To the Graduate Council:

I am submitting herewith a thesis written by George R. Gilbert entitled "Development of a Procedure and Design of an Instrument to Correct for Pulse Pile-Up Distortion." I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

D. W. Bouldin

D. W. Bouldin, Major Professor

We have read this thesis
and recommend its acceptance:

J M Hoog

R. W. Rochelle

Accepted for the Council:

L Evans Roth

Vice Chancellor
Graduate Studies and Research

# DEVELOPMENT OF A PROCEDURE AND DESIGN OF AN INSTRUMENT TO CORRECT FOR PULSE PILE-UP DISTORTION

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

George R. Gilbert

June 1981

3051830

## ABSTRACT

Several areas of science and medicine involve the collection of physical measurements in pulse form where the information is carried by the pulse height. If the pulse rate is high enough, then two or more randomly-spaced pulses may occur closely spaced. If this spacing is sufficiently close, the skirts of the pulse waveshapes will overlap, producing false pulse heights. Current pulse height instruments handle this problem by rejecting the closely-spaced pulses and then increasing experiment time to compensate. It would be desirable, instead, to be able to instrumentally analyze the pulses and correct for the error caused by pile-up. In this thesis, a mathematical development of the nature of pile-up distortion is performed which shows that the pile-up effects for a set of N pulses can be described with a set of N simultaneous linear equations. The parameters and coefficients of the equation set are generated from the false pulse heights and the pulse spacing. The algorithm was tested in a programmable calculator, and then an instrument was designed that could perform on-line pulse pile-up correction.

TABLE OF CONTENTS


| CHAPTER | | PAGE |
|---|---|---|
| I. | INTRODUCTION | 1 |
| II. | ALGORITHM DEVELOPMENT | 6 |
| III. | OVERVIEW OF THE REQUIREMENTS FOR AN INSTRUMENTAL IMPLEMENTATION OF THE SEPARATION ALGORITHM | 22 |
| IV. | A SIMULATION OF THE ALGORITHM IMPLEMENTATION | 32 |
| V. | DESIGN OF THE PULSE PILE-UP CORRECTOR, PHASE 1 | 43 |
| VI. | DESIGN OF THE PULSE PILE-UP CORRECTOR, PHASE 2 | 71 |
| VII. | SUMMARY | 112 |
| LIST OF REFERENCES | | 116 |
| APPENDICES | | 118 |
| VITA | | 134 |

LIST OF TABLES


| TABLE | | PAGE |
|---|---|---|
| IV-1. | Test Point Set Results . . . . . | 40 |
| VI-1. | TMS 9901 Programmable Systems Interface Pin Assignments . | 88 |
| VI-2. | RAM Memory, Designated Storage Assignments . . . . . | 97 |
| VI-3. | Workspace Registers, Designated Uses . . . . . | 98 |
| VI-4. | TMS 990/100M Pulse Pile-Up Corrector Program Listing . . . | 99 |
| A-1. | TI-59 Simulation Program Instruction Sheet . . . . . | 122 |
| A-2. | TI-59 Simulation Program Listing . . . . . | 123 |

LIST OF FIGURES


| FIGURE | | PAGE |
|---|---|---|
| I-1. | Pulse Height Analyzer System | 3 |
| I-2. | Pulse Height Analyzer System with Pile-Up Rejection | 3 |
| I-3. | A Pulse Height Analyzer System with Proposed Pulse Pile-Up Corrector | 5 |
| II-1. | Typical Pulse Waveshapes | 7 |
| II-2. | A Normalized Pulse Waveshape | 12 |
| II-3. | Two Non-Piled-Up Pulses | 12 |
| II-4. | Two Piled-Up Pulses | 14 |
| II-5. | Multiple Overlapping Pulses | 15 |
| II-6. | A Series of n+k+1 Multiple Overlapping Pulses | 16 |
| II-7. | Two Piled-Up Pulses with the Real Peak $Y_1$ and the False Peak $Y_1'$ Not Coincident | 20 |
| III-1. | Block Diagram; Pulse Pile-Up Corrector and Pulse Height Analyzer System | 24 |
| III-2. | Five Mutually Interfering Pulses | 28 |
| III-3. | Spotlight Procedure Error Analysis | 30 |
| IV-1. | Simulation Program; Simplified Flow Chart | 34 |
| IV-2. | Sample Normalized Pulse Waveshape, $f(\Delta t)$ | 38 |
| IV-3. | A Test Pulse Set | 39 |
| V-1. | Pulse Pile-Up Corrector; Design Decision Summary | 44 |
| V-2. | Pile-Up Corrector; Block Diagram | 49 |
| V-3. | Pulse Pile-Up Corrector; Input Data Buffer | 51 |
| V-4. | Pulse Data Structure with Pulse Bursts Evenly Spaced | 53 |
| V-5. | Pulse Data with One Closely-Spaced Group of Pulse Bursts | 53 |
| V-6. | Pulse Pile-Up Corrector Memory Map | 56 |

| FIGURE | | PAGE |
|---|---|---|
| V-7. | Pulse Pile-Up Corrector Flow Chart; Executive | 59 |
| V-8. | Pulse Pile-Up Corrector Flow Chart; Functional Subprograms | 61 |
| VI-1. | The Texas Instruments Incorporated TM 990/100M Micro-Computer Module | 75 |
| VI-2. | Input Buffer Circuit | 77 |
| VI-3. | Input Buffer Address Decoder | 79 |
| VI-4. | Timing Diagram for Input Buffer Read Cycle | 80 |
| VI-5. | Output Buffer Circuit | 81 |
| VI-6. | Timing Diagram for Output Buffer Write Cycle, No Wait Status | 83 |
| VI-7. | Timing Diagram for Output Buffer Write Cycle, One Wait State | 84 |
| VI-8. | Data FIFO Circuit | 85 |
| VI-9. | Time FIFO Circuit | 86 |
| VI-10. | EXECUTIVE/COMMAND Program Flow Chart | 94 |
| VI-11. | TM 990/100M Pulse Pile-Up Corrector Memory Map | 96 |

# CHAPTER I

# INTRODUCTION

## A. STATEMENT OF THE PROBLEM

### The Problem, Pulse Pile-Up Distortion

In certain areas of science, notably nuclear and astrophysics, testing or measurement of a phenomenum generates data in discrete analog pulse form. A single pulse typically represents a single physical event such as a quantum change in energy state and consequent release of energy. Generally these pulse events are X-ray, gamma ray, or charged particle radiation. A class of instrumentation exists, and is commercially available, for measuring and analyzing this type of data. This study is the development of an addition to this instrumentation; an instrument to be used to reduce the source of error known as "Pulse Pile-Up Distortion."

Information may be carried by a pulse by (a) its existence, (b) its existence in a particular time relationship to other events, or (c) its amplitude along with "a" or "b." When the pulse amplitude is important, the instrumentation must be linear with respect to amplitude. That is, any modification to the pulse amplitude must be a multiplication by a known gain or addition of a known constant (i.e., bias). Even a system consisting of properly designed amplifiers will still introduce a troublesome source of amplitude error. Since pulses occur randomly, some may occur in tightly grouped bursts. Normally the pulses are sufficiently separated in time that they do

not interact. However, pulses occasionally occur so close together that their pulse widths overlap. They then become mutally interactive, modifying each other's amplitude. This effect is known as Pulse Pile-Up Distortion.

Typical Pulse Height Analyzer System

Before beginning a discussion of a Pulse Pile-Up Corrector, it will be useful to examine a typical instrument system set up for pulse height analysis. Figure I-1 (Ortec, 1976) shows such an instrumentation system. A process generates pulse data which is read by a detector and converted to an electrical signal. Since the signal is usually of a very low level, some pre-amplification is necessary. Pre-amplification is usually followed by a stage of amplification and pulse wave shaping. The signal is then typically routed through an analog to digital converter (ADC) to a single channel analyzer (SCA) or a multichannel analyzer (MCA). The SCA is set to look at a narrow band of amplitude and count the pulses having that amplitude. The MCA is in effect a series of SCAs each set to a different amplitude band over a wide spectra of amplitudes. Note that the time at which the pulse occurs is not preserved--this point will become important.

Figure I-2 (Ortec, 1976) shows an instrumentation system similar to that of Figure I-1 but with a Pulse Pile-Up Rejector added. This is a commonly available instrument which is used to minimize the effect of pile-up error. It monitors the time interval ($\Delta T$) between pulses, and when it encounters a $\Delta T$ less than a predetermined amount, it assumes the two pulses are interfering and discards them both. This

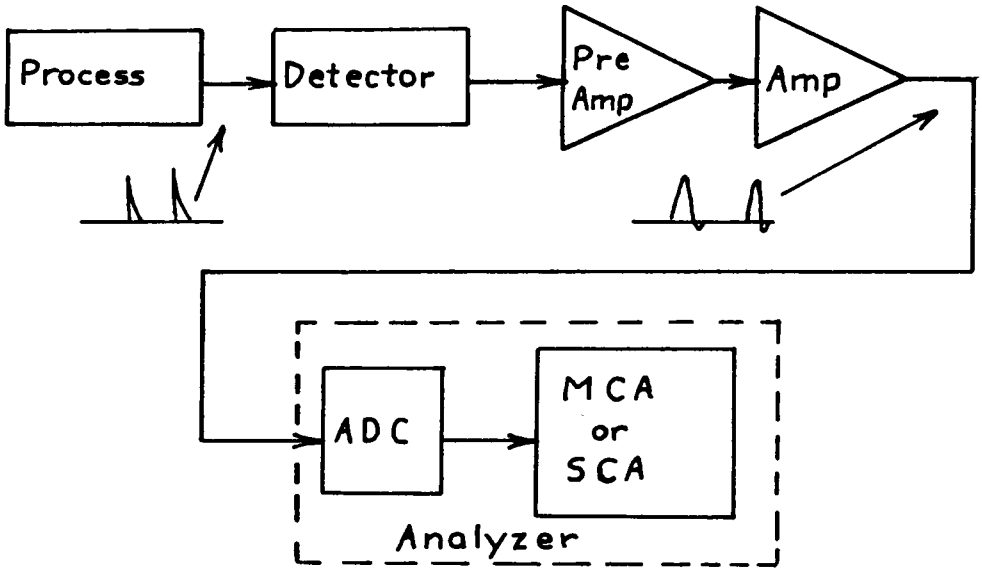



Figure I-1. Pulse Height Analyzer System.

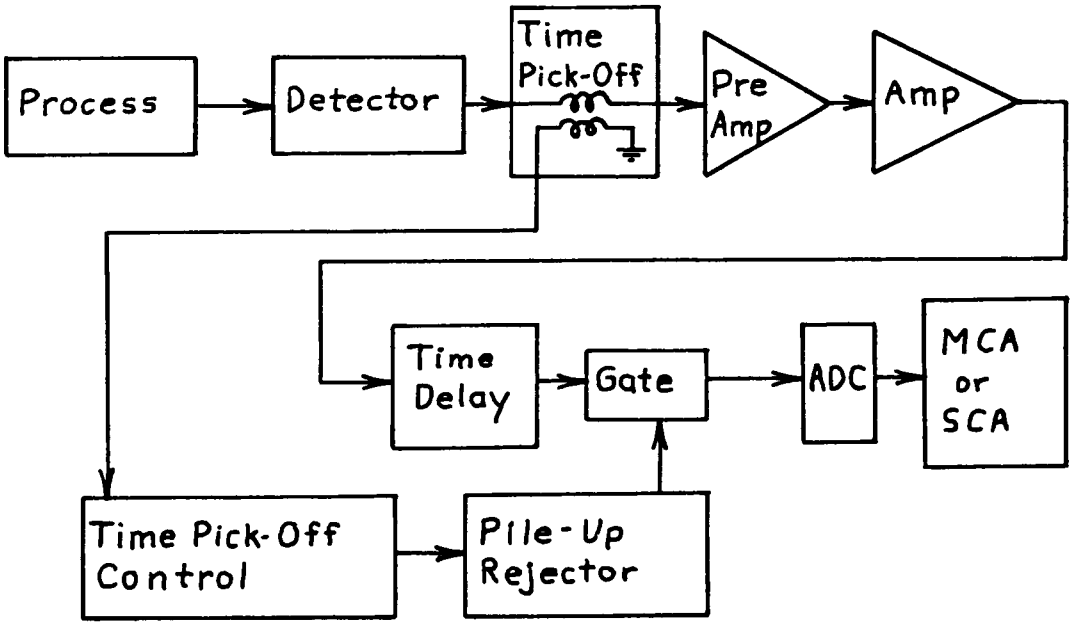



Figure I-2. Pulse Height Analyzer System with Pile-Up Rejection.

type system rejects data; it does not correct for the interference and save the data.

The Concept for a Pulse Pile-Up Correction Instrument

The concept for a Pulse Pile-Up Corrector is as shown in Figure I-3. As above, a time pick-off is used to determine potential pile-up. When a potential pile-up is detected ($\Delta T$ below a preset minimum), then the time pick-off control operates a routing switch to route the interfering pulses to the Pile-Up Corrector. The Corrector then corrects the pulses and presents the corrected data to the analyzer. Noninterfering pulses are routed directly to the analyzer without being diverted to the Corrector.

This study will develop a possible design for a Pulse Pile-Up Corrector as shown in Figure I-3. The Corrector is to be implemented as a digital instrument using microprocessor technology. The work was done in three major parts:

1. Algorithm development and testing (Chapters II, III, and IV).
2. A generalized machine approach to a microprocessor implementation (Chapter V).
3. Application of the design to a selected microprocessor family (Chapter VI).

The analog portion of the instrument (time pick-off control, routing switch, etc.) is not a part of this study except were necessary to specify interfaces. This is because these parts are applications of or simple modifications to currently available instruments and because these items may be specific to a particular instrument system being used.

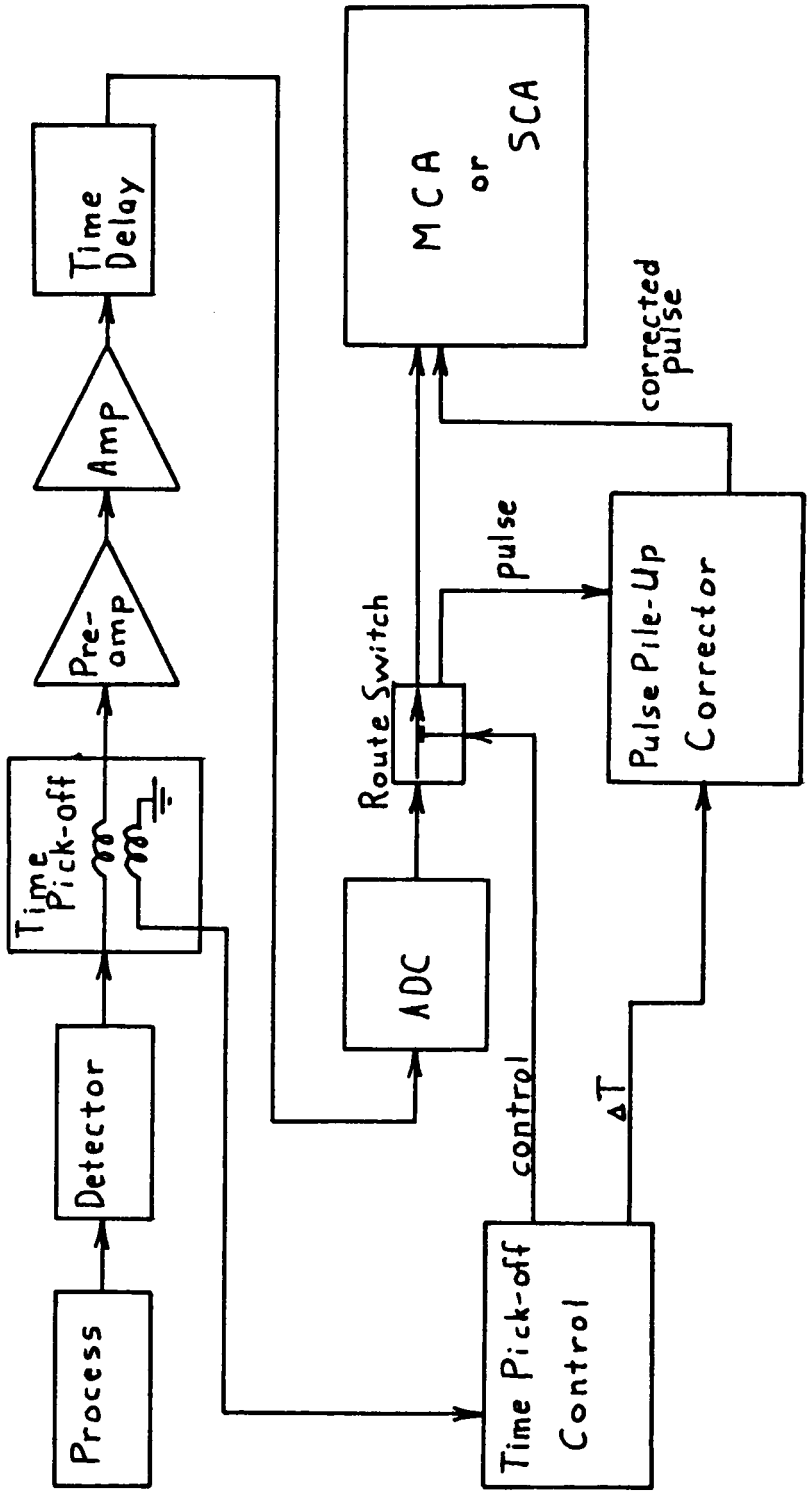



Figure I-3. A Pulse Height Analyzer System with Proposed Pulse Pile-Up Corrector.

# CHAPTER II

## ALGORITHM DEVELOPMENT

### A. THE NATURE OF THE INTERFERENCE BETWEEN CLOSELY SPACED PULSES

Before a design for a Pile-Up Corrector can be attempted it is necessary to examine the nature of the interference between piled-up pulses. Then with the interference understood, a procedure for correcting for the interference can be developed.

A typical event pulse (the pulse generated by the test or process) is a very sharp narrow width pulse (Fairstein, July 1965, p. 37 and Pierce, 1977, p. 40), possibly less than 1 nanosecond. The detector converts the event pulse into an electrical signal which closely follows the event pulse in form. The detected pulse is then amplified to a useful level. Some pulse shaping is also done in the amplifier; this pulse shaping is done to improve resolving time, signal-to-noise ratio, and the precision and linearity of the pulse-height-analyzer response (Fairstein, 1965, p. 83). Figures II-1-a, b, and c show respectively typical event, detected, and amplified pulses.

To examine the pulse conversion behavior let the pulse be represented as $p(t)$, $P(s)$ in time and frequency domains respectively; let the detector transfer function be $D(s)$, and let the amplifier transfer function be $A(s)$. Then the output of the amplifier is clearly:

$$Y(s) = A(s)\, D(s)\, P(s) \qquad \text{II-1}$$

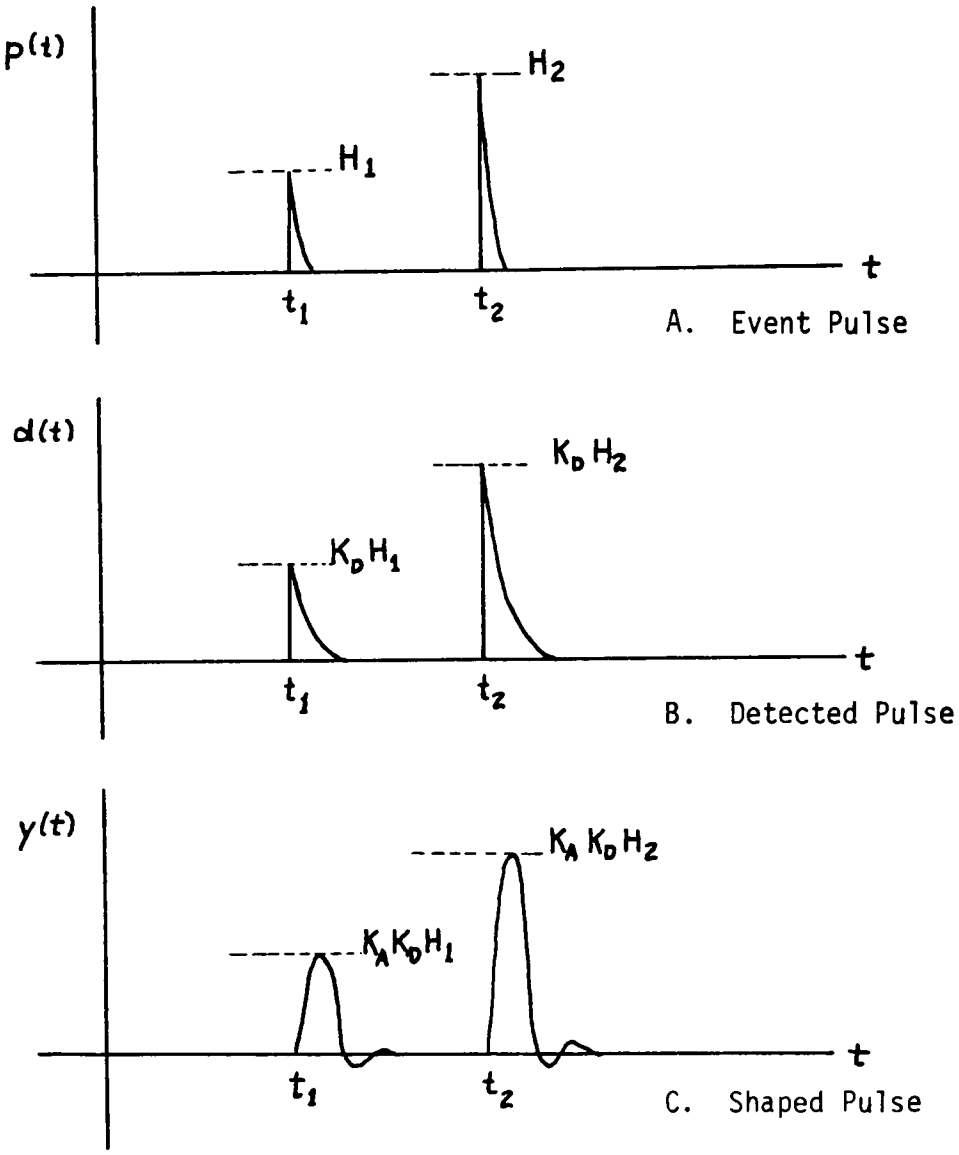



Figure II-1. Typical Pulse Waveshapes.

The output function, $y(t) = \mathcal{L}^{-1}\{Y(s)\}$, will be a pulse of some finite width because the detector and amplifiers are designed to ensure this response. As an example assume the event pulse is an impulse function;

$$p(t) = H\ \delta(t) \qquad \text{II-2}$$

assume the detector transfer function is:

$$D(s) = \left[\frac{K_D}{s + \frac{1}{\tau_D}}\right] \qquad \text{II-3}$$

and assume the amplifier transfer function is:

$$A(s) = \frac{K_A}{(s + \frac{1}{\tau_A})^2 + w_A^2} \qquad \text{II-4}$$

(A study of the papers on pulse shaping and nuclear pulse amplifiers by Fairstein (1965) and Nowlin (N.D.) indicates that the above assumptions are reasonable.) Applying Equation II-1 and taking the inverse Laplace transform gives the time response:

$$y(t)=K_A K_D H\left\{\left[(\frac{1}{\tau_A} - \frac{1}{\tau_D})^2 + w_A^2\right]\dot{e}^{-t/\tau_D} + \left[\cos w_A t + \frac{1}{W_A \tau_D}\sin w_a t\right]\dot{e}^{-t/\tau_A}\right\} \qquad \text{II-5}$$

The decaying exponential terms in II-5 cause the response, $y(t)$, to be a pulse shape of finite effective width.

At this point it has been shown that a single event pulse can be conditioned into some specific form of output pulse by the detector and amplifier. Attention is now turned to the response to two event

pulses. Consider an event pulse $p_1(t)$ occurring at $t_1$, and $p_2(t)$ occurring at $t_2$ as shown in Figure II-1a. These pulses can be written as:

$$p_1(t) = H_1\ \delta(t-t_1) \qquad \text{II-6}$$

$$p_2(t) = H_2\ \delta(t-t_2) \qquad \text{II-7}$$

where the event pulse is assumed to be an impulse function whose Laplace transform is:

$$\mathcal{L}\{\delta(t-t_a)\} = 1 \cdot e^{-t_a s}$$

Then by using the shifting theorem to take the Laplace transform of II-6 and II-7 and then substituting the result in II-1 the system response is:

$$Y(s) = A(s)D(s)H_1e^{-t_1s} + A(s)D(s)H_2e^{-t_2s} \qquad \text{II-9}$$

For simplification, let:

$$F(s) = A(s)D(s) \qquad \text{II-10}$$

then Equation II-9 becomes:

$$Y(s) = F(s)\ H_1e^{-t_1s} + F(s)\ H_2e^{-t_2s} \qquad \text{II-11}$$

This change then makes it easy to see that the time response is:

$$y(t) = H_1\ f(t-t_1)u(t-t_1) + H_2\ f(t-t_2)u(t-t_2) \qquad \text{II-12}$$

Equation II-12 is the key to development of an algorithm for the correction of pile-up error. First note that the development of II-12 placed no restriction on the time between the two pulses, in fact II-12 holds even if $t_1=t_2$. Second note that the system response determines the output pulse shape (assuming event pulses are of the same shape). Third note that the event pulse amplitude (or height) is preserved. Also clearly apparent from Equation II-12 is that $y(t)$ is a pulse train with a pulse at $t_1$ and a pulse at $t_2$ just like the event pulse train. If $f(t-t_1)u(t-t_1) = 0$ everywhere that $f(t-t_2)u(t-t_2) \neq 0$ and vice versa, then the output pulse heights at $t_1$ and $t_2$ are respectively functions of only $p_1(t)$ and $p_2(t)$. If, however, $f(t-t_2)u(t-t_2) \neq 0$ at $t_1$, then the output pulse height at $t_1$ is a function of both $p_1(t)$ and $p_2(t)$; pulse pile-up has occurred. The same is true at $t_2$ if $f(t-t_1)u(t-t_1) \neq 0$ at $t_2$. It can be seen from the development and analysis of Equation II-12 that pulse pile-up distortion is a simple linear addition process.

There may be some concern about possible loss of generality in assuming the event pulse is an impulse function. However, this is not seen as a problem. The event pulse is a finite width pulse. If it is not an impulse function it can be modeled by an impulse function and some G(s) whose output would be a narrow band pulse, that is;

$$\mathcal{L}\{\text{event pulse}\} = 1 \cdot G(s)$$

The G(s) could simply become a factor in F(s) of Equation II-10.

Construction of the Pulse Pile-Up Separation Algorithm

With the development and analysis of Equation II-12 showing that pile up is a simple addition process, construction of the separation algorithm can begin. First a change in notation is made. Previously, notation using the unit step function was used; that is the function was represented as in Figure II-1 as $f(t-t_n)u(t-t_n)$. This notation requires that $t_n$ occur just prior the leading edge of the pulse. In the remaining description it is more convenient to have $t_n$ represent the time that the pulse peak occurs as is shown in Figure II-2. To do this the pulse is represented as $Y_n f(t-t_n)$, where $Y_n$ is the magnitude and $f(t-t_n)$ is a normalized pulse waveshape function as shown in Figure II-2. The function $f(t-t_n)$ can be any function as long as:

$$f(t-t_n) = 1 \quad \text{at} \quad t=t_n$$

$$= 0 \quad \text{at} \quad t \gg t_n \quad \text{or} \quad t \ll t_n$$

This development proceeds by considering two pulses as shown in Figure II-3. The equations for the two pulses as they appear at the amplifier output are:

$$y_1(t) = Y_1 f(t-t_1) \qquad \text{II-13}$$

$$y_2(t) = Y_2 f(t-t_2) \qquad \text{II-14}$$

And applying Equation II-12 the total response at the amplifier output is:

$$y(t) = y_1(t) + y_2(t) = Y_1 f(t-t_1) + Y_2 f(t-t_2) \qquad \text{II-15}$$

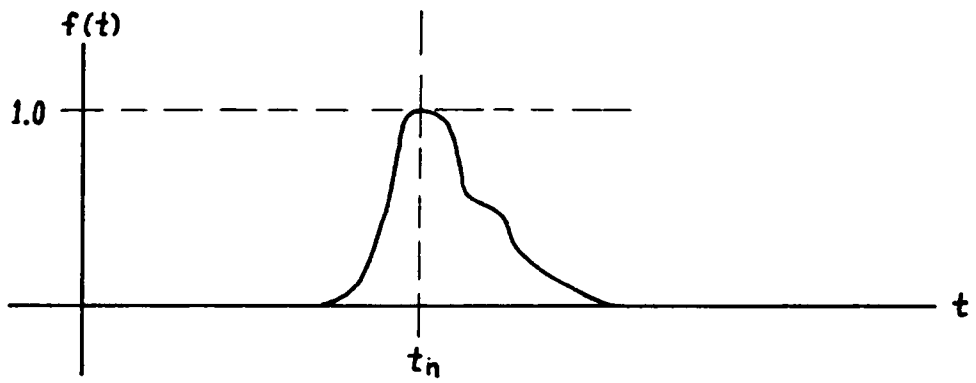



Figure II-2. A Normalized Pulse Waveshape.

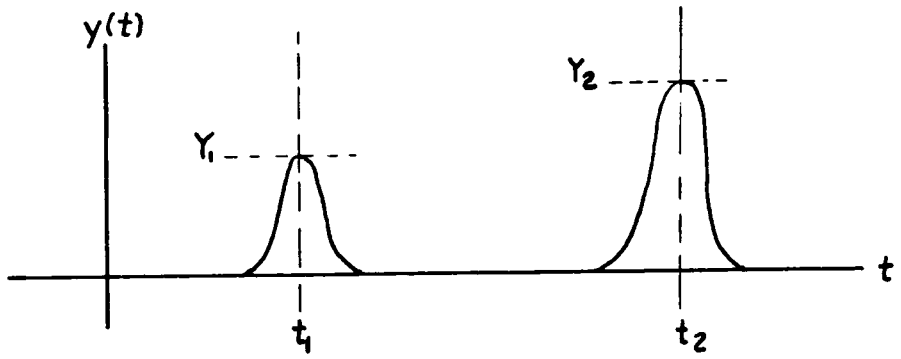



Figure II-3. Two Non-Piled-Up Pulses.

The pulses shown in Figure II-3 are not piled up or interfering; however Figure II-4 shows two pulses that are piled up. The total response for Figure II-4 is still:

$$y(t) = Y_1 f(t-t_1) + Y_2 f(t-t_2) \qquad \text{II-16}$$

In Figure II-4 the total response $y(t)$ is shown as a dashed line. The "false" peaks at $t_1$ and $t_2$ are designated $Y_1'$ and $Y_2'$. These are the values that a pulse height analyzer without pile-up correction would measure. If Equation II-16 is evaluated at $t=t_1$ and at $t=t_2$ the resulting equations are:

$$\text{at } t=t_1,\ y(t_1) = Y_1' = Y_1 f(0) + Y_2 f(t_1-t_2) \qquad \text{II-17}$$

$$\text{at } t=t_2,\ y(t_2) = Y_2' = Y_1 f(t_2-t_1) + Y_2 f(0) \qquad \text{II-18}$$

The pulse waveshape function is fixed by the system and is therefore a known function and it can be evaluated for any $t-t_n$. Thus, the only unknowns are $Y_1$ and $Y_2$. The separation algorithm's function is to find $Y_1$ and $Y_2$.

Thus, it is seen that the separation algorithm is simply the solution of a set of simultaneous linear equations. This analysis can easily be extended to a multiple set of mutually interfering pulses; Figure II-5 and Equations II-19 and II-20 show five mutually piled-up pulses. Finally Figure II-6 and Equation II-21 is a generalization of Figure II-5/Equation II-20. Note that in all cases the principal diagonal of the coefficient matrix consists of f(0) terms and that $f(0)=1$.

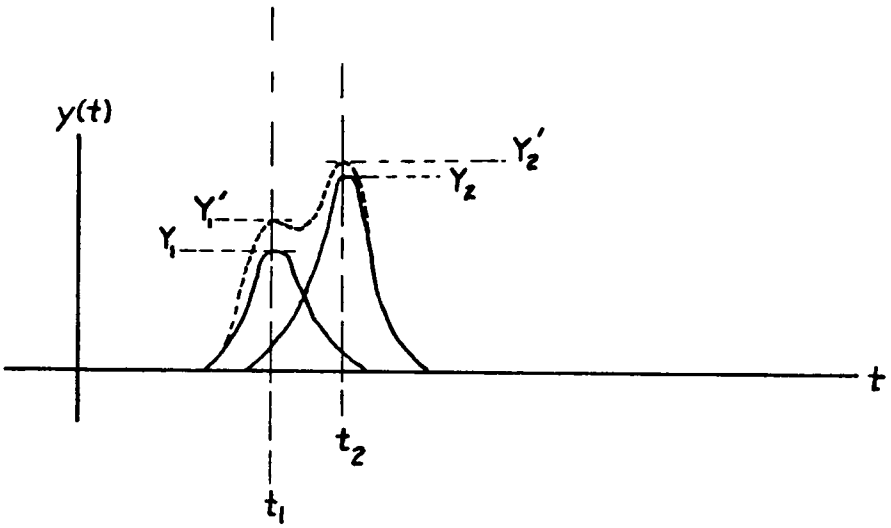



Figure II-4. Two Piled-Up Pulses.

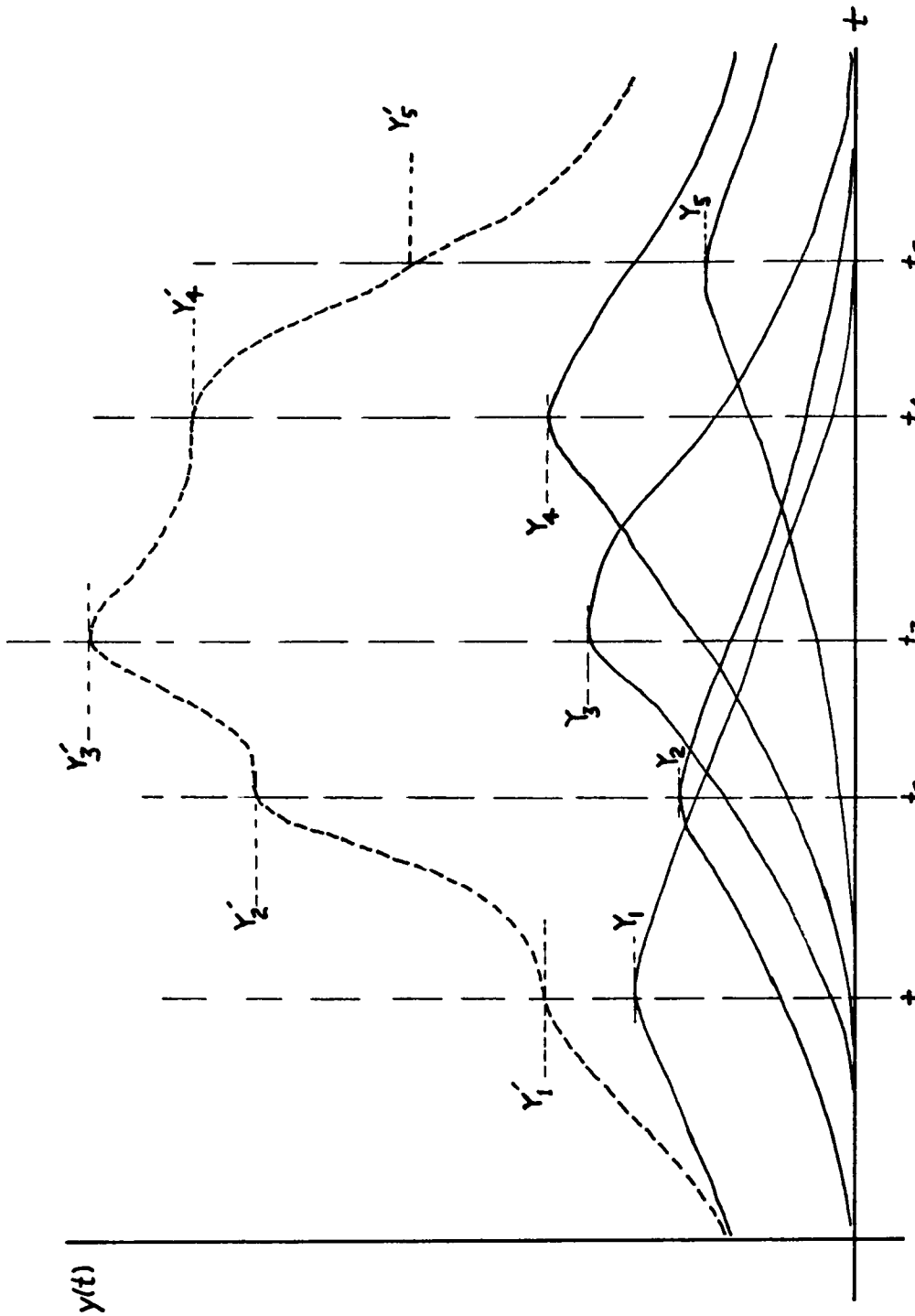



Figure II-5. Multiple Overlapping Pulses.

