

**MICROPROCESSOR FAULT
SIMULATION AND DIAGNOSIS
EXPANSION SYSTEM MA08**

EXPERIMENT MANUAL

Year	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	2100																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
Population	7.7	7.8	7.9	8.0	8.1	8.2	8.3	8.4	8.5	8.6	8.7	8.8	8.9	9.0	9.1	9.2	9.3	9.4	9.5	9.6	9.7	9.8	9.9	10.0	10.1	10.2	10.3	10.4	10.5	10.6	10.7	10.8	10.9	11.0	11.1	11.2	11.3	11.4	11.5	11.6	11.7	11.8	11.9	12.0	12.1	12.2	12.3	12.4	12.5	12.6	12.7	12.8	12.9	13.0	13.1	13.2	13.3	13.4	13.5	13.6	13.7	13.8	13.9	14.0	14.1	14.2	14.3	14.4	14.5	14.6	14.7	14.8	14.9	15.0	15.1	15.2	15.3	15.4	15.5	15.6	15.7	15.8	15.9	16.0	16.1	16.2	16.3	16.4	16.5	16.6	16.7	16.8	16.9	17.0	17.1	17.2	17.3	17.4	17.5	17.6	17.7	17.8	17.9	18.0	18.1	18.2	18.3	18.4	18.5	18.6	18.7	18.8	18.9	19.0	19.1	19.2	19.3	19.4	19.5	19.6	19.7	19.8	19.9	20.0	20.1	20.2	20.3	20.4	20.5	20.6	20.7	20.8	20.9	21.0	21.1	21.2	21.3	21.4	21.5	21.6	21.7	21.8	21.9	22.0	22.1	22.2	22.3	22.4	22.5	22.6	22.7	22.8	22.9	23.0	23.1	23.2	23.3	23.4	23.5	23.6	23.7	23.8	23.9	24.0	24.1	24.2	24.3	24.4	24.5	24.6	24.7	24.8	24.9	25.0	25.1	25.2	25.3	25.4	25.5	25.6	25.7	25.8	25.9	26.0	26.1	26.2	26.3	26.4	26.5	26.6	26.7	26.8	26.9	27.0	27.1	27.2	27.3	27.4	27.5	27.6	27.7	27.8	27.9	28.0	28.1	28.2	28.3	28.4	28.5	28.6	28.7	28.8	28.9	29.0	29.1	29.2	29.3	29.4	29.5	29.6	29.7	29.8	29.9	30.0	30.1	30.2	30.3	30.4	30.5	30.6	30.7	30.8	30.9	31.0	31.1	31.2	31.3	31.4	31.5	31.6	31.7	31.8	31.9	32.0	32.1	32.2	32.3	32.4	32.5	32.6	32.7	32.8	32.9	33.0	33.1	33.2	33.3	33.4	33.5	33.6	33.7	33.8	33.9	34.0	34.1	34.2	34.3	34.4	34.5	34.6	34.7	34.8	34.9	35.0	35.1	35.2	35.3	35.4	35.5	35.6	35.7	35.8	35.9	36.0	36.1	36.2	36.3	36.4	36.5	36.6	36.7	36.8	36.9	37.0	37.1	37.2	37.3	37.4	37.5	37.6	37.7	37.8	37.9	38.0	38.1	38.2	38.3	38.4	38.5	38.6	38.7	38.8	38.9	39.0	39.1	39.2	39.3	39.4	39.5	39.6	39.7	39.8	39.9	40.0	40.1	40.2	40.3	40.4	40.5	40.6	40.7	40.8	40.9	41.0	41.1	41.2	41.3	41.4	41.5	41.6	41.7	41.8	41.9	42.0	42.1	42.2	42.3	42.4	42.5	42.6	42.7	42.8	42.9	43.0	43.1	43.2	43.3	43.4	43.5	43.6	43.7	43.8	43.9	44.0	44.1	44.2	44.3	44.4	44.5	44.6	44.7	44.8	44.9	45.0	45.1	45.2	45.3	45.4	45.5	45.6	45.7	45.8	45.9	46.0	46.1	46.2	46.3	46.4	46.5	46.6	46.7	46.8	46.9	47.0	47.1	47.2	47.3	47.4	47.5	47.6	47.7	47.8	47.9	48.0	48.1	48.2	48.3	48.4	48.5	48.6	48.7	48.8	48.9	49.0	49.1	49.2	49.3	49.4	49.5	49.6	49.7	49.8	49.9	50.0	50.1	50.2	50.3	50.4	50.5	50.6	50.7	50.8	50.9	51.0	51.1	51.2	51.3	51.4	51.5	51.6	51.7	51.8	51.9	52.0	52.1	52.2	52.3	52.4	52.5	52.6	52.7	52.8	52.9	53.0	53.1	53.2	53.3	53.4	53.5	53.6	53.7	53.8	53.9	54.0	54.1	54.2	54.3	54.4	54.5	54.6	54.7	54.8	54.9	55.0	55.1	55.2	55.3	55.4	55.5	55.6	55.7	55.8	55.9	56.0	56.1	56.2	56.3	56.4	56.5	56.6	56.7	56.8	56.9	57.0	57.1	57.2	57.3	57.4	57.5	57.6	57.7	57.8	57.9	58.0	58.1	58.2	58.3	58.4	58.5	58.6	58.7	58.8	58.9	59.0	59.1	59.2	59.3	59.4	59.5	59.6	59.7	59.8	59.9	60.0	60.1	60.2	60.3	60.4	60.5	60.6	60.7	60.8	60.9	61.0	61.1	61.2	61.3	61.4	61.5	61.6	61.7	61.8	61.9	62.0	62.1	62.2	62.3	62.4	62.5	62.6	62.7	62.8	62.9	63.0	63.1	63.2	63.3	63.4	63.5	63.6	63.7	63.8	63.9	64.0	64.1	64.2	64.3	64.4	64.5	64.6	64.7	64.8	64.9	65.0	65.1	65.2	65.3	65.4	65.5	65.6	65.7	65.8	65.9	66.0	66.1	66.2	66.3	66.4	66.5	66.6	66.7	66.8	66.9	67.0	67.1	67.2	67.3	67.4	67.5	67.6	67.7	67.8	67.9	68.0	68.1	68.2	68.3	68.4	68.5	68.6	68.7	68.8	68.9	69.0	69.1	69.2	69.3	69.4	69.5	69.6	69.7	69.8	69.9	70.0	70.1	70.2	70.3	70.4	70.5	70.6	70.7	70.8	70.9	71.0	71.1	71.2	71.3	71.4	71.5	71.6	71.7	71.8	71.9	72.0	72.1	72.2	72.3	72.4	72.5	72.6	72.7	72.8	72.9	73.0	73.1	73.2	73.3	73.4	73.5	73.6	73.7	73.8	73.9	74.0	74.1	74.2	74.3	74.4	74.5	74.6	74.7	74.8	74.9	75.0	75.1	75.2	75.3	75.4	75.5	75.6	75.7	75.8	75.9	76.0	76.1	76.2	76.3	76.4	76.5	76.6	76.7	76.8	76.9	77.0	77.1	77.2	77.3	77.4	77.5	77.6	77.7	77.8	77.9	78.0	78.1	78.2	78.3	78.4	78.5	78.6	78.7	78.8	78.9	79.0	79.1	79.2	79.3	79.4	79.5	79.6	79.7	79.8	79.9	80.0	80.1	80.2	80.3	80.4	80.5	80.6	80.7	80.8	80.9	81.0	81.1	81.2	81.3	81.4	81.5	81.6	81.7	81.8	81.9	82.0	82.1	82.2	82.3	82.4	82.5	82.6	82.7	82.8	82.9	83.0	83.1	83.2	83.3	83.4	83.5	83.6	83.7	83.8	83.9	84.0	84.1	84.2	84.3	84.4	84.5	84.6	84.7	84.8	84.9	85.0	85.1	85.2	85.3	85.4	85.5	85.6	85.7	85.8	85.9	86.0	86.1	86.2	86.3	86.4	86.5	86.6	86.7	86.8	86.9	87.0	87.1	87.2	87.3	87.4	87.5	87.6	87.7	87.8	87.9	88.0	88.1	88.2	88.3	88.4	88.5	88.6	88.7	88.8	88.9	89.0	89.1	89.2	89.3	89.4	89.5	89.6	89.7	89.8	89.9	90.0	90.1	90.2	90.3	90.4	90.5	90.6	90.7	90.8	90.9	91.0	91.1	91.2	91.3	91.4	91.5	91.6	91.7	91.8	91.9	92.0	92.1	92.2	92.3	92.4	92.5	92.6	92.7	92.8	92.9	93.0	93.1	93.2	93.3	93.4	93.5	93.6	93.7	93.8	93.9	94.0	94.1	94.2	94.3	94.4	94.5	94.6	94.7	94.8	94.9	95.0	95.1	95.2	95.3	95.4	95.5	95.6	95.7	95.8	95.9	96.0	96.1	96.2	96.3	96.4	96.5	96.6	96.7	96.8	96.9	97.0	97.1	97.2	97.3	97.4	97.5	97.6	97.7	97.8	97.9	98.0	98.1	98.2	98.3	98.4	98.5	98.6	98.7	98.8	98.9	99.0	99.1	99.2	99.3	99.4	99.5	99.6	99.7	99.8	99.9	100.0

CONTENTS

Chapter 1	Characteristics of Logic
Chapter 2	Microprocessor Signal Lines
Chapter 3	Basic Microprocessor Test Equipment
Chapter 4	Investigation of a Working System
Chapter 5	Fault and Diagnostic Testing
Chapter 6	Address Decode System
Chapter 7	System Expansion
Chapter 8	System Interfaces
Chapter 9	Video Interfaces
Chapter 10	Miscellaneous Circuits
Chapter 11	Fault and Diagnostic Testing (VISA)
Appendix 1	Summary of Single Cycle Executions
Appendix 2	Address Decode PROM'S
Appendix 3	Schematic Diagrams

Introduction

This manual seeks to introduce the reader to the Hardware of a typical Microcomputer System. Having explored the Hardware by means of exploratory exercises the reader is expected to be able to Test the System and diagnose faults which are introduced to an otherwise working system by means of either a switch fault unit which may be connected to this basic module or through switches included on the expansion module.

The system used for these exercises is the EMMA Microprocessor and the VISA Expansion Module.

Objectives are set for each chapter and a short self test included at the end of each chapter.

It is strongly recommended that the reader completes each chapter and its practical exercises in a chronological order.

The reader is also advised to make full use of the EMMA/VISA Software and Technical Manuals since it contains data sheets of the various Integrated Circuits which will in themselves provide very valuable information.

CHARACTERISTICS OF LOGIC

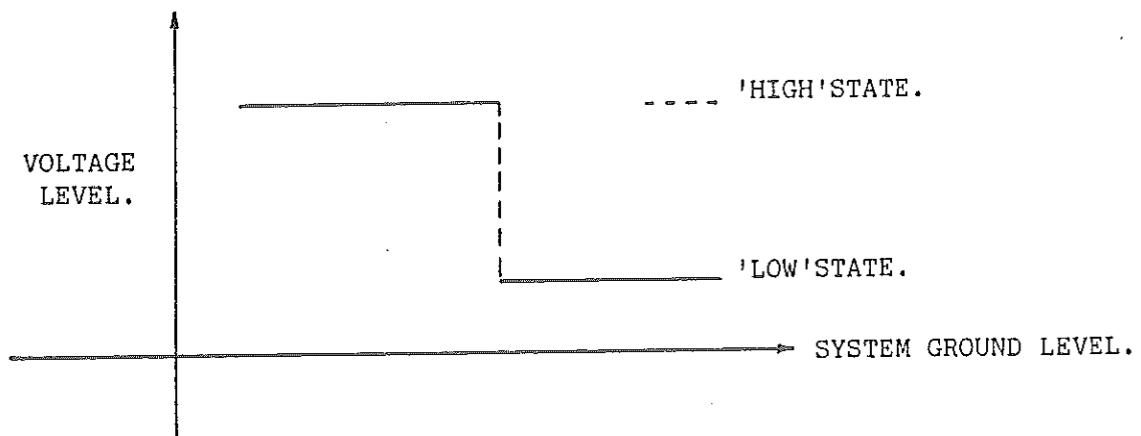
Objectives of this Chapter

Having studied this chapter you should:

- Know TTL Voltage Levels.
- Be familiar with TTL 'Noise' Tolerances.
- Be able to differentiate between different TTL Sub-Families.
- Understand TTL Output Structures such as Totem Pole, Open Collector and Three-State.
- Be conversant with the Working Rules for TTL and MOS Integrated Circuits.

The inputs and outputs of digital devices are represented by two discrete voltage levels or states, which are often referred to as either 'high' or 'low' with respect to each other.

Example:



We also refer to the high state as a 'logical - one' (1) and the low state as a 'logical - zero' (0).

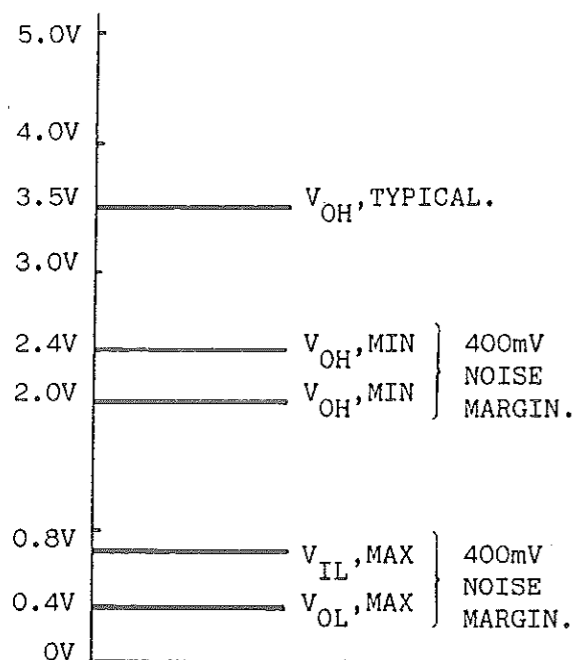
This may well be all we need to know for descriptive purposes. However, for test and fault diagnosis, we will probably be required to predict the actual 'in circuit' input/output levels of any given device within our particular system design. This will involve an intimate knowledge of the system configuration and at least the facility to consult the individual device Manufacturers Specifications.

Numerous logic families are in existence with the most common being Transistor, Transistor Logic (TTL). Indeed you will notice from Manufacturers Data Sheets that many devices are referred to as T.T.L. Compatible. With this in mind we will define the voltage levels associated with the two logic states previously mentioned.

● TTL VOLTAGE LEVELS

• Logic '1'	+2.0V	to	+5.5V
• Logic '0'	+0.4V	to	+0.8V
• UNDEFINED	+0.8V	to	+2.0V

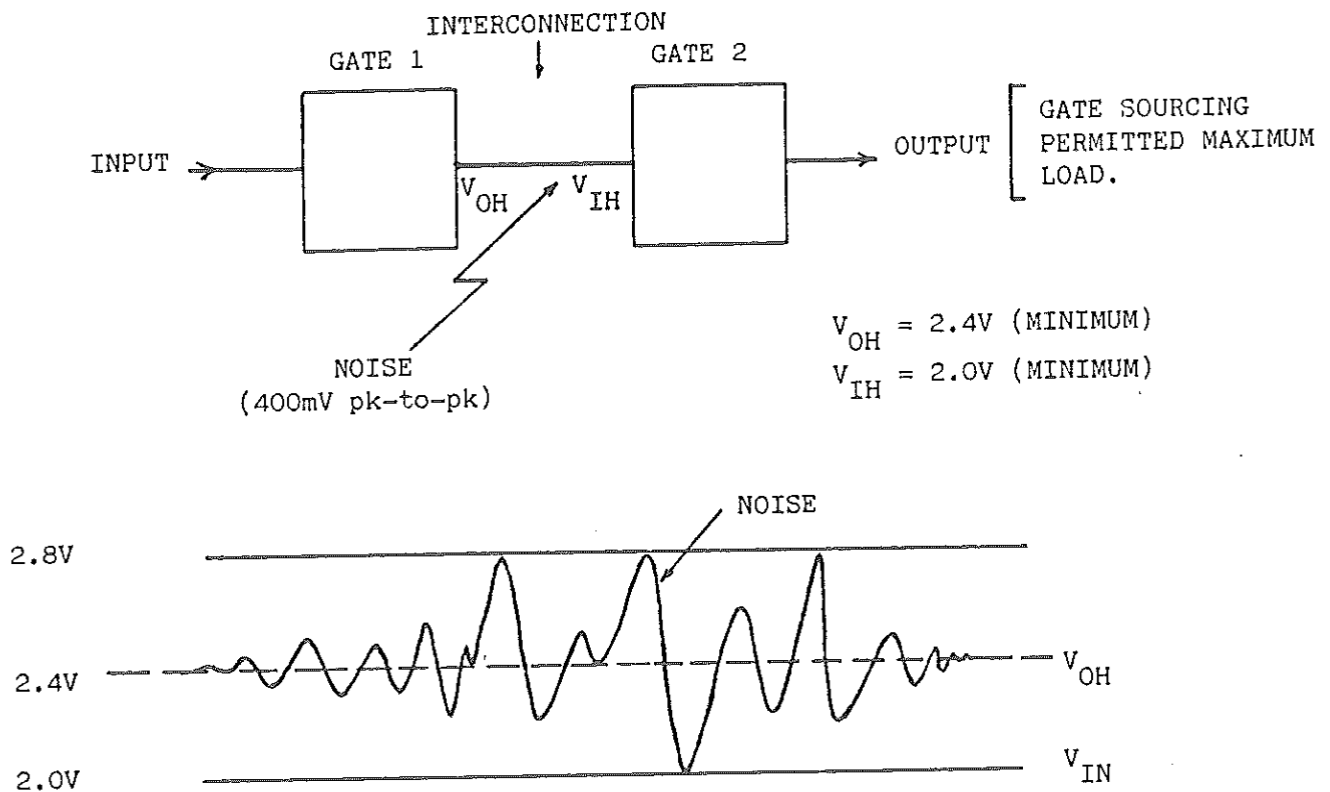
Operationally we need to refine this still further and the diagram below presents a more realistic picture:



TTL VOLTAGE LEVELS

In this diagram the respective levels refer to the input and output of a logic gate. The suffixes IH and OL refer to the 'Input-high' and 'Output-low' states respectively. If this specification is met, then a 400mV peak-to-peak noise level induced in an inter-gate connection should not cause malfunctioning of the receiving gate.

Example:

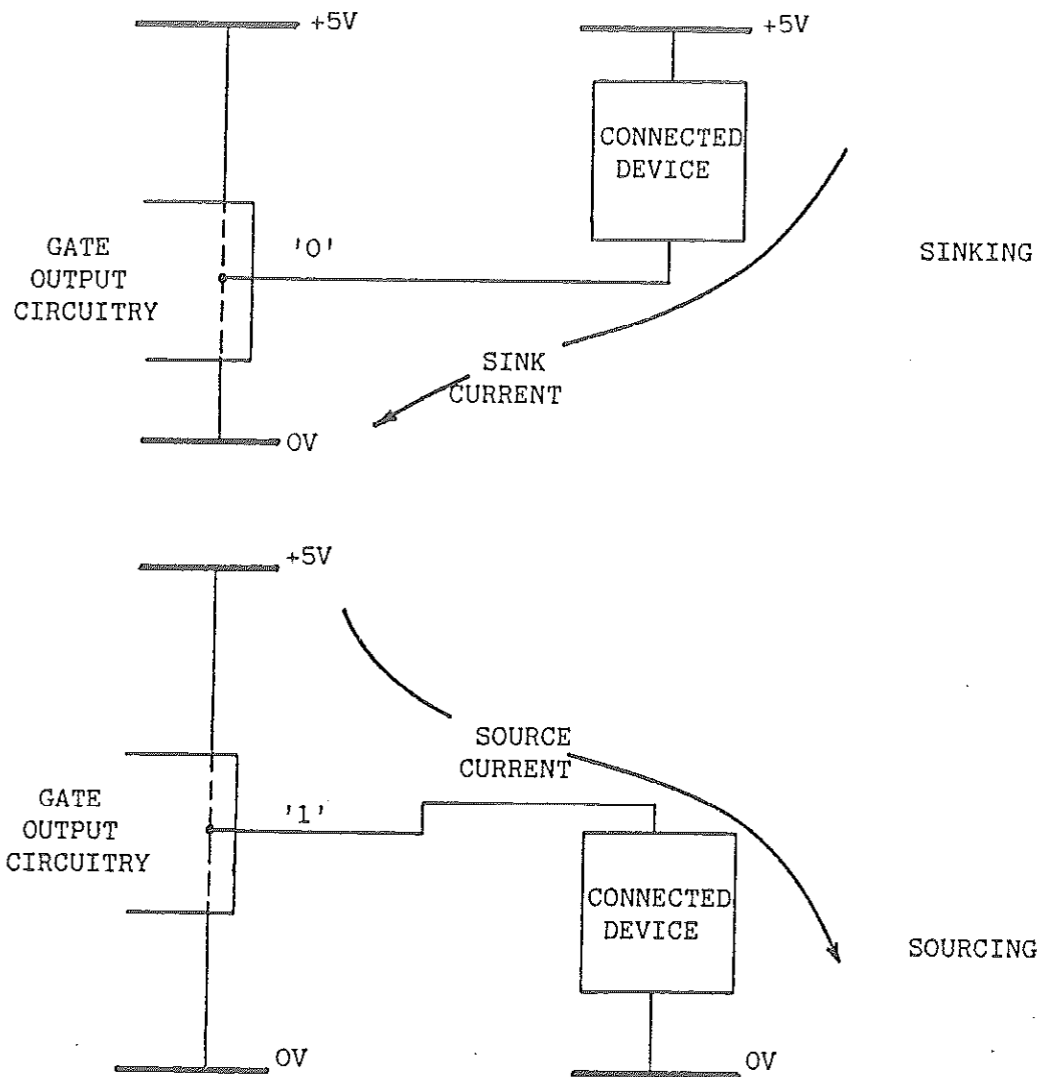


The same tolerance to noise is inherent at the logic '0' level.

● Gate Source and Sink Capability

In addition to voltage requirements, each device will possess a current input and output specification. We refer to a device being capable of 'Sourcing' current to another device(s) when at logical '1' and 'Sinking' current from another device when at logical '0'.

Example:



A very common Logic Family is the 74. This designatory number would be followed by a two or three digit number to represent a unique device within the 74 family. Example, 7400 refers to an Integrated Circuit (IC) which contains four (quad)-2 Input NAND Gates.

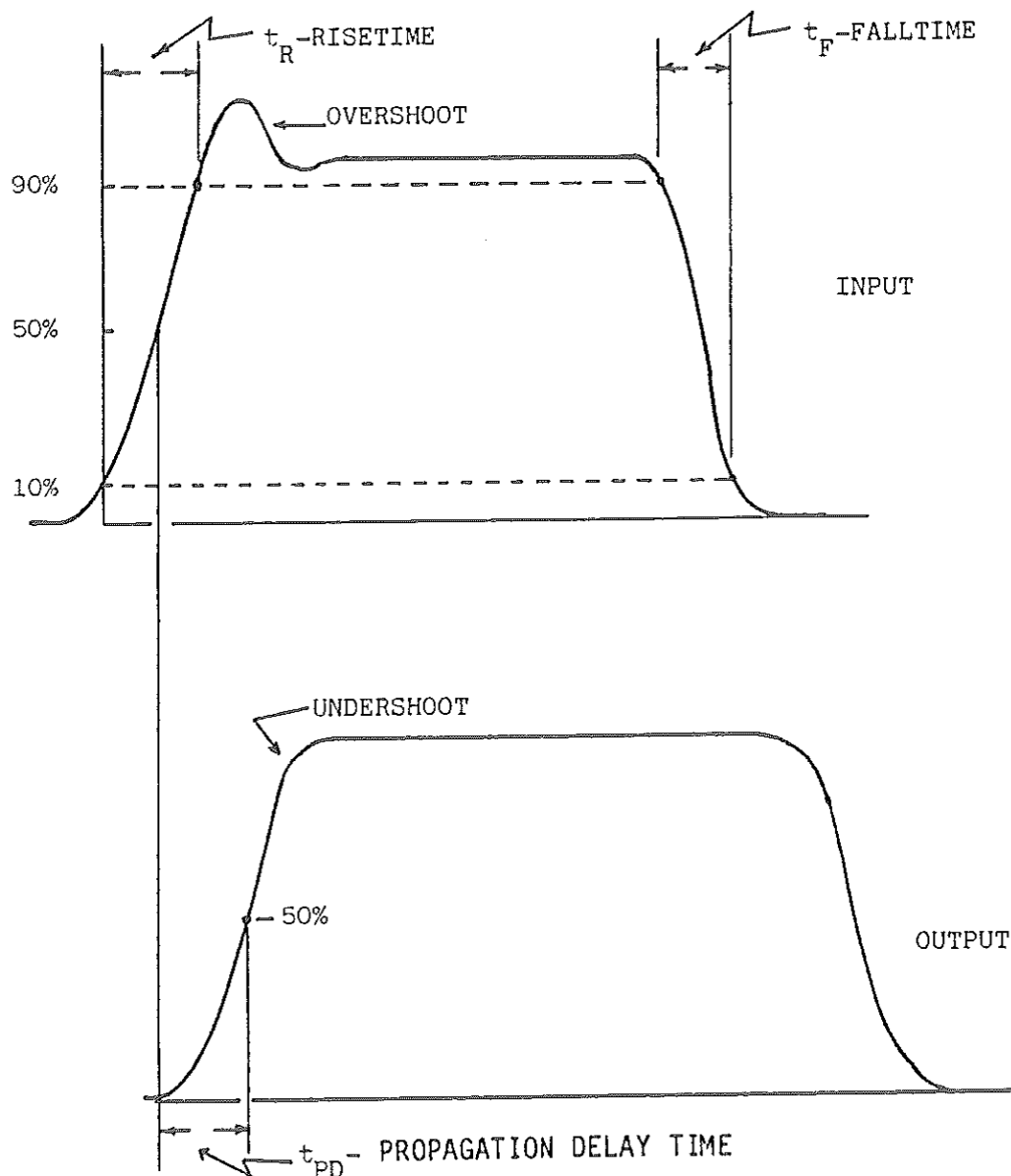
Within the 74 family are a number of Sub-Families each developed to meet a particular requirement.

We can tabulate these as:

- 74 Standard
- 74H High Power
- 74L Low Power
- 74S Schottky
- 74LS Lower Power Schottky

Each device has a parameter termed its 'Propagation Delay'. This refers to the time taken for an input pulse to change the state of the device output. It is measured from the 50% point respectively of the input and output pulses.

Example:



The diagrams also specifies other parameters of a 'Pulse' which are frequently used in design specifications.

To decrease the propagation delay (increase speed of operation) more power must be used, hence the 74H series. The premium is of course a higher chip power dissipation. The 74L takes advantage of lower power dissipation at the expense of increased propagation delay. The 74S series takes advantage of newer technology, uses less power and provides more speed than the 74H series. The 74H is now little used. The 74LS optimises on both power and speed and is in fact an IC much favoured by designers.

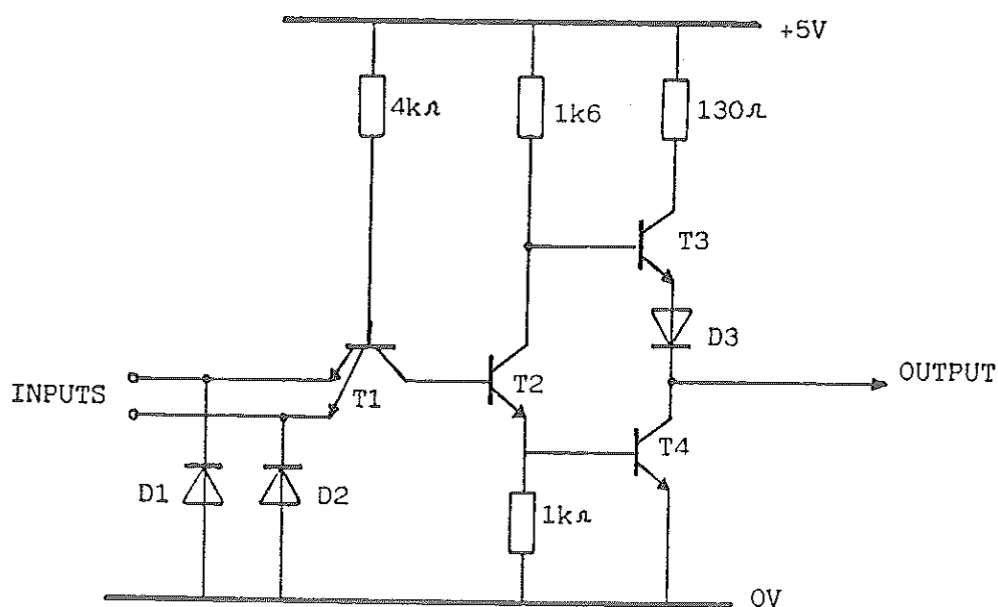
We often use the product of power/gate and propagation delay to compare different logic families.

Most of the logic IC's used on the equipment in this set are 74LS therefore it is to this that we will confine the rest of our discussions.

● Device Input and Output Structures

● Totem Pole Output.

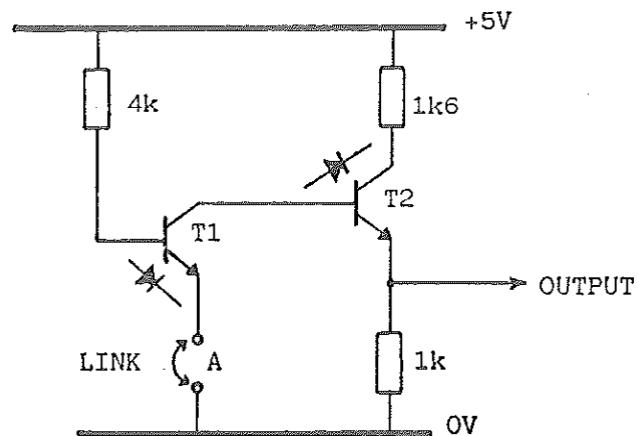
We will consider a standard TTL NAND gate the circuit of which is shown below:



Lets start at the input A and temporarily ignore input B. The two diodes D1 and D2 we will also consider later.

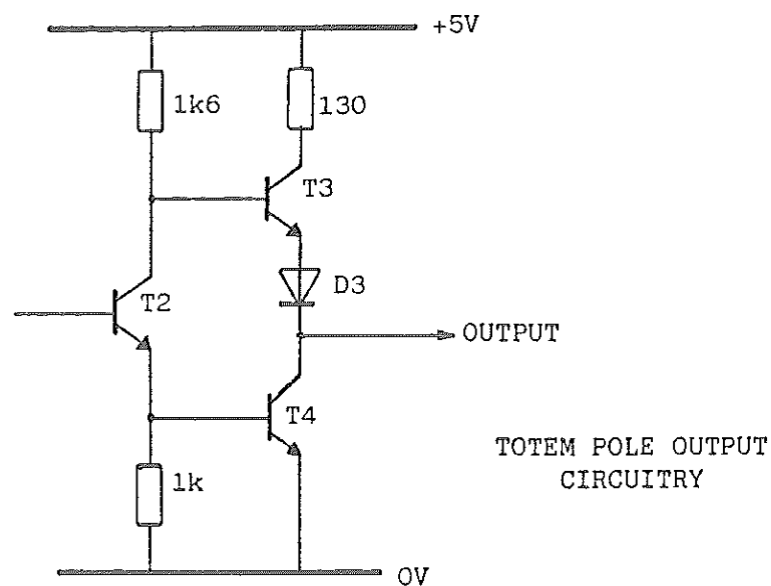
First we will ground input A.

Transistor T1 should be seen as having no collector source current by virtue of the base/collector junction of T2.



The effect will be for T1 to conduct due to a base current of approximately 1.1mA. This base current will be sufficient for T1 to saturate with T1, V_{ce} at approximately 0.1 volts. T2 will thus be turned off due to V_b being negative to V_c .

Now lets consider the effect of this upon the output circuitry. The output arrangement is termed 'Totem Pole' by virtue of the fact that one transistor 'sits' upon the top of another.



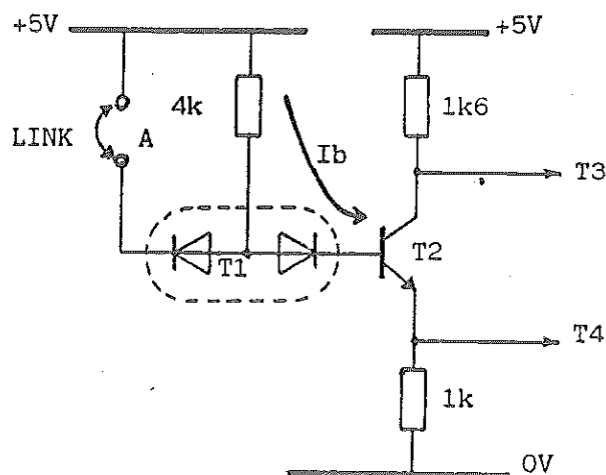
Now, with T2 OFF, and hence high impedance, the bases of T3 and T4 will be pulled high and low respectively. T3 will thus be turned ON and T4 turned OFF. The effect is to pull the gate output HIGH.

A standard TTL gate output is designed to source 10 standard TTL gate inputs (loads). We refer to this as a FAN-OUT of ten. Each output will require a source current of $40\mu\text{A}$. A standard TTL gate therefore must be capable of sourcing $400\mu\text{A}$. The base current to T3 required to provide this current is small resulting in a very small voltage drop across the $1.6\text{k}\Omega$ resistor. Assuming therefore a voltage drop of 0.7 volts across both D3 and T3, base emitter junction, the output voltage will be no more than 3.6 volts. This value is much higher than the legal minimum for a logic '1'.

Now lets again ignor input B and take input A high.

No path exists for T1 base to emitter current, hence T1 will be turned off. However, an ohmic view of T1 will show that there is a current path to the base of T2.

Example:



T2 will therefore be turned ON where previously it was OFF.

The emitter of T2 will now provide sufficient base current to turn the previously OFF T4, ON so pulling the output down to logical '0'. T3 will also be turned OFF and held in that condition by virtue of the fact that its base voltage will settle at approximately 0.8V ($T4 - V_{be} = 0.7V$ plus $T2 - V_{ce.sat} = 0.1V$). T3 base voltage (0.8V) is considerably less than the 1.4V ($T3 - V_{be} = 0.7V$ plus $V_{d3} = 0.7V$ required to maintain conduction).

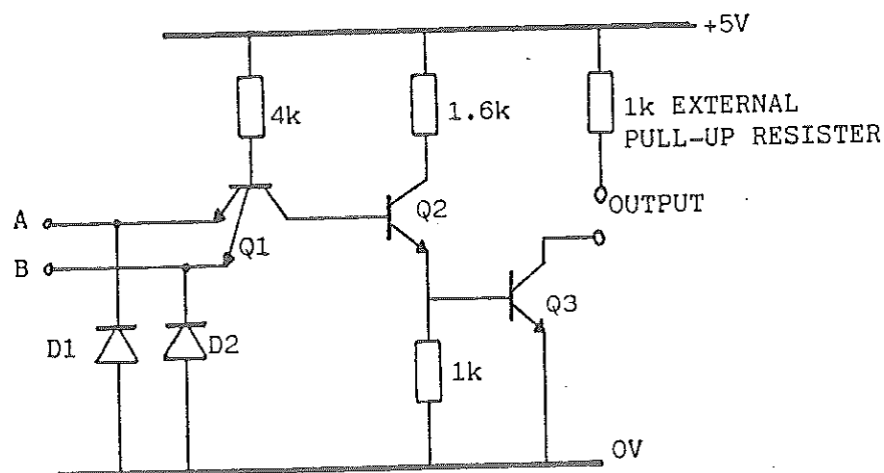
A similar line of reasoning for different combinations of A and B inputs will show the circuit to represent a two-input NAND.

Now let's turn our attention to the 130Ω resistor and diode D3 in the output circuit. The inclusion of D3 ensures that T3 and T4 do not conduct simultaneously. Some overlap may occur since the time taken to turn a transistor OFF is greater than that to turn it ON. The 130Ω resistor will limit this current which will occur during output level changes. Nevertheless large current peaks will be impressed upon the supply system during gate output level changes. These may cause considerable trouble unless inter-device decoupling capacitors are used.

We will consider these later.

● Open Collector Output

One of the problems with totem pole outputs is that if two outputs were connected together then a short circuit condition effectively exists when one gate tries to output 'high' and the other 'low'. One or other of the devices will inevitably be 'burned-out'. Since we frequently desire to make such a connection, manufacturers have designed IC's which have an output circuit configuration termed 'Open-Collector'. The arrangement is shown below:



You will notice that the upper totem pole transistor and associated diode are missing. Indeed the output circuit has to be completed by means of an external 'PULL-UP' resistor to ensure that the gate output is always in a known state and not 'floating'. We refer to this method of 'pull-up' as being 'passive' whereas the totem-pole arrangement gives 'active' pull-up. We will leave it to you to prove that no damage will result from one gate output going low and the other gate output going high. Take notice too that a low output gate will always pull the combined arrangement low. We refer to the method of connection as being 'Wired-NOR'.

● **Three State Output**

Three-State or TRI-State outputs have the normal high and low states plus a 'high-impedence' state. This third state is created by turning both output transistors OFF by means of a separate 'ENABLE' input. The advantage of three-state outputs is that many device outputs can be connected together using a common bus arrangement. Only one sourcing device is 'enabled' at one time all other device outputs being disabled, hence no conflict takes place. The advantage over open-collector outputs is of course the more rapid pull-up that can be achieved. A disadvantage is the extra input required to enable the device.

● **Metal Oxide Semi Conductor Devices**

TTL devices require a large chip area and have a relatively high power dissipation. This precludes using this technology for large-scale integration (LSI). MOS technology overcomes those problems, hence its use for larger circuit configurations such as microprocessor chips.

Many of these devices are termed TTL compatible but, a wide variety of supply voltages are in use and in consequence a similar range of input/output voltage levels. Fortunately, semi-conductor manufacturers now concentrate on making MOS devices operate on a single +5V supply. The LSI components you will meet using this equipment fall into this later category.

We are not going to embark upon a discussion of the operation of MOS devices as we have with TTL. However, we will make a comparison sufficient for our purposes.

● **Logic Technology Comparison**

The input of MOS transistors (TTL use bipolar transistors) is electrically isolated from the rest of the device. This creates handling problems due to static electricity. Protective diodes within the device help but care should still be taken.

The high input impedance draws approximately 10pA in either logic state and presents a capacitance of 5-7pF to a sourcing device.

CMOS devices possess much larger and variable propagation delay times than TTL. This can cause timing problems in some circuits.

Power dissipation is the main advantage of CMOS when operating at low frequencies. Above 1MHz these advantages begin to cancel out.

CMOS current sinking capability does not make it totally compatible with TTL. For example, a CMOS device operating on a 5V supply can only sink 0.4mA at an output low of 0.4 Volts. This is insufficient to provide the 16mA requirement of a standard TTL gate logic '0' input.

We will terminate our discussions on logic by providing a few rules for working with the different technologies.

● **Integrated Circuit Working Rules**

● **TTL**

1. Tie unused inputs to ground or V.cc with a 1k resistor.
2. The outputs of TTL gates should not be connected together.
Exceptions to this rule are open-collector and three-state TTL.
3. Maximum V.cc is 7V.
4. The maximum signal input is 5.5V at a V.cc of 5V.

5. There is a maximum fan out of 10 gates within each TTL family.
6. An $0.1\mu\text{F}$ or $0.01\mu\text{F}$ by-pass capacitor should be installed between V_{cc} and ground next to each IC.
7. Open connecting leads should not be greater than 12 to 14 inches for standard TTL and 3 to 4 inches for Schottky TTL.

● MOS

1. Integrated circuits should be stored in conductive foam (usually dull black in colour) or aluminium foil. Never put the chips in white styrofoam which is an excellent static generator.
2. Ground to an earth ground your soldering-iron tip, your metal bench and the chassis of any test equipment used.
3. In a low-humidity, high-static condition, connect your wrist to earth ground with a length of wire and a 1M series resistor. The resistor prevents you from getting a fatal shock if you touch a voltage source, but allows static to drain off.
4. As with all integrated circuits do not remove or insert MOS ICs with the power on.
5. Do not apply signals to an input with the circuit power off. With no V_{cc} present, a low-output impedance signal generator may forward bias the protection diodes and force enough current through to destroy them.
6. Connect all unused input leads to V_{cc} or V_{dd} whichever keeps the desired logic for the gate, never leave them open because open inputs accumulate static and float to unpredictable logic levels.
7. Put resistors in series with any MOS inputs connected to points off a printed circuit board.

● CMOS

1. Observe static electricity precautions as with any MOS logic.
2. Make sure all unused inputs are tied to V.cc or ground.
3. Turn off low-impedance signal sources at the same time, or before turning off CMOS circuits.
4. Do not attempt to drive standard TTL with a standard CMOS output.
5. Do not tie two CMOS outputs together unless they are both three-state and then make sure they are not both enabled at the same time.
6. Minimise capacitive loading on outputs to keep Propagation Delay (t_{pd}) as short as possible.
7. In higher frequency circuits watch for timing problems caused by the relatively long t_{pd} for CMOS.

POST TEST

Chapter 1

- 1.1 Define three distinct voltage level found in TTL devices.
- 1.2 Explain why TTL devices have a higher minimum output voltage to the minimum input voltage.
- 1.3 Explain what is meant by the terms 'SINK' and 'SOURCE' when applied to TTL device output circuitry.
- 1.4 Explain the advantages of 74LS over other 74 Sub-Families.
- 1.5 Describe each of the following parameters applicable to 'PULSES'.
 - t_R
 - t_{PD}
 - Overshoot
- 1.6 Explain the term 'FAN-OUT'.
- 1.7 Discuss the advantages/disadvantages of the following output configurations;
 - Totem Pole, Open Collector and Three-State.

