LONDON, METEOROLOGICAL OFFICE.

Met.O.15 Internal Report No. 005.

Microprocessors for data-acquisition in Met.O.15.
By CONWAY,B.J.

London,Met.Off.,Met.O.15 Intern.Rep.No.005,1976,
31cm.Pp.iii+17,10 pls.Abs.p.ii.

An unofficial document - not to be quoted in print.

METEOROLOGICAL OFFICE
London Road, Bracknell, Berks.

MET.O.15 INTERNAL REPORT

No.005

MICROPROCESSORS FOR DATA-ACQUISITION IN MET.O.15

by

B. J. CONWAY

DATE : DECEMBER 1976

Cloud Physics Branch (Met.O.15)

Abstract

Software-controlled processors are considered as alternatives to special-purpose hardware for data-acquisition in field experiments. Systems based on microprocessors are examined and a comparison with present methods is made for the case of the ASSP 100.

ELG/

CONTENTS

Microprocessors for Data-Acquisition in Met O 15

1. Introduction

Cloud physics experiments increasingly involve the use of automatic instruments which produce large numbers of readings that must be collected, organised and stored for subsequent computer-analysis.

Until now, our approach to this data-collection problem has been to build a special-purpose electronic logic unit to handle the output of a given instrument, to display that output and to record it on tape.

Because the outputs of different instruments have to be handled in different ways, the hardware-logic unit for one instrument is not suitable for another.

When a new instrument is brought into use a new hardware unit has to be designed, built and "debugged". This has proved a long and expensive process, particularly for the sort of device (such as the ASSP) which we are now using. If these methods continue to be employed every time we commission a new instrument we shall probably spend more time developing data-processors than doing cloud physics or even than developing cloud physics instruments.

2. Alternative Methods: Hardware and Software processing

The difficulty of defining the final system before an instrument has been tried in use and the problem of modification to meet later requirements is well-recognised. Until now the only defence has been to try to incorporate all possible future requirements in the initial design, recognising that some of the features will remain unused, but hoping that changes after completion will be minimised. This approach was employed on the Mk II ASSP–display now used on the aircraft. The price paid for inclusion of subsequently redundant operating modes is an unnecessarily large and complicated device which takes longer to build and debug and with an enhanced ability to go wrong.

Once a hardware data-logger has been built and tested it is difficult to make even minor modifications - this is particularly true if a high component-density has been used to keep the original unit as small as possible. Changes often mean additional Integrated Circuits and there may not be room to accommodate them.

Even small changes often produce unforeseen effects, which require further debugging. In general, hardware devices are so inflexible that any modification is likely to prove difficult and time-consuming.

The above problems can be overcome if the instrument outputs are handled not by special-purpose hardware but by software, that is by a program in a computer. The only requirement on the instrument is that it should present its readings in digital form; connection to the computer is via a standard interface connected to a multi-line bus. A different instrument is accommodated by a different software program - the hardware remains the same. Indeed, several different instruments may be connected to the I/O bus at once. They are serviced in turn by the computer, each device having its own unique "address" and sending data along the bus only when the processor calls it with that address.

Operating changes are incorporated by rewriting the program instead of by making hardware changes. Small changes can still produce unexpected results, but software development is much quicker and easier than the corresponding hardware process and it is not difficult to delete the changes and return to the original system.

The computer contains all the individual functions employed in a hardware processor but instead of wiring the required functions together in a permanent fashion for a particular instrument, the functions are ordered and organised by the software program.

Until recently the use of a software-controlled processor as an alternative to special-purpose hardware has implied the use of a mini-computer probably costing well over £1,000. Further, if an instrument is dependent upon a minicomputer which is installed on the C130 we cannot use that instrument anywhere else (eg for fog-studies). If a sufficiently cheap software-processor can be found these difficulties disappear.

In looking for a cheap alternative to a minicomputer we look at the sort of functions performed by hardware processors. The operations are necessarily very simple and consist mainly of counting pulses and storing groups of bits and of recognising certain preset conditions (eg "Total count = N"), selected from a limited set of options by the operator, via the front-panel switches. Data reorganisation, multiplexing and final storage on tape are accomplished by special-purpose logic and timing circuits. Arithmetic operations are generally avoided in these units whose job is just to get the information on tape.

We see that the above tasks could be performed by a processor with a limited instruction set, using fixed, relatively small programs. The sophisticated programming aids that are a feature of modern mini-computer systems could be dispensed with for small programs, changed infrequently.

There is now available a class of devices known as microprocessors. A microprocessor is a central processor unit (CPU) on a single integrated circuit. It contains the features of CPU's found in minicomputers and in larger machines, viz. arithmetic/logic unit, control unit, status and address registers and buffers for external connections. Program instructions and data are stored in external memory just as in conventional computers.

There is no conceptual difference between a microprocessor and a minicomputer CPU. Indeed some minicomputers are now available employing microprocessors instead of conventional CPU's. Differences are of degree: generally speaking, minicomputers have greater speed, larger instruction sets, longer words, and use larger memory and more complicated software than do microprocessor systems. They are also much more expensive.

3. The Elements of a Microprocessor System

3.1 The microprocessor unit (MPU)

This is shown in Fig 1. It is a box containing the main processing elements: the microprocessor and the memory units. There is also a clock which controls the timing of the operations inside the microprocessor. The clock incorporates a power-on/manual reset which is used to start the processor at the correct program location.

The microprocessor and the memory units are linked by a multi-line signal bus which carries Data, Address, Control and Interrupt lines, the latter only being used by devices external to the MPU.

The bus finds its way to the outside world via 2 I/O "ports" labelled "Master" and "Slave". The significance of the 2 ports is explained later in section 5.2 (Multiprocessor systems).

3.2 Memory

Fig 1 shows 3 types of memory in the MPU. Main stores in conventional computers still tend to use ferrite cores which have the advantage of being non-volatile (the stored information remains when the power is switched off). Microprocessors tend to use semiconductor-memory exclusively, some types being volatile and others non-volatile.

(1) RAM (Random Access Memory)

Volatile Read/Write memory used for temporary storage and as general workspace during program-execution. It may also be used for program storage provided that the program is reloaded every time the power is interrupted.

(2) ROM (Read-only Memory)

Data are fixed into this type of memory by the manufacturer or by the user by irreversibly burning out links in a diode-matrix. This type of memory can be read from but not written into during program-execution and it is of course non-volatile. ROM's are used for storing fixed data such as reference tables and for fixed "loader-programs" for getting other programs into RAM when a suitable storage peripheral (eg disc or papertape) exists. They are not suitable for storing programs which may have to be changed.

(3) EPROM (Eraseable, Programmable, Read-Only Memory)

This is read-only memory which retains its contents when power is turned off but which is capable of being erased and reprogrammed by the user.

The most common type is the so-called "UV EPROM" which retains its contents until exposed to UV light through a quartz window built into the top of the chip. A second type of device which is electrically reprogrammable (no UV needed) has now appeared on the market.

Data are entered into these devices from a keyboard or papertape using a special-purpose machine. Met O 16 have such a machine which is at present set up for the MM5203Q 256 x 8-bit UV EPROM. It can be adapted to program other EPROMS.

The value of the EPROM is that it may hold a program which will then remain stored for indefinite use. When the program is to be modified the EPROM is merely erased and re-written. There is no need to buy a new ROM.

The availability of EPROM's is perhaps the most important single factor in making microprocessors an attractive proposition for our sort of work.

Cost of EPROM is about 6p/word for 16-bit words.

Semiconductor memory of any type is built up to the size required using a number of identical blocks each accommodated on a dual-in-line chip. A single chip may contain typically 1K to 4K bits organised in 4 or 8 bit bytes.

3.3 The I/O Bus

This is the multi-line signal-cable which connects the MPU with external devices. Fig 1 shows the I/O bus as an extension of the MPU memory-bus, so that external devices appear as pseudo-memory - that is, the processor never "knows" whether it is communicating with internal memory or with an external device. The memory map in Fig 2 reflects this, showing external devices as a block of locations within memory.

The I/O bus contains several groups of lines:-

Data lines

If the microprocessor has an n-bit word-length there will be n bi-directional data lines.

Only the MPU or a single device may attempt to control the state of a bus-line at any instant. The data-lines carry information in both directions between the MPU and its external devices. By having one line for each bit in the word, the bus is able to carry words in parallel form during transfer operations.

All transfers via the bus are between a device and the MPU. Devices do not communicate directly with one another via the bus. Thus at all times the MPU controls bus-usage and the flow of information from one device to another. This scheme is part of the essential difference between a software-controlled processing system and the hardware processors used in the past.

