DESIGN AND DEVELOPMENT STUDY

FOR A SPACE BASE DIGITAL COMMAND SYSTEM

ER71-4005

FINAL REPORT

Period 1 January - 31 December 1970

27 December 1970

CR 114817

DESIGN AND DEVELOPMENT STUDY

FOR A SPACE BASE DIGITAL COMMAND SYSTEM

ER71-4005

FINAL REPORT

Period 1 January - 31 December 1970

27 December 1970

Prepared Under

NASA CONTRACT NAS 9-10419

for

NASA Manned Spacecraft Center
Houston, Texas

Prepared by: _____
A.J. Stiffler
Technical Director

_____
R. Adams
Senior Engineer

RAYTHEON COMPANY
EQUIPMENT DIVISION
Aerospace Systems
Sudbury, Massachusetts 01776

CONTENTS

1.0 INTRODUCTION

This report documents the results of a design and development study for a Space Base Digital Command System. The purpose of the command system is to serve as the primary control link between ground stations and a space base and between the space base and its associated satellites. Thus, the system must be capable of accepting messages from the ground and relaying them either to a specified location within the space base or to the appropriate subsidiary satellite. It must also be able to accept messages originating on the base and to multiplex them in order of priority with those emanating from the ground for transmission to the associated satellites.

The system is subject to the following basic objectives:

a. The command messages are to be so encoded that the probability of an erroneous command does not exceed 10^{-18}.

b. The system is to be capable of sending up to 1000 messages per second, each message consisting of 18 bits of information plus redundancy sufficient to achieve the necessary error immunity.

c. The equipment is to be designed so that failure of any critical component does not render the system inoperative and so that a second failure of this same component results in a fail-safe condition.

d. The design is to be sufficiently flexible to allow future expansion with minimum modification.

In addition, since the command system is to be space-borne, power consumption, weight, and volume are all critical parameters and must be kept to a minimum consistent with the system objectives. This is particularly true of that portion of the system which is to be included in each of the subsidiary satellites.

The following sections of this report describe the proposed system and discuss the considerations which lead to the recommended configuration. Section 2 examines in greater detail the system requirements and outlines the overall system organization.

Section 3 is concerned with the command controller; i.e., the device for storing and routing the various command messages in order of their priority and for keeping track of whether or not each message has been received at its intended location.

Section 4 then investigates a number of coding alternatives and compares the complexities of the resulting decoders.

Section 5 examines the impact of micro-electronic techniques on the system implementation.

Section 6 is concerned with methods for achieving the desired system reliability. Finally, the major conclusions of the study are summarized in Section 7.

2.0 SYSTEM REQUIREMENTS AND OVERALL ORGANIZATION

2.1 SYSTEM REQUIREMENTS

2.1.1 Priority Requirements

The Apollo command link (up-data link) recognizes three classes of commands: real-time commands (RTC) to turn equipment on and off; computer data transfer to read data into the guidance computer; and central timing equipment up-date instructions. Other satellite and deep-space command systems have generally required only two classes of commands, real-time commands and data transfer or quantitive commands. Although the space base will be a considerably more complex system than any of its predecessors, it seems reasonable to assume that the same two categories, real-time commands and data-transfers (the latter category also possibly including timing up-date instructions) will adequately characterize the types of commands to be handled by the space base command system.

Thus, at least two priority levels are clearly called for: if real-time commands and data transfer commands are both awaiting access to the command link, the former should presumably be serviced first. The extent to which other levels of priority are required is less obvious. At minimum, two additional priority levels would be useful, one to distinguish between emergency and routine real-time commands, and one to identify data which must be transferred relatively quickly.

Further refinements could conceivably also be useful (e.g., a data transfer level which takes precedence over certain real-time commands). For purposes of this study, however, only four priority levels will be considered for two reasons: (1) It is felt that the bulk of the command link traffic (in terms of volume) will be at level four priority and hence that near real-time service should be available whenever it is needed (see Section 2.2.3). (2) Additional sub-levels of priority could be added later within the present framework, should the need arise, so long as the basic operational modes are not drastically changed.

2.1.2 Acknowledgement Delay

Since a command must be kept in standby storage during the interval between its transmission and its acknowledgement, the amount of this delay has an obvious bearing on the system design. It will be assumed here that the maximum acknowledgement delay will not exceed 1/2 second. This represents the time required for a message to travel half way around the earth via a synchronous satellite and return by the same route, with some allowance made for decoding, and other processing delays. It is conceivable that a link involving two synchronous relay satellites might be required to connect the space base to some vehicle, in which case the acknowledgement delay could be as much as one second. At worst, such a possibility could double the required storage

capacity. Since such links even if they do exist, will presumably involve only a small fraction of the total command system traffic, their effect on the amount of storage needed for the system should be minor in any event.

2.1.3 Traffic Requirements

Both the ground to space base and space base to subsidiary vehicle command links are to have a capacity of 1000 real-time commands per second. A real-time command will consist of a 4-bit vehicle address, a 6-bit subsystem address, and an 8-bit instruction for a total of 18 bits in all. (Presumably, the priority assigned to a command can be deduced from the 8-bit instruction. Since this instruction certainly distinguishes between a real-time command and a data transfer, at most one additional bit will be needed to distinguish between level one and two and between level three and four priorities. For the present study, the 18 bit real-time command structure will be assumed adequate to contain the necessary priority information as well.) Since the command is to be encoded prior to transmission, it will be most efficient to block all command sequences into 18-bit segments or command words. Because data transfers may involve considerably more than 18 bits, several command words may be required for each data transfer.

On the basis of prior space mission experience, it seems unlikely that the rate at which real-time commands must be transmitted will ever approach the thousand word/second link capacity, even when the vastly increased complexity of the space base system is taken into account. Thus, the command system design will be predicted on the assumption that real-time commands, since they will be given special priority, and since they are relatively rare, can in fact be accommodated in near real-time.

2.1.4 Multiplexing Requirements

A multiplexing situation arises in the command system due to the fact that commands to be transmitted to remote vehicles can originate both on the ground and in the space base itself. Similarly, since a message received from the ground can be destined either for the space base or for one of its subsidiary vehicles, a demultiplexing capability is also required. So far as the command system multiplexer is concerned two situations can be postulated: (1) The multiplexer is to be designed to accept inputs from the up-link decoder and from any of several sources on the space base. (2) The multiplexer has only three inputs, one from the up-link decoder, one from the space base data bus through which all commands originating on board must pass, and a third consisting of unacknowledged commands which must be retransmitted. Likewise, the demultiplexer could be designed: (1) to

direct commands either to the encoder or to any one of several on-board destinations; or (2) to route commands only to the encoder and to the data bus. In both cases, option (2) will be postulated here. The reason for this is that the resulting design will be more compatible with the internal space base communication system, as it presently appears to be evolving. Again, this assumption while convenient, is not critical to the system design, and could be changed at a later date should the direct access approach prove more attractive. The only condition which will be postulated here, then, (and this condition is not absolutely necessary) is that the data bus capacity be greater than that of the command system by an amount sufficient to treat the bus as nearly instantaneous (i.e., to intoduce delays not exceeding one word period).

2.1.5 Postulated Noise Environment

The signal-to-thermal-noise ratio will be assumed to be such that the uncoded bit-error probability, in the absence of other types of interference, will be on the order of 10^{-4}. This assumption will be applied to both the up-link (from the ground) and the cross-link (to subsidiary vehicles) channels.

Although the preliminary coding trade-offs will be made on the basis of the above operating situation, the coding schemes surviving this first screening will then be evaluated in terms of their performance at much

lower signal-to-noise ratios. Since a link could be either partially or totally blocked for any of a number of reasons during a transmission it is clearly important that the error probability remains low regardless of the actual signal-to-noise ratio.

Since one subject of considerable interest in this study involves the extent to which erasures are useful in achieving the desired error probability, a modulation scheme must be postulated so that the bit error/erasure tradeoff can be determined. The most likely candidate for a modulation scheme, in view both of its performance and of its wide use in command and other digital data links, is two-phase (and possibly four-phase) PSK modulation. This is the modulation scheme which will be postulated here.

2.2 OVERALL SYSTEM ORGANIZATION

A block diagram of a space-base command system is shown in Figure 2.1. It should be emphasized that this is a conceptual design only, included here solely to facilitate the subsequent discussion. The actual system to be evolved in the following sections of this report will not necessarily resemble that of Figure 2.1 in all its details.

2.2.1 Priority Structure

As discussed in Section 2.1.1, four priority levels appear to be adequate, at least for preliminary design purposes. A priority structure which is consistent

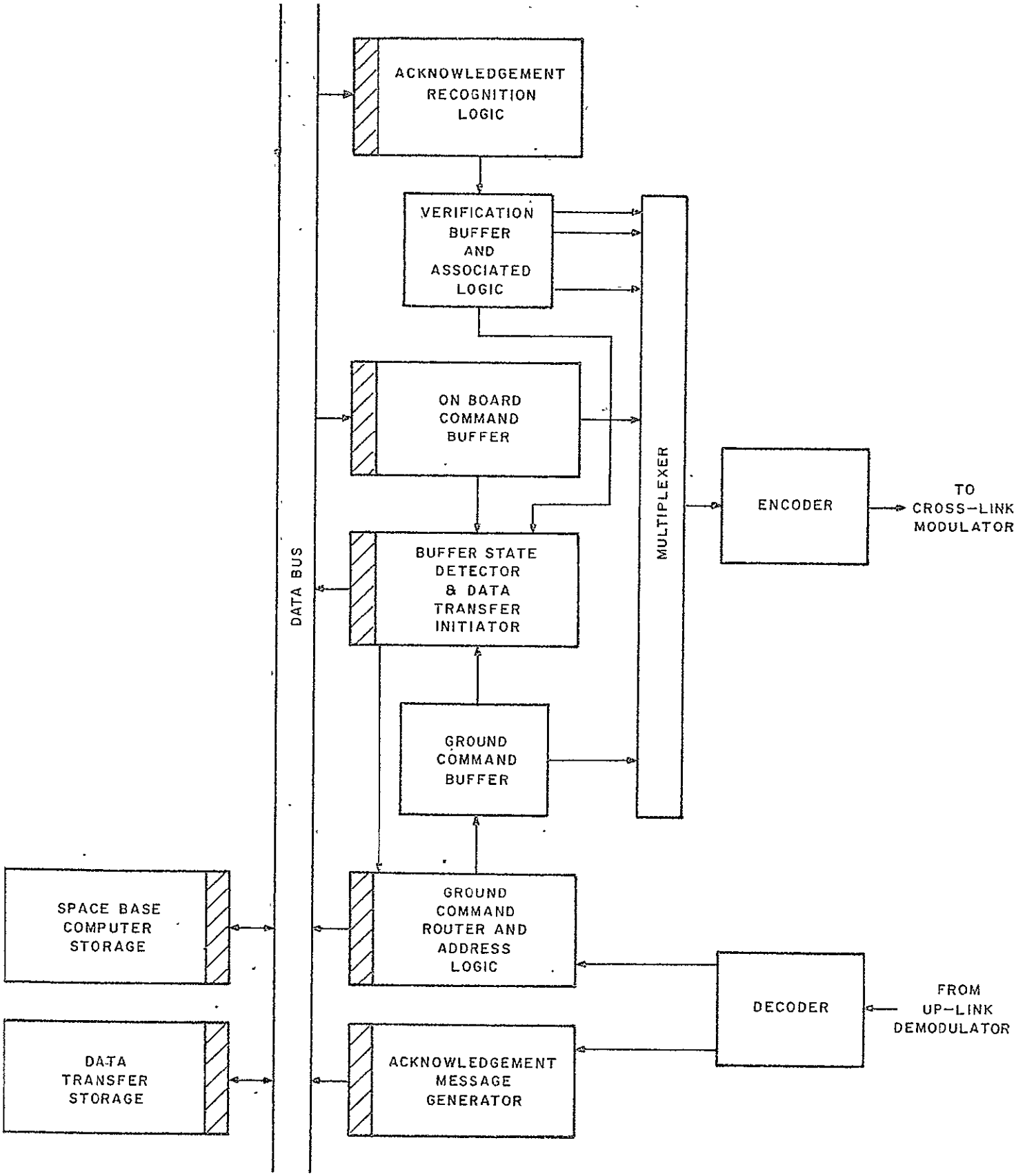


FIGURE 2.1

COMMAND SYSTEM CONCEPTUAL BLOCK DIAGRAM

with the requirements outlined in Section 2.1.1 is summarized in Table 2.1.

In view of the discussion in Section 2.1.1, the only features of this structure which require further justification are the specifics of the various modes of operation. Other emergency modes, in particular, are obviously possible. The system might be designed to transmit the emergency command as every other (or every third, etc.) command word, for example, rather than repeatedly. This would allow other commands to be transmitted without (necessarily) appreciably increasing the delay in getting an emergency command to its destination. This would also allow simultaneous transmission of two (or more) emergency mode commands. On the other hand, it may well be that an emergency mode is needed because the subsidiary vehicle is violently out-of-control so that the command receiver antenna is properly directed only on occasion. In this case, transmitting the emergency command only every other time could conceivably double the expected time needed to get the command through. Because of this possibility, because emergency commands should be rare events, and in order not to complicate the system unnecessarily, it was decided to take the simplest approach which provided a means of successfully transmitting a command in the minimum possible time.

TABLE 2.1 COMMAND SYSTEM PRIORITY LEVELS

Priority Level	Designation	Mode of Operation
1	Emergency	Emergency command is transmitted repeatedly, to the exclusion of all other commands, until its receipt is acknowledged. (Since a number of emergency commands will generally be received before the transmitter receives an acknowledgement, the post-decoder logic must be such that only the first of any specific emergency mode command is acted upon). The emergency mode is interrupted only on receipt of an acknowledgement or of a countermanding command.
2	Real-Time Commands	Level two commands are serviced immediately, in order of receipt, except when an emergency command enters the system.
3	Urgent Date Transfers	Level three commands are sent in order of arrival but only when no level one or level two commands are awaiting transmission.
4	Routine Data Transfers	Transmitted only when all level one, two and three commands have been sent and acknowledged.

Modes two and four are the ordinary modes of operation and require little explanation. Priority four commands are sent only when all higher priority commands have been acknowledged. While this can result in some "down-time", it would seem to be a worthwhile feature in that it allows level four data to be sent in comparatively large blocks without frequent interruptions to retransmit earlier unacknowledged commands. It also assures that urgent data is successfully transmitted before any attempt is made to transmit routine data.

2.2.2 Acknowledgement Procedure

The command acknowledgement routine must clearly satisfy the following minimum requirements:

(1) It must be reliable

(2) It must be unambiguous; i.e., the transmitter must be able to associate each acknowledgement uniquely with some transmitted command.

In addition, it should be designed to minimize the burden on the telemetry link. Presumably, part of the telemetry channel capacity is needed for purposes other than acknowledging commands.

Of the several procedures which were considered one, the "coherent acknowledgement" techniques, seems to be by far the most attractive, at least at this point in the study. This technique satisfies both of the objectives listed above, places minimum burden on the

telemetry link, and in addition provides a constant indication at the transmitter of the status of each of the potential command receivers.

The approach is as follows: Before any commands are transmitted, the transmitter initiates an acquisition procedure in which the receiver phase-locked loops (assuming a coherent system) are brought into lock and bit and word synchronization is established at each of the potential receivers. (This, of course, is necessary regardless of the acknowledgement scheme used.) After this is accomplished, the transmitter sends a command in turn to each of the command receivers. The returned acknowledgements then establishes the delay associated with each receiver. Most of the acknowledgement procedures considered involve an initial acquisition phase of this sort. In the coherent acknowledgement scheme, however, the command receivers are required to acknowledge every transmitted command. If a command decoder recognizes its own address in the decoded command, it it acknowledges the command with a "one"; if it does not recognize its own address, or if it is unable to decode the received sequence, it transmits a "zero". Thus, simply by counting the number of received acknowledgements from any particular recipient, the transmitter is able to keep track of which commands is presently being acknowledged.