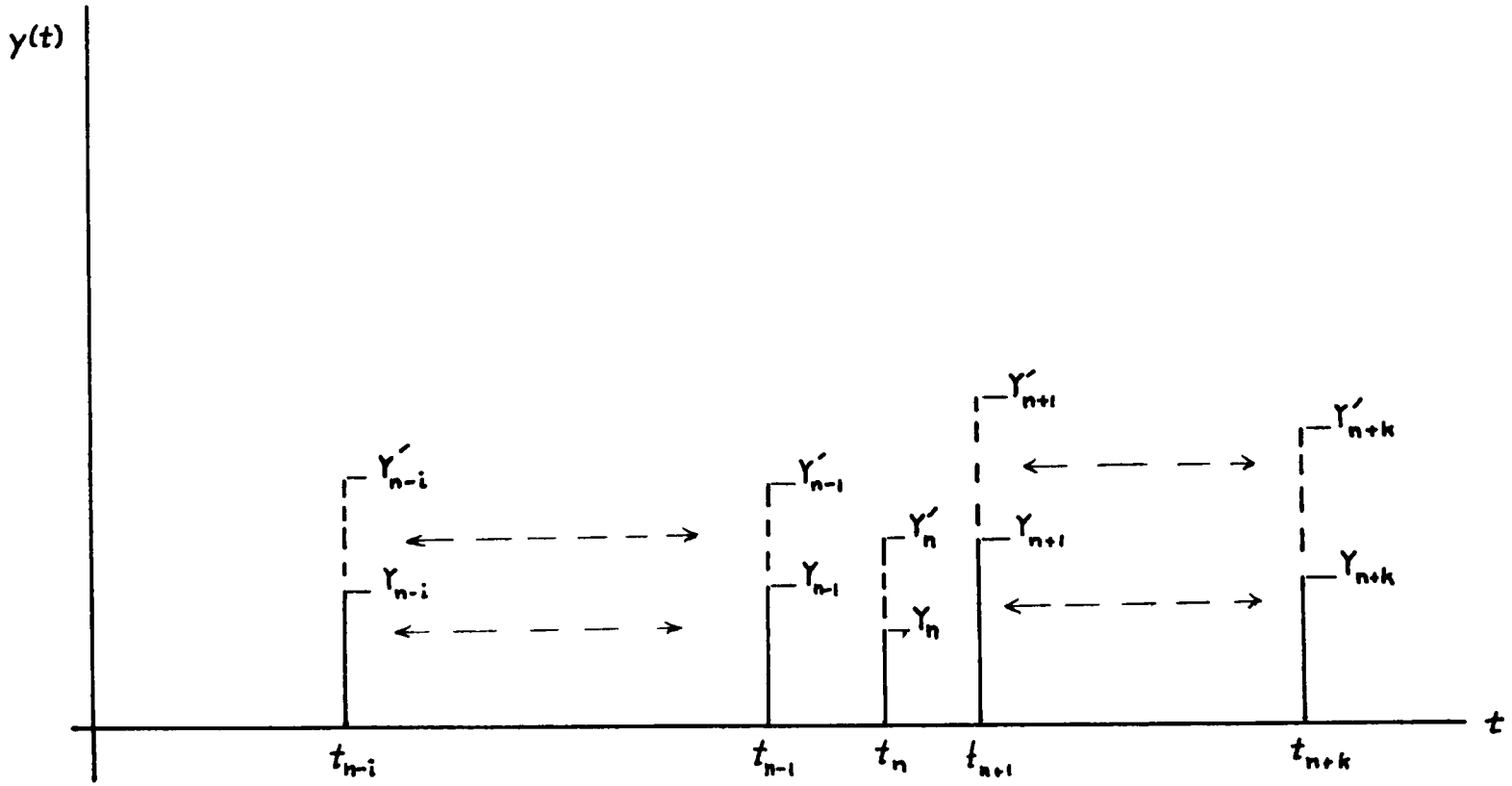



Figure II-6. A Series of n+k+1 Multiple Overlapping Pulses.

From Figure II-5 the equations are:

$$Y_1 f(t_1-t_1)+Y_2 f(t_1-t_2)+Y_3 f(t_1-t_3)+Y_4 f(t_1-t_4)+Y_5 f(t_1-t_5)=Y'_1$$

$$Y_1 f(t_2-t_1)+Y_2 f(t_2-t_2)+Y_3 f(t_2-t_3)+Y_4 f(t_2-t_4)+Y_5 f(t_2-t_5)=Y'_2$$

$$Y_1 f(t_3-t_1)+Y_2 f(t_3-t_2)+Y_3 f(t_3-t_3)+Y_4 f(t_3-t_4)+Y_5 f(t_3-t_5)=Y'_3$$

$$Y_1 f(t_4-t_1)+Y_2 f(t_4-t_2)+Y_3 f(t_4-t_3)+Y_4 f(t_4-t_4)+Y_5 f(t_4-t_5)=Y'_4$$

$$Y_1 f(t_5-t_1)+Y_2 f(t_5-t_2)+Y_3 f(t_5-t_3)+Y_4 f(t_5-t_4)+Y_5 f(t_5-t_5)+Y'_5 \qquad \text{II-19}$$

Equation II-19 can be written, most conveniently, in matrix form as shown below:

$$\begin{vmatrix} f(t_1-t_1) & f(t_1-t_2) & f(t_1-t_3) & f(t_1-t_4) & f(t_1-t_5) \\ f(t_2-t_1) & f(t_2-t_2) & f(t_2-t_3) & f(t_2-t_4) & f(t_2-t_5) \\ f(t_3-t_1) & f(t_3-t_2) & f(t_3-t_3) & f(t_3-t_4) & f(t_3-t_5) \\ f(t_4-t_1) & f(t_4-t_2) & f(t_4-t_3) & f(t_4-t_4) & f(t_4-t_5) \\ f(t_5-t_1) & f(t_5-t_2) & f(t_5-t_3) & f(t_5-t_4) & f(t_5-t_5) \end{vmatrix} \begin{vmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \end{vmatrix} = \begin{vmatrix} Y'_1 \\ Y'_2 \\ Y'_3 \\ Y'_4 \\ Y'_5 \end{vmatrix} \qquad \text{II-20}$$

And following the same pattern for the general case shown in Figure II-6, the equation set II-21 can be written. It is as follows:

$$\begin{vmatrix} f(t_{n-i}-t_{n-i}) & \cdots & f(t_{n-i}-t_{n-1}) & f(t_{n-i}-t_n) & f(t_{n-i}-t_{n+1}) & \cdots & f(t_{n-i}-t_{n+k}) \\ & & f(t_{n-1}-t_{n-1}) & f(t_{n-1}-t_n) & f(t_{n-1}-t_{n+1}) & \cdots & \\ f(t_n-t_{n-i}) & \cdots & f(t_n-t_{n-1}) & f(t_n-t_n) & f(t_n-t_{n+1}) & \cdots & f(t_n-t_{n+k}) \\ & & f(t_{n+1}-t_{n-1}) & f(t_{n+1}-t_n) & f(t_{n+1}-t_{n+1}) & \cdots & \\ f(t_{n+k}-t_{n-i}) & \cdots & f(t_{n+k}-t_{n-1}) & f(t_{n+k}-t_n) & f(t_{n+k}-t_{n+1}) & \cdots & f(t_{n+k}-t_{n+k}) \end{vmatrix} \begin{vmatrix} Y_{n-i} \\ \\ Y_n \\ \\ Y_n \end{vmatrix} = \begin{vmatrix} Y'_{n-i} \\ \\ Y'_n \\ \\ Y'_{n+k} \end{vmatrix} \qquad \text{II-2}$$

Considerations on the Use of the Pulse Pile-Up Separation Algorithm

The separation algorithm is exact and can be safely applied to any piled-up function (as described here) provided (a) the time, $t_n$, at which the real peaks, $Y_n$, occur are known, (b) the value of the total response function, $y(t_n)$, at each time $t_n$ is known, and (c) $f(t-t_n)$ is a known function. Note specifically that item "b" above does not say the false peak $Y'_n$ is $y(t_n)$. Pulse shaping functions may exist that generate false peaks at some time other than $t_n$.

If the $Y'_n$ is not coincident with the $Y_n$ it may be possible to handle the problem by measuring $t_n$ and then evaluating $y(t_n)$ at exactly $t_n$. The time $t_n$ should be available with good accuracy. Using semiconductor detectors and inductively coupled time pick-off, the pulse occurrence time resolution can be in the subnanosecond range (Fairstein, November 1965, p. 52). This method's difficulty would lie

in design of the circuits to evaluate $y(t_n)$ at $t_n$. A better solution, for a problem where pile-up causes significant error, would be to change to a pulse shaping function that keeps the $Y_n'$ and the $Y_n$ coincident.

It is useful to consider the pulse shape characteristics that could cause the real and false peaks to be noncoincident. Figure II-7 shows such a function. The false peak, $Y_1'$ occurs not at $t_1$ but at $t_f$. For this to happen the magnitude of change in $Y_2 f(t-t_2)$ between $t_1$ and $t_f$ must be greater than the magnitude of the change in $Y_1 f(t-t_1)$ between $t_1$ and $t_f$.

Most commonly encountered functions will be some sort of exponential function. If $f(t-t_1)$ and $f(t-t_2)$ are exponential in nature then it could be expected that:

$$f(t-t_1) \propto \exp(\frac{t-t_1}{\tau_a}) \qquad \text{for } t>t_1 \qquad \text{II-22}$$

and

$$f(t-t_2) \propto \exp(\frac{t-t_1}{\tau_b}) \qquad \text{for } t<t_2 \qquad \text{II-23}$$

The rate of change of the functions are then:

$$\frac{d}{dt}f(t-t_1) \propto \frac{1}{\tau_a} \exp(\frac{t-t_1}{\tau_a}) \qquad t>t_1 \qquad \text{II-24}$$

$$\frac{d}{df}f(t-t_2) \propto \frac{1}{\tau_b} \exp(\frac{t-t_2}{\tau_b}) \qquad t<t_2 \qquad \text{II-25}$$

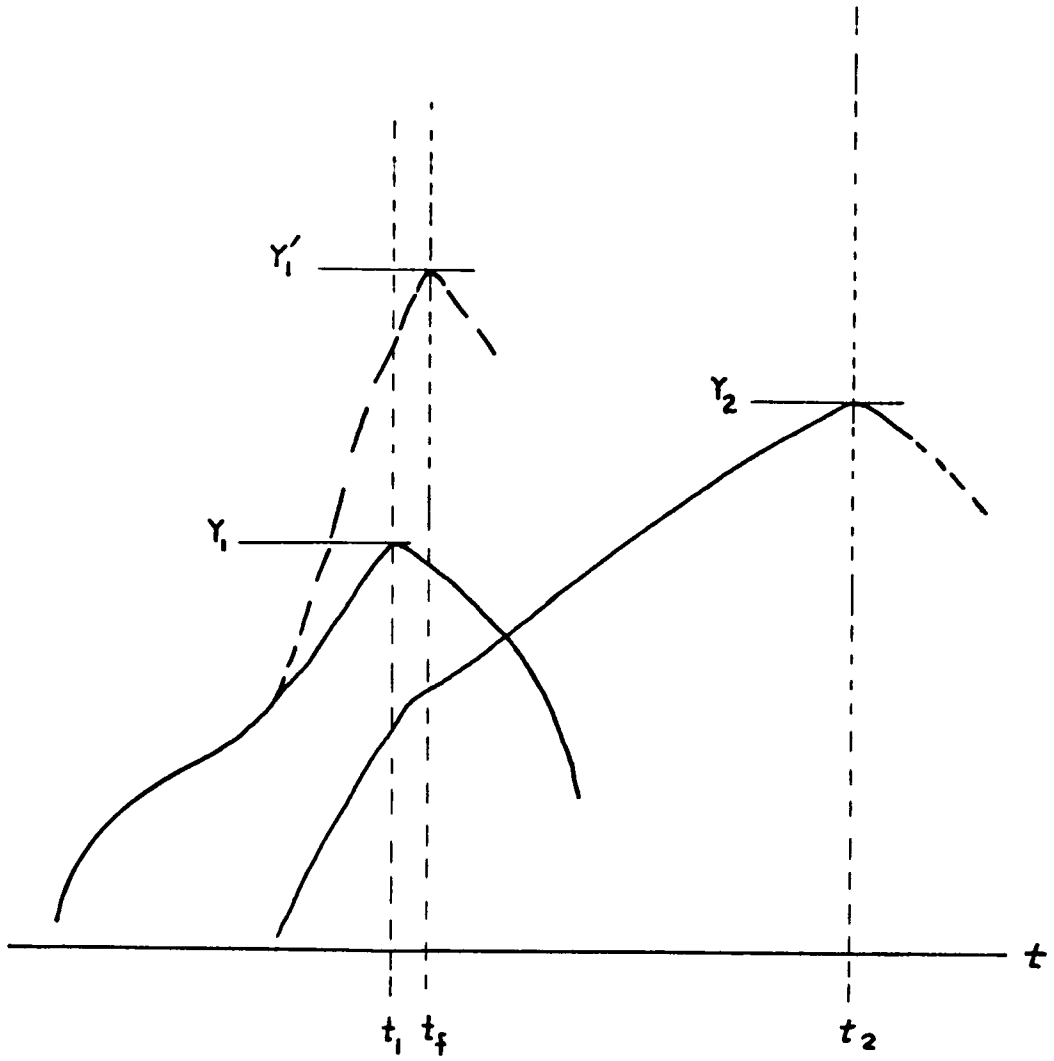



Figure II-7. Two Piled-Up Pulses with the Real Peak $Y_1$ and the False Peak $Y_1'$ Not Coincident.

If $\tau_a$ and $\tau_b$ are similar in magnitude then Equation II-24 will most probably be greater than Equation II-25 in the region where $t_1 < t <<< t_2$, that is much closer to $t_1$ than $t_2$. With this condition the peak would occur at $t_1$ causing $Y_1'$ and $Y_1$ to be coincident.

The conclusion of this discussion is that commonly encountered functions, exponential in nature, will produce coincident false and real peaks and will therefore satisfy the separation algorithm.

# CHAPTER III

## OVERVIEW OF THE REQUIREMENTS FOR AN INSTRUMENTAL IMPLEMENTATION OF THE SEPARATION ALGORITHM

The preceding chapters examined the nature of pulse pile-up distortion and developed an algorithm which would remove the distortion. This chapter begins development of an instrument which will implement the algorithm.

The algorithm requires that three types of data be available. They are (a) the pulse waveshape, (b) the uncorrected (measured) pulse heights, and (c) the pulse spacing. A first thought would be to perform the experiment, collecting all the data, and then afterwards, off-line, reanalyze the data. This requires an instrument that is essentially a memory which must store all the data for the entire experiment. The weakness in this approach is that any real instrument must, of course, have a finite memory size. This finite memory size will limit the length of any experiment which can be conducted; when the memory is full the experiment stops. To avoid this limitation and the delays involved in off-line analyzers, the pulse pile-up corrector should be an on-line instrument operating in real time (or near real time).

Reexamination of conventional pulse height analysis instrumentation (see Chapter I) shows that the needed data is either not available or not available in useful form. The pulse waveshape is fixed by the pulse shaping of the amplifier and can be considered fixed and known. However, pulse heights are individually measured, counted, and

then discarded. They are not retained in any memory. Pulse spacing is not measured at all. Consequently the instrument will need to measure and retain the pulse height and spacing of a set of interfering pulses long enough to perform the correction calculation. It can then output the pulse height data into the normal pulse height data stream and clear its memory to be ready for the next set of interfering pulses.

The pulse height corrector will require memory, will perform data manipulation, and will be required to solve a set of simultaneous equations. These requirements, along with the fact that this is to be a single purpose instrument, strongly suggest use of a microcomputer based on one of the currently available microprocessors. There is an obvious problem though, and that is speed. Most microprocessors operate in microsecond cycle times while the data may be available with nanosecond spacing. The potential solution for this problem is found in a reexamination of the conventional pulse-height-analyzer system. Noted earlier was the fact that the order the pulses were received by the pulse height analyzer was not important. If the interfering pulses can be captured in some type of buffer as they occur, then they can be analyzed by the microcomputer and reinserted into the data stream after some delay. This delay will not affect the accuracy of the pulse height analysis. (This operating mode is what is meant by the earlier phrase "near real time.")

At this point an examination of Figure III-1 should help make the envisioned instrument clearer. The Pile-Up Corrector is shown enclosed in the dashed lines. It contains a conventional microprocessor system with a special high speed buffer added for data input.

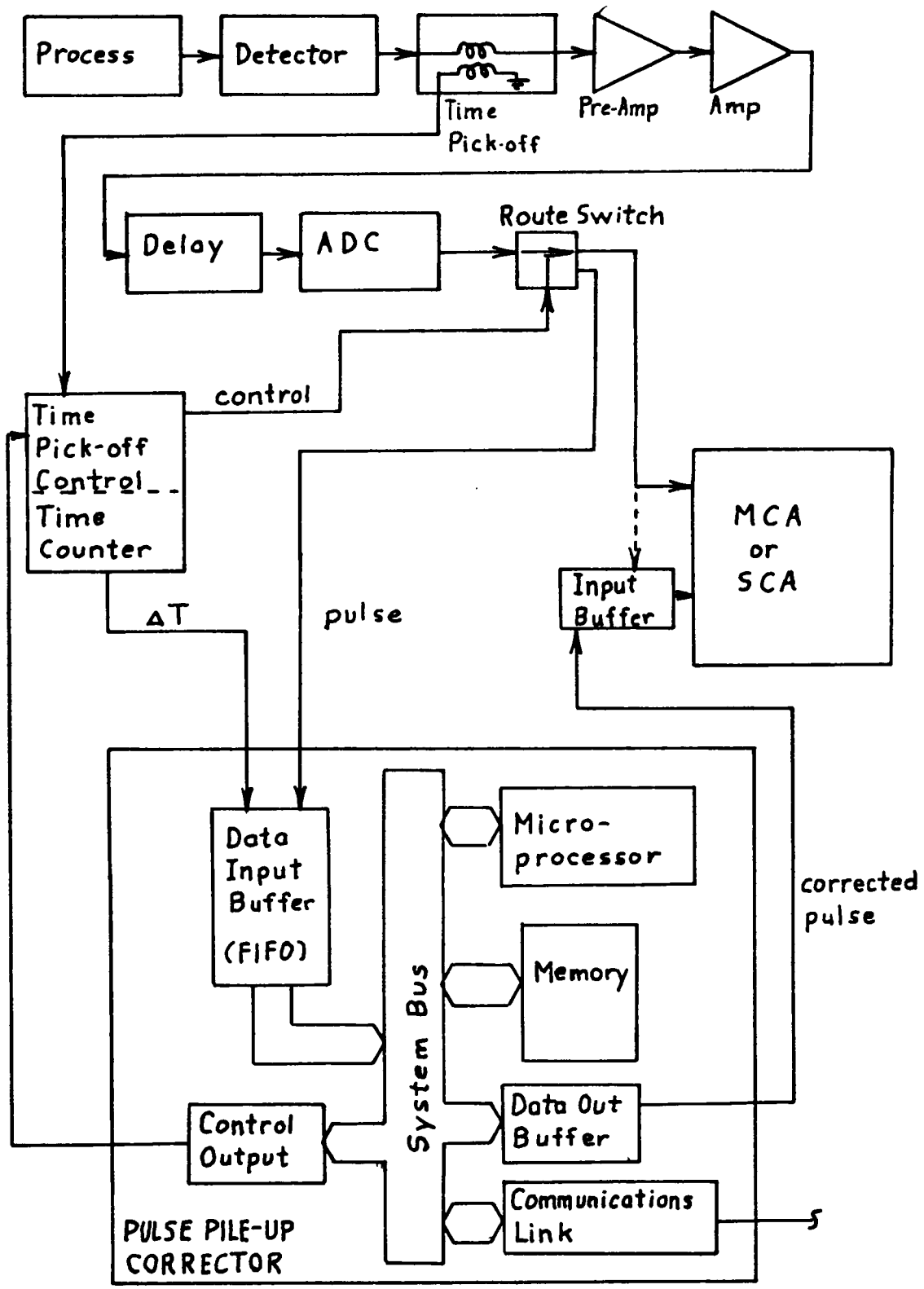



Figure III-1. Block Diagram; Pulse Pile-Up Corrector and Pulse Height Analyzer System.

The Pile-Up Corrector receives the input data via the buffer, performs the separation algorithm, and presents, as output, the corrected pulse heights.

The auxiliary instruments are the time pick-off and time pick-off control, the route switch, and the delay circuit. These devices are adaptations of instruments or circuits now available in pulse height analysis instrumentation. The time pick-off triggers on an event pulse, starting a timer in the time pick-off control. If a second pulse is now detected, by the time pick-off, before the timer completes a preset time interval, then the route switch is operated to divert the two potentially interfering pulses to the Pulse Pile-Up Corrector. This compares with the action of the time pick-off control and gating of the pile-up rejector discussed in Chapter I. In this unit, however, the time differential is digitized and transferred to the Pile-Up Corrector along with the pulse height. The delay circuit serves to delay the pulse propogation to the route switch until the preset time interval in the time pick-off control is exceeded. Finally, there may be a communications link from the Pile-Up Corrector to the pick-off control. This link allows the Corrector to set the time interval in the pick-off control.

The design of the auxiliary instruments is not a part of this work; however, certain requirements on these instruments will need to be specified.

The microcomputer is to be a conventional machine, but with a special input buffer added. This input buffer is built using a FIFO (First In First Out) register stack. Its function is to capture the

data as it occurs and to hold the data until the microcomputer can use it. This gives the computer the slack needed to operate within its own time requirements, and allows program development to be done without considering the need to control the data input. The program will monitor the "bottom" or output of the FIFO and when data is present transfer it into the computer for entry into the separation algorithm. With no data present the program simply idles.

The software's job is to execute the separation algorithm. It takes data from the FIFO, forms it into appropriate arrays, operates on these arrays solving the set of equations and then outputs the corrected heights.

Development of the software must begin by considering the possible data structure. The input to the microcomputer will be a series of mutually interfering pulses. There may be a very limited number of pulses (2 or 3) overlapping or there may be many pulses all mutually interfering. In any given computer configured to execute the separation algorithm, there will be some limit on the maximum number of equations that can be solved simultaneously. This limit could be anything, execution time, memory size, etc.; it is only important to know a limit exists. Call the limit N. Recognizing that there is the limit N leads to the conclusion that there are two cases to be considered in the software design. They are:

Case 1: The maximum number of interfering pulses is less than or equal to N.

Case 2: The maximum number of interfering pulses is greater than N.

There are two different software approaches that can be made, a design for Case 1, or a design for Case 2. (Note that a Case 2 design could handle Case 1 situations, but at the cost of greater execution time.)

The Case 1 design is the simplest to program and it is the fastest operating. If there are n interfering pulses (where $n \leq N$) then the Corrector collects all n pulses. It then solves the n equations to find the n unknowns. Thus, a set of n simultaneous equations is solved one time to produce n outputs. This point seems obvious and has been belabored but will acquire importance when compared to the Case 2 procedures. This procedure used for the Case 1 solution will be called "Batch."

The Case 2 design, which will be called "Spotlight," is more involved. In this case there are n potentially interfering pulses where $n > N$. This means that an exact solution will not be possible, but it should still be possible to get a solution. Figure III-2 shows five pulses which are potentially interfering. This makes n=5. Assume that N=3. With this case data from two pulses will need to be discarded. Examine pulse 3, that is, "Spotlight" pulse 3. The contributions to pulse 3's false height $Y_3'$ from pulse 1 and 5 are neglected and the $Y_3$ is calculated using $Y_2'$, $Y_3'$, and $Y_4'$. This can be expected to produce a smaller error in $Y_3$ than in the other values $Y_2$ and $Y_4$. These later would exhibit greater error since they are more strongly affected by pulses 1 and 5. Consequently only pulse 3 will be calculated from pulses 2, 3, and 4. Likewise, to find $Y_2$, use pulses 1, 2, and 3. The procedure for Case 2 requires correcting each pulse individually. In effect each pulse is, in turn, assigned the position $Y_n$ in Equation II-21, page 18, and then evaluated.

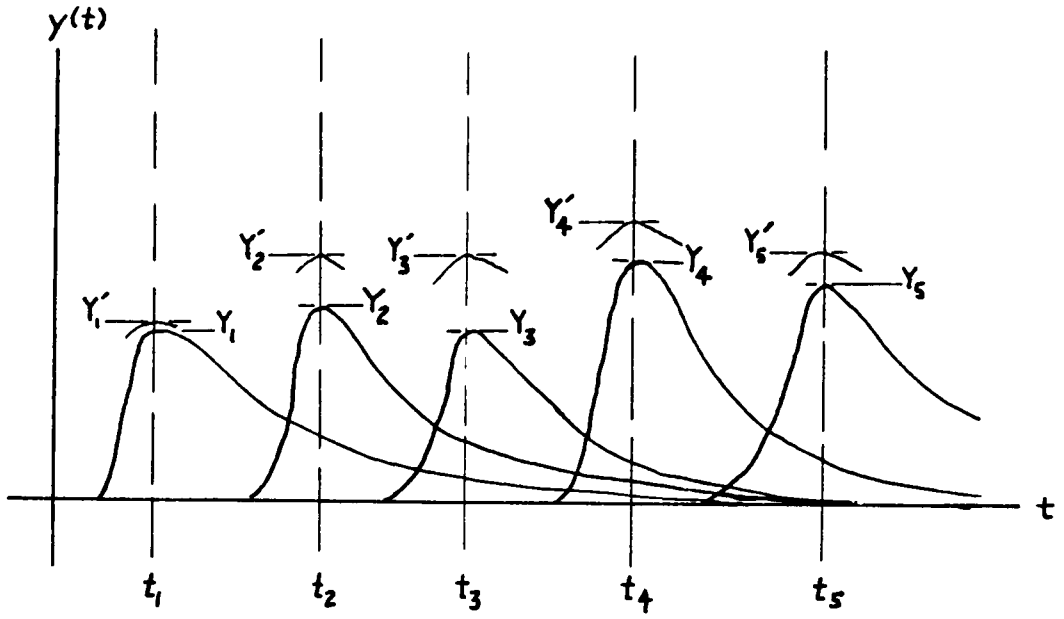



Figure III-2. Five Mutually Interfering Pulses.

The accuracy of each $Y_n$ calculated using "Spotlight" is a function of the number of points considered and the normalized pulse shape function $f(\Delta t)$. An expression for the worst case error can be calculated using Figure III-3. Assume the system can handle i+k+1 pulses. Also assume there is a long stream of pulses (greater than i+k+1) and that the spacing between pulses is the minimum that can be resolved. Let $\varepsilon$ be the greater of the values of $Y_{n-i}f(\Delta t_i)$ and $Y_{n+k}f(\Delta t_k)$. Then the error contributed by each ignored pulse will be less than $\varepsilon$. If the minimum pulse spacing is $\tau$, then the worst case error can be written:

$$E = \sum_{m=1}^{p} Y_{n-i-m} f(\Delta t_i + m\tau) + \sum_{m=i}^{q} Y_{n+k+m} f(\Delta t_k + m\tau) \qquad \text{III-1}$$

Where p+q is the number of potential interfering pulses that are ignored. An upper bound on the error is:

$$E < (p+q)\varepsilon \qquad \text{III-2}$$

It is not likely that the data structure will meet the worst case assumptions so the error actually encountered will probably be less than that shown above. If $\tau$ is assumed to be the average spacing between piled-up pulses instead of the minimum spacing, then III-1 and III-2 can be expected to give the typical error for the data structure.

The spotlight procedure is more generally applicable than the batch procedure, but it is much slower. With spotlight the correction of n points requires n solutions of as many as n sets of simultaneous equations. The batch procedure requires only one solution of n

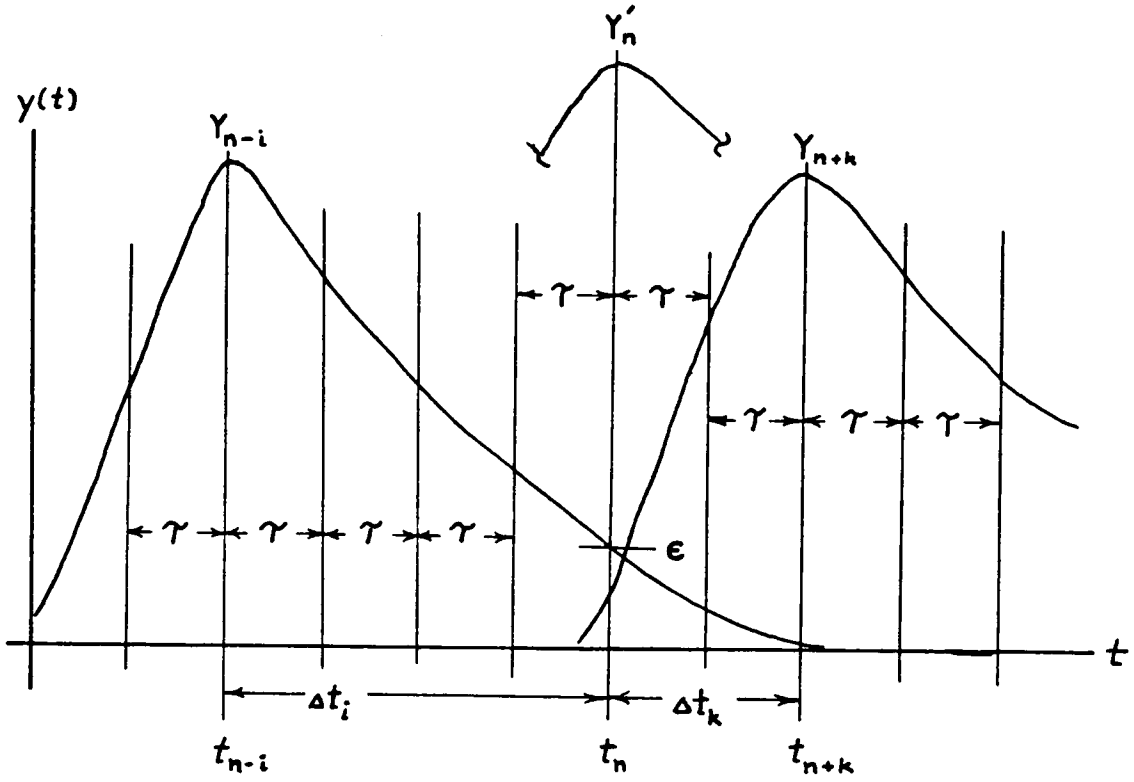



Figure III-3. Spotlight Procedure Error Analysis.

equations. Choice of procedure, again, requires that the data structure be carefully analyzed and trade-offs be made between speed and the number of points that can be handled.

# CHAPTER IV

## A SIMULATION OF THE ALGORITHM IMPLEMENTATION

Before actually beginning design of the Pulse Pile-Up Corrector, it was decided to simulate the instrument by programming the separation algorithm on an existing small machine. This was done to gain a better understanding of the problems that might occur when the algorithm was implemented in the new instrument. The machine chosen for this simulation was a Texas Instruments, Incorporated TI-59 programmable calculator with a PC 100-C printer. The reasons for this choice were:

1. It is a small machine of limited program and data memory, constraints that would likely be part of any microcomputer selected.

2. It uses a programming language somewhat similar in structure to assembly language.

3. It has a preprogrammed routine for solving sets of simultaneous linear equations.

4. It was readily available.

An assumption was then made that the software design should be done for the Case 2 situation (as described in Chapter III). This meant that the data handling system would have to be the "Spotlight" procedure. This choice was made because this procedure requires more involved programming than the Case 1 situation using the "Batch" procedure; and it is more powerful in all respects except speed.

A simplified flow chart describing the algorithm implementation was then prepared (Figure IV-1). In preparing the flow chart the following points were considered:

1. The need to store the "i" data points preceding the spotlighted point, and the "k" points following the spotlighted point.

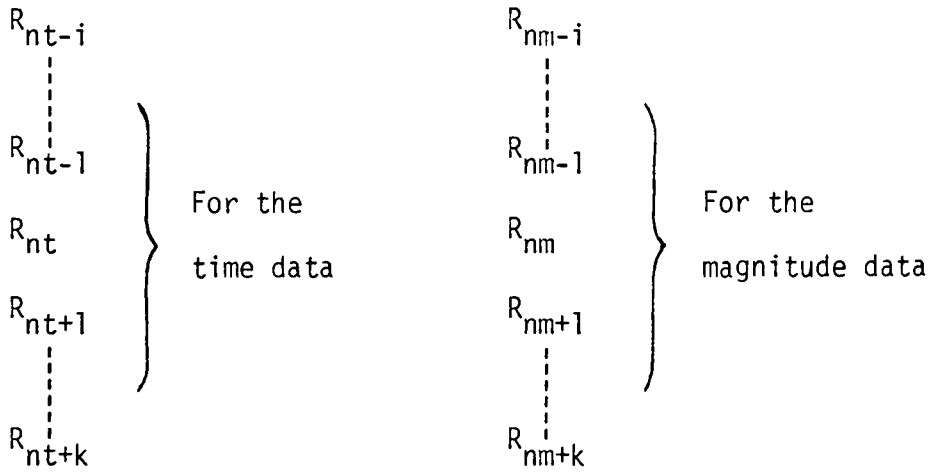
2. The need to store the normalized pulse waveshape $f(\Delta t)$.

3. The need for a routine to generate and handle the arrays of the equation set $[f(\Delta t)][Y] = [Y']$.

4. The need for a procedure to solve the equation set to find $Y_n$.

5. The need to provide for buffer storage that operates independently of the program to store any new data points that occur during the calculation of the currently spotlighted $Y_n$.

Point (1) was handled by designating specific registers to be the data registers as illustrated below:




where nt is the register selected to store the $n^{th}$ time data point and nm is the register selected to store the $n^{th}$ magnitude data point.