Address lines

The MPU allows a device access to the bus by placing the number which is the "address" of that device on the group of bus lines known as "address lines". If the bus contains m address lines then up to 2^m devices may be connected to the bus (subject of course to electrical loading considerations).

Only the MPU is allowed to control the states of the address lines.

Control lines

A few lines (perhaps 3 or 4) for I/O timing, ready-flag sensing, etc. The states of some of these lines are controlled by the addressed device, others by the MPU.

Priority Interrupt (PI) lines.

Priority interrupt facilities are often provided in microprocessors, the Texas Instruments TMS9900 for instance has a hierarchy of 16 PI levels. N PI levels imply N PI lines in the bus but we should probably never need as many as 16 and not all processors supply this many. A reasonable size for the PI bus would probably be about 6 lines.

A PI request from a device, if accepted, causes the processor to make a jump to a subroutine where the programmer has defined a process to deal with that interrupt-level. The processor's last action before returning from the interrupt routine is to reset the interrupt request flag in the device.

The PI feature can be used in 2 ways:-

(1) A slow device (eg digital cassette recorder) can use PI to request data-transfer and save tying up the processor in an inefficient wait-loop.

(2) A clock can generate a PI at regular intervals (eg every 0.01 sec) and cause the processor to scan a number of devices which do not merit their own PI facility (eg front-panel switches).

<u>I/O bus size</u>

Assuming a 16-bit word, ability to address 128 external devices, 6PI levels and 4 control lines the bus must contain 33 signal lines.

3.4 <u>"Peripherals", "Devices" and interfacing</u>

A "peripheral" is a physical unit (eg ASSP 100, cassette recorder, set of front-panel switches) connected to the I/O bus.

A "device" is a register of up to 16 bits (if there are 16 data-bus lines) with a unique address for communication with the MPU.

A peripheral may contain several devices and the MPU may select for I/O any device within a peripheral.

Some devices are input devices, that is they send data <u>to</u> the MPU, others are output devices and receive data <u>from</u> the MPU. The terms "input" and "output" are defined always from the point of view of the MPU.

Fig. 9, showing a system for running the ASSP 100 illustrates the connection of multi-device peripherals to the bus.

A device is connected to the I/O bus through an "interface" unit contained in the peripheral. The main functional form of an interface is shown in Fig. 3.

The job of the interface is to make sure that a device only places information on or takes information off the bus when that device is being addressed by the MPU, and to ensure that the timing of the transfer is correct.

The interface therefore contains an address-decoder as well as gates and latches to strobe data onto the bus or to collect data from the bus.

The interface also contains a "ready-flag" which tells the MPU, via a control line, when the device is ready to send or receive. The MPU resets this flag to "not-ready" when the transfer is performed and will not make another transfer to or from that device until the flag has again been set.

Not all devices require a ready-flag. For example the MPU can read the states of front-panel switches whenever it wishes so these devices send a "ready" signal every time they are addressed.

Devices which are allowed to interrupt the processor have a PI flag connected to a line in the PI bus. This flag is set by the requesting device and reset by the MPU at the end of the PI service routine.

Above are described the main functional elements of any microprocessor system. These are: MPU, I/O Bus, Interfaced I/O devices. Signal-conditioning devices such as level shifters and line drivers are of course incorporated where appropriate. This applies equally to a special purpose hardware system.

Any system, however complex, is built up by connecting together numbers of the above elements.

4. Data-Acquisition Systems employing Microprocessors

We can envisage increasingly complex systems built up from units described in the previous section.

4.1 Single-processor systems

Fig. 4 shows the minimum data-logging system with one measuring instrument.

If the instrument produces an analogue output this is converted to digital form before being connected to the I/O bus through a standard interface. Readings from the instrument are collected by the MPU, processed and recorded on tape.

The processing stage might amount to no more than temporarily storing the readings, but more complicated tasks are obviously possible. Blocking the results for storage on tape would be under the control of the programmer rather than being determined by predesigned hardware.

Fig. 5 shows the natural extension of the simplest system to include more than one measuring instrument and a number of other peripherals. The limit on this sort of system is imposed by the capacity of the MPU to handle the increasingly large amount of data coming in on the bus.

4.2 Multiprocessor systems

It may be beyond the capacity of a single MPU to collect and process the outputs of all the instruments in use in an experiment. A very fast instrument might alone occupy an MPU. It might nonetheless be desirable to bring the collected and processed results from all the instruments to a central point for display and storage. This can be achieved by connecting several MPU's together, and Fig. 6 shows perhaps the simplest way of doing this, the Master/Slave system.

The Master-Slave system uses the 2 I/O ports shown in Fig. 1 to overcome the problem of more than one MPU simultaneously trying to control the states of the I/O bus lines. When an MPU is connected to an I/O bus via its master port it is able to control the states of the address and control lines. It therefore controls access to and use of the bus by everything else connected to that bus.

In contrast, when the slave port is used the MPU is unable to control the address lines and its use of the bus is controlled, like that of any other device, by the MPU which "owns" the bus (ie is connected to it through a "Master" port). It cannot communicate with other devices on that bus other than through the master MPU.

The slave port is essentially a device-interface (appearing as at least one input and one output device) with the slave MPU lurking behind it.

Clearly, only one master MPU can be connected to a given I/O bus but the number of slave MPU's is not limited except by the usual considerations which apply to other peripherals.

A slave MPU is connected to the devices which it services and controls via its own I/O bus which is plugged into its Master port. This arrangement is illustrated in Fig. 6.

4.3 Combined Minicomputer/Microprocessor systems

This is a better version of the multiprocessor system described above and is illustrated in Fig. 7. We replace the Master MPU by a minicomputer and produce a system with advantages over either a minicomputer alone or a system using only microprocessors.

The MPU's perform the very low-level data-collection tasks which would otherwise occupy a wastefully large amount of the minicomputer's time.

Because the minicomputer receives measurements in predigested form from the MPU's it is free to concentrate on processing and displaying the incoming information. These are tasks requiring more complicated programs which are easier to develop on a minicomputer.

Use of the minicomputer minimises the processing and arithmetic done by the MPU's and allows them to use smaller amounts of memory. Use of the slave MPU's releases the more powerful and expensive minicomputer for number-crunching tasks to which it is better suited.

This sort of system is employed by Beukers for the dropsonde data-acquisition task.

5. Choosing a microprocessor

I have tried to list below the main factors to be considered when choosing a microprocessor. The factors, though considered separately, are not truly independent since excellence in one respect can compensate directly for deficiency in another.

(1) Word-length

Main options available are 4, 8 and 16 bits. Also available are the so-called "slices" - usually of 4 bits - which are parallelled together by those users with sufficient enthusiasm to make up a desired word-length of 4n bits.

As with minicomputers, a longer word generally allows a better instruction set and/or faster and easier memory-addressing. Word-length for modern minicomputers is typically 16 bits.

Word-length should be considered in connection with the sort of I/O and memory data to be handled. If this is often naturally organised in groups of more than 8-bits then a 16-bit processor will make life easier for the programmer. It is generally easier to do 8-bit work with a 16-bit processor than vice versa.

(2) Speed

Microprocessors are based on MOS technology in order to achieve the required level of circuit integration. This makes them intrinsically slower than CPU's based on TTL. An order of magnitude difference in speed is typical with a microprocessor needing 5-10 μS for the simplest instruction compared with $< 1\ \mu S$ for a minicomputer.

Within the class of microprocessors a range of speeds is available. Thus the Texas Instruments TMS 9900 will perform a jump in 3.3 μS and a multiply in 17.3 μS while towards the other end of the scale the National Semiconductors ISP-8A/500D takes 20 μS for a jump and has no hardware-multiply instruction.

The speed of the processor must be high enough to service the real-time devices to which it is connected. In this way, connection to real-time devices can impose a definite go/no go speed criterion in a way that programs not operating in real-time do not.

(3) Instruction Set

The instruction set (the list of operations from which programs are constructed) is usually a fixed feature of the processor and varies both in its size and in the types of operations provided, from one microprocessor to another.

In general, the larger the instruction set, the easier a given programming task is likely to be. A good instruction set means fewer program steps and therefore a faster program occupying less memory. The speeds of 2 comparable processors cannot therefore be properly compared without also looking at their instruction sets.

Comparison of two instruction sets is complicated by the fact that many instructions may operate in a variety of addressing modes so that the true power of an instruction set may not be revealed merely in terms of the number of basic operations included.