MICROPROCESSOR SIGNAL LINES

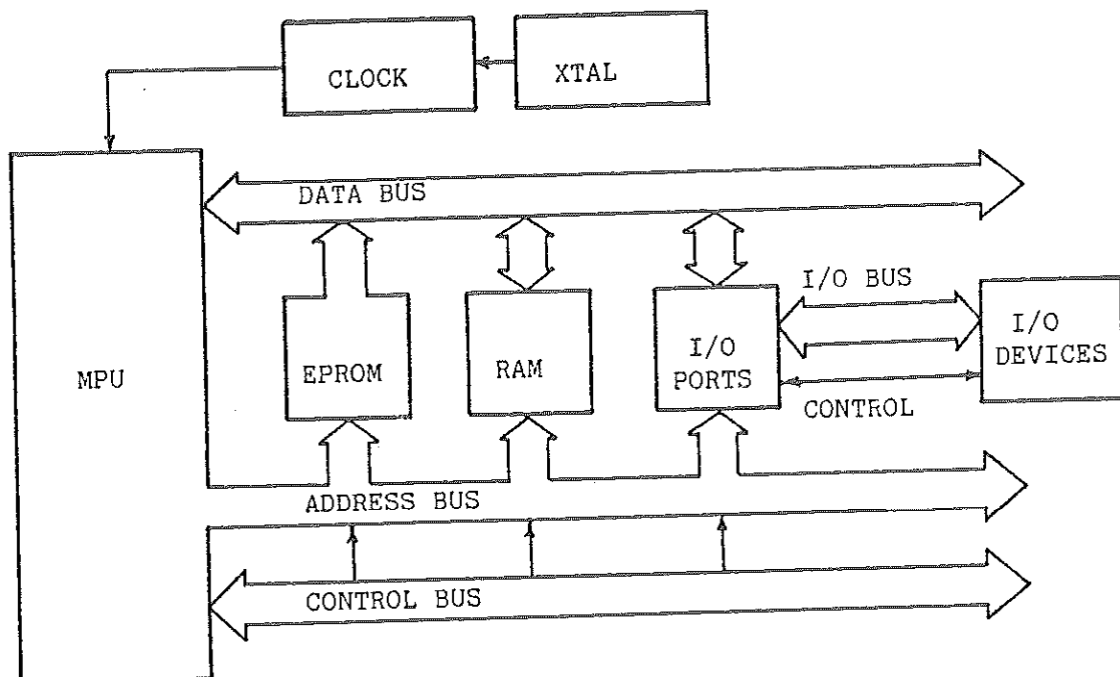
Objectives of this Chapter:

Having studied this chapter you should:

- Understand the concept of a System Bus.
- Be able to produce a Simple Timing diagram.
- Be able to monitor Control Line Signals upon a Microcomputer System using the oscilloscope.
- Write a Simple Program to 'exercise' an I/O Port for test purposes.

Introduction

Before actually considering any signals which may be generated by, or input to the microprocessor, we need to relate the processor to the rest of the system of which it forms part. A diagram which shows us the 'Architecture' of the system will do this for us. A typical generalised system architecture would be:



TYPICAL SYSTEM ARCHITECTURE

From the diagram we can see that the microprocessor chip has a number of input lines and output lines. Some lines are in fact used for both input and output. In some cases where the functions are similar, a number of lines will be grouped together. The grouping we term a BUS.

We can easily identify both lines and buses together with the directions in which the signals pass.

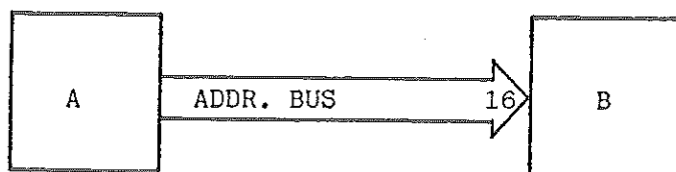
Example:

● Single Signal Line



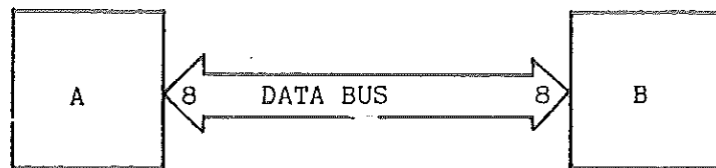
A single line indicates signal path between devices A and B. Arrow head shows direction of signal.

● Signal Bus (Uni-directional)



In this case a number of signal paths are grouped together by function to form a bus. The designation of the bus is indicated as is the number of signal paths forming the bus. We term this bus a 'Uni-directional' bus since the signal paths are always in the same direction.

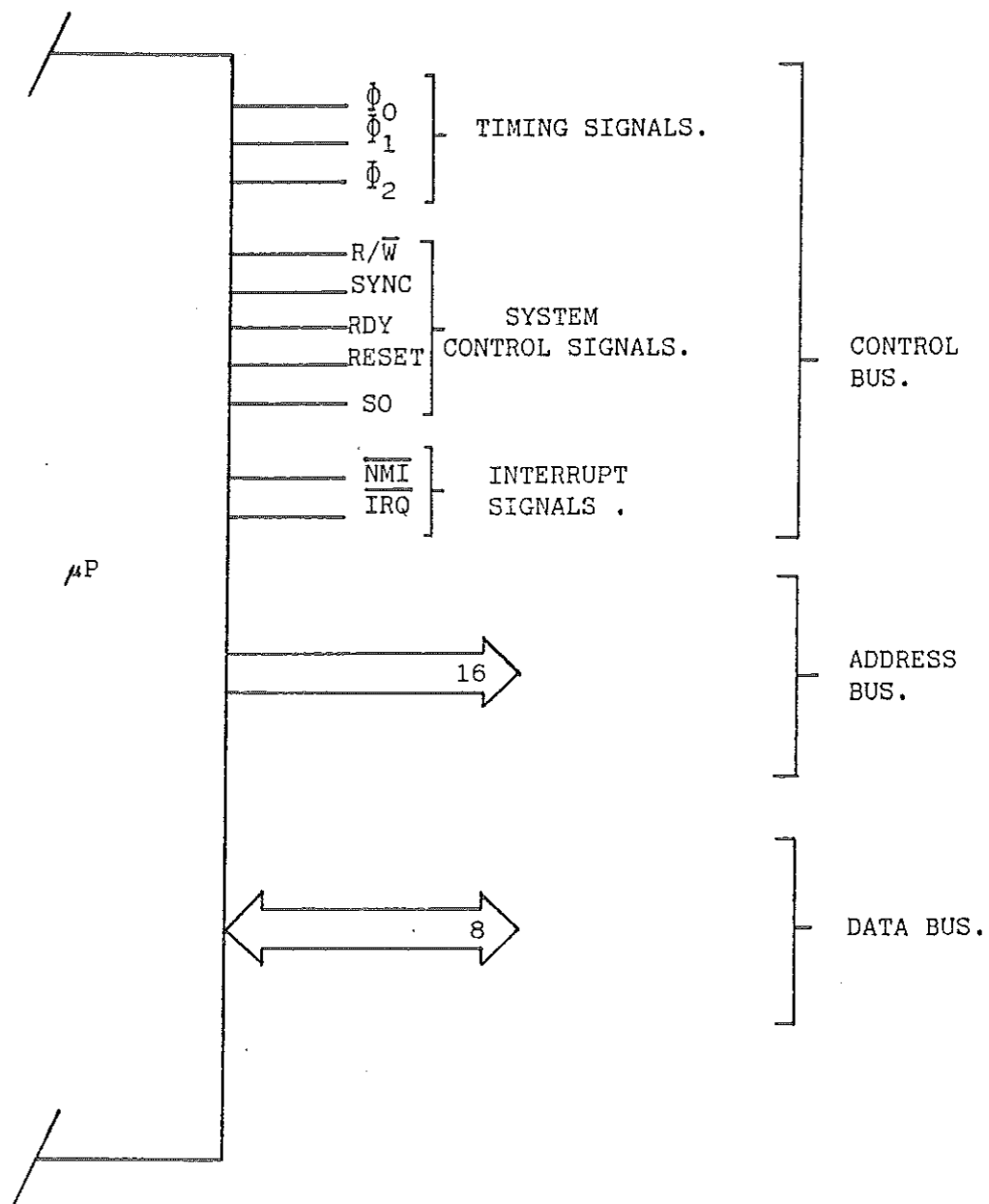
● Signal Bus (Bi-directional)



Arrow heads terminating both ends of the bus indicate its bi-directional nature.

● The Microprocessor Signal Line

From the system architecture diagram we can identify basically three groups of signals, Address Lines, Data Lines, and Control Lines. The diagram overleaf indicates each of the signals passing from or too the microprocessor. We will use this diagram to discuss each signal in turn.



● The Control Bus

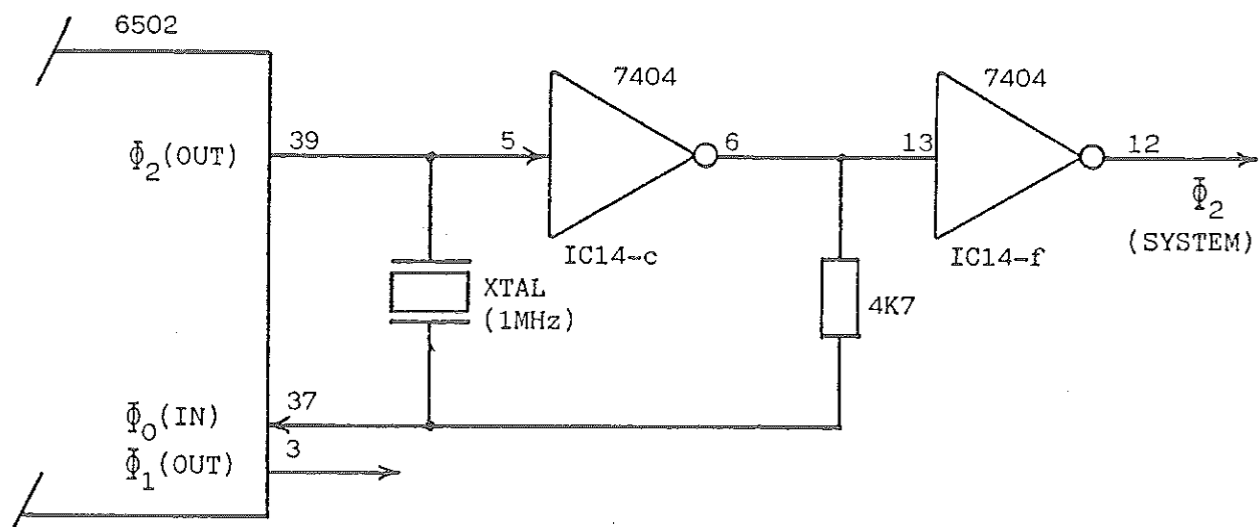
The Control Bus is a 'loose' grouping of signals which are in the main, generated by the microprocessor. They are used to control the whole of the microcomputer configuration. It is for this reason that we often refer to systems which employ a microprocessor as 'Microprocessor Based'.

We will consider each of these control signals in turn and take the opportunity to 'look' at them using an oscilloscope.

● Timing Signals

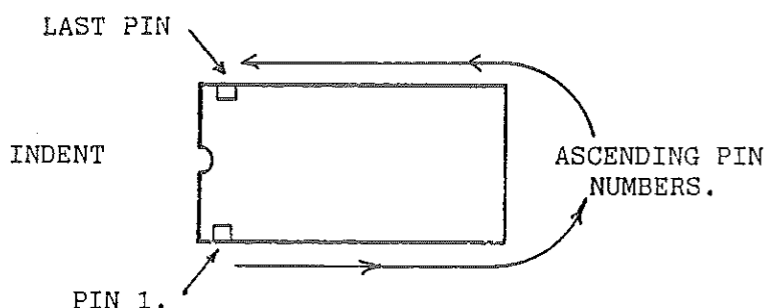
A microcomputer based system is termed a 'sequential' system and as much relies upon 'clock' signals to ensure that the numerous components maintain synchronism with each other. Three distinct clock signals can be identified at the 6502 chip. These are Phase Zero (Φ_0), Phase One (Φ_1) and Phase Two (Φ_2). Φ_0 is an input signal. The actual clock circuitry is combined within the chip on the 6502 but the system 'frequency' is determined by external circuitry employing a crystal for accurate and stable system timing. Φ_1 and Φ_2 signals are both outputs generated by the on-board clock circuitry. These signals are in anti-phase or complementary and hence non-overlapping.

The circuitry external to the 6502 is shown below:



The various numbers shown next to a device and above a signal line refer to the pin number of the device. Generally devices are numbered to a standard such that although the actual pin numbers are not shown on the chip we can identify pin 1 and hence all the others. Chip 'end' identification is usually by means of a small indentation.

Example:

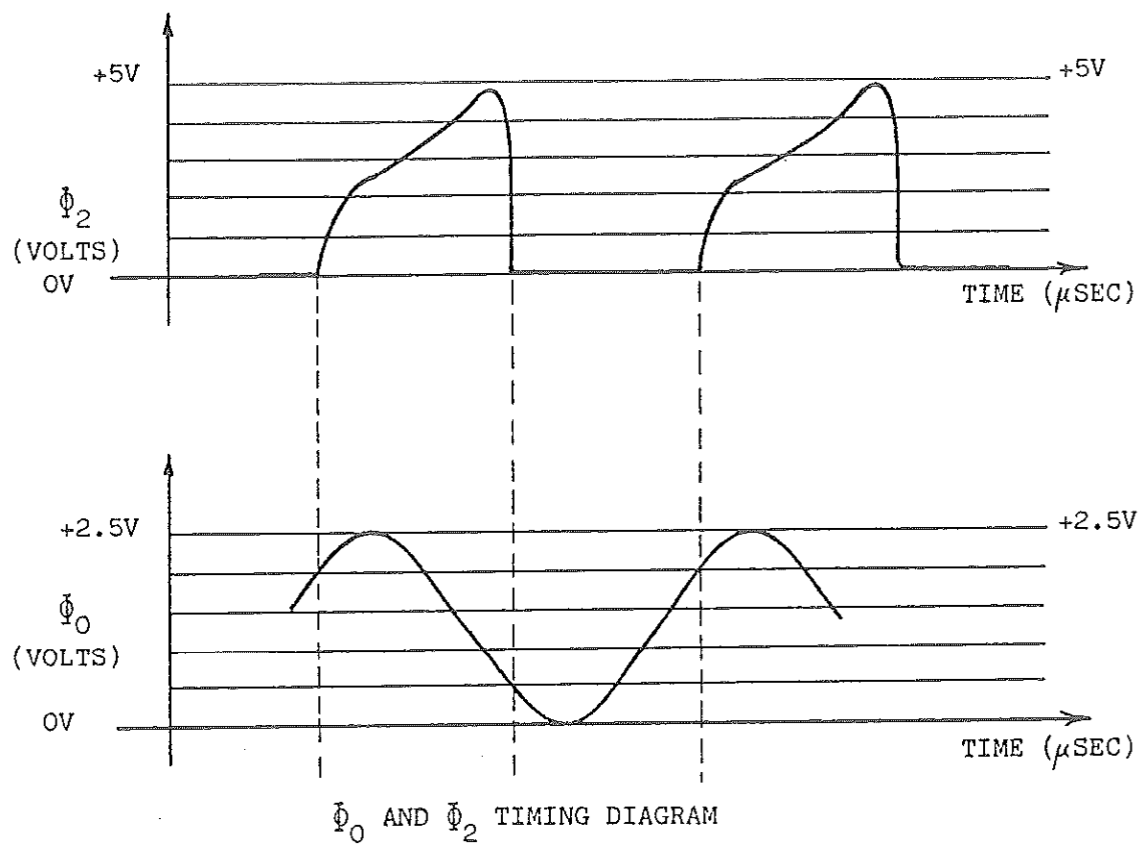


The number above the device, example 7404 indicates the type of device and the number below, example IC14-C, the chip number on the particular printed-circuit-board (PCB). See Appendix for chip lay-out diagram. Now proceed with the following exercises.

● EXERCISE 2.1

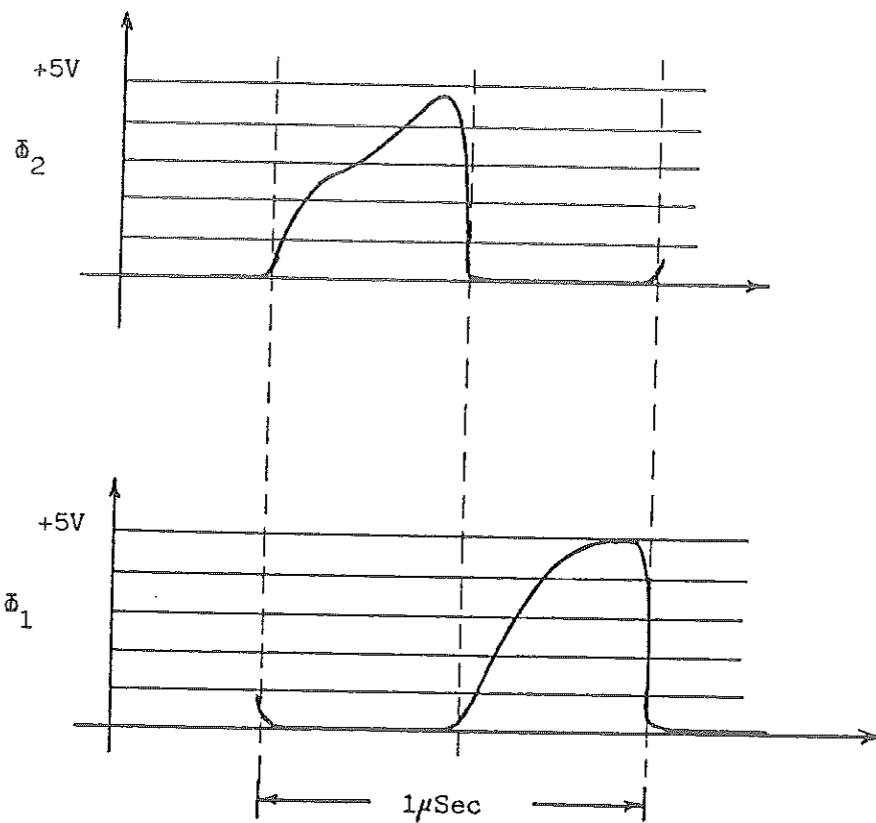
- Set up an EMMA and an oscilloscope. If you are not sure about using the oscilloscope see appendix 'Use of the Oscilloscope'.
- RESET EMMA to obtain the familiar row of dots on the seven-segment display.
- Adjust the oscilloscope to monitor a 1Megahertz signal at approximately 5 volts.
- Simultaneously monitor the ~~40~~ and ~~42~~ signals at the 6502 chip pins. (Use pulse shrouds to prevent accidental short circuit of two adjacent pins).

- Check that the waveforms are similar to those below:

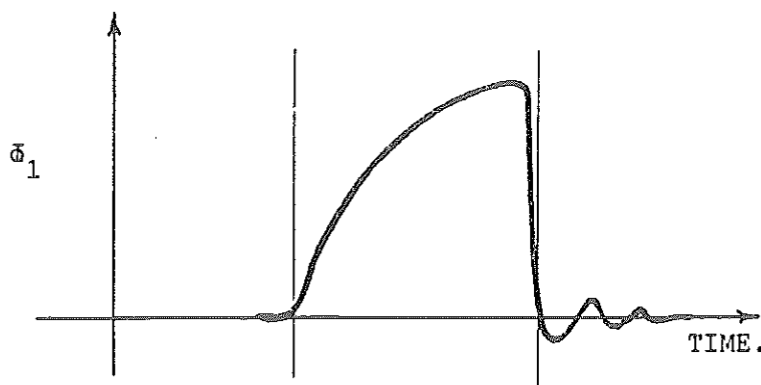


● EXERCISE 2.2

- Simultaneously monitor the Φ_1 and Φ_2 signals, again the 6502 pins.
- Check that the waveforms are similar to those overleaf:



- Take special notice of the non-overlapping characteristic of these two waveforms. A decreased time-base setting will help examine the cross-over points.
- Measure the periodic time and hence determine the period of oscillation.
- In taking your measurements it is assumed that the oscilloscope probe screen (outer sheath) have both been taken to the same system zero voltage line.
- Remove one probe screen and observe the effect upon the associated waveform. You should find that it approximates to that below:



We term this 'ringing'. You should notice that ringing only occurs when the wave shape changes in a 'step' fashion. The actual amount present depends upon the characteristic of the oscilloscope probe as well as the rate-of-change of signal. You will find a discussion on this problem in the appendix under 'The Transmission Line Problem'.

The signals you have been monitoring are the primary clock signals. They are the signals you would expect for a working system.

The Φ_1 clock is not used by the microcomputer system, it is only used inside the 6502 microprocessor. Internal transfers of data take place during Φ_1 high. The system clock is in fact the microprocessor Φ_1 clock buffered by the logic invertors (IC14-C and IC14-f). It is used to 'Strobe' all external data transfers.

● EXERCISE 2.3

- Monitor the Φ_1 clock and the Φ_2 (system) clock. This latter signal can be obtained of the 7404 inverter (pin 12) or on the control signals 0.1" pins top centre of the board.
- Monitor the input and output pins (5 and 12 respectively) of the two invertors in IC14. Notice the 'buffered' effect.

● Read/Write ($R/\overline{W}$)

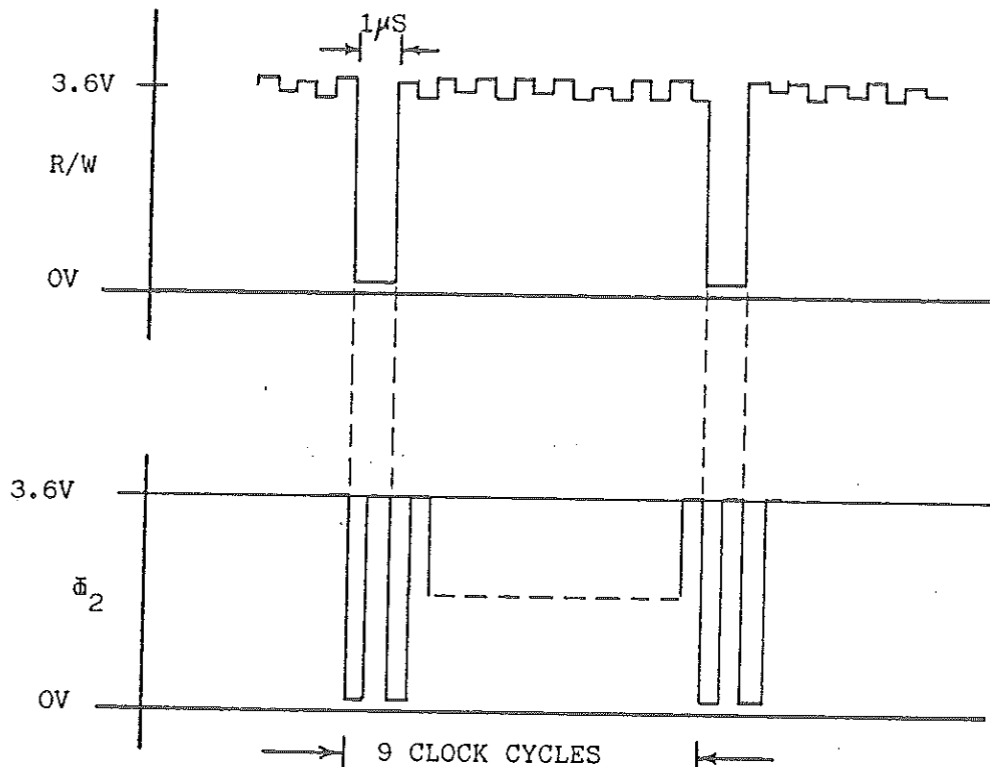
The Read/Write line controls the direction of data transfers between the processor and system support chips. The line is normally high except when 'writing' to memory or peripheral interface devices. All transfers on the line take place during Φ_1 clock pulse (concurrent with address lines). The method allows complete control of the data transfers which takes place during Φ_2 clock pulse.

The $R/\overline{W}$ line high signifies data transfer into the microprocessor while $R/\overline{W}$ line low signifies data transfer out of the microprocessor.

The R/W line is buffered and capable of driving 130pf and one standard TTL load.

● EXERCISE 2.4

- Monitor the Φ_2 (System) clock and the R/W line when the system monitor is operating (dots on display).
- Check that the waveforms are similar to those below:



During the time that the $R/\overline{W}$ is low the microprocessor is writing to memory. We will look at this in more detail later when we will write our own program and predict the times that the $R/\overline{W}$ line should be low.

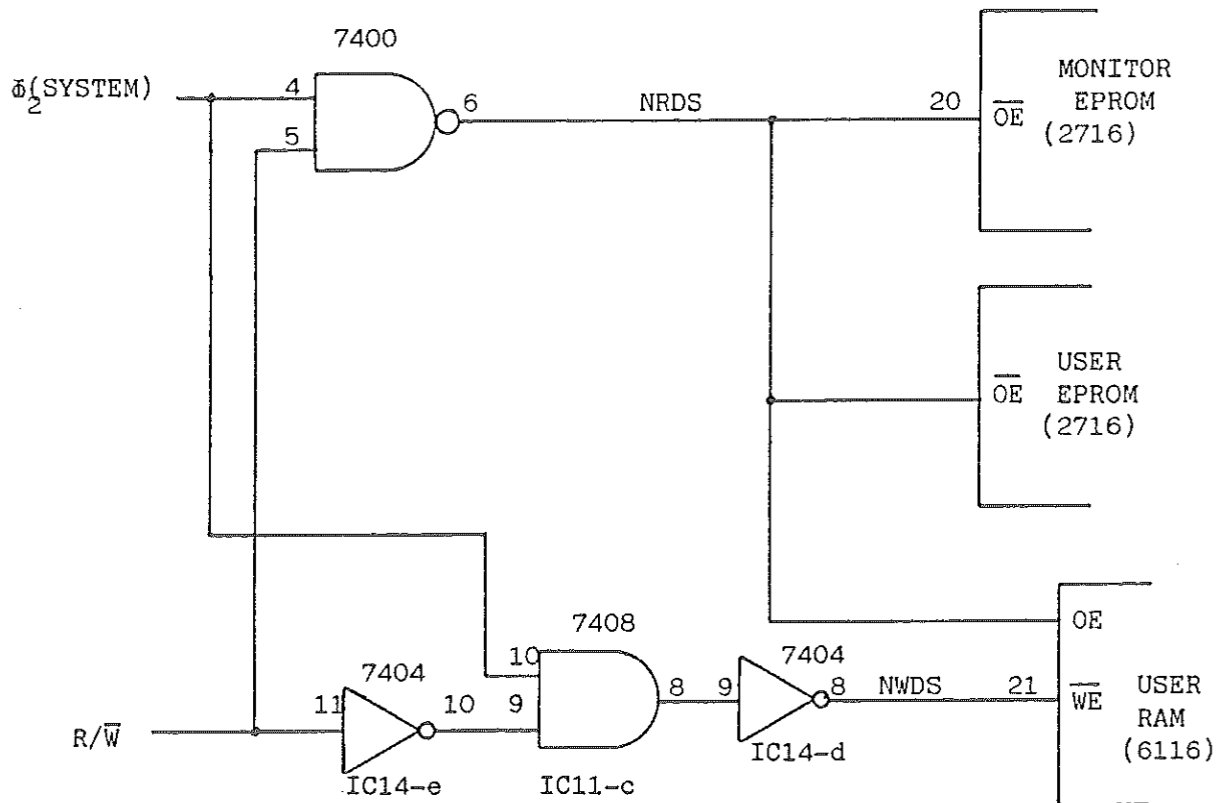
For the time being, notice that both lines are outputting at an expected T.T.L. level.

The 'castleation' effect on the logic high level of the $R/\overline{W}$ line is due to 'cross-talk' - electromagnetic coupling between adjacent lines. The effect is obviously interference but since it is well within T.T.L. noise limits it presents no problem.

● Not Read and Not Write Data Strobes (NRDS, NWDS).

The output enable ($\overline{OE}$ of the 2716 EPROM'S) and the write enable ($\overline{WE}$ of the 6116 RAM) lines of the memory chips are not time compatible with the $R/\overline{W}$ line of the 6502. To obtain the desired compatibility the $R/\overline{W}$ line is gated by the system phase two clock.

The circuitry is detailed below:



MEMORY DATA STROBE GENERATION

● EXERCISE 2.5

- From your knowledge of the timing relationship between the ϕ_2 and $R/\overline{W}$ signals, produce a 'timing diagram' showing the relationship between the ϕ_2 clock and the NRDS. Indicate on your diagram the period during which the Monitor EPROM is being read.
- Check your waveforms by monitoring the ϕ_2 clock at pin 4 on IC13-6 and pin 20 on the Monitor EPROM.

● EXERCISE 2.6

- Repeat exercise 2.5 but for the NWDS. This time indicate when the user RAM is being written to.

We will take a further look at these two signals when we consider inter-chip timing later.

● Sync (SYNC)

This output line identifies those cycles in which the processor is performing an OP.Code fetch. SYNC goes high during ϕ_1 and remains so for the rest of the cycle.

If the Ready (RDY) line is pulled low during the phase one clock pulse in which the SYNC line went high then the processor will stop in its current state. It will remain in this state until the RDY goes high (see also READY below).

● Ready (RDY)

This line is an input line, it allows the user to halt the processor on all cycles except write cycles. On EMMA it is brought out to a 0.1" pin and also held high by means of a pull-up resistor. A negative transition on the RDY line during or coincident with phase one (ϕ_1 high) will halt the microprocessor with the address lines reflecting the current address being fetched. The condition will remain through a subsequent phase two (ϕ_2 high) in which the Ready (RDY) line is low.

If RDY is low during a Write cycle, it will be ignored until the following Read operation. Ready (RDY) transitions must not be permitted during ϕ_2 time.

We will return to the SYNC and RDY lines later.

● Reset (RES)

Again an input line. It is used to initialise the microprocessor from a power-down condition. The line is held high by means of a pull-up resistor and pulled low by User operation of the EMMA RESET pushbutton. Three states, for the processor can be identified:

1. Pre-negative transition on Reset line. Supply line, V_{cc} (+5V) must reach +4.75 volts (Powered-Up).
2. Post - Power-Up. The Reset line must be held low for at least two clock cycles. Both R/W and SYNC lines will then become valid.
3. Post-Reset line low condition. A positive edge on the Reset line will cause the processor to begin an internal initialisation routine lasting six clock cycles. At the end of this period the Interrupt Mask Flag will be set and the processor will load the program counter with the System Reset vectors from System Memory Vector location FFFC and FFFD. The processor will then proceed with a System Reset.

We will use the logic analyser to log this routine later.

● Non-Maskable Interrupt ($\overline{\text{NMI}}$)

Again an Input line.

A Non-Maskable Interrupt sequence is initiated by a negative going transition on the $\overline{\text{NMI}}$ line. The effect is for the signal to interrupt the processor after it completes the instruction currently being processed. There is no way in which the 'interrupt' can be prevented hence 'non-maskable'.

The line, once taken low can be held low without effecting the processor operation and without another interrupt. To defect another interrupt the line must be taken high and then back to low. The $\overline{\text{NMI}}$ must be low for at least two clock cycles for the interrupt to be recognised. Whereupon new program count vectors are fetched.

The $\overline{\text{NMI}}$ is held high by means of a Pull-Up resistor to facilitate Wire-OR operation.

● Interrupt Request ($\overline{IRQ}$)

An input line.

The processor responds to an $\overline{IRQ}$ in much the same way as an $\overline{NMI}$ except that the interrupt is sensitive to signal level rather than signal edge. To effect an interrupt the $\overline{IRQ}$ pin must be taken low AND the Interrupt Inhibit Flag - I (Single bit in the Status Register - S) set logic '0'. The method of implementation means that the $\overline{IRQ}$ signal must be held low until it is recognised. This can only be when the processor has completed its current instruction. If the inhibit flag (I) is set when an $\overline{IRQ}$ occurs then this must be cleared by software control before the interrupt can be recognised.

To ensure the same interrupt cannot be recognised more than once, the inhibit flag (I) is set automatically during the last cycle before the processor begins executing the interrupt software beginning with the 'fetch' of program count.

The $\overline{IRQ}$ is held high by means of a pull-up resistor to facilitate Wired-OR operation.

● Set Overflow (SO)

Also an input line which sets the Overflow Bit in the processor Status Register when a Negative going edge is detected.

The signal is also sampled on the trailing edge of the ϕ_1 clock.

The Set Overflow line is held high by means of a Pull-Up resistor to facilitate Wired-OR operation.

We have now concluded all the control signals to and from the microprocessor.

● EXERCISE 2.7

● Reset EMMA so that the System Monitor is running (Dots on display).

- Use your oscilloscope to monitor the $\overline{NMI}$, $\overline{IRQ}$, RDY, SO and $\overline{RES}$ lines. Compare them with the R/W line. Suitable oscilloscope setting are:

Time-Base $1\mu s$
 Input Sensitivity 2 Volts/cm

- Explain why the amplitude of the five signals is different to that of the R/W line and also why the castle effect evident on the R/W is not present on the other lines.
- Whilst running the Monitor, monitor the $\phi 2$ and SYNC lines. Notice the effect upon these lines of depressing the RESET pushbutton.
- Explain why no transitions are evident upon the SYNC line with the RESET pushbutton held depressed although the $\phi 2$ clock is still operative.
- Monitor the RESET line and notice that the line is pulled-low by depression of the RESET pushbutton.

● Data Lines

Eight data lines form the Data Bus. Each line is regarded as TTL compatible but this refers to output logic levels. The drive capability is one standard TTL load at no greater than 130 pF. The Data Bus is also 'Three-state'.

As the name implies, the Data Bus carries 'Data' through out the System. The Data being carried at any one period in time may have originated from the microprocessor, a memory device or from some connected peripheral device via a device controller or simple Input/Output Port. Hence the Bi-Directional nature of the microprocessor bus input circuitry.

Most of the control signals that we have 'looked' at have been in some way 'periodic'. It could well be that we may find 'periodic' waveforms on the Data Bus, in which case we would be able to examine them on an oscilloscope.

However, invariably this is not the case and difficulty would be experienced in obtaining a suitable trace. This does not preclude us using the oscilloscope on the data lines as we will see in the following exercise.

● Exercise 2.8

- Run the System Monitor.
- Set up your oscilloscope to monitor Data Lines D0 and D1 simultaneously.
- Carefully examine your traces. You should find each channel presents a 'bright' trace with another 'background' trace. You should appreciate that this is due to the aperiodic nature of the data trace. It is possible for the data to be so aperiodic that a sensible trace is unobtainable. Although triggering the oscilloscope from the R/ $\overline{W}$ line may help.

The exercise above does indicate that difficulties do exist when using the oscilloscope on Data Lines (also Address Lines) and for that matter on all

lines carrying aperiodic data. However, a more serious problem exists when we wish to examine more than two data lines simultaneously and over a larger time scale than that permitted by the oscilloscope. Later we will use the Logic Analyser to overcome this problem for us.

● **Address Lines (AB0 - AB15)**

Sixteen address lines form the Address Bus. The only differences between the Address Lines and the Data Lines is that the Address Lines are both bi-state and uni-directional. The reason is that all system addresses are set up by the microprocessor.

● **Chip Select/Chip Enable Lines ($\overline{CS}/\overline{CE}$)**

These lines are generated through address line decode circuitry. They ensure that only one memory or I/O chip is enabled at any given time. Since these lines are generated using TTL logic devices we would expect to find TTL output voltage levels upon them.

● **Exercise 2.9**

- Run the System Monitor.
- Set up your oscilloscope to monitor the Chip Enable ($\overline{CE}$) of the 6116 RAM (Pin No.18) and $\overline{CE}$ of the Monitor 2716 EPROM (Pin No.18).
- Having obtained the best possible trace what can you say about the periodic nature of the two signals.
- Examine carefully both traces. Notice that at no point are both traces 'low'. Why should this be?

Exercise 2.10

- Run the System Monitor.
- Set up your oscilloscope to monitor the Chip Select lines ($\overline{CS2}$ and CS0) of the Keyboard/Display I/O Port (6821 - Pin Nos. 23 and 22 respectively).
- Now monitor $\overline{CS2}$ and CS1 (Pin Nos. 23 and 24 respectively).
- Explain the difference between the traces obtained on pins 22 and 24. Does the difference tell you anything about the way the signals have been generated.

Exercise 2.11

- The user I/O port (6522) resides on page 09 with the data direction register for Port A at memory location 03 and the data register at memory location 01. Write a 'minimal' program to repeatedly increment the port.
- With the program running use your oscilloscope to monitor the chip select lines CS2 and CS1 (Pin Nos. 23 and 24 respectively).
- Examine (while your program is running) the chip select lines for the:
 - i) Monitor EPROM
 - ii) Keyboard/Display I/O port and
 - iii) User RAM.
- From the information obtained state those devices which are active during the running of your program.

POST TEST

Chapter 2

- 2.1 Name two input and two output System Control Bus lines.
- 2.2 Explain why the microprocessor Address Bus need only be uni-directional.
- 2.3 Explain why an oscilloscope trace may show 'ringing'.
- 2.4 What is the function of the R/W line.
- 2.5 What will be the possible effect upon a P.C.B. track of a pulse waveform being carried on an adjacent track.
- 2.6 Explain the reason, in the EMMA System, of the NRDS and NWDS control lines.
- 2.7 Explain why the RESET line is permanently connected to a Pull-Up resistor.

CHAPTER 3

BASIC MICROPROCESSOR TEST EQUIPMENT

Objectives of this Chapter

Having studied this chapter you should:

- Be able to choose the correct Test Instrument for a given area of microcomputer investigation.
- Understand the 'Logic Probe' and how to use it.
- Be familiar with the Logic Analyser and know when and how to use it.

Introduction

The basic 'tools' required to analyse and test a microprocessor based system are:

- a Multimeter (Digital)
- an Oscilloscope
- a Logic Probe
- a Logic Analyser

Both multimeters and oscilloscopes have been basic test instruments for a number of years and will not be considered further. However, setting-up procedures for the oscilloscopes are included in the appendices which also provide useful information regarding oscilloscope voltage probes.

The logic probe and the logic analyser are more recent innovations. The latter is almost indispensable for certain types of fault which may be experienced on microprocessor based equipment.

• **The Logic Probe**

The logic probe is designed to detect out-of-tolerance logic signals, open circuit modes as well as transient events down to 300 nanoseconds. It may be used on DTL/TTL and CMOS/HTL circuitry.

Logic levels are defined when the probe is connected to the system supply of the logic under test or may be user defined by connecting to a suitable external supply.

• **DTL/TTL Logic Levels:**

Logic '1' - 2.25V

Logic '0' - 0.8V

• **CMOS Logic Levels:**

These are determined by the applied Vcc voltage and are:

Logic '1' - 70% Vcc

Logic '0' - 30% Vcc

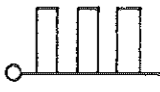
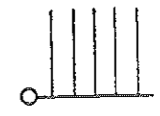
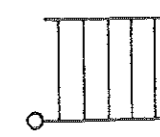
• **The LEDS**

Three LED's provide the signal data available at any point in the circuit under test.

The PULSE LED is activated for 0.1 seconds each time the input signal changes state, for example from a logic 1 0 or logic 0 1. When observing low frequency, low duty cycle signals, the PULSE LED gives an immediate indication of pulse activity at the mode under test. High frequency signals cause the PULSE LED to flash at a 10 Hertz rate.

Interpretation of the LEDS can best be achieved by consulting the following table.

INTERPRETING THE LEDS

LED STATES			INPUT	
HIGH	LO	PULSE	SIGNAL	
				LOGIC '0' NO PULSE ACTIVITY
				LOGIC '1' NO PULSE ACTIVITY
ALL LEDS OFF				
				1. TEST POINT IS AN OPEN CIRCUIT 2. OUT OF TOLERANCE SIGNAL 3. PROBE NOT CONNECTED TO POWER* 4. MODE OR CIRCUIT NOT POWERED
				THE SHARED BRIGHTNESS OF THE HI AND LO LEDS INDICATED A 50% DUTY CYCLE AT THE TEST POINT ($\leq 1.5\text{MHz}$)
				HIGH FREQUENCY SQUARE WAVE 1 MHz ($> 100\text{KHz}$) AT TEST MODE AS THE HIGH FREQUENCY SIGNAL DUTY CYCLE SHIFTS FROM A SQUARE WAVE TO EITHER A HIGH OR LOW DUTY CYCLE PULSE TRAIN EITHER THE LO OR HI LED BECOME ACTIVATED.
				LOGIC '0' ACTIVITY PRESENT POSITIVE GOING PULSES SINCE HI LED NOT 'ON' PULSE TRAIN DUTY CYCLE IS LOW RE ($< 15\%$) IF THE DUTY CYCLE WERE INCREASED ABOVE 15% HI LED WOULD START TO TURN ON.
				LOGIC '1' PULSE ACTIVITY PRESENT NEGATIVE GOING PULSE SINCE LO LED NOT 'ON' PULSE TRAIN DUTY CYCLE IS HIGH RE ($> 85\%$). IF THE DUTY CYCLE WERE REDUCED TO $< 85\%$ 'LO' LED WOULD START TO TURN ON.

-  LED ON
-  LED OFF
-  BLINKING LED

Exercise 3.1

Run the system monitor.

