DEVELOPMENT AND IMPLEMENTATION OF HARDWARE AND SOFTWARE FOR A MINICOMPUTER BASED PROGRAMMABLE INTERVAL TIMER FOR USE IN POINT PROCESS ANALYSIS OF NEURONAL ACTION POTENTIALS

BY KEITH DAVID WILLENBERG

Submitted to the University of Cape Town in partial fulfilment of the requirements for the degree of Master of Science in Medicine in the field of Biomedical Engineering.

SEPTEMBER 1980

The guidance and encouragement of my thesis supervisor, Dr. Rod Douglas, the unfailing support of my family and friends, the interest and the assistance of the Department of Biomedical Engineering, and the Department of Physiology, and the financial assistance of the Council for Scientific and Industrial Research are greatly acknowledged.

A B S T R A C T

A precision programmable interval timer (PIT) for use in point process analysis of the neutronal action potentials derived from multiple extracellular electrode arrays is described. The PIT is based on a commercially available LSI package, INS8253 and is interfaced to a Data General (DG) microNova 16-bit microcomputer via a DG General Purpose Interface (GPI) board.

The programmable clock periods range from 1 μsec to 10 msec in decade multiples, offering a total timing duration of 65 msec to 650 sec. The PIT operates in two Modes :

1. Count-down timer
2. Elapsed time timer (Event resettable timer)

Clock-overrun and data lost flags are provided.
Testing and applications software have also been developed.

TABLE OF CONTENTS

APPENDICES

A. Medical School Computer System
B. INS8253 Data Sheets
C. micrNova GPI Specifications
D. Testing Software
E. Experimental Software
F. PIT Calibration Results
G. Spike Data Results
H. PIT Circuit Layout
I. Parts Listing & Costing

I N T R O D U C T I O N

Single neurones transmit information over long distances in the nervous system by means of sequences of action potentials (spike trains).
In most instances the spike trains generated by a single neurone have nearly identical waveforms, consequently the time of occurrence of the spikes or the instantaneous rate must carry the neural information.

The measurement of interspike (inter-event) intervals is equivalent to the recording of relative times of occurrence of spike events. Thus point process analysis of neuronal action potential derived from single or multiple extracellular electrode arrays requires precise timing of the intervals between spikes. The electrical interaction between neurones can therefore be investigated by considering their outputs to be point processes.

In the laboratory where this project was executed, the interspike intervals were usually measured by means of software implemented timing routines which depended on a computer to count machine cycles. These timing routines monopolised CPU time and for this reason a programmable interval timer was developed. The employment of intelligent devices such as a PIT improves a computer's overall efficiency by allowing the computer to act as a central controller which has control over, one or more control units (peripherals) in a multi-tasking system. A multi-tasking system allows multiple tasks to be performed by intelligent peripherals which can run by themselves with a minimum of programming.
These peripherals generate interrupts whenever they need to be serviced by the computer.

CHAPTER 1

POINT PROCESS ANALYSIS OF NEURONAL SPIKE TRAINS

The intervals between neuronal action potentials exhibit random variations, which phenomenon is taken to reflect an underlying process with a stochastic character.

A neuronal spike train can be described as a stochastic point process because of the brevity of the action potentials relative to the time interval between them. A typical neuronal spike has duration of 0,5- 2 msec. The maximum repition rate of a spike train depends on the absolute refractory period of the neurone under observation.

The average absolute refractory period is about 1 msec which results in a maximum repitition rate of about 1 kHz. However Hiltz (1965) speaks of a cat's spike train having a bandwidth of 4 kHz.

An idealised representation of a point process is shown in FIG.1.

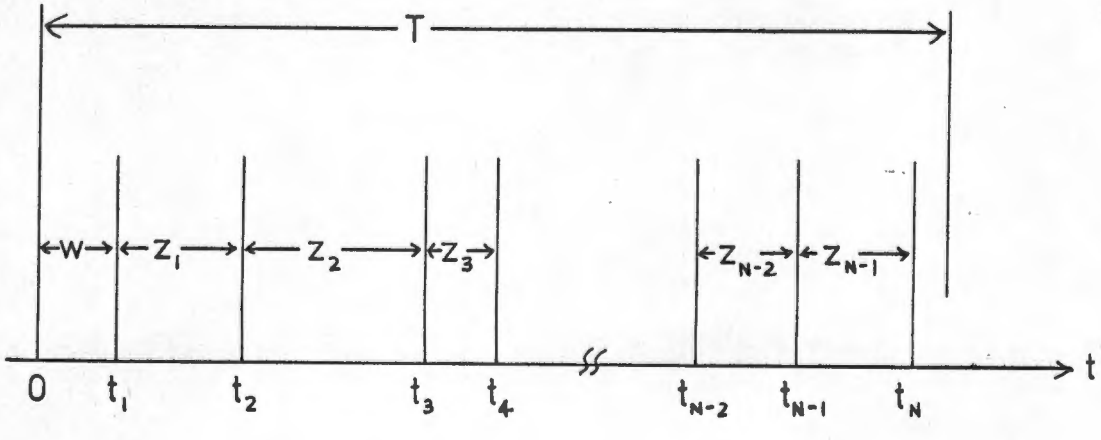


FIG.1. The Event Times and Intervals in a T sec Segment of a Point Process

where T is an epoch during which N spike events occur,

t_i is the absolute occurrence time of the ith spike event

w is the waiting time before the first event

z_i is the interval length between the ith and (i+1)th event

$\{t_i\}$ or $\{z_i\}$ Completely describe the process. (Glaser and Ruchin, 1976).

In order to statistically analyse spike train data it is necessary to measure precisely the time occurrence of each spike. The event times can be recorded as the absolute time relative to some fixed reference time or they can be recorded as the time elapsed since the preceding event. In the latter case, there is less computer memory storage space required since the elapsed time contains fewer significant bits than the absolute time.

The elapsed time or time interval between consecutive events have to be measured with an accurate clock since a noisy timer could generate random intervals in face of a deterministic action potential generator. In this project a PIT is implemented as the accurate real time clock. The interval data obtained from the PIT, $\{z_i\}$, is manipulated in various ways to implement various forms of analysis.

Some forms of analysis performed in this project are:

1) INTERVAL DURATION VS INTERVAL NUMBER GRAPH

The plot of interval duration against interval number is used as an estimate of the mean firing rate of a given spike train, (see FIG.2).

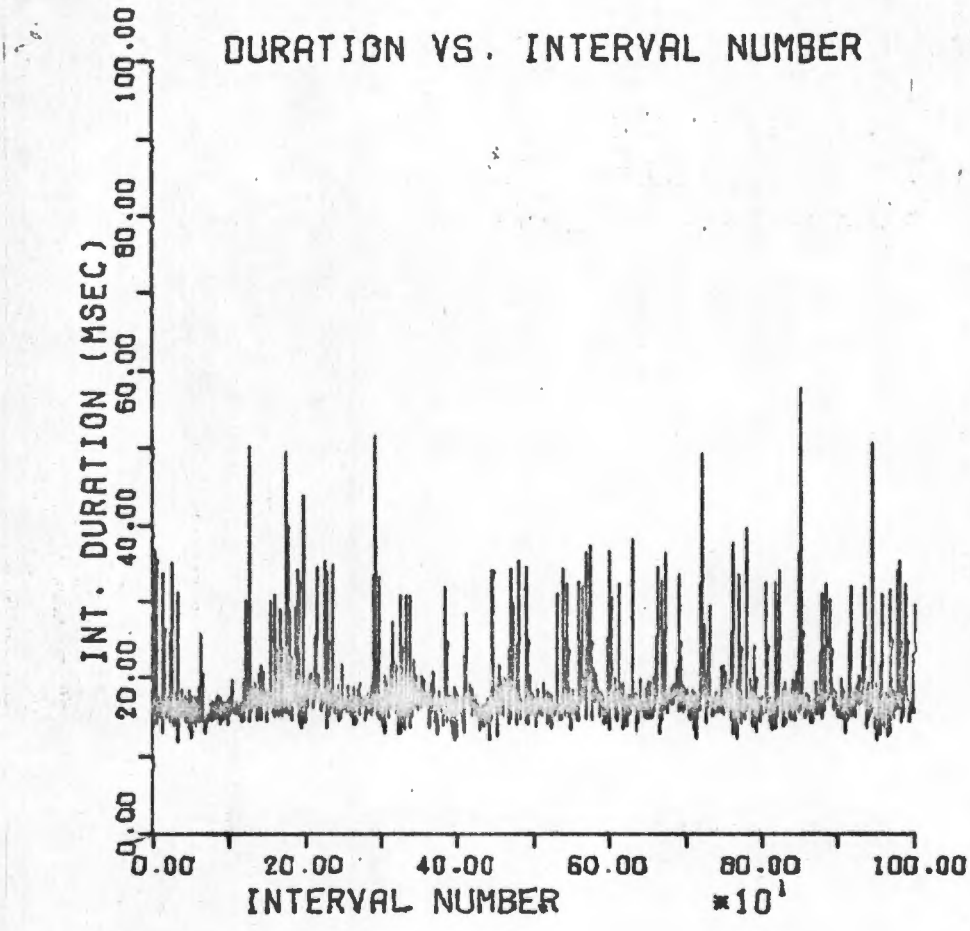


FIG.2. GRAPH OF INTERVAL DURATION VS INTERVAL NUMBER

2. INTERVAL HISTOGRAM

The interval between an event and its immediate successor is defined as a first order interval; the interval between an event and the second one following it, as a second-order interval; and the interval between and event and the nth one following it, as an nth order interval. See FIG.3 for a graphical representation of a first order and higher order intervals. An nth order interval can be descibed as

$$z_i^n = \sum_{k=i}^{i+n-1} z_k \qquad i=1,2,\ldots,N-n+1$$

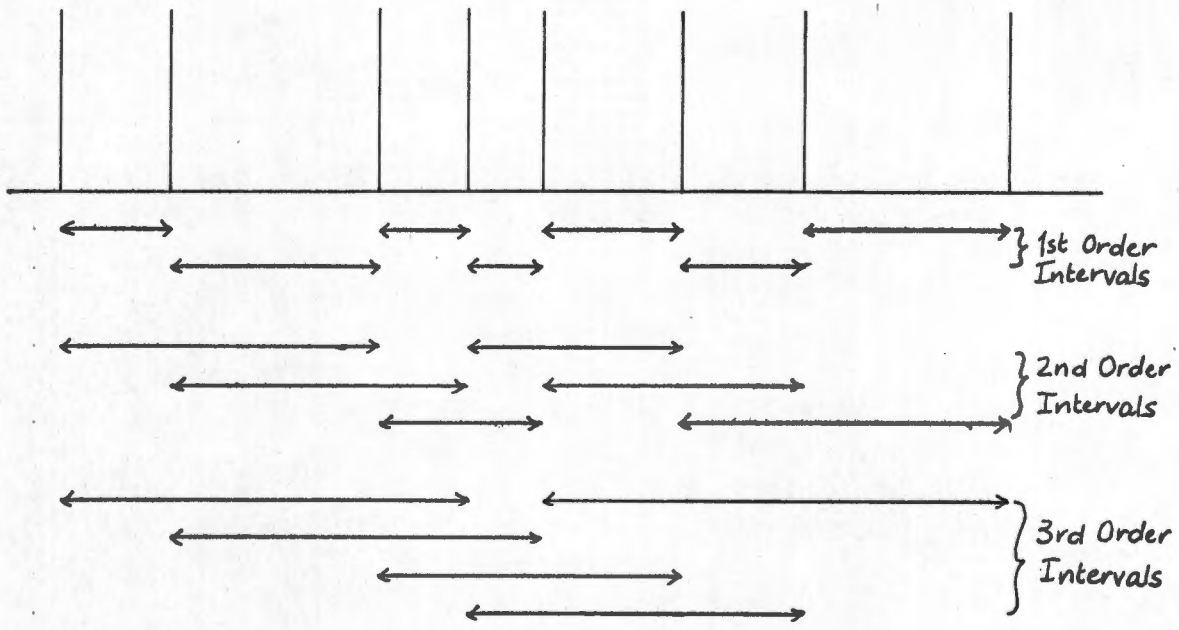


FIG.3. Ordering of time intervals

2(a) FIRST ORDER INTERVAL HISTOGRAM

A sequence of N first order intervals, $\{z_1, z_2, \ldots\ldots\ldots, z_N\}$ obtained in time T, can be compiled into first order interval histogram which is an estimate of the interval distribution. For n=1, the first order interval

$$z_i' = z_i$$

Using the maximum interval length z_{max}, a set of intervals can be partitioned into a specific number of time bins, say B. The length of each bin is

$$\Delta = z_{max} / B$$

An interval z_i falls into bin k if

$$(k-1)\Delta \leqslant z_i < k \cdot \Delta$$

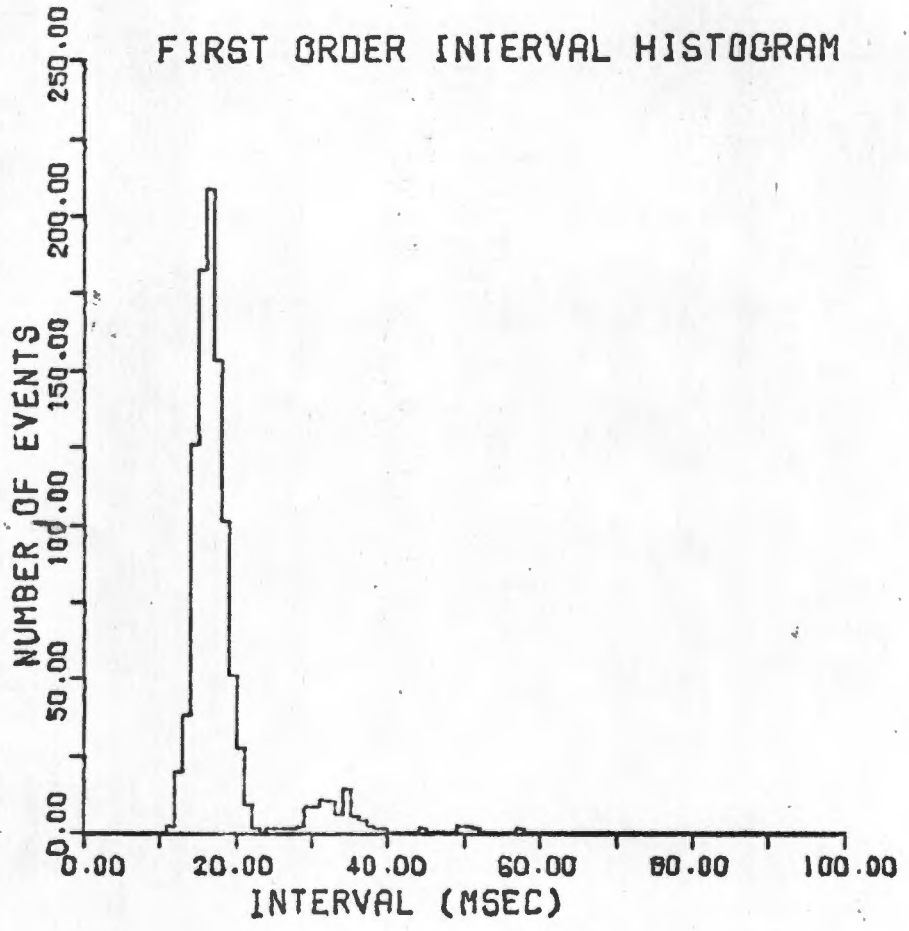


FIG.4 First Order Interval Histogram

The interval falling into the kth bin is assigned the arbitrary value.

$$z_k = (k-\tfrac{1}{2})\Delta \qquad \text{see FIG.4}$$

After sorting N intervals, each bin contains n_k intervals such that

$$N = \sum_{k=1}^{B} n_k$$

2(b) AUTOCORRELATION HISTOGRAM - EXPECTATION DENSITY FUNCTION

The expectation density function specifies the expectation of encountering any event as a function of time after a given event. It is order dependent and involves higher order intervals, eg. 2nd order, 3rd order etc. (See FIG.3.) Interval histograms similar to the first order interval histogram are compiled for each of the sets of higher intervals. Since the autocorrelation is the sum of interval densities of all order, ie $N \rightarrow \infty$, the histograms are summed up and plotted as in FIG.5.

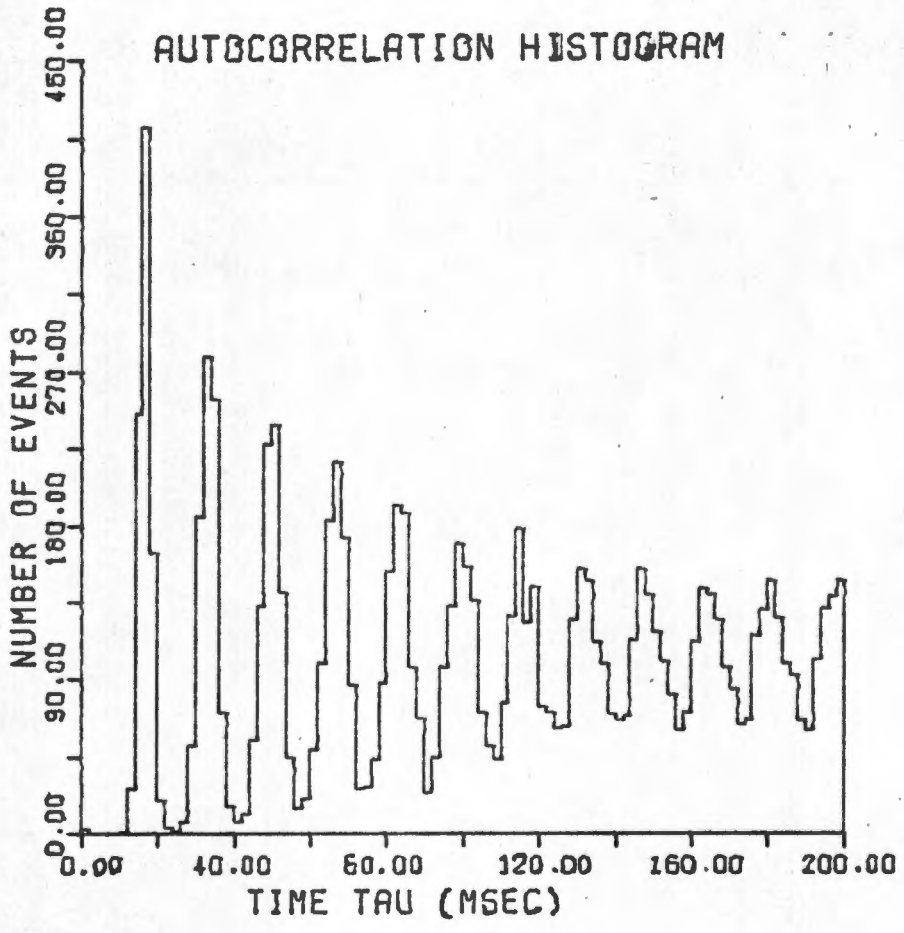


FIG.5. AUTOCORRELATION

The user specifies the maximum range of interval length, t_{obs} which is divided into B bins. The width of each bin is

$$\Delta = T_{obs}/B$$

If the nth order interval z_i^n satisfies the inequality.

$$(k-1)\Delta \leq z_i^n < k.\Delta \quad ,$$

the bin k of the histogram of order n is incremented by one. This histogram is an estimate of the nth interval density.

3. EVENT FREQUENCY VERSUS TIME GRAPH

The graph of event frequency versus time is used to observe trends in the mean firing rate of a neurone over a time period, T_{obs}. The period of observation is divided into 100 time bins. The width of each bin is

$$\Delta = T_{obs}/100$$

The number of events that occur in each bin is counted. This is performed by adding successive intervals, while simultaneously incrementing a particular bin by one for each interval added, until the sum of the intervals exceed the bin width, whereupon the next bin is loaded in the same manner. This is represented graphically in Fig.6.

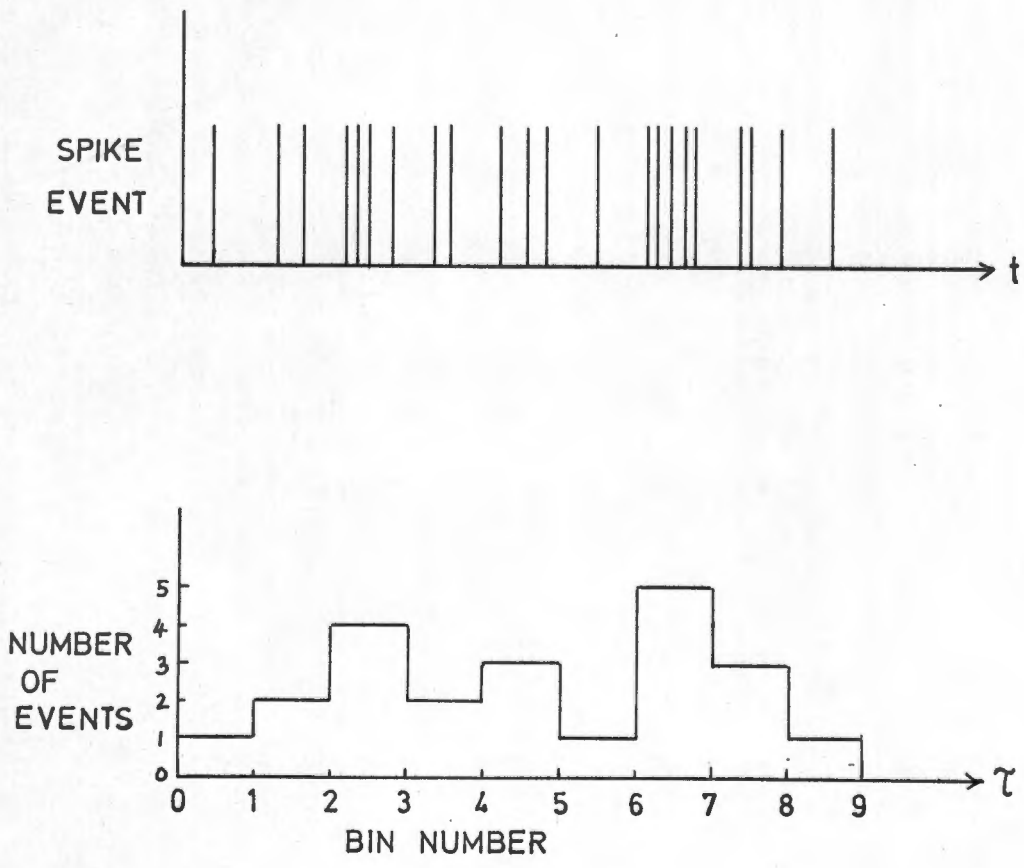


FIG.6.

The number of events in each bin is divided by Δ to give an estimate of the mean frequency for each of the bins. The mean frequency for each bin is then plotted against time (bin number), Fig.7, to display any trends in mean frequency over the observed time period T_{obs}.

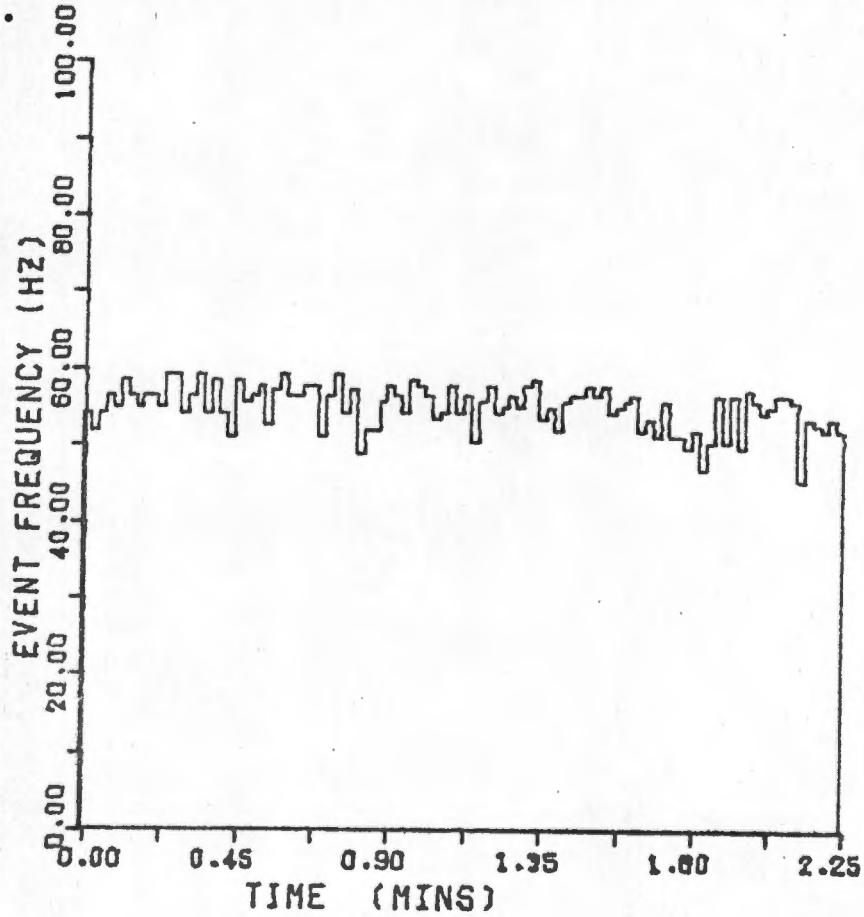


FIG.7. Graph of Event Frequency vs Time

References:

Moore, Perkel and Segundo (1966)

Wyss (1970)

Glaser and Ruchin (1976)

Rhode and Roni (1976)

CHAPTER 2

COMPUTER-BASED LABORATORY FOR NEUROPHYSIOLOGICAL UNIT STUDIES

The laboratory in which this project was executed is almost exclusively concerned with extra-cellular recording and processing of spontaneous and evoked neuronal action potentials from neurons in the brainstem of anaethetised rats.

EXPERIMENTAL SET-UP

Extracellular recording of neuronal action potentials is performed using a microelectrode and high gain amplifier system.
The head of the rat is clamped in a stereotaxic frame and a burr hole, through which the recording microelectrode passes, is made through the skull. The position of the microelectrode, in X, Y and Z coordinates is determined by the stereotaxic frame. By appropriate siting of the electrode according to a stereotaxic atlas, the activity of neurons belonging to a specific population can be observed. A hydraulic microdrive system, which steps the tip of the microelectrode through the brain tissue, allows one to study neurones at various depths in the brain stem.

The experiments are performed in an electromagnetically-screened enclosure (Faraday cage) since any interfering electromagnetic signals such as those caused by unsuppressed motor cars, can corrupt the recording of cellular electrical activity.
Peak amplitudes of neuronal spikes measured using an extracellular electrode are small varying from tens of microvolts to a few millivolts. If the amplitude of a deterministic interfering signal increases relative to the amplitude of the individual spikes, an artifactual component will be induced in the measured spike repitition rate.
However, Johnson (1978) found that random interfering signals do not greatly influence the average intensity (mean firing rate) of a spike train .

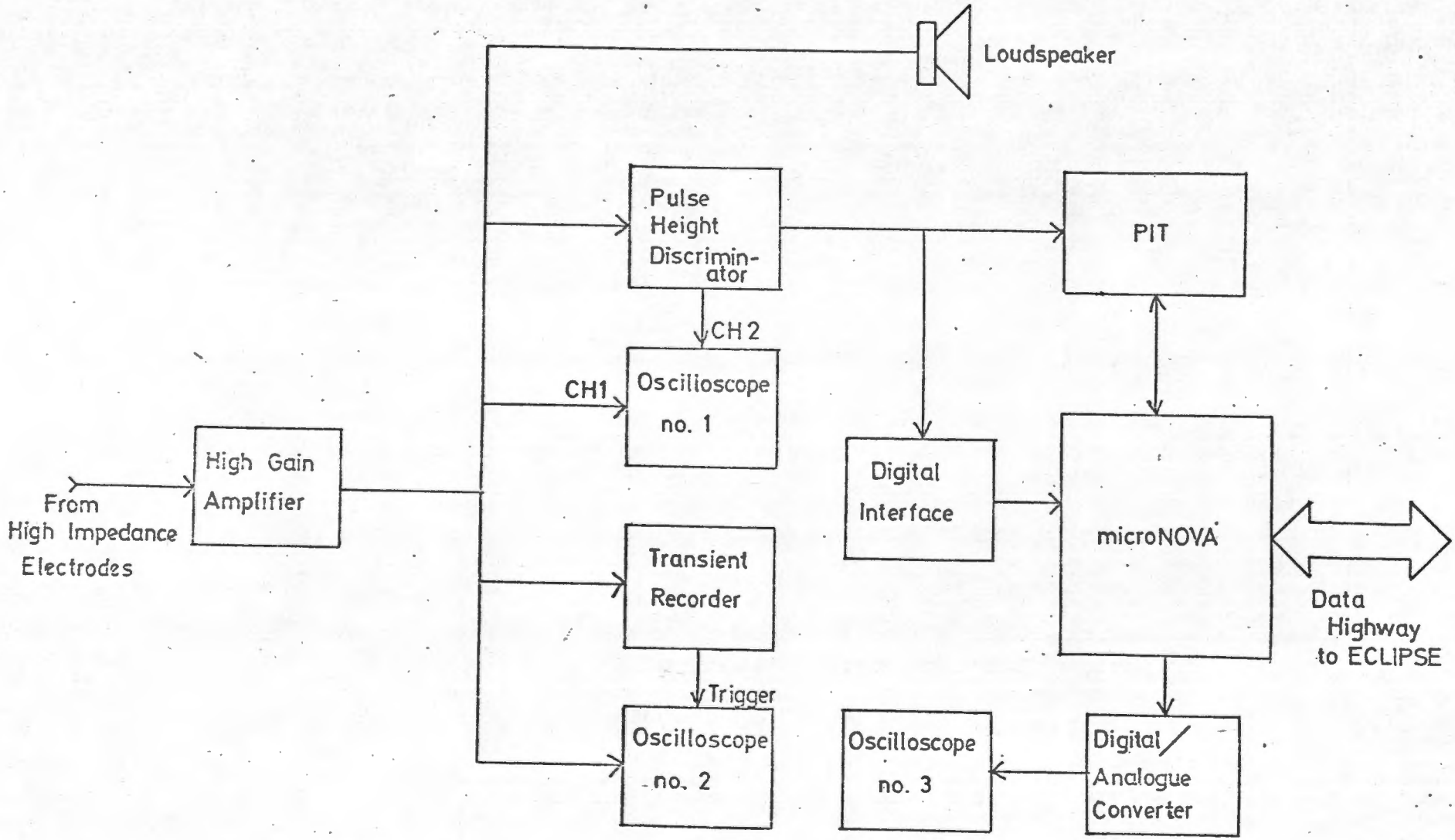


FIG. 8. Apparatus for Spike Data Aquisition

SPIKE DATA ACQUISITION

A schematic representation of the electronic apparatus, which is used in this project for spike data acquisition, is shown in Fig. 8.

The neuronal electrical potentials measured using the microelectrode are amplified by a high input impedance, high gain aplifier and then passed on to the pulse height discriminator, the output of which is a TTL pulse of constant pulse width(0,3 msec). Each output pulse denotes the time at which the spike brain exceeds an arbitrary value, the detection threshold. Oscilloscope 2 which is triggered by the transient recorder displays the transient (spike) which has triggered the TTL pulse. The discriminator level is adjusted until the scope displays an action potential of interest. An actual amplitude vs time recording of a typical extracellular action potential that is normally displayed on the transient triggered scope is shown in Fig. 9.

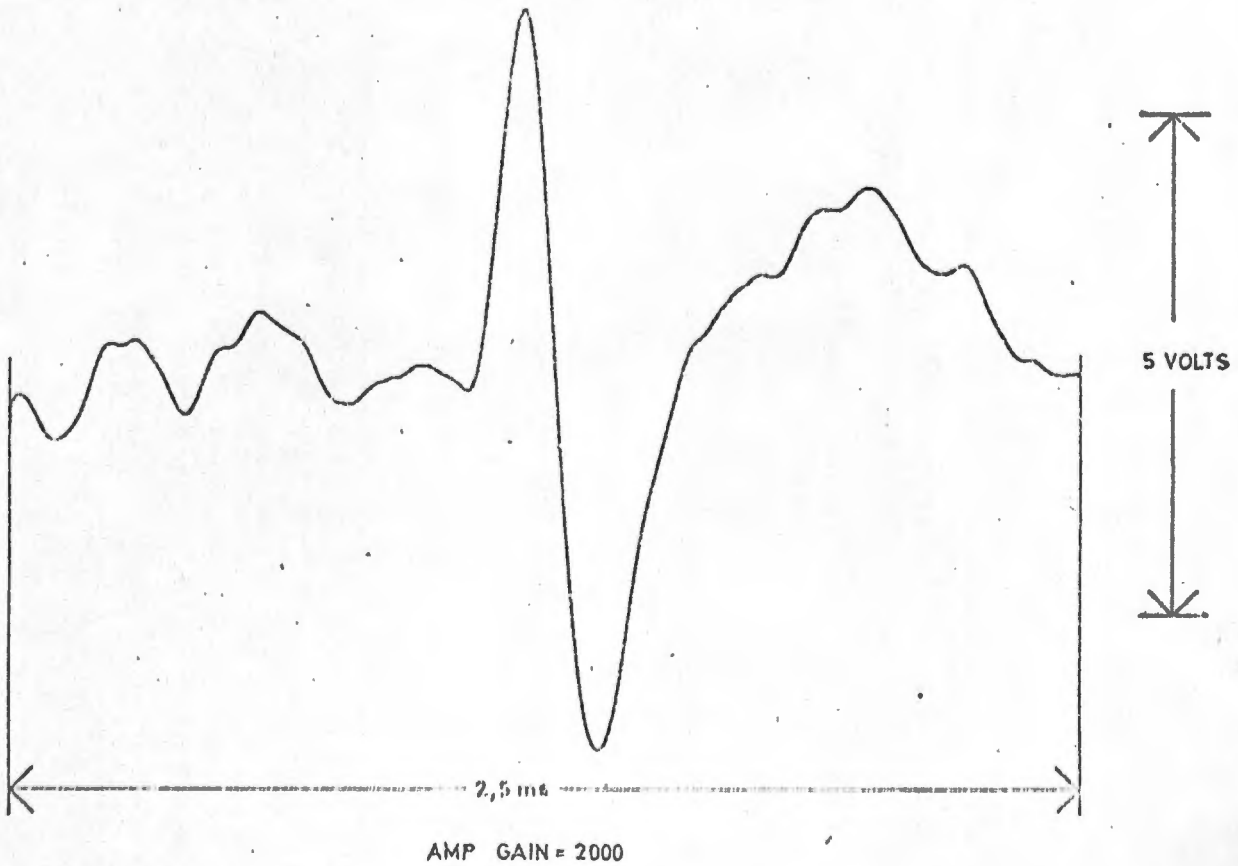


FIG. 9. A Typical Neuronal Action Potential

Amplitude windows achieved by setting up two thresholds on the pulse height discriminator allows various cells in close proximity to the microelectrode to be identified on amplitude criteria. The purpose of the discrimination therefore is to differentiate desired neural discharges from noise and other (smaller) spikes.

The output after pulse height analysis is the stochastic point process consisting of a train of TTL pulses. This train can be subjected to point process analysis directly and is logged by the microNova laboratory computer via its D-D interface.

The D-D interface looks at 16 electrode status lines, each corresponding to an electrode. Unused lines are masked by the computer.
When a spike event occurs on any of the 16 electrode status lines, the current PIT time and the status of the D-D interface are stored by the microNova as a two word couplet in a circular buffer and whenever the Eclipse computer is ready to receive data, data is dispatched from the circular buffer to the Eclipse for processing and analysis.

CHAPTER 3

THE ONLINE MULTIPROGRAMMING COMPUTER SYSTEM

The neurophysiology laboratory and several laboratory experiments in the medical school are interfaced for real time data acquisition and control to a central minicomputer system called the Medical School Computer System (MSCS).
A schematic representation of the computer system is found in Appendix A.

The Data General Eclipse S 130 which is the Central Processing Unit is a 16-bit, floating point processor with 192 K-byte MOS memory. The Advance Operating System (AOS) provides a multi-process environment enabling a number of independent users to operate simultaneously. Memory is expanded by means of 10 M-byte Disc Drive of which 5 M-byte is fixed while the other 5 M-byte is removable. Mass storage and program/data transport is performed on the Standard NRZI 9 track ½" Magnetic Tape Unit. Printing is executed by a Centronix PC101 Dot Matrix Printer. Plotting facilities are available in the Calcomp 565 Digital Incremental Plotter. This is useful for generating interspike interval and other similar histograms. A Tektronix 4025 Graphics Terminal with 16 K graphics is available for use in editing graphs before plotting. An Infoton Keyboard and VDU are used as the laboratory operating console.

The microNova laboratory preprocessor, a 16-bit microprocessor with 24 K-byte main memory, serves as a local intelligence in the laboratory. It is linked to the Eclipse S 130 via an asynchronous 9600 baud line. Real time data acquisition programs are developed on the Eclipse and down line loaded to the microNova. Programming for the microNova is performed in Assembler language on the Eclipse and loaded to the microNova as machine code.

The microNovas act as real time interfaces between the laboratory experiments and the Eclipse and the actual interfacing is implemented using the Data General Model 4210 General Purpose Interface card as is the case for the PIT. A 16 channel 12-bit A/D

converter (30Hz) and a 4 channel 12-bit D/A converter (30Hz) are interfaces situated in the microNova frame.
The microNova laboratory computer system operates under real time disc operating system (RDOS). A microNova debugging routine allows stepwise execution of the program.

THE DATA GENERAL GENERAL PURPOSE INTERFACE

The Data General model 4210 General Purpose Interface (GPI) enables facility in interfacing non-standard equipment to a microNova microcomputer.

The GPI contains an I/O controller (IOC), a function decoder, a data buffer, plus jumpers for device select and polarity select. All data and control lines are brought out to wire wrap pins for user access. Each of the control lines on the I/O Bus is used for a single function, with no additional timing signals needed. Only the control lines corresponding to user implemented functions need to be used.
In this project all the programmed I/O lines except DOC and DIB were used.

The 16 data lines are bidirectional and extend from the IOC in two groups.
One group pass via a 16-bit buffer to wire wrap pins and is normally used for output signals, D (0-15) H, from the CPU to the user device. The other group passes directly to wire wrap pins to be used for input signals, D (0-15) L, from the user device. Input lines are designed to be driven by open-collector TTL drivers.

The I/O transreceiver, IOC, clock driver and I/O control buffer connect the GPI to the I/O Bus. The direction of all information transfers on the I/O Bus is defined relative to the computer. " Output " always refers to moving information from the computer to a controller; " Input " always refers to moving information from a controller to the computer.

In this application of the GPI, status, control and data information are transferred between the computer and the PIT under direct program control, also called " programmed I/O ".

To facilitate programmed I/O, data registers are used to store data as it passes from the computer to a user device or from a user device to the computer. These registers or buffers are needed

because the computer and the device usually operate at different speed. Six programmed I/O instructions allow for 6 device registers to be accessed. Three of these registers are output registers which can be loaded by the computer with either data or control information; the remaining three registers are input registers from which the program can read either data or status information.
The six device registers are :-

Output Register A
Output Register B
Output Register C
Input Register A
Input Register B
Input Register C

The control signals from microNova are asserted low ("1" = HI = 0V).

A summary of the programmed I/O signals each of which can drive up to 10 TTL loads is given below.
Each of these signals is asserted by the IOC when specified by the program.

SIGNAL	FUNCTION
DIA	Load input register A onto data lines
DIB	Load input register B onto data lines
DIC	Load input register C onto data lines
DOA	Load data lines contents into output register A
DOB	Load data lines contents into output register B
DOC	Load data lines contents into output register C
STRT	Start device interface: Set BUSY = 1, DONE = 0
CLR	Clear device interface: Set BUSY = 0, DONE = 0
IOPLS	Initiate a special interface function
IORST	Perform general reset

SET BUSY and SET DONE are signals asserted by the interface to inform the IOC about the status of the interface.

SET BUSY sets the BUSY flag (BUSY = 1) and SET DONE sets the DONE flag (DONE = 1).

A detailed description of these signals as well as a key to the wire wrap pins on the GPI are found in the Appendix. C.

CHAPTER 4

THE PROGRAMMABLE INTERVAL TIMER

A Programmable Interval Timer (PIT) is a real-time clock, which, after being programmed, operates autonomously and generates interrupts to a controlling processor when necessary. A PIT is used for the measurement of the elapsed time between two events or for the temporal control of events.

COMMERCIAL APPROACHES TO PITS

Most commercially available laboratory computers which could be used for data acquisition and processing in the experimental context do not have precision clocks with a time resolution of the order 10 usec or better. A clock with such precision " is more than satisfactory in measuring the event times of the occurrence of neural spikes",(Glaser and Ruchin, 1976).

The scarcity of high resolution real-time clocks compared to the abundance of devices such as A-D converters is due mainly to customer demand. A-D converters come as standard packages and are used extensively in computerisation of analogue instrumentation and in feedback loops for process control. High resolution timers on the other hand are usually only found in neurophysiology and physics laboratories where high event repitition rates are encountered.