The major advantage of this approach is that the logic needed to keep track of the status of each potential receiver is minimized and the sender receives positive information, with minimum delay, as to whether or not a given command reached its destination. The technique does require each command receiver to transmit one bit of information for every transmitted command, regardless of the address of the commands. A possible alternative in the case of the space-base/subsidiary vehicle links would be to have a receiver acknowledge a command only if it recognizes its own address. While this would possibly decrease the average amount of data to be transmitted over any one telemetry link, it would tend to increase the peak demand on these links since, in general, the acknowledgement message would have to contain more information in order to identify which command is being acknowledged. This would not necessarily be true if the command system were kept informed, by other means, of the relative locations of each of the receivers since then it would be able to determine the expected delay between a transmitted command and the receipt of an acknowledgement. (Since one command word period corresponds to a round-trip distance of approximately 93 miles, the precision needed for this purpose is not extreme.) The disadvantage of this last method, of course, is in the increased complexity of the logic needed to associate acknowledgements with commands.

It is apparent that all of approaches are compatible; the major differences have to do with the logic needed to identify which command is being acknowledged. Because the coherent acknowledgement scheme appears to be the easiest to implement, and because it has the additional advantage of providing a constant check on the status of each potential command receiver, it will be postulated for purposes of this study. Should it subsequently develop that one of the other methods is preferable (e.g., if the coherent acknowledgement procedure overburdens the telemetry links) the necessary changes could be made, most likely without undue effect on the rest of the system.

2.2.3 Traffic Routing and Multiplexing Organization

The approach taken here, is as follows (cf. Figure 2.1): All valid real-time commands received from the ground are examined to see if they are intended for the space base itself or are to be relayed on to one of its subsidiary vehicles. In the former instance they are adapted to the data bus format and sent over the data bus to their destination; in the latter case, they are transferred immediately to the real-time command buffer* to await encoding and transmission over the cross-link channel.

*It is convenient for purposes of this discussion to identify a number of different buffers. Actually the functions of many of these buffers will be combined.

Data transfer command words of either level three or level four priority destined for a subsidiary vehicle are placed in data buffer storage until the buffer state detector indicates that the two real-time command buffers are empty. When this happens, urgent data transfer words are retrieved from buffer storage for encoding and transmission. If the verification buffer is also empty (indicating all transmitted commands have been acknowledged) routine data commands are then encoded and transmitted, in order of their receipt. Data transfers intended to remain on board the space base are, of course, routed to their proper destination via the data bus.

Real-time commands originating on board are immediately sent over the data bus to the respective command buffer to await transmission. Data transfers originating on board are treated in exactly the same way as those received from the ground; the data words are placed in data buffer storage from which they are removed for encoding and transmission in order of receipt.

The multiplexer samples the two command buffers and those outputs of the verification buffer corresponding to transmitted commands which were not, but should have been, acknowledged prior to the sampling instant. The command word must be read into the encoder in less time than it takes to transmit the encoded command, since some

time is needed to complete the encoding. Thus, the multiplexer will have sufficient time to check the verification buffer and still find another valid input before the next command actually has to be read into the encoder.

To summarize, the rationale for the command system organization outlined here is based on the assumption that a relatively small fraction of the command system capacity is ever needed for real-time commands alone, so that the excess capacity is sufficient to accommodate all data transfers without undue delay. (The priority structure does make provisions for urgent data transfers, however.) This allows real-time commands to be serviced in near real-time even though these commands can originate from several sources. Moreover, only small buffers are actually needed in multiplexing this data. The bulk of the data, on the other hand, since it need not be handled in real-time, can be held in buffer storage and called for when other, more urgent, commands do not have to be transmitted. Data transfer messages which do exceed the buffer storage capacity are routed to temporary storage via the data bus. These words can later be requested when the data buffer storage has been emptied.

2.3 SUMMARY

A conceptual design for the space base command system has been presented (cf. Figure 2.1). The system and environmental constraint leading to this design and postulated for the remainder of this study are:

(1) The command link traffic can be divided into two basic categories, real-time commands and data transfers. Commands falling in the first category are assumed to be relatively rare. (i.e., The probability is negligible that the rate at which real-time commands enter the system from all sources at any one instant exceeds 1000 commands per second.)

(2) Four priority levels, two for real-time and two for data transfers, are adequate at least for system design purposes.

(3) The time separating the transmission of a command and its acknowledgement will not exceed 1/2 second.

(4) Internal communications within the space base (via the data bus) will not involve appreciable delays (i.e., delays exceeding 1 millisecond, the command word period).

(5) Under ordinary conditions the uncoded bit error probabilities associated with both the up-link and th cross-link will be on the order of 10^{-4}. Command messages are to be transmitted using PSK modulation.

3.0 COMMAND CONTROLLER

3.1 SYSTEM ORGANIZATION (Reference Figure 3.1)

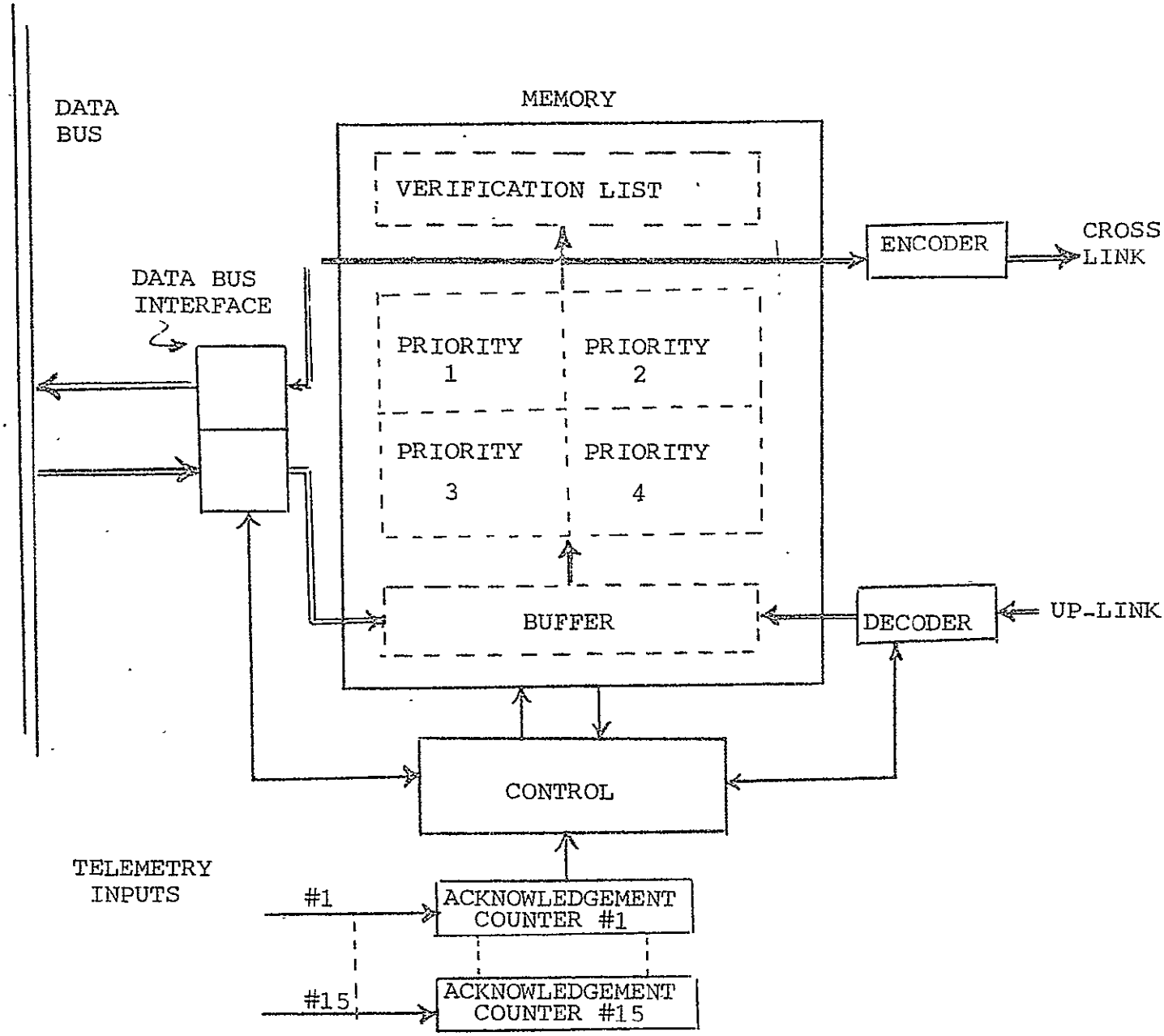
The proposed system organization is based on the recommended constraints of a four level priority system and a "coherent acknowledgement" scheme.

Command words for satellite vehicles, whether from the ground via up-link or within the space base via the data bus, are stored in a buffer memory until they can be forwarded to the encoder for transmittal via cross-link. Command words received via up-link for the space base are transferred directly to the data bus input register.

Commands are stored in the buffer memory under executive program control according to priority. A word which is transmitted is retained until a valid acknowledgement is received from the addressed vehicle. Failure to receive a valid acknowledgement will initiate a retransmission of the command.

3.1.1 Acknowledgement Verification

Under the proposed coherent acknowledgement scheme the command receivers acknowledge every transmitted command. A "one" is returned when a receiving vehicle decodes its own address and a "zero" returned when its own address is not recognized. The command controller contains a clock operating at the maximum transmittal rate of 1000 commands per second. The controller also utilizes a counter (modulo 512) for each of the 15



COMMAND CONTROLLER BLOCK DIAGRAM

FIGURE 3.1

possible receiving vehicles. An acknowledgement received via the telemetry link increments the counter one count.

3.2 ACKNOWLEDGEMENT VERIFICATION IMPLEMENTATION

Two methods of implementing the acknowledgement verification scheme are candidates for use in the command controller. One is based on the use of time-tags which are stored along with a command word in the verification memory. This will be called the "time-tag" verification method. The other scheme, called the "time-location" verification method, stores the transmitted command words in memory locations which correspond to a real-time count at transmission time. Each of the two methods will be examined in detail.

3.2.1 Time-Tag Verification Method

Each time a command is transmitted a verification word (reference Figure 3.2), is assembled and stored in the verification list. The 10 bits allocated for "N" will contain the command clock at the time a word is transmitted. The portion of the verification word designated "$2N-k_i + 1$" indicates the clock count when an acknowledgement is considered overdue. Here again N refers to the clock count at transmittal time, while "k_i" is the contents of the ith acknowledgement counter at that time. A count of 1 is added to allow for doppler affects.

FIGURE 3.2

VERIFICATION WORD FOR COMMAND WORDS OF PRIORITY LEVELS TWO AND THREE

10 BITS	10 BITS	18 BITS
Count Tag for Overdue Acknowledgement $2N-k_i + 1$	N = Local Word Count	Command Word for a Re-transmission

The system operates as follows:

A "1" acknowledgement from a particular receiving vehicle causes an interrupt in the command controller. The acknowledgement counter for the corresponding vehicle is queried and the verification list scanned for agreement with the "N" contents for that vehicle address. When agreement exists, the verification word is cleared, since this was a valid acknowledgement. Should no verification word with a value of N corresponding to the contents of the acknowledgement counter be located, an invalid verification will exist. An alarm condition is initiated and sent to the space base control center.

Overdue type "1" acknowledgements are discovered by scanning the verification list once each millisecond and comparing the present command clock contents with the "$2N-k_i + 1$" portion of the verification word.

Agreement indicates that a command was not acknowledged, in which case the command word is retransmitted and a new verification word assembled.

The acknowledgement verification sequence is flow charted in Figure 3.2.1. This sequence is divided into two parts. Part I is initiated by a logic "1" acknowledgement from one of the fifteen receiving vehicles via telemetry lines. Part II shows the check made every millisecond for any acknowledgements which might be overdue. The tag in the verification word, $2N-k_i + 1 = N + (N-k_i) + 1$, is the command word count, N, at the time a specific word is transmitted, plus $(N-k_i)$. Here k_i is the contents of acknowledgement counter, i, at the time the word is transmitted. Thus, $N-k_i + 1$ equals the round trip message and acknowledgement delay for a particular receiving vehicle, plus one count for doppler affect.

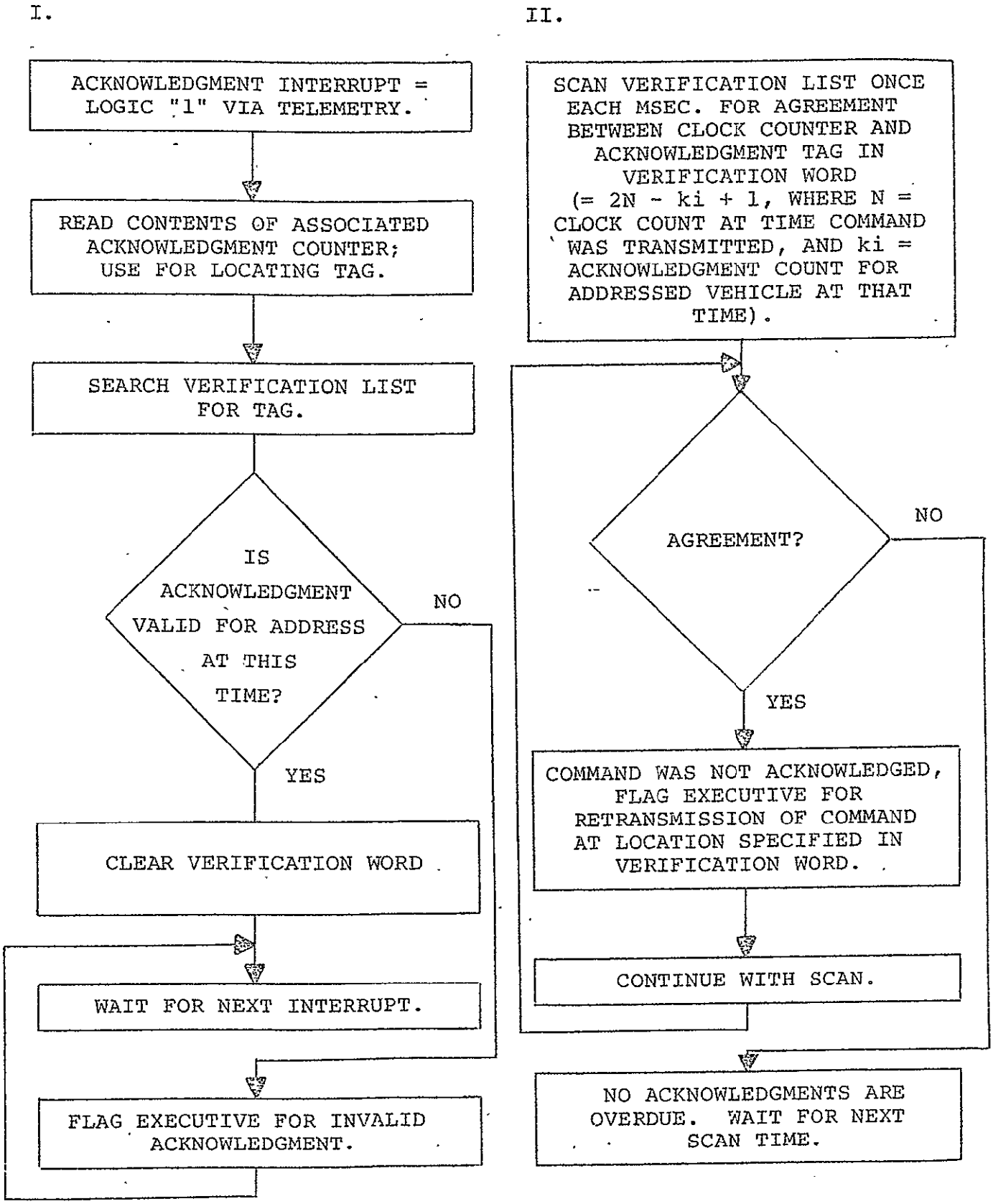
This approach to implementation of acknowledgement verification requirements suggests the use of an associative type memory. This type of memory organization will next be examined.

3.2.1.1 Associative Type Memory System

The type of storage required for the verification list can best be implemented with an associative memory. Level two and three verification words are selected according to content rather than by location. An associative

FIGURE 3.2.1

ACKNOWLEDGEMENT VERIFICATION SEQUENCE



memory can be addressed either by location or by content. A block diagram of the proposed memory system is shown in Figure 3.2.1.1. Basic operation of the system is as follows. The word which is to be used for search or compare is loaded into the interrogate register. All words in the memory matrix are simultaneously requested to form a comparison with the interrogate register, but only for those bits which correspond to a logic "1" in the mask register. Thus, by use of the mask register, the search can be restricted to desired bit positions. Each word in memory has an interrogate output line which indicates whether the data in that word matches the interrogate word.

