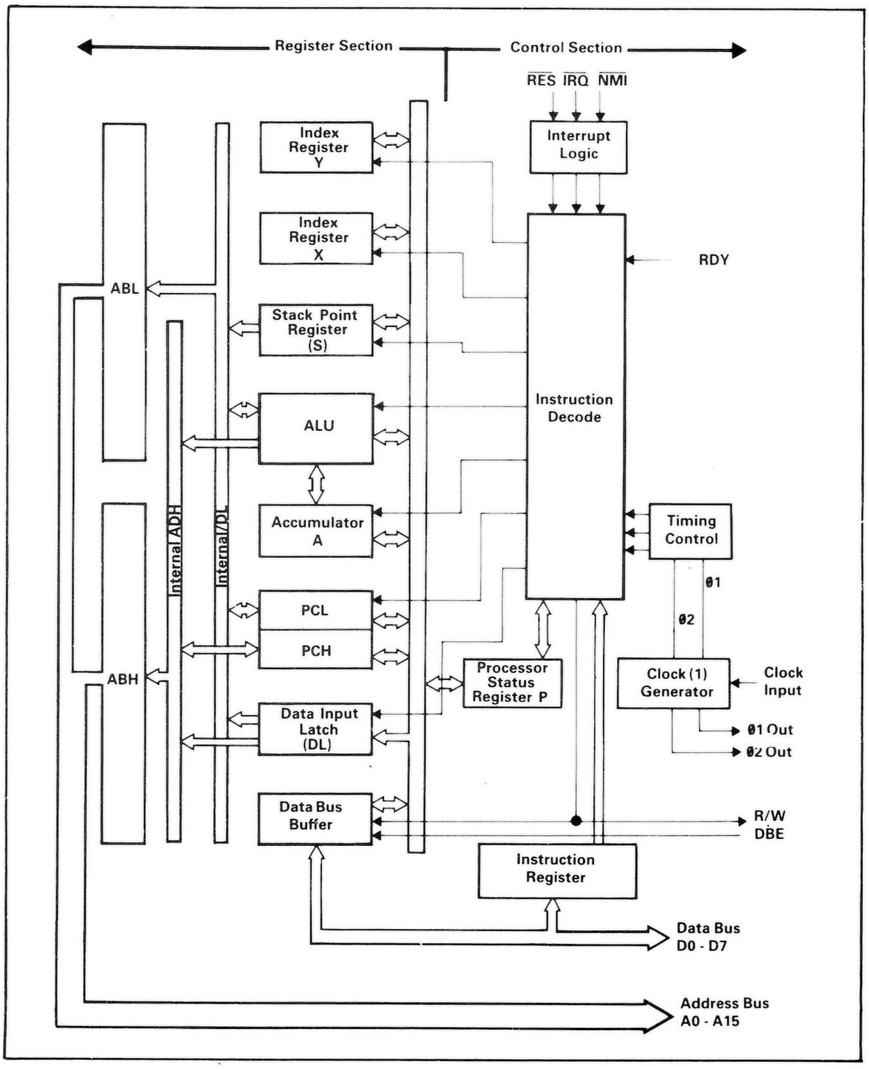
Chapter 2 6502 Microprocessor

As can be seen on the layout diagram on page 3 'IC3' is the 6502 microprocessor. This is a 40-pin I.C., requiring an external crystal to maintain stable operating speed. The EMMA II uses a 1MHz crystal, giving a basic microprocessor cycle time of 1μs.



Internal Architecture of the 6502

The diagram of the internal architecture will be used to discuss the operation of the various pins available outside the chip and in particular those specifically used in the EMMA II Microcomputer System.



With reference to the internal architecture diagram it can be seen where the external pins connect to the structure of the microprocessor. We will examine the external pins in turn.

- **Address Bus (A_0-A_{15})**
  The 16-bit address bus is used to transfer the address generated by the processor to the address inputs of all memory and peripheral interface devices. The address is always provided by the 6502, thus the address bus is unidirectional. The bus lines are TTL compatible, capable of driving one standard TTL load and 130 pF.

- **Clock ($\emptyset_0$,$\emptyset_1$,$\emptyset_2$):**
  The 6502 clock circuits can be driven from a single TTL level square wave, or with the internal oscillator. As explained the **EMMA II** system uses a 1 MHz crystal to control the internal oscillator, giving a stable 1μs cycle time.

- **Data Bus (DB_0-DB_7):**
  The 8-bit data bus is bi-directional, providing a two-way transfer path between the microprocessor and the memory and interface circuits. The data bus is tri-state and is capable of driving one standard TTL load and 130 pF.

- **Ready (RDY):**
  This input signal is provided to allow the user to halt the 6502 on all cycles except write cycles. This feature is of use when low speed PROMS are used, or when fast direct memory access (DMA) transfers are to be performed. The ready signal is disabled (held high), on the EMMA II board.

- **Interrupt Request ($\overline{IRQ}$)**
  When this input is in the low state (logic '0'), the microprocessor is requested to begin an interrupt sequence. The microprocessor will not acknowledge this request until it has completed the current instruction being executed. Then, provided the interrupt mask bit (interrupt disable, within the status register), is not set, the microprocessor will begin an interrupt sequence.

  The interrupt sequence is:
  (i) store Program Counter high byte (PCH) on the stack.
  (ii) store Program Counter low byte (PCL) on the stack.
  (iii) store status register contents on the stack.
  (iv) load PCL from address FFFE.
  (v) load PCH from address FFFF.
  (vi) normal program execution sequences now continue from the memory vector held at FFFF and FFFE.

  On the EMMA II board, the IRQ input is returned to +, through a 4.7kΩ resistor to facilitate the wired-OR operation.

- **Non-Maskable Interrupt ($\overline{\text{NMI}}$)**
  A negative going transition on this input requests that a non-maskable interrupt sequence is begun by the microprocessor. This is not conditional and will always be actioned following completion of the current instruction. The sequence of steps for a $\overline{\text{NMI}}$ is the same as for $\overline{\text{IRQ}}$, except that the vector address will be loaded to PCL and PCH from locations FFFA and FFFB respectively.

  The 6502 has an internal latch which is set by a '1' to '0' transition on the $\overline{\text{NMI}}$ input. Thus a single, short duration, negative going pulse may be used to provide an $\overline{\text{NMI}}$, unlike $\overline{\text{IRQ}}$, which has no latch and requires $\overline{\text{IRQ}}$ to remain low until the interrupt is actioned. $\overline{\text{NMI}}$ will take priority over $\overline{\text{IRQ}}$.

  Wired - OR inputs to the $\overline{\text{NMI}}$ are facilitated by a 4.7 kΩ resistor returned to +5V.

- **Set Overflow Flag (S.O.):**
  A negative going edge on this input sets the overflow flag within the status register. This facility is not required for any of the current 6500 family circuits and the S.O. input is returned to +5V on the EMMA II board.

- **SYNC:**
  This output line identifies those cycles in which the microprocessor is doing a fetch instruction operation. The SYNC line goes high during$\emptyset_1$ of a fetch operation and stays high for remainder of that cycle

  The SYNC signal is used on the EMMA II board for the generation of an interrupt signal required for single step operation.

- **RESET ($\overline{\text{RES}}$):**
  This input is used to reset or start the microprocessor from a power down condition. During the time that this line is held low, writing to or from the microprocessor is inhibited. When a positive edge occurs on the input the microprocessor will immediately begin the reset sequence.

  After a system initialization time of six clock cycles, the mask interrupt flag will be set and the microprocessor will load the program counter from the memory locations FFFC and FFD. This is the start location for program control (FEE0 in in the EMMA II monitor).

  On the EMMA II board, $\overline{\text{RES}}$ is held high through a 4.7kΩ resistor; it is pulled low when the push-button reset switch is depressed.

- **Read/Write (R/$\overline{W}$):**
  This output signal is used to control the direction of data transfer between the processor and other memory circuits on the data bus. R/$\overline{W}$ high signifies data transfer into the 6502, R/$\overline{W}$ low signifies data transfer out of the 6502.

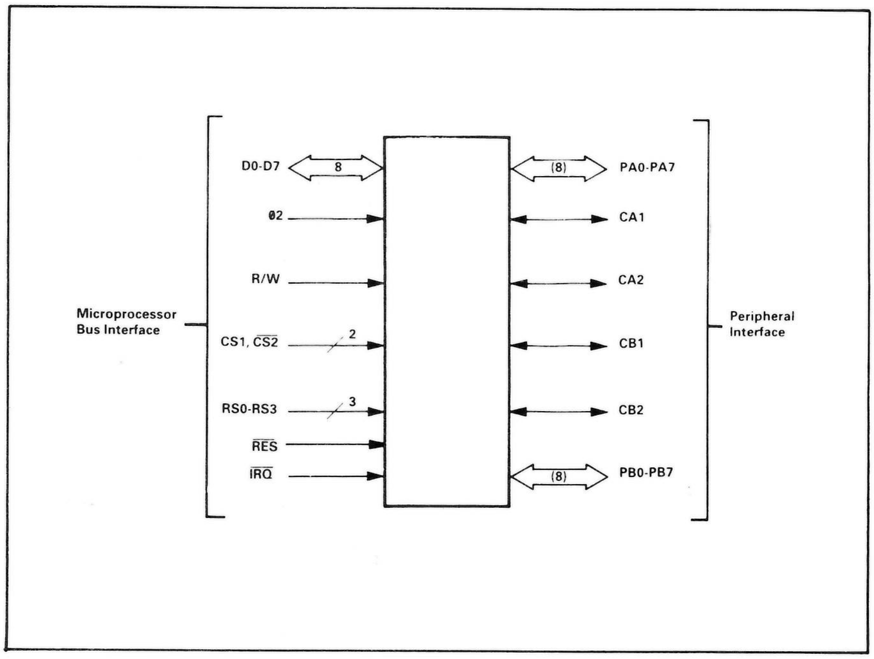
Supply Voltages:

The EMMA II requires a 5V regulated supply, able to provide 700mA approximately. The 6502 has a maximum V_{CC} rating of +7V.

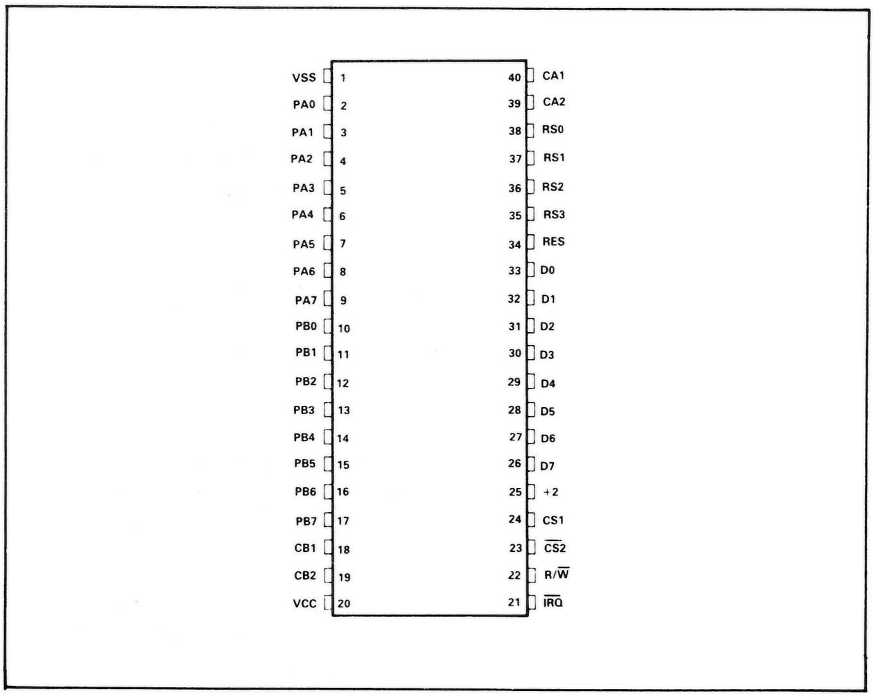
It is strongly recommended that a fixed 5V d.c. supply should be used for the EMMA II. Some variable supplies generate a short duration positive going spike when switched on, which may damage circuits in the 6500 family.

Chapter 3 6522, Versatile Interface Adaptor (VIA)

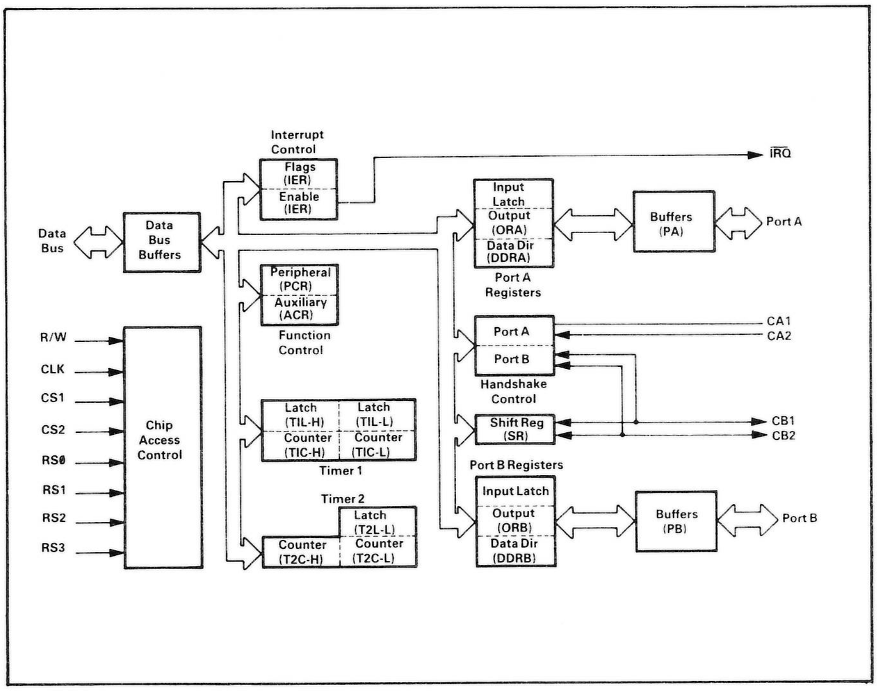
Again referring to the layout of the EMMA II 'IC1' the 6522 can be seen as the I/O Port Controller/Timer. A simplified configuration is shown.



6522 pin configuration is as shown.



The internal structure of the 6522 is shown below:



The 6522 provides

(a) two 8-bit programmable input/output ports.
(b) two 16-bit programmable timer/counters.
(c) one 8-bit serial transfer register.

The pin functions are as follows:

- **$\overline{\text{RES}}$ (Reset)**
  The reset input clears all internal registers to logic 0 (except T1 and T2 latches and counters and the Shift Register). This places all peripheral interface lines to the input state, disables the timers, shift register, etc, and disables interrupts from the chip.

- **(Input Clock):**
  The input clock is the EMMA system Ø2 clock and is used to control all data transfers between the system processor and the 6522.

- **R/$\overline{\text{W}}$ (Read/$\overline{\text{Write}}$)**
  The direction of the data transfers between the 6522 and the system processor is controlled by the R/$\overline{\text{W}}$ line. If R/$\overline{\text{W}}$ is low, data will be transferred out of the processor into the selected 6522 register (write operation). If R/$\overline{\text{W}}$ is high and the chip is selected, data will be transferred out of the 6522 (read operation).

- **DB0-DB7 (Data Bus)**
  The eight bi-directional data bus lines are used to transfer data between the 6522 and the system processor. During read cycles, the contents of the selected 6522 register are placed on the data bus lines and transferred into the processor. During write cycles, these lines are high-impedance inputs and data is transferred from the processor into the selected register. When the 6522 is not being accessed the data bus lines are high-impedance.

- **CS1, CS2 (Chip Selects)**
  The two chip select inputs are normally connected to processor address lines either directly or through decoding. The selected 6522 register will be accessed when CS1 is high and CS2 is low.

  On the EMMA II board, the address decoding provides CS2 low when memory pages 08 to 0B are selected. CS1 is connected to address line A_9, thus the 6522 can be accessed on page 09 or 0B.

- **RS0-RS3 (Register Selects)**
  The four Register Select inputs permit the system processor to select one of the 16 internal registers of the 6522.

  The EMMA II board has the register lines RS0-RS3 connected to address lines A_0-A_3 respectively. Thus, the memory locations for the 6522 registers are as given in the table overleaf.

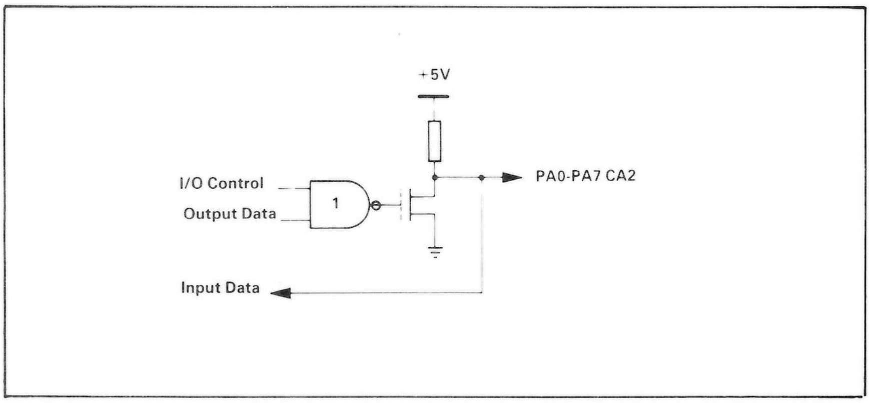
Memory Locations for the 6522 Registers

Address	Register Desig:	Description	
		Write	Read
09x0	ORB/IRB	Output Register "B"	Input Register "B"
09x1	ORA/IRA	Output Register "A"	Input Register "A"
09x2	DDRB	Data Direction Register "B"	
09x3	DDRA	Data Direction Register "A"	
00x4	T1C-L	T1 Low-Order Ltches	T1 Low-Order Counter
09x5	T1C-H	T1 High-Order Counter	
09x6	T1L-L	T1 Low-Order Latches	
09x7	T1L-H	T1 High-Order Latches	
09x8	T2C-L	T2 Low-order latches	T2 Low-order counter
09x9	T2C-H	T2 High-Order Counter	
09xA	SR	Shift Register	
09xB	ACR	Auxiliary Control Register	
09xC	PCR	Peripheral Control Register	
09xD	IFR	Interrupt Flag Register	
09xE	IER	Interrupt Enable Register	
09xF	ORA/IRA	Same as 09x0 Except no "Handshake"	

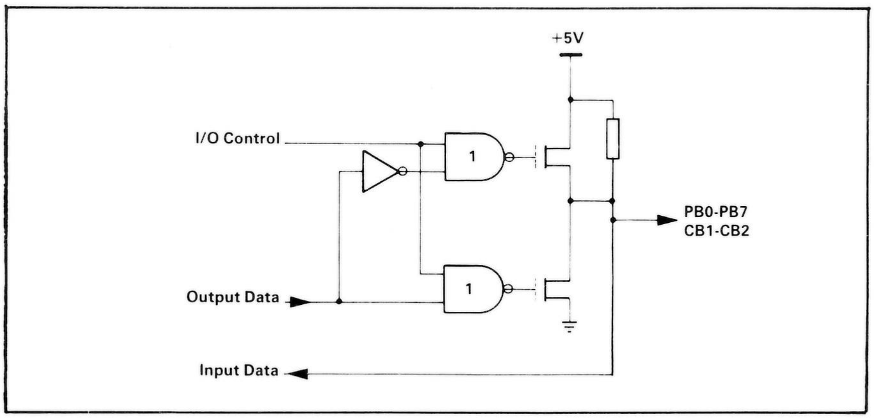
Note: The chip select connections mean that these register locations are duplicated on page 0B. The fact that the register select lines are activated by address lines A_0-A_3, regardless of the state of address line A_4-A_7, means that the hexadecimal code used for X in the previous listing is unimportant.

- **$\overline{IRQ}$ (Interrupt Request)**
  The Interrupt Request output goes low whenever an internal interrupt flag is set and the corresponding interrupt enable bit is a logic 1. This output is "open-drain" to allow the interrupt request signal to be "wire-or'ed" with other equivalent signals in the system.

- **PA0-PA7 (Peripheral A Port)**
  The peripheral A port consists of 8 lines which can be individually programmed to act as inputs or outputs under control of a Data Direction Register. The polarity of output pins is controlled by an Output Register and input data may be latched into an internal register under control of the CA1 line. All of these modes of operation are controlled by the system processor through the internal control registers. These lines represent one standard TTL load in the input mode and will drive one standard TTL load in the output mode. This is illustrated in the following diagram.



- **PB0-PB7 (Peripheral B Port)**
  The Peripheral B port consists of eight bi-directional lines which are controlled by an output register and a data direction register in much the same manner as the PA port. In addition, the polarity of the PB7 output signal can be controlled by one of the interval timers while the second timer can be programmed to count pulses on the PB6 pin. Peripheral B lines represent one standard TTL load in the input mode and will drive one standard TTL load in the output mode. In addition, they are capable of sourcing 1.0mA at 1.5VDC in the output mode to allow the outputs to directly drive Darlington Transistor Circuts. The circuit diagram below displays this.



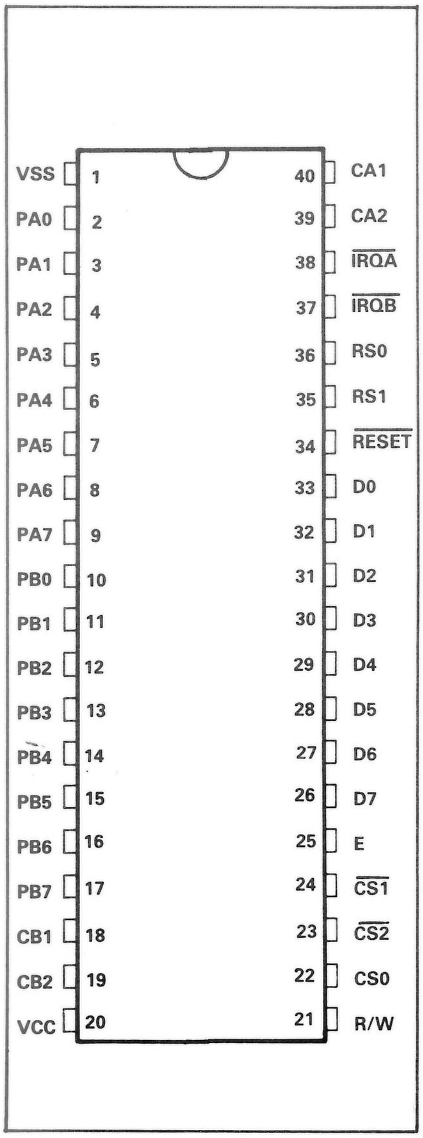
- **CA1, CA2 (Peripheral A Control Lines)**
  The two peripheral A control lines act as interrupt inputs or as handshake outputs. Each line controls an internal interrupt flag with a corresponding interrupt enable bit. In addition, CA1 controls the latching of data on Peripheral A port input lines. CA1 is a high impedance input only while CA2 represents one standard TTL load in the input mode. CA2 will drive one standard TTL load in the output mode.

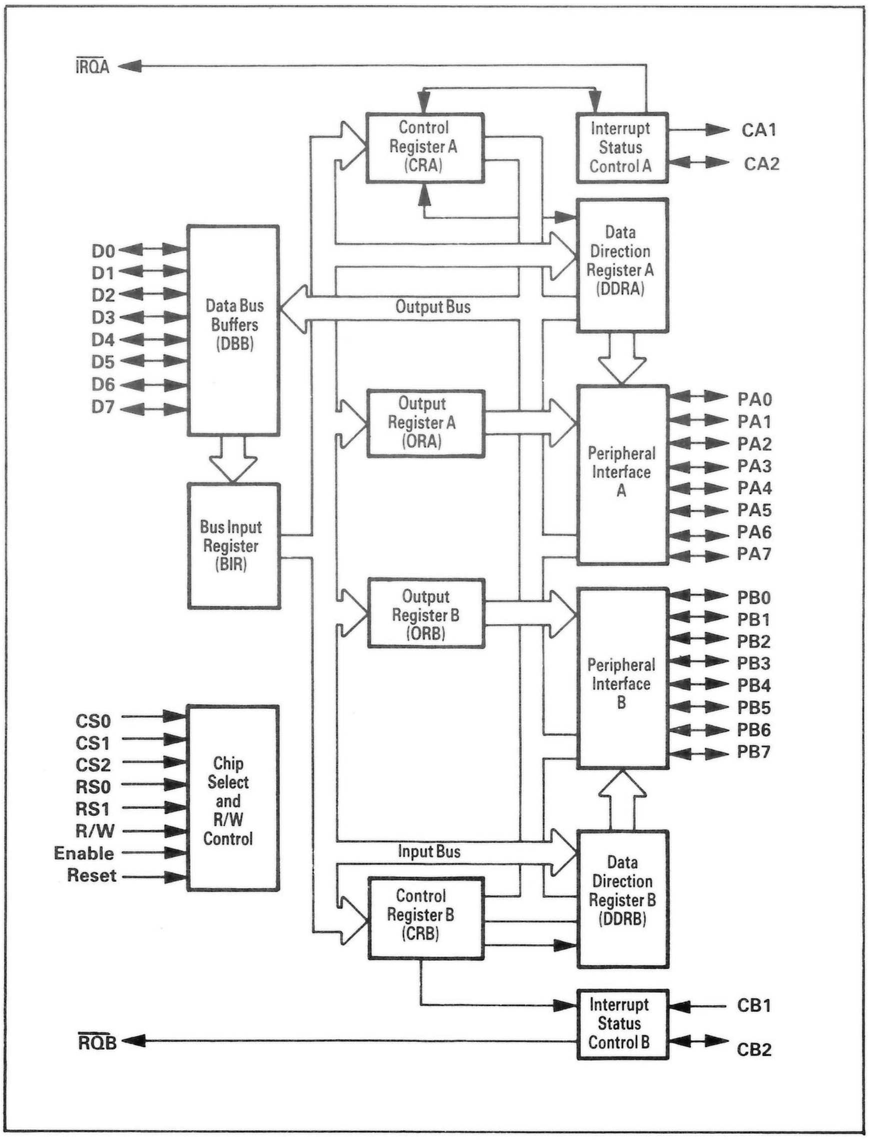
- **CB1, CB2 (Peripheral B Control Lines)**
  The Peripheral B control lines act as interrupt inputs or as handshake outputs. As wth CA1 and CA2, each line controls an interrupt flag with a corresponding interrupt enable bit. In addition, these lines act as a serial port under control of Shift Register. These lines represent one standard TTL load in the input mode and will drive one standard TTL load in the output mode. Unlike PB0-PB7, CB1 and CB2 cannot drive Darlington Transistor Circuits.

Chapter 4 6821 Peripheral Interface Adaptor (PIA)

With reference to the board layout diagram, IC16 (6821) is on the keyboard display section.

The 6821 provides a means of interfacing the keyboard display unit and the cassette interface with the microprocessor.





Pin connections are as shown. Block diagram of the 6821 is as shown.

Pin Functions

- **DB0-DB7 Data Bus**
  Are eight bi-directional data bus lines used to transfer data between 6821 and the system processor. The 6821 data bus buffers are tri-state, giving a high impedance loading to the data bus unless a chip transfer is selected.

- **Enable (E):**
  This is the only timing pulse supplied to the PIA. Timing of all other signals is referenced to the leading and trailing edges of the E pulse.

- **Read/Write (R/W):**
  A low state on the PIA Read/Write line enable the input buffers and data is transferred from the microprocessor to the PIA on the 'E' signal if the device has been selected. A high on the R/W line sets up the PIA for transfer of data to the bus. The PIA output buffers are enabled when the proper address and the enable pulse 'E' are present.

- **Reset:**
  The active low RESET used to reset all register bits in the PIA to a (LOW) zero.

- **Chip Select (CS0, CS1 and CS2):**
  These three input signals are used to select the PIA. CS0 and CS1 must be high and CS2 must be low for selection of the device. Data transfers are then performed under the control of the enable and read/write signals. The chip select lines must be stable for the duration of the Enable pulse. The device is deselected when any of the chip selects are in the inactive state.

- **Register Selects (RS0 and RS1):**
  The two register select lines are used to select the various registers inside the PIA. These two lines are used in conjunction with internal Control Registers to select a particular register that is to be written or read.

  The register and chip select lines should be stable for the duration of the Enable pulse while in the read or write cycle.

- **Interrupt Input (CA1 AND CB1)**
  These are peripheral input lines, CA1 and CB1 are input only lines that set the interrupt flags of the control registers. The active transition for these signals is also programmed by the two control registers.

- **Peripheral Control (CA2)**
  The peripheral control line CA2 can be programmed to act as an interrupt input or as a peripheral control output. As an output, this line is T.T.L. compatible. The function of this signal line is programmed with Control Register A. In **EMMA II** the CA2 line gives indication of data flow via an LED, as a function of the Cassette Interface routine.

- **Peripheral Control (CB2)**
  Peripheral control line CB2 may also be programmed to act as an interrupt input or peripheral control output. As an input this line has high input impedance and T.T.L. compatibility. This line is programmed by Control Register B.

- **Interrupt Request ($\overline{IRQA}$ and $\overline{IRQB}$):**
  The active low interrupt request lines ($\overline{IRQA}$ and $\overline{IRQB}$) act to interrupt the MPU either directly or through interrupt priority circuitry.