Data General, the suppliers of the Medical School Computer System do not provide a PIT as a standard item. They do however provide real-time clocks with a resolution of 1 msec, but this does not fulfil the criterion stated above.

Instead of using a PIT it is possible to implement a real-time processor system. Real-time processor (RTP) systems depend on a computer with high data rates. For optimal process control the processor has to run faster than the process. However, the cost of a RTP system goes up as the data rates increase, since the higher the data rate, the larger the computer.

LSI TIMER PACKAGES

Many electronic component manufacturers offer versatile 8 bit LSI timer packages which can be transformed into working PITs via custom interfacing to any computer system. Motorola offer the MC 6480 Programmable Timer Module while National Semiconductor and Intel produce the 8253 Programmable Interval Timer package. Each of these packages mentioned contains three independent 16-bit interval timers, each of which can be operated in a different mode. Interstil offer a range of timers with an accompanying handbook of potential applications. Most of these LSI timers in general are for the 8-bit microprocessor systems, each one for a particular system eg. the Motorola MC 6480 for the MC 6800 microprocessor (MPU) system and the 8253 for the 8080 MPU system.

LITERATURE APPROACHES TO TIMERS FOR USE IN PHYSIOLOGICAL SIGNAL ANALYSIS

In the last decade very few timer devices, which resemble a PIT, are found in both the IEEE Transactions on Biomedical Engineering and the Journal of Applied Physiology.

In the former, three timing systems are found. Two of these timers, one by Silverman and Eisenberg(1971) and the other by Nolte and Tarby (1977), are basically pulse generating systems. These devices can be used for programming sequences of events such as oscilloscope sweeps and stimuli in physiological and other experiments. They cannot however be used for interval measurements without implementing additional circuitry.

The other timing system which appeared in the IEEE Transactions on Biomedical Engineering is a real-time cross correlator by Arnett and Ellert(1976). This is a hardware device for use in neuronal spike train analysis which is faster than a laboratory computer in analysing spike trains. Circuits for compiling post-stimulus time and interval histograms are included. Although the device described is faster than a laboratory computer, it cannot perform as many types of analysis as the computer can on the acquired data.

In the Journal of Applied Physiology three timing devices are found in journals covering the last ten years. A presettable multichannel digital timer is described by Sabah(1975). This device only serves as a variable frequency generator with a pulse delay selector for monostable operation.

The two other timing devices which appeared in the Journal of Applied Physiology are a "digital display of neuronal average firing frequencies" (Sabah,1976) and a "high accuracy linear rate meter"(Wyss,1975).
The former device counts the number of spike events during fixed time intervals of 1,2,5,10,20 and 50 sec using a 555 timer to generate the fixed time intervals.

The latter device is used to measure the instantaneous occurrence of events. Both timing systems are capable only of performing basic first order analysis of a spike train and demonstrates the need for a computer-based timer if high levels of analysis are required.

Many researchers using computer based methods for point process analysis of neutronal spike trains make use of real-time clocks that are built into computer peripherals.

Wyss (1970) used a versatile laboratory peripheral which included, amongst other features, Schmitt trigger inputs, multiplexed A-D converters and a variable RC-clock. The RC-clock was calibrated manually by adjusting the clock control potentiometers and observing the square-wave output on a CRT oscilloscope. RC-clock cycles are counted by software and this is one of the features that this project is to replace. The RC-clock which is adjusted manually, can be replaced by a crystal controlled clock and a frequency divider circuit, this will eliminate any human errors incurred in adjusting the RC-clock frequency.

In his later work, Wyss with Handwerker (1971) used the built-in real-time clock of a PDP-12 computer. When a spike event occurred, the peak amplitude of the spike and the double-precision clock-time were recorded.
Interval times were calculated by the computer program. This method, where the event time is stored as a double-precision word uses computer memory storage space very inefficiently. Storage space could be utilised more efficiently if interval times were measured by hardware and then stored as a single precision word.

Sanderson and Kobler (1976) describes a computer-based " time interval digitiser " (TIDIG) which is basically a programmable crystal controlled clock capable of generating pulses with a clock period ranging from 20 μsec - 4096 msec. The output

of the clock has a pulse width of 1000 , 100 or 10 usec. These pulse are used to decrement a register in the TIDIG. When the register has been decremented to zero, a pulse is generated as the TIDIG output. From this point on, the timer operates as a software package. Each TIDIG output pulse interrupts the computer and an interrupt service routine increments a computer memory word A. Word A thus serves as a counter of clock pulses. When a data interrupt (spike event) occurs, word A is stored as the elapsed time or interspike interval. Word A is then reset to zero.

The TIDIG is in fact a hardware/software package which is totally dependent on a PDP-8 computer. In addition to the PDP-8 computer a small special purpose computer was used to process the acquired time-interval data. This TIDIG routine is an inefficient method of computer usage. The TIDIG could be replaced by an "intelligent" programmable interval timer, allowing the CPU to service other routines whenever required to do so.

A "Pulse Interval Timer" is described by Brown, et al (1976). This device is capable of digitising intervals between input events (TTL pulses on any of 6 input channels) with a resolution of 10 usec. The device operates in programmed I/O or direct memory access (DMA) mode. Each output word from the timer is a couplet of two 12-bit PDP-12 words.
The output word contains channel code as well as time interval code. The timing circuitry employs five 4-bit binary counters which are used to produce the 18-bit interval code. Application-specific interfacing circuitry had to be designed to implement the pulse interval timer on the PDP-12 system (no general purpose interface board was used).

Although the design of the pulse interval timer is very flexible, the control circuitry is designed specifically for the PDP-12. The basic components of the actual interval timer design, ie. clock frequency divider and counters are similar to those implemented in the PIT described in this thesis.

Caron and Herzog(1977) developed a programmable timer similar to those of Sabah(1975), Silverman and Eisenberg(1971), Nolte and Tarby(1977). The microprogrammable unit, which is used in the control of a pulsed Fourier Transform Nuclear Magnetic Resonance spectrometer, employs a Mostek MK 5009 programmable frequency divider to generate various pulse rates. The rest of the timer design,ie. counters,etc. employs standard TTL.
By comparison the frequency divider used for the PIT in this thesis uses standard TTL while the counters are part of a LSI package.

The majority of timers are designed and built to provide exactly the features required, ie. they are application-specific.

P I T S P E C I F I C A T I O N S

Given a conventional interface structure consisting of :-

a) flags (eg. busy, done) for indicating to the CPU the current status of the interface.

b) a means of generating a CPU interrupt when certain interface criteria are met (eg. done flag asserted).

c) interface data registers to facilitate data transfer between the CPU and the interface.

d) control lines for implementing certain interface functions.

The following characteristics are to be implemented.

1. MODE 1: Event-resettable (Elapsed-time) Timer

1.1 The PIT counts clock pulses until an incoming action potential causes the PIT to

i) latch its current time
ii) reset and commence a new timing cycle
iii) assert an interrupt status flag and a done flag

1.2 If the PIT clock overruns, a clock status flag is set as well as a done flag to inform CPU to compensate for the overrun. The PIT is auto-restarted.

2. MODE 2: Count-down timer

The PIT is loaded with an initial count and decremented on each clock pulse until the count reaches zero. Zero count asserts the done flag. The whole sequence is automatically repeated, ie without computer intervention.

During this mode the PIT can be pulsed to read the current time value.
The PIT clock rate is program selectable. Selectable clock rates are :-

1 µsec, 10 µsec, 100 µsec, 2 msec and 10 msec

Clock-overrun and lost-data flags are provided. The lost data flag is set if an event occurs before previously latched event-time was read from the PIT by the computer.

CHAPTER 5

THE INS 8253 PROGRAMMABLE INTERVAL TIMER

The INS 8253 package was used as the basis for the design for the PIT because, amongst other reasons, there was a time limit to the project. Most LSI timer packages mentioned earlier were unavailable for immediate use as they would have had to be ordered. Thus would have been resulted in a delay of anything from two weeks to a month.

The documentation for the INS 8253 timer was clear and precise compared to that of the Motorola MC 6480 which was the only other LSI timer available in Cape Town at the time. Although the MC 6480 timer can perform the functions of INS 8253, it is designed specifically for the M 6800 Bus. Interfacing the MC 6480 to the microNova general purpose interface would have required more work than for the INS 8253 because of extra control lines. The 8253 has only five control lines compared to the MC 6480 which has 9 control lines connecting the device to the servicing bus. Besides the manufacturers data sheets additional information on the INS 8253 timer was provided in an excellent series of articles by De Jong et al (1978) in which he introduced the INS 8253 and mentioned some of its potential applications.

GENERAL DESCRIPTION OF THE INS 8253

The INS 8253 chip contains 3 independent 16-bit down counters, each of which is capable of independent count rates of DC - 2 MHz. In this application only one of these counters, counter # 0 is used. A schematic for a typical counter in the chip is given in Fig. 10.

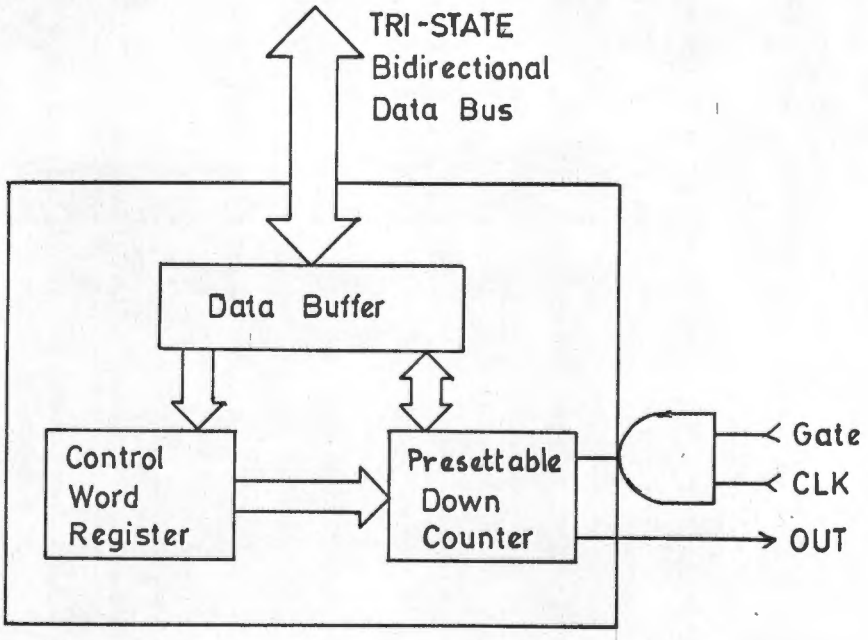


FIG.10 Functional Diagram of each of the three 16-bit interval timers in the 8253 chip (De Jong et al, 1978)

Two address lines, A0 and A1, select which of the internal registers, ie control register or counter, is connected to the data bins. Selecting counter #0 requires the setting A0 = 0 , A1 = 0 while selecting the control word register requires A0 = 1, A1 = 1.

Since only counter #0 is used, these two lines, A0 and A1 can be tied together to form one control line which we shall call A0A1.

Control inputs $\overline{WR}$ and $\overline{RD}$ determine whether WRITE (to the chip) or READ (from the chip) functions are implied respectively. It is not possible to read the contents of the control register.

TABLE 1: CONTROL WORD FUNCTIONS

8253 Data Bus BIT	FUNCTION
D7 D6	Selects one of three 16-bit counters
D5 D4	Selects READ/WRITE Format
D3 D2 D1	Selects 8253 Operation Mode
D0	Selects count sequence BCD/Binary

Since only counter #0 is used, bits D7 and D6 are always zeroes.

For maximum counting range the full 16-bits of the timer are used so bits D5 and D4 are both 1's in the initialisation control word. This bit setting allows the sequential loading or reading of the LS byte first, then the MS byte of the selected counter.

Each timer has 6 modes of operation. These modes are specified by bits D3, D2 and D1.

MODE 0 : Timed interrupt. Out goes high on zero count
MODE 1 : Retriggerable variable width one shot.
MODE 2 : Programmable rate generator.
MODE 3 : Programmable square wave generator.
MODE 4 : Delayed strobe.
MODE 5 : Hardware triggered strobe

A more detailed description of the control word and the operating modes can be found in the series by De Jong (1978) and in the data sheets on the INS 8253 which have been included for the readers convenience. (see Appendix B)

Of the 8253 operating modes, MODE 1 and MODE 2 suits the characteristics of the required PIT best.

In the operation of MODE 1 and MODE 2 the INS 8253 control lines OUT and GATE are significant. If the gate line GATE, is active, negative transitions at the clock input decrement the counter. When the counter reaches zero, OUT becomes active, its actual behaviour depending upon the mode programmed into control register.

In the elapsed-time timer mode with the 8253 timer operating in MODE 1, OUT goes low on the first clock pulse after a rising transistion on the GATE input. OUT goes high again on the terminal count and remains high until the next positive transition on the GATE input. The positive transition of OUT on the terminal count can be used to set a clock overrun flag.

In the Count-down timer Mode with the 8253 operating in MODE 2, OUT goes low for one clock cycle at the end of each Count sequence. The counter repeats Count sequences as long as the GATE input is held high. Any positive transition on the GATE input will start a new count sequence.

CHAPTER 6

PIT STRUCTURE AND PRINCIPLES OF OPERATION

The specifications for the required PIT were implemented in a design based on the INS 8253 programmable interval timer.

1. COMPOSITION OF APPARATUS

The PIT hardware consists of the following components :-

1) Device I/O Controller (IOC)
2) A 1MHz oscillator with frequency divider which divides the 1MHz signal down to 100 Hz in decade steps
3) Two 16-bit input registers
   a) Mode (Control) Register
   b) Input Data Register
4) The INS 8253 programmable interval timer
5) Command Interpreter
6) Timing waveform Generator
7) PIT gating (trigger) circuit
8) Two 16-bit output Registers
   a) Status Register
   b) Output Data Register
9) Interrupt circuitry

A functional diagram of the PIT system is given in FIG.11.

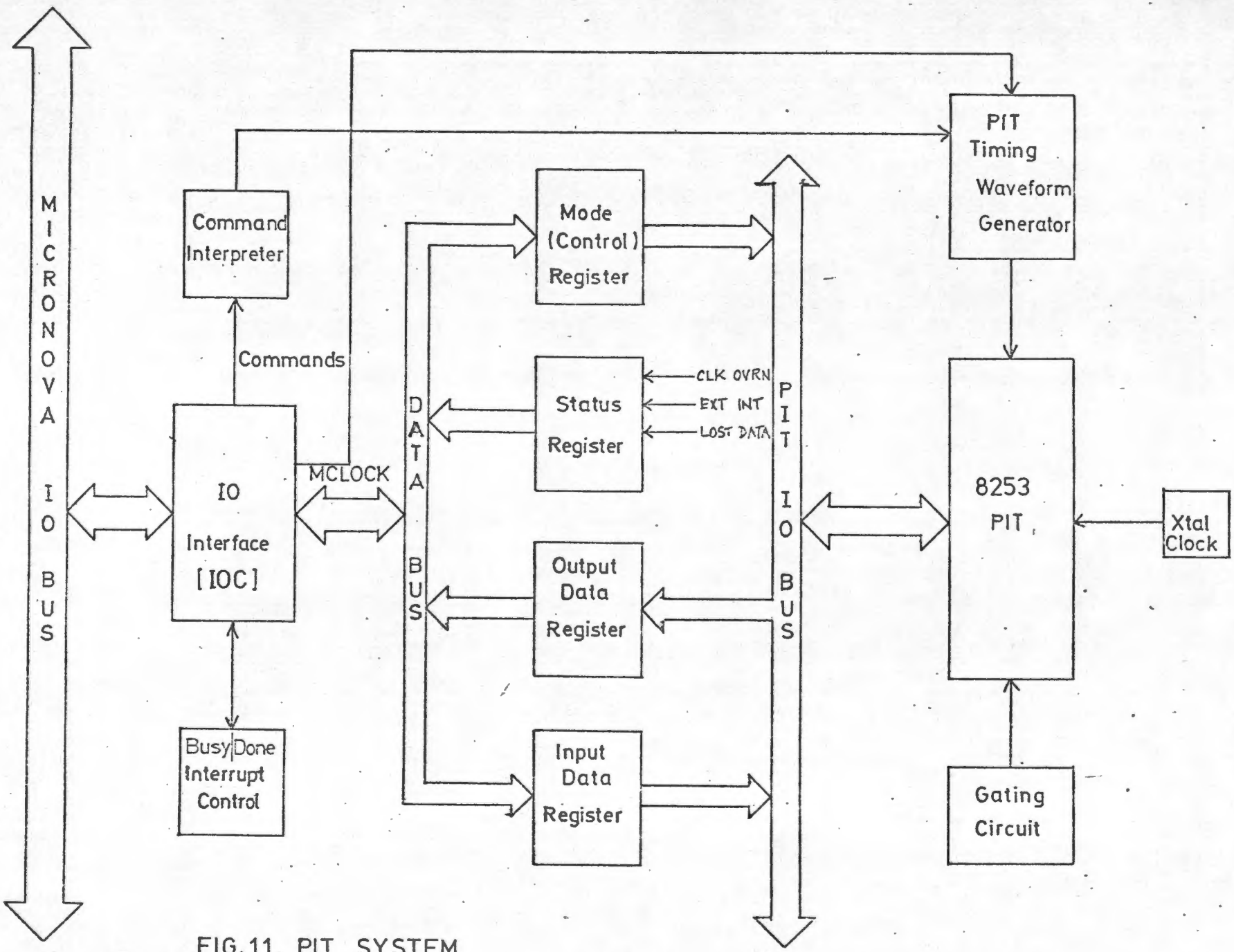


FIG.11. PIT SYSTEM

1.1 THE DEVICE I/O CONTROLLER

The IOC transfers data from the MicroNova I/O bus to the interface data bus and vice versa. Interface registers are used to facilitate data transfers between the interface data bus and device (PIT) data bus. These interface registers are shown in FIG.11 as the Mode (Control) Register, the Data Register, the Status Register and the Output Data Register. In the discussions that follow, these registers may also be called the device's Output Register A, Output Register B, Input Register A and Input Register C respectively, according to the date General convention. An output buffer/register is one into which a specified accumulator's contents are loaded, and an input buffer/register is one whose contents are placed in a specified accumulator according to a specific I/O instruction.

The IOC is also responsible for generating control signals which could be used to control the interface registers and to generate device control functions such as " READ " and " WRITE ". When using the 8-bit INS 8253 timer for example, these IOC generated control signals cannot be used directly, since the timing specifications of GPI IOC control signals do not meet those required for the 8253 timer.
The 8253 timer is designed specifically for the 8080 microprocessor system and requires a minimum READ or WRITE pulse width of 400nsec. By comparison the pulse width of the I/O control signals from the microNova GPI is 240nsec.

1.2 COMMAND INTERPRETER AND TIMING WAVEFORM GENERATOR

The problems experienced with timing are overcome by the implementation of the Command Interpreter and the Timing waveform generator.
The Command Interpreter decides what timing waveform should be generated when specific I/O control signals are asserted. The generated waveforms are typical 8253 input waveforms which meet the manufacturer's specifications.

1.3 FUNCTION OF DEVICE I/O REGISTERS

1.3.1 The Mode Register is loaded with the specified control word by a programmed I/O control signal. The LS 8-bit byte of this control word, the PIT CONTROL WORD is then loaded into the 8253 chip a timing waveform, consisting of a single WRITE pulse, which is specified by the Command Interpreter.

1.3.2 The Input Data Register is loaded with the required clock run-time (initial count) by a programmed I/O control signal. The 16-bit word is then written into the 8253 chip as two sequential 8-bit words. This data transfer requires two WRITE pulses which constitute the timing waveform specified by the Command Interpreter.

1.3.3 The 13 LS bits of the Status Register which are the same as the 13 LS bits of the Mode Register are loaded by the same signal which loads the Mode Register. These bits are used for checking the loaded control word. The 3 MS bits are PIT interface Status flags, viz.Clock Overrun, External Interrupt and Lost Data flags. When this register is read by the computer, ie, a conventional I/O control signal loads the contents of the register onto the microNova I/O Bus, the status flags and the control word can be tested.

1.3.4 The Output Data Register contains the latched clock time which can be read by the computer via a single I/O control signal.

1.4. GATING CIRCUIT

The gating circuit, which is used to start the PIT, is used according to the 8253 manufacturers' specifications. In the 8253 timer Mode 1, the output of the gating circuit is a pulse which starts the timer while in Mode2, the output of the gating circuit goes high to start the timer and is held high as long as counting is required.

1.5. INTERRUPT CIRCUIT

The interrupt circuit is a BUSY/DONE circuit which notifies the CPU about the status of the interface. The DONE flag is asserted whenever the PIT has a clock overrun or an event occurs. Setting the DONE flag interrupts the CPU, thus allowing the computer to read the latched PIT clock time.

2. PRINCIPLES OF OPERATION

The PIT is initialised by loading it with a 16-bit SETUP word which selects both the mode of operation and the PIT clock rate. Special functions that can be performed in each mode are dependent on the SETUP word.

2.1.1. MODE 1 : Event resettable (Elapsed time) Timer

i) The 8253 timer is initialised for MODE 2 operation by issuing it with the respective PIT CONTROL WORD, the LS byte of the SETUP word.

ii) Counter #0 is loaded with the initial counter value, ie the contents of the Input Data Register.

iii) The PIT is started by a pulse from the gating circuit and counter # 0 is then decremented with every clock pulse until an EXTERNAL INTERRUPT (spike event) occurs.

iv) The interface responds by :-

a) Setting the EXTERNAL INTERRUPT flag in the Status Register

b) Asserting the DONE flag and thus interrupting the CPU

c) Latching the current PIT clock time in the Output Data Register

d) Auto-restarting the timer by automatically reloading counter # 0 with the initial count and pulsing the gating trigger input of the 8253 timer

If the PIT clock overruns (counter reaches zero before the next EXTERNAL INTERRUPT

a) Set the CLOCK OVERRUN Status flag in the Status Register

b) Set the DONE flag to interrupt the CPU to notify it about the clock overrun

c) Auto-restart the timer by automatically reloading counter # 0 with the initial count and pulsing the gating (trigger) input of the 8253 timer.

2.1.2. MODE 2: Count down Timer

i) The 8253 timer is initialised for Mode 2 operation by issuing it with the respective PIT CONTROL WORD, the LS byte of the PIT SETUP word, from the Mode Register.

ii) Counter # 0 is loaded with the initial count which is the contents of the Input Data Register.

iii) The PIT is started by a step input to the gating input of counter # 0, GATE 0 and counter # 0 is then decremented with every clock pulse until the counter reaches the end of count.

Then:

a) The DONE flag is asserted to interrupt the CPU causing it to jump to a specified routine.

b) The timer is restarted automatically as long as the gating input of counter # 0, GATE 0, is held high.

The current of time the PIT can be read at any time by pulsing the PIT with a computer generated pulse, IOPLS. This causes the current time to be latched in the Output Data Register without asserting the DONE flag.
The clock is unaffected by the pulse and continues running.
In MODE 1, IOPLS has no effect on the PIT since it is hardware selected as a function in MODE 2 only.

2.2 CLOCK INTERVAL

The clock rate is software selectable. Selectable time-bases are:

1 µsec, 10 µsec, 100 µsec, 1 msec and 10 msec

These give maximum clock run-times of

65536 µsec, 655360 µsec, 6553600 µsec
65536 msec, 655360 msec and 6553600 msec respectively.

The clock rate is specified in the PIT SETUP word.

2.3. REGISTERS

The I/O registers A, B and C exist in the PIT. They have however been renamed in the functional diagram of the PIT system and in this text. The original registers (according to Data General convention) with their new names are listed below.

Output Register A	-	Mode (Control) Register
Output Register B	-	Input Data Register
Input Register A	-	Status Register
Input Register C	-	Output Data Register

2.3.1. OUTPUT REGISTER A - MODE (CONTROL) REGISTER

This register handles the control data, ie the PIT SETUP word, that is received from the microNova for the PIT.

A generalised SETUP word format is given in the following table.

TABLE 2. SETUP WORD AND ITS FUNCTIONS

DG GPIO BUS BIT	8253 DATA BUS BIT	FUNCTION
0	———	———
1	———	———
2	———	———
3	———	M0 } MODE
4	———	M1 } SELECT
5	———	T0 } TIME-BASE
6	———	T1 } SELECT
7	———	T2 }
8	D7	8253
9	D6	PIT
10	D5	CONTROL
11	D4	WORD
12	D3	
13	D2	
14	D1	
15	D0	

Note that the 8253 data lines are numbered D0 - D7 where the D7 is the most significant bit. By contrast, the Data General bit numbering convention has bit 0 as the most significant bit. Thus the 8253 control word received from the microNova, is found in bits 8-15 with bit 8 as the most significant bit.

Since only counter # 0 is used, the bits 8 and 9 in the microNova output to the Mode Register are always zeroes.

TABLE 3 .TIME-BASE SELECT

TIME-BASE	BIT 5= T0	BIT 6= T1	BIT 7= T2
1 µsec	0	0	0
10 µsec	0	0	1
100 µsec	0	1	0
1 msec	0	1	1
10 msec	1	0	0

This time-base table can easily be extended to 100 msec and 1 sec without too much trouble, since using 3 bits to define a time scale allows for 7 different time bases and only the first 5 are presently being utilised. The addition of these extra two time-bases would only require the installation of two decade counters in the clock circuitry.

TABLE 4. MODE SELECT

MODE	BIT 3= M0	BIT 4 = M1
1	0	1
2	1	0

2.3.2. INPUT REGISTER - STATUS REGISTER

This register has generally the same format as the Mode Register has for the SETUP word. The only difference in the format, is the use of Bits 0, 1 and 2 as status flags, see TABLE that follows.

TABLE 5. STATUS REGISTER FORMAT

BIT	FUNCTION	
0	Clock overrun flag	Status flags
1	External interrupt flag	
2	Lost Data flag	
3	Mode	
4	Select	
5	Time-base	
6	Select	
7		
8		
9		
10	PIT	
11	CONTROL	
12	WORD	
13		
14		
15		

The order of the status flags facilitate interrogation of the status register. The method of status interrogation is discussed later under PIT software.

2.3.3. OUTPUT REGISTER B - INPUT DATA REGISTER

This 16-bit register stores the initial count value, TIME, it receives from the microNova for loading to the PIT. The clock may be programmed to run for any length of time from 1 μsec to 655360 msec.

2.3.4. INPUT REGISTER C - OUTPUT DATA REGISTER

The current PIT value is latched in the 16-bit Output data register for transferance to the microNova.

Care should be taken when loading the initial count value. Although the 8253 timer accepts a 16-bit number as an unsigned number, bit 0 is used by the microNova as a sign bit.

This will cause complications if the initial count value has bit 0 = 1 and is subsequently used in an arithmetic operation.

NOTE: The Input Register B and the Output Register C are not used, so DIB and DOC are not valid for the PIT, ie these control lines are not used.

2.4. ERROR CONDITIONS

These error conditions which are only relevant in MODE 1, the Event resettable Mode are the "clock overrun" error and the " event overrun " error.

2.4.1. "CLOCK OVERRUN ERROR"

This error condition arises if the terminal count is reached before a spike event occurs. The interface responds by :

a) Setting the clock overrun Status flag.
b) Auto-restarting the PIT by automatically reloading Counter # 0 and retriggering the timer.
c) Asserting the DONE flag thus interrupting the CPU.

2.4.2. "EVENT-OVERRUN" (LOST DATA) ERROR

This error condition arises if a spike event occurs while the EXTERNAL INTERRUPT flag is still set from a previous spike event, ie the latched value was not read yet. The previously latched value will be lost and replaced by the new elapsed time. The LOST DATA flag is set by the second spike event and the timer is auto-restarted.

2.5. CLEARING, INITIALISING, STARTING AND READING THE PIT

2.5.1. CLEAR or IORST causes a general reset.
This results in BUSY = 0 and DONE = 0 flag settings.

2.5.2. The PIT is initialised by loading the Mode Register with a SETUP word via a DOA instruction.

The counter then receives its initial count value in the Input Data Register via a DOB instruction. A new count value can be loaded at any time after the PIT has been initialised.

2.5.3. START sets BUSY = 1 and DONE = 0.
The first START pulse after a CLEAR or IORST activates the PIT. Once started the PIT does an auto-restart every time it reaches its terminal count or receives an EXTERNAL INTERRUPT.
The restart after an interrupt is performed a soon as the current time of the PIT is latched in the Output Data Register.

One of the reasons for using the INS 8253, that was not mentioned before, is that the automatic reloading of the counter(# 0) does not require computer intervention. No computer - PIT data transfer is needed since a positive transition on the GATE input resets the counter to its originally loaded value. This cuts down in the number of computer instructions in the servicing routine and thus the computer servicing time.

All subsequent START pulses after a DONE = 1 state, will have no effect on the timer. These subsequent START pulses do however reset the BUSY and DONE flags (BUSY =1, DONE =0) since they are part of the microNova GPI

2.5.4. READING THE PIT

a) Status Register

Reading the Status Register of the PIT by computer requires a DIA instruction. This is useful for testing purposes and is included in the testing software which will be discussed later.

b) Output Data Register

In order to read the PIT " on the fly ", ie not to interfere with the counting cycle, it has to be issued with a special control word, ie the FLY word. via a DOA instruction.

The current PIT value is latched by an EXTERNAL INTERRUPT or a NIOP instruction.
The latched time is then read by the microNova via a DIC instruction which transfers the latched time from the PIT to a specified accumulator. The time that is read is not the elapsed time but the initial count value decremented by a number of clock cycles.

3. DESIGN OF PIT COMPONENTS

3.1. DEVICE I/O REGISTERS

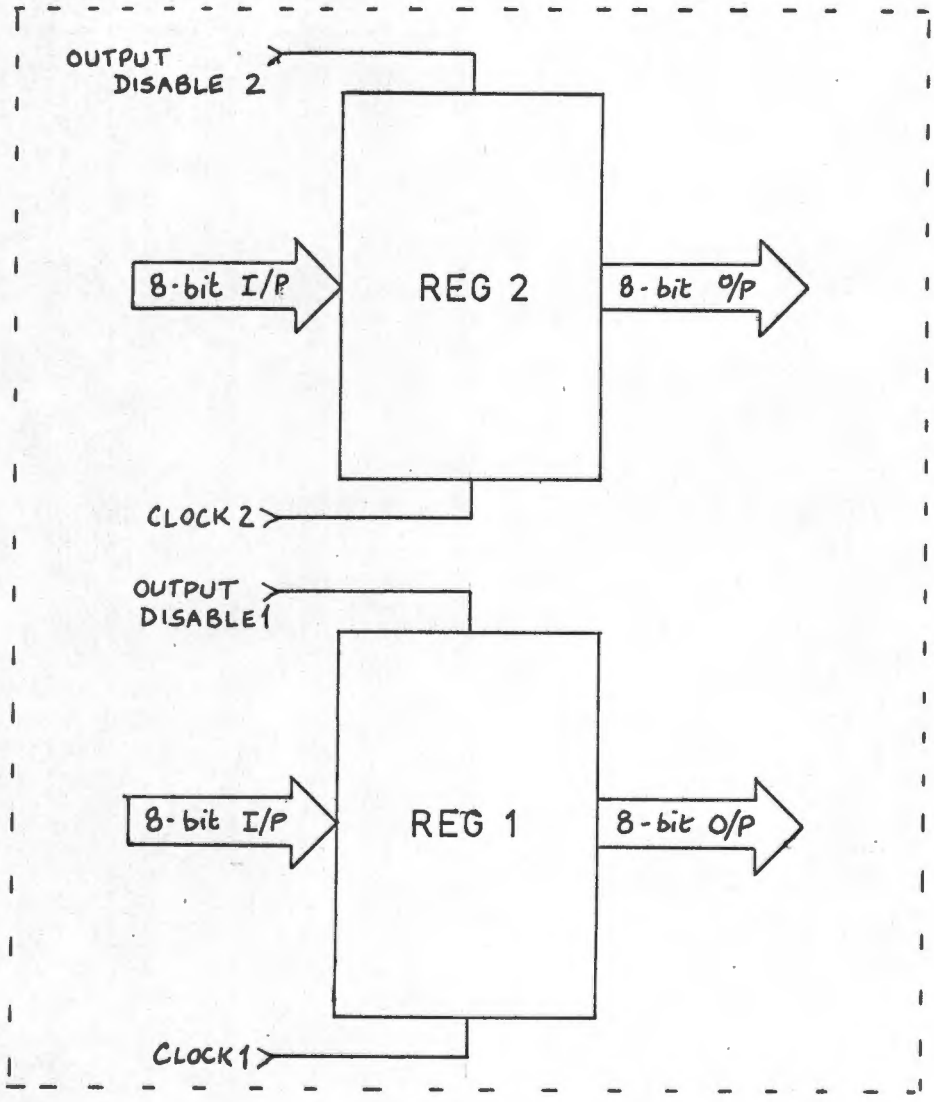


FIG. 12 A typical Interface Register

The four interface registers, each consists of two 74LS374 Tri-state octal D-type flip-flops, see FIG. 12 .
Each of these latches have two control inputs, viz. CLOCK and OUTPUT DISABLE. Data at the D-inputs, meeting the setup and hold time requirements, is transfered to the Q outputs on positive transitions of the CLOCK input. When a high logic level is applied to the OUTPUT DISABLE input, all outputs go to a high impedance state, regardless of the state of the storage elements.
Use is made of this feature to gate data from the two Output Registers, ie. the Mode Register and the Input Data Register to the PIT, thus allowing a 16-bit data word onto a 8-bit bus as two sequential 8-bit data words.

a) MODE REGISTER (OUTPUT REGISTER A)

The Mode Register consists of two 74LS374 tri-state octal D-type flip-flops. Each of these constitute a 8-bit byte. In this application the octal flip-flop representing the LS byte of the Mode Register is called O/P REG 1A while the one representing the MS byte is called O/P REG 2A. These are shown in FIG.13 as IC 21 and IC 22 respectively. The two octal flip-flops are loaded simultaneously from the GPI Data bus via an inverted $\overline{\text{DOA}}$ signal.

The LS byte octal flip-flop has a signal line called $\overline{\text{1A}}$, connected to its OUTPUT DISABLE terminal. When this signal is asserted (1A=0V), the outputs of the octal flip-flop are gated onto the PIT data bus.

The MS byte octal flip-flop that contains the mode select and time-base select bits has its OUTPUT DISABLE terminal tied to ground so that the outputs are never in the high impedance state but always in the state of storage elements.

The mode select outputs are called M0 and M1 while the time-base select outputs are called T0, T1 and T2.

b) INPUT DATA REGISTER (OUTPUT REGISTER B)

This register like the Mode Register, consists of two 74LS374 octal D-type flip-flops. The octal flip-flop representing the LS byte of a 16-bit word is called O/P REG 1B, while the one representing the MS byte is called O/P REG 2B. In FIG.13 O/P REG 2A and O/P REG 2B are represented by IC 19 and IC 20 respectively. The two flip-flops are loaded simultaneously from the GPI data bus via an inverted $\overline{\text{DOB}}$ signal.

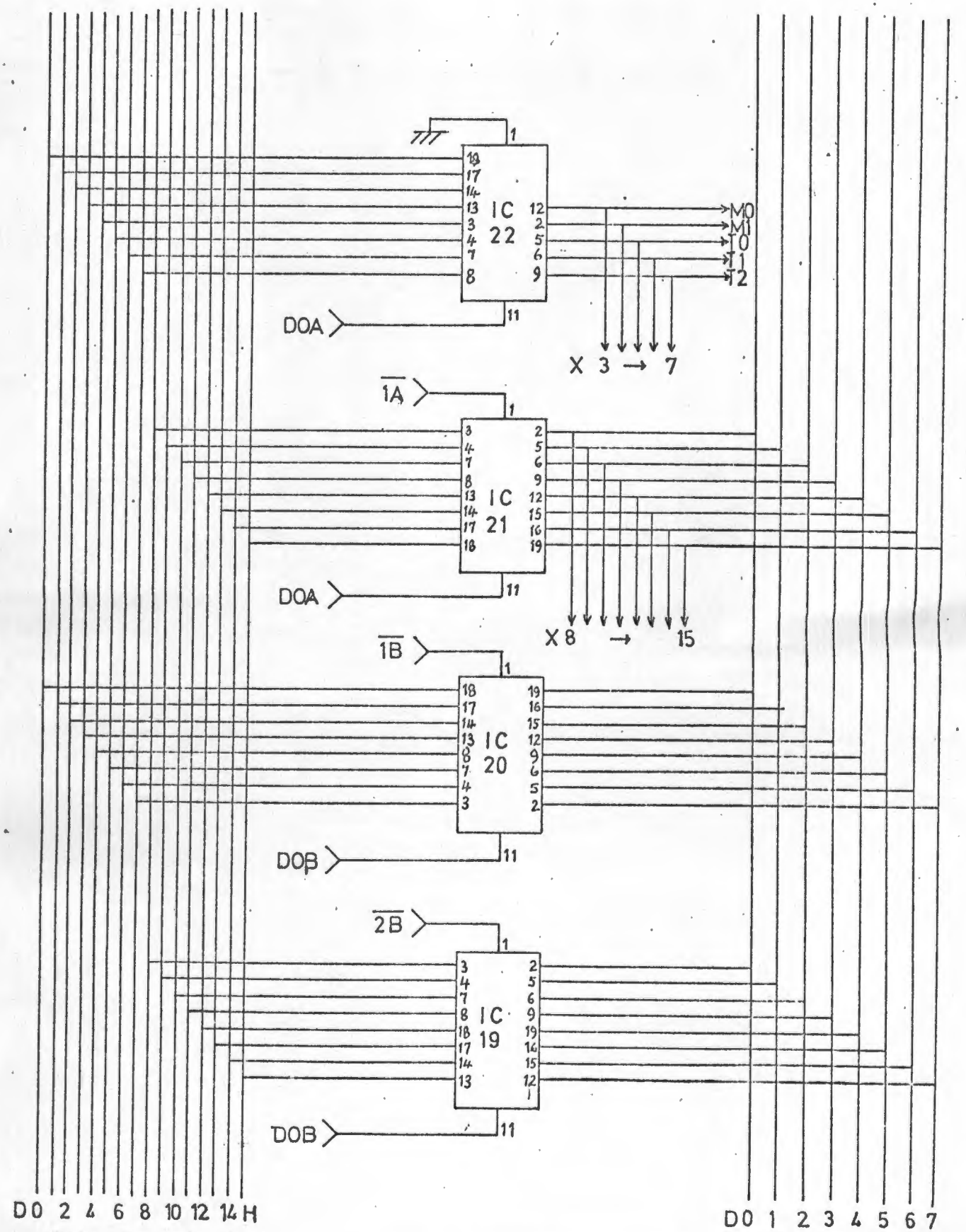


FIG.13 OUTPUT REGISTERS

The LS byte flip-flop has a signal line, $\overline{1B}$, connected to its OUTPUT DISABLE terminal while the MS byte has a signal line, $\overline{2B}$, connected to its OUTPUT DISABLE terminal. The $\overline{1B}$ and $\overline{2B}$ signals are asserted sequentially; this allows for the sequential gating of the outputs of O/P REG 1B and O/P REG 2B respectively, onto the PIT data bus.

c) STATUS REGISTER (INPUT REGISTER A)

This register consists of two 74LS374 octal D-type flip-flops which are called I/P REG 1A and I/P REG 2A.
These two octal flip-flops are used to latch the 13 LS bits of the SETUP word when it is loaded into the Mode Register.
The MS byte, ie the 5 MS bits, is handled by the I/P REG 1A.
These registers represented in FIG. 14 by the IC 24 and IC 23 respectively.
The inputs of these octal flip-flops are connected to the GPI data bus and they are clocked by the same inverted $\overline{DOA}$ signal as the Mode Register. The outputs of these flip-flops feed via 13 open-collector TTL drivers onto the GPI data bus.
This register is used to check the loaded SETUP word.
The OUTPUT DISABLE terminals of both octal flip-flops are grounded so that the outputs represent the state of storage elements.
The 16-bit input to the GPI data bus is completed by the three status flip-flops, viz. CLOCK OVERRUN, EXTERNAL INTERRUPT and LOST DATA which feed into the 3 MS open-collector TTL drivers.
These 16 open-collector TTL drivers, which are shown in FIG. 14 as IC's 31, 32, 33 and 34, allows status data onto the GPI via a DIA instruction.

NOTE: Open collector drivers are specified requirements for the GPI input data lines D(0-15)L.