The next step is to determine the address of the compare word, then read out that word onto the output bus. If more than one compare word is stored in memory, the search and address process must be repeated sequentially.

The interrogate output line for each word is sent to the detector matrix for division into X and Y address components. These components are stored in the X and Y multimatch resolvers and priority given to each resolver. Based on X and Y configurations, a specific word driver is activated for selection. The X and Y resolvers are sequential

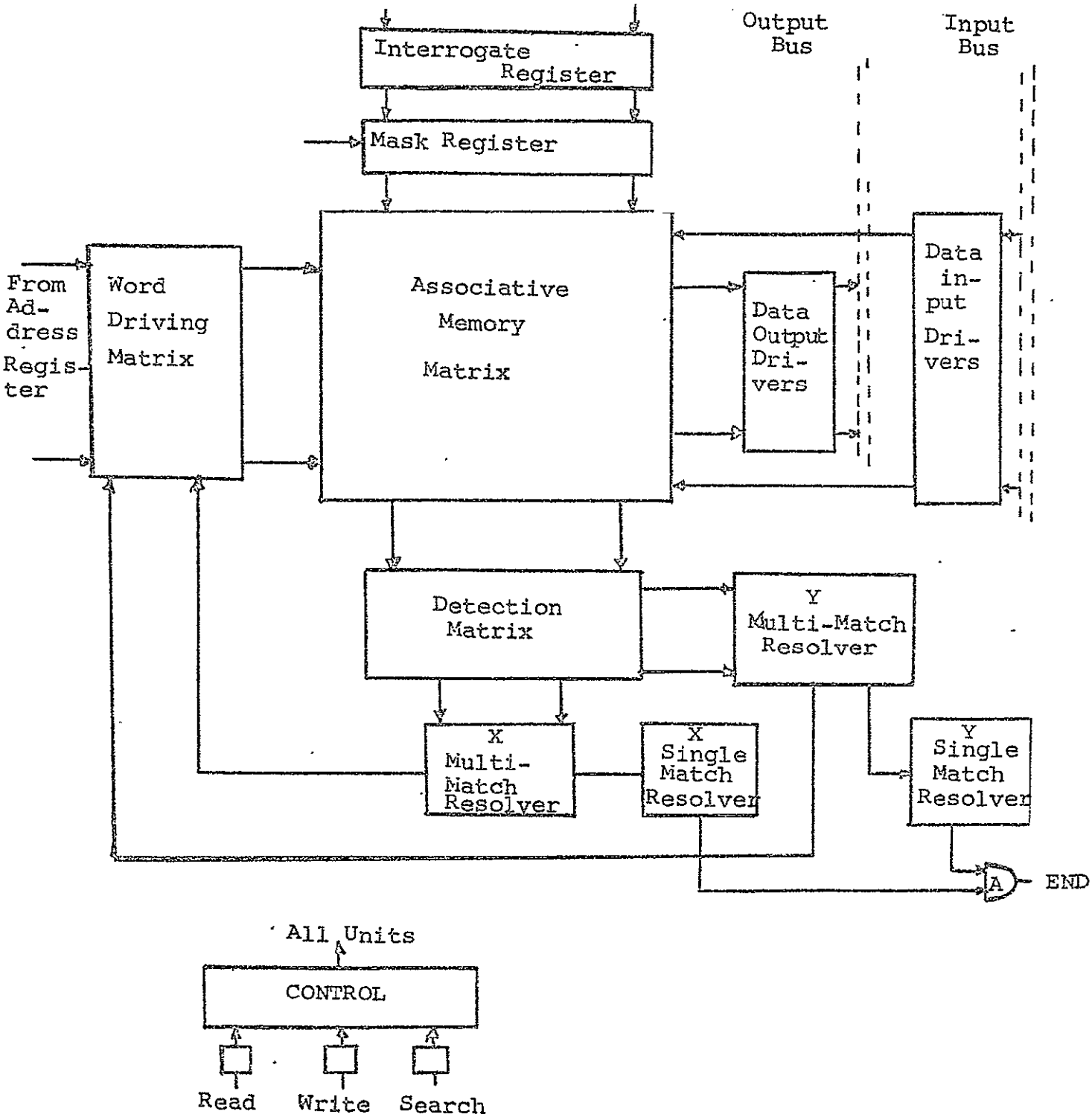


FIGURE 3.2.1.1

ASSOCIATIVE MEMORY SYSTEM

in operation and the response selection continues until the X and Y single match detectors detect the end of the current search.

A detector matrix is shown in Figure 3.2.1.2 for a 128 word memory section. The matching signals generated by each word-sense line of 128 words are sent to the AND gates which are divided into 16 groups as shown in Figure 3.2.1.2. The signals are then gated into the Y-multimatch resolving circuits by the Y set pulse, and are gated into the X-multimatch circuits by the X set pulse.

A logic diagram of the Y-multimatch resolving circuit is shown in Figure 3.2.1.3. A word matching signal in Group 1 will initially set FY-1 when Y set pulse is enabled. Any other matching signals in Groups 2 through 16 will set corresponding flip-flops FY-2 through FY-16. The Y control in Figure 3.2.1.3 is normally at a "1" state. Thus the priority in the order of Y1, Y2, Y3. . .Y16 is established by the Y control. If, for example, at least one of the matching signals in Group 2 and Group 11 are provided, then FY-2 and FY-11 are set, but only Y2 is in a "1" state for processing the matching signals from M_9 through M_{16}. When the processing for Group 2 is over, Y control temporarily becomes a "0" state to reset FY-2 and

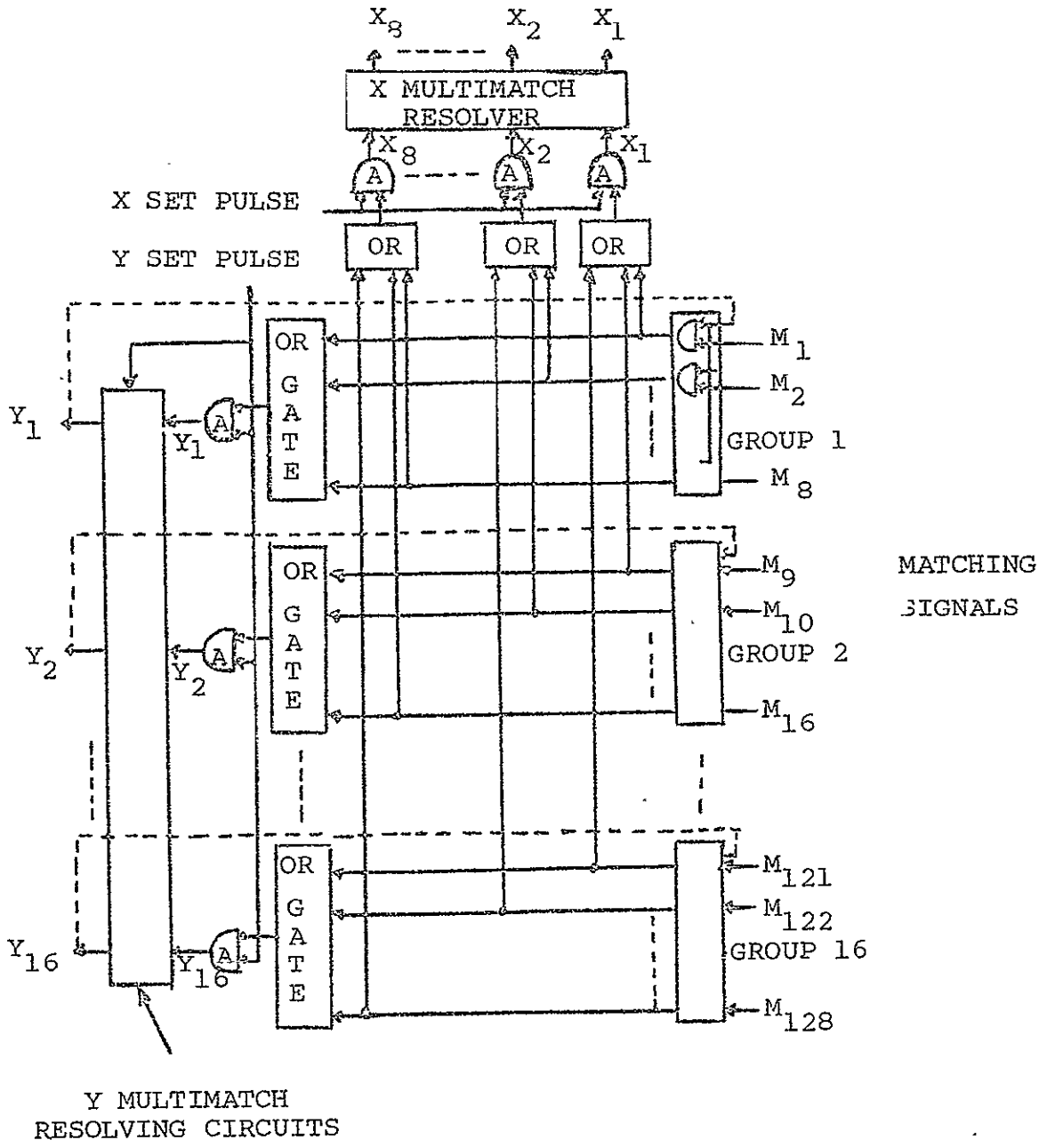


FIGURE 3.2.1.2

DETECTOR MATRIX

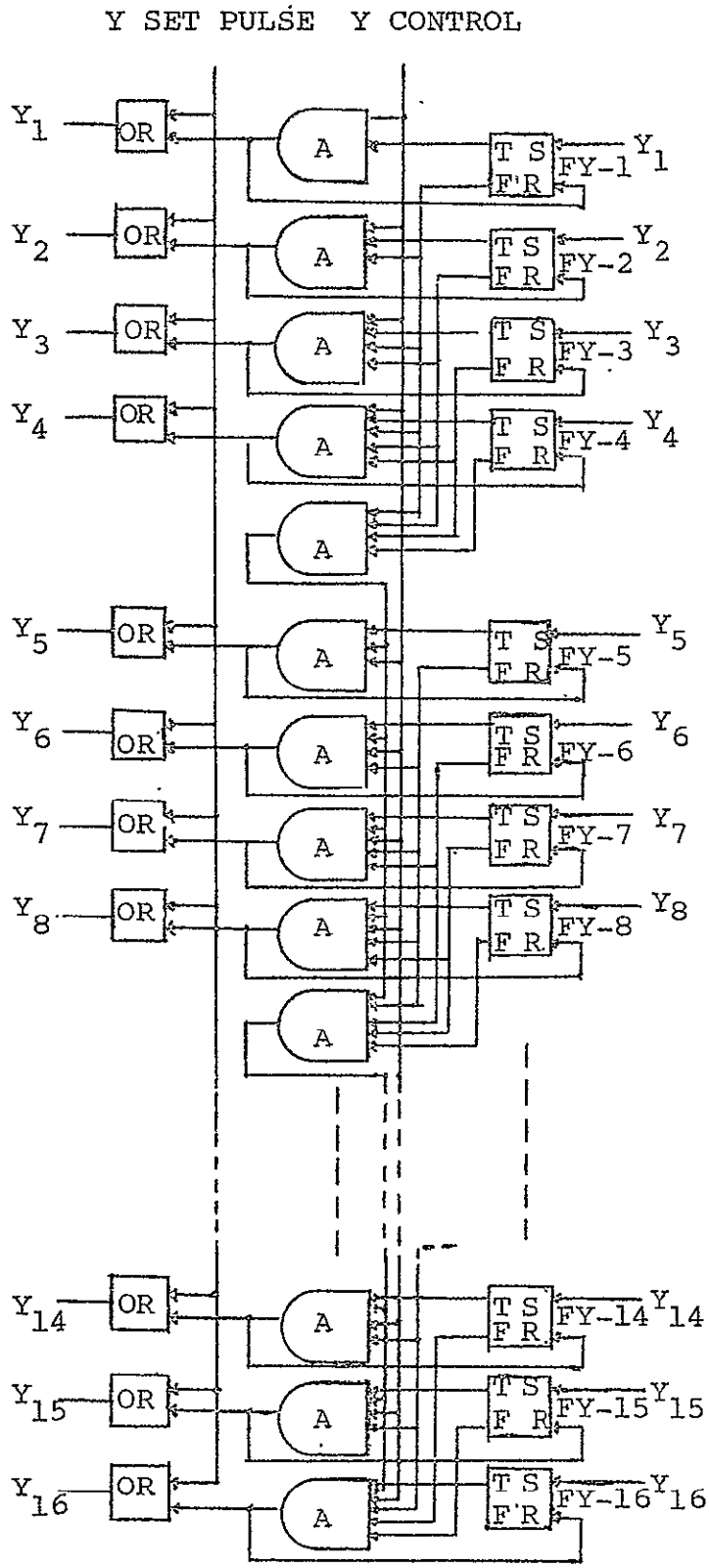


FIGURE 3.2.1.3

Y-MULTIMATCH RESOLVER

the priority control moves down to FY-11. Therefore, for the above example, when Y2 is in "1" state, the states of M_9 through M_{16} are stored in flip-flops FX-1 through FX-8, respectively. The states of eight matching signals in the one group with the highest priority will be stored in FX-1 through FX-8 by enabling X set pulse as shown in Figure 3.2.1.4. As an example, suppose FX-3 and FX-8 are set by matching signals M_{11} through M_{16}. The states of FX-3 and FX-8 show the matching words in Group 2. The word clock in Figure 3.2.1.4 establishes the priority in the order of X1, X2, X3,. . ., X8. The overall timing diagram for the above example is shown in Figure 3.2.1.5. As shown in the diagram, X3 becomes active during the first word clock. Then the first driving signal is generated by Y2 and X3 in the word driving matrix, selecting the first matching word. Following the first word clock, FX-3 is reset and the priority moves down to FX-8, enabling X8. At the second clock time, the word driving signal is generated by Y2 and X3 in the word driving matrix to select a second matching word in Group 2. When the selection of all matching words in Group 2 is finished, Y control resets Y2 and enables the next higher priority Y signal, such as Y11.