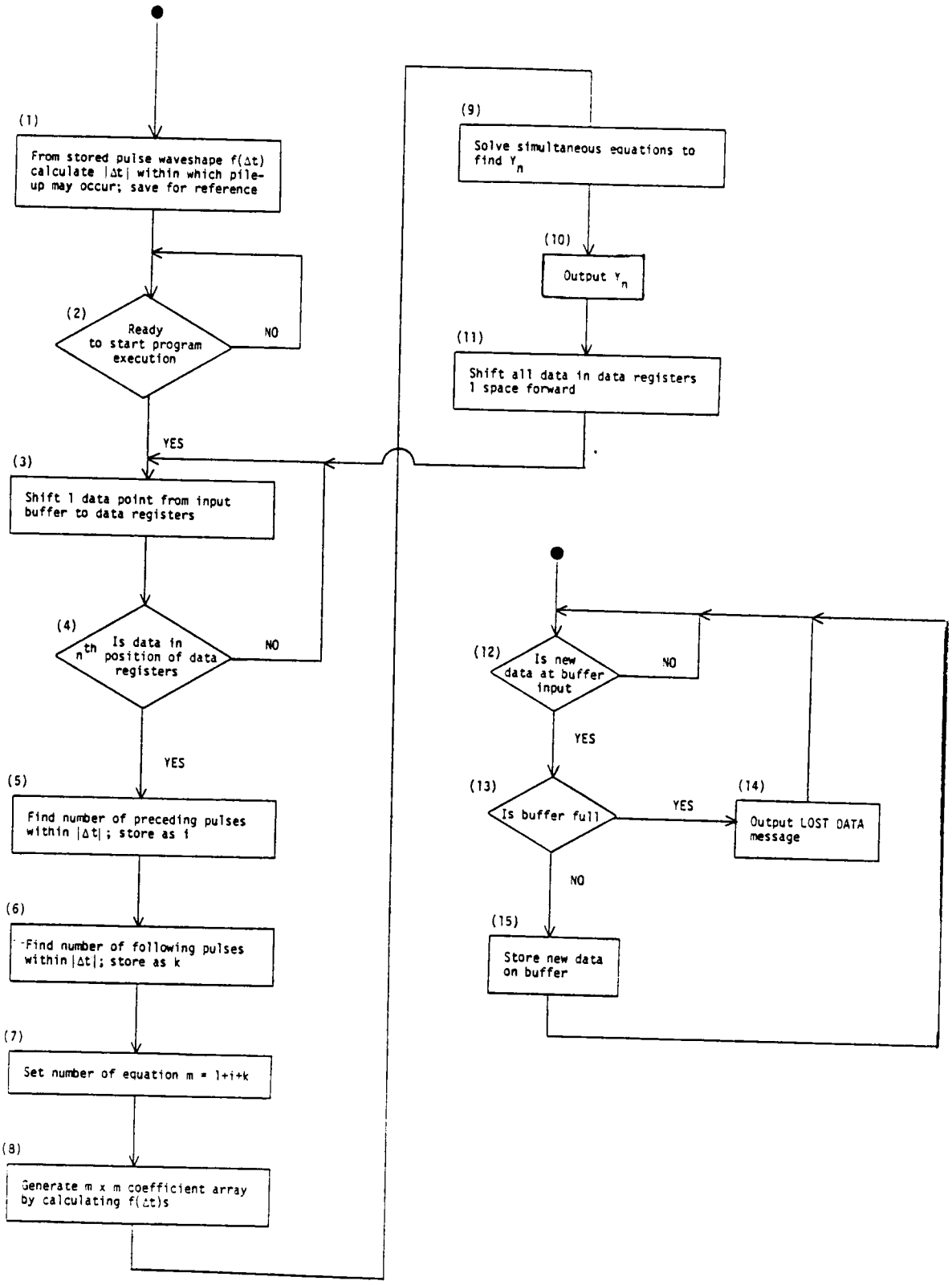



Figure IV-1. Simulation Program, Simplified Flow Chart.

Point (2) concerns the normalized pulse waveshape. The pulse waveshape for all pulses can be represented by some normalized function, $f(\Delta t)=f(t-t_0)$, as already discussed. There are two ways to handle programming the waveshape; enter the function as a look-up table or generate it as a mathematical function. In the actual instrument the look-up table will be preferred because of speed; however, for this simulation a mathematical generation for $f(t-t_0)$ was chosen. This was primarily because of limited memory space in the calculator. (Note that there are two ways in which the time data might be available; it is either collected as the actual time of occurrence of the pulses or as the time spacing between the pulses. The work on this simulation assumed data would be available in the time of occurrence format. A later study of vendor catalogs, notably Ortec (1976) and Canberra Industries (1979-80) indicates that the time spacing would be most easily generated. It appeared that the programming effort for either approach would be similar so it was not judged necessary to change the simulation.)

Points (3) and (4) involve the bulk of the programming work of the simulation and will be covered in the discussion of the simplified flow chart, Figure IV-1 (a detailed flow chart and description is in Appendix A).

Point (5) concerns buffering of continually occurring input data. In the microcomputer this will be handled with a FIFO (First In First Out) memory stack. The FIFO was not simulated in the calculator since the calculator is not handling the data in real time. The data for the calculator is maintained on a pencil and paper listing

and entered manually into the calculator one data point at a time. As soon as the calculator has enough points to perform the calculation on the spotlighted data it stops accepting data and makes the correction calculation. It then is available for new data, if available, and calculation of the next spotlighted point.

The simplified program flow chart was developed considering the above points (see Figure IV-1). Block (1) of the flow chart accesses the normalized pulse waveshape generator to determine the maximum $|\Delta t|=|t-t_o|$ in which pulse pile-up can be expected to occur. (This step is included to accommodate changes in the normalized pulse shape that may occur when the corrector is used in differing instrumentation setups.) Block (2) is a wait function which may or may not be needed in the actual instrument and is primarily included to remind the user to properly initialize the Pile-Up Corrector. The program begins with block (3) by shifting the first data pair (time and magnitude) into the data register pair. The decision in block (4) checks to see if there is data in the $n^{th}$ or spotlighted data register. If not, the program loops back and continues to enter data until the $n^{th}$ data register is filled. Block (5) then calculates the number i of data points preceding the spotlighted data point. Block (6) does the same for the k following data points. Block (7) then calculates the number of simultaneous equations to be solved by adding m=1+i+k. Note that this block also limits the maximum number of equations to within the program's capability. The $f(\Delta t)$ coefficient array is generated by block (8), then block (9) solves the equation set presented by the data and the $f(\Delta t)$ arrays. Block (10) then outputs the corrected

spotlighted pulse height $Y_n$. Finally block (11) shifts all data one register pair ahead to spotlight a new data point and loops back to begin this next calculation.

The normalized pulse waveshape used (and shown in Figure IV-2) requires some further comment. The function chosen for the waveshape was:

$$f(\tau) = \begin{cases} \exp(-4.7\tau^2) & \tau>0 \\ \exp(-4.7\tau^2+3\tau) & \tau\leq 0 \end{cases} \qquad \text{IV-1}$$

This function was selected because it produces a nonsymmetrical pulse about $\tau=0$. It is normalized in time as well as in magnitude; therefore a time scale factor is needed. The normalized pulse width is 2. If the corresponding actual pulse width is PW, then the time scale factor is then T=PW/2. The factor T is entered in the initialization program and applied to the time data entered into the program.

Upon completion of the simulation program a test pulse set was constructed to test the program. This pulse set was prepared by first selecting a time scale and time scale factor (T = 5 microseconds was used) and then spacing several pulse bursts of varying degrees of potential overlap on the time scale (see Figure IV-3). Then each pulse was given some arbitrary pulse height, $Y_n$. Next the corresponding false pulse heights, $Y_n'$, were calculated using Equation IV-1 for the $f(\Delta t)$ coefficients (with $\tau=t/5\mu sec.$) and Equation II-21, page 18, to solve for the $Y_n'$ values. The pulse set was then entered into the calculator algorithm simulation program and a set of results duly generated. These results are presented in Table IV-1.

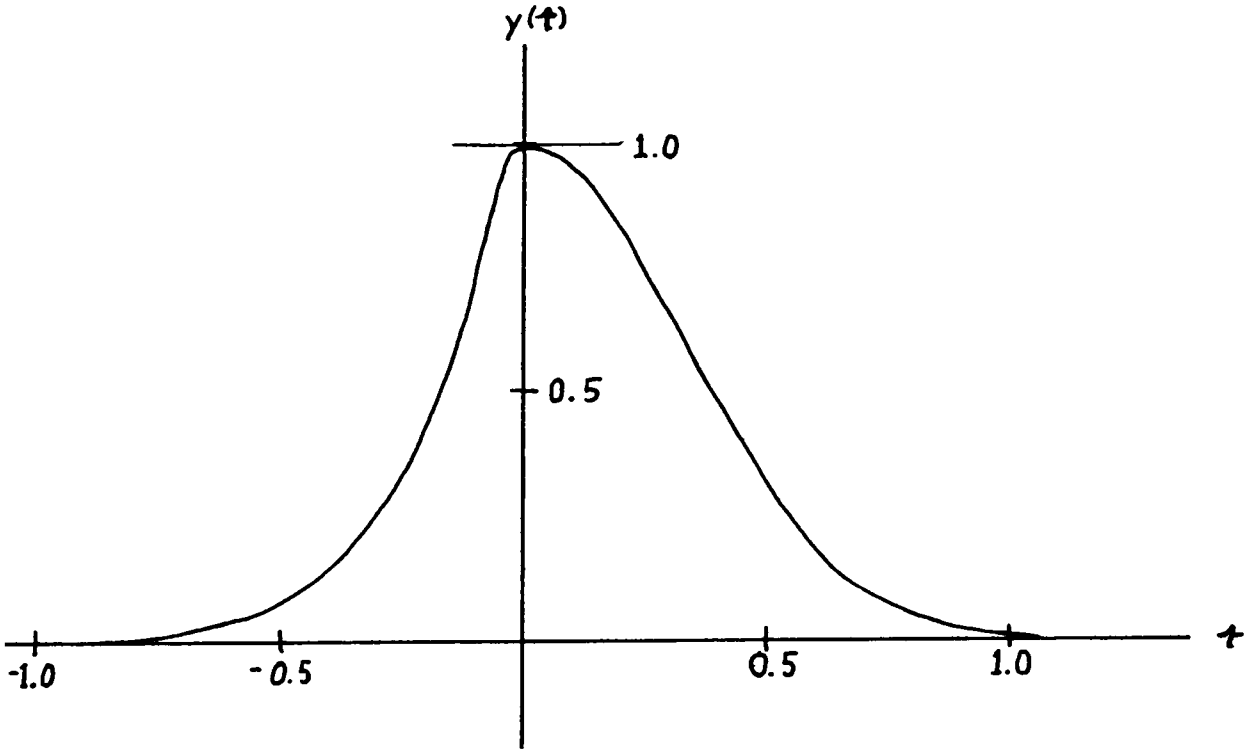



Figure: IV-2. Sample Normalized Pulse Waveshape, f(Δt). This function is the pulse waveshape function used in the TI-59 simulation of the Separation Algorithm.

Note: $\tau=\Delta t$

Defining Equation

$y(\tau) = \exp(-4.7\tau^2), \quad \tau>0$

$y(\tau) = \exp(3\tau-4.7\tau^2), \quad \tau\leq 0$

Tabulated Values

| $\tau$ | $y(\tau)$ |
|---|---|
| -1.0 | 0.0005 |
| -0.5 | 0.0689 |
| -0.2 | 0.4543 |
| 0.0 | 1.0000 |
| 0.2 | 0.8286 |
| 0.5 | 0.3088 |
| 1.0 | 0.0091 |

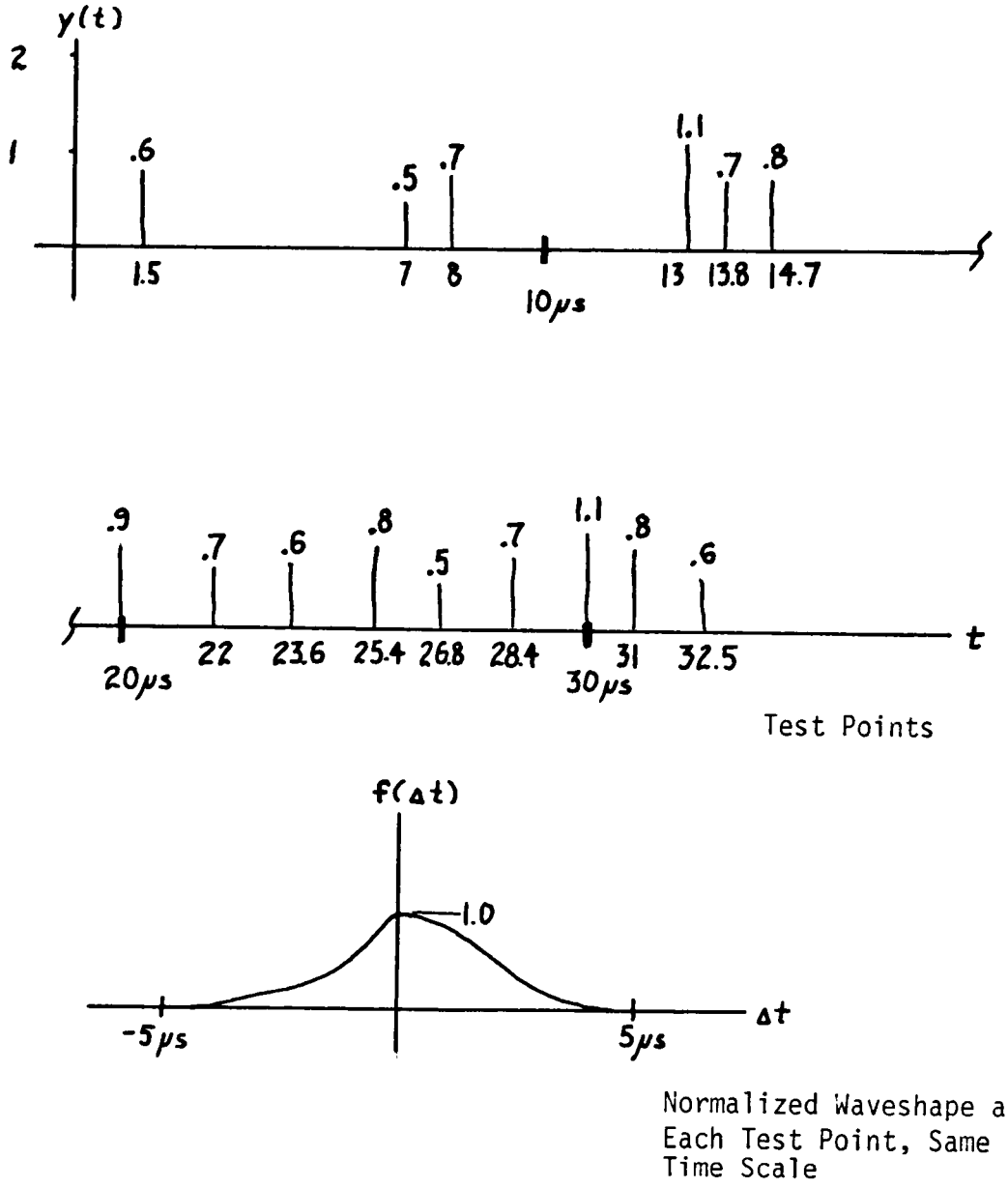



Figure IV-3. A Test Pulse Set. A set of data points used to test the TI-59 simulation of the Separation Algorithm.

TABLE VI-1

TEST POINT SET RESULTS

| Occurrence Time usec | True Height $Y_n$ | False Height $Y'_n$ | Number of Piled-Up Points | Number of Points Used | Algorithm Calculated $Y_n$ | System Error[1] | Algorithm Error[2] | Improvement Index[3] |
|---|---|---|---|---|---|---|---|---|
| 1.5 | 0.6 | 0.60000 | 1 | 1 | 0.60000 | 0.00000 | 0.00000 | n/a |
| 7.0 | 0.5 | 0.81833 | 2 | 2 | 0.49999 | 0.31833 | 0.00001 | $31.8 \times 10^3$ |
| 8.0 | 0.7 | 1.11431 | 2 | 2 | 0.70000 | 0.41431 | 0.00000 | $\infty$ |
| 13.0 | 1.1 | 1.65160 | 3 | 3 | 1.10000 | 0.55160 | 0.00000 | $\infty$ |
| 13.8 | 0.7 | 2.07565 | 3 | 3 | 0.70000 | 1.37565 | 0.00000 | $\infty$ |
| 14.7 | 0.8 | 2.04002 | 3 | 3 | 0.79999 | 1.24002 | 0.00001 | $124 \times 10^3$ |
| 20.0 | 0.9 | 1.00558 | 4 | 3 | 0.90277 | 0.10588 | 0.00277 | 38.1 |
| 22.0 | 0.7 | 1.27847 | 5 | 3 | 0.66879 | 0.57849 | 0.03121 | 18.5 |
| 23.6 | 0.6 | 1.27028 | 6 | 3 | 0.34452 | 0.67028 | 0.25548 | 2.6 |
| 25.4 | 0.8 | 1.38172 | 6 | 3 | 0.50542 | 0.58172 | 0.29458 | 1.9 |
| 26.8 | 0.5 | 1.34284 | 6 | 3 | 0.18109 | 0.84184 | 0.31891 | 2.6 |
| 28.4 | 0.7 | 1.47415 | 6 | 3 | 0.32468 | 0.77415 | 0.37532 | 2.1 |
| 30.0 | 1.1 | 2.02592 | 6 | 3 | 0.76043 | 0.92592 | 0.33957 | 2.7 |
| 31.0 | 0.8 | 2.08803 | 5 | 3 | 0.43565 | 1.28803 | 0.36435 | 2.5 |
| 32.5 | 0.6 | 1.49457 | 4 | 3 | 0.65874 | 0.89457 | 0.05874 | 15.2 |

Note: The results that were obtained when the test point set was entered into the TI-59 Algorithm Simulation Program.

[1]System Error: The error resulting from Pile-Up.

[2]Algorithm Error: The difference between the true height $Y_n$ and the algorithm calculated $Y_n$ (absolute value).

[3]Improvement Index: A measure of the improvement in accuracy of $Y_n$ resulting from the algorithm. Any value greater than 1.0 is an improvement. The Index = (Algorithm Error ÷ System Error).

The single pulse at t=1.5μsecond was passed through with no alteration as it should. The pulse bursts at 7 to 8μsecond and at 13 to 14.7μsecond were both corrected with negligible error. Since the calculator was programmed to handle a maximum of 3 sets of equations (3 pulses in mutual overlap), the negligible error in these bursts consisting of 3 or less pulses is to be expected. The long pulse burst between 20 and 32.5μsecond does exhibit error between the true $Y_n$ and the algorithm calculated $Y_n$. In all cases where there is an Algorithm Error the false peak results from the interference of 4 to 6 pulses. Since the program discards all but 3 points some error is to be expected. An Improvement Index was defined to quantify the improvement of the calculated corrected pulse height $Y_n$ over the false pulse height $Y_n'$. Any index number greater than 1.0 indicates an improvement. (The index is not applicable to a single pulse.) All of the corrected pulses show some improvement indicating that the Separation Algorithm does indeed improve the pulse height measurement system accuracy. The Algorithm Error magnitudes in the long pulse burst does indicate that the 3 equation capability of the TI-59 Simulation Program was inadequate.

The choice of the number of equations required with the spotlight procedure is a function of pulse spacing and pulse waveshape (pulse width). To estimate the number of equations needed, the possible data structure is examined to estimate the typical minimum pulse spacing between pulse pairs in a piled-up pulse burst. Then this time interval is divided into the total pulse width of the pulse waveshape function, $f(\Delta t)$. The result is the needed number of equations.

In the test pulse set, Figure IV-3, the typical minimum pulse spacing is about 1.5μseconds, and the pulse width is 10μseconds. Thus, a capability of handling 7 or 8 equations (pulses) is required.

The TI-59 simulation of the Separation Algorithm implementation was concluded with the confidence that the algorithm worked and could be easily programmed.

# CHAPTER V

## DESIGN OF THE PULSE PILE-UP CORRECTOR, PHASE 1

This chapter begins the actual design of a Pulse Pile-Up Corrector. The design study was divided into two phases. The first phase is development of the working concepts required by such an instrument. These concepts are those not dependent upon the particular microprocessor chosen. The purpose of this first phase is to serve as a guide for building the instrument from any selected microprocessor family. Phase 2 will be an example of the design of the instrument using a specific microprocessor. Phase 2 will be covered in Chapter VI.

Prior to beginning design of a pile-up corrector, a designer needs to make an analysis of his expected data structure. The purpose is, first, to decide if a pile-up corrector will or will not work. The secondary purpose is then to select the procedure, "Batch" or "Spotlight," to be used. In addition, this early examination provides some feel for the trade-offs that might need to be made between providing capability to handle worst case pile-up conditions and optimizing to handle typical or average conditions. To assist in this initial analysis a Decision Summary Figure, Figure V-1 has been developed. This figure is based on both information already presented and on information that is presented in this chapter. Matching a data structure against Figure V-1 is the recommended starting point for design of a pile-up corrector for any particular data structure.

I. Data Structure

A. 1. Piled-up pulse spacing, typical _____

2. Piled-up pulse spacing, minimum _____

B. Pulse burst rate (average) _____

C. Number of pulses per pulse burst, M.

1. Average M _____

2. Typical high M _____

3. Worst case high M _____

II. Input Buffer Capability

| Pulse Spacing | | |
|---|---|---|
| <10 nsec. | 10-100 nsec. | >100 nsec. |
| Pile-up corrector not usable | Pile-up corrector possible with careful selection and design of the Input Buffer | Pile-up corrector use not limited by Input Buffer considerations |

III. Calculate the time to process one burst of M pulses.

A. $T_{PM}$ = [MPY(MPU/MPY) + ADD(MPU/ADD) + TRF(MPU/TRF)]

X (1/clock rate)

MPY--the number of multiplications in the algorithm

ADD--the number of additions in the algorithm

TRF--the number of register-to-register transfers in the algorithm

MPU/MPY, ADD, TRF--the number of microprocessor cycles per MYP, ADD, or TRF

Figure V-1. Pulse Pile-Up Corrector; Design Decision Summary.

| M | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| MPY | 6 | 14 | 28 | 46 | 74 | 98 | 176 | 230 | 342 |
| ADD | 25 | 38 | 57 | 85 | 224 | 168 | 224 | 303 | 491 |
| TRF | 93 | 156 | 255 | 393 | 582 | 789 | 1200 | 1599 | 2499 |

B. Calculate the processing rate, $1/T_{PM}$.

IV. Check to see if the processing rate is adequate.

Compare average pulse burst rate, $P_{BR}$, and $T_{PM}$.

$P_{BR} < \frac{1}{T_{PM}}$ Correction with selected microprocessor is possible if the Input Buffer is correctly sized.

$P_{BR} \geq \frac{1}{T_{PM}}$ Pulse burst rate is too fast. Consider:

1. Using faster microprocessor.
2. If examination was done at worst case M try typical high or average M.
3. If neither 1 or 2 works then a near real time corrector is not possible.

V. Decide whether to use Batch or Spotlight.

A. Base the decision on the typical high M.

B. $M<5$ Use the Batch procedure.

$M \geq 5$ May use Batch or Spotlight, proceed as described in C below to decide.

C. A set of M pulses in one burst does not necessarily require every pulse in M to affect every other pulse in M. For example, if the pulse shape is a step rise and an exponential decay then each pulse can affect following pulses but not proceeding pulses.--Proceed by first determining the number of pulses $M_n$ in M that are involved in modifying any one "spotlighted" pulse. Next using the procedure in III above, calculate $T_{PMn}$ using $M_n$ instead of M. Next find the Spotlight procedure $T_{PMS}$ as:

Figure V-1. (Continued)

$$T_{PMx} = (M/M_n) \times T_{PMn}$$

Now select procedure according to:

| $T_{PMs} << T_{PM}$ | $T_{PMs} \sim T_{PM}$ | $T_{PMs} >> T_{PM}$ |
|---|---|---|
| Use Spotlight | Consider worst case high M and base decision on memory space required for MXM arrays and on any expected programming difficulties | Use Batch |

Figure V-1. (Continued)

As an example consider a measurement discussed in Canberra Industries 1979-80 Product Catalog. A typical Ge(Li) detector is used to count a Zn-65 source where the source geometry and strength are such that 1K counts per second are produced. This data is analyzed with an MCA to produce a particular spectra. Next a Cs-137 source is added such that the count rate is brought up to 5K counts per second. Now the Zn-65 portion of the spectra indicates 3.49 percent less Zn-65. This error is due to pile-up that occurs with a greater number of randomly distributed pulses at 5K than at 1K. This corresponds roughly to 3.59 percent of all pulses being in a pile-up situation, which indicates a pile-up pulse burst rate of 175 counts per second. If additional Cs-137 is added to bring the count rate to 15K then the error is 12.85 percent corresponding to a pulse burst rate of 1.93K counts per second. Analyzing these two examples using Figure V-1, and assuming the microprocessor is a Texas Instruments 9900 operating at 3MHz gives:

| Count = | 5K | 15K |
|---|---|---|
| I. A. | No data available for this section. | |
| B. | 175 counts/sec. | 1.93K counts/sec. |
| C. | No data available for this section. | |
| II. | No data available for this section. | |
| III. | $T_{PM}$=1.148 msec. | $T_{PM}$=1.148 msec. |
| | Both assuming M=3 thus | $1/T_{PM}$=871 bursts/sec. |

IV. $P_{BR}$=175<871

Correction is possible.

$P_{BR}$=1930>871

Correction is not possible. (If a SN 54/74S481 bit slice is substituted for the 9900 then $1/T_{PM}$ becomes 5226 and correction is possible.

V. Data not available for this section.

Assuming that the data structure has been examined, attention is now turned to the Phase 1 stage of this design beginning with a description of the hardware.

The Hardware

Figure III-1, page 24, was taken as the starting point for this design work. First an expansion of the Pile-Up Corrector block of that figure was constructed; this is shown as Figure V-2. This hardware is representative of the microprocessor family eventually to be chosen. Since this is a dedicated instrument, not a general purpose computer, the program storage is to be firmware, or ROM. It is also necessary to provide RAM as the instrument's working memory. A provision is made to output the corrected pulses in a parallel mode for direct presentation to an SCA or MCA. A serial communications link is provided; this link is to be used for external configuration of the instrument, if such is desired. (The configurable items are the normalized pulse waveshape and the time scaling.) Finally there is a special input buffer which consists of a FIFO memory stack.

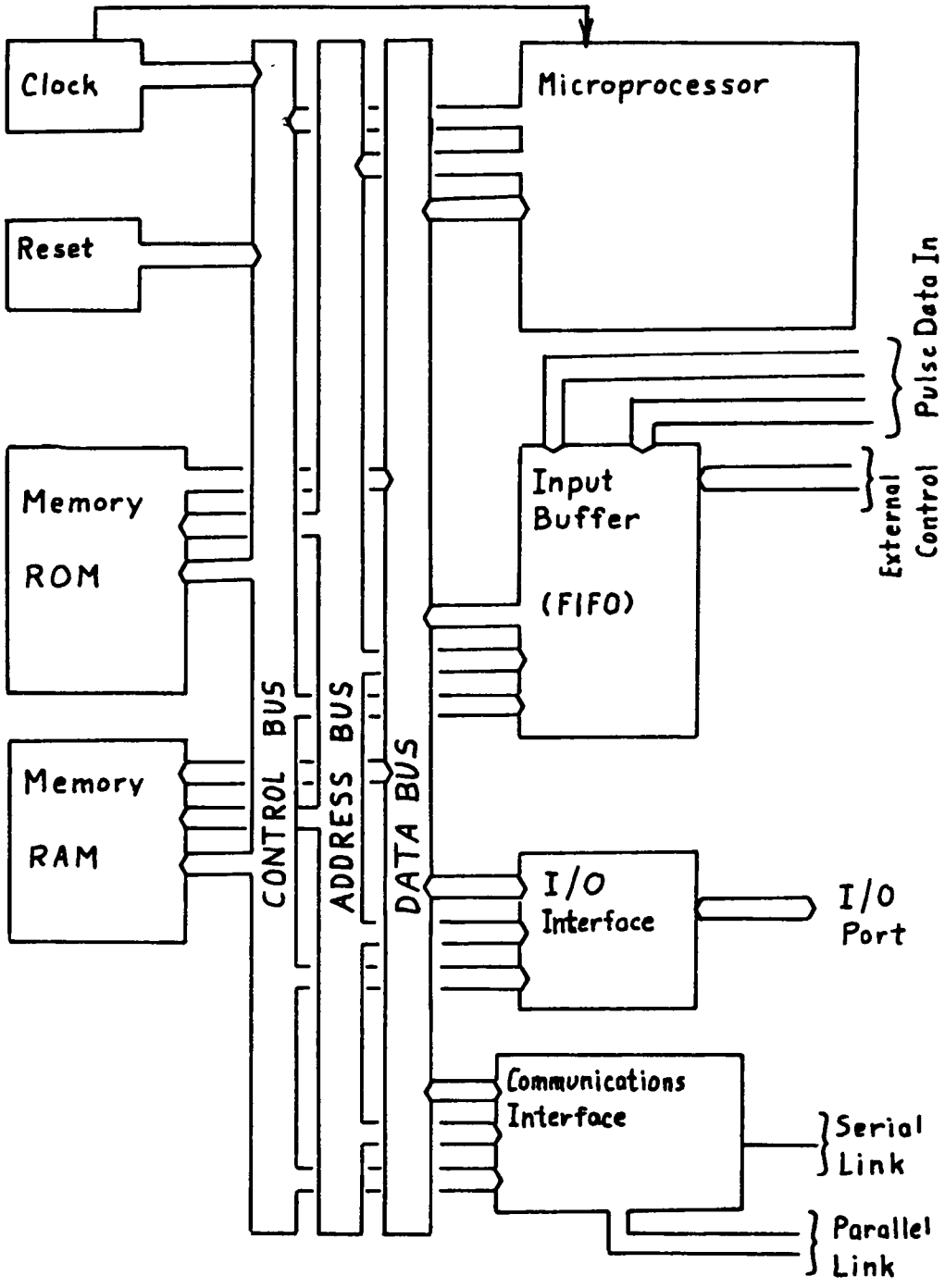



Figure V-2. Pile-Up Corrector; Block Diagram.

The input buffer is to be constructed from commercially available FIFO integrated circuits such as Monolithic Memories 67401/67401A plus switching, address decoding, and control circuits as shown in Figure V-3. Its input is the digital pulse magnitude and pulse time data entered directly and asynchronously by the other system instruments. The FIFO output is tied to two designated addresses in the instrument's memory.

The input buffer's function, as discussed in Chapter III, is to capture pulses with very close spacing and to hold them for processing by the microprocessor. Consequently, an examination of the FIFO's performance is in order. The FIFO's maximum input rate sets the minimum pulse spacing that can be captured by the Corrector. Monolithic Memories 67401's shift-in speed is 12 MHz for a pulse spacing of 83.3 nanoseconds, and as per Motorola Application Note AN-730, a FIFO constructed of MECL MC10143's has a data rate of 70 MHz for a pulse spacing of 14.3 nanoseconds. Thus, fairly high input speed requirements can be met.

There is a second important speed criterion which is a function of the speed at which the complete separation algorithm can be processed. A pulse burst of M pulses will require some time, $T_{PM}$, to be processed. Assuming all pulse bursts contain M pulses, then the maximum pulse burst processing rate, $P_{BR}$, is equal to $1/T_{PM}$.

The time $T_{PM}$ is a function of the number of potentially interfering pulses M; however, the dependence is not linear. Solution of the separation algorithm involves solving M simultaneous equations; as M becomes larger the number of operations increases greatly.

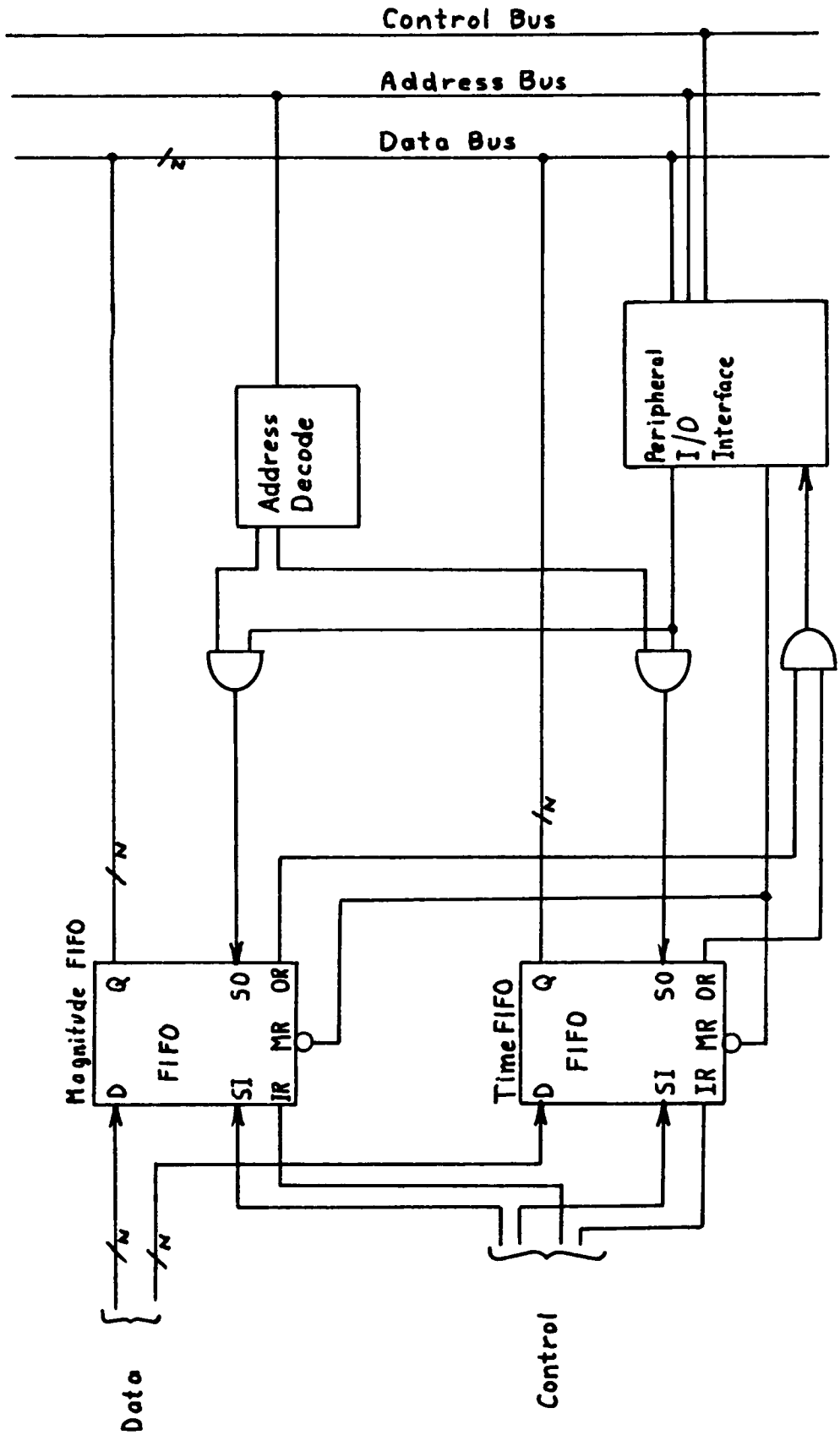



Figure V-3. Pulse Pile-Up Corrector; Input Data Buffer.

Figure V-1, Section III, page 44, lists the estimated number of multiplications, additions, and register-to-register transfers required for processing the pulse burst for M from 2 to 10. This estimate is developed from a study of the software presented later in this chapter.