In the end, the only way to choose between processors for a given task is to try to identify the critical parts of that task and to code these parts in terms of the instruction sets of the contenders. At the moment there are between 30 and 40 microprocessors on the market and only a few of these are fast 16-bit devices. The range is probably therefore still just small enough for us to be able to choose from it in a rational way.

(4) Interrupt-handling

Facilities for handling interrupts are virtually indispensible in a processor used for data acquisition from a variety of real-time devices. Given the existence of such a facility, processors differ from one another in the number of interrupt priority-levels available and in the way a jump to the interrupt service routine is accomplished. The value of sophisticated interrupt-handling relative to that of other features can only really be assessed in a particular application.

(5) Price

The price-range for currently-available microprocessors is from about £10 to about £60 in small quantities. These prices are for commercial-grade devices. Military spec. versions, where available, are more expensive.

Fast 16-bit devices are at the top end of the price range, slow 8-bit devices at the bottom.

Given that one's application should narrow the choice of units anyway, the difference in price between suitable alternatives is unlikely to be a significant factor, if only a few units are to be bought.

For an MPU using one of the more expensive processors and say 1K of memory at 6p/word the cost of the electronics items in the MPU will be of order £150-£200. Cost of plugs and sockets and putting it in a box (the usual neglected items) will probably amount to at least another 50% of this.

The price is still an order of magnitude lower than a minicomputer.

I estimate that it might amount to three month's work for one man to "breadboard" a prototype system and get it working, assuming we choose to make our own MPU.

6. A microprocessor system to replace existing special-purpose hardware An example - the ASSP 100

6.1 Present system

Data from the ASSP 100 are currently handled by the Mk II data-logger/display* whose internal structure I have illustrated in Fig. 8.

Three outputs are produced by the ASSP:

* referred to hereafter simply as "the Mk II".

(1) Total Particle Count. Every particle seen by the ASSP produces a pulse which increments the contents of a 24 bit counter in the Mk II.

(2) Strobe and Channel-number. When a droplet is fortunate enough to pass through the inner sampling volume the ASSP guesses its size, presents this as a 4-bit channel number and generates a strobe pulse to say that it has done so.

(3) Particle Transit-time. An 8-bit number which represents the average time taken by droplets to pass through the sampling-volume. We usually call this number the "velocity".

The Mk II accumulates measurements from the ASSP, displays them, and, when it has enough, stores them on a tape cassette (via an external recorder).

At the end of a collection period the contents of the "Total Particle Count" counter are latched into 3 bytes of a 40-byte store (1 byte = 8 bits) which occupies the rear half of the Mk II and is known amongst other things as the "GPIU" (general purpose interface unit).

At the same moment the velocity (transit time) is latched at the input to the Mk II and into 1 byte of the GPIU.

Each of the 15 size-channels has its own 12-bit counter. During the collection period, whenever a strobe is received from the ASSP the appropriate channel counter is incremented, as is a separate counter which contains the total of the channel contents. At the end of the collection period these channel counts are latched into 26 bytes in the GPIU. The contents of individually selected channels can be divided by ten (during accumulation, not after) by switching any of 15 decade counters in between the channel counters and the ASSP. These switch-settings are latched into 2 bytes in the GPIU.

The Mk II also contains two clocks, a GMT clock (recorded in 2½ GPIU bytes) and an "elapsed time" clock which records the collection period to 1/100th sec (provided it is less than 41 seconds) and which takes 1½ bytes in the GPIU.

The operator uses the front panel switches to select, from a limited number of fixed options, such things as the duration of the collection period or the total number of strobes to be accumulated.

Because the switches are hard wired into the elements they control, changing a switch-setting during accumulation of a spectrum can render that spectrum meaningless.

46 bits are required to encode all the front panel options currently available.

The GMT clock is displayed in numerals. All other values are displayed as binary numbers by linear LED arrays. In the case of the spectrum this is actually quite good since the spectrum appears in graphical form with a logarithmic vertical axis.

The front-panel options, such as the values of the 4 different allowed collection periods, could be changed, but only by making hardware alterations. Because the switches have specific functions and are hard wired in, their functions are essentially permanent - we cannot change the meanings of the switches just as we please.

Control and timing circuits are dedicated to specific tasks, and because the extra complications of multiplexing all the internal data transfers have been avoided we find there are about 300 signal wires going into the GPIU.

The operations performed in the Mk II are seen to be quite simple and its storage-capacity is meagre. It does however consume a lot of hardware. The unit is the size of a suitcase and weighs 45 lbs. It contains 30 printed circuit boards and about 320 integrated circuits which, with the displays, draw 10 to 12 amps from the low-voltage supply and generate enough heat to require forced cooling. The unit took about $\frac{1}{2}$ man-year to put together.

6.2 Alternative using an MPU

The organisation of a replacement system based on an MPU is shown in Fig. 9. The MPU at the top of the diagram is the same as that shown in Fig. 1. It could be accommodated on a single printed circuit board but in practice it might be convenient to put such items as I/O bus drivers on a separate board.

The rest of the system is organised as 4 peripherals:-

(1) ASSP 100. This contains 4 devices and a PI flag which is set by the strobe signal. The Total Particle Count counter has been moved into the ASSP because its input pulses arrive too fast to be handled by the MPU. It is still allowed 24 bits and with the 8 bits of the transit time occupies two 16-bit input devices. (Each of the small rectangles in Fig. 9 represents a 16-bit input or output device. The output

devices latch the information from the MPU during transfer).

The encoded channel-number occupies the 4 least significant bits of an otherwise empty 16-bit input device. This presents it in convenient form for the strobe servicing routine. Arrival of a strobe sets the PI request flag which may be either used or ignored by the MPU according to a mask set by the program.

The 4th device is an output device which is used by the MPU to control the ASSP. Bits in this control-word start and stop the ASSP, set the range and switch the de-icing heaters on and off.

(2) Front Panel. This is seen as a peripheral in its own right and is plugged into the bus just like any other peripheral.

It is shown with 3 input devices and 5 output devices.

The 3 input devices are front-panel switch settings. They provide a total of 48 bits which are more than enough to encode all the currently available options. The switch-settings are just bits in a word. They have no predefined meanings. If we wish to change the meaning of a particular switch we just change the program in the MPU and change the label beside that switch. No rewiring is involved.

The output devices are concerned with displays. I have drawn them as though the transit time, elapsed time and Total Particle Count were to be displayed in binary form as at present. If numerical displays were chosen instead, conversion to BCD could be performed by logic circuits in the front panel, or by the MPU before transmission, in which case the output device organisation would look slightly different.

Two output devices are used for the droplet-spectrum display, one for X (the channel number) and one for Y (the number of particles in the channel. I have indicated only 4 bits being used for X which implies only one channel at a time in the matrix being addressed.

A simpler LED-matrix display can be built with less wiring than at present if we scan the display, channel by channel. With a step interval of, say, 0.01 sec the whole spectrum is scanned at 6 Hz which should be adequate.

If we should later wish to install a physically different front panel, say with a keyboard or a different display, this could be built and just plugged in when ready, without taking the system out of commission.

(3) Real-Time clock. This combines the functions of GMT clock and elapsed time clock. It occupies 2 input devices and registers GMT to 0.01 sec accuracy.

A corresponding "clock-mask" occupies 2 output devices and is set to any required time by the MPU. A digital comparator constantly compares clock and clock-mask and generates a PI request when they are equal. In this way the MPU is able to preset any data collection period.

The real-time clock also generates a second PI every 0.01 sec, which is used by the MPU as a reminder to update the spectrum display.

(4) Cassette Recorder. This has 2 input and 2 output devices and a PI flag. One O/P device is used for the byte being recorded and the other O/P device enables the MPU to control the function of the tape recorder (play, record, etc). One I/P device is used for the replayed byte when in playback mode and the other gives the MPU status information like "end of tape".

The PI flag is set by the recorder after a byte has been recorded or replayed to attract the attention of the MPU. This is because the recorder is such a slow unit ($27\frac{1}{2}$ ms/byte at present; 7 ms/byte for the new recorders).

Note that this system permits replay for display of the recorded spectra. This facility is not available on the Mk II.

Service Routines

Fig. 10 is a flow chart of the overall program for the system described above. It is by no means the only possible program. Storage on cassette is shown as a separate process at the end of the collection period. It could instead be performed during the next collection period so as to achieve continuous sampling.

The critical part of the program is the routine which accepts channel numbers - the Strobe Service Routine (SSR). It should be fast enough for few strobes to be lost.

The SSR is shown in Fig. 11. Its job is to sort incoming measurements and accumulate them as a histogram in a set of consecutive locations "Base + 1" to

"Base + 15" as shown in Fig. 12. It also has to be able to recognise a "spectrum-complete" condition such as "any channel full" or "Strobe total = N". The example in Fig. 11 assumes the latter case.