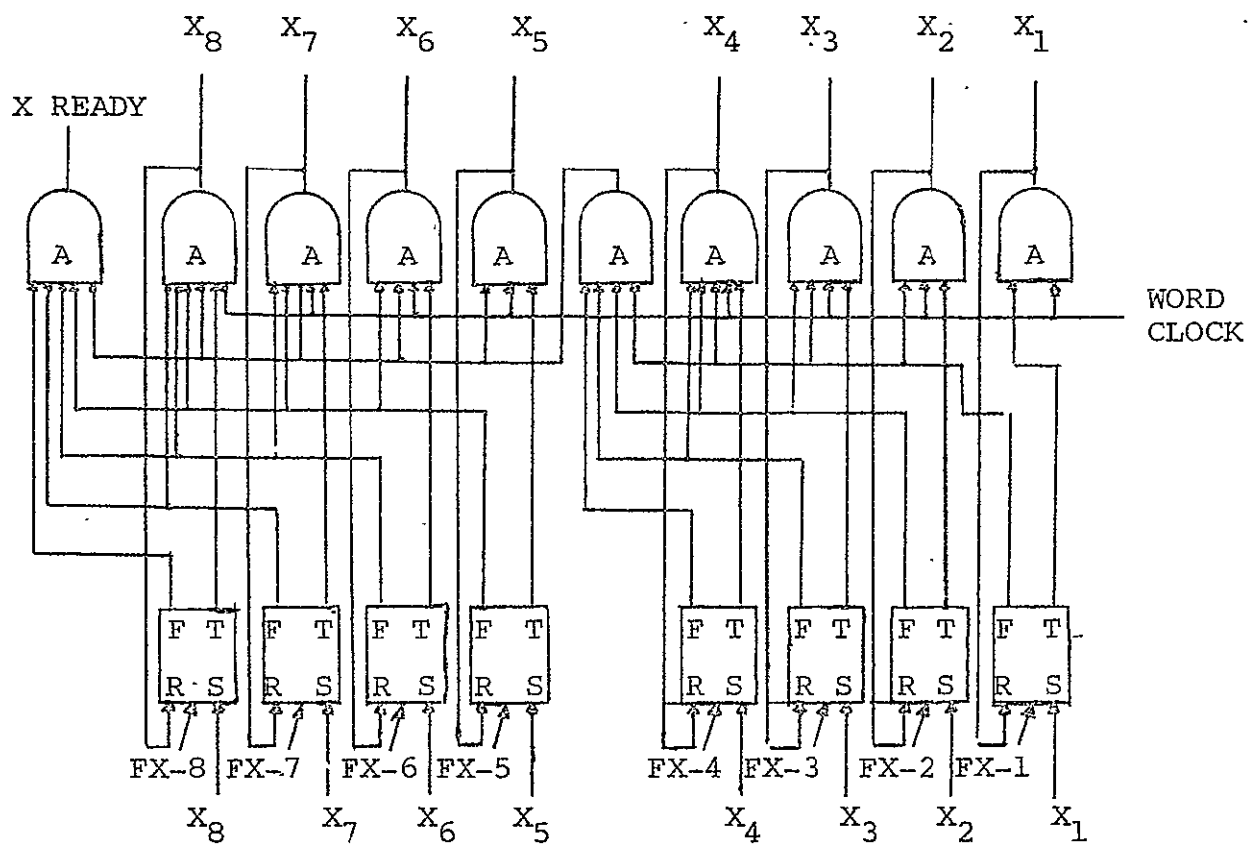


FIGURE 3.2.1.4

-MULTIMATCH RESOLVER

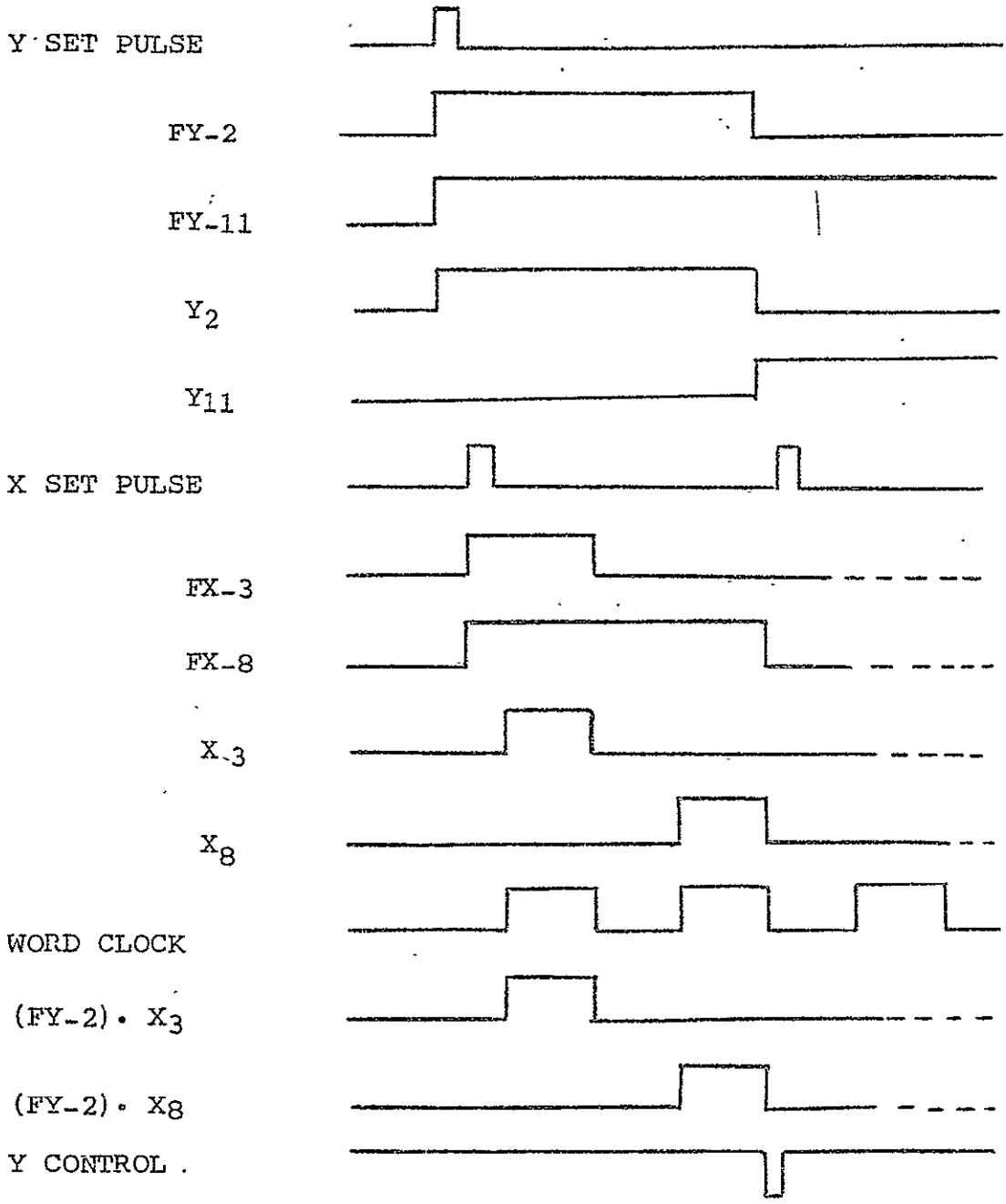


FIGURE 3.2.1.5

TIMING DIAGRAM

The states of matching signals M_{81} through M_{88} are now stored in FX-1 through FX-8. Under the X priority control the selection of matching words of M_{81} through M_{88} is then performed. The selection process continues in the same manner under the X and Y priority controls until X-singlematch and Y-singlematch detectors detect the end of the process.

The logic diagram of a Y-singlematch detector is shown in Figure 3.2.1.6.

3.2.1.2 Verification List Search

This acknowledgement verification method for level two and three command words requires that a search be conducted on the verification list once each millisecond to detect overdue acknowledgements. The search tag is the "$2N-k_i + 1$" portion of the verification word. The possibility exists that an acknowledgement from more than one of the addressed vehicles could be overdue at a given time and therefore meet the search criteria. It is for this reason that the multi-match resolving logic as described in the example is required.

3.2.1.3 Memory Implementation

From a power requirement standpoint an association memory matrix fabricated using MOS technology is highly desirable. A 1024 word

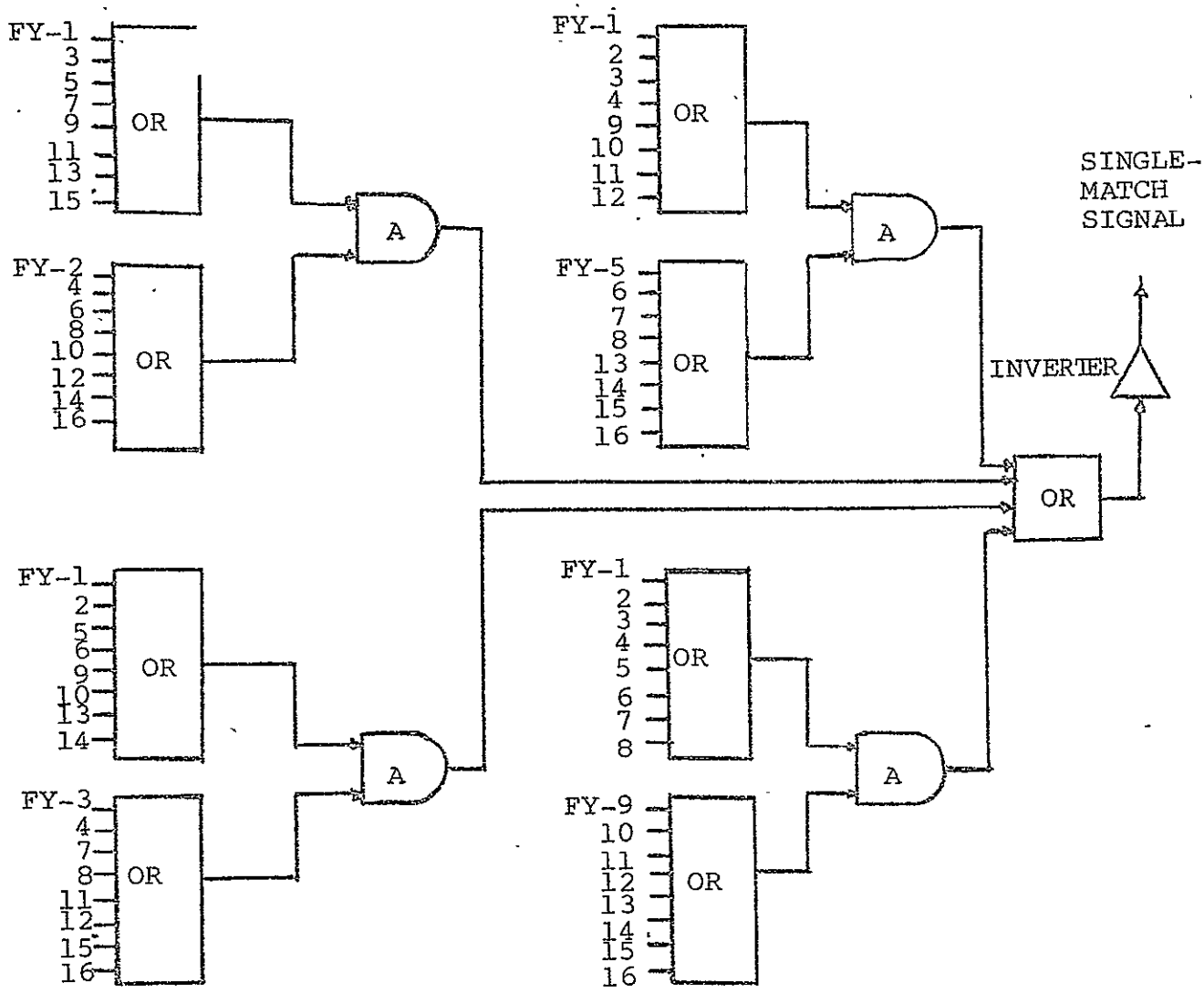


FIGURE 3.2.1.6

Y SINGLE-MATCH DETECTOR

by 38 bit memory matrix such as required for the verification list would require less than one watt of power. However, the type of peripheral circuits utilized will largely determine its total power dissipation.

Other possible methods of implementing the association memory involve the use of bipolar semiconductor logic and plated wire bit steering techniques. Memories using either of these technologies dissipate considerable more power than does the MOS type memory. An advantage of the plated wire system is that it is nonvolatile in case of power loss. This disadvantage, shared by the MOS and bipolar memories, can be overcome by the use of a rechargeable battery, provided the power consumption is not too great.

In the standby mode of operation, the power required by a MOS memory matrix could be held to milliwatts.

3.2.2 Time-Location Verification Method

As the priority level one, two and three command words are transmitted, they are also stored in the verification memory. The location in memory is determined by the count of the real-time counter at that time. If the command word is priority level one, the word is stored in the verification memory only for the first

transmission. (Priority level one commands are transmitted continually until verified.)

Every millisecond, the contents of each telemetry counter is transferred in turn to the verification memory address register. The command word at that location is read into the "B" register where the vehicle address is decoded. If the vehicle address corresponds to the telemetry counter from which the address was obtained, then the last acknowledgement received at that counter should have been a type "1" (valid acknowledgement). A type "0" acknowledgement would indicate that the word was not accepted and must be retransmitted.

When the vehicle address contained in the command word does not correspond to the telemetry counter, the acknowledgement should be a type "0" since the command was not meant for that vehicle. A type "1" acknowledgement would be invalid, in which case the control center is notified via the data bus.

Each time an acknowledgement failure is determined for a receiving vehicle, the acknowledgement failure counter is incremented. The same counter is decremented for each valid acknowledgement /if c (counter) > 0/. Three successive acknowledgement failures will cause the two-stage counter to overflow, initiating an alarm condition to the control center.

3.2.2.1 Verification Memory Timing Considerations

The most active sequence of controller operation from a speed standpoint occurs during the one millisecond verification check for priority one, two and three acknowledgements. During each millisecond period the verification memory must be accessed 15 times (once for each telemetry counter). For the worst case situation where all 15 acknowledgements received are failures, the 15 words read from the verification memory must be stored back in the proper section of buffer memory for retransmission. Reading each word from the verification memory and checking vehicle address, priority versus acknowledgement received takes approximately one memory cycle time. Transferring the command word to the proper section of buffer memory requires approximately two memory cycle times.

Assuming the selection of a relatively slow speed memory system (such as MOS technology), a typical access time is 1 microsecond and cycle time is 3 microseconds. The total number of memory cycle times required in the above example is 48. At 3 microseconds per cycle, the total time required is 144 microseconds. During this millisecond total time period the controller must

also handle at least two input and one output sequence. The remaining time of approximately 850 microseconds is ample time for these sequences.

3.2.3 Time-Tag Method Versus Time-Location Method, Conclusions

The time-location method of implementing the acknowledgement verification scheme has been selected, based on the following advantages:

1. Less logic required for implementation.
2. Random access semiconductor memories are presently available in both MSI and LSI configurations, while associative type memory units are available only in small bit quantities per package.
3. Power requirements are less for the available random access memories.

While the associative organization enables the search of an entire memory in a short time span, it was shown in Section 3.2.2.1 that ample time is available for utilization of the time-location method.

3.3 PRIORITY LEVEL FOUR DATA TRANSFER COMMANDS

Level four data commands are transmitted in blocks which consist of a header word, a starting address word, data words and an end of message word. The header word contains the vehicle address and the subsystem address. All words following

the starting address are treated as data by the receiving vehicle until the end of message is received.

Level four data commands are retained in buffer memory following transmission until they are verified. The first valid type "1" acknowledgement from the addressed vehicle occurs when the contents of the telemetry acknowledgement counter $k_i = N$, the real-time clock count at the time the first word was transmitted. If the acknowledgement for the header word or starting address is a type "0", or non-verify an interrupt is generated which restarts the transmission. Each data word location is sequentially addressed as the acknowledgement counter increments, and a tag bit written into depending upon the type of acknowledgement. A type "1", or verify, acknowledgement will cause a logic "1" to be written into the tag bit at that location. A non-verify will cause a logic "0" to be written into the tag bit. When the final word of the data block has been acknowledged, a search for unverified words is initiated, starting with the first data word. A word containing the starting address is formulated for each non-verified data word or word sequence located. This word and the header must be transmitted along with the data word(s), all followed by an end of message word.

Verification of the retransmission is handled in a manner similar to that for a normal transmission. This process is repeated for any additional non-verified words detected during the search. Should an end of transmission word not be verified, it is retransmitted as a priority one

command since the receiving vehicle would treat any other command as data. When a command word of higher level priority interrupts a level four data transfer, an end of message word is transmitted following completion of the word in process. An interrupted search routine for non-verified words in the level four data word buffer will restart at the beginning of the data block when control is returned.

3.4 CONTROLLER DETAILED SYSTEM DESIGN (reference Figure 3.4)

3.4.1 Buffer Memory

The number of locations in buffer memory for the four types of priority level command words have arbitrarily been assigned as follows:

Priority Level One Commands	-	4 locations
Priority Level Two Commands	-	64 locations
Priority Level Three Commands	-	64 locations
Priority Level Four Commands	-	1024 locations

Level four commands are allotted the greater portion of the memory for two reasons. First, according to present ground rules, level four commands cannot be transmitted until all higher priority commands have been transmitted and verified. Secondly, level four data words are expected to form the major part of all transmissions.