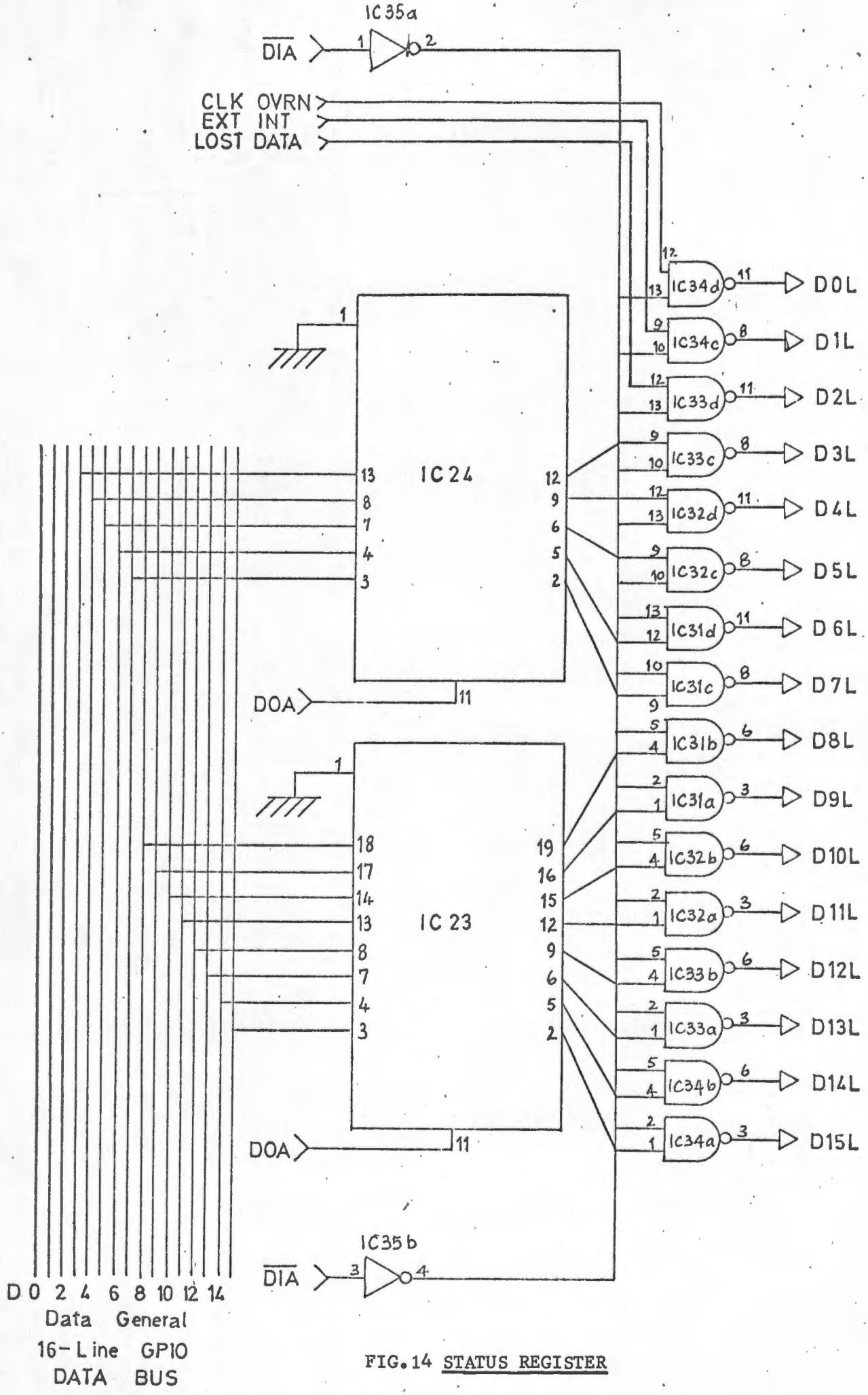


FIG. 14 STATUS REGISTER

d) OUTPUT DATA REGISTER (INPUT REGISTER C)

This register consists of two 74LS374 octal D-type flip-flops, I/P REG 1C and I/P REG 2C which are shown in FIG. 15 as IC 26 and IC 25 respectively. These two flip-flops are used to latch the current time of the PIT whenever an event occurs or the PIT is pulsed. The inputs of these two octal flip-flops are connected to the PIT data bus, and I/P REG 1C and I/P REG 2C are clocked sequentially by signals S1 and S2 respectively. This allows the two 8-bit bytes from the PIT to be arranged as a 16-bit word for the GPI data bus.

The OUTPUT DISABLE terminal of both the octal flip-flops are grounded so that the outputs represent the state of the storage elements. The outputs of the flip-flops feed onto the GPI data bus via 16-open-collector TTL drivers which are represented in FIG. 15 by IC's 27, 28, 29 and 30. Data from this register is loaded onto the GPI bus via a $\overline{\text{DIC}}$ instruction.

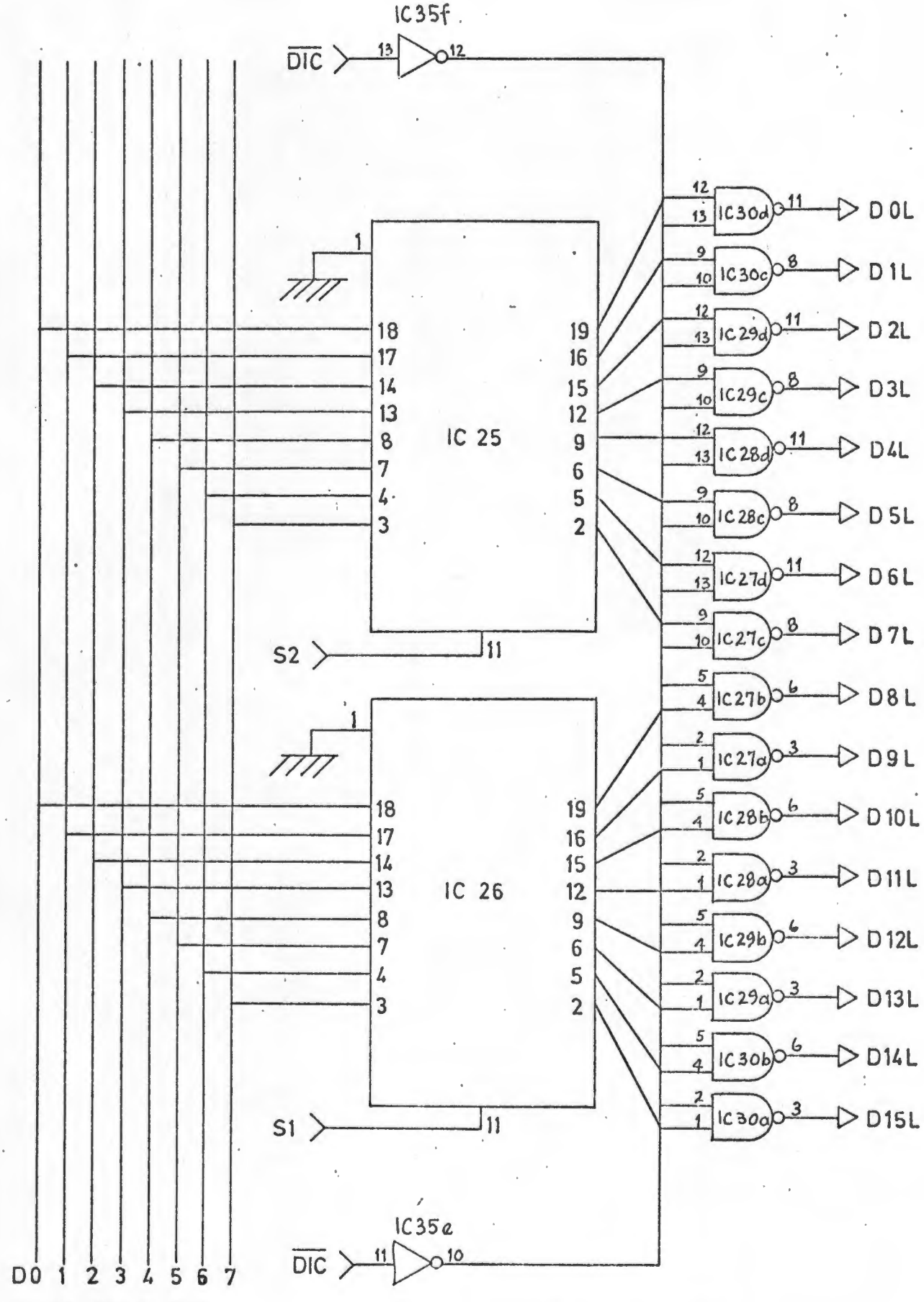


FIG. 15 OUTPUT DATA REGISTER

3.2. COMMAND INTERPRETER AND PIT TIMING WAVEFORM GENERATOR

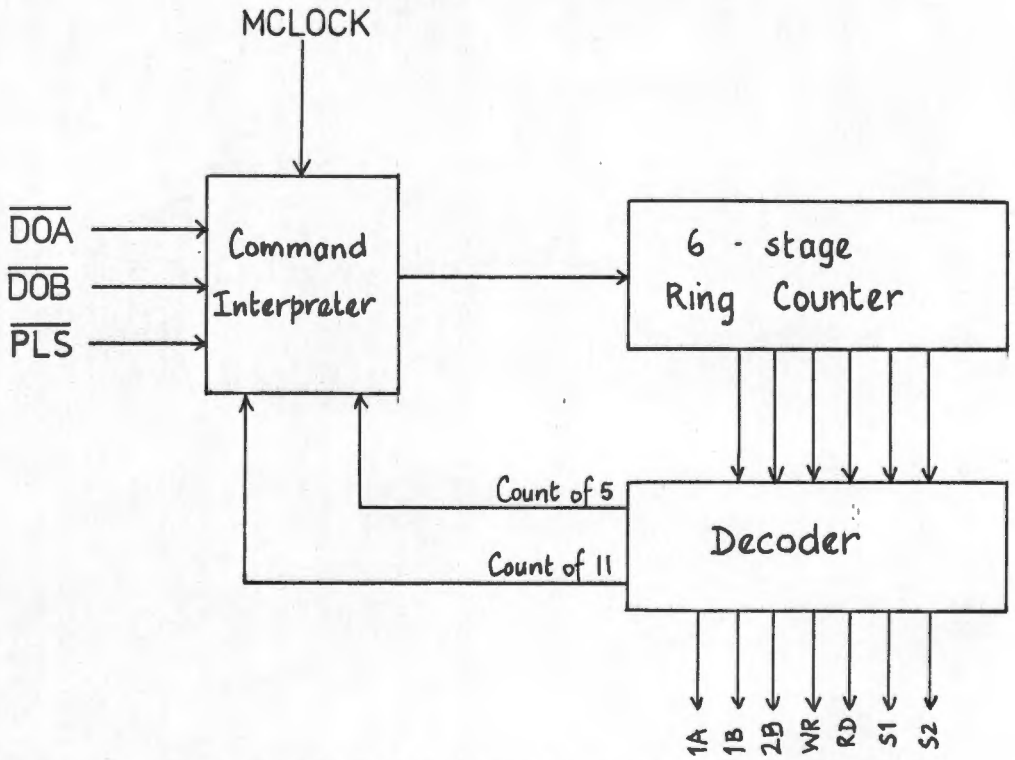


FIG.16 Schematic diagram of Command Interpreter and Timing waveform generator

There are three instances when timing waveforms need to be generated. These correspond to the inputs to the Command Interpreter, FIG. 16 and are given as :

a) Loading the PIT CONTROL WORD into the 8253 timer via a $\overline{DOA}$ signal.

b) Loading the initial count, TIME into the 8253 timer via a $\overline{DOB}$ signal.

c) Reading the current value of the PIT via a $\overline{PLS}$ signal.

At first it was decided to generate timing waveforms by using monostables. However these these monostables are dependent on resistive and capacitive components which are unreliable if precise timing is required.

It was then decided to generate timing waveforms by counting MCLOCK (a GPI signal line) pulses via a 6-stage ring counter. The unique out states produced by the ring counter are decoded to generate specific signals. This method is similar to the one used by Waterfall (1979) in interfacing a 16-bit IBM computer to a 24-bit Ferranti Display Control Module (two non compatible parallel equipment).

The timing waveforms that were decided upon, meet the INS 8253 manufacturers I/O specifications, eg the $\overline{RD}$ and $\overline{WR}$ pulse width exceeds the specified minimum of 400 ns. The signals that are used to enable the outputs of registers, which load data onto the PIT data bus, meet the setup and hold time requirements of the 8253 bus.

Table 6. displays the 6-stage ring counter states for each count.

TABLE 6: RING COUNTER OUTPUT STATES

COUNT	Q1	Q2	Q3	Q4	Q5	Q6
0	0	0	0	0	0	0
1	1	0	0	0	0	0
2	1	1	0	0	0	0
3	1	1	1	0	0	0
4	1	1	1	1	0	0
5	1	1	1	1	1	0
6	1	1	1	1	1	1
7	0	1	1	1	1	1
8	0	0	1	1	1	1
9	0	0	0	1	1	1
10	0	0	0	0	1	1
11	0	0	0	0	0	1
12	0	0	0	0	0	0

= is used to generate $\overline{RD}$ or $\overline{WR}$ signals

= is used to enable register outputs

Four unique states, as shown in TABLE 6 are used to generate the generalised timing waveforms. These states give rise to the following signals :

$$\overline{WR1} = \overline{\overline{Q4}.\ \overline{Q6}} = Q4 + Q6$$

$$\overline{OP1} = \overline{\overline{Q5}.\ \overline{Q6}} = Q5 + Q6$$

$$\overline{WR2} = \overline{Q4.\ Q6}$$

$$\overline{OP2} = \overline{Q5.\ Q6}$$

SEE FIG. 17

$\overline{WR1}$ and $\overline{WR2}$, each of which are 480 nsec negative going pulses $\overline{WR}$ and $\overline{RD}$ pulses for the 8253, while $\overline{OP1}$ and $\overline{OP2}$, each of which are 600 nsec negative going pulses are used to enable outputs of the registers that are used to

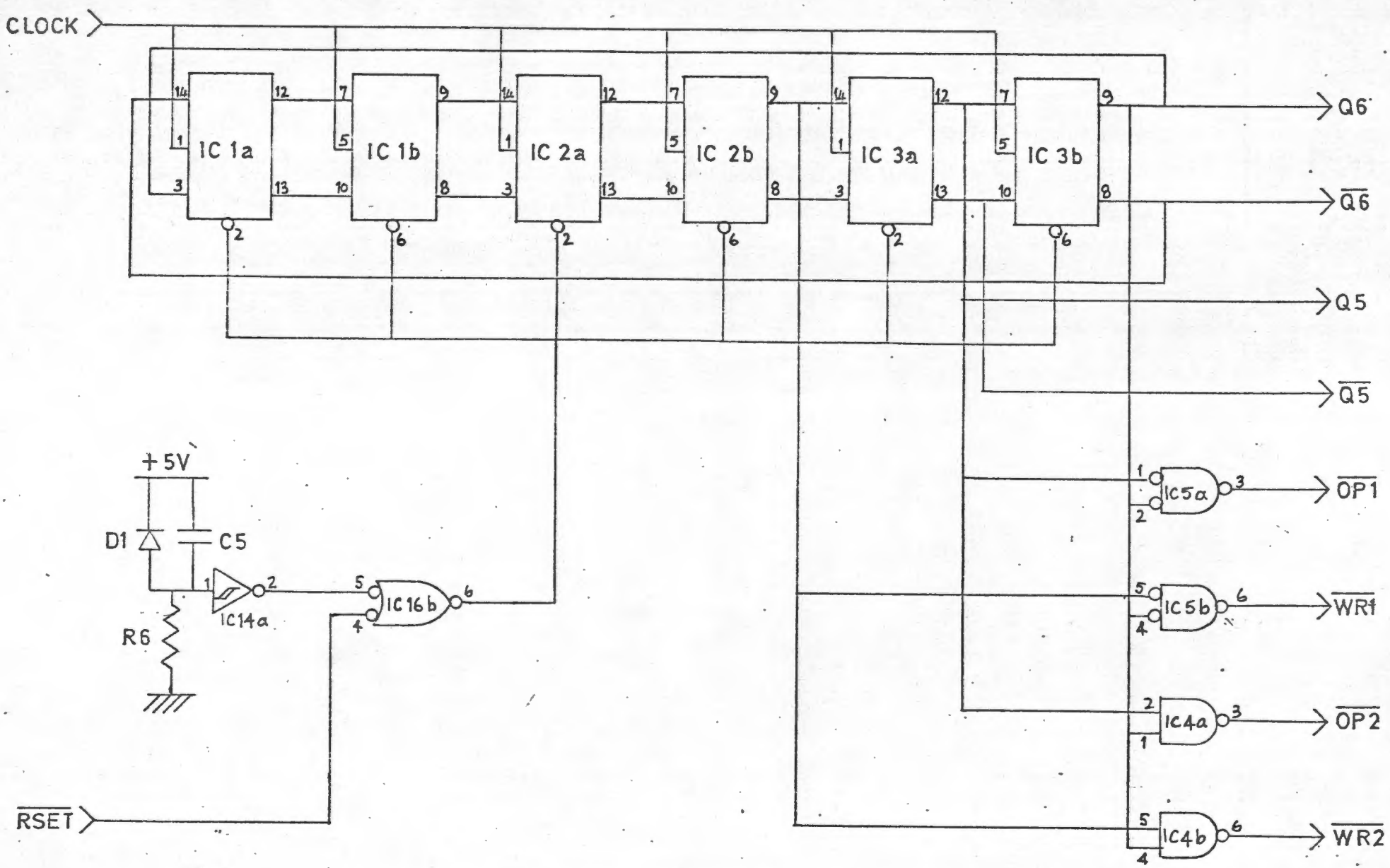


FIG.17. RING COUNTER – Timing Waveform Generator

load data into the 8253 timer. The signals are manipulated by the Command Interpreter according to the type of signal it receives, FIG.s 18 and 19.
When only one $\overline{WR}$ pulse is required the CLOCK is disabled by the $\overline{RSET}$ signal after the count of 5, see FIG. 20 .
However if a normal transfer, ie loading or reading two 8-bit words occur the CLOCK is only disabled after the count of 11.

a) LOADING THE PIT CONTROL WORD

The PIT CONTROL Word is the LS byte of the SETUP word which is loaded into the Mode Register via a $\overline{DOA}$ instruction.
The timing waveforms that are initiated by the $\overline{DOA}$ instruction are given in FIG. 21.
The $\overline{DOA}$ signal enables the clock by clocking a flip-flop IC 11(a) in FIG. 18 causing the output EA to go a high logic level ("HI"). EA causes the clock to pulse the ring counter. These pulses are shown as CLOCK in FIG. 18 . Since the PIT CONTROL WORD transfer is only one 8-bit word transfer, only one $\overline{WR}$ pulse is required to load the word into the 8253 timer. The data is setup for loading by enabling the output of the O/P REG 1A. The signal $\overline{1A}$, a derivative of $\overline{O/P\ 1}$, is used to enable the O/P REG 1A output. The data is then written into the 8253 by the positive edge of the $\overline{WR}$ pulse. The timing sequence is terminated at the count of 5.

b) LOADING THE INITIAL COUNT, TIME

The initial count is loaded into the INPUT DATA REGISTER via a $\overline{DOB}$ instruction. This $\overline{DOB}$ instruction initiates the timing waveforms that are shown in FIG. 22 .

The $\overline{DOB}$ signal clocks the flip-flop IC 11(b), FIG. 18 , causing the output EB to go HI. This enables the CLOCK to pulse the ring counter. The generalised waveforms are generated and the Command Interpreter converts these waveforms into $\overline{1B}$, $\overline{2B}$ and $\overline{WR}$ signals, FIG.19 . $\overline{WR}$ consists of the two sequential pulses $\overline{WR1}$ and $\overline{WR2}$.

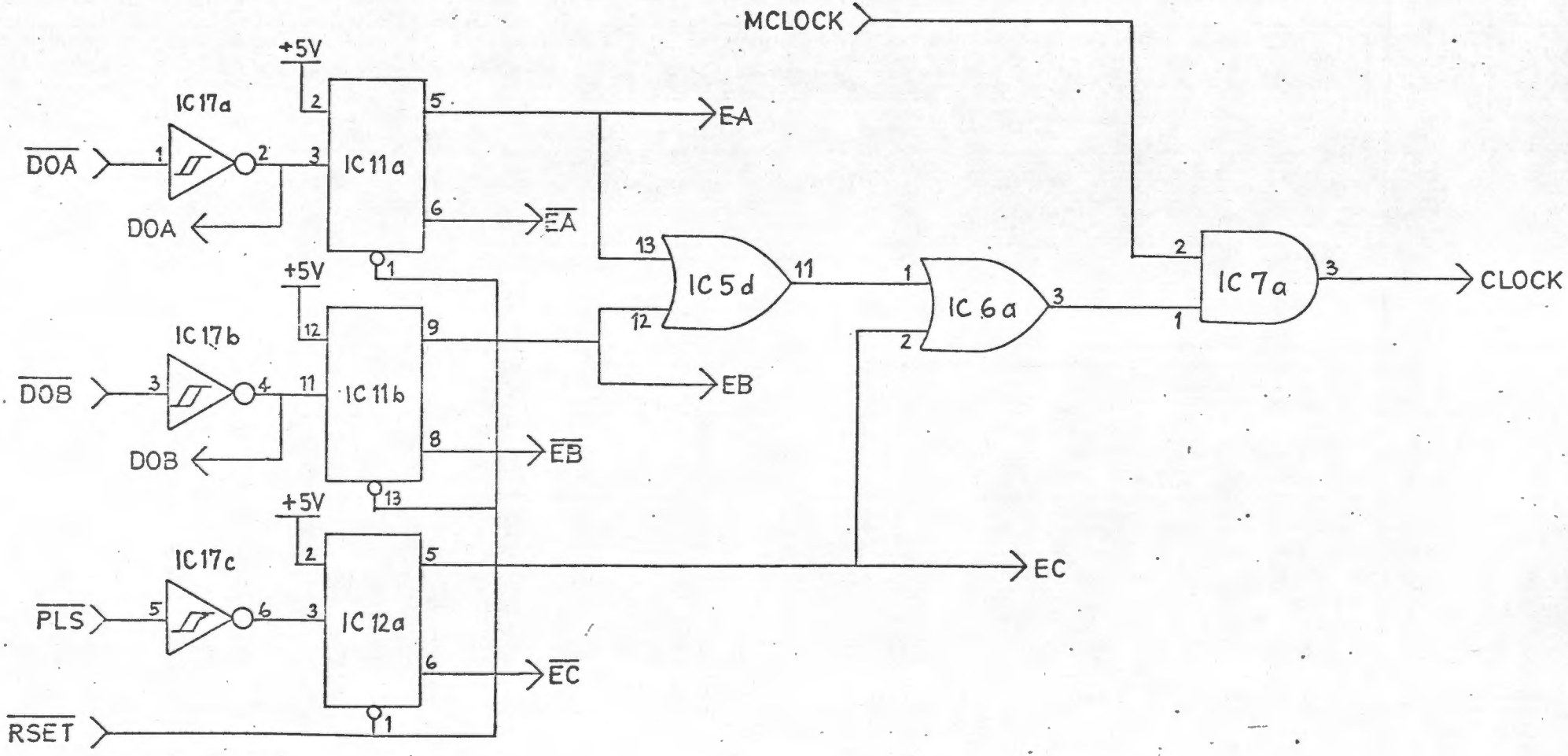


FIG. 18. COMMAND INTERPRETER

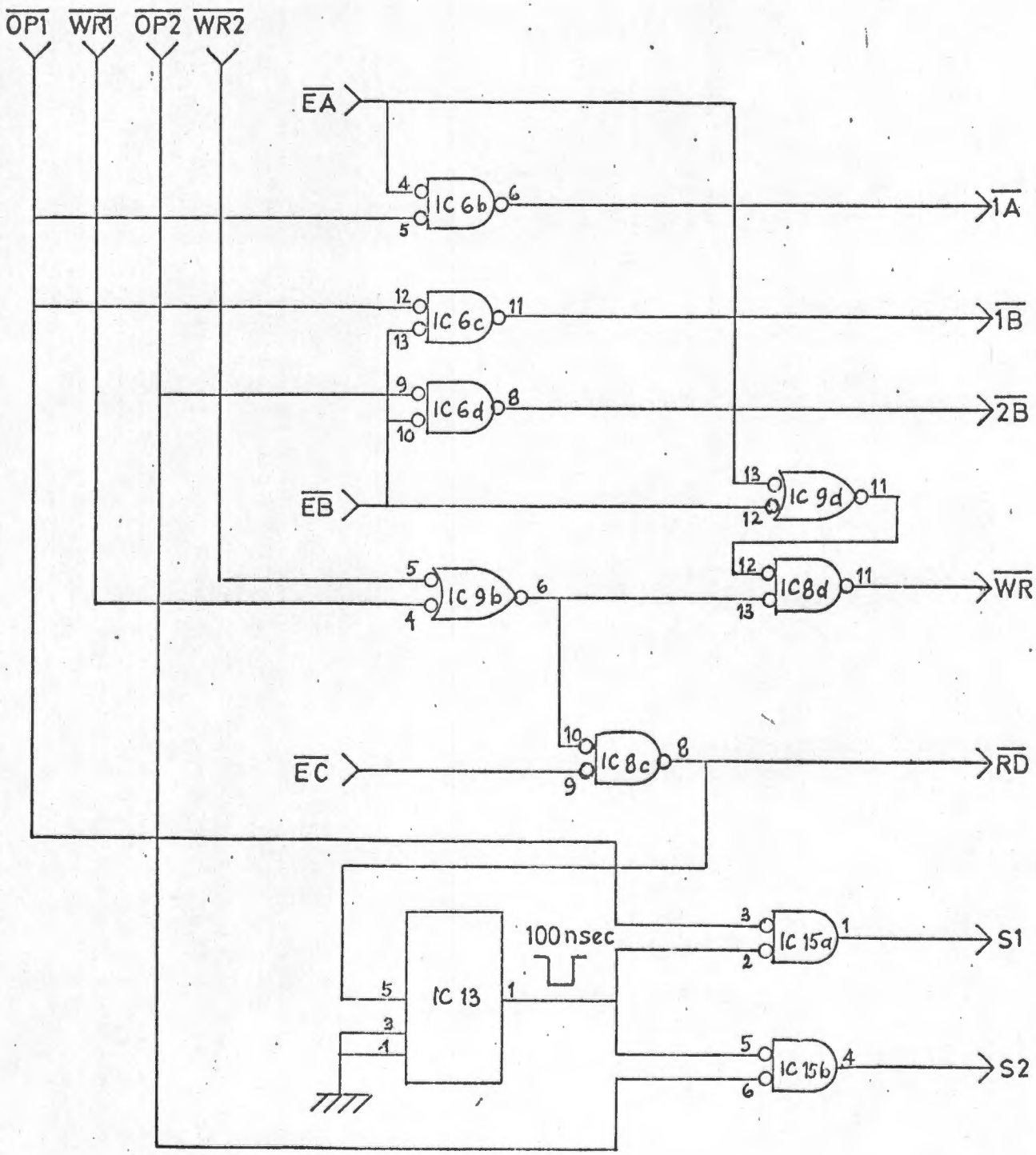
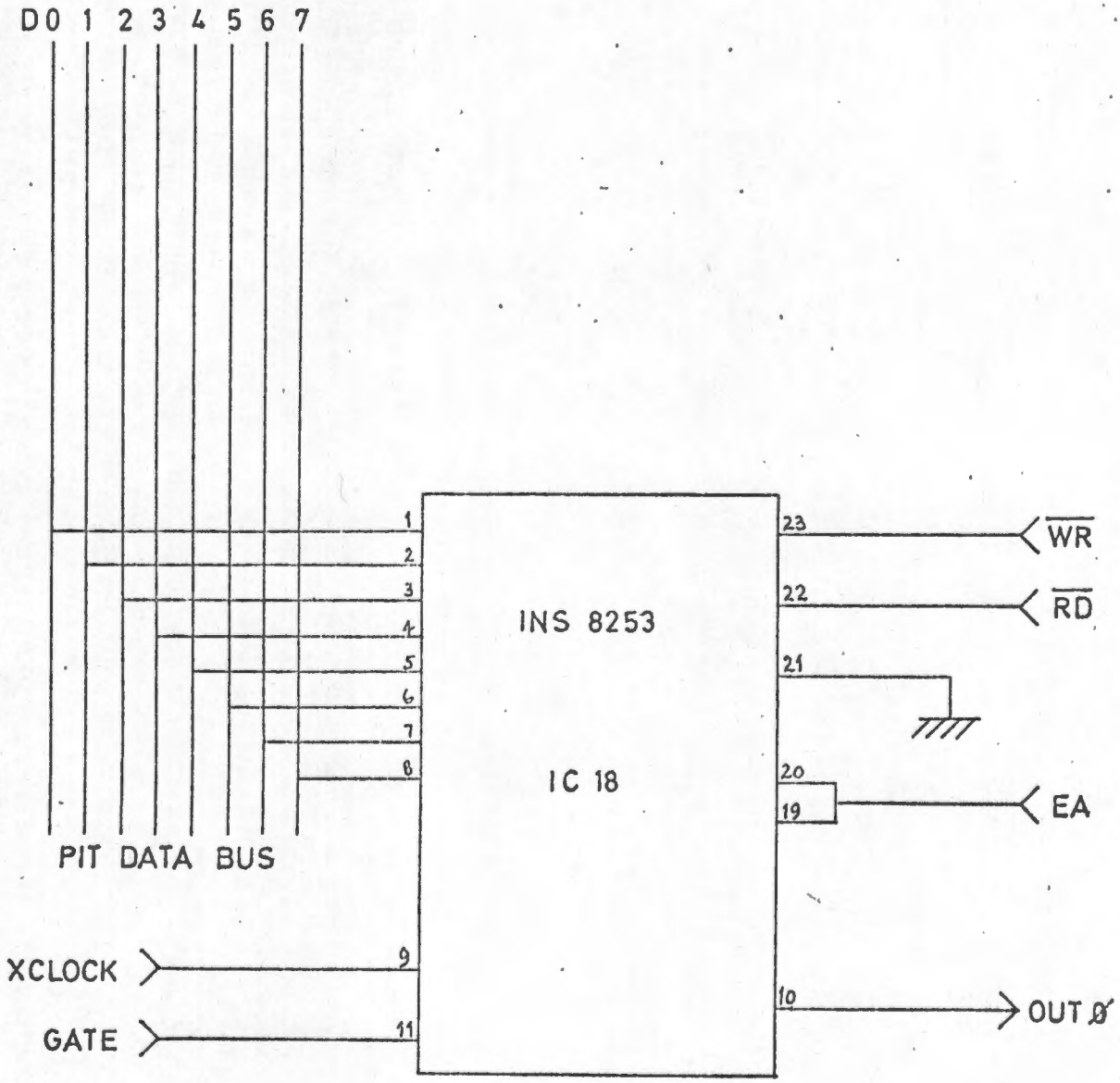


FIG. 19. SIGNAL "PROCESSOR"


D0 1 2 3 4 5 6 7
PIT DATA BUS
INS 8253
IC 18
1
2
3
4
5
6
7
8
9
11
23
22
21
20
19
10
WR
RD
EA
XCLOCK
GATE
OUT Ø

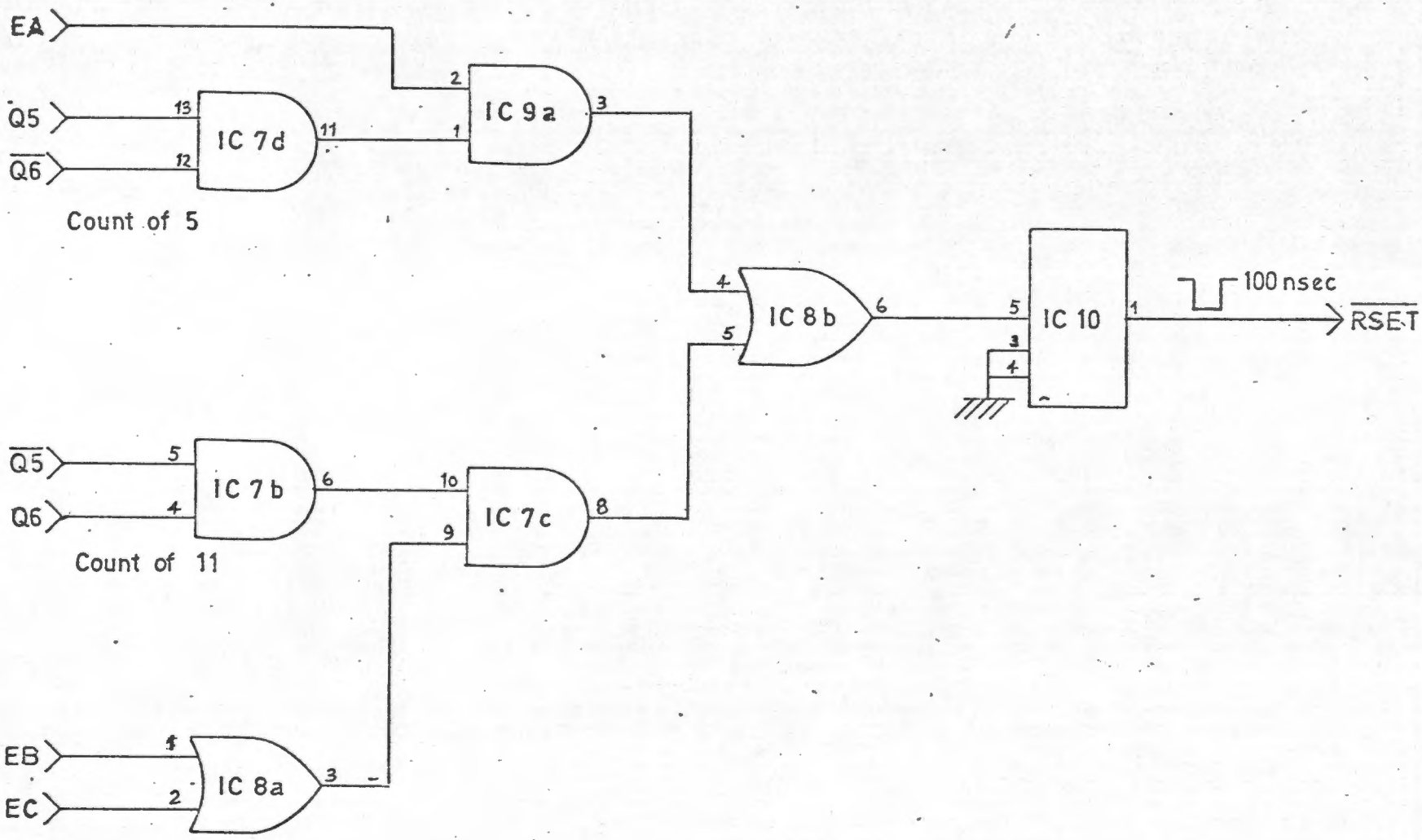


FIG.20. RESETTING CIRCUIT

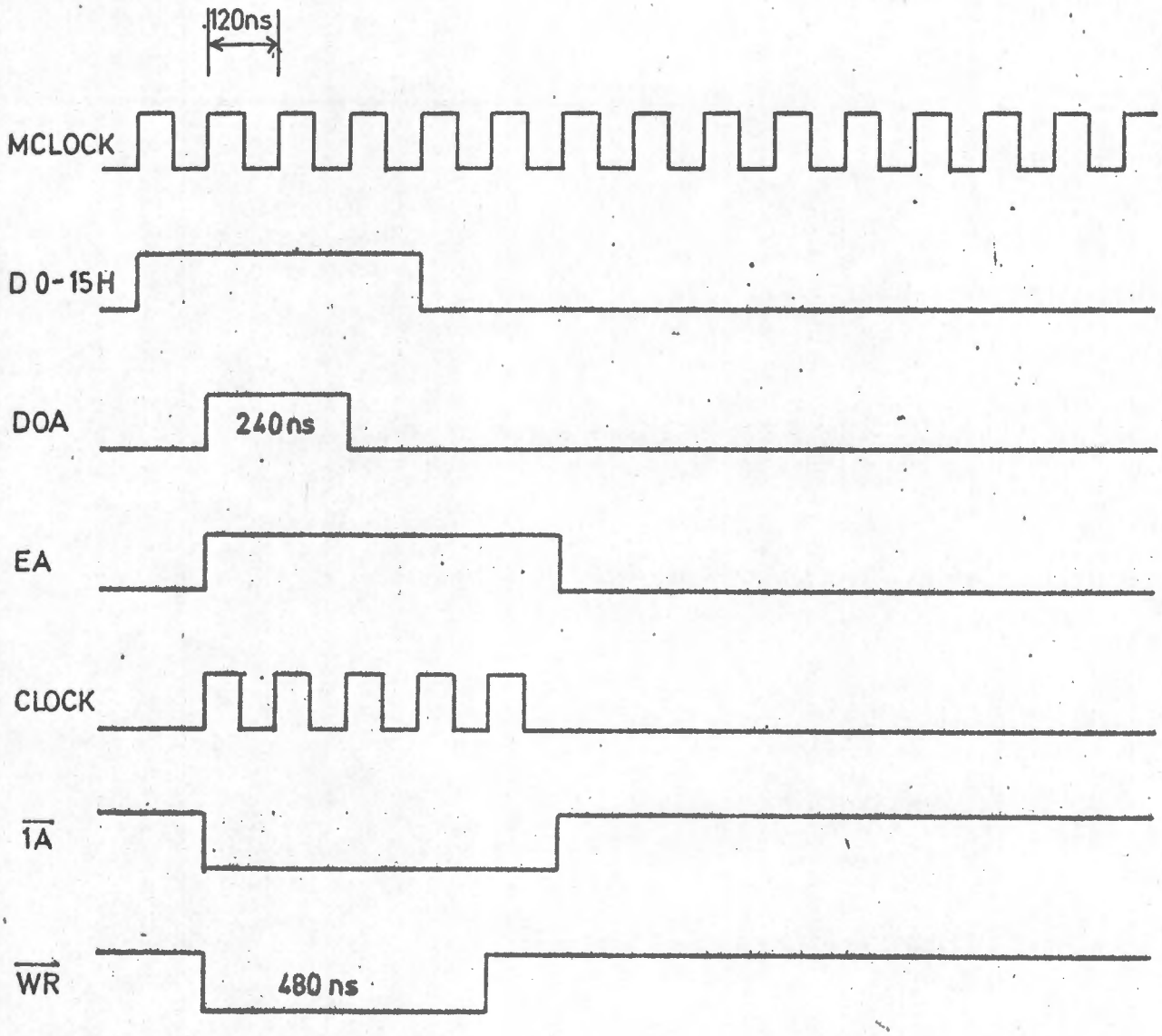


FIG. 21. LOADING CONTROL WORD

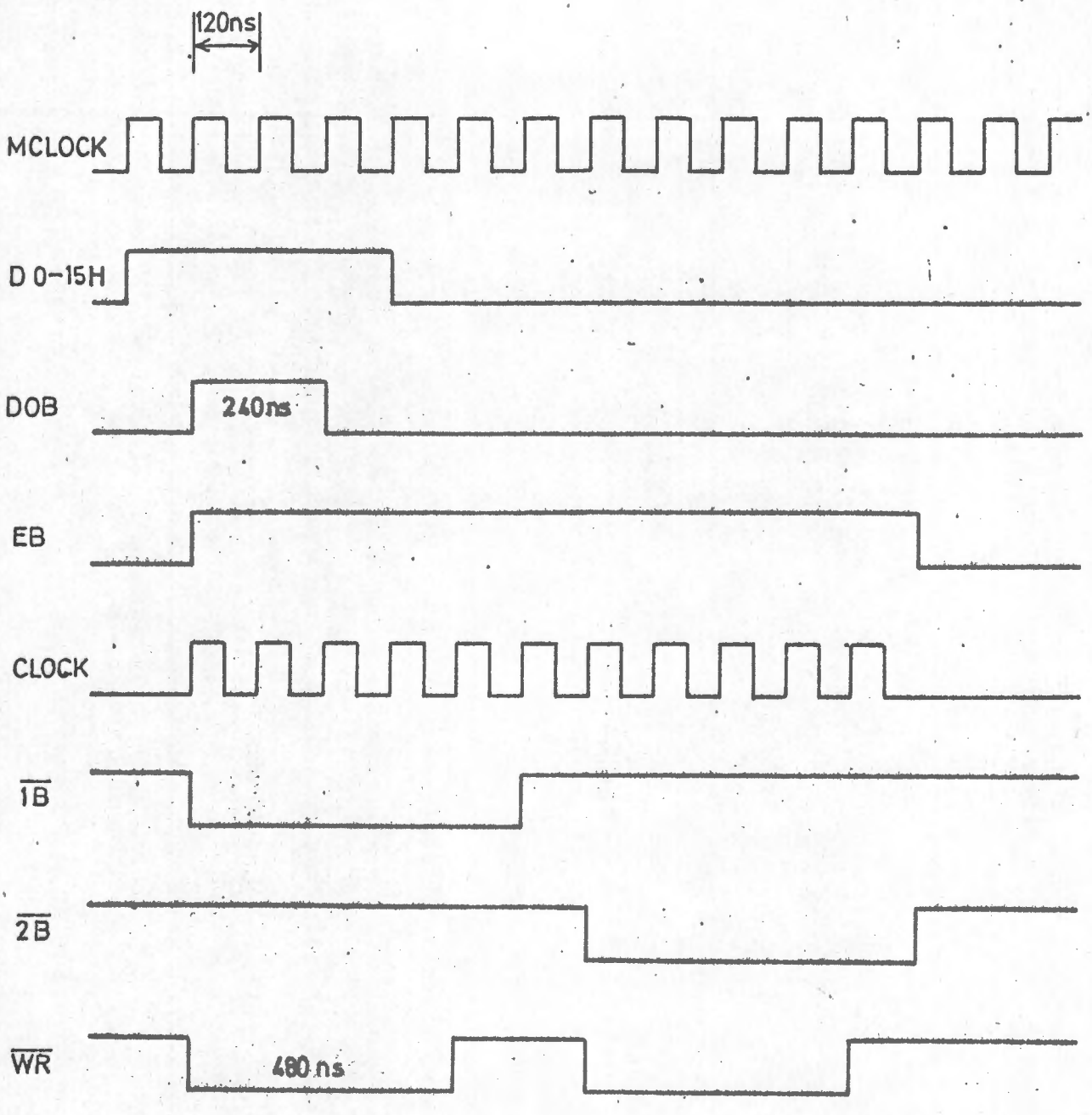


FIG. 22. LOADING INITIAL COUNT

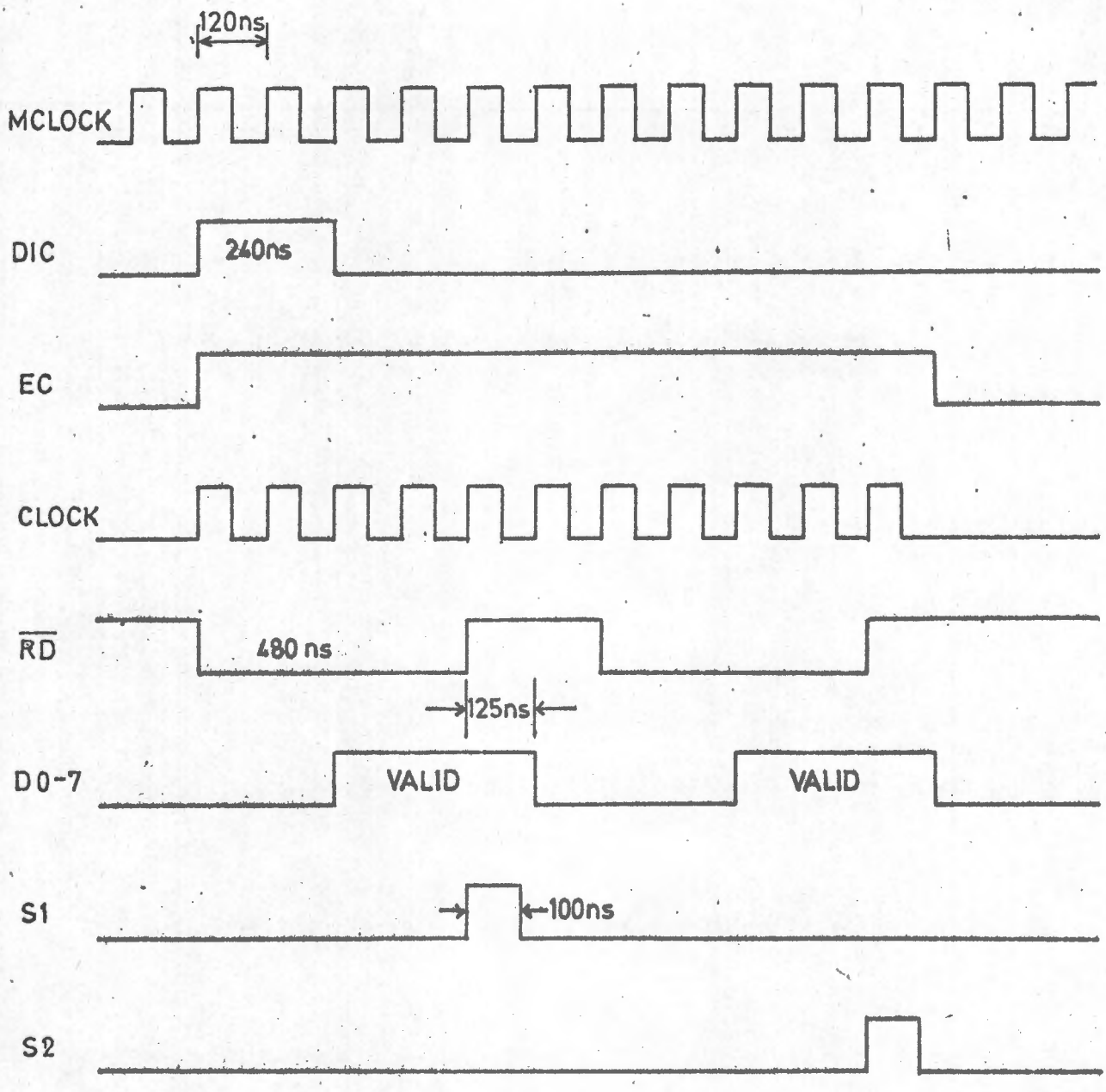


FIG. 23. READING THE PIT

The signal $\overline{1B}$ enables the outputs of O/P REG 1B and $\overline{WR1}$ writes the LS byte of data into the 8253 timer.
The signal $\overline{2B}$ enables the outputs of O/P REG 2B and $\overline{WR2}$ then writes the MS byte of data into the 8253 timer.
The CLOCK is disabled on the count of 11.

c) READING THE PIT

The $\overline{PLS}$ signal that comes from the interrupt circuitry is used to read the PIT.