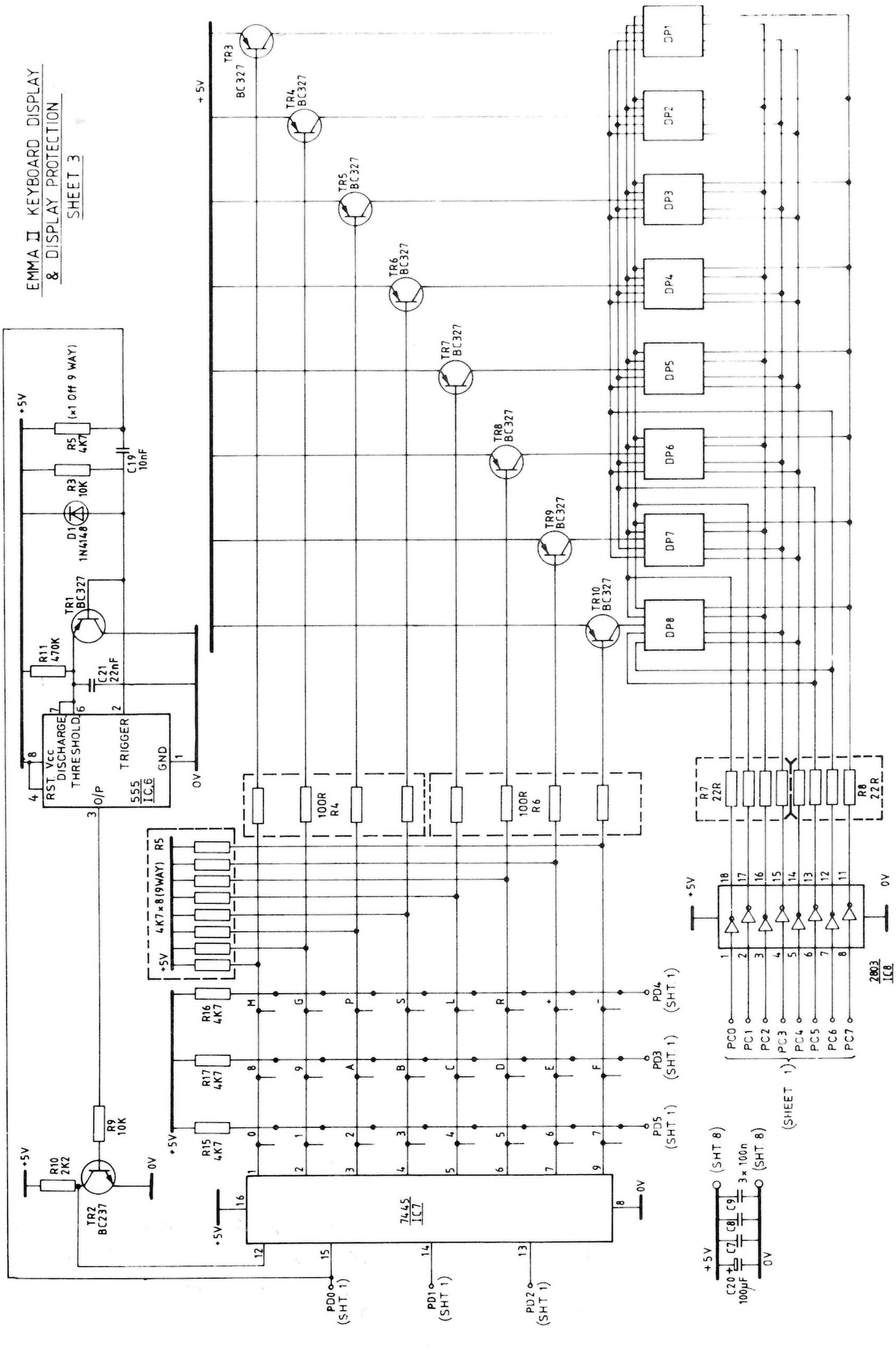
  Each interrupt Request line has two internal interrupt flag bits that can cause the interrupt Request line to go low. Each flag bit is associated with a particular peripheral interrupt line. Also, four interrupt enable bits are provided in the PIA which may be used to inhibit a particular interrupt from a peripheral device. In the EMMA II diagram it can be seen that $\overline{IRQA}$ and $\overline{IRQB}$ are tied together and form part of the $\overline{IRQ}$ connectors throughout the board.

- **Port A (PA0-PA7):**
  This is used by the monitor program for the seven segment display codes (OUTPUTS).

- **Port B (PB0-PB7)**
  This port is mainly used for strobing the keyboard and display and also used as input and output from the cassette.

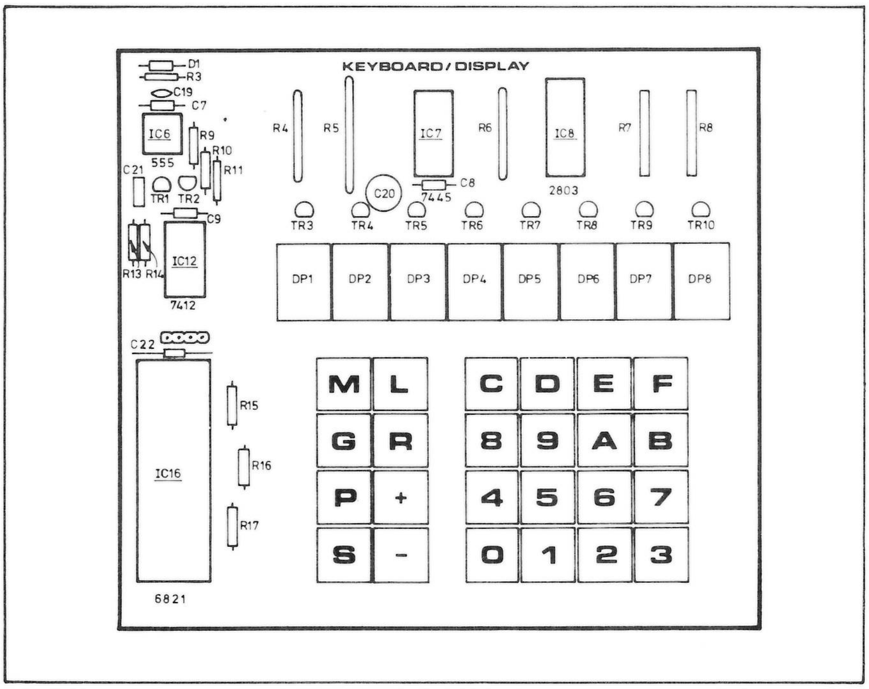
PB0-PB2	Keyboard Strobe
PB3-PB5	Keyboard Column
PB6	Output to Cassette
PB7	Input from Cassette

Since, on the EMMA II board, the 6821 performs the function of display/keypad interface, and is not likely to perform any other task for most users. Functional detail and characteristics are not included in this section of the manual.


EMMA II KEYBOARD DISPLAY
& DISPLAY PROTECTION
SHEET 3
+5V
R5 4K7 (x1 Off 9 WAY)
C19 10nF
R3 10K
D1 1N4148
TR1 BC327
R11 470K
C21 22nF
RST
Vcc
DISCHARGE
THRESHOLD
TRIGGER
GND
O/P
555 IC6
0V
TR2 BC237
R10 2K2
R9 10K
4K7x8(9WAY) R5
R15 4K7
R17 4K7
R16 4K7
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
M
G
P
S
L
R
+
-
PD5 (SHT 1)
PD3 (SHT 1)
PD4 (SHT 1)
7445 IC7
PD0 (SHT 1)
PD1 (SHT 1)
PD2 (SHT 1)
100R R4
100R R6
TR3 BC327
TR4 BC327
TR5 BC327
TR6 BC327
TR7 BC327
TR8 BC327
TR9 BC327
TR10 BC327
DP1
DP2
DP3
DP4
DP5
DP6
DP7
DP8
R7 22R
R8 22R
2803 IC8
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
(SHEET 1)
C20 100µF
C7
C8
C9
3x100n
(SHT 8)

Chapter 5 Keyboard/Display

This section is a supplement to the operations of the 6821 as the interface between the Keyboard/Display and the microprocessor.



A circuit diagram of the keyboard and display is shown opposite on page 21.

The EMMA II Keyboard/Display section comprises seven-segment display and 24 key switches. 'IC7' is a binary decoder; its decoded output is used for key column and display scanning simultaneously.

The method of key detection is as follows:

- The monitor program sends a repeating binary count of 111 to 000 on PB0, PB1, PB2.
- At each step of the count PB5, PB3, PB4 are accessed to see if a key has been pressed. If a key has been presed a '0' will appear in the appropriate column. With no key pressed, PB5, PB3, PB4 inputs will be '1'.
- The monitor program is able to deduce which key is pressed by assessing the binary count and column on which an '0' appears.
- The monitor program will not read a hex key until a control key has been pressed.

An amount of key debouncing is achieved by counting 111 000 twice before acting upon a key function.

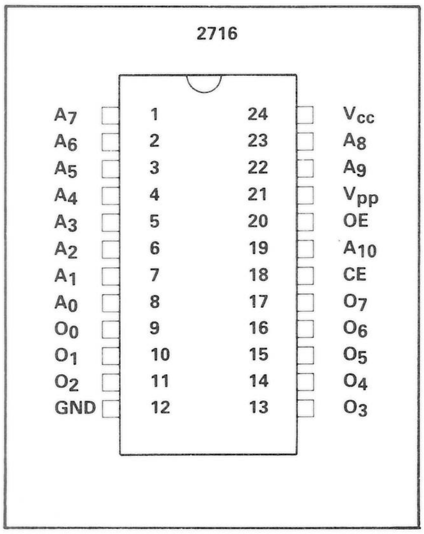
The display uses 8 HDSP-5501 seven segment LED displays (Common Anode). These are strobed continuously by the monitor feeding the 7445 decoder a binary count as explained; the decoder output of the 7445 in turn switches the anode of the displays on to the 5V rail. The seven segment codes for each display are provided via port 'A' of the 6821, this signal is inverted by IC8. Current limitation is provided by resistor packages R7 and R8.

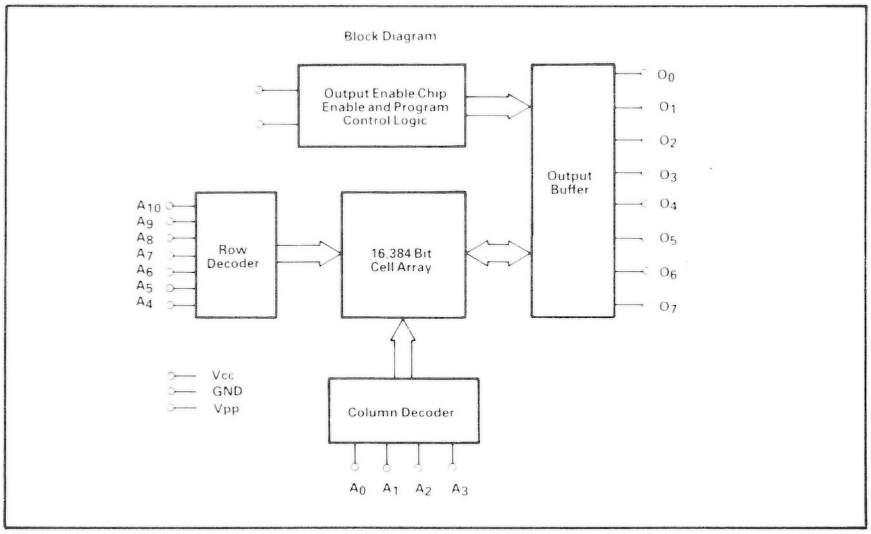
'IC6' is a retriggerable monopulse generator used to detect keyboard scanning via 'PB0'; hence the display is turned 'OFF' when not being strobed. This is a safety feature, to ensure that the displays are not overheated.

Chapter 6 2716 Erasable Programmable Read Only Memory (EPROM)

There are two EPROM sockets on the EMMA II. One is used for the monitor program; the other is available to the user. The 2716 is a 16,384-bit electrically programmable,ultra-violet erasable read-only memory. The 2716 is organized as 2048 eight-bit words.

The 2716 requires only a single 5V power supply with a current required of approximately 100mA. Pin connections for the 2716 are as shown:





The 2048 locations are accessed by an address placed on the eleven lines, A_0-A_{10}. Data from the accessed location is output on the eight data lines, D_0-D_7, if the two control signals 0E and CE are low. Both 0E and CE are high during programming. The decoding on the EMMA II places the monitor EPROM D8 - FF and the user EPROM F0 - F7.

EPROM Erasure

Erasure of the 2716 begins to occur when exposed to light with wave lengths shorter than 4,000 angstroms. Certain types of light contain wave lengths in this range. Although exposure to sunlight or fluorescent light for a relatively long time is required to cause erasure, it is advised to avoid exposure to light for extended periods.

If operated while exposed to ambient light, the minute photoelectric currents generated may cause the EPROM to malfunction. Covering the EPROM 'window' with some opaque material is strongly advised.

Deliberate erasure of the 2716 is accomplished by exposing the device to ultra violet light with a wavelength of 2,537 angstroms. A 12 mW/cm^2 u-v lamp placed one inch from the 2716 will require approximately 20 minutes for complete erasure. Over-erasure is not thought by EPROM manufacturers to be a problem, but it is advised that erasure times greater than 10 times the necessary period should be avoided. Under-erasure will result in programming error. When erased,all bits are in the '1' state.

EPROM Programming

When fully erased (or blank) all the bits of a 2716 will be at logic '1'. Therefore programming of a single location (8-bits) is achieved by changing the required bits to logic '0'

The procedure for programming a 2716 EPROM is given below:

- Set up the following voltages:V_{PP} =+25V
  V_{CC} =+5V
  CE/PGM '0'
- Set up the address location to be programmed om A0 to A10.
- Set up the data to be programmed on D0 to D7.
- Pulse the CEE/PGM pin high for a period of 45 to 55 ms

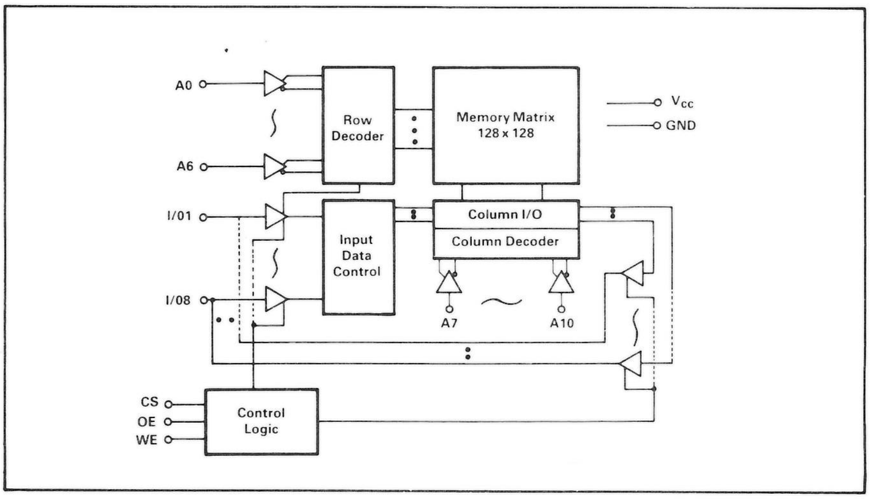
Notes:
1. No pin should be left open circuit
2. The CE/PGM pin must not remain high for longer than 55ms
3. It is possible to program several 2716 EPROMS in parallel

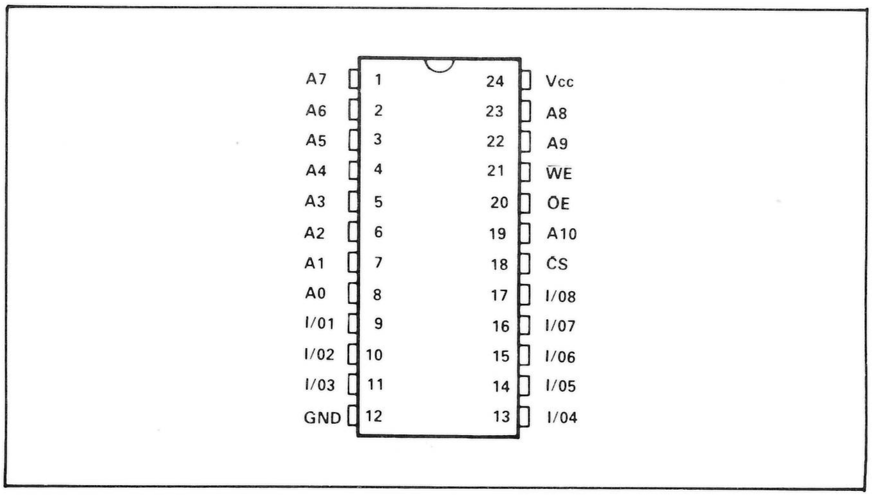
User EPROM/RAM

As indicated on EMMA II, the user EPROM socket can be used for inserting a 6116 RAM. An alteration to the decode select header is needed. The link from pin 7-10 needs to be placed between 8-9. This enables the NWDS to be used in conjunction with the RAM R/$\overline{W}$ input.

Chapter 7 6116 Random Access Memory (RAM)

The RAM on the EMMA II is provided by a 16384 - bit static RAM type '6116'. This is organized into,2048 eight bit words;the package is fabricated by CMOS Technology, and operates from a 5 volt supply.





The 6116 is compatible with the 5517 RAM. A feature of the 6116 is output enable and chip enable input; that is OE for fast memory access and CS for a minimum standby current mode.

On EMMA II the OE pin is tied to the NRDS line, and CS is chip select from the decoding.

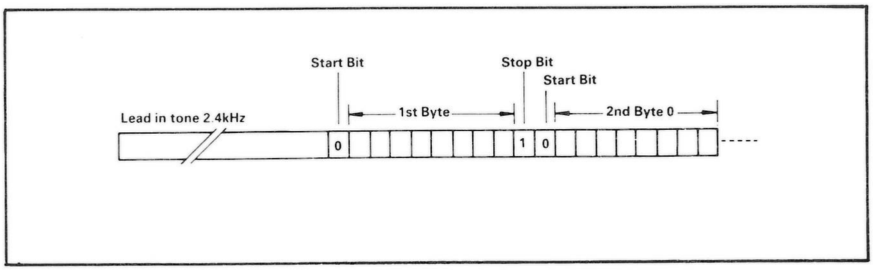
As explained earlier the USER EPROM/RAM socket can be used as a RAM location. Alterations to the Decode Select Header are needed.

Chapter 8 Cassette Interface

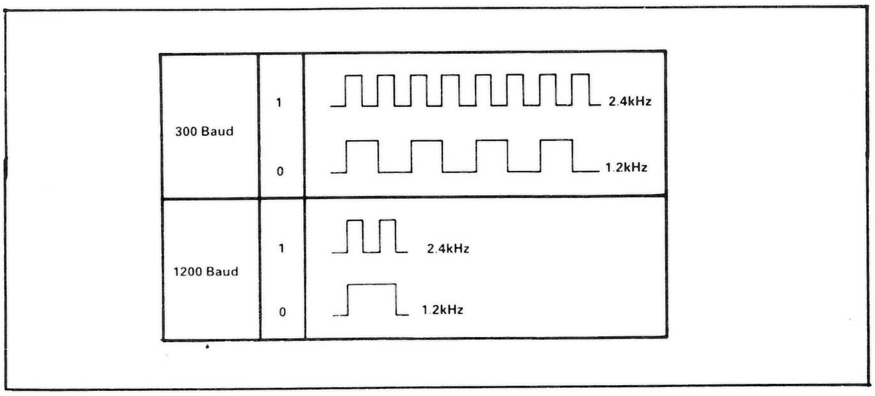
The EMMA II uses a computer users tape standard (CUTS) interface for cassette tape recording of programs.

The monitor program in EMMA II controls the routine for saving data on cassette. This routine offers the user two BAUD rates, 300 and 1200 BAUD, (BAUD means bits/sec transmitted). Before sending data to the cassette a 2.4kHz tone is generated and sent preceeding data. This gives a lead in tone of approximately 4 seconds for finding the start of programs on tape, then data is sent.

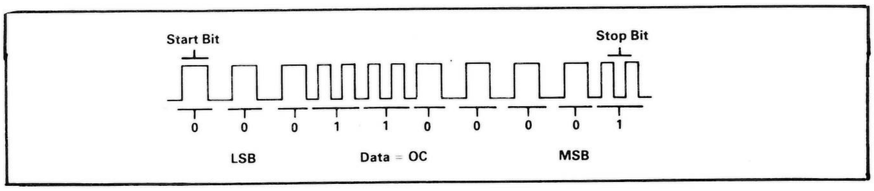
Format is as follows for data transmissions:



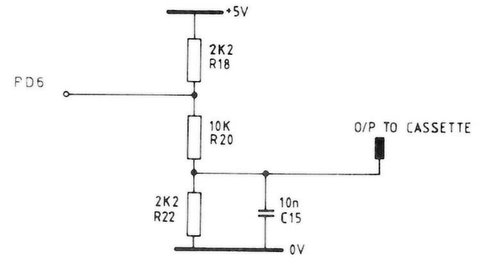
The method of transmission can be seen on the following chart.



Example: A byte at 1200 BAUD of data sent out appears least significant bit first.



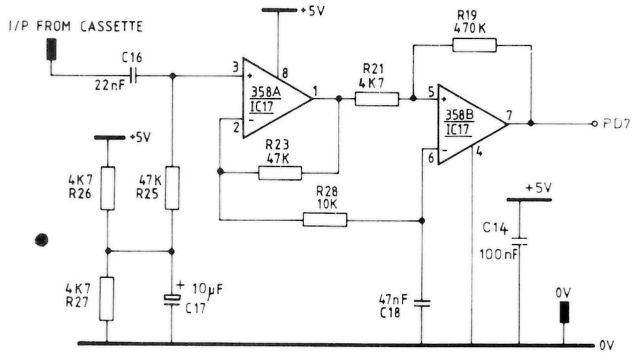
Output to cassette is shown below:



Output To Cassette

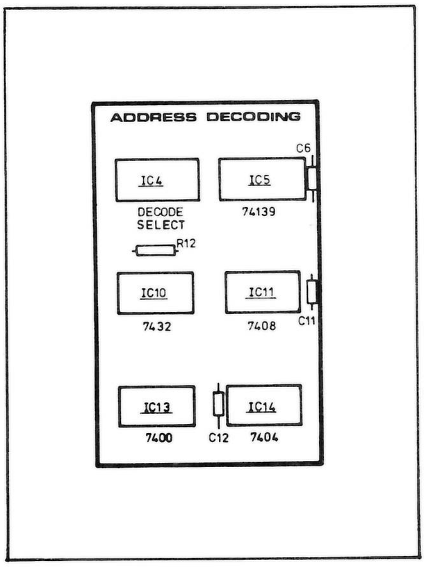
The output consists of a simple attentuation and d.c. level control; C15 acts as a filter for spikes that may occur.

Input from cassette consists of a dual operational amplifier stage (358). The potential divider connected to pin3 ensures that both inputs and the output of the first stage will be biased at the mid point of supply voltage (2.5V; with 5V supply; the second amplifier is a non-inverting stage. The circuit diagram is shown below:

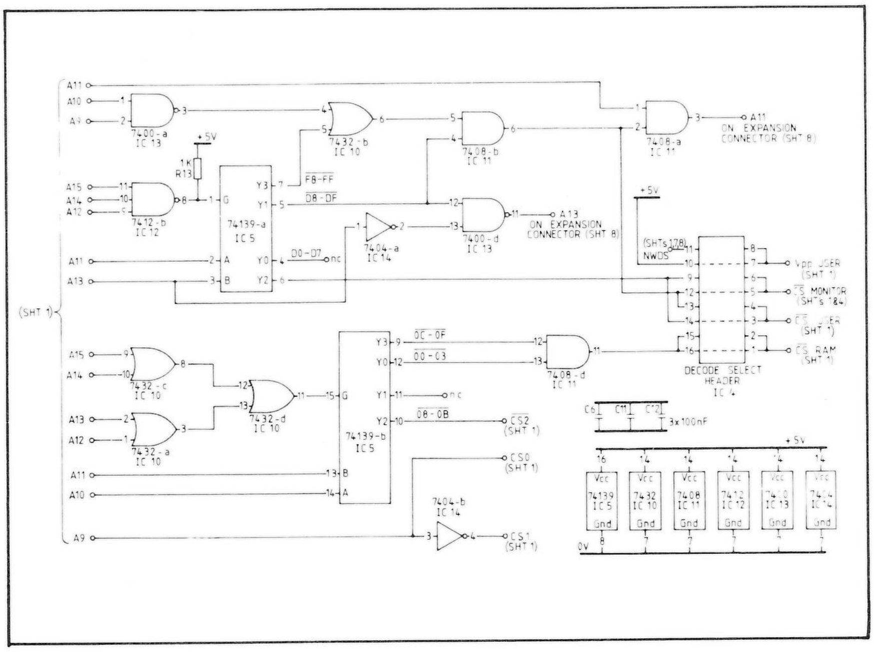


LED 1 is activated from CA2 of '6821'; this indicates when transmission of data is occurring.

Chapter 9 EMMA II Address Decoding



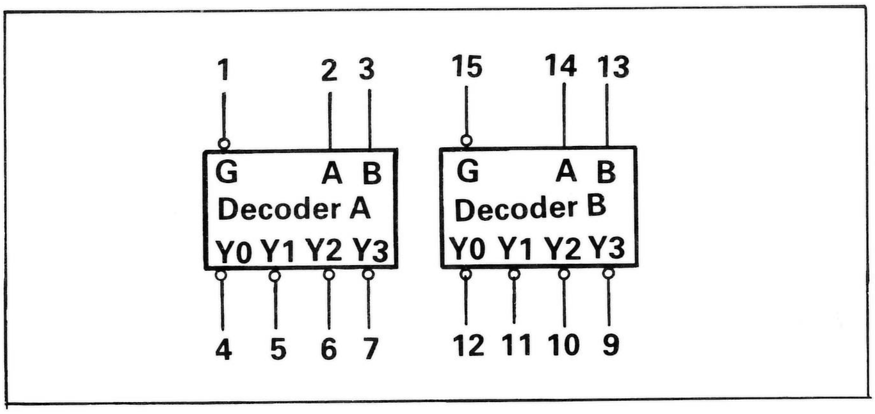
In the EMMA II system address lines A_0-A_{15} are decoded to give the chip select signals required for the memory and I/O circuits.



The circuit diagram for the address decoding.

As can be seen in the diagram,'IC5' plays an important part in this section.

'IC5' (74139) ia dual 2-4 Decoder/Demodulator. It consists of two independent decoders.



Each of these decoders accepting two inputs and providing four mutually exclusive active 'low' outputs, they also have an active 'low' enable input. Pin names are as below:

A B	Address·Input
G	Enable Input
Y_0 - Y_3	Active Low Outputs

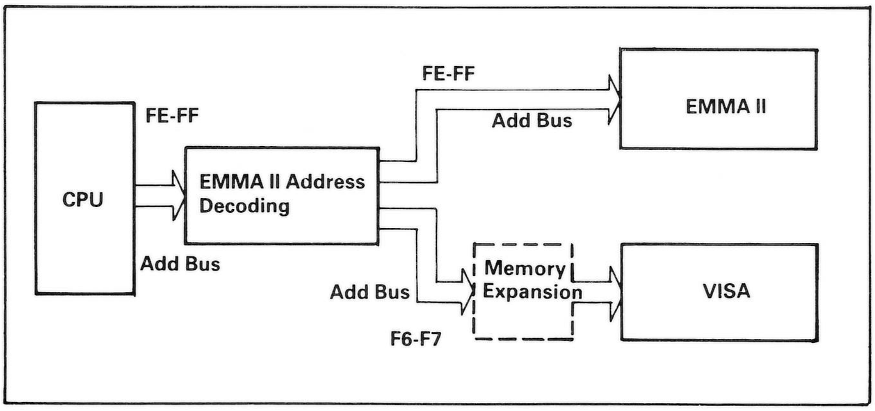
Truth table for one decoder:

Inputs			Outputs			
G	A	B	Y_0	Y_1	Y_2	Y_3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	H	L	H	L	H	H
L	L	H	H	H	L	H
L	H	H	H	H	H	L

H = High Voltage Level L = Low Voltage Level X = Don't Care

The first decoding stage
The enable for pin 1 of the 2-4 decoder is provided by an AND function of A15, A14, A12 at logic '1'. The outputs of the decoder Y3 to Y0 are enabled for address in the range F8-FF, D8-DF, F0-F7 and D0-D7 respectively, these are selected by inputs at pin 3 and pin 2 (A13, A11). The truth table above shows the order of selection.

In order to maintain compatibility with past EMMA models and previous VISA expansion units, the address lines A_{13}, A_{11} have been arranged to give a modified address to the expansion connector. The modified address lines are depicted on the diagram as A13 and A11. A block diagram of this is shown below:



When a monitor program is called for with VISA connected:

FE-FF is modified to F6-F7 for VISA address lines. Nothing exists at this address on VISA so there is no clash with the EMMA II monitor.

When an address in the range D8-DD is asked for with memory expansion and VISA connected:

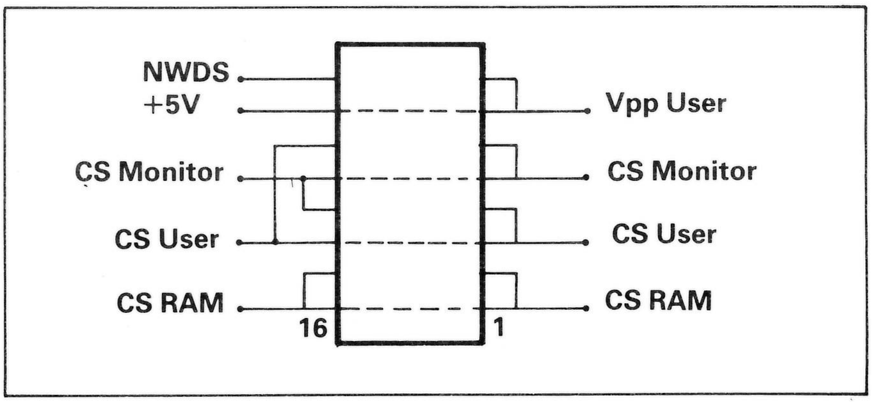
D8-DD is modified to the range F0-F5. This has the effect of setting up the data direction buffers in the correct direction.

The logic gating previous to A11', A13' enable these alterations to be made.

The second section of the address decoding :
Enable pin 15 is provided by A12, A13, A14, A15 through three 'OR' functions, providing any one of those are '0' the 2-4 decoding is enabled and areas 0C-0F, 00-03, 08-0B, are selected by A11 and A10. A9 selects the 6522 and 6821 interface controllers.

Decode Select Header

Of the total addressing capability only a limited amount is decoded. The required address decode is selected by links placed in an IC Header. The EMMA II decode select is shown below:-

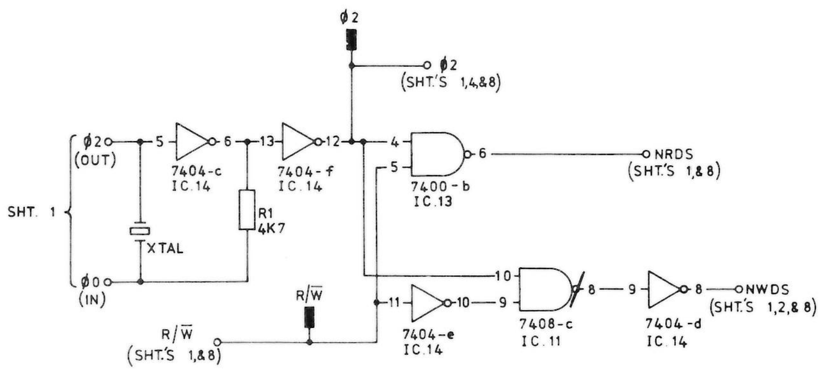


It is important to use the correct header to perform the functions specified by the EMMA II manuals. The select header is provided so that the decoding can be easily adapted to different peripheral devices.

Note: If a RAM is used in the user EPROM socket the header link 7-10 needs to be placed from pin 8-9. This enables the NWDS line for the 6116 RAM.

Chapter 10 Data Strobe Generation

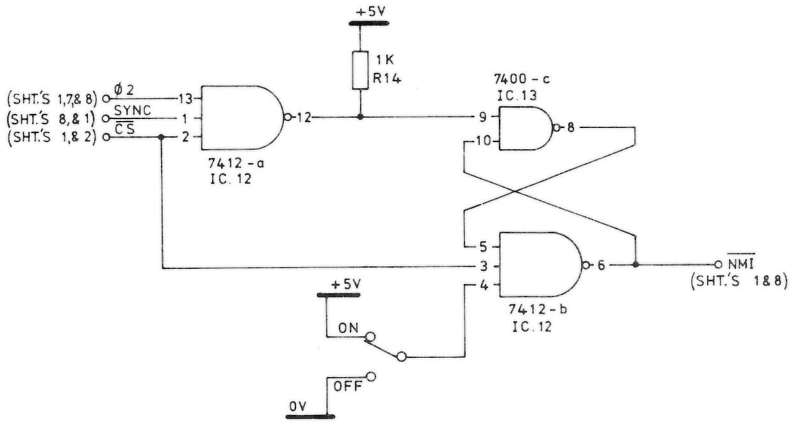
The Ø2 clock signal is shaped and buffered by two inverters as shown below:



DATA STROBE GENERATION CIRCUIT

This is then gated with the 6502 R/$\overline{\text{W}}$ signal to provide the two data strobes required for the RAM and EPROM. These signals are also supplied to the expansion connector.