Use the Logic Probe to monitor and record the following signals:

a) $\overline{\phi}_2$ (System)

b) $\overline{R/W}$

c) $\overline{RES}$

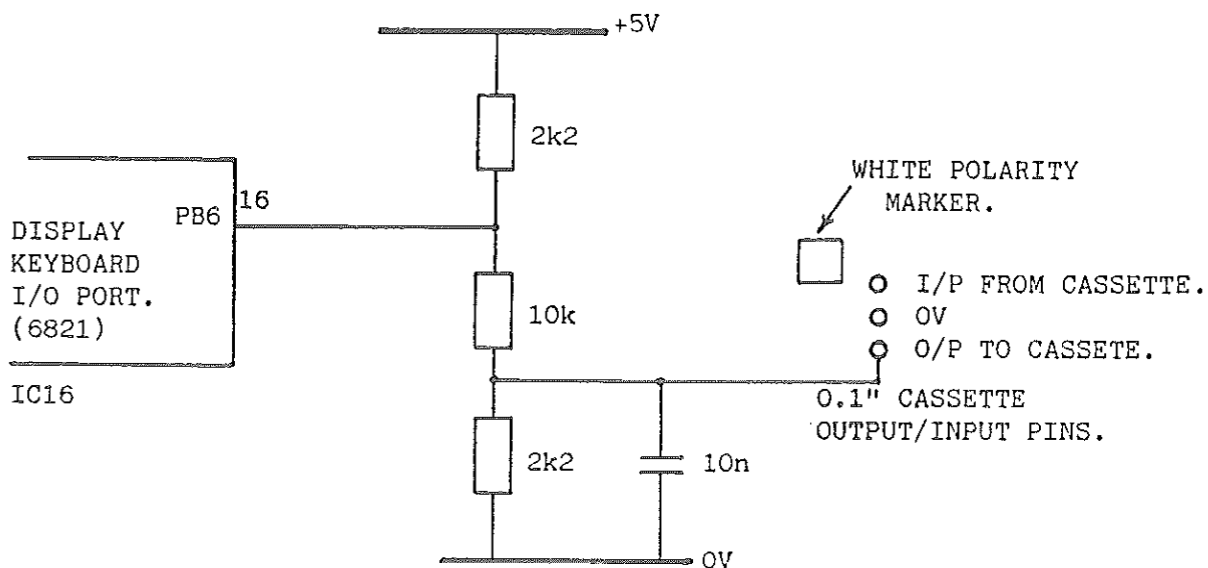
d) SYNC

e) RDY

From the information obtained via the LEDs indicate the nature of each signal.

Exercise 3.2

The cassette interface output circuitry is shown below:



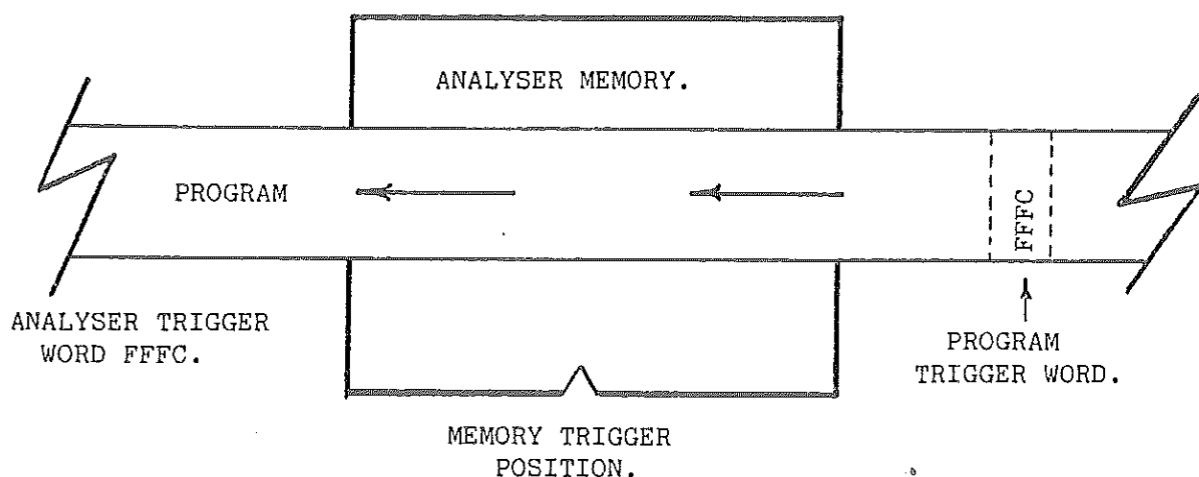
- Set-up the interface to output pages F0-F7 (F000 - F7FF) at 300 Baud. This will give you approximately 70 seconds in which to make the following tests. See User Manual for SAVE procedure.
- With this interface LED ON, use the logic probe to monitor pin 16 on the Display/Keyboard I/O Port Chip and the 0.1" cassette output/input pins.
- Notice that the pulse output recorded at pin 16 is not 'seen' at the 0.1" pin. Explain why this should be (Hint, study diagram).

THE LOGIC ANALYSER

Introduction

In simple terms the Logic Analyser comprises a memory into which is 'clocked' data from a number of test nodes in the logic circuitry under test. The data will continuously pass through memory until a particular event occurs when the current contents of the memory is 'frozen'. The 'particular event' may be the recognition of a predetermined bit pattern appearing at the nodes under test. Receipt of this bit pattern would trigger the freezing of memory. The predetermined bit pattern we refer to as a 'trigger - word'.

We can depict the above diagrammatically as below.



Initially the Trigger Word is SET and the Analyser is ARMED (made ready). The program under analysis is now run and passes through the analyser memory. When the Analyser Trigger Word (in example FFFC) is detected in the program and at the centre of the memory, the memory is frozen, although the program continues to run. Since the trigger event occurs at the centre of the memory, half the memory will contain the pre-trigger data and the other half, past-trigger data.

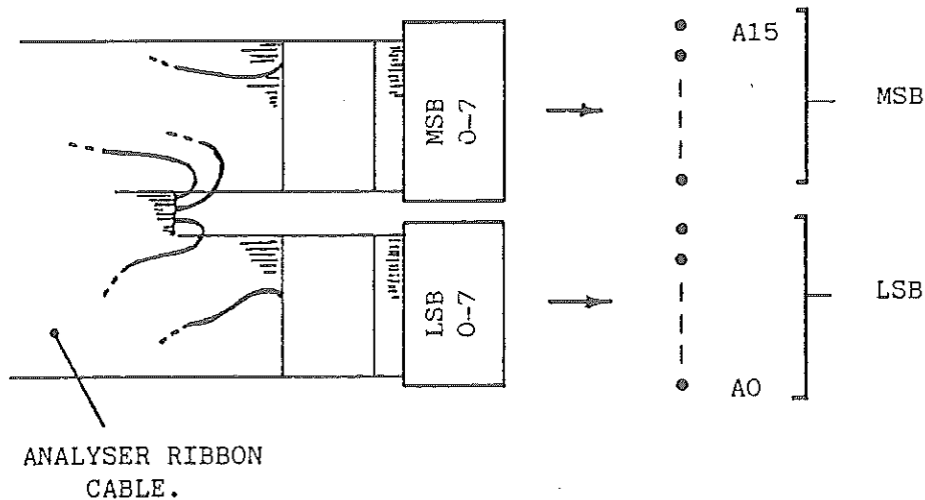
The analyser has other useful features which allow us for example to 'qualify' both the trigger-word or the clock input. We will consider these later.

Now let's use the Analyser.

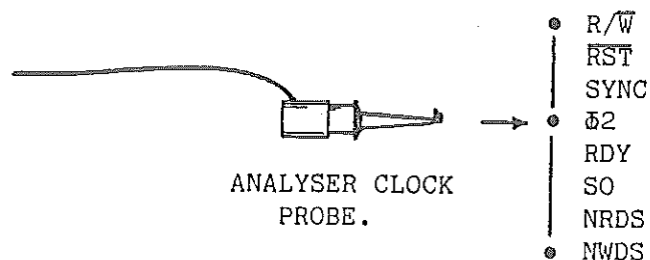
Exercise 3.3

- Connect the Logic Analyser data ribbon cable to the EMMA address bus using the 0.1" pins.

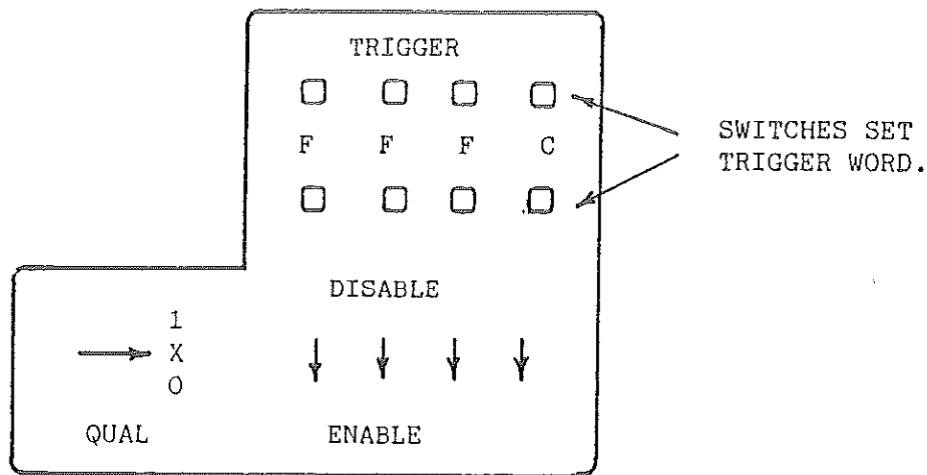
Notice: The ribbon cable is terminated as Least Significant Byte (LSB) and Most Significant Byte (MSB).



- Connect the following Analyser ribbon cable miniture clip-on probes:
 - a) Three-Ground (GND) probes to available EMMA zero volt 0.1" pins.
 - b) Clock probe to EMMA system Ø2, 0.1" pin.



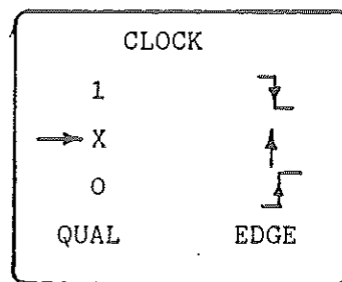
• Set Analyser 'TRIGGER' section to:



These settings enable the analyser to be triggered off the TRIGGER WORD FFFC only, the trigger qualifier being set to 'X' (don't care).

Notice the trigger word can be DISABLED or ENABLED as a complete or selected part word.

• Set Analyser 'CLOCK' section to:



These setting, clock data into the analyser memory on the trailing (↓) edge of the EMMA system Ø2 clock. Using this setting ensures that any captured data will be stable by the time the clock edge occurs. Again, the qualifier is set to 'don't care' and as such does not influence the data capture.

Analyser settings are now complete.

- Run the EMMA system monitor (Dots' on Display).
- Momentarily move the ABORT/ARM switch to ABORT and then to ARM (notice, the switch has a spring-return-to-neutral action).

The Analyser should now be ARMED (awaiting data capture) with both CURSOR and DATA displays extinguished (apart from LED indicating ARMED).

- To capture data, press EMMA RESET pushbutton. Both CURSOR and DATA displays should now be illuminated.
- Increment (INC) or Decrement (DEC) CURSOR position using INC/DEC switch. (Notice Auto Repeat feature). Find CURSOR zero position.

The Data indicated should be FFFC indicating that the Analyser triggered off the preset TRIGGER WORD.

The data captured are the first 63 Addresses appearing on the address bus from the instant the low byte RESET VECTOR appeared and following release of the EMMA RESET pushbutton.

The program actually captured the initial part of the EMMA Monitor which causes the Dots' to appear on the display and the keyboard to be scanned awaiting user data entry.

You should find the following DATA.

CURSOR	DATA	
00	FFFC	6502 MICROPROCESSOR RESET VECTOR MEMORY LOCATIONS
01	FFFD	
02	FEE0	START ADDRESS (RESET VECTOR) OF EMMA RESET ROUTINE
03	FEE1	

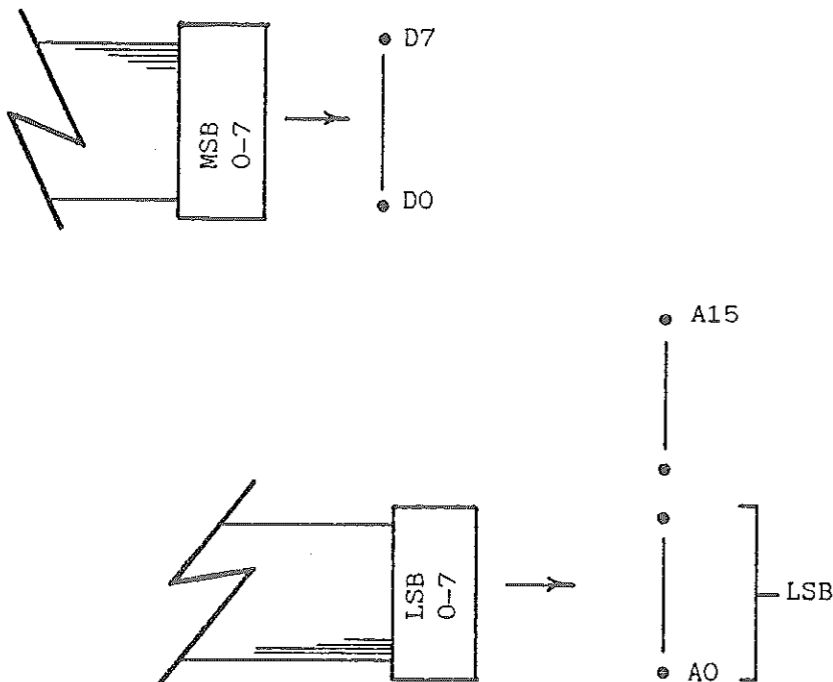
You should have now actually captured Data, in this case addresses, from a known location in memory. You should appreciate that at this point the Data may not mean much to you.

Now lets capture some data from the Data Bus.

● Exercise 3.4

For the Data on the Data Bus to be meaningful, it must relate to the Address Bus Data. We are required therefore to capture simultaneously both Address Bus Data and Data Bus Data. To accomplish this requires a total of 24 bits Address Bus and 8 bits Data Bus. We have only 16 bits available to us therefore we will capture 8 bits of Data Bus and the 8, least significant bits of the Address Bus. Having already logged all 16 Bits of the Address Bus for the same program we should then easily be able to bring together the two listing to form a complete composite one. The CURSOR will also help us in this task.

- Reconnect the Analyser to EMMA as follows:



CLOCK → SYSTEM Ø2
GND(S) → GROUNDS

- Set Analyser as:

TRIGGER WORD	EOFC	(ENABLED)
TRIGGER QUALIFIER	X	
CLOCK EDGE	TRAILING	
CLOCK QUALIFIER	X	

- Run EMMA MONITOR.
- ABORT/ARM
- RESET EMMA
- RECORD first few Data items from CURSOR position 00 to 04.
- Bring together Data captured from Exercise 3.3 and this exercise.
You should find they co-relate as:

● EXERCISE 3.3

CURSOR	DATA
	ADDR. BUS
	MSB LSB
00	FFFC
01	FFFD
02	FEE0
03	FEE1
04	FEE2

● EXERCISE 3.4

CURSOR	DATA
	(BUS) DATA ADDR.
	(BYTE) - LSB
00	E0 FC
01	FE FD
02	AE E0
03	03 E1
04	F8 E2

Co-Relation

CURSOR	ADDR. BUS	DATA BUS	
00	FFFC	E0	} RESET
02	FFFD	FE	
02	FEE0	AE	} VECTORS
03	FEE1	03	
04	FEE2	F8	} FIRST RESET
			} INSTRUCTION
			} (LDX F803)

Note: It may not always be possible to co-relate using CURSOR alone hence we have captured the Address low byte (changes more often than high byte) on each data log and used this to co-relate.

Now lets enter an easily understood program and use it to investigate the validity of Trigger Words and the use of Qualifiers.

● Exercise 3.5A

● Enter the following Machine Code program into EMMA.

0200	A9,FF	LDA# FF
2	85,10	STA Zero Page,10
4	4C,04,02	JMP 02 04

The program uses instructions which READ and WRITE memory locations before terminating with a repeating JUMP instruction.

- Connect Analyser to capture Address Bus Data.
- Set the Analyser TRIGGER to 0200. Use same setting for TRIG. and CLK. QUAL. and CLock Edge as before.
- Abort and Arm as before.
- Run program from 0200 to capture Address Bus Data. Make a record of capture up to program repeat.

● Exerrcise 3.5B

● Re-connect to capture EMMA low byte Address Bus and EMMA Data Bus. Connect as:

ANALYSER	EMMA
MSB	- (DATA BUS)
LSB	LSB (ADDRESS BUS)

- Set TRIGGER WORD to A900 and ARM.
- Run program from 0200 and Record.
- Combine Records to co-relate both Address and Data Bus Data.

Exercise 3.5C

Now answer the following:

- From your combined listing, explain what the microcomputer was doing at:

CURSOR	ADDR.BUS	DATA BUS
04	0010	FF

- TRIGGER WORDS' used were 0200 and A900 respectively. Other TRIGGER WORDS' could have been used and the CURSOR would have referenced the TRIGGER WORD from which the capture took place.

Lets try some of these TRIGGER WORDS' with the analyser left connected as for Exercise 3.5B, try to capture VALID data using FF01 and 8502. Now look at your original program to find other TRIGGER WORDS'. List THREE of them.

- Now reconnect the Analyser to capture the Address Bus. From your original program, list THREE more VALID TRIGGER WORDS.

Exercise 3.5D

When capturing the Address Bus we noticed that 0010 appeared on the Bus when loading FF into that memory. It does seem therefore that 0010 would also be a VALID TRIGGER WORD. You will notice that the Analyser will appear to ARM when the ABORT/ARM switch is depressed but will immediately trigger when released. The reason for this is that 0010 (and many other combinations) appear on the Address Bus when the Monitor is running.

The moral is, take care when selecting TRIGGER WORDS and always make sure that Data Captured is VALID.

Now lets look at the Qualifiers.

Exercise 3.6

Connect the Analyser as follows:

Data Pod	to	→	Address Bus
Clock		→	System Ø2
Trigger Qualifier		→	R/W

Set Analyser as follows:

Clock Qualifier	X
Clock Edge	TRAILING
Trigger Word	0203 (Enable)
Trigger Qualifier	1

Arm and Run previous programs (Ex. 3.5). Note that both ARMING and Capture is possible as previously. Now:

Change Trigger Qualifier to 0, ARM and RUN. Notice that the Analyser does not trigger. The reason is that the Analyser will now only trigger if both the TRIGGER WORD and Trigger Qualifier settings occur simultaneously. In our case the Trigger Qualifier Signal is at logical 1 (microprocessor is READING memory) when address 0203 appears but the Analyser is set to expect a logical '0' (microprocessor WRITE to memory). Obviously the condition never occurs.

Now lets try a different TRIGGER WORD, for example 0010. We have already found that this was none valid. The particular reason for its none validity is that the memory 0010 is the address of the seven-segment code for the extreme left-most Seven-Segment Display .

Immediately following RESET, the location (0010) will be WRITTEN too with 80 (Seven-Segment Code for decimal point). It will then be READ to display the decimal point. If now we try to RUN our program we must first press G to set-up the program start address. At the instant G was pressed the code BD (G) is initially loaded (WRITTEN) into 0010 and then continually READ for display. It does appear therefore that it should be possible to ARM the Analyser (with TRIGGER WORD 0010) after having first pressed G to set-up program start address AND if the Trigger Qualifier is connected to the Read/Write (R/W) line and set to logical '0'.

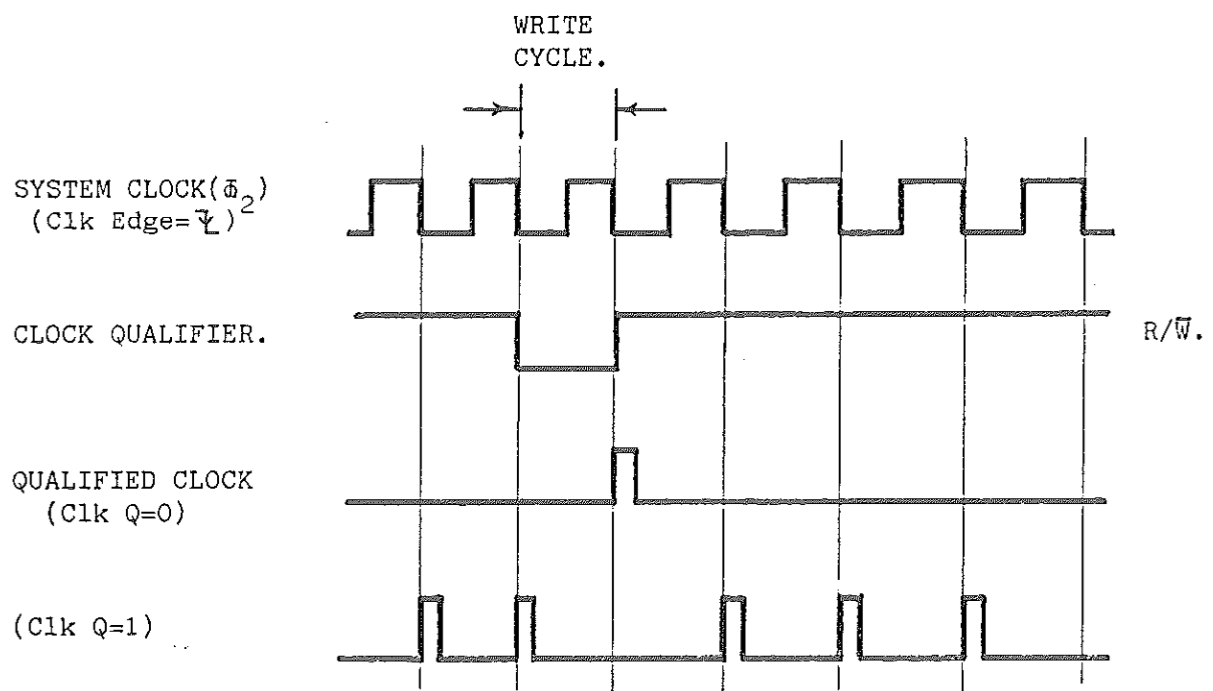
Exercise 3.7

Prove that the preceeding remarks are true by successfully capturing our program from 0010 with TRIG.QUAL set to '0' and monitoring R/W.

Needless to say we can use the TRIG.QUAL, in many useful ways, - we will meet some later. Now lets look at the CLOCK QUALIFIER.

The Analyser will only clock data into its memory when the set CLOCK EDGE appears AND the CLOCK QUALIFIER is VALID.

For example:



The qualified clock above is internal to the Analyser and is used as its system clock. If the Clock Qualifier is connected to the R/W line and set to '0' then only WRITE cycles would be captured. This assumes that the TRIGGER WORD and TRIGGER QUALIFIER were appropriately set. Conversely if CLK.QUAL were set to '1' then only READ cycles would be captured.

Exercise 3.8

- Write and enter into EMMA a simple program to load FF into zero page locations 30, 31 and 32. Terminate your program with a JMP to start of program. Commence your program at 0200.
- Connect the Analyser to capture the Address Bus only.
- Connect clock qualifier to R/W.
- Set Clock Edge to 'trailing'.
- Set TRIGGER WORD to 0200, TRIG.QUAL to X and CLOCK QUALIFIER to '1'. ARM Analyser and RUN.
- Log data captured. Notice that only READ cycles have been captured.
- Change TRIG.WORD to 0030 and CLK.QUAL to '0'. ARM and RUN.
- Log Data captured. Notice that only WRITE cycles have been captured.
- Explain why the Analyser will not trigger under the following conditions:

CLK QUAL = X; TRIG QUAL $\overline{R/W}$ AND

- a) TRIG.WORD 0200, TRIG.QUAL '0'
- b) TRIG.WORD 0030, TRIG.QUAL '1'

The qualifiers, as seen, allow us to selectively capture data, a feature which we will find extremely useful when fault finding.

You should now be at least familiar with the Logic Analyser as a stand-alone test instrument. We can, however, output the whole of the Analyser memory to an oscilloscope. We will now consider this mode of operation.

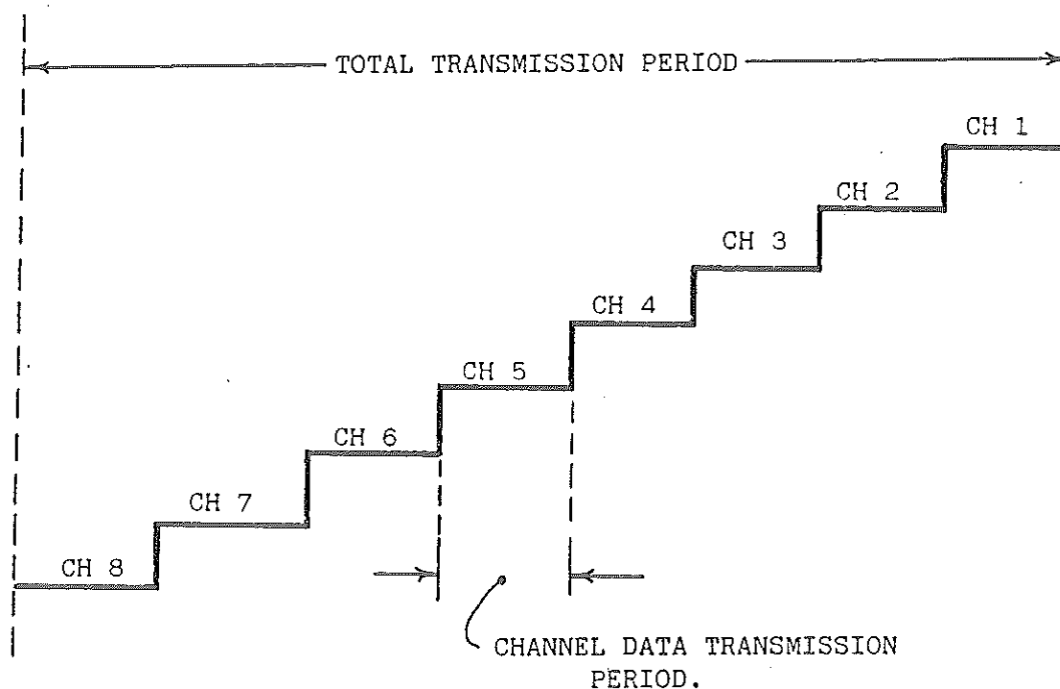
Each input channel to the Analyser stores 128 Data Bits. There are also two by eight bit Data channels. Thus the Analyser memory comprises 128, two byte memories.

Example:

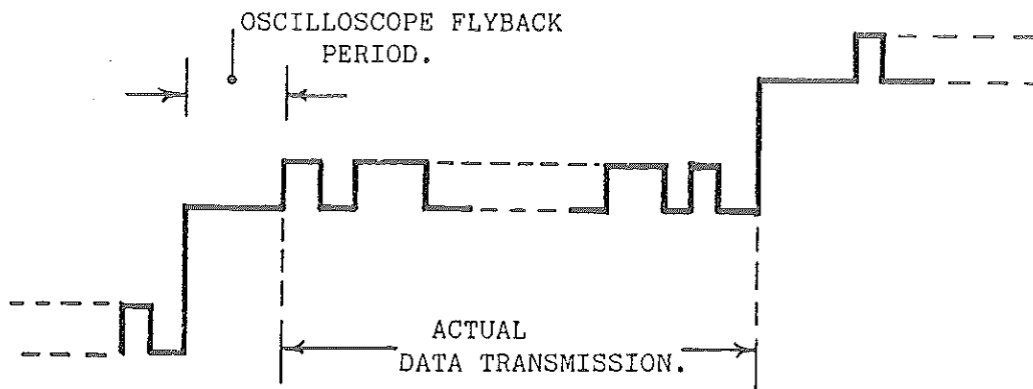
		128 DATA BITS.														
CHANNEL	1	1	0	1	1	-	-	-	-	-	-	-	1	0	1	] LSB
	8	1	0	1	0	-	-	-	-	-	-	-	0	0	1	
	9	0	1	0	0	-	-	-	-	-	-	-	0	1	0	
CHANNEL	16	0	0	1	1	-	-	-	-	-	-	-	0	0	0	] MSB

The stored data for each channel can be output serially for display on an oscilloscope. To enable inter-channel timing problems to be investigated, eight channels are also serially transmitted but modulated by an eight-step, stair-case.

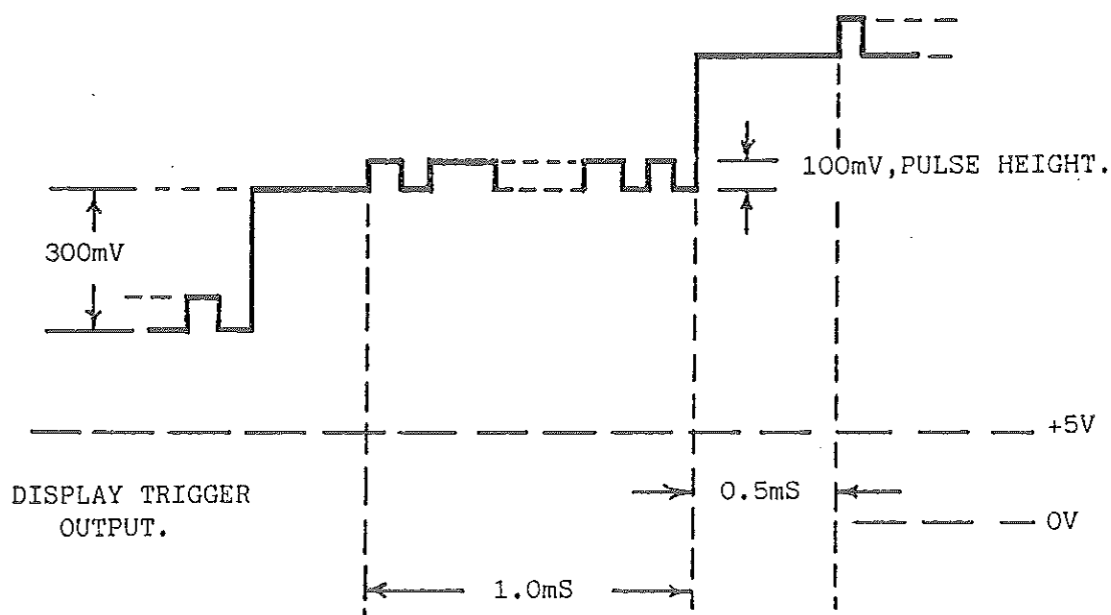
Example:



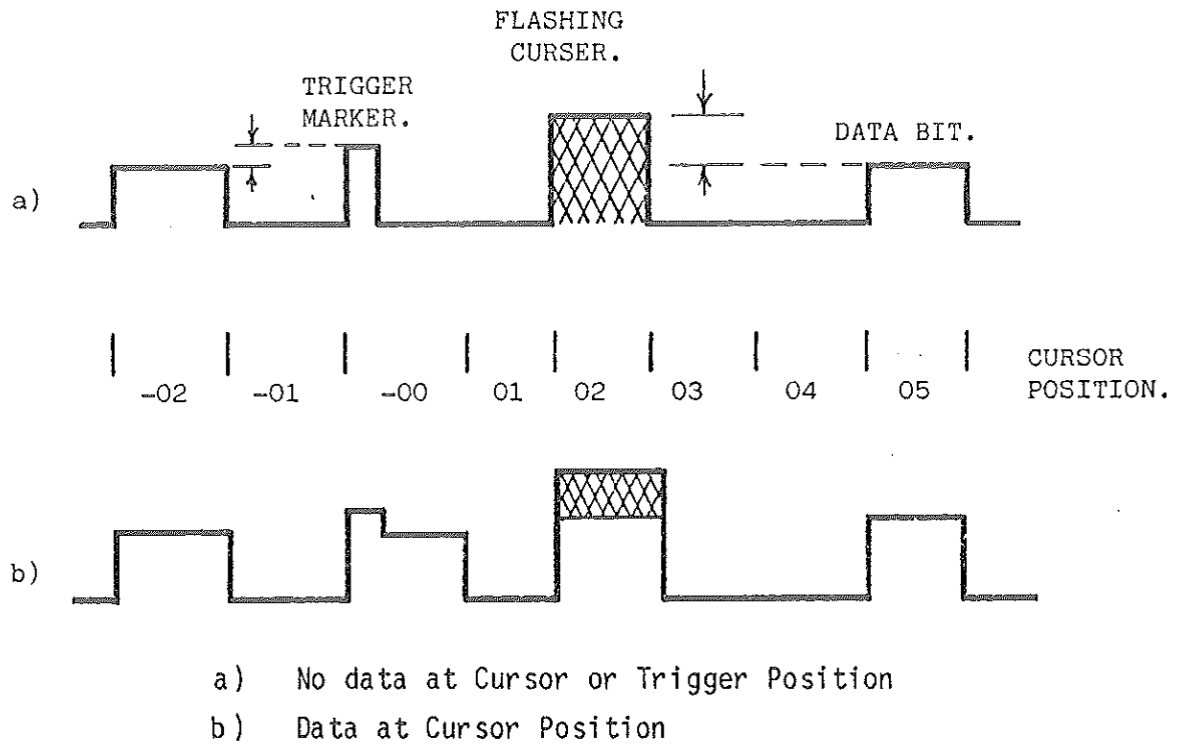
Each channel Data Transmission Period actually comprises two periods as shown below:



Associated with these is an internally generated trigger output for display purposes. The relationship, together with actual voltage levels is:



Finally, also modulated upon each channel data transmission as two markers: a fixed one which corresponds with the trigger event (Centre of memory) and a flashing, variable position one which marks the cursor position.



From the foregoing it should be appreciated that if an oscilloscope is externally triggered with a time base setting of $100\mu\text{sec/division}$ then, assuming 10 divisions in the X axis, one data channel output period would occupy the width of the screen. However trigger pulses arrive at intervals of 1.5mS and at the start of each data channel data burst, therefore all eight channels are effectively stacked one above the other. Obviously we also need to set sensitivity of the oscilloscope Y-amplifier.

Assuming 6 divisions in the vertical axis, and eight channels requiring 300mV amplitude each then to occupy the whole of the screen. A suitably sensitivity setting would be:

$$\frac{300\text{mV} \times 8 \text{ channels}}{6 \text{ Divisions}} = 400\text{mVolts/Div}$$

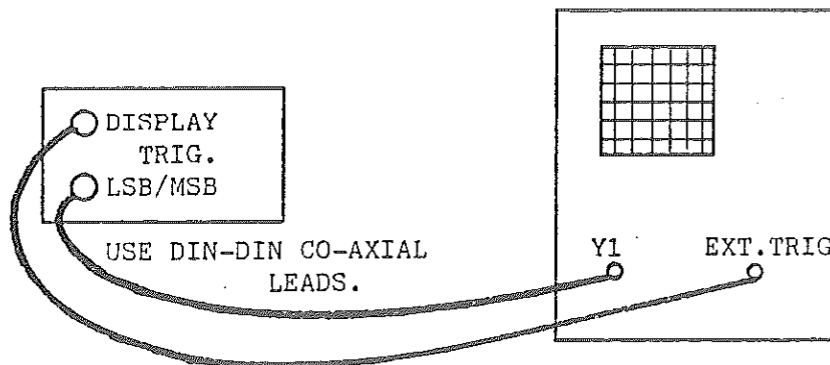
The nearest actual setting available is likely to be $500\text{mVolts/Division}$ and this would obviously be used.

One final word. The Display Trigger is a negative transition therefore the oscilloscope triggering should be accordingly.

Now lets get the oscilloscope in action!

Exercise 3.9

Connect the Oscilloscope and Logic Analyser as follows:



Set oscilloscope to:

Y1 SENSITIVITY	- 0.5V/Division
TIME BASE	- 0.1mA/Division

TRIGGER

Function	- External
Mode	- D.C. Coupled
Edge	- Negative
Trace	- Single

With the Analyser and Oscilloscope switched ON, adjust the Trigger Level until eight stable displays appear on the screen. Adjust traces to centralise on screen.

Load the following program

0200	A9,FF	LDA# FF
2	85,30	STA Zero Page, 30
4	40,04,02	JMP 02 04

- Set Analyser to:

TRIGGER WORD	0200
QUAL	X
CLOCK EDGE	TRAILING
QUAL	X

Connections -

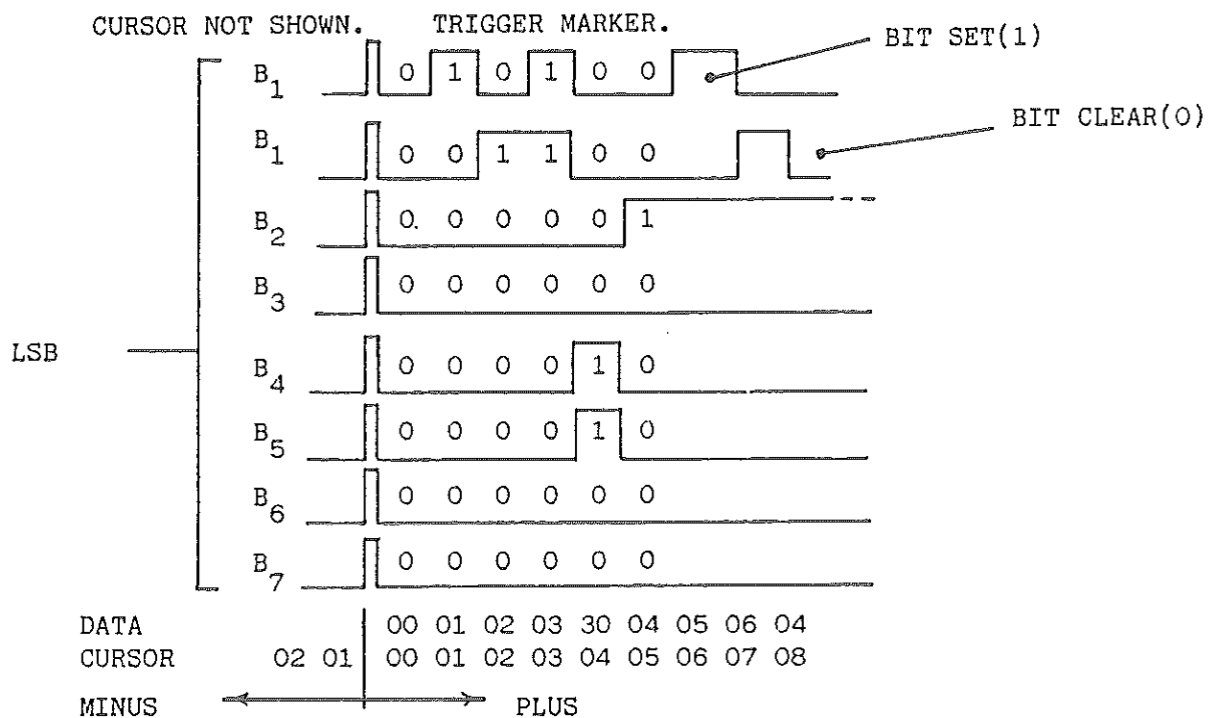
ANAL.	LSB	-	EMMA	ADDR	LSB
ANAL.	MSB	-	EMMA	ADDR	MSB

- ARM Analyser and RUN program from 0200.
- Switch Analyser DISPLAY switch to LSB.

Example:

DATA	DISPLAY
0 2 0 0	LSB
	↑
	MSB
MSB LSB	

- The Analyser DATA DISPLAY will now show 0200. When the Cursor is set to 00. The oscilloscope will show to the right of the TRIGGER position:



You may initially find it easier to READ oscilloscope display by 'expanding' it using the times 5 pushbutton.

- Increment cursor to move towards the right of the oscilloscope display and READ the DATA displayed for each position of the CURSOR.
- Now select MSB on the Aanalyser DISPLAY switch and READ the Data in each cursor position comparing it with that shown on the Analyser in the MSB DATA window.

You should now be very familiar with the Logic Analyser and its use. We will expand this knowledge in subsequent chapters of this manual.

POST TEST

Chapter 3

- 3.1 Choose an appropriate Test Instrument for diagnostic testing of the following lines:
- Address Bus
 - System Clock
 - System Reset
 - $R/\overline{W}$
- 3.2 Explain what is meant by the terms 'Trigger Word', 'Armed' and 'Qualified' when applied to Logic Analysers.
- 3.3 Explain why a Logic Analyser requires a 'Clock'.
- 3.4 Briefly explain how you would 'log' 16 address lines and 8 data lines when using a logic analyser capable of monitoring only 16 lines.
- 3.5 Explain why some 'trigger words' may not always be suitable for trigger purposes.
- 3.6 Explain why selection of the 'correct' clock edge is important when clocking the Logic Analyser.
- 3.7 Explain briefly how 8-channels of data can be displayed upon a single-channel oscilloscope.

CHAPTER 4

INVESTIGATION OF A WORKING SYSTEM

Objectives of this Chapter

Having studied this chapter you should:

- Be able to understand the data appearing on the 6502 microprocessor R/ $\overline{W}$ and Sync Lines.
- Be able to understand the data appearing on the Address and Data Lines.

Introduction

You will already be familiar with the Data passing along the Control Bus, Address Bus and Data Bus if you have studied chapters 2 and 3. The procedure we are going to adopt in this chapter will follow closely that you would be expected to follow when investigating a faulty system. We will again be working with EMMA only.

The purpose of this session is to investigate a 'working system' using three basic instruments - Multimeter, Oscilloscope and Logic Analyser.

● Exercise 4.1

● SYSTEM SUPPLY CHECK

- Connect-up the EMMA microcomputer to the System Power 90, 5 volt supply. Using a digital multi-meter, check the supply voltage and record.
- Switch off the supply and reconnect the multi-meter to record current drain.

Current Drain in mA (NO-display)

Current Drain in mA (Display-A8888.88)

● Exercise 4.2

CLOCK WAVEFORMS

Use an oscilloscope to observe and record the ϕ_0 , ϕ_1 and ϕ_2 clock waveforms.

- ϕ_0 - Available on lower lead of crystal.
- ϕ_1 - Available on pin 3 on 6502.
- ϕ_2 - (Buffered). This is the system clock and available on monitoring pin.
- ϕ_2 - (Unbuffered). Available on upper lead of crystal.
- Make a neat sketch of the timing relationship of all three waveforms.

● Exercise 4.3

- Reset EMMA to run the system monitor. If dots appear on the display then it is a fair assumption that you have at least a basic working system.
- Now load the following program:

Address	Hex. Machine Code	Program
0200	A9, 00	
0202	85, 30	
0204	85, 31	
0206	85, 32	
0208	4C, 00, 02	

You have met a similar routine before. There is nothing particularly special about it. It simply repeatedly stores data in the three locations 0030, 0031, 0032. Our objective is now to construct a timing diagram indicating the relationship between the system ϕ_2 clock, Read/Write (R/W) and the Sync line (SYNC).

CHAPTER 4

INVESTIGATION OF A WORKING SYSTEM

Objectives of this Chapter

Having studied this chapter you should:

- Be able to understand the data appearing on the 6502 microprocessor R/W and Sync Lines.
- Be able to understand the data appearing on the Address and Data Lines.