The ASSP 100 cannot generate strobes at intervals of less than 11 μS because of internal delays. As the particle density and airspeed are increased an increasing fraction of strobes is rejected because optical coincidences are detected and electronic "lockout" conditions occur.

At an airspeed of 90 m.sec^{-1} the strobe rate peaks at a particle-density of about 250 cm^{-3}. Droplets are then passing through the inner sampling volume at a rate of 12000 sec^{-1} but because of coincidences only about 37% of these produce strobes which therefore arrive separated by a mean interval of about 220 μS.

For the TMS 9900 microprocessor, operated at its maximum clock frequency of 3MHz the SSR shown in Fig. 11 has a minimum loop-time of 22 μS (= 10% of the strobe interval). The SSR is shown using a wait-loop. If instead a strobe-generated PI were used to initiate the SSR this would add a further 19.3 μS making 41.3 μS. This is still less than 20% of the minimum average interval between strobes so we conclude that this processor would be capable of servicing the ASSP 100 output.

The SSR is the only critical section of the program. All the other devices in the suggested system make fewer demands on the processor.

7. Conclusion

I have shown that a microprocessor can be used as a data handler for a fast cloud-physics measuring instrument, the ASSP 100. Most instruments would impose a less severe burden on a processor though we might expect in future to use instruments comparable with the ASSP 100.

A second ASSP 100 is about to be commissioned for MRF. Data-logging arrangements must be made for it and the choice is between a special-purpose hardware unit and a software controlled processor.

If a special purpose hardware unit is built we either design a new unit or we make a copy of the Mk II. If the former we have another design and debugging problem and if the latter we must construct a large and complicated unit that will take a long time to build. The physical size alone of the Mk II is likely to make it difficult or impossible to use both ASSP 100's simultaneously on the C130.

Either way we will be back to square 1 when the time comes to accommodate the next new instrument.

Alternatively if we opt for the MPU system we can build a smaller, simpler but more powerful and flexible system which is relatively easy to modify and extend. We will also be in a very good position to accommodate a new instrument of a different type with a minimum of delay.

It must be a matter of concern to everyone involved to see the kind of delay that at present seems inevitable in getting a new instrument into operation. Only a part of this delay is caused by development of the instrument itself; another major contributor is the provision of suitable data acquisition facilities.

I believe that the introduction of software-controlled general purpose data-loggers is the only way to get new instruments into use quickly and that micro-processors offer us the best means of making this step.

GENERAL MICROPROCESSOR UNIT (MPU) - INTERNAL STRUCTURE.

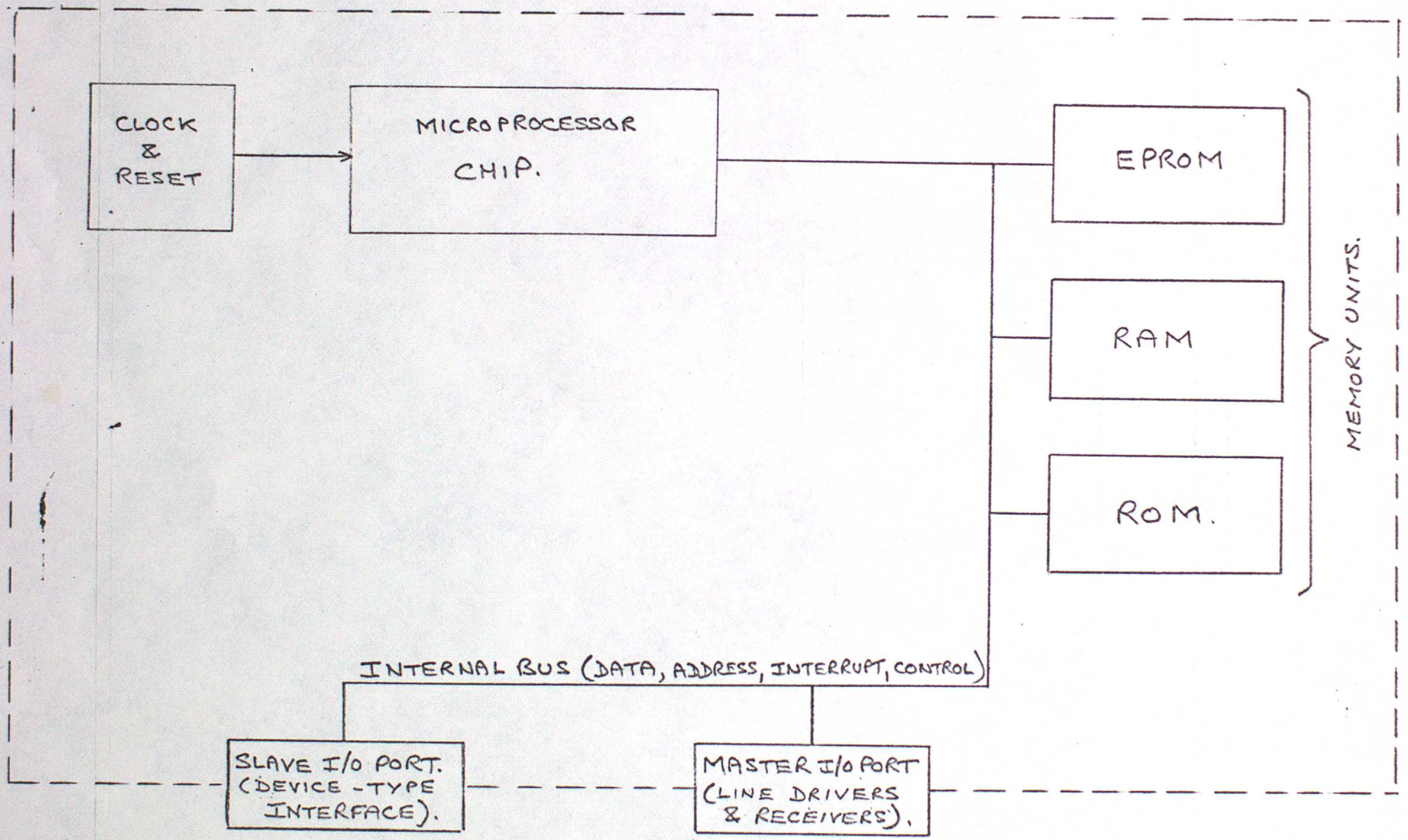

CLOCK & RESET
MICROPROCESSOR CHIP.
EPROM
RAM
ROM.
MEMORY UNITS.
INTERNAL BUS (DATA, ADDRESS, INTERRUPT, CONTROL)
SLAVE I/O PORT. (DEVICE -TYPE INTERFACE).
MASTER I/O PORT (LINE DRIVERS & RECEIVERS).

Fig. 1

Fig. 2

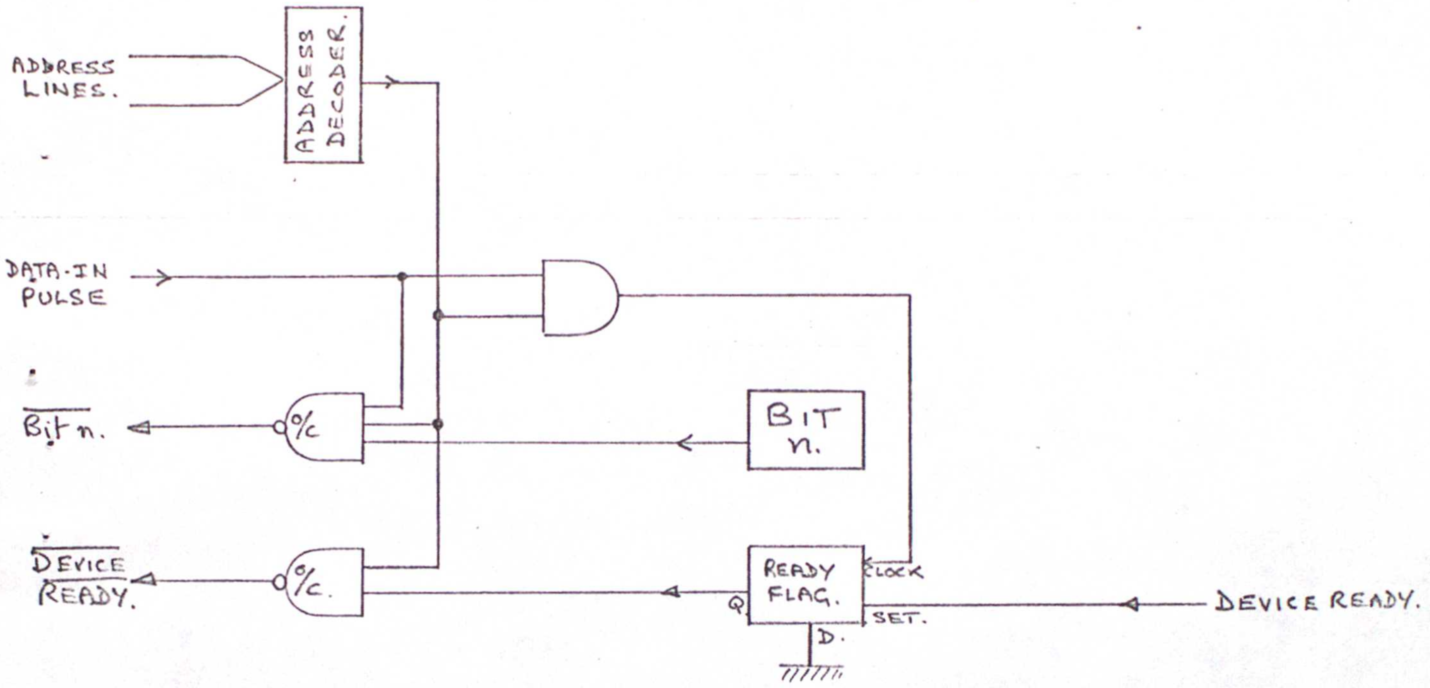
MEMORY MAP - MAIN FEATURES.