Chapter 11 EMMA II Single Step Function



SINGLE STEP CIRCUIT

This facility is part of the program debugging facilities. It enables the user to single step through a program and examine the contents of accumulator, X and Y registers

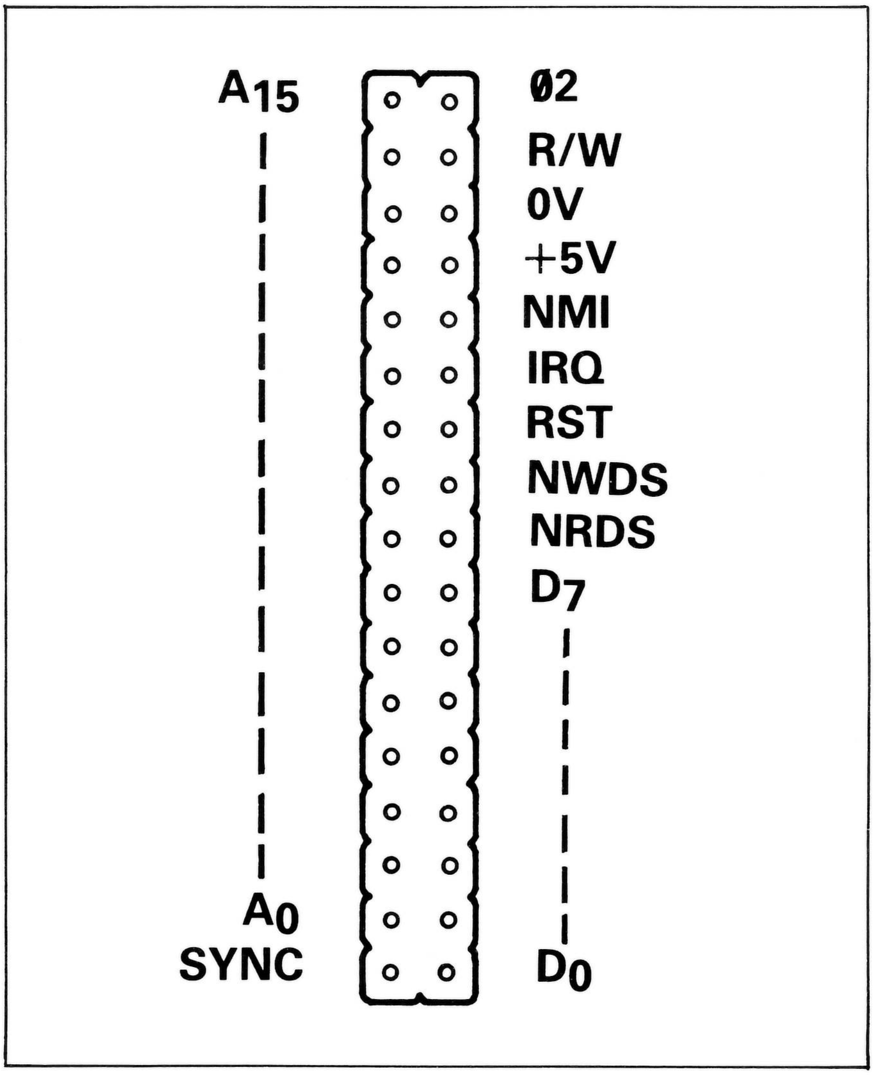
As can be seen above an interrupt can occur on the rising edge of Ø2 only when both SYNC and CS MONITOR EPROM are high. This signal sets the S.R. latch which resets when Eprom enable goes low again. By using open collecter NAND gates (74LS12) the output can be connected directly to the $\overline{\text{NMI}}$ line.

Chapter 12 EMMA II Interconnection Details

The EMMA II microcomputer is based on a 303mm x 228mm printed circuit board. The system layout has been designed to facilitate teaching. Because of this the principles involved have been to:

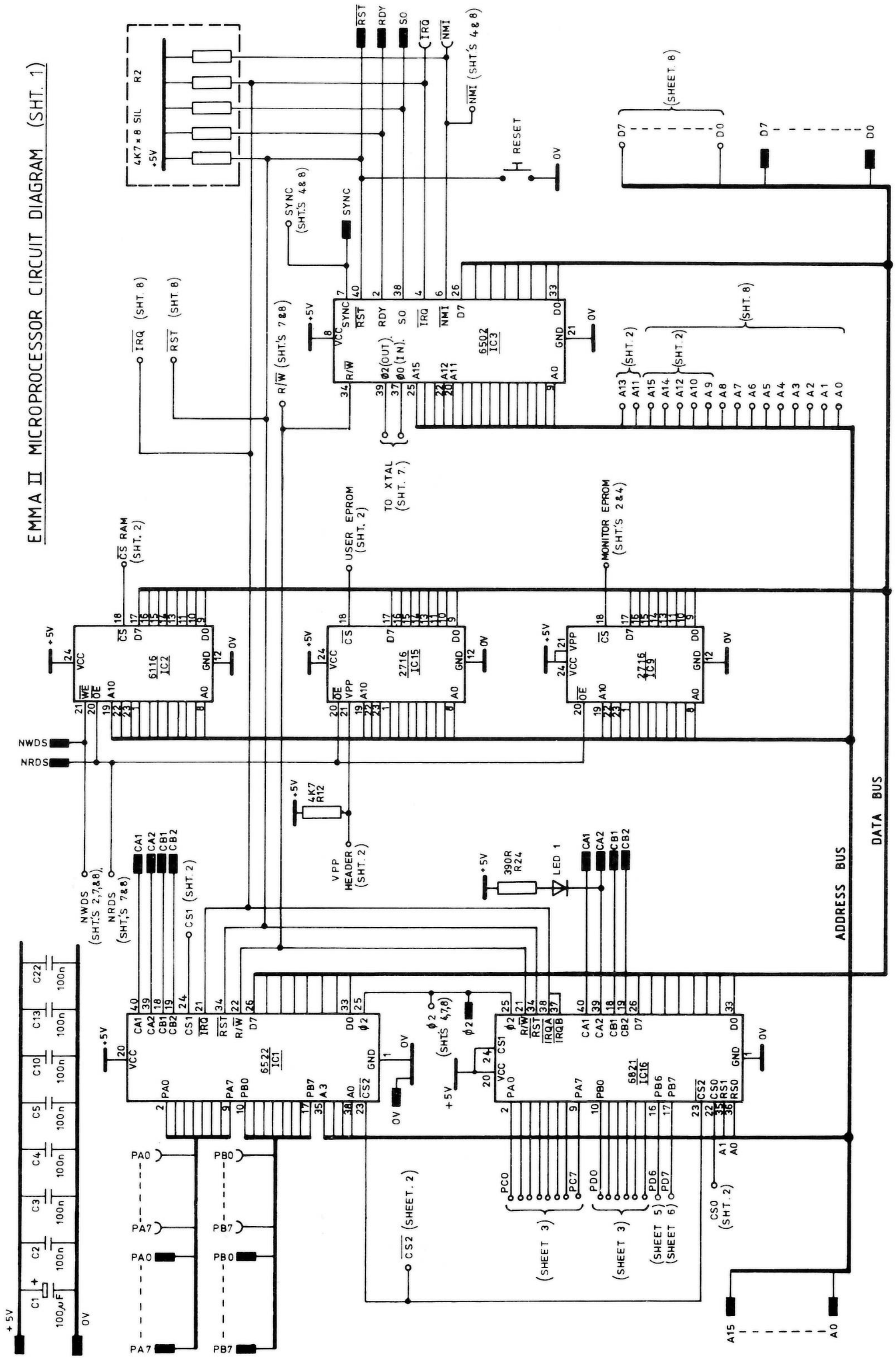
a) Separate the elements into identifiable blocks
b) Make all control bus, data bus and address lines accessible (and identifiable) on 0.1″ spacing connector pins.
c) Have easy access to the user I/O ports via standard 4mm sockets, and 0.1″ connector pins.

The microprocessor and support chip diagram for EMMA II is given overleaf.



Expansion connector pin allocations:

EMMA II Microprocessor System - Circuit Diagram



Part 2 EMMA II Software

This section of the manual describes the programming requirements of the EMMA microcomputer system. It also details the programming requirements and facilities of the 6522 and 6821 Interface Adaptors.

Chapter 13 The Microprocessor Structure

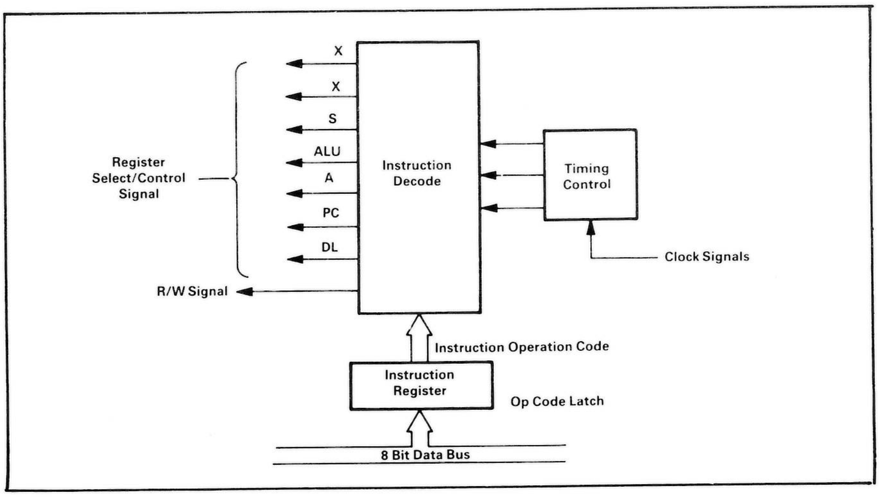
Internal Operation

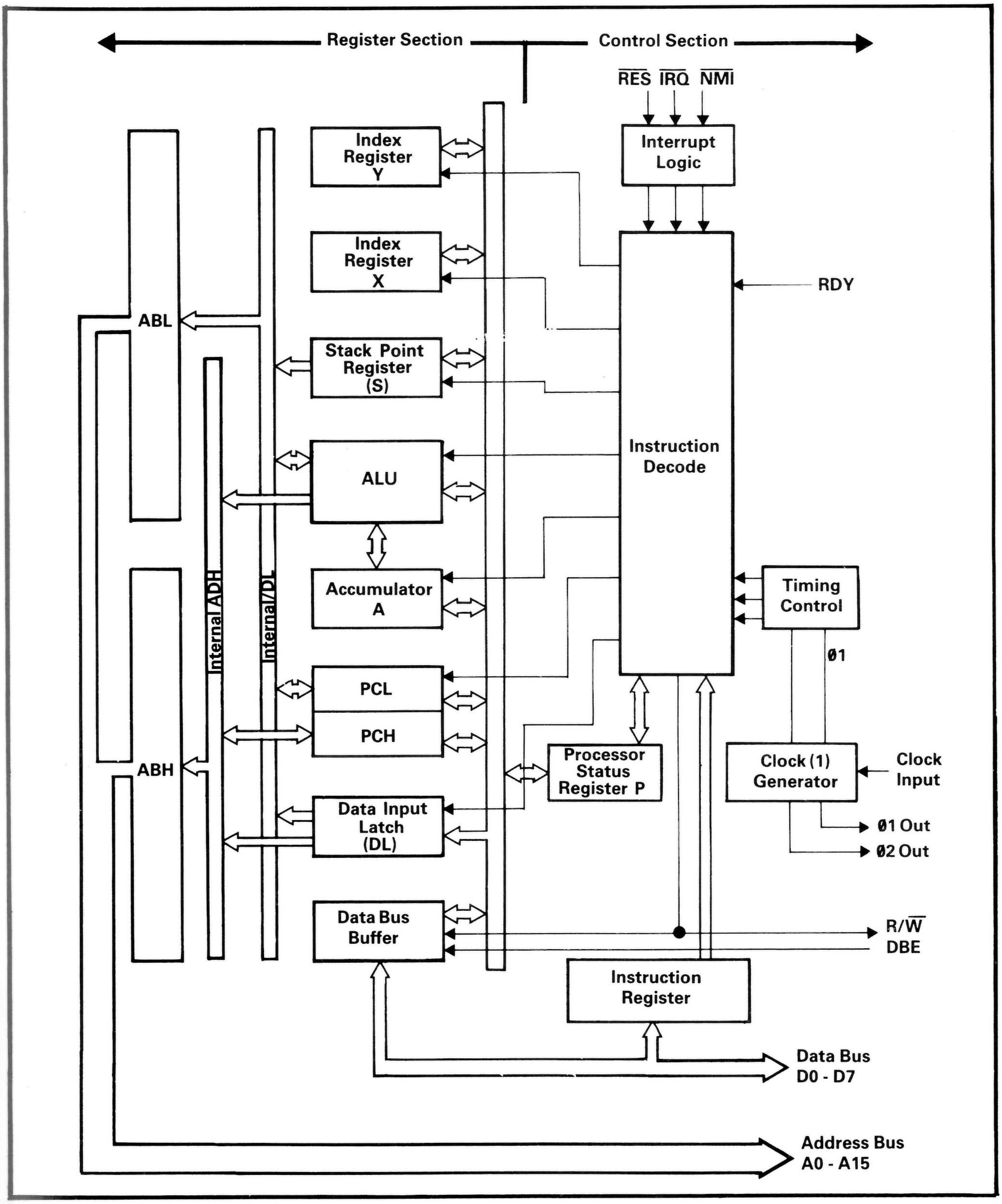
The processor is split into two distinct sections: Register Section and Control Section. The Register Section implements all the data transfers from external memory together with any data arithmetic/logic operations that may be required. The control section which decodes user program instructions accessed sequentially from external memory. The program operation codes (Op.Codes) enter the processor via the Data Bus and are latched into the Instruction Register for subsequent decoding. All memory addresses are generated by the processor and appear on the Address Bus.

Each program Instruction requires a number of discrete operations to effect it. It is the function of the Timing Control Unit to correctly sequence these. At the beginning of each instruction the unit is reset to zero and advanced at the beginning of each Ø1 clock pulse until the instruction is completely executed.

The diagram: 'Control Section - Instruction Decode' indicates the scheme. An Op.Code appearing on the data bus is latched into the instruction register. This is eventually decoded together with timing control signals, to provide an appropriate sequence of single- cycle control signals for corresponding registers in the chip register section. The following diagram: 'Register Section - Data Bus Accumulator' indicates the registers involved in transferring data from the data bus to the accumulator. The sequence of events would be determined by the control unit and as dictated by the preceeding Op.Code. For a simple LDA# (Load accumulator - immediate addressing mode) instruction, data appearing on the data bus would be latched into the Data Latch (DL) via the data bus buffer for subsequent transfer to the accumulator via the internal data bus.

Control Section Instruction Decode

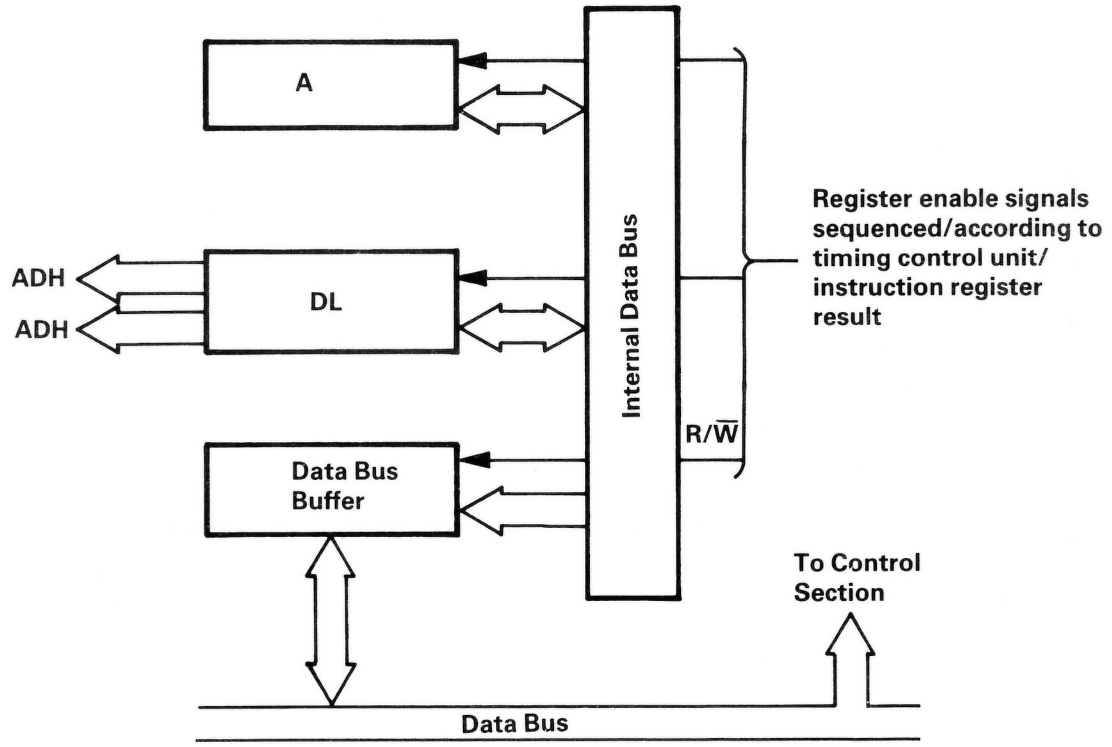



Register Section
Control Section
RES IRQ NMI
Interrupt Logic
Index Register Y
Index Register X
Stack Point Register (S)
ALU
Accumulator A
PCL
PCH
Data Input Latch (DL)
Data Bus Buffer
ABL
ABH
Internal ADH
Internal/DL
Instruction Decode
RDY
Timing Control
Ø1
Processor Status Register P
Clock (1) Generator
Clock Input
Ø1 Out
Ø2 Out
R/W
DBE
Instruction Register
Data Bus D0 - D7
Address Bus A0 - A15

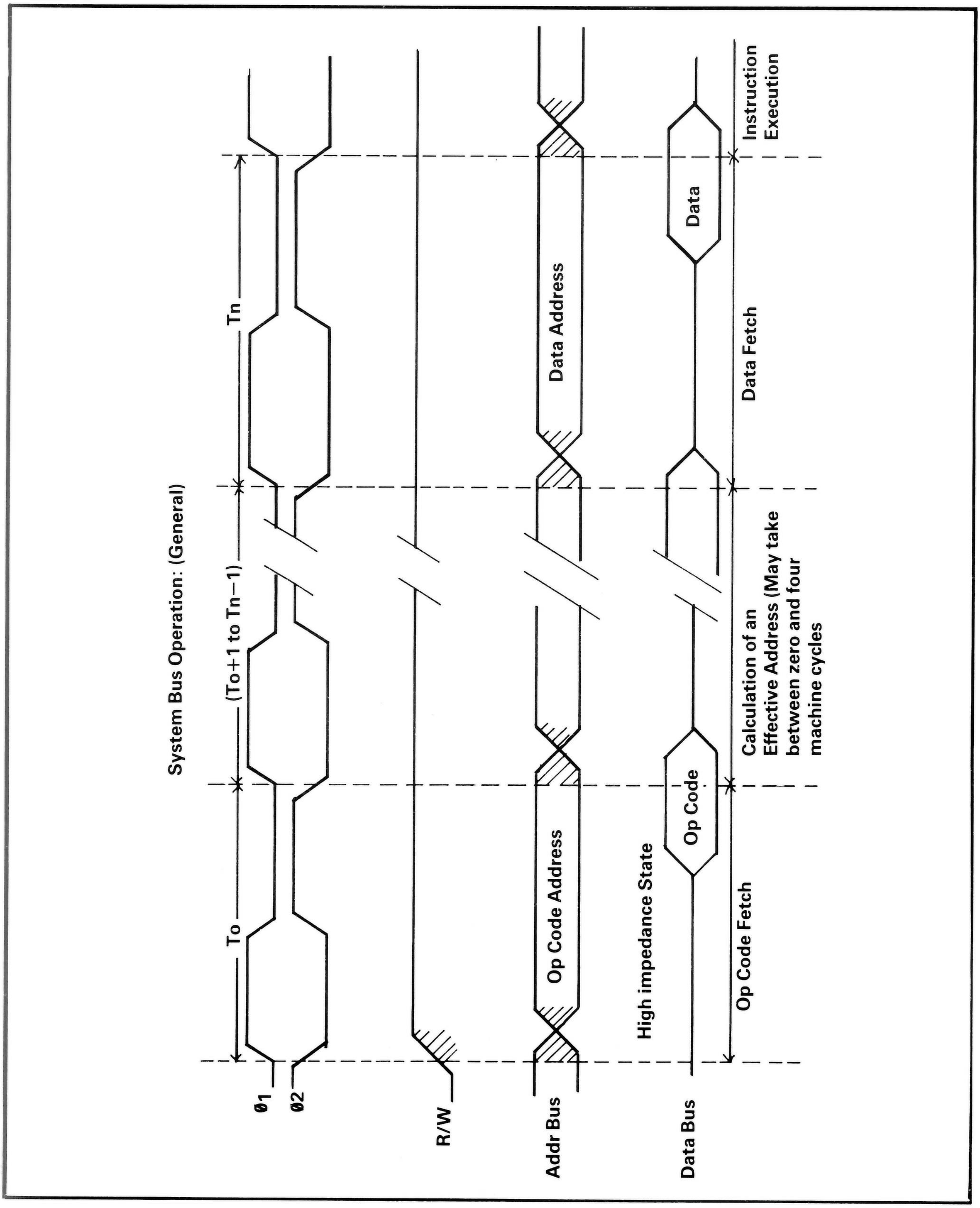
Register Section - Data Bus - Accumulator

Register Section

Data Bus - Accumulator

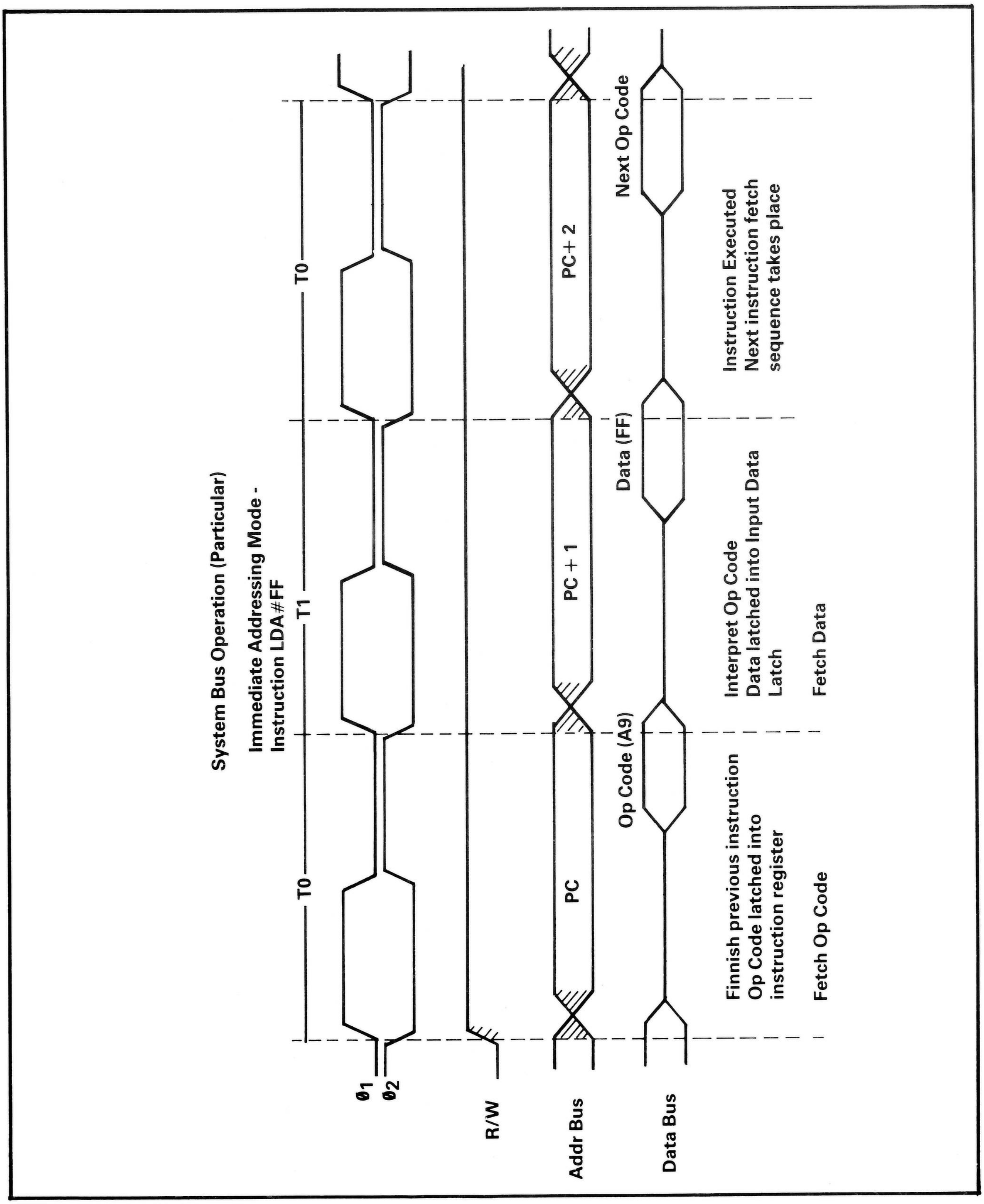

A
ADH
ADH
DL
Internal Data Bus
Data Bus Buffer
R/$\overline{W}$
Register enable signals sequenced/according to timing control unit/ instruction register result
To Control Section
Data Bus

System Bus Operation (General)



System Bus Operation: (General)

System Bus Operation (Particular)



System Bus Operation (Particular)

Immediate Addressing Mode -
Instruction LDA#FF

System Bus Operation

From a test and fault finding point of view we do not need to know the exact flow of data or signals internal to the chip. What is required however is an intimate knowledge of the signals appearing on the external address bus, data bus and control lines for each clock cycle during the execution of any valid program instruction or external control signal. The diagrams, System Bus Operation - General and Particular indicate the relative signal timing.

It should be observed from these diagrams that the Address Bus always contains known data except for a brief period (when it is unstable due to change) at the start of $\emptyset_1$ clock cycle, while the data bus only becomes stable during the later part of $\emptyset_2$ and remains so for a very brief period of $\emptyset_1$. During the rest of the time the data bus is 'floating'.

All internal data transfers take place during $\emptyset_1$ and all inter-chip data transfer during $\emptyset_2$.

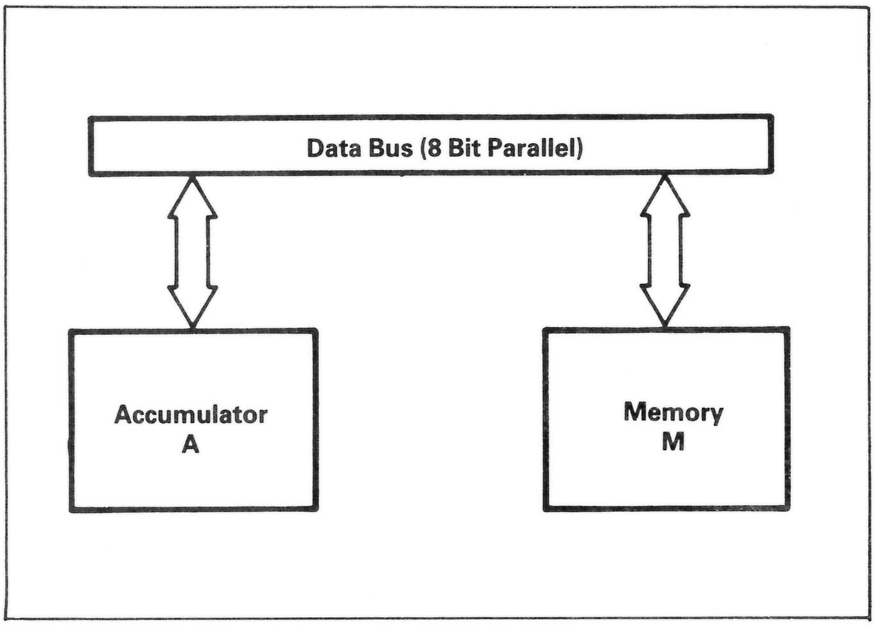
In addition to the registers already mentioned the following may also be involved in any given data transfer/manipulation: Index Register (X and Y), Satck Pointer (S), Arithmetic/Logic Unit (ALU), Program Counter (PC) and the Process Status Register (P).

So far the internal functioning of the processor has been considered as being simply under program control. However, the sequence of program instructions can also be affected by the Interrupt Logic and the Process Status Register. The Interrupt Logic provides an interface between the interrupt lines, Reset ($\overline{RES}$), Interrupt Request ($\overline{IRQ}$) and Non Maskable Interrupt ($\overline{NMI}$) and the processor to ensure correct timing, enabling and sequencing of these inputs to the processor. The processor Status Register controls certain processor operations which may be required as the result of the processor arithmetic and logic operations.

Chapter 14 Data Manipulation

Data Bus

Data is transferred to and from the microprocessor via the 8-bit parallel data bus. The diagram shows how external memory is connected to the internal 8-bit accumulator register by the bi-directional data bus.



Accumulator

All transfers between memory locations must be made through a microprocessor internal register. The accumulator, or an index register, must be used for this purpose.

- Consider the following instructions:

  Load Accumulator with Memory, LDA

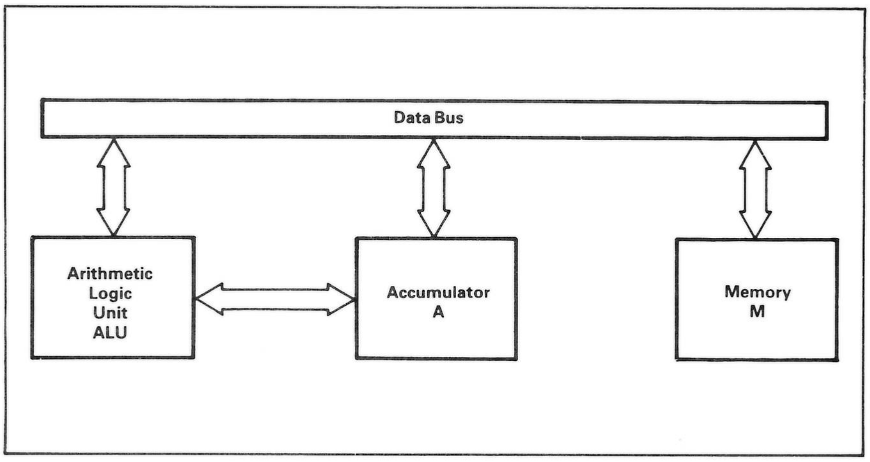
  The instruction LDA, causes the microprocessor to transfer data from memory into the accumulator. The symbolic representation for this instruction is (M)→A, where the arrow is read as "transfer to" and the brackets as "contents of".

  Store Accumulator in Memory, STA

  The instruction STA, transfers the contents of the accumulator into memory. Symbolically (A)→M.