When this signal is asserted IC 12(a) FIG.18 is clocked causing EC to go HI. The CLOCK is enabled by this causing it to pulse the ring counter. The generalised signals $\overline{WR1}$ and $\overline{WR2}$ are converted by the Command Interpreter to $\overline{RD}$ signal consisting of two pulses, See FIG. 19 .
Data becomes valid on the PIT data bus and remains valid for 125 nsec after each $\overline{RD}$ pulse. The positive going edge of each $\overline{RD}$ pulse is then used to trigger a 100 nsec monostable.

The output pulses of the monostable are gated to the two octal flip-flop clock inputs as S1 and S2 by the $\overline{OP1}$ and $\overline{OP2}$ signals respectively. (FIG.19)

The clock is disabled on the count of 11.

A diagram of the Command Interpreter resetting circuit is given in FIG. 20 .

This circuit initiates the count of 5 and count of 11 reset.

3.3. INTERRUPT CIRCUITRY

The circuitry, FIG.24 involves the setting of the interface BUSY and DONE flags. The BUSY and DONE flags exist in the GPI IOC and are set via the $\overline{\text{SET BUSY}}$ and $\overline{\text{SET DONE}}$ signals respectively.

The setting of the DONE flag is used to interrupt the CPU, thus allowing it to jump to an interrupt handler routine.

The BUSY and DONE flags are cleared or set to zero, by the $\overline{\text{CLR}}$ and $\overline{\text{IORST}}$ signals when requested by the program.

The $\overline{\text{STRT}}$ signal is used to set the BUSY flag to 1.

Only two signals can set the DONE flag. These are the EXTERNAL INTERRUPT signal, which is asserted when an event occurs, and the CLOCK OVERRUN signal, which is asserted when a positive transmission occurs at the OUT 0 terminal of the 8253 timer. The EXTERNAL INTERRUPT signal can only set the DONE flag in Mode 1 ie when M1 = 1. This allows for uninterrupted counting in Mode 2. The BUSY flag must be set before the DONE flag can be set.

3.4. GATING (TRIGGER) CIRCUIT

A diagram of the trigger circuit is give in FIG.25 .

The following signals start/restart the 8253 timer.

i) START
ii) CLOCK OVERRUN
iii) S1
iv) LOST DATA

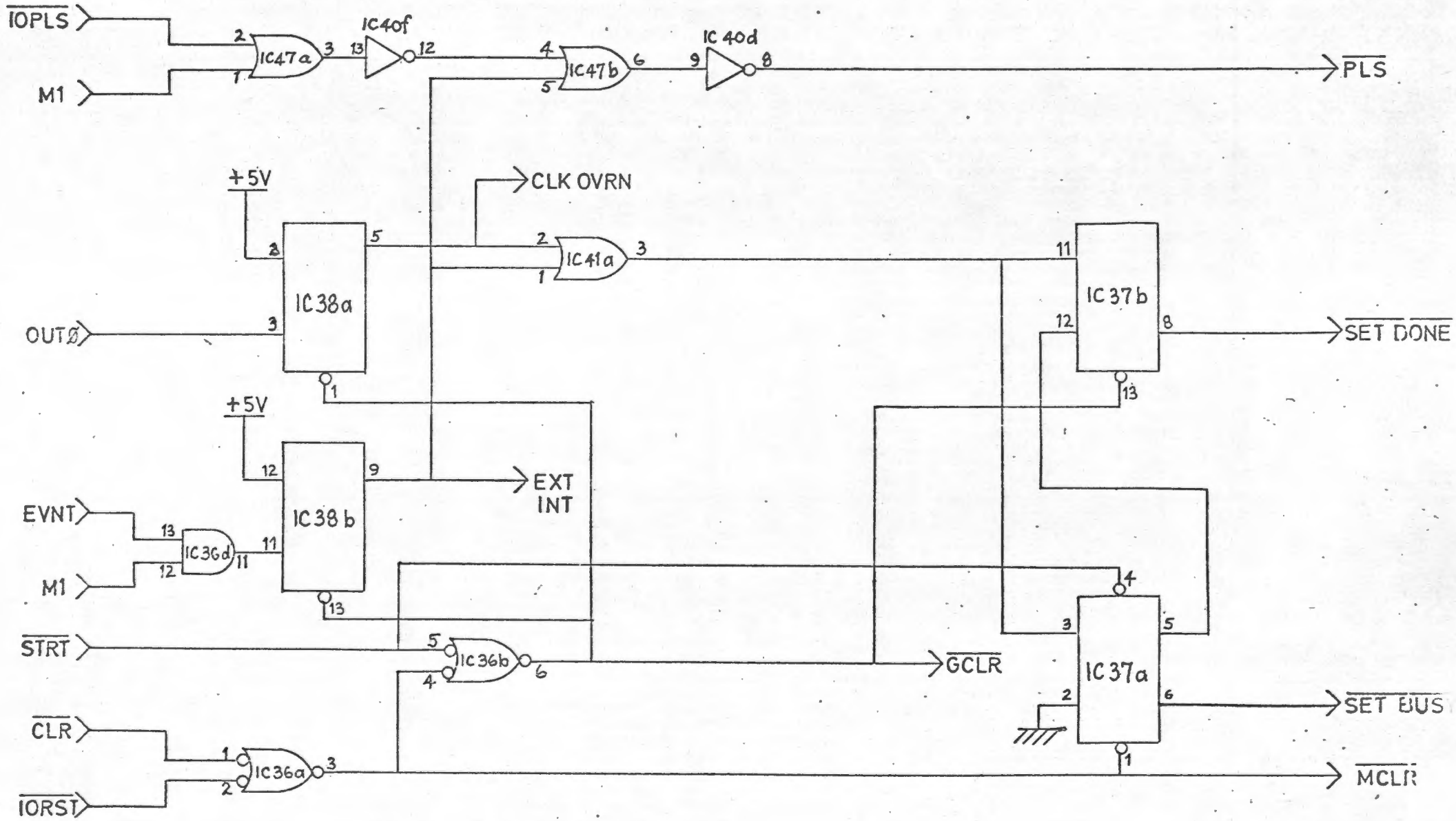


FIG. 24. INTERRUPT CIRCUIT

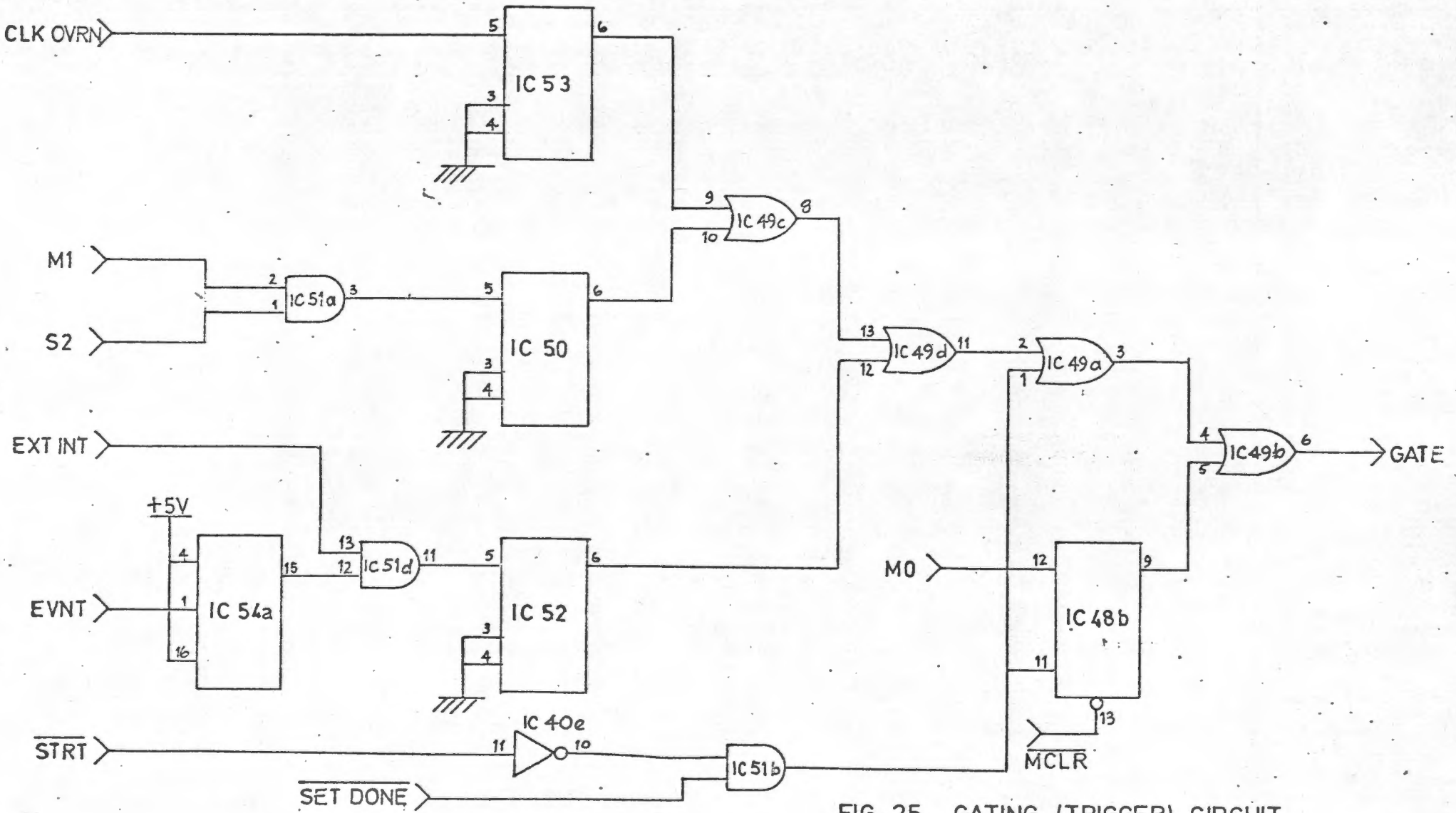


FIG. 25. GATING (TRIGGER) CIRCUIT

i) The START signal is only effective after a CLEAR or a IORST signal.
In mode 1, ie M0 = 0. START results in the signal GATE, being a positive pulse, the positive rising edge of which is used to trigger the 8253 timer, causing it to start counting down from its initial count value. However in the Mode 2, ie M0 = 1, START causes the output of IC. 48b to follow its input M0=1 and remain in that "H1" state. This meets the requirements for the 8253 Mode 2 operation. If the DONE flag is set, START has no effect on the 8253 : this is to allow for Mode 2 effects.

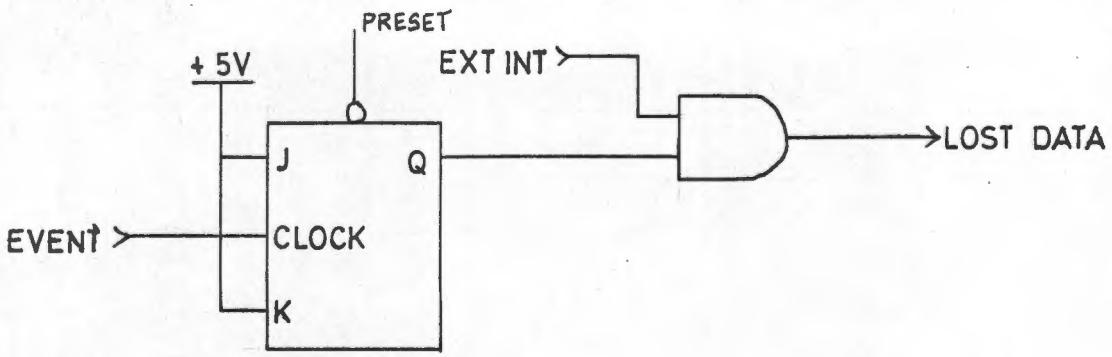
ii) When the CLOCK OVERRUN flag is set, the positive transition triggers a monostable, IC 53 and this results in the GATE signal being a 100ns pulse. The positive rising edge of this pulse is used to trigger the 8253 timer causing it to reload its initial count value and start counting down again.

iii) Each time the PIT is read after an EXTERNAL INTERRUPT, the two latching signals S1 and S2 are generated. S2 is used to retrigger the 8253 via a monostable (100ns pulse) to start a new timing cycle. This method restart is only possible in Mode 1, ie when M1 = 1.

When a NIOP is asserted the data is read from the PIT but the clock carries on unaffected by the "READ"

iv) When the LOST DATA flag is set, ie. if an event occurs while the EXTERNAL INTERRUPT flag is still set from a previous event, the positive transition is used to restart the 8253 pulse from monostable IC 52, FIG. 25.

The LOST DATA flag is generated by the following circuit



The JK flip-flop, IC 54(a) in FIG. 25 is preset so that its output Q is High. When an event clocks the flip-flop, the output Q goes low. However if the next event occurs before the flip-flop is preset again, the output of the flip-flop will go high. This state in the presence of an uncleared EXTERNAL INTERRUPT flag, (from the previous event) results in the LOST DATA flag being set.

Note that in Mode 2, all the restarting features become redundant. Since any START pulse in Mode 2 will result in the GATE signal" remaining high due to the flip-flop IC 48b in FIG. 25.

3.5. CRYSTAL CONTROLLED CLOCK CIRCUIT

The 1MHz crystal oscillator, which consists of IC 40 and related components is shown in FIG. 26 . The output of the oscillator which is TTL compatible is fed into 4 divide by 10 dividers giving a 100Hz signal at the end of the dividers. The 5 outputs, ie clock signal and four signals from the dividers are fed to a time-base select circuit comprising of IC45, IC47 (c) and (d), IC41 (b),(c) and (d), and IC46. The time-base is selected by three inputs T0, T1 and T2 which feed into IC45,a 4 to 16 line decoder. These inputs are specified in the PIT SETUP word.
The output of of IC46, XCLOCK , clocks the 8253 counter # 0 via its CLOCK 0 input, thereby decrementing the counter.

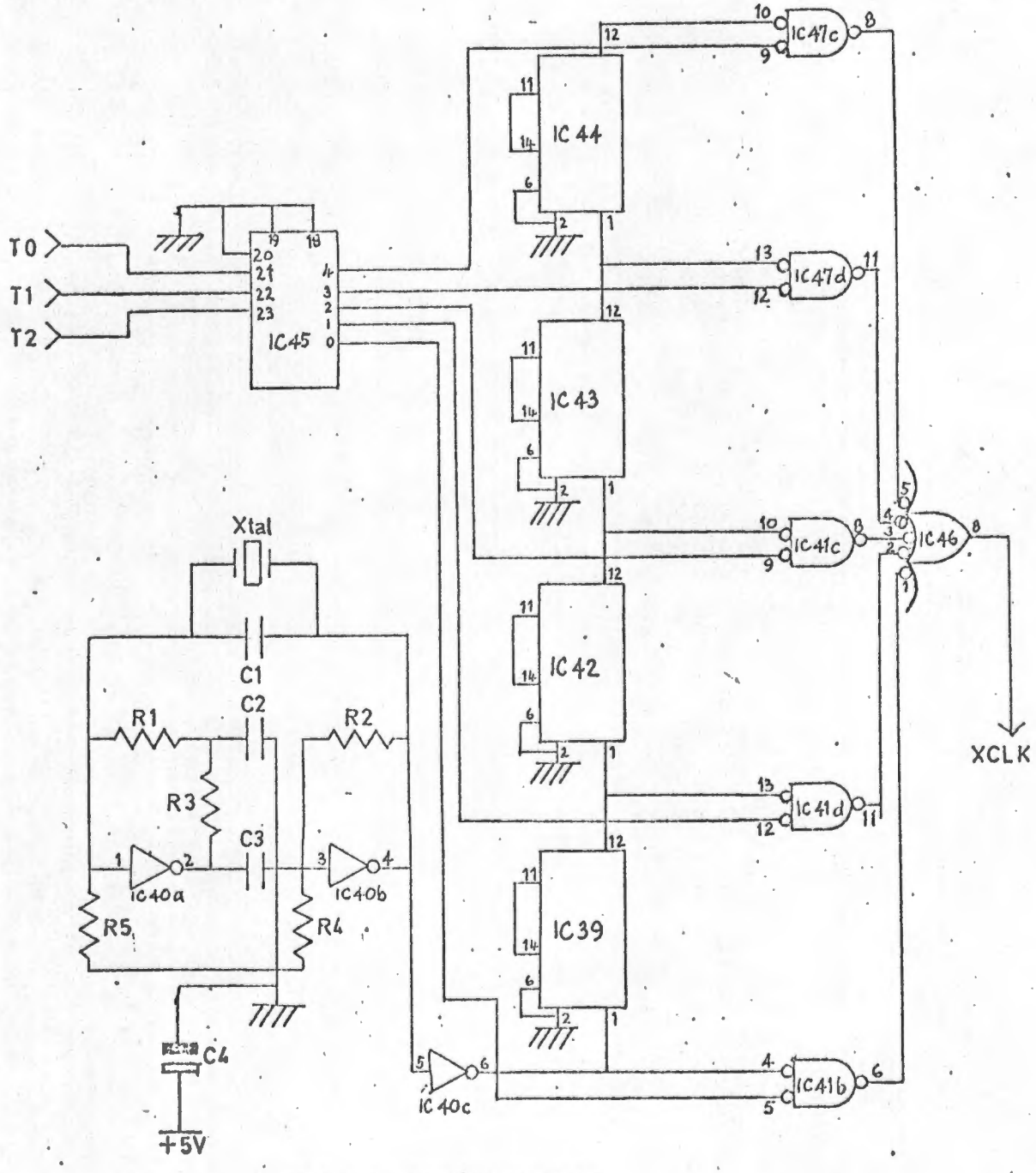


FIG. 26. CLOCK CIRCUIT

CHAPTER 7

TESTING SOFTWARE

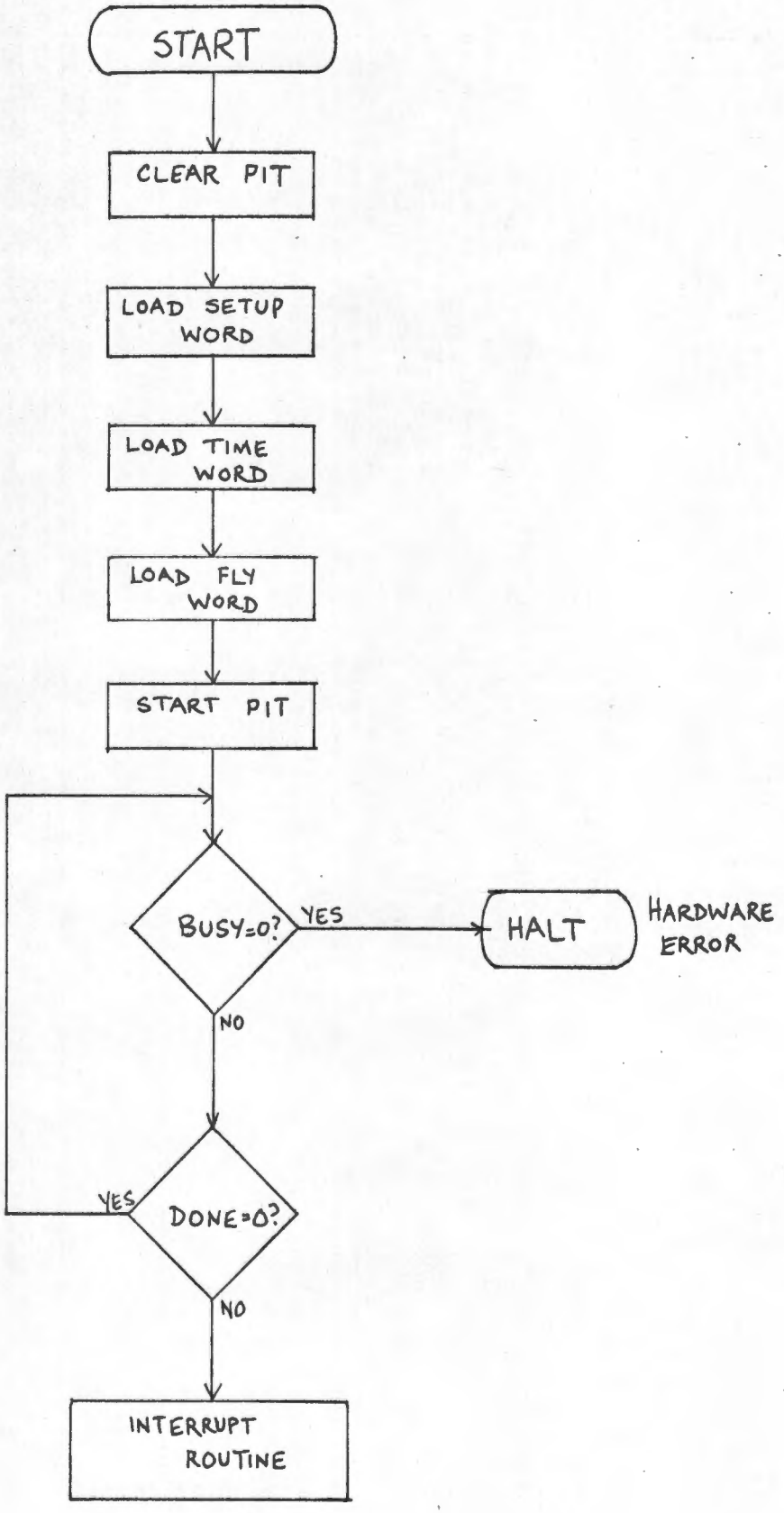
The PIT is programmed in D.G. Assembler language. All the programs discussed here were developed on the Eclipse and then down line loaded to the microNova, to which the PIT is interfaced. The initialisation section of all the PIT programs are the same. This section involves the loading of a control word to the 8253, loading the initial count value and loading a "read on the fly" word. Programs only differ in the routines they perform, when requested to do so.

A quick reference to the PIT SETUP control words, for operating in either the Elapsed-time timer or count down modes is given in TABLE 7. The control words are given in OCTAL. The read on the fly word "FLY" is the same as the SETUP word, except for bits 10 and 11 which are set to zeroes in the FLY word. The FLY word contains Mode and time-base information.

TABLE 7. QUICK SETUP WORDS

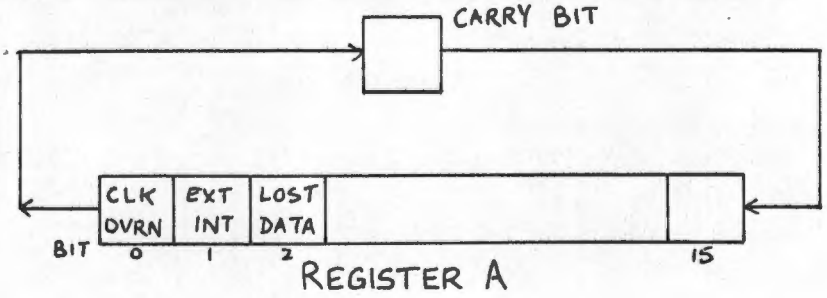
MODE	TIME-BASE	SETUP WORD	CORRESPONDING FLY WORD
1	1 µsec	04062	04002
	10 µsec	04462	04402
	100 µsec	05062	05002
	1 msec	05462	05402
	10 msec	06062	06002
2	1 µsec	10064	10004
	10 µsec	10464	10404
	100 µsec	11064	11004
	1 msec	11464	11404
	10 msec	12064	12004

A TYPICAL INITIALISATION FLOWCHART



ETIM. SR : PROGRAM FOR TESTING PIT IN ELAPSED-TIME TIMER MODE

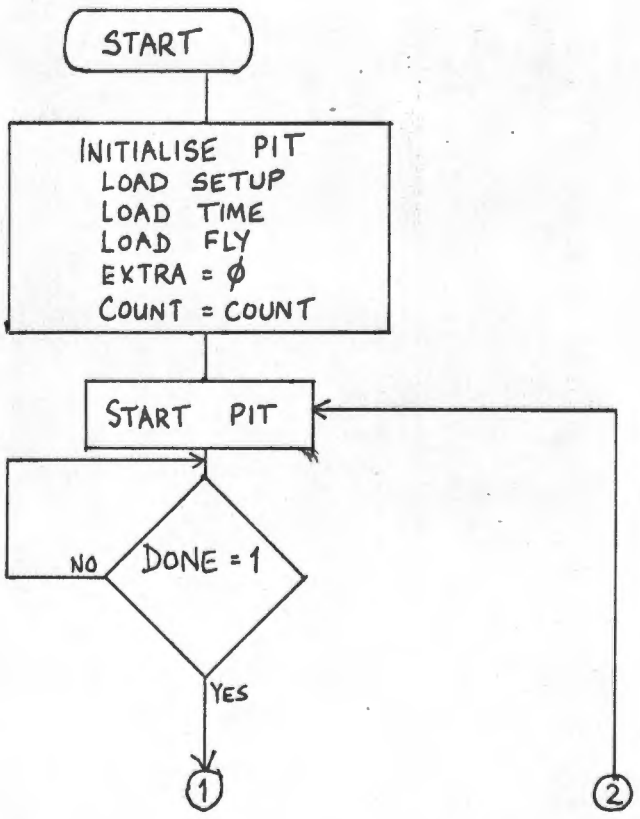
This program halts on error, or at checkpoint where status registers can be interrogated. When the DONE flag is set to "1" or BUSY goes low, due to the occurrence of an event or clock overrun, this Status Register (REGISTER A) is interrogated to determine the source of interrupt. This is done by repeatedly rotating the accumulator, into which Register A was loaded, to the left and examining the carry bit.

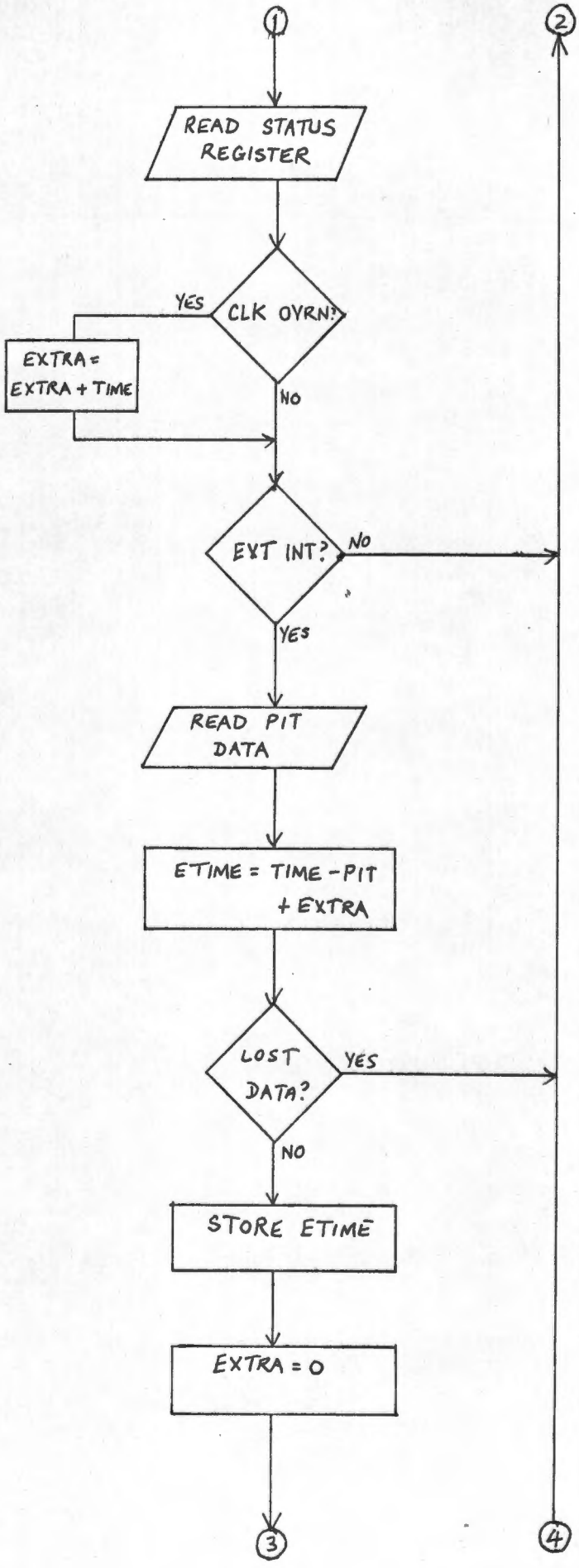


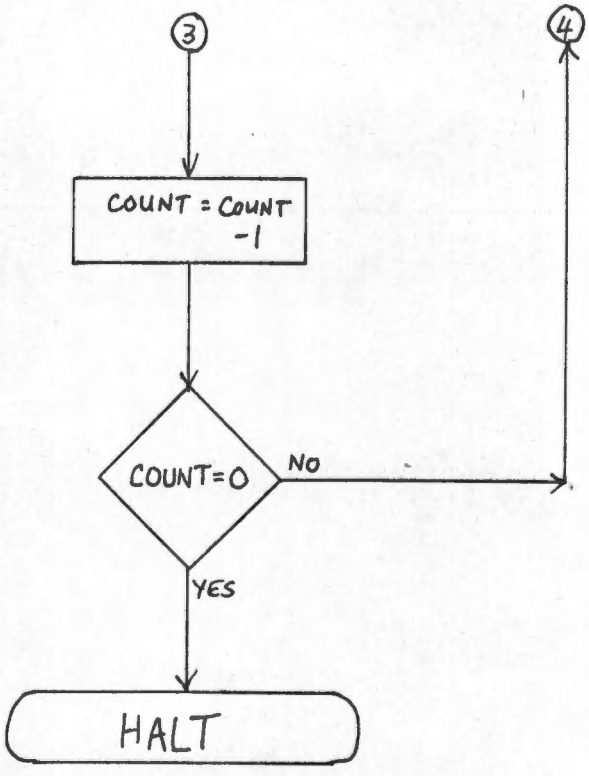
The program then executes a specific routine for each type of interrupt.

Since the value that is read from the PIT is the initial count value decremented by a certain number of clock pulses, the real elapsed time is the initial count, TIME minus the read time.

The program is described in the following flow chart.




1
2
READ STATUS REGISTER
CLK OVRN?
YES
NO
EXTRA = EXTRA + TIME
EXT INT?
NO
YES
READ PIT DATA
ETIME = TIME - PIT + EXTRA
LOST DATA?
YES
NO
STORE ETIME
EXTRA = 0
3
4



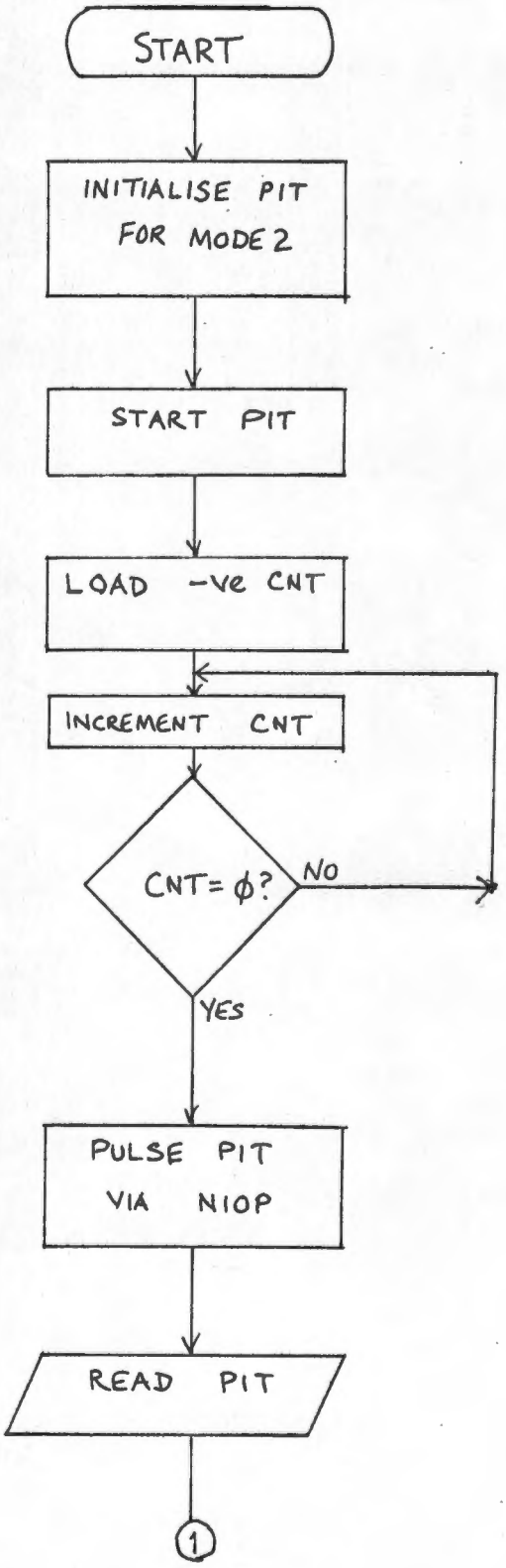
The interrupt testing routines are given at the start of the ETIM.SR program.

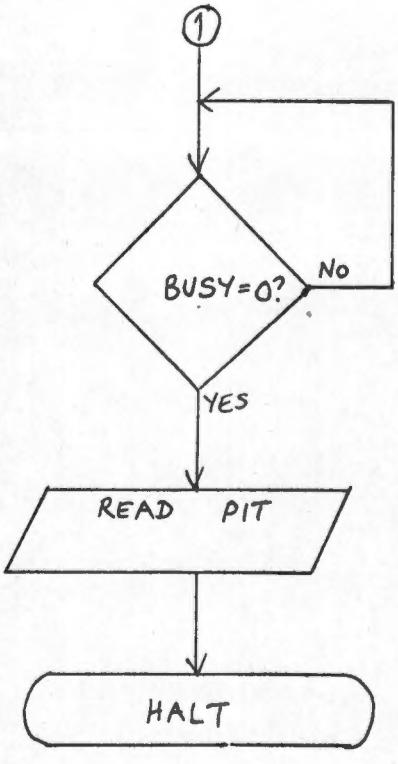
This program will halt if there are any faults in the interrupt circuit.

PIT. SR : PULSED TIMER IN COUNT DOWN MODE

The PIT is initialised for Mode 2 and pulsed according to a counting cycle.

PIT.SR FLOW CHART



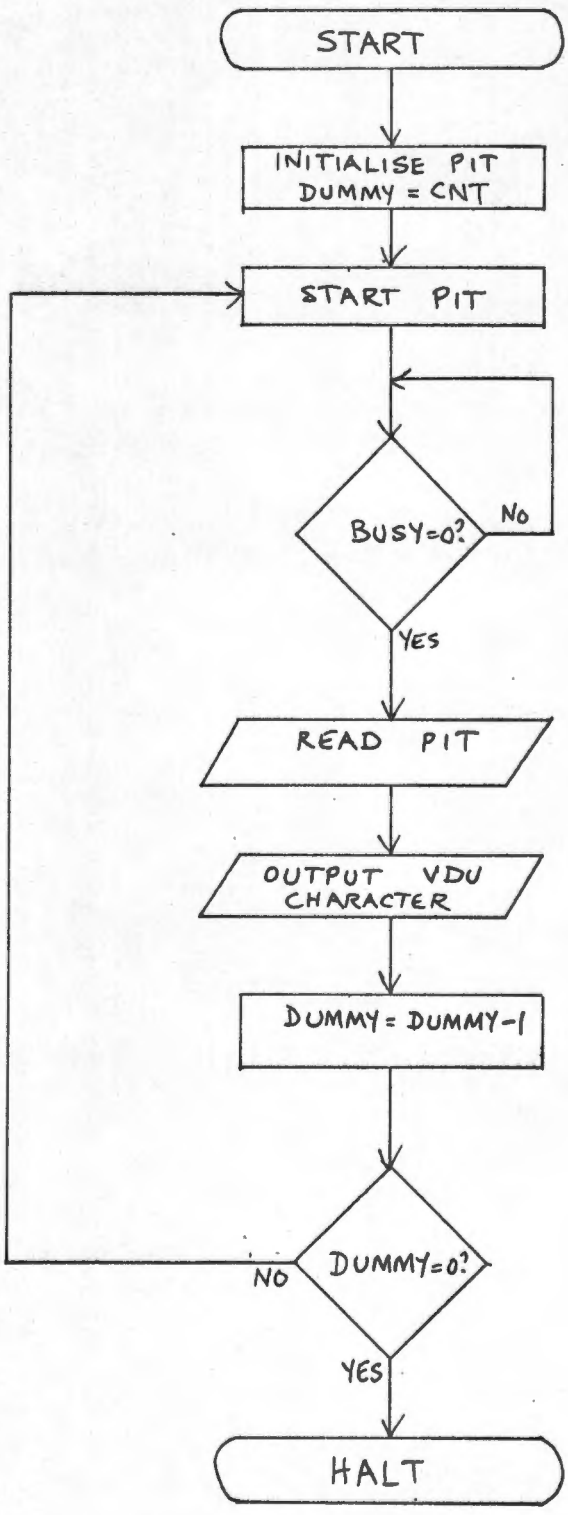


This program is used to test the NIOP pulsing feature of Mode 2 operation.