Introduction

You will already be familiar with the Data passing along the Control Bus, Address Bus and Data Bus if you have studied chapters 2 and 3. The procedure we are going to adopt in this chapter will follow closely that you would be expected to follow when investigating a faulty system. We will again be working with EMMA only.

The purpose of this session is to investigate a 'working system' using three basic instruments - Multimeter, Oscilloscope and Logic Analyser.

● Exercise 4.1

- SYSTEM SUPPLY CHECK
- Connect-up the EMMA microcomputer to the System Power 90, 5 volt supply. Using a digital multi-meter, check the supply voltage and record.
- Switch off the supply and reconnect the multi-meter to record current drain.

Current Drain in mA (NO-display)

Current Drain in mA (Display-A8888.88)

● Exercise 4.2

CLOCK WAVEFORMS

Use an oscilloscope to observe and record the ϕ_0 , ϕ_1 and ϕ_2 clock waveforms.

- ϕ_0 - Available on lower lead of crystal.
- ϕ_1 - Available on pin 3 on 6502.
- ϕ_2 - (Buffered). This is the system clock and available on monitoring pin.
- ϕ_2 - (Unbuffered). Available on upper lead of crystal.
- Make a neat sketch of the timing relationship of all three waveforms.

● Exercise 4.3

- Reset EMMA to run the system monitor. If dots appear on the display then it is a fair assumption that you have at least a basic working system.
- Now load the following program:

Address	Hex. Machine Code Program
0200	A9, 00
0202	85, 30
0204	85, 31
0206	85, 32
0208	4C, 00, 02

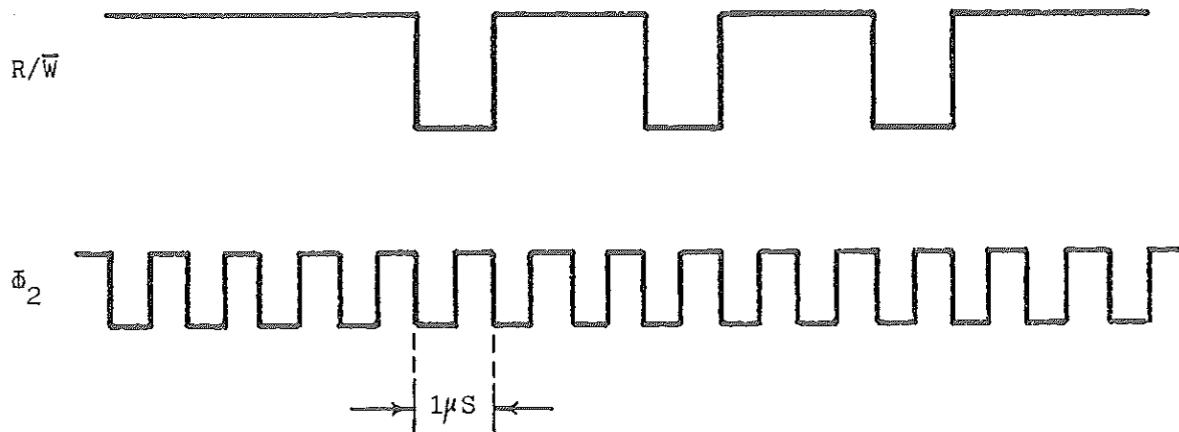
You have met a similar routine before. There is nothing particularly special about it. It simply repeatedly stores data in the three locations 0030, 0031, 0032. Our objective is now to construct a timing diagram indicating the relationship between the system ϕ_2 clock, Read/Write (R/W) and the Sync line (SYNC).

Now follow the procedure below:

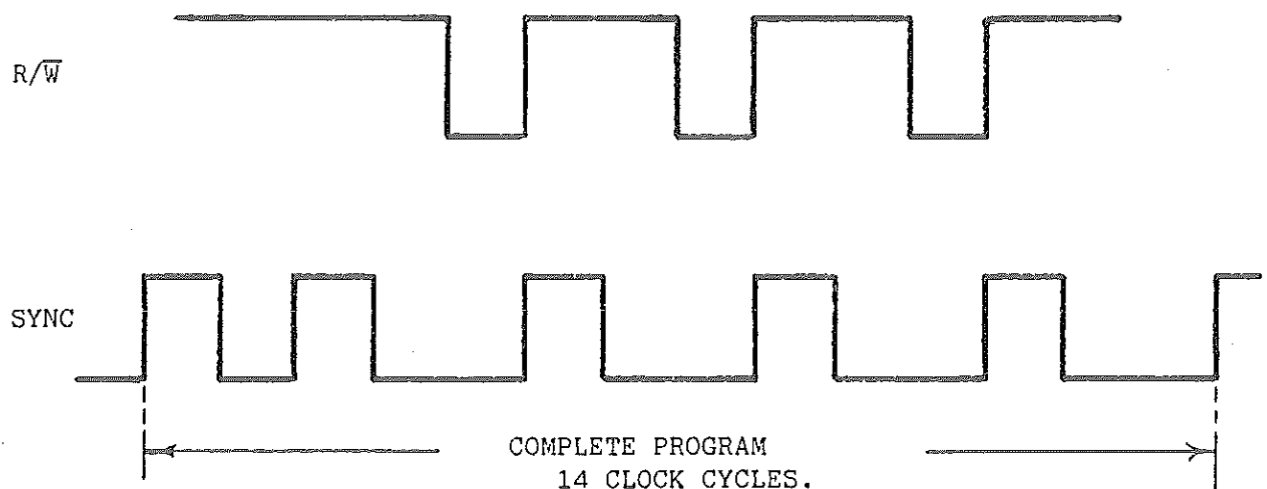
- Connect the $R/\overline{W}$ to channel 1 (CH1) of the oscilloscope and trigger from this signal.
- Connect the system ϕ_2 clock to channel 2 (CH2).
- Adjust the time-base to obtain a stable base with approximately 5 clock cycles/division.

You should observe the $R/\overline{W}$ line going low in groups of three transitions.

Example:



- Now remove ϕ_2 signal and compare $R/\overline{W}$ with SYNC using the same time-base setting. You should observe the relationship:



Now bring together your observations to form a composite timing diagram showing R/W, SYNC and Ø2 clock for the complete program. Then from your knowledge of the program and work in chapter 2, indicate a period on the timing diagram when:

- a) An Op.Code is being 'Fetched'.
- b) An Operand is being 'Fetched'
- c) A write to external memory is taking place.
- d) The LDA# 00 instruction takes place.

● Exercise 4.4

Now use the Logic Analyser to examine the Address and Data Buses.

- Record Complete program (Addr. and Data)
 - Record Memory READ operations only.
 - Record Memory WRITE operations only.
-

We are now going to 'have a look' at the System Software - at least at the initial part of the Monitor Program. We are interested in this for two reasons.

Firstly, we need to acquire skill in interpretation of otherwise unknown programs that a dedicated system may be running.

Secondly, we must understand what the machine does after a cold start RESET.

We can partially achieve both these objectives by using the Logic Analyser triggered from the RESET line.

● Exercise 4.5

Set up the Logic Analyser (no need to use the oscilloscope) so that it will trigger when the RESET pushbutton on EMMA is released.

Example:

- TRIGGER
 - Word - Disabled
 - Qualifier - Logical One

- CLOCK
 - Edge - Negative
 - Qualifier - Don't Care

- DATA POD
 - a) ANALYSER EMMA
 - LSB LSB
 - MSB MSB

 - b) LSB LSB (ADDR.BUS)
 - MSB DATA BUS

- PROCEDURE

For both connections a) and b) above, hold RESET pushbutton depressed, ARM Analyser, release RESET pushbutton.

- Collate the first 50 cursor increments (from 00).

The information you have obtained is the total Address and Data Bus activity immediately before and after a system Hardware Reset signal. We are not particularly interested in what happened before RESET only what happens after the RESET pushbutton is released.

The first 6 or 7 clock pulses capture random data. The bus activity is associated with reset of the microprocessor internal registers. After completion of this activity the microprocessor places FFFC upon the Address.

Bus and READ the Data at that address. It will treat this as the low byte of a USER RESET Routine. FFFD will then appear on the Address Bus to READ in the high byte of the User Reset Routine. The term used to refer to the 'fetched' data is USER RESET VECTORS.

In the EMMA system the reset vectors are FEE0, high byte and low byte respectively. The start of the Analyser data we are interested in is therefore:

ADDR BUS	DATA BUS
FFFC	OE
FFFD	FE
FEE0	XX

At this point we will have to digress to consider the 6502 Manufacturers 'Summary of Simple Cycle Executions' before we can proceed with interpretation of our captured data.

● EXECUTION OF SINGLE CYCLES

The Instruction Set provides sufficient information to program a microcomputer. To investigate intelligently a program being executed we require to know exactly what data appears on the address and data lines for each machine cycle of any given instruction. The 'Summary of Single Cycle Executions' provide this.

This list is included in the Appendix and reference to it should be made. It is fairly self explanatory. Consider A.2.3 reproduced below:

A.2.3 Absolute Addressing (4 cycles)

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch low order Effective Address byte
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

The information given shows the state of the Address Bus, Data Bus and Read/Write (R/W) line for an 'Internal Execution on Memory Data' instruction such as LDX, for each system clock cycle. Using this data, together with that collected using the logic analyser, it is possible to investigate programs in detail and also deduce faults which may be present upon system hardware such as the processor chip, address and data lines.

At this point we are only interested in analysing our data (we will assume no fault exists).

To set you off we will analyse the first few data lines captured after FE0E. These should be:

ADDR BUS	DATA BUS
FEE0	AE
E1	03
E2	F8
F803	F8
FEE3	D0
E4	0E
F3	A0
F4	80

Now the data (AE) stored in FEE0 MUST BE AN OP.CODE. Our instruction set tells us that AE is the Op. code for a Load X-Register from an absolute address instruction. The next two bytes MUST THEN BE the address of the data to be fetched and loaded into the X- Register.

Hence:

<u>ADDR</u>	<u>HEX CODE</u>	<u>SYMBOLIC CODE</u>
FEE0	AE,03,F8	LDX F8 03

However, the actual data appearing as given in the 'Summary of Single Cycle Executions' and confirmed by the Logic Analyser is:

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	FEE0	AE	1	Fetch OP. Code
T1	FEE1	03	1	Fetch low order Effective Address Byte
T2	FEE2	F8	1	Fetch high order Effective Address Byte
T3	FE03	F8	1	Fetch Data
T0	FEE3	D0	1	Next Instruction

From the above, (D0) is the OP. Code of the next instruction which the instruction set gives as a 'Branch Instruction'. Now turn to A.5.8 in the 'Summary of Single Cycle Executions'. The actual instruction is a (BRANCH on NOT EQUAL to zero' (BNE). We are not sure at this point how many machine cycles are involved (2 or 3) since we do not know if a page is crossed by the OFFSET or not, but:

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	FEE3	D0	1	Op. Code
T1	FEE4	OE	1	Offset
T2*	FEF3	A0	1	Op. Code Next Instruction
T3**	⌘			

$$\begin{array}{rcl} \text{⌘} & \left\{ \begin{array}{l} \text{PC} + 2 \\ \text{OFFSET} \end{array} \right. & \begin{array}{r} \text{FEE5} \\ \text{OE} \\ \hline \text{FEF3} \end{array} \end{array}$$

* Last Arithmetic Operation resulting from an Instruction loaded caused the X-Register to be loaded with FF not 00, hence we would expect a Branch to be taken.

** NO CARRY (PAGE) was generated hence we would not expect this cycle.

● Exercise 4.6

Use the Instruction Set and Summary of Single Cycle Executions to extract the actual program being run from FEE0 and for the Data you collated from that obtained using the Logic Analyser in Exercise 4.5.

● Exercise 4.7

Having completed Exercise 4.6 and obtained the program, state what the microprocessor does during the first seven instructions following the loading of the Reset Vectors (FEE0) into the Program Counter.

You may wonder why the processor is made to check F803. Well the answer is that the EMMA Monitor checks to 'see' if the VISA Expansion Module is connected. If it is not then F8 will be returned if it is then 00 will be returned and the Branch not taken.

You should now have a working knowledge of what to expect on the Control, Address and Data lines of a correctly functioning microprocessor in the 65XX family. In the next chapter we will call upon this knowledge to consider Fault and Diagnostic Testing.

POST TEST

Chapter 4

- 4.1 Explain why, if 'Dots' appear on the EMMA Display, it is a fair assumption that 'at least' a basic working system is in operation.
- 4.2 Sketch the relationship between the $R/\overline{W}$ and SYNC signals when executing a 3-Byte Instruction such as STA.
- 4.3 Explain the random activity upon the System Buses immediately following a System Reset.
- 4.4 Quote the memory addresses of the 6502 System Reset Vectors.
- 4.5 Explain the relationship between the 'Instruction Set' and the 'Summary of Single Cycle Executions'.
- 4.6 Quote the EMMA Reset Vectors.

Fault and Diagnostic Testing

Objectives of this Chapter

Having studied this chapter you should:

- Be able to classify Microprocessor Based System Faults.
- Be familiar with the general principles of fault finding.
- Be able to Design a Simple Semi-Automated Test Routine.
- Be able to diagnose faults externally applied to a microprocessor based system.

Introduction

Microprocessor based systems fall into two categories from a fault and diagnostic point of view; those from Production or Repair and those from a working environment with a suspected fault. In either case we should approach the system as being faulty until proved otherwise. Lets consider some of the types of faults which may be present.

● **Classifications of System Faults**

Four can easily be recognised.

● **Wiring Faults**

These may be due to incorrect connections between System Units or Simply Short or Open Circuit faults on a particular unit. It is unlikely that design faults will be experienced unless the system under test is a development or preproduction prototype.

• **Component Failure**

Components fail in many different ways. For example, a component may experience partial Failure - it fails to meet its specification after a period of use or it may fail Catastrophically - it fails to function at all.

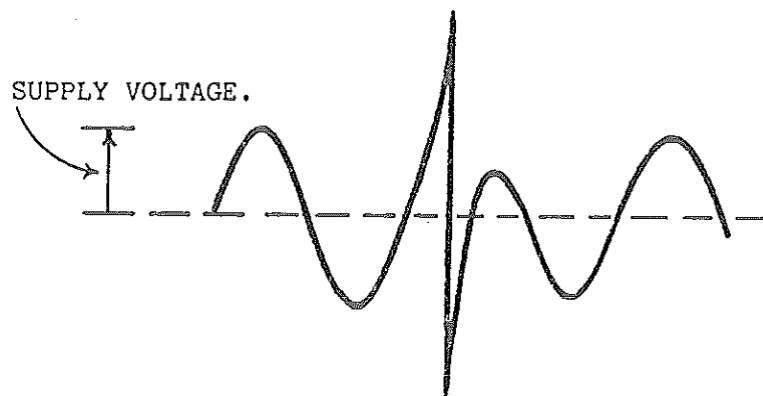
We do not intend to explore this topic further since much has been written regarding the reliability of electronic equipment and many statistical tables produced to help find 'figures of merit' for both components and systems working in various environments. You would be well to acquaint yourself with such tests since, at a very basic level, a knowledge of those components more prone to failure can greatly reduce the time taken to diagnose a fault.

• **Noise Induced Problems**

Noise problems inherent to the equipment should have been overcome by the time equipment reaches the production stage. However, noise is an environmental hazard and is frequently a problem in otherwise well designed equipments.

Typically, trouble can be experienced due to 'switching' other equipments connected to the same power system. The problem arises from the induced 'noise-spike' resulting from the switching action. At the supply termination the 'spike' may be several thousand volts.

Example:



Supply Noise Spike

Disturbances of this nature can filter through transformers, smoothing circuits and regulators, much attenuated, but still sufficiently large to upset the normal running of the equipment. We call these 'glitches'. Faults of this nature may only be diagnosed in the working environment and not on the test bench.

- **Software Bugs**

Software can fall into two broad categories - User Originated and System Manufacture Originated. An example of the latter is the System Monitor. In general, it is reasonable to assume when a User is developing programs but experiencing trouble running them, that it is HIS software at fault rather than the System Software or Hardware. Obviously this is not always so. Don't be too dogmatic!

- **Testing - General Principle**

Systems are normally broken down into convenient sub-systems for test purposes. Having satisfied ourselves at this level it is important to test the system as a whole since this will include terminations and interfaces which may not have been effectively tested at the sub-system stage.

Most testing techniques are based upon a 'Comparison' method. This involves comparing an existing component, board or system with a 'standard'. The standard may be a written specification, a physical piece of hardware or, for microprocessor based systems, software.

The real problem exists not so much in what we want to compare but more in how do we make the comparison.

In our own factory we frequently use existing working 'test' EMMA'S to test other items of equipments which we manufacture.

This would involve designing a dedicated software item to run the test. The test procedure is then made semi-automatic. Even EMMA'S (as you will see) are tested at some stage in this fashion. This type of test will generally, provide a PASS/FAIL result only. Although the point in the test program at which failure occurs may provide the Test Technician with sufficient information to allow him to look in a particular area.

He will then rely upon conventional test equipment to actually pin-point the problem. Obviously success depends to a large extent upon an intimate knowledge of the system under consideration.

Lets consider a Semi-Automated Test Routine. You may even use the developed routine yourself together with others we will suggest you design.

● Design of a Semi-Automated Test Routine

This test routine is designed to test a single component (a RAM chip). it is software based and requires a working EMMA.

The test is based upon the fact that we should be able to both WRITE to, and READ from, Random Access Memory.

The level at which we chose to write the program is entirely up to ourselves. At the lowest level the program may simply WRITE/READ each location, comparing the data Read with the data Written. If the test is successful then it terminates to RESET the EMMA, if not it does not exit from the routine.

At the highest level, the program will cause the Display to indicate either PASS or FAIL together with the addresses of all those locations which failed.

The program below is written to an intermediate level, it simply WRITE's and READ's each location sequentially in a given page of RAM. The 'page' is set-up before the program is RUN. If the test is successful then - PASSED - is displayed. If unsuccessful, the address of the first memory location only to fail is indicated. Display XXXX FAIL.

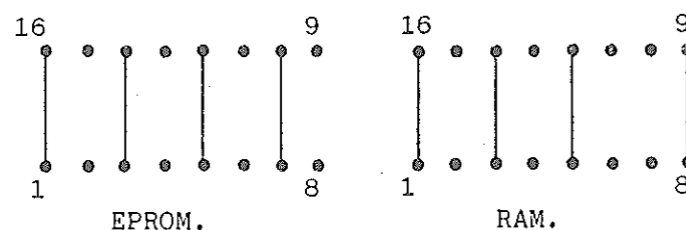
The program is listed below:

0200	A9	LDA	#80	020E	C9	CMP	#FF
0202	85	STA	OE	0210	D0	BNE	0231 (1F)
0204	A2	LDX	#00	0212	A9	LDA	#FF
0206	A9	LDA	#FF	0214	9D	STA	F000,X
0208	9D	STA	F000,X	0217	BD	LDA	F000,X
020B	BD	LDA	F000,X	021A	C9	CMP	#FF

021C	D0	BNE	0231	(13)	023F	4A	LSR	A
021E	E8	INX			0240	A8	TAY	
021F	F0	BEQ	0223	(02)	0241	B9	LDA	FFEA,Y
0221	D0	BNE	0206	(E3)	0244	85	STA	12
0223	A2	LDX	#00		0246	8A	TXA	
0225	B5	LDA	90,X		0247	29	AND	#0F
0227	95	STA	10,X		0249	A8	TAY	
0229	E8	INX			024A	B9	LDA	FFEA,Y
022A	E0	CPX	#08		024D	85	STA	13
022C	D0	BNE	0225	(F7)	024F	A2	LDX	#00
022E	20	JSR	FE0C		0251	B0	LDA	0098,X
0231	A9	LDA	#71		0254	9D	STA	0014,X
0233	85	STA	10		0257	E8	INX	
0235	A9	LDA	#3F		0258	C9	CMP	#04
0237	85	STA	11		025A	D0	BNE	0251 (F5)
0239	8A	TXA			025C	20	JSR	FE0C
023A	29	AND	#F0					
023C	4A	LSR	A		0090	40	73	77 6D
023D	4A	LSR	A		0094	6D	79	5E 40
023E	4A	LSR	A		0098	71	77	30 38

● Exercise 5.1

- Study the program so that you know how it works.
- To test the program place a 2k RAM at F0-F8, You will need to change the Address Decode Header links:



- Enter the page number (say F0) to be tested at location 0020.
- Enter the program at 0200 and RUN from that address.

- Change the Header back to EPROM. Re-run the program. The test should fail at the first location since it cannot WRITE to the EPROM. Display ?

F000 FAIL

- If you can 'burn' EPROMS, then modify the program as below:

```
0212    LDA# FF    (was 00)
021A    CMP# FF    (was 00)
```

Then 'burn' an otherwise 'blank' EPROM (all location at FF) so that one location, (say FOBA) is at FE.

Run the program and notice that the display indicates:

FOBA FAIL

● Exercise 5.2

Refine the program to allow the testing of any desired number of memory locations within a 1k block of memory by designating the first memory location, together with the total number of locations, you wish to test.

● Exercise 5.3

Write a program to test the I/O port ports.

● Basic System Testing

Remember our system may be from the Production line and hence never been 'powered-up' before, returned faulty from its working environment or may have just been repaired but not yet tested.

Either way, it is worth performing a visual inspection of the board before attempting to use any test equipment.

● **Fault Diagnosis**

- Check all the System Control Signals with an oscilloscope. It is pointless using a relatively sophisticated instrument such as a Logic Analyser initially since, if the control signals are not functioning correctly, there is little hope of collecting valid data with an analyser.
- Analyse the System Software being run, using the Logic Analyser. This should pin-point problems associated with the Address and Data Bus or even devices connected to them.
- Use a Logic Probe on items of logic hardware such as the Address Decoder Circuitry and Cassette Interface. It may be advantageous to develop a special program to 'exercise' circuitry of this nature.

Now lets get some practice actually fault finding.

● **Using The Switched Fault Unit**

The switched fault unit plugs into the expansion connector of the EMMA board. It enables ten short circuit faults to be switched into the system, each of which results in different failure symptoms.

Faults 1 to 4 involve System Control Signals whilst faults 5 to 10 involve Data and Address Buses. You should now appreciate that control signal faults can best be approached using the oscilloscope while Data and Address Bus problems generally require or are best tackled with a multi-channel instrument such as the Logic Analyser.

● **The Role of the Schematic Diagram**

You will need to refer to the Schematic Diagram frequently. Apart from telling you what the system does it provides important information such as device pin-out numbers. Copies are provided in the appendix but for your convenience use the one provided at the end of this section.

● Exercise 5.4

- Connect the Switched Fault Unit to the EMMA board and Power-Up.
- Select Fault Number 1.
- Press the RESET pushbutton to run the System Monitor.

OBSERVATION

'No dots' appear on the Display.

Obviously there could be many things incorrect such as:

- a) Missing or incorrect Monitor EPROM.
- b) Missing or incorrect Decode Header.
- c) Missing or incorrect Microprocessor Chip etc., etc.

Since we expect the board to be operating correctly without the fault (presumably it was!) then we would not concern ourselves with these details.

Now let's logically go through the CONTROL SIGNALS.

- The first one we could suspect is the RESET line to the microprocessor. We can check this at the 0.1" pin provided by monitoring either with the Logic Probe or the Oscilloscope. We would expect to 'see' a transition from positive 5 volts. (Reset is passively pulled-up via a 4k7 resistor - Remember!) to zero volts as the RESET pushbutton is depressed.

Even if we see a transition take place at this point there is no guarantee that the signal actually arrives at the microprocessor chip. The only way to be sure is to check!

If you cannot see a transition at the 'test pin' then you should go to the source of the signal - in this case the RESET pushbutton - and progressively trace your way forwards (away from source).

In a larger system (a number of boards connected together) we would adopt what is termed a 'half-split' method. This would involve testing at a point mid-way between source and destination and then continue tracing in the appropriate direction. Consult the schematic diagram and check thoroughly the RESET line.

- Having checked the RESET line thoroughly you will realise that this is not at fault - Sorry!

A better method, rather than test thoroughly a given signal line first, is to quickly monitor all those lines which are likely to affect the running of the Monitor.

Quickly use the oscilloscope to check these. you will notice that they all appear on test pins and are grouped together.

You should find that the System Ø2 Clock Signal is at fault since the line has present a series of positive and negative ringing spikes. The condition suggests that the signal is being fed into a short to the zero-volt line, the spikes highlighting the fact that the connection between signal source and actual fault is inductive.

Now let's see how you manage on your own.

● **Exercise 5.5**

- Remove all other faults and select Fault Number 2.
- Follow a procedure similar to that we used in Exercise 5.6 to diagnosis the fault.

● **Exercise 5.6**

- Fault Number 3.
- Start as for previous Exercises.

● **Exercise 5.7**

- Fault Number 4.

- Proceed as for earlier exercises. This should identify a control line which has a regularly occurring fault. Monitor this line on one channel of the oscilloscope and use the other channel to compare with other lines (including Address and Data).

If you can identify a line with the same trace then you should perform an ohmic test (remove Supply and use suitable meter) to determine whether or not these lines are physically connected together through the fault condition.

● **Exercise 5.8**

- Fault Number 5.

- Proceed as always checking the control lines. However, you will probably soon realise that you require a Logic Analyser. If you suspect a hardware problem remember that this can effectively affect the software being run. You should now proceed by deciding whether or not you can formulate a valid trigger word. If you are still not sure, go back to previous chapters to recapitulate.

If you have managed this, Congratulations! Your learning.

● **Exercise 5.9**

- Fault Number 6.

● **Exercise 5.10**

- Fault Number 7.

● **Exercise 5.11**

- Fault Number 8.

● **Exercise 5.12**

● Fault Number 9

● **Exercise 5.13**

● Fault Number 10.

POST TEST

Chapter 5

- 5.1 Briefly classify the various faults which may be experienced by a microprocessor based system.
- 5.2 Briefly outline the order in which tests should be carried out on a suspected faulty P.C.B. based microprocessor system.

Objectives of this chapter

Having studied this chapter you should:

- Understand a Microcomputer Address Decode System.
- Understand a Standard Cassette Interface.
- Be able to perform a functional test on a Cassette Interface.

Introduction

This chapter considers the miscellaneous circuitry required to effect a working microcomputer system such as EMMA.

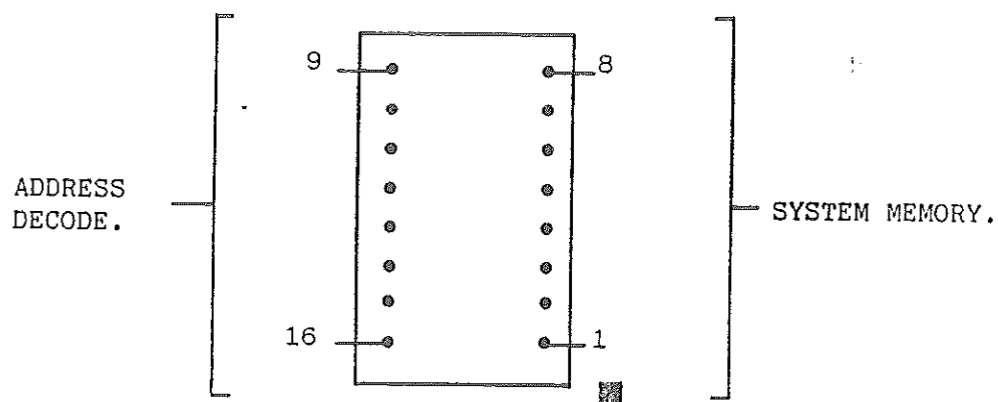
● **Address Decode System**

In the EMMA system, the address lines A11 - A15 are partially decoded to directly address 2k byte blocks of memory. Additional decoding is provided for the user I/O ports and the Keyboard/Display Interface. For the expanded system (VISA), additional decoding is done local to the VISA board.

An 'address decode header' is used on the EMMA which allows the user to change the address at which various memory blocks actually reside. For example, the EMMA reset and interrupt vectors can easily be made to 'appear' at the EMMA User EPROM socket. The facility also enables one to run dedicated system EPROM's simply by changing the decode linkage. In addition to all this, the user EPROM socket can be used to supplement the User RAM. Again by changing the decode linkage.

Decode Header

The 'Standard' Header has links in the Positions shown:



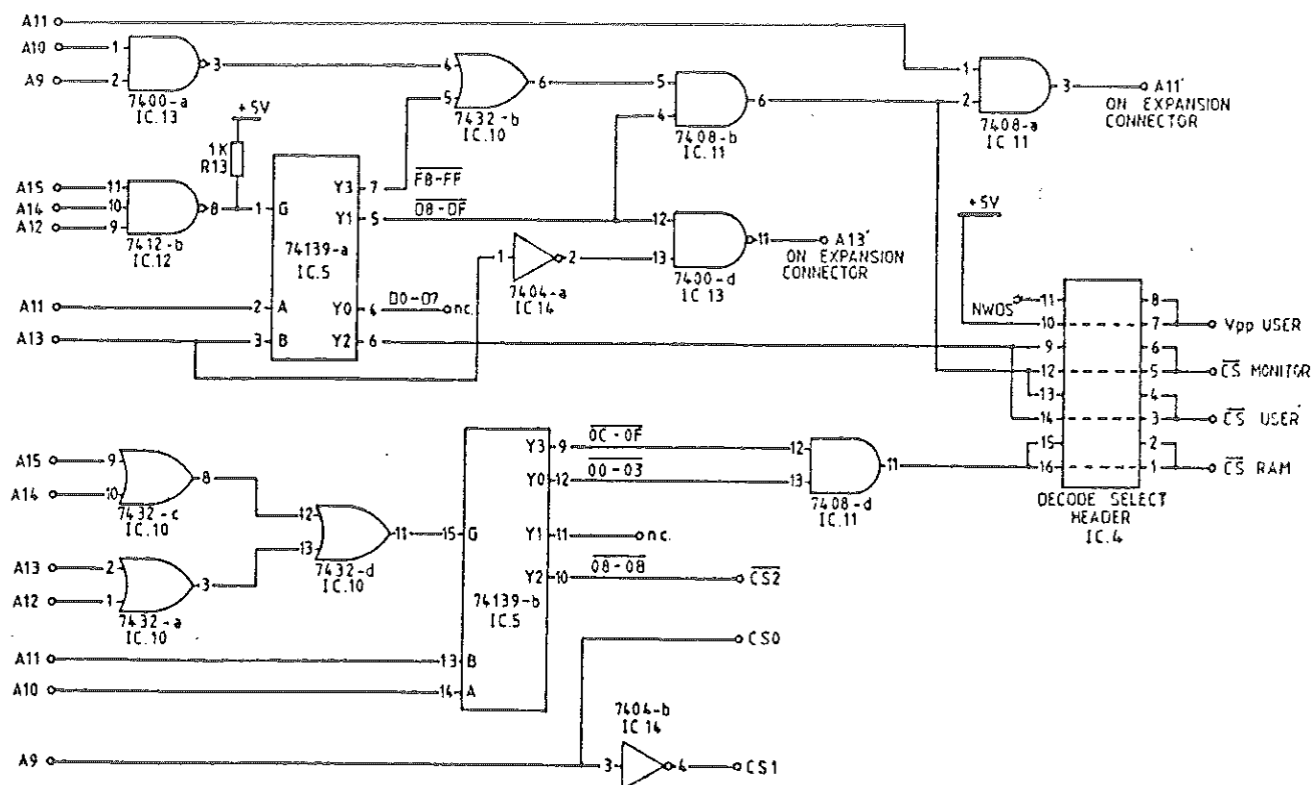
To place the RESET/INTERRUPT vectors at the User EPROM socket then the following link changes are required:

Remove Links	3 - 14
and	5 - 12
Replace at	4 - 15
and	6 - 11

To use RAM in place of EPROM at the User EPROM socket then:

Remove Link	7 - 10
Replace at	8 - 9

Address Decode Circuitry



The actual Decode IC used is a 74LS139 (IC5). This is a dual package providing two (A and B), one-of-four decoders.

The Pin designations are:

- A and B - Output Selects (Inputs)
- G - Chip Enable (Input)
- $Y_0 - Y_3$ - Decoder Outputs

The TRUTH TABLE for this device is:

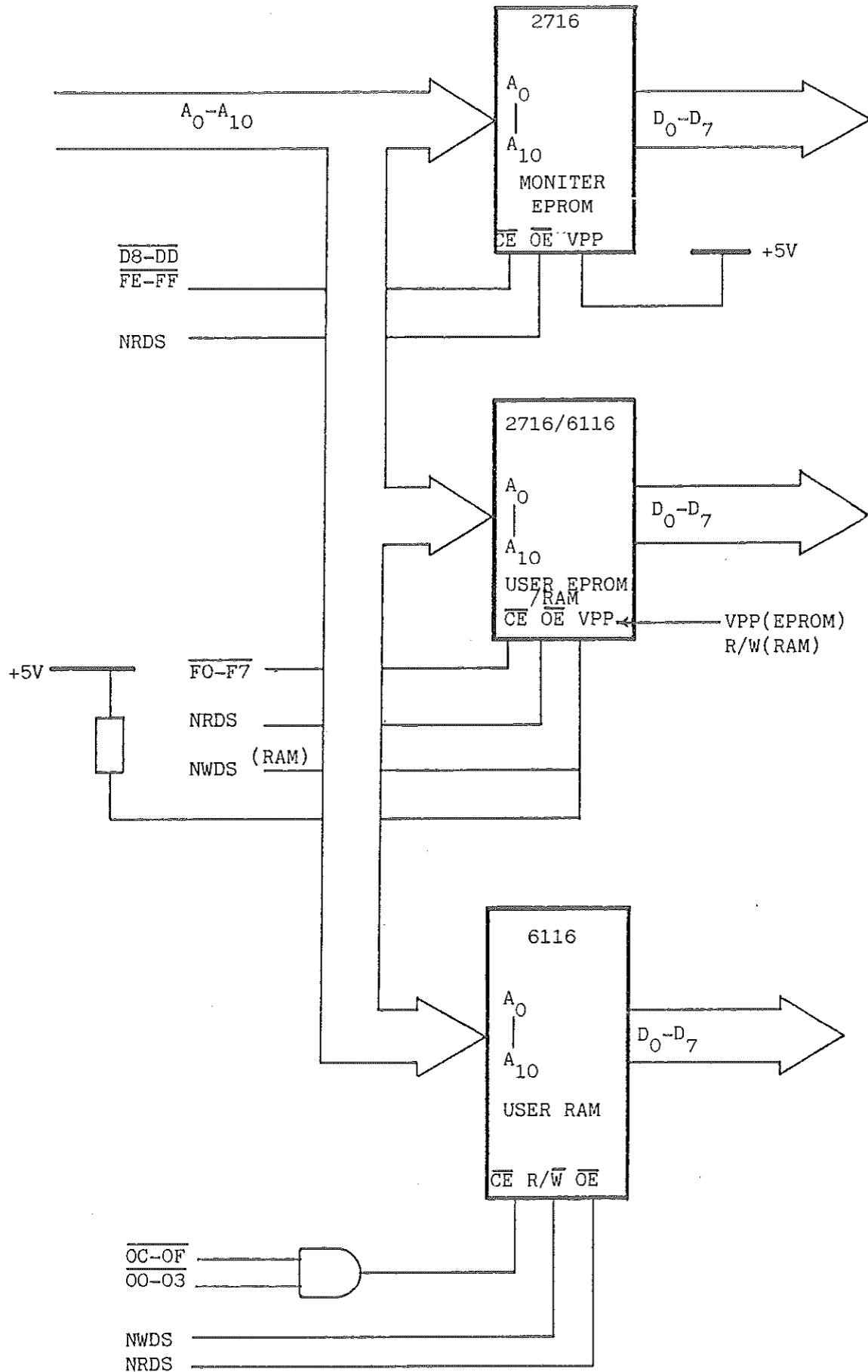
INPUTS			OUTPUTS			
G	A ₀	B ₁	Y ₀	Y ₁	Y ₂	Y ₃
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	H	L	H	L	H	H
L	L	H	H	H	L	H
L	H	H	H	H	H	L

H = HIGH Voltage Level

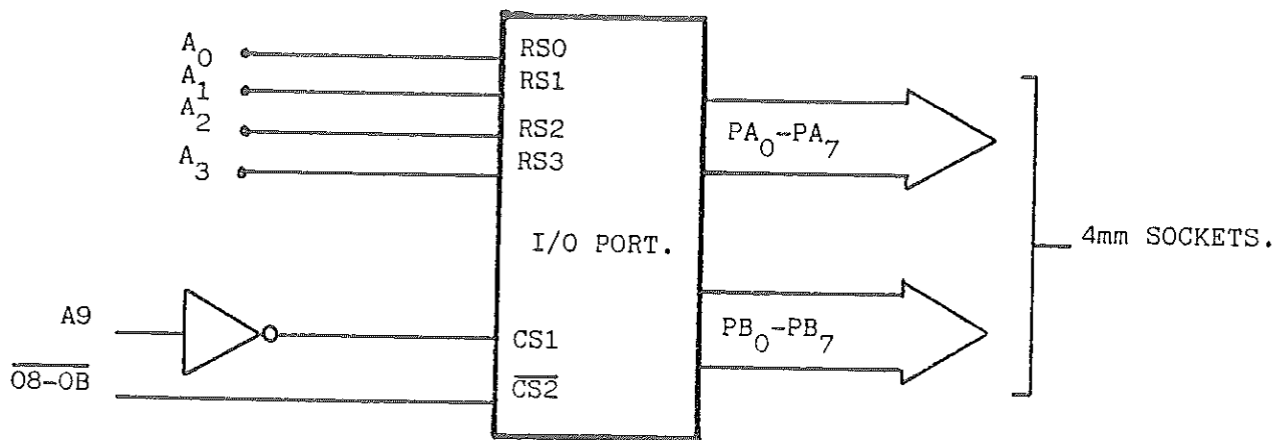
L = LOW Voltage Level

X = Don't Care

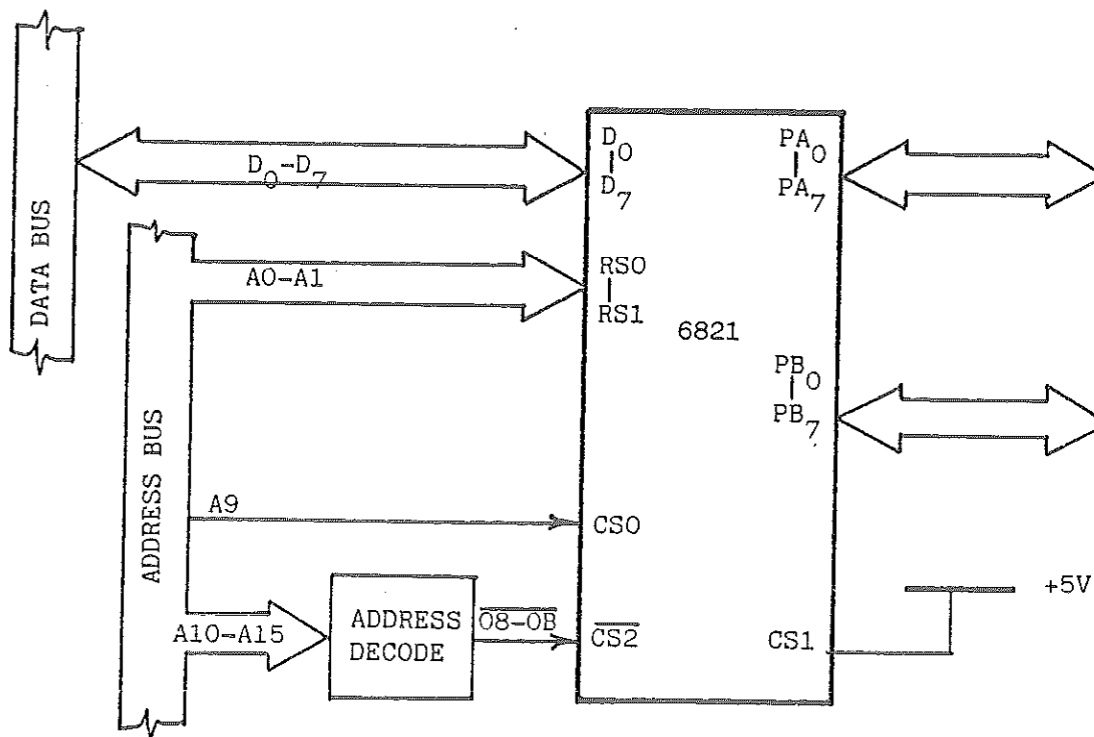
Memory Chip Selects (EMMA)



6522 (USER I/O)



6821 (KEYBOARD/DISPLAY PIA)

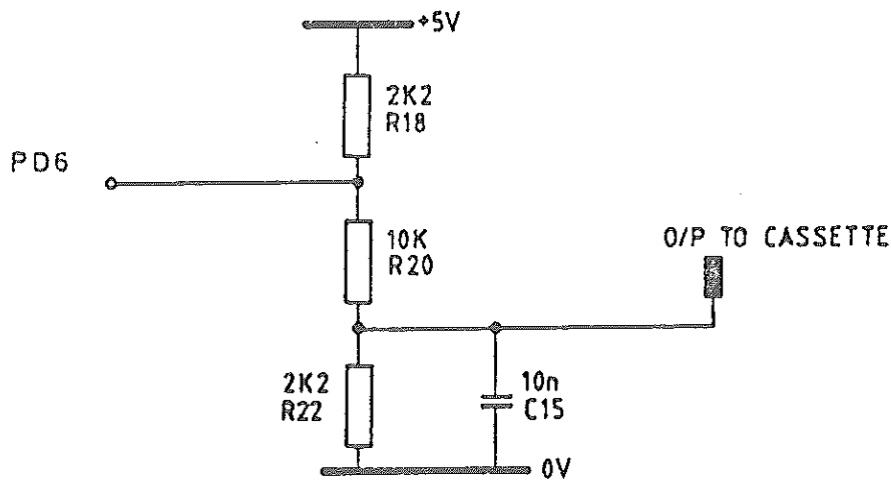


Cassette Interface

The Cassette Interface can be considered in two parts - output to, and input from, a magnetic tape recorder.