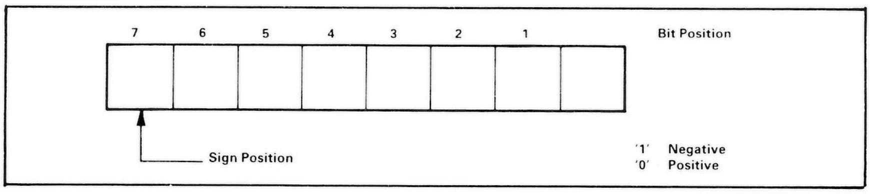
Chapter 15 Arithmetic Unit

The 6502 microprocessor has an 8-bit arithmetic unit, capable of performing arithmetic and logical operations on data. The arithmetic unit interconnects with the data bus and the accumulator as shown below:



Since the arithmetic unit handles 8-bit binary numbers, the range is 0 to 255, giving 256 values. If an arithmetic operation gives a result which is larger that 225, then a ninth bit is generated. This bit is termed the 'carry'

ADC Add Memory to Accumulator with Carry
This instruction adds the contents of the memory and the carry to the accumulator, placing the result in the accumulator. Symbolically $A + M + C \rightarrow A$. The arithmetic unit can operate in true binary or in binary coded decimal mode as selected by the user. In decimal mode the arithmetic unit will treat the 8-bit numbers as two 4-bit codes, each having a value of 0 to 9. Multiple precision arithmetic can be performed, allowing numbers larger than 255 to be operated upon. Numbers within the microprocessor may be considered as positive or negative, depending upon the value of the most significant bit to represent the sign of the number means that the number range becomes +127 to −128. Within the microprocessor there is a binary digit, termed the overflow flag, which indicates when an error has occured in operations involving signed numbers. The format of signed numbers is indicated in the diagram.



SBC Subtract Memory from Accumulator with Borrow.
This instruction subtracts the value of memory and borrow from the contents of the accumulator, using two's complement arithmetic, and stores the result in the accumulator. Borrow is defined as the complement of the carry. When SBC is used on single precision numbers, the carry must be set before the subtraction is performed. Symbolically this is A - M - C → A.

Note: Decimal mode addition and subtraction can only be performed using unsigned numbers. In binary arithmetic the carry flag is set by a result greater than 256; in decimal mode the flag is set with result greater than 99.

The arithmetic unit also has the facility to perform logical operations.

AND Memory with Accumulator
This instruction performs a logical AND of each bit of the accumulator with each bit of the selected memory, placing the result in the accumulator. If result is zero, the zero flag sets. If the most significant bit of the result is '1', the negative flag is set. Symbolically; A ∧ M → A.

ORA Memory with Accumulator
This instruction performs a logical 'OR' of each bit of the accumulator with each bit of the selected memory, placing the result in the accumulator. Symbolically: A ∧ M → A.

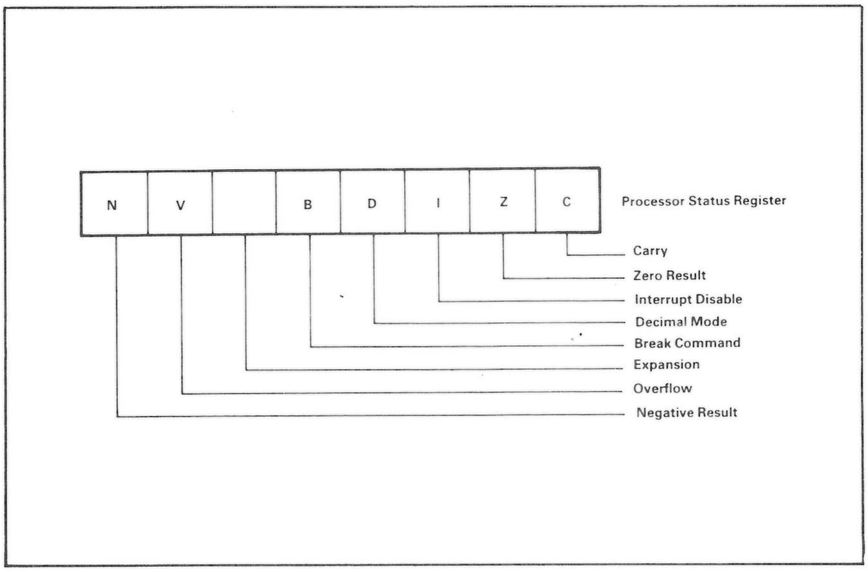
EOR Exclusive OR Memory with Accumulator.
This instruction performs the logical Exclusive OR with accumulator and memory, on a bit-by-bit basis, placing the result in the accumulator.

Note: That the EOR instruction may be used to complement data, e.g.

```
LDA    10101101
EOR    11111111
STA    01010010
```

Chapter 16 Status Register

The flag signals already mentioned (carry, zero, negative, overflow) are part of an 8-bit register within the microprocessor. This register is termed the processor status register, P. Each bit of the P register represents a totally independent flag:



CARRY Flag (C):
This flag is modified by arithmetic operations, as mentioned earlier. It is used by rotate and shift instructions as a ninth bit. The CARRY bit can be set or reset by the programmer. A SEC instruction will set and a CLC instruction will reset the flag.

ZERO flag (Z):
This flag is set by the microprocessor during any data movement or calculation operation when the 8-bit result is zero.

INTERRUPT DISABLE flag (I):
This flag disables the IRQ input. The I flag is set by the microprocessor during reset and interrupt commands; it can also be set or reset by the programmer, using the SEI or CLI instruction.

DECIMAL MODE flag (D):
This determines whether the arithmetic unit performs true binary or binary coded decimal arithmetic. The programmer can set or reset the flag, using the SED or CLD instruction.

BREAK COMMAND flag (B):
This flag is set only by the microprocessor and is used to determine during an interrupt service sequence whether or not the interrupt was caused by the break instruction or by a $\overline{\text{NMI}}$ or $\overline{\text{IRQ}}$. There are no instructions which can set or reset this bit.

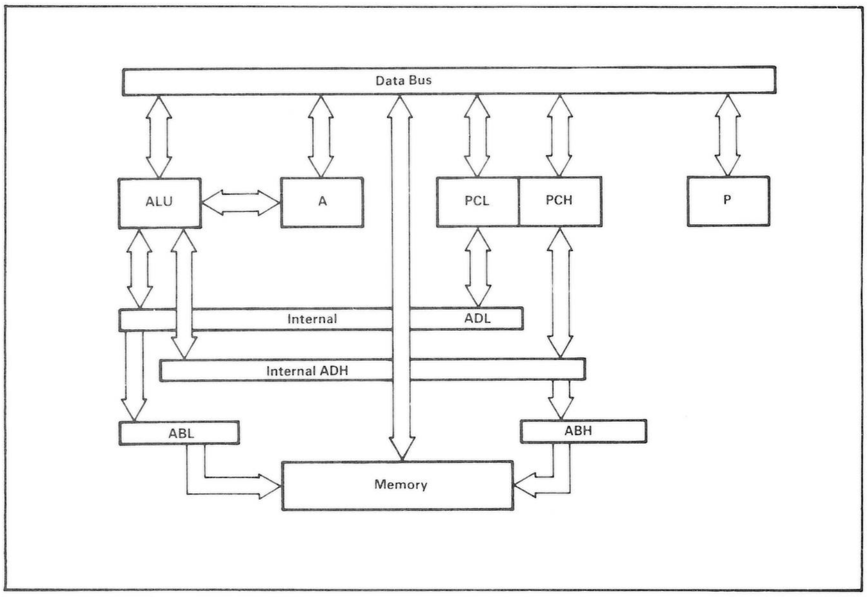
EXPANSION BIT:
This unused flag may appear as a '1' or '0' when the status register is examined. This flag will be used in expanded versions of the 6502.

OVERFLOW flag (V):
When using signed numbers, this flag will indicate when the 7bit result overflows into the sign bit. If not using signed numbers this flag can be ignored. If this flag is set after a signed number add or subtract, the programmer must apply a sign correction routine. The V flag is also used to indicate the state of bit 6 on the data tested with the BIT instruction.

NEGATIVE flag (N):
The N flag takes the state of bit 7 of the resulting value in all data movement and arithmetic operations. Thus, when using signed numbers, the N flag indicates whether data is positive ('0') or negative ('1').

Chapter 17 Program Counter

To work through a sequence of instructions as it automatically executes a program, the microprocessor has to have a register which keeps track of the address of the next location to be accessed. The interconnection of microprocessor registers, including the program counter, is shown below:



The program counter is a 2-byte register, giving an address which can access 65,536 (64K) locations. The lower 8-bits of address are provided by the program counter low register (PCL), the upper 8-bits are provided by the program counter high register (PCH).

The 16-bit address held in the program counter is transmitted to the external memory via the address bus.

Having been initially set to the address at which a program commence, the program counter will increment automatically after each fetch operation. To change the sequence of the program, certain instructions are provided which allow the programmer to modify the contents of the program counter.

JMP Jump to New Location
In this instruction, the data from the memory location accessed next in sequence after the JMP code is loaded into PCL and the data in the next location after that is loaded into PCH.

Example of JMP instruction:

Address	Data	Comments
0024	JMP	Jump to location 0937
0025	37	New PCL byte
0026	09	New PCH byte
0937	Next instruction	

This method of forming the address from the next two bytes in sequence after the instruction is termed the absolute addressing mode. The jump instruction can also make use of indirect addressing as described later.

Note: that the jump instruction is not conditional on any test; i.e., whenever a jump instruction is accessed in a program the program counter will always be modified.

To allow for conditional modification of the program counter, dependant upon the state of a flag, the 6502 has a number of branch instructions. All branch instructions are relative; i.e.,the program counter contents will be increased or decreased by an amount given as data by the programmer.

BMI Branch on Result Minus
This instruction branches if the N flag is set.

BPL Branch on Result Plus
This instruction is complementary to BMI; i.e.,branch occurs when N flag is reset.

BCC Branch On Carry Clear
This instruction branches conditional on the C flag being reset.

BCS Branch on Carry Set
Branch occurs when carry is set.

BEQ Branch on Result Zero
Branch occurs conditional on the Z flag being set.

BNE Branch on Result Not Zero
Branch when Z flag reset.

BVS Branch on Overflow Set
Branch occurs when the V flag is set.

BVC Branch on Overflow Clear
Branch occurs when the V flag is reset.

Note: That a branch is limited to +127 or −128 relative to the current program counter location. Complementary branch instructions are given with the 6502 microprocessor to facilitate branching outside this range. Consider that a branch of several pages is required if the carry is set.

Address	Data	Comments
0031	BCC	Branch on carry clear
0032	+3	
0033	JMP	Jump to address 1243
0034	43	PCL
0035	12	PCH
0036	Next instruction	

If carry is not set the jump instruction will be by-passed. If carry is set the jump will modify the program counter to 1243.

Note: that when the branch instruction at address 0031 is to be obeyed, the data at address 0032 will be fetched. The program counter will then hold 0033, therefore the data at 0032 must increment (or decrement) the program counter from 0033. In the example, 0033 + 3 gives address 0036 as the address of the next instruction.

There are two 6502 instruction which are designed only to set flags, prior to testing for a branch instruction.

CMP Compare Memory and Accumulator This instruction subtracts the contents of memory from the contents of the accumulator. Symbolically; A - M. This instruction does not change the accumulator or the memory, it simply sets or resets flags as a result of the subtraction.

Z flag sets if A - M = Ø, reset if A - M ≠ Ø
N flag sets if result negative (bit 7 '1')
C flag sets if M is less than, or equal to, A.

BIT Test Bits in Memory with Accumulator
This instruction performs a logical AND between a memory location and the accumulator. Symbolically; M ∧ A. This instruction does not change the accumulator or the memory; it simply sets or resets flags.

N flag takes the value of bit 7 of the memory being tested.
V flag takes the value of bit 6 of the memory being tested.
Z flag sets if the result of the AND is zero.

Full details of the 6502 instruction set are given in the Appendix of the EMMA II User Manual.

Chapter 18 Addressing Modes

The 6502 uses a 16-bit address bus, enabling the microprocessor to access 64K locations. There are a number of methods available to the programmer for forming the address required for use during the execution of an instruction.

These addressing modes fall into two categories, non-indexed and indexed.

To consider first the non-indexed methods.

Implied Addressing

The instruction will consist of a 1-byte operation code, causing an operation internal to the microprocessor. No external memory address is necessary for this type of instruction; e.g.,CLD clear decimal flag.

Immediate Addressing

The instruction will comprise 2-bytes. The first byte will be an operation code, the second byte will be the data to be operated upon.

Address	Data	Comments
0024	A9	This is a load accumulator instruction
0025	FF	Data FF is loaded to accumulator
0026	next instruction	

Absolute Addressing

Gives a 3-byte instruction. Instruction comprises a 1-byte operation code, followed by a low-byte address, then by a high-byte address.

Address	Data	Comments
0042	AD	This is a load accumulator instruction
0043	03	The accumulator is loaded with the data stored at 0903
0044	09	
0045	Next instruction	

Zero Page Addressing

Gives 2-byte instructions. 1-byte is given to the operation code, with the second byte being a location on zero page of the memory.

Address	Data	Comments
0E10	A5	This is a load accumulator instruction
0E11	32	The accumulator loads with the data at 0032
0E12	Next instruction	

Zero page is a form of absolute addressing but being a 2-byte instruction it takes only 3 machine cycles, unlike absolute addressing which takes 4 machine cycles. Program execution can be made more rapid by locating frequently accessed data on page zero and thus reducing the number of machine cycles.

Relative Addressing

This type of addressing is used only by branch instructions. The instructions are two byte, a 1-byte operation code for a branch being followed by a 1-byte offset which can modify the program counter to a maximum of +127 to −128.

Address	Data	Comments
0214	30	Branch on result minus (N flag = '1'
0215	42	Offset of 42, ie program counter + 42
0258	Next instruction. Note: that the program counter contains 0216 before it is incremented by 42.	

The offset is interpreted as a signed hexadecimal number.

More powerful addressing methods are possible if the microprocessor computes the address using the index register

Absolute Indexed

Absolute indexed addressing is absolute addressing with an index register added to the absolute address.

Address	Data	Comments
0231	BD	Load accumulator, absolute, X. The accumulator will be loaded with the data held at address (034C + contents of index register X).
0232	4C	
0233	03	

Absolute indexed addressing may make use of register X or register Y.

Zero Page Indexed

The operation code is followed by a 1-byte address, to which is added the contents of index register X.

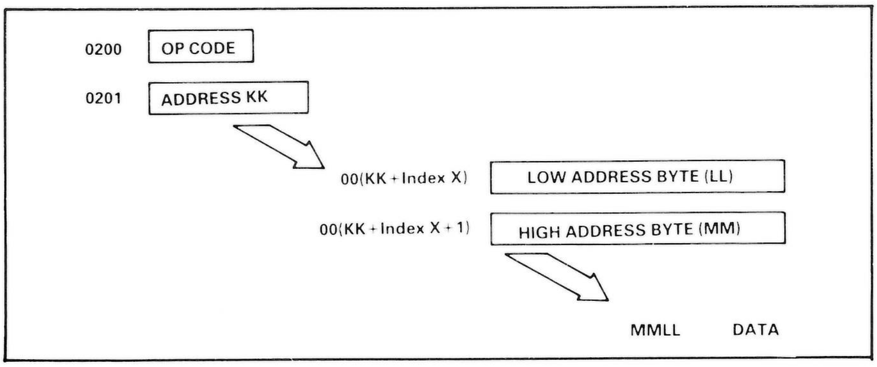
Address	Data	Comments
0200	B5	Load accumulator, zero page X The accumulator is loaded with the data held at address (00F3 + X)
0201	F3	

Note that this indexing mode cannot change page; i.e.,should the modified address generate a carry it will be ignored. For the above example, if index register X contained 15, then the accumulator would be loaded from F3 + 15 =08.

Indexed Indirect Addressing

Instructions will comprise 2-bytes: a 1-byte operation code followed by a 1-byte zero-page address. This address will be added to the contents of the X register to give the zero page location of the low address byte. The next location will provide the high address byte. This is shown pictorially.

Pictorially:

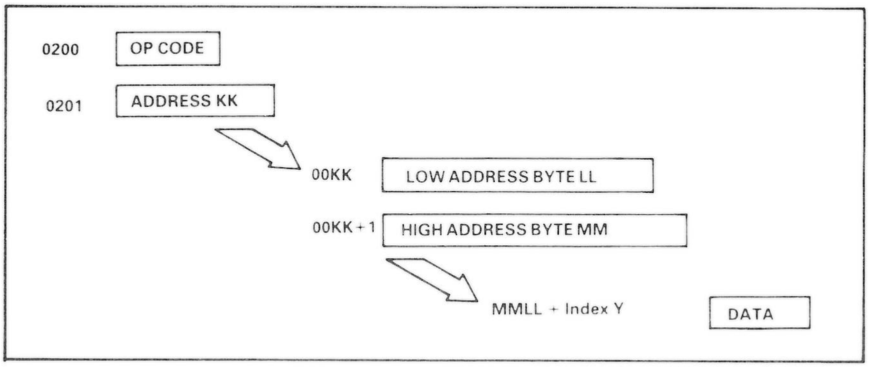


Whilst this addressing code does provide a method of accessing a 2-byte address using only a 2-byte instruction, it does necessitate devoting part of zero page to address storing. It also takes 6 machine cycles to execute this form of instruction:

Note that the computed address formed will always be on page zero, if a carry is generated during the addition of index register X it is ignored, in similar manner to that mentioned in zero page indexed mode.

Indirect Indexed Addressing
This is an extension of the previous mode, giving a computed address on any memory page.

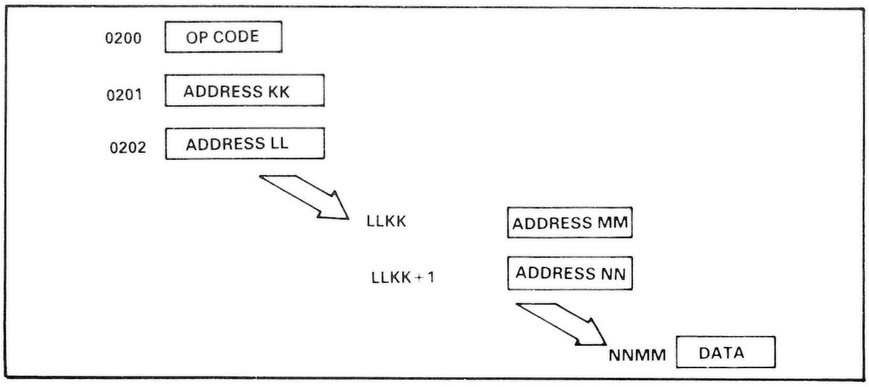
Pictorially:



This method does permit page crossing, i.e. if LL + Y generates a carry then this will be added to MM.

Indirect Absolute Addressing
This mode applies only to the JUMP instruction, the 2-byte address code, following the operation code gives access to the low byte data address, incrementing by 1, gives the high byte data address.

Pictorially:



Index Register Applications

The primary use of the index registers, X and Y, is as offset registers and counters for computed addresses. Both registers have instructions for increment, decrement, compare with memory, transfer to and from accumulator, load from memory and store in memory. This range of instructions makes the index registers useful for interim storage of data.

Chapter 19 Stack Processing

For most programming operations it is convenient to use known memory locations for the storage of data. However, the precise memory location is not known, only the order in which data may be sequenced being known. This type of programming is called re-entrant coding and is often used in servicing interrupts.

To implement this type of addressing the microprocessor has a separate address generator, termed the stack pointer, S, which is used by the program to access memory.

The stack pointer uses the push down stack concept, for example consider 4 data bytes which are to be stored in sequence, $DATA_1$ $DATA_2$ $DATA_3 DATA_4$. The stack pointer will store $DATA_1$, then decrement, store $DATA_2$, then decrement, etc. After 4 store operations we will have

	Address	Data
	(A)	$DATA_1$
	(A-1)	$DATA_2$
	(A-2)	$DATA_3$
Stack Pointer →	(A-4)	

The data is stored in sequential locations, with the stack pointer always pointing to the next 'empty' location. In recalling the data from stack, the stack pointer will repeatedly increment. Thus, in the example given, $DATA_4$ is accessed first, then $DATA_3$, then $DATA_2$, then $DATA_1$. Leaving the stack pointer pointing to location A.

For the 6502, the stack pointer is an 8-bit register, with most significant address byte set to 01. Thus, the stack uses page 1, with the pointer resetting to FF. Allowing all page 1 for stack operations gives a 256 location stack.

6502 Instructions Which Utilize The Stack Area

A list of the 6502 instructions which utilize the stack can be found overleaf on page 57.

JSR Jump to Subroutine
This instruction changes the program counter contents to the start location of the subroutine, but before this change occurs, the last address accessed is stored on the stack. This permits the microprocessor to return to continue the program when the subroutine has been completed.

To illustrate the JSR instruction:

Program Counter	Memory Contents	
0400	JSR	
0401	1E	Subroutine starts at 2F1E
0402	2F	
Stack Pointer	**Stack Contents**	
01FF	04	Return address 0402 is placed on the stack
01FE	02	

The JSR instruction takes 6 machine cycles. During the return from subroutine instruction the program counter is incremented causing a return to the next program instruction , i.e. for the above example, the program counter contents of 0402 are restored and then incremented to 0403 before the return from subroutine is completed.

RTS Return from Subroutine
This instruction loads the program counter low and program counter high from the stack to the program counter. The stack pointer will increment twice during this instruction.

The complete sequence for executing a subroutine is:

Address	Data		
0200	JSR		
0201	12		
0202	03	Address	Data
0203	Next instruction	0312	First Instruction of subroutine
		:	
		:	
		03XX	RTS

PHA Push Accumulator on Stack
This instruction transfers the current value of the accumulator to the next location on the stack, automatically decrementing the stack to pointer to the next empty location.

The symbolic notation for this operation is A ↓ . Note that the notation means push onto stack, ↑ means pull from the stack.

PLA Pull Accumulator from Stack
This instruction increments the stack pointer, then loads the accumulator from this incremented pointer address.
Symbolically: A ↑ .

The stack push and pull instructions enable the programmer to make use of the stack for temporary data storage during programs.

TXS Transfer Index X to Stack Pointer
This instruction transfers the value in the index register X to the stack pointer. Symbolic notation is X→S.

TSX Transfer Stack Pointer to Index X
This instruction transfers the value in the stack pointer to the index register, X.
Symbolic notation is S→X.

PHP Push Processor Status on Stack

This instruction transfers the contents of the processor status register to the stack. Symbolically, P ↓ . Note that the status register contents are automatically saved when an interrupt occurs, but not saved on a jump to subroutine.

PLP Pull Processor Status from the Stack
This instruction increments the stack and transfers the data accessed to the status register.

Chapter 20 Reset and Interrupts

Reset

When a reset signal is applied to the microprocessor it fetches a 2-byte address from pre-determined memory addresses and loads this address into the program counter. The 2-bytes forming the fetched address are termed vectors.

The 6502 fetches its reset vectors from addresses FFFC (PCL) and FFFD (PCH). For the 'EMMA', the monitor program resides at pages FE and FF, thus the reset vectors are within the monitor.

Interrupts

The 6502 can be interrupted during the execution of a program by an external signal. Before responding to an interrupt the microprocessor will complete its current instruction. The current program counter address (2-bytes) and the status register contents are then placed on the stack so that the microprocessor can return to its current program after the interrupt has been serviced.

Two types of interrupt are available with the 6502:

(i) Non-maskable interrupt, $\overline{\text{NMI}}$, this is an edge sensitive input to the 6502, an internal flag is set whenever the $\overline{\text{NMI}}$ input goes from '1' to '0'. This input cannot be disabled by the microprocessor. The $\overline{\text{NMI}}$ vectors are obtained from FFFA and FFFB.

(ii) Interrupt request, $\overline{\text{IRQ}}$ this is a level sensitive input to the 6502. An interrupt will occur when the $\overline{\text{IRQ}}$ input is at '0' if the interrupt disable flag (bit 2 of the status register) is reset. The $\overline{\text{IRQ}}$ vectors are obtained from FFFE and FFFF.

If both $\overline{\text{NMI}}$ and $\overline{\text{IRQ}}$ occur the 6502 will give priority to the $\overline{\text{NMI}}$. Peripheral devices may be wire OR'd to the $\overline{\text{NMI}}$ and $\overline{\text{IRQ}}$ lines, but it must be noted that the edge triggering of the $\overline{\text{NMI}}$ does make a difference to the interrupt handling. With the $\overline{\text{IRQ}}$ line, any number of devices may hold this line low. Each device will be serviced before the line goes high, clearing the $\overline{\text{IRQ}}$. Once a device has set the $\overline{\text{NMI}}$ flag, other 1→0 edges will be ignored.

All interrupt routines should end with the single byte return from interrupt instruction, RTI, which restores the interrupted program and continues normal execution.

The time taken to initiate interrupts means that only devices with byte transfer rates below about 40kHz should use the interrupt facility.

BRK Break Command

The break command is an instruction which causes an interrupt sequence. Typical use for the break command is in program debugging, where the sequence of instruction execution may be interrupted at any chosen point. When a break command is received by the microprocessor, the contents of the program counter +2 is placed on the stack, and the program counter is loaded with the vectors at FFFE and FFFF. Symbolic notation for this is PC2 ↓ (FFFE)→PCL, (FFFF)→ PCH. The status register (P) is also pushed onto the stack (at stack pointer +1) when a BREAK occurs. The programmer can examine the BREAK flag within this register to determine whether the interrupt was caused by a software BREAK or a hardware $\overline{\text{IRQ}}$.

Chapter 21 Further Arithmetic and Logical Commands

LSR Logical Shift Right
This instruction shifts either the accumulator or a specified memory contents 1 bit to the right. The high bit of the result is always set to '0', the low bit is shifted into the carry flag.

Effect of LSR command on the status register:

(i) The shift right does not affect the overflow flag.
(ii) The N flag is always reset.
(iii) The Z flag indicates the condition of the result.
(iv) The carry flag is set equal to bit 0 of the input.

ASL Arithmetic Shift Left
This instruction shifts either the accumulator or a specified memory contents 1 bit to the left. The low bit of the result is always set to '0' the high bit is shifted into the carry flag.

Effect of ASL command on the status register:

(i) overflow flag not affected.
(ii) the N flag is set equal to bit 7 of the result (bit 6 of input).
(iii) the Z flag is set if result equal zero, otherwise reset.
(iv) the carry flag is set equal to bit 7 of the input.

ROL Rotate Left
This instruction shifts either the accumulator or a specified memory location 1 bit to the left. Input carry is transferred into bit 0, bit 7 is transferred into the carry. As well as affecting the carry flag, this instruction sets N equal to input bit 6, sets Z flag if result is zero.

ROT Rotate Right
As for ROL, but shift is to the right, with carry to result bit 7, bit 0 into carry.

INC Increment Memory by One
This instruction adds 1 to the addressed memory location. Symbolic notation M +1 →M.

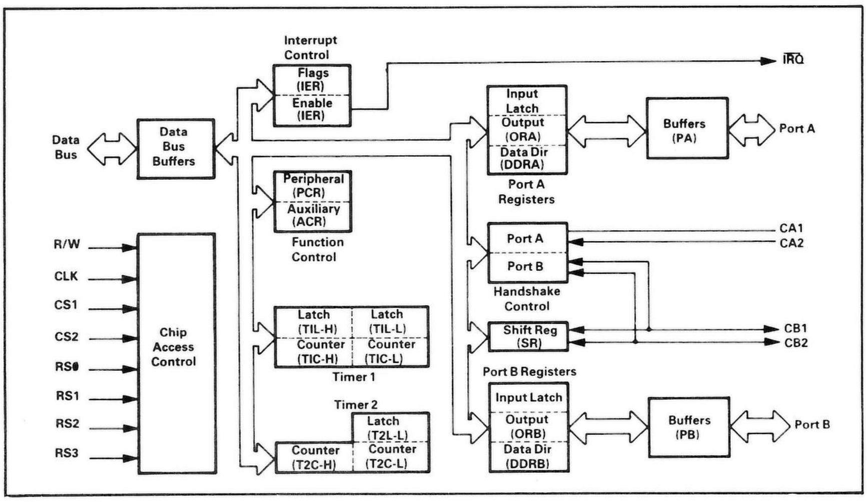
DEC Decrement Memory by One
This instruction subtracts 1, in 2's complement from, from the contents of the addessed memory location.

Chapter 22 6522, Versatile Interface Adaptor (VIA)

This device has been explained to some extent already in the user manual. This section endeavours to look at it in more detail.

This device contains two 8-bit buffer registers which access port A and port B on the 'EMMA' board. In addition, the 6522 provides two powerful interval timers a serial to parallel/parallel to serial shift register and data latching on the peripheral ports.

A complete block diagram of the 6522 is given below:



The register select lines, RS0, RS1, RS2, RS3, are connected to the lower 4-bits of the address bus. The code selected for these determine the 6522 register with which the microprocessor will communicate. The decoding for the chip select signal, CS1, CS2, place the 6522 on page 09 of the 'EMMA' memory map.

PERIPHERAL PORTS, (PA_0-PA_7), (PB_0-PB_7)
The peripheral ports each comprise eight lines which can be individually programmed to act as an input or an output under control of a data direction register. With '0' loaded into a bit of the data direction register, the corresponding bit of the port will act as an input. A '1' causes the bit of the port to act as an output.

Address	Register Function
0900	Data Register, B
0901	Data Register, A
0902	Data Direction Register, B.
0903	Data Direction Register, A.

Each peripheral pin is also controlled by a bit in the Output Register (ORA, ORB) and an Input Register (IRA, IRB). When the pin is programmed as an output, the voltage on the pin is controlled by the corresponding bit of the Output Register. A '1' in the Output Register will cause the output to go high, and a '0' causes the output to go low; this my be written into the Output Register bits corresponding to pins which are programmed as inputs. In this case, however, the output signal is unaffected.

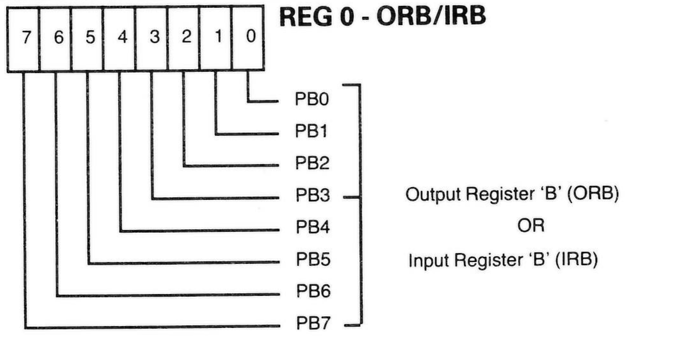
Reading a peripheral port causes the contents of the Input Register (IRA, IRB) to be transferred onto the Data Bus. With input latching disabled, IRA will always reflect the levels on the PA pins. With input latching enabled, IRA will reflect the levels on the PA pins at the time the latching occurred (via CA1).

The IRB register operates in a similar way to the IRA register. However, for pins programmed as outputs there is a difference. When reading IRA, the level on the pin determines whether a 0 or a 1 is sensed. When reading IRB, however, the bit stored in the output register, ORB, is the bit sensed.

Thus, for outputs which have large loading effects and which pull an output "1" down or which pull an output "0" up, reading IRA may result in reading a "0" when a "1" was actually programmed, and reading a "1" when a "0" was programmed. Reading IRB, on the other hand, will read the "1" or "0" level actually programmed, no matter what the loading on the pin. The following diagrams illustrate the formats of the port registers.