It is necessary to develop some relationship concerning the required size, or depth, of the Input Buffer FIFO. The FIFO's depth, D, is the number of individual pulses (both time and magnitude data) that it can hold. If the pulses are evenly spaced as in Figure V-4, then the required D is just M, or one pulse burst. Unfortunately the distribution of piled-up pulses cannot be expected to be even, so the FIFO must have sufficient depth to capture any pulse bursts that occur while the processor is busy. For any Pile-Up Corrector the $P_{BR}$ must be less than or equal to $1/T_{PM}$ no matter how the pulses are distributed. If the data structure is found to contain a group of X closely spaced pulse bursts as in Figure V-5, then in some time $T_c$ there should be an additional Y pulse bursts such that:

$$\frac{X + Y}{T_c} \leq \frac{1}{T_{PM}} \qquad \text{V-1}$$

If X is the largest number of closely spaced (less than $T_{PM}$) then the FIFO must be sized to hold all X pulse bursts. Assuming each pulse burst contains M pulses, then the required size is related to X by:

$$X + 1 \leq \frac{1}{T_{PM}} \qquad \text{V-2}$$

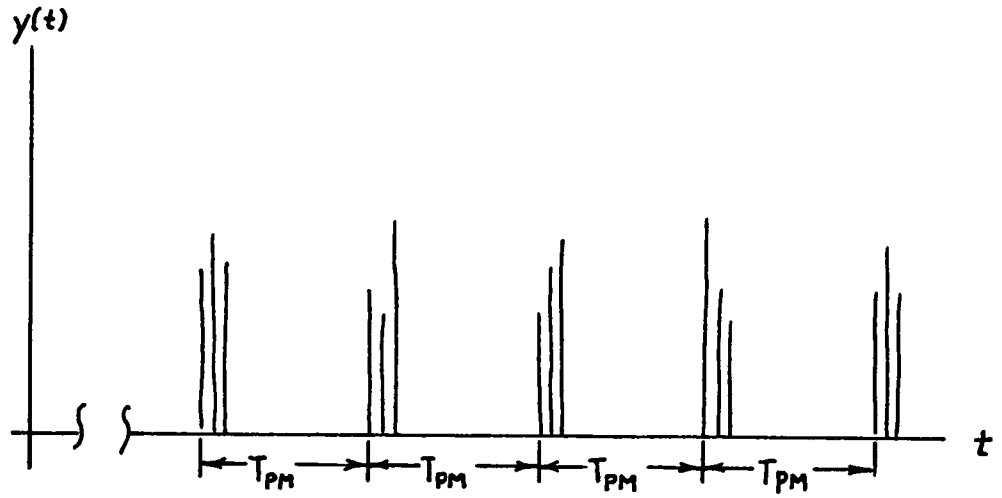



Figure V-4. Pulse Data Structure with Pulse Bursts Evenly Spaced.

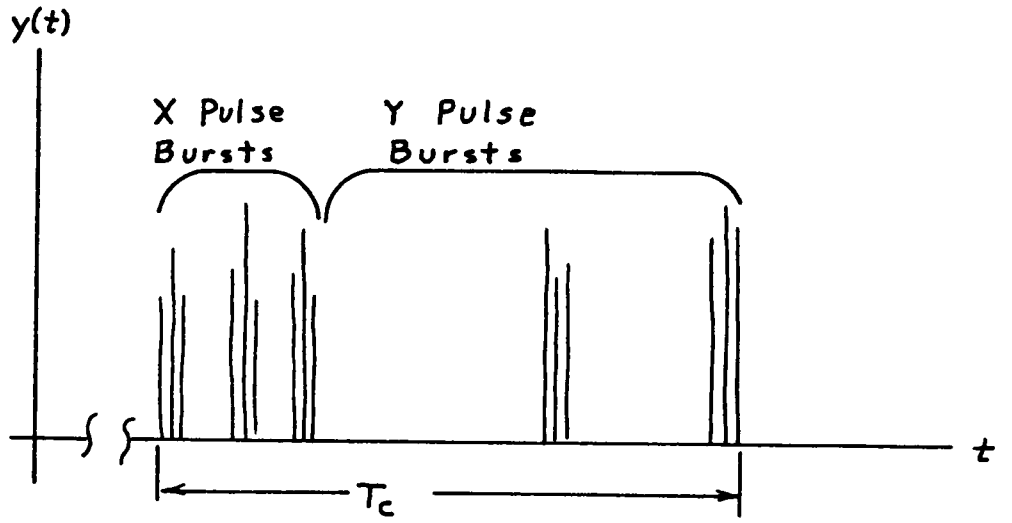



Figure V-5. Pulse Data with One Closely-Spaced Group of Pulse Bursts.

The 1 on the left-hand side of V-2 is to allow for the pulse burst that may be in the FIFO just ready for processing at the time the group of X bursts begins.

Before completing the hardware discussion there are two more items that must be mentioned. The first is to note that the logic levels used by the microprocessor and those used by available pulse height analyzer systems may not be the same. Therefore signal level conversion circuits must be included as part of the Pulse Pile-Up Corrector. The design of these circuits is not considered in this study. Second a specification describing the performance of "other system components" must be included. The other system components are (a) the Time Pick-off and Pick-off Control and Register, (b) the Delay Module, (c) the ADC, (d) the Route Switch, and (e) the Return Buffer.

The Time Pick-off and Pick-off Control and Register is a derivative of the Time Pick-off Control used with a Pulse Pile-Up Rejector (see Figure III-1, page 24). It is preset (either manually or by signal from the Pile-Up Corrector) with the maximum pulse spacing having the potential of pile-up. An event pulse, through the Time Pick-off, starts a timer. If the time runs until it exceeds the preset time then the Route Switch is set to allow the pulse to pass directly to the Pulse Height Analyzer. If, however, a second event pulse triggers the Time Pick-off before the preset time is reached then the accumulated time is loaded down to an output register. The Control unit also sets the Route Switch to route the digitized pulse height to the output register. When the output register contains both the pulse height and the $\Delta T$ (differential time), then the output

register transfers the data to the Input Buffer of the Pile-Up Corrector. The second pulse also clears and restarts the timer to look for the next following pulse. The ADC is the fast analog to digital converter commonly used with pulse height analysis instrumentation. The return buffer is a memory register and control unit which receives corrected pulse heights from the Corrector and inserts them into the data lines to the Pulse Height Analyzer.

Note that pulse time data obtained above is the time difference between two pulses, not a time of occurrence as in Equation II-21, page 18.

The Memory Map

With the hardware defined, attention was turned to developing a memory allocation plan. The memory map shown in Figure V-6 was the result. The first section of memory is the program memory space which is to be ROM (or PROM). Nonviolate program memory is used to permit the instrument to be a start-up on power-up device. As such, the ROM also contains the power-up vectors and initializations required by the microprocessor. The second major section is the $f(\Delta t)$ look up table or map. A 128 word space is reserved for the normalized waveshape $f(\Delta t)$. This should provide sufficient precision however, a larger memory space to increase precision could be easily provided, if needed. The $f(\Delta t)$ table may be either ROM, PROM, or RAM. ROM would be used if the table was never to be changed, PROM if changes were probable but were to be infrequent, and RAM for a frequently changed waveshape. This section of memory should also contain the time scaling factor. The remainder of the memory is RAM. A section of this RAM

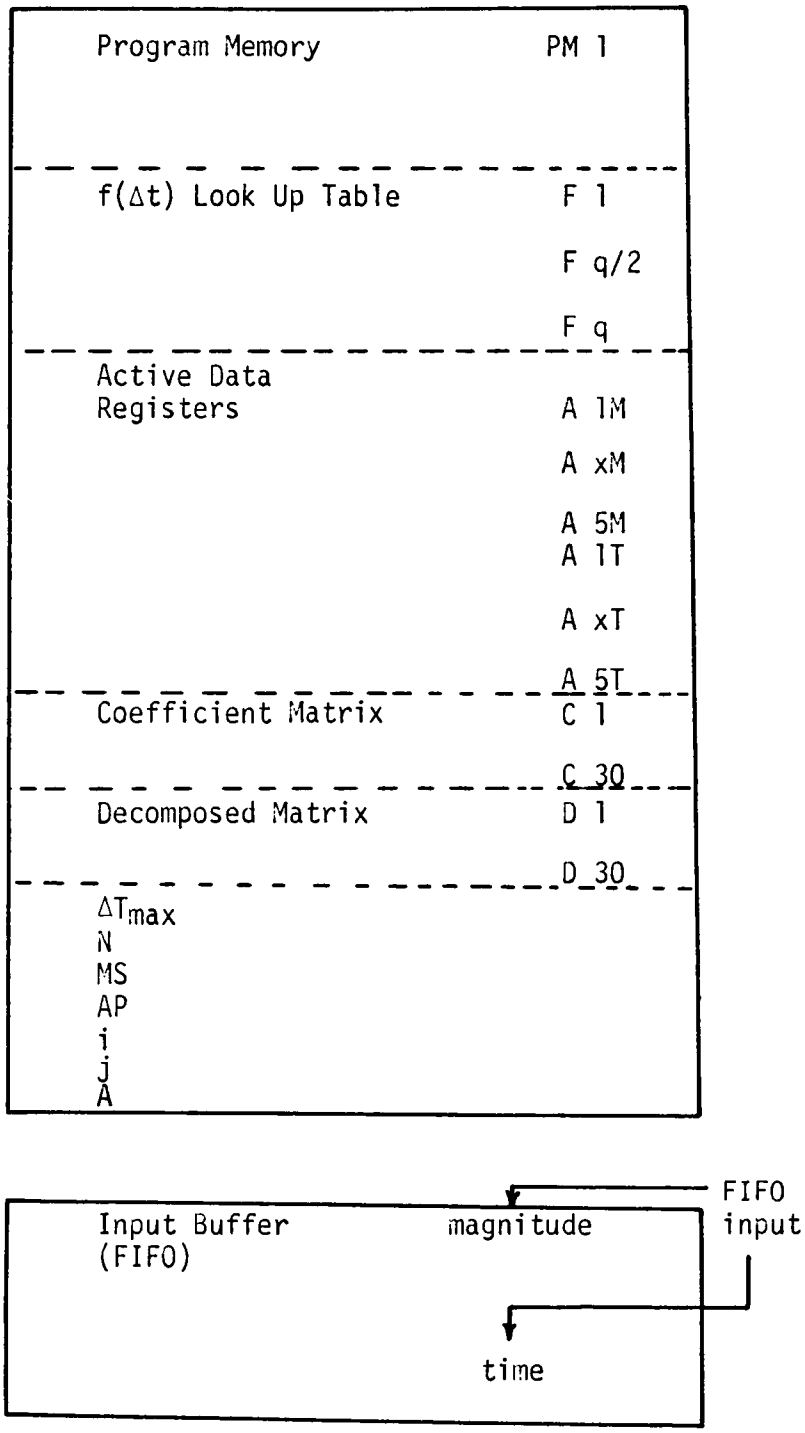



Figure V-6. Pulse Pile-Up Corrector Memory Map.

is designated Active Data Registers. These registers store the pulse burst magnitude data in locations $A_{1m}$ to $A_{5m}$ and the time data at $A_{1t}$ to $A_{5t}$. Note that only five data points are provided for; this will be discussed later. Another section is reserved for the coefficient matrix, a 5X6 array stored in locations $C_1$ to $C_{30}$, and the decomposed matrix, also a 5X6 array, stored in $D_1$ to $D_{30}$. The remainder of RAM is considered scratchpad, although some spaces are reserved for several constants, indexes, counters, etc. generated during the program operation. Finally the Input Buffer's output is located and addressed as memory. Reference to Figure V-3, page 51, shows the Input Buffer is actually two buffers, one for the magnitude and one for the time data. Two consecutive addresses are used for the output of these two buffers. Care needs to be taken in read operations to always read both a magnitude and a time at every reading. This keeps the data pairs in the FIFO properly aligned.

The Software

The first step of the software design was to define a typical data structure for the separation algorithm to handle. It was assumed at the outset that the pulse burst rate, $P_{BR}$, would be less than $1/T_{PM}$, since that is a requirement for a successful Pile-Up Corrector. Also it was assumed that the minimum pulse spacing was not less than the FIFO's capability. This left only the requirement to specify M, the number of pulses per pulse burst. The typical data source generates pulses randomly and pulse pile up is just the random coincidence of these random pulses (Canberra Industries, 1979-80, p. 34;
Thus finding M for a given data structure is a probability

problem. The probability of a small number of pulses being coincident is greater than the probability of a large number of pulses being coincident. Instrument vendors literature (Ortec, Canberra Industries) and the papers on nuclear pulse amplifiers by Fairstein and Hahn (1965) do not directly address the number of pulses in a typical pile-up, but they do leave the impression that only a few (2 or 3) will be involved. Consequently, the software was developed for a data structure characterized by:

1. Minimum pulse spacing satisfactory for capture by the Input Buffer.

2. Number of pulses per burst, M, normally 2 or 3.

3. $P_{BR}$ always less than $1/T_{PM}$.

This design work done thus assumes normal pile-up conditions involve two or three mutually interfering pulses. A capability of handling five simultaneous equations, in the Batch mode, was therefore incorporated in the program. A program step that determines the actual number of equations involved in any single pulse burst is part of the program. By solving only the needed number of equations, processing time per pulse burst was reduced. The limitation to five pulses results in no real loss of generality since it affects only memory space requirements for array, and indexing limits.

The basic software structure is similar to that for the TI-59 simulation. An Executive program is used to guide the operation through a series of functional subprograms. The Executive (Figure V-7) begins by vectoring, on power-up, to an initialization routine particular to the microprocessor system selected. It then presents the

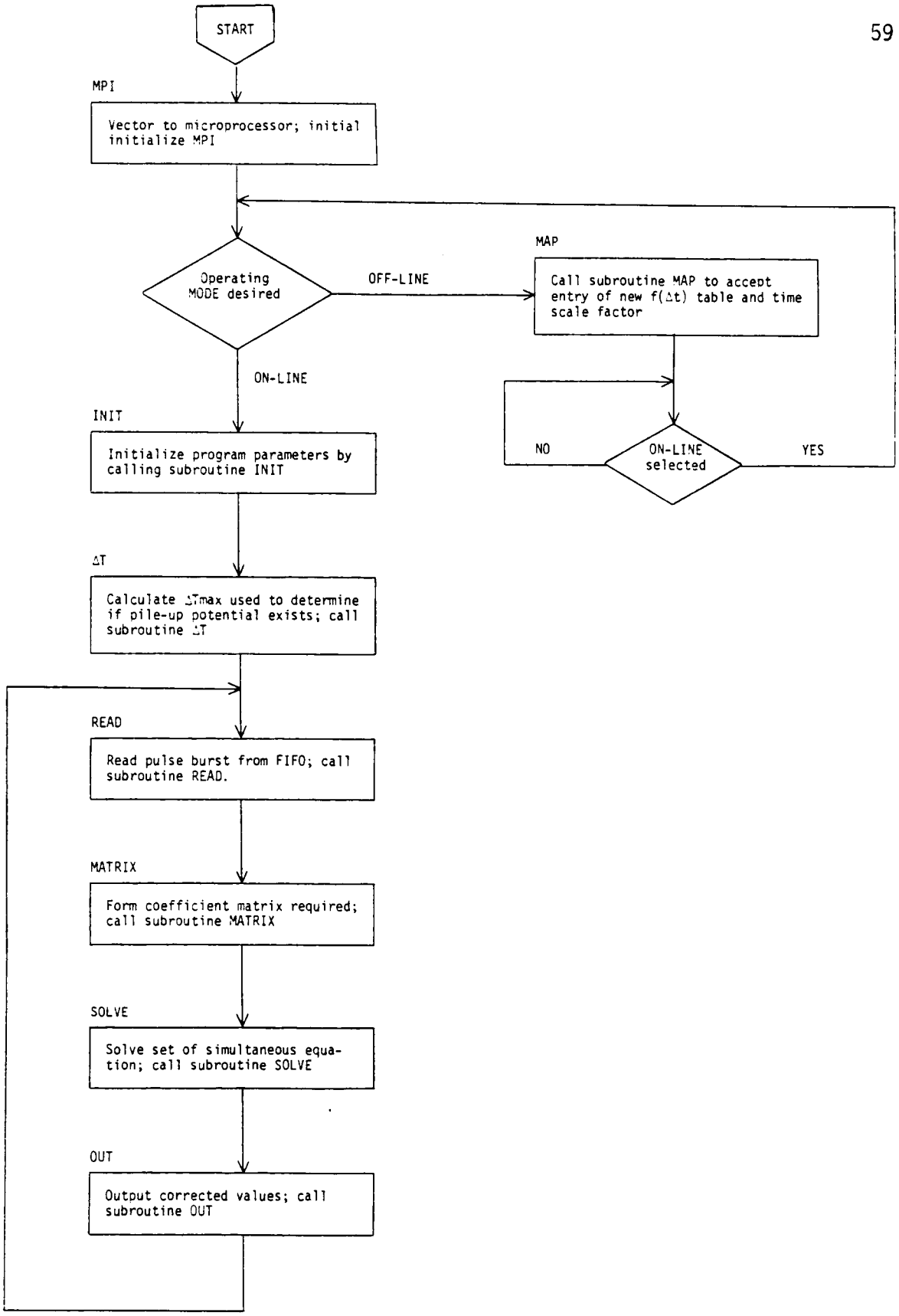



Figure V-7. Pulse Pile-Up Corrector Flow Chart, Executive.

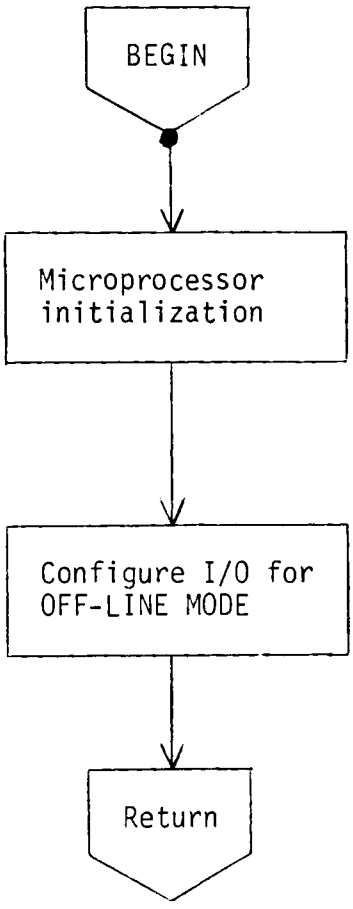
instrument user an opportunity to go "off-line" and reconfigure the normalized waveshape, $f(\Delta t)$, look-up table and enter the time scale factor. Then the Executive directs the program through the several functional subprograms. The Executive is shown in Figure V-7 and the series of functional subprograms in Figure V-8.

The first functional subprogram is the off-line program MAP. This controls the interaction with an external device, such as a desk-top computer, in the process of entering the normalized pulse waveshape $f(\Delta t)$ map, and entry of the time scale factor. It is also possible to have the map and time scale stored in PROM. Then a change in this item can be done by exchanging PROM, in which case the subprogram MAP would not be needed.

The next subprogram is the initialization subprogram, INIT. INIT first reconfigures all I/O peripherals to the on-line mode, then clears both the all memory space (RAM). Next it identifies the beginning of the coefficient matrix and loads 1 in the principal diagonal. Then control is returned to the Executive.

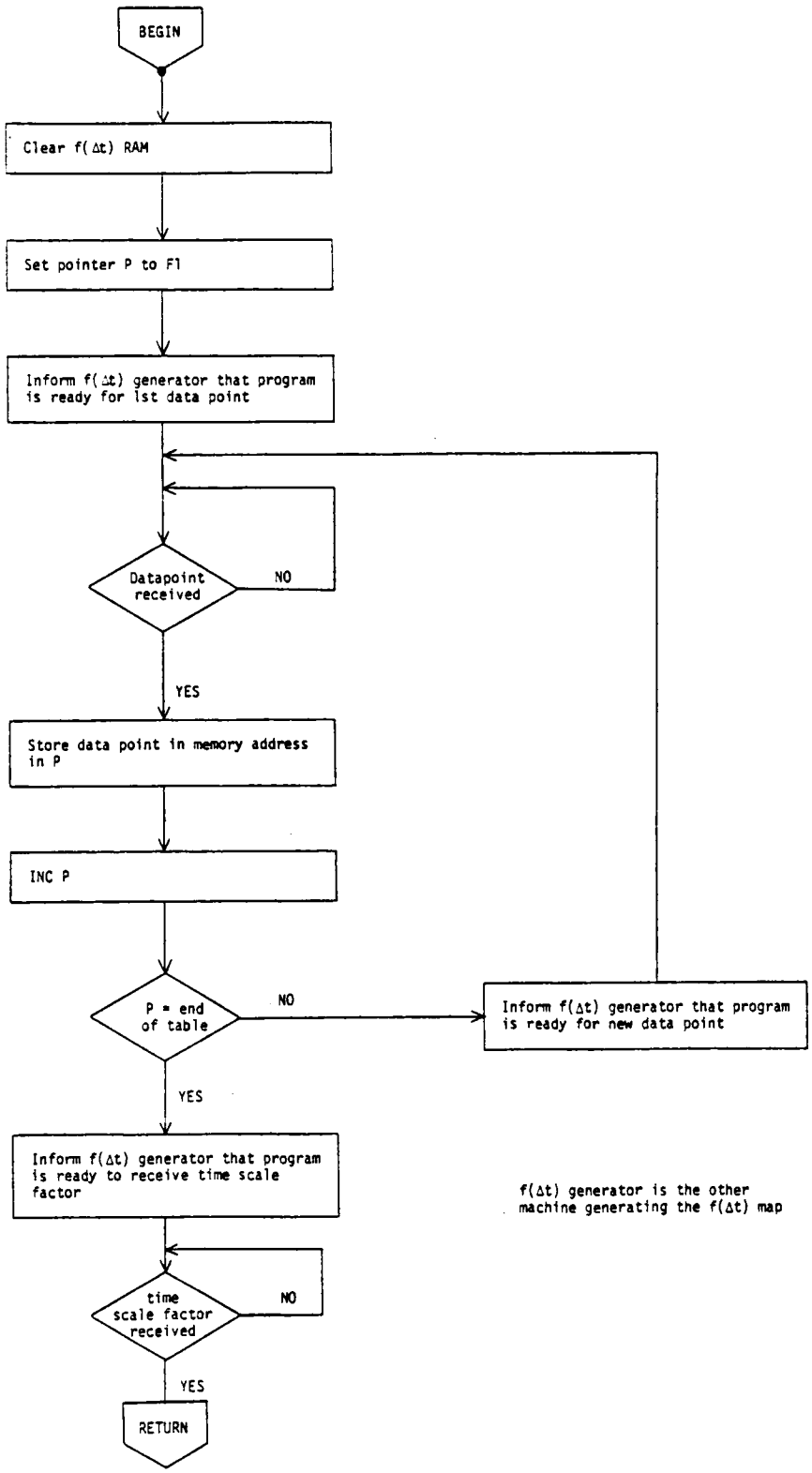
Note that INIT uses a short subroutine, AP (Array Pointer). This program takes the numbers "i" and "j" which identify the array element "ij" and returns the address for the corresponding element in the memory. For example if $C_1$ is address $0200_{16}$, then the address of array element 23 is $020B_{16}$. Subroutine AP is used throughout the program.

Subprogram $\Delta T$ is another of the start-up subprograms. It searches the $f(\Delta t)$ map to find the addresses (corresponding to $\Delta t$ values) at which the $|f(\Delta t)|$ falls below a preset minimum $\varepsilon$ where



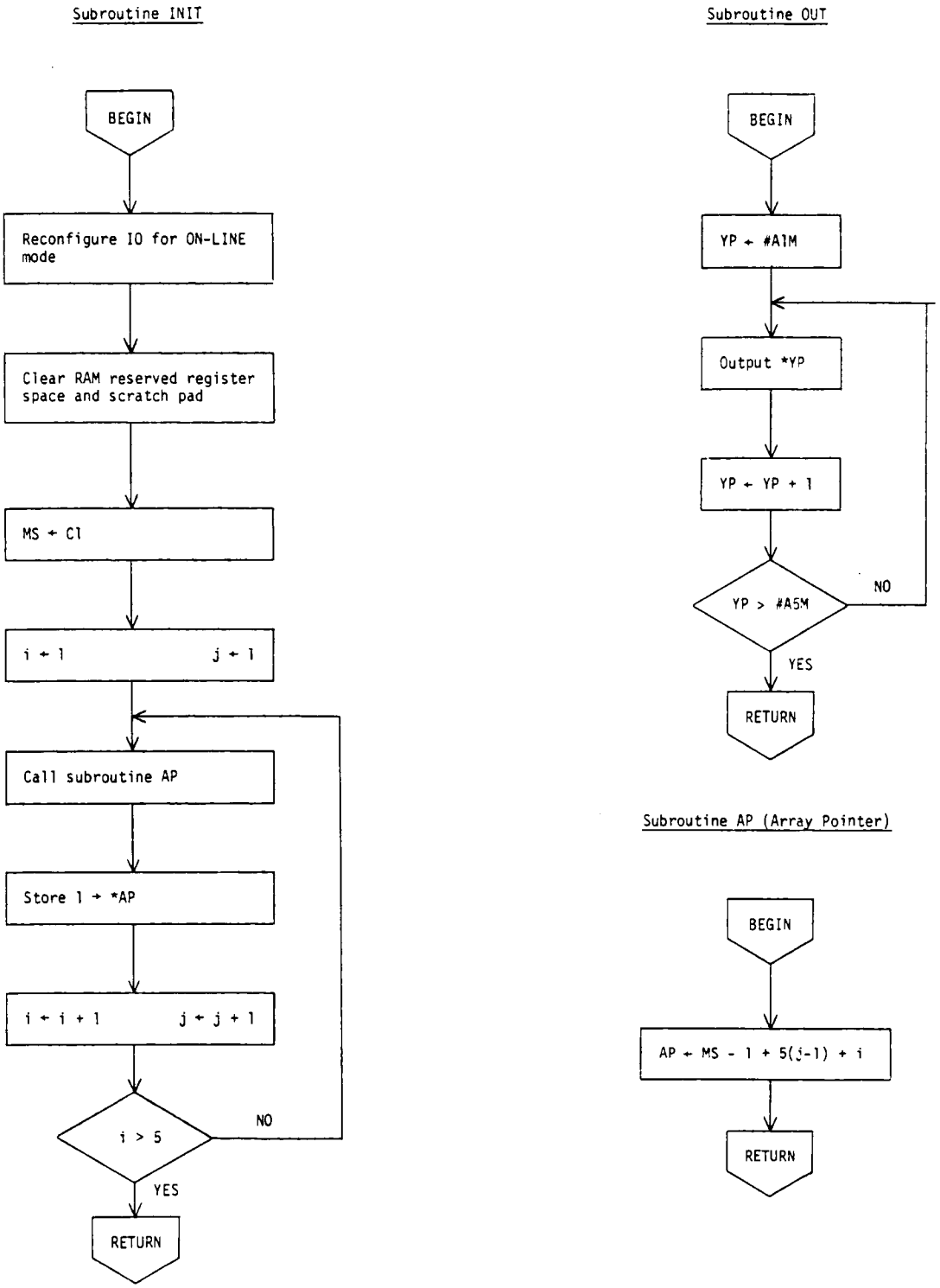

Subprogram MPI

Figure V-8. Pulse Pile-Up Corrector Flow Chart, Functional Subprograms.



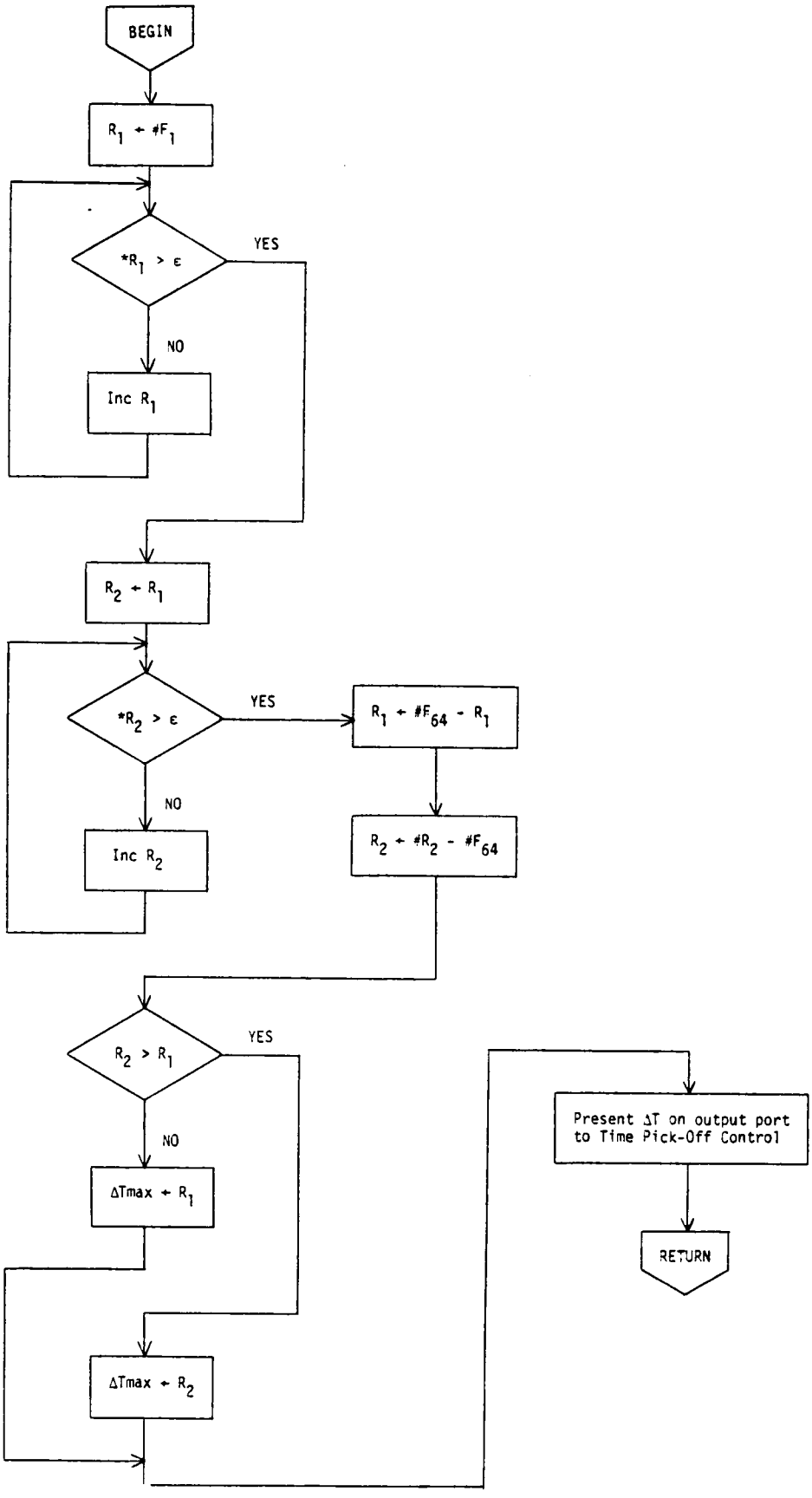

Subprogram MAP

Figure V-8. (Continued)



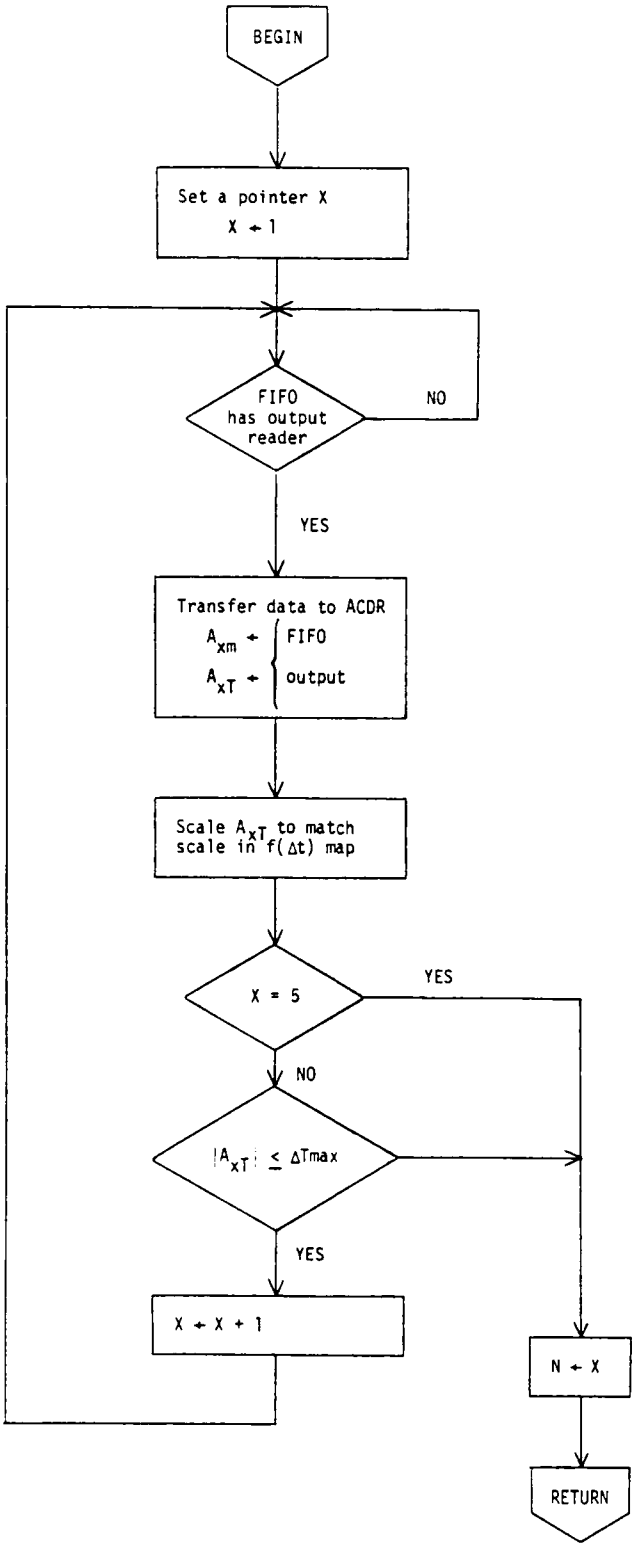

Subprogram INIT

Figure V-8. (Continued)



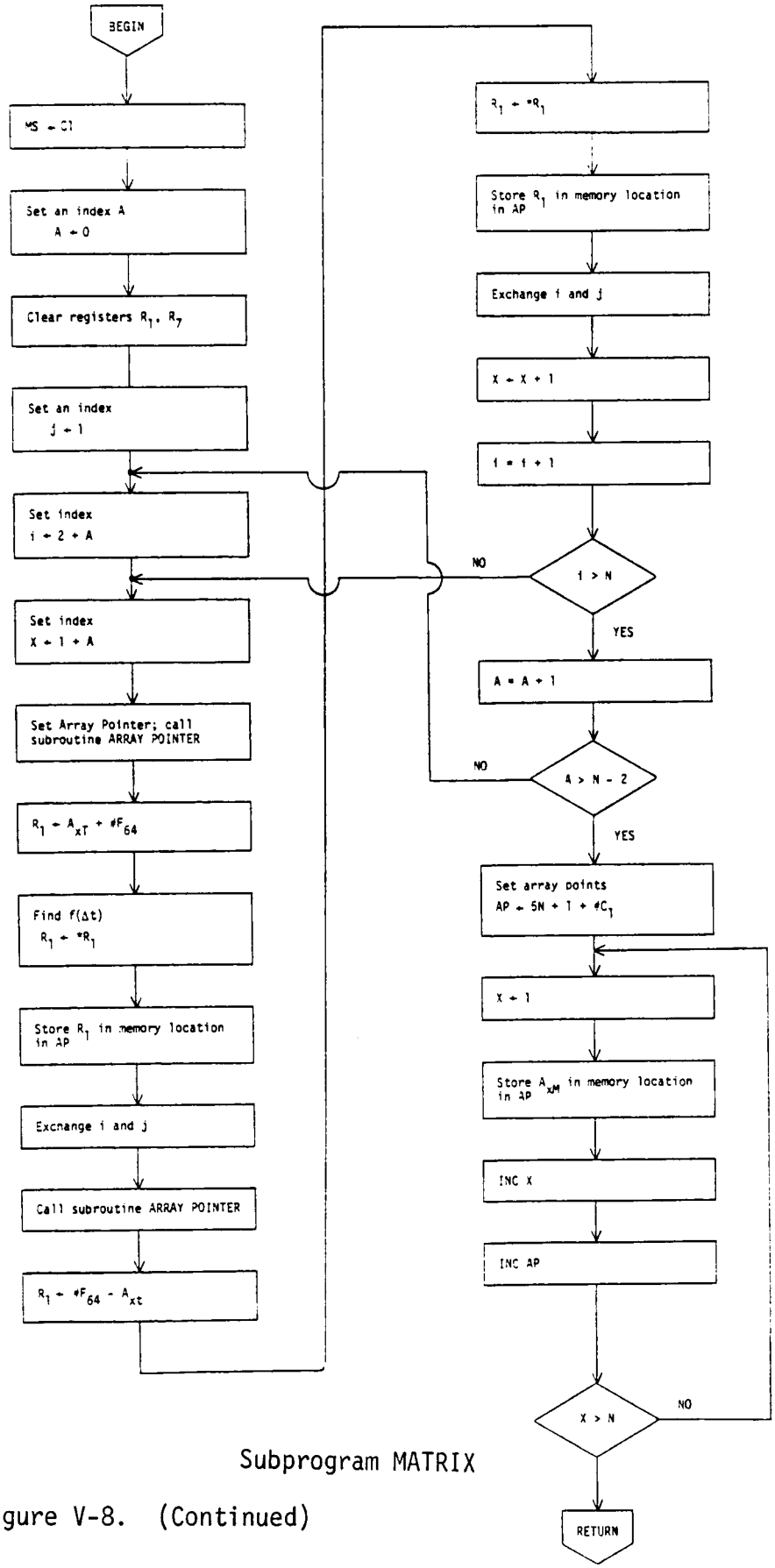

Subprogram $\Delta T$

Figure V-8. (Continued)



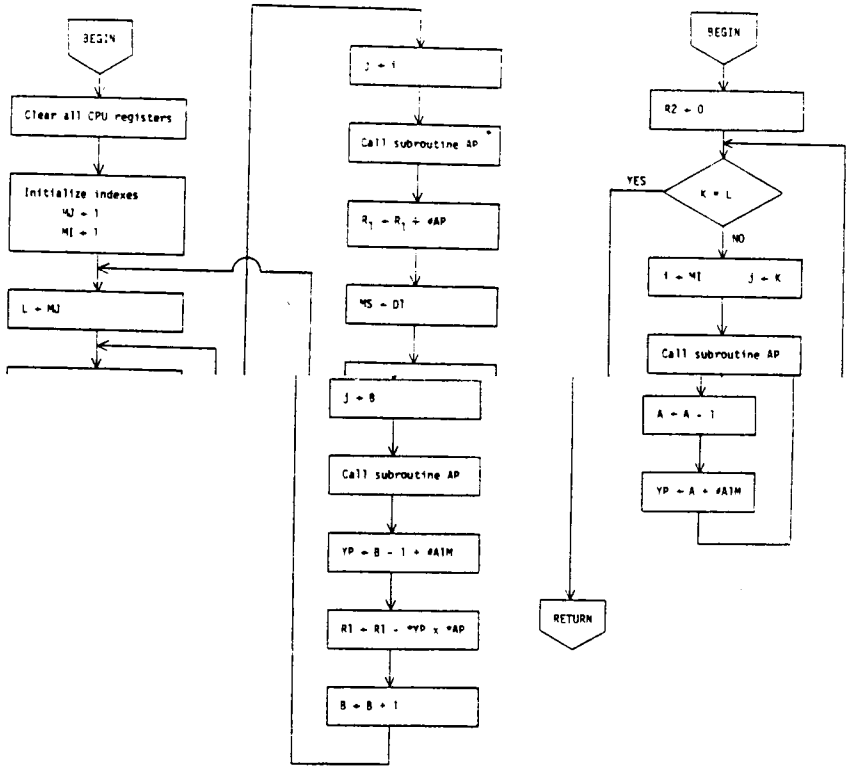

Subprogram READ

Figure V-8. (Continued)




Subprogram MATRIX

Figure V-8. (Continued)




Subprogram SOLVE

Figure V-8. (Continued)

pile-up interference is negligible. In operation the program steps through the table to find the upper and lower points at which $|f(\Delta t)|$ is less than $\varepsilon$. The difference between the address of these points and the address of the center point (at which $f(\Delta t)=1$) is a number corresponding to a $\Delta t$ value. The program then selects the larger of these two numbers as the $\Delta t$ in which pile-up may exist. This value is stored at memory address $\Delta T_{max}$. It is also presented to the Time Pick-off Control.