BYTE ADDRESS$_{16}$	CONTENTS.	MEMORY TYPE.
0000 007E	64 words (≡ 128 bytes) reserved for interrupt vectoring etc. (TMS 9900)	EPROM
Size (in multiples of 256 wds) depends on application. Probably <1K. wds	General program area. Non-volatile but eraseable.	
Expandable to suit application	General read-write area. Workspace and temporary storage. Volatile memory.	RAM.
— " —	Fixed data & Reference tables Non-eraseable.	ROM.
ddØØ ddFE	128 16-bit locations each correspond-ing to 1 input or 1 output device on the I/O bus. Address limits chosen so that hardware can recognise any address of form dd-- as I/O.	PSEUDO-MEMORY (devices on I/O bus)

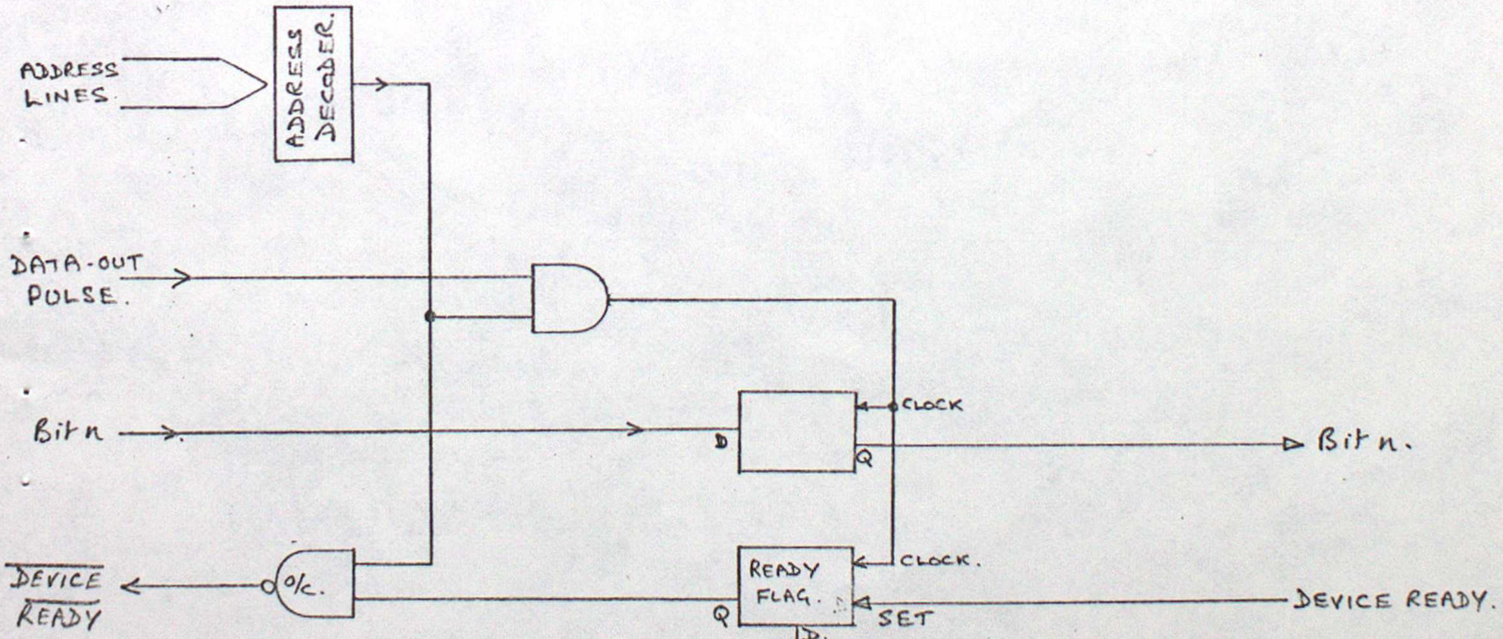
DEVICE INTERFACE.

Fig 3.

INPUT DEVICE.

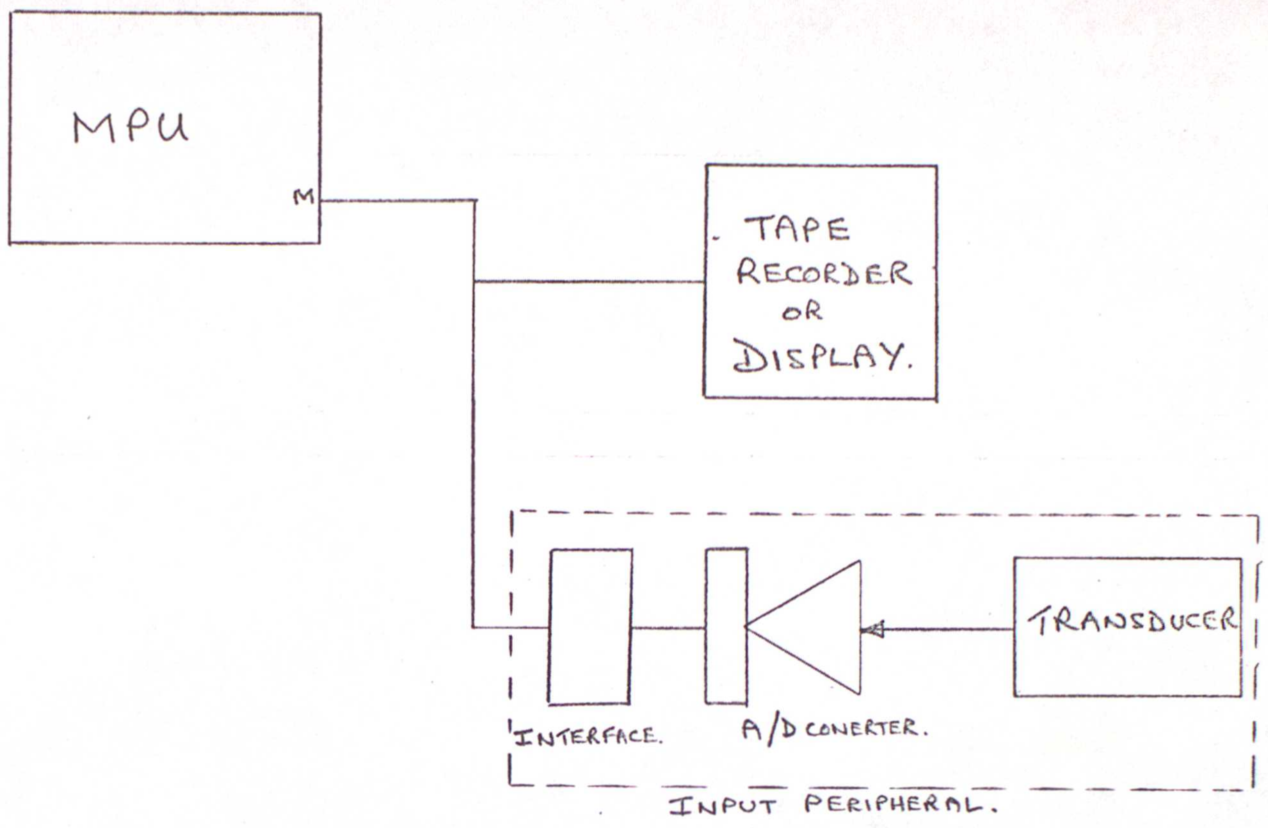


OUTPUT DEVICE.