The input latching modes are selected by programming of the Auxiliary Control Register.

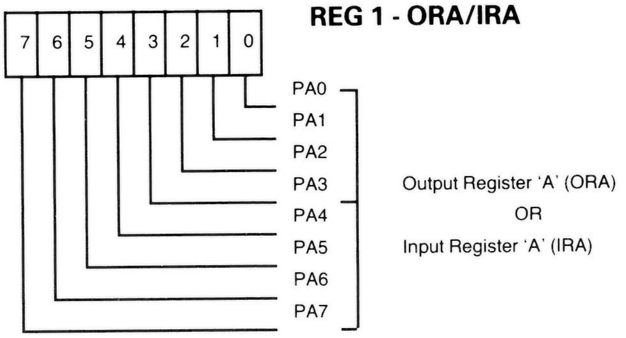
Output Register B (ORB)
Input Register B (IRB)



Pin Data Direction Selection	Write	Read
DDRB = '1' (Output)	MPU Writes Output Level (ORB)	MPU Reads Output Register Bit in ORB pin. Level has no affect
DDRB = '0' (Input) (Input latching disabled)	MPU Writes into ORB but No effect on pin level until DDRB changed	MPU reads input level on PB pin
DDRB = '0' (Input) (Input latching enabled)		MPU reads IRB bit which is the level of the PB pin at the time of the last CB1 active transition

Output Register A (ORA)
Input Register A (IRA)

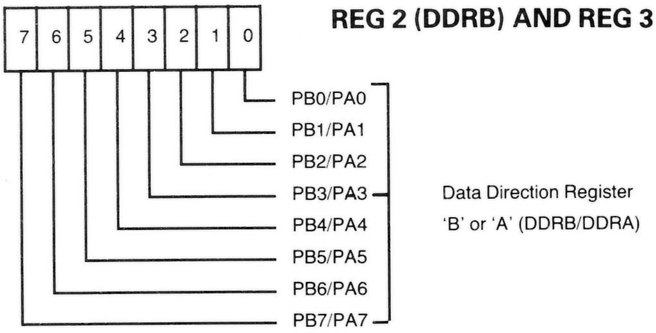
REG 1 - ORA/IRA



Pin Data Direction Selection	Write	Read
DDRA = '1' (Output) (Input Latching Disabled)	MPU Writes Output Level (ORA)	MPU Reads Level on PA Pin
DDRA = '1' (Output) (Input Latching Enabled)		MPU reads IRA bit which is the Level of the PA Pin at the time of the last CA1 active transition
DDRA = '0' (Input) (Input Latching Disabled)	MPU Writes into ORA but No effect on pin level until DDRA changed	MPU reads level on PA Pin
DDRA = '0' (Input) (Input latching enabled)		MPU reads IRA bit which is the level of the PA pin at the time of the last CA1 active transition

Data Direction Registers (DDRB, DDRA)

REG 2 (DDRB) AND REG 3 (DDRA)



'0' Associated PB pin is an input (high impedance)

'1' Associated PB pin is an output whose level is determined by ORB register bit

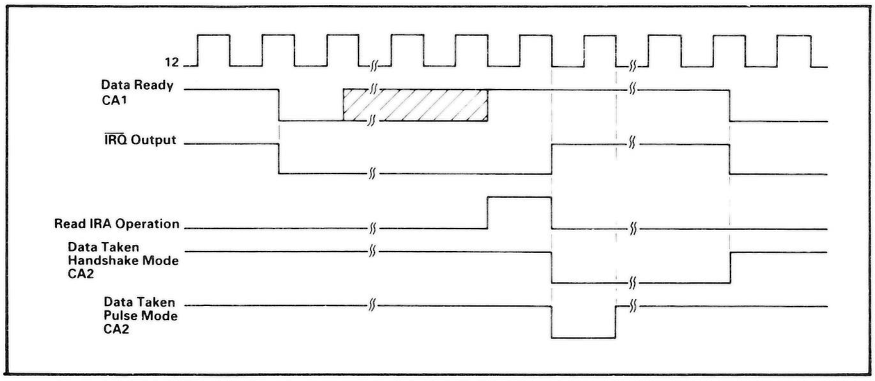
Handshake Control of Data Transfer

The 6522 allows positive control of data transfers between the system processor and peripheral devices through the operation of "handshake" lines. Port A lines (CA1, CA2) handshake data on both a read and a write operation while the Port B lines (CB1, CB2) handshake on a write operation only.

Read Handshakes

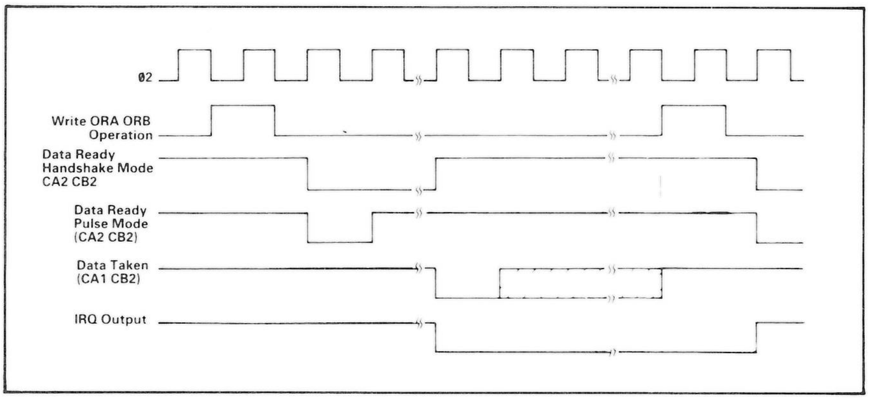
Positive control of data transfers from peripheral devices into the system processor can be accomplished very effectively using Read Handshaking. In this case, the peripheral device must generate the equivalent of a "Data Ready" signal to the peripheral port. This signal normally interrupts the processor, which then reads the data, causing generation of a "Data Taken" signal. The peripheral device responds by making new data available. This process continues until the data transfer is complete.

In the 6522, automatic "Read" Handshaking is possible on the Peripheral A port only. The CA1 interrupt input pin accepts the "Data Ready" signal will set an internal flag which may interrupt the processor or which may be polled under program control. The "Data Taken" signal can either be a pulse or a level which is set low by the system processor and is cleared by the "Data Ready" signal. These options are shown below which illustrates the normal Read Handshaking sequence.



Write Handshake

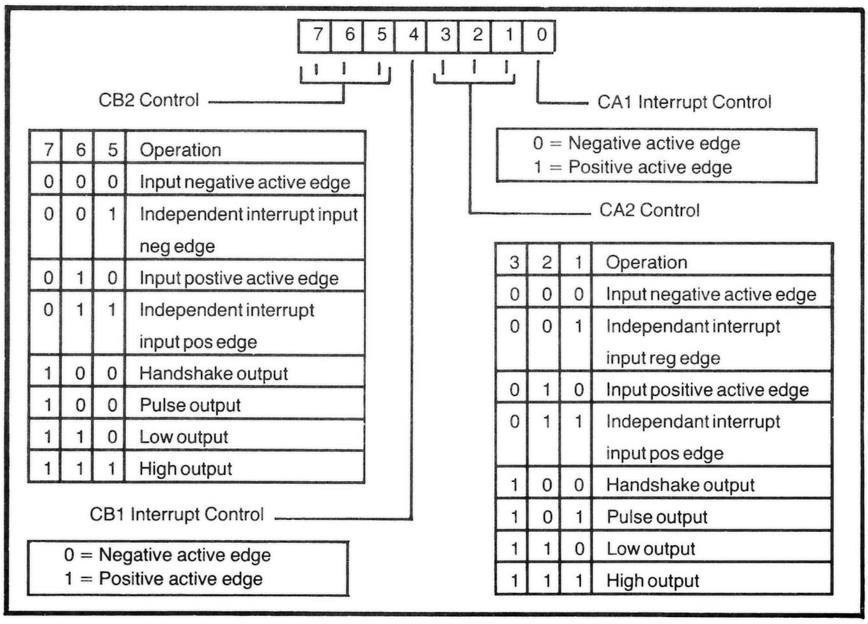
The sequence of operation which allows handshaking data from the system processor to a peripheral device is very similar to that described for READ Handshaking. However, for Write Handshaking, the 6522 generates the "Data Ready" signal and the peripheral device must respond with the "Data Taken" signal. This can be accomplished on both the PA port and the PB port on the 6522. CA2 or CB2 act as a "Data Ready" output in either the handshake mode or pulse mode and CA1 or CB1 accept the "Data Taken" signal from the peripheral device, setting the interrupt flag and clearing the "Data Ready" output. This sequence is shown in the timing diagram below.


ø2
Write ORA ORB Operation
Data Ready Handshake Mode CA2 CB2
Data Ready Pulse Mode (CA2 CB2)
Data Taken (CA1 CB2)
IRQ Output

Selection of operating modes for CA1, CA2, CB1 and CB2 is accomplished by the Peripheral Control Register (PCR).

The mode of operation of the "handshake" control lines is determined by the programming of the Peripheral Control Register. Detail of this control register is given below.

Peripheral Control Register (PCR)



CB2 Control

7	6	5	Operation
0	0	0	Input negative active edge
0	0	1	Independent interrupt input neg edge
0	1	0	Input postive active edge
0	1	1	Independent interrupt input pos edge
1	0	0	Handshake output
1	0	0	Pulse output
1	1	0	Low output
1	1	1	High output

CB1 Interrupt Control

0 = Negative active edge
1 = Positive active edge

CA1 Interrupt Control

0 = Negative active edge
1 = Positive active edge

CA2 Control

3	2	1	Operation
0	0	0	Input negative active edge
0	0	1	Independant interrupt input reg edge
0	1	0	Input positive active edge
0	1	1	Independant interrupt input pos edge
1	0	0	Handshake output
1	0	1	Pulse output
1	1	0	Low output
1	1	1	High output

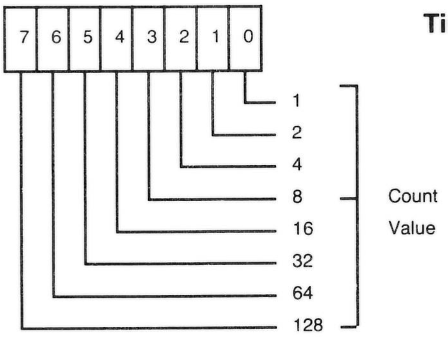
Timer Operation

Interval Timer T1 consists of two 8-bit latches and a 16-bit counter. The latches are used to store data which is to be loaded into the counter. After loading, the counter decrements at Ø2 clock rate. Upon reaching zero, an interrupt flag will be set, and IRQ will go low if the interrupt is enabled. The timer will then disable any further interrupts, or will continue to decrement. In addition, the timer may be programmed to invert the output signal on a peripheral pin each time it "times-out". Each of these modes is discussed separately below.

The T1 counter and latches are depicted below:

(A) T1 Counter Registers

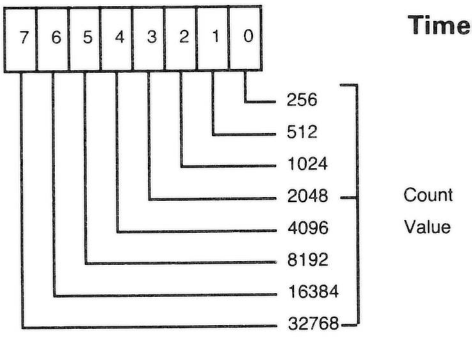
Timer 1 Low Order Counter



Write - 8 bits loaded into T1 low order latches latch contents are transferred into low order counter at the time the high order counter is loaded (Reg 5)

Read - 8 bits from T1 low order counter transferred to MPU in addition T1 interrupt flag is reset (Bit 6 in interrupt flag register).

Timer 1 High-Order Counter

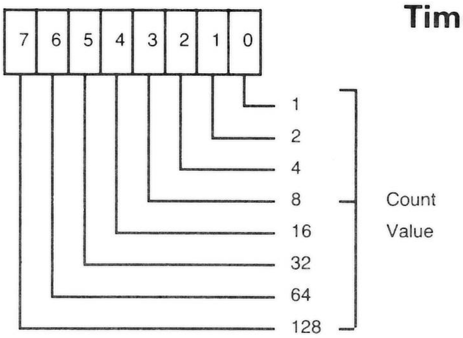


Write - 8 bits loaded into T1 high order latches. Also at this time both high and low order latches transferred into T1 counter T1 interrupt flag also is reset

Read - 8 bits from T1 high order counter transferred to MPU

(B) T1 Latch Registers

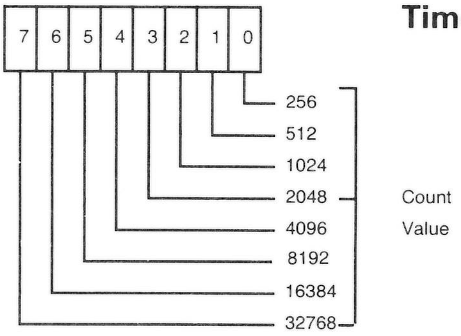
Timer 1 High-Order Latches



Write - 8 bits loaded into T1 low order latches/this operation is no different than a write into Reg 4.

Read - 8 bits from T1 low order latches transferred to MPU. Unlike Reg 4 operation this does not cause reset of T1 interupt flag.

Timer 1 Low-Order Latches

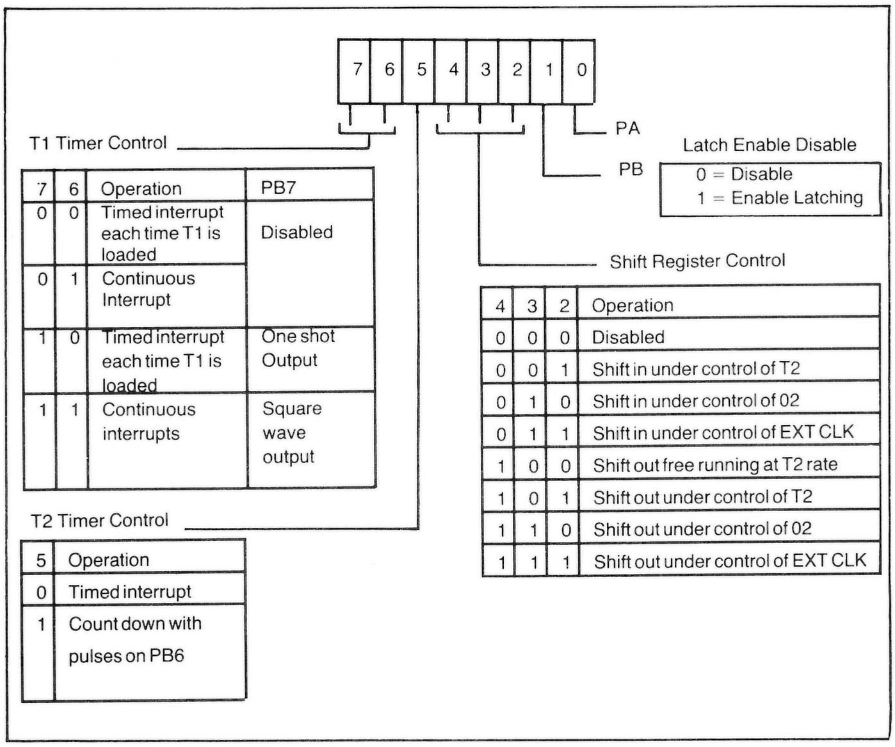


Write - 8 bits loaded into T1 high order latches unlike reg 4 operation.
No latch to counter transfers take place
Read - 8 bits from T1 high order latches transferred to MPU.

Two bits are provided in the Auxiliary Control Register (bits 6 and 7) to allow selection of the T1 operating modes. The four possible modes are depicted overleaf.

Note: The processor does not write directly into the low order counter (TIC-L). Instead, this half of the counter is loaded automatically from the low order latch when the processor writes into the high order counter. In fact, it may not be necessary to write to the low order counter in some applications since the timing operation is triggered by writing to the high order counter.

Auxiliary Control Register



T1 Timer Control

7	6	Operation	PB7
0	0	Timed interrupt each time T1 is loaded	Disabled
0	1	Continuous Interrupt	
1	0	Timed interrupt each time T1 is loaded	One shot Output
1	1	Continuous interrupts	Square wave output

T2 Timer Control

5	Operation
0	Timed interrupt
1	Count down with pulses on PB6

Latch Enable Disable

0 = Disable
1 = Enable Latching

Shift Register Control

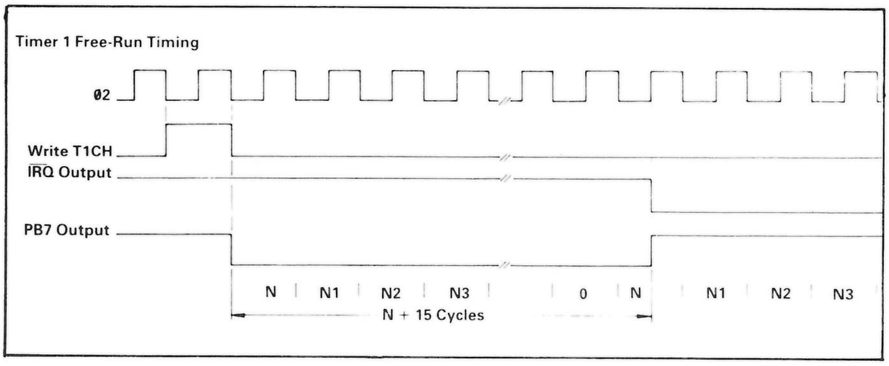
4	3	2	Operation
0	0	0	Disabled
0	0	1	Shift in under control of T2
0	1	0	Shift in under control of 02
0	1	1	Shift in under control of EXT CLK
1	0	0	Shift out free running at T2 rate
1	0	1	Shift out under control of T2
1	1	0	Shift out under control of 02
1	1	1	Shift out under control of EXT CLK

Timer 1 One-Shot Mode

The interval timer one-shot mode allows generation of a single interrupt for each timer load operation. As with any interval timer, the delay between the "write TIC-H" operation and generation of the processor interrupt is a direct function of the data loaded into the timing counter. In addition to generating a single interrupt, Timer 1 can be programmed to produce a single negative pulse on the PB7 peripheral pin. With the output enabled (ACR7=1) a "write TIC-H" operation will cause PB7 to go low. PB7 will return high when Timer 1 times out. The result is a single programmable width pulse.

In the one-shot mode, writing into the high order latch has no effect on the operation of Timer 1. However, it will be necessary to ensure that the low latch contains the proper data before initiating the count-down with a "write TIC-H" operation. When the processor writes into the high order counter, the T1 interrupt flag will be cleared, the contents of the low order latch will be transferred into the low order counter, and the timer will begin to decrement at system clock rate. If the PB7 output is enabled, this signal will go low on the phase two following the write operation. When the counter reaches zero, the T1 interrupt flag will be set, the IRQ pin will go low (interrupt enabled), and the signal on PB7 will go high. At this time the counter will continue to decrement at system clock rate. This allows the system processor to read the contents of the counter to determine the time since interrupt. However, the T1 interrupt flag cannot be set again unless it has been cleared as described in this section.

Timing for the 6522 interval timer one-shot modes is shown below:

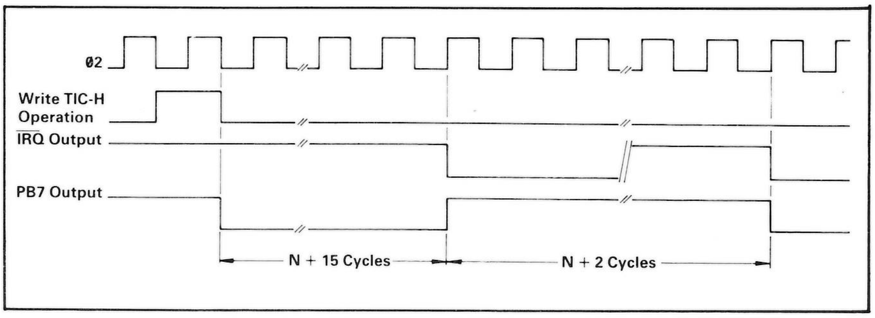


Timer 1 Free-Run Mode

The most important advantage associated with the latches in T1 is the ability to produce a continuous series of evenly spaced interrupts unaffected by variations in the processor interrupt response time. This is accomplished in the "free-running" mode.

In the free-running mode, the interrupt flag is set and the signal on PB7 is inverted each time the counter reaches zero. However, instead of continuing to decrement from zero after a time-out, the timer automatically transfers the contents of the latch into the counter (16 bits) and continues to decrement from there. The interrupt flag can be cleared by writing TIC-H, by reading TIC-L, or by writing directly into the flag as described later. However, it is not necessary to rewrite the timer to enable setting the interrupt flag on the next time-out.

All interval timers in the 6522 are "re-triggerable". Rewriting the counter will always re-initialize the time-out period. In fact, the time-out can be prevented completely if the processor continues to rewrite the timer before it reaches zero. Timer 1 will operate in this manner if the processor writes into the high order counter (TIC-H). However, by loading the latches only, the processor can access the timer during each down counting operation without affecting the time-out in process. Instead, the data loaded into the latches will determine the length of the next time-out period. This capability is particularly valuable in the free running mode with the output enabled. In this mode, the signal on PB7 is inverted and the interrupt flag is set with each timeout. By responding to the interrupts with new data for the latches, the processor can determine the period of the next half cycle during each half cycle of the output signal on PB7. In this manner, very complex waveforms can be generated. Timing for the free-running mode is shown:

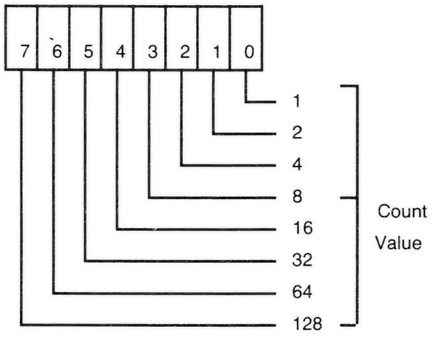


Note: A precaution to take in the use of PB7 as the timer output concerns the Data Direction Register contents for PB7. Both DDRB bit 7 and ACR bit 7 must be 1 for PB7 to function as the timer output. If one is 1 and the other is 0, then PB7 functions as a normal output pin controlled by ORB bit 7.

Timer 2 Operation

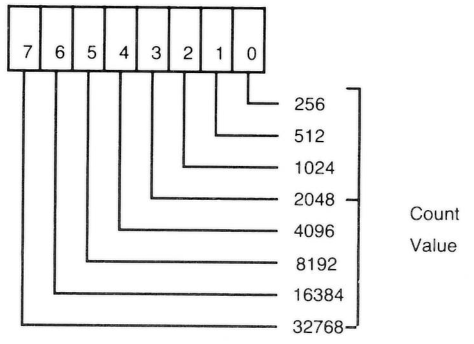
Timer 2 operates as an interval timer (in the "one-shot" mode only), or as a counter for counting negative pulses on the PB6 peripheral pin. A simple control bit is provided in the Auxiliary Control Register to select between these two modes. This timer consists of a 'write-only' low order latch (T2L-L), a 'read only' low-order counter and a read/write high order counter. The counter registers act as a 16-bit counter which decrements at 02 rate. The diagram below illustrates T2 counter Registers.

Timer 2 Low-Order Latch/Counter



Write - 8 Bits loaded into T2 low order latch
Read - 8 Bits from T2 low order counter transferred to MPU T2 interrupt flag is reset

Timer 2 High-Order Latch/Counter



Write - 8 Bits loaded into T2 high order counter also low order latch transferred to low-order counter in addition. T2 interrupt flag is reset

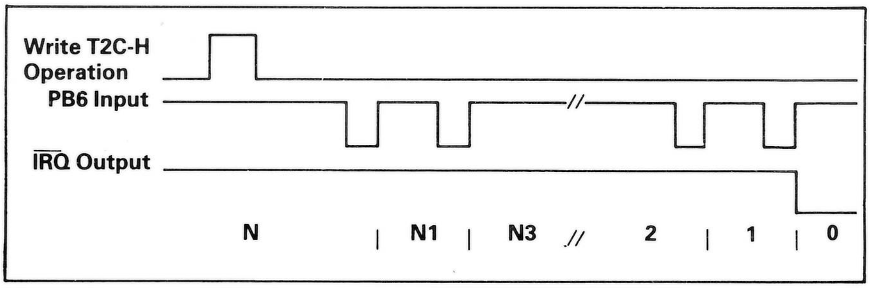
Read - 8 Bits from T2 high order counter transferred to MPU

Timer 2 One-Shot Mode

As an interval timer, T2 operates in the "oneshot" mode similar to Timer 1. In this mode, T2 provides a single interrupt for each "write T2C-H" operation, After timing out, the counter will continue to decrement. However, setting of the interrupt flag will be disabled after initial time-out so that it will not be set by the counter continuing to decrement through zero. The processor must rewrite T2C-H to enable setting of the interrupt flag. The interrupt flag is cleared by reading T2C-L or by writing T2C-H.

Timer 2 Pulse Counting Mode

In the pulse counting mode, T2 serves primarily to count a predetermined number of negative-going pulses on PB6. This is accomplished by first loading a number into T2. Writing into T2C-H clears the interrupt flag and allows the counter to decrement each time a pulse is applied to PB6. The interrupt flag will be set when T2 reaches zero. At this time the counter will continue to decrement with each pulse on PB6. However, it is necessary to rewrite T2C-H to allow the interrupt flag to set on subsequent down-counting operations. Timing for this mode is shown below. The pulse must be low on the leading edge of Ø2.



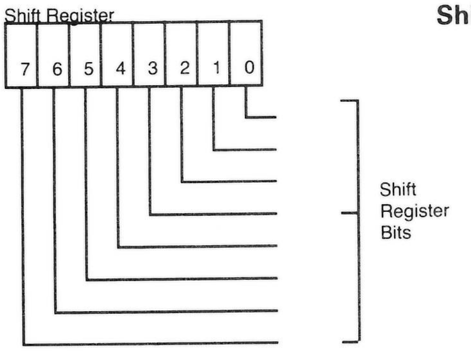
Shift Register Operation

The Shift Register (SR) performs serial data transfers into and out of the CB2 pin under control of an internal modulo-8 counter. Shift pulses can be applied to the CB1 pin from an external source or, with the proper mode selection, shift pulses generated internally will appear on the CB1 pin for controlling external devices.

The control bits which select the various shift register operating modes are located in the Auxiliary Control Register. Figure below illustrates the configuration of the SR data bits and the SR control bits of the ACR.

SR and ACR Control Bits

Shift Register

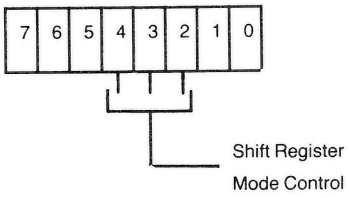


Notes:

1 When shifting out bit 7 is the first bit out and simultaneously is rotated back into bit 0

2 When shifting in bit 5 initially enter bit 0 and are shifted towards bit 7

Auxiliary Control Register



4	3	2	Operation
0	0	0	Disabled
0	0	1	Shift in under control of T2
0	1	0	Shift in under control of 02
0	1	1	Shift in under control of EXT CLK
1	0	0	Shift out free running at T2 rate
1	0	1	Shift out under control of T2
1	1	0	Shift out under control of 02
1	1	1	Shift out under control of EXT CLK

Operation of the Various Shift Register Modes

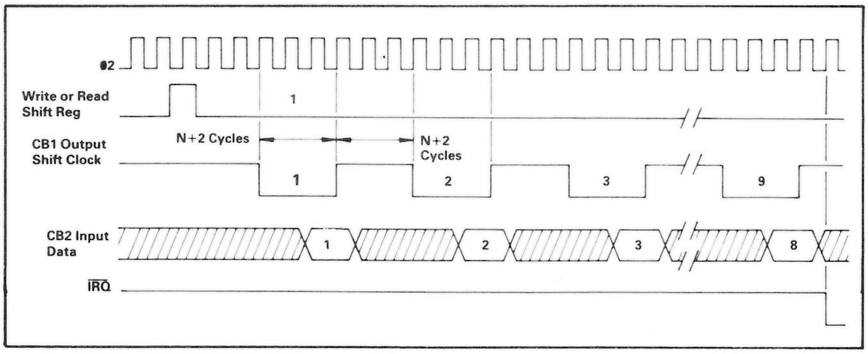
SR Disabled (000)

The 000 mode is used to disable the Shift Register. In this mode the microprocessor can write or read the SR but the shifting operation is disabled and operation of CB1 and CB2 is controlled by the appropriate bits in the Peripheral Control Register (PCR). In this mode the SR Interrupt Flag is disabled (held to logic 0).

Shift in Under Control of T2 (001)

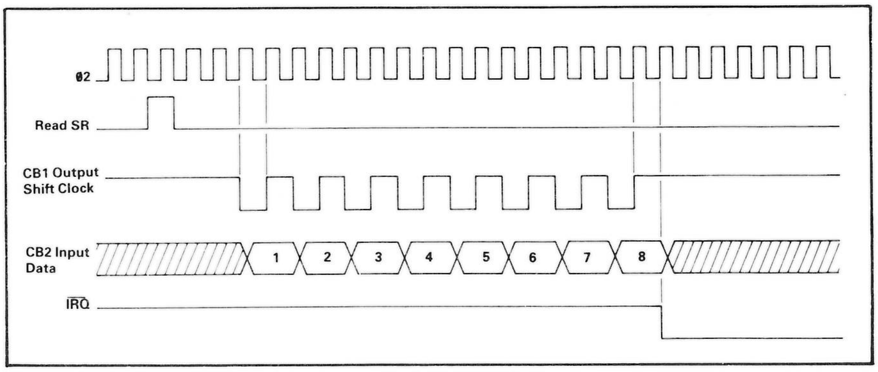
In the 001 mode the shifting rate is controlled by the low order 8 bits of T2. Shift pulses are generated on the CB1 pin to control shifting in from external devices. The time between transitions of the output clock is a function of the system clock period and the contents of the low order T2 latch (N).

The shifting operation is triggered by writing or reading the shift register. Data is shifted first into the low order bit of SR and is then shifted into the next higher order bit of the shift register on the negative edge of each clock pulse. The input data should change before the positive going edge of the CB1 clock pulse. This data is shifted into the shift register during the Ø2 clock cycle following the positive going edge of the CB1 clock pulse. After 8 CB1 clock pulses the shift register interrupt flag will be set and $\overline{IRQ}$ will go low.



Shift in Under Control of Ø2 (010)

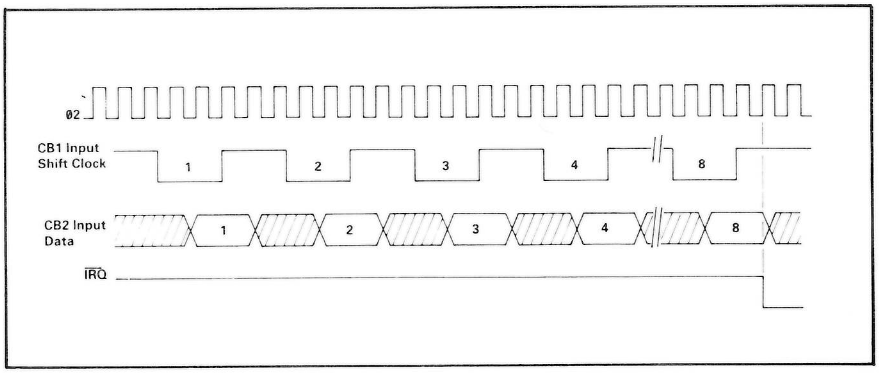
In mode 010, the shift rate is a direct function of the system clock frequency. CB1 becomes an output which generates shift pulses for controlling external devices. Timer 2 operates an independant interval timer and has no effect on SR. The shifting operation is triggered by reading or writing the Shift Register. Data is shifted first into bit 0 and is then shifted into the next higher order bit of the shift register on the trailing edge of each Ø2 clock pulse. After 8 clock pulses, the shift register interrupt flag will be set, and the output clock pulses on CB1 will stop.