From a 'test' point of view we can use the System Monitor to output data from a selected age in memory, RECORD on cassette, PLAY BACK and then examine the memory to determine whether the RECORD/PLAYBACK has been successful. If this fails there are a possible three major areas of failure - The System Monitor, Cassette Recorder or the Cassette Interface itself. At this stage we will interest ourselves only with the Cassette Interface.

Cassette Interface Output Circuitry



OUTPUT TO CASSETTE

The output consists of a simple attenuator and D.C. Level control. The capacitor C15 acts as a simple filter.

The following program will output a 1200 Baud signal of equal mark-space ratio at Port 36 on the 6821 PIA.

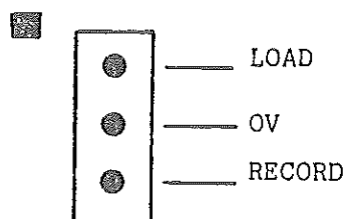
```

0200      A9      LDA# 00
          2      8D      STA 0A03
          5      A9      LDA# 40
          7      8D      STA 0A02
          A      A9      LDA# 04
          C      8D      STA 0A03
          F      A9      LDA# 40
         11      8D      STA 0A02
          4      A2      LDX# A4
          6      CA      DEX
          7      D0      BNE 0216 (FD)
          9      A9      LDA# 00
          B      8D      STA 0A02
          E      A2      LDX# A3
          0      CA      DEX
         21      D0      BNE 0220 (FD)
          3      EA      NOP
          4      4C      JMP 020F
0227

```

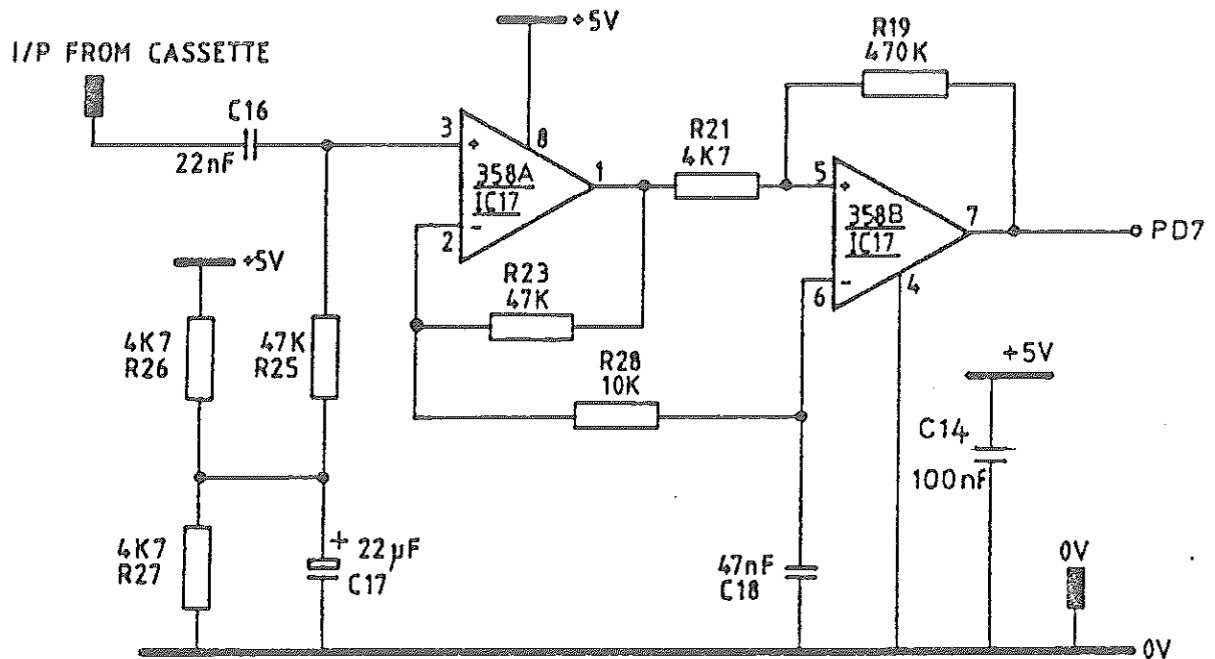
● Exercise 6.1

- Input the program above and record the output voltage waveform at PIN 16 on the 6821.
- Calculate the expected amplitude at the output to the cassette (lower of the three cassette output 0.1" pins).



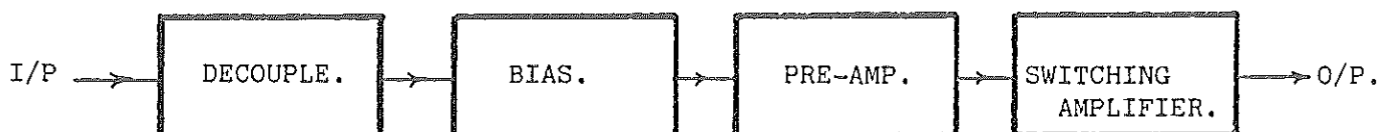
Does your calculated value and measured value agree.

Cassette Interface Input Circuitry



Input from Cassette

The 'input from cassette' interface can be considered by means of the block diagram shown below:



Capacitor C16 effectively decouples the cassette recorder output circuitry from the cassette interface. Bias is obtained via the potential divider network R26/R27 and R25.

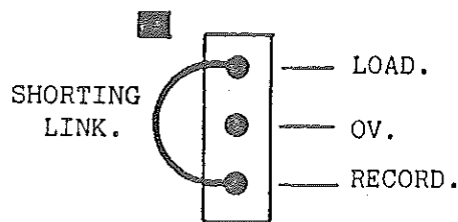
Both non-inverting inputs of the operational amplifiers are held at essentially bias voltage by virtue of the feedback resistor R23 and the capacitor C18.

Any excursion due to a modulating signal being imposed on the bias condition will cause a 'switching' action to take place at PIN 1. However, the feedback by virtue of R23, R28 and C18 and the differentiation effect of C16 will always mean that PIN 1 tends to settle towards bias potential.

The second amplifier stage possesses positive feedback and hence operates as a bi-stable, changing its output (PIN 7) sympathetically with the output (PIN 1) of the pre-amp. PIN 7 output swing is between zero voltage and 3.5 volts. Since this is within T.T.L. specifications no further signal conditioning is required for it to be input to Port B, bit 7 or the 6821 PIA.

● Exercise 6.2

- Run the program given previously to provide an output signal at 1200 Baud.
- Place a shorting link between the two 0.1" pins RECORD/LOAD.



- Use your oscilloscope to record the following signals:

PIN 16	-	6821
PIN 3	}	358
PIN 2		
PIN 1		
PIN 7		



Exercise 6.3

- Change the time-wastes in the program given to obtain a bit-rate of 300 Baud.
- Run your new program and make a magnetic tape recording of approximately two minutes duration of the 'signal'.
- Replay the recorded signal into the cassette interface input. You now have two minutes to make the following observations (re-run tape if necessary).

PIN 3	}	358
PIN 1		
PIN 7		

- Comment upon the shape of the output waveform from the cassette recorder.

POST TEST

Chapter 6

- 6.1 Briefly explain the function of the System Address Decode Circuitry.
- 6.2 Construct a Truth Table for the 74139, one-of-four Decoder.
- 6.3 How many address lines, and which, are normally used to directly address the memory locations of a 2k EPROM.
- 6.4 Which address lines would be decoded to place the EPROM above in a particular place in the Memory Map.
- 6.5 Explain why the output circuitry of the Cassette Interface cannot be 'monitored' by the Logic Probe.
- 6.6 The tests carried out by 'test software' shows the cassette interface to be operating correctly, yet it seems impossible to 'store' a program under normal operating conditions. Explain what may be responsible for this.

CHAPTER 7

Objectives of this Chapter

Having studied this chapter you should:

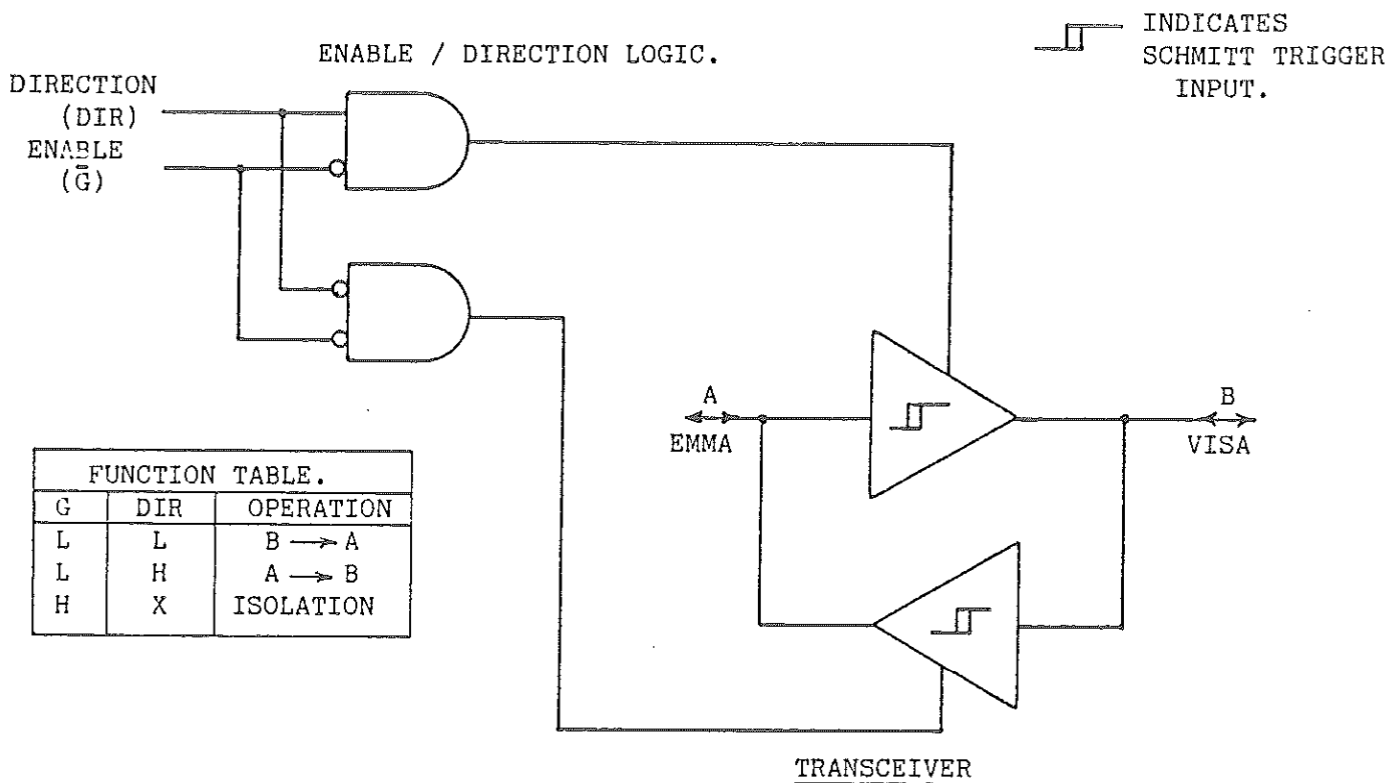
- Understand the problems associated with expanding a system.
- Understand the use of such devices as Bus Transceivers and Line Drivers.
- Understand how Control Signals can be delayed to overcome timing problems.

● **System Expansion**

Problems are likely to arise in any expanded system due to the inability of signal source I.C's to deliver sufficient drive currents. To overcome the problem, Bus Transceivers or Line Drivers are used to enable these lines to effectively source and sink currents in excess of the normal one T.T.L. load drive capability of most microprocessor I.C's. The VISA expansion board employs Line Drivers for the Address and Control Lines and Transceivers for the Data Lines.

• Data Bus Transceivers

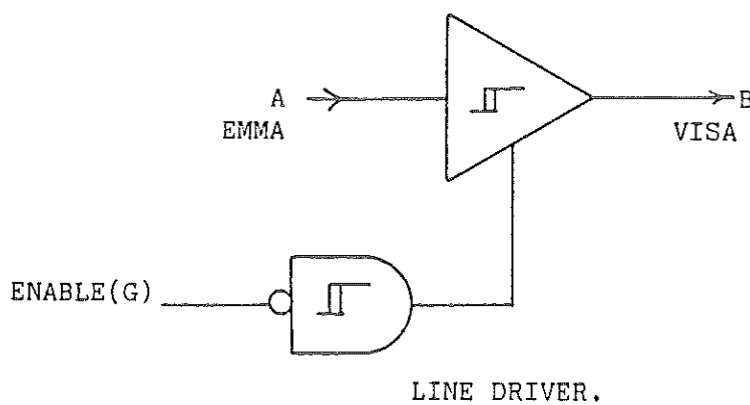
Diagrammatically a Single Data line Transceiver could be represented as:



The device actually comprises two uni-directional, three-state line drivers connected back-to-back. Since the outputs are effectively connected to the inputs and the busses capable of Duplex communication only, then individual Enable (G) and Direction (DIR) Control lines must be provided.

• Address Bus/Control Line, Line Drivers

Diagrammatically:



● Exercise 7.1

- Consult the VISA II Bus Buffering and Address Decoding diagram (in Appendix).
- Use the diagram to answer the following:
 - (a) Identify the Data Bus Transceivers, Address Bus and Control Lines, Line Drivers. For each state the I.C. Reference No and the Manufacturers type number.
 - (b) State how the Address/Control Line Drivers are Enabled.
 - (c) State the direction (A → B or B → A), of the Data Bus during a READ operation.
 - (d) Consult your Technical Manual - Manufacturers Data Sheet Appendix - and determine the SINK and Source Currents for both types of Buffer. How do these compare with the 6502 Microprocessor I.C.

● Address Decoding

Address Decode on the EMMA board was provided by means of a 1-out-of-4 Decode I.C. On the VISA board, programmable Read Only Memories (PROM's) are used. These are preprogrammed by the Manufacturer to a specification provided by the System Development Engineer. The actual PROM coding sheet is reproduced and can be found in the Appendix of the Technical Manual.

● Exercise 7.2

- Consult the PROM coding sheet.
- The Hexadecimal Code 16 appears on the Address Lines A15-A11. Determine the state of the PROM Data output lines D1-D8.

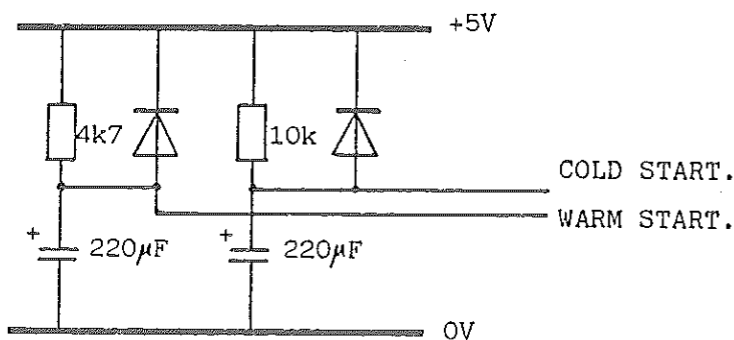
- For the condition above what will be the state of the Address E (ADE) pin on the B-PROM.

NOTE: a) Both PROM's have a SELECT (S) pin taken directly to ground - the PROM's are hence permanently enabled.

b) The outputs, D1, D2 and D3 of the A-PROM have pull-down resistors connected. This ensures that the 6264 RAM CE lines are 'earthy' during power-down conditions so that the current drain is correct for Battery Back-up conditions.

• System Cold/Warm Start Circuitry

The circuitry is reproduced below:



The circuitry provides the RESET conditions for the following I.C's:

Cold Start

VISA Keyboard VIA	(6522)	$\overline{\text{RES}}$
-------------------	--------	-------------------------

Warm Start

EMMA Microprocessor	(6502)	$\overline{\text{RES}}$
EMMA Keyboard VIA	(6821)	$\overline{\text{RESET}}$
EMMA User VIA	(6522)	$\overline{\text{RES}}$
VISA Video Controller CRTC	(6845)	$\overline{\text{RST}}$
EPROM Programmer PPI	(8255)	RESET
Floppy Disc-Controller FDC	(1770)	$\overline{\text{MR}}$

● **System Cold Start**

On initial Switch-on, both cold and warm start lines will be in a low condition effectively resetting internal registers within the various resettable I.C's. The system microprocessor (6502) will begin it's RESET routine by fetching the reset vectors upon receipt of a rising edge on the RES line. This is effectively obtained from the Warm Start line which rises at a faster rate than the Cold Start line. The System Monitor is then entered whereupon it immediately interrogates the VISA keyboard VIA. If the VIA is still in a RESET condition (due to the longer time constant of the Cold Start network) it will assume a COLD START RESET.

● **System Warm Start**

Occasionally, during system operation, it is advantageous to reset the microprocessor without resetting the internal registers of other resettable I.C's. This can be done by use of the EMMA RESET pushbutton 'EMMA' will reset to Monitor which again will check the VISA Keyboard VIA. Since the supply has not been removed, 'reading' the VIA will indicate a non-reset condition. This will then be interpreted as a WARM START.

● **Exercise 7.3**

- Consider the Cold/Warm Start Circuitry.
- Calculate the time-constant for the two circuits.
- How would you expect the circuits to react to a power-down condition.

● **Exercise 7.4**

- From a knowledge of the Time-Constants, set-up an oscilloscope to monitor simultaneously both cold and warm start line. Trigger the oscilloscope externally from the system +5V line.
- Do the observed traces confirm your calculated values for the time constant.

Exercise 7.5

- Set up as for Exercise 7.4 above, but monitor the cold start line immediately before and after the Line Driver (Pins 1A1 and 1Y1 respectively).
- Observe the Schmitts Trigger action of the Line Driver.
- Now observe the outputs 1Y1 and 1Y2. Determine the time interval between warm and cold start buffered lines.

Exercise 7.6

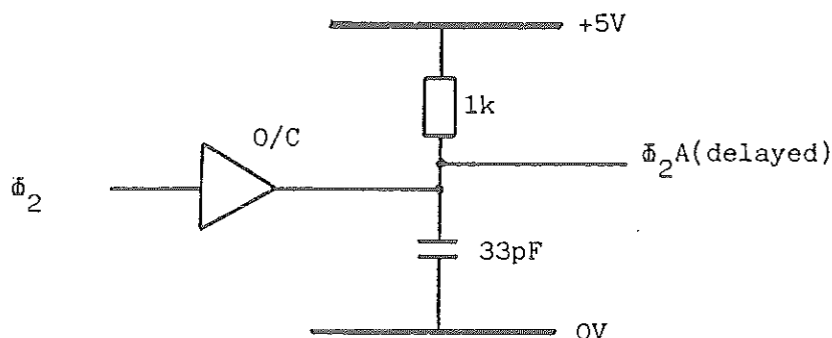
- The RESET line taken to EMMA is provided via a 7406 open-collector gate. Can you suggest a reason for this.

VISA System Clocks

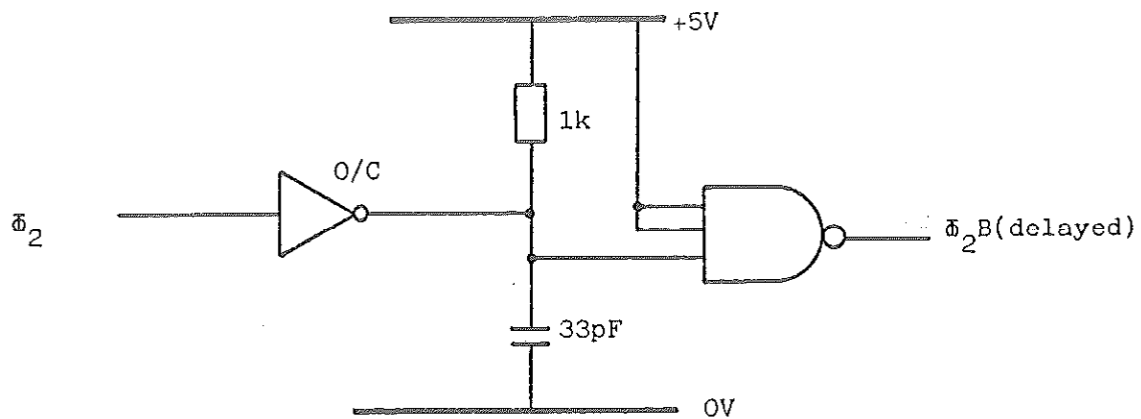
The VISA uses the same system clock as EMMA, namely ϕ_2 . However, two ϕ_2 clocks are generated, one with a delayed leading edge and one with a delayed trailing edge.

The circuitry to perform this is:

- Delayed Leading Edge.



Delayed Trailing Edge.



Exercise 7.7

- Use an oscilloscope to compare the ϕ_2 System Clock with the ϕ_2A and ϕ_2B clocks.
- Determine the respective time delays to the leading and trailing edges.

VISA System Read/Write Line

Two are used, R/W as for EMMA and R/W Delayed.

Exercise 7.8

- What are the differences, if any, between the Delayed R/W circuitry and that for the ϕ_2 delayed clocks?
- Which edge of the R/W is delayed?

POST TEST

Chapter 7

- 7.1 Briefly explain the problems associated with attempting to 'expand' a microprocessor based system.
- 7.2 Explain the difference between 'Line-Drivers' and 'Bus-Transceivers'.
- 7.3 Explain how a PROM (as used on VISA) can take the place of the 74139's (as used on EMMA).
- 7.4 Briefly explain the principle of operation of the Cold and Warm Start circuitry.
- 7.5 Explain, with a diagram if necessary, how basically the same circuitry can be used to delay a clock leading and trailing edge.

● **System Cold Start**

On initial Switch-on, both cold and warm start lines will be in a low condition effectively resetting internal registers within the various resettable I.C's. The system microprocessor (6502) will begin it's RESET routine by fetching the reset vectors upon receipt of a rising edge on the RES line. This is effectively obtained from the Warm Start line which rises at a faster rate than the Cold Start line. The System Monitor is then entered whereupon it immediately interrogates the VISA keyboard VIA. If the VIA is still in a RESET condition (due to the longer time constant of the Cold Start network) it will assume a COLD START RESET.

● **System Warm Start**

Occasionally, during system operation, it is advantageous to reset the microprocessor without resetting the internal registers of other resettable I.C's. This can be done by use of the EMMA RESET pushbutton 'EMMA' will reset to Monitor which again will check the VISA Keyboard VIA. Since the supply has not been removed, 'reading' the VIA will indicate a non-reset condition. This will then be interpreted as a WARM START.

● **Exercise 7.3**

- Consider the Cold/Warm Start Circuitry.
- Calculate the time-constant for the two circuits.
- How would you expect the circuits to react to a power-down condition.

● **Exercise 7.4**

- From a knowledge of the Time-Constants, set-up an oscilloscope to monitor simultaneously both cold and warm start line. Trigger the oscilloscope externally from the system +5V line.
- Do the observed traces confirm your calculated values for the time constant.

Exercise 7.5

- Set up as for Exercise 7.4 above, but monitor the cold start line immediately before and after the Line Driver (Pins 1A1 and 1Y1 respectively).
- Observe the Schmitts Trigger action of the Line Driver.
- Now observe the outputs 1Y1 and 1Y2. Determine the time interval between warm and cold start buffered lines.

Exercise 7.6

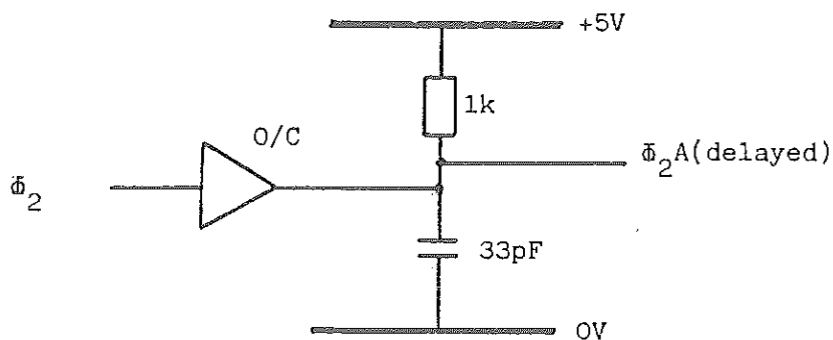
- The RESET line taken to EMMA is provided via a 7406 open-collector gate. Can you suggest a reason for this.

VISA System Clocks

The VISA uses the same system clock as EMMA, namely Φ_2 . However, two Φ_2 clocks are generated, one with a delayed leading edge and one with a delayed trailing edge.

The circuitry to perform this is:

- Delayed Leading Edge.



Objectives of this chapter

Having studied this chapter you should:

- Understand a typical Keyboard Matrix for a microprocessor based system.
- Understand a typical Floppy Disk Drive Interface System.
- Be able to write test routines to exercise both keyboards and floppy disk controllers.

System Interfaces

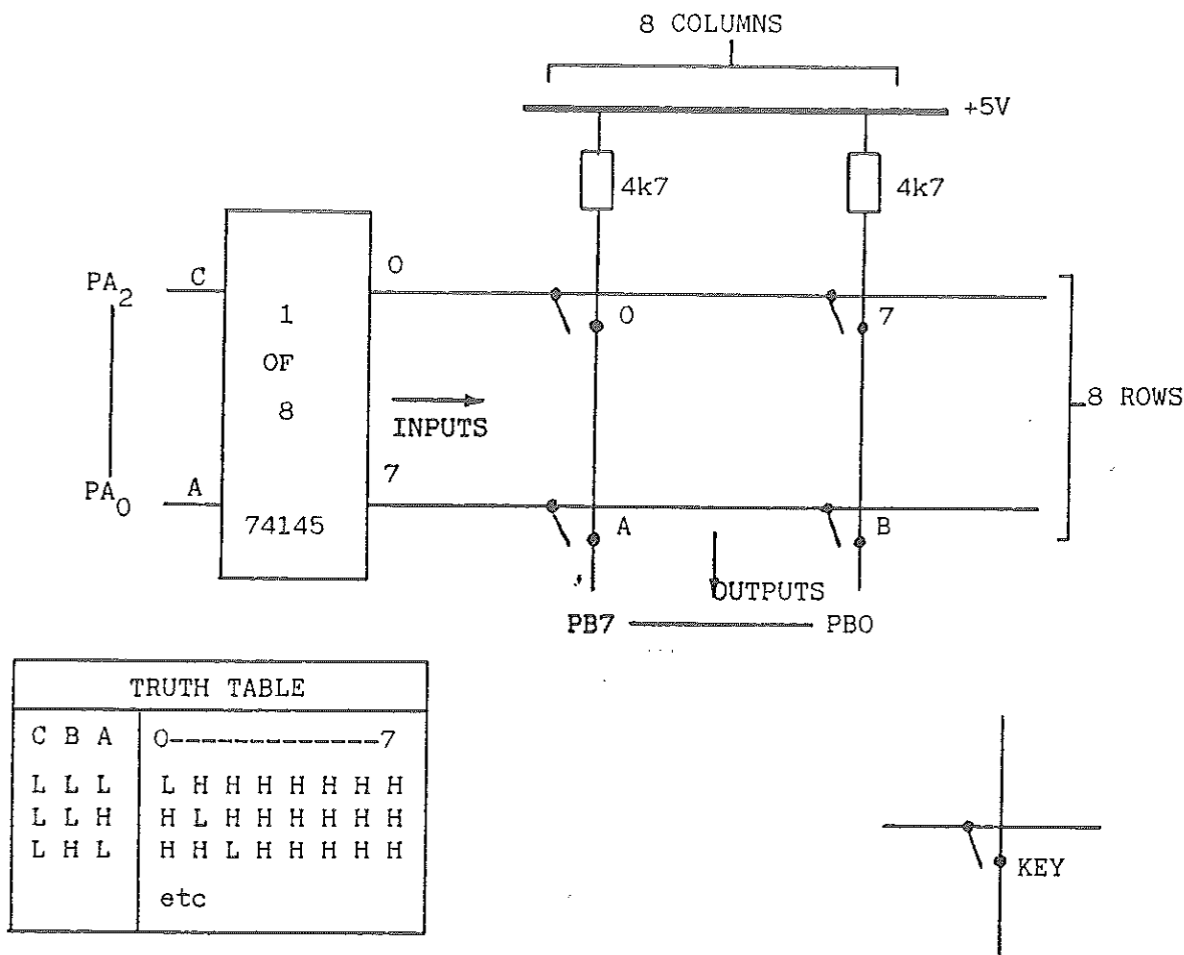
For the microprocessor to be of any use it must interface to other input/output devices. The VISA system is representative of any microcomputer configuration in that it provides for direct interface to a VDU, QUERTY keyboard, Floppy Disc etc. We will examine each of these in turn including an interface for Eprom Programming - a feature not normally found on most microcomputer boards.

● **Keyboard Interface**

The keyboard interface is not dissimilar to that used for EMMA. The main difference is that EMMA uses a simpler VIA (6821) while the VISA uses the more sophisticated 6522.

The principle of operation is as follows:

The keyboard is arranged in an 8 x 8 matrix array giving 8 columns and 8 rows. The columns are scanned by 8 bits of Port B of the 6522 while successive rows are pulled low (walking-zero), by a one-out-of-eight. Decoder (74145).



The method is to look for coincidence of bit pattern on the rows and columns.

Lets assume that a binary count is input to the decoder from PA0 - PA2. The decoder will then output a 'walking-zero'. Now, if key 'B' is depressed, when the 'zero' reaches row 5, PB5 of the 6522 will be pulled low. The codes on the I/O ports will be:

	7	6	5	4	3	2	1	0	
PORT A	X	X	X	X	X	H	L	H	(OUTPUT)
DECODER	H	H	H	L	H	H	H	H	
PORT B	H	H	L	H	H	H	H	H	(INPUT)

The actual decoding of the keyboard is performed in software (Keyboard Sub-Routine, System Monitor). The procedure is complex since it must consider the possibilities of Key-bounce and Key-Roll-Over. Briefly, after a system RESET (Cold or Warm start), the monitor enters the Keyboard Scan Routine which effectively repeatedly strobes the keyboard rows. When a key is

depressed, this will be detected by Port B reading the columns. The row strobe will immediately be stopped with a low held at the row in which the pressed key is detected. The Keyboard Routine will now, after a suitable time delay, read the columns until the same key-code is returned ten times. This effectively debounces the key. The Keyboard Routine is then entered again to check that the key is still pressed, if so, then the routine scans for levels of shift e.g. SHIFT, CTRL and FN. No debounce is provided for these keys since it would be normal to press one of these (if required) before pressing other keys.

The key or keys are then decoded and a unique Key Code returned to the accumulator.

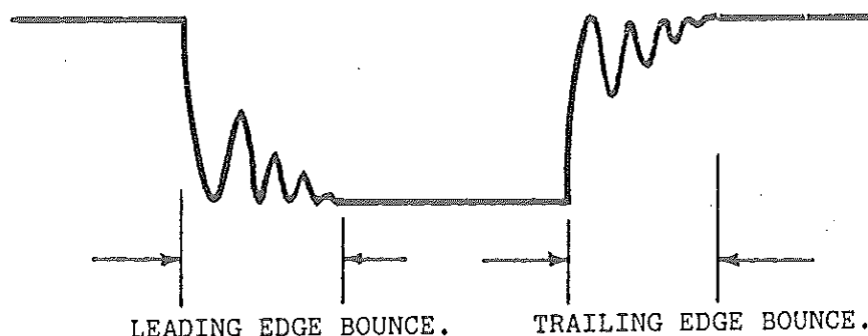
● Exercise 8.1

- With the Monitor running from a RESET condition, use an oscilloscope to determine the relationship between inputs A, B and C to the 74LS145.
- From the timing diagram obtained predict, using the 'truth table' for the 74LS145, the relationship between the inputs and outputs.
- Use an oscilloscope to try and monitor these. You will probably find that the only 'sensible' trace will be with a sensitivity of 0.5 Volts/cm.

This is probably not what you would have expected. Why? (Hint - consider the output configuration for the 74LS145).

● Keyboard Test Routines

Since the Keyboard Routine is complex it may be to our advantage to develop a simple routine to test the keyboard. First lets consider what happens when the mechanical contacts of a key are made to close. They 'bounce' for a short period of time before staying together.



The problem can be overcome using either hardware or software to effectively delay reading a key as valid until 20mSec have elapsed.

● Exercise 8.2

- Develop a program using EMMA RAM, to strobe continuously the Keyboard Matrix Array rows.

Note: 6522 (IC32)

<u>REGISTER</u>	<u>PORT</u>	<u>ADDR</u>
Data Register	B	8800
Data Register	A	8801
Direction Register	B	8802
Direction Register	A	8803

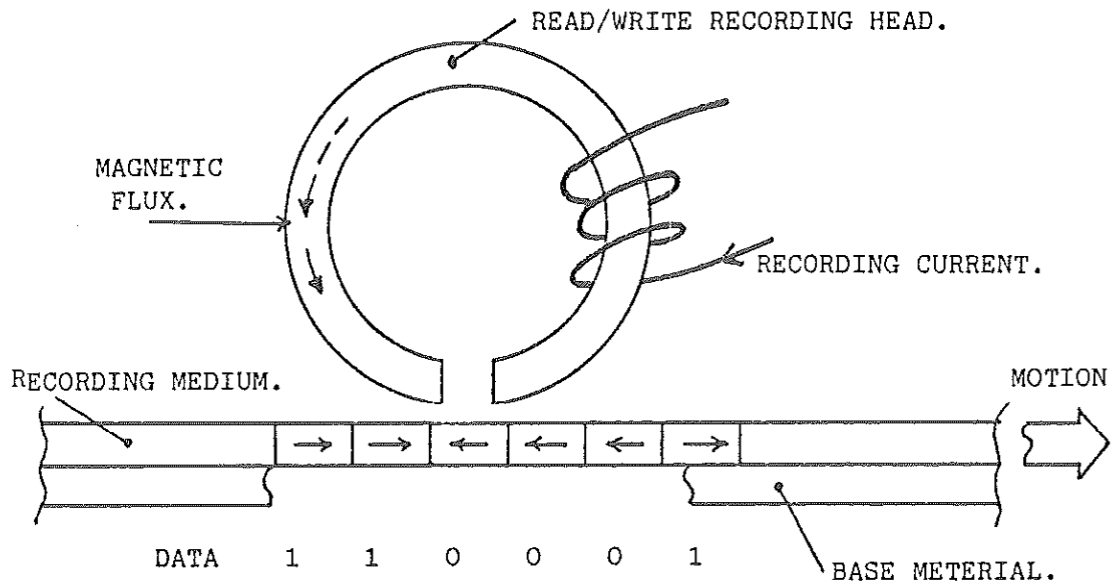
- Run the program and use a logic probe to check the signal at the input of the 74145 (IC33).
- Use an oscilloscope/probe to check a key return at each of the column outputs. Check at both PBO - PB7 and KCO - KC7.

The 0.1" Test Pins (KRO - KR7 and KCO - KC7) immediately below the 74145 provide suitable monitoring points for the Rows and Columns respectively.

● Floppy Disc Interface

• General

Metallic oxide films are convenient materials for storing large amounts of data in the form of magnetic field. The direction of magnetic field determines the data bit logic level.

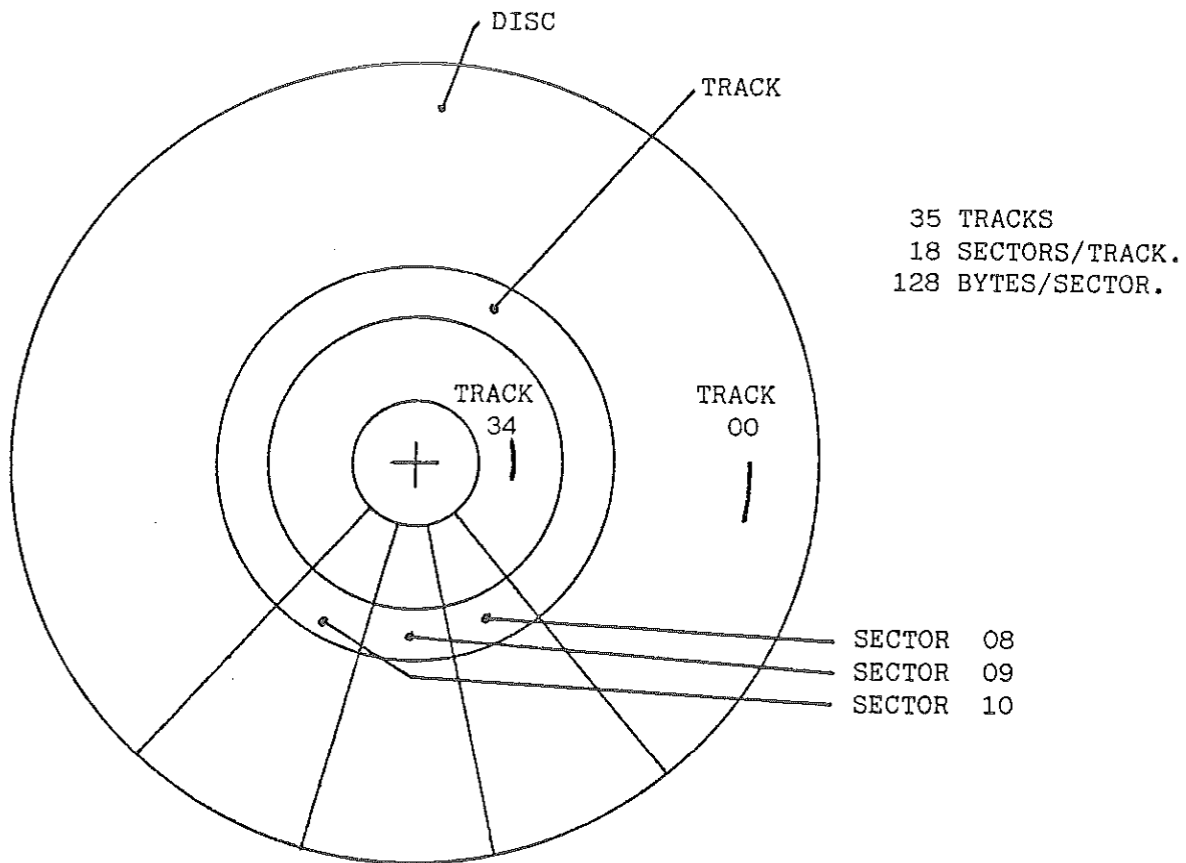


The principle is the same regardless as to whether the Recording Media is on a 'tape' base or 'disc' base. Data is 'written' to the base coating simply by passing it beneath the recording head in which a magnetic flux is established due to 'data' current flowing through it in the appropriate direction. The reverse process takes place to 'read' data, the flux in the recording head now being established by virtue of movement of the recorded medium and previously recorded magnetically stored data.

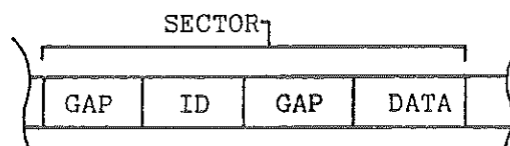
Our prime concern is with 'disc' rather than 'tape' and in particular the 'mini-floppy' which is simply a 5 1/4" disc of Milar forming the base onto which a suitable coating of metallic oxide is deposited. The disc itself rotates within a protective sleeve in which a radial disc access cut-out is formed to allow the recording head to make contact with the disc.

An indexing hole is also provided to the disc/sleeve. The disc itself is rotated and recorded upon by means of a 'disc-drive'.

The disc is artificially split-up into track and sectors.



Each sector will be format as:



Exercise 8.3

- Calculate the total number of bytes which a mini-floppy disc can store.
- Does this represent the maximum amount of 'User-data' which the disc can accommodate?

Data transfer rates are determined by the Disc Drive and can be summarised as follows:

Transfer Rate:	125K.BITS/SECOND (SINGLE)
	250K.BITS/SECOND (DOUBLE)
Access Time:	40mS Track-to-Track
Settling Time:	10mS.
Head Loading Time:	75mS.
Rotational Speed:	300 r.p.m.

● **Disc Drive Mechanism**

The disc drive provides all the mechanical functions and the electronics required to directly rotate the diskette and access the recorded data.

The recording head is moved radially across the disc by means of a stepper motor. Once the head is in the correct position it is lowered so as to make contact with the disc. Recording/Retrieval of data can then take place.

For satisfactory operation the disc-drive as a unit requires the following.

● **Drive Select:**

Selects Drive number 1 or 2 if mechanism is dual.

● **Side Selects:**

Selects Disc side if double sided discs are in use.

● **Step Pulse (STEP):**

Steps the drives R/W head from track to track.

● **Direction (DIRC):**

Sets the direction in which the recording head is to travel.

- **Motor ON (MO):**

Enables spindle motor prior to READ, WRITE or STEP OPERATIONS.

- **Write Gate (WG):**

Used to determine the R/W operation of the recording head.

- **Write Data (WD):**

Serially presented clock and data pulses to be written to disc use this line.

The disc drive itself generates the following signals for use by the microcomputer system to which it is indirectly connected via a device controller.

- **Write Protect:**

Provided by an optical device which sense the presence or otherwise of a 'write-protect' tab.

- **Index Pulse (IP):**

Indicates that the physical index hole on the disc has been encountered.