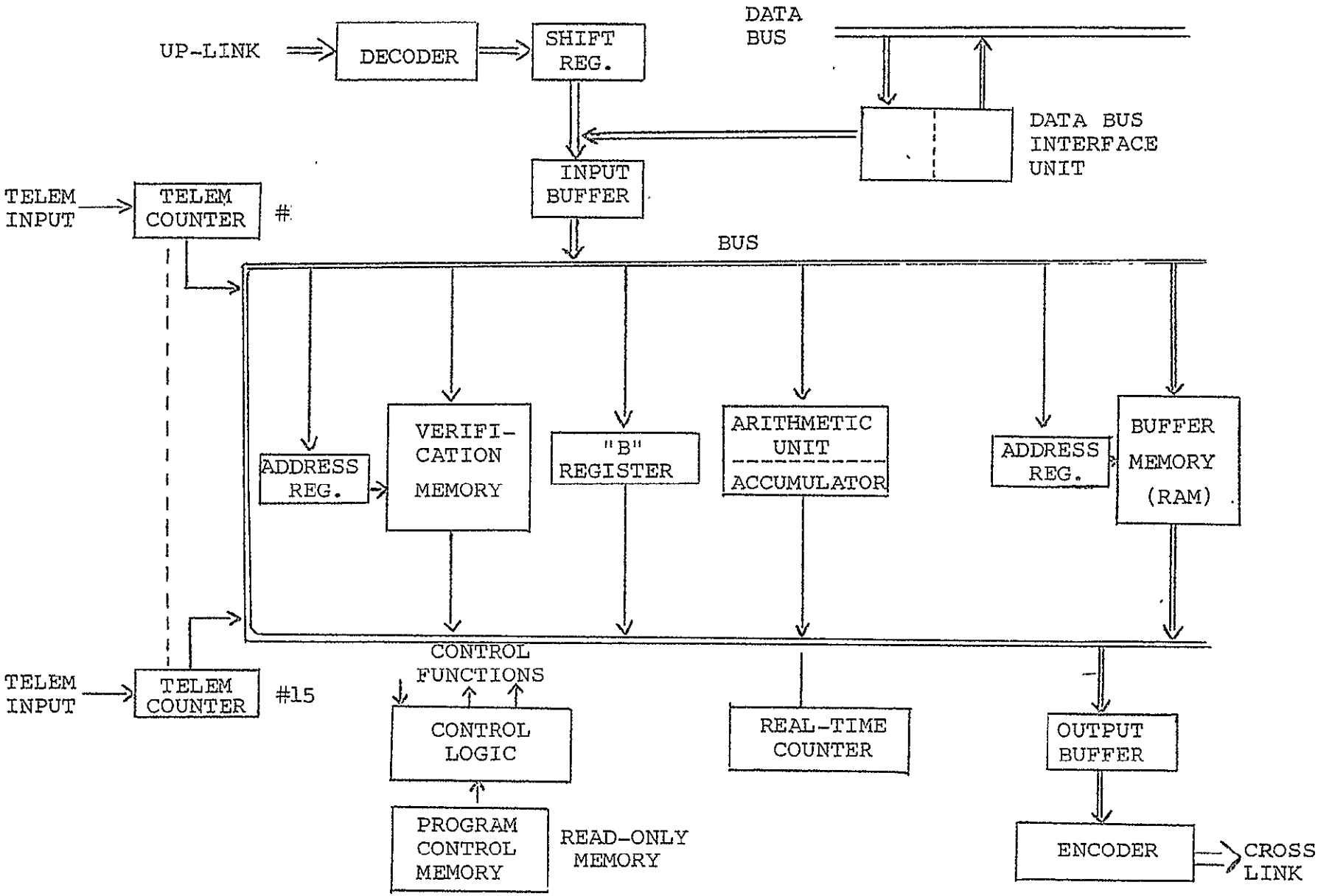


FIGURE 3.4

COMMAND CONTROLLER BLOCK DIAGRAM

Storage is required for levels one, two and three commands in the event the command processor is waiting for a verification acknowledgement for a priority one command which is being continuously transmitted. The time interval before verification could be as long as 1/2 second if the receiving vehicle were the maximum distance from the space base. Thus, as many as 512 words could be received during this time. Few of these words are expected to be additional priority level one commands. Should the capacity of the storare buffer be exceeded, the additional words are transmitted via data bus to temporary storage.

3.4.2 Input Pointers, Output Pointers and Word Counters

Each section of the buffer memory assigned to the four priority levels of command words has a corresponding input pointer, output pointer and word counter. These pointers and counters consist of unique locations in the buffer memory. The counters always indicate the number of words stored for each priority level. Counters are incremented each time a word is stored and decremented when a word is retrieved for transmission.

The input pointers indicate the location for storing the next input word and are incremented following each storage operation. In a similar manner, the output pointers indicate the location from which the next word is to be retrieved for transmission. Since the buffer

storage functions on a first in-first out basis, the word counter must be checked before a word is stored to determine if that buffer section is full. In this case the input pointer contains the same address as does the output pointer, and if a word were to be stored, the word at that location would be lost.

3.4.3 Telemetry Counters (Figure 3.4.3)

The command controller utilizes a counter (modulo 512) for each of the 15 possible receiving vehicles. An acknowledgement, whether a "1" or "0", received via the telemetry link increments the counter one count.

3.4.4 Real-Time Counter (Figure 3.4.4)

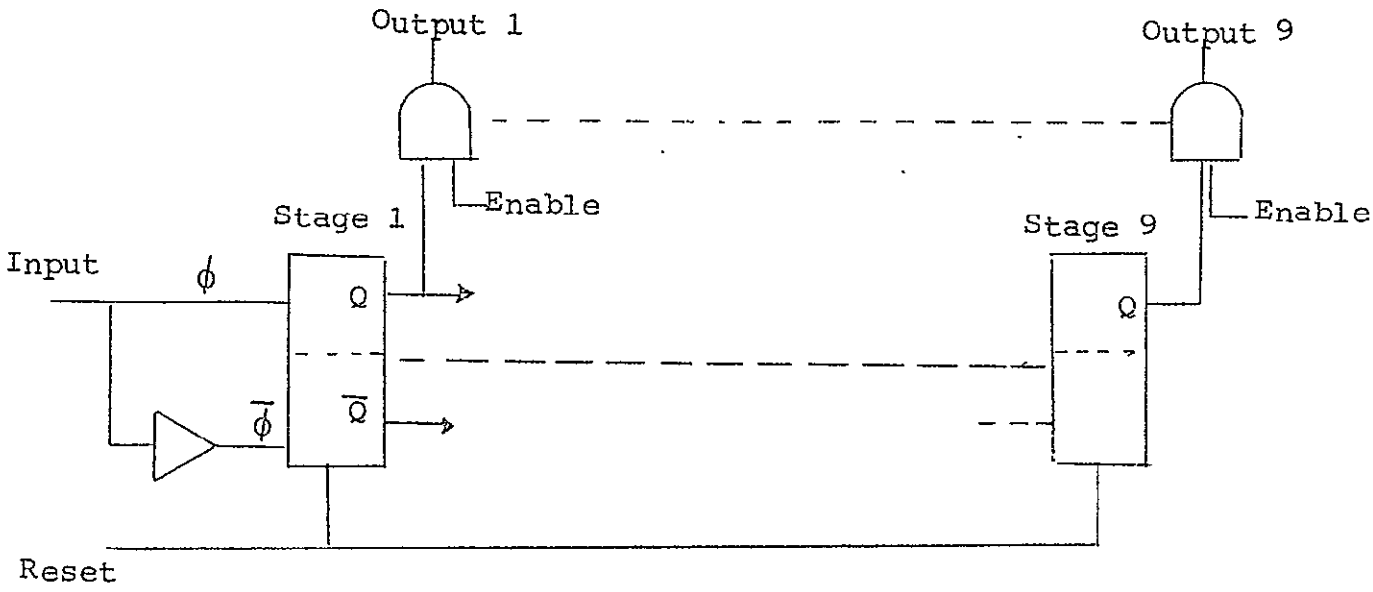
The command controller also contains a real-time counter (modulo 512) operating at the maximum transmittal rate of 1000 commands per second. The modulo 512 count is based on the maximum time from transmission to acknowledgement of 500 milliseconds.

3.4.5 Verification Failure Counters

The command controller contains a two stage verification failure counter for each satellite vehicle. A successful verification decrements the counter, while a verification failure increments the counter. Three successive failures fill the counter, initiating an alarm condition which signals the control center via data bus.

FIGURE 3.4.3

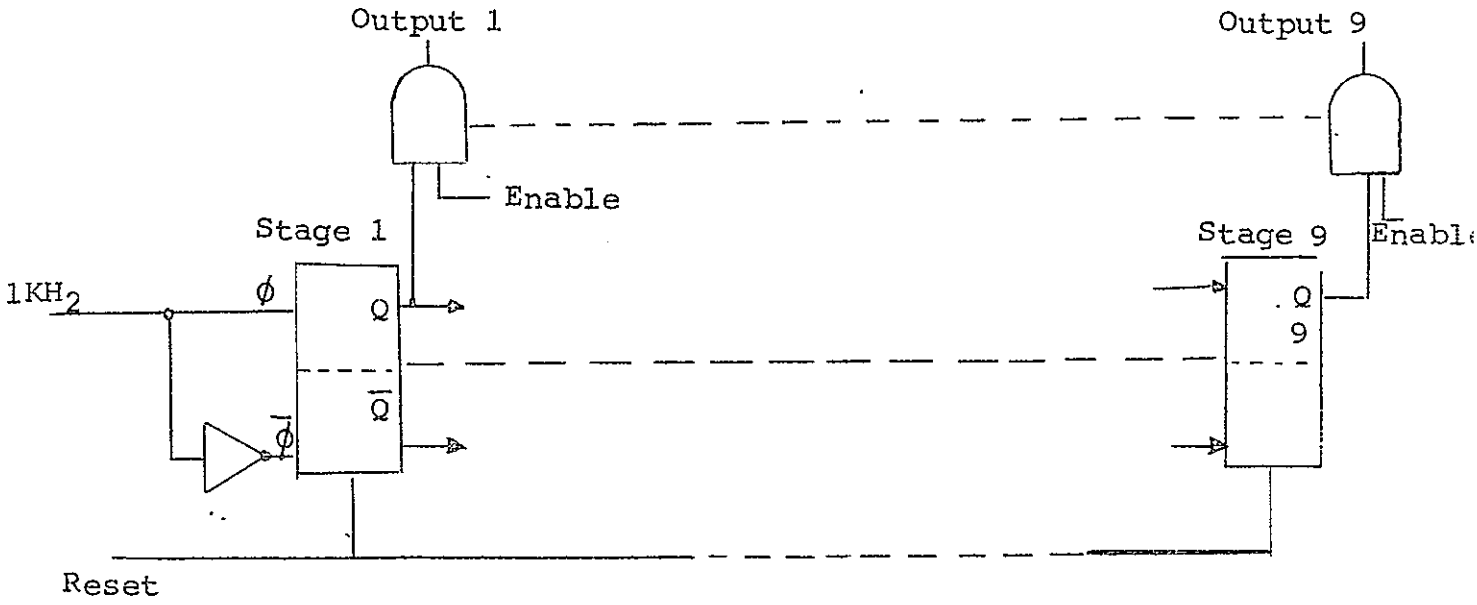
MODULO 512 TELEMETRY COUNTER



15 COUNTERS REQUIRED

FIGURE 3.4.4

MODULO 512 REAL-TIME COUNTER
(9 Stage)



3.4.6 Verification Memory

A 512 word random access memory serves as a verification list for priority level one, two and three command words. At the time one of these command words is transmitted, the contents of the real-time counter is transferred to the address register for the verification memory. The command word being transmitted is then stored at that location.

3.4.7 Arithmetic Unit

A twelve bit arithmetic unit in the command controller performs increment and decrement operations on memory addresses, as well as equality and relative magnitude operations which are mentioned in the detailed description of operation.

A block diagram of the proposed ARU is shown in Figure 3.4.7. The arithmetic operations are performed on the contents of the accumulator and/or the contents of the bus according to the function code applied to the adder. The function code is decoded within the adder logic to enable the desired operation. The results of the arithmetic or logic operation are stored in the accumulator from where it is gated onto the bus when required.

3.4.8 Adder Implementation

The equivalent logic diagram for one bit of the adder is shown in Figure 3.4.8. The specific adder

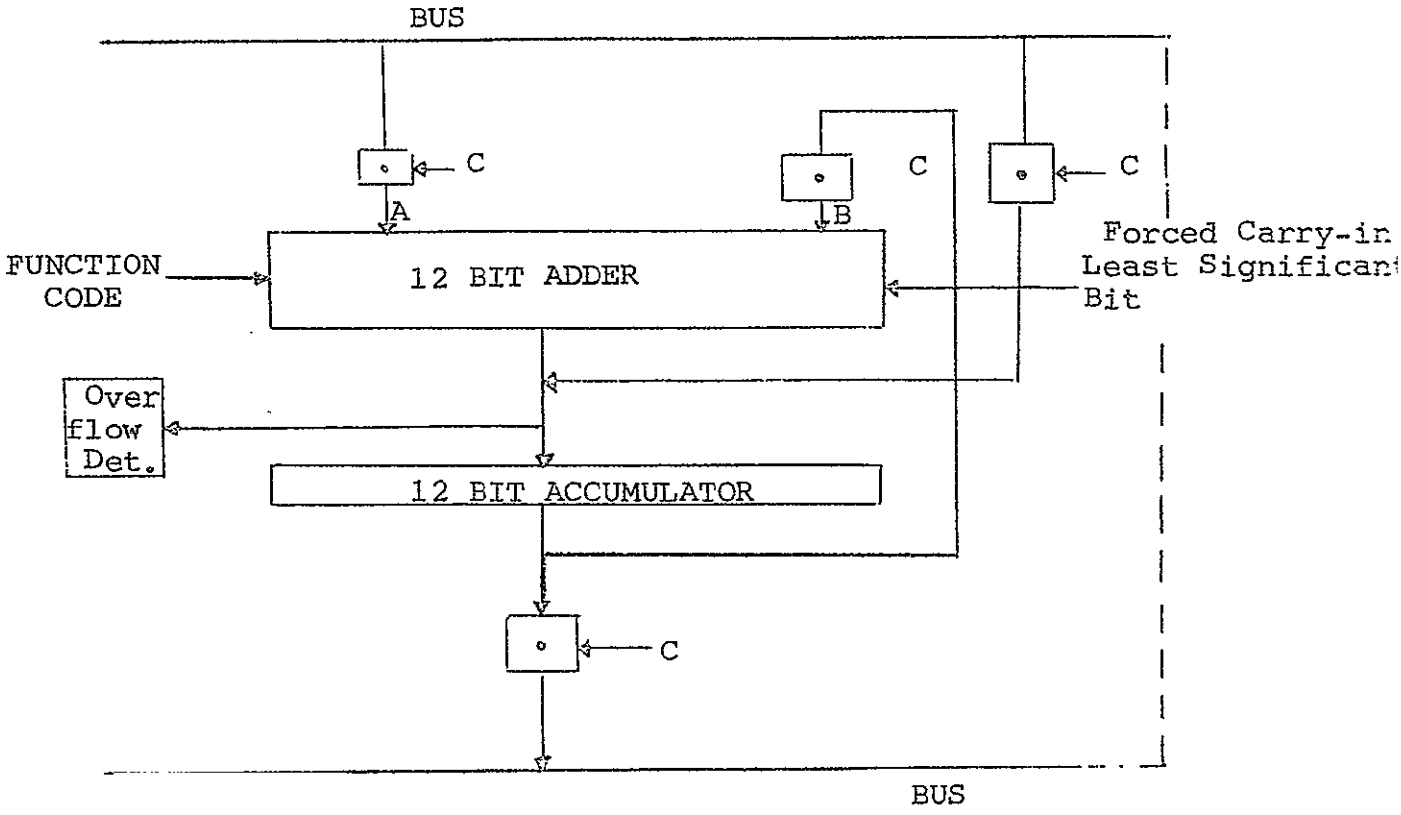
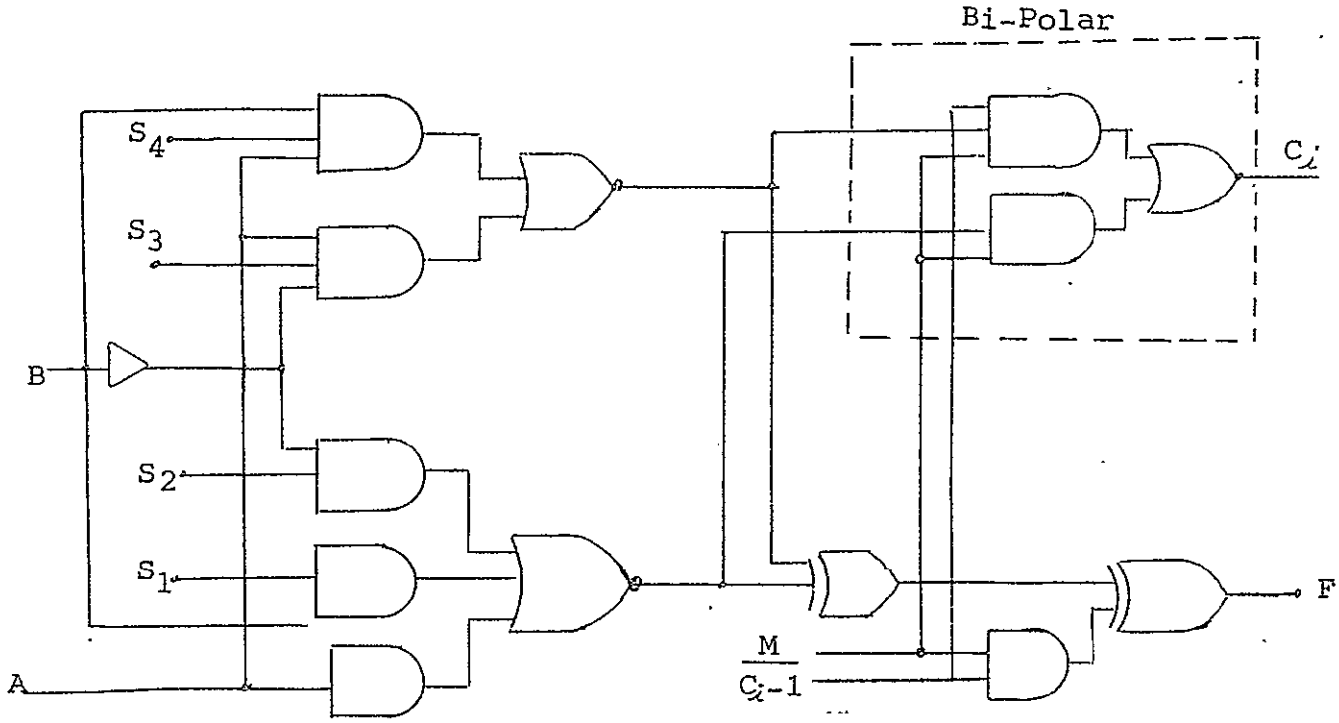


FIGURE 3.4.7

ARITHMETIC UNIT (ARU)

FIGURE 3.4.8

LOGIC FOR ONE BIT OF ADDER



ADDER FUNCTIONS

OUTPUT FUNCTION	FUNCTION SELECT				
	S_1	S_2	S_3	S_4	M
A plus B	1	0	0	1	1
A minus B-1	0	1	1	0	1
A shifted 1 Bit	0	0	1	1	1
A + B	0	1	1	1	0
AB	1	1	0	1	0
$\overline{A \oplus B}$	0	1	1	0	0
$A \oplus B$	1	0	0	1	0

function is selected by inputs S1-S4 in conjunction with mode control input, M. The adder operates in the arithmetic mode when M is high and the logic mode, where the carry-in has no effect, when M is low.