KPIT.SR : PROGRAM FOR DEMONSTRATING PIT INTERRUPTS

This program operates in both modes 1 and 2.
In mode 1 it stores the elapsed time at incrementing addresses in RAM. Each time an event occurs (Mode 1) or the clock overruns (mode 2), the program puts out a character onto the VDU screen

KPIT. SR FLOWCHART



INTV. SR : PROGRAM TO MONITOR MULTIPLE ELECTRODES FOR OCCURRENCE OF NEURONAL SPIKES

This program monitors the Digital Interface of the microNova. When a zero to one transition occurs on any one of the 16 input lines, ie a spike event (EXTERNAL INTERRUPT) occurs, the digital input status and the elapsed time since the last event are stored as a couplet in a buffer.

This program is vitually the same as the one for ETIM.SR. The only difference is that when the PIT is read for each event, the current status of the digital interface is also stored.

The listings of the test programs are found in Appendix D .

CHAPTER 8

DATA COLLECTION

PITCB.SR and PTCONT.SR are two programs that were developed for collecting spike train data. Both these programs monitor a 16 line digital interface and whenever a transition occurs on any one of these lines, the digital input status and the elapsed time since the last event are written into a circular buffer.

PITCB.SR records data for a specific number of events as specified by a binary reader program called BINRDT.

PTCONT.SR records data over a specific time period as specified by a binary reader program BINRDC. Both binary reader programs were written by Dr. Rod Douglas and are part of a library of programs used in neuronal spike data analysis.

The PITCB.SR and PTCONT.SR have PIT initialisation routines that are used to select the PIT mode of operation and clock time-base.

INFPT.SR serves as an interrupt handler for the circular buffer and the digital interface and PIT user routines, ie PITCB.SR and PTCONT.SR.

PITCB.SR and PTCONT.SR, when requested, write data to a circular buffer. The circular buffer routine was written by Dr. Rod Douglas and is part of the library of programs used in neuronal spike data analysis. Data is read from the circular buffer to the host computer (Eclipse) via a BINRDT or BINRDC binary reader routines.

The user routines of PITCB.SR and PTCONT.SR are similar to that of the testing program ETIM.SR

Diagram of test setup?

PIT CALIBRATION TEST

The PIT was initialised for Mode 1 and time-base 100 usec. A Hewlett Packard Pulse Generator was then used to provide an input signal, representing a regular spike train, to the PIT. The pulse period of the pulse generator was varied and inter-pulse interval data was recorded for pulse periods of 10,16 msec.

49,46 msec and 100,52 msec as measured in a Monsanto Programmable counter/timer. These were stored in files PER10.16 PER49.46 and PER100.52 respectively.

1. DATA ACQUISITION OF A SPECIFIED NUMBER OF SPIKE EVENTS

This data was read using the BINRDT program with PITCB.SR. Graphs of Interval Duration versus Interval Number, the first order interval histogram and the auto correlation histogram were plotted for each of these pulse periods. The graphs are labelled with the file name being specified as the cell number.

These graphs are found in Appendix F . Due to problems with the computer real-time clock in the binary reader and circular buffer routines, the plots for short pulse periods show irregularities. However for higher pulse periods the plot of interval duration vs interval is constant as a value equivalent to the pulse period. The first order interval histogram shows a single spike at a interval length equivalent to the pulse period.

The single spike in the interval histogram and the high degree of correlation in the autocorrelation histogram show that the data originates from some deterministic process, viz. the pulse generator.

2. CONTINUOUS DATA ACQUISITION

Using the PTCONT.SR routine, continuous acquisition of generated pulse data was performed by the BINRDC binary reader. This was performed for pulse frequencies of 10 Hz and 40 Hz. Graph of event frequency vs time were plotted over 4 mins for each of the above frequencies.

The average event frequency as seen from the graphs compare well to the actual setting of the pulse generator, as measured on the Monsanto Programmable Counter/Timer.

SPIKE TRAIN DATA COLLECTION RUN

Using the PITCB.SR routine as in the PIT calibration test, data was collected from a neuronal spike train input. The BINRDT binary reader was used to load 2048 couplets of data each time at time 0 mms, 5 mins, 10 mins, 15 mins, 40 mins.

These batches of data were stored in files DATA0, DATA5, DATA 10 DATA 15 and DATA 40 respectively. Plots of the graphs mentioned under the PIT calibration test were plotted for these sets of data. The resultant plots are shown in Appendix G .

The graphs are labelled with the data file name being specified as the cell number. The tall spike in the first order histogram plots represent the firing of a single cell. Any smaller peaks are due to the discriminator setting not being optimal.

All the graphs discussed are generated by a program called CELLREPORT. The theory behind these graphs have been discussed under point process analysis in Chapter 1 .

CONCLUSIONS

The total cost of the PIT was less than R150.00. It is impossible to buy a machine as powerful as the PIT for that price.

When comparing period measurements using the expensive Monsanto Programmable Counter/timer to that using the PIT, the PIT proved to be as efficient. This can be seen from the calibration graphs.

The only problems experienced with the PIT at the moment is that of measuring very short intervals. This is not due to the PIT, but to the supporting Software which will have to be refined.

REFERENCES

1. Wyss,U.R., and Handwerker,H: Stap-12: A Library system for on-line assimilation and off-line analysis of Event/time Data. Computer Progr. Biomed.1,209-218 (1971).
2. Johnson, D.H. The Relationship of post-stimulus time and Interval Histogram to the timing characteristics of spike trains. Biophys.J.22, 413-430 (1978).
3. Wyss, U.R., Statp-8: A Subroutine package for statistical on-line analysis of simultaneous neurol spike trains.
4. Caron,F., and Herzog, R.F: A new programmable timer designed for Pulsed NMR. J. Magn.Reson. 31, 357-362 (1978).
5. Brown P.B., Duffy, D., and McIntyre, T.W: Three Computer interfaces for neurophysiologists. In: Computer technology in neuroscience. Brown, P.B.(ed), pp 569-590. New York: Wiley and Sons 1976.
6. Rhode, W.S., and Soni,V.: Neural unit data analysis system. In: Computer technology in neuroscience. Brown, P.B.(ed), pp 253 -270. New York. Wiley and Sons (1976).
7. Sanderson, A.C., and Kobler,B., Sequential interval histogram analysis on non-stationary spike train data. In: Computer technology in neuroscience. Brown, P.B.(ed) pp 271-291. New York: Wiley and Sons (1976).
8. Glaser, E.M: Separation of neuronal activity by waveform analysis. Adv. Biomed. Eng.1, 77-135 (1970).
9. "MicroNova Computer Systems Technical Reference". Data General Corporation, Southboro, Mass. 1976
10. "Nova and Eclipse Line Computers Interface Designers' Reference". Data General Corporation, Southboro, Mass. 1975.
11. INS8253 Programmable Interval Timer Data Sheet, National Semiconductor Corp., Santa Clara, Calif. 1978.
12. Graf, R.F: Modern Dictionary of Electronics. Howard W. Sams and Co., Indianapolis. 1972.

13. De Jong, M.L., Titus, C.A, Larsen, D.G., Rony, P.R, Titus J.A: The 8253 programmable interval timer. Pulse. December 1978.

14. Larsen, D.G., De Jong, M.L., Rony, P.R., Titus, J.A., Titus,C.A: Demonstration program for the 8253 timer. Pulse. January. 1979.

15. Moore, G.P., Perkel, D.H., Segundo, J.P., Statistical analysis and functional interpretation of neuronal spike data. Ann. Rev. Physiol. 28 (1969).

16. Glaser, E., Ruchin, D.J., Principals of Neurobiological Signal Analysis. Academic Press Inc., New York. 1976.

17. Moore, G.P., Segundo, J.P., Perkel, D.H., Levitan, H: Statistical signs of synaptic interaction in neurons. Biophys.J. 10, 876-900. (1970).

18. Gerstein, G.L., Perkel, D.H: Mutual temporal relationships among neuronal spike trains. Biophys.J.12, 453 - 473. (1972).

19. Arnett, D.W. and Ellert, B.M: A real time cross correlator for neurophysiological research. IEEE T-BME. Jan 65-70(1976).

20. Silverman, G., Eisenberg,L.: Programmable Parallel timing system. IEEE T-BME. May. 201-205. (1971).

21. Nolte, J., Tarby, T.J.; An inexpensive Modular Pulse Generating System. IEEE T-BME. Sept. 480-482 (1977).

22. Sabah, N.H: A presettable multichannel digital timer. J. Appl. Physiol. 38, 757-759 (1975).

23. Sabah, N.H: A digital display of neuronal average firing frequencies. J.Appl. Physiol. 41, 98-100. (1976).

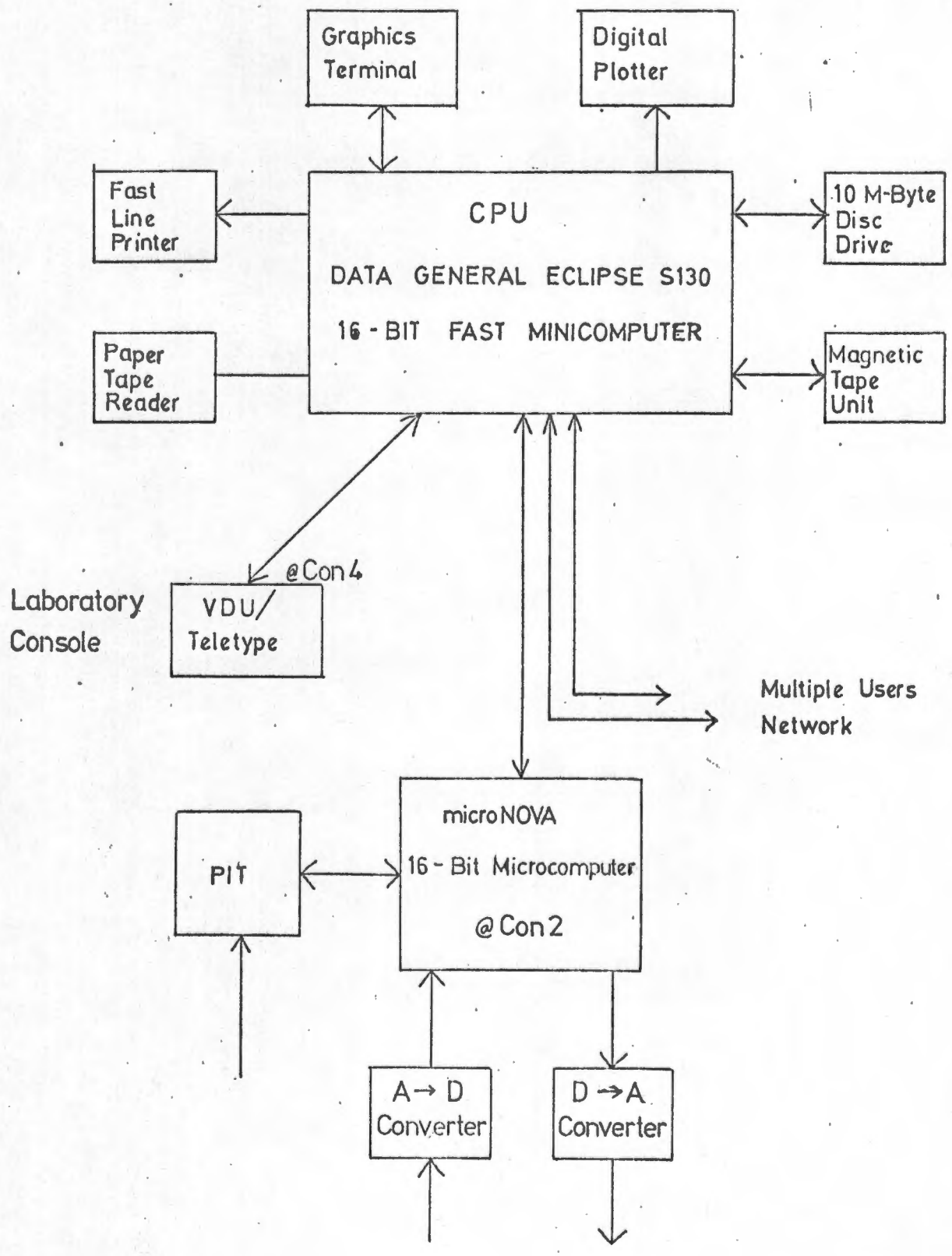
24. Wyss, C.R: A high accuracy linear rate meter. J. Appl. Physio. 39, 327-330 (1975).

25. Waterfall, R.C: Digital Interface Design. Lecture Notes,UMIST (1979).

APPENDIX A.

MEDICAL SCHOOL COMPUTER SYSTEM

MEDICAL SCHOOL COMPUTER SYSTEM



APPENDIX B.

INS8253 DATA SHEETS

National Semiconductor

JULY 1978

INS8253 Programmable Interval Timer

General Description

The INS8253 is a programmable timer/counter device contained in a standard, 24-pin dual-in-line package. The chip, which is fabricated using N-channel silicon gate technology, provides counting or time-out services in a microcomputer system. The various operating modes and other functional characteristics of the INS8253 are programmed by the system software.

The INS8253 provides three independent 16-bit down counters, each of which is capable of count rates in the range DC to 2MHz. Through software initialization, each counter can be made to operate in any one of six modes. The modulus and counting system used are also specified by system software. The operating characteristics of any individual counter can be modified by the software at any time to meet changing system requirements.

The modulus of any given counter can be changed at the program's discretion by loading a new value into the counter. A counter load operation may be limited to the counter's least significant byte or to its most significant byte, or it may revise both halves of the counter.

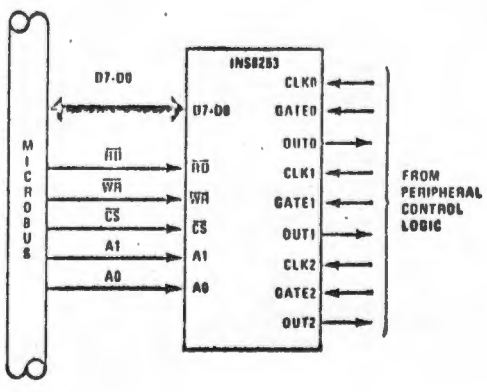
Count sequences may be in either binary or BCD. This choice is also individually specified for each counter by the software.

The contents of each counter may be read either directly or through an auxiliary register. A direct reading of the counter can be made whenever the counter is inhibited from counting. A count value can also be read without interfering with the counting process. This is done by transferring the counter's current value to an auxiliary register and then reading that register. This counter-to-register transfer can be executed without affecting the normal count sequence.

Features

- 3 Individually Programmable 16-Bit Counters
- 6 Operating Modes
- DC to 2MHz Count Rates
- Individual Count Rate and Modulus for Each Counter
- Selectable Counting System (Binary or BCD) for Each Counter
- TRI-STATE® TTL Drive Capability for Bidirectional Data Bus
- Single +5 Volt Power Supply
- 24-Pin Dual-In-Line Package
- MICROBUS™* Compatible

INS8253 MICROBUS Configuration



*A trademark of National Semiconductor Corporation.

DA-S25M78/Printed in U.S.A.

Absolute Maximum Ratings

Ambient Temperature Under Bias ..... 0°C to +70°C
Maximum Voltage to Any Input
with Respect to GND ..... −0.5V to +7V
Storage Temperature ..... −65°C to +150°C
Power Dissipation ..... 1 Watt

Note: Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended and should be limited to those conditions specified under DC Electrical Characteristics.

DC Electrical Characteristics

(T_A = 0°C to +70°C; V_{CC} = 5V ± 5%)

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage	2.2	V_{CC} + 0.5V	V	
V_{OL}	Output Low Voltage		0.45	V	Note 1
V_{OH}	Output High Voltage	2.4		V	Note 2
I_{IL}	Input Load Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} to 0V
I_{CC}	V_{CC} Supply Current		140	mA	

Note 1: INS8253, I_{OL} = 1.6mA.
Note 2: INS8253, I_{OH} = −150μA.

Capacitance

T_A = 25°C; V_{CC} = GND = 0V.

Symbol	Parameter	Min	Max	Unit	Test Conditions
C_{IN}	Input Capacitance		10	pF	f_C = 1MHz
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured pins returned to V_{SS}

AC Electrical Characteristics

T_A = 0°C to +70°C; V_{CC} = 5.0V ± 5%; GND = 0V

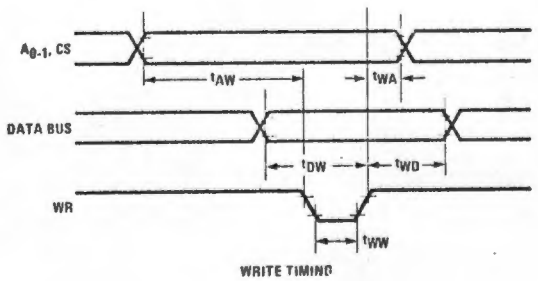
Bus Parameters:

Symbol	Parameter	Min	Max	Unit
READ CYCLE				
t_{AR}	Address Stable Before $\overline{READ}$	50		ns
t_{RA}	Address Hold Time for $\overline{READ}$	5		ns
t_{RR}	$\overline{READ}$ Pulse Width	400		ns
t_{RD}	Data Delay from $\overline{READ}$ (Note 2)		300	ns
t_{DF}	$\overline{READ}$ to Data Floating	25	125	ns
WRITE CYCLE				
t_{AW}	Address Stable Before $\overline{WRITE}$	50		ns
t_{WA}	Address Hold Time for $\overline{WRITE}$	30		ns
t_{WW}	$\overline{WRITE}$ Pulse Width	400		ns
t_{DW}	Data Setup Time for $\overline{WRITE}$	300		ns
t_{WD}	Data Hold Time for $\overline{WRITE}$	40		ns
t_{RV}	Recovery Time Between $\overline{WRITE}$s	1		ns

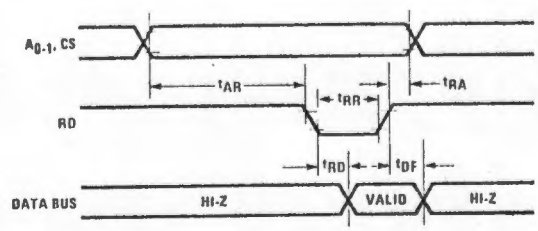
Note 1: AC timings measured at V_{OH} = 2.2V, V_{OL} = 0.8V.

Note 2: Test conditions: INS8253, C_L = 100pF.

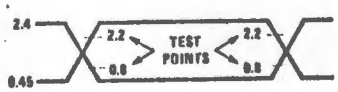
Input Waveforms for AC Tests



WRITE TIMING



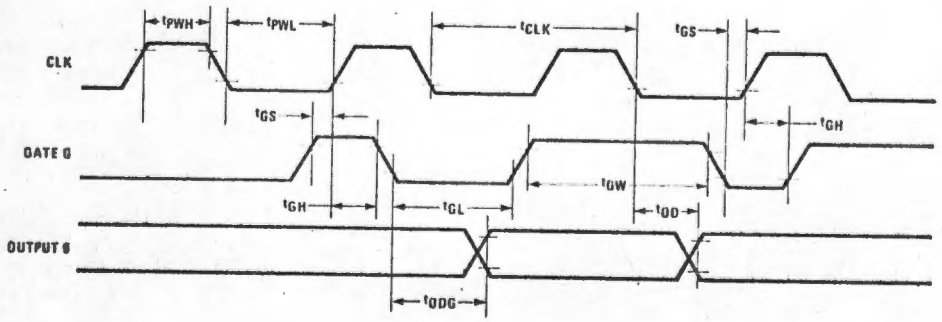
READ TIMING



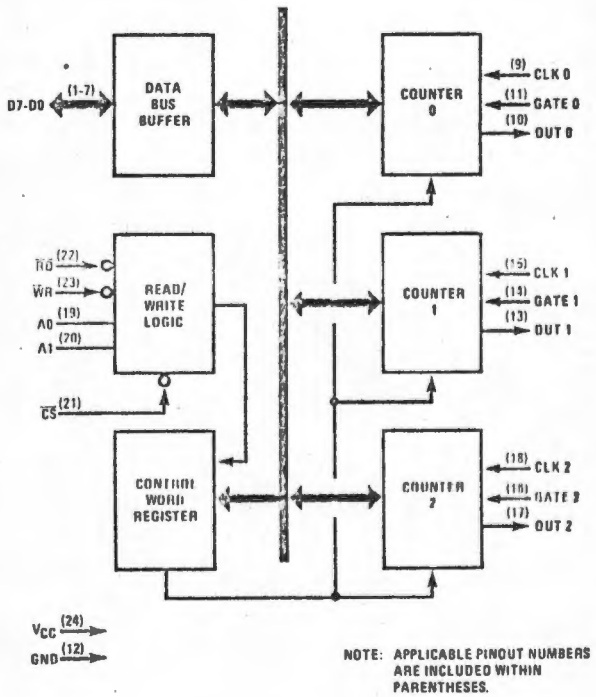
Clock and Gate Timing

Symbol	Parameter	Min	Max	Unit
t_{CLK}	Clock Period	380	DC	ns
t_{PWH}	High Pulse Width	230		ns
t_{PWL}	Low Pulse Width	150		ns
t_{GW}	Gate Width High	150		ns
t_{GL}	Gate Width Low	100		ns
t_{GS}	Gate Setup Time to CLK ↑	100		ns
t_{GH}	Gate Hold Time After CLK ↑	50		ns
t_{OD}	Output Delay From CLK ↓ (Note 1)		400	ns
t_{ODG}	Output Delay From Gate ↓ (Note 1)		300	ns

Note 1: Test conditions: INS8253, C_L = 100pF.



INS8253 Functional Block Diagram



NOTE: APPLICABLE PINOUT NUMBERS ARE INCLUDED WITHIN PARENTHESES.

INS8253 Functional Pin Description

The following describes the functions of all INS8253 input/output pins. Some of these descriptions refer to internal circuits.

NOTE

In the following descriptions, a low represents a logic 0 (0 Volt, nominal) and a high represents a logic 1 (+2.4 Volts, nominal).

INPUT SIGNALS

Chip Select ($\overline{CS}$): When low, the chip is selected. This enables communication between the INS8253 and the microprocessor.

Read ($\overline{RD}$): When low, allows the microprocessor to read contents of counter specified by A0, A1.

Write ($\overline{WR}$): When low, writes control word into control word register or loads new count value into selected counter. Destination of data (control word register or counter 0, 1 or 2) is specified by A0, A1.

A0, A1: These inputs are used to select one of the counters for reading or writing or to select the control word register for writing. A0, A1 may be controlled via address bus lines.

Clock (CLK0–CLK2): Each counter has a separate clock input that drives the counter.

Gate (Gate 0–Gate 2): Each counter is individually controlled by a separate Gate input (1 = enable, 0 = inhibit). In some modes, the positive edge of Gate is used to initiate the counting process. Specific use of Gate depends on the counter's operating mode. Details are provided in the section entitled INS8253 Programming.

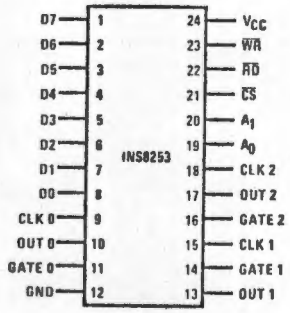
OUTPUT SIGNALS

Output (Out 0–Out 2): Each counter has a single output that indicates whether or not the counter has reached its terminal count. Specific operation of this output depends on the counter's mode. Details are provided in the section entitled INS8253 Programming.

INPUT/OUTPUT SIGNALS

Data (D7–D0) Bus: This bus, which comprises eight TRI-STATE input/output lines, provides for bidirectional communication between the INS8253 and the microprocessor. Control words and count value bytes are transferred over these lines.

INS8253 Pin Configuration



INS8253 Programming

This section provides basic information for programming the INS8253 and describes the methods for reading counter status. Table 1 summarizes the control signals needed to write command words and new count values into the INS8253 and to read the contents of individual counters.

Table 1. Bus Control for INS8253 I/O Operations

Output Operations	$\overline{CS}$	$\overline{WR}$	$\overline{RD}$	A1	A0
LOAD COUNTER 0	0	0	1	0	0
LOAD COUNTER 1	0	0	1	0	1
LOAD COUNTER 2	0	0	1	1	0
WRITE CONTROL WORD	0	0	1	1	1
Input Operations					
READ COUNTER 0	0	1	0	0	0
READ COUNTER 1	0	1	0	0	1
READ COUNTER 2	0	1	0	1	0

WRITING CONTROL WORDS

Each counter's mode and counting system (binary or BCD) are specified by an eight-bit control word. See figure 1. An I/O write operation with A0, A1 = 11 will load the control word into the control word register. The control word contains four fields:

- D7, D6 (SC1, SC0) — This field specifies which counter will be affected by the other control fields.
- D5, D4 (RL1, RL0) — A bit pattern of 00 in this field causes the contents of the selected counter to be latched in an auxiliary register. The count value can then be read without inhibiting the counter. The other three bit patterns specify which byte(s) of the selected counter will be affected by any subsequent read/write operations addressed to that counter.
- D3, D2, D1 (M2, M1, M0) — This field specifies the mode of operation for the selected counter.
- D0 (BCD) — This one-bit field specifies the counting system to be used by the selected counter.

Any time after a counter is initialized by a control word, its initial count value can be loaded. This is done by means of a write operation addressed to that counter. Details are given in the section entitled Loading Initial Count Value.

Programming of the three counters can be executed in any sequence, with only two requirements.

1. A counter must be issued a control word before it is given an initial count value.
2. Read and write operations addressed to a counter must conform to the byte-selection rules specified by the RL1, RL0 field in the control word. For example, if the counter's RL1, RL0 bits = 10, subsequent counter load operations addressed to that counter must be intended for the most significant byte only.

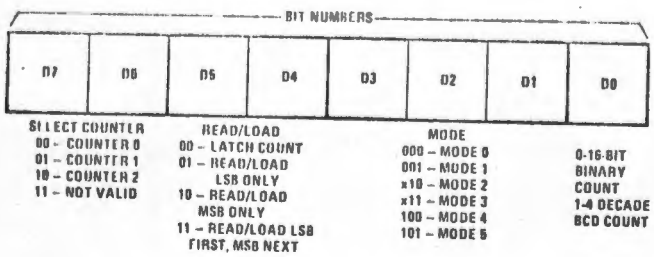


Figure 1. Control Word Format

COUNTER MODE DESCRIPTIONS

Figure 2 provides timing information for the six INS8253 operating modes.

- Mode 0, Timed Interrupt — In this mode, OUT goes low when the mode is set. The counter begins counting CLK cycles when the count is loaded. OUT remains low until the terminal count is reached, at which point it goes high and remains high until either the mode or count is reloaded. The gate input will inhibit the count when low.

  If the counter is loaded with a new value during a count cycle, counting will stop when the first byte is loaded and will begin decrementing from the new value after the second byte is loaded.

- Mode 1, Retriggerable One Shot — In this mode, OUT goes low on the first CLK after a rising transition on Gate. OUT goes high again on the terminal count.

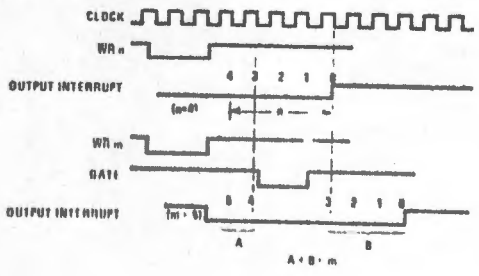
  Gate can be used to retrigger the counter. Each positive transition of Gate causes the counter to begin decrementing from the initial count value.

  If a new initial count value is loaded during a count cycle, the new value will not take effect until the next rising transition of Gate.

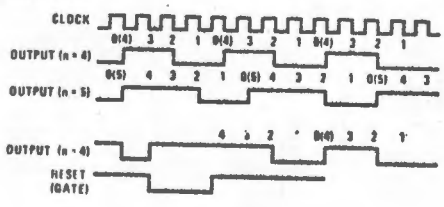
- Mode 2, Rate Generator — In this mode, OUT goes low for one CLK cycle at the end of each count sequence. The leading edge of each pulse occurs at the start of the terminal CLK cycle. The counter will repeat count sequences as long as Gate remains high. Any positive transition of Gate will start a new count sequence at the initial count value. This allows the counter to be synchronized by Gate.

  If a new initial count value is loaded during a count sequence, the current sequence will run to completion and the following sequence will then start at the new initial count value.

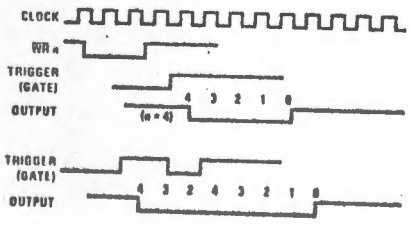
MODE 0



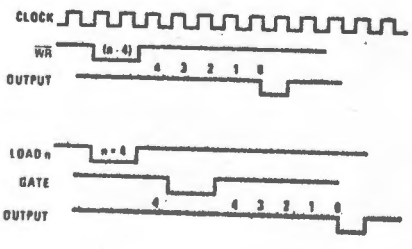
MODE 3



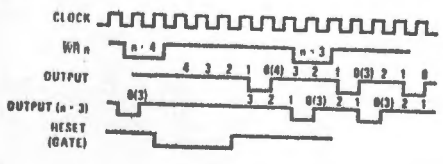
MODE 1



MODE 4



MODE 2



MODE 5

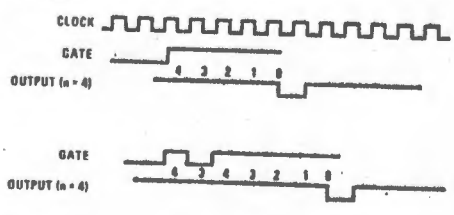


Figure 2. Mode Timing Waveforms

- Mode 3, Square Wave Generator — In this mode, the counter generates a square wave signal at the OUT pin so long as Gate remains high. The period of the square wave is equal to one count cycle. If the initial count value is even, OUT will be high for the first half of each count sequence and low during each second half. For an odd count, OUT is high for (N + 1)/2 counts and low for (N – 1)/2 counts.

  If a new initial count value is loaded during a count sequence, the current sequence will run to completion and the following sequence will then start at the new initial count value.

  Any positive transition of Gate will start a new count sequence at the initial count value. This allows the counter to be synchronized by Gate.

- Mode 4, Software Triggered Strobe — In this mode, OUT is normally high and goes low for one CLK cycle after the terminal count is reached. Counting is enabled when Gate is high. Counting is initiated by loading the modulus.

  If a new initial count value is loaded during a count sequence, the current sequence will run to completion and the following sequence will then start at the new initial count value.

  A low on the Gate input inhibits the count.

- Mode 5, Hardware Triggered Strobe — In this mode, any positive transition of Gate will initiate a new count sequence. OUT then goes low for one CLK cycle when the terminal count is reached.

LOADING INITIAL COUNT VALUE

Each counter's modulus is determined by presetting the counter to the desired value. This is done by means of one or two I/O write operations with A1, A0 selecting the counter to be preset. The write operation loads the contents of the data bus (D7-D0) into the upper or lower half of the selected counter, as determined by the control word's RL1, RL0 field. Figure 3 summarizes the various counter loading conditions.

After a counter's initial count value is loaded, it is ready for operation in the specified mode. It begins counting CLK cycles when its Gate input goes high. Each CLK decrements the enabled counter by one until the full count cycle has been completed.

The initial count value of any counter can be changed by loading a new value into the counter's:

- LSB only (RL1, RL0 = 01),
- MSB only (RL1, RL0 = 10), or
- LSB first, and then MSB (RL1, RL0 = 11).

Read/Load Conditions		Effect of Subsequent Write Operation
RL1	RL0	
0	1	$\overline{WR}$ loads D7-D0 into LSB of counter selected by A1, A0.*
1	0	$\overline{WR}$ loads D7-D0 into MSB of counter selected by A1, A0.
1	1	First $\overline{WR}$ loads D7-D0 into LSB of counter selected by A1, A0. Next $\overline{WR}$ loads D7-D0 into counter's MSB.

*A1	A0	
0	0	Selects Counter 0
0	1	Selects Counter 1
1	0	Selects Counter 2

Figure 3. Initial Count Loading Summary

READING COUNT VALUES

The current status of a count sequence can be examined at any time by the program. This can be done either by reading the counter contents directly or by latching the counter contents into an auxiliary register and then reading that register.

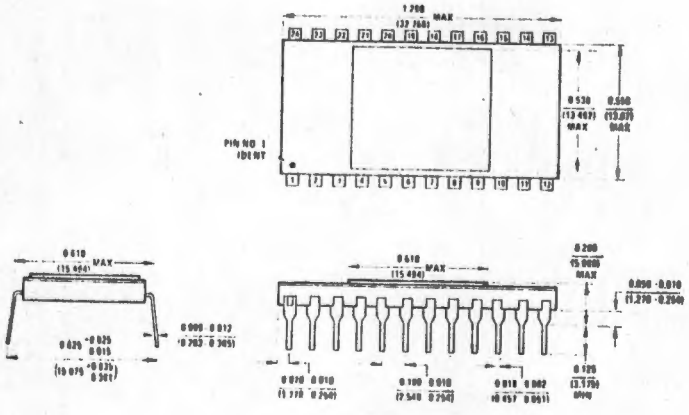
A counter can be read directly with the following bus conditions:

	$\overline{RD}$	A1	A0
To Read Counter 0	0	0	0
To Read Counter 1	0	0	1
To Read Counter 2	0	1	0

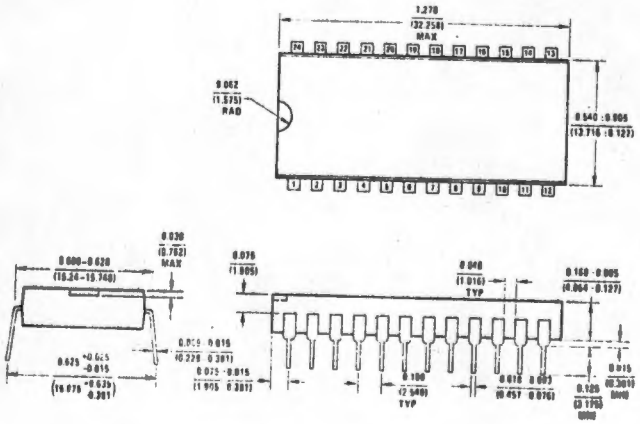
The count should remain stable during direct reading of a counter. Stability is assured by holding the Gate input low or inhibiting the CLK input (by means of external logic) for the duration of the read operation. Counter status can also be sampled without inhibiting the count sequence. This is done by issuing a control word to the counter with RL1, RL0 = 00, followed by an I/O read of that counter's location. The RL1, RL0 bits cause the contents of the addressed counter to be latched into the auxiliary register. The subsequent read operations access the auxiliary register.

When reading either a counter or the auxiliary register, the read operation must follow the format programmed for that counter by RL0 and RL1. Note that issuing a latch command of RL1, RL0 = 00 does not alter the previously programmed RL0 and RL1.

Physical Dimensions
inches (millimeters)



24-Lead Hermetic DIP (D)
NS Package Number D24A



24-Lead Molded DIP (N)
NS Package Number N24A

National Semiconductor Corporation
2900 Semiconductor Drive
Santa Clara, California 95051
Tel: (408) 737-5000
TWX: (910) 339-9240

National Semiconductor GmbH
8000 München 21
Eisenheimerstrasse 61/2
West Germany
Tel.: 089/9 15027
Telex: 05-22772

NS International Inc., Japan
Miyake Building
1-9 Yotsuya, Shinjuku-ku 160
Tokyo, Japan
Tel: (03) 355-3711
TWX: 232-2015 NSCJ-J

National Semiconductor (Hong Kong) Ltd.
8th Floor,
Cheung Kong Electronic Bldg
4 Hing Yip Street
Kwun Tong
Kowloon, Hong Kong
Tel.: 3-411241-8
Telex: 73866 NSEHK HX
Cable: NATSEMI

NS Electronics Do Brasil
Avda Brigadeiro Faria Lima 844
11 Andar Conjunto 1104
Jardim Paulistano
Sao Paulo, Brasil
Telex:
1121008 CABINE SAO PAULO

NS Electronics Pty. Ltd.
Cnr. Stud Rd. & Mtn. Highway
Bayswater, Victoria 3153
Australia
Tel.: 03-729-6333
Telex: 32096

APPENDIX C.

MICRONOVA GPI SPECIFICATIONS

SUMMARY OF GPIO BUS SIGNALS

The fifty-five signals which comprise the GPIO bus can be divided into five groups:

Data

D(0-15)H Data Out. All data for both data channel and programmed I/O are transferred from the IOC to the device interface via these 16 lines. Each line is buffered to drive 10 standard TTL loads. The contents of the polarity bit (controlled by jumper W5) determines whether a low level should be interpreted as a 0 or a 1.

D(0-15)L Data Input. All data and addresses for both data channel and programmed I/O are transferred from the device interface to the IOC via these 16 input lines. The interrupt disable mask bit is determined by one of these lines when the MSKO signal is asserted. The device code, external register select bit, and the polarity bit are carried on these lines when the signal **IORST** is asserted (see IORST, MSKO, and Jumpers). The device interface should drive these lines with open collector gates. The contents of the polarity bit determines whether a low level should be interpreted as a 0 or a 1.

Programmed I/O

The following control signals are asserted low (1=0V). They can drive up to 10 TTL loads.

DIA Data In A. Asserted by the IOC upon receipt of its DIA instruction.To be used by the device interface to place the contents of its A input buffer on **D(0-15)L**.

DIB Data In B. Asserted by the IOC upon receipt of its DIB instruction.To be used by the device interface to place the contents of its B input buffer on **D(0-15)L** if external registers are enabled (see Jumpers).

DIC Data In C. Equivalent to DATIA, except that it applies to the C input buffer.

DOA Data Out A. Asserted by the IOC upon receipt of its DOA instruction.To be used by the device interface to load the contents of **D(0-15)H** into its A output buffer.

DOB Data Out B. Asserted by the IOC upon receipt of its DOB instruction. To be used by the device interface to load the contents of **D(0-15)H** into its B output buffer

DOC Data Out C. Equivalent to DATOB, except that it applies to the C output buffer.

STRT Start. Asserted by the IOC upon the receipt of any of its non-skip I/O instructions in which bits 8 and 9 = 01 (i.e., instructions in which the S control function is specified). Asserted during DIA, DIB, DIC, DOA, DOB, and DOC instructions after the data transfer has occurred. Usually used to initiate the device interface by setting the Busy flag to 1 and the Done flag to 0.

CLR Clear. Asserted by the IOC upon the receipt of any of its non-skip I/O instructions in which bits 8 and 9 = 10 (i.e., instructions in which the C control function is specified). Asserted during DIA, DIB, DIC, DOA, DOB, and DOC instructions after the data transfer has occurred. Usually used to terminate device operation by setting the Busy and Done flags to 0.

IOPLS I/O Pulse. Asserted by the IOC upon receipt of any of its non-skip I/O instructions in which bits 8 and 9 = 11 (i.e., instructions in which the P control function is specified). Asserted during DIA, DIB, DIC, DOA, DOB, and DOC instructions after the data transfer has occurred. Usually used to initiate special device operations.

SET BUSY Asserted by the interface when it is busy and should not be disturbed by the IOC. Sets the Busy flag in the IOC to 1.

SET DONE Asserted by the interface to notify the IOC that it has completed its operation. In the IOC it sets the Done flag to 1 and the Busy flag to 0.

Program Interrupt

INT SYNC Interrupt Synchronize. Asserted by the interface to notify the program that it has completed its operation. In the IOC it directly initiates a program interrupt request without disturbing either the Busy or Done flags.

