To the Graduate Council:

I am submitting herewith a thesis written by Nabil Ibrahim Tmeizeh entitled "An Investigation of Different Microprocessor-Based Organizations for Systems Generating Histograms." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Robert E. Bodenheimer
Major Professor

We have read this thesis and recommend its acceptance:

Robert W. Rochelle

Donald W. Bouldin

Accepted for the Council:

L. Evans Peck
Vice Chancellor
Graduate Studies and Research

# AN INVESTIGATION OF DIFFERENT MICROPROCESSOR-BASED ORGANIZATIONS FOR SYSTEMS GENERATING HISTOGRAMS

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Nabil Ibrahim Tmeizeh

June 1981

3051882

ACKNOWLEDGMENTS

The author would like to express gratitude to Dr. Robert E. Bodenheimer for suggesting the thesis topic and for helping to develop, guide, and proofread this thesis. Thanks are extended to Dr. D. W. Bouldin and to Dr. R. W. Rochelle for serving on the author's committee.

A word of thanks goes to Student Aid International for their partial financial support.

Finally and most important, I wish to express my indebtedness and appreciation to my parents, Ibrahim and Na'eema, for their total support, patience, and understanding throughout my academic career.

ABSTRACT

This thesis investigated the limitations of different microprocessor-based organizations for systems generating Time (interval and dwell) and amplitude histograms. Applying successive design techniques on the existing hardware developed an optimal architecture for the interval histogram processor. The same architecture was utilized to generate dwell and amplitude histograms. Analysis on the above processors revealed that architectures for histogram systems were best obtained by using multiprocessing and software minimization. The Interval processor was built by laboratory kits to demonstrate the feasibility of multiprocessing.

TABLE OF CONTENTS


| CHAPTER | | PAGE |
|---|---|---|
| 1. | INTRODUCTION | 1 |
| 2. | HISTOGRAMS | 3 |
| | Time Histograms | 3 |
| | Amplitude Histogram | 5 |
| | Possible Methods for Implementation | 7 |
| | Time histograms | 7 |
| | Amplitude histogram | 9 |
| 3. | TIME HISTOGRAMS | 11 |
| | Interval Histogram | 11 |
| | Single microprocessor system | 11 |
| | Two-microprocessor system with ADC interface | 24 |
| | Two-microprocessor system with Schmidt Trigger interface | 38 |
| | Modified two-microprocessor system | 47 |
| | Dwell Histogram | 56 |
| | Input MPU operation | 57 |
| | Display MPU operation | 63 |
| | Analysis | 64 |
| 4. | AMPLITUDE HISTOGRAM | 66 |
| | Input Microprocessor | 66 |
| | Display Microprocessor | 70 |
| | Analysis | 72 |
| | Conclusion | 73 |
| 5. | RESULTS | 74 |
| | Results | 74 |
| 6. | CONCLUSIONS | 79 |
| LIST OF REFERENCES | | 81 |
| APPENDIXES | | 83 |
| | Appendix A | 84 |
| | Appendix B | 105 |
| | Appendix C | 107 |
| VITA | | 114 |

LIST OF TABLES


| TABLE | | PAGE |
|---|---|---|
| 3.1. | Memory Size and Counter's Order Requirements as a Function of $f_{CLK}$ . . . . . | 55 |
| 3.2. | Counter's Clock Frequency as a Function of $T_h$ . . . . . | 55 |
| C-1. | Expected and Experimental Histogram Using $f_{CLK}$ = 1 kHz . . . | 109 |
| C-2. | Expected and Experimental Histogram Using $f_{CLK}$ = 10 kHz . . | 110 |
| C-3. | Expected and Experimental Histogram Using $f_{CLK}$ = 1 MHz . . . | 111 |
| C-4. | Expected Histogram Using $f_{CLK}$ = 100 kHz . . . . . | 112 |
| C-5. | Experimental Histogram for Intervals of Table C-4 . . . . . | 113 |

LIST OF FIGURES


| FIGURE | | PAGE |
|---|---|---|
| 2.1. | Histogram Modes (a) Nonsequential and (b) Sequential . . . | 4 |
| 2.2. | Period Detection against Threshold . . . . . | 4 |
| 2.3. | Dwell Interval Measurement . . . . . | 6 |
| 2.4. | Amplitude Histogramming by Sampling . . . . . | 6 |
| 2.5. | Interval Measurement with High $f_{CLK}$ . . . . . | 8 |
| 2.6. | Threshold Detector Performance (a) Input Signal and (b) Output Signal . . . . . | 10 |
| 3.1. | Analog Signal and Its Intervals Relative to Threshold . . . | 12 |
| 3.2. | Block Diagram: Single MPU Interval Processor . . . . . | 13 |
| 3.3. | Single MPU System Software Flowchart (a) Initialization Program, (b) Clock ISR Routine, (c) Data Acquisition and Threshold Detection Program, (d) Histogram Program, and (e) Display Program . . . . . | 14 |
| 3.4. | Block Diagram: Two-Microprocessors Interval Processor . . . . . | 25 |
| 3.5. | Interval Processor with ADC Interface Input MPU Flowchart: (a) Initialization Routine, (b) Data Acquisition and Threshold Detection Routine, (c) Clock ISR Routine, and (d) Histogramming and Data Transfer Routine . . . . . | 26 |
| 3.6. | Interval Processor with ADC Interface: Display MPU Flowchart (a) Initialization Routine, (b) Data Update ISR Routine, and (c) Display Routine . . . . . | 33 |
| 3.7. | Schmidt Trigger's Input and Output . . . . . | 39 |
| 3.8. | Block Diagram: Interval Processor with Schmidt Trigger Interface . . . . . | 40 |
| 3.9. | Interval Processor with Schmidt Trigger Interface: Input MPU Flowchart (a) Initialization Program, (b) Interval Measurement Routine, and (c) Schmidt Trigger ISR Routine . . . . . | 41 |

| FIGURE | | PAGE |
|---|---|---|
| 3.10. | Modified (Final) Interval Processor: Block Diagram and Interface . . . . . | 48 |
| 3.11. | Modified Interval Processor: Input MPU Flowchart (a) Initialization and Startup Routine and (b) Interval Read and Data Transfer ISR Routine . . . . . | 50 |
| 3.12. | Modified Interval Processor Input MPU Interface Timing Diagrams . . . . . | 52 |
| 3.13. | Dwell Processor Block Diagram and Interface . . . . . | 58 |
| 3.14. | Dwell Processor Timing Diagrams . . . . . | 59 |
| 3.15. | Dwell Processor Flowchart (a) Initialization Routine, (b) Polling Routine, (c) Schmidt Trigger's Output ISR Routine, and (d) Inverted Schmidt Trigger's Output ISR Routine . . . . . | 60 |
| 4.1. | Amplitude Processor Block Diagram and Interface . . . . . | 67 |
| 4.2. | Amplitude Histogram: Input MPU Flowchart (a) Initialization and Startup Routine and (b) Data Acquisition and Transfer ISR Routine . . . . . | 68 |
| 4.3. | Control of ADC and MPU2 Timing Diagrams . . . . . | 71 |
| 5.1. | Tested Interval Processor: Block Diagram and Interface . . . . . | 75 |
| 5.2. | Experimental Time Histograms Plots (a) Table C-1 Plot, (b) Table C-2 Plot, and (c) Table C-3 Plot . . . . . | 77 |
| A.1. | Zero Crossing Uncertainty . . . . . | 86 |
| A.8.1. | Generated Random Signal of Example A.8.1 . . . . . | 104 |
| B.1. | Frequency Generation and Binary Counter Interface . . . . | 106 |

# CHAPTER 1

## INTRODUCTION

The limitations of microprocessor-based systems designed to perform real-time signal processing impose bounds on the signal's frequency content. Fast time signals require a high sampling rate by the microprocessor and, hence, a fast interface (1). In the case of slow signals a single MPU system might be sufficient to acquire data, process it, and display results on a cathode-ray-tube (CRT) screen. However, such a system is too slow to process fast analog signals (2).

The above discussion raises several questions concerning microprocessor-based digital signal processing. For a specific design configuration what is the system's speed and its sampling rate? If determined, what are the implications on the signal's frequency content? How does an intended task dictate the design scheme? Is a single microprocessor system sufficient or should multiprocessing be considered? (3)

This thesis provides some answers to these questions by considering the design constraints on systems performing real-time processing on signals. Measuring parameters of time and the distribution of amplitude is an important aspect of the digital processing of neurological signals (4). In studying the design constraints of microprocessor-based systems, two tasks are to be implemented: (1) the measurement of time parameters to produce time histograms and (2) the measurement of

amplitude distributions to produce amplitude histograms (4). In both cases the histograms are to be displayed on a CRT screen.

In Chapter 2 the nature of interval, dwell, and amplitude histograms is presented to shed some light on possible systems for performing these tasks. An initial system is proposed in Chapter 3 to produce an interval histogram. Alternative schemes are introduced and analyzed. Further improvements yield a two-processor system in which the limitations and capabilities are evaluated. Performing the dwell histogram is made feasible by slight adaptations of the interval processor. Chapter 4 is dedicated to the design of a microprocessor system which constructs the amplitude histogram. The interval processor designed in Chapter 3 was built and tested and the results are illustrated in Chapter 5. Finally, conclusions and considerations for future work are presented in Chapter 6.

# CHAPTER 2

# HISTOGRAMS

The histograms to be considered involve measurement of time parameters and distribution of amplitude in waveforms. These histograms can be generated in two ways, nonsequentially and sequentially (1). In the first case (Fig. 2.1a) the value of the measurement defines the address of a memory location. The content of that location is incremented by one each time a measurement occurs. For the sequential mode (Fig. 2.1b) the values of successive measurements are placed in successive memory locations (4). In this thesis the nonsequential mode of histogramming is chosen.

There are two categories of histograms to be generated: (1) time histograms, and (2) amplitude histograms. Time histograms include two parameters to be measured:

1. Interval time.
2. Dwell time.

The second category includes the measurement of amplitudes.

## 2.1 Time Histograms

The interval histogram represents the number of occurrences (i.e. the frequency) of every period present in the analog signal. For example, if period $T_o$ is present in the waveform of Figure 2.2, the address location corresponding to $T_o$ is incremented by one whenever

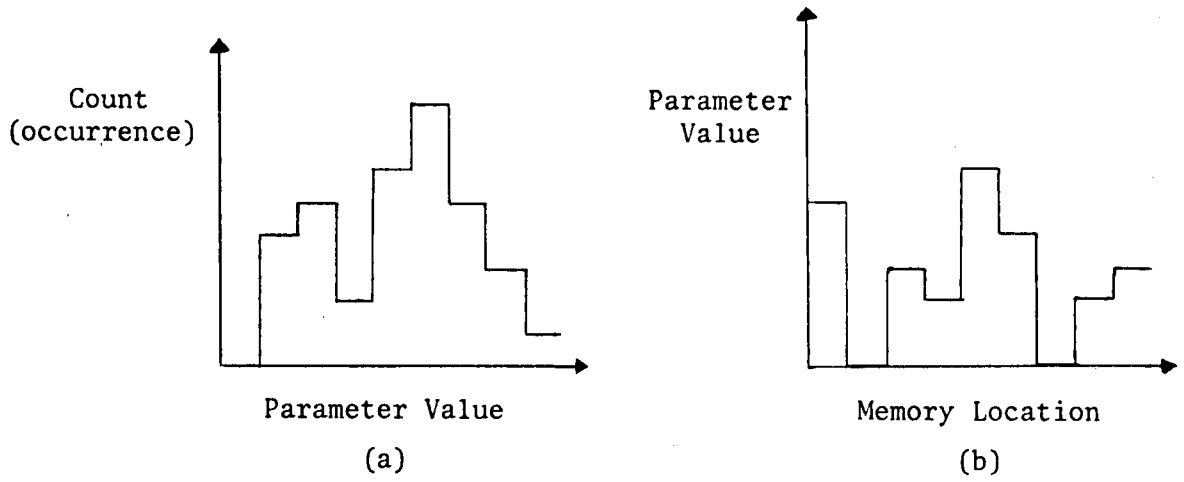



Figure 2.1. Histogram Modes (a) Nonsequential and (b) Sequential.

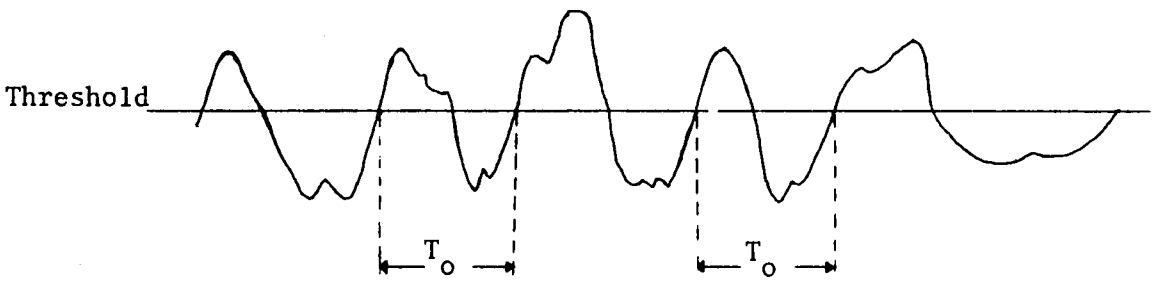



Figure 2.2. Period Detection against Threshold.

$T_o$ is detected. There are exactly 2 periods of length $T_o$ in the given waveform.

Occurrence of a pulse is detected when the signal crosses a preset threshold level in the positive-going direction. Similarly, by detecting both the positive and negative-going edge of an analog signal through a preset threshold level (Fig. 2.3), the length of time the signal stays above that level is represented as a dwell histogram. The nonsequential mode quantizes these dwell times as memory addresses and increments the content of that address location for each dwell time detected (4).

## 2.2 Amplitude Histogram

The second category of histograms to be generated is the amplitude histogram. This histogram plots the amplitude distribution between a succession of waveform samples. By quantizing the measured voltages after sampling, an amplitude histogram is generated in the following way. Each voltage sample is quantized as an address location. The content of that location is to be incremented by one each time the signal is quantized to that address (4). As an example, there are 10 points in the signal of Figure 2.4 where the amplitude equals $A_i$. Seven of those points were quantized. It follows that the address location corresponding to $A_i$ is incremented seven times and finally holds the value 7.

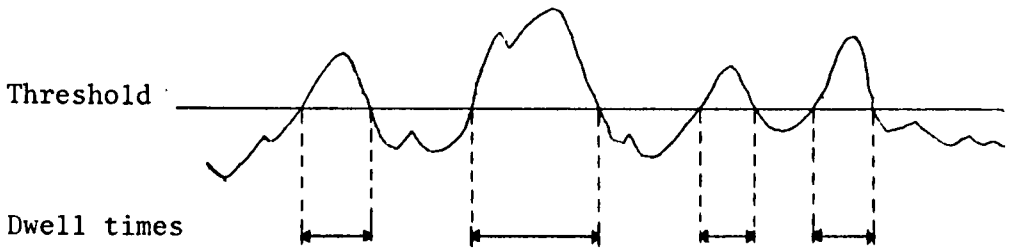



Figure 2.3. Dwell Interval Measurement.

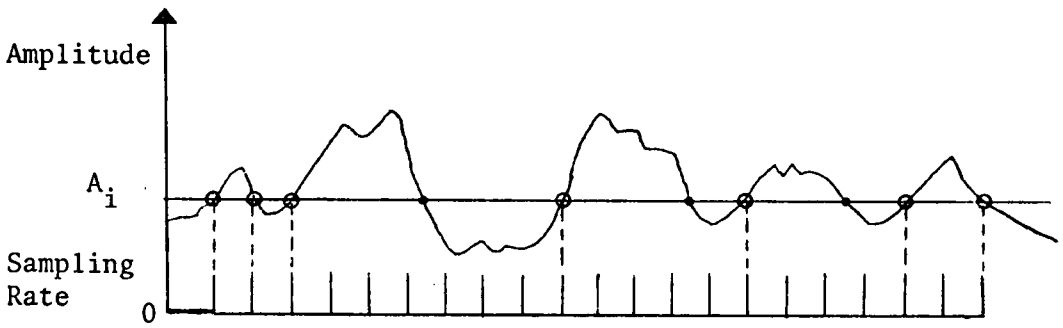



Figure 2.4. Amplitude Histogramming by Sampling.

## 2.3 Possible Methods for Implementation

### 2.3.1 Time Histograms

In performing time histograms, equivalent techniques apply to both types with few variations. Therefore, by considering the interval histogram, the dwell is easily implemented using the same basic approach. In an interval histogram the idea is to detect the end of a period of the analog signal. There are two techniques attempted here:

1. Digitizing the signal with an analog-to-digital Converter (ADC),

and

2. Detecting threshold level crossing by a Threshold Detector.

In the first technique the ADC acquires data, hopefully at a fast rate, and inputs it to the microprocessor (MPU) which processes the samples to determine where the signal crosses the preset threshold level. As the conversion and the detection process continues, an external frequency clock interrupts the MPU at defined periods of time. For each interrupt the MPU software increments a count memory location. As soon as the MPU detects that the analog signal has made 3 threshold level crossings, it disables the clock, reads the count value, increments the corresponding memory location, clears the count, and re-enables the clock (Fig. 2.5). The process continues until an interval histogram is obtained.

The second technique replaces the ADC by a Threshold Detector which produces an output that has a digital form of the input analog signal. There are several types of threshold detectors (5). The one chosen

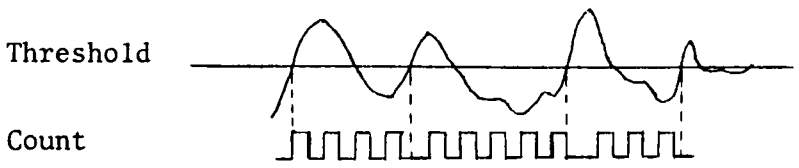



Figure 2.5. Interval Measurement with High $f_{CLK}$.

for the interval histogram produces a constant voltage, $V_o$, whenever the signal makes a positive-going transition above the preset threshold level. The output stays at that level, $V_o$, until the signal makes a negative-going transition below the preset threshold level, at which point the output of the detector becomes a zero voltage.

The output of the threshold detector (Fig. 2.6b) for the input signal of Figure 2.6a is interfaced to the MPU such that the detector's output interrupts the MPU every time the output makes a positive-going transition corresponding to the end of a period.

There is little difference, beyond this stage, between this method and the previous one. The high-frequency clock counts until another positive-going edge occurs at which point the clock is disabled, the count is read, the histogram is collected, the count initialized, and the clock is resumed. Notice that in both techniques the MPU waits until the first positive-going edge of the signal is encountered, at which time initialization begins, and the interrupts are enabled.

### 2.3.2 Amplitude Histogram

The fact that this histogram displays plots of the distribution of amplitudes requires that every sample be an input to the MPU and that it be processed. Therefore, it dictates that the first method of digitization be utilized. Figure 2.4 illustrates the fundamentals of this process. It is, however, of utmost importance to have a sampling rate as high as possible in order to guarantee a fairly accurate histogram (1).

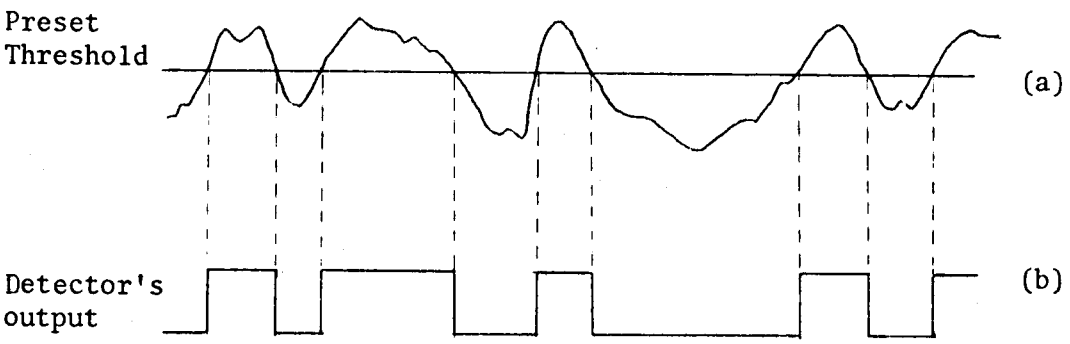



Figure 2.6. Threshold Detector Performance
(a) Input signal and (b) Output signal.

# CHAPTER 3

# TIME HISTOGRAMS

## 3.1 Interval Histogram

As discussed in Chapter 2, the design of a system for generating the interval histogram constitutes the main body of this chapter. Several systems are designed for which the capabilities and limitations are discussed.

### 3.1.1 Single Microprocessor System

In attempting to construct an interval histogram for random analog signals similar to that of Figure 3.1, an initial design configuration was considered by interfacing an analog-to-digital converter (ADC), a frequency clock, and a display unit to a microprocessor as shown in Figure 3.2.

The block diagram of Figure 3.2 illustrates that in addition to acquiring data, the MPU is burdened with two more tasks: the counting process which determines the length of a period and the display of results on a Cathode-Ray Tube (CRT).

Initially the MPU, as indicated by the flowchart shown in Figure 3.3a, waits for the signal's first positive-going transition. At this point the MPU enables the free running clock's interrupt and initializes the count. The clock continues to interrupt the MPU causing count to be incremented (Fig. 3.3b). As soon as the end of

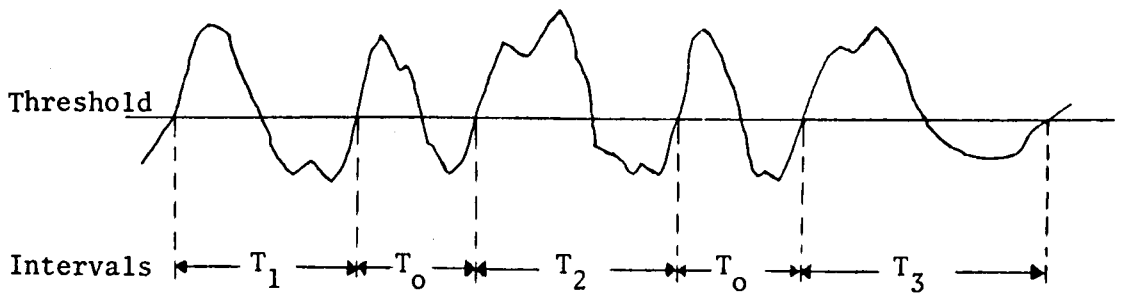



Figure 3.1. Analog Signal and Its Intervals Relative to Threshold.