Substraction is performed by one's complement addition with the one's complement of the subtrahend generated internally. The output is A-B-1 which requires a forced carry into the last significant bit to provide A-B.

The proposed design for the adder unit employs MOS type gates for the major part of the logic. Selection of MOS logic involves the trade-off of power and speed. The low power consumption of the MOS logic favors its use wherever the propagation delays can be tolerated.

3.4.9 Accumulator

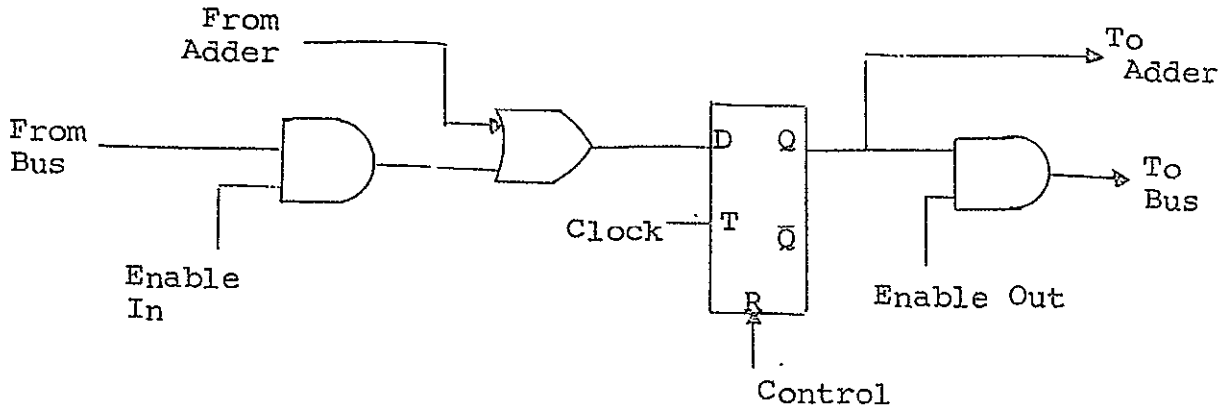
The accumulator register (reference Figure 3.4.9) is constructed with D-type flip-flops of MOS or low-power TTL technolgoy. Gated clocks permit selected data transfer.

3.4.10 ARU Speed

A true measurement of ARU speed is the maximum clock rate at which the ARU may be operated. This time includes decoding of bus enable signals, adder transfer time, carry ripple time (arithmetic operations), output bus, and register delays.

FIGURE 3.4.9

ACCUMULATOR STAGE



Arithmetic Operations

Minimum clock period for sum of n bits:

T = Bus Enable + Ripple Generate + Ripple Propagate + Adder + Register Setup.

Using 70 ns/stage MOS

10 ns/stage Bipolar

T = (150 + 140 +(20n) + 280 + 140)ns

T = 710 + (20n)ns. (For n = 12 bits, T = 950ns)

Logical Operations

Minimum clock period, independent of the number of bits:

T = Bus Enable + Adder + Register Setup

T = (150 + 280 + 140)ns

T = 570ns

3.4.11 Counter and Register Implementation

Figures 3.4.11.1 through 3.4.11.3 are equivalent logic diagrams of the proposed counters and registers in the system. Specific device selection will depend largely on functional speed requirements of each unit.

3.4.12 Control Section

Figure 3.4.12 is a block diagram of the control section required to implement the command controller. The operations performed by the control unit are determined by the contents of the microprogram control memory. The use of the microprogram control memory differs here from the common application where program

FIGURE 3.4.11.1

18 BIT BUFFER REGISTER

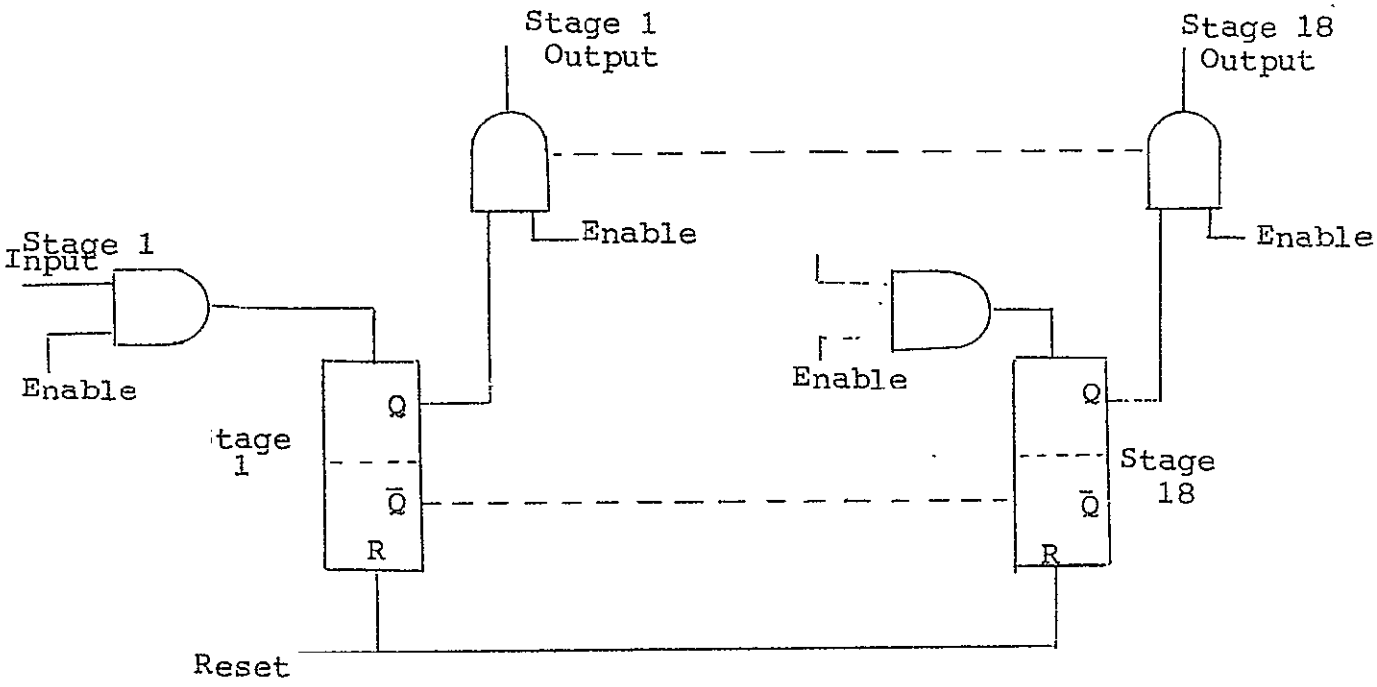


FIGURE 3.4.11.2

18 STAGE SHIFT REGISTER

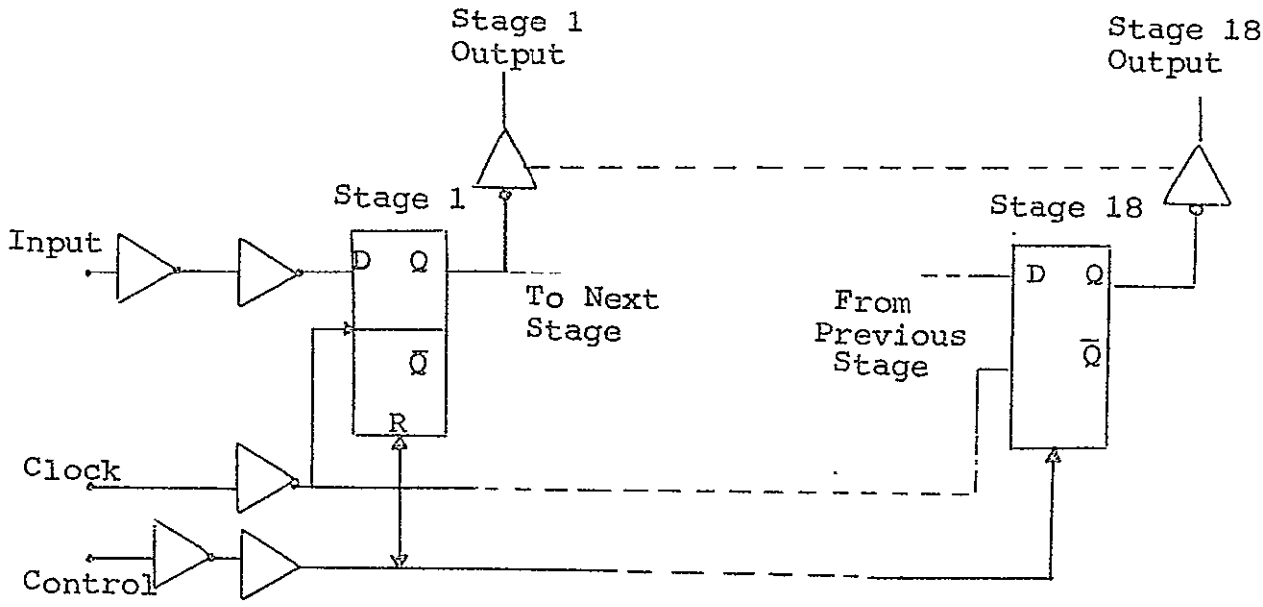
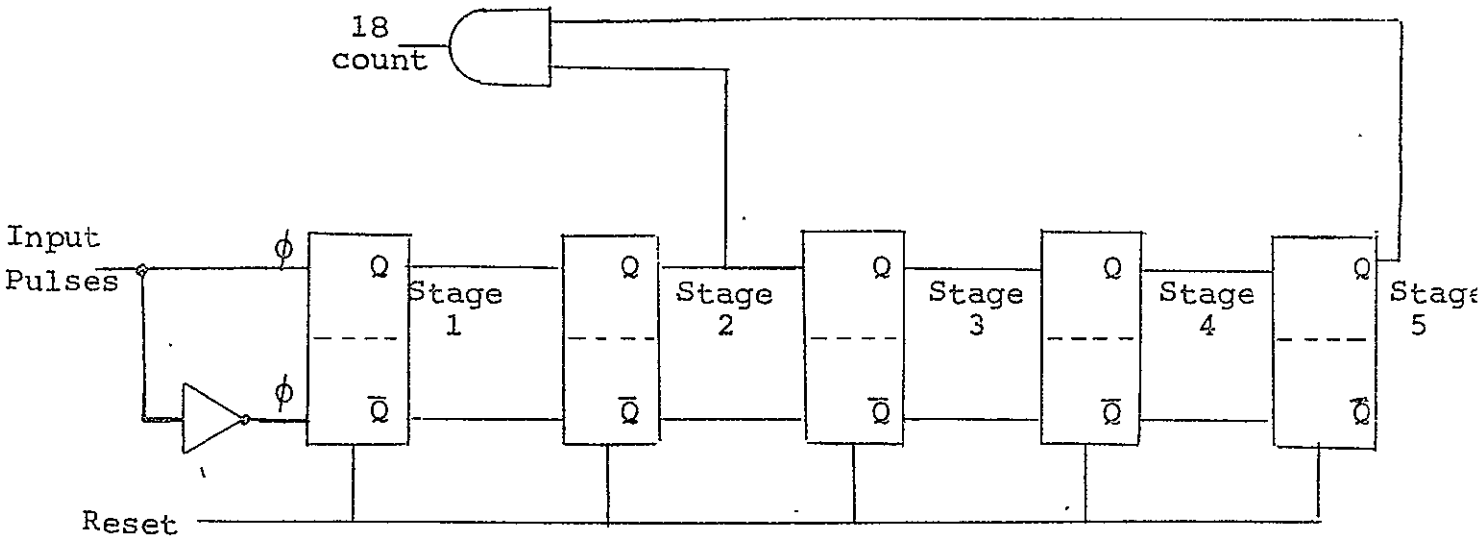
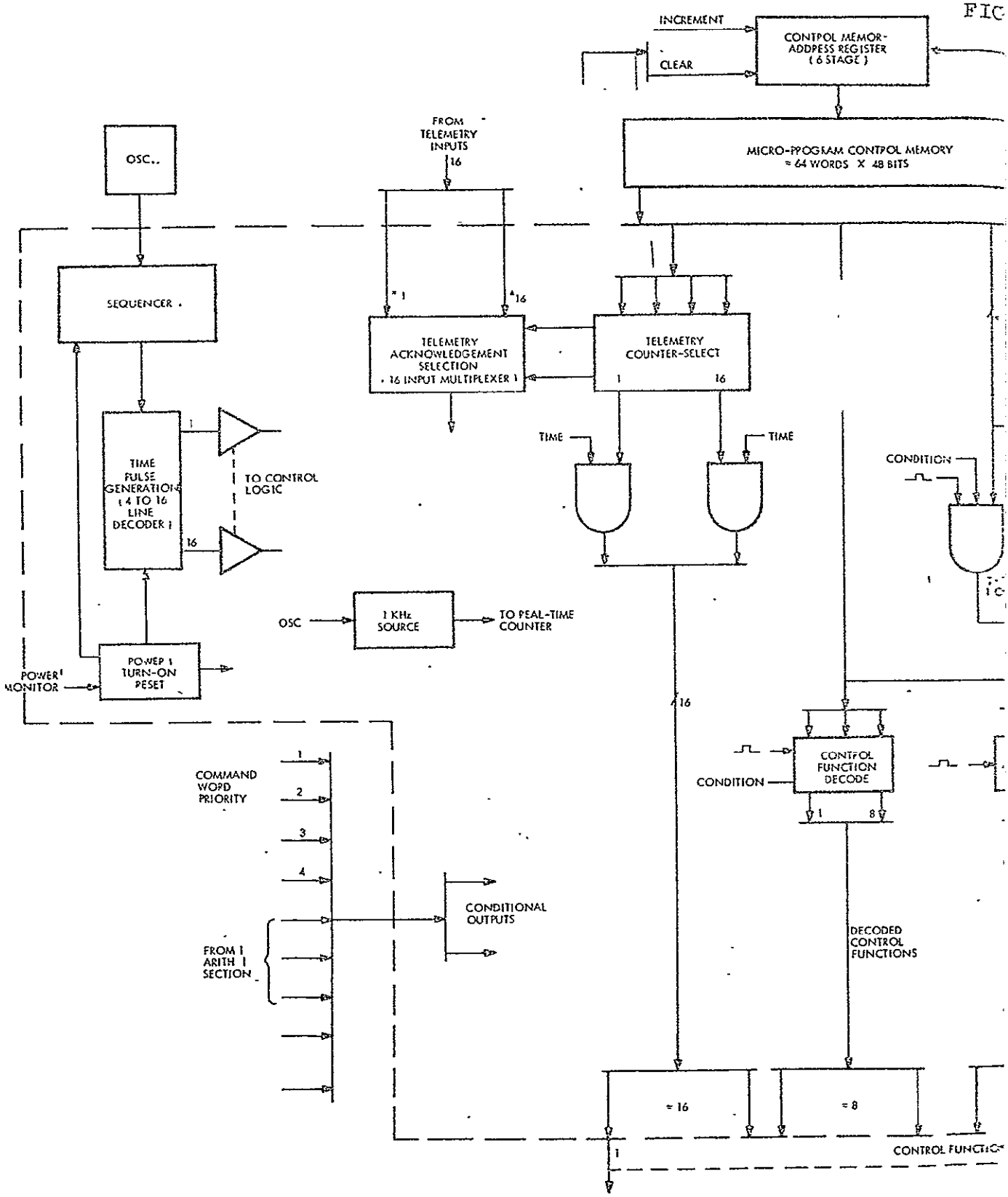