- **Track 00 ($\overline{\text{TR00}}$):**

Indicates that the recording head is over track 00 (outertrack)

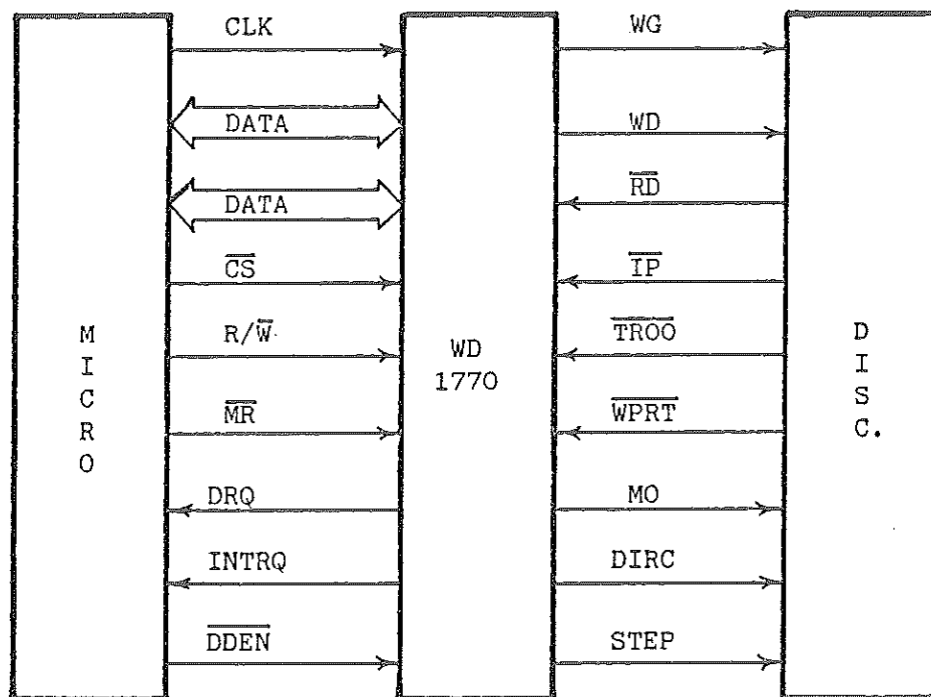
- **Read Data ($\overline{\text{RD}}$):**

Serially presented clock and data pulses provided from disc via the recording head.

• Disc Drive/Interface

The actual interface is provided by a Floppy Disc Controller/Formatter (WD1770). The only additional devices required are buffers for some of the controller - Disc Drive interconnections. 7406 inverting Drivers are used for this purpose.

The block diagram below indicates the position of the Device Controller relative to the Microcomputer and the Disc Drive Mechanism.



The signal lines between the Disc Drive Unit and its Device Controller have already been outlined. Below are listed the function of the signal lines between the Microcomputer and the Device Controller.

• Clock (CLK):

System Clock (8MHz)

• Data (D0 - D7):

Eight bi-directional bus lines used for Raw Data, Control Commands and Status. The lines are capable of driving one standard TTL load are three-state and enabled by the $\overline{CS}$ and $R/\overline{W}$ signals.

- **Address (A0 - A1):**

Used for internal register select:

$\overline{CS}$	A1	A2	R/ $\overline{W}$ = 1	R/ $\overline{W}$ = 0
0	0	0	STATUS	COMMAND
0	0	1	TRACK	TRACK
0	1	0	SECTOR	SECTOR
0	1	1	DATA	DATA

- **Chip Select ($\overline{CS}$):**

Selects Device (WD1770)

- **Read/Write (R/ $\overline{W}$):**

Control direction of data flow from or to selected register.

- **Master Reset ($\overline{MR}$):**

Resets device and initialises the STATUS REGISTER. Has internal pull-up.

- **Data Request (DRQ):**

Indicates that the DATA REGISTER is full (READ) or Empty (WRITE).

- **Interrupt Request (INTRQ):**

Line SET (high) at the completion of any COMMAND or RESET upon a READ of the STATUS REGISTER.

- **Double Density Enable ($\overline{DDEN}$):**

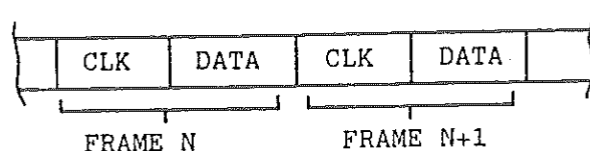
Selects Single or Double density writing to disc.

Architecture - Floppy Disc Controller

Introduction

Before considering in detail the operation of the controller we need some idea of the format of the recorded data. We have already seen that information within a sector is stored with a gap between a sector ID and the actual data recorded within the sector.

All data is recorded in binary format, normally either a logic '0' or a logic '1'. There is no intermediate state - the method is referred to as Non-Return to zero (NTZ). Additionally Frequency Modulation (FM) is used with each data bit appearing in the middle of a 'frame' sandwiched between two successive clock pulses. Diagrammatically:

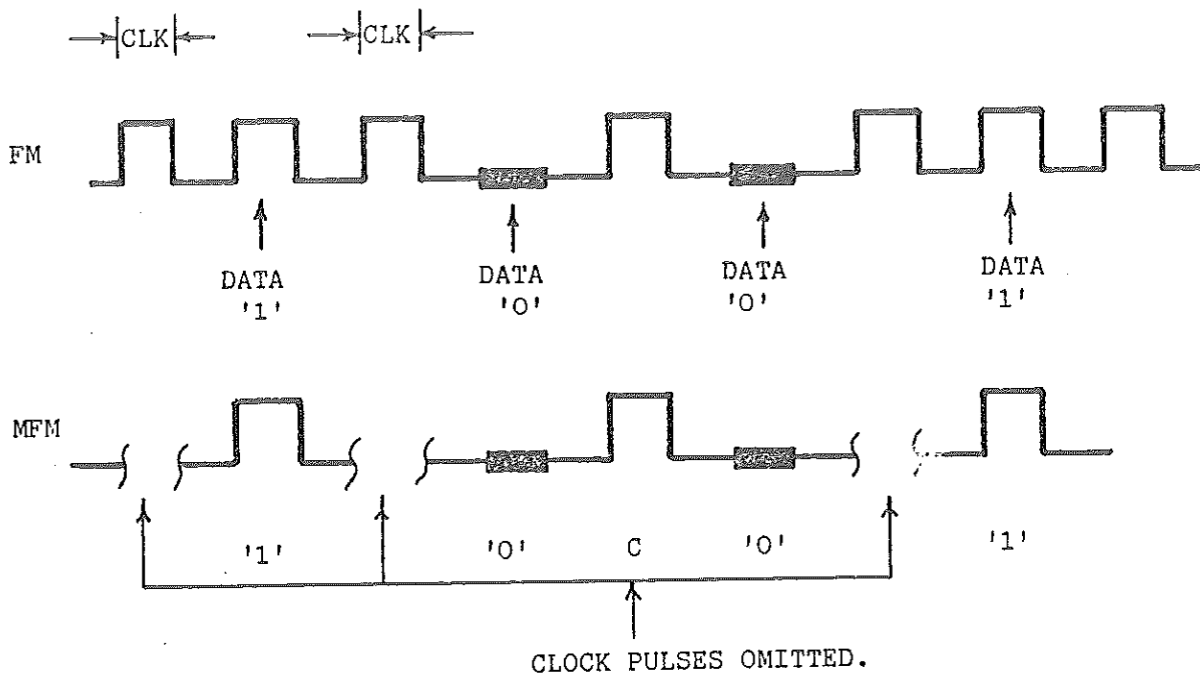


For a data transfer rate of 125k Bits/Sec (single Density), each 'frame' will have a period of 8 s.

To increase the bit density it is possible to eliminate repetitive clock or data bits using a method termed Modified Frequency Modulation (MFM). The principle is that:

- a) Data must still appear in the middle of a Bit Frame.
- b) The Clock Bit is written at the beginning of a frame only if two conditions are met:
 - NO Data Bit will appear in a current Frame and,
 - there was NO Data in the previous Frame.

Example:



Exercise 8.4

Draw Recorded Data diagrams for the following Data Streams:

. 1 0 1 0 1

. 1 0 0 0 1

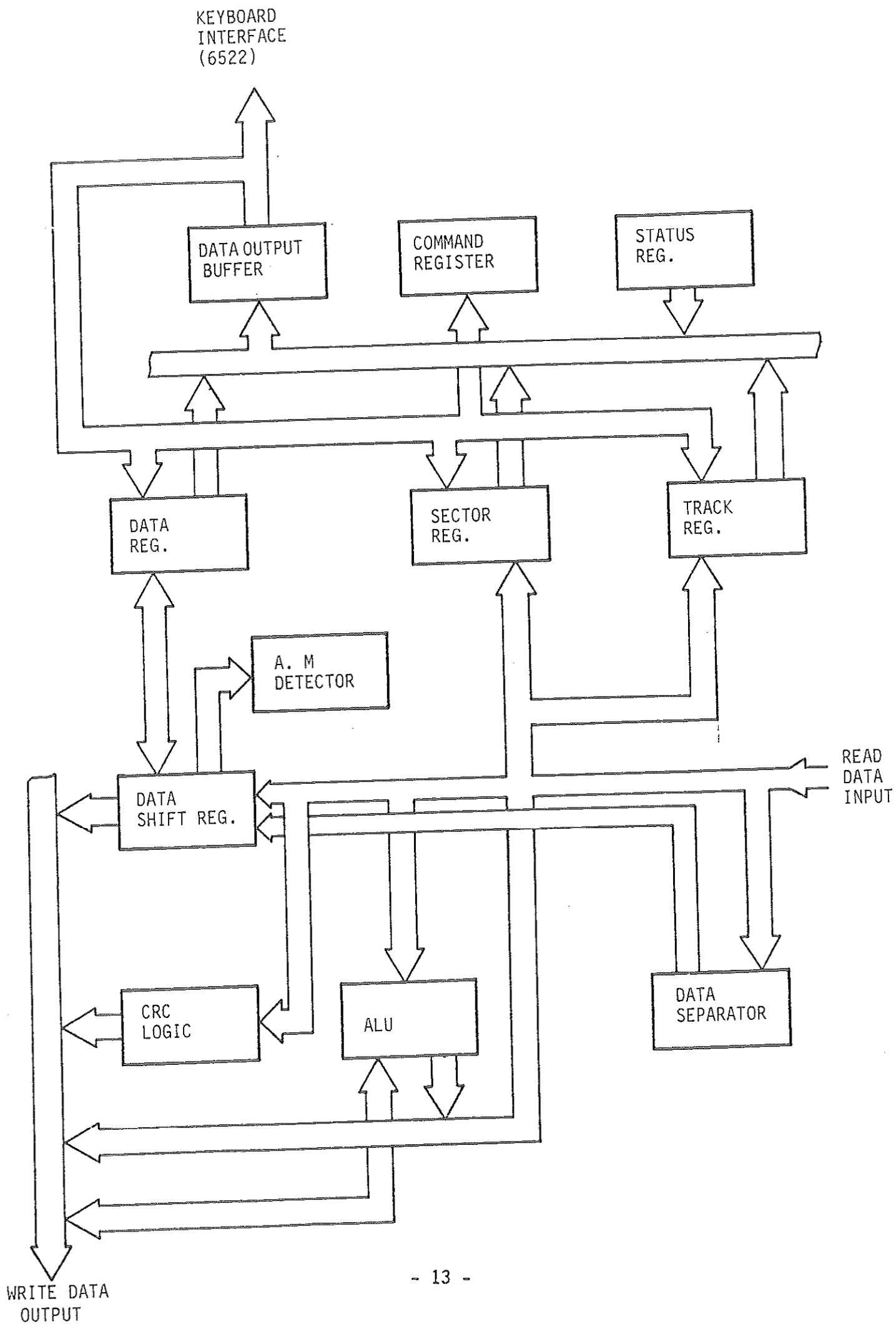
. 1 1 1 0 1

The Architecture of the Floppy Disc Device Controller (WD1770) is shown on the following page.

The device comprises a number of registers the function of each, referring to the diagram, is as follows:

Data Register

An eight-bit register which is used to hold Data during Disc READ and WRITE operations. Data enters the register via the Keyboard PIA (6522) for writing to Disc or via a Data Shift Register when Reading from Disc. When executing the SEEK command, the Data Register holds the address of the desired Track position.



- **Data Shift Register**

An eight-bit register which assembles serial data from the Read Data (RD) input during Disc Read operations and transmits serial data to the Write Data (WD) output during Disc Write operations.

- **Track Register**

An eight-bit register which holds the track number of the current Read/Write Head position. The register is incremented for each step of the Head inwards and decremented for each step outward (towards track 00).

The contents of the register are compared with the recorded track number in the ID field during Read, Write and Verify modifications. The register can be loaded from or transferred to the Data Access Lines. The register should not be loaded when the Device is busy.

- **Sector Register**

An eight-bit register which holds the address of the desired sector position. The contents of the register are compared with the recorded sector number in the ID field during Disc Read and Write operations. The register can be loaded from or transferred to the Data Access lines. The register should not be loaded when the Device is busy.

- **Command Register**

An eight-bit register which holds the COMMAND code currently being executed. The register should not be loaded when the device is busy unless the new command is a forced interrupt. The register can be loaded but not Read via the Data Access lines.

- **Status Register**

An eight-bit register which holds the device STATUS information. The meaning of the status register is a function of the type of command previously executed. The register can be Read but not loaded via the Data Access lines.

In addition to the above registers two further logic units are provided:

- **Arithmetic/Logic Unit**

This is a serial comparator, incrementor and decrementor. Its function is to modify registers and perform comparisons with the Disc recorded ID field.

- **Cyclic Redundancy Check Logic**

The logic embraces an 16-bit CRC register which is preset to ones prior to data being shifted through the circuit. The logic generates or checks the CRC code which is used to detect errors in a block of words.

- **Timing and Control**

Internal device timing is generated from an external crystal clock operating at 8MHz. The ADDRESS MARK DETECTOR is also part of this circuitry as is the DATA SEPARATOR.

- **Address Mark Detector**

This detects the ID, Data and Index address marks during Read and Write operations.

- **Data Separator**

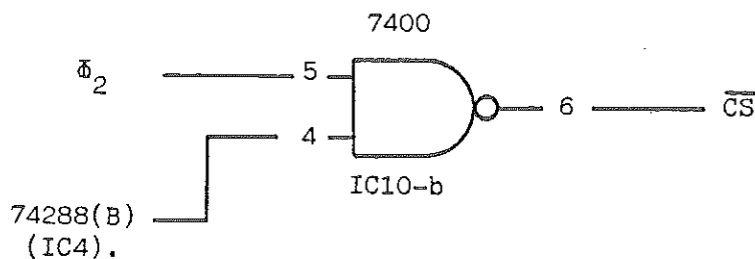
This is a digital data separator which consists of a Ring Shift Register and Data Window detection logic. The separator provides Read Data and a recovery clock to the Address Mark Detector.

- **Signal Investigation**

Now we have some knowledge of the Floppy Disc System we will investigate some of the control signals. Leave Floppy Disc Drive Unit Disconnected.

- **Chip Select ($\overline{CS}$)**

Chip select is provided by a Decode PROM and gated by the Φ_2 Buffered Clock:



- **Exercise 8.5**

- Consult the PROM coding sheet (Appendix) to determine the address of the Floppy Disc Controller (FDC) Chip Select.
- Write a short program to repeatedly select the FDC $\overline{CS}$ line, by loading the accumulator from the $\overline{CS}$ address. Write your program from 0C00 using the EMMA Hexadecimal Keys.
- Determine, from your program the repetitive pulse rate of your $\overline{CS}$ signal.
- Examine, together with the Φ_2 clock, the gated and ungated CS lines.
- Examine gated $\overline{CS}$ at the WD1770.

● Exercise 8.6

- Monitor the following signals at the WD1770.
- DCLK - Determine the frequency of this clock signal.
- R/W - Run the EMMA Monitor.

● Exercise 8.7

- Write a series of short programs to repeatedly select:
 - . Drive 1 , Side A
 - . Drive 1 , Side B
 - . Drive 2 , Side A
 - . Drive 2 , Side B
- Run program and monitor with an oscilloscope at the inputs to the Drivers immediately prior to the Floppy Disc Drive connector.
- Explain why the signals cannot be monitored at the Floppy Disc Drive connector without a slight modification or connection of the Drive Unit.

● Exercise 8.8

- Check, using the oscilloscope that the MASTER RESET ($\overline{MS}$) can be pulled low by means of the EMMA RESET pushbutton.
- Check that the Double Density Enable ($\overline{DDEN}$) pin is in fact pulled-up. Defaults to Single Density.

• Drive Commands

We will now try to exercise the Floppy Disc Controller Device itself. This will involve writing to various registers within the FDC. The registers involved and their addresses can be seen from the table overleaf.

ADDR	READ ($R/\bar{W} = 1$)	WRITE ($R/\bar{W} = 0$)
8C 00	Status Register	Command Register
8C 01	Track Register	Track Register
8C 02	Sector Register	Sector Register
8C 03	Data Register	Data Register

The COMMAND Register will accept eleven commands. Commands are divided into four types and are summarised below:

TYPE	COMMAND	7	6	5	4	3	2	1	0
I	Restore	0	0	0	0	h	V	r ₁	r ₀
I	Seek	0	0	0	1	h	V	r ₁	r ₀
I	Step	0	0	1	u	h	V	r ₁	r ₀
I	Step-in	0	1	0	u	h	V	r ₁	r ₀
I	Step-out	0	1	1	u	h	V	r ₁	r ₀
II	Read Sector	1	0	0	m	h	E	0	0
II	Write Sector	1	0	1	m	h	E	P	a ₀
III	Read Address	1	1	0	0	h	E	0	0
III	Read Track	1	1	1	0	h	E	0	0
III	Write Track	1	1	1	1	h	E	P	0
IV	Force Interrupt	1	1	0	1	l ₃	l ₂	l ₁	l ₀

FLAG SUMMARY

TYPE I COMMANDS			
h = Motor On Flag (Bit 3)			
h = 0, Enable Spin-Up Sequence			
h = 1, Disable Spin-Up Sequence			
V = Verify Flag (Bit 2)			
V = 0, No Verify			
V = 1, Verify on Destination Track			
r₁, r₀ = Stepping Rate (Bits 1, 0)			
r ₁	r ₀	WD1770	WD1772
0	0	6 ms	2 ms
0	1	12 ms	3 ms
1	0	20 ms	5 ms
1	1	30 ms	6 ms
u = Update Flag (Bit 4)			
u = 0, No Update			
u = 1, Update Track Register			

TYPE II & III COMMANDS	
m = Multiple Sector Flag (Bit 4)	
m = 0, Single Sector	
m = 1, Multiple Sector	
a₀ = Data Address Mark (Bit 0)	
a ₀ = 0, Write Normal Data Mark	
a ₀ = 1, Write Deleted Data Mark	
E = 30ms Settling Delay (Bit 2)	
E = 0, No Delay	
E = 1, Add 30ms Delay	
P = Write Precompensation (Bit 1)	

TYPE IV COMMANDS	
l₃-l₀ Interrupt Condition (Bits 3-0)	
l ₀ = 1, Don't Care	
l ₁ = 1, Don't Care	
l ₂ = 1, Interrupt on Index Pulse	
l ₃ = 1, Immediate Interrupt	
l ₃ -l ₀ = 0, Terminate without Interrupt	

Step Commands are type I. Lets see if we can provide a step pulse train for the Drive Unit. We will again not connect the unit.

• Step

Upon receipt of this command the WD1770 issues one stepping pulse to step the recording head in the same direction as the previous step. A verification takes place after a delay determined by rate field, if the V flag is ON. If the U flag is ON the Track register is updated. The h bit allows the motor ON option (Disabled on VISA) at the start of the command. An interrupt is generated at the completion of the command.

• Step-In

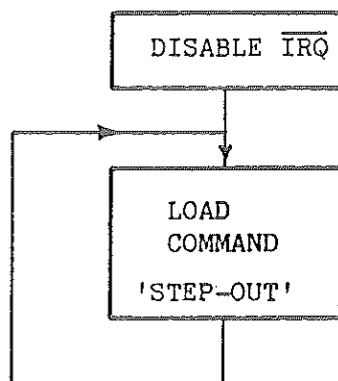
Issues a step pulse towards inside of Disc (Track 76). Otherwise similar to step-in.

• Step-Out

Similar to Step-In but in opposite direction.

Now lets write a program to exercise the STEP PULSE and DIRECTION outputs.

The flow chart below should suffice:



NOTES:

- COMMAND Register Addr - 8C00

- STEP-OUT COMMAND Formatt:

B ₇	B ₆	B ₅	B ₄	B ₃	B ₂	B ₁	B ₀
0	1	1	U	h	V	r ₁	r ₀

U = Update	- NO	Bit = 0
h = Motor enable	- NO	Bit = 1
V = Verify	- NO	Bit = 0
r ₁ r ₂ = Step Rate	- Fastest	Bits = 00

The STEP-OUT COMMAND required is thus 0 1 1 0 1 0 0 0 (68 HEX).

Our program thus becomes:

```

0200    78      SEI
          A9,68    LDA# 68
          8D,00,8C STA 8C00
          4C,01,02 JMP 0201

```

Exercise 8.9

- Ensure Disc Drive is not connected.

- Load the program above to provide a continuous pulse train in the outwards direction.

- Run the program and monitor the following:

- STEP-PULSE (PIN 16 - WD1770)
- DIRECTION (PIN 17 - WD1770)

- Determine the pulse repetition rate and pulse width.

- Explain why we set the 6502 Interrupt Flag.

● Exercise 8.11

- Rewrite program to continuously step inwards.
 - Prove.
-

● Restore

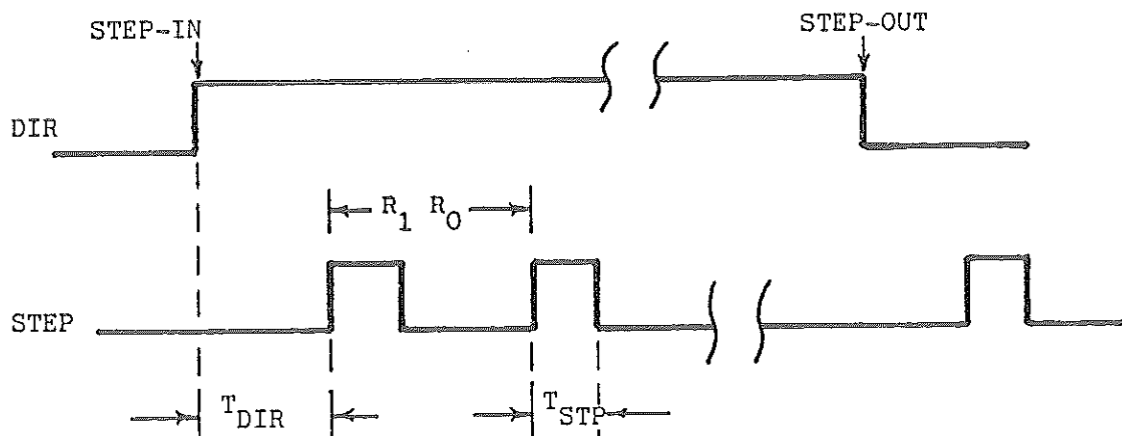
The RESTORE (Seek Track 00) command can be used to generate a continuous pulse train. Upon receipt of this COMMAND the track 00 ($\overline{TR00}$) input is sampled. If low it indicates that the Recording Head is positioned over Track 00. The Track Register will contain zeroes and an interrupt generated. If $\overline{TR00}$ is high, then stepping pulses will continue until Track Zero is found. If Track 00 is not found after 255 pulses the operation is terminated.

The Track 00 input is pulled-high on the VISA board. With no Drive Unit connected it cannot go low.

● Exercise 8.11

- Write a program to cause the controller to RESTORE. For the purposes of monitoring, your program needs to repetitively RESTORE.
 - Observe using an oscilloscope.
-

The timing waveforms for the STEP PULSE and DIRECTION are:



- TDIR = $24\mu\text{S}$. An additional $30\mu\text{S}$ takes place to ensure the Head is settled if the V or E flags are set.
- TSTP = $4\mu\text{S}$ (MFM) or $8\mu\text{S}$ (FM)
- $R_1-R_0 = 6\text{mS} - 30\text{ms}$ as set.

- **Seek**

The SEEK command seeks for a given Track. The command assumes that:

- Track register contains track number of current position of Read/Write Head.
- Data register contains desired track number.

The controller will automatically update the Track register and issue step pulses in the appropriate direction until the contents of the Track register equal the contents of the Data register.

A verification takes place if the V flag is set.

An interrupt is generated at the completion of the command.

- **Exercise 8.12 - Drive Disconnected**

- Write a program to load the TRACK REG with 00, the DATA REG with 02 and then use the SEEK COMMAND.
- At the completion of SEEK an interrupt will be caused. Write an interrupt service routine to handle this, insert a time delay and then return to the start of the main program.

POST TEST

Chapter 8

- 8.1 Briefly explain, use a sketch if necessary, how the VISA Keyboard is arranged and scanned as an item of hardware.
- 8.2 In using a keyboard, what two operational possibilities must be taken into account if successful keyboard scanning is to take place.
- 8.3 Briefly explain the purpose of a Floppy Disc Controller.
- 8.4 Explain the terms, Frequency Modulation and Modified Frequency Modulation as applied to Floppy Disc Recording.
- 8.5 Briefly explain the purpose of the Commands; Restore, Seek and Step.

CHAPTER 9

Objectives of this Chapter

Having studied this chapter you should:

- Understand how a typical Video Interface operates.
- Be familiar with a Cathode Ray Tube Controller operates.
- Understand Memory Mapped Displays.
- Understand the signal requirements of a Visual display unit.
- Be able to write programs to exercise a CRT Controller for test purposes.

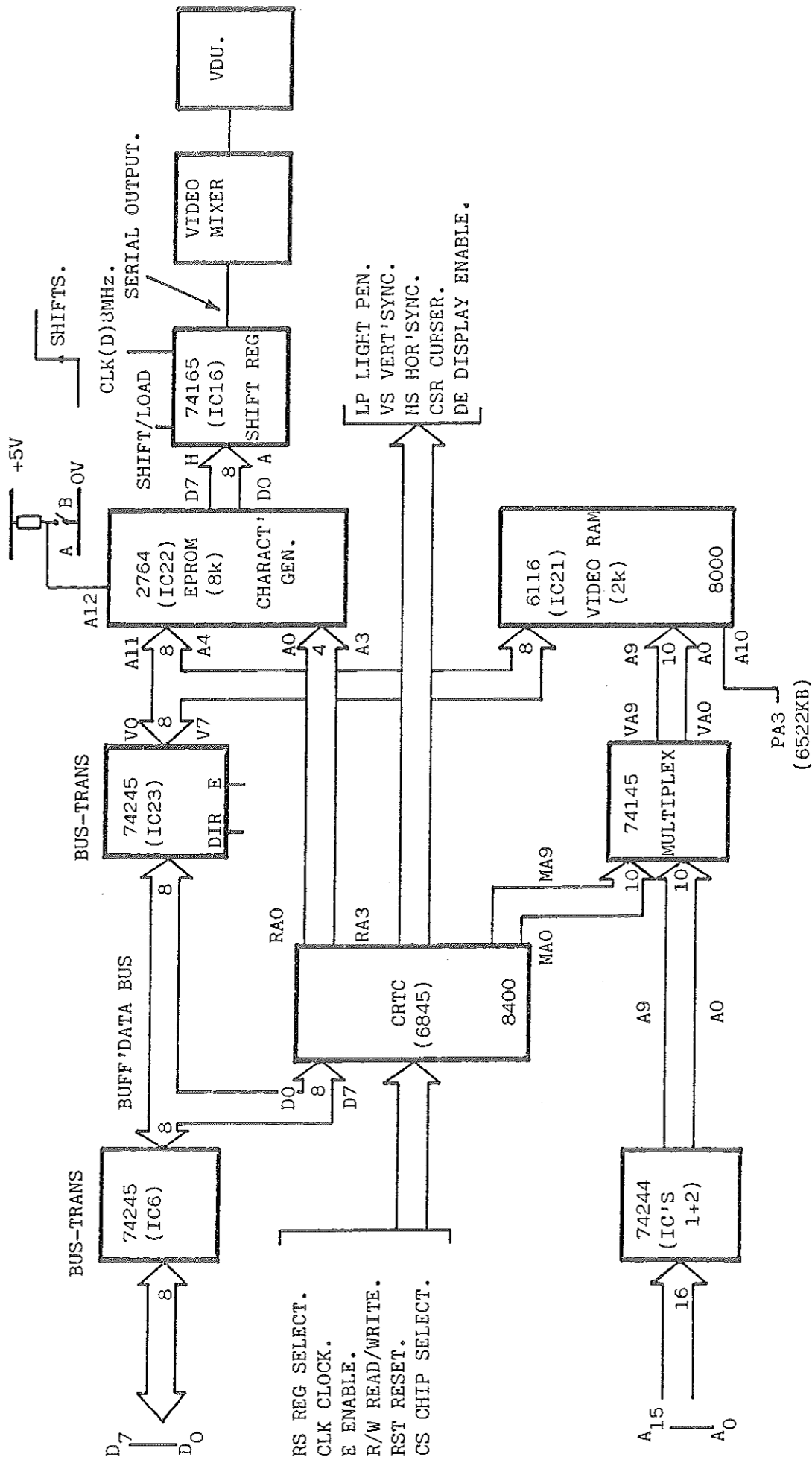
The Video Interface

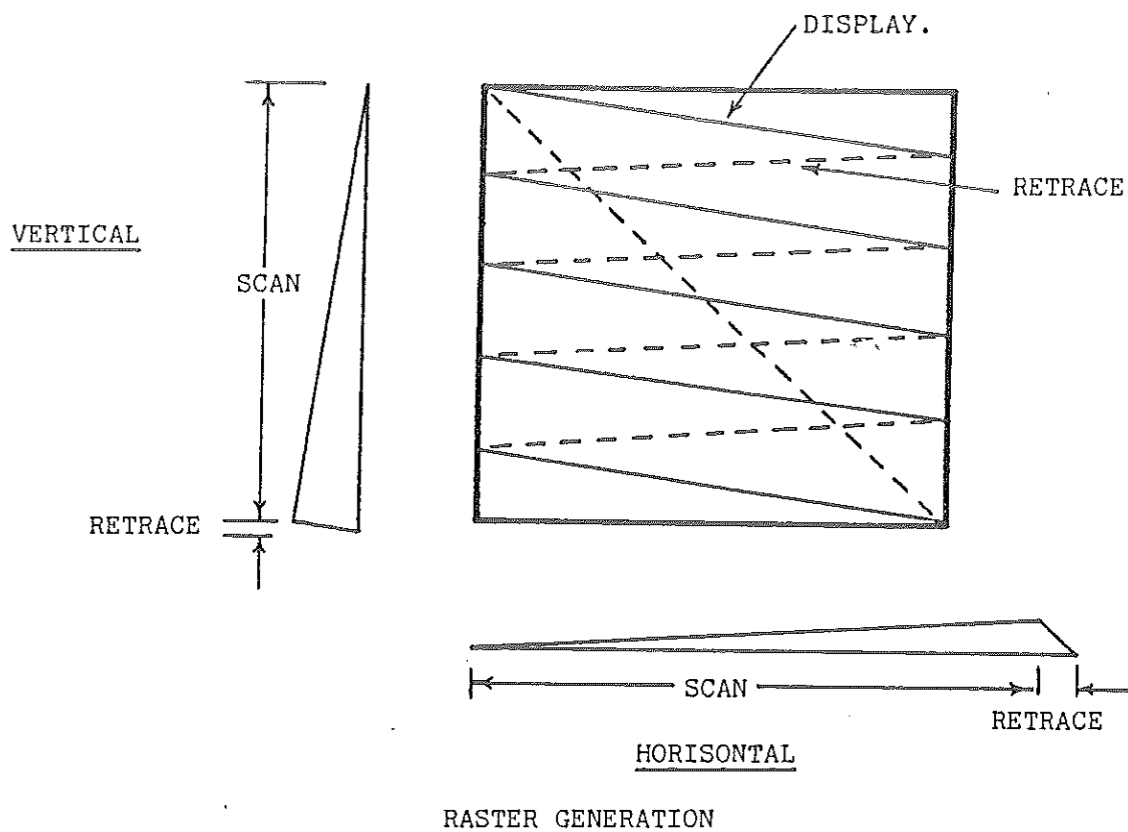
The Video Interface enables the System User to communicate visually with the system. It is essentially hardware based but is supported by software.

The block diagram on sheet 2 provides an outline of the interface. At the heart of the Interface is a Cathode Ray Tube Controller (CRTC) which performs the interface between the microprocessor and a raster-scan Visual Display Unit (VDU).

● The Cathode Ray Tube Controller (CRTC)

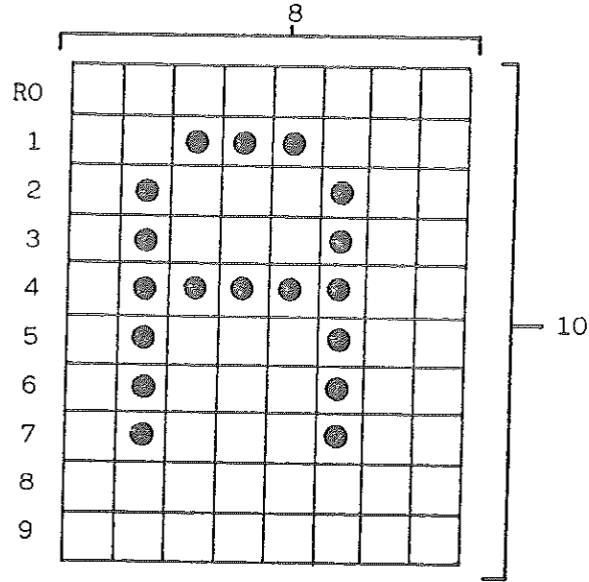
The CRTC generates the signals necessary to interface a digital system to a raster scan CRT display or VDU. The principle is that an electron beam starts in the upper left-hand corner of the display, moves quickly across the screen and returns. This is termed a horizontal scan. After each horizontal scan the beam is incrementally moved down in the vertical direction until it has reached the bottom. At this stage one 'frame' is said to have been displayed. The diagram overleaf indicates:





The number of lines shown are illustrative purposes only

In order to display a character(s) on the VDU frames must be continually repeated. The data to be displayed is stored in a Refresh Memory (6116 RAM) by the microprocessor (EMMA - 6502). Since the data is written in ASCII code it cannot directly be displayed as characters hence a Character Generator (2764-EEPROM) is used to convert the ASCII codes into the 'dot' pattern required to make-up each displayed character. Each character is a matrix of dots (8 x 10) in the VISA System. A typical display is as shown below together with the output for the first two horizontal scan lines.



1st SCAN 

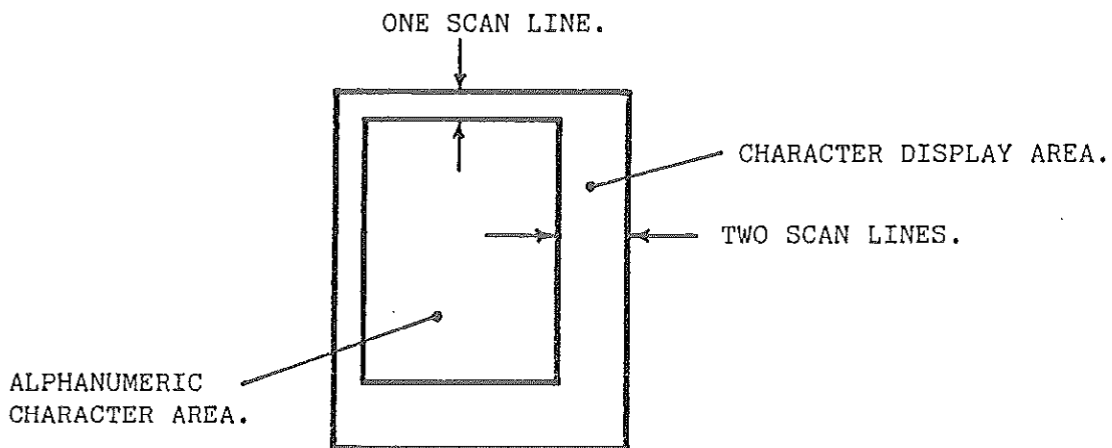
2nd SCAN 

3rd SCAN 

Exercise 9.1

Sketch the horizontal scan lines for the rest of the character block above indicating the scan line number.

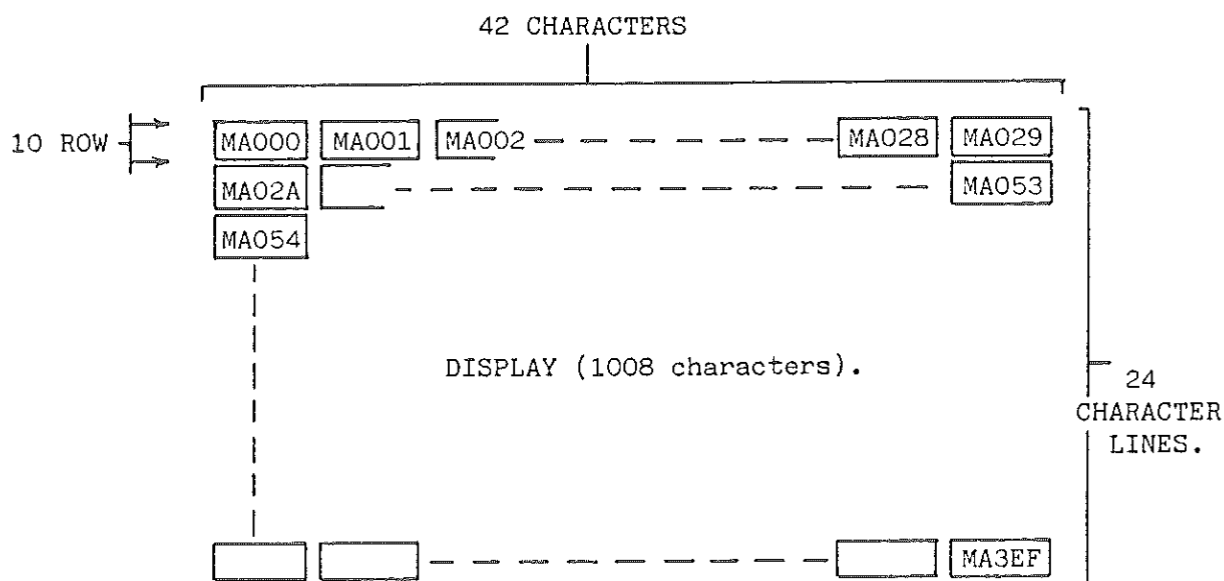
The actual position of an Alphanumeric Character within the Character Display Area is 5 x 7 and biased towards the top left-hand corner:



The depth of Character Display Area and the actual character size are a function of the CRT Controller initialisation and the predetermined Data entered into the Character Generator EPROM. The width of the character is determined by the Character Clock Rate (CCLK - 1MHz) and the number of dots/character by the Dot Clock Rate (DCLK - 8MHz). Both these 'clocks' are generated by the 'Video Clock Generator'. The CRTC is therefore driven by the CCLK and the Parallel - Serial Shift register which generates the serial output for each horizontal scan line by the DCLK.

Now, referring again to the block diagram it should be seen that the CRT Controller (6845) generates the Video Refresh Addresses (MA0-MA9) and the character Row Addresses (RA0-RA4). The data output to these by the CRTC together with the Horizontal and Vertical Syncs determine the exact position of a character row on the Display and the actual data displayed.

Diagrammatically:

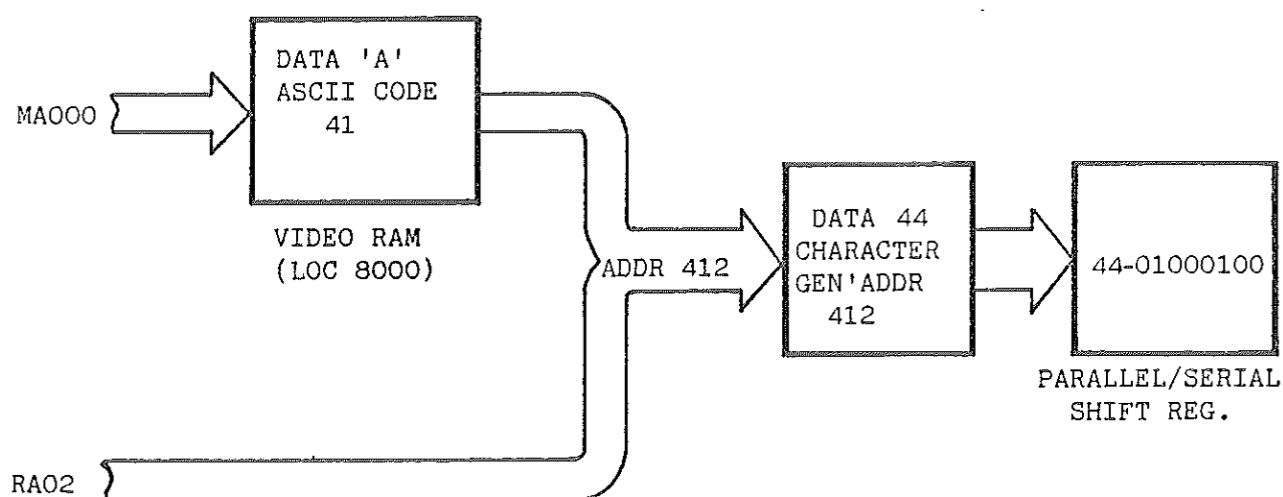


We say that the Display is Memory Mapped and as far as the microprocessor is concerned MA000 is seen as memory location 8000.

The actual display area and the start address of data within the Video Refresh RAM is set-up by CRTC internal registers - Start Address Register (R12-R13), Horizontal Displayed Register (R1) and Vertical Displayed Register (R6). These registers are all initialised by the EMMA/VISA Monitor.

We have already mentioned that the Video RAM contains data in ASCII code which is converted by the Character Generator into a Character Dot Code. Lets assume that the Video RAM location 8000 contains the alphabetic character 'A' (ASCII Code 41).

If the CRTC outputs MA00 and RA02 then diagramtically:



Exercise 9.2

Consult the character 'A' given earlier in text and determine the data required in the Character Generator and the Serial data output when the Row Address becomes RA04.

Exercise 9.3

The letter H is encoded by the ASCII Code 48. If this letter is to be displayed at the beginning of the second character line determine:

- The Memory Address (MAXXX) to be output by the CRTC when this character is being 'refreshed'.

The address in the Character Generator Eprom when the CRTC outputs the Row Address RA04.

- The Parallel/Serial data input to and output from the Video Shift Register.