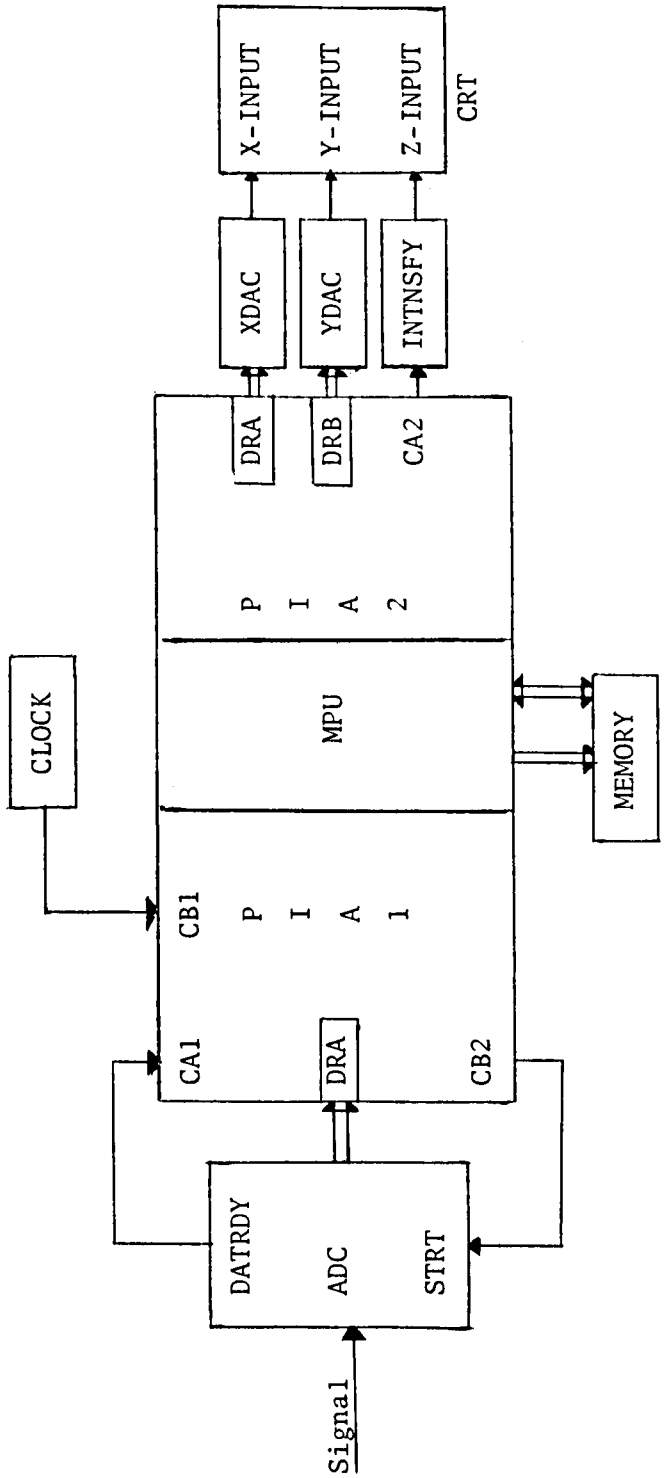
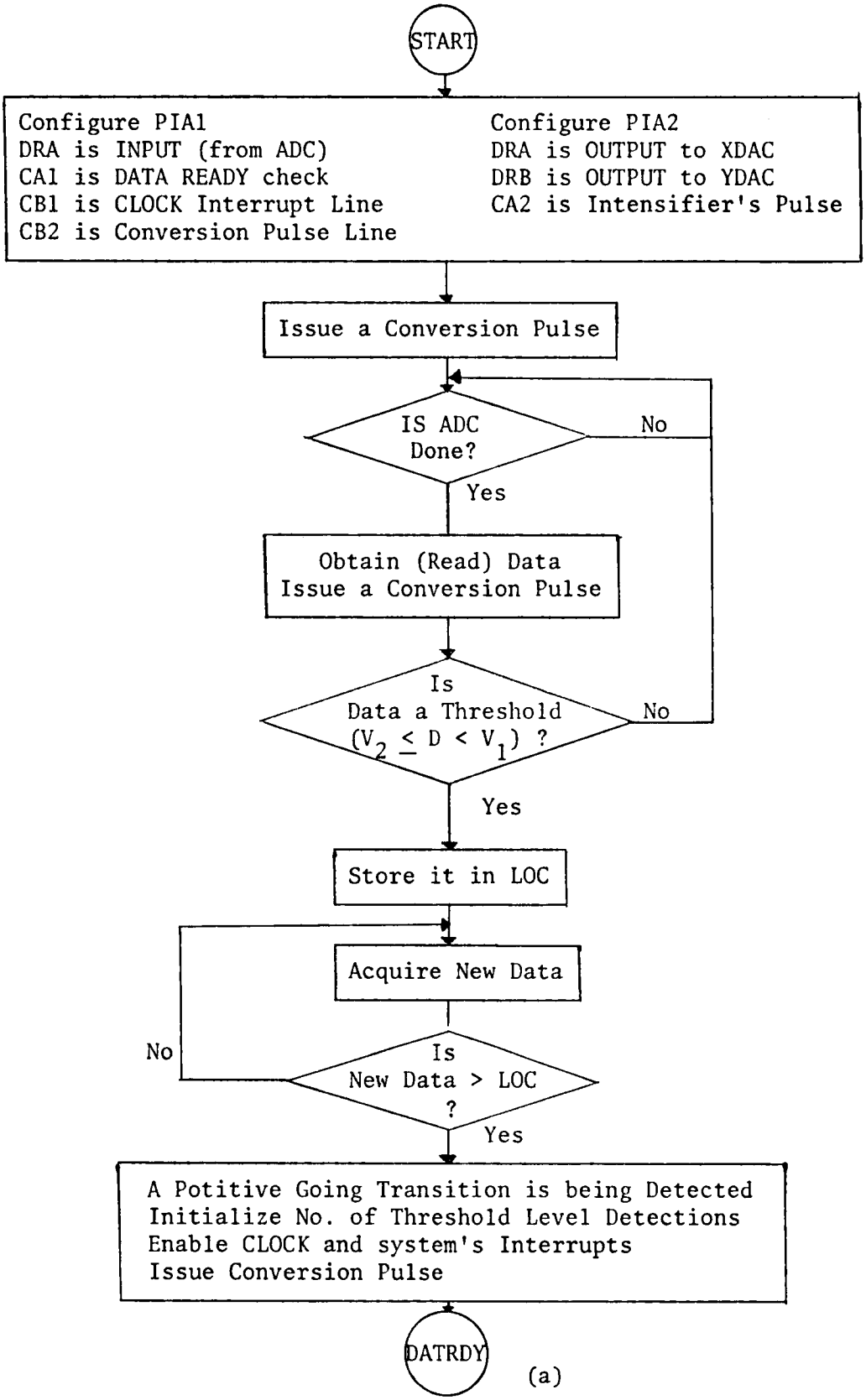


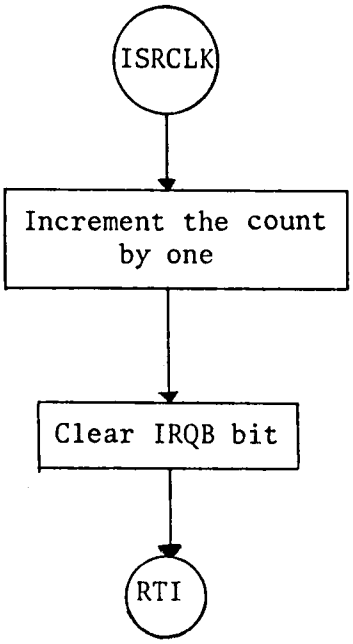

Figure 3.2. Block Diagram: Single MPU Interval Processor.

Figure 3.3. Single MPU System Software Flowchart (a) Initialization program, (b) Clock ISR routine, (c) Data acquisition and threshold detection program, (d) Histogram program, and (e) Display program.


START
Configure PIA1
DRA is INPUT (from ADC)
CA1 is DATA READY check
CB1 is CLOCK Interrupt Line
CB2 is Conversion Pulse Line
Configure PIA2
DRA is OUTPUT to XDAC
DRB is OUTPUT to YDAC
CA2 is Intensifier's Pulse
Issue a Conversion Pulse
IS ADC Done?
No
Yes
Obtain (Read) Data
Issue a Conversion Pulse
Is Data a Threshold ($V_2 \leq D < V_1$) ?
No
Yes
Store it in LOC
Acquire New Data
No
Is New Data > LOC ?
Yes
A Potitive Going Transition is being Detected
Initialize No. of Threshold Level Detections
Enable CLOCK and system's Interrupts
Issue Conversion Pulse
DATRDY

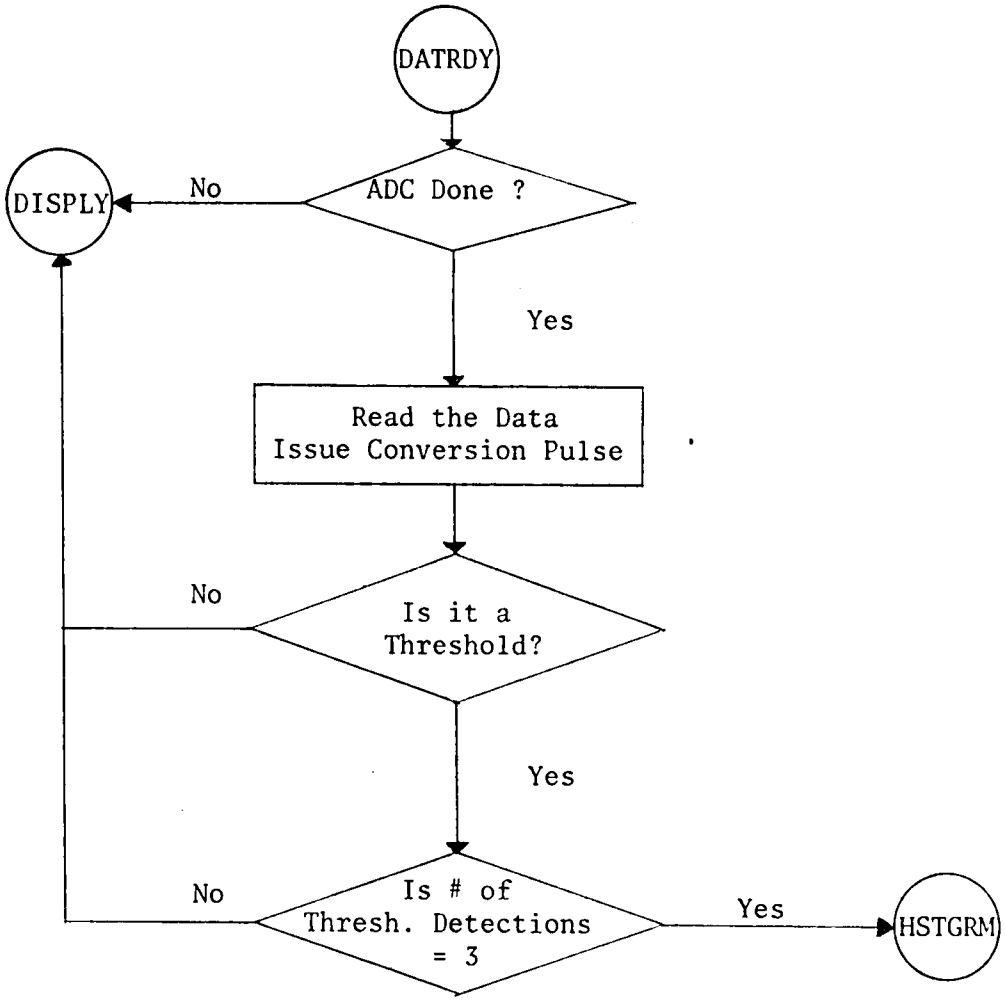
(a)

Figure 3.3



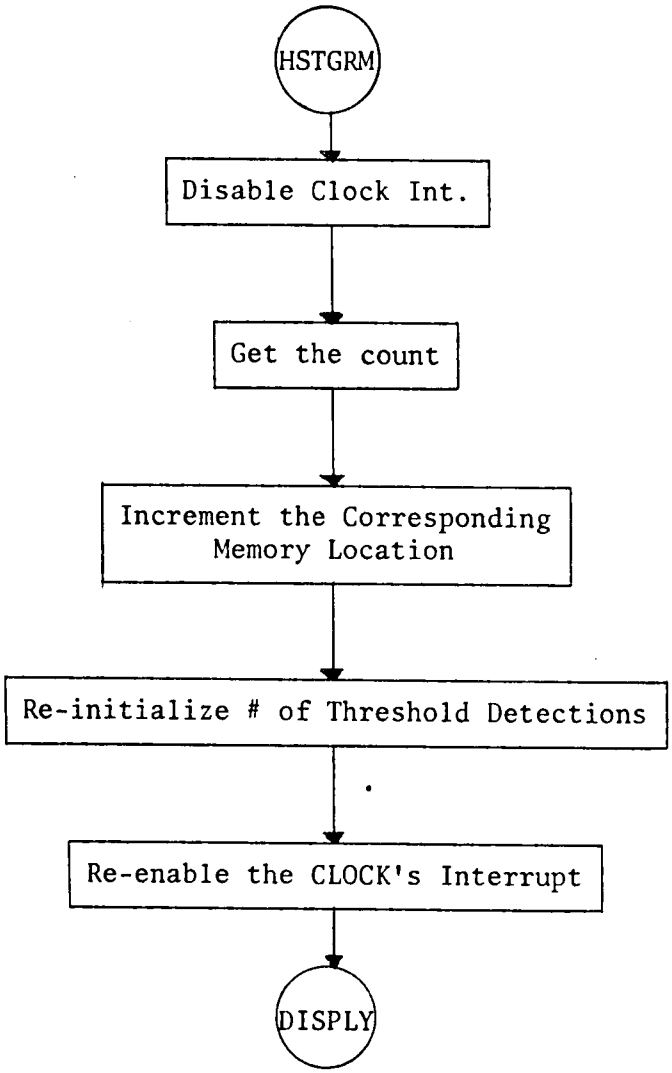

(b)

Figure 3.3 continued



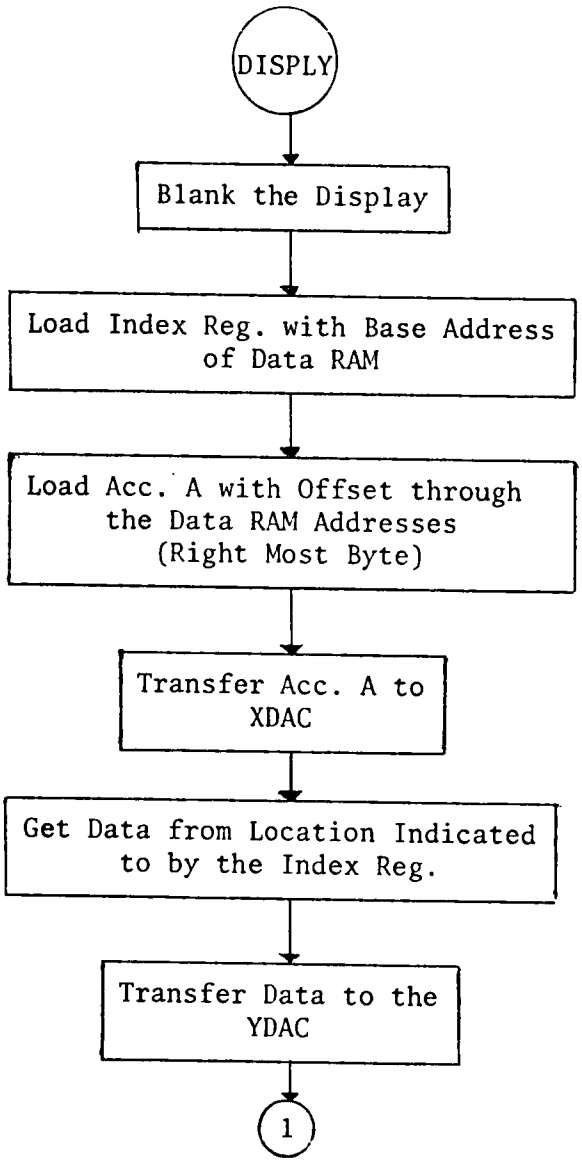

(c)

Figure 3.3 continued



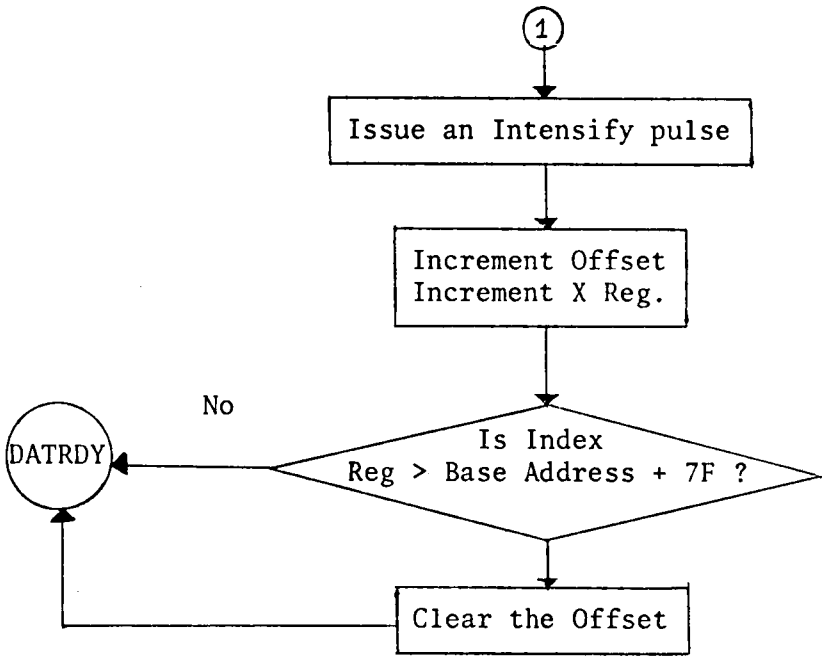

(d)

Figure 3.3 continued




(e)

Figure 3.3 continued




(e) continued

Figure 3.3 continued

one interval is detected (Fig. 3.3c), the MPU disables the clock interrupt, reads the count, treats count as an address, increments that address by one, reinitializes count, and re-enables the clock interrupt (Fig. 3.3d).

As data acquisition for the interval histogram procedes, the microprocessor is constantly running a display routine that plots the form of the histogram on a CRT (Fig. 3.3e). Under program control the address location, count, is transferred to the x-axis digital-to-analog Converter (XDAC), and the content of that address is transferred to the y-axis digital-to-analog Converter, YDAC (Fig. 3.2, p. 13). Both XDAC and YDAC are connected to the x and y inputs of the CRT while the third input, z, is controlled by the MPU to intensify the point displayed.

3.1.1.1 Analysis. The program listed in Appendix A.1 for the Motorola 6800 MPU indicates that the vast majority of the incoming data, excluding those points where the threshold level detections occur, is insignificant to the data acquisition routine. Therefore, the performance of the system is greatly deterred by testing every sample for threshold level crossing. Consequently, some speed restrictions are imposed on the data sampling rate, on the clock frequency, and on the the signal's frequency content.

Of significance, the execution time of four tasks determines the limitations on the system's parameters. The data acquisition execution time denoted $T_{ACQ}$, the histogramming time denoted $T_{HST}$, the execution time of the clock interrupt service routine, $T_{ISR}$, and the execution time taken by the display routine, $T_{DISP}$.

Most severe is the restriction imposed on the data sampling rate which, at the worst case, can be expressed as

$$T_s = T_{ACQ} + T_{HST} + T_{ISR} + T_{DISP} \cdot \tag{3.1}$$

However, at least four data samples are needed for the detection of one period in a signal. Thus, one has to sample at a new sampling period, in accordance with the sampling theorem (6), denoted $T_s'$ such that

$$T_s' = 4\ T_s \cdot \tag{3.2}$$

Based upon the program listed in Appendix A.1, the time parameters of Equation 3.1 were evaluated. Calculating

$T_{ACQ}$ = 53 cycles × 1.628 μsec/cycle,

$T_{HST}$ = 28 cycles × 1.628 μsec/cycle,

$T_{ISR}$ = 32 cycles × 1.628 μsec/cycle,

and

$T_{DISP}$ = 63 cycles × 1.628 μsec/cycle,

a sampling period of 1.14 msec is computed which dictates that the highest frequency content, $f_o$, in the analog signal cannot exceed 0.877 kHz.

Furthermore, the clock frequency is controlled by the program execution time, excluding the histogram time $T_{HST}$ during which the clock is disabled. Thus, the maximum clock frequency is given by

$$f_{clk} = \frac{1}{T_{ACQ} + T_{ISR} + T_{DISP}} . \qquad (3.3)$$

which requires that $f_{clk}$ be less than 4.167 kHz.

On the other hand, the display routine should be executed so frequently that a refreshing rate of at least 30 times/second is maintained. In one second, the number of display refreshes is given by

$$\text{Display Rate} \simeq \frac{1 - (T_{ISR})(f_{clk})}{(\text{No. RAM Locations})\ (T_{DISP} + T_{ACQ})} . \qquad (3.4)$$

where $T_{ISR} \times f_{clk}$ is the time consumed by the clock interrupts in one second. Equation 3.4 allows 27 refreshes per second which is below the minimum requirement. It is worth noting that the higher the clock frequency, the worse the display rate. For instance, a 10-kHz clock does not permit more than 17 refreshes per second which leaves the display in a serious flickering state.

Finally, the ADC speed is governed by the shortest path the MPU takes to check the ADC status a second time. This includes the display time, $T_{DISP}$, and the time taken to check the ADC status, $T_{DATRDY}$. Therefore, the ADC speed is given by

$$T_{ADC} = T_{DATRDY} + T_{DISP} , \qquad (3.5)$$

which requires that the ADC be faster than 116 μsec. Strictly speaking, the ADC can be much faster since new conversions occur only under program control, not on an interrupt basis.

Remember that the above results are entirely based on this analysis and on the execution time of the program provided in Appendix A.1.

### 3.1.2 Two-Microprocessor System with ADC Interface

The analysis made on the previous design scheme revealed how impotent a single MPU could become in performing data acquisition, time histogramming, and displaying of results. More powerful design concepts are to be introduced in order to improve the system's performance. The fact that a single MPU produces a flickering display makes the concept of multiprocessing so tempting that introducing a second microprocessor becomes a necessity. Considerable upgrading in performance is expected once the double-MPU system is configured. As depicted in the block diagram of Figure 3.4, tasks are distributed among both MPUs, where MPU1, referred to as the Input MPU, is in charge of data acquisition and time histogramming, and where MPU2, referred to as the Display MPU is solely displaying results on the CRT.

3.1.2.1 Input microprocessor. As illustrated in the flowchart of Figure 3.5 and the software listed in Appendix A.2, initialization, data acquisition, and interval measurement (Fig. 3.5a, b, c) are accomplished by the Input MPU in the same manner as in the previous system. However, a communication protocol between both microprocessors is established (Fig. 3.5d). The input MPU does not enable its interrupts (including system's) until the first positive-going threshold crossing occurs, after which it sends a high pulse on control line CA2 requesting the Display MPU to become operational (Fig. 3.5a). Should the content of a memory location in the Input MPU memory, M1, change, the Input MPU places the address of the corresponding

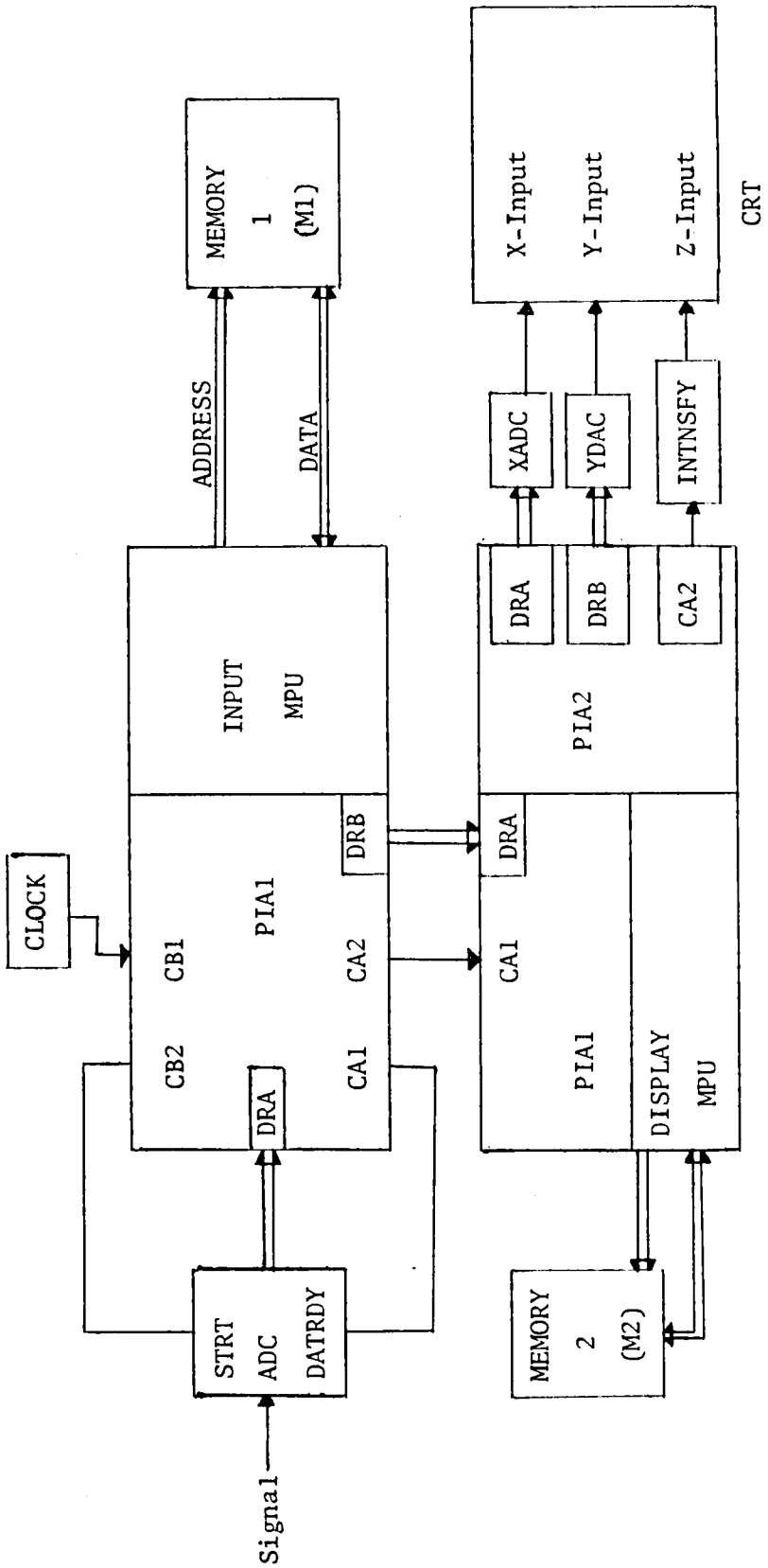
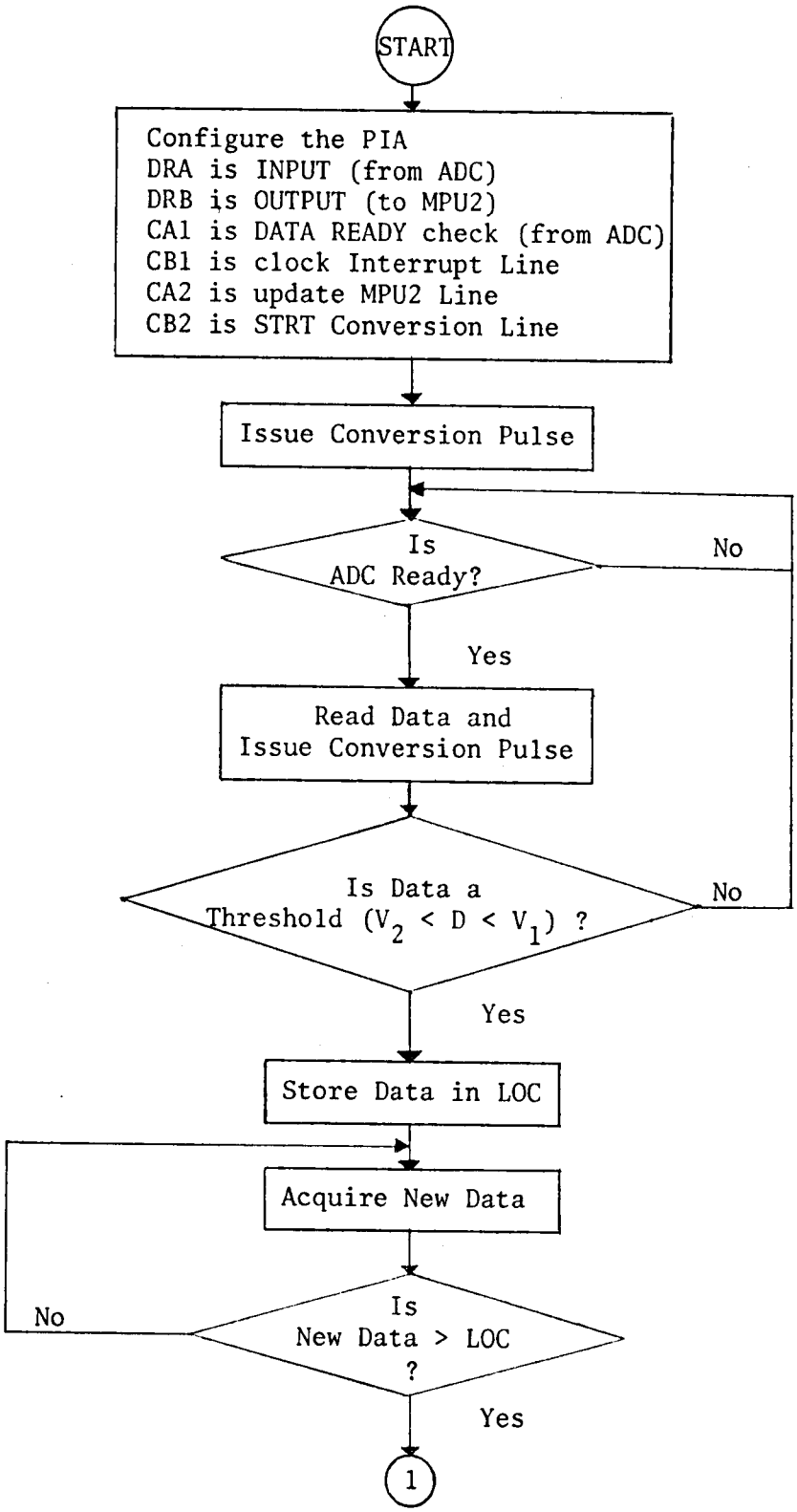


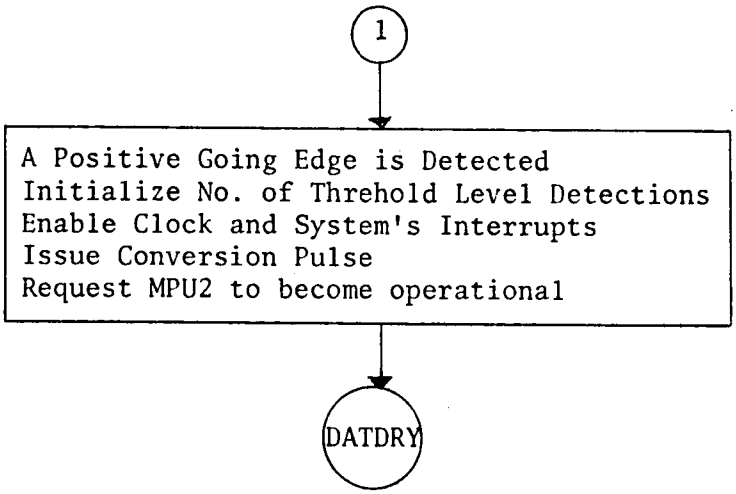

Figure 3.4. Block Diagram: Two-Microprocessors Interval Processor.