Shift In Under Control of External CB1 Clock (011)

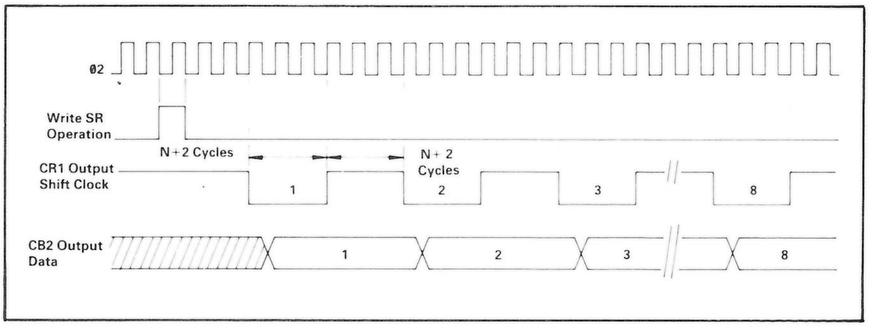
In mode 011 CB1 becomes an input. This allows an external device to load the shift register at its own pace. The shift register counter will interrupt the processor each time 8 bits have been shifted in. However, the shift register counter does not stop the shifting operation it acts simply as a pulse counter. Reading or writing the Shift Register resets the interrupt flag and initialise the SR counter to count another 8 pulses.

Note that the data is shifted during the first system clock cycle following the positive-going edge of the CB1 shift pulse. For this reason, data must be held stable during the first full cycle following CB1 going high.



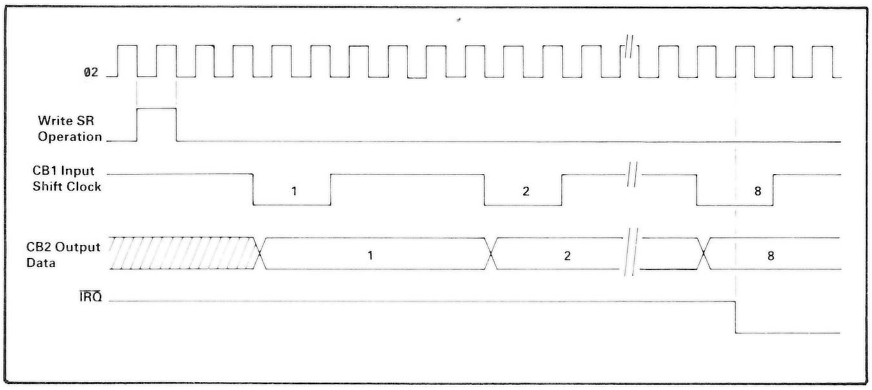
Shift Out Free Running At T2 Rate (100)

Mode 100 is very similar to mode 101 in which the shifting rate is set by T2. However, in mode 100 the SR counter does not stop the shifting operation. Since the Shift Register bit 7 (SR7) is recirculated back into bit 0, the 8 bits loaded into the shift register will be clocked onto CB2 repetitively. In this mode the shift register counter is disabled.



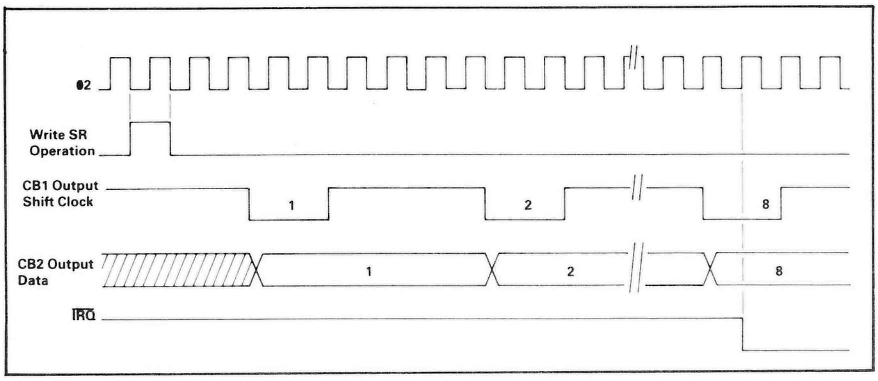
Shift Out Under Control of T2 (101)

In mode 101 the shift rate is controlled by T2 (as in the previous mode). However, with each read and write of the shift register the SR Counter is reset and 8 bits are shifted onto CB2. At the same time, 8 shift pulses are generated on CB1 to control shifting in external devices. After the 8 shift pulses, the shifting is disabled, the SR Interrupt Flag is set and CB2 remains at the last data level.



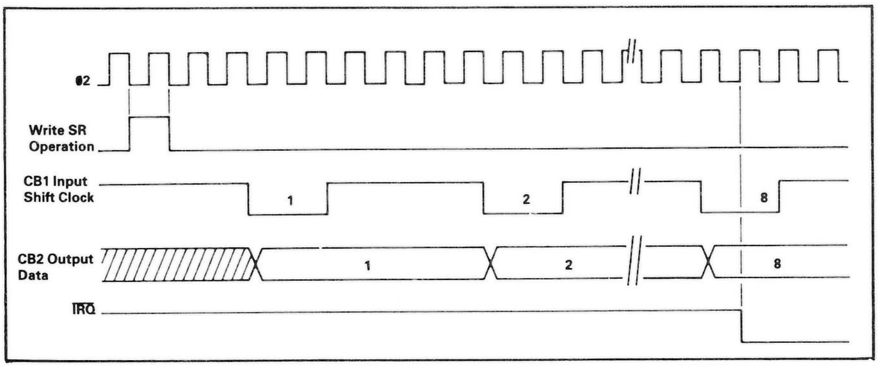
Shift Out Under Control of T2 (110)

In mode 110, the shift rate is controlled by the Ø2 system clock.



Shift Out Under Control Of External CB1 Clock (111)

In mode 111 shifting is controlled by a pulse applied to the CB1 pin by an external device. The SR counter sets the SR Interrupt flag each time it counts 8 pulses but it does not disable the shifting function. Each time the microprocessor writes or reads the shift register, the SR Interrupt flag is reset and the SR counter is initialised to begin counting the next 8 shift pulses on pin CB1. After 8 shift pulses, the interrupt flag is set. The microprocessor can then load the shift register with the next byte of data.



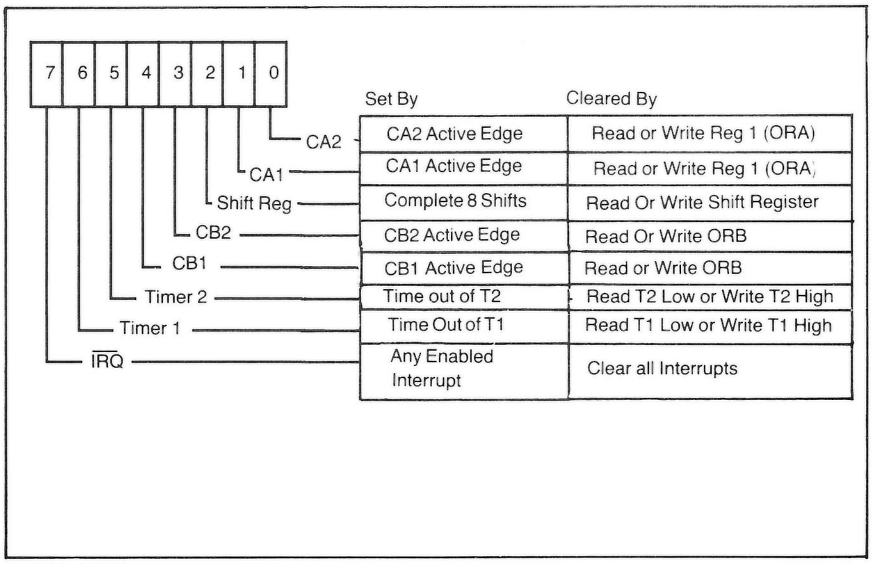
Interrupt Operation

Controlling interrupts within the 6522 ınvolves three principal operations. These are flagging the interrupts, enabling interrupts and signalling to the processor that an active interrupt exists within the chip. Interrupt flags are set by an interrupting condition which exists within the chip or on inputs to the chip. These flags normally remain set until the interrupt has been serviced. To determine the source of an interrupt, the microprocessor must examine these flags in order from highest to lowest priority. This is accomplished by reading the flag register into the processor accumulator, shifting this register either right or left and then using conditional branch instructions to detect an active interrupt.

Associated with each interrupt flag is an interrupt enable bit. This can be set or cleared by the processor to enable interruption of the processor from the corresponding interrupt flag. If an interrupt flag is set to a logic 1 by an interrupting condition, and the corresponding interrupt enable bit is set to a 1, the Interrupt Request Output ($\overline{IRQ}$) will go low. $\overline{IRQ}$ is an "open-collector" output which can be 'wired-or-ed' with other devices in the system to interrupt the processor.

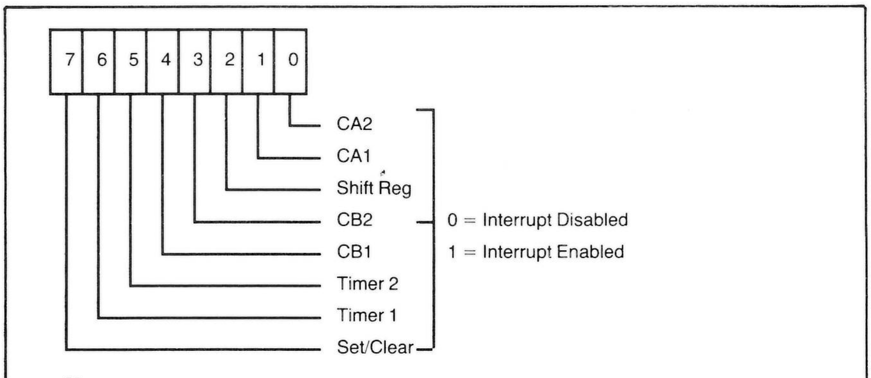
Within the 6522, all the interrupt flags are contained in the Interrupt Flag Register (IFR). In addition, bit 7 of this register will be read as a logic 1 when an interrupt exists within the chip. This allows very convenient polling of several devices within a system to locate the source of an interrupt. The Interrupt Flag Register (IFR) and Interrupt Enable Register (IER) are depicted below:

Interrupt Flag Register (IFR)



	Set By	Cleared By
CA2	CA2 Active Edge	Read or Write Reg 1 (ORA)
CA1	CA1 Active Edge	Read or Write Reg 1 (ORA)
Shift Reg	Complete 8 Shifts	Read Or Write Shift Register
CB2	CB2 Active Edge	Read Or Write ORB
CB1	CB1 Active Edge	Read or Write ORB
Timer 2	Time out of T2	Read T2 Low or Write T2 High
Timer 1	Time Out of T1	Read T1 Low or Write T1 High
$\overline{IRQ}$	Any Enabled Interrupt	Clear all Interrupts

Interrupt Enable Register (IER)



Notes

1 If bit 7 is a '0' then each '1' in Bits 0-6 disables the corresponding interrupt

2 If bit 7 is a '1' then each '1' in Bits 0-6 enables the corresponding interrupt

3 If a read of this register id done, Bit 7 will be '0' and all other bits will reflect their enable/disable state

The IFR may be read directly by the processor. In addition, individual flag bits may be cleared by writing a "1" into the appropriate bit of the IFR. When the proper chip select and register signals are applied to the chip, the contents of this register are placed on the data bus. Bit 7 indicates the status of the IRQ output. This bit corresponds to the logic function: $\mathrm{IRQ} = \mathrm{IFR6} \times \mathrm{IER6} + \mathrm{IFR5} \times \mathrm{IER5} + \mathrm{IFR4} \times \mathrm{IER4} \times \mathrm{IFR3} \times \mathrm{IER3} + \mathrm{IFR2} \times \mathrm{IER2} + \mathrm{IFR1} \times \mathrm{IER1} + \mathrm{IFR0} \times \mathrm{IER0}$.
Note: X = logic AND, + = Logic OR.

The IFR bit 7 is not a flag. Therefore, this bit is not directly cleared by writing a logic 1 into it. It can only be cleared by clearing all the flags in the register or by disabling all the active interrupts.

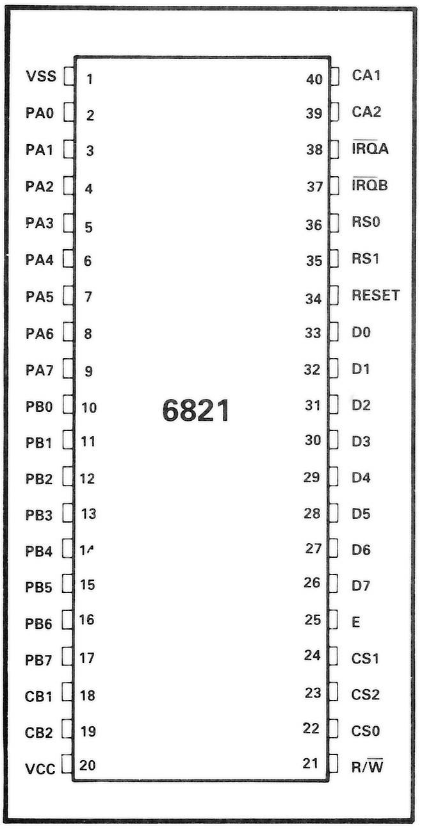
For each interrupt flag in IFR, there is a corresponding bit in the Interrupt Enable Register. The system processor can set or clear interrupts without affecting others. This is accomplished by writing to address 1110 (IER address). If bit 7 of the data placed on the system data bus during this write operation is a 0, each 1 in bits 6 through 0 clears the corresponding bit in the Interrupt Enable Register. For each zero in bits 6 through 0, the corresponding bit is unaffected.

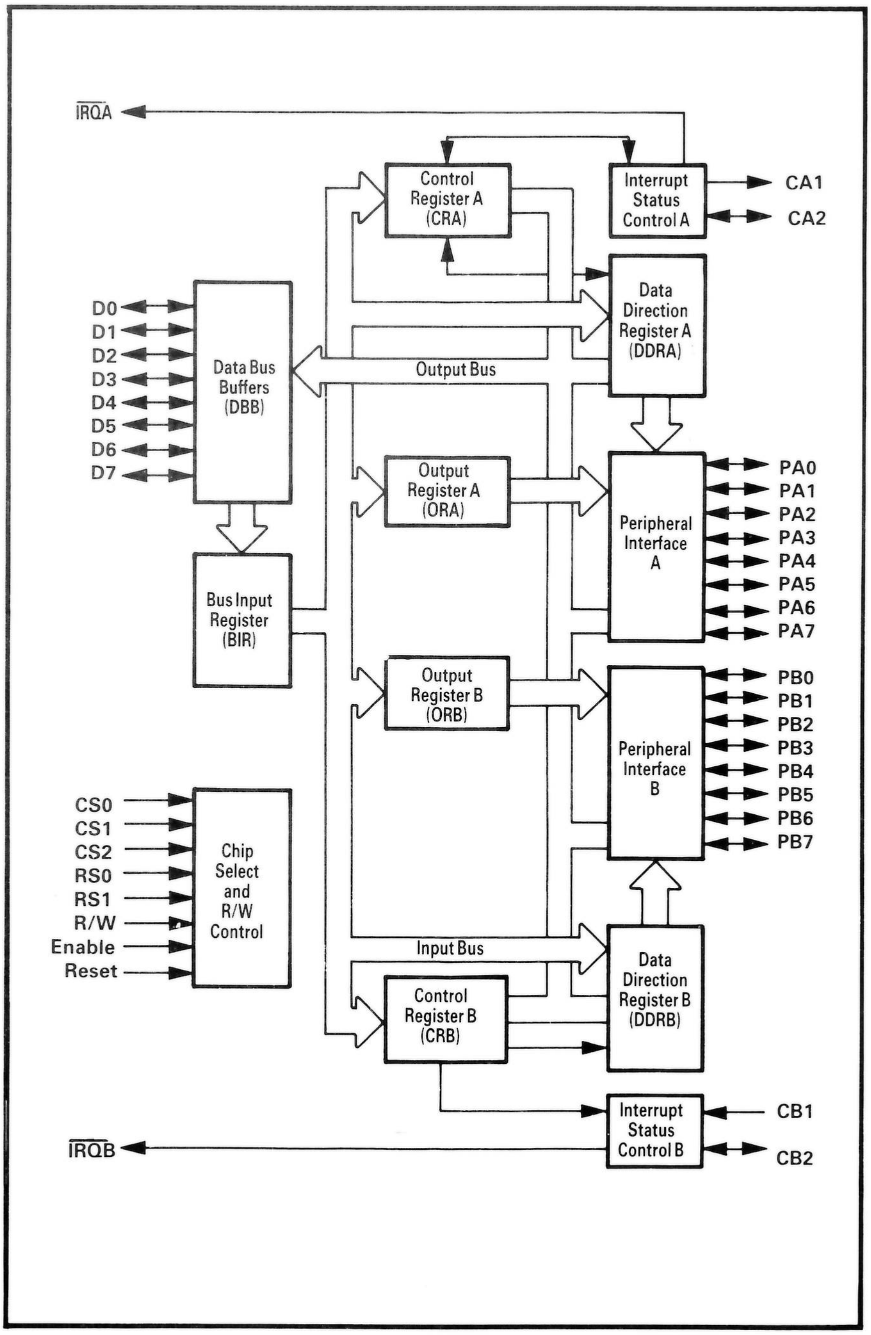
Setting selected bits in the Interrupt Enable Register is accomplished by writing to the same address with bit 7 in the data word set to a logic 1. In this case, each 1 in bits 6 through 0 will set the corresponding bit. For each zero, the corresponding bit will be unaffected. This individual control of the setting and clearing operations allows very convenient control of the interrupts during system operation.

In addition to setting and clearing IER bits, the processor can read the contents of this register by placing the proper address on the register select and chip select inputs with the R/W line high. Bit 7 will be read as a logic 0.

Chapter 23 6821 Keyboard and Display Interface

The functional configuration of the 6821 is programmed by the MPU during system initialisation. The 6821 is dedicated on the **EMMA II** board for the function of keyboard and display interfacing. It is similar to the 6522 in some functions, in that it contains two 8-bit Bidirectional I/O Ports. The ports are set up to function as explained in the hardware section. Internal block diagram of the 6821 is shown below.





As seen on the block diagram the 6821 contains:

- Peripheral register that loads either input or output data. When used for output this register is latched, and when used for input is unlatched.
- Two data direction registers (DDRA, DDRB); the bits in these registers determine whether the corresponding data register bits are inputs (0) or outputs (1).
- Control registers that hold the status signals required for handshaking, and other bits that select operating conditions within the 6821.
- Two control lines that are configured by the control registers.

The 6821 occupies four memory addresses as selected by **EMMA II** address decoding and the register select lines. Since there are six registers (2 peripheral, 2 data direction, 2 control) in the 6821 the Data Direction registers and Peripheral registers for each port share an address. Thus a further bit is required to address these individually. When a port is addressed bit 2 of the associated control register determines whether the Data Direction register or the Peripheral register is accessed.

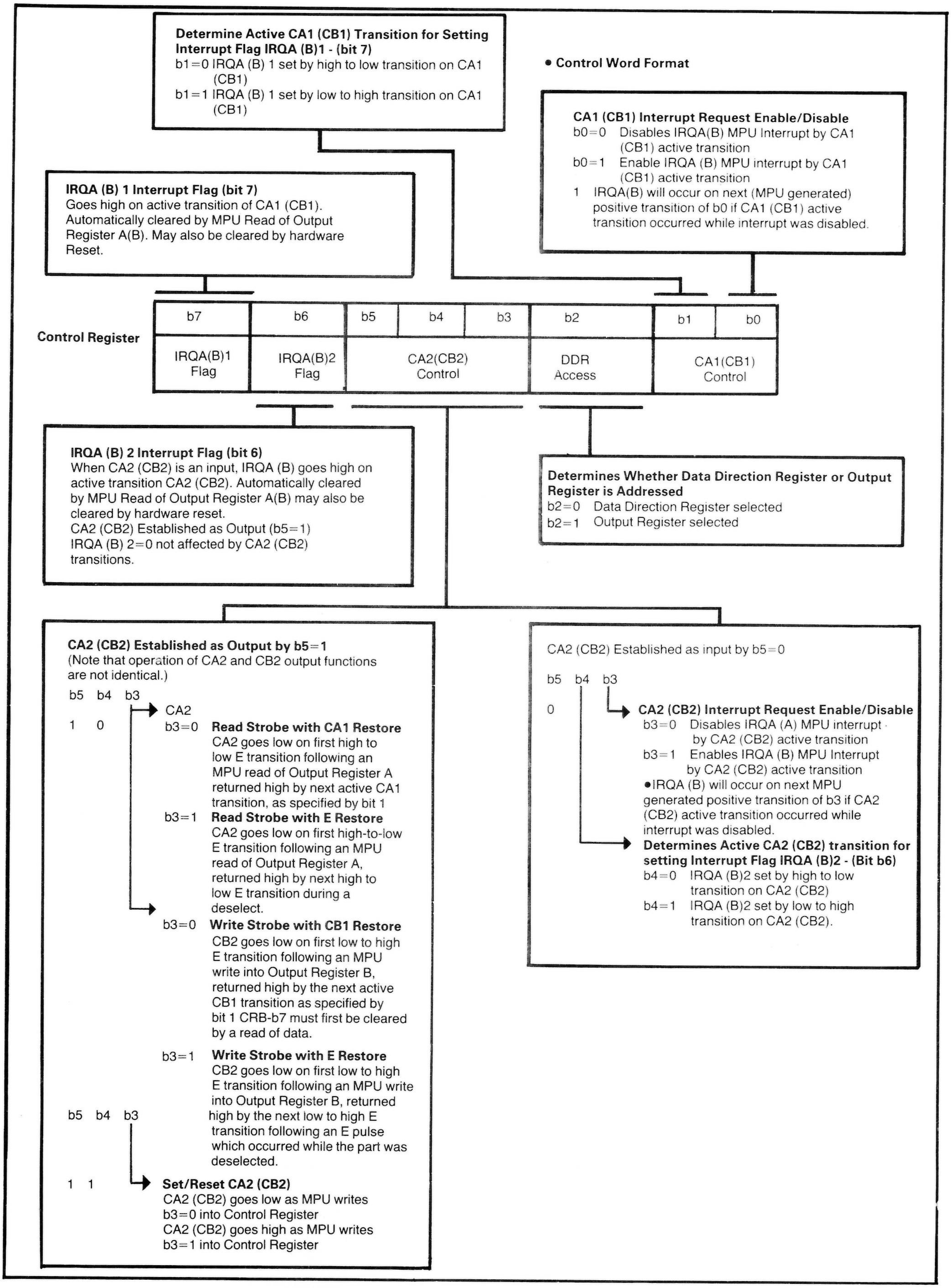
The sharing of an external address means:

- A program must change the bit in the conti ol register in order to use the register that is not currently being addressed.
- The programmer must be aware of the contents of the control register in order to know which register is being addressed.
- RESET clears the control register and thus addresses the data direction register.

Addressing The 6821 Internal Registers

	Control Register Bit		
Address Lines	CRA 2	CRB 2	
0 A 0 0	1	X	Peripheral Register A
0 A 0 0	0	X	Data Direction Register
0 A 0 1	X	X	Control Register A
0 A 0 2	X	1	Peripheral Register B
0 A 0 2	X	0	Data Direction Register B
0 A 0 3	X	X	Control Register B
X = Either 0 or 1			

The table shown gives the addresses of the various registers in the 6821. Control register format is shown opposite on page 82.



• **Control Word Format**

Determine Active CA1 (CB1) Transition for Setting Interrupt Flag IRQA (B)1 - (bit 7)
b1=0 IRQA (B) 1 set by high to low transition on CA1 (CB1)
b1=1 IRQA (B) 1 set by low to high transition on CA1 (CB1)

CA1 (CB1) Interrupt Request Enable/Disable
b0=0 Disables IRQA(B) MPU Interrupt by CA1 (CB1) active transition
b0=1 Enable IRQA (B) MPU interrupt by CA1 (CB1) active transition
1 IRQA(B) will occur on next (MPU generated) positive transition of b0 if CA1 (CB1) active transition occurred while interrupt was disabled.

IRQA (B) 1 Interrupt Flag (bit 7)
Goes high on active transition of CA1 (CB1). Automatically cleared by MPU Read of Output Register A(B). May also be cleared by hardware Reset.

Control Register

b7	b6	b5	b4	b3	b2	b1	b0
IRQA(B)1 Flag	IRQA(B)2 Flag	CA2(CB2) Control			DDR Access	CA1(CB1) Control	

IRQA (B) 2 Interrupt Flag (bit 6)
When CA2 (CB2) is an input, IRQA (B) goes high on active transition CA2 (CB2). Automatically cleared by MPU Read of Output Register A(B) may also be cleared by hardware reset.
CA2 (CB2) Established as Output (b5=1)
IRQA (B) 2=0 not affected by CA2 (CB2) transitions.

Determines Whether Data Direction Register or Output Register is Addressed
b2=0 Data Direction Register selected
b2=1 Output Register selected

CA2 (CB2) Established as Output by b5=1
(Note that operation of CA2 and CB2 output functions are not identical.)

b5 b4 b3

1 0 → CA2

b3=0 **Read Strobe with CA1 Restore**
CA2 goes low on first high to low E transition following an MPU read of Output Register A returned high by next active CA1 transition, as specified by bit 1

b3=1 **Read Strobe with E Restore**
CA2 goes low on first high-to-low E transition following an MPU read of Output Register A, returned high by next high to low E transition during a deselect.

b3=0 **Write Strobe with CB1 Restore**
CB2 goes low on first low to high E transition following an MPU write into Output Register B, returned high by the next active CB1 transition as specified by bit 1 CRB-b7 must first be cleared by a read of data.

b3=1 **Write Strobe with E Restore**
CB2 goes low on first low to high E transition following an MPU write into Output Register B, returned high by the next low to high E transition following an E pulse which occurred while the part was deselected.

b5 b4 b3

1 1 → **Set/Reset CA2 (CB2)**
CA2 (CB2) goes low as MPU writes b3=0 into Control Register
CA2 (CB2) goes high as MPU writes b3=1 into Control Register

CA2 (CB2) Established as input by b5=0

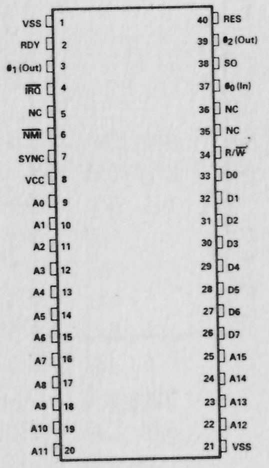
b5 b4 b3

0

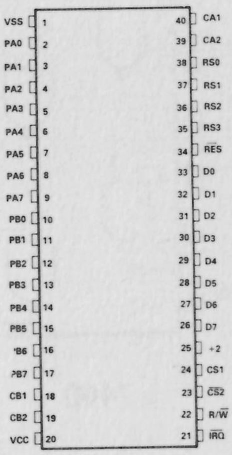
→ **CA2 (CB2) Interrupt Request Enable/Disable**
b3=0 Disables IRQA (A) MPU interrupt by CA2 (CB2) active transition
b3=1 Enables IRQA (B) MPU Interrupt by CA2 (CB2) active transition
•IRQA (B) will occur on next MPU generated positive transition of b3 if CA2 (CB2) active transition occurred while interrupt was disabled.

→ **Determines Active CA2 (CB2) transition for setting Interrupt Flag IRQA (B)2 - (Bit b6)**
b4=0 IRQA (B)2 set by high to low transition on CA2 (CB2)
b4=1 IRQA (B)2 set by low to high transition on CA2 (CB2).