The next program, READ, controls the transfer of a pulse burst from the Input Buffer (FIFO) to the Active Data Register in the RAM. The Buffer presents two memory words, one containing magnitude and one containing time data. The two words are addressed at two successive memory location. The subprogram reads both data words and transfers the magnitude to location $A_{xm}$ and the time to location $A_{xt}$. After it reads a data pulse it checks the time data to see if it is less than $\Delta T_{max}$. If the test is true the program knows there is a following pulse in the pulse burst. If the test is false then the read operation stops. READ also counts the number of pulses it reads and stores that number at location N.

Before describing subprogram matrix, it is necessary to rewrite Equation II-20. This is required to accommodate receiving time data as pulse spacing instead of time of occurrence as noted earlier. To begin first note from Figure II-5, page 15, that the following can be written:

$$\Delta t_1 = t_2 - t_1 \qquad \Delta t_3 = t_4 - t_3$$

$$\Delta t_2 = t_3 - t_2 \qquad \Delta t_4 = t_5 - t_4$$

Then Equation II-20 is rewritten as:

$$\begin{vmatrix} 1 & f(-\Delta t_1) & f(-\Delta t_1-\Delta t_2) & f(-\Delta t_1-\Delta t_2-\Delta t_3) & f(-\Delta t_1-\Delta t_2-\Delta t_3-\Delta t_4) \\ f(\Delta t_1) & 1 & f(-\Delta t_2) & f(\Delta t_2-\Delta t_3) & f(-\Delta t_2-\Delta t_3-\Delta t_4) \\ f(\Delta t_1+\Delta t_2) & f(\Delta t_2) & 1 & f(-\Delta t_3) & f(-\Delta t_3-\Delta t_4) \\ f(\Delta t_1+\Delta t_2+\Delta t_3) & f(\Delta t_2+\Delta t_3) & f(\Delta t_3) & 1 & f(-\Delta t_4) \\ f(\Delta t_1+\Delta t_2+\Delta t_3+\Delta t_4) & f(\Delta t_2+\Delta t_3+\Delta t_4) & f(\Delta t_3+\Delta t_4) & f(\Delta t_4) & 1 \end{vmatrix} \quad |Y| = |Y'| \qquad \text{V-3}$$

Now note that the argument for each element, ij, in the upper triangular matrix is the negative of the argument for the element, ji, in the lower triangular matrix.

Subprogram MATRIX builds the coefficient matrix of Equation V-3 and stores it in memory beginning at location $C_1$. The program indexes through elements in the lower triangular matrix and at each element enters the $f(\Delta t)$ map to return the mapped value for storage in the array. Then it negates the argument, interchanges i and j and maps through $f(\Delta t)$ for the corresponding value in the upper triangular matrix. If there are n overlapping pulses (number stored at N) then an nXn array is generated. After building the nXn coefficient array, the program then transfers the magnitude data $A_{xm}$ into the n+1 column of the coefficient matrix. This locates both the $[f(\Delta t)]$ and [Y'] matrices of Equation V-3 in proper position for solution of the set of linear equations by Cholesky's method (see Appendix B).

Subprogram SOLVE is a direct implementation of Cholesky's method for solving a set of simultaneous linear equations. The program

handles from two to five equations (the limitation to five is not in the subprogram; it is caused by the limit on reserved space for array storage). The program first constructs the decomposed matrix, [L/T¦K], from the augmented coefficient matrix, [f(Δt)¦Y'], then using the T and K elements calculates the solution values, the Y values. Finally, this subroutine stores these corrected pulse heights in the magnitude column of the Active Data Register.

The final functional subprogram is OUT. This program outputs the n corrected pulse heights.

# CHAPTER VI

## DESIGN OF THE PULSE PILE-UP CORRECTOR, PHASE 2

This chapter completes the final phase of development of the Pulse Pile-Up Corrector concept. The universal hardware/software design of Chapter V was used as the foundation to design a real instrument. A primary goal of this design effort was to have an instrument that would be useful in a typical pulse-height-analyzer system.

The first step was an examination of typical analysis systems. (Vendor literature was the source for the needed information, primarily Ortec Incorporated and Canberra Industries product catalogs.) Recalling that the channel number of an MCA is related to the number of data lines by:

$$2^N = \text{channel number} \qquad \text{VI-1}$$

where N is the number of lines, typical channel capacities range from 256 to 8192 with the lower numbers appearing most frequently. Note that this corresponds to 8 to 13 data lines. For this design an 11-line data word supplying a 2048-channel analyzer was chosen. This size MCA appeared frequently enough in the catalog study to be considered typical.

Choice of an 11-bit data word eliminates use of 8-bit microprocessors. It was thus necessary to select a 16-bit microprocessor.

Typical pulse data rates were the next consideration. Performance of the ADCs used in these analysis systems was taken as a guide to pulse rate performance. Some examples of ADC conversion time are:

| | | |
|---|---|---|
| Ortec model 800: | 8192-channel: | (3+0.01N)μs |
| | 4096- : | (1.75+0.01N)μs |
| | 2048- : | (1.25+0.01N)μs |
| | 1024- : | (0.75+0.01N)μs |
| Canberra model: | 8192 : | (2.5+0.01N)μs |
| | 8060 : | (3.5+0.01N)μs |

The number N is the resultant channel number of the data. For a 2048-channel system the conversion times for several N are:

| N | Ortec 800 | Canberra 8192 |
|---|---|---|
| 100 | 2.25μs | 3.5μs |
| 500 | 6.25 | 8.5 |
| 1500 | 16.25 | 17.5 |
| 2000 | 21.25 | 22.5 |

From these times it can be seen that the minimum spacing between pulses is on the order of a microsecond. Thus the Input Buffer's FIFOs can capture any pulse that can be digitized by the ADCs. A pulse spacing of, for example, 15μs corresponds to a count rate of $6.7 \times 10^4$ counts/second. Ortec specifies the count rate stability for many of its pulse amplifiers at a maximum count rate of $5 \times 10^4$ counts/second. It would be expected that the actual operating count rate would be lower. Assuming an operating count rate of $5 \times 10^3$ as typical seems reasonable based on amplifier and ADC performance.

A data structure that has a count rate of $5 \times 10^3$ counts/second and experiences 10 percent pile-up would have a piled-up pulse burst rate of:

| Pulses Per Burst | Pulse Burst Rate |
|---|---|
| 2 | 250 bursts/second |
| 3 | 167 |
| 4 | 125 |

If 250 bursts/second is the performance requirement then the required time to process on pulse burst, $T_{PM}$, is 1/250 or 4 milliseconds. This

number is in the range of possibility for a real microprocessor based instrument.

At this point is has been decided:

1. To design for an 11-bit data word corresponding to a 2048-channel MCA.

2. To use a 16-bit microprocessor.

3. That the microprocessor operating time and the pulse analysis system time requirements are of the same order of magnitude.

## Hardware Design

The hardware design for the Pile-Up Corrector is based upon the concepts presented in Chapter V. The goal of this design study was to develop a realizable instrument design. To avoid spending a lot of effort in designing a computer and to improve the odds for success it was decided to base the design on a manufactured single board computer based on a single-chip microprocessor. These computer modules are available both from the microprocessor manufacturers and from other vendors such as Pro-Log Corporation. It is recognized that this is probably not the approach that would be taken by an Original Equipment Manufacturer (OEM) in the pulse instrument field. However, the work here could be readily transferred by an OEM into a design that conformed to the NIM or CAMAC standards if appropriate. The OEM would have to design his own circuit boards using the microprocessor family chip set.

Selection of the microprocessor family begins with two points (a) it should be a 16-bit device, and (b) the algorithm requires multiplication. Since speed is important it was required that the multiplication

be accomplished by either being included as one of the instructions, or by use of a separate hardware multiplier.

The microprocessor. There are now several 16-bit microprocessor chips or chip sets on the market. They include the DEC LSI 11, the TMS 9900, the Intel 8086, the Motorola 68000, and the Zilog Z8000. The Texas Instruments Incorporated TMS 9900 was chosen for this design. This was primarily because complete documentation for the TMS 9900 family was readily available to this author. Also it was believed that the 9900's memory-to-memory architecture would be useful in the book-keeping involved in building and using the arrays required by the algorithm. Also the instruction set included a multiply function.

Texas Instruments offers the 9900 microprocessor in single board modules as the TM 990 series Micro-Computer modules. For the Pulse Pile-Up Corrector the following was required:

1. TM 990/100M Micro-Computer; 1K-word ROM; 512-word RAM; EIA-RS-232-C serial I/O.
2. TM990/512 Universal prototyping card.
3. TM990/510 OEM card cage.
4. Power supply.

The prototyping card is for the input and output buffers. These have to be specially built for the Pile-Up Corrector. (The prototyping card is for the first, or prototype, instrument. Also for prototyping, EPROM would be preferred to ROM.)

The TM 990/100M is shown in block diagram form in Figure VI-1. Memory on the board consists of 1K word of ROM (or EPROM) for program memory and 512 words RAM for working memory and registers. There is one TMS 9901 Programmable Systems Interface for parallel I/O and one

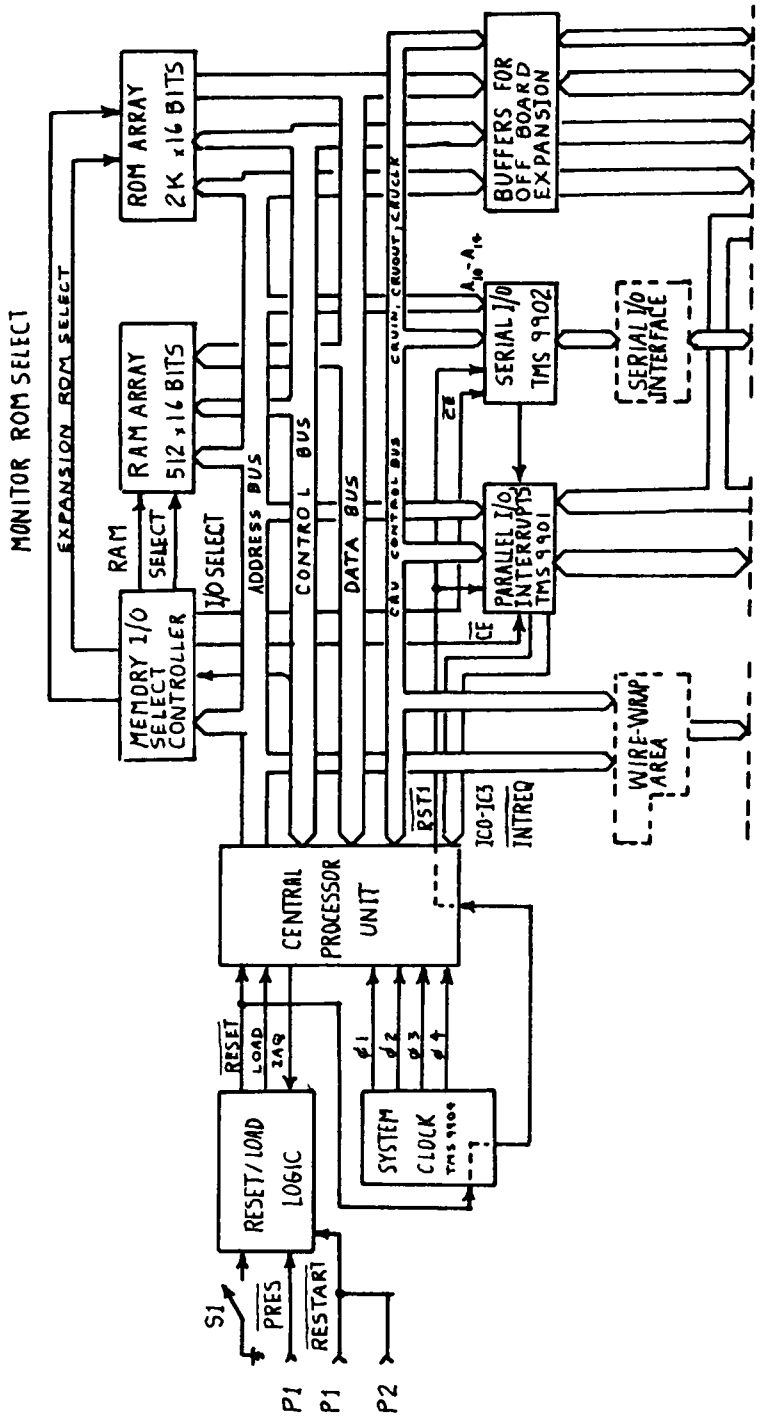



Figure VI-1. The Texas Instruments Incorporated TM 990/100M Micro-Computer Module. A single-board micro-computer based on the TMS 9900 Microprocessor.

TMS 9902 Asynchronous Communications Controller for serial I/O. The card connectors are:

P1: Bus connector.

P2: RS-232 serial connector.

P3: Parallel I/O connector.

P4: Spare connector for on-board prototyping area.

Reference should be made to the Texas Instruments 9900 Family Systems Design and Data Book and the TM 990/100M instruction manuals for a full description of this microcomputer.

Input and output buffers. The input buffer is a special requirement for the Pulse Pile-Up Corrector, so a design for this item was prepared. The output buffer was originally considered to be one of the external instruments and not a part of the Corrector. However, during the design work it was realized that the output buffer was very similar to the input buffer and was a natural fit into the instrument design so it was included. The input and output buffer designs were carried through to completed logic diagrams.

Figure VI-2 shows the Input Buffer Circuit. There are two halves to the circuit, the Time FIFO half and the Data FIFO half. They are identical except for size of the FIFO. Control of the FIFO input is done by the Time Pick-Off Control Instrument. When it has a new pulse consisting of magnitude data and time pulse it samples the FIFO's Input Ready (IR) line. If both IRs are high the control loads the data to the FIFO's data input port I. It then transfers the data by sending a Shift In (SI) signal to the FIFO. The output side of the input buffer is controlled by the TM 990/100M. Memory mapped I/O is used for the

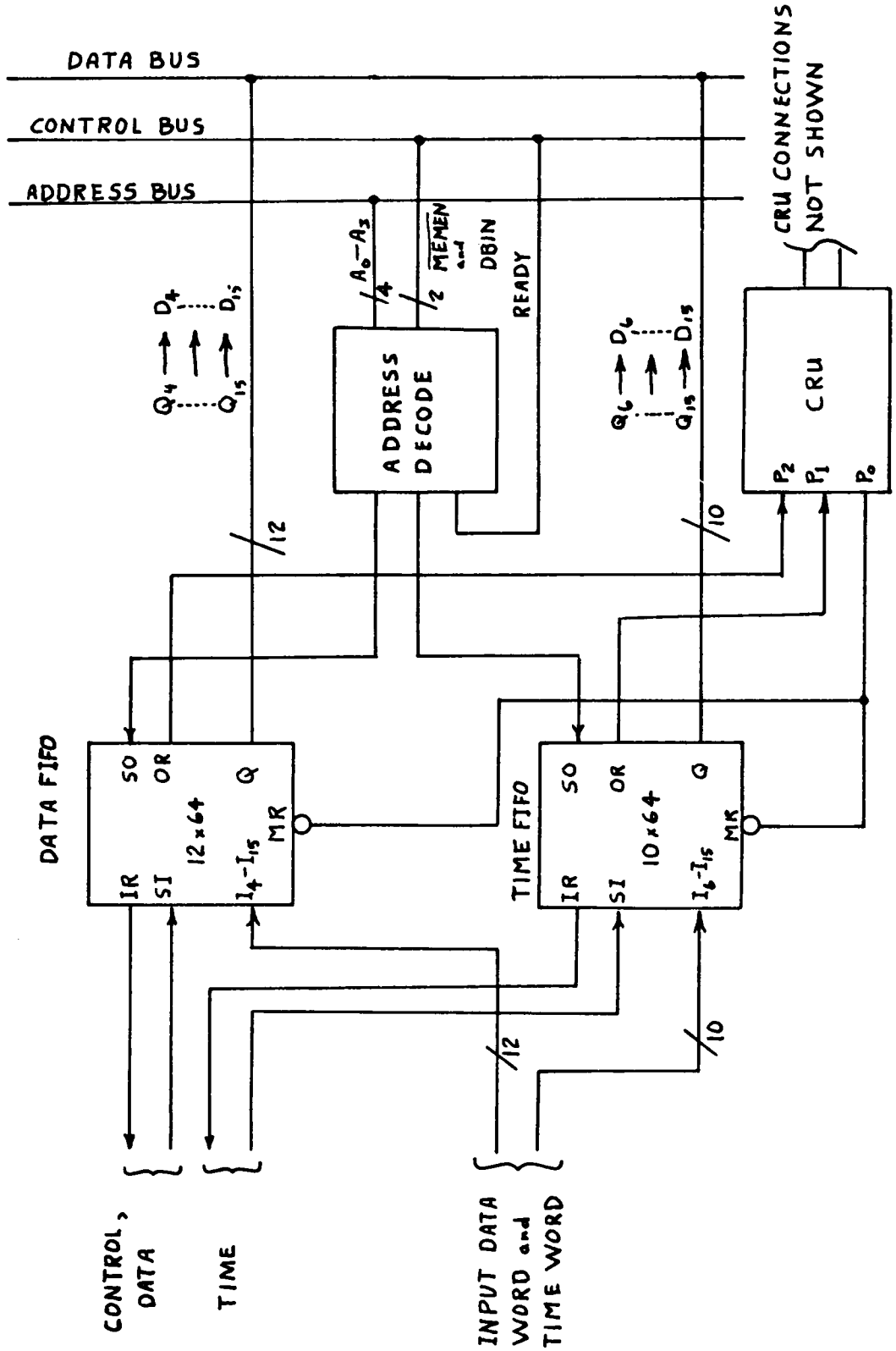



Figure VI-2. Input Buffer Circuit.

input buffer (and the output buffer), so address decoding has to be provided. There is a large block of unused memory addresses available. This is used to advantage by selecting buffer addresses that require only 4 of the 16 address bits available. The selected addresses are:

$CXXX_{16}$.....data or magnitude address

$DXXX_{16}$.....time address

where the Xs indicate "don't cares." An address decode circuit was then designed (Figure VI-3). It generates the Data Shift Out (SO), Time SO, and READY signal. The READY signal is generated when either SO line is active, and it prevents the microprocessor from going into a wait cycle during a read of the input buffer.

In operation the computer samples the input buffer Output Ready (OR) signals via the TMS 9901. If the OR are high and an input buffer is desired, then the program sets the proper address to select first Data FIFO and then Time FIFO. (It must always read a data/time pair to keep the data aligned with time in the two FIFOs.) When the address is valid and DBIN and $\overline{\text{MEMEN}}$ are active (see Figures VI-3 and VI-4) then the SO signal is set to 1. The data is held valid at the FIFO as long as SO is held high. The computer samples the data on phase one of clock cycle two, and at the end of the cycle sets DBIN and $\overline{\text{MEMEN}}$ inactive and may change the address. This causes the SO to go low and the FIFO discards the data on its output register and brings the next data to the register. Designing the input buffer control in this way allows the data to be read off the input buffer as if it were just another location in memory.

The output buffer is similar to the input buffer. It is shown in Figure VI-5. The SO and OR and Q lines are under control of the receiving device which directs reinsertion of the corrected pulse data

Address Assignment

date SO ——— $C000_{16}$

date SO ——— $D000_{16}$

Decode Equations:

data SO = $A_0 \cdot A_1 \cdot \overline{A_2} \cdot \overline{A_3} \cdot DBIN \cdot \overline{MEMEN}$

time SO = $A_0 \cdot A_1 \cdot \overline{A_2} \cdot A_3 \cdot DBIN \cdot MEMEN$

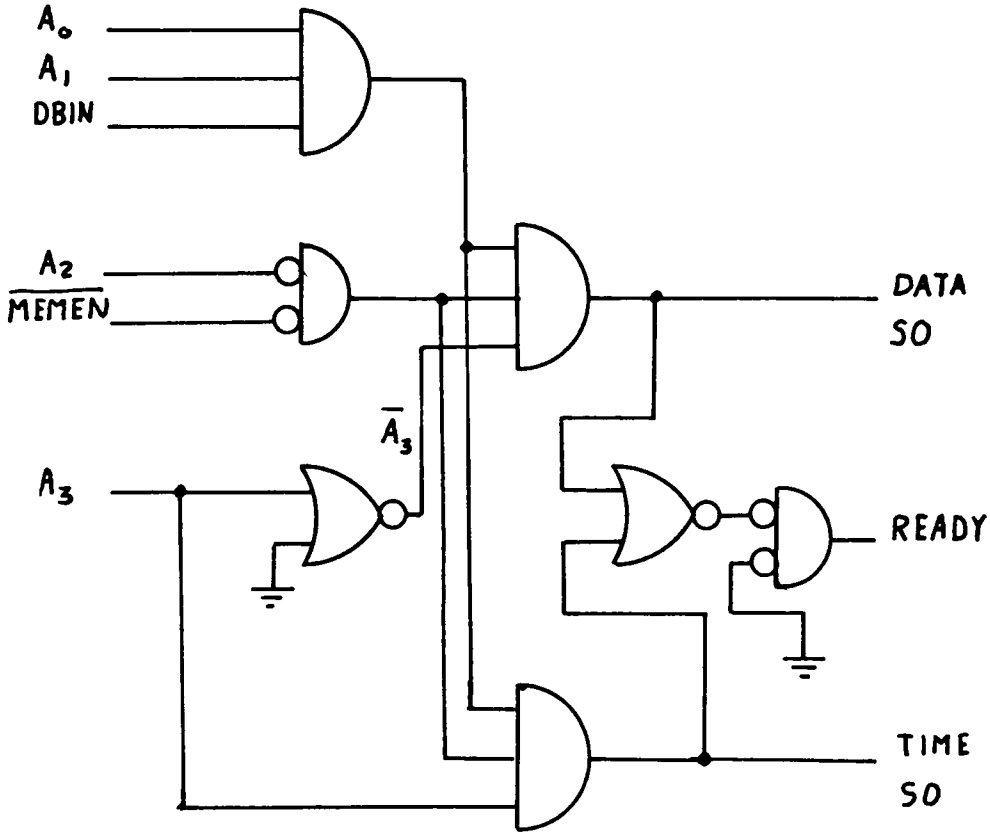



Figure VI-3. Input Buffer Address Decoder. Decoder Circuit is Constructed Using an SN74LS11 Triple Three Input and an SN74LS02 Quad Two Input NCR.

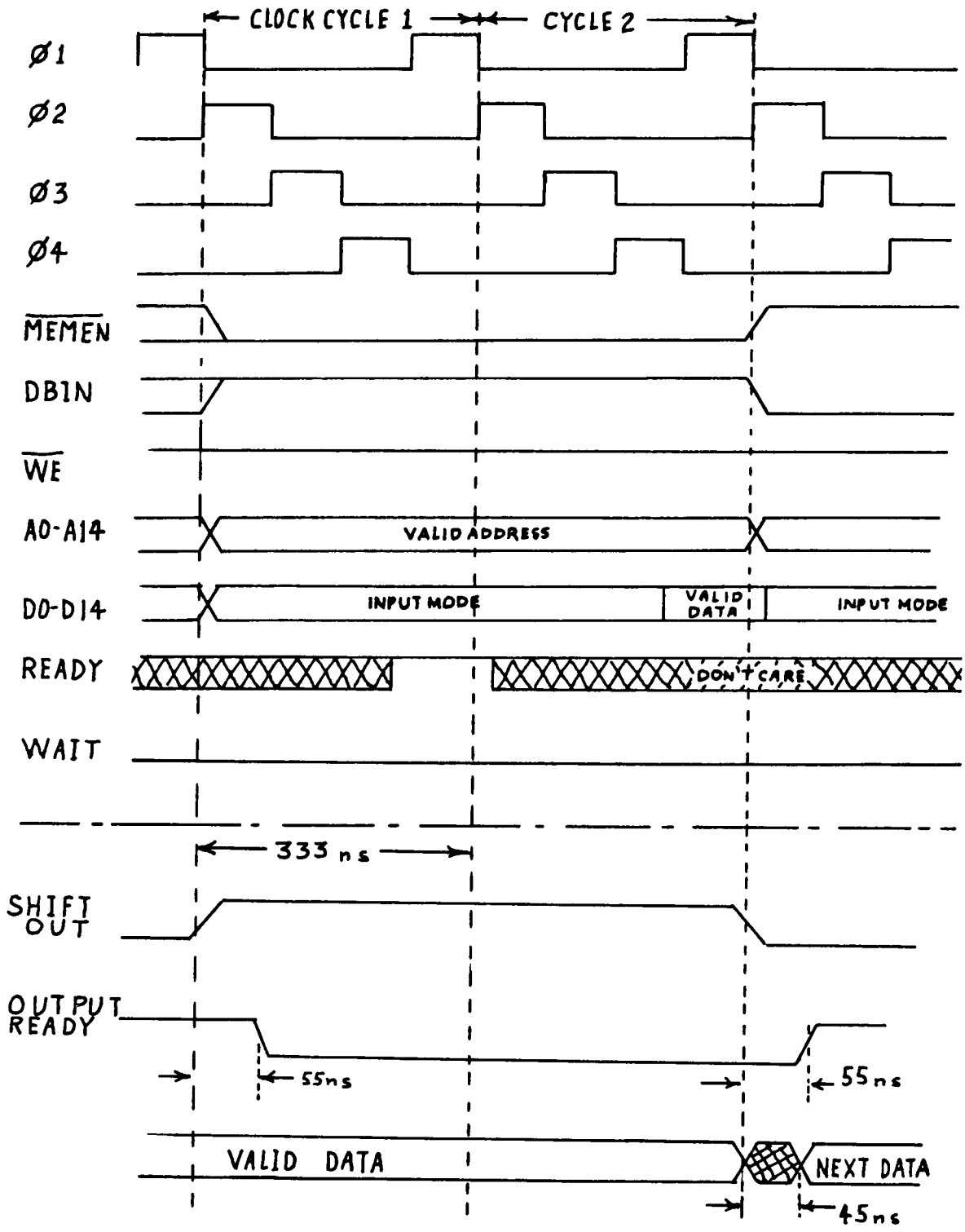



Figure VI-4. Timing Diagram for Input Buffer Read Cycle. TI-9900 information is above the dashed line, and FIFO information is below the dashed line.

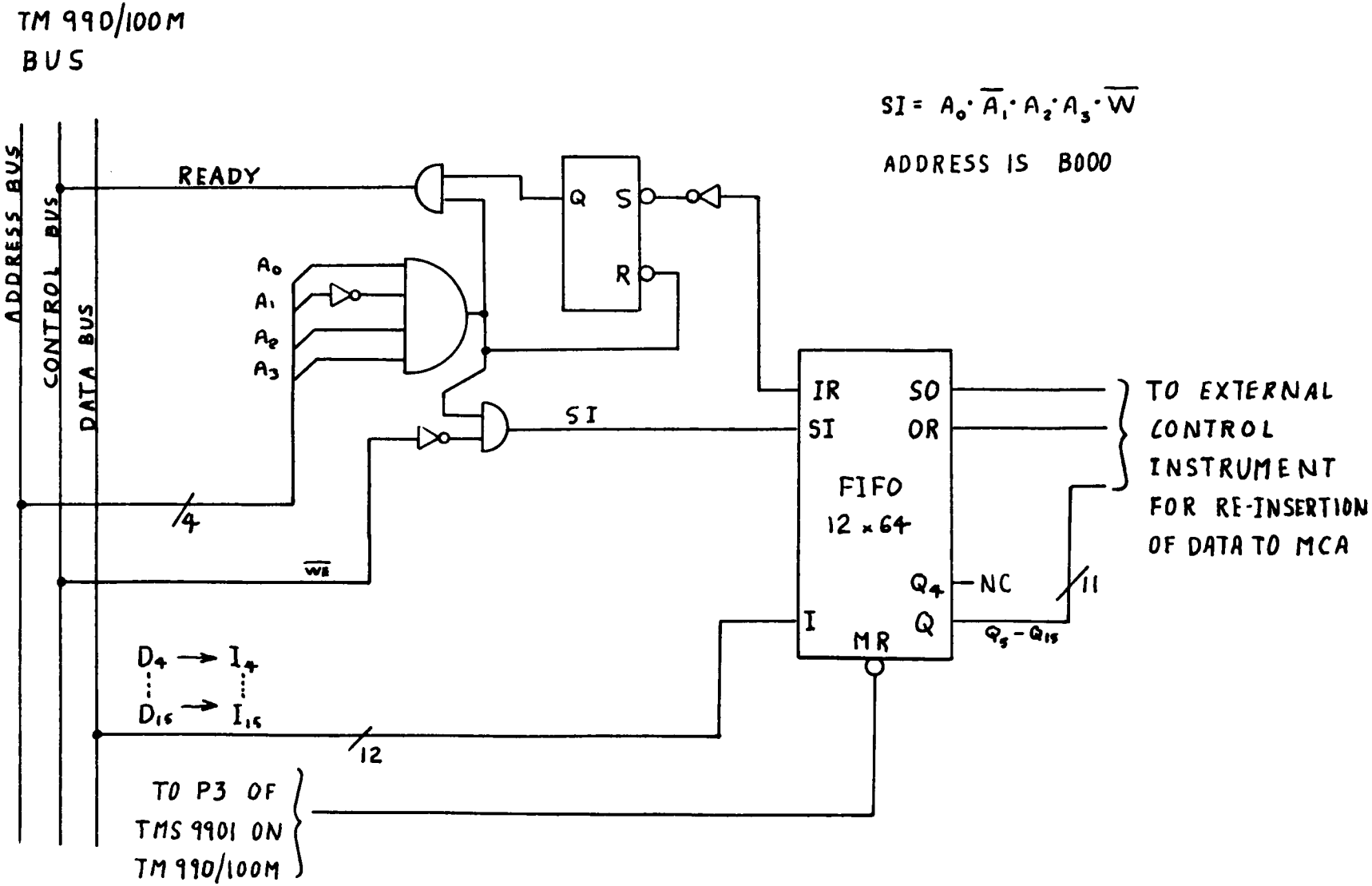



Figure VI-5. Output Buffer Circuit.

into the main data stream to the MCA. The input side is under control of the TM 990/100M. The computer's access to the output buffer is via address $CXXX_{16}$. The address decode circuit is also shown in Figure VI-5. The buffer is configured as an always open port to the computer provided the FIFO input register is empty. IR is used as a control and check signal. Because the FIFO is fast, IR would normally be high when the buffer is accessed. Addressing the output buffer allows IR to set a latch which sets the READY signal high, preventing the computer from entering a wait state (see Figure VI-6 for timing). When the 9900 $\overline{WE}$ goes active low (indicating the 9900 is writing to memory) the FIFO SI is set high and the data on the bus is written into the FIFO. When $\overline{WE}$ goes inactive SI goes low and the write cycle is complete. Figure VI-7 shows the timing that occurs if IR happens to be low when SI goes high. The memory write goes into a wait state until IR goes high and is able to latch the READY signal.

The FIFO circuits are paralleled sets of Monolithic Memories 67401A 4x64 FIFO's or 67402A 5x64 FIFO's. Figure VI-8 is the circuit for the input buffer Data FIFO and the output buffer FIFO. Figure VI-9 is the FIFO for the input buffer Time FIFO. The Data FIFO is a 12-bit structure where only 11 bits are needed. The extra bit is the MSB of the FIFO. The I4 bit of the input buffer should be held at a logic 0. The Q4 bit on the output buffer can be left floating.