Figure 3.5. Interval Processor with ADC Interface Input MPU Flowchart: (a) Initialization routine, (b) Data acquisition and threshold detection routine, (c) Clock ISR routine, and (d) Histogramming and data transfer routine.


START
Configure the PIA
DRA is INPUT (from ADC)
DRB is OUTPUT (to MPU2)
CA1 is DATA READY check (from ADC)
CB1 is clock Interrupt Line
CA2 is update MPU2 Line
CB2 is STRT Conversion Line
Issue Conversion Pulse
Is ADC Ready?
No
Yes
Read Data and Issue Conversion Pulse
Is Data a Threshold ($V_2$ < D < $V_1$) ?
No
Yes
Store Data in LOC
Acquire New Data
Is New Data > LOC ?
No
Yes
1

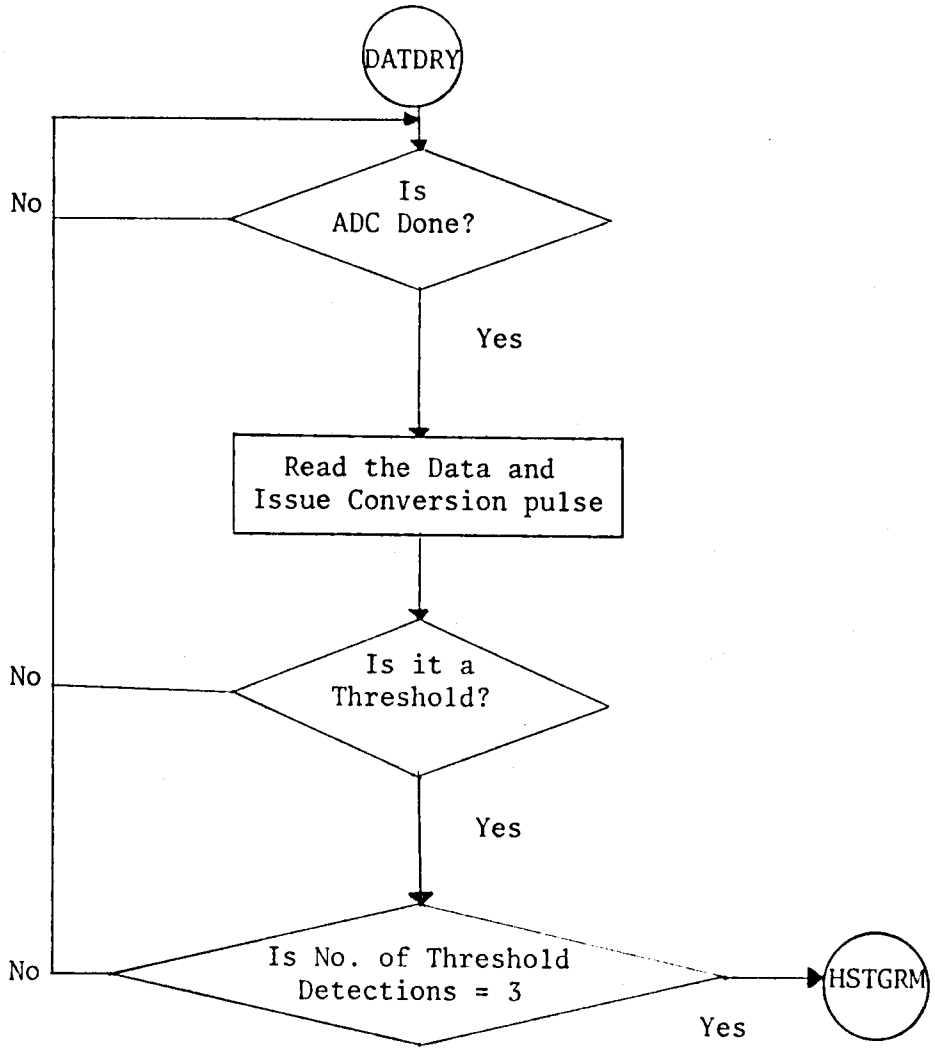
(a)

Figure 3.5



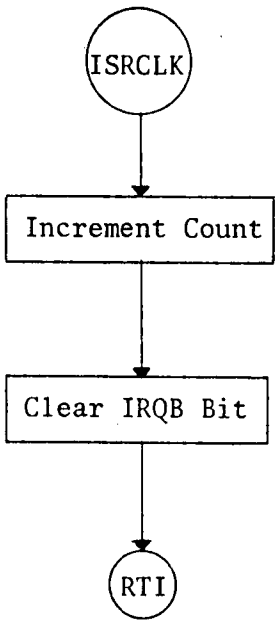

(a) continued

Figure 3.5 continued



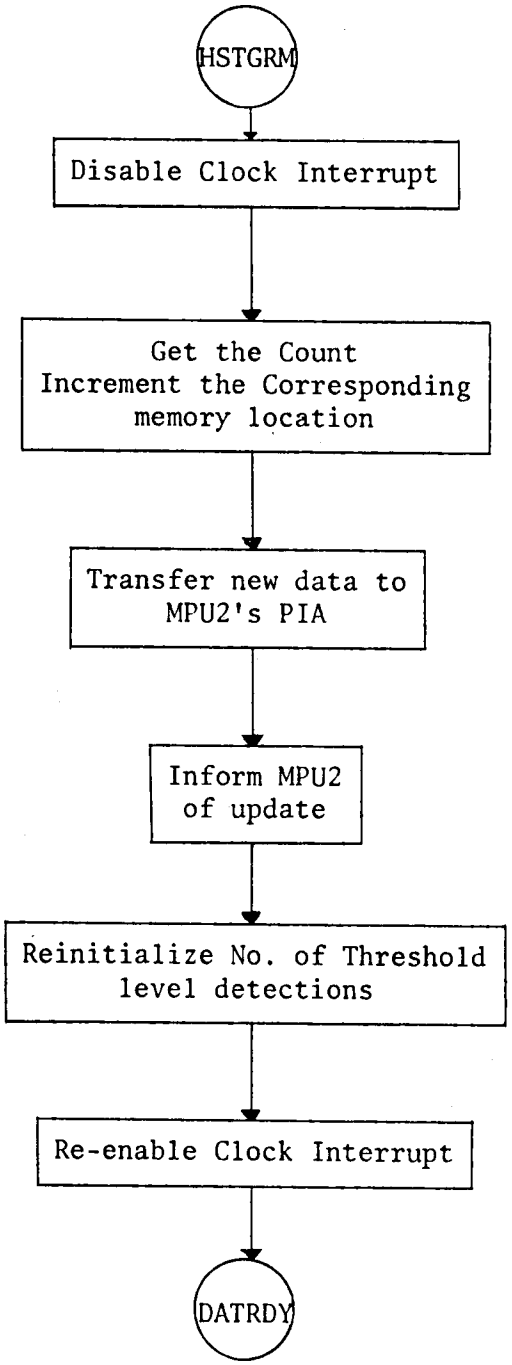

(b)

Figure 3.5 continued




(c)

Figure 3.5 continued




(d)

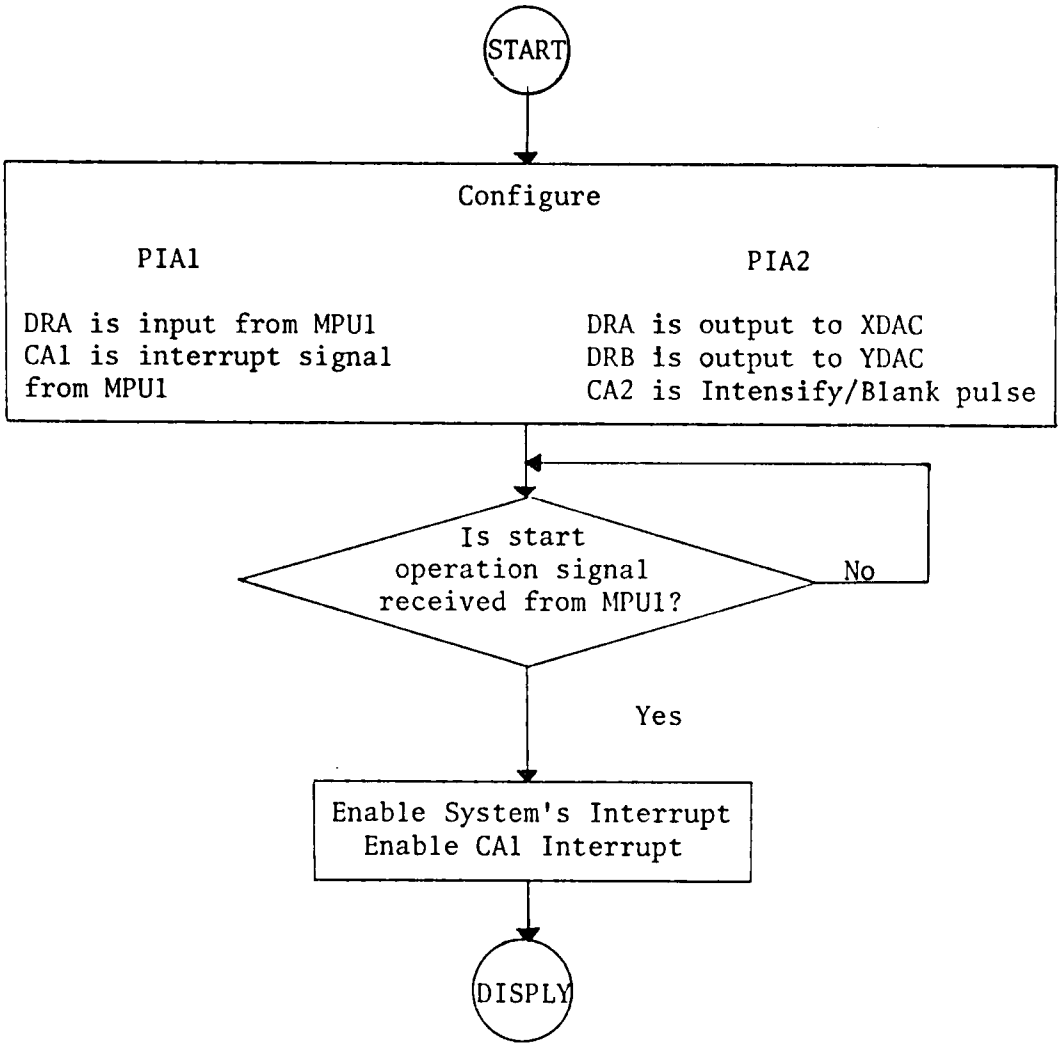
Figure 3.5 continued

memory location on the Display microprocessor's Peripheral Interface Adapter (PIA) and promptly informs that MPU, through a control line of the data update (Fig. 3.5d). Assuming that no new updates can occur by MPU1 before MPU2 gets the previous data, the Input MPU resumes data acquisition and interval measurement without waiting for an acknowledge signal from the Display MPU.

3.1.2.2 Display microprocessor. Operation of the Display MPU, as Figure 3.6a explains, is resumed at the request of MPU1. Exclusively, the Display MPU is constantly displaying RAM addresses from its own memory, M2, on the x-axis of the CRT and their contents on the CRT's y-axis (Fig. 3.4, p. 25). Upon interrupt from the Input MPU, the Display MPU, whose PIA is hooked up to that of the Input MPU, reads in the count (address) from its own PIA, increments the corresponding memory location (Fig. 3.6b) and resumes displaying upon return from interrupt (Fig. 3.6c). Flowcharting for the program listed in Appendix A.2 is detailed in Figure 3.6.

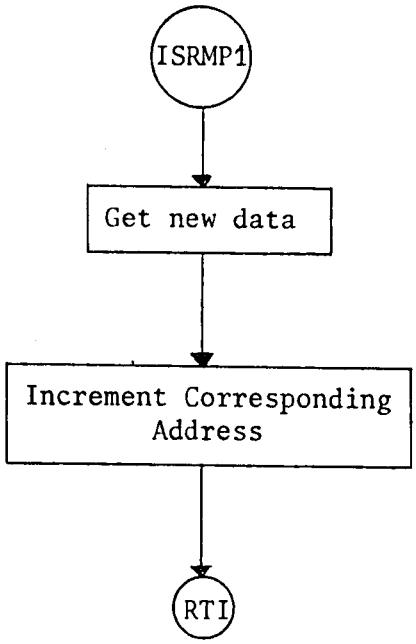
Notice that the Input MPU can transfer data to the Display MPU directly without storing it in memory, M1. Basically, data is being stored in M1 as a test measurement to ensure the validity of data transfer to MPU2.

3.1.2.3 Analysis. As observed in the program of Appendix A.2, the Input microprocessor has been relieved from the display burden and improvements in the sampling rate and signal's frequency content are expected. In particular Equation 3.1 becomes



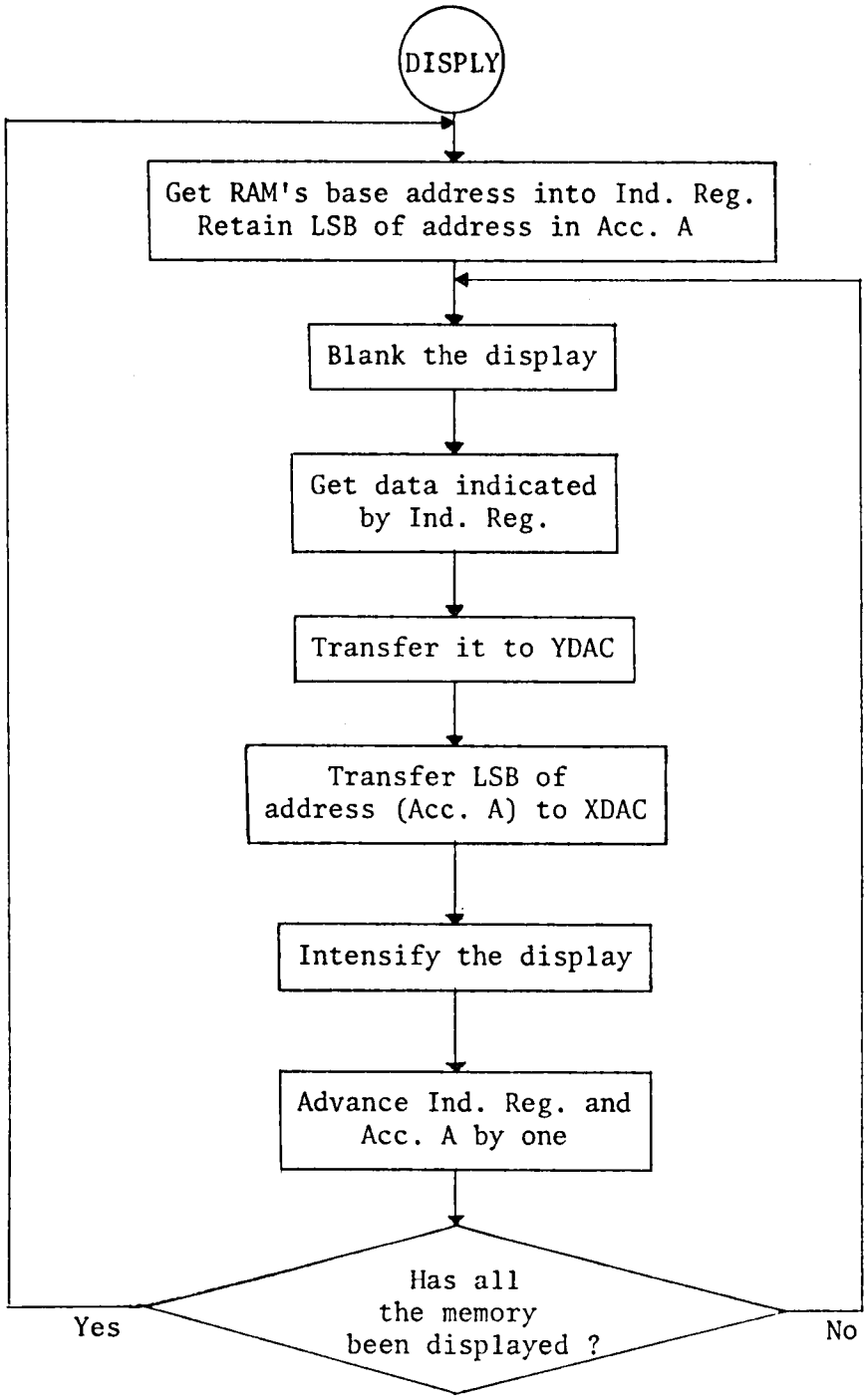

(a)

Figure 3.6. Interval Processor with ADC Interface: Display MPU Flowchart (a) Initialization routine, (b) Data update ISR routine, and (c) Display routine.




(b)

Figure 3.6 continued




(c)

Figure 3.6 continued

$$T_s \geq T_{ACQ} + T_{HST} + T_{ISR} + T_{UPDATE} , \qquad (3.6)$$

where $T_{UPDATE}$ is the time taken by the Input MPU to update the Display MPU with data changes. Based on the analysis made on the program, Appendix A.2, execution times of the above time parameters were evaluated. Calculating $T_{ACQ}$ = 86.3 μsec, $T_{HST}$ = 44 μsec, $T_{ISR}$ = 52.1 μsec, and $T_{UPDATE}$ = 37.5 μsec, a sampling period, $T_s'$, of 0.880 msec is computed from Equation 3.2, page 22. This limits the signal's highest frequency to less than 1.14 kHz. Similarly, the clock frequency is increased to 7.23 z as Equation 3.3, page 23, becomes

$$f_{clk} \leq \frac{1}{T_{ACQ} + T_{ISR}} , \qquad (3.7)$$

where $T_{HST}$ and $T_{UPDATE}$ are excluded. Although a higher clock frequency provides a more accurate count, inhibiting the clock during histogramming and data update could become a source of error. For instance, if $T_{HST}$ and $T_{UPDATE}$ both add up to a one-clock period then one count will be lost. Thus, it is required to add, in software, the estimated number of counts lost during $T_{HST}$ and $T_{UPDATE}$ to the next interval measurement.

Evidently, the restrictions on the analog signal's frequency content and on the data sampling rate have not been considerably improved by this system. On the other hand, the original motivation behind the current design scheme, mainly to maintain a continuous display, is being successfully fulfilled. Task distribution leaves the second MPU predominantly occupied with display and with acknowledging the one-way communication between both microprocessors. Essentially, the CRT display refreshing rate is determined by

$$\text{Display Rate} = \frac{1 - T_{DATUPD} \times f_o}{(\text{No. RAM Locations})(T_{DISP})}, \tag{3.8}$$

where $T_{DATUPD}$ is the time MPU2 takes to acknowledge data transfer from MPU1 and to update itself (Fig. 3.6b). Particularly, given $T_{DISP}$ = 83 μsec, $T_{DATUPD}$ = 70 μsec, and $f_o$ = 1140, Equation 3.8 permits 87 refreshes per second which is a fairly sufficient refreshing rate to maintain a nonflickering display. Strictly speaking, since the highest frequency in the analog signal is less than 1.14 kHz not more than 1140 updates to MPU2 can occur in one second. Only 0.079 seconds ($1140 \times T_{DATUPD}$) is consumed by MPU2 in updating itself during one second. Therefore, the display routing has control of 92.1% of the time (0.921 seconds), and Equation 3.8 modifies to

$$\text{Display Rate} \simeq \frac{1}{(\text{No. RAM Locations})(T_{DISP})}, \tag{3.9}$$

Consequently, a refreshing rate of 94 times/second is obtained for displaying 128 locations, and, obviously three RAM's of the same size can be refreshed at a rate of 31 times/second which is slightly above the minimum requirement.

Even with the display task being successfully accomplished by the Display microprocessor, the Input processor remains incapable of processing fast analog signals and, thus, further modification has to be considered.

### 3.1.3 Two-Microprocessor System with Schmidt Trigger Interface

In this section, attention is directed to the Input microprocessor and its interface in an effort to remove the strict limitations imposed by the previous design schemes on the input analog signal.

The fact that most of the data samples are insignificant to the MPU and only the threshold level crossing points are of importance (Fig. 3.5b, p. 29), urges that one uses a device which produces a digital format of the signal at the desired data points. Seemingly, a Schmidt Trigger, which is a threshold detector device, satisfies this task. The Schmidt Trigger device described in Chapter 2 is considered to produce a voltage, $V_o$, as long as the signal is above the threshold level. A zero voltage is output when the signal goes below the threshold level (Fig. 3.7).

3.1.3.1 Input microprocessor. As depicted in the system's block diagram, Figure 3.8, the output of the Schmidt Trigger is fed to CA1 interrupt line of the Input MPU's PIA. Under program control, CA1 interrupts the MPU on the occurrence of the positive going edges indicating the end of one interval. Similarly, control line CB1 is connected to the external clock output which functions as the clock for the interval measurement.

Initially, as illustrated in the program flowchart (Fig. 3.9a), the input microprocessor waits for the occurrence of the first rising transition on the interrupt line CA1 after which occurrence the MPU performs the following:

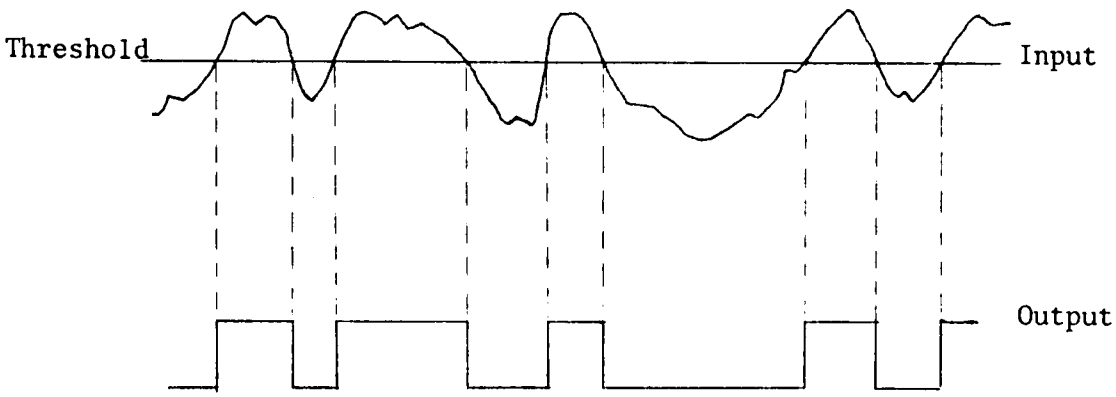



Figure 3.7. Schmidt Trigger's Input and Output.