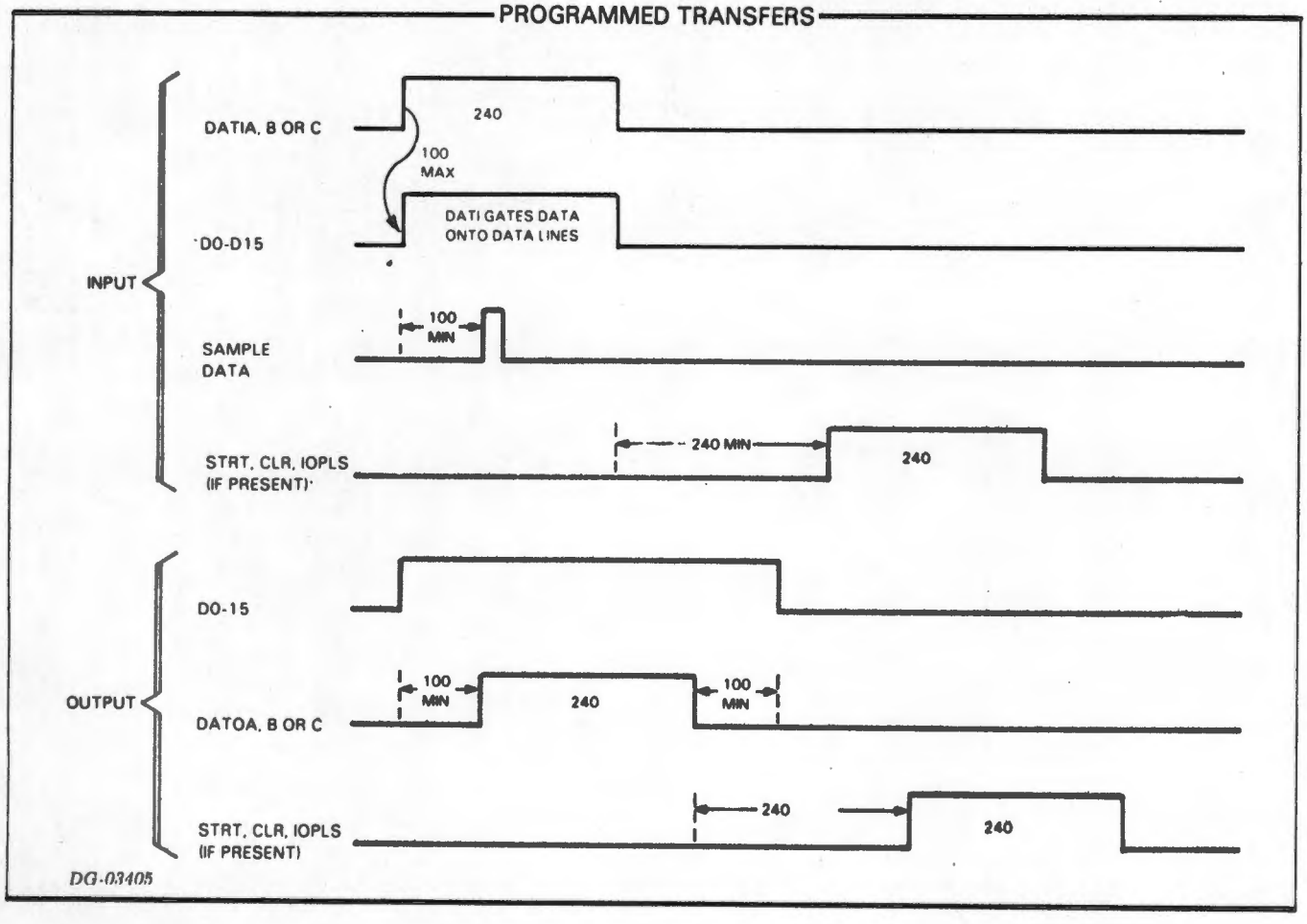
MSKO Mask Out. Asserted by the IOC during the execution of a MSKO instruction. Loads the selected Data line into the priority mask bit register.

Data In - The I/O controller asserts **DATIA, DATAIB,** or **DATIC.** It also asserts **STRT, CLR** or **IOPLS** if they are specified by the I/O instruction. When the signal **SAMPLE DATA** occurs, any data on the GPIO data bus (D0-D15)L lines will be gated from the interface, through the IOC, and onto the I/O data bus. This signal, however, is internal to the IOC and is shown for reference only.

Data Out - The I/O controller (IOC) places the data received from the CPU onto the GPIO data bus **D(0-15)H** lines and asserts **DATOA, DATOB,** or **DATOC.** It also asserts **STRT, CLR** or **IOPLS** if they are specified by the I/O instruction.

Data Channel Transfers

An information transfer occurring under data channel control moves a block of data, one word at a time, between the IOC and the device.



Jumper W5 selects the polarity bit. The polarity bit is a 1-bit register that determines the sense of the data bits transmitted and received via the IOC. If W5 is in, the polarity bit is set to a 1 and a low level (0V) on the data pins of the IOC is interpreted as a 0. A high level (5V) is interpreted as a 1. If W5 is out, the polarity bit is set to a zero and a low level on the data pins of the IOC is interpreted as a 1. The high level is interpreted as a 0. Note that the buffered outputs, D(0-15)H, are inverted.

W5	DATA POLARITY (GPIO BUS)	
IN	D(0-15)H	ZERO=+5V, ONE= 0V
OUT		ZERO= 0V, ONE=+5V
IN	D(0-15)L	ZERO= 0V, ONE=+5V
OUT		ZERO=+5V, ONE= 0V

Jumper W4 controls the selection of the external (device interface) or internal (IOC) memory address and word count registers.

W4	Location of Registers
IN	DEVICE INTERFACE
OUT	IOC

The interrupt priority mask bit is selected by jumpering the mask signal (MSKO, pin 44) to one of the D(0-15)L lines.

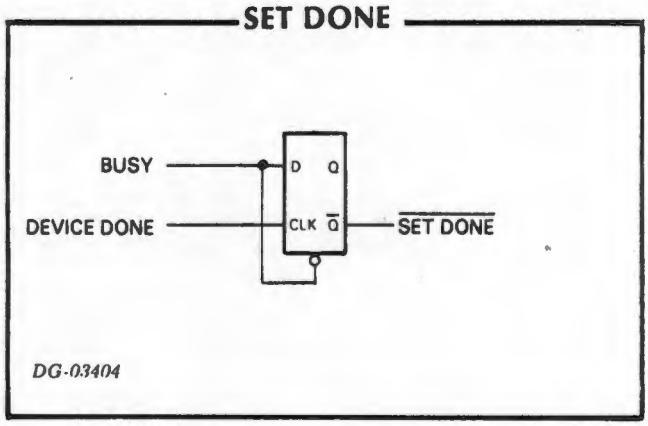
Data Lines and Drive Capability

The outputs of the I/O controller (IOC) chip are capable of driving only 1 TTL load. Therefore, all the data out lines, D(0-15)H have been TTL buffered, and are capable of sinking 16mA. The outputs of the 4 to 16 decoder are also capable of sinking 16mA. The data input lines, D(0-15)L, should be driven with open collector drivers. Each control signal to the IOC (INTSYNC, pin 23; DCHSYNC, pin 22; SET BUSY, pin1; and SET DONE, pin 2) constitute 1 TTL input load.

The supply voltages required (+5Vdc, pin 58; +15Vdc, pin 57; and -5Vdc, pin 51) must be supplied to the board by the system into which it is installed. See the section on Power Supply Assemblies for proper supply voltage sequencing. The maximum current drain on the +5Vdc should be 1 ampere.

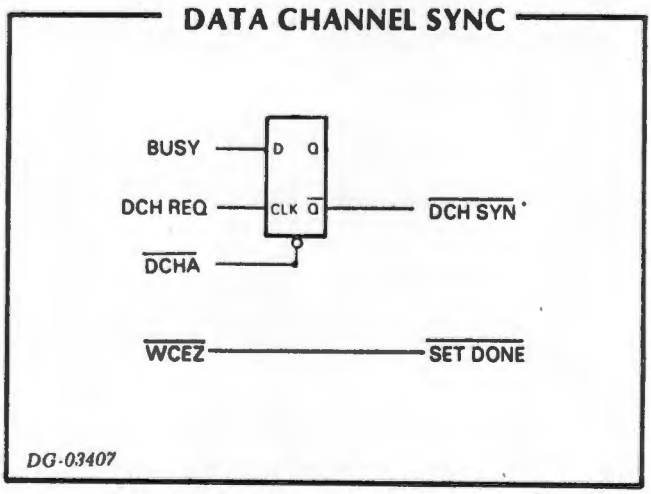
Busy/Done

A suggested circuit for generating the SET BUSY and SET DONE signals is given below.



Interface Wire Wrap Pins

Wire wrap pins are provided in the IOC section of the model 4211 board to facilitate the connection of the GPIO bus to the custom device controller. Below are listed the wire wrap pins associated with each bus signal. The location of the pins may be found by referring to the physical layout of the board. The model 4210 GPIO board does not include wire wrap pins, but features etched circuit holes in the same locations.



WIRE WRAP PINS (GPIO BOARD)

PIN	SIGNAL	PIN	SIGNAL
1	$\overline{\text{SET BUSY}}$	30	$\overline{\text{DONE}}$
2	$\overline{\text{SET DONE}}$	31	D9H
3	$\overline{\text{MASTER CLOCK}}$	32	D9L
4	D12H	33	D13H
5	D12L	34	D13L
6	D11H	35	D14H
7	D11L	36	D14L
8	D10H	37	D15L
9	D10L	38	D15H
10	D4H	39	$\overline{\text{DIB}}$
11	D4L	40	$\overline{\text{DOA}}$
12	D3H	41	$\overline{\text{CLR}}$
13	D7H	42	$\overline{\text{DCHA}}$
14	D7L	43	$\overline{\text{CLK}}$
15	D6H	44	$\overline{\text{MSKO}}$
16	D6L	45	$\overline{\text{DOC}}$
17	D5H	46	$\overline{\text{DIA}}$
18	D5L	47	$\overline{\text{STRT}}$
19	D3L	48	$\overline{\text{IORST}}$
20	D1H	49	$\overline{\text{WCEZ}}$
21	D1L	50	$\overline{\text{DCHO}}$
22	$\overline{\text{DCH SYN}}$	51	-5V
23	$\overline{\text{INT SYN}}$	52	$\overline{\text{BUSY}}$
24	D0H	53	$\overline{\text{DOB}}$
25	D0L	54	$\overline{\text{DIC}}$
26	D2H	55	$\overline{\text{OPLS}}$
27	D2L	56	$\overline{\text{DCHI}}$
28	D8H	57	+15V
29	D8L	58	+5V

DG-03408

APPENDIX D.

TESTING SOFTWARE

```
 0001  ETIME    AOS ASSEMBLER REV 02.03          14:26:59 08/20/80
                        .TITLE ETIME
 02                 ; PROGRAM NAME - ETIM.SR
 03                        .NREL
 04                 ; PROGRAM FOR TESTING PROGRAMMABLE INTERVAL
 05                 ; TIMER
 06                 ; PROGRAM WILL RUN AND HALT ON ERROR
 07                 ; CONSULT PROGRAM LISTING FOR ERROR DESCRIPTION
 08 00000'060205 ETIMT:   NIOC    5                    ; GENERAL RESET
 09 00001'063505          SKPBZ   5
 10 00002'063077          HALT                ; BUSY/DONE HARDWARE ERROR
 11 00003'063705          SKPDZ   5
 12 00004'063077          HALT                ; BUSY/DONE HARDWARE ERROR
 13 00005'020464          LDA     0,SETUP     ; LOAD CONTROL WORD
 14 00006'061005          DOA     0,5
 15 00007'000401          JMP     .+1
 16 00010'064405          DIA     1,5         ; CHECK CONTROL
 17 00011'063077          HALT                ; CHECKPOINT
 18                 ; TYPE 1A TO CHECK IF LSB IN AC1 EQUALS
 19                 ; LSB IN SETUP
 20 00012'030460          LDA     2,TIME      ; LOAD INITIAL TIME
 21 00013'072005          DOB     2,5
 22 00014'020457          LDA     0,FLY       ; LOAD READ ON FLY WORD
 23 00015'061005          DOA     0,5
 24 00016'000401          JMP     .+1
 25 00017'064405          DIA     1,5         ; CHECK CONTROL
 26 00020'063077          HALT                ; CHECKPOINT
 27                 ; TYPE 1A TO CHECK IF LSB IN AC1 EQUALS
 28                 ; LSB IN FLY
 29 00021'063505          SKPBZ   5
 30 00022'063077          HALT                ; ERRONEOUS START
 31 00023'063705          SKPDZ   5
 32 00024'063077          HALT                ; ILLEGAL INTERRUPT
 33 00025'020453          LDA     0,.DATA     ; LOAD DATA START ADDRESS
 34 00026'040020          STA     0,20
 35 00027'020447          LDA     0,TINT      ; LOAD TYPE OF INTERRUPT START ADDR
 36 00030'040022          STA     0,22
 37 00031'020446          LDA     0,XTA       ; LOAD VALUE OF OVERRUN START ADDRE
 38 00032'040023          STA     0,23
 39 00033'030441          LDA     2,DPTS      ; LOAD NUMBER OF DATA POINTS
 40 00034'050441          STA     2,DUMMY     ; CREATE DUMMY VARIABLE
 41 00035'152440          SUBO    2,2
 42 00036'060105 START:   NIOS    5           ; START PIT
 43 00037'063405          SKPBN   5
 44 00040'063077          HALT                ; START NOT BEING SET
 45 00041'063605          SKPDN   5           ; WAIT FOR INTERRUPT
 46 00042'000777          JMP     .-1         ; THEN PROCEED
 47 00043'064405          DIA     1,5         ; LOAD AC1 WITH PIT STATUS FLAGS
 48 00044'000401          JMP     .+1
 49 00045'046022          STA     1,@22       ; RECORD TYPE OF INTERRUPT
 50                 ; AN INTERRUPT HAS SET DONE=1
 51                 ; HAVE TO CHECK FOR SOURCE OF INTERRUPT
 52                 ; BY CHECKING STATUS FLAGS IN REGISTER-A
 53                 ; THIS IS DONE BY REPEATEDLY ROTATING
 54                 ; AC1 TO THE LEFT AND EXAMINING THE CARRY BIT
 55 00046'125103          MOVL    1,1,SNC ; SHIFT AC1 LEFT
 56                 ; CHECK IF CLOCK OVERRUN FLAG IS SET
```

```
57                ; IF NOT SHIFT AC1 LEFT AGAIN
58 00047'000403           JMP     .+3
59 00050'020422           LDA     0,TIME  ; LOAD OVERRUN TIME
60 00051'113000           ADD     0,2     ; TOTAL OVERRUN
```

```
 0002  ETIME
01 00052'125103          MOVL    1,1,SNC ; SHIFT AC1 LEFT
02                ; CHECK IF EXTERNAL INTERRUPT FLAG IS SET
03                ; IF NOT LOOP AND WAIT
04 00053'000763          JMP     START
05 00054'052023          STA     2,@23
06 00055'076405          DIC     3,5     ; LOAD AC3 FROM PIT
07 00056'020414          LDA     0,TIME  ; LOAD AC0 WITH INITIAL TIME
08 00057'162400          SUB     3,0     ; CALCULATE ELAPSED TIME
09 00060'143000          ADD     2,0     ; ADD IN OVERRUN TIME
10                ; RESULT IS PLACED IN AC0
11 00061'152440          SUBO    2,2     ; CLEAR AC2
12 00062'125102          MOVL    1,1,SZC ; SHIFT AC1 LEFT
13                ; CHECK IF LOST DATA FLAG IS SET
14                ; IF SET RESTART PIT
15 00063'000753          JMP     START
16 00064'042020          STA     0,@20   ; STORE AC0 AT INCREMENTING ADDRESS
17                ; CORRECTED TIME IS STORED
18 00065'102440          SUBO    0,0     ; CLEAR AC0
19 00066'014407          DSZ     DUMMY   ; ALL DATA POINTS LOADED?
20 00067'000747          JMP     START
21 00070'063077          HALT
22 00071'000000 SETUP: 0
23 00072'000000 TIME: 0
24 00073'000000 FLY: 0
25 00074'000012 DPTS: 10.
26 00075'000000 DUMMY: 0
27 00076'003777 TINT: 3777
28 00077'004777 XTA: 4777
29 00100'000100'. DATA:  DATA-1
30 00101'001750 DATA:    .BLK    1000.
31                       .END    ETIMT

**00000 TOTAL ERRORS, 00000 FIRST PASS ERRORS
```

PIT.SR

```
 0001  PINT      AOS ASSEMBLER REV 02.03          14:53:00 07/11/80
                          .TITL   PINT
02                        .NREL
03
04 00000'060205 PINT:     NIOC    5                ; CLEAR THE PIT
05 00001'020423           LDA     0,SETUP
06 00002'061005           DOA     0,5
07 00003'020422           LDA     0,FLY            ; LOAD THE COUNT ON FLY WOR
08 00004'061005           DOA     0,5
09 00005'024421           LDA     1,TIME
10 00006'066005           DOB     1,5              ; SEND THE START TIME
11 00007'060105 TIMR:     NIOS    5                ; START THE TIMER
12 00010'030417           LDA     2,CNT
13 00011'151404           INC     2,2,SZR
14 00012'000777           JMP     .-1
15 00013'060305           NIOP    5
16 00014'000401           JMP     .+1              ; WASTE SOME TIME
17 00015'072405           DIC     2,5
18
19                ; WAIT FOR END OF TIMING CYCLE
20
21 00016'000401           JMP     .+1
22 00017'063505           SKPBZ   5
23 00020'000777           JMP     .-1
24 00021'000401           JMP     .+1
25 00022'076605           DICC    3,5
26 00023'063077           HALT
27
28 00024'024064 SETUP:    024064           ; READ 2 BYTES,MODE 2,BINARY
29 00025'024000 FLY:      024000
30 00026'077777 TIME:     077777           ; STARTING TIME
31 00027'177700 CNT:      177700           ; LOOP COUNT
32                        .END    PINT

**00000 TOTAL ERRORS, 00000 FIRST PASS ERRORS
```

```
 0001  PINT     AOS ASSEMBLER REV 02.03          14:45:17 08/21/80
02 KPIT.SR                .TITL   PINT
02                        .NREL
03
04 00000'060205 PINT:     NIOC    5                ; CLEAR THE PIT
05 00001'020434           LDA     0,SETUP
06 00002'061005           DOA     0,5
07 00003'024433           LDA     1,TIME
08 00004'066005           DOB     1,5              ; SEND THE START TIME
09 00005'020432           LDA     0,FLY            ; READ ON THE FLY WORD
10 00006'061005           DOA     0,5
11 00007'024431           LDA     1,CNT
12 00010'044431           STA     1,DUMMY          ; CREATE DUMMY VARIABLE
13 00011'020433           LDA     0,.BUFF          ; LOAD BUFFER START ADDRESS
14 00012'040020           STA     0,20
15 00013'060105 START:    NIOS    5                ; START THE TIMER
16 00014'000401           JMP     .+1              ; WASTE SOME TIME
17 00015'063505           SKPBZ   5
18 00016'000777           JMP     .-1
19 00017'076405           DIC     3,5
20 00020'020416           LDA     0,TIME
21 00021'162400           SUB     3,0
22 00022'042020           STA     0,@20
23 00023'030417 VDU:      LDA     2,SYM            ; OUTPUT CHARACTER
24 00024'060211           NIOC    TTO
25 00025'000401           JMP     .+1
26 00026'071111           DOAS    2,TTO
27 00027'000401           JMP     .+1
28 00030'063511           SKPBZ   TTO              ; TTO FINISHED?
29 00031'000777           JMP     .-1
30 00032'014407           DSZ     DUMMY            ; ALL DATA LOADED?
31 00033'000760           JMP     START
32 00034'063077           HALT
33 00035'006062 SETUP:    6062              ; READ 2 BYTES, MODE 1, BINARY
34 00036'000777 TIME:     000777            ; STARTING TIME
35 00037'006002 FLY:      6002
36 00040'000012 CNT:      12
37 00041'000000 DUMMY:    0
38 00042'037076 SYM:      .TXT    ">>"
39       000000
40 00044'000044'.BUFF:    BUFF-1
41 00045'001750 BUFF:     .BLK    1000.
42                        .END    PINT
```

```
0001   INTV      AOS ASSEMBLER REV 02.03          10:16:02 08/21/80
1                ; THIS PROGRAM MONITORS THE DIGITAL INTERFACE OF THE MICRONOVA
2                ; WHEN A ZERO TO ONE TRANSITION OCCURS ON ANY ONE OF THE
3                ; 16 INPUT LINES, I.E. AN EXTERNAL INTERRUPT OCCURS
4                ; THE DIGITAL INPUT STATUS AND THE ELAPSED TIME SINCE THE LAST
5                ; EVENT ARE STORED AS A COUPLET IN A BUFFER
6                ;
7                ; THE PROGRAM IS USED TO MONITOR MULTIPLE ELECTRODES FOR
8                ; THE OCCURRENCE OF NEURONAL ACTION POTENTIALS
9                ;
0                ;
1                ; KEITH WILLENBERG --- DEPT. OF BIOMEDICAL ENGINEERING, UCT
                         .TITLE INTV
3                        .NREL
4  00000'060242  INTV:   NIOC    DIO              ; CLEAR DIGITAL INTERFACE
5  00001'060205          NIOC    5                ; CLEAR PIT
6  00002'020445          LDA     0,DPTS           ; LOAD THE NUMBER OF DATA POINTS
7  00003'040445          STA     0,DUMMY          ; CREATE DUMMY VARIABLE
8  00004'020446          LDA     0,.BUFF          ; LOAD BUFFER START ADDRESS
9  00005'040020          STA     0,20             ; TO AUTO-INCREMENTING REGISTER
0  00006'102440          SUBO    0,0              ; CLEAR ACC0
1  00007'152440          SUBO    2,2              ; CLEAR ACC2
2  00010'060142          NIOS    DIO              ; START DIGITAL INTERFACE
3  00011'060105  START:  NIOS    5                ; START PIT
4  00012'063605          SKPDN   5                ; WAIT FOR INTERRUPT
5  00013'000777          JMP     .-1              ; THEN PROCEED
6  00014'064405          DIA     1,5              ; LOAD THE PIT STATUS FLAGS
7                ; AN INTERRUPT HAS SET DONE=1
8                ; HAVE TO CHECK SOURCE OF INTERRUPT
9                ; BY CHECKING STATUS FLAGS IN PIT REGISTER-A
0                ; THIS IS DONE BY REPEATEDLY ROTATING ACC1 TO LEFT
1                ; AND EXAMINING THE CARRY BIT
2                ;
3                ;
4  00015'125103          MOVL    1,1,SNC          ; SHIFT ACC1 LEFT
5                ;
6                ; CHECK IF CLOCK OVERRUN FLAG IS SET
7                ; IF NOT, SHIFT ACC1 LEFT AGAIN
8                ;
9  00016'000403          JMP     .+3
0  00017'020426          LDA     0,TIME
1  00020'113000          ADD     0,2
2  00021'125103          MOVL    1,1,SNC          ; SHIFT ACC1 LEFT
3                ;
4                ; CHECK IF EXTERNAL INTERRUPT FLAG IS SET
5                ; IF NOT, LOOP BACK AND WAIT
6                ;
7  00022'000767          JMP     START
8  00023'061442          DIB     0,DIO            ; LOAD CURRENT DIG. INT. STATUS
9  00024'040425          STA     0,TEMP           ; STOE ELECTRODE STATUS IN TEMP.
0  00025'076405          DIC     3,5              ; LOAD CURRENT PIT VALUE
1  00026'020417          LDA     0,TIME
2  00027'162400          SUB     3,0              ; CALCULATE ELAPSED TIME
3  00030'143000          ADD     2,0              ; CORRECT FOR ANY OVERRUN
4  00031'152440          SUBO    2,2              ; CLEAR ACC2
5  00032'125102          MOVL    1,1,SZC          ; SHIFT ACC1 LEFT
6                ;
```

```
57                      ; CHECK IF LOST DATA FLAG IS SET
58                      ; IF SETY, RESTART PIT - NO DATA TRANSFER
59                      ;
60 00033'000756                 JMP     START
```

```
0002  INTV
1 00034'034415            LDA     3,TEMP
2 00035'056020            STA     3,@20           ; WRITE ELECTRODE STATUS TO BUFF
3 00036'042020            STA     0,@20           ; WRITE TIME TO NEXT LOCATION
4 00037'102440            SUBO    0,0             ; CLEAR ACC0
5 00040'040411            STA     0,TEMP          ; CLEAR TEMP. REGISTER
6 00041'014407            DSZ     DUMMY           ; ALL DATA PAIRS LOADED?
7 00042'000747            JMP     START
8 00043'063077            HALT
9 00044'006062 SETUP: 6062
0 00045'000777 TIME: 777
1 00046'006002 FLY: 6002
2 00047'000012 DPTS: 12
3 00050'000000 DUMMY: 0
4 00051'000000 TEMP: 0
5 00052'000052'. BUFF: BUFF-1
6 00053'001750 BUFF:    . BLK     1000.
7.                      . END     INTV

*00000 TOTAL ERRORS, 00000 FIRST PASS ERRORS
```

APPENDIX E.