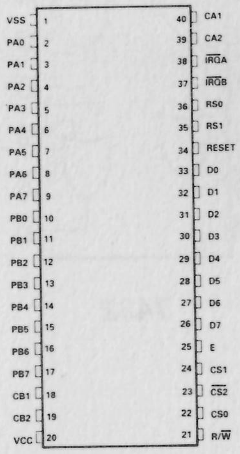
Appendix 1 Pin Allocations



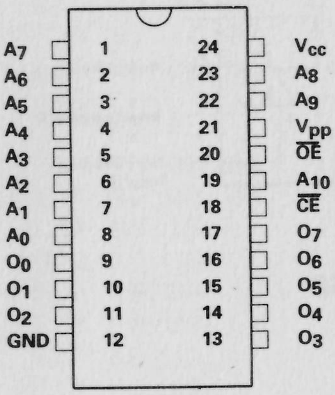
6502



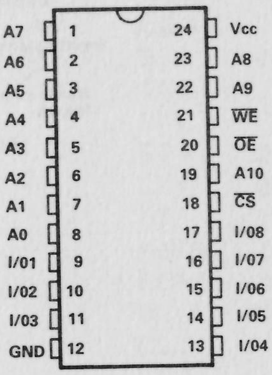
6522



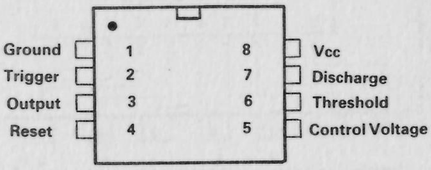
6821



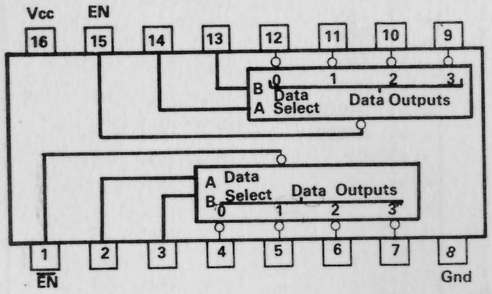
2716



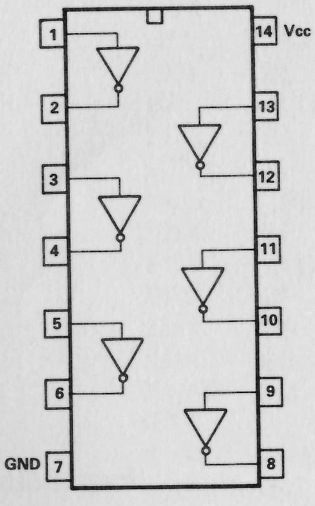
6116



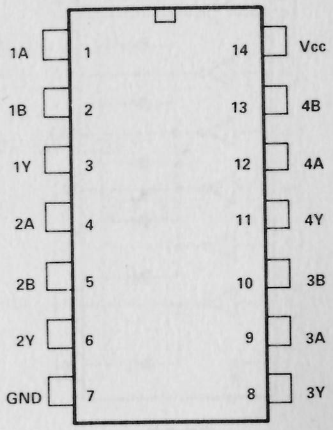
555



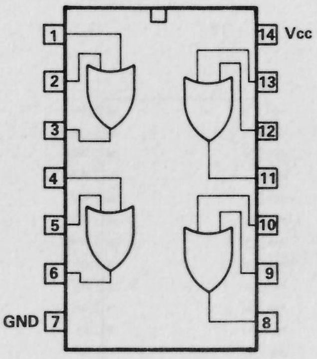
74139



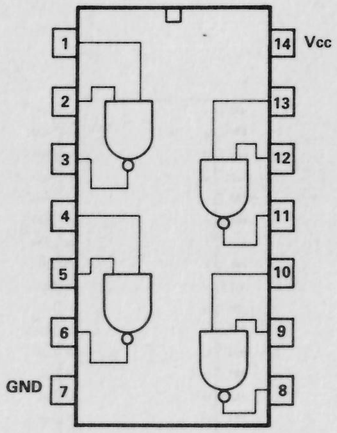
7404



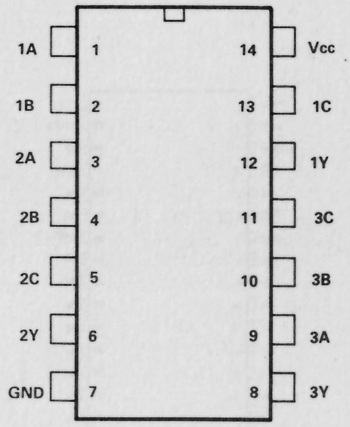
7408



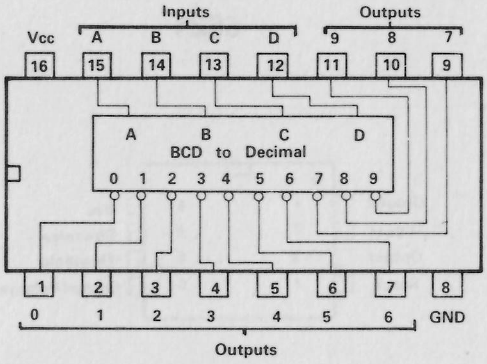
7432



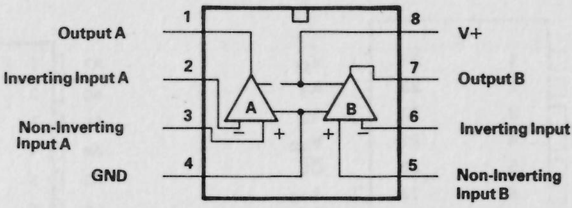
7400



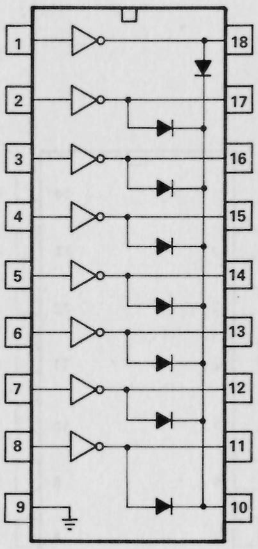
7412



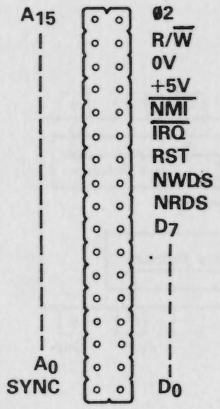
7445



358



2803



Expansion connector pin allocations

```
                                ;EMF3R5-EMMA2 MONITOR+CASSETTE ROU
TINES
                                ;+USEFUL ROUTINES.14/6/84,NC.R12/9
/84
                                ORG  FE00
 FE00 A0 06         QUAD:       LDY#  06
 FE02 B5 00                     LDA   00,X
 FE04 20 6F FE                  JSR    DHEXTD
 FE07 CA                        DEX
 FE08 88                        DEY
 FE09 88                        DEY
 FE0A 10 F6                     BPL   F6 (FE02)
 FE0C 86 1A         DISPLAY:    STX   1A
 FE0E 38                        SEC
 FE0F A2 0F         NEWSCAN:    LDX#  0F
 FE11 8A            NEXT:       TXA
 FE12 29 07                     AND#  07
 FE14 A8                        TAY
 FE15 8D 02 0A                  STA   0A02
 FE18 B9 10 00                  LDA   0010,Y
 FE1B 8D 00 0A                  STA   0A00
 FE1E C8                        INY
 FE1F D0 FA                     BNE   FA (FE1B)
 FE21 8C 00 0A                  STY   0A00
 FE24 AD 02 0A                  LDA   0A02
 FE27 85 19                     STA   19
 FE29 29 38                     AND#  38
 FE2B 49 38                     EOR#  38
 FE2D F0 09                     BEQ   09 (FE38)
 FE2F 18                        CLC
 FE30 A5 19                     LDA   19
 FE32 49 18                     EOR#  18
 FE34 29 1F                     AND#  1F
 FE36 85 0D                     STA   0D
 FE38 CA                        DEX
 FE39 10 D6                     BPL    NEXT (FE11)
 FE3B A6 0E                     LDX   0E
 FE3D 10 0A                     BPL    OUT (FE49)
 FE3F 90 04                     BCC   04 (FE45)
 FE41 84 0F                     STY   0F
 FE43 B0 CA                     BCS    NEWSCAN (FE0F)
 FE45 A6 0F                     LDX   0F
 FE47 30 C5                     BMI    NEWSCAN-01 (FE0E)
 FE49 C6 0F         OUT:        DEC   0F
 FE4B A6 1A                     LDX   1A
 FE4D A5 0D                     LDA   0D
 FE4F C9 10                     CMP#  10
 FE51 60                        RTS
 FE52 8D 02 0A      INIT2:      STA   0A02
 FE55 A9 04                     LDA#  04
 FE57 8D 01 0A                  STA   0A01
 FE5A 8D 03 0A                  STA   0A03
```

```
FE5D 60          51              RTS
FE5E A1 00       52   MHEXTD:    LDA  (00,X)
FE60 A0 06       53              LDY# 06
FE62 D0 0B       54              BNE   DHEXTD (FE6F)
FE64 A0 03       55   QHEXTD1:   LDY# 03
FE66 B5 00       56   QHEXTD2:   LDA  00,X
FE68 20 6F FE    57              JSR   DHEXTD
FE6B 88          58              DEY
FE6C 88          59              DEY
FE6D B5 01       60              LDA  01,X
FE6F C8          61   DHEXTD:    INY
FE70 48          62              PHA
FE71 20 7A FE    63              JSR   HEXTD
FE74 88          64              DEY
FE75 68          65              PLA
FE76 4A          66              LSRA
FE77 4A          67              LSRA
FE78 4A          68              LSRA
FE79 4A          69              LSRA
FE7A 84 1A       70   HEXTD:     STY  1A
FE7C 29 0F       71              AND# 0F
FE7E A8          72              TAY
FE7F B9 EA FF    73              LDA   FONT,Y
FE82 A4 1A       74              LDY  1A
FE84 99 10 00    75              STA  0010,Y
FE87 60          76              RTS
FE88 20 64 FE    77   QDATFET:   JSR   QHEXTD1
FE8B 20 0C FE    78              JSR   DISPLAY
FE8E B0 20       79              BCS   RETURN (FEB0)
FE90 A0 04       80              LDY# 04
FE92 0A          81              ASLA
FE93 0A          82              ASLA
FE94 0A          83              ASLA
FE95 0A          84              ASLA
FE96 0A          85              ASLA
FE97 36 00       86              ROL  00,X
FE99 36 01       87              ROL  01,X
FE9B 88          88              DEY
FE9C D0 F8       89              BNE  F8 (FE96)
FE9E F0 E8       90              BEQ   QDATFET (FE88)
FEA0 F6 06       91   COM16:     INC  06,X
FEA2 D0 02       92              BNE  02 (FEA6)
FEA4 F6 07       93              INC  07,X
FEA6 B5 07       94   NOINC:     LDA  07,X
FEA8 D5 09       95              CMP  09,X
FEAA D0 04       96              BNE   RETURN (FEB0)
FEAC B5 06       97              LDA  06,X
FEAE D5 08       98              CMP  08,X
FEB0 60          99   RETURN:    RTS
FEB1 4C 00 DC   100   PUTBYTE:   JMP   PUT
FEB4 AD 03 0A   101              LDA  0A03              ;RTI CHECK
FEB7 AA         102              TAX
FEB8 29 03      103              AND# 03
```

```
 FEBA F0 07         104              BEQ    07 (FEC3)
 FEBC A5 ØA         105              LDA    ØA                    ;RELOAD REGIS
TERS
 FEBE A6 ØB         106              LDX    ØB
 FECØ A4 ØC         107              LDY    ØC
 FEC2 4Ø            108              RTI
 FEC3 8A            109              TXA
 FEC4 Ø9 Ø2         11Ø              ORA#   Ø2
 FEC6 8D Ø3 ØA      111              STA    ØAØ3
 FEC9 A2 Ø2         112              LDX#   Ø2
 FECB 2Ø D2 FE      113              JSR     VECSET
 FECE A9 Ø1         114              LDA#   Ø1
 FEDØ DØ 62         115              BNE    62 (FF34)
 FED2 A9 B3         116    VECSET:   LDA#   B3
 FED4 9D FA ØE      117              STA    ØEFA,X
 FED7 A9 FF         118              LDA#   FF
 FED9 9D FB ØE      119              STA    ØEFB,X
 FEDC 6Ø            12Ø              RTS
 FEDD 4C 9A DC      121    GETBYTE:  JMP     GET
 FEEØ AE Ø3 F8      122    RST:      LDX    F8Ø3
 FEE3 DØ ØE         123              BNE     RESET  (FEF3)
 FEE5 CA            124              DEX
 FEE6 8E ØØ ØA      125              STX    ØAØØ
 FEE9 2Ø 83 FF      126              JSR     INIT1
 FEEC 4C ØØ F8      127              JMP    F8ØØ
 FEEF EA            128              NOP
 FEFØ EA            129              NOP
 FEF1 EA            13Ø              NOP
 FEF2 EA            131              NOP
 FEF3 AØ 8Ø         132    RESET:    LDY#   8Ø
 FEF5 A2 Ø9         133              LDX#   Ø9
 FEF7 94 ØE         134              STY    ØE,X
 FEF9 CA            135              DEX
 FEFA 1Ø FB         136              BPL    FB (FEF7)
 FEFC 9A            137              TXS
 FEFD 8E ØØ ØA      138              STX    ØAØØ
 FFØØ 2Ø 83 FF      139              JSR     INIT1
 FFØ3 D8            14Ø              CLD
 FFØ4 2Ø ØC FE      141    RESTART:  JSR     DISPLAY
 FFØ7 9Ø EA         142              BCC    EA (FEF3)
 FFØ9 29 Ø7         143    SEARCH:   AND#   Ø7
 FFØB C9 Ø4         144              CMP#   Ø4
 FFØD 9Ø 25         145              BCC     FETADD (FF34)
 FFØF FØ 6F         146              BEQ     LOAD (FF8Ø)
 FF11 C9 Ø6         147              CMP#   Ø6
 FF13 FØ Ø9         148              BEQ     UP (FF1E)
 FF15 BØ ØF         149              BCS     DOWN (FF26)
 FF17 DØ 9B         15Ø              BNE    9B (FEB4)
 FF19 2Ø EF DD      151              JSR     SET                  ;SET FLAG & I
NT VECTORS
 FF1C DØ 7B         152              BNE    7B (FF99)
 FF1E F6 ØØ         153    UP:       INC    ØØ,X
 FF2Ø DØ ØC         154              BNE    ØC (FF2E)
```

```
FF22 F6 Ø1                 INC    Ø1,X
FF24 BØ Ø8                 BCS    Ø8 (FF2E)
FF26 B5 ØØ        DOWN:    LDA    ØØ,X
FF28 DØ Ø2                 BNE    Ø2 (FF2C)
FF2A D6 Ø1                 DEC    Ø1,X
FF2C D6 ØØ                 DEC    ØØ,X
FF2E 2Ø 64 FE              JSR     QHEXTD1
FF31 4C 45 FF              JMP     MODIFY
FF34 84 16        FETADD:  STY    16
FF36 84 17                 STY    17
FF38 9Ø 4F                 BCC    4F (FF89)
FF3A 8A                    TXA
FF3B ØA                    ASLA
FF3C AA                    TAX
FF3D EA                    NOP
FF3E 2Ø 88 FE              JSR     QDATFET
FF41 EØ Ø2                 CPX#   Ø2
FF43 BØ 15                 BCS    15 (FF5A)
FF45 2Ø 5E FE     MODIFY:  JSR     MHEXTD
FF48 2Ø ØC FE              JSR     DISPLAY
FF4B BØ BC                 BCS     SEARCH (FFØ9)
FF4D A1 ØØ                 LDA   (ØØ,X)
FF4F ØA                    ASLA
FF5Ø ØA                    ASLA
FF51 ØA                    ASLA
FF52 ØA                    ASLA
FF53 Ø5 ØD                 ORA    ØD
FF55 81 ØØ                 STA   (ØØ,X)
FF57 4C 45 FF              JMP     MODIFY
FF5A DØ Ø3                 BNE    Ø3 (FF5F)
FF5C 6C Ø2 ØØ     GO:      JMP   (ØØØ2)
FF5F EØ Ø4                 CPX#   Ø4
FF61 FØ B6                 BEQ    B6 (FF19)
FF63 2Ø 8D DD     STORE:   JSR     BAUDE1
FF66 A1 Ø6                 LDA   (Ø6,X)
FF68 2Ø B1 FE              JSR     PUTBYTE
FF6B 2Ø AØ FE              JSR     COM16
FF6E DØ F6                 BNE    F6 (FF66)
FF7Ø AD Ø1 ØA              LDA    ØAØ1
FF73 29 2Ø                 AND#   2Ø
FF75 DØ F9                 BNE    F9 (FF7Ø)
FF77 A9 ØØ                 LDA#   ØØ
FF79 8D Ø2 ØA              STA    ØAØ2
FF7C FØ 2C                 BEQ    2C (FFAA)
FF7E EA                    NOP
FF7F EA                    NOP
FF8Ø 4C CD DD     LOAD:    JMP     LOADE
FF83 E8           INIT1:   INX
FF84 A9 47                 LDA#   47
FF86 4C 52 FE              JMP     INIT2
FF89 AØ 8Ø                 LDY#   8Ø
FF8B 84 15                 STY    15
FF8D AA                    TAX
FF8E BD 95 FF              LDA     CHAR,X
```

```
FF91 85 1Ø                          STA    1Ø
FF93 9Ø A5                          BCC    A5 (FF3A)
FF95 F7 BD F3 F1
                      CHAR:         DAB   F7 BD F3 F1
FF99 A1 ØØ            POINT:        LDA   (ØØ,X)
FF9B FØ Ø6                          BEQ    Ø6 (FFA3)
FF9D 85 18                          STA    18
FF9F A9 ØØ                          LDA#   ØØ
FFA1 FØ Ø2                          BEQ    Ø2 (FFA5)
FFA3 A5 18                          LDA    18
FFA5 81 ØØ                          STA   (ØØ,X)
FFA7 2Ø 5E FE                       JSR     MHEXTD
FFAA 4C Ø4 FF         WAYOUT:       JMP     RESTART
FFAD 6C FC ØE                       JMP   (ØEFC)
FFBØ 6C FE ØE                       JMP   (ØEFE)
FFB3 85 ØA            BREAK:        STA    ØA
FFB5 86 ØB                          STX    ØB
FFB7 84 ØC                          STY    ØC
FFB9 68                             PLA
FFBA 48                             PHA
FFBB 85 ØD                          STA    ØD
FFBD A2 ØD                          LDX#   ØD
FFBF A9 FF                          LDA#   FF
FFC1 85 ØE                          STA    ØE
FFC3 2Ø ØØ FE                       JSR     QUAD
FFC6 BA                             TSX
FFC7 86 13                          STX    13
FFC9 C8                             INY
FFCA 84 12                          STY    12
FFCC D8                             CLD
FFCD 68                             PLA
FFCE 29 1Ø                          AND#   1Ø
FFDØ 4A                             LSRA
FFD1 4A                             LSRA
FFD2 4A                             LSRA
FFD3 85 1B                          STA    1B
FFD5 38                             SEC
FFD6 68                             PLA
FFD7 E5 1B                          SBC    1B
FFD9 85 11                          STA    11
FFDB 48                             PHA
FFDC 68                             PLA
FFDD 68                             PLA
FFDE E9 ØØ                          SBC#   ØØ
FFEØ 85 1Ø                          STA    1Ø
FFE2 48                             PHA
FFE3 9A                             TXS
FFE4 A2 13                          LDX#   13
FFE6 4C FA DD                       JMP     BRKCON          ;DISPLAY&CONT
INUE
FFE9 EA                             NOP
FFEA 3F Ø6 5B 4F 66 6D 7D Ø7 7F 6F 77 7C 39 5E 79 71
                      FONT:         DAB   3F Ø6 5B 4F 66 6D 7D Ø7 7F 6F
77 7C 39 5E 79 71
```

```
 FFFA AD FF EØ FE BØ FF
                      259    VECTORS:     DAB   AD FF EØ FE BØ FF
                      260                 ORG   DCØØ                  ;CASSETTE ROU
TINES.24/4/84
 DCØØ 85 1B           261    PUT:         STA    1B
 DCØ2 8A              262                 TXA
 DCØ3 48              263                 PHA
 DCØ4 AD Ø1 ØA        264                 LDA    ØAØ1
 DCØ7 29 2Ø           265                 AND#   2Ø
 DCØ9 DØ 44           266                 BNE     SEND (DC4F)
 DCØB A9 75           267                 LDAL    INT
 DCØD 8D FE ØE        268                 STA    ØEFE
 DC1Ø A9 DC           269                 LDAH    INT
 DC12 8D FF ØE        270                 STA    ØEFF
 DC15 A9 4Ø           271    HEAD:        LDA#   4Ø
 DC17 8D ØB Ø9        272                 STA    Ø9ØB
 DC1A AD Ø1 ØA        273                 LDA    ØAØ1
 DC1D Ø9 3Ø           274                 ORA#   3Ø
 DC1F 29 F7           275                 AND#   F7
 DC21 8D Ø1 ØA        276                 STA    ØAØ1
 DC24 78              277                 SEI
 DC25 29 Ø2           278                 AND#   Ø2
 DC27 4A              279                 LSRA
 DC28 A8              28Ø                 TAY
 DC29 B9 73 DC        281                 LDA     DATA,Y
 DC2C 85 1E           282                 STA    1E
 DC2E A2 CØ           283                 LDX#   CØ
 DC3Ø 8E ØE Ø9        284                 STX    Ø9ØE
 DC33 A9 CE           285                 LDA#   CE
 DC35 8D Ø4 Ø9        286                 STA    Ø9Ø4
 DC38 85 ØC           287                 STA    ØC
 DC3A A9 ØØ           288                 LDA#   ØØ
 DC3C 8D Ø5 Ø9        289                 STA    Ø9Ø5
 DC3F 58              29Ø                 CLI
 DC4Ø A9 55           291                 LDA#   55
 DC42 85 1C           292                 STA    1C
 DC44 A9 6Ø           293    BACK4:       LDA#   6Ø
 DC46 85 1F           294                 STA    1F
 DC48 A5 1F           295                 LDA    1F
 DC4A DØ FC           296                 BNE    FC (DC48)
 DC4C CA              297                 DEX
 DC4D DØ F5           298                 BNE     BACK4 (DC44)
 DC4F A9 Ø4           299    SEND:        LDA#   Ø4
 DC51 85 ØC           3ØØ                 STA    ØC
 DC53 A2 Ø9           3Ø1                 LDX#   Ø9
 DC55 DØ Ø4           3Ø2                 BNE    Ø4 (DC5B)
 DC57 46 1B           3Ø3    NEXTC:       LSR    1B
 DC59 BØ 11           3Ø4                 BCS     HIGH (DC6C)
 DC5B A9 66           3Ø5                 LDA#   66                 ;1.2KHZ
 DC5D 85 1C           3Ø6    CON2:        STA    1C
 DC5F A5 1E           3Ø7                 LDA    1E
 DC61 85 1F           3Ø8                 STA    1F
 DC63 A5 1F           3Ø9                 LDA    1F
 DC65 DØ FC           31Ø                 BNE    FC (DC63)
```

```
DC67 CA          311             DEX
DC68 30 06       312             BMI    END1 (DC70)
DC6A D0 EB       313             BNE    NEXTC (DC57)
DC6C A9 55       314   HIGH:     LDA#   55                  ;2.4KHZ
DC6E D0 ED       315             BNE    CON2 (DC5D)
DC70 68          316   END1:     PLA
DC71 AA          317             TAX
DC72 60          318             RTS
DC73 10 04       319   DATA:     DAB  10 04
DC75 48          320   INT:      PHA
DC76 AD 04 09    321             LDA    0904
DC79 A5 1C       322             LDA    1C
DC7B 6A          323             RORA
DC7C 66 1C       324             ROR    1C
DC7E 8D 02 0A    325             STA    0A02
DC81 C6 0B       326             DEC    0B
DC83 D0 11       327             BNE     RET (DC96)
DC85 C6 0C       328             DEC    0C
DC87 D0 0D       329             BNE     RET (DC96)
DC89 AD 01 0A    330             LDA    0A01
DC8C 29 C7       331             AND#   C7
DC8E 8D 01 0A    332             STA    0A01
DC91 A9 40       333             LDA#   40
DC93 8D 0E 09    334             STA    090E
DC96 C6 1F       335   RET:      DEC    1F
DC98 68          336             PLA
DC99 40          337             RTI
DC9A 8A          338   GET:      TXA                        ;FEDD
DC9B 48          339             PHA
DC9C A9 03       340             LDA#   03
DC9E 8D 04 09    341             STA    0904
DCA1 A9 08       342             LDA#   08
DCA3 85 1D       343             STA    1D
DCA5 A9 00       344             LDA#   00
DCA7 8D 02 0A    345             STA    0A02
DCAA 85 1C       346             STA    1C
DCAC A9 40       347             LDA#   40
DCAE 8D 0B 09    348             STA    090B
DCB1 20 50 DD    349             JSR     EDGE
DCB4 AD 01 0A    350             LDA    0A01
DCB7 29 30       351             AND#   30
DCB9 D0 14       352             BNE     BACK1 (DCCF)
DCBB A0 06       353   HSEARCH:  LDY#   06
DCBD 20 41 DD    354   YES:      JSR     TIMER
DCC0 E0 09       355             CPX#   09
DCC2 90 F7       356             BCC     HSEARCH (DCBB)
DCC4 E0 11       357             CPX#   11
DCC6 B0 F3       358             BCS     HSEARCH (DCBB)
DCC8 C6 0B       359             DEC    0B
DCCA D0 F1       360             BNE     YES (DCBD)
DCCC 88          361             DEY
DCCD D0 EE       362             BNE     YES (DCBD)
DCCF 20 41 DD    363   BACK1:    JSR     TIMER
DCD2 E0 17       364             CPX#   17
```

```
DCD4 90 F9                BCC    BACK1 (DCCF)
DCD6 85 1C                STA    1C
DCD8 AD Ø1 ØA             LDA    ØAØ1
DCDB 29 F7                AND#   F7
DCDD Ø9 3Ø                ORA#   3Ø
DCDF 8D Ø1 ØA             STA    ØAØ1
DCE2 29 Ø2                AND#   Ø2
DCE4 DØ Ø3                BNE    Ø3 (DCE9)
DCE6 2Ø 2C DD             JSR    WAIT
DCE9 2Ø 5Ø DD    CON1:    JSR    EDGE
DCEC A2 ØØ                LDX#   ØØ                ;EDGE COUNTER
DCEE AØ 21                LDY#   21                ;TIME COUNTER
DCFØ AD Ø2 ØA    BACK2:   LDA    ØAØ2
DCF3 29 8Ø                AND#   8Ø
DCF5 C5 1B                CMP    1B                ;NEW EDGE?
DCF7 FØ Ø3                BEQ    NO (DCFC)
DCF9 E8                   INX
DCFA 85 1B                STA    1B
DCFC 88          NO:      DEY
DCFD DØ F1                BNE    BACK2 (DCFØ)
DCFF EØ Ø2                CPX#   Ø2
DDØ1 66 1F                ROR    1F
DDØ3 AD Ø1 ØA             LDA    ØAØ1
DDØ6 29 Ø2                AND#   Ø2
DDØ8 AA                   TAX
DDØ9 DØ Ø3                BNE    Ø3 (DDØE)
DDØB 2Ø 36 DD             JSR    WAIT1
DDØE C6 1D                DEC    1D
DD1Ø DØ D7                BNE    CON1 (DCE9)
DD12 8A                   TXA
DD13 DØ ØA                BNE    SKIP (DD1F)
DD15 2Ø 5Ø DD             JSR    EDGE
DD18 2Ø 41 DD    SBIT:    JSR    TIMER
DD1B EØ 12                CPX#   12
DD1D BØ F9                BCS    SBIT (DD18)
DD1F 68          SKIP:    PLA
DD2Ø AA                   TAX
DD21 AD Ø1 ØA             LDA    ØAØ1
DD24 29 D7                AND#   D7
DD26 8D Ø1 ØA             STA    ØAØ1
DD29 A5 1F                LDA    1F
DD2B 6Ø                   RTS
DD2C AØ AØ       WAIT:    LDY#   AØ
DD2E 88                   DEY
DD2F DØ FD                BNE    FD (DD2E)
DD31 A9 ØD                LDA#   ØD
DD33 8D Ø5 Ø9             STA    Ø9Ø5
DD36 AD Ø4 Ø9    WAIT1:   LDA    Ø9Ø4
DD39 AD ØD Ø9    BACK3:   LDA    Ø9ØD
DD3C 29 4Ø                AND#   4Ø
DD3E FØ F9                BEQ    BACK3 (DD39)
DD4Ø 6Ø                   RTS
DD41 A2 ØØ       TIMER:   LDX#   ØØ
DD43 E8          TIME:    INX
```

```
DD44 AD Ø2 ØA    419              LDA   ØAØ2
DD47 29 8Ø       42Ø              AND#  8Ø
DD49 C5 1B       421              CMP   1B
DD4B FØ F6       422              BEQ    TIME  (DD43)
DD4D 85 1B       423              STA   1B
DD4F 6Ø          424              RTS
DD5Ø AD Ø2 ØA    425   EDGE:      LDA   ØAØ2
DD53 29 8Ø       426              AND#  8Ø
DD55 C5 1C       427              CMP   1C
DD57 DØ F7       428              BNE    EDGE  (DD5Ø)
DD59 AD Ø2 ØA    429   EDGE1:     LDA   ØAØ2
DD5C 29 8Ø       43Ø              AND#  8Ø
DD5E C5 1C       431              CMP   1C
DD6Ø FØ F7       432              BEQ    EDGE1  (DD59)
DD62 85 1B       433              STA   1B
DD64 6Ø          434              RTS
DD65 AD Ø1 ØA    435   BAUD:      LDA   ØAØ1
DD68 Ø9 Ø2       436              ORA#  Ø2
DD6A 8D Ø1 ØA    437              STA   ØAØ1
DD6D 2Ø EC E2    438              JSR   E2EC
DD7Ø A2 Ø8       439              LDX#  Ø8
DD72 86 1B       44Ø              STX   1B
DD74 BD 84 DD    441   BACK:      LDA    MESS,X
DD77 2Ø 5D EØ    442              JSR   EØ5D
DD7A C6 1B       443              DEC   1B
DD7C A6 1B       444              LDX   1B
DD7E 1Ø F4       445              BPL    BACK  (DD74)
DD8Ø 2Ø EC E2    446              JSR   E2EC
DD83 6Ø          447              RTS
DD84 44 55 41 42 2Ø 3Ø 3Ø 32 31
                 448   MESS:      DAB   'DUAB ØØ21'
DD8D 66 1Ø       449   BAUDE1:    ROR   1Ø                    ;"T"
DD8F A2 Ø8       45Ø              LDX#  Ø8
DD91 2Ø 88 FE    451              JSR    QDATFET
DD94 2Ø A7 DD    452              JSR    BAUDE
DD97 Ø6 1Ø       453              ASL   1Ø
DD99 2Ø 64 FE    454              JSR    QHEXTD1
DD9C A2 Ø4       455              LDX#  Ø4
DD9E B5 Ø5       456   N1:        LDA   Ø5,X
DDAØ 2Ø B1 FE    457              JSR    PUTBYTE
DDA3 CA          458              DEX
DDA4 DØ F8       459              BNE    N1  (DD9E)
DDA6 6Ø          46Ø              RTS
DDA7 AØ Ø8       461   BAUDE:     LDY#  Ø8
DDA9 B9 C5 DD    462   B1:        LDA    MESSE,Y
DDAC 99 1Ø ØØ    463              STA   ØØ1Ø,Y
DDAF 88          464              DEY
DDBØ 1Ø F7       465              BPL    B1  (DDA9)
DDB2 2Ø ØC FE    466              JSR    DISPLAY
DDB5 A8          467              TAY
DDB6 AD Ø1 ØA    468              LDA   ØAØ1
DDB9 Ø9 Ø2       469              ORA#  Ø2
```

```
DDBB C0 01          470               CPY#  01
DDBD.F0 02          471               BEQ   02 (DDC1)
DDBF 49 02          472               EOR#  02
DDC1 8D 01 0A       473               STA   0A01
DDC4 60             474               RTS
DDC5 7C 77 3E 5E 40 00 00 00
                    475    MESSE:     DAB  7C 77 3E 5E 40 00 00 00
DDCD 20 A7 DD       476    LOADE:     JSR    BAUDE
DDD0 A2 04          477               LDX#  04
DDD2 20 DD FE       478    B2:        JSR    GETBYTE
DDD5 95 05          479               STA   05,X
DDD7 CA             480               DEX
DDD8 D0 F8          481               BNE    B2 (DDD2)
DDDA 20 DD FE       482    B3:        JSR    GETBYTE
DDDD 81 06          483               STA   (06,X)
DDDF 20 A0 FE       484               JSR    COM16
DDE2 D0 F6          485               BNE    B3 (DDDA)
DDE4 4C F3 FE       486               JMP    RESET
DDE7 A2 1A          487               LDX#  1A                   ;S KEY PATCH
DDE9 20 9F E2       488               JSR   E29F                 ;FROM VISA
DDEC 4C 07 E4       489               JMP   E407
DDEF AD 03 0A       490    SET:       LDA   0A03                 ;SET FLAG &
DDF2 09 01          491               ORA#  01                   ;INT VECTORS
DDF4 8D 03 0A       492               STA   0A03
DDF7 4C D2 FE       493               JMP    VECSET
DDFA 20 00 FE       494    BRKCON:    JSR    QUAD                ;BRK PATCH
DDFD 4C 07 FF       495               JMP    RESTART+03
                    496               ;NEW KEYBOARD ROUTINES.
                    497               ORG  D800
D800 20 E3 D8       498    KDISPC:    JSR    INIT
D803 D0 06          499               BNE    06 (D80B)
D805 20 E3 D8       500    KDISP:     JSR    INIT
D808 20 0C FE       501    KEY:       JSR   FE0C
D80B B0 22          502               BCS    CONVERT  (D82F)
D80D 38             503               SEC
D80E A9 08          504               LDA#  08
D810 E5 0A          505               SBC   0A
D812 AA             506               TAX
D813 BD 91 0E       507    BA2:       LDA   0E91,X
D816 9D 90 0E       508               STA   0E90,X
D819 B5 11          509               LDA   11,X
D81B 95 10          510               STA   10,X
D81D E8             511               INX
D81E E0 07          512               CPX#  07
D820 90 F1          513               BCC    BA2 (D813)
D822 A4 0D          514               LDY   0D
D824 8C 97 0E       515               STY   0E97
D827 B9 EA FF       516               LDA   FFEA,Y
D82A 85 17          517               STA   17
D82C 4C 08 D8       518               JMP    KEY
D82F A2 00          519    CONVERT:   LDX#  00
D831 A0 07          520               LDY#  07
D833 B9 8F 0E       521    BA3:       LDA   0E8F,Y
D836 0A             522               ASLA
```

```
D837 ØA           523                 ASLA
D838 ØA           524                 ASLA
D839 ØA           525                 ASLA
D83A 19 9Ø ØE     526                 ORA    ØE9Ø,Y
D83D 95 1C        527                 STA    1C,X
D83F E8           528                 INX
D84Ø 88           529                 DEY
D841 88           53Ø                 DEY
D842 1Ø EF        531                 BPL     BA3  (D833)
D844 A6 ØB        532                 LDX    ØB
D846 A4 ØA        533                 LDY    ØA
D848 6Ø           534                 RTS
D849 A9 ØØ        535     DISP8:      LDA#   ØØ
D84B FØ Ø6        536                 BEQ    Ø6  (D853)
D84D A5 1B        537     DISP:       LDA    1B
D84F 85 1Ø        538                 STA    1Ø
D851 A9 Ø2        539                 LDA#   Ø2
D853 85 ØC        54Ø                 STA    ØC
D855 86 ØB        541                 STX    ØB
D857 A2 ØØ        542                 LDX#   ØØ
D859 86 11        543                 STX    11
D85B AØ Ø6        544                 LDY#   Ø6
D85D B5 1C        545     BA5:        LDA    1C,X
D85F 2Ø 6F FE     546                 JSR    FE6F
D862 C4 ØC        547                 CPY    ØC
D864 FØ Ø5        548                 BEQ    Ø5  (D86B)
D866 88           549                 DEY
D867 88           55Ø                 DEY
D868 E8           551                 INX
D869 DØ F2        552                 BNE     BA5  (D85D)
D86B AØ ØØ        553                 LDY#   ØØ
D86D A6 ØC        554                 LDX    ØC
D86F B5 1Ø        555     BA7:        LDA    1Ø,X
D871 C9 3F        556                 CMP#   3F
D873 DØ Ø7        557                 BNE     END  (D87C)
D875 94 1Ø        558                 STY    1Ø,X
D877 E8           559                 INX
D878 EØ Ø7        56Ø                 CPX#   Ø7
D87A DØ F3        561                 BNE     BA7  (D86F)
D87C A6 ØB        562     END:        LDX    ØB
D87E 2Ø ØC FE     563                 JSR    FEØC
D881 6Ø           564                 RTS
D882 18           565     ADD:        CLC
D883 65 ØA        566                 ADC    ØA
D885 85 ØA        567                 STA    ØA
D887 A5 ØB        568                 LDA    ØB
D889 69 ØØ        569                 ADC#   ØØ
D88B 85 ØB        57Ø                 STA    ØB
D88D 4C AF DA     571                 JMP      ADDP
                  572                 ;
D89Ø A5 ØB        573     MU1Ø:       LDA    ØB
D892 48           574                 PHA
D893 A5 ØA        575                 LDA    ØA
D895 A4 ØC        576                 LDY    ØC
```

```
D897 20 AC D8                    JSR    SHIFT
D89A 20 AC D8                    JSR    SHIFT
D89D 18                          CLC
D89E 65 0A                       ADC    0A
D8A0 85 0A                       STA    0A
D8A2 68                          PLA
D8A3 65 0B                       ADC    0B
D8A5 85 0B                       STA    0B
D8A7 98                          TYA
D8A8 65 0C                       ADC    0C
D8AA 85 0C                       STA    0C
D8AC 06 0A          SHIFT:       ASL    0A
D8AE 26 0B                       ROL    0B
D8B0 26 0C                       ROL    0C
D8B2 60                          RTS
D8B3 A0 04          MU16:        LDY#   04
D8B5 18             NEXT2:       CLC
D8B6 A5 0A                       LDA    0A
D8B8 65 0A                       ADC    0A
D8BA 85 0A                       STA    0A
D8BC A5 0B                       LDA    0B
D8BE 65 0B                       ADC    0B
D8C0 85 0B                       STA    0B
D8C2 A5 0C                       LDA    0C
D8C4 65 0C                       ADC    0C
D8C6 85 0C                       STA    0C
D8C8 88                          DEY
D8C9 D0 EA                       BNE     NEXT2  (D8B5)
D8CB 60                          RTS
D8CC A9 00          RESET1:      LDA#   00
D8CE 85 0A                       STA    0A
D8D0 85 0B                       STA    0B
D8D2 85 0C                       STA    0C
D8D4 60                          RTS
D8D5 A0 07          CLEAR:       LDY#   07
D8D7 A9 00                       LDA#   00
D8D9 99 10 00       C1:          STA    0010,Y
D8DC 99 90 0E                    STA    0E90,Y
D8DF 88                          DEY
D8E0 10 F7                       BPL     C1  (D8D9)
D8E2 60                          RTS
D8E3 84 0A          INIT:        STY    0A
D8E5 86 0B                       STX    0B
D8E7 A0 FF                       LDY#   FF
D8E9 84 0E                       STY    0E
D8EB A0 03          ZERO:        LDY#   03
D8ED A9 00                       LDA#   00
D8EF 99 1C 00       C2:          STA    001C,Y
D8F2 99 90 0E                    STA    0E90,Y
D8F5 99 94 0E                    STA    0E94,Y
D8F8 88                          DEY
D8F9 10 F4                       BPL     C2  (D8EF)
D8FB 60                          RTS
D8FC 6C FA 0E       MULT:        JMP   (0EFA)
```

```
                                     ;HEX-DEC-HEX CONVERSION ROUTINE.
                                     ORG   D900
D900 20 D5 D8           ENT:         JSR    CLEAR
D903 A9 F6                           LDA#   F6
D905 85 10                           STA    10
D907 85 1B                           STA    1B
D909 20 0C FE                        JSR    FE0C
D90C A5 1B              CONT:        LDA    1B
D90E 49 F6                           EOR#   F6
D910 F0 58                           BEQ     HEX  (D96A)
D912 D8                 DECIMAL:     CLD
D913 A2 05                           LDX#   05
D915 A9 90                           LDAL    MU10
D917 8D FA 0E                        STA    0EFA
D91A A9 D8                           LDAH    MU10
D91C 8D FB 0E           CONH1:       STA    0EFB
D91F 20 D5 D8                        JSR     CLEAR
D922 8A                              TXA
D923 A8                              TAY
D924 A5 1B                           LDA    1B
D926 85 10                           STA    10
D928 20 00 D8                        JSR     KDISPC
D92B A5 0D                           LDA    0D
D92D C9 14                           CMP#   14
D92F F0 46                           BEQ     FIN  (D977)
D931 20 CC D8                        JSR     RESET1
D934 A2 02                           LDX#   02
D936 20 FC D8           NEXT1:       JSR     MULT
D939 B5 1C                           LDA    1C,X
D93B 48                              PHA
D93C 4A                              LSRA
D93D 4A                              LSRA
D93E 4A                              LSRA
D93F 4A                              LSRA
D940 18                              CLC
D941 69 00                           ADC#   00
D943 20 82 D8                        JSR     ADD
D946 20 FC D8                        JSR     MULT
D949 68                              PLA
D94A 29 0F                           AND#   0F
D94C 18                              CLC
D94D 69 00                           ADC#   00
D94F 20 82 D8                        JSR     ADD
D952 CA                              DEX
D953 10 E1                           BPL     NEXT1  (D936)
D955 A2 02                           LDX#   02
D957 B5 0A              BA6:         LDA    0A,X
D959 95 1C                           STA    1C,X
D95B CA                              DEX
D95C 10 F9                           BPL     BA6  (D957)
D95E A5 1B                           LDA    1B
D960 49 28                           EOR#   28
D962 85 1B                           STA    1B
D964 20 4D D8                        JSR     DISP
```

```
D967 4C ØC D9               JMP    CONT
D96A F8         HEX:        SED
D96B A2 Ø4                  LDX#  Ø4
D96D A9 B3                  LDAL   MU16
D96F 8D FA ØE               STA   ØEFA
D972 A9 D8                  LDAH   MU16
D974 4C 1C D9               JMP    CONH1
D977 4C F3 FE   FIN:        JMP    FEF3
D97A DE F6      COM:        DAB  DE F6
                            ;REL1-BRANCH&RELOCATOR
                            ORG  D98Ø
D980 D8                     CLD
D981 2Ø D5 D8   RESTARTB:   JSR    CLEAR
D984 2Ø EB D8               JSR    ZERO
D987 A9 F1                  LDA#  F1
D989 85 1Ø                  STA   1Ø
D98B A2 1C                  LDX#  1C
D98D 2Ø 88 FE               JSR   FE88
D99Ø A9 F8                  LDA#  F8
D992 85 1Ø                  STA   1Ø
D994 A2 1E                  LDX#  1E
D996 2Ø 88 FE               JSR   FE88
D999 2Ø D5 D8               JSR    CLEAR
D99C A5 1C                  LDA   1C
D99E E9 7E                  SBC#  7E
D9AØ 85 1C                  STA   1C
D9A2 BØ Ø3                  BCS   Ø3  (D9A7)
D9A4 C6 1D                  DEC   1D
D9A6 38                     SEC
D9A7 A5 1E                  LDA   1E
D9A9 E5 1C                  SBC   1C
D9AB AA                     TAX
D9AC A5 1F                  LDA   1F
D9AE E5 1D                  SBC   1D
D9BØ DØ 13                  BNE    ERROR  (D9C5)
D9B2 8A                     TXA
D9B3 49 8Ø                  EOR#  8Ø
D9B5 2Ø 6Ø FE               JSR   FE6Ø
D9B8 A9 BF                  LDA#  BF
D9BA 85 1Ø      CB1:        STA   1Ø
D9BC 2Ø ØC FE               JSR   FEØC
D9BF C9 14                  CMP#  14
D9C1 DØ BE                  BNE    RESTARTB  (D981)
D9C3 FØ 4B                  BEQ    END2  (DA1Ø)
D9C5 A9 F9      ERROR:      LDA#  F9
D9C7 DØ F1                  BNE    CB1  (D9BA)
                            ;RELOCATOR ROUTINE.
D9C9 2Ø CC D8   RELOC:      JSR    RESET1
D9CC 2Ø EB D8               JSR    ZERO
D9CF 2Ø D5 D8               JSR    CLEAR
D9D2 A9 F1                  LDA#  F1                   ;"F"
D9D4 85 1Ø                  STA   1Ø
D9D6 A2 1C                  LDX#  1C
D9D8 2Ø 88 FE               JSR   FE88
```

```
D9DB A9 F8      739           LDA#  F8                 ;"T"
D9DD 85 1Ø      74Ø           STA   1Ø
D9DF A2 1E      741   ERR:    LDX#  1E
D9E1 2Ø 88 FE   742           JSR   FE88
D9E4 A2 16      743           LDX#  16
D9E6 2Ø A6 FE   744           JSR   FEA6
D9E9 BØ F4      745           BCS     ERR (D9DF)
D9EB A9 DE      746           LDA#  DE                 ;"D"
D9ED 85 1Ø      747           STA   1Ø
D9EF A2 ØA      748           LDX#  ØA
D9F1 2Ø 88 FE   749           JSR   FE88
D9F4 A2 16      75Ø           LDX#  16
D9F6 A5 1D      751           LDA   1D
D9F8 C5 ØB      752           CMP   ØB
D9FA DØ Ø4      753           BNE   Ø4 (DAØØ)
D9FC A5 1C      754           LDA   1C
D9FE C5 ØA      755           CMP   ØA
DAØØ BØ 11      756           BCS     DEC (DA13)
DAØ2 A1 Ø6      757   R1:     LDA   (Ø6,X)
DAØ4 91 ØA      758           STA   (ØA),Y
DAØ6 C8         759           INY
DAØ7 DØ Ø2      76Ø           BNE   Ø2 (DAØB)
DAØ9 E6 ØB      761           INC   ØB
DAØB 2Ø AØ FE   762           JSR   FEAØ
DAØE DØ F2      763           BNE     R1 (DAØ2)
DA1Ø 4C F3 FE   764   END2:   JMP   FEF3
DA13 38         765   DEC:    SEC
DA14 A5 1E      766           LDA   1E
DA16 E5 1C      767           SBC   1C
DA18 85 1A      768           STA   1A
DA1A A5 1F      769           LDA   1F
DA1C E5 1D      77Ø           SBC   1D
DA1E 85 1B      771           STA   1B
DA2Ø 18         772           CLC
DA21 A5 1A      773           LDA   1A
DA23 65 ØA      774           ADC   ØA
DA25 85 ØA      775           STA   ØA
DA27 A5 1B      776           LDA   1B
DA29 65 ØB      777           ADC   ØB
DA2B 85 ØB      778           STA   ØB
DA2D CØ ØØ      779   CON:    CPY#  ØØ
DA2F DØ Ø2      78Ø           BNE   Ø2 (DA33)
DA31 C6 ØB      781           DEC   ØB
DA33 88         782           DEY
DA34 A5 1E      783           LDA   1E
DA36 DØ Ø2      784           BNE   Ø2 (DA3A)
DA38 C6 1F      785           DEC   1F
DA3A C6 1E      786           DEC   1E
DA3C A1 Ø8      787           LDA   (Ø8,X)
DA3E 91 ØA      788           STA   (ØA),Y
DA4Ø 2Ø A6 FE   789           JSR   FEA6
DA43 DØ E8      79Ø           BNE     CON (DA2D)
DA45 FØ C9      791           BEQ     END2 (DA1Ø)
```

```
                792              ;CHECKSUM ROUTINE.
                793              ORG   DA5Ø
DA5Ø 2Ø D5 D8   794              JSR     CLEAR
DA53 2Ø EB D8   795              JSR     ZERO
DA56 A9 F1      796              LDA#   F1
DA58 85 1Ø      797              STA    1Ø
DA5A A2 1E      798              LDX#   1E
DA5C 2Ø 88 FE   799              JSR    FE88
DA5F 2Ø D5 D8   8ØØ              JSR     CLEAR
DA62 A9 B7      8Ø1              LDA#   B7
DA64 85 1Ø      8Ø2              STA    1Ø
DA66 A2 1C      8Ø3              LDX#   1C
DA68 2Ø 88 FE   8Ø4              JSR    FE88
DA6B 2Ø CC D8   8Ø5              JSR     RESET1
DA6E A2 ØØ      8Ø6              LDX#   ØØ
DA7Ø 4C 8E DA   8Ø7              JMP     DECC
DA73 18         8Ø8    ADDS:     CLC
DA74 A1 1E      8Ø9              LDA    (1E,X)
DA76 65 ØA      81Ø              ADC    ØA
DA78 85 ØA      811              STA    ØA
DA7A 9Ø Ø6      812              BCC    Ø6 (DA82)
DA7C E6 ØB      813              INC    ØB
DA7E DØ Ø2      814              BNE    Ø2 (DA82)
DA8Ø E6 ØC      815              INC    ØC
DA82 E6 1E      816    INC:      INC    1E                  ;INC ADDRESS
DA84 DØ Ø2      817              BNE    Ø2 (DA88)
DA86 E6 1F      818              INC    1F
DA88 A5 1C      819    CMP:      LDA    1C
DA8A Ø5 1D      82Ø              ORA    1D
DA8C FØ ØB      821              BEQ     ENDS (DA99)
DA8E A5 1C      822    DECC:     LDA    1C                  ;DEC BYTES
DA9Ø DØ Ø2      823              BNE    Ø2 (DA94)
DA92 C6 1D      824              DEC    1D
DA94 C6 1C      825              DEC    1C
DA96 4C 73 DA   826              JMP     ADDS
DA99 A9 B9      827    ENDS:     LDA#   B9
DA9B 85 1Ø      828              STA    1Ø
DA9D A5 ØA      829              LDA    ØA
DA9F 85 1C      83Ø              STA    1C
DAA1 A5 ØB      831              LDA    ØB
DAA3 85 1D      832              STA    1D
DAA5 A5 ØC      833              LDA    ØC
DAA7 85 1E      834              STA    1E
DAA9 2Ø 51 D8   835              JSR     DISP+Ø4
DAAC 4C Ø7 FF   836              JMP    FFØ7
DAAF A5 ØC      837    ADDP:     LDA    ØC
DAB1 69 ØØ      838              ADC#   ØØ
DAB3 85 ØC      839              STA    ØC
DAB5 6Ø         84Ø              RTS
```