The Time FIFO is shown as a 10-bit device. The design assumes that the Time Pick-Off Control functions as a time counter and that it can count up to $2^{10}$ counts. The requirements on the time counter are that it have a count rate fast enough that the minimum measurable time interval is small (less than 1/128 with the $F(\Delta t)$ map used) relative to

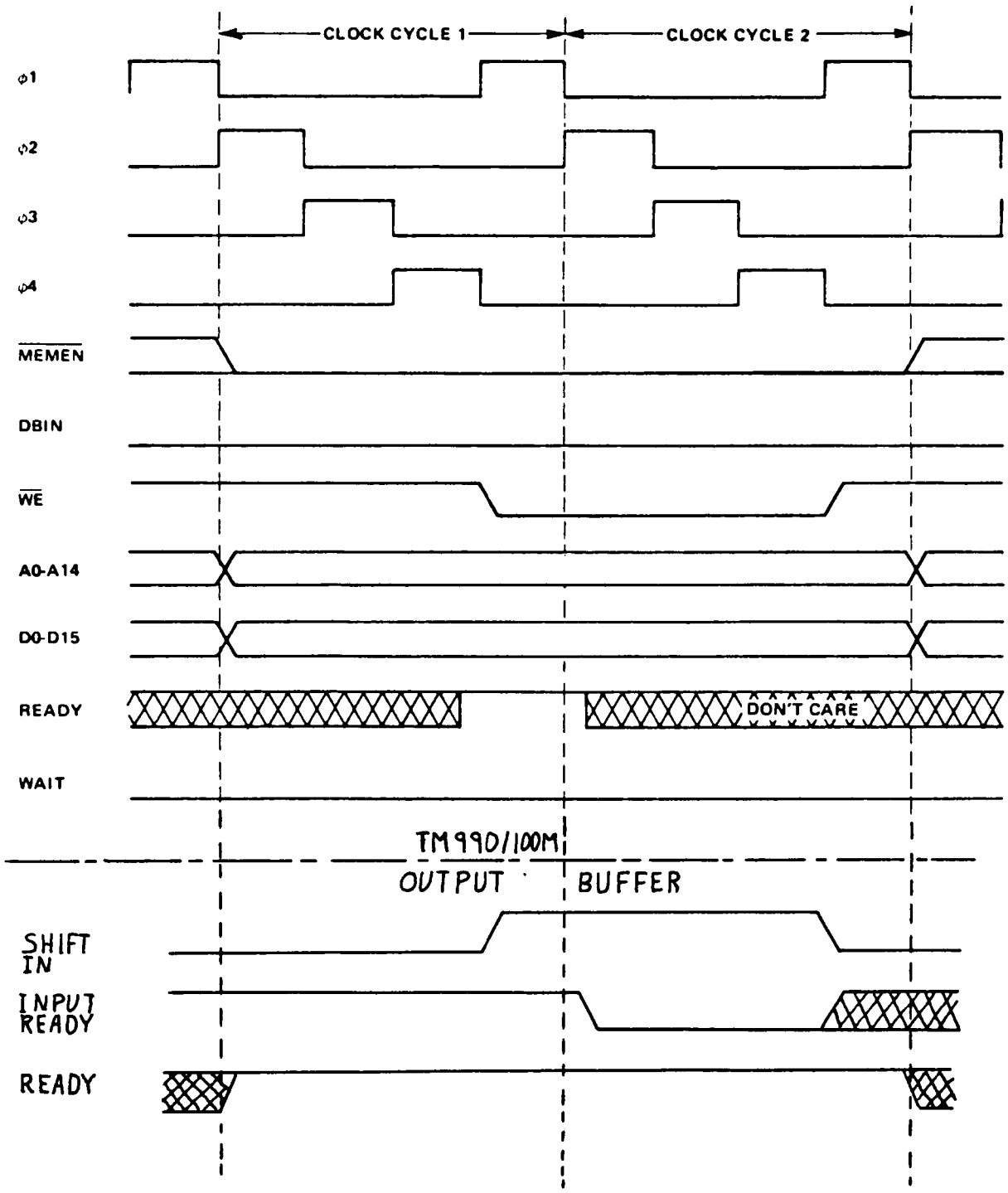



Figure VI-6. Timing Diagram for Output Buffer Write Cycle, No Wait Status.

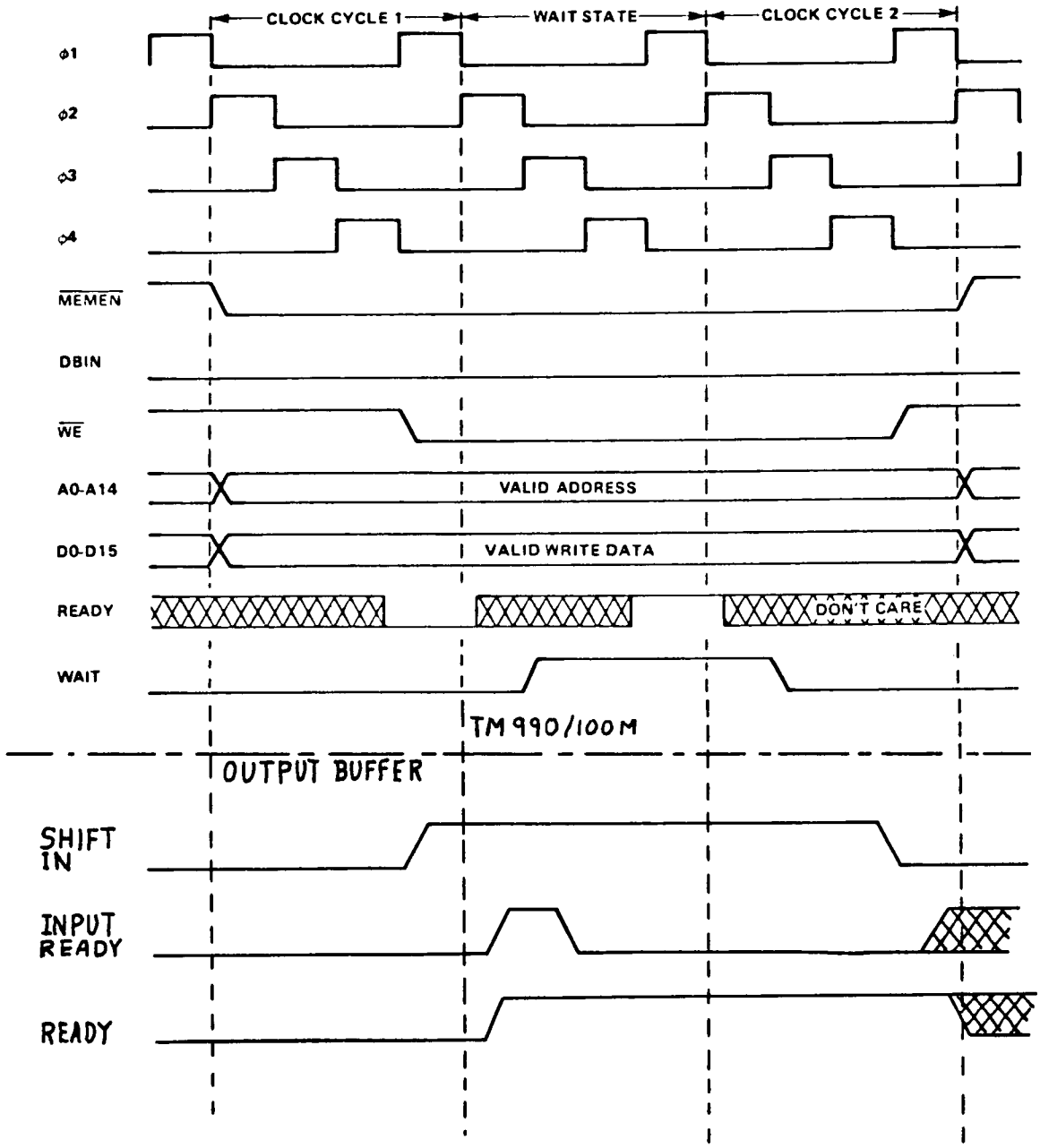



Figure VI-7. Timing Diagram for Output Buffer Write Cycle, One Wait State.

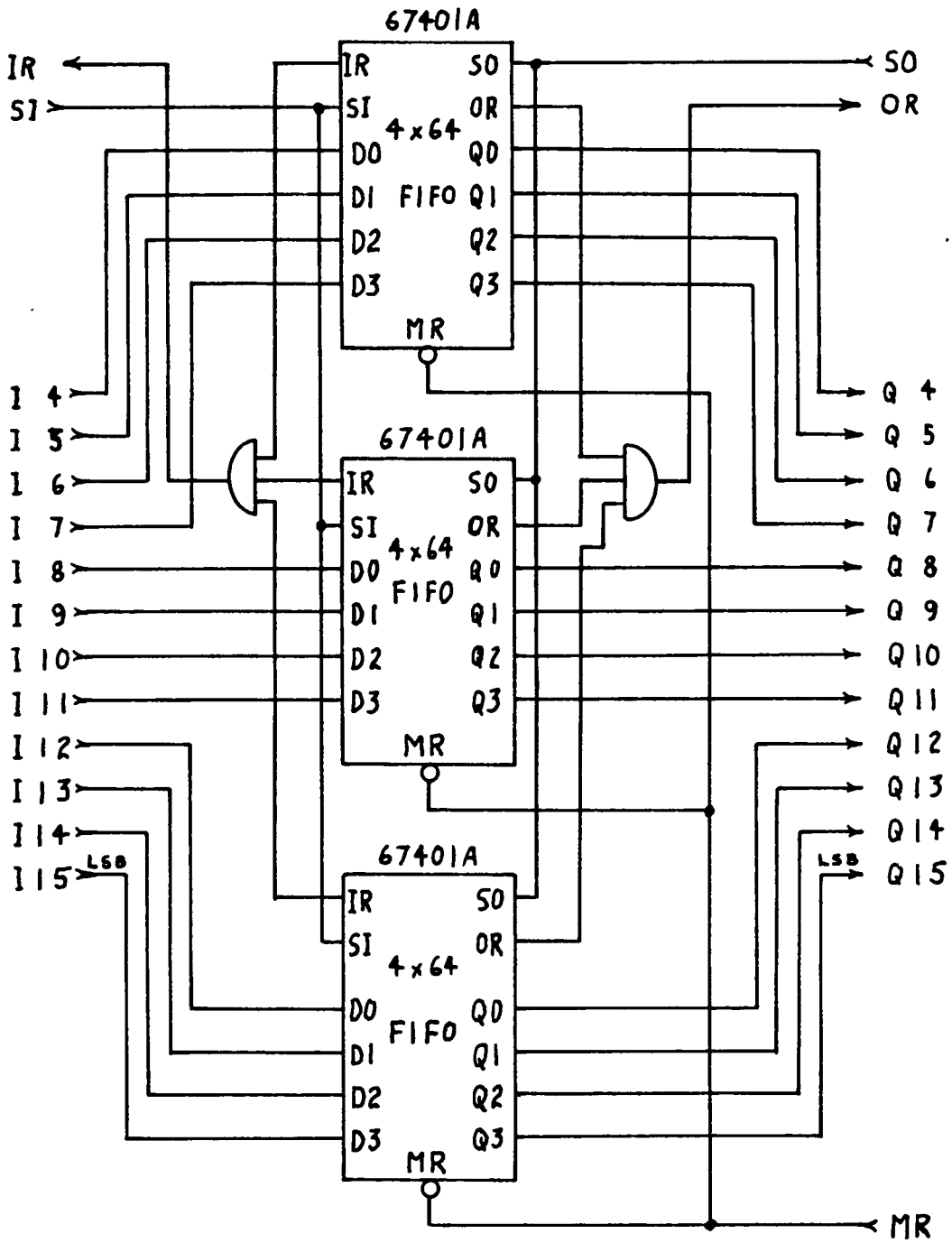



Figure VI-8. Data FIFO Circuit. Circuit is constructed using two Monolithic Memories 67401A FIFO's in a parallel arrangement.

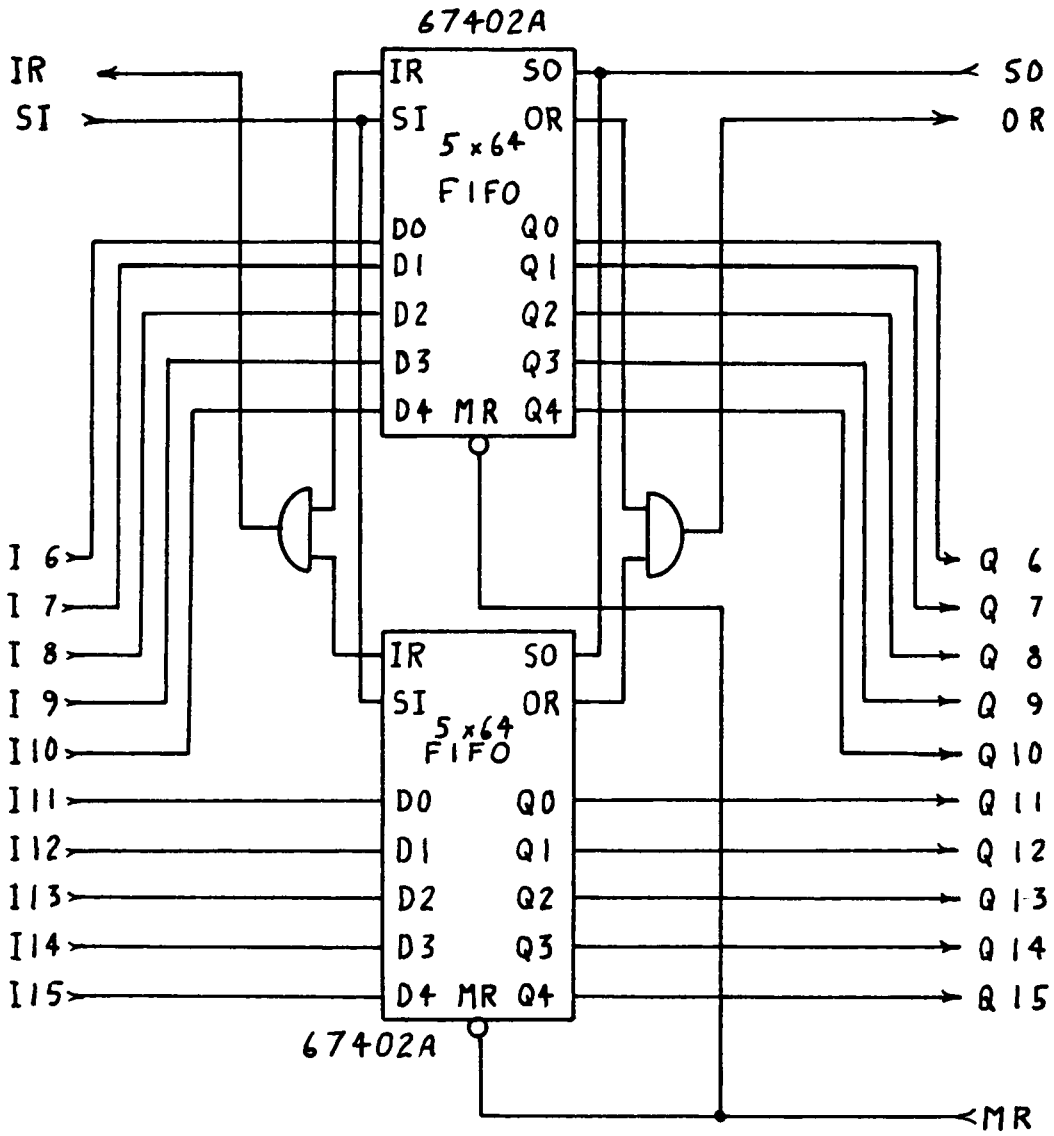



Figure VI-9. Time FIFO Circuit. Circuit is constructed using three Monolithic Memories 67402A FIFO's in a parallel arrangement.

the pulse waveshape. The timer should be able to record a number somewhat larger than the maximum time separation in which pile-up may occur. The 10-bit timer provides a 1024:1 maximum-to-miminum time ratio.

This hardware design will be a good base for developing a prototype instrument. It has all the needed performance requirements and it is well supported by the manufacture with development systems.

## The Software

The software design was a direct application of the generalized software developed in Chapter V. Only a few deviations were made and these generally to match the characteristics of the TMS 9900. As an example the 9900 addressed bytes not words. Consequently two successive words in memory differ in address by 2 instead of 1. This required considerable care in adjusting indexing.

A point that is perhaps a cross between software and hardware interests is the I/O pin assignments for the TMS 9901 Programmable Systems Interface. These connections are shown in the circuit diagrams and for convenience they are also listed in Table VI-1.

Data representation. At this point a discussion of the data structure representation is in order. The algorithm will manipulate numbers that come from either the input false magnitudes, $Y'_n$, or from the normalized $f(\Delta t)$ map. The 11-bit $Y'_n$ has a range of $0_{16}$--$FF_{16}$; $f(\Delta t)$ varies from 0-1. With floating point arithmetic this would be no problem. Floating point arithmetic would, however, have to be done in software introducing a processing speed penalty. So it was decided to avoid floating point work.

TABLE VI-1

TMS 9901 PROGRAMMABLE SYSTEMS INTERFACE PIN ASSIGNMENTS

| Pin Signature | Connection | Function | Use |
|---|---|---|---|
| P0 | Input Buffer MR | Output | Master Reset |
| P1 | Input Buffer Time OR | Input | Output Ready |
| P2 | Input Buffer Data OR | Input | Output Ready |
| P3 | Output Buffer MR | Output | Master Reset |
| P4 | Ready for Data | Input | Handshake with Time |
| P5 | Data Ready to Send | Output | Pick-Off Control |
| P6 | Time Pick-Off Control Input | Output | New Max T Count for the Time Pick-Off Control LSB |
| P7 | | | |
| P8 | | | |
| P9 | | | |
| P10 | | | |
| P11 | | | |
| P12 | | | |
| P13 | | | |
| P14 | | | |
| P15 | | | MSB |

The $Y_n'$ numbers consist of at most the 11 LSBs of the 16-bit computer data word. If the $f(\Delta t)$ uses only the first bit as it varies from 0-1, then large round-off errors can be expected. An improvement is made by assuming the decimal (binary) point is between bits 5 and 6 of the computer word instead of to the right of the LSB, bit 15. This has the effect of mathematically changing the spans of $Y_n'$ and $f(\Delta t)$ to:

$$0.000_{16} \leq Y_n' \leq 1.FFC_{16} \qquad \text{VI-2}$$

$$0.000_{16} \leq f(t) \leq 1.000_{16} \qquad \text{VI-3}$$

Since no real changes have taken effect the scaling on $Y_n$ will be removed as it passes to the output buffer.

This data representation will cause no problems with addition or subtraction as illustrated below:

$$\begin{array}{rcl} 0000\ 1011\ 0000\ 0001 & & 0000\ 10.1100\ 0000\ 01 \\ +0000\ 0101\ 0010\ 0001 & \text{and} & +0000\ 01.0100\ 1000\ 01 \\ \hline 0001\ 0000\ 0010\ 0010 & & 0001\ 00.0000\ 1000\ 10 \end{array} \qquad \text{VI-4}$$

The bit patterns are the same for both additions.

Multiplication is a little different though. Consider the multiplication of two binary numbers:

$$\begin{array}{rl} & xxxxxx.xxxx\ xxxx\ xx \\ X & yyyyyy.yyyy\ yyyy\ yy \\ \hline & zzzzzzzzzzzz.zzzz\ zzzz\ zzzz\ zzzz\ zzzz \end{array} \qquad \text{VI-5}$$

From arithmetic recall that the number of decimal (binary) places to the right of the point in the product is equal to the combined number of digits right of the point in both the multiplier and the multiplicand.

The TMS 9900 multiplies two 16-bit numbers to form a 32-bit product which is placed in two registers. As an example (with the assumed point):

```
      0000 01.10 1000 0001
X     0000 01.00 0000 0000
-------------------------------
000000000001.10100000100000000000
-------------  -------------
  1st Register    2nd Register
```

If the assumed decimal point is to work then the answer should be a 16-bit word residing in one register with the point in the correct location. With the point between bits 5 and 6 multiplication will produce a two-register number with 20 fraction bits where 10 are excess. Shifting the entire two-register word pair left 6 places produces:

```
0000 01.10 1000 0001 0000 0000 00xx xxxx
-------------------  -------------------
  1st Register          2nd Register
```

The resulting number in the first register is the desired result. This is the multiplication procedure used in the program.

There is no similar problem with division. Shifting the decimal (binary) point an equal amount in 2 numbers has no effect on the ratio of the numbers.

The programming done in this design study uses this assumed decimal point notation.

Time scaling. The $f(\Delta t)$ map for the pulse waveshape is normalized in time as well as magnitude. As demonstrated in Chapter III a time scale factor, TSF, is needed to convert the real pulse $\Delta t$ into normalized time, $\tau$, for entry into the $f(\Delta t)$ function. There are also two other time scales that occur and must be considered. These scales are the result of digitizing the time interval. The single bit 1 from the time counter represents a time of 1/CR, where CR is the count rate in counts per second. Also the number $1_{16}$ represents a minimum time unit in the computer. Call this time $t_u$. (Note $t_u$ is not a function of the computer clock or of any timing function.) Then actual time is:

$$(\text{computer word}) \times (t_u) = \text{actual time} \qquad \text{VI-6}$$

The count rate of the time counter can be expressed in units of $t_u$ as:

$$C = CR \times t_u \qquad \text{VI-7}$$

where C is in counts per time unit and $t_u$ is in seconds per time unit. CR can be a very large number for very fine time resolution; it may be difficult to store CR in the computer. However, C can be made reasonably small by careful choice of $t_u$. The principal restriction is that

$$t_u > \frac{1}{CR} \qquad \text{VI-8}$$

so that C is greater than 1, and it is desirable that

$$t_u = n\left(\frac{1}{CR}\right) \qquad \text{VI-9}$$

where n is an integer. This makes C = n and avoids round-off errors that would occur when the fractional part of C was truncated.

If the TSF (time scaling factor) is in units of $t_u$ instead of seconds, then some useful relations result. First TSF is the half-pulse width of the pulse waveshape and is thus related to the maximum number of time counts that the time pick-off control needs to look for a pile-up.

$$\text{time counts}_{\max} = \frac{\text{TSF } t_u}{(1/CR)} = \frac{\text{TSF } t_u}{(t_u/C)} = \text{TSF } C \qquad \text{VI-10}$$

In the program this result is called $\Delta T\max$.

Next a conversion factor between the time count and the $\Delta$ address in the $f(\Delta t)$ map can be written:

$$\text{address} = \frac{\text{time count}}{(C/t_u)} \quad \frac{1}{\text{TSF } t_u} \quad (64)$$

$$= \frac{\text{time count}}{C} \quad \frac{1}{\text{TSF}} \quad (64) \qquad \text{VI-11}$$

Both Equation VI-10 and VI-11 are used in the time scaling in the Pile-Up Corrector program.

The program. The program flow charts developed in Chapter V were followed very closely in writing the program. There was, however, one significant change. It was assumed that a link was available between the Pile-Up Corrector and an external control instrument. The external instrument could change the $f(\Delta t)$ map and the constants TSF, C, and $\varepsilon$. The communications protocol uses a few ASCII code letters as described below:

| Code | As Sent from Corrector | As Sent to Corrector |
|---|---|---|
| S | none | halt and idle |
| M | none | new $f(\Delta t)$ coming |
| T | none | new TSF, C, and $\varepsilon$ coming |
| B | ready to begin execution | begin execution |
| ? | am waiting for a code | |

A flow chart of this EXECUTIVE/COMMAND program is shown in Figure VI-10.

The remainder of the documentation in this chapter is the software package for the TM 990/100M Pulse Pile-Up Corrector. The program was written in assembly language and was hand-assembled. The program is to be resident in ROM (EPROM in the prototype) as firmware. The memory map, Figure VI-11, shows the memory allocation. It closely follows the memory map of Figure V-5, page 81. Tables VI-2 and VI-3 list the designated usage of the RAM. Finally, Table VI-4 is the program listing. Again all the functional subprograms are well described in Chapter V and need not be repeated here.

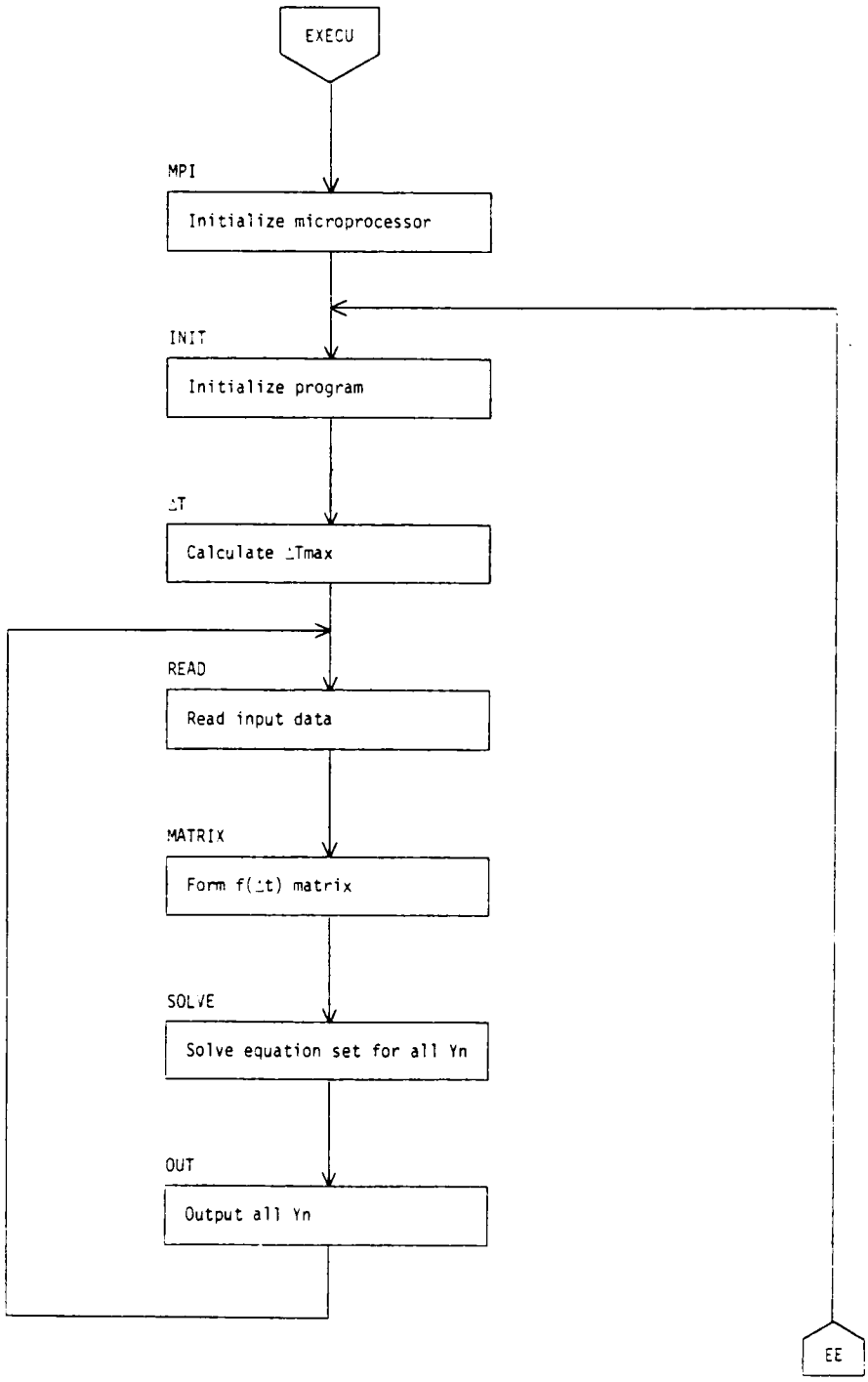



Figure VI-10. EXECUTIVE/COMMAND Program Flow Chart.

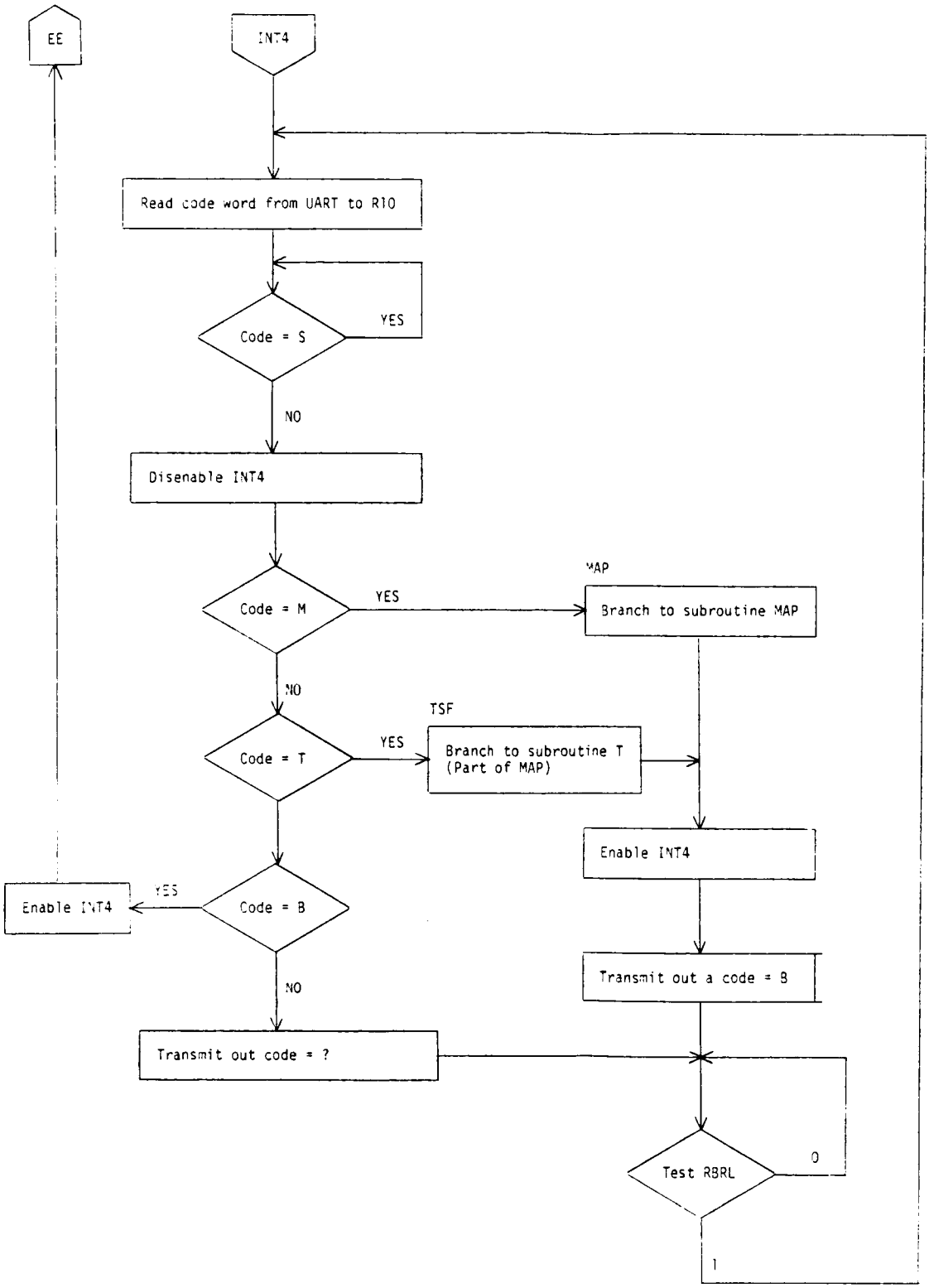



Figure VI-10. (Continued)

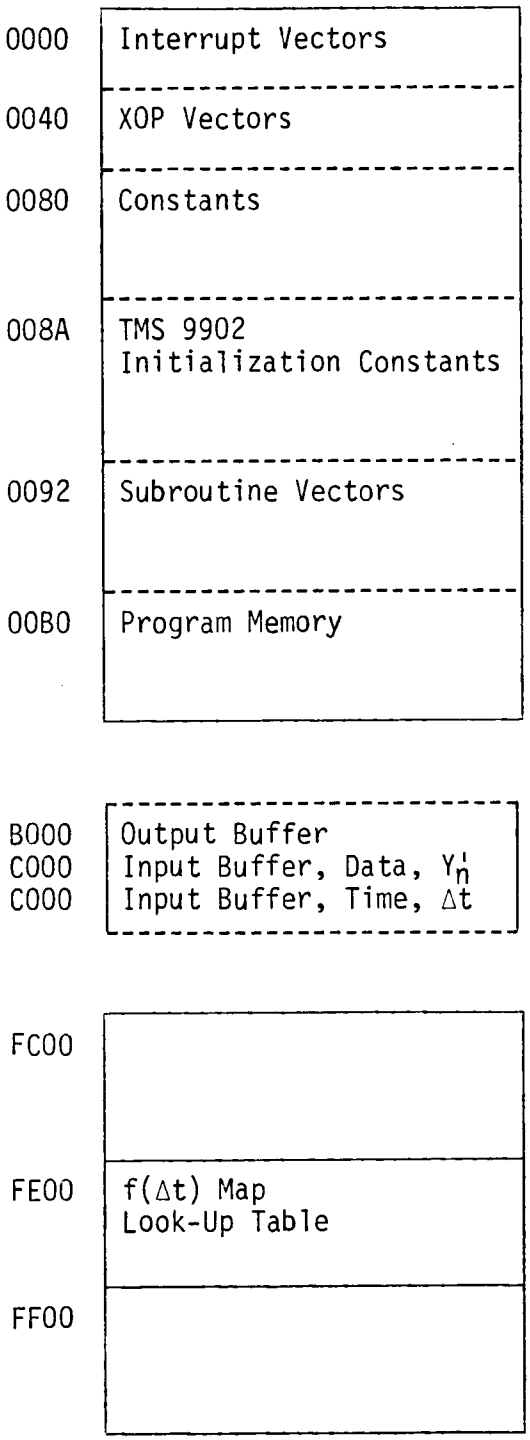



Figure VI-11. TM 990/100M Pulse Pile-Up Corrector Memory Map.