Basic MPU Data-logging system

Fig 4



Multi-peripheral system

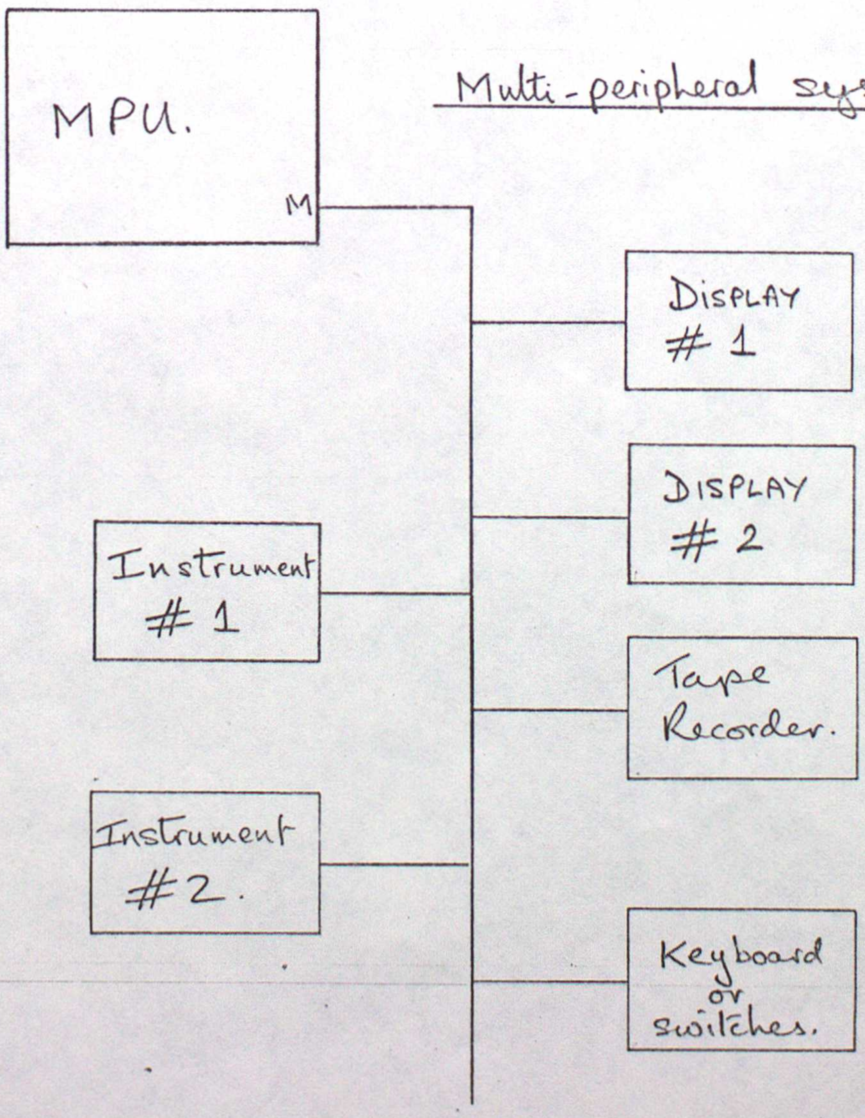
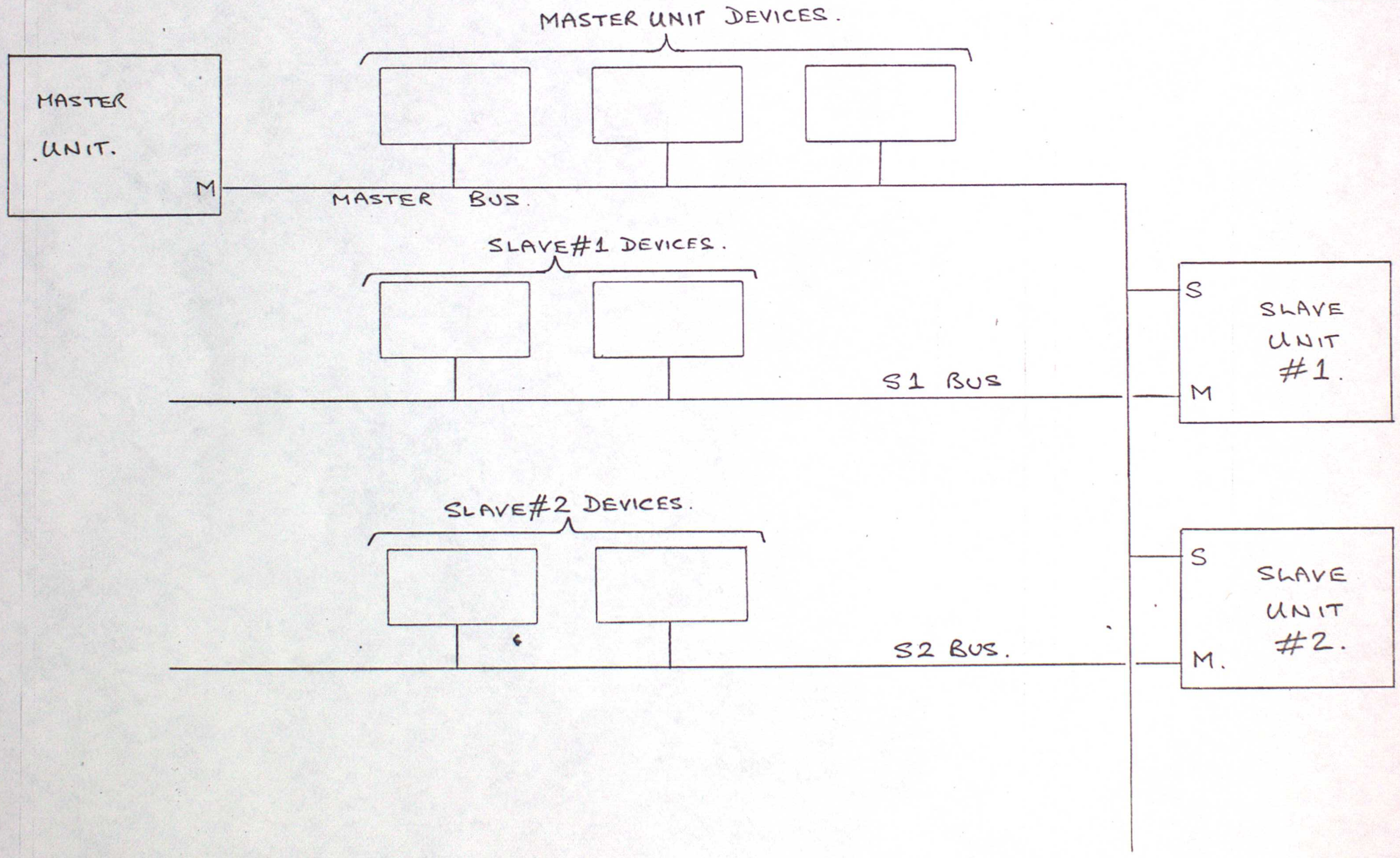


Fig 5.