EXPERIMENTAL SOFTWARE

```
 0001   TPROG    AOS ASSEMBLER REV 02.03          14:00:19 09/11/80
01                ; PITCB.SR        THE USER ROUTINE, I.E. THE DIGITAL INTERFACE AND PIT
02                ;                 SERVICE ROUTINE, MONITORS THE 16 INPUT LINES OF THE
03                ;                 DIGITAL INTERFACE AND WHENEVER A TRANSITION OCCURS
04                ;                 ON ANY ONE OF THESE LINES, THE DIGITAL INPUT STATUS
05                ;                 AND THE ELAPSED TIME SINCE THE LAST EVENT ARE
06                ;                 WRITTEN TO A CIRCULAR BUFFER.
07
08
09                        .TITLE  TPROG
11                        .ENT    .TPROG .DIPT .MSG .RSTRT DPTS WRFLG
12                        .EXTD   .DISMS DADLY DACLK .CBINT .WRIT ACFLG
13
14                        .ZREL
45 00000-000000'. TPROG:  TPROG
16 00001-000036'. DIPT:   DIPT
17 00002-000007': MSG:    MSG
18 00003-000102'. RSTRT:  RSTRT
19 00004-000000$. ACFLG:  ACFLG
20 00005-000000 DPTS:     000
21 00006-000000 WRFLG:    000                          ; USER WRITE FLAG
22
23       000005           .DUSR   PIT=5                ; PIT DEVICE CODE = 5
24       000376           .LOC    376
25 00376 002000-          JMP     @.TPROG
26
27                        .NREL
28
29                ; ********************************************************************
30                ;
31                ;
32                ; THIS IS THE START-UP SECTION OF PITCB.SR
33                ;
34                ;
35 00000'061077 TPROG:    DOA     0,CPU          ; GENERAL RESET
36 00001'000401           JMP     .+1
37 00002'004415           JSR     PINT                 ; INITIALISE USER ROUTINE
38 00003'006000$          JSR     @.CBINT              ; INITIALISE CIRC. BUFFER
39 00004'060177           NIOS    CPU                  ; ENABLE INTERRUPTS
40 00005'071077           DOA     2,CPU                ; ENABLE REAL TIME CLOCK
41 00006'000400           JMP     .                    ; WAIT HERE FOR FIRST INTERRUPT
42                ;
43                ;
44                ; ********************************************************************
45                ;
46                ;
47                ; THIS SECTION SETS THE DIPT ROUTINE WRITE FLAG AND THE CIRCBUFF ACTIVE FLAG&
48                ;
49                ;
50 00007'101420 MSG:      INCZ    0,0                  ; INCR. NO. OF DATA POINTS
51 00010'101420           INCZ    0,0                  ; TO COMPENSATE FOR
52 00011'101420           INCZ    0,0                  ; USE OF DSZ IN DIPT ROUTINE
53 00012'040005-          STA     0,DPTS               ; STORE THE NO. OF DATA POINTS
54                                                     ; TO BE TRANSMITTED TO HOST
55 00013'102520           SUBZL   0,0                  ; GENERATE +1
56 00014'040006-          STA     0,WRFLG              ; SET DIPT ROUTINE WRITE FLAG
```

```
57 00015'042004-          STA     0,@.ACFLG               ; SET CIRC BUFF ACTIVE FLAG
58 00016'002000$          JMP     @.DISMS
59                        ;
60                        ;


 0002  TPROG
01                        ; ********************************************************************
02                        ;
03                        ;
04                        ; THIS SECTION INITIALISES THE DIGITAL INTERFACE AND PIT
05                        ;
06                        ;
07 00017'060242 PINT:     NIOC    DIO                     ; CLEAR DIGITAL INTERFACE
08 00020'060205           NIOC    PIT                     ; CLEAR PIT
09 00021'020412           LDA     0,SETUP                 ; LOAD PIT CONTROL WORD/TIME-BASE
10 00022'061005           DOA     0,PIT
11 00023'020411           LDA     0,TIME                  ; LOAD INITIAL TIME
12 00024'062005           DOB     0,PIT
13 00025'020410           LDA     0,FLY                   ; LOAD READ ON FLY WORD
14 00026'061005           DOA     0,PIT
15 00027'152440           SUBO    2,2                     ; CLEAR ACC2
16 00030'060142           NIOS    DIO                     ; START DIGITAL INTERFACE
17 00031'060105           NIOS    PIT                     ; START PIT
18 00032'001400           JMP     0,3                     ; RETURN TO START-UP ROUTINE
19
20 00033'005062 SETUP:    5062            ; MODE 1 , TIME-BASE 100USEC
21 00034'007777 TIME:     7777
22 00035'005002 FLY:      5002            ; ENABLE READ ON THE FLY
23                        ;
24                        ;
25                        ; ********************************************************************
26                        ;
27                        ;
28                        ; DIGITAL INTERFACE AND PIT SERVICE ROUTINE
29                        ;
30                        ;
31 00036'061442 DIPT:     DIB     0,DIO                   ; LOAD CURRENT DIG. INTF. STATUS
32 00037'024447           LDA     1,MASK                  ; LOAD MASK
33 00040'123420           ANDZ    1,0
34 00041'040446           STA     0,TEMP1                 ; MASK UNUSED LINES
35 00042'064405           DIA     1,PIT                   ; LOAD PIT STATUS FLAGS
36 00043'125103           MOVL    1,1,SNC                 ; SHIFT ACC1 LEFT TO CHECK
37                                                        ; CLOCK OVERRUN FLAG
38                        ; IF NOT SET, SHIFT ACC1 LEFT AGAIN
39                        ; OTHERWISE.RECORD OVERRUN TIME
40
41 00044'000403           JMP     .+3
42 00045'020767           LDA     0,TIME                  ; OVERRUN TIME
43 00046'113000           ADD     0,2                     ; ADD IN FOR OVERRUN
44 00047'125103           MOVL    1,1,SNC                 ; SHIFT ACC1 LEFT TO CHECK
45                                                        ; EXTERNAL INTERRUPT FLAG
46                        ; IF NOT SET, WAIT
47                        ; OTHERWISE RECORD EVENT
48
49 00050'002003-          JMP     @.RSTRT
50 00051'076405           DIC     3,PIT                   ; LOAD CURRENT PIT VALUE
51 00052'020762           LDA     0,TIME
52 00053'162400           SUB     3,0                     ; CALCULATE ELAPSED TIME
53 00054'143000           ADD     2,0                     ; CORRECT FOR ANY CLK OVERRUN
54 00055'040433           STA     0,TEMP2                 ; STORE TIME IN TEMPORARY REGISTER 2
55 00056'152440           SUBO    2,2                     ; CLEAR ACC2
56 00057'125102           MOVL    1,1,SZC                 ; SHIFT ACC1 LEFT TO CHECK
```

```
57                                                      ; LOST DATA FLAG
58              ; IF SET, RESTART - NO DATA TRANSFER
59              ; OTHERWISE TRANSFER DATA TO CIRCULAR BUFFER
60
```

```
 0003  TPROG
01 00060'002003-        JMP     @.RSTRT
02 00061'024006-        LDA     1,WRFLG                 ; IS USER WRITE FLAG SET?
03 00062'125025         MOVZ    1,1,SNR
04 00063'002003-        JMP     @.RSTRT                 ; NO - THEN RESTART
05 00064'020423 ELEC:   LDA     0,TEMP1                 ; WRITE ELEC. STATUS TO CIRCBUFF
06 00065'006000$        JSR     @.WRIT
07 00066'000401         JMP     .+1
08 00067'014005-        DSZ     DPTS                    ; DECREMENT NO. OF DATA POINTS
09 00070'000402         JMP     TIM
10 00071'000406         JMP     CBCLR                   ; IF "DPTS" DATA POINTS SENT
11                                                      ; RE-INITIALISE CIRCBUFF
12 00072'020416 TIM:    LDA     0,TEMP2                 ; WRITE ELAPSED TIME TO CIRCBUFF
13 00073'006000$        JSR     @.WRIT
14 00074'000401         JMP     .+1
15 00075'014005-        DSZ     DPTS                    ; IF "DPTS" DATA POINTS SENT
16                                                      ; RE-INITIALISE CIRCBUFF
17                                                      ; OTHERWISE RESTART
18 00076'002003-        JMP     @.RSTRT
19
20              ; ***************************************************************
21              ;
22              ; RE-INITIALISATION OF CIRCULAR BUFFER
23              ;
24              ;
25 00077'102440 CBCLR:  SUBO    0,0                     ; RESET USER WRITE FLAG
26 00100'040006-        STA     0,WRFLG
27 00101'006000$        JSR     @.CBINT
28
29              ; ***************************************************************
30              ;
31              ; RESTART ROUTINE
32              ;
33 00102'060105 RSTRT:  NIOS    PIT                     ; RESTART PIT
34 00103'002000$        JMP     @.DISMS                 ; RETURN TO CALLING PROGRAM
35
36 00104'000000 ACC0:   000
37 00105'000000 ACC3:   000
38 00106'100000 MASK:   100000
39 00107'000000 TEMP1:  000
40 00110'000000 TEMP2:  000
41                      .END    TPROG

**00000 TOTAL ERRORS, 00000 FIRST PASS ERRORS
```

```
 0001  TPROG    AOS ASSEMBLER REV 02.03          13:55:56 09/11/80
 01              ; PTCONT.SR      THE USER ROUTINE, I.E. THE DIGITAL INTERFACE AND PIT
 02              ;                SERVICE ROUTINE, MONITORS THE 16 INPUT LINES OF THE
 03              ;                DIGITAL INTERFACE AND WHENEVER A TRANSITION OCCURS
 04              ;                ON ANY ONE OF THESE LINES, THE DIGITAL INPUT STATUS
 05              ;                AND THE ELAPSED TIME SINCE THE LAST EVENT ARE
 06              ;                WRITTEN TO A CIRCULAR BUFFER.
 07
 08
 09                       .TITLE  TPROG
 11                       .ENT    .TPROG .DIPT .MSG .RSTRT DPTS WRFLG
 12                       .EXTD   .DISMS DADLY DACLK .CBINT .WRIT ACFLG
 13
 14                       .ZREL
 15 00000-000000'.TPROG:  TPROG
 16 00001-000041'.DIPT:   DIPT
 17 00002-000007'.MSG:    MSG
 18 00003-000101'.RSTRT:  RSTRT
 19 00004-000000$.ACFLG:  ACFLG
 20 00005-000000 DPTS:    000
 21 00006-000000 WRFLG:   000                      ; USER WRITE FLAG
 22
 23       000005          .DUSR   PIT=5            ; PIT DEVICE CODE = 5
 24       000376          .LOC    376
 25 00376 002000-         JMP     @.TPROG
 26
 27                       .NREL
 28
 29              ; ********************************************************************
 30              ;
 31              ;
 32              ; THIS IS THE START-UP SECTION OF PITCB.SR
 33              ;
 34              ;
 35 00000'061077 TPROG:   DOA     0,CPU            ; GENERAL RESET
 36 00001'000401          JMP     .+1
 37 00002'004420          JSR     PINT             ; INITIALISE USER ROUTINE
 38 00003'006000$         JSR     @.CBINT          ; INITIALISE CIRC. BUFFER
 39 00004'060177          NIOS    CPU              ; ENABLE INTERRUPTS
 40 00005'071077          DOA     2,CPU            ; ENABLE REAL TIME CLOCK
 41 00006'000400          JMP     .                ; WAIT HERE FOR FIRST INTERRUPT
 42              ;
 43              ;
 44              ; ********************************************************************
 45              ;
 46              ;
 47              ; THIS SECTION SETS THE DIPT ROUTINE WRITE FLAG AND THE CIRCBUFF ACTIVE FLAG&
 48              ;
 49              ;
 50 00007'101122 MSG:     MOVZL   0,0,SZC          ; IF BIT0=0 THIS IS TRIGGER WD
 51 00010'000405          JMP     STPTR            ; IF BIT0=1 STOP TRANSMISSION
 52
 53 00011'102520          SUBZL   0,0              ; GENERATE +1
 54 00012'040006-         STA     0,WRFLG          ; SET DIPT ROUTINE WRITE FLAG
 55 00013'042004-         STA     0,@.ACFLG        ; SET CIRC BUFF ACTIVE FLAG
 56 00014'002000$         JMP     @.DISMS
```

```
                                                    ; CLEAR ACC0
00015'102440 $TPTR:  SUBO    0,0
00016'040006-        STA     0,WRFLG                ; CLEAR THE USER WRITE FLAG
00017'006000$        JSR     @.CBINT                ; RE-INITIALISE THE CIRC BUFF


.TITL  TPROG
00020'060105         NIOS    PIT                    ; RESTART PIT
00021'002000$        JMP     @.DISMS                ; RETURN TO  INTERRUPT HANDLER
                     ;
                     ; ***************************************************************
                     ;
                     ;
                     ; THIS SECTION INITIALISES THE DIGITAL INTERFACE AND PIT
                     ;
                     ;
00022'060242 PINT:   NIOC    DIO                    ; CLEAR DIGITAL INTERFACE
00023'060205         NIOC    PIT                    ; CLEAR PIT
00024'020412         LDA     0,SETUP                ; LOAD PIT CONTROL WORD/TIME-BASE
00025'061005         DOA     0,PIT
00026'020411         LDA     0,TIME                 ; LOAD INITIAL TIME
00027'062005         DOB     0,PIT
00030'020410         LDA     0,FLY                  ; LOAD READ ON FLY WORD
00031'061005         DOA     0,PIT
00032'152440         SUBO    2,2                    ; CLEAR ACC2
00033'060142         NIOS    DIO                    ; START DIGITAL INTERFACE
00034'060105         NIOS    PIT                    ; START PIT
00035'001400         JMP     0,3                    ; RETURN TO START-UP ROUTINE

00036'005062 SETUP:  5062            ; MODE 1 , TIME-BASE 100USEC
00037'007777 TIME:   7777
00040'005002 FLY:    5002            ; ENABLE READ ON THE FLY
                     ;
                     ;
                     ; ***************************************************************
                     ;
                     ;
                     ; DIGITAL INTERFACE AND PIT SERVICE ROUTINE
                     ;
                     ;
00041'061442 DIPT:   DIB     0,DIO                  ; LOAD CURRENT DIG.  INTF.  STATUS
00042'024443         LDA     1,MASK                 ; LOAD MASK
00043'123420         ANDZ    1,0                    ; MASK UNUSED LINES
00044'040442         STA     0,TEMP1                ;
00045'064405         DIA     1,PIT                  ; LOAD PIT STATUS FLAGS
00046'125103         MOVL    1,1,SNC                ; SHIFT ACC1 LEFT TO CHECK
                                                    ; CLOCK OVERRUN FLAG
                     ; IF NOT SET, SHIFT ACC1 LEFT AGAIN
                     ; OTHERWISE RECORD OVERRUN TIME

00047'000403         JMP     .+3
00050'020767         LDA     0,TIME                 ; OVERRUN TIME
00051'113000         ADD     0,2                    ; ADD IN FOR OVERRUN
00052'125103         MOVL    1,1,SNC                ; SHIFT ACC1 LEFT TO CHECK
                                                    ; EXTERNAL INTERRUPT FLAG
                     ; IF NOT SET, WAIT
                     ; OTHERWISE RECORD EVENT

00053'002003-        JMP     @.RSTRT
00054'076405         DIC     3,PIT                  ; LOAD CURRENT PIT VALUE
00055'020762         LDA     0,TIME
00056'162400         SUB     3,0                    ; CALCULATE ELAPSED TIME
```

```
 57 00057'143000           ADD     2,0                     ; CORRECT FOR ANY CLK OVERRUN
 58 00060'040427           STA     0,TEMP2                 ; STORE TIME IN TEMPORARY REGISTER 2
 59 00061'152440           SUBO    2,2                     ; CLEAR ACC2
 60 00062'125102           MOVL    1,1,SZC                 ; SHIFT ACC1 LEFT TO CHECK


  0003  TPROG
 01                                                        ; LOST DATA FLAG
 02                ; IF SET, RESTART -> NO DATA TRANSFER
 03                ; OTHERWISE TRANSFER DATA TO CIRCULAR BUFFER
 04
 05 00063'002003-          JMP     @.RSTRT
 06 00064'024006-          LDA     1,WRFLG                 ; IS USER WRITE FLAG SET?
 07 00065'125025           MOVZ    1,1,SNR
 08 00066'002003-          JMP     @.RSTRT                 ; NO - THEN RESTART
 09 00067'020417 ELEC:     LDA     0,TEMP1                 ; WRITE ELEC. STATUS TO CIRCBUFF
 10 00070'006000$          JSR     @.WRIT
 11 00071'000401           JMP     .+1
 12 00072'020415 TIM:      LDA     0,TEMP2                 ; WRITE ELAPSED TIME TO CIRCBUFF
 13 00073'006000$          JSR     @.WRIT
 14 00074'000401           JMP     .+1
 15 00075'002003-          JMP     @.RSTRT
 16
 17                ;*************************************************************************
 18                ;
 19                ; RE-INITIALISATION OF CIRCULAR BUFFER
 20                ;
 21                ;
 22 00076'102440 CBCLR:    SUBO    0,0                     ; RESET USER WRITE FLAG
 23 00077'040006-          STA     0,WRFLG
 24 00100'006000$          JSR     @.CBINT
 25
 26                ;*************************************************************************
 27                ;
 28                ; RESTART ROUTINE
 29                ;
 30 00101'060105 RSTRT:    NIOS    PIT                     ; RESTART PIT
 31 00102'002000$          JMP     @.DISMS                 ; RETURN TO CALLING PROGRAM
 32
 33 00103'000000 ACC0:     000
 34 00104'000000 ACC3:     000
 35 00105'100000 MASK:     100000
 36 00106'000000 TEMP1:    000
 37 00107'000000 TEMP2:    000
 38                        .END    TPROG

**00000 TOTAL ERRORS, 00000 FIRST PASS ERRORS
```

```
0001  INFPT    AOS ASSEMBLER REV 02.03        20:17:39 08/26/80
01                        ;
02                        ; INFPT.SR       THIS ROUTINE SERVES AS AN INTERRUPT HANDLER FOR
03                        ;                (1) THE CIRCULAR BUFFER VIA TTO AND TTI
04                        ;                (2) THE DIGITAL INTERFACE AND PIT USER ROUTINES
05                        ;
06                        ;                ROUTINE - INFPT.SR - INTERRUPT FOR PIT
07                        ;                                     --       -  - -
08
                                .TITLE  INFPT
10                              .ENT    DACLK DADLY .DISMS
11                              .EXTD   .CBTIM .SEND .RECV .DIPT
12
13          000001              .LOC    1
14 00001 000000'.INTR:          INTR
15 00002 000024'.CLK:           CLOCK
16
17                              .ZREL
18          000005              .DUSR   PIT=5                   ; PIT DEVICE CODE = 5
19 00000-000012'.DISMS:         DISMS
20 00001-000000 DACLK:          000
21 00002-000000 DADLY:          000
22
23                        ; ********************************************************************
24                        ;
25                        ; COME HERE TO SERVICE INTERRUPTS
26                        ;
27                              .NREL
28 00000'040420 INTR:           STA     0,ACC0                  ; PRESERVE ACCUMULATORS
29 00001'044420                 STA     1,ACC1                  ;
30 00002'050420                 STA     2,ACC2                  ;
31 00003'054420                 STA     3,ACC3                  ;
32 00004'063711                 SKPDZ   TTO                     ; TTO INTERRUPT?
33 00005'002000$                JMP     @.SEND                  ; YES - GO TO CIRCBUFF TRANSMITTER
34 00006'063710                 SKPDZ   TTI                     ; NO - TRY TTI INTERRUPT
35 00007'002000$                JMP     @.RECV                  ; YES - HONOUR THE HANDSHAKE
36
37                        ; ********************************************************************
38                        ;
39                        ; USER DEVICE INTERRUPT - I.E. PIT
40                        ;
41 00010'063705                 SKPDZ   PIT                     ; WAIT HERE FOR PIT INTERRUPT
42 00011'002000$                JMP     @.DIPT                  ; GO TO PIT SERVICE ROUTINE IF
43                                                              ; INTERRUPT OCCURS
44
45                        ; ********************************************************************
46                        ;
47                        ; COME HERE TO DISMISS INTERRUPTS
48                        ;
49 00012'020406 DISMS:          LDA     0,ACC0                  ; RESTORE ACCUMULATORS
50 00013'024406                 LDA     1,ACC1                  ;
51 00014'030406                 LDA     2,ACC2                  ;
52 00015'034406                 LDA     3,ACC3                  ;
53 00016'060177                 NIOS    CPU                     ; ENABLE INTERRUPTS
54 00017'002000                 JMP     @0                      ; RETURN TO INTERRUPTED PROGRAM
55
56 00020'000000 ACC0:           000
```

```
57 00021'000000 ACC1:    000
58 00022'000000 ACC2:    000
59 00023'000000 ACC3:    000
60
```

```
 0002  INFPT
01                ; ***************************************************************************
02                ;
03                ; COME HERE TO SERVICE A RTC INTERRUPT
04                ;
05 00024'040774 CLOCK:   STA      0,ACC0
06 00025'044774          STA      1,ACC1
07 00026'050774          STA      2,ACC2
08 00027'054774          STA      3,ACC3
09 00030'002000$         JMP      @.CBTIM                    ; GO TO CIRCBUFF TIMEOUT ROUTINE
10
11                       .END     INTR

**00000 TOTAL ERRORS, 00000 FIRST PASS ERRORS
```

```
; CBFPIT.SR
;
; THIS ROUTINE PROVIDES A CIRCULAR BUFFER FOR TRANSMISSION FROM MICRONOVA
; TO HOST.  THE DATA IS TRANSMITED IN RECORDS 'RECLN' WORDS LONG.  THE
; ROUTINE WAITS FOR A HANDSHAKE WORD FROM THE HOST AFTER THE TRANSMISSION
; OF EACH RECORD.  IN FUTURE DEVELOPMENTS OF THIS PROGRAM THE HANDSHAKE
; WORD WILL BE USED TO IMPLEMENT A CHECKSUM.
;
; DESIGNED FOR USE IN A STAND-ALONE ASSEMBLER ENVIRONMENT, WITH A SIMPLE
; INTERRUPT STRUCTURE.  ( SEE THE INTERRUPT HANDLER INFPT.SR FOR DETAILS. )
;
; ROD DOUGLAS AND DAVE BOONZAIER        7 FEB 79        REV 1.00
;

        .TITL   CBFPIT
        .EXTD   .DISMS .MSG WRFLG
;
; WRFLG IS THE USER WRITE ENABLE FLAG
;
        .ENT    .CBINT .WRIT .SEND .RECV .CBTIM ACFLG

        .ZREL
.CBIN:  CBINT                   ; ENTRY FOR INITIALISATION
 WRIT:  WRITE                   ; ENTRY FOR USER CALL TO CIRC BUFF
.SEND:  SEND                    ; ENTRY FOR TTO INTERRUPT SERVICE
 RECV:  RECV                    ; ENTRY FOR TTI INTERRUPT SERVICE
.CBTIM: CBTIM                   ; ENTRY FOR CIRC BUFF TIMEOUTS
WRPTR:  000                     ; CIRC BUFFER WRITE POINTER
RDPTR:  000                     ; CIRC BUFFER READ POINTER
ACFLG:  000                     ; CIRC BUFFER ACTIVE FLAG (1=ACTIVE)


        .NREL

CBFPT:

; COME HERE TO WRITE USER DATA TO THE CIRCULAR BUFFER;
;
; CALLING SEQUENCE;
; ------------------        JSR     @.WRIT          (ACC0 = USER DATA WORD)
;                           <BUFFER FULL>
;                           <NORMAL RETURN>
;
; ACC1,ACC2 PRESERVED
;

WRITE:  STA     1,ACC1          ; PRESERVE ACCUMULATORS
        STA     2,ACC2
        LDA     2,WRPTR         ; LOAD THE CURRENT WRITE-POINTER
        INC     2,2             ; INCREMENT THE POINTER
        LDA     1,MODU          ; LOAD THE MASK FOR MODULO-256 COUNTING
        ANDZ    1,2             ; MASK THE POINTER MODULO-256
        LDA     1,RDPTR         ; LOAD THE CURRENT READ-POINTER
        SUB#    1,2,SNR         ; SKIP IF WRPTR .NE. RDPTR
        JMP     FULL            ; CAUGHT UP WITH OUR TAIL
        STA     2,WRPTR         ; STORE THE UPDATED WRITE-POINTER
        LDA     1,.BUFF         ; LOAD THE CIRC BUFF START ADDRESS
```

```
        ADDZ    1,2             ; COMPUTE THE CURRENT WRITE POSITION IN
                                ; CIRC BUFF.  (.BUFF+WRPTR MUST BE LT 177777
        STA     0,0,2           ; STORE THE USER DATA WORD IN CIRC BUFF
        SUBO    0,0             ; GENERATE A ZERO IN ACC0
        STA     0,BFLG          ; AND STORE IN BUFFER FLAG
                                ; 0 = BUFFER NOT EMPTY   1 = BUFFER EMPTY
        LDA     1,ACC1          ; RESTORE ACCUMULATORS
        LDA     2,ACC2
        JMP     1,3             ; NORMAL RETURN TO USER CALLING PROGRAM

FULL:   LDA     1,ACC1          ; RESTORE ACCUMULATORS
        LDA     2,ACC2
        JMP     0,3             ; ABNORMAL RETURN - BUFFER FULL


MODU:   007777          ; MASK FOR 4096 WORD BUFFER
ACC1:   000
ACC2:   000
BFLG:   001                     ; SET BUFFER EMPTY FLAG

; COME HERE TO SERVICE AN INTERRUPT FROM TTO;

SEND:   LDA     0,HSFLG         ; LOAD HANDSHAKE FLAG
        MOVZR#  0,0,SZC         ; TEST FLAG, 1=WAITING FOR HANDSHAKE
        JMP     @.DISMS         ; RETURN TO THE INTERRUPT HANDLER
        LDA     1,RDPTR         ; IF NO SEND ANOTHER BYTE,
        INC     1,1             ; BY LOADING RDPTR AND INCREMENTING
        LDA     0,MODU          ; LOAD MASK FOR MODULO-256 COUNTING
        ANDZ    0,1             ; MODULO-256 MASK THE READPOINTER
        LDA     0,WRPTR         ; LOAD WRITEPOINTER
        SUB#    0,1,SNR         ; SKIP IF BUFFER NOT EMPTY IE RDPTR+1 NE WRP
        JMP     EMPTY           ; ELSE WAIT FOR NEW DATUM TO SEND
        STA     1,RDPTR         ; STORE INCREMENTED READPOINTER
        LDA     2,.BUFF         ; LOAD CIRCULAR BUFFER ZEROTH ADDRESS
        ADDZ    1,2             ; COMPUTE CURRENT READ POSITION IN CIRC
                                ; BUFF.  (.BUFF+RDPTR MUST BE LT 1777777)
        LDA     0,0,2           ; LOAD A DATA WORD
        MOVS    0,1             ; SWAP BYTES
        DOAS    1,TTO           ; SEND HIGH-ORDER BYTE FIRST
        SKPBZ   TTO
        JMP     .-1
        DOAS.   0,TTO           ; THEN LOW-ORDER BYTE
        SKPBZ   TTO
        JMP     .-1
        NIOC    TTO             ; CLEAR TTO INTERRUPT
        DSZ     ICNT            ; CHECK FOR END OF RECORD
        JMP     @.DISMS         ; IF NOT END RETURN TO INTERRUPT HANDLER
                                ; W/O SETTING HANDSHAKE FLAG
        LDA     0,RECLN         ; END OF RECORD -RESTORE RECORD WORD COUNTER
        STA     0,ICNT
        SUBZL   0,0             ; GENERATE 1=AWAITING HANDSHAKE
        STA     0,HSFLG         ; STORE IN HANDSHAKE FLAG
        JMP     @.DISMS         ; RETURN TO INTERRUPT HANDLER

EMPTY:  SUBZL   0,0             ; SET BIT 1 IN ACC0
        STA     0,BFLG          ; AND SET THE BUFFER FLAG (1=EMPTY)
        NIOC    TTO             ; CLEAR THE TTO INTERRUPT
```

```
        JMP     @.DISMS         ; AND RETURN TO INTERRUPT HANDLER

HSFLG:  000                     ; HANDSHAKE FLAG (1=WAITING ON HOST HDSHAKE

; COME HERE TO INITIALISE THE CIRCULAR BUFFER;
;
; CALLING SEQUENCE;
; ----------------
;
;                       JSR     @.CBINT
;                       <NORMAL RETURN>
;
; ALL ACCUMULATORS PRESERVED
;

CBINT:  STA     0,ACCUMO
        LDA     0,RECLN         ; LOAD NUMBER OF WORDS PER RECORD
        STA     0,ICNT
        SUBO    0,0             ; CLEAR ACC0 AND CARRY BIT
        STA     0,WRPTR         ; INITIALISE REGISTERS
        STA     0,RDPTR
        STA     0,ACFLG
        STA     0,HSDLY
        STA     0,TMOUT
        SUBZL   0,0             ; GENERATE +1
        STA     0,BFLG

        NIOC    TTO             ; CLEAR TRANSMISSION INTERFACES
        NIOC    TTI
        JMP     .+1
        NIOS    TTI             ; RESTART THE RECEIVE INTERFACE
        STA     0,ACCUMO                ; RESTORE ACC0
        JMP     0,3             ; RETURN TO CALLING PROGRAM

ACCUMO: 00                      ; ACC0 TEMP STORE
RECLN:  64.                     ; OUTPUT RECORD LENGTH IN WORDS
ICNT:   00                      ; RECORD WORD COUNTER


; COME HERE TO SERVICE A TTI INTERRUPT;
;
; FIRST ESTABLISH WHETHER THE HANDSHAKE FLAG IS SET;
; YES -- THEN THIS TRANSMISSION MUST BE THE HANDSHAKE....
; NO  -- THEN THIS TRANSMISSION IS EITHER A MESSAGE (IF CIRCBUF IS INACTIVE)
;        OR AN ERROR (IF CIRCBUF IS ACTIVE), SO CHECK ACFLG....
;
;

RECV:   DIAS    0,TTI           ; LOAD THE HIGH BYTE OF THE TRANSMITTED WORD
        MOVS    0,1             ; TRANSFER TO ACC1 & SWAP BYTES
        SKPBZ   TTI
        JMP     .-1
        DIAS    0,TTI           ; LOAD THE LOW BYTE
        ADD     1,0             ; GENERATE THE COMPLETE WORD
```

```
        LDA     1,HSFLG         ; LOAD THE HANDSHAKE FLAG
        MOVZ#   1,1,SNR         ; SKIP IF AWAITING HANDSHAKE
        JMP     MESSG           ; ELSE CHECK FOR MESSAGE
        SUBO    1,1             ; CLEAR ACC1
        STA     1,HSFLG         ; AND CLEAR THE HANDSHAKE FLAG
;
; TEST BIT 0 OF HANDSHAKE - IF BIT 0 = 0  THEN CONTINUE TRANSMISSION
;                           IF BIT 0 = 1 THEN STOP TRANSMISSION
; CHECKSUM HERE IN FUTURE.
;
        MOVZL#  0,0,SNC
        JMP     @.DISMS         ; BIT0=0 - DISMS. INT. - CONT. TRANSMISSION
RESET:  JMP     @.MSG           ; BIT0=1 IS MESSAGE TO STOP TRANSMISSION


MESSG:  JMP     @.MSG           ; CALL THE MESSAGE INTERPRETER

; COME HERE FROM A REAL-TIME CLOCK INTERRUPT SERVICE ;
;
; THIS ENTRY (1) MONITORS THE HANDSHAKE FLAG, AND STARTS A TIMEOUT
;            WHEN THE CIRC BUFF IS WAITING FOR A RECORD END HANDSHAKE
;            FROM THE HOST.
;            (2) MONITORS THE BUFFER EMPTY FLAG, AND RESTARTS CIRC BUFFER
;            TRANSMISSION WHEN BFLG IS DOWN (CIRC BUFF NON-EMPTY)
;

CBTIM:  LDA     0,BFLG          ; LOAD THE BUFFER FLAG
        MOVZR#  0,0,SZC         ; SKIP IF BFLG=0 (BUFFER NON-EMPTY)
        JMP     EMBF            ; ELSE CHECK FOR END OF USER TRANSMISSION,
                                ; SINCE CIRC BUFF IS EMPTY
        SKPBZ   TTO             ; TTO BUSY ?
        JMP     HNDSK           ; YES - GO TO HNDSK MONITOR
        JSR     SEND            ; NO - RESTART CIRC BUFF TRANSMISSION

EMBF:   LDA     0,WRFLG         ; TEST THE USER WRITE ENABLE FLAG
        MOVZ#   0,0,SZR
        JMP     HNDSK           ; FLG SET, USER STILL TRANSMITTING...
        LDA     0,ACFLG         ; TEST CIRC BUFF ACTIVE FLAG
        MOVZ#   0,0,SZR
        JSR     @.CBINT         ; IF USER TRANSMISSION DISABLED (COMPLETE)
                                ; AND CIRCBUFF EMPTY, REINITIALISE CIRCBUFF
;**** DANGER HERE **** HSFLG WILL BE CLEARED EVEN IF WE'RE STILL WAITING
; ON FINAL HANDSHAKE....

HNDSK:  LDA     0,HSFLG         ; LOAD THE HANDSHAKE FLAG
        MOVZR#  0,0,SNC         ; SKIP IF SET (WAITING FOR HANDSHAKE)
        JMP     CLEAR           ; ELSE RETURN
        LDA     0,TMOUT         ; LOAD THE TIME-OUT FLAG
        MOVZR#  0,0,SNC         ; SKIP IF SET (TIMEOUT IN PROGRESS)
        JMP     START           ; ELSE START A TIMEOUT
        DSZ     HSDLY           ; DECREMENT THE TIMEOUT COUNTER
        JMP     @.DISMS         ; NORMAL RETURN (TIMEOUT IN PROGRESS)
        LDA     2,ERFLG         ; LOAD AN ERROR FLAG TO ACC2
        HALT
ERFLG:  000002                  ; FAILURE OF HOST HANDSHAKE

START:  LDA     0,HSTIM         ; LOAD THE HANDSHAKE DELAY VALUE
```

```
        STA     0,HSDLY         ; AND STORE IN COUNTER
        JMP     @.DISMS         ; NORMAL RETURN AFTER STARTING TIMEOUT

CLEAR:  SUBO    0,0             ; GENERATE A ZERO
        LDA     0,TMOUT         ; AND CLEAR THE TIMEOUT FLAG
        JMP     @.DISMS                 ; NORMAL RETURN TO INTERRUPT HANDLER

HSTIM:  100.                    ; TIMEOUT=240 MSEC
HSDLY:  000                     ; TIMEOUT COUNTER
TMOUT:  000                     ; TIMEOUT FLAG  1=TIMEOUT IN PROGRESS

; CIRCULAR BUFFER STORAGE AREA;

.BUFF:  BUFF
BUFF:   .BLK    4096.           ; RESERVE 4096 WORD BUFFER BLOCK

        .END    CBFPT
```

```
;  BINRDC.SR
;
;  THIS ROUTINE READS DATA TRANSMITTED FROM THE MICRONOVA IN HANDSHAKE
;  MODE.  IT IS SIMILAR TO BINREAD.SR, BUT IT DOES NOT REQUIRE INFORMATION
;  ABOUT TOTAL NO.  OF WORDS TO BE SENT.
;
;  THE ROUTINE SENDS TWO CHARACTERS AS A START TRIGGER FOR THE MICRONOVA
;  PROGRAM.  THEN IT READS A BUFFER RECLN WORDS LONG, AND RETURNS A ONE WORD HAN
;  TO THE MICRONOVA
;  THE DATA READ IN FROM THE MICRONOVA IS WRITTEN TO <FILENAME> IN BINARY
;  FORMAT.
;  DATA TRANSMISSION OCCURS UNTIL A USER DEFINED INTERVAL HAS ELAPSED.
;  THE PROGRAM HAS A SUBTASK THA WRITES THE NUMBER OF BUFFERS LOADED TO @OUTPUT
;  EVERY 60 SECS
;
;
;  REV 1.0    ROD DOUGLAS       2 APR 79
;  REV 2.0     RJD     22 JUN 79
;  TIMEOUT ADDED.
;  REV 3.0         RJD     26 AUG 80
;  MODIFIED TO PROVIDE CONTINOUS READ FROM MICRONOVA, OVER USER SUPPLIED
;  TIME PERIOD.
;
;  THE PROGRAM CONTAINS THREE TASKS;
;  (1)    TIMER - ALLOWS BINRDC TO RUN FOR USER DEFINED PERIOD
;  (2)    COMMUNICATOR - WRITES VALUE OF NBUFS (TOTAL DATA BUFFERS READ)
;         TO THE USER AT @CONSOLE AT ONE MINUTE INTERVALS.
;  (3)    BINARY READER - READS 64 WORD BUFFERS FROM THE MICRONOVA PORT
;         AND WRITES THESE TO <DATAFILE>.  DATA TRANSFER IS INITIATED BY
;         SENDING A TRIGGER WORD (000000) TO THE MICRONOVA.  AFTER EACH
;         BUFFER IS RECEIVED A HANDSHAKE WORD (000000) IS SENT TO MICRONOVA
;         PROVIDED THAT TRANSMISSION IS STILL ENABLED BY TIMER.  IF TIMER
;         HAS DISABLED FURTHER TRANSMISSION, A STOP WORD (100000) IS SENT
;         TO MICRONOVA AND CONTROL RETURNS TO AOS CLI.
;
;
;  CALLING SEQUENCE;
;  -----------------
;
;         XEQ BINRDT <MICRONOVA PORT NAME> <OUTPUT FILENAME>
;
;  (<MICRONOVA MESSAGE> IS A DECIMAL NUMBER WITH MAX 32000)
;
;
         .TITLE  BINREAD
         .ENT    BINRD
         .TXTM 2

         .ZREL
.TIMWD:  TIMWD                    ; TIMER WORD
.INBUF:  INBUF                    ; ADDRESS OF FIRST WORD OF MN INPUT RECORD
SRTWD:   000              ; START-UP AND HANDSHAKE WORD
STPWD:   100000           ; STOP TRANSMISSION WORD
.OUTBF:  OUTBF                    ; ADDRESS OF OUTPUT WORD
.ERROR:  ERR                      ; ADDRESS OF ERROR EXIT
TFLG:    001                      ; TRANSMISSION FLAG (1=TRANSM ENABLED)
```

```
        NREL
        .TSK    4

; FIRST GET THE INITIALISATION MESSAGES;

BINRD:  ?GTMES  MSPK1           ; GET ARG 1 (MICRONOVA PORT NAME)
        JMP     @.ERROR
        ?GTMES  MSPK2           ; GET ARG 2 (OUTPUT FILE NAME)
        JMP     @.ERROR
        ?GTMES  MSPK3           ; GET ARG 3 (HIGH WORD OF MILLISEC TIMER)
        JMP     @.ERROR
        STA     1,@.TIMWD               ; STORE BINARY EQUIVALENT OF ARG 3

; NOW OPEN THE FILES;

        ?OPEN   MNOUT           ; OUTPUT TO MICRONOVA
        JMP     @.ERROR
        ?OPEN   MNINP           ; INPUT FROM MICRONOVA
        JMP     @.ERROR
        ?OPEN   DATFL           ; DATA OUTPUT FILE
        JMP     @.ERROR
        ?OPEN   USER            ; USER CONSOLE OUTPUT
        JMP     @.ERROR

; START THE TASKS;

        ?TASK   TPK1            ; START THE TIMER
        JMP     @.ERROR
        ?TASK   TPK2            ; START THE COMMUNICATOR
        JMP     @.ERROR
        ?TASK   TPK3            ; START BINARY READER
        JMP     @.ERROR

; NOW KILL THE CURRENT TASK;

        ?KILL
TPK1:   1                       ; STANDARD PACKET
        0                       ; USE CALLER'S PRIORITY
        2                       ; ID=2
        TIMER                   ; START ADDRESS
        0                       ; NO MESSAGE
        STK1                    ; TASK STACK ADDRESS
        30.                     ; 30 WORD STACK
        -1                      ; NO DEFAULT ROUTINE
        0                       ; NO TASK SPECIFICATION FLAGS
        0                       ; NO OVERLAY
        1                       ; INITIATE ONE TASK ONLY

TPK2:   1
        0
        3
        COMM
        0
        STK2
        30.
        -1
```

```
        0
        0
        1

TPK3:   1
        0
        4
        BIN
        0
        STK3
        30.
        -1
        0
        0
        1

STK1:   .BLK    30.
STK2:   .BLK    30.
STK3:           .BLK    30.

MNINP:  0                               ; PARAMETER PACKET FOR INPUT OF ONE
        ?ICRF+?IBIN+?OFIN+?RTDY         ; 128 BYTE RECORD FROM MICRONOVA
        0
        INBUF*2
        0
        128.
        0
        0
        0
        CON*2
        -1
        -1


DATFL:  0                               ; PARAMETER PACKET FOR OUTPUT TO @LIST
        ?ICRF+?IBIN+?OFOT+?RTDY
        0
        INBUF*2
        0
        128.
        0
        0
        0
        FNAME*2
        -1
        -1

MNOUT:  0                               ; PARAMETER PACKET FOR OUTPUT TO MICRONOVA
        ?ICRF+?IBIN+?OFOT+?RTDY
        0
        OUTBF*2
        0
        2
        0
        0
        0
        CON*2
```

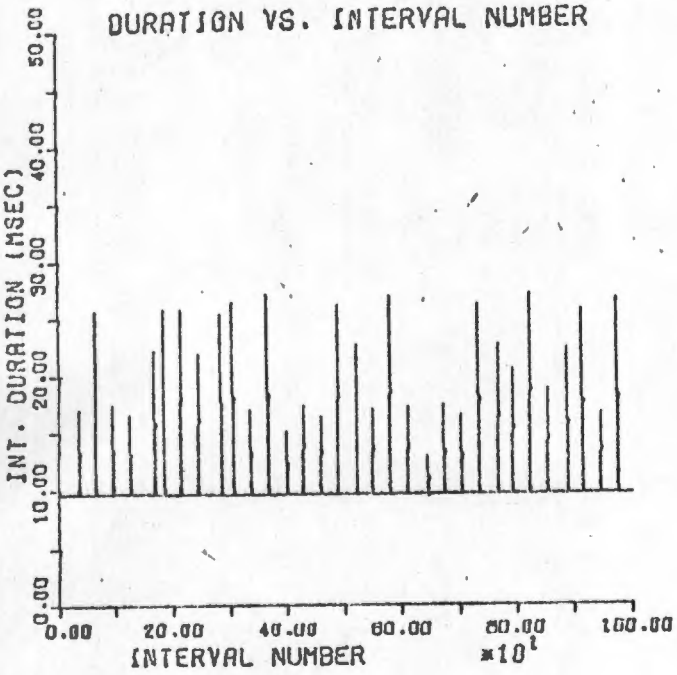
```
        -1
        -1

USER:   0                               ; PARAMETER PACKET FOR USER CONSOLE
        ?ICRF+?IBIN+?OFOT+?RTDY
        0
        UBUF*2
        0
        10.
        0
        0
        0
        UCON*2
        -1
        -1

MSPK1:  ?GARG                           ; GET THE FIRST ARGUMENT
        1
        0
        CON*2
MSPK2:  ?GARG                           ; GET THE SECOND ARGUMENT
        2
        0
        FNAME*2                         ; BYTE POINTER TO <FILENAME>
MSPK3:  ?GARG                           ; GET THE THIRD ARGUMENT
        3
        0
        MSBF3*2             ; TEMP STORE FOR ASCII DIGITS
SWORD:  2                               ; 2 BYTE RECORD
RWORD:  128.                            ; 128 BYTE RECORD
RECLN:  32.                             ; NUMBER OF WORDS PER INPUT RECORD
                                        ; (INCLUDING CHECKSUM WORD)
UBUF:   .BLK    10.
FNAME:  .BLK 10.
CON:    .BLK 10
OUTBF:  .BLK 2
INBUF:  .BLK 68.
MSBF3:  .BLK    6
;

ERR:    LDA     2,FLAG
        ?RETURN
        JMP     @.ERROR

FLAG:   ?RFCF+?RFER+?RFEC

;-------------------------------------------------------------------------------
;
;  TASK 1     TIMER
;
TIMER:  LDA     1,LOWD                  ; LOW ORDER TIMER WORD (60 SEC)
        LDA     0,TIMWD                 ; LOAD THE HIGH ORDER TIME WORD
        ?DELAY
        JMP     @.ERROR

        LDA     0,STPWD                 ; LOAD THE SIUP MESSAGE
```

```
        STA     0,@.OUTBF
        ?WRITE  MNOUT           ; AND SEND TO MICRONOVA
        JMP     @.ERROR

; NOW TERMINATE THE PROGRAM;

        LDA     1,MSG           ; GET THE SWANSONG
        LDA     2,FLAGS         ; AND THE ?RETURN FLAGS
        ?RETURN
        JMP     @.ERROR

FLAGS:  ?RFCF+33
MSG:    .+1*2
        .TXT    "THIS IS BINRDC SINGING OFF"

LOWD:   60000.                  ; 60 SECS IS TIMER MULTIPLE
TIMWD:  000                     ; TIMER INTERVAL * 60000 MSEC

;-----------------------------------------------------------------------
;
; TASK 2    COMMUNICATOR
;
COMM:   SUBO    0,0             ; CLEAR THE HIGH TIMER WORD
        LDA     1,BTIM          ; LOAD 60000. MSEC INTERVAL
        ?DELAY
        JMP     @.ERROR
        LDA     0,ACC0          ; BYTE POINTER TO CONSOLE NAME
        LDA     1,ACC1          ; BYTE POINTER TO MESSAGE TO BE SENT
        LDA     2,ACC2          ; SOME OTHER SEND PARAMETERS
        ?SEND
        JMP     @.ERROR
        LDA     0,NBUFS         ; LOAD THE NUMBER OF DATA BUFFERS READ
; **** SOME MORE TO DO HERE!
        JMP     COMM

BTIM:   60000.          ; ONE MINUTE TIMER
NBUFS:  000                     ; UPDATED BY BINARY READER
ACC0:   UCON*2
ACC1:   MSG1*2
ACC2:   001042          ; 32 CHARS, @CONSOLE

UCON:   .TXT    "@CONSOLE"
MSG1:   .TXT "NUMBER OF WORDS TRANSFERED = 64 * "

;-----------------------------------------------------------------------
;
; TASK 3    BINARY READER
;
; FIRST SEND A TRIGGER TO MICRONOVA TO INITIATE TRANSMISSION
;
BIN:    SUBO    0,0             ; <NULL><NULL>
        STA     1,@.OUTBF
        ?WRITE  MNOUT
        JMP     @.ERROR
```

```
; NOW COMMENCE BINARY READ, 64 WORDS A BUFFER;

RECORD: ?READ   MNINP           ; AND READ A NEW RECORD
        JMP     @.ERROR

        LDA     0,SRTWD         ; LOAD THE CONTINUATION HANDSHAKE WORD
        STA     0,@.OUTBF
        ?WRITE  MNOUT           ; AND SEND TO MICRONOVA
        JMP     @.ERROR
        ?WRITE  DATFL           ; WRITE THE RECORD TO OUTPUT FILE
        JMP     @.ERROR
        ISZ     NBUFS           ; INCREMENT THE BUFFER RECEIVED COUNT
        JMP     RECORD

;-------------------------------------------------------------------------------

        .END    BINRD
```

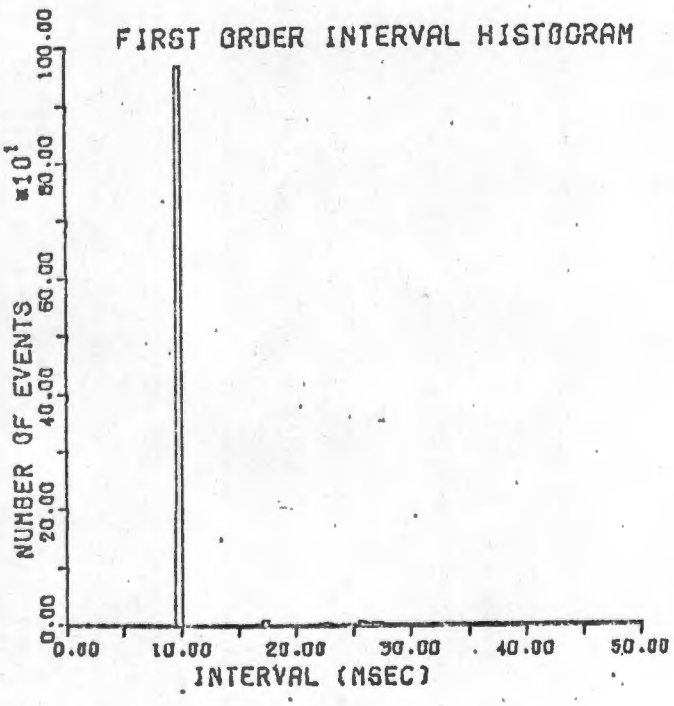
APPENDIX F.