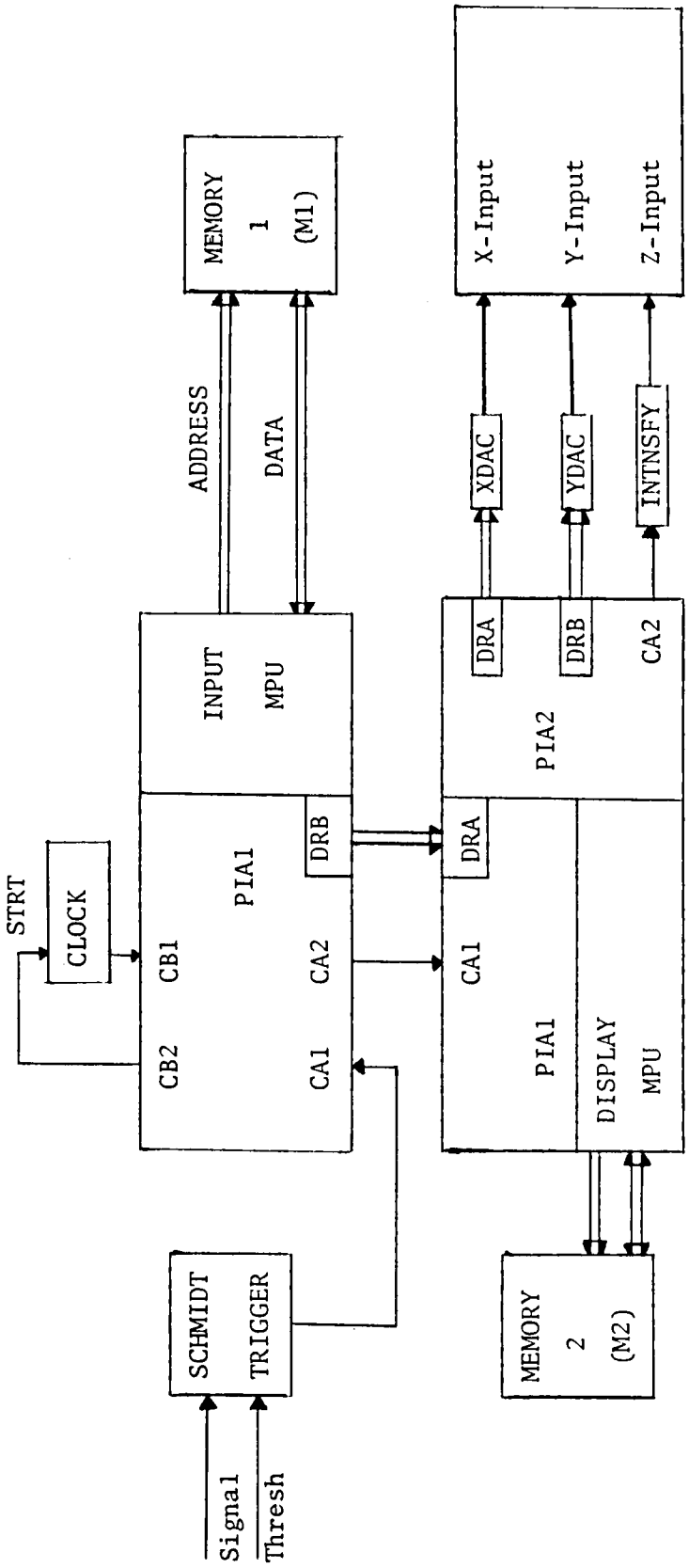
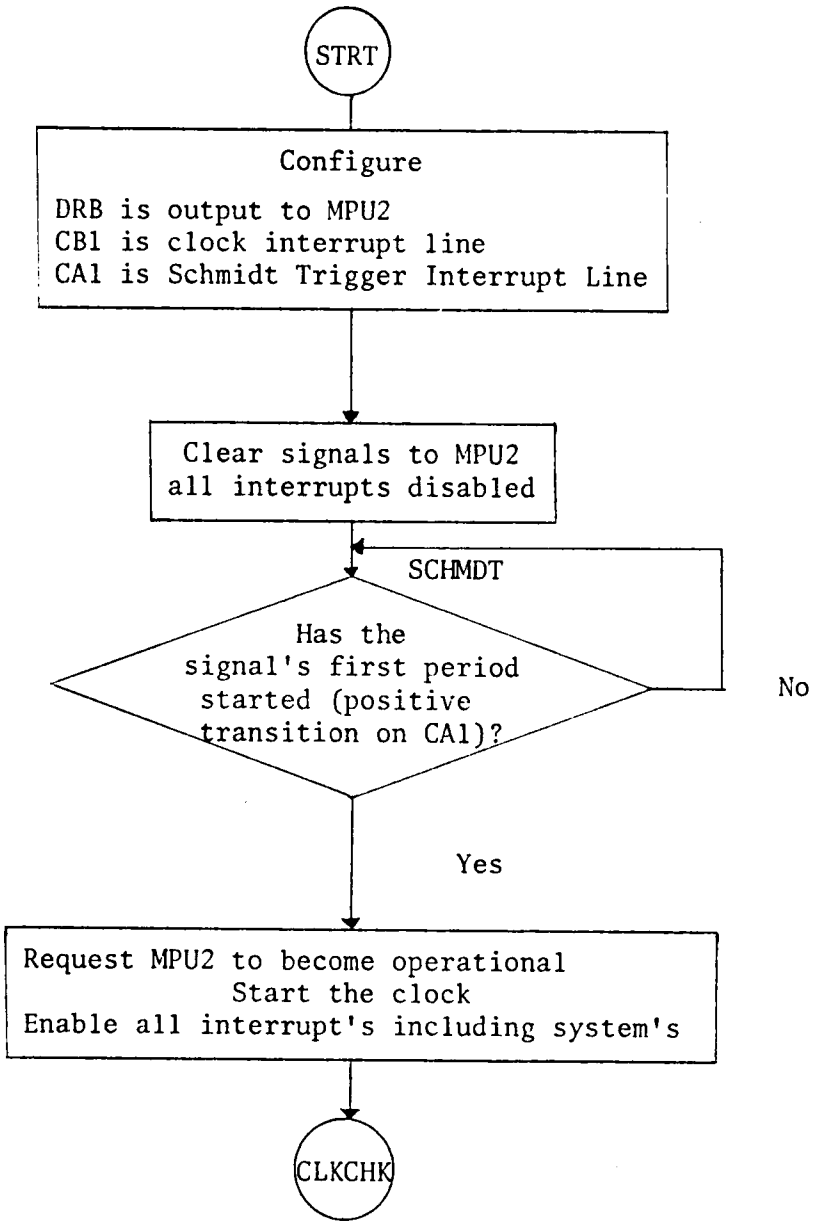
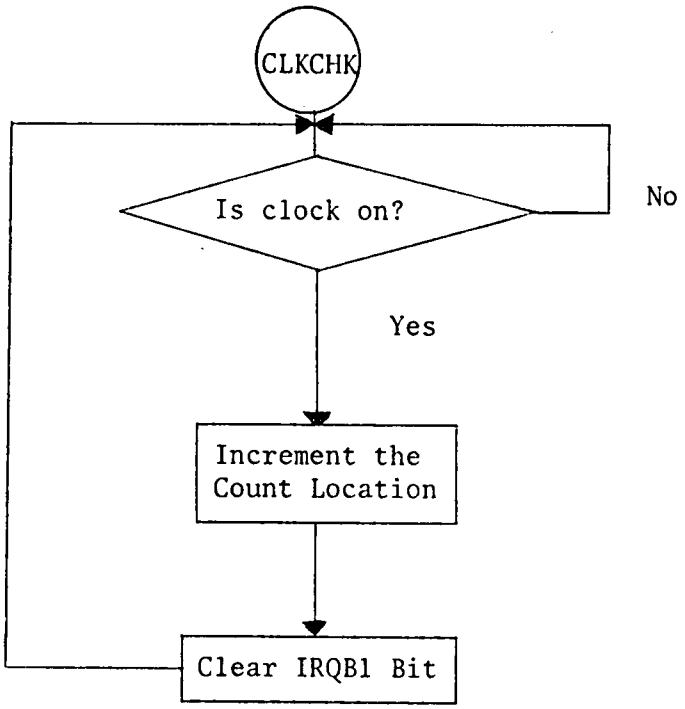



Figure 3.8. Block Diagram: Interval Processor with Schmidt Trigger Interface.



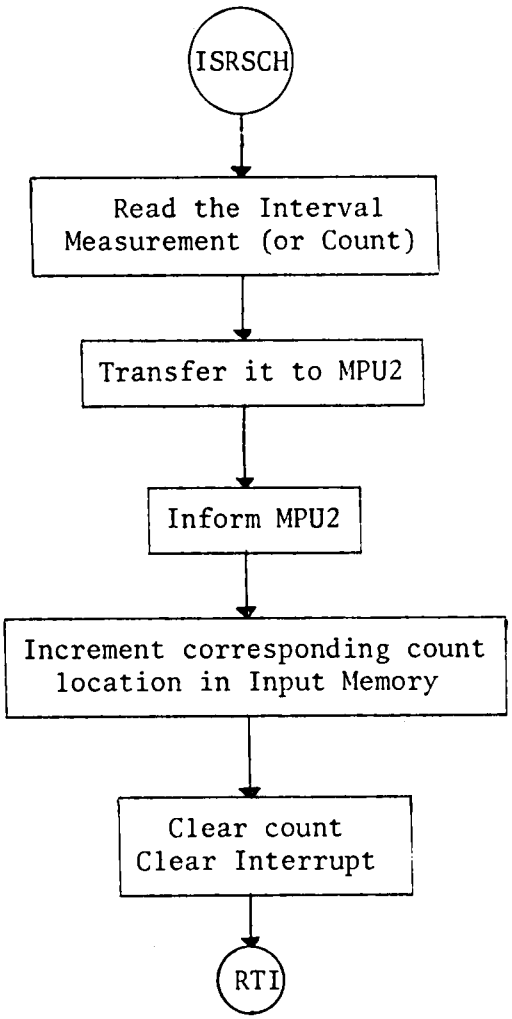

(a)

Figure 3.9. Interval Processor with Schmidt Trigger Interface: Input MPU Flowchart (a) Initialization program, (b) Interval measurement routine, and (c) Schmidt Trigger ISR routine.




(b)

Figure 3.9 continued




(c)

Figure 3.9 continued

1. Enables system's and Schmidt output's interrupts.
2. Starts the clock through control line CB2.
3. Requests the Display MPU to become operational by sending a pulse on CA2.

Beyond this stage the input MPU is constantly examining IRQB1 bit in control register B, the status of which when set, causes the MPU to increment a count memory location and to reset the bit thereafter (Fig. 3.9b). Should the $\overline{IRQ}$ line go high indicating an interrupt from the Schmidt Trigger, the MPU reads the interval measurement (count), increments the corresponding memory location, transfers to the Display MPU the measurement (count), issues a Data Transfer signal, and returns from interrupt (Fig. 3.9c).

3.1.3.2 Display MPU. No modifications are necessary in the Display MPU which simply operates in the same previous fashion. The occurrence of the first positive transition on the interrupt line CA1, which is connected to the output of CA2 of the Input MPU, causes the Display MPU to enable its system's interrupt and CA1 interrupt (Fig. 3.6a, p. 33). Thereafter, the Display MPU resumes displaying RAM addresses of memory M2 and their contents on the CRT's x-axis and y-axis DACs respectively (Fig. 3.6c, p. 35). Should an interrupt occur on CA1 indicating data transfer from the Input MPU, the Display MPU suspends the display and updates itself with the new address (count), already residing on its own PIA. The content of that address is incremented by one and the Display MPU returns from interrupt (Fig. 3.6b, p. 34). The assumption is that data update is performed

by MPU2 before new data is transferred by the Input MPU. When such an assumption is not valid, the Input MPU should wait for a Done signal from the Display MPU before it can send new data.

3.1.3.3 Analysis. With the implementation of the Schmidt Trigger, the software size and, thus, the execution time has been reduced considerably (Appendix A.3). The effect of this reduction is obvious in softening the restrictions on the highest frequency content in the input signal. For the existing interface (Fig. 3.8, p. 40) and its software (Appendix A.3), the program execution time requires that the minimum period, $T_o$, be higher than the sum of $T_{ISR}$ and $T_{CLKCHK}$. Therefore, the highest frequency, $f_o$, in the analog signal is given by

$$f_o \leq \frac{1}{T_{ISR} + T_{CLKCHK}} , \tag{3.10}$$

where $T_{ISR}$ is the interrupt Service Routine execution time and $T_{CLKCHK}$ is the time consumed by the CPU in the interval measurement counting process. Calculating $T_{ISR}$ = 105 μsec, and $T_{CLKCHK}$ = 35.8 μsec (Appendix A.3), yields a $f_o$ as high as 7.05 kHz.

Similarly, the counting clock frequency is increased substantially by the fact that the clock is serviced only whenever its corresponding PIA's control register's bit is set. Thus, $f_{CLK}$ is absolutely determined by $T_{CLKCHK}$ and is simply given by

$$f_{CLK} \leq \frac{1}{T_{CLKCHK}} , \tag{3.11}$$

which implies that the clock frequency be less than 27.9 kHz. Ideally, at the occurrence of $T_o$, the clock yields a count of only one where two or more counts are lost during the servicing of the Schmidt Trigger's interrupt preceding $T_o$. Hence, as was previously done, the lost number of counts in any interval measurement is calculated as

$$\text{Number of Counts} = (T_{ISR})(f_{CLK}) \ , \tag{3.12}$$

and added, in software, to every interval count reading. There are exactly 2.96 counts (rounded to 3) lost in this scheme.

As concluded from the above computations, the replacement of the ADC by the threshold level detector has appreciably improved the bounds on both the highest frequency content of the signal and the clock frequency. Although the higher $f_{CLK}$ improves the accuracy of the count, it places a worse bound on the lowest frequency in the signal which is given by

$$f_L \geq f_{CLK}/\text{size of RAM} \ . \tag{3.13}$$

For instance, with an 8-bit RAM (size = 256), and with $f_{CLK}$ given as 27.9 Khz, the lowest frequency in the signal should be greater than 109 Hz.

On the other hand, the operation of the Display MPU is not altered by this scheme. However, the display refreshing rate, in accordance with Equation 3.8, page 37, has decreased to 39 times/sec per 128 locations. At the minimum refreshing rate of 30 times/sec only 168 RAM addresses can be displayed with no flicker.

### 3.1.4 Modified Two-Microprocessor System

The motivation for this section is to reconfigure the previous design scheme such that a wide range of clock frequencies is feasible in the counting process. Additional improvements of the bounds on the input signal are intended.

Although better bounds on the analog signal and a higher clock frequency were obtained, one important consideration remains: How accurate can an interval measurement be?

Signals containing intervals so close to each other in length will not be histogrammed accurately. Since the maximum clock frequency attained so far is 27.9 kHz, the counts obtained for any two intervals differing by one clock period in length are equal or, at the most, differ by two. For instance, given $T_1$ and $T_2$ with lengths 170 μsec and 200 μsec respectively, according to Equation 3.12, $T_1$ counts to 4 or 5 and $T_2$ counts to 5 or 6. If a 1-MHz clock is used, the corresponding counts for the above two intervals are 170 and 200 respectively and, thus, a much better resolution is obtained.

Therefore, a hardware approach is sought to replace the software technique in performing the interval counting. Binary counters are easy to utilize in accomplishing the counting process external to the microprocessor such that little interaction is required with the MPU.

Indeed, the new system as configured in the block diagram of Figure 3.10 is interfaced to an 8-bit counter, where all 8 output bits are fed into port A of the PIA. The binary counter is reset by a high pulse on control line CB2 and resumes counting upon clearing control line CA2 which enables the free running clock.

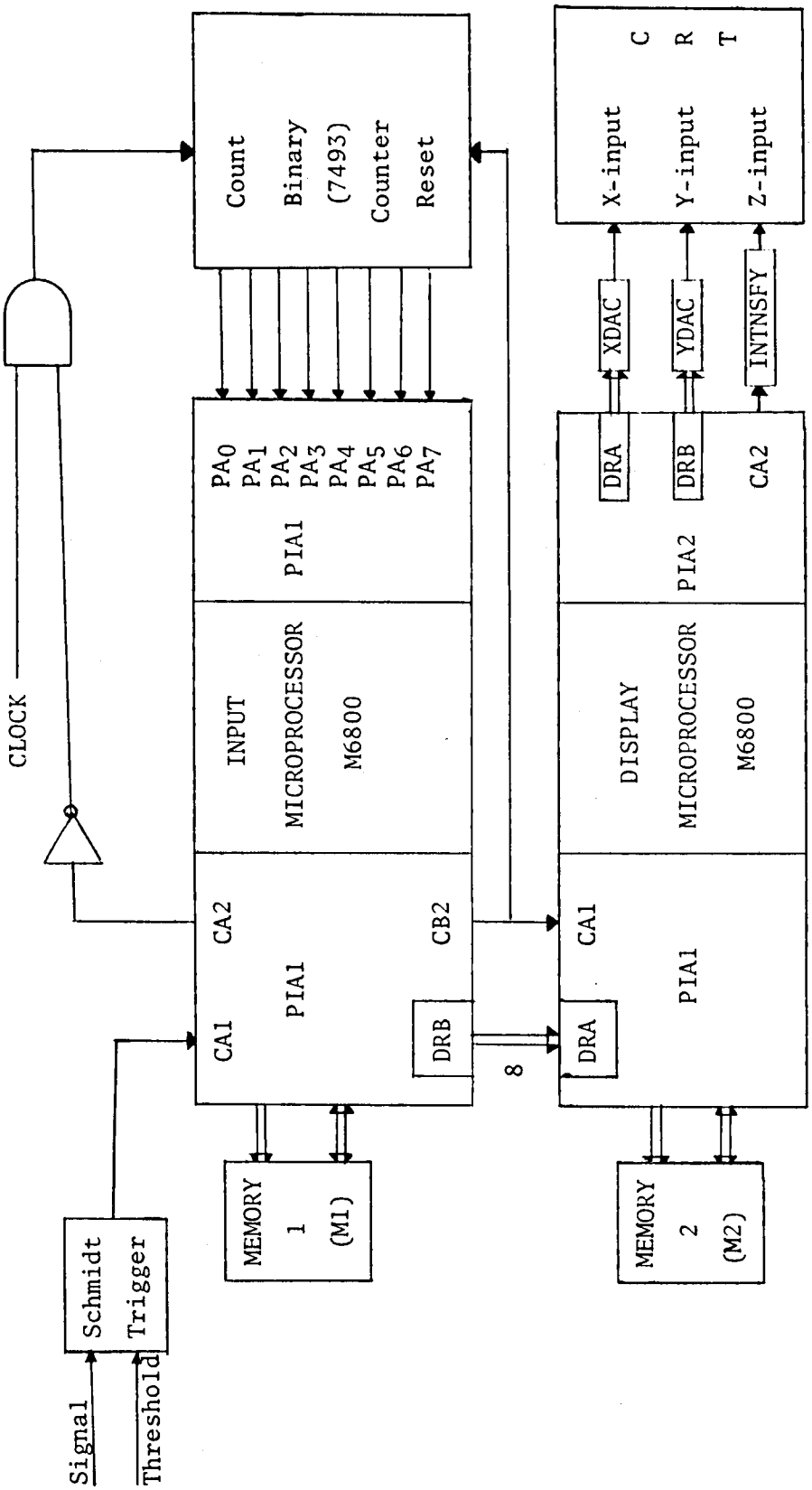


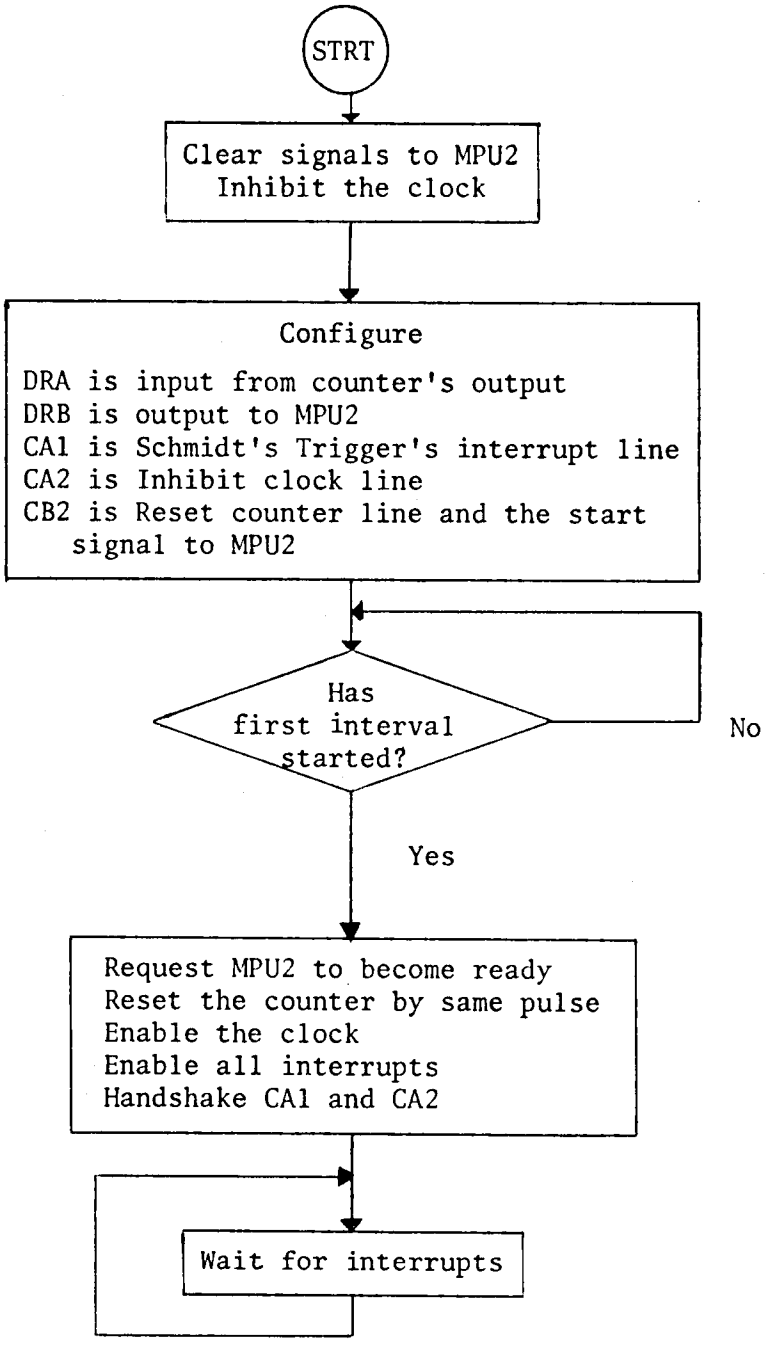

Figure 3.10. Modified (final) Interval Processor: Block Diagram and Interface.

Initially, as seen by the flowchart (Fig. 3.11a), the clock is enabled at the occurrence of the first positive transition on the CA1 interrupt line, which is already connected to the Schmidt Trigger's output. Figure 3.11a indicates that the Input microprocessor, after the beginning of the first interval, performs the following functions:

1. requests the Display MPU to resume operation by transmitting a high pulse on control line CB2,
2. resets the counter to zero by the same pulse on CB2,
3. enables its system's interrupt and CA1 interrupt, and
4. enables the free running clock by clearing CA2.

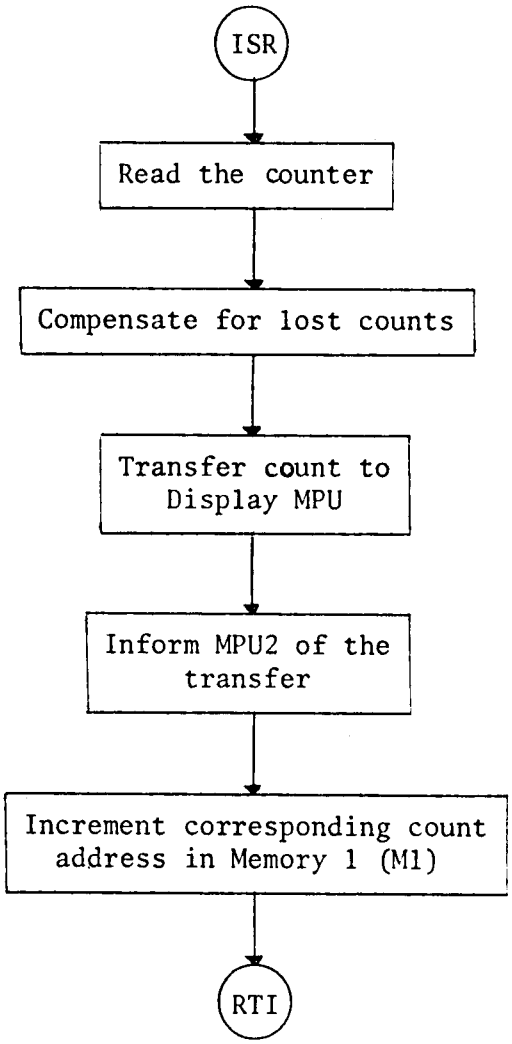
Meanwhile, CA1 and CA2 are programmed to function in the handshake mode. The next positive transition on CA1, indicating the end of the interval, forces CA2 to go high and remain so until the corresponding PIA Data Register is read (Fig. 3.11b). This, as illustrated by the timing diagrams of Figure 3.12, disables the clock during the "Data Read" operation. Immediately, the count is placed on port B, which is connected to port A of the Display MPU's PIA, and a high pulse is sent on CB2 (Fig. 3.11b). CB2, being attached to CA1 of the Display MPU, causes an interrupt to the Display MPU which, in turn, reads the new data from the PIA and updates itself (Fig. 3.6b, p. 34).

On the other hand, the Display microprocessor is interfaced and operated in the same manner as previously described. Its system




(a)

Figure 3.11. Modified Interval Processor: Input MPU Flowchart (a) Initialization and startup routine and (b) Interval Read and data transfer ISR routine.