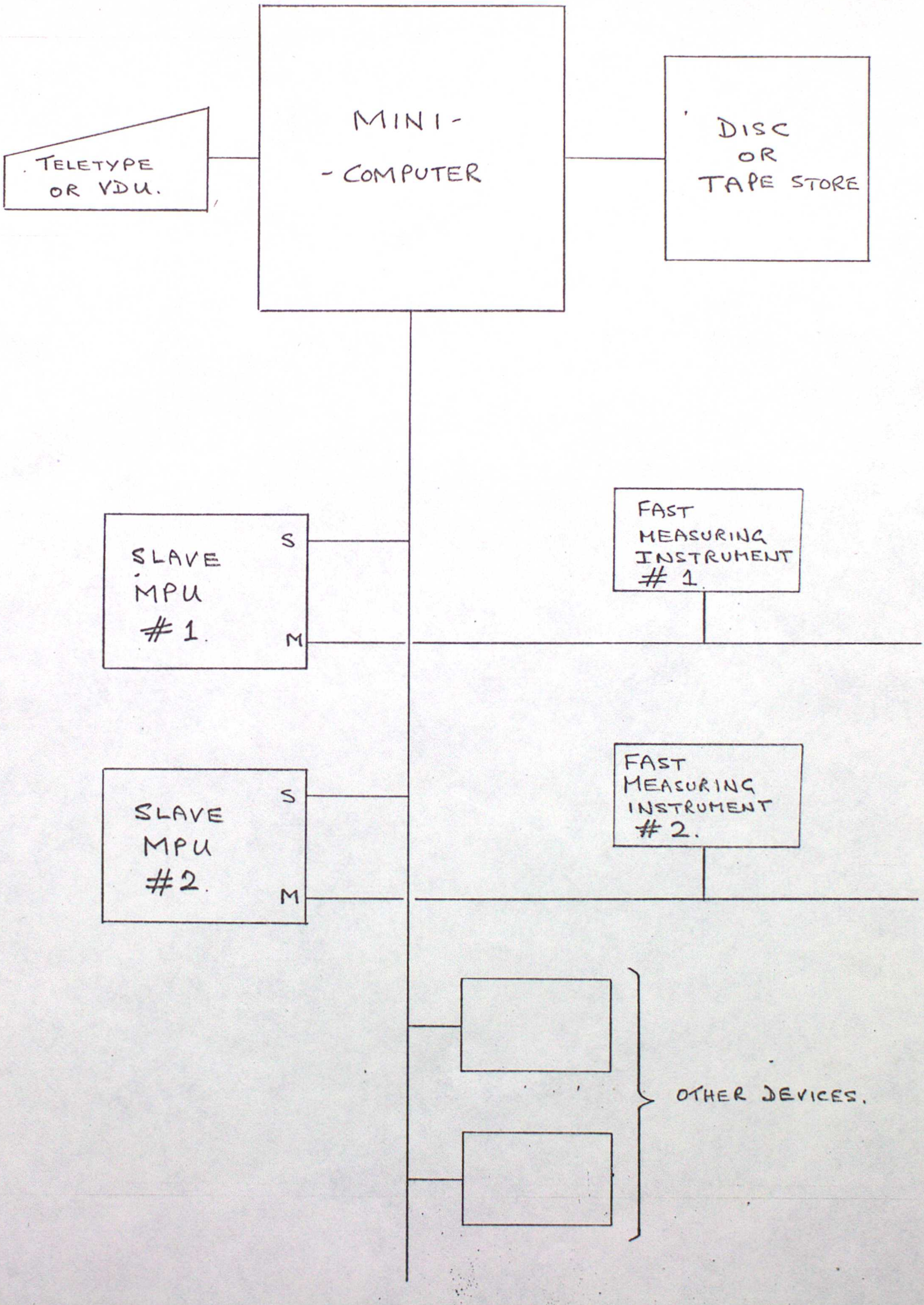
MASTER-SLAVE SYSTEM FOR MORE THAN ONE PROCESSOR.

Fig 6.

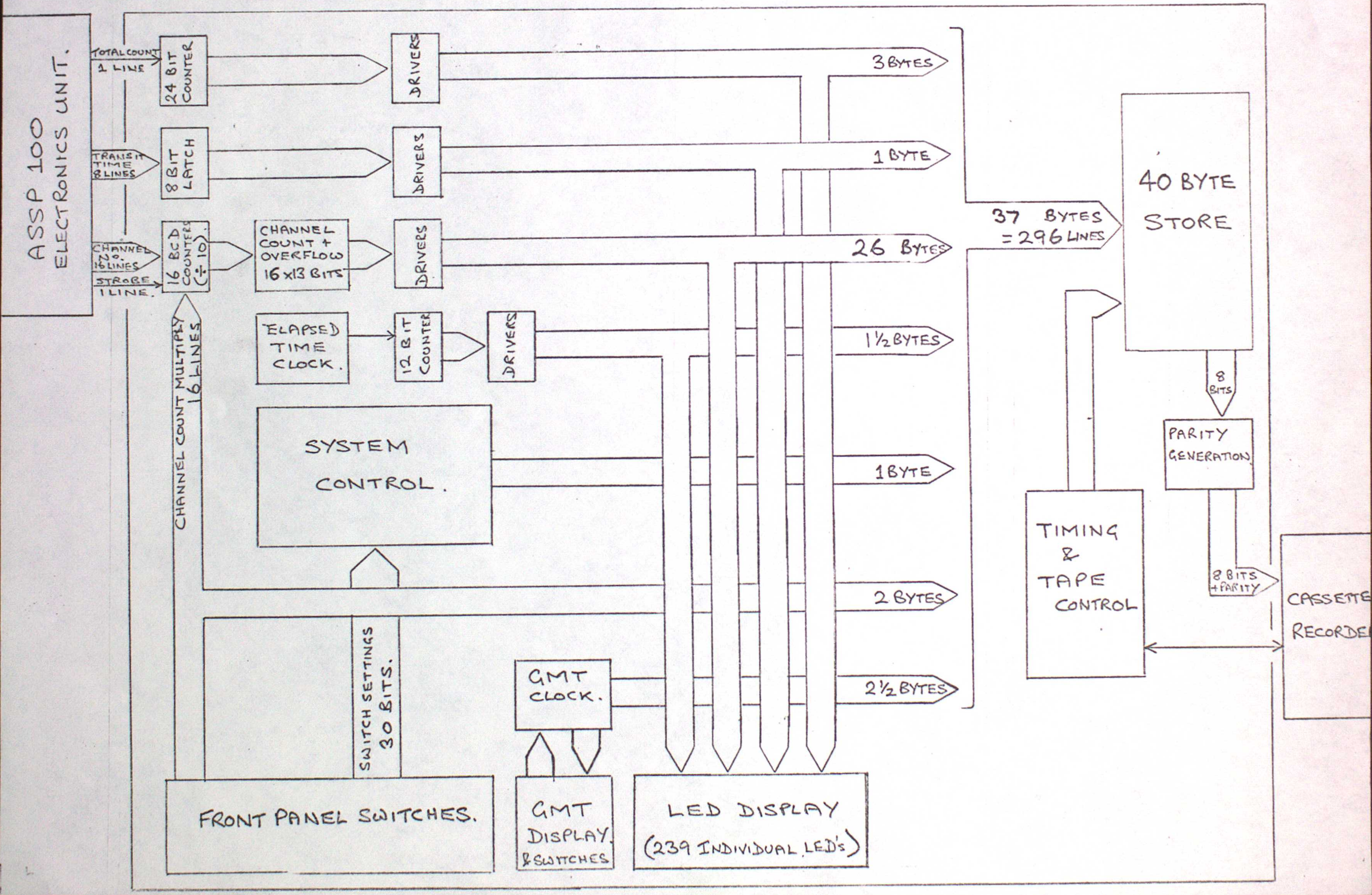


COMBINED MINICOMPUTER / MPU SYSTEM.

Fig 7.