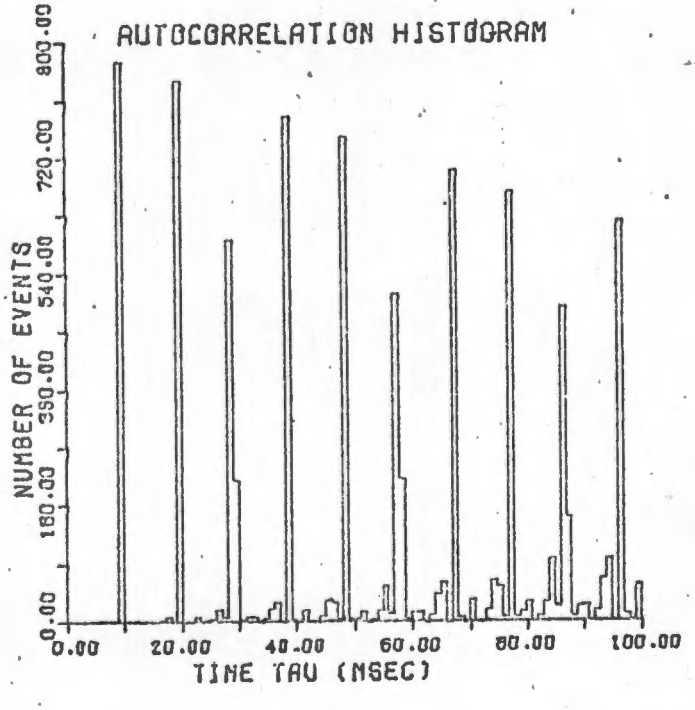
PIT CALIBRATION RESULTS

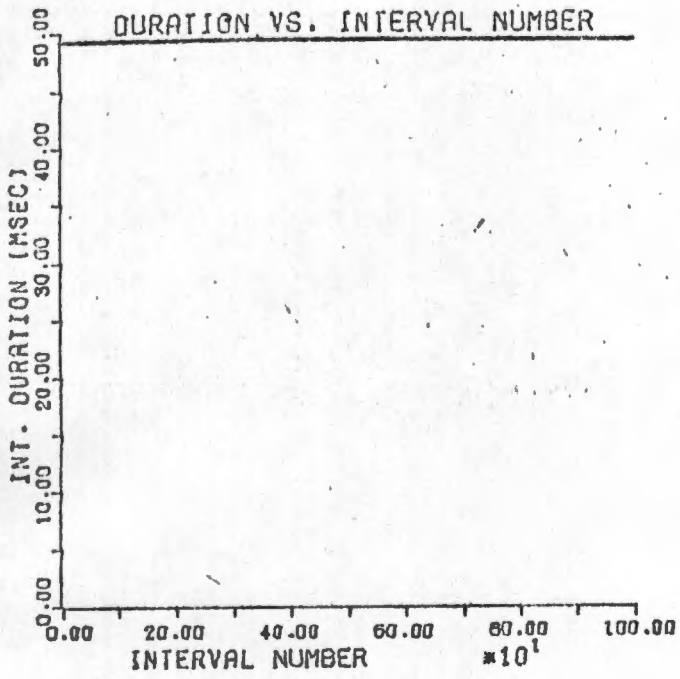


CALIBRATION OF PROG.INT.TIMER

HP PULSE GENERATOR 11SEPT80
CELL NUMBER PER10.16
PROGRAM CELLREPORT.FR REV1.0

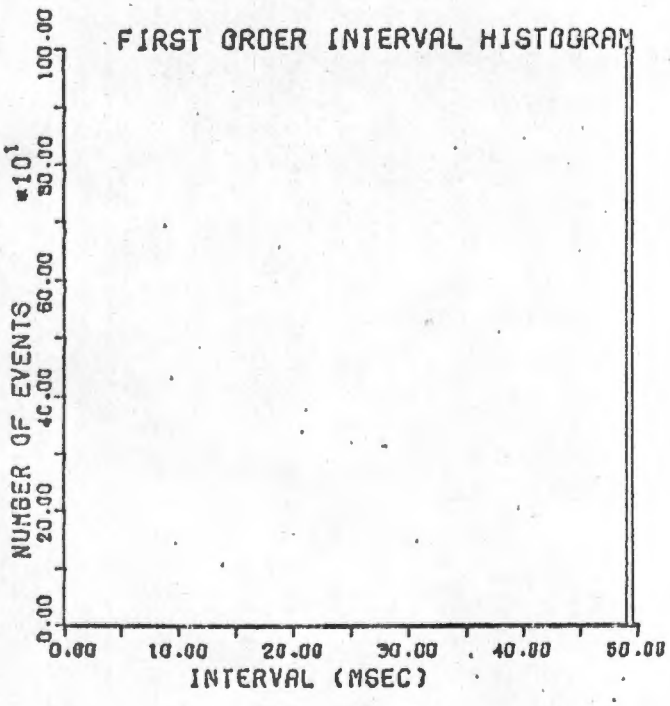


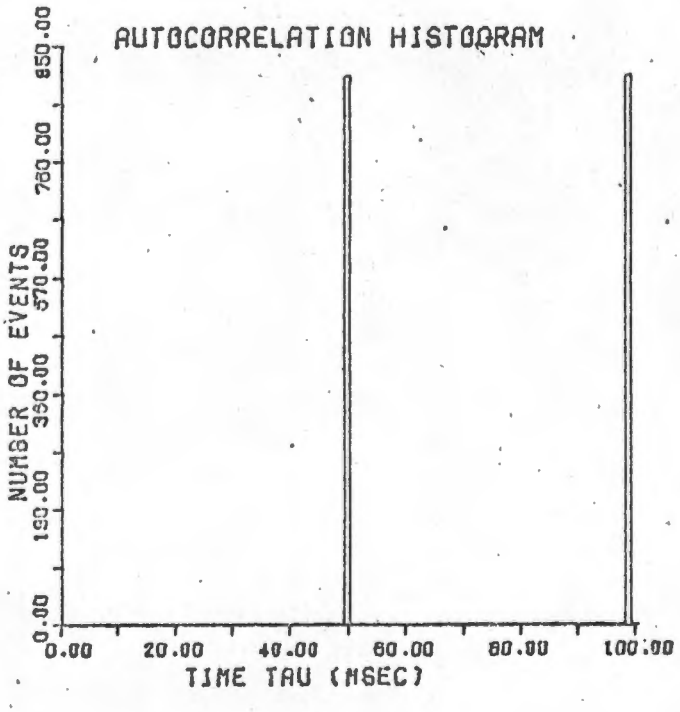


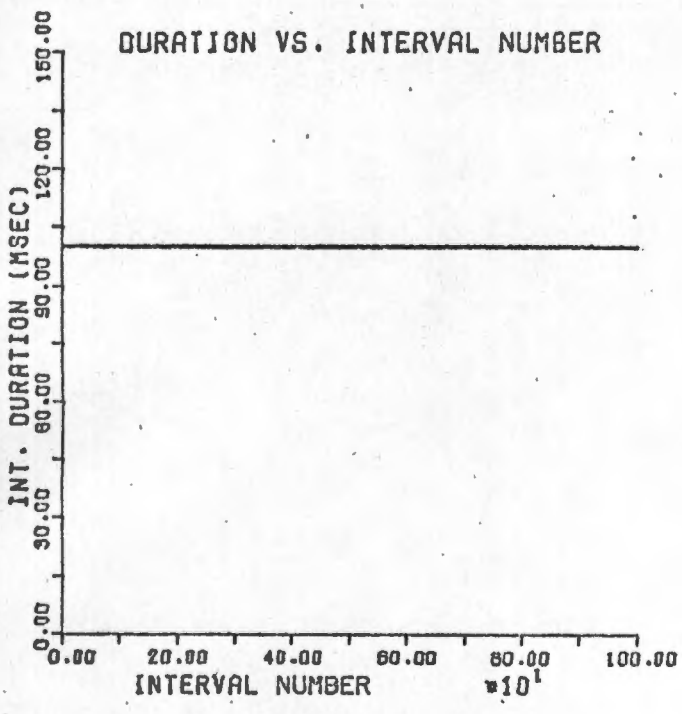


CALIBRATION OF PROG.INT.TIMER

HP PULSE GENERATOR 11SEPT80
CELL NUMBER PER49.46
PROGRAM CELLREPORT.FR REV1.0

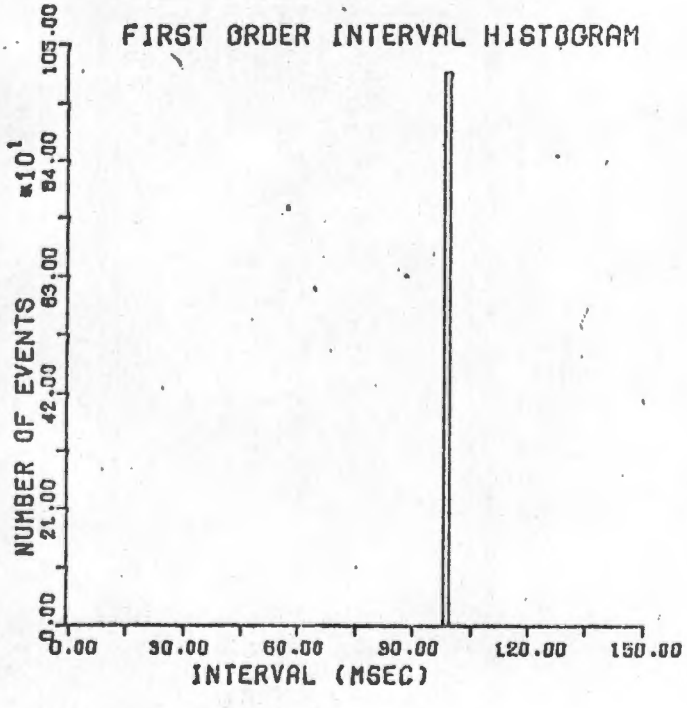


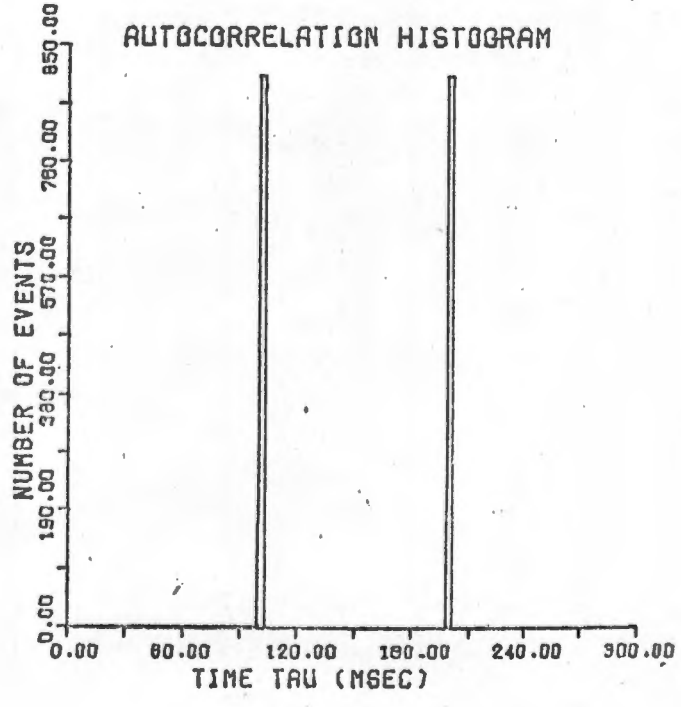




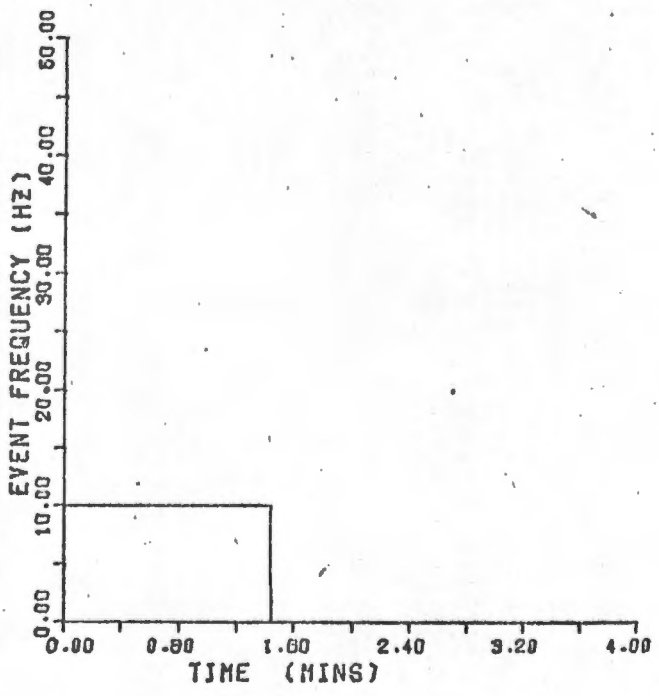
CALIBRATION OF PROG.INT.TIMER

HP PULSE GENERATOR 11SEPT80
CELL NUMBER PER100.52
PROGRAM CELLREPORT.FR REV1.0

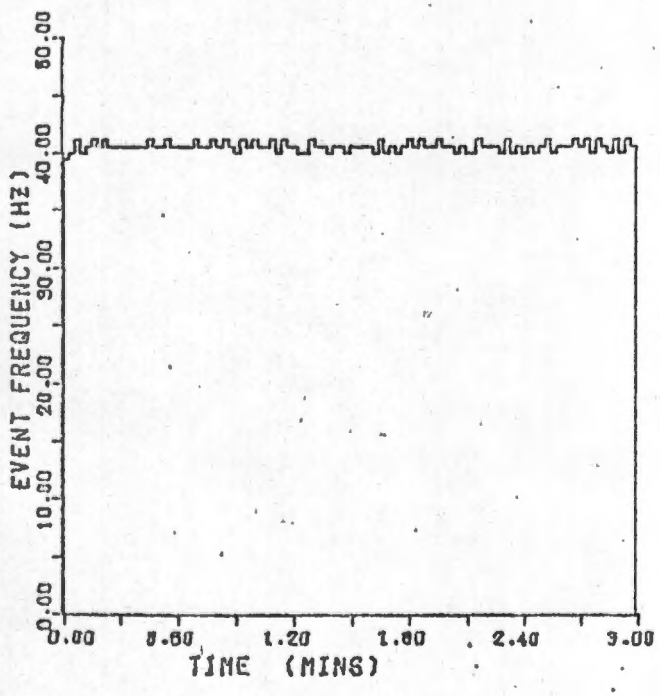




CALIBRATION OF PROG.INT.TIMER
HP PULSE GENERATOR 11SEPT80

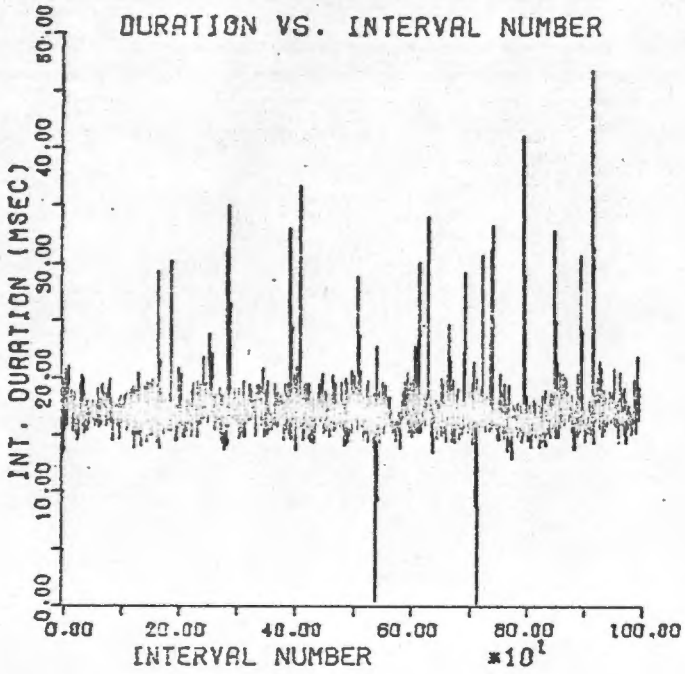


CALIBRATION OF PROG.INT.TIMER
HP PULSE GENERATOR 11SEP80



APPENDIX G.

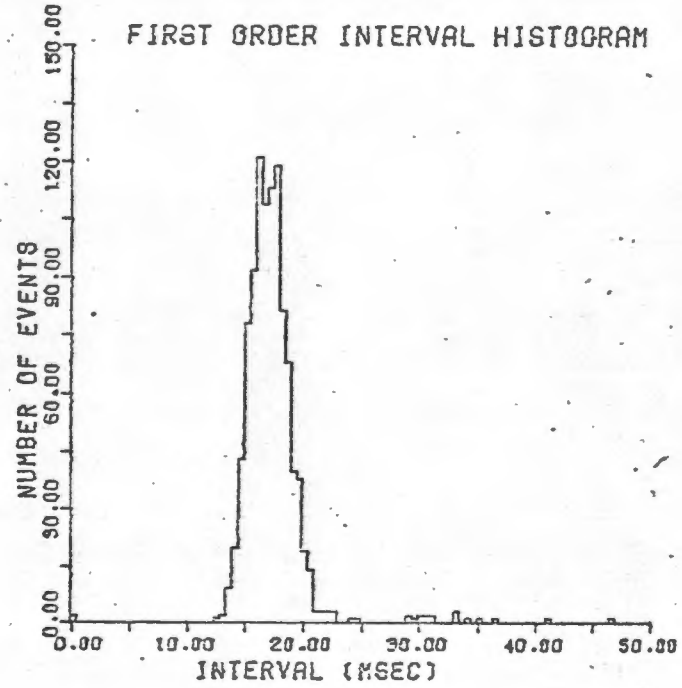
SPIKE DATA RESULTS

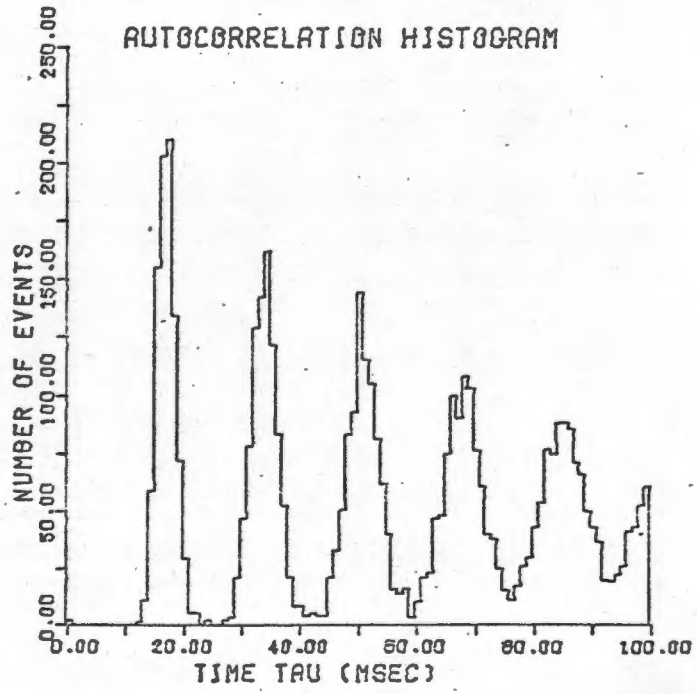


TEST OF PROG. INTERVAL TIMER

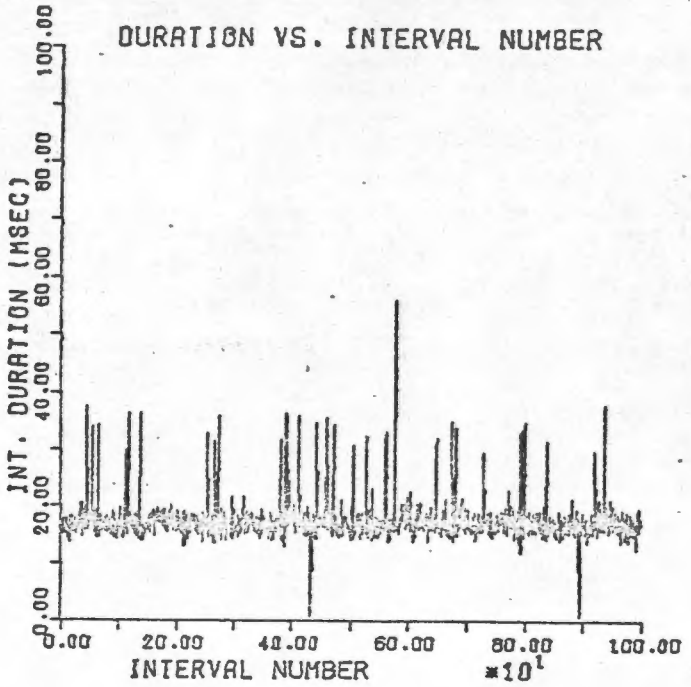
NUCL.GIGANTOCELL. DATA
CELL NUMBER DATA0

PROGRAM CELLREPORT.FR REV1.0





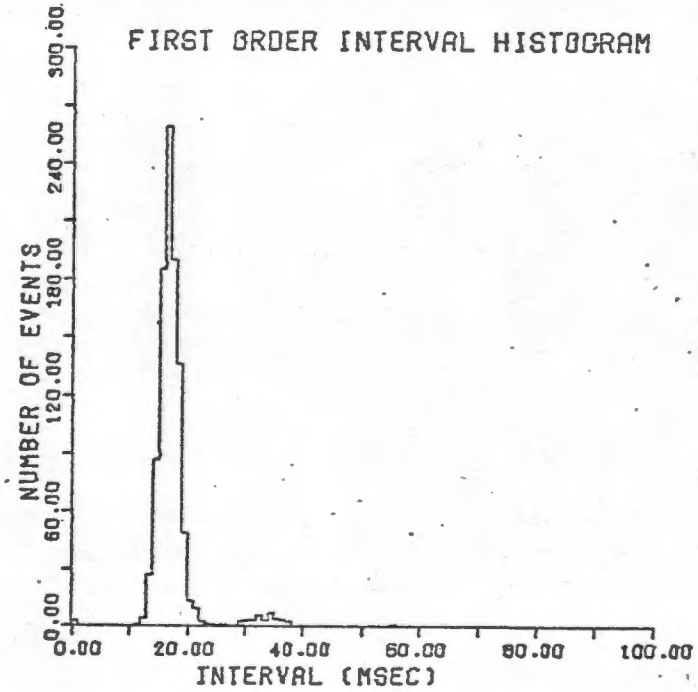
DURATION VS. INTERVAL NUMBER



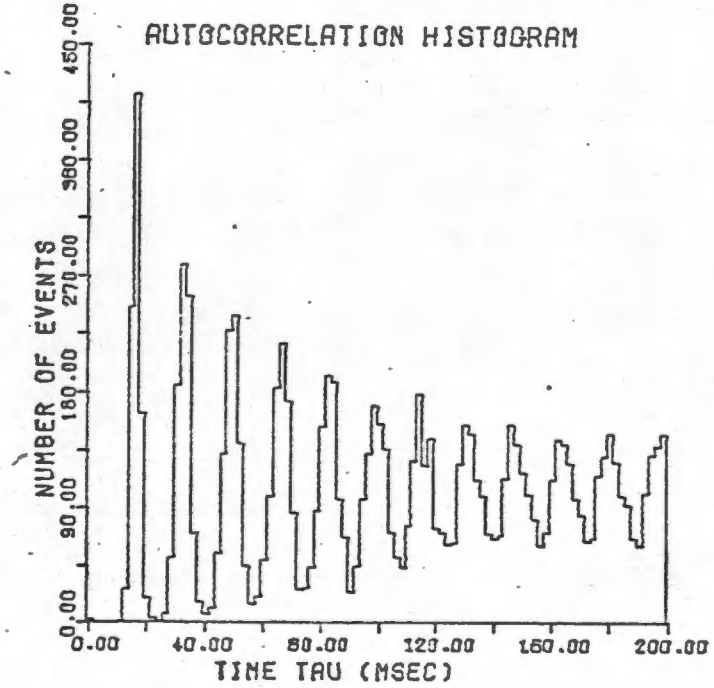
TEST OF PROG. INTERVAL TIMER

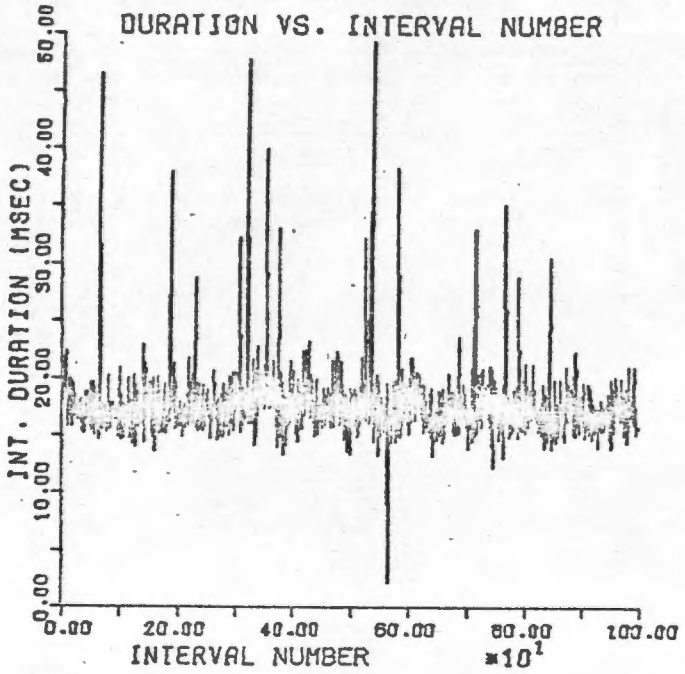
NUCL.GIGANTOCELL. DATA
CELL NUMBER DATA5
PROGRAM CELLREPORT.FR REV1.0

FIRST ORDER INTERVAL HISTOGRAM



AUTOCORRELATION HISTOGRAM

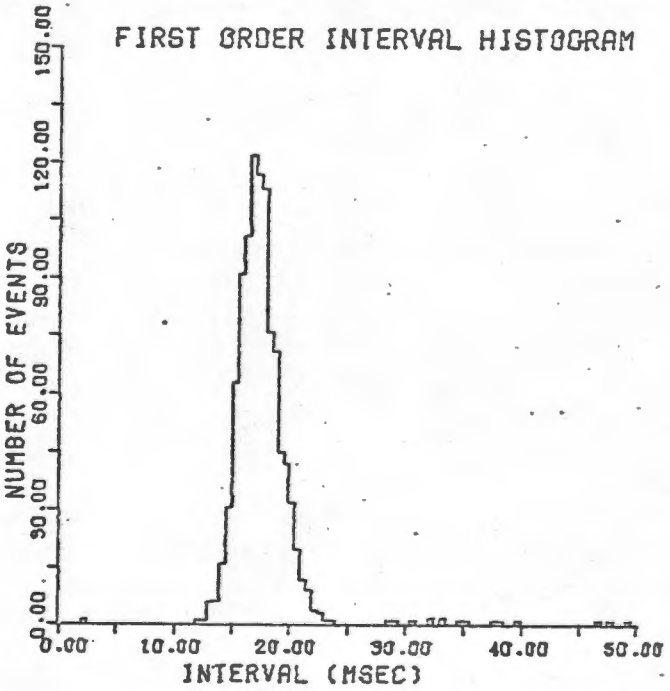


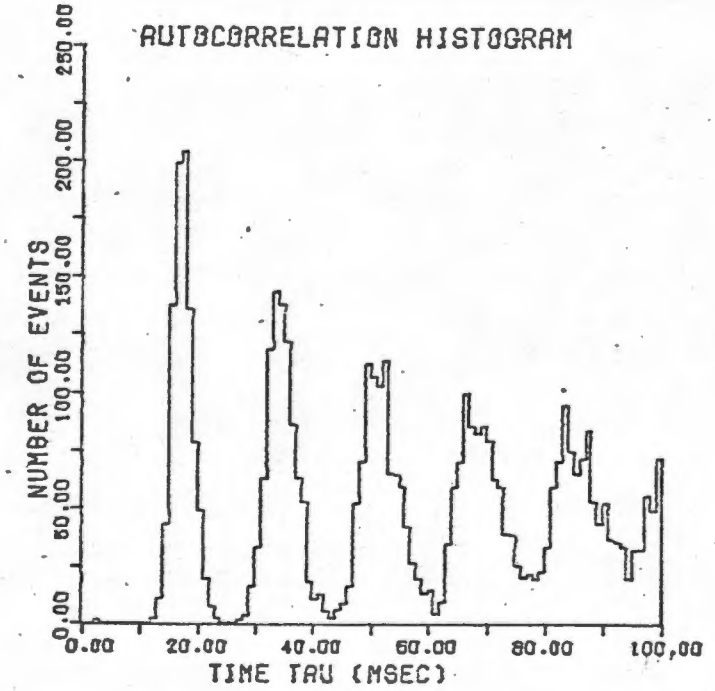


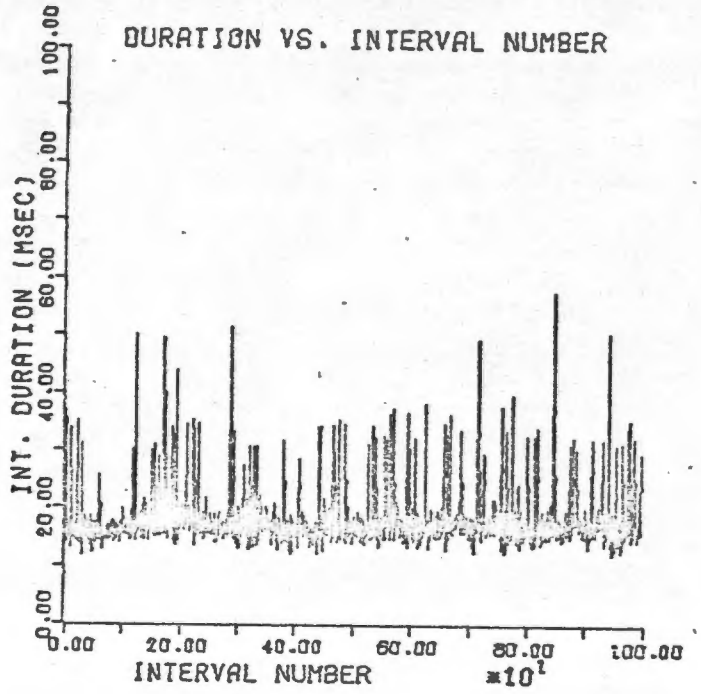
TEST OF PROG. INTERVAL TIMER

NUCL.GIGANTOCELL. DATA
CELL NUMBER DATA10

PROGRAM CELLREPORT.FR REV1.0



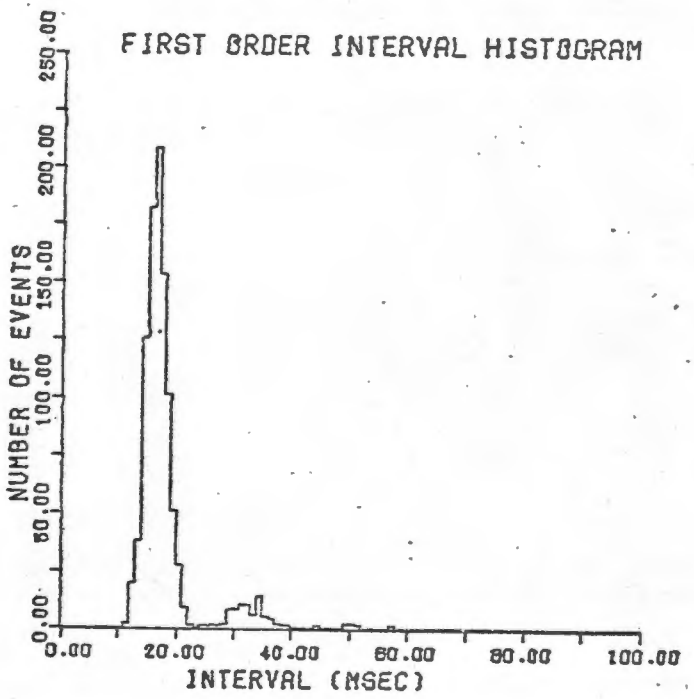


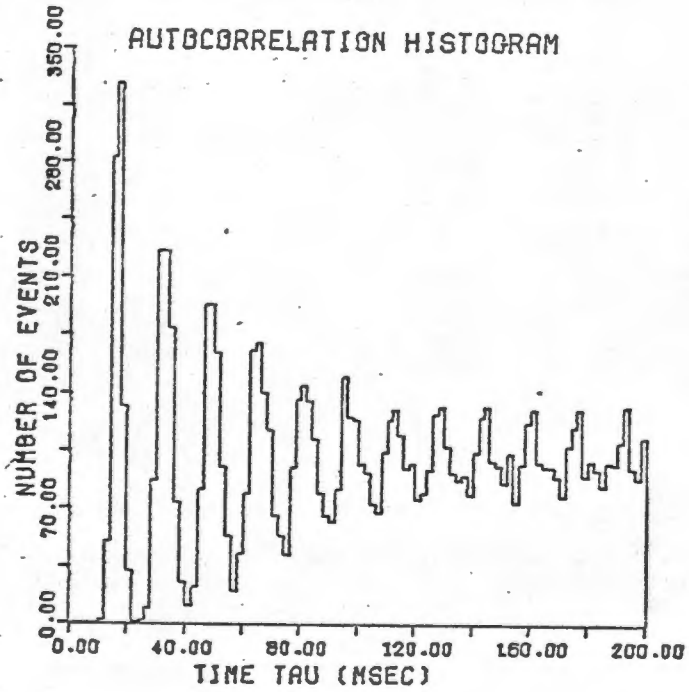


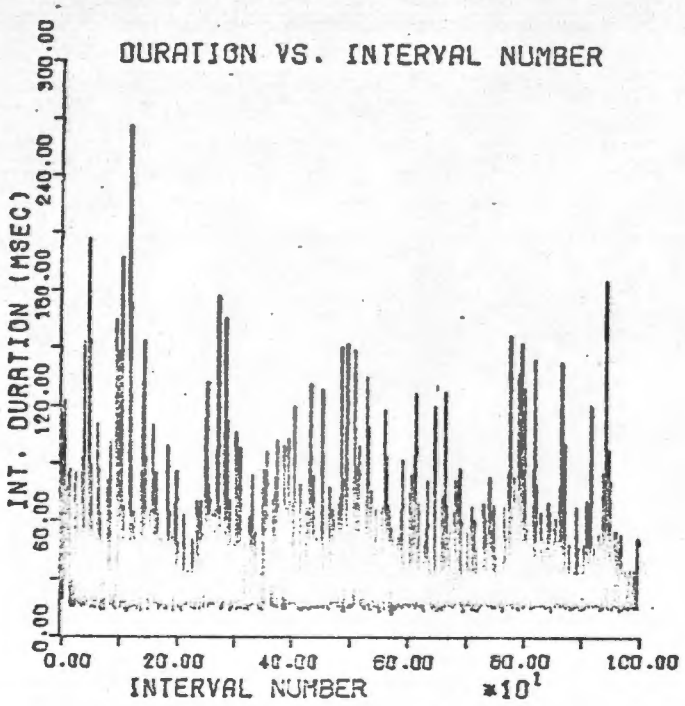
TEST OF PROG. INTERVAL TIMER

NUCL.GIGANTOCELL. DATA
CELL NUMBER DATA15

PROGRAM CELLREPORT.FR REV1.0



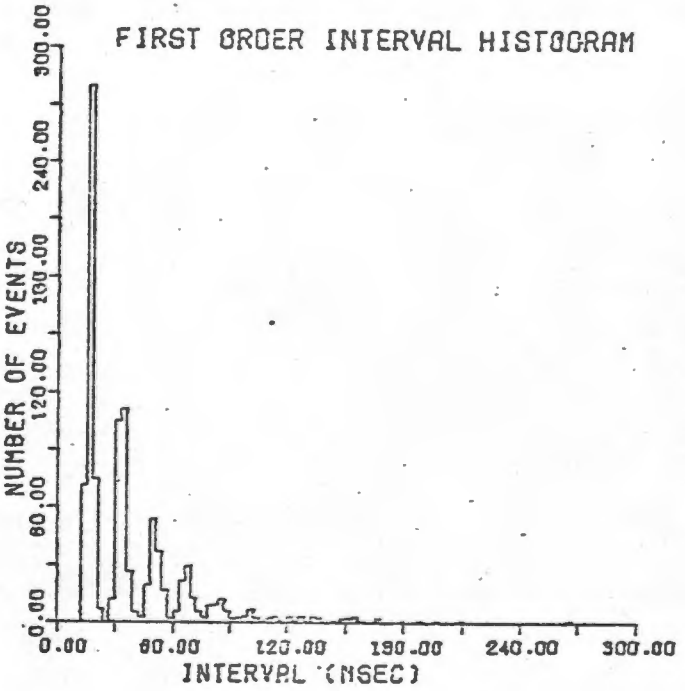


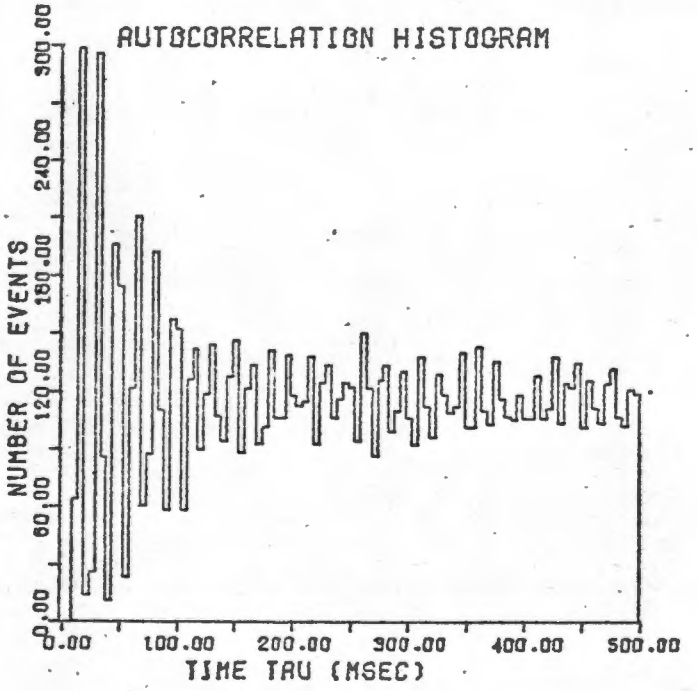


TEST OF PROG. INTERVAL TIMER

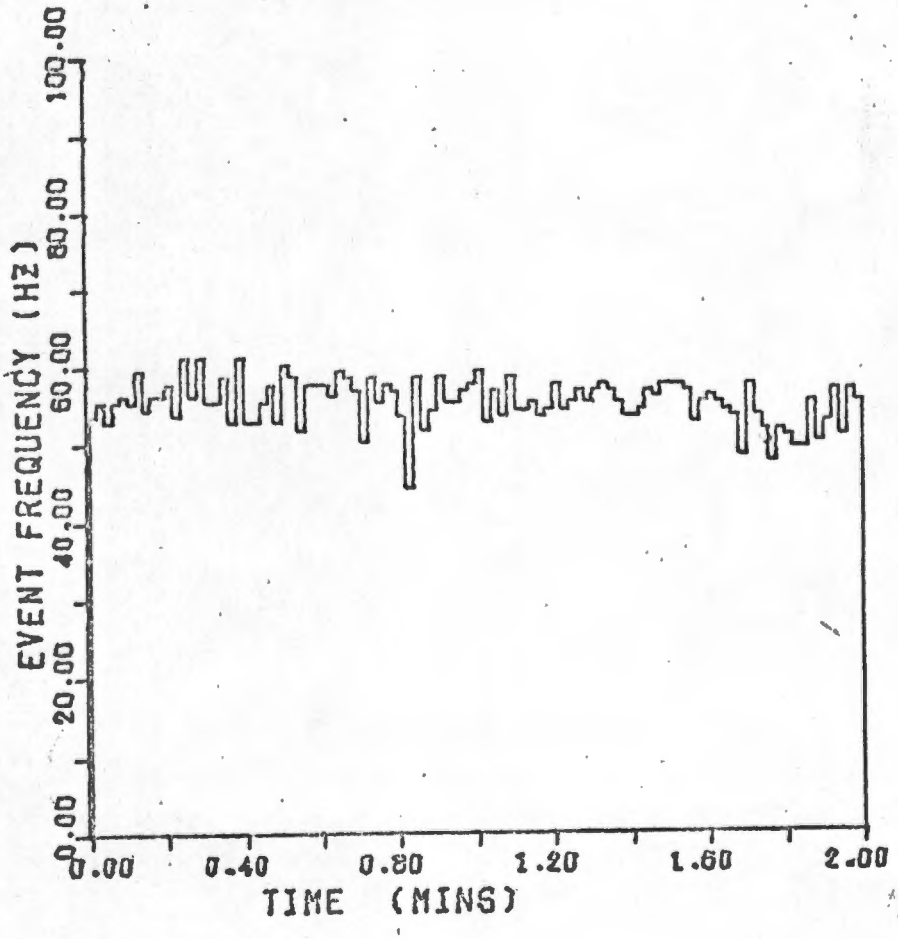
NUCL.GIGANTOCELL. DATA
CELL NUMBER DATA40

PROGRAM CELLREPORT.FR REV1.0





TEST OF PROG.INTERVAL TIMER
CONTINUOUS NUCL.GIGANTOCELL. DATA



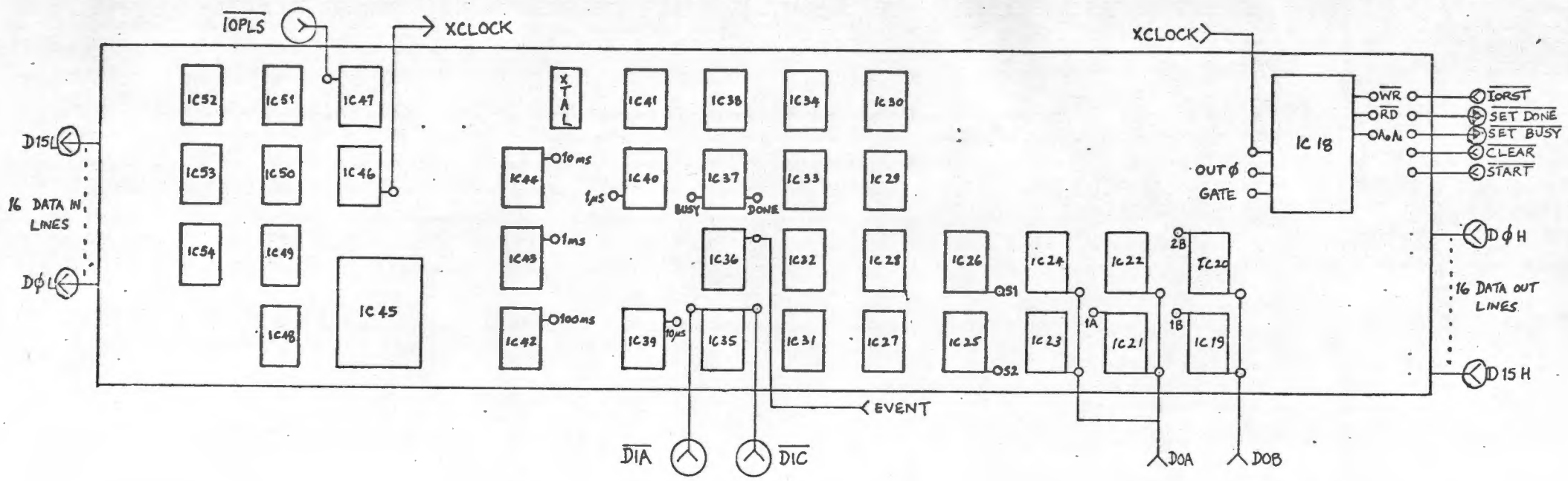
APPENDIX H.

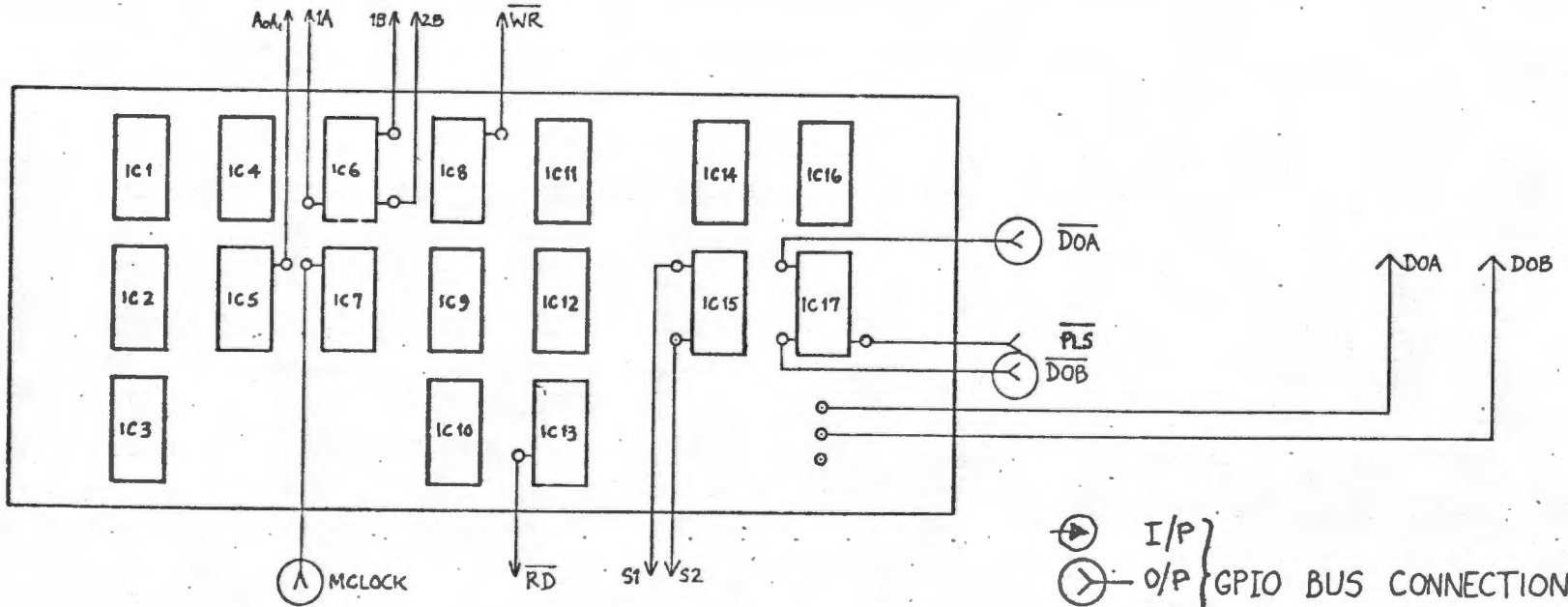
PIT CIRCUIT LAYOUT

The PIT circuit layout diagram shows points where control signals can be tested. These points are wire-wrap pins that are soldered into the veroboard.

Non of the circuitry that was designed is fitted onto the GPI board. All the designed circuitry is built on veroboard and it is connected to the GPI board via edge connectors and ribbon-cable.

The maximum current drawn by the circuit is 0,75A and this allows the circuit to be attached to the +5V line on the GPI. The GPI +5V line allows for a maximum current of 1A.





PIT CIRCUIT LAYOUT

APPENDIX I.

PARTS LISTING

PARTS LISTING

1. INTERGRATED CIRCUITS

IC NO.	DESCRIPTION	COST R.C	IC NO.	DESCRIPTION	COST R.C
1	7473	.40	28	7438	.50
2	7473	.40	29	7438	.50
3	7473	.40	30	7438	.50
4	7400	.30	31	7438	.50
5	7432	.30	32	7438	.50
6	7432	.30	33	7438	.50
7	7408	.30	34	7438	.50
8	7432	.30	35	7404	.40
9	7408	.30	36	7408	.30
10	74121	.50	37	7474	.50
11	7474	.50	38	7474	.50
12	7474	.50	39	7490	.50
13	74121	.50	40	7404	.40
14	7414	.30	41	7432	.30
15	7402	.30	42	7490	.50
16	7408	.30	43	7490	.50
17	7414	.30	44	7490	.50
18	INS8253	14.00	45	74154	3.00
19	74LS374	2.00	46	7430	.30
20	74LS374	2.00	47	7432	.30
21	74LS374	2.00	48	7474	.50
22	74LS374	2.00	49	7432	.30
23	74LS374	2.00	50	74121	.50
24	74LS374	2.00	51	7408	.30
25	74LS374	2.00	52	74121	.50
26	74LS374	2.00	53	74121	.50
27	7438	.50	54	7476	.80

1 MHz Crystal R10.00
54 Sockets at .40 cents each

2. MISCELLANEOUS COMPONENTS

R1	510R	C1	10nF	
R2	1K	C2	10nF	
R3	510R	C3	10nf	
R4	3K3	C4	22 uF	16V tantalum
R5	3K3	C5	1 uF	
R6	2K2			

D1	1N914

Veroboard	R5.00
Edge Connectors	R5.00
Wire wrap pins	R8.00
Ribbon Cable	R3.00