(b)

Figure 3.11 continued

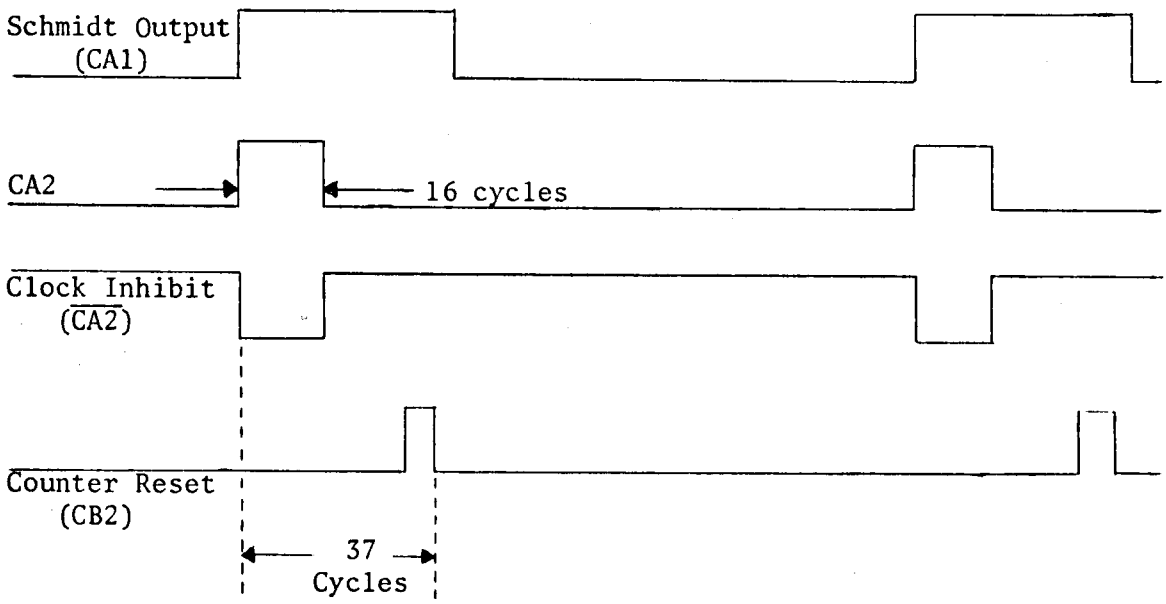



Figure 3.12. Modified Interval Processor Input MPU Interface Timing Diagrams.

interrupt and CA1 interrupt are enabled and display is resumed as soon as the first high pulse on CA1 arrives. Future high pulses on CA1, indicating data transfer from MPU1, interrupt the Display MPU. The MPU reads new data from the PIA, increments the corresponding memory location, and returns from interrupt (Fig. 3.6b, p. 34). Again, there is no need to send back an acknowledge signal to MPU1. The assumption is that new data transfers are taking place at a guaranteed slow rate, as is proven later in the analysis.

3.1.4.1 Analysis. One major system parameter, the clock frequency, has attained remarkable improvement with the implementation of this hardware technique. Introducing the binary counter has made feasible the use of high frequencies in the counting process, thus, obtaining a better histogram accuracy. Basically, any frequency available could be used so long as the longest interval, $T_h$, in the analog signal does not exceed a certain limit. Actually, the limit on the clock frequency is governed by the RAM size and is given by

$$f_{CLK} \leq \frac{\text{RAM size}}{T_h} , \tag{3.14}$$

where $T_h$ is the longest period in the signal. Specifically, with a 256 word × 8-bit RAM and a $T_h$ equal to 255 μsec, the clock frequency should be less than 1 MHz. Incidently, since the maximum count produced by an 8-bit counter is 255, an overflow will occur if $T_h$ exceeds 255 μsec, assuming that $f_{CLK}$ is 1 MHz.

Expanding the size of the memory and increasing the order of the binary counter relieves the above restrictions on $f_{CLK}$ and $T_h$. As derived previously in Section 3.1.3, page 38, the number of counts in an interval measurement is given by

$$\text{count} = T_{INTVL} \times f_{CLK} \,, \tag{3.15}$$

where $T_{INTVL}$ is the length of the interval. In particular, for a given $T_h$, the maximum count is

$$\text{count max} = T_h \times f_{CLKMAX} \,, \tag{3.16}$$

where $f_{CLKMAX}$ is the highest clock frequency available. The above equation yields the highest corresponding location in RAM and, thus, the size of the RAM and the order of the counter are determined. As an example, Table 3.1 shows that, for $T_h$ = 4096 μsec and $f_{CLKMAX}$ = 2 MHz, an 8K word × 8-bit RAM and a 13-bit counter are required. Assuming that the memory size is fixed at 64 K, and, thus, requiring a 16-bit counter, Table 3.2 presents $f_{CLK}$ as a function of $T_h$.

Evidently, the count is more accurate as $T_h$ decreases since a higher clock frequency is used. In general, long intervals do not exist in fast signals and high clock frequencies are applied confidently as long as they match the requirement of the binary counter.

A slight improvement in the highest frequency content is achieved due to the fact that the counting is done externally; the amount of software and, thus, the CPU execution time, has been reduced (Appendix A.4).

Table 3.1 Memory Size and Counter's Order Requirements as a Function of $f_{CLK}$

| $f_{CLKMAX}$ (Hz) | RAM Size (Locations of width 8-bits) | Counter order |
|---|---|---|
| 10k | 41 | 6-bit |
| 500k | 2048 = 2k | 11-bit |
| 1M | 4096 = 4k | 12-bit |
| 2M | 8192 = 8k | 13-bit |
| 16M | 32 = 64k | 16-bit |

($T_h$ is fixed at 4096 μsec.)

Table 3.2 Counter's Clock Frequency as a Function of $T_h$

| $T_h$ (sec) | $f_{CLK(Hz)}$ |
|---|---|
| 256 μ | 4M |
| 1024 μ | 1M |
| 4096 μ | 250k |
| 40.96M | 25k |
| 204.8M | 5k |

(Ram size is fixed at 1k byte)

Essentially, the shortest period in the signal is exclusively determined by $T_{ISR}$ and $T_{LOOP}$ and is given by

$$T_o = T_{ISR} + T_{LOOP} \,, \tag{3.17}$$

where $T_{LOOP}$ is the "wait for interrupt" loop time. Equation 3.17, as calculations in Appendix A.4 show, requires that $T_o$ be higher than 110 μsec. This implies that the highest frequency content in the analog signal should not exceed 9 Khz.

Consequently, the display refreshing rate as given by Equation 3.8, page 37, is 35 times/sec per 128 locations. A higher $f_o$ places a limitation on how fast the input signal should be so that a nonflickering display is maintained. In particular, at the minimum refreshing rate of 30 times/sec, given that 128 addresses are displayed, the maximum $f_o$ as given by Equation 3.8, page 37, should not exceed 9.7 kHz.

## 3.2 Dwell Histogram

As previously defined in Chapter 2, a dwell time is the length of time the analog signal stays above a preset threshold level. These dwell times are quantized as memory addresses. The contents of the address location is incremented by one for each dwell time detected.

As was proposed in Chapter 3, the interval processor is slightly altered to construct a dwell histogram. Since the dwell time is acquired by detecting both the signal's positive and negative going edges, the dwell histogram may be accomplished by reconfiguring

the interrupt scheme of the interval processor and by reorganizing the counter's reset-count sequence.

A dwell processor is designed, as illustrated in the block diagram (Fig. 3.13), with few modifications to the previous system. The strategy is to allow the Schmidt trigger's output on CA1 to interrupt the input microprocessor on the positive transition. By means of the handshake mode, CA2 resets the counter which resumes counting. Meanwhile, the discriminator's output is inverted and fed into CB1. The occurrence of a positive transition on CB1 interrupts the MPU indicating the end of one dwell time. Upon interrupt from CB1, CB2 inhibits the clock and uses the handshake mode to freeze the counter during the "Read" operation. The timing diagrams, Figure 3.14, provide a complete explanation of these sequences. In regard to the counting pulse generation, it is evident that the upper Nand gate, Figure 3.13, is responsible for the actual dwell measurement, whereas the lower Nand gate is inserted to provide a series of clock pulses so that CA2 can reset the counter at the arrival of the next clock pulse.

### 3.2.1 Input MPU Operation

As presented in the flowchart, Figure 3.15a, the MC6800 program, listed in Appendix A.5, waits for the initial positive threshold crossing. At this point the program performs the following steps:

1. requests MPU2 to get ready by sending a high pulse on CB2 of PIA2,
2. resets the counter by issuing a high pulse on CA2 of PIA1,

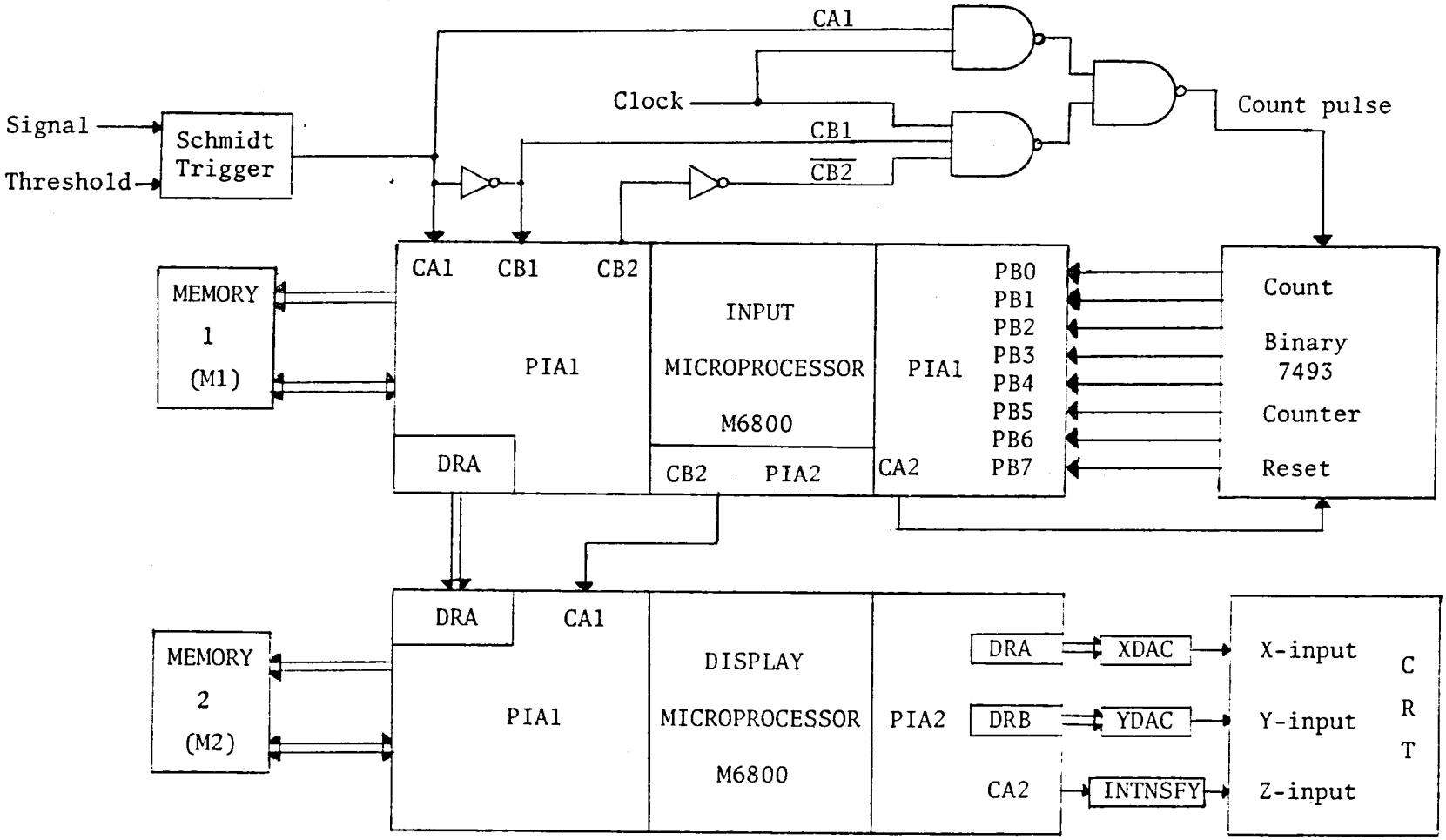



Figure 3.13. Dwell Processor Block Diagram and Interface.

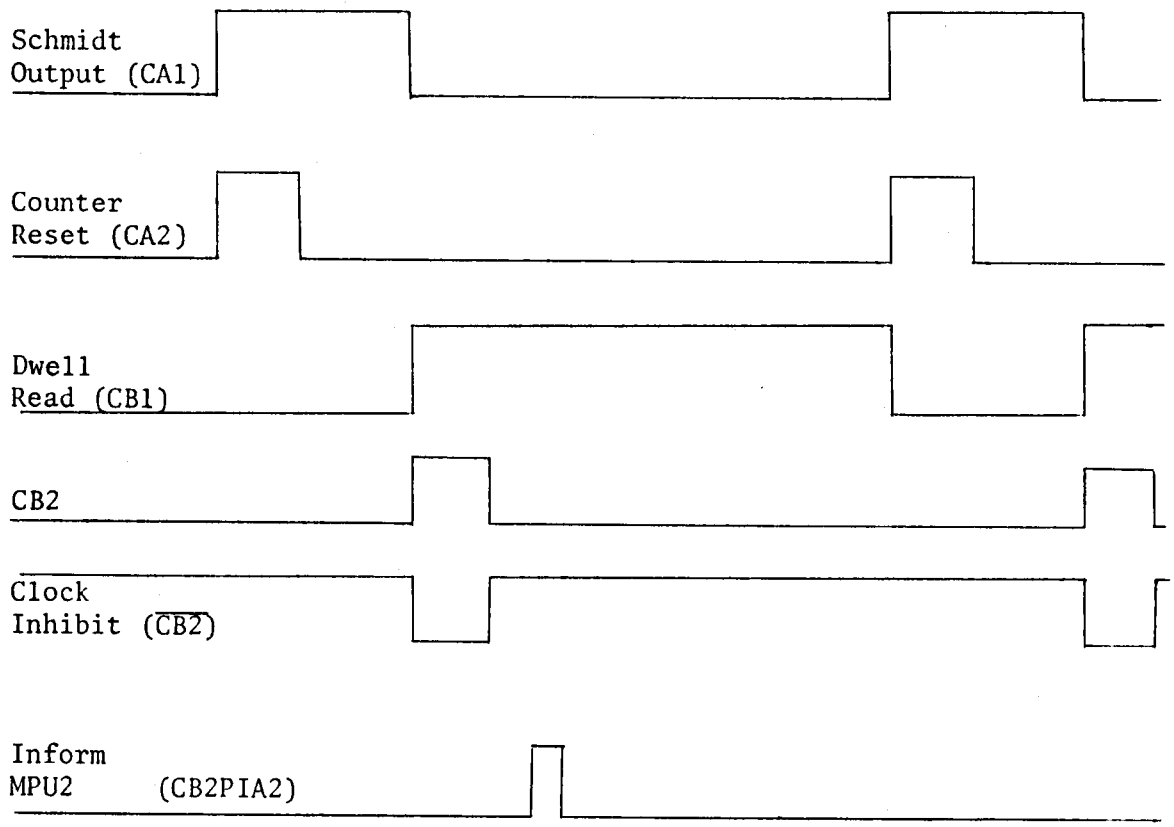
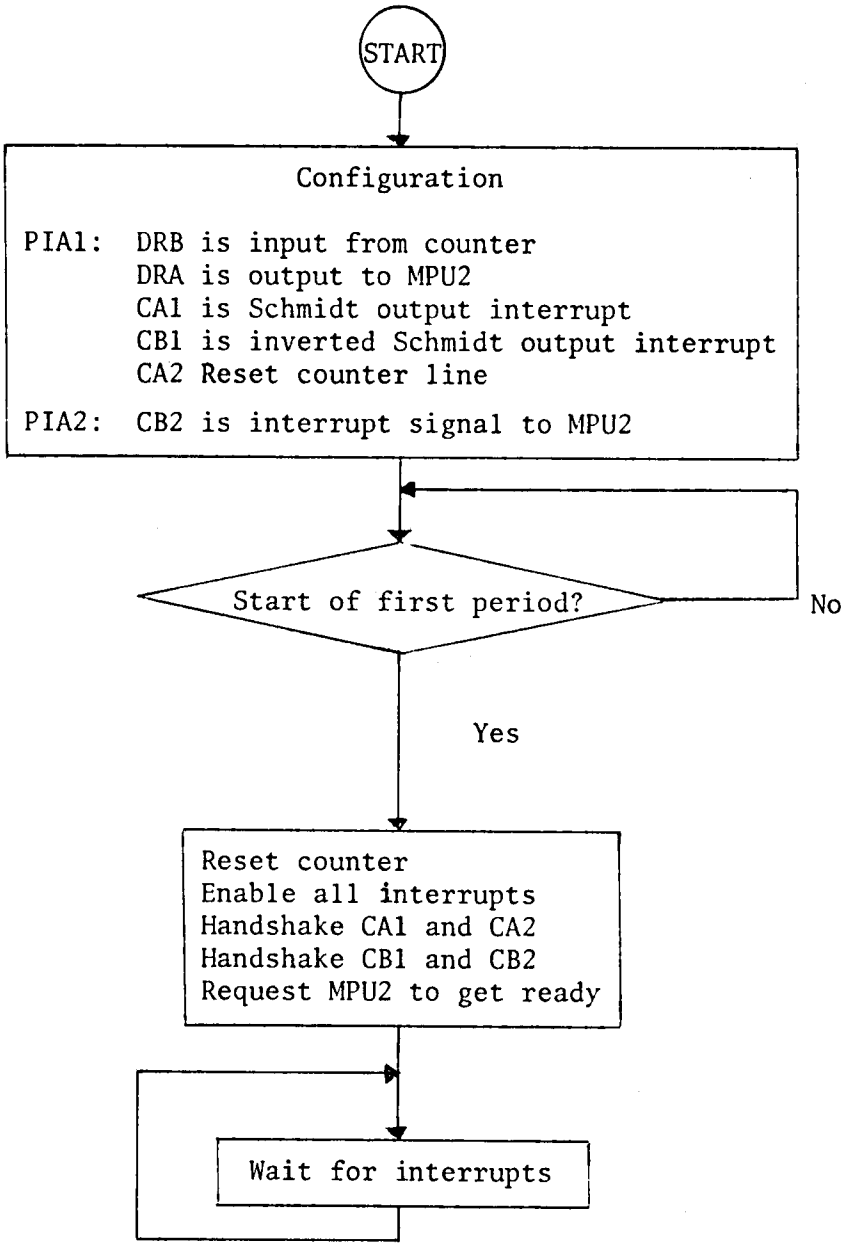
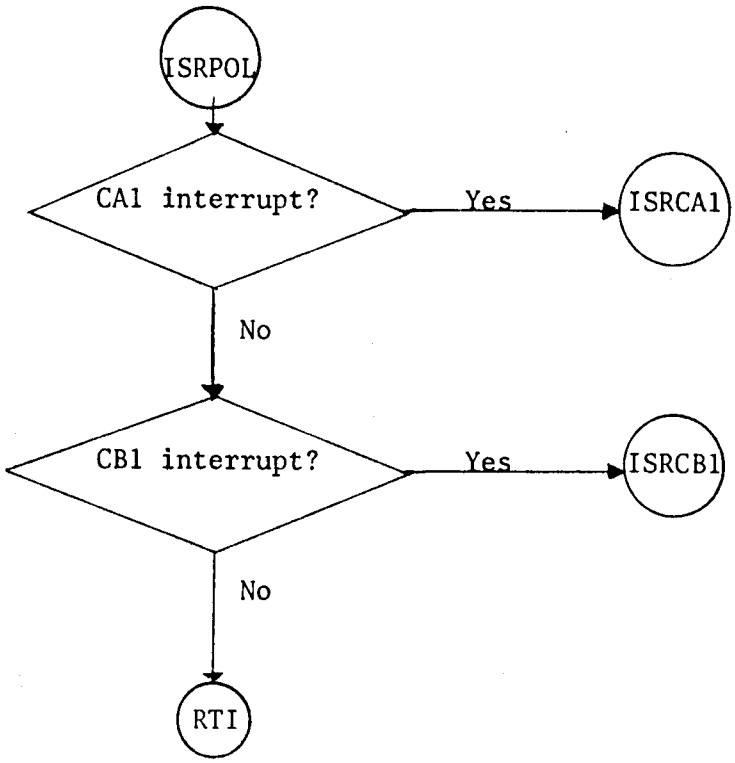



Figure 3.14. Dwell Processor Timing Diagrams.

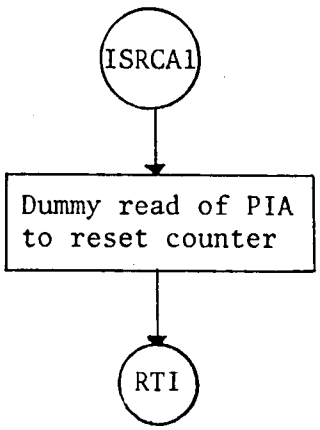



(a)

Figure 3.15. Dwell Processor Flowchart (a) Initialization routine, (b) Polling routine, (c) Schmidt Trigger's output ISR routine, and (d) Inverted Schmidt Trigger's output ISR routine.

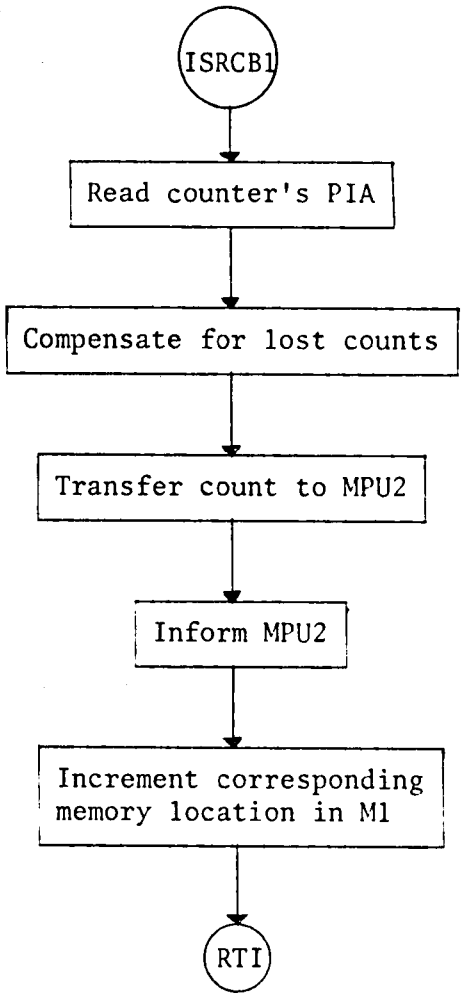



(b)




(c)

Figure 3.15 continued




(d)

Figure 3.15 continued

3. handshakes CA1 with CA2 and CB1 with CB2 of PIA1 and enables their interrupts, and
4. enables system's interrupt.

Following this initialization process, the Input MPU waits for interrupts and, whenever the $\overline{IRQ}$ line is pulled high, a polling routine is executed to determine the interrupting device (Fig. 3.15b). Those interrupts recognized as coming from CA1 (Fig. 3.15c) cause the MPU to read the corresponding data register, DRA1. This forces CA2 to go high which resets the counter. Dwell counting process, thereafter, resumes. A CB1 interrupt as illustrated in Figure 3.15d, causes the MPU to

1. read the counter's data register, DRB1,
2. transfer the count to the Display MPU by placing it on DRA1,
3. inform MPU2 of the new data transfer by sending a high pulse on CB2 of PIA2,
4. increment the corresponding memory address, and
5. return from interrupt.

### 3.2.2 Display MPU Operation

Configuration, interface, and operation of the Display MPU are exactly the same as those for the nterval Processor, (Figs. 3.4 and 3.6, pp. 25 and 33). Memory addresses and their contents are displayed on the CRT's x-axis and y-axis DACs, respectively. Furthermore, transfer signals from MPU1 interrupt the Display MPU to pass the data.

### 3.2.3 Analysis

Major parameters of the Dwell processor are the limitations on the signal's frequency content, the clock speed and its effect on count accuracy, and the refreshing rate of the display. The analysis made on the program listed in Appendix A.5, indicates that the signal's shortest period is determined by

$$T_o \geq T_{ISRCA1} + T_{ISRCB1} \text{ ,} \tag{3.18}$$

where $T_{ISRCA1}$ and $T_{ISRCB1}$ are the times taken by MPU to poll and service CA1 and CB1 interrupts, respectively. Computing $T_{ISRCA1}$ = 55.3 μsec and $T_{ISRCB1}$ = 128.5 μsec, Equation 3.18 yields a shortest interval, $T_o$, of 183.8 μsec. This indicates that the signal's highest frequency, $f_o$, should not exceed 5.44 kHz.

Strictly speaking, a new signal parameter should be considered. Notice that once CA1 interrupts no new interrupts can occur on CB1 until a period of $T_{ISRCA1}$ = 55.3 μsec has elapsed (Appendix A.5). Thus, there is a limitation on the minimum dwell time, defined as $T_{dmin}$ that can be measured, and is simply given by

$$T_{dmin} \geq T_{ISRCA1} \text{ ,} \tag{3.19}$$

which demands that the minimum dwell time be greater than 55.3 sec.

In regard to the longest period, $T_h$, present in a signal, Equation 3.14, page 53, is applied. For a given memory size Table 3.2, page 55, determines $T_h$ for a fixed clock frequency.

Similarly, the clock frequency is controlled by Equation 3.14, (p. 53) also. Any clock frequency can be used so long as there is sufficient memory to fit the measurement of a given $T_h$. In other words, if $f_{CLK}$ is fixed at 1 Mhz and $T_h$ is increased to 1024 μsec, then a 1K RAM is needed.

On the other hand, the display refreshing rate given by Equation 3.8 (p. 37), for a 128 × 8-bit RAM is 58 times/sec. At the minimum refreshing rate, 30 times/sec, only 245 locations can be displayed effectively. As previously concluded, at the minimum rate, assuming a 128 × 8-bit RAM, the signal's highest frequency content, $f_o$, cannot exceed 9.7 kHz.