FIGURE 3.4.11.3

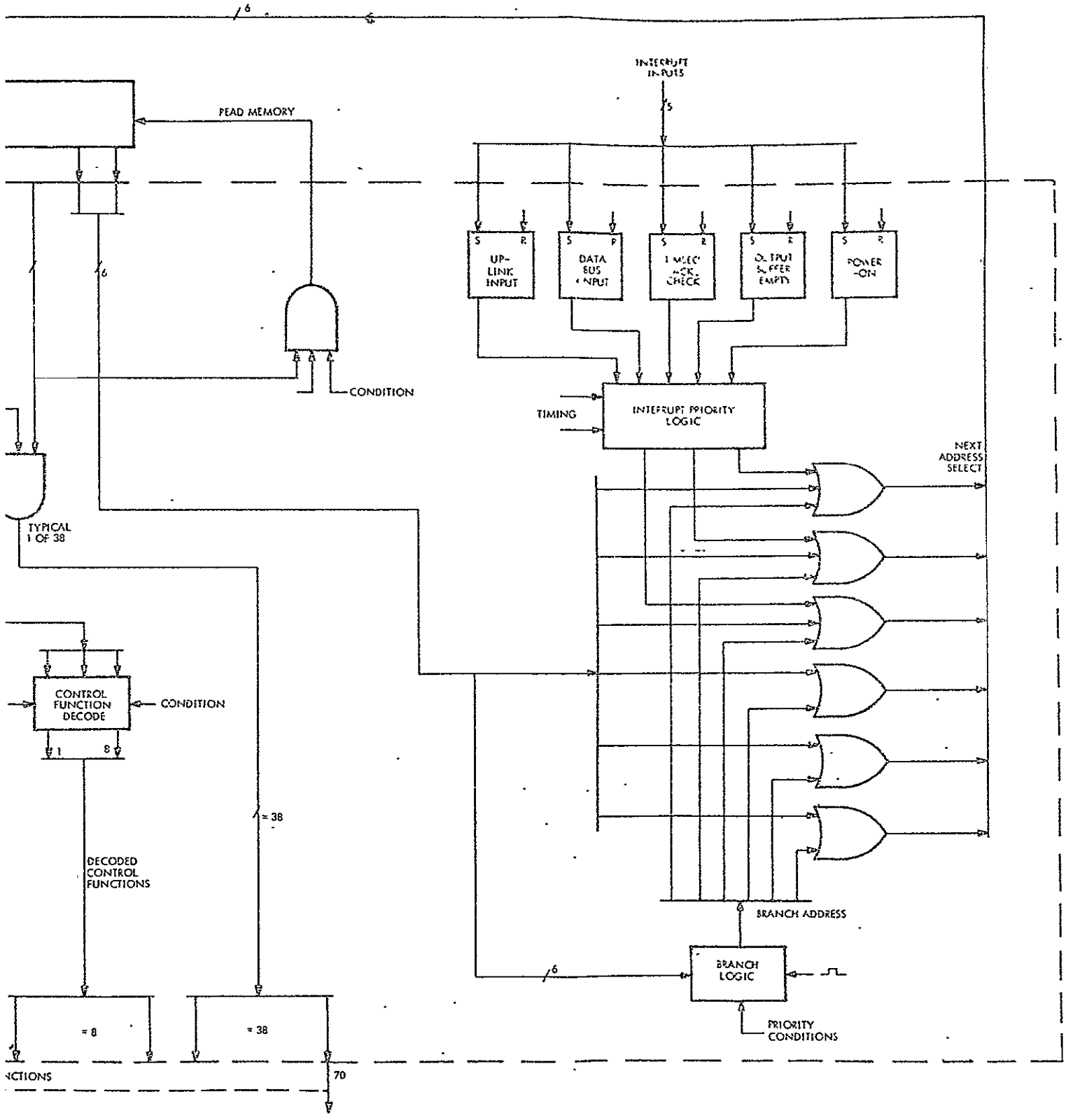
5 STAGE COUNTER (MODULO 18)





FOLDOUT FRAME

FIGURE 3.4.12 CONTROL SECTION



FOLDOUT FRAME 2

information is stored separately at the macro level. The macro-instructions would then specify certain locations in the microprogram memory from which the control functions would be generated.

In this application, the entire program required for the command controller is coded at the micro level. Both program changes and control function changes are made by altering the microprogram memory.

Control section operation for an input sequence would be as follows:

> The shift register full signal sets an interrupt flip-flop which, at the proper time, selects a specific location in the program control memory. The control functions, such as transfer contents of shift register to buffer register, are determined by the information bits contained in the addressed memory location. The timing for the control function is provided by ANDing a timing generator output with the memory output.

A portion of the memory output (from 1 to 6 bits) specifies the next memory location where additional micro-control functions are stored. This location is addressed when the present cycle is completed. The next location could be that immediately following the

present location, in which case the address register is incremented. A conditional input signal such as a buffer memory full, could generate an address for the next control cycle far removed from the original location.

Approximately 70 control functions are needed to implement the various controller sequences. Table 3.4.12 lists most of the control functions required. A program memory of 64 words x 48 bits per word is considered to be the approximate size required. Approximately 38 of the 48 memory outputs are direct control functions, while the remaining ten bits are decoded to provide an additional 32 control functions.

Time pulses are generated by an oscillator, sequencer and a 4-to-16 line decoder. The specific width of control pulses will depend largely upon the final logic selected for implementation of the command controller.

3.5 BASIC OPERATION OF CONTROLLER SEQUENCES

The basic operation of the input sequence, output sequence and acknowledgement verification sequences is described next with the aid of flow charts, Figure 3.5.1.

3.5.1 Input Sequence (Up-Link Data) (Figure 3.5.1)

Data which is accepted by the decoder is transferred to the input buffer register and a valid acknowledgement transmitted via down-link telemetry.

TABLE 3.4.12

CONTROL FUNCTIONS (S. B. COMMAND SYSTEM)

1	C(Bus) ⟶ Buffer Memory
2	C(Bus) ⟶ Verification Memory
3	C(Bus) ⟶ Address Register, (Buffer Memory)
4	C(Bus) ⟶ Accumulator
5	C(Bus) ⟶ B Register
6	C(Bus) ⟶ Address Register, (Verify Memory)
7	C(Shift Register) ⟶ Input Buffer
8	C(Input Buffer) ⟶ Bus
9	C(Data Bus Register) ⟶ Input Buffer
10	C(Bus) ⟶ Data Bus Input Register
11	C(Tel. CTR. #1) ⟶ Bus
12	C(Tel. CTR. #2) ⟶ Bus
13	C(Tel. CTR. #3) ⟶ Bus
14	C(Tel. CTR. #4) ⟶ Bus
15	C(Tel. CTR. #5) ⟶ Bus
16	C(Tel. CTR. #6) ⟶ Bus
17	C(Tel. CTR. #7) ⟶ Bus
18	C(Tel. CTR. #8) ⟶ Bus
19	C(Tel. CTR. #9) ⟶ Bus
20	C(Tel. CTR. #10) ⟶ Bus
21	C(Tel. CTR. #11) ⟶ Bus
22	C(Tel. CTR. #12) ⟶ Bus
23	C(Tel. CTR. #13) ⟶ Bus
24	C(Tel. CTR. #15) ⟶ Bus
25	C(Tel. CTR. #16) ⟶ Bus

TABLE 3.4.12
(Cont)

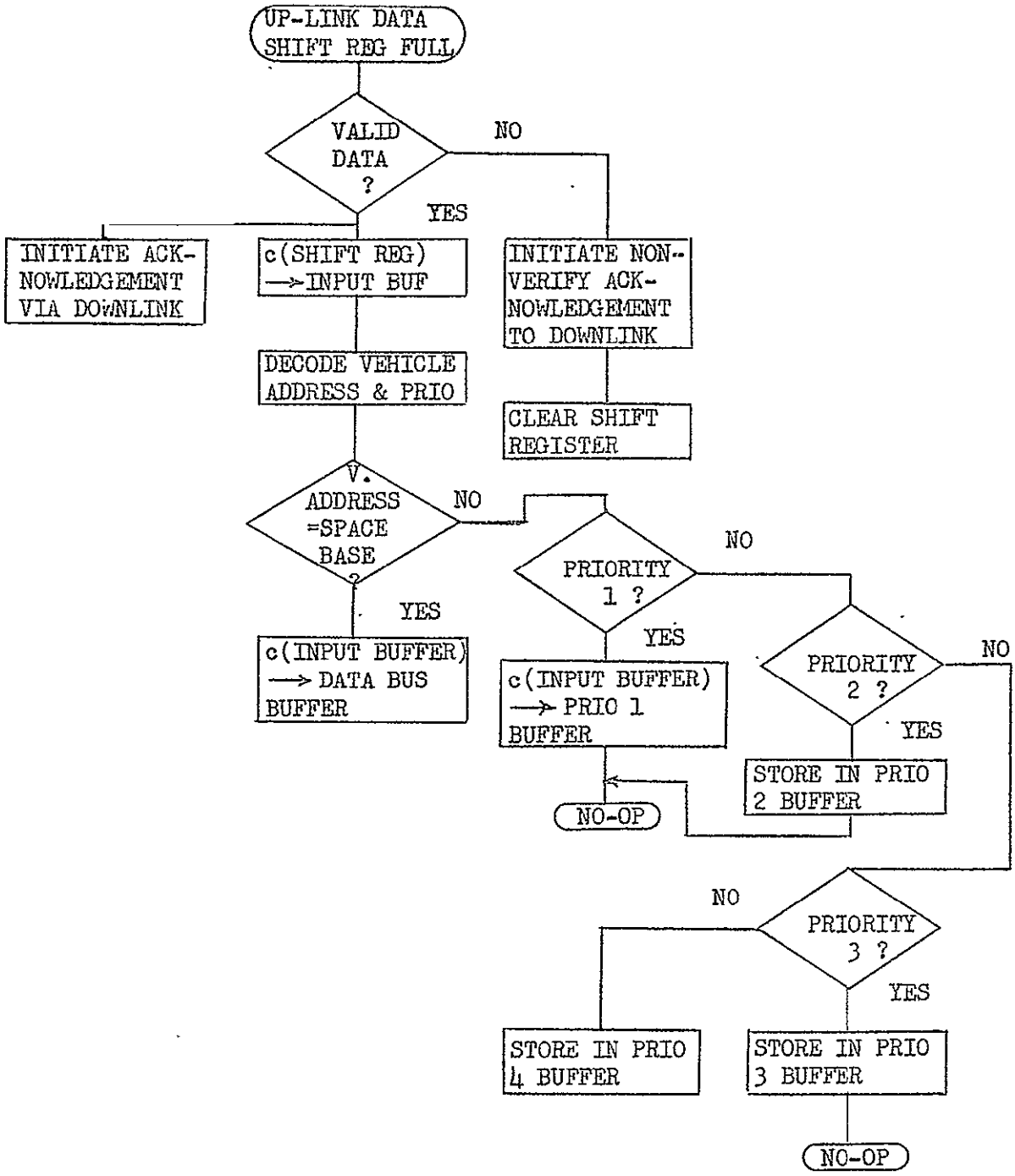
26	C(B Register) ⟶ Bus
27	C(Accumulator) ⟶ Bus
28	Arith. Function ⟶ Increment
29	Arith. Function ⟶ Decrement
30	Arith. Function ⟶ Comparison
31	Arith. Function ⟶ Relative Magnitude
32	C(Real-Time Counter) ⟶ Bus
33	C(Bus) ⟶ Output Buffer
34	C(Output Buffer) ⟶ Encoder
35	Jam Address - Prio. 1 Word Counter
36	Jam Address - Prio. 1 Input Pointer
37	Jam Address - Prio. 1 Output Pointer
38	Jam Address - Prio. 2 Word Counter
39	Jam Address - Prio. 2 Input Pointer
40	Jam Address - Prio. 2 Output Pointer
41	Jam Address - Prio. 3 Word Counter
42	Jam Address - Prio. 3 Input Pointer
43	Jam Address - Prio. 3 Output Pointer
44	Jam Address - Prio. 3 Verification Pointer
45	Jam Address - Prio. 3 Acknowledgement Pointer
46	Jam Address - Prio. 4 Word Counter
47	Jam Address - Prio. 4 Input Pointer
48	Jam Address - Prio. 4 Output Pointer
49	Jam Address - Prio. 4 Verification Pointer
50	Jam Address - Prio. 4 Acknowledgement Pointer
51	Jam Address - - - - - Temporary Storage
52	Write Buffer Memory
53	Read Buffer Memory

TABLE 3.4.12

(Cont)

54	Write Verification Memory
56	Read Verification Memory
57	Set B Register to 4
58	Set B Register to 64
59	Set B Register to 1000
60	Set B Register to 3
61	Clear Accumulator
2	Clear B Register
3	Clear Address Register (Buffer Memory)
4	Clear Input Buffer
5	Clear Output Buffer
6	Clear Address Register (Verification Memory)
7	Read Micro-Program Memory
8	Increment Micro-Program Counter
9	Jam Flag Word to Data Bus

FIGURE 3.5.1 INPUT SEQUENCE



If the data word is not accepted by the decoder, a non-valid acknowledgement is initiated and the shift register cleared.

The vehicle address and priority of the command are obtained by decoding the proper bits of the data in the input buffer. Data for the space base is transferred to the data bus buffer register.