TABLE VI-2

RAM MEMORY, DESIGNATED STORAGE ASSIGNMENTS

| Addresses | Name of Contents |
|---|---|
| FC00--FC1E | Workspace 1 |
| FC20--FC3E | Workspace 2 |
| FC40--FC5E | Workspace 3 |
| FC60--FC68 | Data Input, $A_{1M}$--$A_{5M}$ |
| FC6A--FC72 | Time Input, $A_{1T}$--$A_{5T}$ |
| FC74 | not designated |
| FC76--FCB0 | f($\Delta$t) Coefficient Array, C1--C30 |
| FCB2--FCEC | Decomposed Matrix, D1--D30 |
| FCEE--FDFE | not designated |
| FF00 | Time Scale Factor, TSF |
| FF02 | Tolerance Limit, $\varepsilon$ |
| FF04 | Number of Interfering Pulses, N |
| FF06 | Time Count Rate of Time Pick-Off Control, C |
| FF08--FF0E | not designated |
| FF10--FF2E | Workspace 4 |
| FF30--FFFA | not designated |
| FFFC--FFFE | LOAD Vectors |

TABLE VI-3

WORKSPACE REGISTERS, DESIGNATED USES

| | Designated Uses | | | |
|---|---|---|---|---|
| Register | Workspace 1 | Workspace 2 | Workspace 3 | Workspace 4 |
| R0 | | | | |
| R1 | | | | |
| R2 | | YP | | |
| R3 | | $\#A_{XT}$/MJ/A | | |
| R4 | YP | #X / MI/ B | | |
| R5 | i | i | | |
| R6 | j | j | | |
| R7 | MS | MS | | |
| R8 | A | A / L | | |
| R9 | AP | AP | (C)(TSF) | Transmit Data |
| R10 | | K | ΔTmax | Receive Data |
| R11 | | PC Contents (BL) | | |
| R12 | | CRU Base Address | | |
| R13 | | WP Register Contents | | |
| R14 | | PC Contents | | |
| R15 | | ST Register Contents | | |

TABLE VI-4

TMS 990/100M PULSE PILE-UP CORRECTOR PROGRAM LISTING

| Address | Machine Code | Label | Assembly Code | Comments |
|---|---|---|---|---|
| 0000 | FF10 | | DATA>FF10 | Reset WP. |
| 0002 | 00B0 | | DATA>00B0 | Reset PC. |
| 0010 | FF10 | WS4 | DATA>FF10 | Interrupt 4 WP. |
| 0012 | 00CE | SVINT | DATA>00CE | Interrupt 4 PC. |
| 0080 | 0040 | | DATA>0040 | Number $64_{10}$. |
| 0082 | FC6A | | DATA>FC6A | $\#A_{1T}$. |
| 0084 | FC68 | | DATA>FC68 | $\#A_{1T}$ - 2. |
| 0086 | FC5E | | DATA>FC5E | $\#A_{1M}$ - 2. |
| 0088 | FE80 | | DATA>FE80 | Center of f(Δt) map. |
| 008A | 00A3 | | DATA>00A3 | Control register. |
| 008C | 009C | | DATA>009C | Interval register. |
| 008E | 01A1 | | DATA>01A1 | RDR. |
| 0090 | 01A1 | | DATA>01A1 | TDR. |
| 0092 | FC00 | WS1 | DATA>FC00 | Vector INT WP. |
| 0094 | 01CA | VINIT | DATA>01CA | PC. |
| 0096 | FC40 | WS3 | DATA>FC40 | Vector DLT WP. |
| 0098 | 021C | VDLT | DATA>021C | PC. |
| 009A | FC40 | WS3 | DATA>FC40 | Vector READ WP. |
| 009C | 0264 | VREAD | DATA>0264 | PC. |
| 009E | FC20 | WS2 | DATA>FC20 | Vector MATRIX WP. |
| 00A0 | 02B8 | VMATRX | DATA>02B8 | PC. |
| 00A2 | FC20 | WS2 | DATA>FC20 | Vector SOLV WP. |
| 00A4 | 0344 | VSOLV | DATA>0344 | PC. |
| 00A6 | FC00 | WS1 | DATA>FC00 | Vector OUT WP. |
| 00A8 | 0430 | VOUT | DATA>0430 | PC. |
| | | | | ********** |
| | | | | Begin EXECUTIVE program. |
| | | | | ********** |
| 00B0 | 06A0 | EXECU | BL @MPI | Call subroutine MPI. |
| 00B2 | 0188 | | | |
| 00B4 | 0420 | NEXT | BLWP @VINIT | Call subroutine INIT. |
| 00B6 | 0092 | | | |
| 00B8 | 0420 | | BLWP @VDLT | Call subroutine ΔT. |
| 00BA | 0096 | | | |
| 00BC | 0420 | CONT | BLWP @VREAD | Call subroutine READ. |
| 00BE | 009A | | | |
| 00C0 | 0420 | | BLWP @VMATX | Call subroutine MATRIX. |
| 00C2 | 009E | | | |
| 00C4 | 0420 | | BLWP @VSOLV | Call subroutine SOLVE. |
| 00C6 | 00A2 | | | |
| 00C8 | 0420 | | BLWP @VOUT | Call subroutine OUT. |
| 00CA | 00A6 | | | |

TABLE VI-4 (Continued)

| Address | Machine Code | Label | Assembly Code | Comments |
|---|---|---|---|---|
| 00CC | 10EE | | JMP CONT | Loop back for next set. |
| | | | | ********** |
| | | | | The following section is the interrupt service routine for servicing interrupts generated by a communication from the control instrument. |
| | | | | ********** |
| 00CE | 04CA | SVINT | CLR R10 | |
| 00D0 | 020C | | LI R12,>0080 | Set CRU base. |
| 00D2 | 0080 | | | |
| 00D4 | 1F15 | INTR | TB 21 | Wait for RBRL = 1. |
| 00D6 | 16FC | | JNE INTR | |
| 00D8 | 360A | | STCR R10, 8 | Store code in R10. |
| 00DA | 1E12 | | SBZ 18 | Reset RBRL. |
| 00DC | 028A | STOP | CI R10,>B300 | Compare R10 with S. |
| 00DE | B300 | | | |
| 00E0 | 13FA | | JEQ STOP | Hold if R10 = S. |
| 00E2 | 0300 | | LIMI 3 | Disable interrupt 4. |
| 00E4 | 0003 | | | |
| 00E6 | 028A | | CI R10,>BC00 | Compare R10 and M. |
| 00E8 | BC00 | | | |
| 00EA | 1322 | | JEQ NMAP | Jump if equal. |
| 00EC | 028A | | CI R10,>B400 | Compare R10 and T. |
| 00EE | B400 | | | |
| 00F0 | 1334 | | JEQ NEWT | Jump if equal. |
| 00F2 | 028A | | CI R10,>C200 | Compare R10 and B. |
| 00F4 | C200 | | | |
| 00F6 | 1310 | | JEQ BNEXT | |
| 00F8 | 020A | | LI R10,>BF00 | R10 = ? |
| 00FA | BF00 | | | |
| 00FC | 1010 | | SBO 16 | Turn transmitter on. |
| 00FE | 1F16 | ATST | TB 22 | Wait for XBRE = 1. |
| 0100 | 16FC | | JNE ATST | |
| 0102 | 320A | | LDCR R10, 8 | Transmit a ? |
| 0104 | 1E10 | | SBZ 16 | Turn transmitter off. |
| 0106 | 101C | | JMP BTST | |
| 0108 | 0300 | BNEXT | LIMI 4 | Enable interrupt 4. |
| 010A | 0004 | | | |
| 010C | 10AE | | JMP NEXT | Begin program. |
| 010E | 06A0 | NMAP | BL @MAP | Call subroutine MAP. |
| 0120 | 0140 | | | |
| 0122 | 0300 | NMAPA | LIMI 4 | Enable interrupt 4. |

TABLE VI-4 (Continued)

| Address | Machine Code | Label | Assembly Code | Comments |
|---|---|---|---|---|
| 0124 | 0004 | | | |
| 0126 | 020A | | LI R10,>C200 | R10 = B. |
| 0128 | C200 | | | |
| 012A | 1D10 | | SBO 16 | Turn transmitter on. |
| 012C | LF16 | CTST | TB 22 | Wait for XBRE = 1. |
| 012E | 16FC | | JNE CTST | |
| 0130 | 320A | | LCD R10, 8 | Transmit B. |
| 0132 | 1E10 | | SBZ 16 | Turn transmitter off. |
| 0134 | 1F15 | BTST | TB 21 | Test RBRL = 1. |
| 0136 | 16FC | | JNE BTST | |
| 0138 | 10AC | | JMP INTR | |
| 013A | 06A0 | NEWT | BL @SNEWT | Call subroutine SNEWT. |
| 013C | 0166 | | | |
| 013E | 10E4 | | JMP NMAPA | |
| | | | | ********** |
| | | | | Subroutine MAP, an external control instrument, instructs the program to accept new f(Δt) map, TSF, ε, and C constants. |
| | | | | ********** |
| 0140 | 020C | MAP | LI R12,>0080 | Set CRU base. |
| 0142 | 0800 | | | |
| 0144 | 0201 | | LI R1,>FE00 | Set a pointer to FE00. |
| 0146 | FE00 | | | |
| 0148 | 0209 | | LI R9,>BC00 | Load in the MSByte or R9 the ASCII M. |
| 014A | BC00 | | | |
| 014C | 1D10 | | SBO 16 | Turn transmitter on. |
| 014E | 1F16 | AA | TB 22 | Wait for XBRE = 1. |
| 0150 | 16FC | | JNE AA | |
| 0152 | 3209 | | LDCR R9, 8 | Transmit M. |
| 0154 | 1E10 | | SBZ 16 | Turn transmitter off. |
| 0156 | 1F15 | BB | TB 21 | Wait for XBRE = 1. |
| 0158 | 161F | | JNE BB | |
| 015A | 3608 | | STCR R10, 8 | Read a data byte to R10. |
| 015C | 1E12 | | SBZ 18 | Reset RBRL. |
| | | | | ********** |
| | | | | 128 data words are loaded byte at a time beginning with the MSByte of each data word followed by the LSByte. |
| | | | | ********** |

TABLE VI-4 (Continued)

| Address | Machine Code | Label | Assembly Code | Comments |
|---|---|---|---|---|
| 015E | D29A | | MOVB R10, *R1 | Move byte to MAP. |
| 0160 | 0281 | | CI R1,>FEFF | Test for last byte. |
| 0162 | FEFF | | | |
| 0164 | 1AF0 | | JL BB | |
| 0166 | 0209 | SNEWT | LI R9,>B400 | Set R9 = T. |
| 0168 | B400 | | | |
| 016A | 1D10 | | SBO 16 | |
| 016C | 1F16 | CC | TB 22 | |
| 016E | 16FC | | JNE CC | |
| 0170 | 3209 | | LCDR R9, 8 | Transmit T. |
| 0172 | 1E10 | | SBZ 16 | |
| 0174 | 0201 | | LI R1,>FF00 | Set R1 = address of TSF. |
| 0176 | FF00 | | | |
| 0178 | 1F15 | DD | TB 21 | |
| 017A | 16FC | | JNE DD | |
| 017C | 3608 | | STCR R10, 8 | Read data byte into R10. |
| 017E | 1E12 | | SBZ 18 | |
| | | | | ********** |
| | | | | Read successively first TSF, then ε, then C. |
| | | | | ********** |
| 0180 | D29A | | MOVB R10, *R1 | |
| 0182 | 0281 | | CI, R1,>FF07 | Test all 3 entered ? |
| 0184 | FF07 | | | |
| 0186 | 045B | | RTWP | Return from subroutine. |
| | | | | ********** |
| | | | | Begin subroutine MPI. |
| | | | | ********** |
| 0188 | 020C | MPI | LI R12,>0100 | Set CRU base. |
| 018A | 0100 | | | |
| 018C | 1D00 | | SBO 0 | Enter clock mode. |
| 018E | 1E0F | | SBZ 15 | Reset TMS 9901. |
| 0190 | C820 | | MOV @0000, @FFFC | Set LOAD vector WP. |
| 0192 | 0000 | | | |
| 0194 | FFFC | | | |
| 0196 | C820 | | MOV @0002, @FFFE | Set LOAD vector PC. |
| 0198 | 0002 | | | |
| 019A | FFFE | | | |
| 019C | 1000 | | NOP | |
| 019E | 1000 | | NOP | |
| 01A0 | 1D10 | | SBO 16 | Set input and out buffer MR's = 0. |

TABLE VI-4 (Continued)

| Address | Machine Code | Label | Assembly Code | Comments |
|---|---|---|---|---|
| 01A2 | 1D13 | | SBO 19 | Buffer MRs = 0. |
| 01A4 | 102C | | LI R12,>0800 | Set CRU Base for 9902. |
| 01A6 | 0800 | | | |
| 01A8 | 1E1F | | SBZ 31 | Reset the TMS 9902. |
| 01AA | 3220 | | LCDR @008A, 8 | Initialize 9902 control. |
| 01AC | 008A | | | |
| 01AE | 3220 | | LCDR @008C, 8 | Initialize 9902 interval. |
| 01B0 | 008C | | | |
| 01B2 | 32D0 | | LCDR @008E, 11 | Initialize 9902 RDR. |
| 01B4 | 008E | | | |
| 01B6 | 3320 | | LCDR @0090, 12 | Initialize 9902 TDR. |
| 01B8 | 0090 | | | |
| | | | | ********** |
| | | | | The TMS 9902 is initialized for 18-bit data words, even parity, and 1200 baud. |
| | | | | ********** |
| 01BA | 1D12 | | SBO 18 | Enable interrupt 4. |
| 01BC | 020C | | LI R12,>0100 | |
| 01BE | 0100 | | | Interrupt 4 is activated by interrupt from the 9902. All other interrupts are masked. |
| 01C0 | 1E00 | | SBZ 0 | |
| 01C2 | 1D04 | | SBO 4 | |
| 01C4 | 0300 | | LIMI 4 | |
| 01C6 | 0004 | | | |
| 01C8 | 045B | | RTWP | |
| | | | | ********** |
| | | | | Begin subroutine INIT. |
| | | | | ********** |
| 01CA | 020C | INIT | LI, R12,>0120 | Set CRU base. |
| 01CC | 0120 | | | |
| 01CE | 1D00 | | SBO 0 | Set 9901 in clock mode. |
| 01D0 | 1E0F | | SBZ 15 | Reset 9901. |
| 01D2 | 1E10 | | SBZ 16 | Set P0 as output. |
| 01D4 | 1E13 | | SBZ 19 | Set P3 as output. |
| 01D6 | 020C | | LI R12,>0126 | Set CRU base. |
| 01D8 | 0126 | | | |
| 01DA | 04C1 | | CLR R1 | |
| 01DC | 328A | | LCDR R1, 10 | Set bits P6-P15 as outputs. |
| 01DE | 0201 | CLEAR | LI R1,>FC00 | Clear RAM to FFFA. |

TABLE VI-4 (Continued)

| Address | Machine Code | Label | Assembly Code | Comments |
|---|---|---|---|---|
| 01E0 | FC00 | | | |
| 01E2 | 04D1 | | CLR *R1 | |
| 01E4 | 05C1 | | INCT R1 | |
| 01E6 | 0281 | | CI R1,>FFFA | |
| 01E8 | FFFA | | | |
| 01EA | 12EC | | JLE CLEAR | |
| 01EC | 0207 | | LI R7,>FC76 | Load a 1 in the princi- |
| 01EE | FC76 | | | pal diagonal of the C |
| 01F0 | 0205 | | LI R5, 1 | array. |
| 01F2 | 0001 | | | |
| 01F4 | 0206 | | LI R6, 1 | |
| 01F6 | 0001 | | | |
| 01F8 | 06A0 | DIAG | BL @APTR | |
| 01FA | 0448 | | | |
| 01FC | 0200 | | LI R0, 1 | |
| 01FE | 0001 | | | |
| 0200 | C680 | | MOV R0, *R9 | |
| 0202 | 0585 | | INC R5 | |
| 0204 | 0586 | | INC R6 | |
| 0206 | 0285 | | CI R5, 5 | |
| 0208 | 0005 | | | |
| 020A | 12EC | | JLE DIAG | |
| 020C | 0200 | | LI R12,>0100 | |
| 020E | 0100 | | | |
| 0210 | 1E13 | | SBZ 19 | Reset output buffer. |
| 0212 | 1E10 | | SBZ 16 | Reset input buffer. |
| 0214 | 1E10 | | SBO 19 | |
| 0218 | 1D10 | | SBO 16 | |
| 021A | 0380 | | RTWP | Return from subroutine. |
| | | | | ********** |
| | | | | Begin subroutine ΔT. |
| | | | | ********** |
| 021C | 0201 | DLT | LI R1,>FE00 | Set R1 = map start address. |
| 021E | FE00 | | | |
| 0220 | C160 | | MOV @>FFC2, R5 | Place ε in R5. |
| 0222 | FFC2 | | | |
| 0224 | C0D1 | TESTA | MOV *R1, R3 | Find f(Δt) and hold in |
| 0226 | 0743 | | ABS R3 | R3. |
| 0228 | 8143 | | C R3, R5 | Jump if R3 > R5. |
| 022A | 1B04 | | JH LOWPT | |
| 022C | 05C1 | | INCT R1 | R1 = R1 + 2 |
| 022E | 10CC | | JMP TESTA | |

TABLE VI-4 (Continued)

| Address | Machine Code | Label | Assembly Code | Comments |
|---|---|---|---|---|
| 0230 | 0202 | LOWPT | LI R2,>FEFE | Set R2 = map end address. |
| 0232 | FEFE | | | |
| 0234 | C0D2 | TESTB | MOV *R2, R3 | Find f(Δt) and hold in |
| 0236 | 0743 | | ABS R3 | R3. |
| 0238 | 8143 | | C R3, R5 | Compare and jump if |
| 023A | 1B04 | | JH HIGHPT | R3 > R5. |
| 023C | 0642 | | DECT R2 | R2 = R2 - 2 |
| 023E | 10EE | | JMP TESTB | |
| 0240 | 8042 | HIGHPT | C R2, R1 | Compare and jump if R2 |
| 0242 | 1B04 | | JH LODT | high. |
| 0244 | C281 | | MOV R1, R10 | ΔTmax to R10. |
| 0246 | 1002 | | JMP TPOC0 | |
| 0248 | C282 | LODT | MOV R2, R10 | ΔTmax to R10. |
| 024A | C220 | TPOC0 | MOV @>FF00, R8 | Determine the maximum |
| 024C | FF00 | | | allowed time count for |
| 024E | 3AA0 | | MPY @>FF06, R8 | the Time Pick-Off Con- |
| 0250 | FF06 | | | trol. |
| 0252 | 020C | | LI R12,>0100 | Set CRU base. |
| 0254 | 0100 | | | |
| 0256 | 1D15 | | SBO 21 | Transmit max time. |
| 0258 | 1F14 | TESTBT | TB 20 | Count to Time Pick-Off |
| 025A | 16FC | | JNE TESTBT | Control. |
| 025C | 020C | | LI R12,>0120 | |
| 025E | 1020 | | | |
| 0260 | 3289 | | LDCR R9, 10 | |
| 0262 | 0380 | | RTWP | End subroutine. |
| | | | | ********** |
| | | | | Begin subroutine READ. |
| | | | | ********** |
| 0264 | 020C | READ | LI R12,>0100 | Set CRU base. |
| 0268 | 0100 | | | |
| 026A | C2A0 | | MOV @>FF00, R8 | Form the C TSF product. |
| 026C | FF00 | | | |
| 026E | 3EA0 | | MPY @>FF06, R8 | |
| 0270 | FF06 | | | |
| 0272 | 0201 | | LI R1,>FC60 | Set R1 = $A_{1M}$ start |
| 0274 | FC60 | | | address. |
| 0276 | 0802 | | LI R2,>FC6A | Set R2 = $A_{1T}$ start |
| 0278 | FC6A | | | address. |
| 027A | 1F11 | TSTFF | TB 17 | Test for OR active from |
| 027C | 16FC | | JNE TSTFF | input buffer. |
| 027E | 1F12 | | TB 18 | |
| 0280 | 16F8 | | JNE TSTFF | |
| 0282 | C160 | | MOV @>C000, R5 | Read data to $A_{XM}$. |

TABLE VI-4 (Continued)

| Address | Machine Code | Label | Assembly Code | Comments |
|---|---|---|---|---|
| 0284 | C000 | | | |
| 0286 | 0245 | | ANDI R5,>07FF | Mask unused bits. |
| 0288 | 07FF | | | |
| 028A | CC45 | | MOV R5, *R1+ | |
| 028C | C160 | | MOV @>D000, R5 | Read time count to R5. |
| 028E | D000 | | | |
| 0290 | 0245 | | ANDI R5,>03FF | Mask unused bits. |
| 0292 | 03FF | | | |
| 0294 | 3960 | | MPY @>0080, R5 | 64xCount to R5. |
| 0296 | 0080 | | | |
| 0298 | 3D48 | | DIV R8, R5 | R5÷(C TSF) to R5. |
| 029A | CC85 | | MOV R5, *R2+ | Move to an $A_{XT}$ array. |
| 029C | 0281 | | CI R1,>FC6A | Test and skip ahead to |
| 029E | FC6A | | | SETN if 5 points done. |
| 02A0 | 1308 | | JEQ SETN | |
| 02A2 | 0745 | | ABS R5 | Compare and jump ahead |
| 02A4 | 814A | | C R10, R5 | if R10 > R5. |
| 02A6 | 1402 | | JHE SETN | |
| 02A8 | 10D0 | | JMP TSTFF | |
| 02AA | C041 | SETN | MOV R2, R1 | Calculate N and store at |
| 02AC | 6060 | | S @>0082, R1 | FF04. |
| 02AE | 0082 | | | |
| 02B0 | 0811 | | SRA R1, 1 | |
| 02B2 | C801 | | MOV R1, @>F004 | |
| 02B4 | FF04 | | | |
| 02B6 | 0380 | | RTWP | Return from subroutine. |
| | | | | ********** |
| | | | | Begin subroutine MATRIX. |
| | | | | ********** |
| 02B8 | 0207 | MATRIX | LI R7,>FC76 | R7 = MS = C1. |
| 02BA | FC76 | | | |
| 02BC | 04C8 | | CLR R8 | Clear a pointer in R8. |
| 02BE | 04C1 | | CLR R1 | |
| 02C0 | 04C2 | | CLR R2 | |
| 02C2 | 0206 | | LI R6, 1 | Set j = 1. |
| 02C4 | 0001 | | | |
| 02C6 | 0205 | LOOPA | LI R5, 2 | |
| 02C8 | 0002 | | | |
| 02CA | A148 | | A R8, R5 | Set i = A + 2. |
| 02CC | C108 | | MOV R8, R4 | Set X in R4. |
| 02CE | 0584 | | INC R4 | X = 1 + A |
| 02D0 | 0A14 | | SLA R4, 1 | X = 2X. |
| 02D2 | 06A0 | LOOPB | BL @APTR | Call AP. |
| 02D4 | 0448 | | | |

TABLE VI-4 (Continued)

| Address | Machine Code | Label | Assembly Code | Comments |
|---|---|---|---|---|
| 02D6 | C0C4 | | MOV R4, R3 | R3 = #$A_{xT}$ = 2X + ($A_{1T}$ - 2). |
| 02D8 | 0A13 | | SLA R3, 1 | |
| 02DA | A0E0 | | A @>0084, R3 | |
| 02DC | 0084 | | | |
| 02DE | A093 | | A *R3, R2 | Add R3 + R2. |
| 02E0 | C042 | | MOV R2, R1 | Add R2 + FE80 and store |
| 02E2 | A060 | | A @>0088, R1 | in R1. |
| 02E4 | 0088 | | | |
| 02E6 | C651 | | MOV *R1, *R9 | Find f(Δt) and store in |
| 02E8 | 1000 | | NOP | C array. |
| 02EA | C045 | | MOV R5, R1 | Exchange i and j. |
| 02EC | C146 | | | |
| 02EE | C181 | | | |
| 02F0 | 06A0 | | BL @APTR | Call AP. |
| 02F2 | 0448 | | | |
| 02F4 | C042 | | MOV R2, R1 | |
| 02F6 | 6060 | | S @>0088, R1 | R1 = FE80 - R2. |
| 02F8 | 0088 | | | |
| 02FA | C651 | | MOV *R1, *R9 | Find f(Δt) and store in |
| 02FC | 1000 | | NOP | C array. |
| 02FE | C045 | | MOV R5, R1 | Exchange i and j. |
| 0300 | C146 | | MOV R6, R5 | |
| 0302 | C181 | | MOV R1, R6 | |
| 0304 | 05C3 | | INCT R3 | X = X + 2. |
| 0306 | 0585 | | INC R5 | i = i + 1. |
| 0308 | 8805 | | C R5, @>FF04 | Compare i and N. |
| 030A | 12CA | | JLE LOOPB | |
| 030C | 04C1 | | CLR R1 | |
| 030E | 04C2 | | CLR R2 | |
| 0310 | 0588 | | INC R8 | R8 = R8 + 1. |
| 0312 | C060 | | MOV @>FF04, R3 | |
| 0314 | FF04 | | | |
| 0316 | 0643 | | DECT R3 | |
| 0318 | 80C8 | | C R8, R3 | Compare A and N - 2. |
| 031A | 12AE | | JLE LOOPA | |
| 031C | 3860 | | MPY @>FF04, R1 | |
| 031E | FF04 | | | |
| 0320 | 0581 | | INC R1 | |
| 0322 | 0209 | | LI R9,>FC76 | |
| 0324 | FC76 | | | |
| 0326 | A241 | | A R1, R9 | Set AP = 5N + 1 + #C1. |
| 0328 | 04C4 | | CLR R4 | |
| 032A | 05C4 | | INCT R4 | Set X = 2. |
| 032C | C0C4 | LOOPC | MOV R4, R3 | |
| 032E | A060 | | A @>0086, R3 | Set #$A_{xM}$ = X + $A_{1M}$ - 2. |

TABLE VI-4 (Continued)

| Address | Machine Code | Label | Assembly Code | Comments |
|---|---|---|---|---|
| 0330 | 0086 | | | |
| 0332 | C663 | | MOV *R3, *R9 | Move $*A_{XM}$ to *AP. |
| 0334 | 05C4 | | INCT R4 | X = X + 2. |
| 0336 | 05C9 | | INCT R9 | AP = AP + 2. |
| 0338 | C044 | | MOV R4, R1 | |
| 033A | 0811 | | SRA R1, 1 | |
| 033C | 8801 | | C R1, @>FF04 | Compare X and N. |
| 033E | FF04 | | | |
| 0340 | 12EA | | JLE LOOPC | |
| 0342 | 0380 | | RTWP | Return from subroutine. |
| | | | | ********** |
| | | | | Begin subroutine SOLVE. |
| | | | | ********** |
| 0344 | 0203 | SOLVE | LI R3,>0001 | Set R3 = MJ = 1. |
| 0346 | 0001 | | | |
| 0348 | 0404 | | CLR R4 | Set R4 = MI = 0. |
| 034A | C203 | SLOPA | MOV R3, R8 | Set R8 = L = MJ. |
| 034C | 0584 | SLOPB | INC R4 | MI = MI + 1. |
| 034E | 06A0 | | BL @ABLE | Call ABLE, find $l_{ij}$. |
| 0350 | 03EC | | | |
| 0352 | 0207 | | LI R7,>FCB2 | Set R7 = MS = #D1. |
| 0354 | FCB2 | | | |
| 0356 | 06A0 | | BL @APTR | Call AP. |
| 0358 | 0488 | | | |
| 035A | C641 | | MOV R1, *R9 | Move $l_{ij}$ to D array. |
| 035C | 8804 | | C R4, @>FF04 | Compare MI and N. |
| 035E | FF04 | | | |
| 0360 | 1AEA | | JL SLOPB | |
| 0362 | C143 | | MOV R3, R4 | Set MI = MJ. |
| 0364 | 0583 | | INC R3 | MJ = MJ + 1. |
| 0366 | C204 | | MOV R4, R8 | R8 = L = MI. |
| 0368 | 06A0 | SLOPC | BL @ABLE | Call ABLE, find $t_{ij}l_{ii}$. |
| 036A | 03EC | | | |
| 036C | C185 | | MOV R5, R6 | Set j = i. |
| 036E | 06A0 | | BL @APTR | |
| 0370 | C183 | | MOV R3, R6 | Set j = MJ. |
| 0372 | C183 | | MOV R1, R2 | Divide $(t_{ij}l_{ii})$ by $l_{ii}$ |
| 0376 | 0461 | | CLR R1 | to find $t_{ij}$ and store |
| 0378 | 3C59 | | DIV *R9, R1 | in R1. |
| 037A | 0207 | | LI R7,>FCB2 | Set R7 = MS = #D1. |
| 037C | FCB2 | | | |
| 037E | 06A0 | | BL @APTR | |
| 0380 | 0448 | | | |

TABLE VI-4 (Continued)

| Address | Machine Code | Label | Assembly Code | Comments |
|---|---|---|---|---|
| 0382 | C5C1 | | MOV R1, *R7 | Move $t_{ij}$ to D array. |
| 0384 | 0583 | | INC R3 | Set R3 = MJ = MJ + 1. |
| 0386 | 0202 | | LI R2,>0001 | Set R2 = 1. |
| 0388 | 0001 | | | |
| 038A | A0A0 | | A @>FF04, R2 | R2 = N + 1. |
| 038C | FF04 | | | |
| 038E | 8083 | | C R3, R2 | Compare MJ and N + 1. |
| 0390 | 12DA | | JLE SLOPC | |
| 0392 | 8804 | | C R4, @>FF04 | Compare MI and N. |
| 0393 | FF04 | | | |
| 0394 | 1406 | | JHE FRWD | |
| 0396 | C0C4 | | MOV R4, R3 | MJ = MI. |
| 0398 | 10AE | | JMP SLOPA | |
| | | | | ********** |
| | | | | Being back substitution for $Y_n$'s. |
| | | | | ********** |
| 039A | 0200 | FWRD | LI R7,>FBC0 | Set R7 = MS = #C1. |
| 039C | FBC0 | | | |
| 039E | C0E3 | | MOV @>FF04, R3 | Set R3 = A = N. |
| 03A0 | C103 | BACKA | MOV R3, R4 | |
| 03A2 | 0584 | | INC R4 | Set R4 = B = A + 1. |
| 03A4 | C143 | | MOV R3, R5 | Set R5 = I = A. |
| 03A6 | C1A0 | | MOV @>FF04, R6 | |
| 03A8 | FF04 | | | |
| 03AA | 0586 | | INC R | Set R6 = j = N + 1. |
| 03AC | 06A0 | | BL @APTR | |
| 03AE | 0448 | | | |
| 03B0 | C059 | | MOV *R9, R1 | Set R1 = *AP. |
| 03B2 | 8804 | BACKB | C R4, @>FF04 | Compare B and N. |
| 03B4 | FF04 | | | |
| 03B6 | 1B1E | | JH BACKC | |
| 03B8 | C144 | | MOV R4, R6 | Set R6 = J = B. |
| 03BA | 06A0 | | BL @APTR | |
| 03BC | 0448 | | | |
| 03BE | C084 | | MOV R4, R2 | Set the YP in R2. |
| 03C0 | 0A12 | | SLA R2, 1 | YP = $\#A_{1M}$ - 2 + 2B. |
| 03C2 | 0222 | | AI R2,>FC5E | |
| 03C4 | FC5E | | | |
| 03C6 | C259 | | MOV *R9, R9 | Multiply *YP by *AP. |
| 03C8 | 3A52 | | MPY *R2, R9 | Subtract from R1 and |
| 03CA | 0A69 | | SLA R9, 6 | store in R1. |
| 03CC | 09AA | | SRL R10, 10 | |
| 03CE | A24A | | A R10, R9 | |

TABLE VI-4 (Continued)

| Address | Machine Code | Label | Assembly Code | Comments |
|---|---|---|---|---|
| 03D0 | 6049 | | S R9, R1 | |
| 03D2 | 0584 | | INC R4 | Set B = B + 1. |
| 03D4 | 10E0 | | JMP BACKB | |
| 03D6 | C083 | BACKC | MOV R3, R2 | |
| 03D8 | 0A12 | | SLA R2, 1 | |
| 03DA | 0222 | | AI R2,>FC5E | Set R2 = YP = 2A + ($\#A_{1M}$*2). |
| 03DC | FC5E | | | |
| 03DE | C401 | | MOV R1, *R2 | *YP = R1. |
| 03E0 | 0283 | | CI R3,>0001 | Compare A and 1. |
| 03E2 | 0001 | | | |
| 03E4 | 1604 | | JNE BACKD | |
| 03E6 | 0380 | | RTWP | Return from subroutine. |
| 03E8 | 0603 | BACKD | DEC R3 | |
| 03EA | 10BC | | JMP BACKA | |
| | | | | ********** |
| | | | | Begin subroutine ABLE. |
| | | | | ********** |
| 03EC | C80B | ABLE | MOV R11, @>FF08 | Save PC in FF08. |
| 03EE | 0207 | | LI R7,>FC76 | Set R7 = L = MI. |
| 03F0 | FC76 | | | |
| 03F2 | 020A | | LI R10,>0001 | Set R10 = K = 1. |
| 03F4 | 0001 | | | |
| 03F6 | 04C2 | | CLR R2 | |
| 03F8 | 8288 | ABLEA | C R8, R10 | Compare L and K. |
| 03FA | 1322 | | JEQ ABLEA | |
| 03FC | C144 | | MOV R4, R5 | Set R5 = i = MI. |
| 03FE | C18A | | MOV R10, R6 | Set R6 = j = K. |
| 0400 | 06A0 | | BL @APTR | |
| 0402 | 0488 | | | |
| 0404 | C051 | | MOV *R9, R1 | Set R1 = *AP. |
| 0406 | C14A | | MOV R10, R5 | Set R5 = i = K. |
| 0408 | C183 | | MOV R3, R6 | Set R6 = j = MJ. |
| 040A | 06A0 | | BL @APTR | |
| 040C | 0488 | | | |
| 040E | C001 | | MOV R1, R0 | Multiply R1 by *AP and |
| 0410 | 3819 | | MPY *R9, R0 | store in R1. |
| 0412 | 0A60 | | SLA R0, 6 | |
| 0414 | 09A0 | | SRL R1, 10 | |
| 0416 | A040 | | A R0, R1 | |
| 0418 | A081 | | A R1, R2 | R2 = R2 + R1. |
| 041A | 058A | | INC R10 | Set R10 = K = K + 1. |
| 041C | 10DA | | JMP ABLEA | |
| 041E | C144 | ABLEB | MOV R4, R5 | Set R5 = i = MI. |

TABLE VI-4 (Continued)

| Address | Machine Code | Label | Assembly Code | Comments |
|---|---|---|---|---|
| 0420 | C183 | | MOV R3, R6 | Set R6 = j = MJ. |
| 0424 | 0488 | | | |
| 0426 | C059 | | MOV *R9, R1 | |
| 0428 | 6042 | | S R2, R1 | Set R1 = *AP - R2. |
| 042A | C2E0 | | MOV @>FF08, R11 | Retrieve PC. |
| 042C | FF08 | | | |
| 042E | 0480 | | RTWP | Return from subroutine. |
| | | | | ********** |
| | | | | Begin subroutine OUT. |
| | | | | ********** |
| 0430 | 0204 | OUT | LI R4,>FC60 | Set R4 = YP = FC60. |
| 0432 | FC60 | | | |
| 0434 | C060 | | MOV @>FF04, R1 | Set R1 = N. |
| 0436 | FF04 | | | |
| 0438 | C814 | OUTA | MOV *R4, @>B000 | Move $Y_n$ to output buffer. |
| 043A | B000 | | | |
| 043C | 05C4 | | INCT R4 | YP = YP + 2. |
| 043E | 0640 | | DEC R1 | N = N - 1. |
| 0440 | 0281 | | CI R1, 0 | Test for N point out. |
| 0442 | 0000 | | | |
| 0444 | LBF2 | | JH OUTA | |
| 0446 | 0380 | | RTWP | Return from subroutine. |
| | | | | ********** |
| | | | | Begin subroutine AP. |
| | | | | ********** |
| 0448 | 0209 | APTR | LI R9, 5 | Calculate from i and j |
| 044A | 0005 | | | and the array starting |
| 044C | 3A46 | | MPY R6, R9 | address the array |
| 044E | C24A | | MOV R10, R9 | pointer AP. |
| 0450 | A245 | | A R5, R9 | |
| 0452 | 020A | | LI R10, 6 | AP = 2(5J + i - 6) + MS. |
| 0454 | 0006 | | | |
| 0456 | 624A | | S R10, R9 | Store AP in R9. |
| 0458 | 0A19 | | SLA R9, 1 | |
| 045A | A246 | | A R6, R9 | |
| 045C | 045B | | RTWP | Return from subroutine. |

# CHAPTER VII

## SUMMARY

In this study the problem of designing a Pulse Pile-Up Corrector using microprocessor technology has been examined. It was first necessary to examine the nature of pile-up in pulse height analysis systems so that an algorithm could be developed. This algorithm was shown to be a simple set of simultaneous linear equations with the coefficient matrix being a function of the pulse spacing and the pulse waveshape.

With the development of the algorithm there was no question that a microprocessor based instrument could solve the algorithm. A simulation of the algorithm in a programmable calculator demonstrated that. The real question throughout the study was could the microprocessor solve the algorithm fast enough to operate in near real time. It was shown that if the piled-up pulse burst rate was less than the reciprocal of the microprocessor's pulse burst processing time then near real time operation would be possible. It was, as described, necessary to add a FIFO input buffer to average the fluctuation in pulse burst rate throughout an experiment.

With selection of pulse burst processing time, $T_{PM}$, as a performance benchmark, a universal design for a microprocessor-based Pile-Up Corrector was prepared. This was then followed with the design of an instrument using real hardware. The result was the TM 990/100M Pulse Pile-Up Corrector described in Chapter VI.