# CHAPTER 4

# AMPLITUDE HISTOGRAM

Generation of the amplitude histogram is the second category for which a system was to be designed. As mentioned in Chapter 2, this histogram is constructed by quantizing successive measured voltages of signal's samples. Each quantized sample represents an address location. The content of that address location is incremented by one every time the corresponding sample is detected.

The design scheme followed in this chapter will not deviate from the final pattern of the interval processor. An ADC will be used to form the main interface between the signal and the MPU. Therefore, a fast sampling rate is required for accurate acquisition of amplitude histograms.

## 4.1 Input Microprocessor

The Input MPU (M6800) is controlling the ADC as detailed in the block diagram of Figure 4.1. Conversion pulses to the ADC are applied through control line CA2 which functions in the pulse-mode with CA1. Meanwhile, ready data from the ADC is acknowledged through interrupt line CA1.

As illustrated by the flowchart of Figure 4.2, the Input MPU transmits on control line CB2 a Ready signal to the Display MPU just prior to starting the ADC. Immediately following the first conversion pulse to the ADC, control line CA2 is programmed to function in the

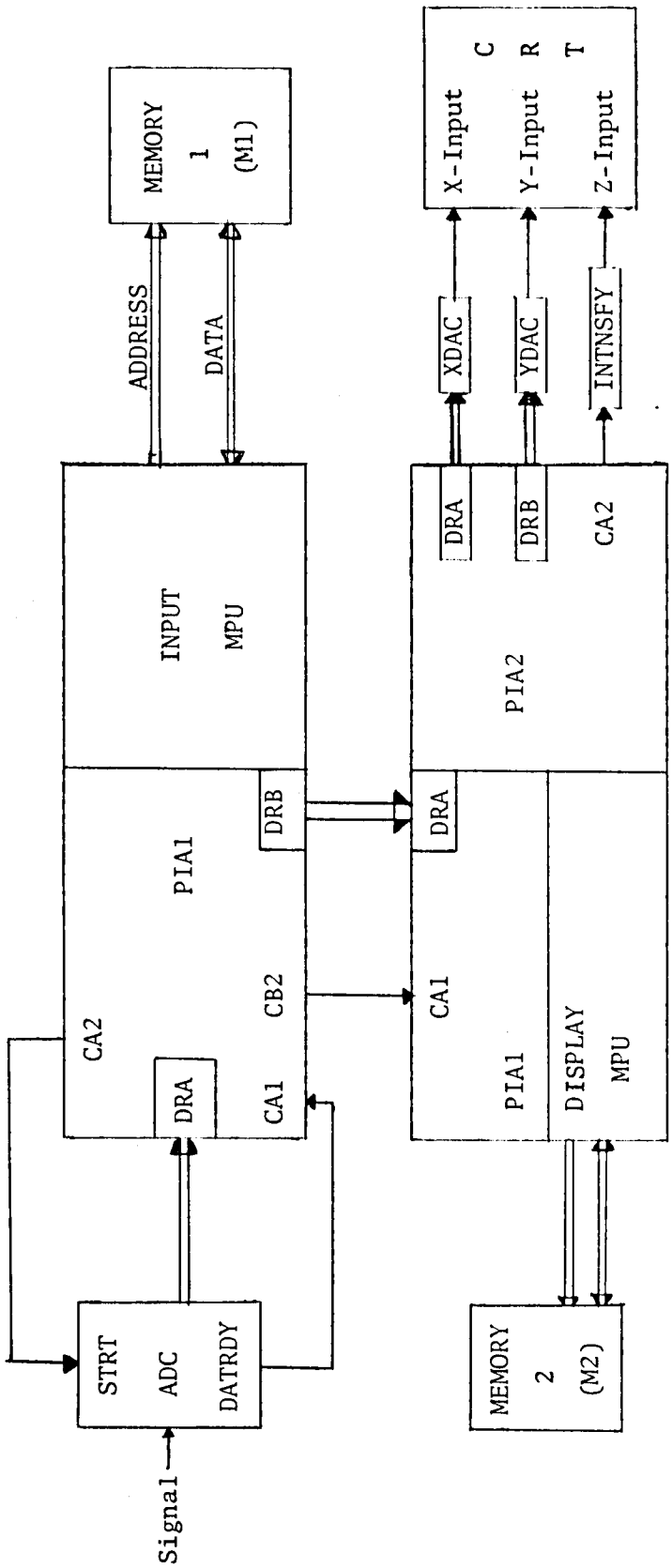
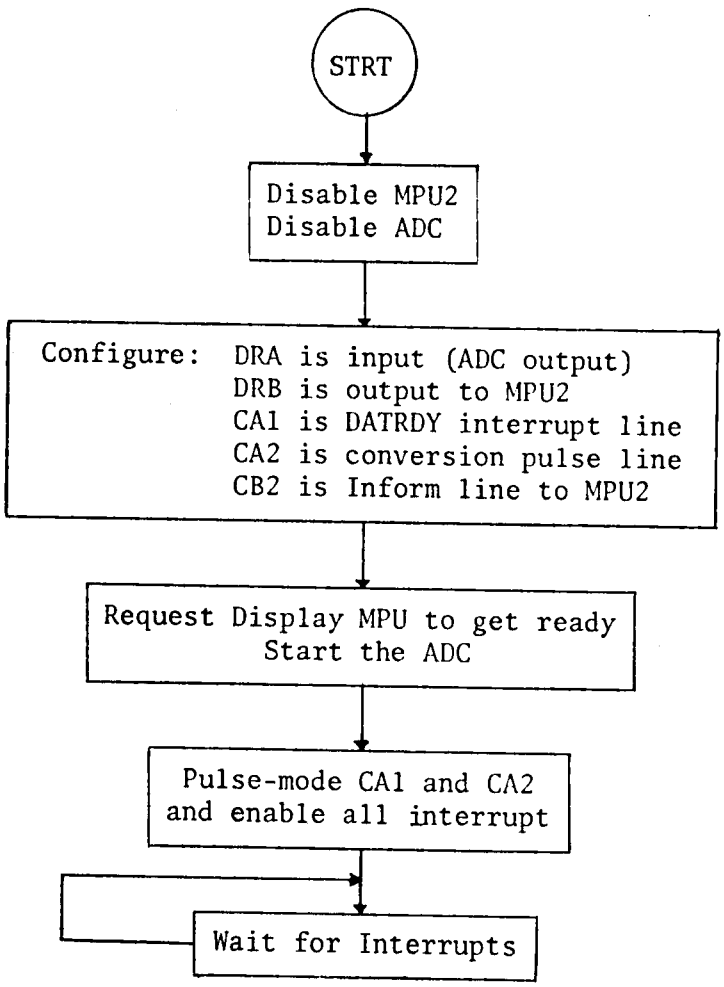
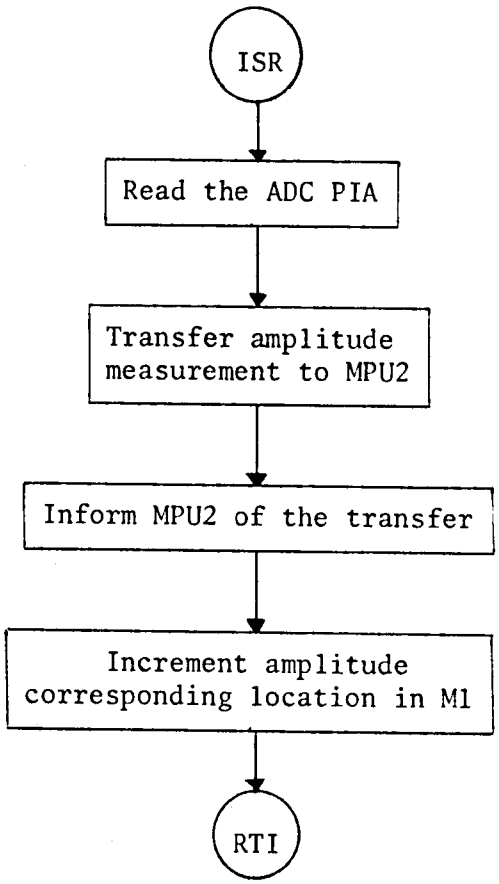



Figure 4.1. Amplitude Processor Block Diagram and Interface.




(a)

Figure 4.2. Amplitude Histogram: Input MPU Flowchart (a) Initialization and startup routine and (b) Data acquisition and transfer ISR routine.




(b)

Figure 4.2 continued

pulse-mode with CA1 (Fig. 4.2a). This automates the control of the ADC by issuing conversion pulses right after the ADC's PIA is read (Fig. 4.3).

Upon interrupts from CA1, indicating ADC end of conversion, the Input MPU reads the ADC port, transfers the amplitude sample to the Display MPU, transmits a data transfer signal, increments the corresponding address in memory 1 (M1), and returns from interrupt (Fig. 4.2b).

## 4.2 Display Microprocessor

In displaying the amplitude distribution, the interface and configuration (Fig. 4.1, p. 67) of the Display MPU and its software flowchart are the same as those in the last interval processor. The only difference is the nature of the data being histogrammed. On the CRT's x-axis RAM addresses that represent amplitudes are being displayed and their contents (the frequencies of occurrences) are being displayed on the CRT's y-axis (Fig. 3.6a, p. 33).

Initially, the Display MPU is started upon receiving the first high pulse on CA1 (Fig. 3.6b, p. 34). At this point the Display MPU enables all its interrupts. Future high pulses on CA1, indicating data transfers from the Input MPU, causes the Display MPU to read the new amplitude measurement from its PIA, increment the corresponding location in memory 2 (M2), and return from interrupt (Fig. 3.6b, p. 34).

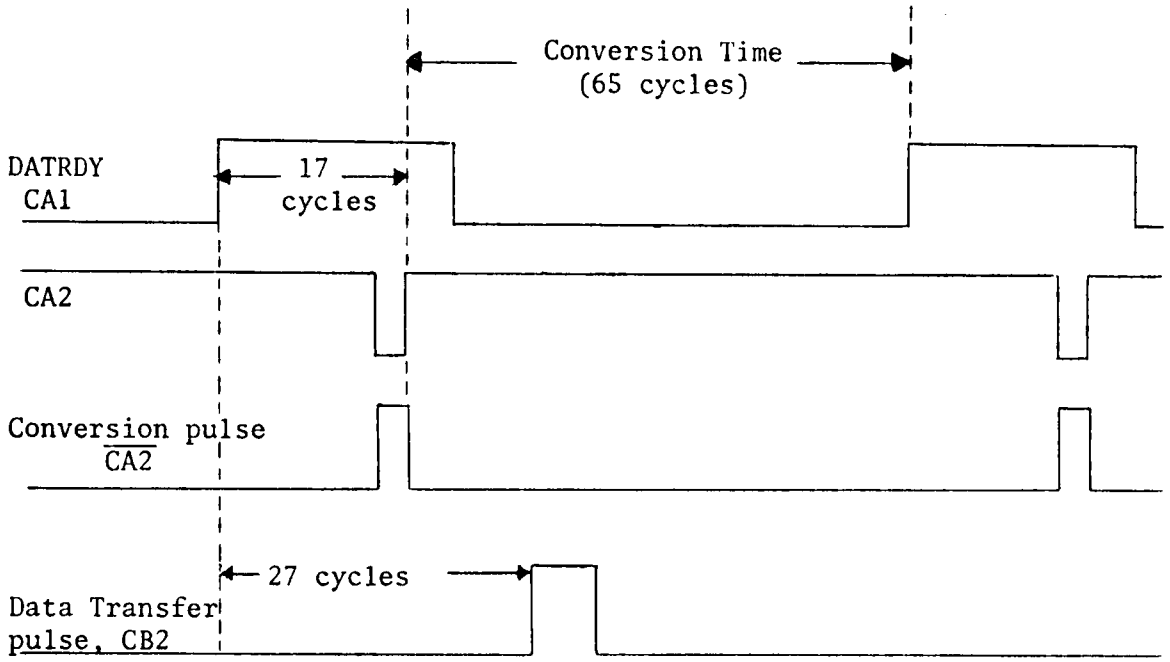



Figure 4.3. Control of ADC and MPU2 Timing Diagrams.

## 4.3 Analysis

The nature of the amplitude histogram requires a fast sampling rate to construct an accurate distribution. However, the program listed in Appendix A.6 indicates that ADC interrupts cannot occur faster than $T_{ISR}$ which is the time taken to service the interrupt. This time is the only factor that determines the sampling period which is given by

$$T_s \geq T_{ISR} \cdot \tag{4.1}$$

Calculating $T_{ISR}$ = 65 cycles × 1.628 μsec/cycle ≃ 106 μsec (Appendix A.6) indicates that the sampling rate, $f_s$ (which is the reciprocal of $T_s$), be less than 9.4 kHz. In accordance with Equation 3.2 (p. 22), the above sampling period limits the signal's shortest period to 423 μsec. This, in turn, limits the signal's highest frequency, $f_o$, to 2.36 kHz which is a severe restriction. In performing an amplitude histogram, there is no limit on the signal's lowest frequency content.

The ADC speed is controlled by two time factors, $T_{ISR}$ and $T_{READ}$ and is given as

$$T_{ADC} = T_{ISR} - T_{READ} , \tag{4.2}$$

where $T_{READ}$ is the time it takes the $\overline{CA2}$ line to go high. The timing diagrams of Figure 4.3 show that the ADC conversion time has to be longer than the time difference between the falling edge of $\overline{CA2}$ and the rising edge of CA1. This indicates that the ADC speed should not be slower than 65 cycles or 105 μsec.

On the other hand, the Display microprocessor is being interrupted at the rate of $f_s$ = 9.4 kHz for data transfer and update. Reconsidering Equation 3.8 (p. 37), the display refreshing rate is given by

$$\text{Refreshing Rate} = \frac{1 - T_{DATUPD} \times f_s}{(\text{RAM size})(T_{DISP})}, \tag{4.3}$$

where $f_s$, the sampling frequency, replaces $f_o$ in Equation 3.8 (p. 37). It follows that a refreshing rate of 32 times/sec is attained which is just above the minimum requirement. Therefore, the minimum memory size that can be displayed at the above given $f_s$ is 128 locations. This is another strict limitation of the amplitude processor which would become more stringent if a higher sampling rate was present.

## 4.4 Conclusion

As already concluded from the previous analysis, the limitation on the amplitude histogram is twofold:

1. The Input MPU is limiting the sampling rate, not only producing an inaccurate amplitude distribution but also limiting the signal's highest frequency, $f_o$.
2. The Display MPU is unable to display more than 128 locations at the given updating rate, $f_s$, and will deteriorate more once $f_s$ is increased.

These limitations are introduced predominantly by the existence of the microprocessor which is considered a slow device in signal processing.

# CHAPTER 5

# RESULTS

During the course of this project several interval processors were designed. The final scheme described in Chapter 3 (Section 3.1.4, p. 47) was built and tested (block diagram, Fig. 5.1). Supporting the two microprocessors were the various hardware components described in Chapter 3. A clock generator circuit was built around a 1-MHz crystal. Subfrequencies were obtained through cascading four decade counters, as diagrammed in Appendix B, producing fundamentals of 1M, 100k, 10k, and 1 kHz frequencies.

Due to the unavailability of a CRT and its associated interface, the second MPU displayed histograms on its own LEDs. Memory addresses were displayed on the four left most LEDs and their contents on the other two LEDs. A delay of approximately one second separated the consecutive address displays in order to enable one's eye to read the display. Appendix A.7 presents a complete description of the program.

A random test signal was generated by a third microcomputer which functioned as a discriminator to generate a preset number of intervals with preset known lengths (Appendix A.8).

## 5.1 Results

The pseudorandom signal having the series of intervals given in Table C-1, Appendix C, was applied to the input microcomputer selecting the counter's clock frequency at 1 kHz. Repeating the process at 10 kHz

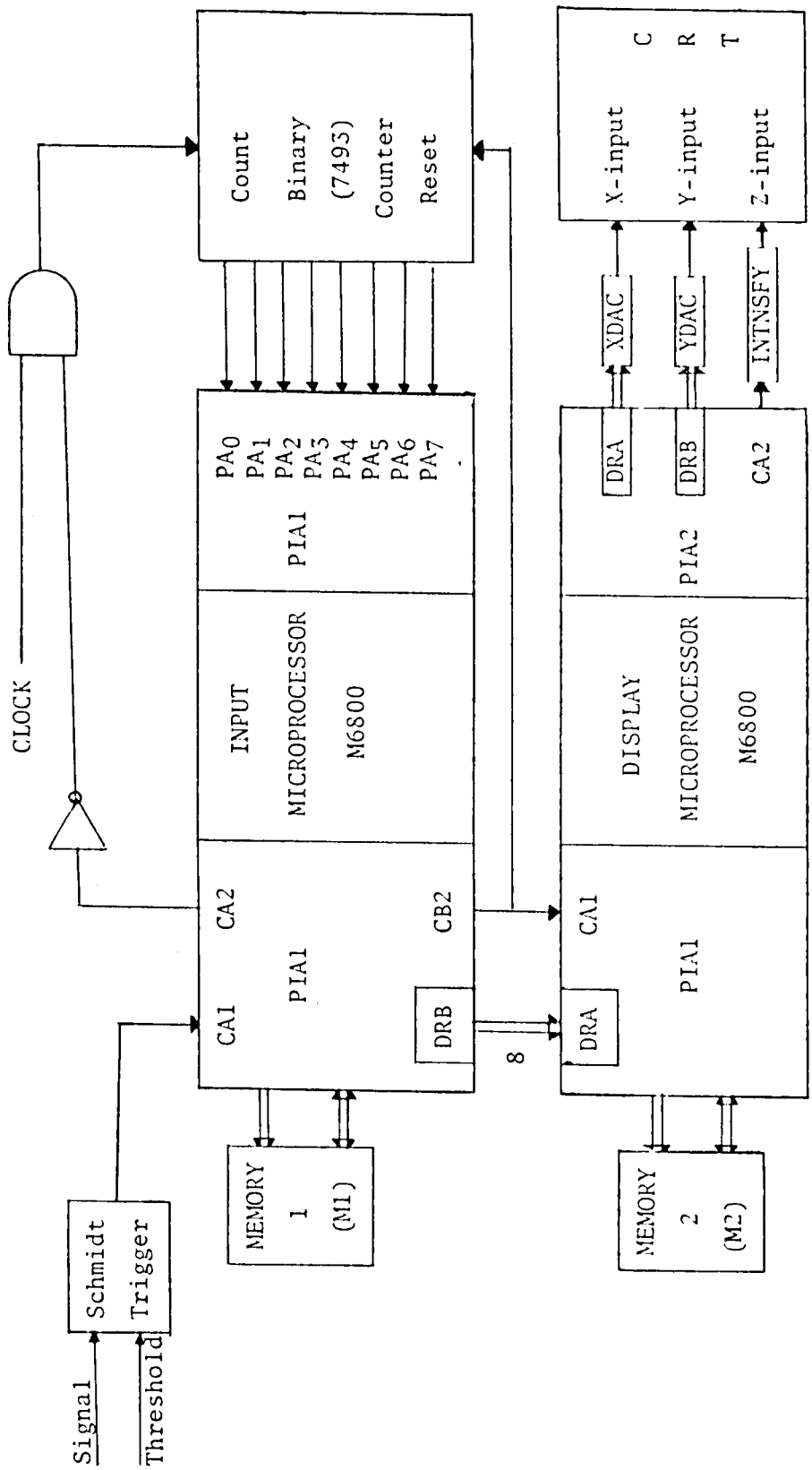
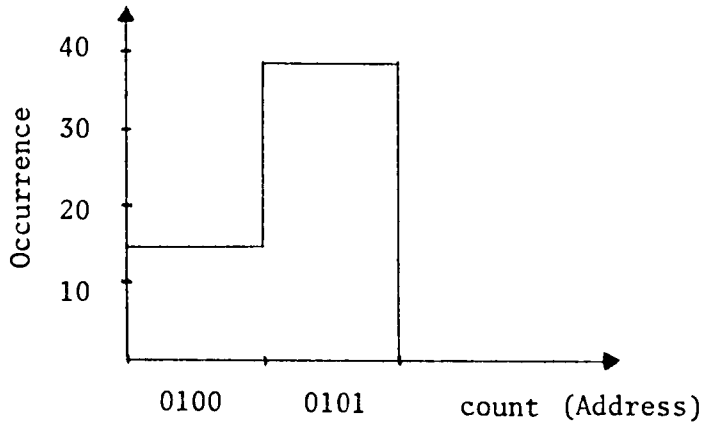



Figure 5.1. Tested Interval Processor: Block Diagram and Interface.

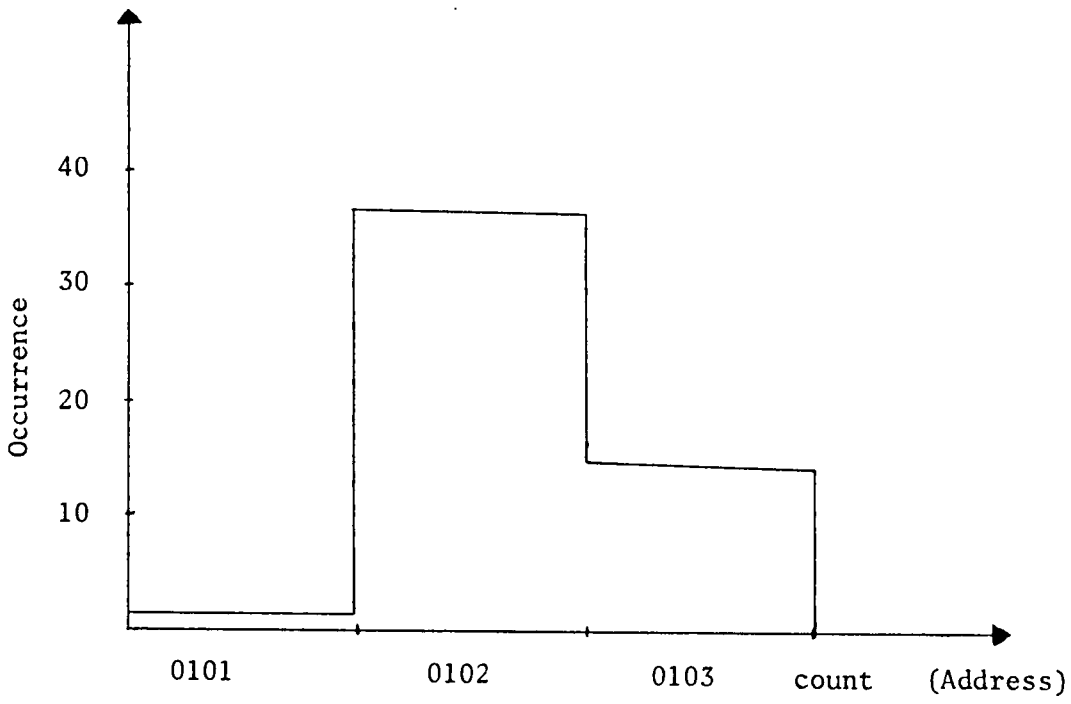
and 1 MHz frequencies, three separate time histograms were constructed. These histograms given in Tables C-1, C-2, and C-3 of Appendix C are plotted in Figure 5.2. Accumulation of interval measurements around the zero count in Figure 5.2a indicated that signals with periods close to each other, using low clock frequencies, produced distorted histograms. Higher clock frequencies as depicted in Figure 5.2c resulted in an appropriate interval distribution with a much better resolution. All histograms were constructed within an accuracy of ± 1 count as expected.

Since the nature of the pseudorandom signal program could not generate a period shorter than 143.5 sec (Table C-1), the highest signal's frequency was 7 kHz instead of 9 kHz. However, when the maximum interval limit, $T_h$, was exceeded the counter overflowed and produced an erroneous histogram value.

Finally, microcomputer-to-microcomputer communication was accomplished successfully as sufficient data obtained in Tables C-4 and C-5 proved correct synchronization and data transfers between the Input MPU and the Display MPU.

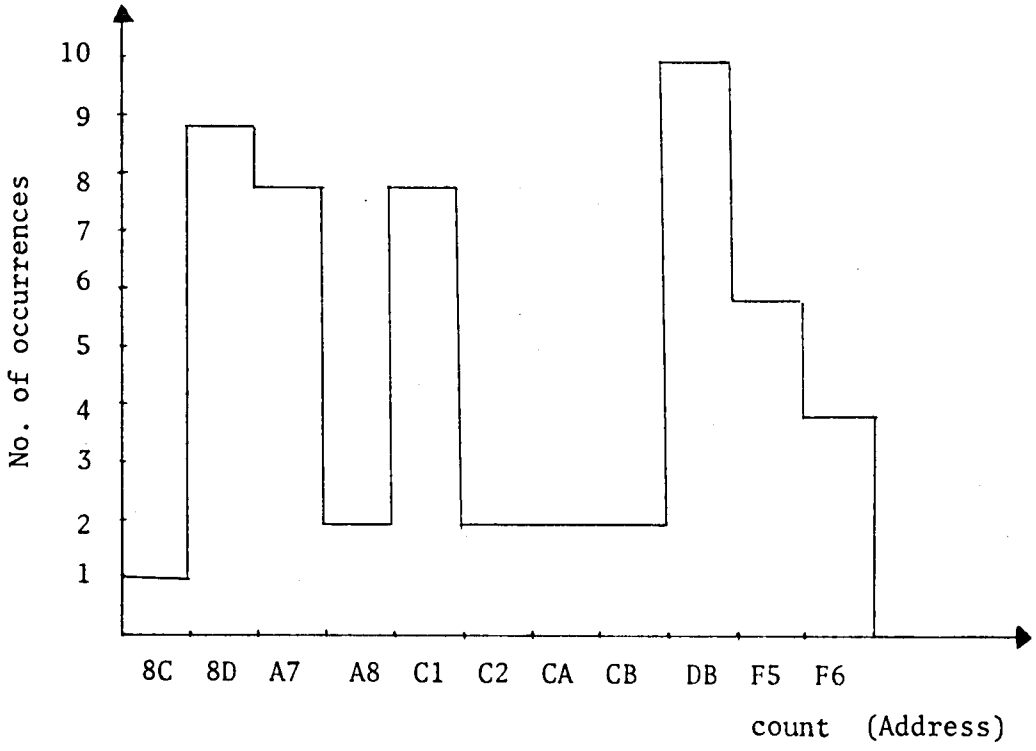



(a)




(b)

Figure 5.2. Experimental Time Histograms Plots (a) Table C-1 Plot, (b) Table C-2 Plot, and (c) Table C-3 Plot.