Appendix 3 Summary of Single Cycle Execution

This section contains an outline of the data on both the address bus and the data bus for each cycle of the various processor instructions. It tells the system designer exactly what to expect while single cycling through a program.

Note that the processor will not stop in any cycle where R/$\overline{W}$ is a 0 (write cycle). Instead, it will go right into the next read cycle and stop there. For this reason, some instructions may appear to be shorter than indicated here.

All instructions begin with TØ and the fetch of the OP CODE and continue through the required number of cycles until the next TØ and the fetch of the next OP CODE.

Definitions of some of the terms used in this appendix are given below:

Op Code

The first byte of the instruction containing the operator and mode of address.

Base Address

The address in indexed addressing modes which specifies the location in memory to which indexing is referenced. The high order of byte of the base address (AB08 to AB15) is BAH (Base Address High) and the low order byte of the base address (AB00 to AB07) is BAL (Base Address Low).

Effective Address

The destination in memory in which data is to be found. The effective address may be loaded directly as in the case of Page Zero and Absolute Addressing or may be calculated as in Indexing operations. The high order byte of the effective address (AB08 to AB15) is ADH and the low order-byte of the effective address (AB00-AB07) is ADL.

Indirect Address

The address found in the operand of instructions utilizing (indirect), Y which contains the low order byte of the base address. IAH and IAL represent the high and low order bytes.

JUMP ADDRESS The value to be loaded into Program Counter as a result of a Jump instruction.

A.1 Single Byte Instructions

ASL
CLC
CLD
CLI
CLV
DEX
DEY
INX
INY
LSR
NOP
ROL
SEC
SED
SEI
TAX
TAY
TSX
TXA
TXS
TYA

These single byte instructions require two cycles to execute. During the second cycle the address of the next instruction in program sequence will be placed on the address bus. However, the OP CODE which appears on the data bus during the second cycle will be ignored. This same instruction will be fetched on the following cycle at which time it will be decoded and executed. The ASL, ROL and LSR instructions apply to the accumulator mode of address.

A.2 Internal Execution on Memory Data

ADC
AND
BIT
CMP
CPX
CPY
EOR
LDA
LDX
LDY
ORA
SBC

The instructions listed above will execute by performing operations inside the microprocessor using data fetched from the effective address. This total operation requires three steps. The first step (one cycle) is the OP CODE fetch. The second (zero to four cycles) is the calculation of an effective address. The final step is the fetching of the data from the effective address. Execution of the instruction takes place during the fetching and decoding of the next instruction.

A.2.1 Immediate Addressing (2 cycles)

Tn	Address bus	Data Bus	R/W	Comments
TØ	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	Data	1	Fetch Data
TØ	PC + 2	OP CODE	1	Next instruction

A.2.2 Zero Page Addressing (3 cycles)

Tn	Address bus	Data Bus	R/W	Comments
TØ	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch effective address
T2	00,ADL	Data	1	Fetch Data
TØ	PC + 2	OP CODE	1	Next instruction

A.2.3 Absolute Addressing (4 cycles)

Tn	Address bus	Data Bus	R/W	Comments
TØ	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch Data
T2	PC + 2	ADH	1	Fetch high order effective address Byte
T3	ADH, ADL	Data	1	Fetch Data
T0	PC + 3	OP CODE	1	Next instruction

A.2.4 Indirect, X Addressing (6 cycles)

Tn	Address bus	Data Bus	R/W	Comments
TØ	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch Page Zero Base Address
T2	00,BAL	Data discard	1	
T3	00,BAL + X	ADL	1	Fetch low order byte of effective address
T4	00,BAL + X + 1	ADH	1	Fetch high order byte of effective address
T5	ADH,ADL	Data	1	Fetch Data
TØ	PC + 2	OP CODE	1	Next instruction

A.2.5 Absolute, X Absolute, Y Addressing (4 or 5 cycles)

Tn	Address bus	Data Bus	R/W	Comments
TØ	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch low order byte of Base Address
T2	PC + 2	BAH	1	Fetch high order byte of Base Address
T3	ADL: BAL + Index Register ADH: BAH + C	Data*	1	Fetch data (no page crossing) Carry is Ø or 1 as required from previous add operation
T4	ADL: BAL + Index Register	Data	1	Fetch data from next page
TØ	PC + 1	OP CODE	1	Next instruction

* If the page boundary is crossed in the indexing operation, the data fetched in T3 is ignored. If page boundary is not crossed, the T4 cycle is bypassed.

A.2.6 Zero Page, X or Zero Page, Y Addressing Modes (4 cycles)

Tn	Addresss Bus	Data Bus	R/W	Comments
TØ	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch Page Zero Base Address
T2	00, BAL	Data (Discarded)	1	
T3	00, BAL + Index Register	Data	1	Fetch Data (no page crossing)
TØ	PC + 2	OP CODE	1	Next instruction

A.2.7 Indirect, Y Addressing Mode (5 or 6 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
TØ	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	IAL	1	Fetch Page Zero Indirect Address
T2	00, IAL	BAL	1	Fetch low order byte of Base Address
T3	00, IAL + 1	BAH	1	Fetch high order byte of Base Address
T4	ADL: BAL + Y	Data*	1	Fetch Data from same page
	ADH: BAH + C			Carry is 0 or 1 as required from previous add operation
T5*	ADL: BAL + Y ADH: BAH + 1	Data	1	Fetch Data from next page
TØ	PC + 2	OP CODE	1	Next instruction

* If page boundry is crossed in indexing operation, the data fetch in T4 is ignored. If page boundary is not crossed, the T5 cycle is bypassed.

A.3 Store Operations

STA
STX
STY

The specific steps taken in the Store Operations are very similar to those taken in the previous group (Internal execution on memory data). However, in the Store Operation, the fetch of data is replaced by a WRITE (R/$\overline{W}$ = 0) cycle. No overlapping occurs and no shortening of the instruction time occurs on indexing operations.

A.3.1 Zero Page Addressing (3 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
TØ	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch Zero Page Effective Address
T2	00, ADL	Data	0	Write internal register to memory
TØ	PC + 2	OP CODE	1	Next instruction

A.3.2 Absolute Addressing (4 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
TØ	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch low order byte of Effective Address
T2	PC+2	ADH	1	Fetch high order byte of Effective Address
T3	ADH, ADL	Data	0	Write internal register to memory
TØ	PC + 3	OP CODE	1	Next instruction

A.3.3 Indirect, X Addressing (6 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
TØ	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch Page Zero Base Address
T2	00, BAL	Data (Discarded)	1	
T3	00, BAL + X	ADL	1	Fetch low order byte of Effective Address
T4	00, BAL X + 1	ADH	1	Fetch high order byte of Effective Address
T5	ADH, ADL	Data	0	Write internal register to memory
TØ	PC + 2	OP CODE	1	Next instruction

A.3.4 Absolute, X or Absolute, Y addressing (5 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
TØ	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch low order byte of Base Address
T2	PC + 2	BAH	1	Fetch high order byte of Base Address
T3	ADL: BAL + Index Register ADH: BAH + C	Data (Discarded)	1	
T4	ADH, ADL	Data	0	Write internal register to memory
TØ	PC + 3	OP CODE	1	Next Instruction

A.3.5 Zero Page, X or Zero Page, Y Addressing Modes (4 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC+1	BAL	1	Fetch Page Zero Base Address
T2	00, BAL	Data (Discarded)	1	
T3	ADL: BAL + index register	Data	0	Write internal register to memory
T0	PC+2	OP CODE	1	Next instruction

A.3.6 Indirect, Y Addressing Mode (6 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC+1	IAL	1	Fetch Page Zero Indirect Address
T2	00, IAL	BAL	1	Fetch low order byte of Base Address
T3	00,IAL+1	BAH	1	Fetch high order byte of Base Address
T4	ADL: BAL + Y ADH: BAH	Data (Discarded)	1	
T5	ADH, ADL	Data	0	Write Internal Register to memory
T0	PC+2	OP CODE	1	Next Instruction

A.4 Read -- Modify -- Write Operations

ASL	LSR
DEC	ROL
INC	ROR

The -- Read -- Write operations involve the loading of operands from the operand address, modification of the operand and the resulting modified data being stored in the original location.

Note: the ROR instruction will be available on MCS650X microprocessors after June 1976

A.4.1 Zero Page Addressing (5 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
TØ	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch Page Zero Effective Address
T2	00, ADL	Data	1	Fetch Data
T3	00, ADL	Data	0	
T4	00, ADL	Modified Data	0	Write modified Data back to memory
TØ	PC + 2	OP CODE	1	Next instruction

A.4.2 Absolute Addressing (6 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
TØ	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch low order byte of Effective Address
T2	PC + 2	ADH	1	Fetch high order byte of Effective Address
T3	ADH, ADL	Data	1	
T4	ADH, ADL	Data	0	
T5	ADH, ADL	Modified Data	0	Write modified Data back into memory
TØ	PC + 3	OP CODE	1	Next instruction

A.4.3 Zero Page, X Addressing (6 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
TØ	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch Page Zero Base Address
T2	00, BAL	Data (Discarded)	1	
T3	ADL: BAL + X (without carry)	Data	1	Fetch Data
T4	ADL: BAL + X (without carry)	Data	0	
T5	ADL: BAL + X (without carry)	Modified Data	0	Write modified Data back into memory
TØ	PC + 2	OP CODE	1	Next instruction

A.4.4 Absolute, X Addressing (7 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
TØ	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch low order byte of Base Address
T2	PC + 2	BAH	1	Fetch high order byte of Base Address
T3	ADL: BAL + X ADH: BAH + C	Data (Discarded)	1	
T4	ADL: BAL + X ADH: BAH + C	Data	1	Fetch Data
T5	ADH, ADL	Data	0	
T6	ADH, ADL	Modifed Data	0	Write modified Data back into memory
TØ	PC + 3	OP CODE	1	New instruction

A.5 Miscellaneous Operations

BCC
BCS
BEQ
BMI
BNE
BPL

BRK
BVC
BVS
JMP
JSR
PHA

PHP
PLA
PLP
RTI
RTS

A.5.1 Push Operation - PHP, PHA (3 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
TØ	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	OP CODE (Discarded)	1	
T2	Stack pointer*	Data	0	Write Internal Register into Stack
TØ	PC + 1	OP CODE	1	Next instruction

*Subsequently referred to as 'Stack Ptr.'

A.5.2 Pull Operations - PLP, PLA (4 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
TØ	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	OP CODE (Discarded)	1	
T2	Stack Ptr	Data (Discarded)		
T3	Stack Ptr + 1	Data	1	Fetch Data from Stack
T0	PC + 1	OP CODE	1	Next Instruction

A.5.3 Jump to Subroutine - JSR (6 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
TØ	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch low order byte of Subroutine Address
T2	Stack Ptr	Data (Discarded)	1	
T3	Stack Ptr	PCH	0	Push high order byte of program counter to stack
T4	Stack Ptr - 1	PCL	0	Push low order byte of program counter to stack
T5	PC + 2	ADH	1	Fetch high order byte of Subroutine Address
TØ	Subroutine Address (ADH, ADL)	OP CODE	1	Next instruction

A.5.4 Break Operation - (Hardware Interrupt) - BRK (7 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch BRK OP CODE (or force BRK)
T1	PC + 1 (PC on hardware interrupt)	Data (Discarded)	1	
T2	Stack Ptr	PCH	0	Push high order byte of program counter to stack
T3	Stack Ptr - 1	PCL	0	Push low order byte of program counter to stack
T4	Stack Ptr - 2	P	0	Push Status Register to stack
T5	FFFE (NMI-FFFA) (RES-FFFC)	ADL	1	Fetch low order byte of interrupt vector
T6	FFFF (RES-FFFD)	ADH	1	Fetch high order byte of Interrupt Vector
T0	Interrupt Vector (ADH, ADL)	OP CODE	1	Next instruction

A.5.5 Return from Interrupt - RTI (6 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	Data (Discarded)	1	
T2	Stack Ptr	Data (Discarded)	1	
T3	Stack Ptr + 1	Data	1	Pull P from Stack
T4	Stack Ptr + 2	Data	1	Pull PCL from Stack
T5	Stack Ptr + 3	Data	1	Pull PCH from Stack
T0	PCH, PCL	OP CODE	1	Next instruction

A.5.6 Jump Operation-JMP

A.5.6.1 Absolute Addressing Mode (3 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch low order byte of Jump Address
T2	PC + 2	ADH	1	Fetch high order byte of Jump Address
T0	ADH, ADL	OP CODE	1	Next instruction

A.5.6.2 Indirect Addressing Mode (5 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
TØ	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	IAL	1	Fetch low order byte of Indirect Address
T2	PC + 2	IAH	1	Fetch high order byte of Indirect Address
T3	IAH, IAL	ADL	1	Fetch low order byte of Jump Address
T4	IAH, IAL + 1	ADH	1	Fetch high order byte of Jump Address
TØ	ADH, ADL	OP CODE	1	Next instruction

A.5.7 Return from Subroutine - RTS (6 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
TØ	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	Data (Discarded)	1	
T2	Stack Ptr	Data (Discarded)	1	
T3	Stack Ptr + 1	PCL	1	Pull PCL from Stack
T4	Stack Ptr + 2	PCH	1	Pull PCH from Stack
T5	PCH, PCL (from Stack)	Data (Discarded)	1	
TØ	PCH, PCL + 1	OP CODE	1	Next instruction

A.5.8 Branch Operation -- BCC, BCS, BEQ, BMI, BNE, BPL, BVC, BVS, (2, 3, or 4 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
Branch Not Taken				
TØ	PC	Op Code	1	Fetch OP CODE
T1	PC + 1	Offset	1	Fetch Branch Offset
T0	PC + 2	Next Op Code		
Branch Taken				
TØ	PC	Op Code		
T1	PC+1	Offset		
T2	PC+2	Next Op Code		Discard
TØ	PC+2+OFF	Op Code		New Branch
Branch Taken Crossing Page Boundary				
TØ	PC	Op Code		
T1	PC+1	Offset		
T2	PC+2	Op Code		Discard
T3	PC+2+OS (W/OC)			Discard
TØ	PC+2+OS (WC)			Next Instruction

L.J. Technical Systems

L.J.Technical Systems Ltd.
Francis Way,
Bowthorpe Industrial Estate,
Norwich. NR5 9JA. England.
Telephone: (0603) 748001
Telex: 975504
Fax: (0603) 746340

L.J.Technical Systems Inc.
19, Power Drive
Hauppauge, N.Y. 11788
U.S.A.
Telephone: (516) 234 2100
Fax: (516) 234 2656

Distributor

Issue Number MP117/E

Designed, Typeset and Produced By LJ Publications Dept. © 1990