It was shown that a pile-up correction algorithm existed and that an instrument to implement that algorithm can be designed. It remains

to estimate the performance of that instrument. This estimate was made by assuming typical pile-ups of 2 or 3 pulses and then manually tracing through the program execution, counting the number of microprocessor cycles. This is presented below:

| Program Subroutine | N = 2 | N = 3 |
|---|---|---|
| READ | 916 | 1228 |
| MATRIX | 1278 | 2902 |
| SOLVE | 5003 | 12522 |
| OUT | 182 | 260 |
| Totals | 7379 | 16912 |

With the 3MHz clock in the TM 990/100M these total times correspond to 2.46ms and 5.64ms.

At the beginning of Chapter VI an estimate of a typical data structure was constructed. In that estimate a processing time, $T_{PM}$, of 4ms was seen to be typical. Therefore the performance of the TM 990/100M system is in the right range.

These performance results are encouraging, particularly when it is realized that they can easily be improved. First a system built with a recently announced Texas Instruments Incorporated TM 990/1481 Micro-Computer would operate at 3 times the speed of the TM 990/100M. Second a computer built from Texas Instruments SN 54/74s481 bit slice and microcoded to emulate the 9900 instruction set would be 6 times faster than the TMS 9900 used in the TM 990/100M. Another option is to use bit slice devices to design a limited function computer tailored specifically for pile-up correction. There is also some potential of streamlining the software. If the data structure allows assuming that there

will never be more than 2 pulses piled up, then most of the array building and accessing routines can be simplified and the simultaneous equation algorithm does not need to be general.

The accuracy of the corrector is affected by accuracy of the ADC conversion of the pulse data, accuracy of measuring pulse time spacing, the accuracy and stability of the $f(\Delta t)$ waveshape, and arithmetic errors introduced in the computations. ADC conversion accuracy considerations that will affect the Pile-Up Corrector will equally affect the MCA. The ADC and its accuracy are already part of the system; adding the corrector will cause no change. The pulse spacing will probably be measured using a high-speed oscillator and pulse counter or a precise RC timer followed by ADC. The timer will have to be carefully designed both to provide accuracy and to have a sufficiently high count in the time range of interest to ensure precision.

The study of pulse waveshaping for pulse amplifiers is a large field in itself. A sampling of the literature of this field (Fairstein, 1965; Pierce, 1967; Nowlin; Ortec Product Catalog) indicates that pulse amplifiers are specifically designed to produce particular pulse waveshapes. Because these waveshapes are designed in and not just obtained by chance it seems reasonable to assume they are well defined. So problems from waveshape inaccuracies should be small.

The Pile-Up Corrector program digitizes the $f(\Delta t)$ magnitude axis in steps of 1/2048 while the time axis is digitized in steps of 1/128. It is possible that this difference will cause accuracy problems. If so the map could easily be expanded to a larger table in memory. However, it may not be a problem. The slope function $df(\Delta t)/d\Delta t$ is large near

$\Delta t = 0$ but typically becomes small at larger $\Delta t$ where the pulse slope flattens out. In a random distribution of pulses where most pulses are not piled up then most piled-up pulses can be expected to be overlapped at the larger $\Delta t$ spacing where the slope function is smaller. With a small slope a large $d\Delta t$ produces a small $df(\Delta t)$.

Arithmetic errors are held down by 2 factors. First the coefficient matrix of the algorithm has a principal diagonal of 1 and off-diagonal elements less than 1. This matrix is not ill-conditioned; consequently an accurate solution using fixed digit arithmetic can be expected (Ralston, 1965). Second the data representation presented in Chapter VI keeps the numbers away from the range limits of the 16-bit word, thus minimizing round-off problems.

The overall conclusion of this study is that a microprocessor based Pulse Pile-Up Corrector can be designed and that its performance will be within the range required by typical pulse analysis methods.

# LIST OF REFERENCES

# LIST OF REFERENCES


1. Ortec Product Catalog Number 1004. Ortec Incorporated, Oak Ridge, Tennessee, 1976.

2. Fairstein, E., and J. Hahn. "Nuclear Pulse Amplifiers--Fundamentals and Design Practice," Nucleonics, Vol. 23, No. 7, No. 11, 1965, Vol. 24, No. 1, 1966.

3. Pierce, J. F., B. W. McNeill, and T. L. Mayhugh. "A Low Power Linear Channel for Use with Photomultiplier Tubes," A report prepared for the Goddard Space Flight Center, University of Tennessee, 1977.

4. Nowlin, C. H. "Pulse Shaping for Nuclear Pulse Amplifiers," A research paper sponsored by the U.S. Atomic Energy Commission under contract with Union Carbide Corporation, N.D.

5. Canberra 1979-80 Product Catalog. Canberra Industries, Meriden, Connecticut, 1979.

6. Farish, R. F., C. D. O'Grady, and R. A. Olivia. Personal Programming, A Complete Owner's Manual for the TI-59 Programmable 58/59. Dallas: Texas Instruments Incorporated, 1977.

7. TI Programmable 58/59 Master Library. Dallas: Texas Instruments Incorporated, 1977.

8. Simpson, W. D., G. Luecke, D. L. Cannon, and D. H. Clemens. 9900 Family Systems Design and Data Book. Dallas: Texas Instruments Incorporated, 1978.

9. Monolithic Memories Bipolar LS Data Book. Monolithic Memories Monolithic Memories Inc., Sunnyvale, California, 1980.

10. Blood, B. "A High Speed FIFO Memory Using the MECL MC10143 Register File," Motorola Application Note AN730. Motorola Semiconductor Products Incorporated, Phoenix, Arizona, 1974.

11. Ketter, R. L., and S. P. Prawel, Jr. Modern Methods of Engineering Computation. New York: McGraw-Hill Book Company, 1969.

12. Ralston, A. A First Course in Numerical Analysis. New York: McGraw-Hill Book Company, 1965.

# APPENDICES

# APPENDIX A

## TI-59 ALGORITHM SIMULATION DETAILED FLOW CHART AND PROGRAM LISTING

The simplified flow chart, Figure IV-1, page 34, was the basis for the detailed flow charts and subsequent program listing presented here in Appendix A.

This work was begun by designating the data memory registers. The Master library program number 2 (ML-02) was used to solve the set of simultaneous equations. To use ML-02 the following registers had to be reserved:

| | | |
|---|---|---|
| R00 | R03 | R06 |
| R01 | R04 | R07 |
| R02 | R05 | R08--Rq |

Where $q=N^2+2N+7$ and N is the number of equations in the equation set. The calculator has a limited memory; the utilization of memory required choosing N=3. This choice limits the power of the program, but not the generality of the method. Since this calculator simulation was an investigation of the method of implementing the separation algorithm, limiting N to a low value will not be a consequential factor. With N=3 the number q=22. With N set at 3, it was decided to give the program a storage capability of five data points. This allows for all expected combinations of two pulses interfering with the spot-lighted pulse.

To store 5 data points 10 registers were reserved for the magnitude and time data. They are:

| | |
|---|---|
| R30--$t_{n-2}$ | R35--$Y'_{n-2}$ |
| R31--$t_{n-1}$ | R36--$Y'_{n-1}$ |
| R32--$t_n$ | R37--$Y'_n$ the spotlighted pulse |
| R33--$t_{n+1}$ | R38--$Y'_{n+1}$ |
| R34--$t_{n+2}$ | $R_{39}$--$Y'_{n+2}$ |

The other registers reserved are:

| | |
|---|---|
| R40--index | R45--index |
| R41--index | R46--index |
| R42--i | R47--index |
| R43--k | R48--index, time scale factor |
| R44--index | R49--index |

The program was divided into three major sections and a fourth special section. These are (1) Initialization, (2) Executive, and (3) the functional subprograms labeled LnX, CE RCL, $y^x$, and LOG. The fourth special program is labeled GTO. Initialization sets the program up for a continuing series of data points entered one at a time. Executive allows the functional subprograms to be developed independently. The functional subprograms do the real work of the algorithm. These subprograms are labeled LnX, CE, RCL, $y^x$, LOG, and GTO. Subprogram LnX counts the number of preceding points, i, and CE counts the following points, k, that are in potential pile-up with the spotlighter pulse, $Y_n$. Subprogram RCL limits the combined number of points i+k+1 to be no greater than the program capability, M. It does this by discarding the points farthest from $Y_n$. Subprogram $y^x$ then forms the arrays and accesses ML-02 to solve the equation set for $Y_n$. Subprogram LOG controls

the manual entry of new data points. Finally the fourth section, GTO, generates the normalized waveshape $f(\Delta t)$. A program instruction sheet and a complete program listing follow.

TABLE A-1

TI-59 SIMULATION PROGRAM INSTRUCTION SHEET

| Step | Procedure | Enter | Press | Display | Print |
|---|---|---|---|---|---|
| 1 | Initialize program | | E' | .99-99 | |
| 2 | Enter a negative number greater than the time scale factor | -N | STO 34<br>STO 33 | -1<br>-1 | |
| 3 | Enter time scale factor | T | R/S | .11-11 | DATA |
| 4 | Enter magnitude data | $Y'_1$ | R/S | .22-22 | |
| 5 | Enter time data | $t_1$ | R/S | .11-11 | |
| 6 | Enter 2nd magnitude | $Y'_2$ | R/S | .22-22 | |
| 7 | Enter 2nd time | $t_2$ | R/S | .11-11 | |
| 8 | Enter 3rd magnitude | $Y'_3$ | R/S | .22-22 | |
| 9 | Enter 3rd time | $t_3$ | R/S | | BEGIN |
| 10 | Program calculates results of first 3 data points | | | | N<br>t<br>Y'<br>Y |
| 11 | Enter magnitude | $Y'_n$ | | .11-11 | DATA |
| 12 | Enter time | $t_n$ | | .22-22 | |
| 13 | Program loops back to step 10. | | | | |

TABLE A-2

TI-59 SIMULATION PROGRAM LISTING

| Loc | Code | Key | Loc | Code | Key | Loc | Code | Key |
|---|---|---|---|---|---|---|---|---|
| 000 | 76 | LBL | 043 | 43 | RCL | 086 | 61 | GTO |
| 001 | 10 | E' | 044 | 71 | SBR | 087 | 15 | E |
| 002 | 47 | CMS | 045 | 45 | $Y^X$ | 088 | 76 | LBL |
| 003 | 00 | 0 | 046 | 32 | X:T | 089 | 45 | $Y^X$ |
| 004 | 35 | 1/X | 047 | 25 | CLR | 090 | 53 | ( |
| 005 | 91 | R/S | 048 | 69 | OP | 091 | 01 | 1 |
| 006 | 42 | STO | 049 | 00 | 00 | 092 | 85 | + |
| 007 | 48 | 48 | 050 | 57 | ENG | 093 | 43 | RCL |
| 008 | 03 | 3 | 051 | 03 | 3 | 094 | 42 | 42 |
| 009 | 42 | STO | 052 | 07 | 7 | 095 | 85 | + |
| 010 | 01 | 01 | 053 | 69 | OP | 096 | 43 | RCL |
| 011 | 76 | LBL | 054 | 04 | 04 | 097 | 43 | 43 |
| 012 | 29 | CP | 055 | 53 | ( | 098 | 54 | ) |
| 013 | 71 | SBR | 056 | 43 | RCL | 099 | 36 | PGM |
| 014 | 28 | LOG | 057 | 32 | 32 | 100 | 02 | 02 |
| 015 | 97 | DSZ | 058 | 65 | x | 101 | 11 | A |
| 016 | 01 | 01 | 059 | 43 | RCL | 102 | 32 | X:T |
| 017 | 29 | CP | 060 | 48 | 48 | 103 | 01 | 1 |
| 018 | 25 | CLR | 061 | 54 | ) | 104 | 39 | CDS |
| 019 | 69 | OP | 062 | 69 | OP | 105 | 01 | 1 |
| 020 | 00 | 00 | 063 | 06 | 06 | 106 | 01 | 1 |
| 021 | 01 | 1 | 064 | 22 | INV | 107 | 36 | PGM |
| 022 | 04 | 4 | 065 | 57 | ENG | 108 | 02 | 2 |
| 023 | 01 | 1 | 066 | 03 | 3 | 109 | 12 | B |
| 024 | 07 | 7 | 067 | 00 | 0 | 110 | 53 | ( |
| 025 | 02 | 2 | 068 | 06 | 6 | 111 | 03 | 3 |
| 026 | 02 | 2 | 069 | 05 | 5 | 112 | 02 | 2 |
| 027 | 02 | 2 | 070 | 69 | OP | 113 | 75 | - |
| 028 | 04 | 4 | 071 | 04 | 04 | 114 | 43 | RCL |
| 029 | 03 | 3 | 072 | 43 | RCL | 115 | 42 | 42 |
| 030 | 01 | 1 | 073 | 37 | 37 | 116 | 54 | ) |
| 031 | 69 | OP | 074 | 69 | OP | 117 | 42 | STO |
| 032 | 01 | 01 | 075 | 06 | 06 | 118 | 49 | 49 |
| 033 | 69 | OP | 076 | 03 | 3 | 119 | 76 | LBL |
| 034 | 05 | 05 | 077 | 00 | 0 | 120 | 55 | ÷ |
| 035 | 98 | ADV | 078 | 69 | OP | 121 | 53 | ( |
| 036 | 76 | LBL | 079 | 04 | 04 | 122 | 03 | 3 |
| 037 | 15 | E | 080 | 32 | X:T | 123 | 02 | 2 |
| 038 | 71 | SBR | 081 | 69 | OP | 124 | 75 | - |
| 039 | 23 | LNX | 082 | 06 | 06 | 125 | 43 | RCL |
| 040 | 71 | SBR | 083 | 98 | ADV | 126 | 42 | 42 |
| 041 | 24 | CE | 084 | 71 | SBR | 127 | 54 | ) |
| 042 | 71 | SBR | 085 | 28 | LOG | 128 | 42 | STO |

TABLE A-2 (Continued)

| Loc | Code | Key | Loc | Code | Key | Loc | Code | Key |
|---|---|---|---|---|---|---|---|---|
| 129 | 47 | 47 | 175 | 37 | P/R | 221 | 54 | ) |
| 130 | 76 | LBL | 176 | 01 | 1 | 222 | 36 | PGM |
| 131 | 65 | x | 177 | 36 | PGM | 223 | 02 | 02 |
| 132 | 53 | ( | 178 | 02 | 02 | 224 | 16 | A |
| 133 | 73 | RC* | 179 | 14 | D | 225 | 73 | RC* |
| 134 | 47 | 47 | 180 | 53 | ( | 226 | 01 | 01 |
| 135 | 75 | - | 181 | 03 | 3 | 227 | 92 | RTN |
| 136 | 73 | RC* | 182 | 07 | 7 | 228 | 76 | LBL |
| 137 | 49 | 49 | 183 | 75 | - | 229 | 37 | P/R |
| 138 | 54 | ) | 184 | 43 | RCL | 230 | 25 | CLR |
| 139 | 71 | SBR | 185 | 42 | 42 | 231 | 92 | RTN |
| 140 | 61 | GTO | 186 | 54 | ) | 232 | 76 | LBL |
| 141 | 72 | ST* | 187 | 42 | STO | 233 | 28 | LOG |
| 142 | 01 | 01 | 188 | 47 | 47 | 234 | 25 | CLR |
| 143 | 01 | 1 | 189 | 76 | LBL | 235 | 69 | OP |
| 144 | 44 | SUM | 190 | 75 | - | 236 | 00 | 00 |
| 145 | 01 | 01 | 191 | 73 | RC* | 237 | 01 | 1 |
| 146 | 44 | SUM | 192 | 47 | 47 | 238 | 06 | 6 |
| 147 | 47 | 47 | 193 | 71 | SBR | 239 | 01 | 1 |
| 148 | 53 | ( | 194 | 03 | 03 | 240 | 03 | 3 |
| 149 | 43 | RCL | 195 | 55 | 55 | 241 | 03 | 3 |
| 150 | 43 | 43 | 106 | 01 | 1 | 242 | 07 | 7 |
| 151 | 85 | + | 197 | 44 | SUM | 243 | 01 | 1 |
| 152 | 03 | 3 | 198 | 47 | 47 | 244 | 03 | 3 |
| 153 | 03 | 3 | 199 | 53 | ( | 245 | 69 | OP |
| 154 | 54 | ) | 200 | 43 | RCL | 246 | 01 | 01 |
| 155 | 32 | X:T | 201 | 43 | 43 | 247 | 69 | OP |
| 156 | 43 | RCL | 202 | 85 | + | 248 | 05 | 05 |
| 157 | 47 | 47 | 203 | 03 | 3 | 249 | 05 | 5 |
| 158 | 22 | INV | 204 | 08 | 8 | 250 | 42 | STO |
| 159 | 67 | EQ | 205 | 54 | ) | 251 | 00 | 00 |
| 160 | 65 | x | 206 | 32 | X:T | 252 | 03 | 3 |
| 161 | 01 | 1 | 207 | 43 | RCL | 253 | 09 | 9 |
| 162 | 44 | SUM | 208 | 47 | 47 | 254 | 42 | STO |
| 163 | 49 | 49 | 209 | 22 | INV | 255 | 44 | 44 |
| 164 | 43 | RCL | 210 | 67 | EQ | 256 | 09 | 9 |
| 165 | 49 | 49 | 211 | 75 | - | 257 | 35 | 1/X |
| 166 | 22 | INV | 212 | 25 | CLR | 258 | 95 | = |
| 167 | 67 | EQ | 213 | 36 | PGM | 259 | 91 | R/S |
| 168 | 55 | ÷ | 214 | 02 | 02 | 260 | 71 | SBR |
| 169 | 36 | PGM | 215 | 15 | E | 261 | 36 | PGM |
| 170 | 02 | 02 | 216 | 53 | ( | 262 | 05 | 5 |
| 171 | 13 | C | 217 | 43 | RCL | 263 | 42 | STO |
| 172 | 32 | X:T | 218 | 42 | 42 | 264 | 00 | 00 |
| 173 | 00 | 0 | 219 | 85 | + | 265 | 03 | 3 |
| 174 | 67 | EQ | 220 | 01 | 1 | 266 | 04 | 4 |

TABLE A-2 (Continued)

| Loc | Code | Key | Loc | Code | Key | Loc | Code | Key |
|---|---|---|---|---|---|---|---|---|
| 267 | 42 | STO | 313 | 03 | 03 | 359 | 33 | $X^2$ |
| 268 | 44 | 44 | 314 | 54 | 54 | 360 | 85 | + |
| 269 | 04 | 4 | 315 | 25 | CLR | 361 | 07 | 7 |
| 270 | 93 | . | 316 | 69 | OP | 362 | 95 | = |
| 271 | 05 | 5 | 317 | 00 | 00 | 363 | 42 | STO |
| 272 | 35 | 1/X | 318 | 02 | 2 | 364 | 01 | 01 |
| 273 | 95 | = | 319 | 01 | 1 | 365 | 43 | RCL |
| 274 | 91 | R/S | 320 | 01 | 1 | 366 | 05 | 05 |
| 275 | 53 | ( | 321 | 03 | 3 | 367 | 32 | X:T |
| 276 | 24 | CE | 322 | 04 | 4 | 368 | 43 | RCL |
| 277 | 55 | ÷ | 323 | 01 | 1 | 369 | 07 | 07 |
| 278 | 43 | RCL | 324 | 02 | 2 | 370 | 22 | INV |
| 279 | 48 | 48 | 325 | 07 | 7 | 371 | 77 | GE |
| 280 | 54 | ) | 326 | 03 | 3 | 372 | 03 | 03 |
| 281 | 71 | SBR | 327 | 07 | 7 | 373 | 54 | 54 |
| 282 | 36 | PGM | 328 | 69 | OP | 374 | 01 | 1 |
| 283 | 92 | RTN | 329 | 04 | 04 | 375 | 44 | SUM |
| 284 | 76 | LBL | 330 | 69 | OP | 376 | 01 | 01 |
| 285 | 36 | PGM | 331 | 05 | 05 | 377 | 73 | RC* |
| 286 | 32 | X:T | 332 | 43 | RCL | 378 | 01 | 01 |
| 287 | 73 | RC* | 333 | 42 | 42 | 379 | 22 | INV |
| 288 | 44 | 44 | 334 | 32 | X:T | 380 | 67 | EQ |
| 289 | 32 | X:T | 335 | 43 | RCL | 381 | 03 | 03 |
| 290 | 72 | ST* | 336 | 43 | 43 | 382 | 74 | 74 |
| 291 | 44 | 44 | 337 | 77 | GE | 383 | 43 | RCL |
| 292 | 01 | 1 | 338 | 42 | STO | 384 | 07 | 07 |
| 293 | 22 | INV | 339 | 01 | 1 | 385 | 44 | SUM |
| 294 | 44 | SUM | 340 | 22 | INV | 386 | 01 | 01 |
| 295 | 44 | 44 | 341 | 44 | SUM | 387 | 01 | 1 |
| 296 | 32 | X:T | 342 | 42 | 42 | 388 | 44 | SUM |
| 297 | 97 | DSZ | 343 | 61 | GTO | 389 | 05 | 05 |
| 298 | 00 | 00 | 344 | 43 | RCL | 390 | 43 | RCL |
| 299 | 36 | PGM | 345 | 76 | LBL | 391 | 03 | 03 |
| 300 | 92 | RTN | 346 | 42 | STO | 392 | 72 | ST* |
| 301 | 76 | LBL | 347 | 01 | 1 | 393 | 01 | 01 |
| 302 | 43 | RCL | 348 | 22 | INV | 394 | 92 | RTN |
| 303 | 53 | ( | 349 | 44 | SUM | 395 | 76 | LBL |
| 304 | 43 | RCL | 350 | 43 | 43 | 396 | 39 | COS |
| 305 | 42 | 42 | 351 | 61 | GTO | 397 | 43 | RCL |
| 306 | 85 | + | 352 | 43 | RCL | 398 | 37 | 37 |
| 307 | 43 | RCL | 353 | 00 | 0 | 399 | 92 | RTN |
| 308 | 43 | 43 | 354 | 92 | RTN | 400 | 76 | LBL |
| 309 | 54 | ) | 355 | 42 | STO | 401 | 23 | LNX |
| 310 | 32 | X:T | 356 | 03 | 03 | 402 | 25 | CLR |
| 311 | 02 | 2 | 357 | 43 | RCL | 403 | 42 | STO |
| 312 | 77 | GE | 358 | 07 | 07 | 404 | 42 | 42 |

TABLE A-2 (Continued)

| Loc | Code | Key | Loc | Code | Key | Loc | Code | Key |
|---|---|---|---|---|---|---|---|---|
| 405 | 76 | LBL | 451 | 42 | STO | 497 | 00 | 0 |
| 406 | 32 | X:T | 452 | 43 | 43 | 498 | 00 | 0 |
| 407 | 01 | 1 | 453 | 76 | LBL | 499 | 00 | 0 |
| 408 | 44 | SUM | 454 | 34 | ⌐X | 500 | 00 | 0 |
| 409 | 42 | 42 | 455 | 01 | 1 | 501 | 00 | 0 |
| 410 | 02 | 2 | 456 | 44 | SUM | 502 | 00 | 0 |
| 411 | 09 | 9 | 457 | 43 | 43 | 503 | 00 | 0 |
| 412 | 32 | X:T | 458 | 03 | 3 | 504 | 00 | 0 |
| 413 | 53 | ( | 459 | 05 | 5 | 505 | 00 | 0 |
| 414 | 03 | 3 | 460 | 32 | X:T | 506 | 00 | 0 |
| 415 | 02 | 2 | 461 | 53 | ( | 507 | 00 | 0 |
| 416 | 75 | - | 462 | 03 | 3 | 508 | 00 | 0 |
| 417 | 43 | RCL | 463 | 02 | 2 | 509 | 00 | 0 |
| 418 | 42 | 42 | 464 | 85 | + | 510 | 00 | 0 |
| 419 | 54 | ) | 465 | 43 | RCL | 511 | 00 | 0 |
| 420 | 42 | STO | 466 | 43 | 43 | 512 | 76 | LBL |
| 421 | 45 | 45 | 467 | 54 | ) | 513 | 61 | GTO |
| 422 | 67 | EQ | 468 | 42 | STO | 514 | 32 | X:T |
| 423 | 33 | $X^2$ | 469 | 45 | 45 | 515 | 02 | 2 |
| 424 | 53 | ( | 470 | 67 | EQ | 516 | 22 | INV |
| 425 | 43 | RCL | 471 | 35 | 1/X | 517 | 77 | GE |
| 426 | 32 | 32 | 472 | 53 | ( | 518 | 37 | P/R |
| 427 | 75 | - | 473 | 43 | RCL | 519 | 02 | 2 |
| 428 | 73 | RC* | 474 | 32 | 32 | 520 | 94 | +/- |
| 429 | 45 | 45 | 475 | 75 | - | 521 | 77 | GE |
| 430 | 54 | ) | 476 | 73 | RC* | 522 | 37 | P/R |
| 431 | 71 | SBR | 477 | 45 | 45 | 523 | 00 | 0 |
| 432 | 61 | GTO | 478 | 54 | ) | 524 | 77 | GE |
| 433 | 50 | IxI | 479 | 71 | SBR | 525 | 05 | 5 |
| 434 | 32 | X:T | 480 | 61 | GTO | 526 | 39 | 39 |
| 435 | 93 | . | 481 | 50 | IxI | 527 | 53 | ( |
| 436 | 00 | 0 | 482 | 32 | X:T | 528 | 32 | X:T |
| 437 | 01 | 1 | 483 | 93 | . | 529 | 33 | $X^2$ |
| 438 | 22 | INV | 484 | 00 | 0 | 530 | 65 | x |
| 439 | 77 | GE | 485 | 01 | 1 | 531 | 04 | 4 |
| 440 | 32 | X:T | 486 | 22 | INV | 532 | 93 | . |
| 441 | 76 | LBL | 487 | 77 | GE | 533 | 07 | 7 |
| 442 | 33 | $X^2$ | 488 | 34 | ⌐X | 534 | 94 | +/- |
| 443 | 01 | 1 | 489 | 76 | LBL | 535 | 54 | ) |
| 444 | 22 | INV | 490 | 35 | 1/X | 536 | 22 | INV |
| 445 | 44 | SUM | 491 | 01 | 1 | 537 | 23 | LNX |
| 446 | 42 | 42 | 492 | 22 | INV | 538 | 92 | RTN |
| 447 | 92 | RTN | 493 | 44 | SUM | 539 | 53 | ( |
| 448 | 76 | LBL | 494 | 43 | 43 | 540 | 32 | X:T |
| 449 | 24 | CE | 495 | 92 | RTN | 541 | 65 | x |
| 450 | 25 | CLR | 496 | 00 | 0 | 542 | 53 | ( |

TABLE A-2 (Continued)

| Loc | Code | Key | Loc | Code | Key | Loc | Code | Key |
|---|---|---|---|---|---|---|---|---|
| 543 | 24 | CE | 548 | 94 | +/- | 552 | 54 | ) |
| 544 | 65 | x | 549 | 85 | + | 553 | 22 | INV |
| 545 | 04 | 4 | 550 | 03 | 3 | 554 | 23 | LNX |
| 546 | 93 | . | 551 | 54 | ) | 555 | 92 | RTN |
| 547 | 07 | 7 | | | | | | |

# APPENDIX B

## SOLUTION OF SIMULTANEOUS EQUATIONS BY CHOLESKY'S METHOD

The software development of the Pulse Pile-Up Corrector required selection of a method for solving the sets of simultaneous linear equations. Two possible approaches were considered:

1. An Iterative Approach
2. A Direct Approach

The iterative technique was highly attractive, serious consideration was given to using the Gauss-Seidel method (Ketter, p. 85). This is a straightforward successive iteration technique which is easily programmed. Also this approach would not require division which would potentially be troublesome with fixed point arithmetic. To test the method two simple equations similar to those expected with the pulse pile-up algorithm were programmed for a Gauss-Seidel solution on a TI-59 programable calculator. The calculator counted the iterations and found that $10^3$ to $10^6$ iterations were required. In addition the iterative program was much slower than a preprogrammed solution technique based on a direct method. A consequent review of references on numerical technique by Ralston (1965) and Ketter (1969) led to selection of a direct approach known as Cholesky's (or sometimes Crout's) method.

Cholesky's method is a direct method of solution of a set of linear equations. It is a highly organized form of Gaussian Elimination. The method is not limited in any way (except round off), it can handle

any number of equations with any random coefficient distribution. Being a direct method it is not bothered by convergence problems as an iterative method (The TI-59 uses Cholesky's method). A description of Cholesky's method follows. The primary reference for this material is Ketter (1969).

Given the need to solve a set of linear equations:

$$[A][X] = [C] \qquad \text{B-1}$$

The brackets denote matrices. [A] is an nXn coefficient matrix, [X] is a nX1 unknown matrix, and [C] is a nX1 constant matrix. In expanded form B-1 would look like:

$$\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & \\ a_{31} & & & \\ \vdots & & & \\ a_{n1} & & & \end{vmatrix} \begin{Vmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Vmatrix} = \begin{vmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{vmatrix} \qquad \text{B-2}$$

If, instead of B-1 and B-2 the equations were in the form of:

$$\begin{vmatrix} 1 & t_{12} & t_{13} & \cdots & t_{1n} \\ 0 & 1 & t_{23} & \cdots & t_{2n} \\ 0 & 0 & 1 & t_{34} \cdots & t_{3n} \\ \vdots & \vdots & & & \\ 0 & 0 & 0 \cdots 0 & & 1 \end{vmatrix} \begin{Vmatrix} x_1 \\ x_2 \\ x_3 \\ \\ x_n \end{Vmatrix} = \begin{vmatrix} k_1 \\ k_2 \\ k_3 \\ \\ k_n \end{vmatrix} \qquad \text{B-3}$$

Then the unknowns could easily be solved by back substitution.

$$x_n = k_n$$

$$x_{n-1} = k_{n-1} - x_n t_{n-1,n}$$

etc. B-4

The problem then is to convert B-2 to the form of B-3. In matrix notation the starting equations are:

$$[A][X] = [C] \qquad \text{B-1}$$

$$[T][X] = [K] \qquad \text{B-5}$$

Where [T] was shown to be an upper triangular matrix. Now define a lower triangular matrix [L]:

$$[L] = \begin{vmatrix} l_{11} & 0 & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 & \cdots & 0 \\ l_{31} & l_{32} & l_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & & \\ l_{n1} & & & & l_{nn} \end{vmatrix} \qquad \text{B-6}$$

Matrix [L] will be used to convert B-1 into B-2. This will require that [L] satisfy the relationship:

$$[L][T] = [A] \qquad \text{B-7}$$

Now it becomes necessary to evaluate [L]. First multiply Equation B-5 by [L]:

$$[L][T][X] = [L][K] \qquad \text{B-8}$$

Then substitute [A] for [L][K].

$$[A][X] = [L][K] \qquad \text{B-9}$$

And substitute from B-1 for [A][X]:

$$[C] = [L][K] \qquad \text{B-10}$$

Equation B-10 establishes the relationship between the two constant matrices [C] and [K], of B-1 and B-5.

At this point the concept of an augmented matrix is introduced. An augmented matrix is formed by joining the square coefficient matrix with the column matrix of constants as shown in B-11:

$$[A\,|\,C] = \left|\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & c_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & c_2 \\ \vdots & & \ddots & & \\ a_{n1} & & \cdots & a_{nn} & c_n \end{array}\right| \qquad \text{B-11}$$

Equation B-11 is read as [A] augmented with [C]. Now an examination of B-7 and B-10 shows that:

$$[A][C] = \{[L][T]\,|\,[L][K]\} \qquad \text{B-12}$$

It can be shown that B-12 can also be written as:

$$[A][C] = [L][T\,|\,K] \qquad \text{B-13}$$

(multiply out and reduce the LHS's of B-12 and B-13 to show the equivalence).

Now B-13 is written in expanded form for a 3X3 system:

$$\left|\begin{array}{ccc:c} a_{11} & a_{12} & a_{13} & c_1 \\ a_{21} & a_{22} & a_{23} & c_2 \\ a_{31} & a_{32} & a_{33} & c_3 \end{array}\right| = \left|\begin{array}{ccc} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{array}\right| \left|\begin{array}{ccc:c} 1 & t_{12} & t_{13} & k_1 \\ 0 & 1 & t_{33} & k_2 \\ 0 & 0 & 1 & k_3 \end{array}\right| \qquad \text{B-14}$$

If the R.H.S. of Equation B-14 is multiplied out and equated term-for-term with the L.H.S. and the resulting equations solved for the l's and t's the general result is:

$$l_{ij} = a_{ij} - \sum_{k=1}^{j=1} l_{ik}t_{ij};\ l_{i1}=a_{i1} \text{ for } i>j \qquad \text{B-15}$$

$$t_{ij} = \frac{1}{l_{ij}}\ a_{ij} - \sum^{j=1} l_{ik}t_{kj}\ ;\ t_{1j} = \frac{a_{1j}}{a_{11}} \text{ for } i<j \qquad \text{B-16}$$

Note that the k terms (from B-14) in an nXn system are represented in Equations B-15 and B-16 as the $t_{i,n+1}$ terms.

Both the $l_{ij}$ terms and $t_{ij}$ terms depend on other l and t terms. This implies a definite order in which the terms should be calculated. An easy method to visualize the required order is to first construct from matrices [L][T¦K] a decomposed matrix that omits the zero terms.

$$\left|\begin{array}{ccccc:c} l_{11} & t_{12} & t_{12} & \cdots & t_{1n} & k_1 \\ l_{21} & l_{22} & t_{23} & \cdots & t_{2n} & k_2 \\ l_{31} & l_{32} & l_{33} & t_{34} & \cdots & k_3 \\ & & & & t_{n-1,n} & \vdots \\ l_{n1} & \cdots & & & l_{nn} & k_n \end{array}\right| \qquad \text{B-17}$$

By inspection of B-15 and B-16 modify B-17 into a series of sectors. Then it is necessary to first calculate all terms in the first sector before any terms in the second sector and so forth.

$$\left|\begin{array}{cccc} l_{11} & t_{12} & t_{13} & t_{14}\text{-----1st} \\ l_{21} & l_{22} & t_{23} & t_{24}\text{-----2nd} \\ l_{31} & l_{32} & l_{33} & t_{34}\text{-----3rd} \\ l_{41} & l_{42} & l_{43} & l_{44}\text{-----4th} \\ \vdots & \vdots & \vdots & \vdots \end{array}\right| \qquad \text{B-18}$$

Finally it is only necessary to write a concise expression for the $x_m$ and the equations of the algorithm B-15, B-16, and B-19 are complete:

$$x_m = k_m - \sum_{j=m+1}^{n} t_{mj} x_j \qquad \text{B-19}$$

Programming the algorithm is a bookkeeping exercise. First the decomposed matrix B-18 must be formed from the [A¦C] matrix using Equation B-15 and B-16, then the $x_m$'s are found from the [T¦K] matrix using expression B-19.

# VITA

George Roland Gilbert was born in Pulaski, Tennessee on October 8, 1946. He attended elementary and secondary schools in Nashville, Tennessee and was graduated from Donelson High School in June 1964. In 1964 he entered Vanderbilt University and in June 1968 received the Bachelor of Engineering degree in Electrical Engineering.

Upon graduation from Vanderbilt University, Mr. Gilbert entered the United States Navy, where he served for five years as a Naval Aviator and as a manager of Navy technical schools.

After leaving the Navy in 1973, he was employed as an Electrical Engineer with the United States Atomic Energy Commission, Monsanto Textiles Company, and Aramco Services Company before accepting his present position as an Electrical Engineer with the Tennessee Eastman Company.

Mr. Gilbert entered The University of Tennessee in 1974 and began study toward a Master's degree. This work was concurrent with his regular employment.

The author is a registered Professional Engineer in Tennessee, a member of the Institute of Electrical and Electronics Engineers, the Instrument Society of America, Tau Beta Pi, and Eta Kappa Nu.

He is married to the former Sylvia Annelle Salazar of Corpus Christi, Texas, and has a daughter, Jennifer Lynn.