(c)

Figure 5.2 continued

# CHAPTER 6

## CONCLUSIONS

The interval, dwell, and amplitude processors that have been designed do not produce a significant improvement in the overall system performance even with extra modifications on the existing hardware. Both the dwell and interval processors had limited the signal's highest frequency, $f_o$, to less than 10 kHz. More severe was the restriction by the Amplitude processor on $f_o$ which was kept at a low 2.36 kHz. Moreover, the maximum memory size that could be displayed by the Display MPU was 384 locations at the given $f_o$. The amplitude processor placed even greater limitations on the display. This is a direct consequence of using microprocessors in signal processing.

Better performance demands that a faster CPU crystal replace the present 614.4 kHz MC6800 crystal. For instance, a 2-MHz crystal improves the restrictions by a factor of $2M/614.4K \simeq 3.255$ and, thus, an $f_o$ of 31.6 kHz will be permitted by the interval processor.

Even with the current schemes, the microprocessor family remains incapable of processing signals containing frequencies higher than 100 kHz. Therefore, logical substitutes are to use more hardwired design schemes, particularly in performing the display, or to introduce a faster technology such as the 2900 bit-slice family.

On the other hand, this work has developed several design procedures that are important in developing system design's techniques

for real-time applications. Multiprocessing was proved feasible and successful by introducing task distribution.

Finally, this work could be expanded to incorporate more signal processors such as a signal averager or an autocorrelator.

# LIST OF REFERENCES

## LIST OF REFERENCES


1. B. Soucek, Minicomputers in Data Processing and Simulation. New York, NY: Wiley-Interscience, 1972.

2. J. Allen, "Computer architecture for signal processing," Proceedings of the IEEE, Vol. 63, No. 4, pp. 624-633, April 1975.

3. F. B. Hiner III, "Distributed processing for signal processing using the building block signal processor," IEEE Proceedings of the 1976 International Conference on Parallel Processing.

4. Biomation/Datalab Technical Material, "Biomac 1000 signal analysis computer," Biomation, 1076 East Meadow Circle, Palo Alto, CA 94303.

5. J. Millman and C. C. Halkias, Integrated Electronics: Analog and Digital Circuits and Systems. New York, NY: McGraw-Hill, 1972.

6. A. Bruce Carlson, Communication Systems. New York, NY: McGraw-Hill, 1975.

# APPENDIXES

# APPENDIX A

## Software

### A.1 Single Microcomputer System Program

| CPU Cycle | Label | Instruction | | Comments |
|---|---|---|---|---|
| | | LDAA | # 01 | ; Location MEM is Interval |
| | | STAA | $MEM-1 | ; Counter. Address 0100 is base of Ram 3 |
| | | CLR | $MEM | ; Initialize counter to zero |
| | | CLR | B | ; Acc. B counts # of zero-crossings |
| | | LDAA | # 32 | ; CA1PIA1 sets IRQA1 at positive |
| | | STAA | CRA1 | ; transitions |
| | | STAA | CRB1 | ; Disable ADC by clearing CB2 PIA1 |
| | | STAA | CRA2 | ; CA2 PIA2 blanks display intensifier |
| | | CLR | DDRA1 | ; DRAPIA1 is input from ADC |
| | | CLR | CRB2 | ; reference DDRB2 |
| | | LDAA | # FF | ; |
| | | STAA | DDRA2 | ; DRBPIA1 is output to XDAC (location DAC) |
| | | STAA | DDRB2 | ; DRAPIA2 is output to YDAC (count DAC) |
| | | LDAA | # 36 | |
| | | STAA | CRA1 | ; |
| | | STAA | CRB1 | ; Reference all DRAs. |
| | | STAA | CRA2 | ; |
| | | STAA | CRB2 | ; |
| | | JSR | CONVRT | ; issue ADC conversion pulse |
| | THRESH: | LDAA | CRA1 | ; wait until ADC is done |
| | | BPL | THRESH | ; |
| | | LDAA | DRA1 | ; Obtain digitized data |
| | | JSR | CONVRT | ; conversion pulse |
| | | CMPA | V1 | |
| | | BHI | THRESH | ; $V_2 \leq$ is DATA < $V_1$ ? |
| | | CMPA | $V_2$ | |
| | | BLS | THRESH | ; No, get new data |
| | | STAA | LOC | ; Yes, retain current data (zero crossing) |
| | CHK: | LDAA | CRA1 | ; |
| | | BPL | CHK | ; Wait for next data |
| | | JSR | CONVRT | ; conversion pulse |
| | | LDAA | DRA1 | ; Get new data |

| CPU Cycle | Label | Instruction | | Comments |
|---|---|---|---|---|
| | | CMPA | LOC | ; is new data above the threshold? |
| | | BLT | CHK | ; No: negative edge ignored, wait |
| | | | | ; for a positive edge |
| | | INCB | | ; yes: positive transition, |
| | | | | ; initialize # zero crossings |
| | | INC | MEM | ; increment count by one |
| | | LDAA | # 37 | ; CB1PIA1 interrupt (clock) is |
| | | STAA | CRB1 | ; enabled. Counting starts. |
| | | CLI | | ; Enable system's interrupt |
| | | LDAA | #01 | |
| | | STAA | SAVE-1 | ; Base address 0100 is set up again |
| | | CLR | SAVE | ; to be used by the Display routine |
| 4 | DATRDY: | LDAA | CRA1 | ; Is ADC done? |
| 4 | | BPL | DISPLY | ; No: display |
| 4 | | LDAA | DRA1 | ; Yes: Get data |
| 2 | | LDAA | #3F | |
| 5 | | STAA | CRB1 | ; issue conversion pulse |
| 2 | | LDAA | #37 | ; clear pulse |
| 5 | | STAA | CRB1 | |
| 2 | | CMPA | #V1 | |
| 4 | | BHI | DISPLY; | ; |
| 2 | | CMPA | #V2 | ; Is data a threshold crossing? |
| 4 | | BLS | DISPLY | ; No: display |
| 2 | | INCB | | ; Yes: update # of threshold |
| | | | | ; crossings |
| 2 | | CMPB | #03 | ; Has 3 crossings occurred? |
| 4 | | BNE | DISPLY | ; No: display |
| 2 | | LDAA | #36 | ; Yes: One interval has elapsed. |
| 5 | | STAA | CRB1 | ; Disable clock interrupt. |
| 4 | HSTGRM: | LDX | MEM-1 | ; Get the interval count |
| 7 | | INC | 0,X | ; Increment corresponding address |
| 2 | | LDAB | #01 | ; Reinitialize # of threshold |
| | | | | ; crossings |
| 4 | | STAB | MEM | ; Reinitialize count to one |
| 2 | | LDAA | #37 | |
| 5 | | STAA | CRB1 | ; Enable clock interrupt |
| 4 | | JMP | DISPLY | ; go display |
| | CONVRT: | STAA | TEMP | ; |
| | | LDAA | #3E | ; Routine to issue a |
| | | STAA | CRB1 | ; conversion pulse |
| | | LDAA | #36 | ; |
| | | STAA | CRB1 | |
| | | LDAA | TEMP | |
| | | RTS | | |

| CPU Cycle | Label | Instruction | | Comments |
|---|---|---|---|---|
| 4 | DISPLY: | LDX | SAVE-1 | ; Get current location's address |
| 3 | | LDAA | SAVE | ; Get LSB of address (right 8-bits) |
| 5 | | STAA | DRA2 | ; Display it on XDAC (address DAC) |
| 5 | | LDAA | 0,X | ; Get content of the address |
| 5 | | STAA | DRB2 | ; Display it on YDAC (count DAC) |
| 2 | | LDAA | #3E | ; Intensify display |
| 5 | | STAA | CRA2 | ; |
| 2 | | LDAA | #36 | ; Blank display |
| 5 | | STAA | CRA2 | ; |
| 6 | | INC | SAVE | ; Advance to next address |
| 4 | | INX | | |
| 3 | | CPX | #0180 | ; Is all memory displayed? |
| 4 | | BNE | DATRDY | ; Not yet |
| 6 | | CLR | SAVE | ; Yes: initialize pointer to base ; of Data RAM |
| 4 | | BRA | DATRDY | ; Always check status of ADC. |
| 6 | ISRCLK: | INC | MEM | : Proceed interval count by one |
| 4 | | LDAA | DRB1 | ; Clear IRQB1 bit |
| 10 | | RTI | | ; Return from interrupt |

Threshold crossing detection

A zero crossing is determined as lying within the voltage range from $V_2$, a slightly negative voltage, to $V_1$, a slightly positive voltage (Figure A.1).

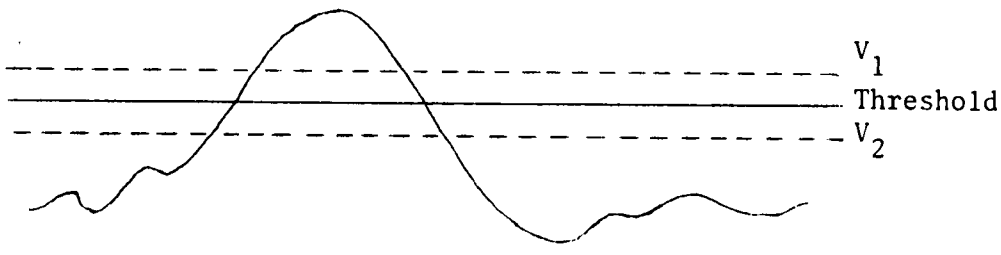



Figure A.1. Zero Crossing Uncertainty.

Time Parameters

TACQ: Data acquisition time includes the program segment starting at label DATRDY and ending at label HSTGRM. Summing up CPU cycles, TACQ amounts to 53 cycles × 1.628 μsec/cycle = 86 μsec.

THST: Data histogramming time includes program segment starting at label HSTGRM And ending at label CONVRT. It amounts to 28 cycles × 1.628 μsec/cycle ≃ 45.6 μsec.

$T_{ISR}$: Clock interrupt service routine which amounts to 32 cycles × 1.628 μsec/cycle ≃ 52 μsec. Extra 12 cycles are needed by the MPU before it access the first instruction in ISR.

$T_{DISP}$: Display routine time starting at label DISPLY and ending at label ISRCLK. It amounts to about 102.5 μsec.

$T_{DATRDY}$: Time to check ADC status. It includes two instructions beginning at label DATRDY and equals 8 cycles × 1.628 μsec/cycle ≃ 13 μsec.

## A.2 Two-Microcomputer System Programs

### Input Processor Program

| CPU Cycle | Label | Instruction | | Comments |
|---|---|---|---|---|
| | STRT: | LDAA | #32 | ; |
| | | STAA | CRA | : No signals to MPU2 |
| | | STAA | CRB | ; Disable ADC |
| | | LDAA | #01 | ; |
| | | STAA | MEM-1 | ; Set up base address of RAM 3 |
| | | CLR | MEM | ; initialize counter |
| | | CLRB | | ; clear zero-crossings counter |
| | | LDAA | #FF | ; |
| | | STAA | DDRB | ; DRB is output to MPU2 |
| | | CLR | DDRA | ; DRA is input from ADC |
| | | LDAA | #36 | ; |
| | | STAA | CRA | ; DRA is referenced. |
| | | STAA | CRB | ; DRB is referenced. |
| | | JSR | CONVRT | ; issue conversion pulse |
| | THRESH: | LDAA | CRA | ; wait until ADC is done |
| | | BPL | THRESH | ; |
| | | LDAA | DRA | ; Obtain digitized data |
| | | JSR | CONVRT | ; conversion pulse |
| | | CMPA | V1 | |
| | | BHI | THRESH | ; $V_2 \leq$ is Data < $V_1$ ? |
| | | CMPA | $V_2$ | |
| | | BLS | THRESH | ; No, get new data |
| | | STAA | LOC | ; Yes, retain current data |
| | | | | ; (zero crossing) |
| | CHK: | LDAA | CRA | ; |
| | | BPL | CHK | ; Wait for next data |
| | | JSR | CONVRT | ; conversion pulse |
| | | LDAA | DRA | ; Get new data |
| | | CMPA | LOC | ; is new data above the threshold ? |
| | | BLT | CHK | ; No: negative edge ignored, wait |
| | | | | ; for a positive edge |
| | | INCB | | ; Yes: positive transition, |
| | | | | ; initialize # zero crossings |
| | | INC | MEM | ; increment count by one |
| | | LDAA | #37 | ; Enable clock |
| | | STAA | CRB | ; interrupt |
| | | CLI | | ; ENable system's interrupt |
| | | LDAA | #3E | ; Request MPU2 to get ready |
| | | STAA | CRA | ; CA2 = 1 |

| CPU Cycle | Label | Instruction | | Comments |
|---|---|---|---|---|
| | | LDAA | #36 | ; clear request (CA2 = 0) |
| | | STAA | CRA | |
| 4 | DATRDY: | LDAA | CRA | ; Is ADC done? |
| 4 | | BPL | DISPLY | ; No: display |
| 4 | | LDAA | DRA | ; Yes: Get data |
| 2 | | LDAA | #3F | |
| 5 | | STAA | CRB | ; issue conversion pulse |
| 2 | | LDAA | #37 | ; clear pulse |
| 5 | | STAA | CRB | |
| 2 | | CMPA | #V1 | |
| 4 | | BHI | DISPLY | ; Is Data a threshold crossing? |
| 2 | | CMPA | #$V_2$ | |
| 4 | | BLS | DISPLY | ; No: display |
| 2 | | INCB | | ; Yes: update # of threshold crossin ; crossings |
| 2 | | CMPB | #03 | ; Has 3 crossings occurred? |
| 4 | | BNE | DATRDY | ; No: display |
| 2 | | LDAA | #36 | ; Yes: One interval has elapsed. |
| 5 | | STAA | CRB | ; Disable clock interrupt. |
| 4 | HSTGRM: | LDX | MEM-1 | ; Get the interval count |
| 7 | | INC | 0,X | ; Increment corresponding address |
| 4 | | LDAA | MEM | ; Get count |
| 5 | | STAA | DRB | ; Transfer it to MPU2 |
| 2 | | LDAB | #01 | ; initialize zero crossings |
| 4 | | STAB | MEM | ; initialize count |
| 2 | | LDAA | #3E | |
| 5 | | STAA | CRA | ; inform MPU2 of data transfer |
| 2 | | LDAA | #36 | |
| 5 | | STAA | CRA | ; clear inform signal |
| 2 | | LDAA | #37 | |
| 5 | | STAA | CRB | ; Enable clock interrupt |
| | CONVRT: | STAA | TEMP | ; |
| | | LDAA | #3E | ; Routine to issue a |
| | | STAA | CRB | ; conversion pulse |
| | | LDAA | #36 | |
| | | STAA | CRB | |
| | | LDAA | TEMP | |
| | | JSR | | |
| 6 | ISRCLK: | INC | MEM | ; increment counter by one |
| 4 | | LDAA | DRB | ; clear IRQB1 (Dummy Read) |
| 10 | | RTI | | |

Input MPU Time Parameters

Except for $T_{DISPLY}$, which is replaced by $T_{UPDATE}$, all other parameters are the same for the Input MPU.

$T_{UPDATE}$: is the time taken to update the Display MPU with new data. It is embodied in the histogram portion starting at label HSTGRM. Instructions responsible for the update sum up to 23 cycles, thus, producing $T_{UPDATE}$ of 37.5 μsec.

Display MPU Program

```
CPU
Cycle  Label      Instruction                  Comments

                  LDAA    #01        ; Set up base address (0100)
                  STAA    LOC-1      ; of display RAM
                  CLR     CRA1
                  CLR     CRA2       ; Reference DDRAs
                  CLR     CRB2
                  CLR     DDRA1      ; DRAPIA1 is input from MPU1
                  LDAA    #FF
                  STAA    DDRA2      ; DRAPIA2 is output to XDAC
                  STAA    DDRAB2     ; DRBPIA2 is output to YDAC
                  LDAA    #36        ; CA1PIA1 sets IRQA1 PIA1 at
                  STAA    CRA1       ; positive pulses from MPU1
                  STAA    CRA2       ; Blank display (CA2PIA2 = 0)
                  STAA    CRB2       ; DRAs are all referenced

        RDY:      LDAA    CRA1       ; wait for start signal
                  BPL     RDY        ; from MPU1
                  LDAA    #37        ; Get ready:
                  STAA    CRA        ; Enable CA1 interrupt
                  LDAA    DRA1       ; clear IRQA1 bit
                  CLI                ; Enable system's interrupt

 3      DISPLY:   LDX     #0100
 2                CLRA               ; Acc. A holds LSB of address

 2      MORE:     LDAB    #34        ; CA2PIA2 = 0
 5                STAB    CRA2       ; Blank display
 5                LDAB    0,X        ; Get data indicated by X
 5                STAB    DRB2       ; display at y-axis
 5                STAA    DRA2       ; display LSB of address at x-axis
```

| CPU Cycle | Label | Instruction | | Comments |
|---|---|---|---|---|
| 2 | INTSFY: | LDAB | #3C | ; |
| 5 | | STAB | CRA2 | ; Intensify the display |
| 2 | | INCA | | ; Advance to next address LSB |
| 4 | | INX | | ; Advance to next address in Ram |
| 3 | | CPX | #0180 | ; Have 127 Locations been displayed? |
| 4 | | BNE | MORE | ; No: Continue through Ram |
| 4 | | BRA | DISPLY | ; Yes: start from base address |
| | | | | ; (i.e. refresh display) |
| 4 | ISRMP1: | LDAA | DRA1 | ; Get the count (updated value) |
| 4 | | STAA | LOC | ; Prepare for Ind. Reg. loading |
| 4 | | LDX | LOC-1 | ; count is being constructed as |
| | | | | ; an address |
| 7 | | INC | 0,X | ; increment the corresponding |
| | | | | ; location |
| 10 | | RTI | | |

Display MPU Time Parameters

$T_{DISPLY}$: Is the time taken by the display routine starting at label STRT and ending at label ISRMP1. $T_{DISPLY}$ amounts to about 51 cycles × 1.628 μsec/cycle ≃ 83 μsec.

$T_{DATUPD}$: Is MPU1 interrupt service routine time during which MPU2 updates itself with new data from MPU1. $T_{DATUPD}$, including 12 CPU cycles to reach ISR, sums up to 41 cycles × 1.628 μsec/cycle ≃ 66.8 μsec.

## A.3 Two Microcomputer System (Schmidt Trigger Interface) Program

```
CPU
Cycle  Label        Instruction                  Comments

                    LDAB    #34          ; No signals to MPU2.  Bit 0 is
                                         ; cleared
                    STAB    CRA          ; Schmidt's interrupt disabled
                    CLR     CRB
                    LDAA    #FF
                    STAA    DDRB         ; DRB is output to MPU2
                    STAB    CRB          ; clock and its interrupt disabled
                    CLR     MEM          ; clear counter
                    LDAB    #01          ; set up base address of
                    STAB    MEM-1        ; data RAM3.

        SCHMDT:     LDAA    CRA          ; wait for first positive zero
                    BPL     SCHMDT       ; crossing of signal.
                    LDAA    #3E          ; Request MPU2 to become
                    STAA    CRA          ; operational
                    LDAA    #36
                    STAA    CRA
                    STAA    CRB          ; CB2 = 1 start clock
                    INC     MEM          ; compensate for a lost count
                    LDAA    DRA          ; IRQA1 is cleared
                    CLI                  ; Enable system's interrupt
                    LDAA    #37
                    STAA    CRA          ; Enable Schmidt interrupt.

  4     CLKCHK:     LDAA    CRB          ; is clock interrupting?
  4                 BPL     CLKCHK       ; No: loop
  6                 INC     MEM          ; Yes: Increment count
  4                 LDAA    DRB          ; Clear IRQB1 bit
  4                 BRA     CLKCHK       ;

  3     ISRSCH:     LDAA    MEM          ; Read count
  5                 STAA    DRB          ; Transfer it to MPU2
  2                 LDAA    #3F
  5                 STAA    CRA          ; Tell MPU2
  2                 LDAA    #37
  5                 STAA    CRA          ; clear pulse
  4                 LDX     MEM-1        ; Get equivalent count address
  7                 INC     0,X          ; Increment corresponding address
  6                 CLR     MEM          ; Clear count
  4                 LDAA    DRA          ; Clear IRQA1 bit
 10                 RTI
```

Time Parameters

$T_{CLKCHK}$: Time consumed in searching for a clock pulse. It starts at label CLKCHK and ends at label ISRSCH. Summing up CPU cycles yields $T_{CLKCHK}$ = 22 cycles × 1.628 μsec/cycle ≃ 35.8 μsec.

$T_{ISR}$: Is Schmidt Trigger's interrupt service routine which sums up to 106 μsec.

Display MPU

The program for the Display MPU is the same as that used in Appendix A.2.

## A.4 Two-Microcomputer System (Schmidt Trigger Interface) Program

### Input Microprocessor Program

| CPU Cycle | Label | Instruction | | | Comments |
|---|---|---|---|---|---|
| | CLRMEM: | LDX | # 0100 | ; | |
| | CLEAR: | CLR | 0,X | ; | clear address indicated by X |
| | | INX | | ; | |
| | | CPX | # 0200 | ; | Have all memory been cleared |
| | | BNE | CLEAR | ; | No: loop |
| | STRT: | LDAA | # 32 | ; | CB2=0, No signals to MPU2 |
| | | STAA | CRB | | |
| | | LDAA | # 3A | ; | $\overline{CA2}$=0, inhibit the clock |
| | | STAA | CRA | | |
| | | CLR | DDRA | ; | DRA is input device from counter |
| | | LDAA | # FF | | |
| | | STAA | DDRB | ; | DRB is output to MPU2 |
| | | LDAA | # 01 | ; | Set up base address of |
| | | STAA | MEM-1 | ; | the data RAM |
| | THRSH: | LDAA | CRA | ; | wait for the start of the |
| | | BPL | THRSH | ; | first period (first positive edge) |
| | | LDAA | # 3E | ; | CB2=1, Request MPU2 to operate |
| | | STAA | CRB | ; | same pulse resets counter |
| | | LDAA | # 36 | ; | clear pulse and thus |
| | | STAA | CRB | ; | counter can resume counting |
| | | STAA | CRA | ; | $\overline{CA2}$=1 clock is enabled |
| | | LDAA | DRA | ; | clear IRQA1 bit |
| | | CLI | | ; | enable system's interrupt |
| | | LDAA | # 27 | ; | CA1/CA2 are handshaked |
| | | STAA | CRA | ; | Schmidt interrupt is enabled |
| 2 | WAIT: | LDAA | # 00 | ; | Dummy load |
| 4 | | BRA | WAIT | ; | Wait for interrupts |
| 4 | ISR: | LDAA | DRA | : | Read the Counter; enable clock rightafter |
| 2 | | ADDA | # C | ; | correction: compensate for lost counts |
| 5 | | STAA | DRB | ; | Transfer count to Display MPU |
| 2 | | LDAB | # 3E | ; | |
| 5 | | STAB | CRB | ; | Inform MPU2 and reset counter |
| 2 | | LDAB | # 36 | ; | clear the pulse |
| 5 | | STAB | CRB | ; | counter can resume counting |

| CPU Cycle | Label | Instruction | | Comments |
|---|---|---|---|---|
| 4 | | STAA | MEM | ; store count: prepare for |
| 5 | | LDX | MEM-1 | ; Index register loading |
| 7 | | INC | 0,X | ; Increment corresponding address |
| 10 | | RTI | | ; Return from interrupt. |

$T_{ISR}$: is the Interrupt Service Routine Execution time which adds up to 63 cycles (100 μsec).

$T_{LOOP}$: is the execution time of the "Wait for Interrupt Loop" which adds up to 6 cycles (9.7 μsec).

## Display Microprocessor Program

The program is the same as that of Appendix A.2.