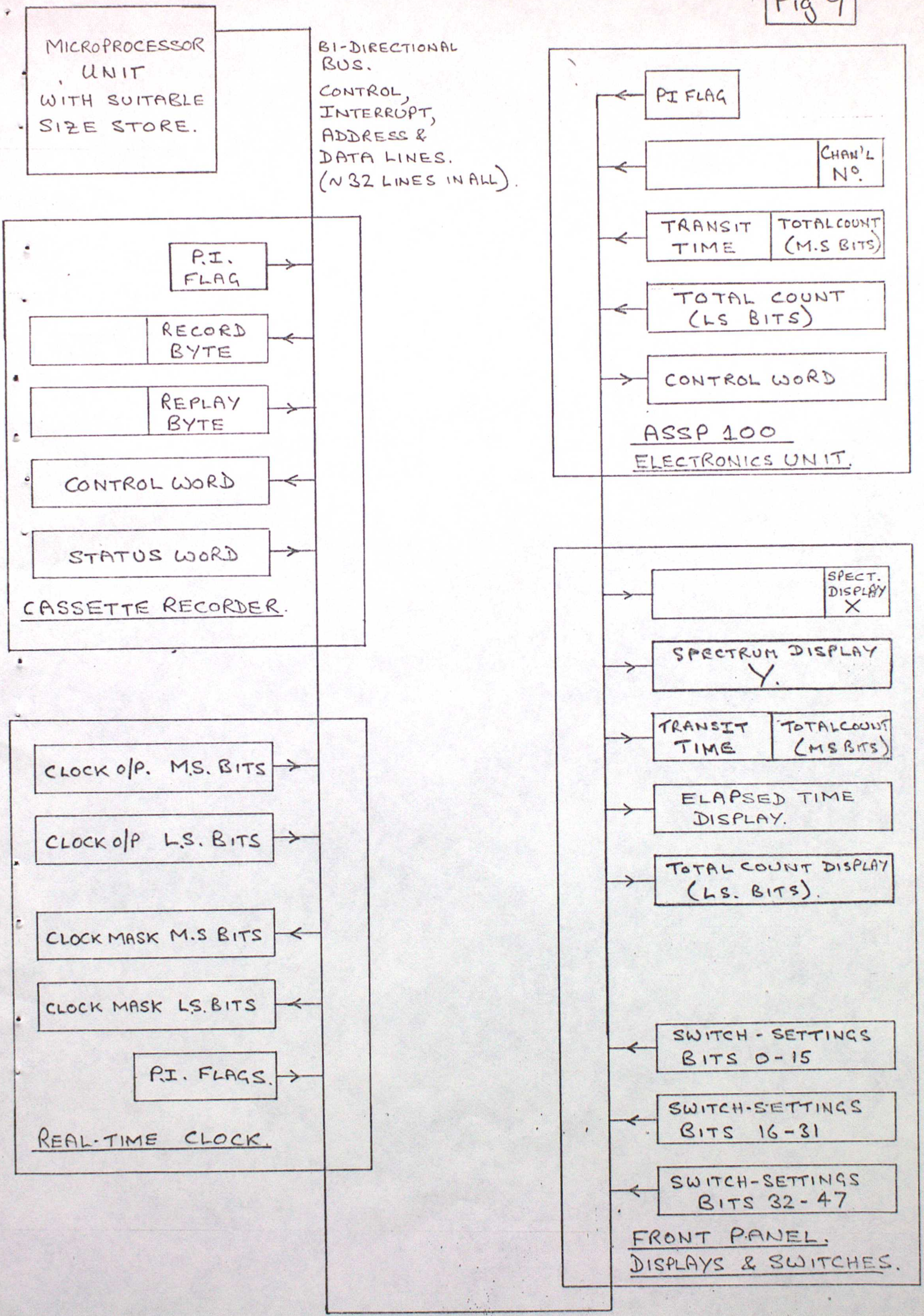


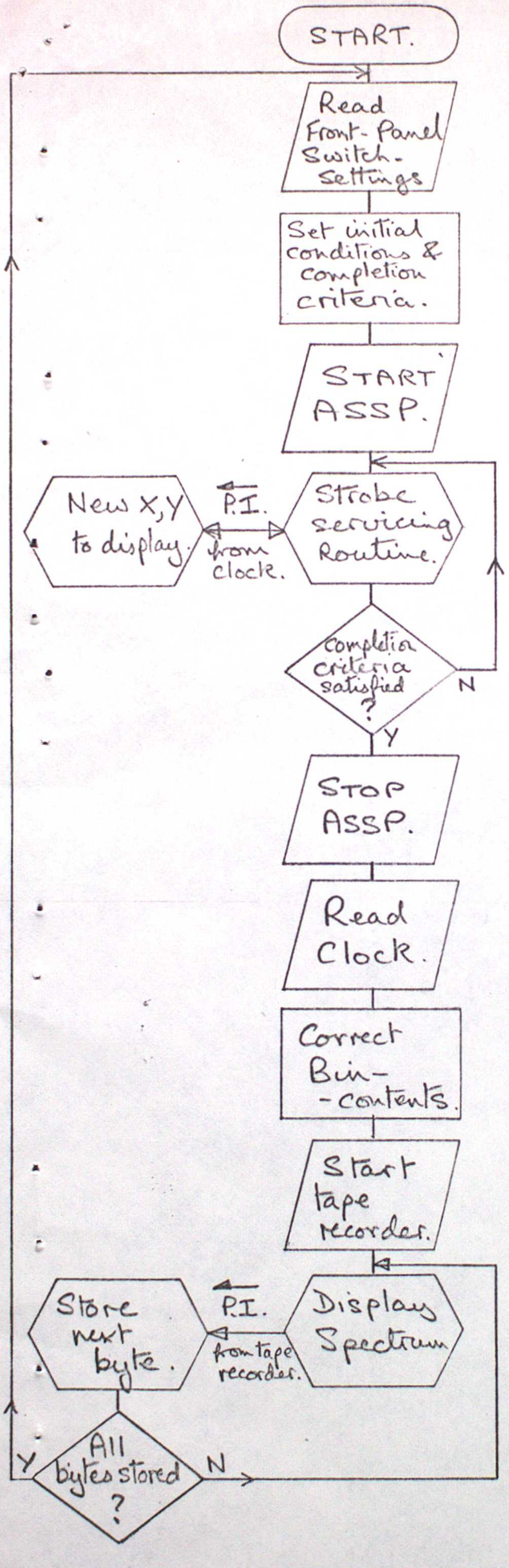
ASSP 100. Mk2 DISPLAY & DATA LOGGER. SIMPLIFIED BLOCK DIAGRAM. Fig 8.



ASSP100 SYSTEM USING MICROPROCESSOR

Fig 9





Get operating-mode for next spectrum.

eg set channel-total counter to −N or put −n in all channels, or set clock mask for current time + T.

And set range, zero total-count.

Loop continues until final condition set up above is satisfied. Interrupts at 0·01 sec intervals from clock are used to update the display. (The display can show either this spectrum or the last).

Time of completion of spectrum.

eg If −n placed in all channels at start, add n to all channels now.

Display this spectrum, also elapsed time, transit time, and total count.

Store next byte in sequence whenever tape recorder interrupts.

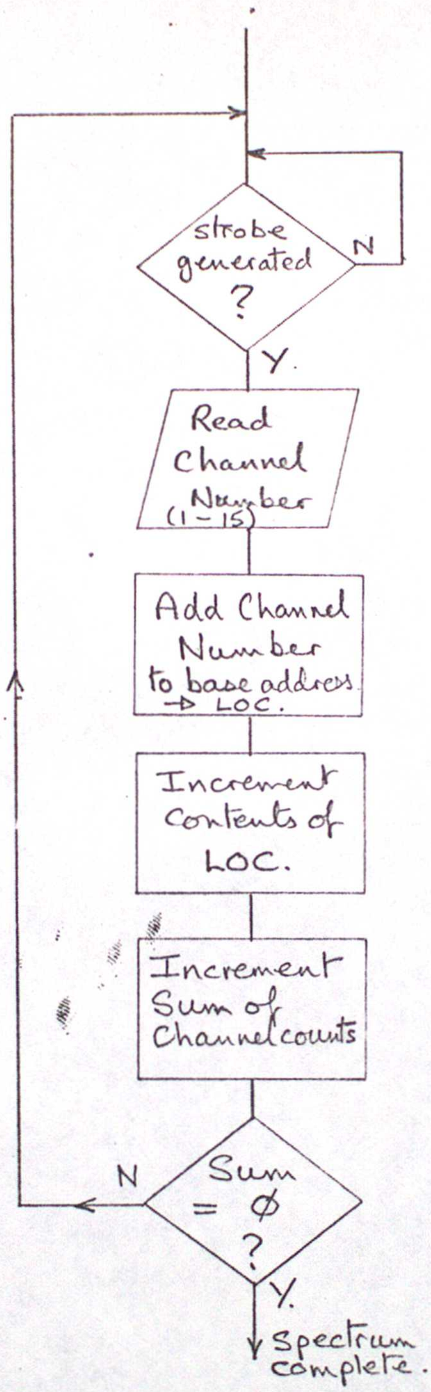
When all bytes stored on tape, prepare to start next spectrum.

ASSP 100. POSSIBLE OPERATING, DISPLAY & STORAGE ROUTINE

Fig 10

STROBE - SERVICING ROUTINE - EXAMPLE.

Fig 11.


strobe generated ?
N
Y.
Read Channel Number (1 - 15)
Add Channel Number to base address → LOC.
Increment Contents of LOC.
Increment Sum of Channel counts
Sum = Ø ?
N
Y.
Spectrum complete.

Processor waits for particle to be sized and strobe generated. Alternative would be to allow strobe to generate priority interrupt.

ASSP 1ØØ presents particle size as 4-bit binary coded channel number in range 1 to 15

Use base address and channel number (x) to form address of location where accumulated channel x counts are stored. (see map below)

Increment channel x count.

Increment sum of channel counts = number of strobes received.

This example assumes completion condition is that channel sum = N.
Program therefore sets channel sum = -N at a start of measuring sequence and then increments it to zero.

Storage of accumulated channel counts.

Fig 12

ADDRESS	CONTENTS.
BASE ADDRESS.	————
BASE + 1	Channel 1 counts.
BASE + 2	Channel 2 counts
BASE + 14	Channel 14 counts
BASE + 15	Channel 15 counts.