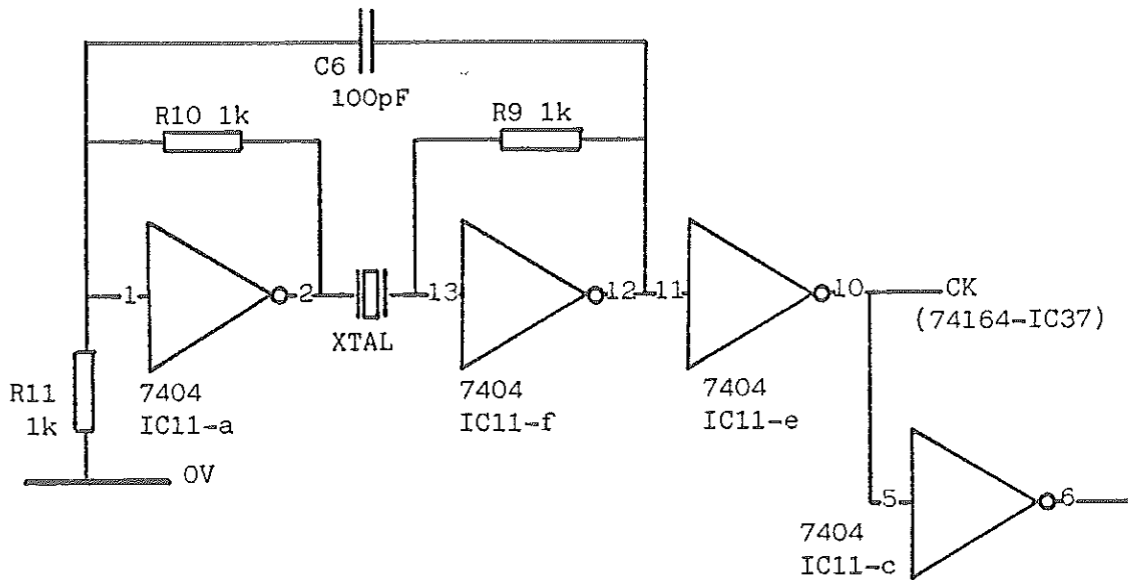
So far we have considered data being output from the Video (Refresh) RAM. We have intimated that the RAM must be addressable from the microprocessor since it forms part of the memory map for the system. Indeed the Refresh Address provided by the CRTC and the Video RAM Address Set-up by the microprocessor are multiplexed to provide the Video Address Bus (VA0-VA9). Writing can thus be achieved to the video RAM (Refresh RAM) by the microprocessor for Display up-date. Display up-date must be made during periods when the Display is not being refreshed by the CRTC. The Display Enable (DE) output from the CRTC permits this. The Monitor Program causes the DE output to Set-up a flag such that writing to Video RAM can be suppressed until periods of horizontal or vertical fly-back occur.

Finally, the Bus-Transceiver (74245 - IC23) isolates the Video Data Bus (V0-V7) from the Buffered Microprocessor Data Bus D0-D7. This is required since the CRTC registers must be capable of being initialised by the microprocessor. The CRTC is addressed via its Chip Select ($\overline{CS}$) line and individual registers by the internal Address Register via the Data Bus.

Now lets investigate practically some of the signals associated with the interface as a whole.

- Video Clock Generator

The basic clock circuitry is as shown below:

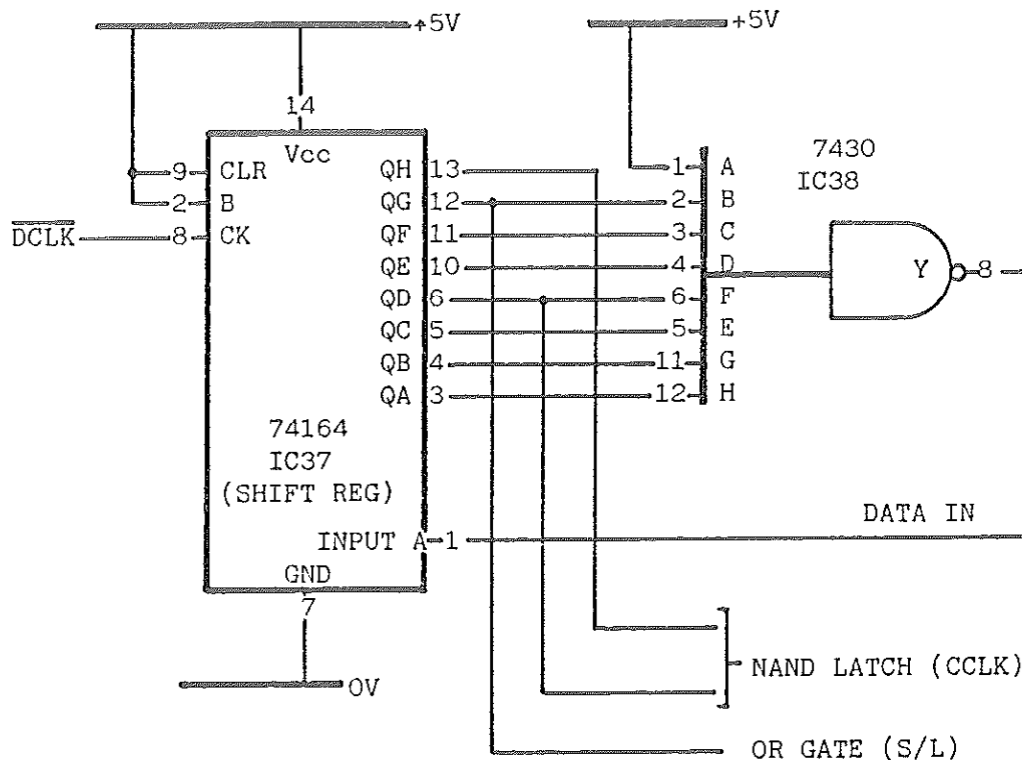


The crystal (XTAL) is operating at 8MHz.

● Exercise 9.4

● Use the oscilloscope to monitor the waveform at Pin 6 of the 7404 Inverter (IC11). This is the Dot Rate Clock (DCLK) and is used to clock the Parallel/Serial Shift Register which serialises the Video Data. It is also used by the Floppy Disc Controller.

In addition to the Basic clock a clock is required by the CRTC to mark each character. Since a character is effectively 8 dots wide and the Dot Rate is 8MHz, the character clock (CCLK) runs at 1MHz. It is derived from the DCLK by the circuitry overleaf:



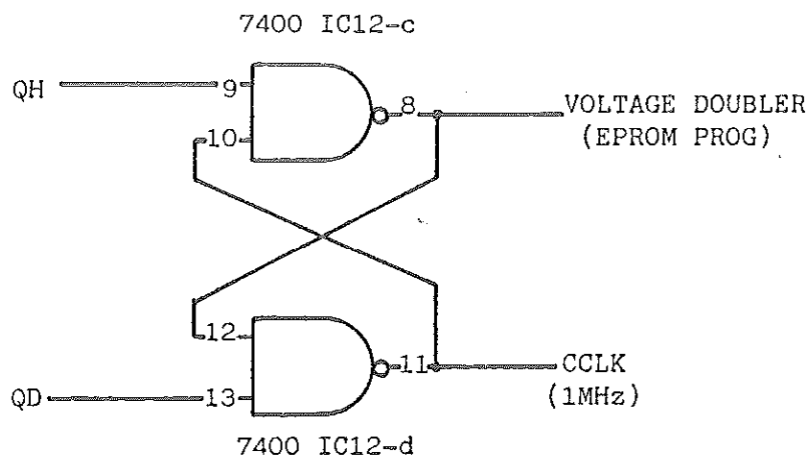
The 74164 (IC37) is used as a Serial/Parallel Shift Register providing eight output bits. Data can only be shifted to the right (QA QB QC QH)

In operation, the DCLK is input at Pin 8. Upon a negative to positive (positive edge) of the DCLK the output data is shifted by one to the right. Assuming an initial condition of QA QH = HIGH then the 7430 NAND (IC38) will be LOW thus providing a logic zero at the Shift Register input. Upon receipt of the first rising DCLK edge QA goes to logic 0. The zero progresses through the device at DCLK rate to be lost after QH, whereupon the next zero awaits clocking to QA.

Exercise 9.5

- Use the oscilloscope to produce a Timing Chart for outputs QD, QG and QH and relate these to the CK (Pin 8) input.
- Notice the mark - space ratios of the Q outputs.

Now consider the circuit below:



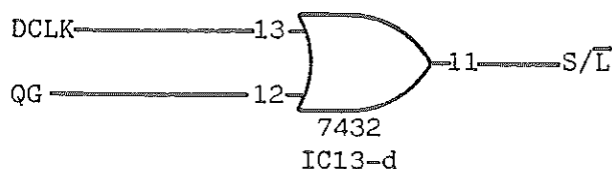
This is a simple NAND latch, which effectively conditions the QD and QH outputs to provide an equal mark - space ratio CCLK related to the DCLK by a one-to-eight rate ratio.

● Exercise 9.6

- Use the oscilloscope to produce a Timing Chart for the QD and QH outputs related to the CCLK.
- Add to your diagram the DCLK so that DCLK and CCLK can be compared.
- Monitor also the output (Pin 8 - IC12).

Notice that this is in anti-phase to the CCLK. It is used by the Voltage Doubler circuit which we will consider later for programming EPROMS.

The Shift/Load input to the Video Data Shift Register is derived as shown below:



● Exercise 9.7

- Use the oscilloscope to produce a Timing Chart for the CCLK, DCLK and S/L.
- The S/L output is also used to clock the Display Enable and Cursor outputs from the CRTC for eventual Video Mixing with the Horizontal and Vertical Sync and Video Data outputs for presentation to the Video Display Unit.

We will now concentrate upon the Video Mixing Circuitry. The diagram for which is on page 12.

The function of the circuitry is to mix the various signals (to produce a COMPOSITE signal) required by the Video Display Unit and to condition them to the required input levels.

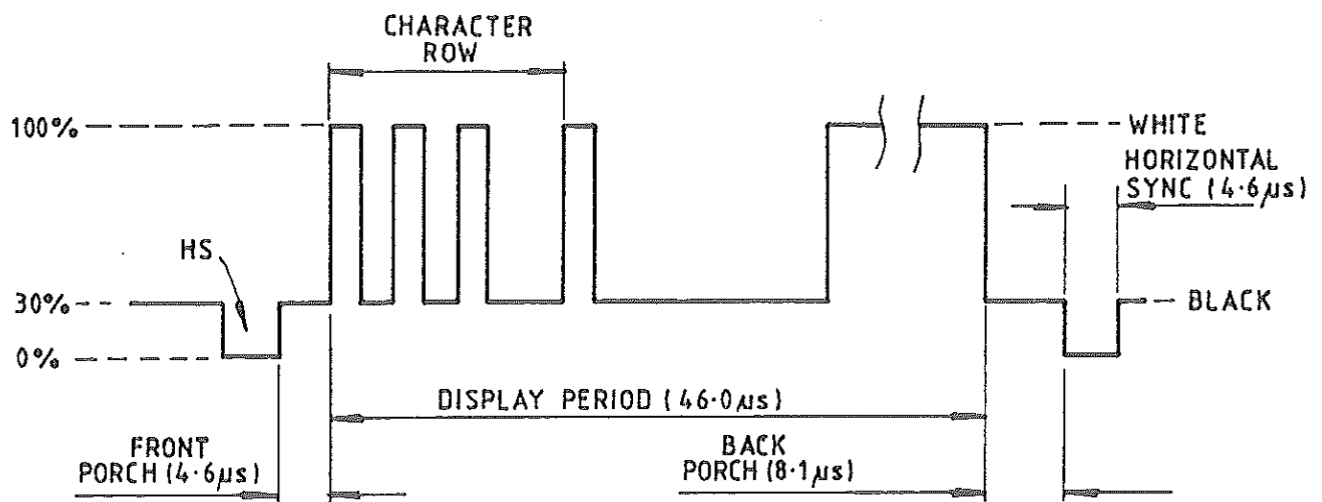
The signals normally required are:

- Vertical Sync
- Horizontal Sync
- Cursor
- Data (Dots)

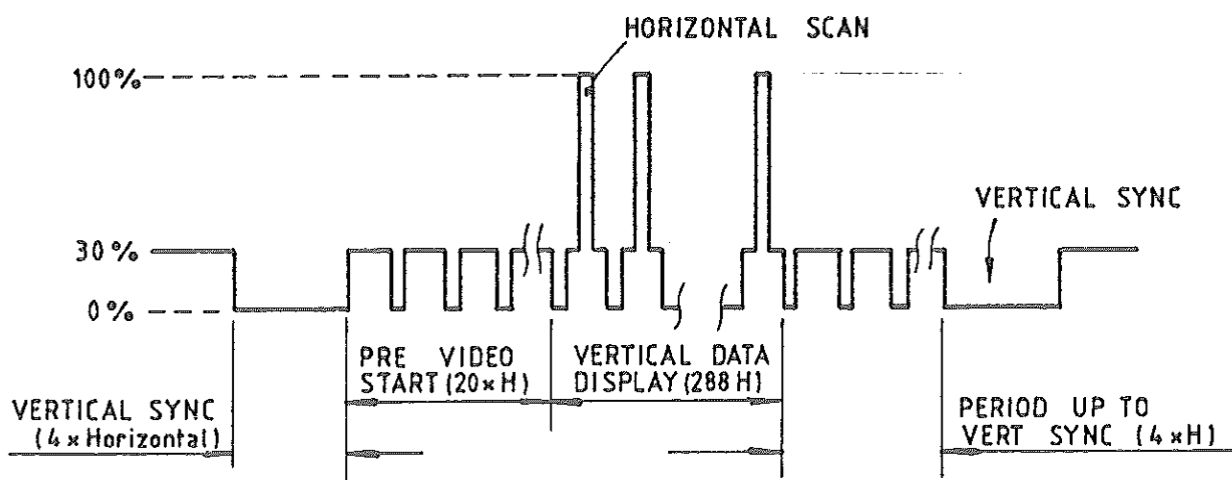
The first three are produced directly by the CRTC and the fourth, indirectly by the CRTC, from the Character Generator and associated Serial Shift Register.

Lets start from the Composite Video Output and work backwards. The line is driven by the BC237 transistor stage which is normally biased ON by virtue of R19. The Composite Video Signal would then be normally HIGH. This would be pulled LOW by either the vertical or Horizontal Sync Signals by virtue of the inverter 7406 (IC7). This meets VDU conditions for negative Sync Signals.

These two signals with respect to the Data can be shown as below:



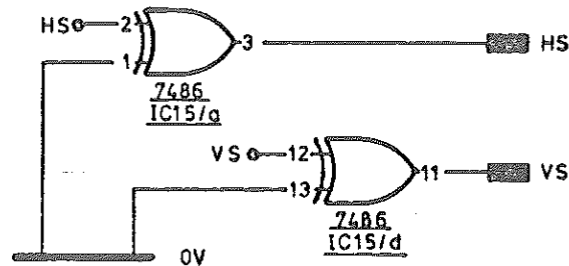
HORIZONTAL SYNC + DATA



VERTICAL SYNC PERIOD

Exercise 9.8

- Trigger the oscilloscope externally from the vertical Sync.
- Monitor the Vertical Sync Signal from the Pin provided. Notice that this signal goes high by virtue of the exclusive OR gates feeding these monitoring pins.



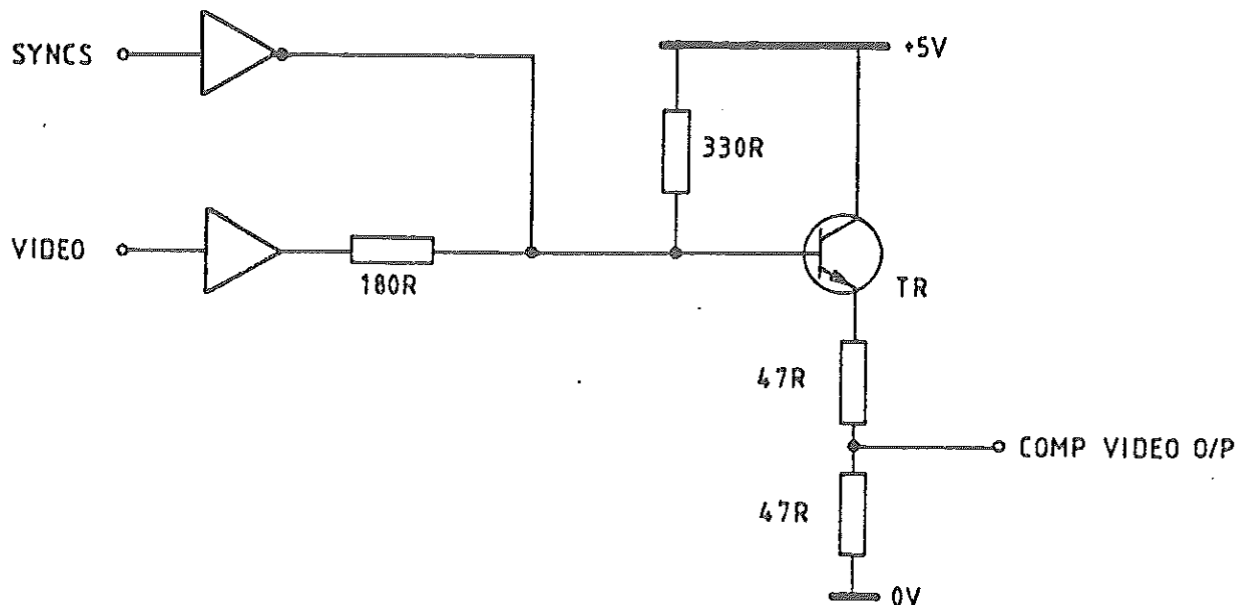
- Measure the period between Vertical Sync Pulses.

Exercise 9.9

- Trigger the oscilloscope externally from the Horizontal Sync.
 - Monitor the Horizontal Sync Signal and measure the Pulse period.
 - Now blank the VDU and HOME the Cursor.
 - Obtain a second trace but of the VIDEO Signal.
 - With the brightness (of the oscilloscope) suitably adjusted you should be able to see the flashing cursor as data.
 - Now observe the VIDEO Signal and while doing so fill the whole screen with the graphic symbol  (under A key)*
- *You may like to write a short routine to do this for you. The ASCII code for the Graphic Symbol is 8A.
- Notice the Data field will progressively become brighter with a mark - space ratio of 1 to 7 (one 'White' and seven 'Black').

- Using the first trace (Horizontal Sync) as a marker, measure the Front and Back Porch periods and the Display Period. Also measure the period between horizontal Sync pulses.

Now consider the mixing of the VIDEO and SYNC Signals.



A positive SYNC from the CRTC will pull the base of TR down to 0V so turning TR OFF and passively pulling the COMP.VIDEO input to VDU, low (Negative Sync). The video input will however be changing between 'black' - low and 'white' - high as appropriate to the data being transmitted. A 'low' will not take the base of TR to 0 volts due to the 330 Ω and 180 Ω potentiometer network. The actual values will cause TR to hold COMP.VIDEO at approximately the required 30% of maximum 100% being 'white'.

Moving back still further notice that the VIDEO Signal is 'mixed' with the CURSOR output from the CRTC as clocked by the Shift/load signal. This prevents Video Data AND Cursor appearing on the VIDEO line together. Second, the cursor will be clocked by a transition of $S/\overline{L}$ to low.

● Exercise 9.10

- Make a Timing Chart with the aid of the oscilloscope of the relationship between: $S/\overline{L}$; cursor (output from CRTC - pin 12 on IC14 or C monitoring pin); QH (fill display with Data) and the output of the Exclusive OR gate.

The Display Enable output from the CRTC is also gated by the S/L output before 'mixing' with the now 'combined' Video Data/Cursor signals. The NAND gate ensures that the VIDEO line remains low until the Display Enable line signals that data can be output to the Display.

● Exercise 9.11

- Make a Timing Chart of the output of the Exclusive OR, the Q output (DE) of the 7474 and the VIDEO line.

Now lets use a Logic Analyser to look at the Video Buses.

For this we will assume that we have no VISA Monitor to handle the Display. Our only means of communicating with the VISA will be via the EMMA keyboard and Seven Segment Display as enabled by the EMMA Monitor.

● Exercise 9.12

- Connect the EMMA/VISA system and obtain the EMMA Monitor.
- Notice that only a blank flashing cursor remains.

We can remove the cursor also by writing to the CRTC Register Address Register (8400) with the address of the cursor start register (0A) then loading the cursor start register with 20 (non-display) by addressing the CRTC Register Data Register (8401).

If we put this on the end of our previous program then:

(020A		* JMP *)	Remove
020A	A9,0A	LDA# 0A	
C	8D,00,84	STA 8400	
F	A9,20	LDA# 20	
11	8D,01,84	STA 8401	
4	4C,04,02	* JMP *	

- Now we will extend the program to load the ASCII codes from 00 to FF into the Video RAM starting on line Seven.

The seventh line starts at memory location $8000 + (42 \times 6) + 1 = 80FD$ (HEX) therefore:

0214	8A	TXA	
5	9D,FD,8D	STA	80FD,X
8	E8	INX	
9	D0,F9	BNE	F9
1B	4C,1B,02	JMP	*

Notice 256 characters are displayed. The first 20 (HEX) ASCII codes are in fact control codes and are output by the Character Generator as all zero's.

- Now change the JUMP TO address from 0218 to 0214. This will have the effect of continuously writing to the display at the same time as the CRTC is trying to refresh the Display. The effect is to produce 'snow'.

Now lets try using the logic analyser on some of the Video Buses.

● Exercise 9.13

- Change the program above to load the alphanumeric character A, B and C into display locations 8000-8002 and D into 8041.
- Run the program to obtain AB and C at first three locations and display and D at last location on the first line.
- Connect the logic analyser to the Video Address lines and Row Address lines.

Analyser MSB	→	V0 - V7
Analyser LSB	→	RA0 - RA3

- Clock the Analyser from the rising edge of the Character Clock (CCLK). Connect the Clock Qualifier to DE set to 1.

- Set the Analyser Trigger to Memory Address 41 x 3. Disable the upper nibble of L.S. Byte. Connect Trigger Qualifier to Reset line and set to 1.
- Hold Reset low and ARM Analyser.
- Release Reset.
- Observe that the Analyser Display indicates the ROW number and the data output in each character position.
- Predict the effect of setting the Trigger word to 41 x 9. Capture data to prove!

POST TEST

Chapter 9

- 9.1 Explain the purpose of a Cathode Ray Tube Controller.
- 9.2 Explain, using sketches if necessary, how a character displayed on a VDU is actually formed.
- 9.3 Explain why a Character Generator is a necessary component of a Video Interface.
- 9.4 Explain the purpose of the Video Refresh RAM.
- 9.5 Explain the difference between the Dot Rate Clock and the Character Clock.
- 9.6 Explain the function of the Video Mixing Circuitry.
- 9.7 State the levels (as a percentage) of the Sync and Data Signals required by a typical VDU.

CHAPTER 10

Objectives of this chapter

Having studied this chapter you should:

- Understand a typical RAM Back-Up System.
- Understand a Software Controlled EPROM Programmer.

MISCELLANEOUS CIRCUITS

● BATTERY BACK-UP

The Battery Back-up circuitry is designed to trickle charge a NICAD battery and automatically connect it to the RAM power supply lines in the event of loss of power. It also holds the RAM Chip Enable ($\overline{CE1}$) line high so preventing access to the RAM and causing substantial current drain.

With normal supply connected TR4 will remain in a conducting state so holding low the RAM $\overline{CE1}$ line. In the event of the Power Supply voltage dropping below 3.6V the zenner (D4) will cease to conduct, TR4 will turn OFF and $\overline{CE1}$ will go high as determined by the battery voltage. Vbb will also draw current from the battery via the diode D3.

Again with normal supply connected TR2 and TR3 will both conduct. TR3 will thus trickle charge the battery via the 1k limiting resistor R26.

The capacitor (C8) acts as a reservoir and takes care of a current surge normally drawn by the 6254 RAMs upon switch-off. C7 is a high frequency decouple.

● Exercise 10.1

- Monitor the RAM $\overline{CE1}$ pin with normal power supply on.
- Remove the power supply and notice that $\overline{CE1}$ goes high.
- Monitor the V.bb pin on slow time base and notice the transient change for switching both ON and OFF.

● EPROM Programmer

The EPROM programmer requires a supply of +24volts. However, some EPROM's require less and some more than this for successful programming.

2716 (2k Bytes) - VPP = 26.5V

2764 (8k Bytes) - VPP = 22.0V

Since the Programmer is capable of programming both 2k and 8k EPROMS a voltage doubler together with associated regulator and EPROM size selection circuitry is provided.

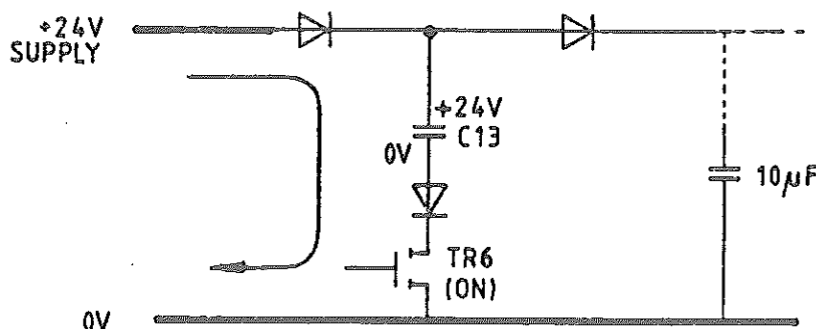
Programming of EPROMS is in fact effected through Software, via an 8255 Programmable Interface, within the VISA Monitor. Also two programming speeds may be selected having entered the EPROM Program Routine.

● Voltage Doubler

Refer to schematic in Appendix.

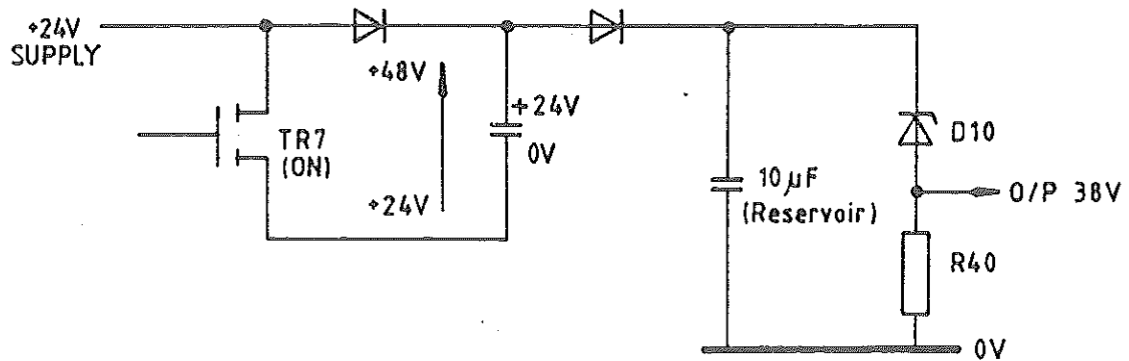
TR6 is turned ON and OFF by a 1MHz signal derived from CCLK.

With TR6 ON, capacitor CB is charged to 24 volts with no current drain to EPROM.



TR6 then switches OFF and TR7 ON. This puts the previously 0V connection of C13 to supply +VE (24 volts).

When the capacitor is charged the circuit output voltage will be pushed to 48 volts (24V supply + 24V capacitor).



The output circuitry of D10 (zenner) and R40 provide approximately 38V for the EPROM Programmer Circuitry and is input via a voltage regulator.

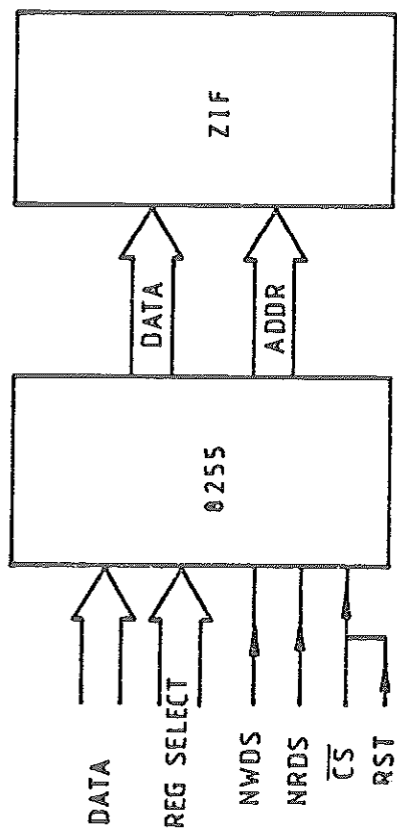
● EPROM Programmer - Program Voltage Source

Refer to Schematic page 4.

The 38V provided by the Voltage Doubler circuitry sources an adjustable voltage regulator (REG1). The output of the regulator provides two voltage levels (25V for 2716 and 21V for 2764 EPROMS).

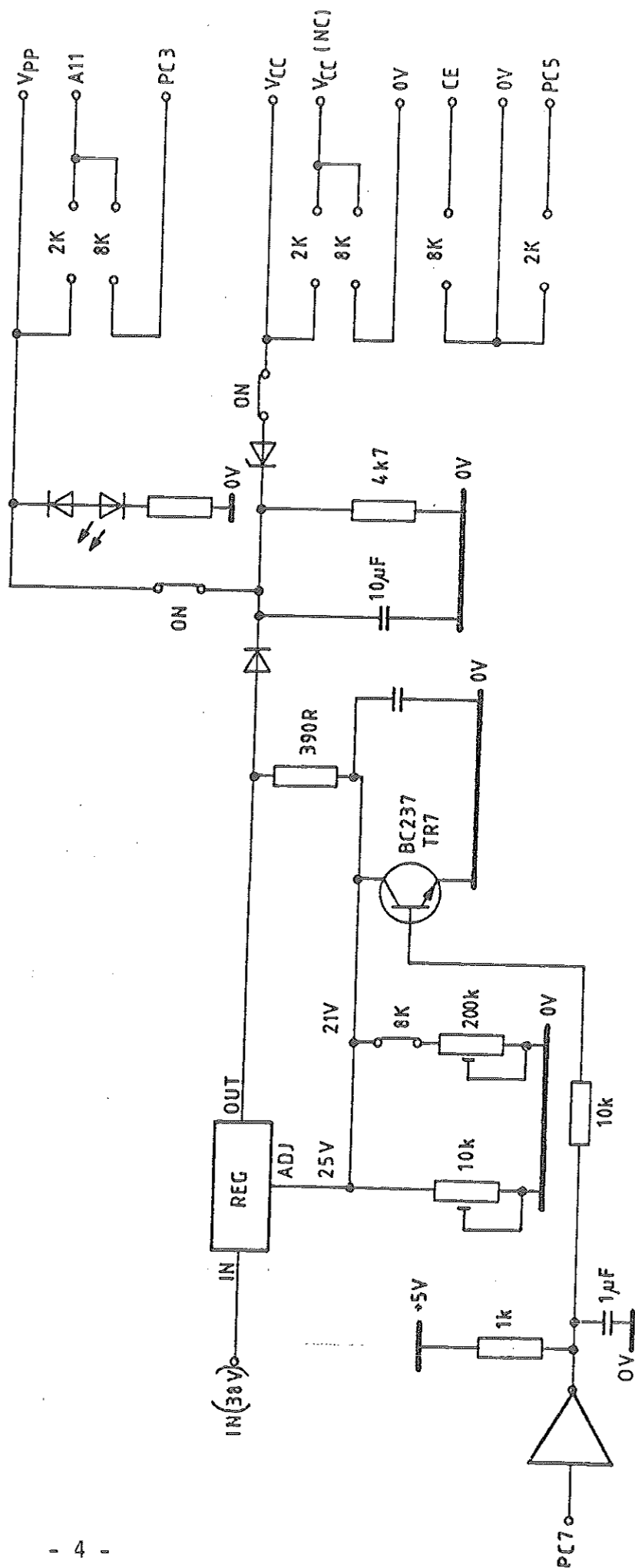
Under program control PC7 (8255 PPI) goes high to give programming voltage. 2k/8k selected switch determines actual output voltage shunting 200k Trim resistor across 10k Trim resistor to give 21 volts. With PC7 low TR7 conducts to take the Regulator 'adjust' to zero volts thus effectively turning off the program supply.

The output (PC3) from the interface adapter provides the necessary address change for 8k EPROMS. Outputs PC5 and PC6 provide the output enable ($\overline{OE}$) for purposes of verification and the program enable ($\overline{PGM}$) for programming.



PC7 ————— PROGRAMMING VOLTAGE

PC3 ————— 8K (ADDRESS CHANGE)



Objectives of this Chapter

Having studied this chapter you should:

- Be able to logically split-up a microprocessor based system for fault finding purposes.
- Have gained experience fault finding on a typical VDU/Qwerty Keyboard microprocessor based system.

Fault and Diagnostic Testing (VISA)

● **Introduction**

From a fault and diagnostic point of view this chapter follows on from Chapter 5 where we considered the EMMA system.

Lets have a quick appraisal of the components of our now expanded microcomputer. The block diagram on page 2 presents this diagrammatically for us.

● **System Software**

The System Software directly affects the operation of the following Hardware devices.

- Cassette Interface
- EMMA Keyboard/Display
- Visa Keyboard/Floppy Disk
- EPROM Programmer.

SYSTEM SOFTWARE

System (EMMA)
MONITOR

System (VISA)
EMMA/VISA MONITOR

BASIC
INTERPRETOR

ASSEMBLER /
DISASSEMBLER

TEXT EDITOR

SYSTEM HARDWARE

MEMORY
DECODE

SYSTEM
RAM

SYSTEM (EMMA)
MONITOR EPROM

MEMORY
DECODE

SYSTEM
RAM

SYSTEM MONITOR
EMMA/VISA

I/O PORT
(USER)

USER
EPROM

KEYBOARD
INTERFACE

VDU
INTERFACE

KEYBOARD
INTERFACE

FLOPPY DISC
INTERFACE

SYSTEM PERIPHERALS

CASSETTE
INTERFACE

KEYBOARD
(EMMA)

SEVEN
SEGMENT
DISPLAY

EPROM PROGRAMMER

EPROM
PROG

V.D.U.

KEYBOARD
(VISA)

FLOPPY
DISC

From the list practically every item of hardware that interfaces with the user can be seen to be affected by the system software. We could therefore be excused if the first thing we suspected when fault conditions exist is the System Software. However, this is not usually the case. Suspect User Developed Software but not System Software! This leaves us with Hardware.

● **System Hardware**

Obviously some items of Hardware affect the whole system. For example the microprocessor itself. In our particular system we can split off the VISA Expansion Module from the EMMA. This we should always do in the event of a fault. If the EMMA Board is operating satisfactorily then we can reconnect the VISA with the knowledge that at least 30-40% of the total system is working.

We can also disconnect the Floppy Disk but as a device it does not connect directly to the System address, data or control buses.

Now, in the following, we make no attempt to discuss the numerous possible faults, you must rely upon your knowledge of the system which by now should be fairly familiar to you. What we would reiterate is the logical order in which to approach a fault condition.

- Monitor System Control Lines (Oscilloscope)
- Monitor System Address/Data Line (Logic Analyser)
- Monitor Secondary System Address/Data Lines e.g. Video Address/Data Lines (Logic Analyser)
- Monitor Chip Enable Lines (Oscilloscope)
- Monitor Serial Signal Lines.
- Monitor Interface Chip Output Lines e.g. CRTC Rows, Keyboard (Qwerty) Row and Column lines, Floppy Disk Controller etc.

In all these tests, since we are starting - off with a working EMMA System we could devise User Test Routines to exercise identifiable Sub-Systems. This you have already done for some areas. You are encouraged to do this for yourself.

● **VISA Switched Faults**

The fault number represents the order of difficulty of the fault. You are encouraged to approach the faults in a logical manner rather than rely upon inspired (or otherwise) guess work.

● **Exercise 11.1**

Diagnose the fault, and if possible locate the point of application, for each of the eight switched faults.

APPENDIX 1

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/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 T0 and the fetch of the OP CODE and continue through the required number of cycles until the next T0 and the fetch of the next OP CODE.

While the basic terminology used in this appendix is discussed in the main Manual, it has been defined below for ease of reference while studying Single Cycle Execution.

- | | |
|-------------------|--|
| OP CODE | The data on which the operation specified in the OP CODE is performed. |
| 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 to AB07) is ADL. |

INDIRECT ADDRESS The address found in the operand of instructions utilizing (Indirect), Y which contains the low order byte of the 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	DEX	NOP	TAX	TYA
CLC	DEY	ROL	TAY	
CLD	INX	SEC	TSX	
CLI	INY	SED	TXA	
CLV	LSR	SEI	TXS	

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.

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	OP CODE (Discarded)	1	
T0	PC + 1	OP CODE	1	Next Instruction

A. 2. INTERNAL EXECUTION ON MEMORY DATA

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

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	Data	1	Fetch Data
T0	PC + 2	OP CODE	1	Next Instruction

A. 2.2. Zero Page Addressing (3 cycles)

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch Effective Address
T2	00, ADL	Data	1	Fetch Data
T0	PC + 2	OP CODE	1	Next Instruction

A. 2.3. Absolute Addressing (4 cycles)

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch low order Effective Address byte
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)

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
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	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
T0	PC + 2	OP CODE	1	Next Instruction

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

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	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 0 or 1 as required from previous add operation
T4*	ADL: BAL + index register ADH: BAH + 1	Data	1	Fetch data from next page
T0	PC + 3	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)

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
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	00, BAL + index register	Data	1	Fetch Data (no page crossing)
T0	PC + 2	OP CODE	1	Next Instruction

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

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
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 + C	Data*	1	Fetch Data from same page 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
T0	PC + 2	OP CODE	1	Next Instruction

*If page boundary is crossed in indexing operation, the data fetch in T4 is ignored. If page boundary is not crossed, the T5 cycle is by-passed.

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/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)

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	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
T0	PC + 2	OP CODE	1	Next Instruction

A. 3.2. Absolute Addressing (4 cycles)

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	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
T0	PC + 3	OP CODE	1	Next Instruction

A. 3.3. Indirect, X Addressing (6 cycles)

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
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	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
T0	PC + 2	OP CODE	1	Next Instruction

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

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	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
T0	PC + 3	OP CODE	1	Next Instruction

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

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch Page Zero Base Address
T2	OO, 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)

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	IAL	1	Fetch Page Zero Indirect Address
T2	OO, IAL	BAL	1	Fetch low order byte of Base Address
T3	OO, 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--Modify--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)

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	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
T0	PC + 2	OP CODE	1	Next Instruction

A. 4.2. Absolute Addressing (6 cycles)

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	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
T0	PC + 3	OP CODE	1	Next Instruction

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

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
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 + 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
T0	PC + 2	OP CODE	1	Next Instruction

A. 4.4. Absolute, X Addressing (7 cycles)

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	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	Modified Data	0	Write modified Data back into memory
T0	PC + 3	OP CODE	1	New Instruction

A. 5. MISCELLANEOUS OPERATIONS

BCC	BRK	PHP
BCS	BVC	PLA
BEQ	BVS	PLP
BMI	JMP	RTI
BNE	JSR	RTS
BPL	PHA	

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

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	OP CODE (Discarded)	1	
T2	Stack Pointer*	Data	0	Write Internal Register into Stack
T0	PC + 1	OP CODE	1	Next Instruction

*Subsequently referred to as "Stack Ptr."