## A.5 Dwell Processor Program

Input MPU

| CPU Cycle | Label | Instruction | | Comments |
|---|---|---|---|---|
| | START: | LDAA | #32 | |
| | | STAA | CRB2 | ; No signals to MPU2 (CB2PIA2 = 0) |
| | | CLR | CRB1 | |
| | | CLR | DDRB1 | ; DRBPIA1 is input from counter |
| | | CLR | CRA1 | |
| | | LDAA | #FF | |
| | | STAA | DDRA1 | ; DRAPIA1 is output to MPU2 |
| | | LDAA | #36 | ; CA1PIA1 is configured to set |
| | | STAA | CRA1 | ; IRQA1 bit at positive transitions. |
| | | LDAA | #01 | |
| | | STAA | LOC-1 | ; fix base address |
| | THRESH: | LDAA | CRA1 | |
| | | BPL | THRESH | ; Wait for start of first interval |
| | | LDAA | #3F | ; $\overline{CA2}$PIA1 = 1; Reset counter |
| | | STAA | CRA1 | ; Counting resumed. Interrupt |
| | | | | ; enabled. |
| | | LDAA | #27 | |
| | | STAA | CRA1 | ; handshake CA1 and CA2 |
| | | STAA | CRB1 | ; handshake CB1 and CB2 |
| | | LDAA | #3E | |
| | | STAA | CRB2 | ; Request MPU2 to get ready |
| | | LDAA | #36 | |
| | | STAA | CRB2 | ; Clear request pulse |
| | | CLI | | |
| | WAIT: | LDAA | #00 | |
| | | BRA | WAIT | ; Wait for interrupts |
| 4 | ISRPOL: | LDAA | CRA1 | ; If interrupt comes from |
| 4 | | BMI | ISRCA1 | ; CA1, service it |
| 4 | | LDAA | CRB1 | ; If no: check if it comes from |
| 4 | | BMI | ISRCB1 | ; CB1: service it |
| 10 | | RTI | | ; If no: Return from interrupt |
| 4 | ISRCA1: | LDAA | DRA1 | ; Dummy read to reset the counter |
| 10 | | RTI | | |
| 4 | ISRCB1: | LDAB | DRB1 | ; Read counter |
| 2 | | ADDB | #N | ; Compensate for lost # of counts. |
| 5 | | STAB | DRA1 | ; Transfer count to MPU2 |

| CPU Cycle | Label | Instruction | | Comments |
|---|---|---|---|---|
| 2 | | LDAA | #3E | |
| 5 | | STAA | CRB2 | ; Inform it |
| 2 | | LDAA | #36 | |
| 5 | | STAA | CRB2 | |
| 4 | | STAB | LOC | ; Prepare for Ind. Reg. loading |
| 5 | | LDX | LOC-1 | ; |
| 7 | | INC | 0,X | ; Increment corresponding location. |
| 10 | | RTI | | ; Return from Interrupt. |

## A.6 Amplitude Histogram Program

| CPU Cycle | Label | Instruction | | Comments |
|---|---|---|---|---|
| | CONFIG: | LDAA | #32 | |
| | | STAA | CRB | ; Disable MPU2 (CB2 = 0) |
| | | LDAA | #3A | |
| | | STAA | CRA | ; Disable ADC ($\overline{CA2}$ = 0) |
| | | CLR | DDRA | ; DRA is input from ADC |
| | | LDAA | #FF | |
| | | STAA | DDRB | ; DRB is output to MPU2 |
| | | LDAA | #3E | |
| | | STAA | CRB | ; CB2 = 1, request MPU2 to get ready |
| | | LDAA | #36 | |
| | | STAA | CRA | ; $\overline{CA2}$ = 1, start the ADC |
| | | LDAA | #2F | |
| | | STAA | CRA | ; Pulse-mode CA1 and CA2 |
| | | CLI | | |
| | | LDAA | #01 | |
| | | STAA | LOC-1 | ; set up RAM's base address. |
| 2 | WAIT: | LDAA | #00 | ; Dummy load |
| 4 | | BRA | WAIT | ; Wait for interrupts |
| 4 | ISR: | LDAA | DRA | ; Read the ADC PIA |
| 5 | | STAA | DRB | ; Transfer amplitude to MPU2 |
| 2 | | LDAB | #3E | |
| 5 | | STAB | CRB | ; Inform MPU2 |
| 2 | | LDAB | #36 | |
| 5 | | STAB | CRB | ; clear pulse |
| 4 | | STAA | LOC | ; |
| 5 | | LDX | LOC-1 | |
| 7 | | INC | 0,X | ; Increment corresponding address |
| 10 | | RTI | | |

$T_{ISR}$: is the time taken to service ISR added to it 4 cycles (worst case) from the "WAIT" for interrupt loop. It adds up to 65 cycles.

$T_{READ}$: is the time taken to reach ISR (12 cycles) added to it 4 cycles of the "Read PIA" instruction.

## A.7 LED Display Program

```
Label      Instruction                  Comments

CNFGR:     CLR     CRA      ;
           CLR     DDRA     ; DRA is input (output of MPU1)
           LDAA    # 36     ; IRQA1 bit gets set at positive
           STAA    CRA      ; transitions on CA1. Schmidt's
                              INT. disabled
           LDAA    # 01     ; MSB of RAM addresses is
           STAA    MEM-1    ; retained (to be used by
                              Histgrm ISR)

RDY:       LDAA    CRA      ;
           BPL     RDY      ; Wait for a signal from MPU1
           LDAA    DRA      ; clear IRQA1 bit
           LDAA    # 37     ;
           STAA    CRA      ; Enable CA1 interrupts
           CLI              ; Enable system's interrupt
           JMP     DISPLY

DSPLY:     CLR     $A00C    ; Sixth LED Buffer holds Ø
           CLR     $A00D
           INC     $A00D    ; fifth LED Buffer holds 1

STRT:      LDX     # 0100   ;
           LDAB    # 00     ; start with LSB of base address

LOAD:      STAB    TEMP     ; retain address pointer
           ANDB    # 0F     ; retain least signif. nipple
                              of LSB
           STAB    $A00F    ; send it to third LED Buffer
           LDAB    TEMP     ; Restore address pointer
           LSRB             ;
           LSRB             ; Shift left to
           LSRB             ; retain most signif. nipple
           LSRB             ; of LSB.
           STAB    $A00E    ; fourth LED Buffer loaded
           LDAA    0,X      ; Get contents of location specified
                              by X
           ANDA    # 0F     ; Retain least signif. nipple
                              of LSB
           STAA    $A011    ; Load it to first LED Buffer
           LDAA    0,X      ; Get the same data again
           LSRA             ;
           LSRA             ; shift left four times
           LSRA             ; to obtain most signif.
           LSRA             ; nipple of LSB
```

```
Cycle   Label     Instruction                  Comments

                  STAA    $A010      ; store at second LED Buffer
                  STX     LOC        ; save Ind. Reg.
                  LDAB    # 50       ; set up display refresh counter

        REFRSH:   LDX     # DISBUF   ; Get starting address of Buffers
                  LDAA    # 20       ; count for
                  STAA    $A01C      ; scan counter

        OUTDS1:   LDAA    0,X        ; Get first digit
                  INCA
                  INX
                  STX     XDSBUF     ; save pointer
                  LDX     #DIGTBL-1

        OUTDS2:   INX
                  DECA               ; point to pattern
                  BNE     OUTDS2
                  CLR     SCNREG     ; Blank display
                  LDAA    0,X        ; Get pattern
                  STAA    DISREG     ; Set up segments
                  LDAA    SCNCNT     ;
                  STAA    SCNREG     ; Select digit
                  LDX     XDSBUF     ; Recover pointer
                  CPX     #DISBUF+6; Have six Buffers been displayed?
                  BEQ     MORE       ; Yes: refresh display
                  LSR     SCNCNT     ; No: prepare to display
                  BRA     OUTDS1     ; the rest of Buffers.

        MORE      DECB               ;
                  BNE     REFRSH     ; Accomplish delay between displays.

        CONTNU:   LDAB    TEMP       ; Recover pointer to Data RAM
                                       addresses
                  LDX     LOC        ; Recover Ind. Reg.
                  INX
                  CPX     # 0200     ; Has all memory been displayed?
                  BEQ     STRT       ; Yes: start again with base address
                  INCB               ; No: advance to next location
                  JMP     LOAD

  4     ISRHST:   LDAA    DRA        ; Read counter
  4               STAA    MEM        ; Prepare for Ind. Reg. loading
  5               LDX     MEM-1      ; Get equivalent address of count
  7               INC     0,X        ; increment it by one
 10               RTI                ; Return from interrupt.
```

## A.8 Random Signal Generator

### Program

| CPU Cycle | Label | Instruction | | Comments |
|---|---|---|---|---|
| | | CLR | MEM-1 | ; MEM is used by delay counter |
| | CONFGR: | CLR | CRA | ; |
| | | LDAA | # FF | ; Bit Ø of DRA is an output |
| | | STAA | DDRA | |
| | | LDAA | # 04 | |
| | | STAA | CRA | ; Reference DRA |
| | | CLR | DRA | |
| | | LDX | # N0 | ; $N_0$ different Interval groups |
| | | LDAA | # $N_2$ | ; $N_2$ determines initial delay |
| 4 | NEWPRD: | STAA | MEM | ; MEM is delay counter |
| 2 | | LDAB | # $N_1$ | ; Each Interval group repanted |
| | | LDAB | | ; $N_1$ times |
| 6 | | CLR | DRA | ; No intervals yet |
| 5 | | STX | LOC | ; |
| 3 | | LDX | # OA | ; |
| 4 | MINT0: | DEX | | ; Delay to be used to guarantee |
| 4 | | BNE | MINT0 | ; that shortest period > $T_o$ |
| 4 | | LDX | LOC | |
| 6 | STRT: | INC | DRA | ; start the interval (positive edge) |
| 9 | | JSR | DLYSUB | ; Delay |
| 6 | | CLR | DRA | ; Negative edge |
| 9 | | JSR | DLYSUB | ; Delay |
| 2 | | DECB | | ; Has $N_2$ intervals of same |
| 4 | | BNE | STRT | ; group been generated? No |
| 6 | | INC | DRA | ; Yes: positive edge (start of new group) |
| 6 | | CLR | DRA | ; |
| 2 | | ADDA | # D | ; increase initial delay to generate |
| 4 | | DEX | | ; next interval group |
| 4 | | BNE | NEWPRD | ; All groups generated? |
| | STOP: | BRA | STOP | ; Yes: stop |
| 5 | DLYSUB: | STX | TEMP | ; Save X |
| 4 | | LDX | MEM-1 | ; MEM holds the delay variable |
| 4 | DLY: | DEX | | |
| 4 | | BNE | DLY-1 | ; Decrement until delay is over |

| CPU Cycle | Label | Instruction | | Comments |
|---|---|---|---|---|
| 4 | | LDX | TEMP | |
| 5 | | RTS | | ; Return from subroutine |
| | | Routine for longer Delays | | |
| 4 | DLYSUB: | STAB | TEMP-2 | |
| 3 | | LDAB | TEMP-3 | ; Use memory change to enter second counter |
| 5 | | STX | TEMP | ; |
| 4 | DLY1: | LDX | MEM-1 | |
| 4 | DLY2: | DEX | | |
| 4 | | BNE | DLY2 | ; Inner loop |
| 2 | | DECB | | |
| 4 | | BNE | DLY1 | ; Outer loop |
| 4 | | LDX | TEMP | |
| 3 | | LDAB | TEMP-2 | |
| 5 | | RTS | | |

Intervals generated: Length and Number

There are $N_0 + 1$ different interval groups, each group, with the exception of one group, is repeated $N_1$ times. Length of first interval group, $T_1$, is determined by $N_2$ and calculated as

$$T_1 = [2(9 + 5 + 4 + (4 + 4)N_2 + 4 + 5 + 6) + 2 + 4]\ 1.628\ \mu sec \tag{A.1}$$

or

$$T_1 = (1.628)[2(33 + 8N_2) + 6]\ . \tag{A.2}$$

Based upon $T_1$, the second interval group, $T_2$, is obtained as

$$T_2 = 1.628\ [2(33 + 8(N_2 + D)) + 6] \tag{A.3}$$

Where D is the incremental length. Generally, any interval group is given by

$$Ti = 1.628\ [2(33 + 8(N_2 + (i - 1)\ D)) + 6] \tag{A.4}$$

where $i = 1, 2, \ldots, N_0$

Due to the extra software involved in generating the interval groups, an intermediate interval is generated between each group, the length of which is given by

$$\begin{aligned} T_{Intr.} &= [6 + 2 + 4 + 4 + 4 + 2 + 6 + 5 + 3 + (4 + 4)(10) + \\ &\qquad 4 + 6]\ 1.628 \\ &= 205\ \mu sec\ . \end{aligned} \tag{A.5}$$

There are exactly $N_0$ - 1 intermediate intervals all having the same length. The generated signal is depicted in Figure A.8.1.

Intervals produced by the long delay routine have length of

$$T = (78 + 20B + 16BX)\ 1.628\ \mu sec\ . \qquad (A.6)$$

where B = contents of acc. B

and X = contents of Ind. Reg.

Example A.8.1 consider the generation of a signal with 3 different interval groups each interval group is repeated twice. Initial delay $N_2$ = 1 and incremental length D = 1.

There are $N_0$ = 3 different groups, each group repeated $N_1$ = 2 times. Length of $T_1$ is given as 143.5 μsec, $T_2$ as 169.5 μsec and $T_3$ as 195.6 μsec, as given by Equation A.4. There will be $N_0$ - 1 = 2 intermediate intervals; each of length 205 μsec as given by $T_{Intr}$. Drawn in Figure A.8.1 is a sketch of the generated signal.

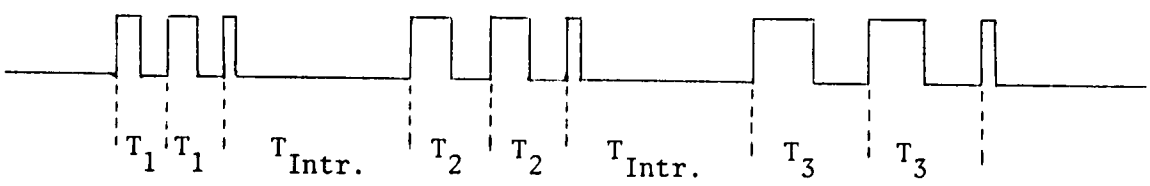



Figure A.8.1. Generated Random Signal of Example A.8.1.

# APPENDIX B

## Interface Hardware

### B.1 Frequency generation

A fundamental 10 MHz frequency was produced by powering up a 10 MHz crystal as illustrated in Figure B.1. Slicing this fundamental frequency to lower fundamentals, basically to 1 MHz, 100 kHz, 10 kHz, and 1 kHz, was achieved through cascading four decade counters as shown in Figure B.1. Applying 10 MHz at input A of the first decade counter while connecting the B input to the $Q_A$ pin, produced a frequency divided by 10, or 1 MHz, at the output $Q_D$. Generally, feeding $Q_D$ of previous counters to input A of next counter, a frequency divided by 10 is obtained at its $Q_D$.

### B.2 Binary counter

Two 4-bit binary counters, Figure B.1, were cascaded to produce an 8-bit counter, the outputs of which were connected to the Input MPU's PIA. The chosen clock frequency was applied to input A of the first counter, while its B input was tied to output $Q_A$ for binary counting. Output $Q_D$ of the first counter was fed into input A of next counter to extend the counting sequence to 8-bits.

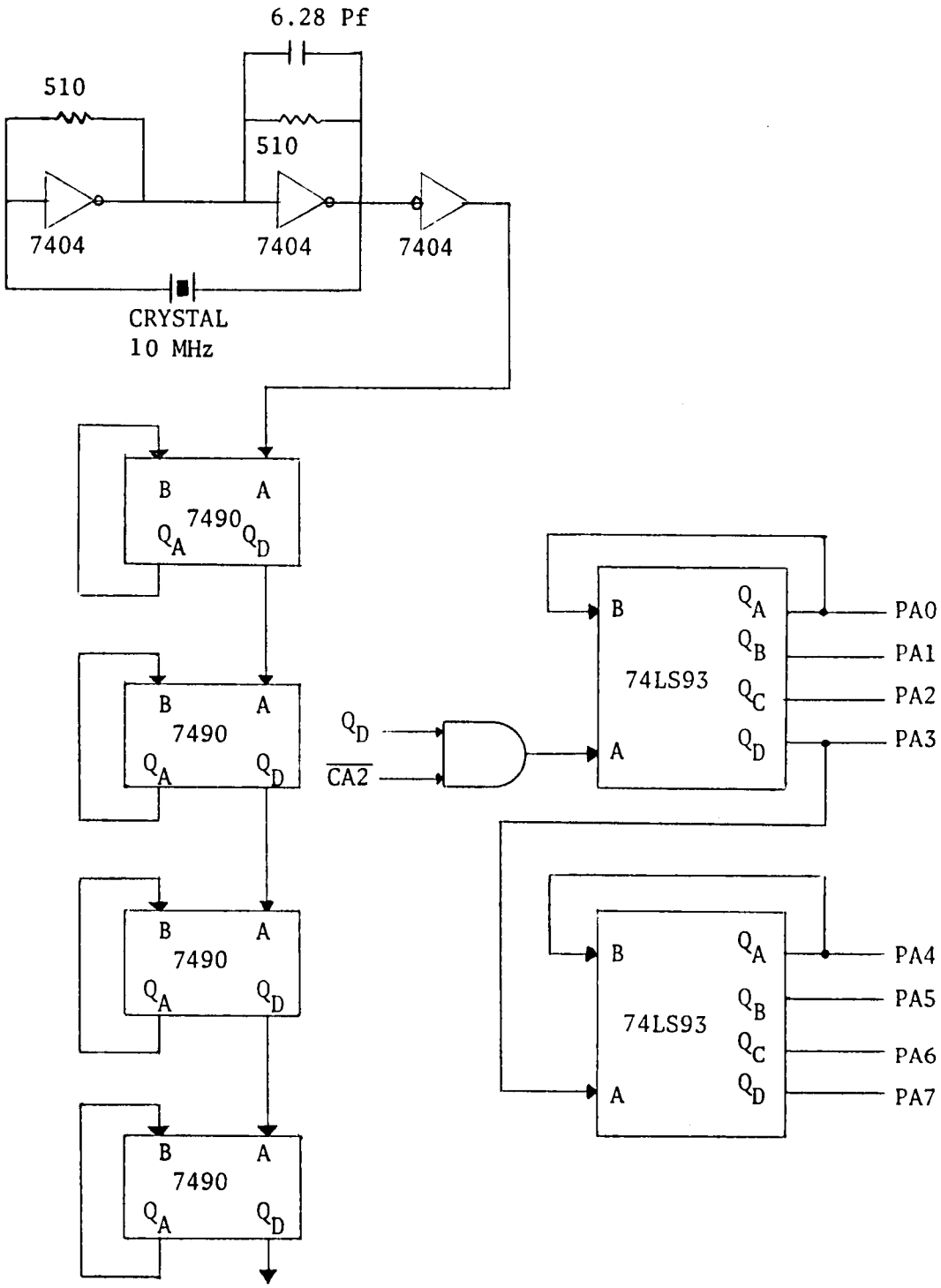



Figure B.1. Frequency Generation and Binary Counter Interface.

# APPENDIX C

## Experimental Data

The intervals given in Tables C-1, C-2, and C-3 were generated by the pseudo-random signal generator program given in Appendix A.8 by setting $N_0 = 5$, $N_1 = 10$, $N_2 = 1$, and $D = 1$.

Table C-1 constructed an interval histogram utilizing a 1 MHz counter clock. In theory interval $T_1$, for instance, will produce 143 ± 1 counts since number of counts = $(T)(f_{CLK})$. This count corresponds to address 018E, 018F, or 0190. Results in the right columns showed that most of the counts were accumulated in 018D which is within -1 count from 018E.

Similarly, Table C-2 was constructed using a lower clock frequency, 10 kHz. The process is repeated with a 1 kHz clock.

Different set of intervals was generated to test the validity of the interaction between the Input MPU and the Display MPU. The intervals given in Table C-4 were produced by setting $N_0 = 9$, $N_1 = 10$, $N_2 = 5$, and $D = 4$ in the pseudo-random signal generator of Appendix A.8. Expected behavior of the histogram, Table C-4, was checked experimentally and results are shown in Table C-5. Address locations 0114 and 0115 correspond to the intermediate intervals discussed in Appendix A.8, the number of which is $N_0 - 1$, all having the same length. All counts were proved to be within ±1 count, except at address 011A where a deviation of + 2 counts was obtained in measuring $T_1 = 248.5$ μsec.

First run shows the successful transfer of data from the Input MPU to the Display MPU. When signal was generated twice to the Input MPU, while the Display MPU is displaying, accumulation of new data was successful as the right column of Table C-5 shows.

Table C-1. Expected and Experimental Histogram Using $f_{CLK}$ = 1 kHz

| | | Theory | | | Experiment | |
|---|---|---|---|---|---|---|
| Period | Length(μs) | Expected Count | Corresponding Address | Decimal Content | Result Address | Decimal Content |
| $T_1$ | 143.5 | 0-1 | 0100-0101 | 10 | 0100<br>0101 | 15<br>39 |
| $T_2$ | 169.5 | 0-1 | 0100-0101 | 10 | | |
| $T_3$ | 195.6 | 0-1 | 0100-0101 | 10 | | |
| $T_4$ | 221.6 | 0-1 | 0100-0101 | 10 | | |
| $T_5$ | 247.7 | 0-1 | 0100-0101 | 10 | | |
| $T_{Intr}$ | 205 | 0-1 | 0100-0101 | 4 | | |
| Total No. of Intervals | | | | 54 | | 54 |

Table C-2. Expected and Experimental Histogram Using $f_{CLK}$ = 10 kHz

| | | Theory | | | Experiment | |
|---|---|---|---|---|---|---|
| Period | Length(μs) | Expected Count | Corresponding Address | Decimal Content | Result Address | Decimal Content |
| $T_1$ | 143.5 | 1-2 | 0101-0102 | 10 | 0101<br>0102 | 01<br>09 |
| $T_2$ | 169.5 | 1-2 | 0101-0102 | 10 | 0102 | 10 |
| $T_3$ | 195.6 | 1-2 | 0101-0102 | 10 | 0102<br>0103 | 09<br>01 |
| $T_4$ | 221.6 | 2-3 | 0102-0103 | 10 | 0102<br>0103 | 04<br>06 |
| $T_5$ | 247.7 | 2-3 | 0102-0103 | 10 | 0102<br>0103 | 02<br>08 |
| $T_{Intr}$ | 205 | 2-3 | 0102-0103 | 4 | 0102<br>0103 | 02<br>02 |
| Total No. of Intervals | | | | 54 | | 54 |

Table C-3. Expected and Experimental Histogram Using $f_{CLK}$ = 1 MHz

| Period | Length (μs) | Theory | | | Experiment | |
|---|---|---|---|---|---|---|
| | | Expected Count | Corresponding Address | Decimal Content | Result Address | Decimal Content |
| $T_1$ | 143.5 | 143-144 | 018F-0190 | 10 | 018C<br>018D | 01<br>09 |
| $T_2$ | 169.5 | 169-170 | 01A9-01AA | 10 | 01A7<br>01A8 | 08<br>02 |
| $T_3$ | 195.6 | 196-197 | 01C3-01C4 | 10 | 01C1<br>01C2 | 08<br>02 |
| $T_4$ | 221.6 | 221-222 | 01DA-01DB | 10 | 01DB | 10 |
| $T_5$ | 247.7 | 247-248 | 01F7-01F8 | 10 | 01F5<br>01F6 | 06<br>04 |
| $T_{Intr.}$ | 205 | 205 | 01CD | 4 | 01CA<br>01CB | 02<br>02 |
| Total No. of Intervals | | | | 54 | | 54 |

Table C-4. Expected Histogram Using $f_{CLK}$ = 100 kHz

| Period | Length(μs) | Expected Count | Corresponding Address | Input MPU Decimal Content | Display MPU Decimal Content |
|---|---|---|---|---|---|
| $T_1$ | 248.5 | 24-25 | 0118-0119 | 10 | 10 |
| $T_2$ | 352.5 | 35-36 | 0123-0124 | 10 | 10 |
| $T_3$ | 456.5 | 45-46 | 0120-012E | 10 | 10 |
| $T_4$ | 560.6 | 56 | 0138 | 10 | 10 |
| $T_5$ | 664.6 | 66-67 | 0142-0143 | 10 | 10 |
| $T_6$ | 768.6 | 76-77 | 014B-014C | 10 | 10 |
| $T_7$ | 872.6 | 87-88 | 0157-0158 | 10 | 10 |
| $T_8$ | 976.6 | 97-98 | 0161-0162 | 10 | 10 |
| $T_9$ | 1080.6 | 108 | 016C | 10 | 10 |
| $T_{Intr}$ | 205.0 | 205 | 0114 | 8 | 8 |

Table C-5. Experimental Histogram for Intervals of Table C-4

| | Run #1: Signal generated once | | | Run #2: Signal generated twice | |
|---|---|---|---|---|---|
| | Decimal Content | | | Decimal Content | |
| Address | Input MPU | | Display MPU | Input MPU | Display MPU |
| 0114 | 06 | = 8 | 06 | 07 | 13 |
| 0115 | 02 | | 02 | 01 | 03 |
| 0118 | 02 | | 02 | 03 | 05 |
| 0119 | 07 | = 10 | 07 | 06 | 13 |
| 011A | 01 | | 01 | 01 | 02 |
| 0123 | 08 | = 10 | 08 | 10 | 18 |
| 0124 | 02 | | 02 | 00 | 02 |
| 012D | 04 | = 10 | 04 | 04 | 08 |
| 012E | 06 | | 06 | 06 | 12 |
| 0138 | 10 | = 10 | 10 | 10 | 20 |
| 0142 | 08 | = 10 | 08 | 08 | 16 |
| 0143 | 02 | | 02 | 02 | 04 |
| 014C | 04 | = 10 | 04 | 02 | 06 |
| 014D | 06 | | 06 | 08 | 14 |
| 0157 | 08 | = 10 | 08 | 08 | 16 |
| 0158 | 02 | | 02 | 02 | 04 |
| 0161 | 04 | = 10 | 04 | 03 | 07 |
| 0162 | 06 | | 06 | 07 | 13 |
| 016B | 01 | = 10 | 01 | 01 | 02 |
| 016C | 09 | | 09 | 09 | 18 |

# VITA

Nabil Ibrahim Tmeizeh was born in AL-Khalil (Hebron), Jordan, on September 18, 1953. He graduated from Hussien Ben Ali High School of AL-Khalil in June 1972. After an intensive English language course at the Haikazian College, Beirut, Lebanon, he joined the American University of Beirut (AUB) in February 1974. He transferred to the University of New Hampshire (UNH), Durham, New Hampshire, USA, in June 1976. He subsequently received a Bachelor of Science degree in Electrical Engineering on May 20, 1979. He entered The University of Tennessee, Knoxville, Tennessee, in September 1979 where he received a Master of Science degree in Electrical Engineering in June 1981.

The author is married to Sumaia Mustafa of Idna, Jordan. They have two sons, Aladdin and Ibrahim.