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

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	OP CODE (Discarded)	1	
T2	Stack Ptr.	Data (Discarded)	1	
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)

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	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
T0	Subroutine Address (ADH, ADL)	OP CODE	1	Next Instruction

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

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
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 (NMI-FFFFB) (RES-FFFFD)	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)

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
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)

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
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)

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	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
T0	ADH, ADL	OP CODE	1	Next Instruction

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

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	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	
T0	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)

<u>Tn</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>R/W</u>	<u>Comments</u>
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	Offset	1	Fetch Branch Offset
T2*	PC + 2 + offset (w/o carry)	OP CODE	1	Offset Added to Program Counter
T3**	PC + 2 + offset (with carry)	OP CODE	1	Carry Added

*Skip if branch not taken

**Skip if branch not taken; skip if branch operation doesn't cross page boundary.

APPENDIX 2

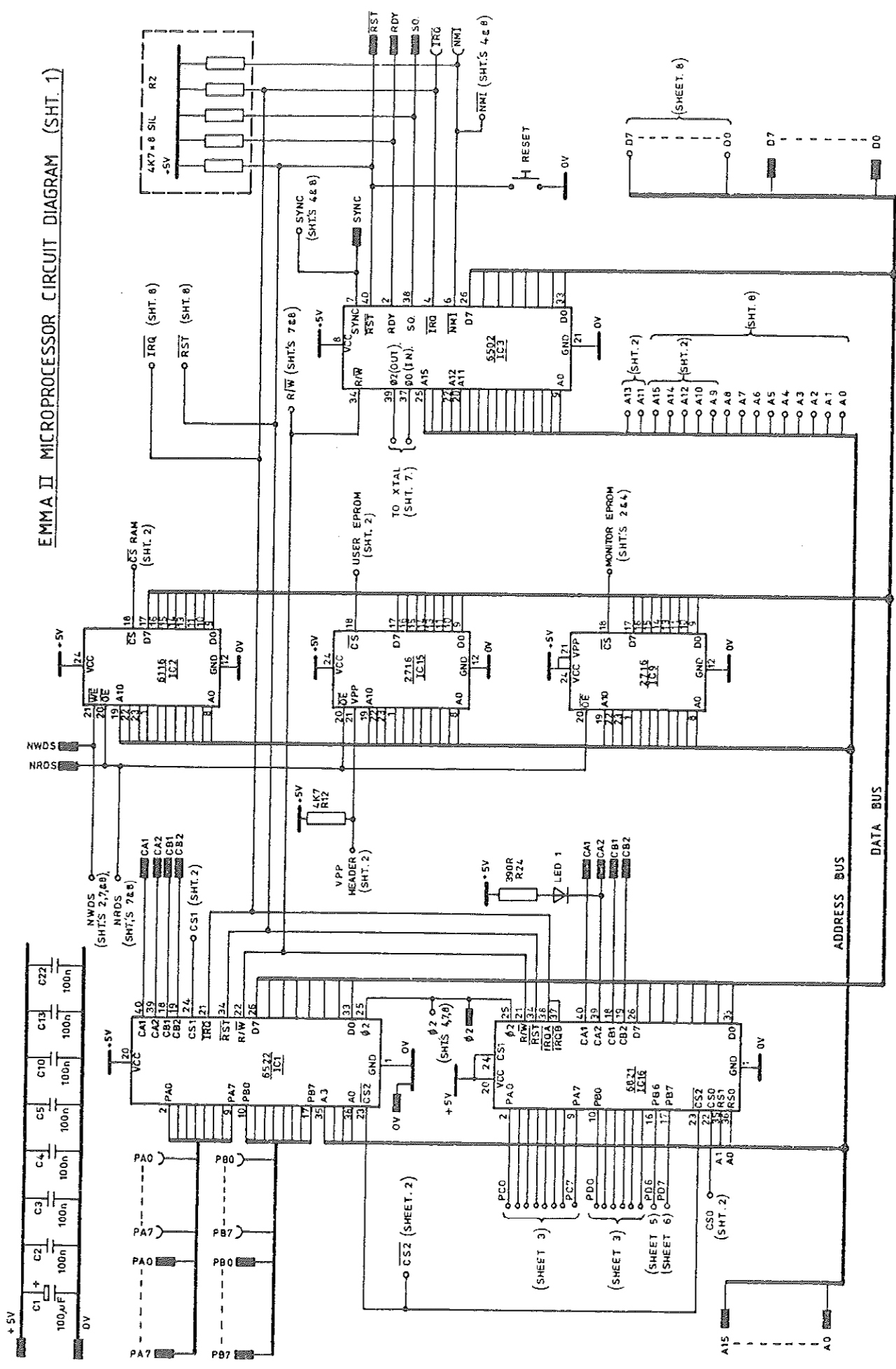
PROM 'B' CODING

ADDRESS BUS					DATA BUS							
D1	A13	A12	A11	A10	D8	D7	D6	D5	D4	D3	D2	D1
0	0	0	0	0	1	0	1	1	1	0	1	1
0	0	0	0	1	1	0	1	0	0	1	1	1
0	0	0	1	0	1	0	0	1	0	1	1	1
0	0	0	1	1	1	1	1	1	0	1	1	1
0	0	1	0	0	0	0	1	1	0	1	1	1
0	0	1	0	1	0	0	1	1	0	1	1	1
0	0	1	1	0	1	0	1	1	0	1	0	1
0	0	1	1	1	1	0	1	1	0	1	1	0
0	1	0	0	0	1	0	1	1	0	1	1	1
0	1	0	0	1	1	0	1	1	0	1	1	1
0	1	0	1	0	1	0	1	1	0	1	1	1
0	1	0	1	1	1	0	1	1	0	1	1	1
0	1	1	0	0	1	0	1	1	0	1	1	1
0	1	1	0	1	1	0	1	1	0	1	1	1
0	1	1	1	0	1	0	1	1	0	1	1	1
0	1	1	1	1	1	0	1	1	0	1	1	1
1	0	0	0	0	1	0	1	1	0	1	1	1
1	0	0	0	1	1	0	1	1	0	1	1	1
1	0	0	1	0	1	0	1	1	0	1	1	1
1	0	0	1	1	1	0	1	1	0	1	1	1
1	0	1	0	0	1	0	1	1	0	1	1	1
1	0	1	0	1	1	0	1	1	0	1	1	1
1	0	1	1	0	1	0	1	1	0	1	1	1
1	0	1	1	1	1	0	1	1	0	1	1	1
1	1	0	0	0	1	0	1	1	0	1	1	1
1	1	0	0	1	1	0	1	1	0	1	1	1
1	1	0	1	0	1	0	1	1	0	1	1	1
1	1	0	1	1	1	0	1	1	0	1	1	1
1	1	1	0	0	1	0	1	1	0	1	1	1
1	1	1	0	1	1	0	1	1	0	1	1	1
1	1	1	1	0	1	0	1	1	0	1	1	1
1	1	1	1	1	1	0	1	1	0	1	1	1
1	1	1	1	1	1	0	1	1	0	1	1	1

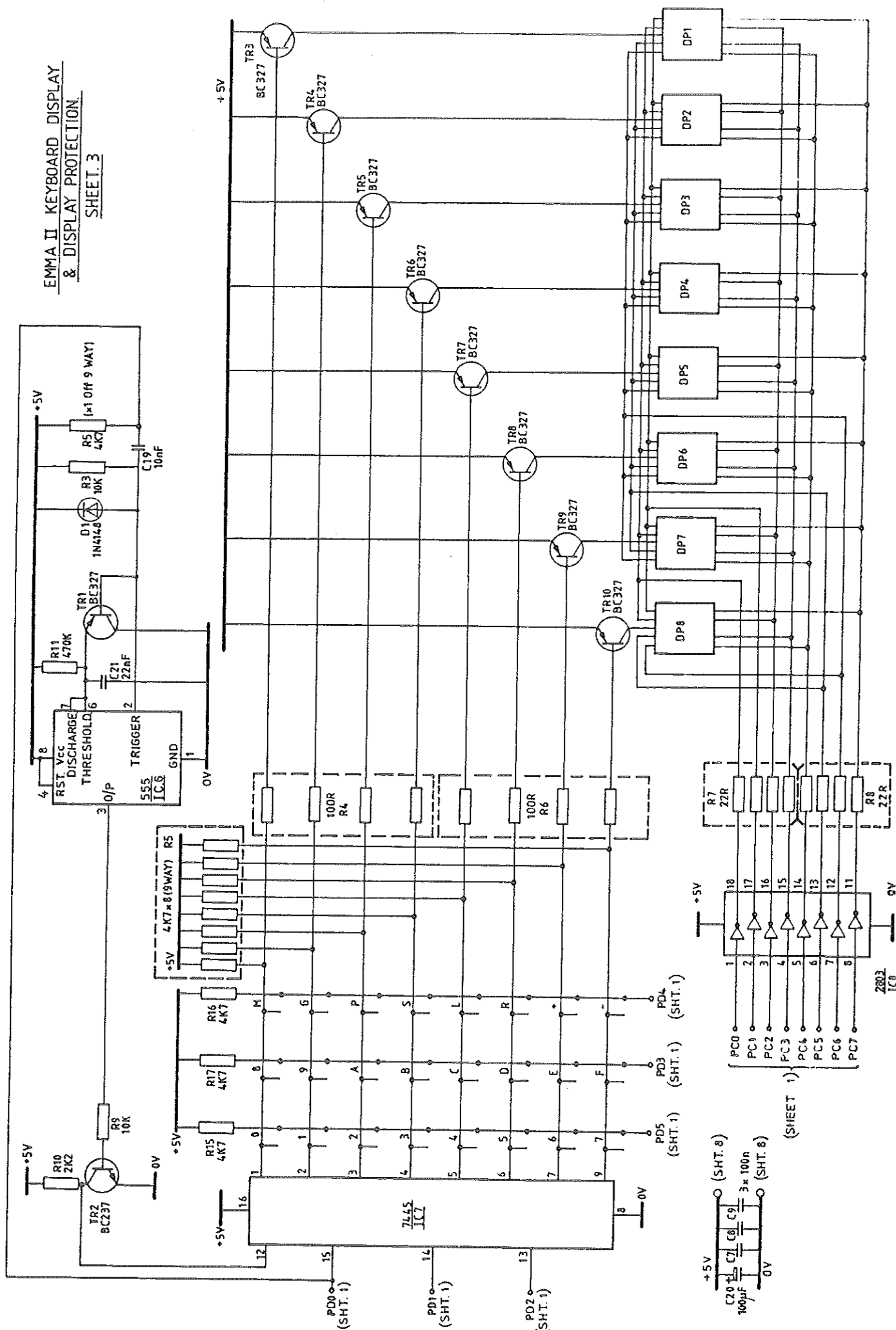
APPENDIX 3

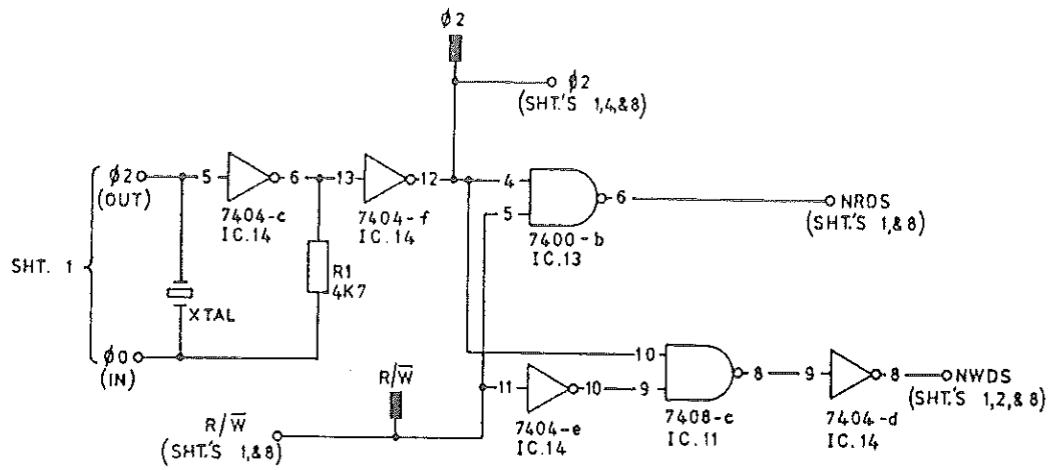
EMMA II SCHEMATICS

EMMA II MICROPROCESSOR CIRCUIT DIAGRAM (SHT. 1)

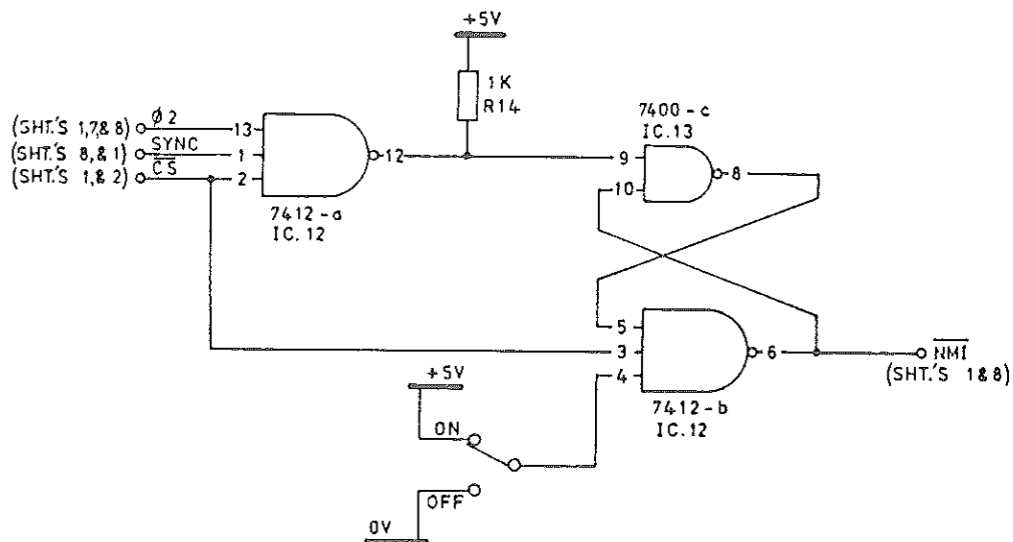


EMMA II KEYBOARD DISPLAY & DISPLAY PROTECTION SHEET. 3



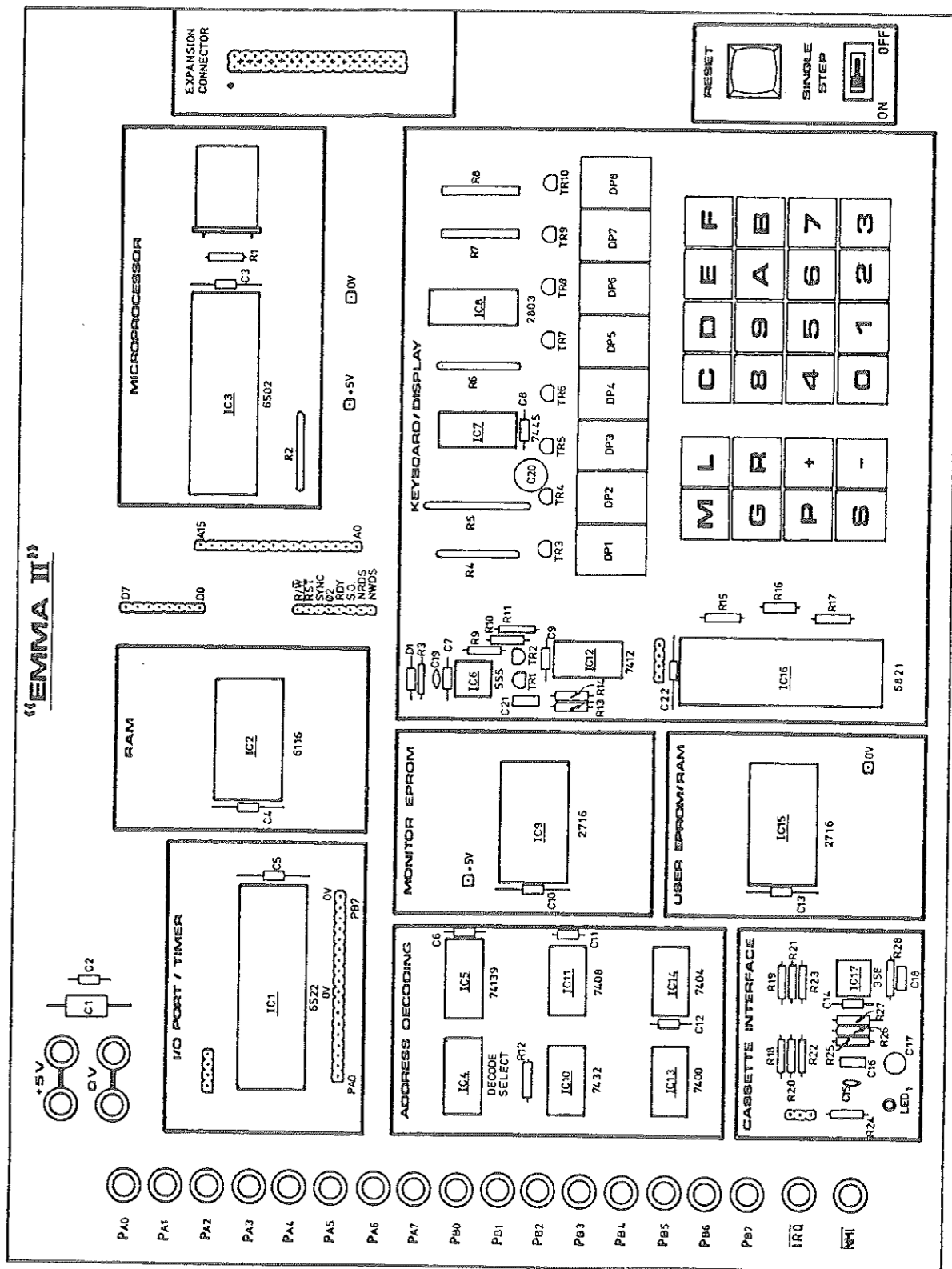


SHEET. 7

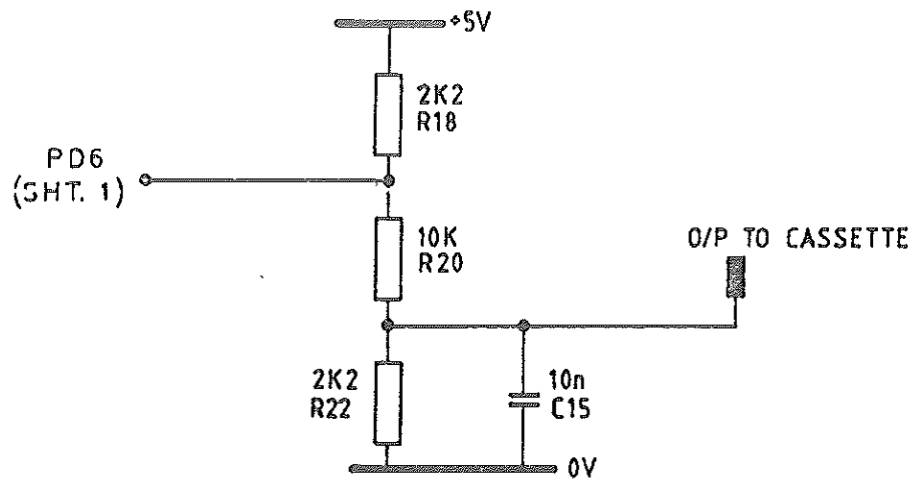


SHEET. 4

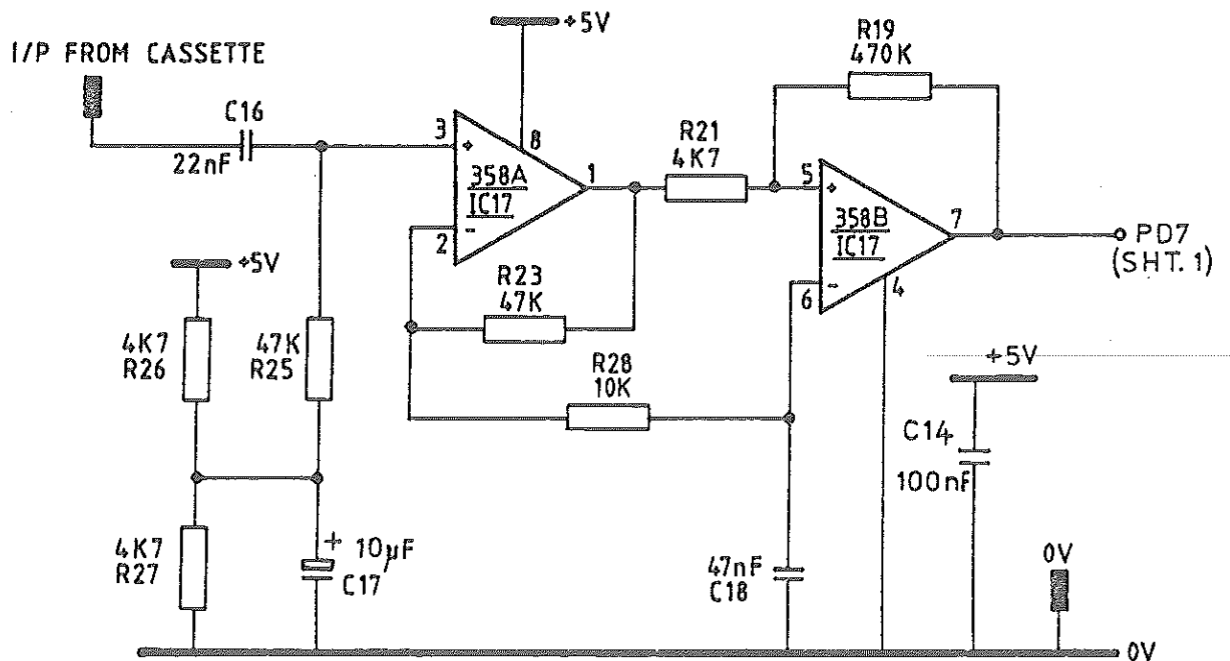
"EMMA II"



EMMA II CASSETTE INTERFACE



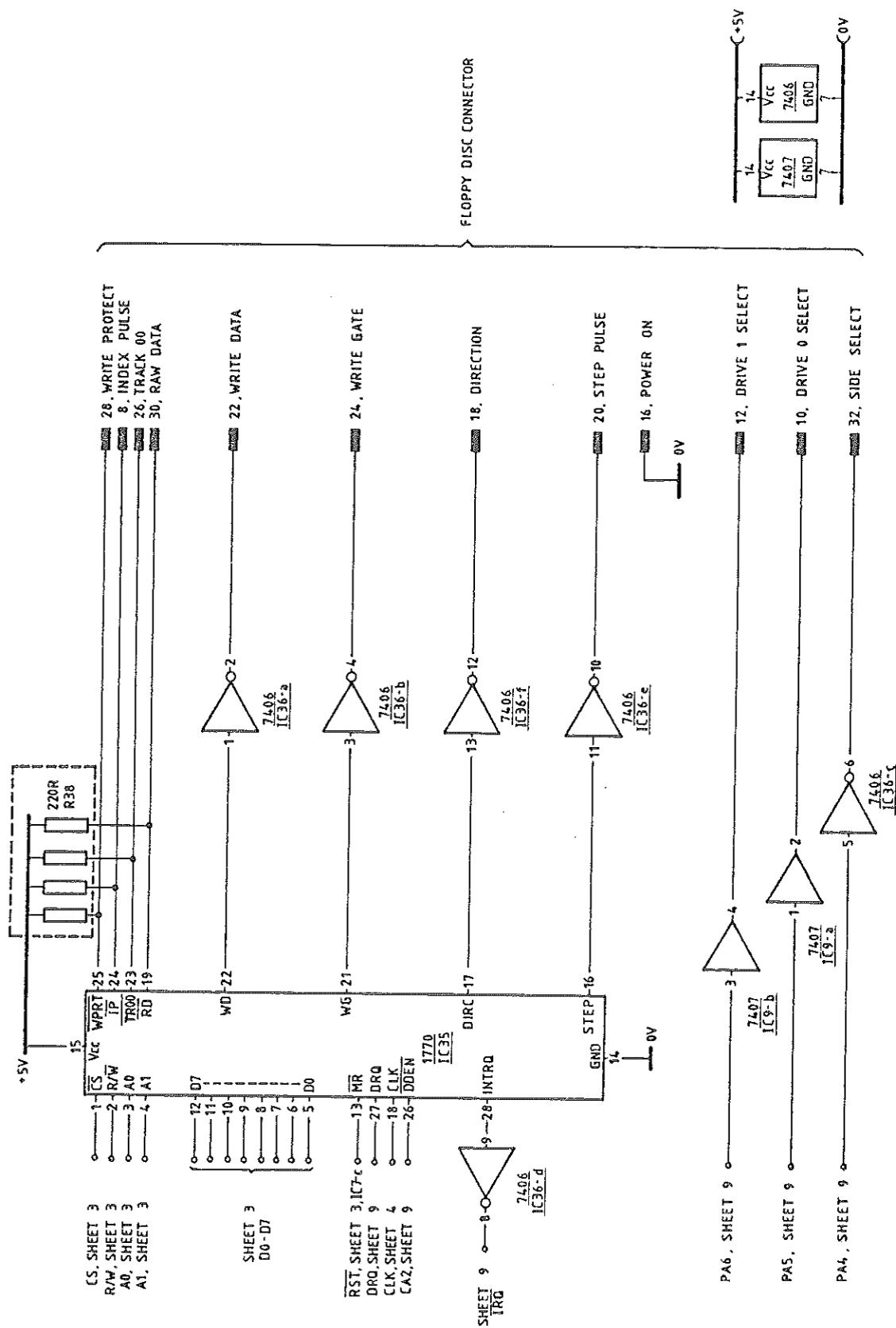
Output To Cassette
SHEET. 5



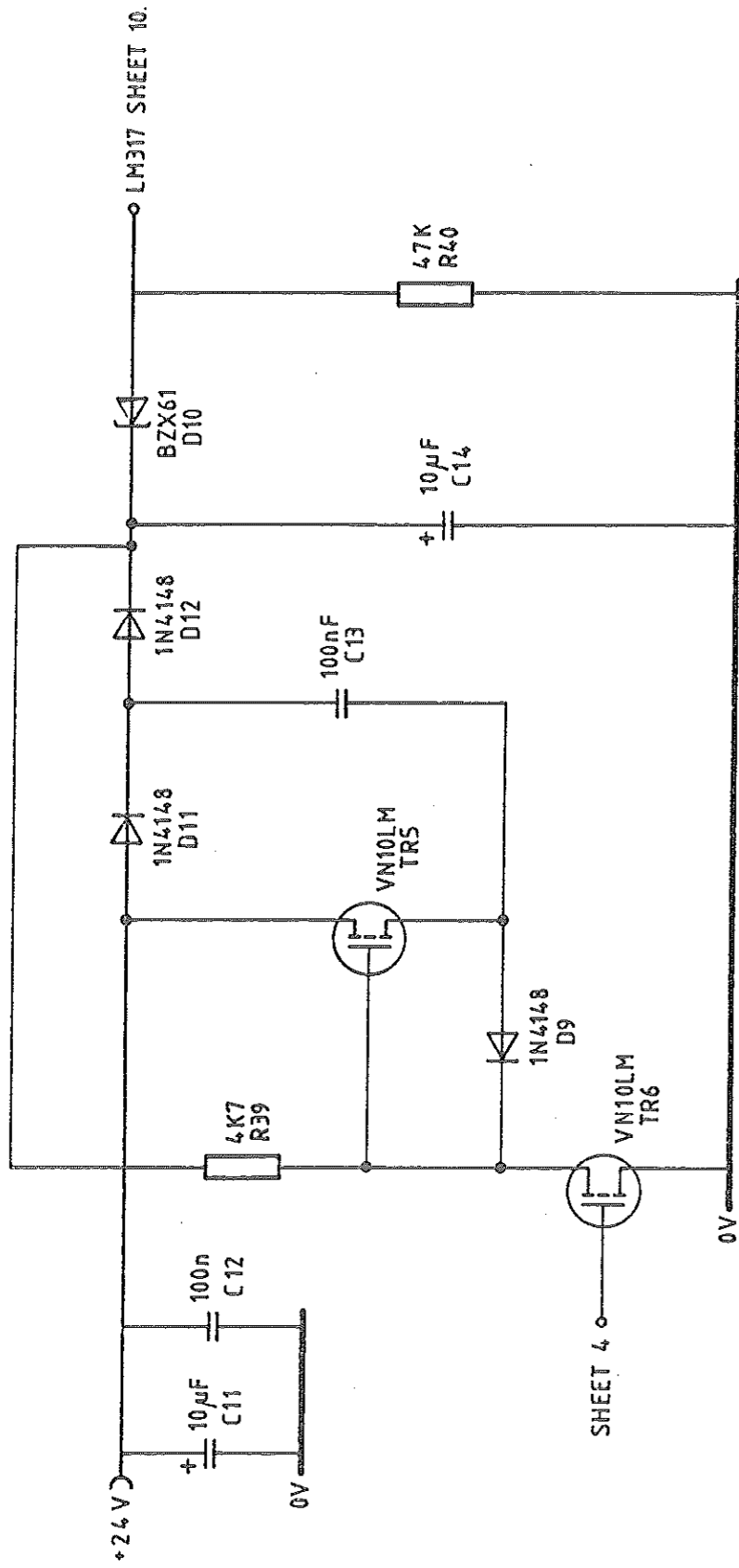
Input From Cassette
SHEET. 6

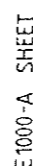
VISA II SCHEMATICS

VISA II FLOPPY DISC INTERFACE:



VISA II VOLTAGE DOUBLER

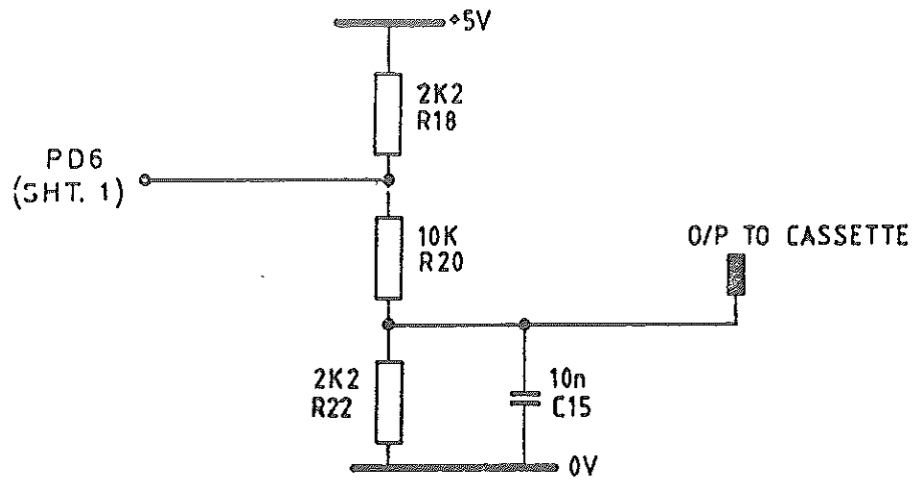




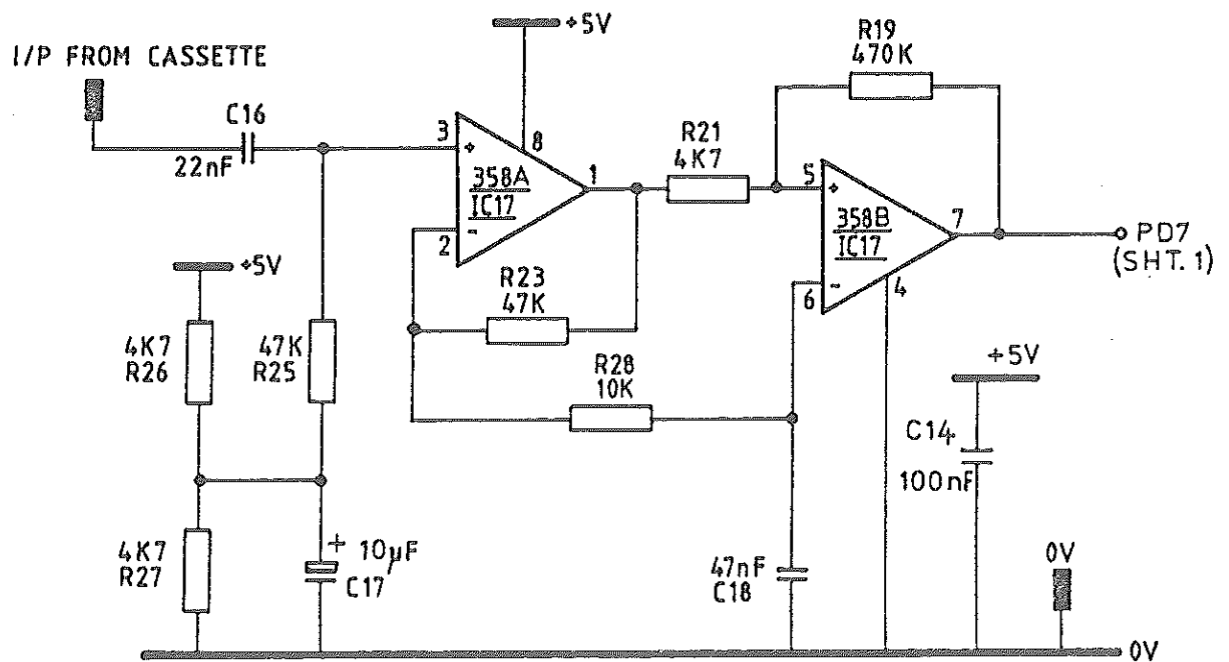


VISA II VIDEO BUSES

EMMA II CASSETTE INTERFACE

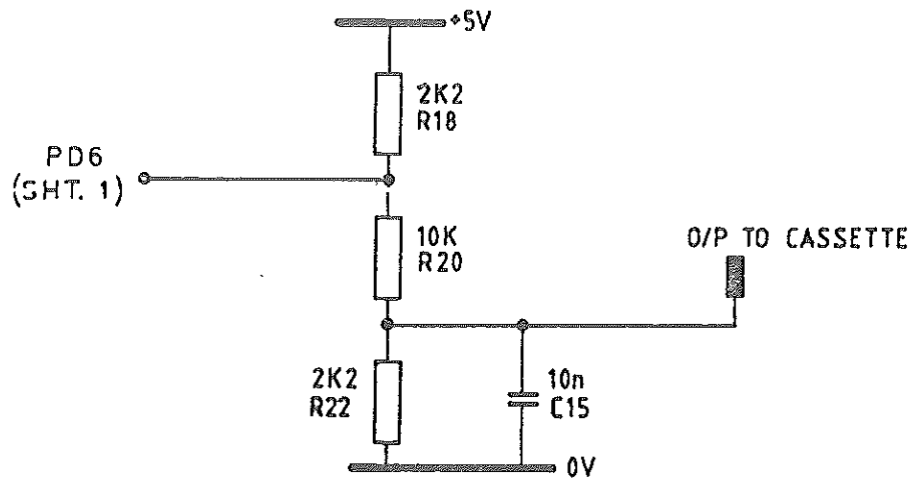


Output To Cassette
SHEET. 5



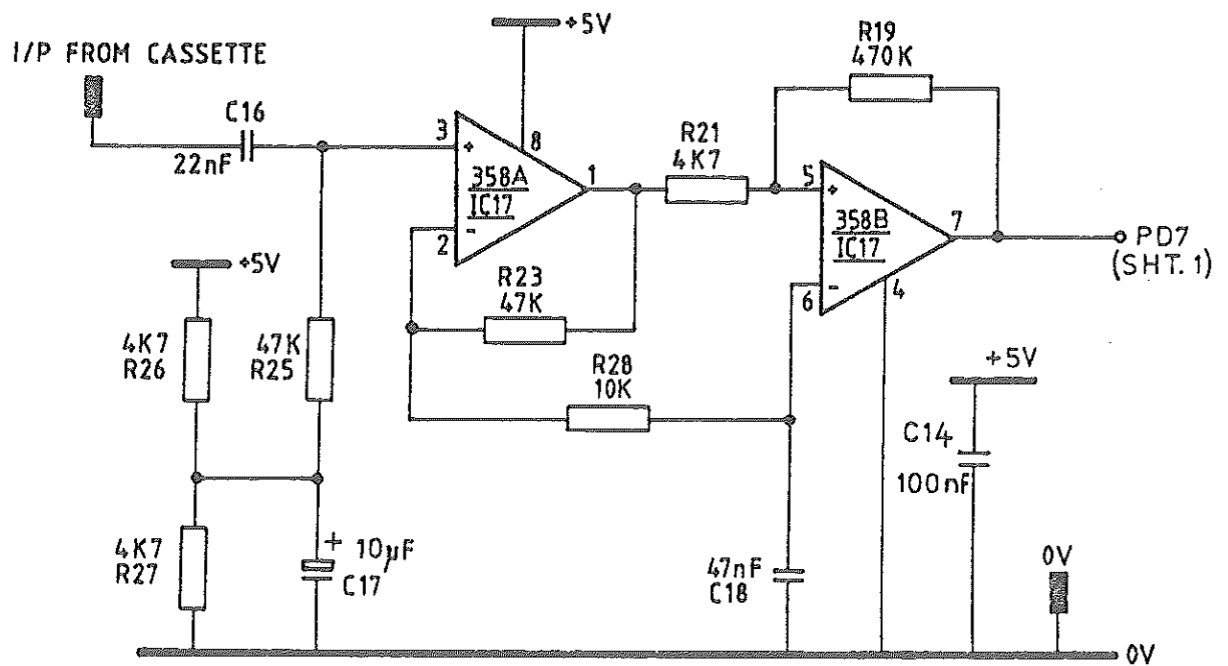
Input From Cassette
SHEET. 6

EMMA II CASSETTE INTERFACE



Output To Cassette

SHEET. 5

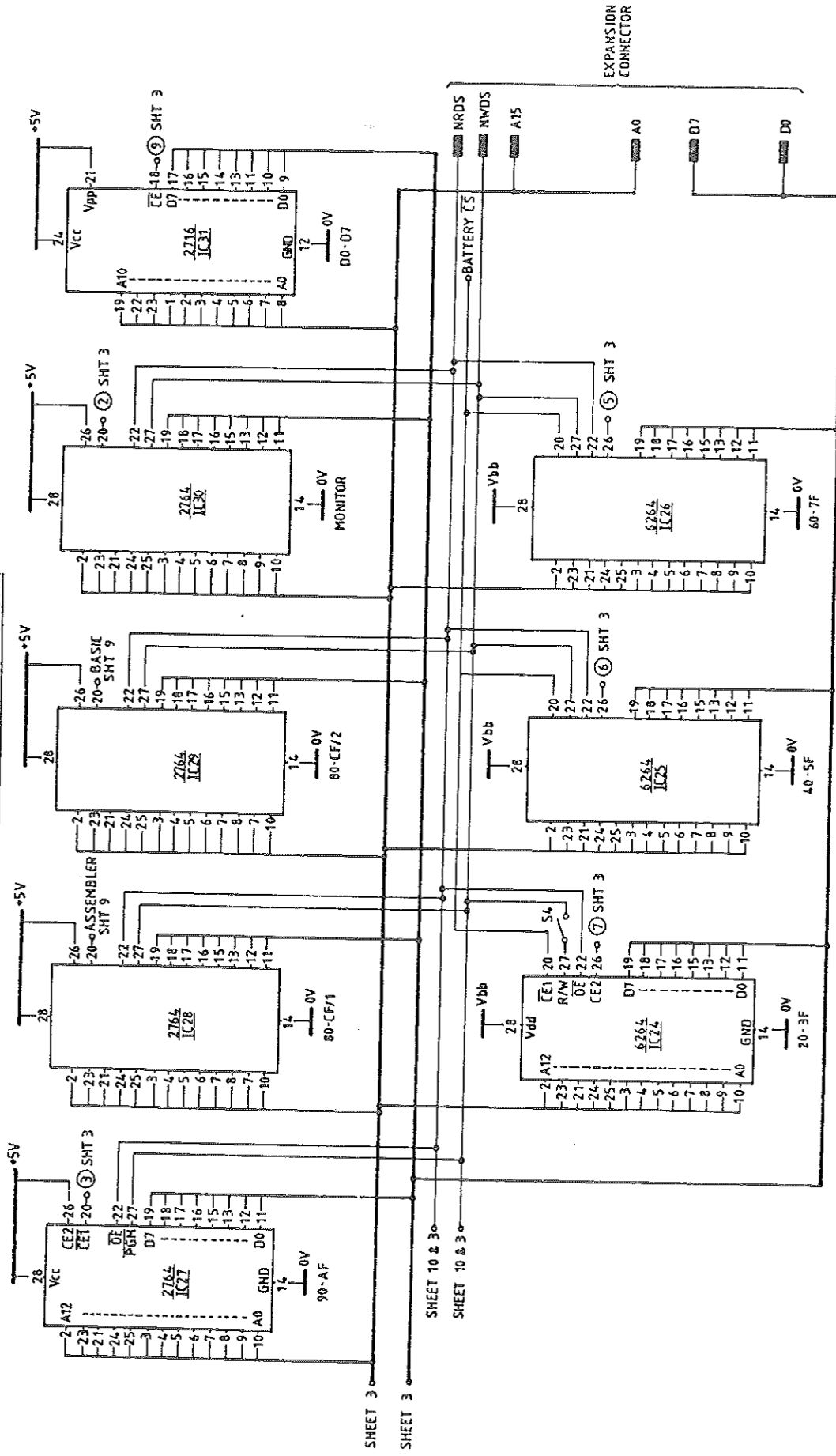


Input From Cassette

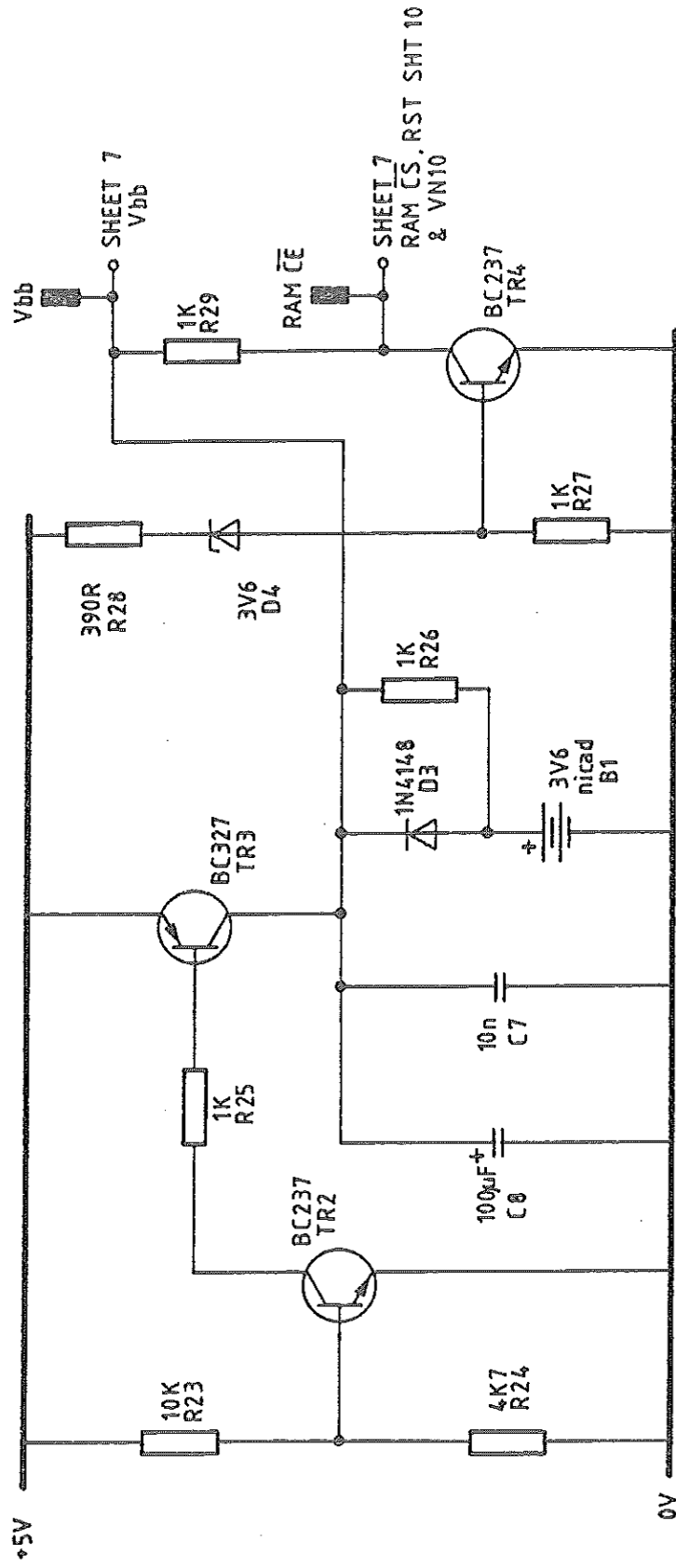
SHEET. 6

[illegible]

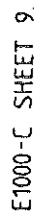
VISA II SYSTEM MEMORY



VISA II BATTERY BACK-UP CIRCUIT.



VISA II KEYBOARD INTERFACE



[illegible]