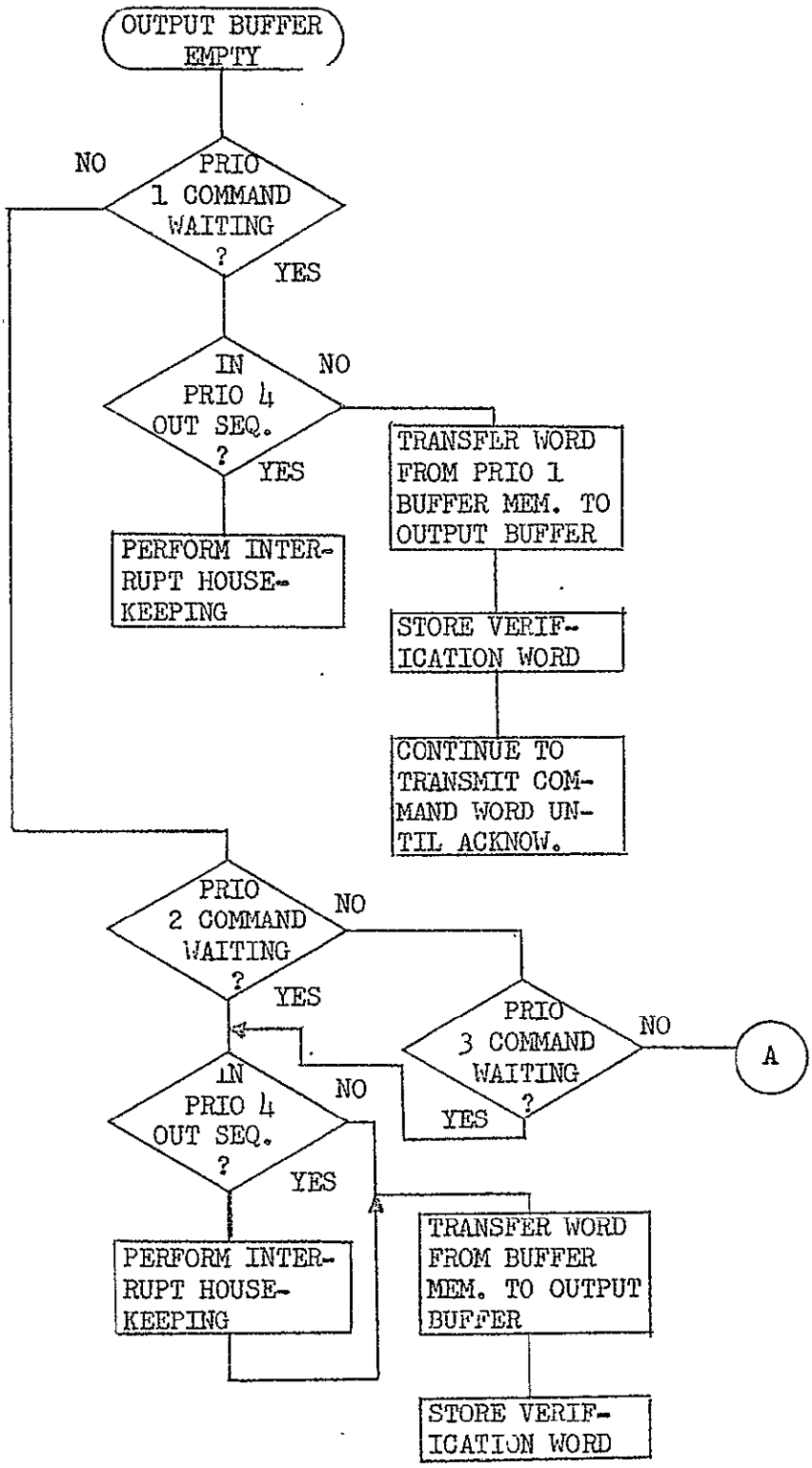
Data for external vehicles is transferred to a section of the buffer memory according to priority. If a section of buffer memory is full, any additional words of that priority are transferred to external storage via the data bus.

3.5.2 Output Sequence (Figures 3.5.2 and 3.5.3)

An output buffer empty signal initiates a search of the buffer memory for data on the basis of priority level. Data in the buffer is indicated by the contents of the corresponding word counter being greater than zero. As words are stored in the buffer, the word counter is incremented and when words are removed, the counter is decremented.

Priority level four words cannot be transmitted until all previously transmitted words have been acknowledged and verified. The priority level one, two and three words are also transferred to verification memory each time they are transferred from the output buffer to the encoder. The location in verification memory is determined by the real-time count at that time.

FIGURE 3.5.2 OUTPUT SEQUENCE



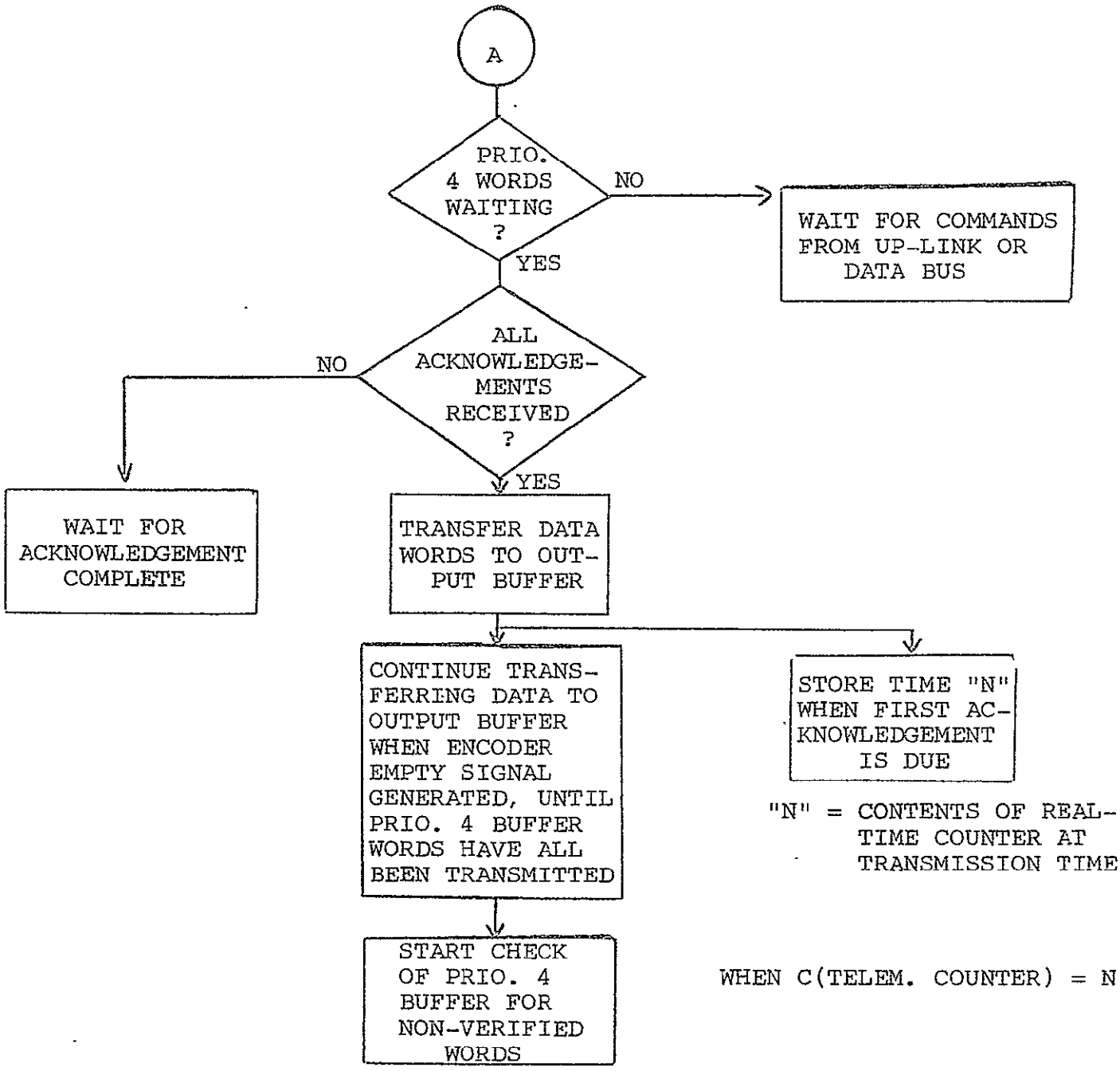


FIGURE 3.5.3

OUTPUT SEQUENCE

(CONT)

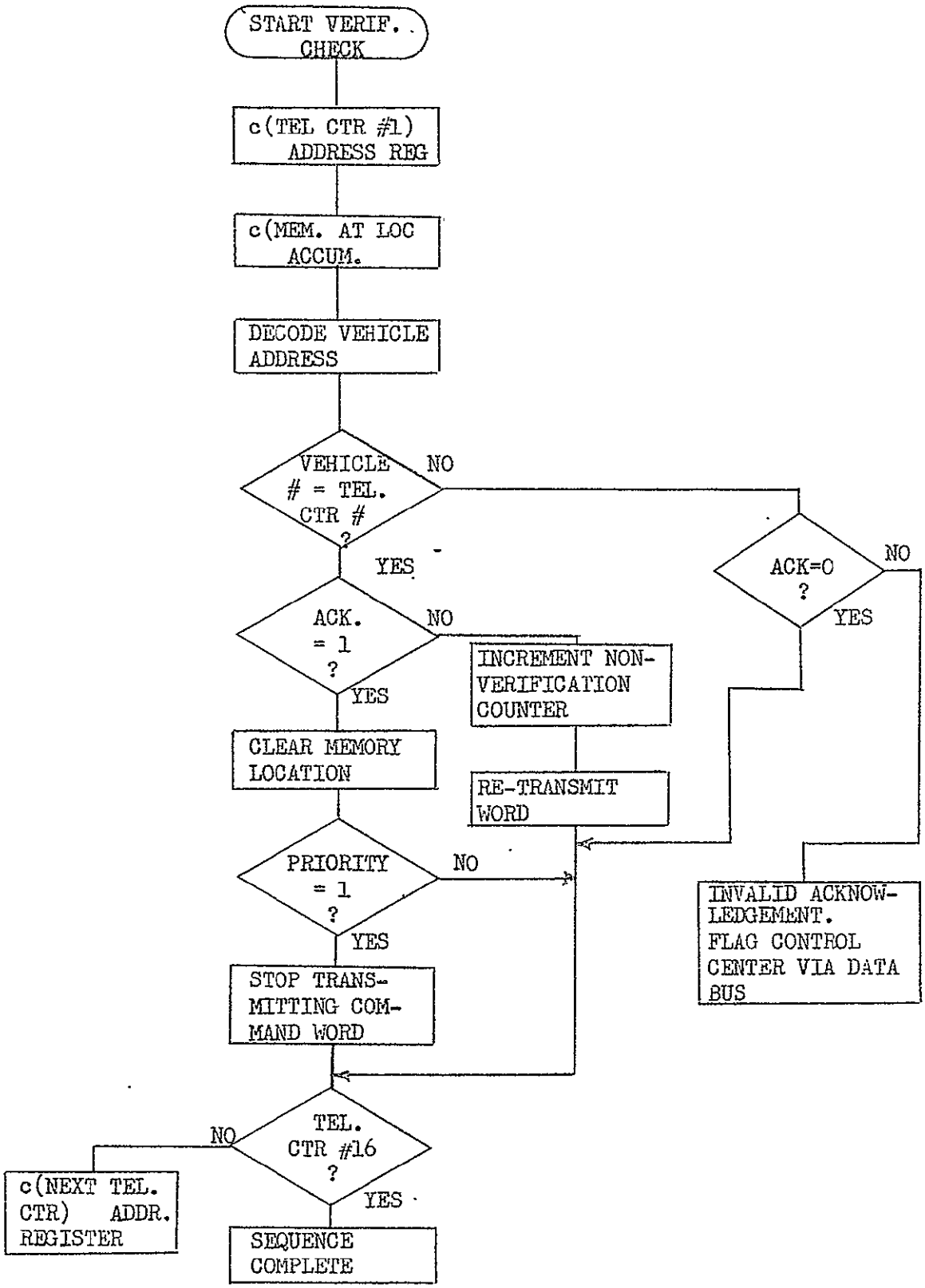
If a block of priority four data is being transmitted, the sequence will be interrupted whenever a word of higher priority level is received. Before the new word can be transmitted, an end of block word must be transmitted and information stored which indicates where in the data block to resume transmission. When the first word of a data block is transmitted, the contents of the real-time counter, N, must be stored to indicate the time a valid acknowledgement is due.

3.5.3 Priority One, Two and Three Acknowledgement Verification

The acknowledgement verification sequence shown in the flow chart (Figure 3.5.4) is repeated every millisecond. The contents of telemetry counter #1 is transferred to the verification memory address register. The command word at that location is transferred to the accumulator where the vehicle address and priority are determined. If the vehicle address is #1, then the last acknowledgement received from that vehicle should be a type "1". A type "0" acknowledgement indicates the word was not accepted, in which case the command word is transferred to the proper section of buffer memory for retransmission.

If the vehicle address is not #1, then the last acknowledgement received from vehicle #1 should be a type "0" to indicate that its own address was

FIGURE 3.5.4 PRIORITY ONE, TWO & THREE ACKNOWLEDGEMENT VERIFICATION



not decoded. The same operation is performed using the contents of each remaining telemetry counter as an address for a word in the verification memory.

3.5.4 Priority Four Acknowledgement Verification (Figure 3.5.5)

The first valid type "1" acknowledgement is received when the contents of the telemetry counter, k_i is equal to N, the real-time count at the time the first word of the data block was transmitted. If the acknowledgement received for the first or second words of the data block (header and starting address respectively) is a type "0", the entire block of data is re-transmitted. A type "0" acknowledgement received for any of the following words, except for the last word, will cause a logic "0" tag bit to be written into the corresponding location in buffer memory. The last word which must be re-transmitted as priority 1 if it is not acknowledged.

3.5.5 Retransmission of Non-Verified Priority Four Words (Figure 3.5.6)

Following the elapse of sufficient time for the first data word of the block to have been acknowledged, a check of the tag bits for each data word is started. A tag bit of logic "0" indicates that the word must be re-transmitted. The header word and new starting address word must precede the data word transmission, which is followed by an end of block word.

FIGURE 3.5.5 PRIORITY FOUR ACKNOWLEDGEMENT CHECK

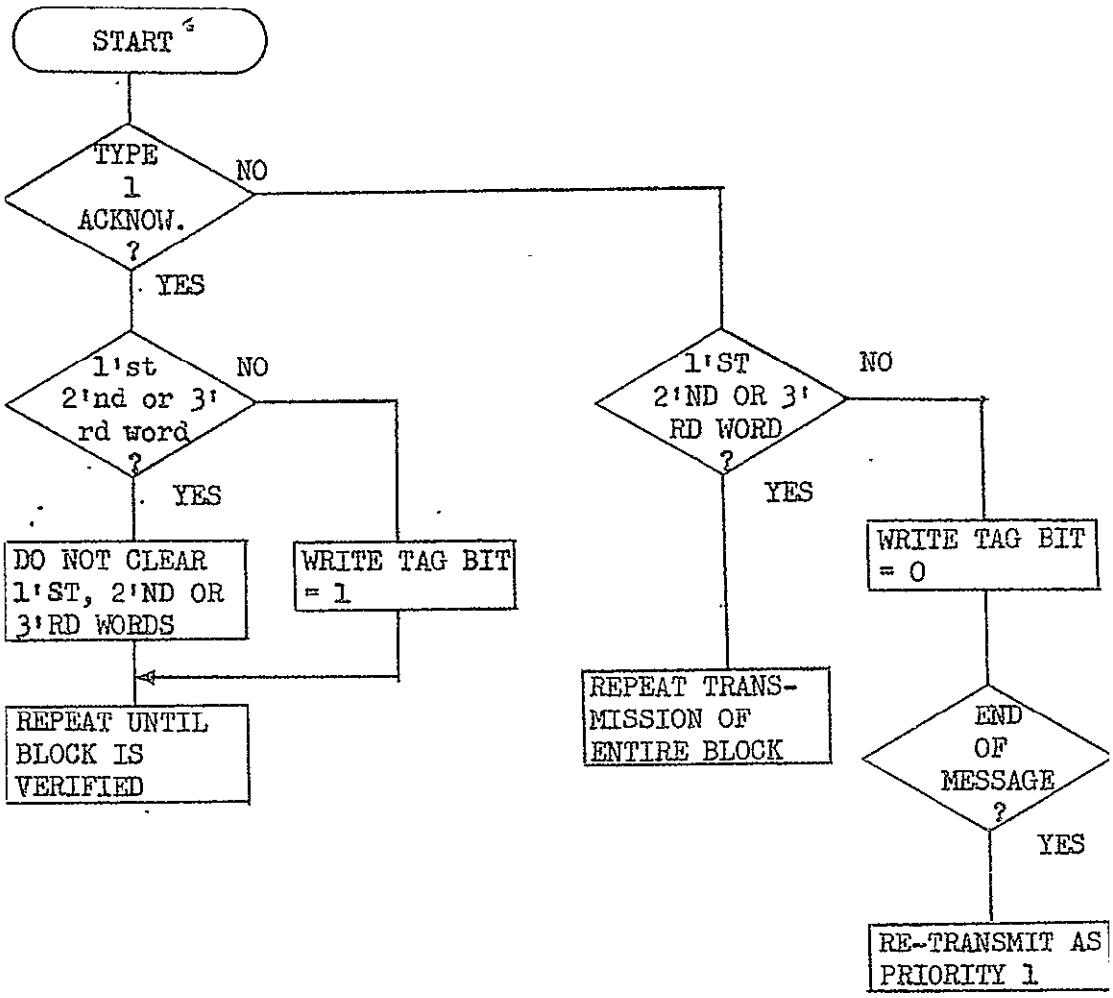
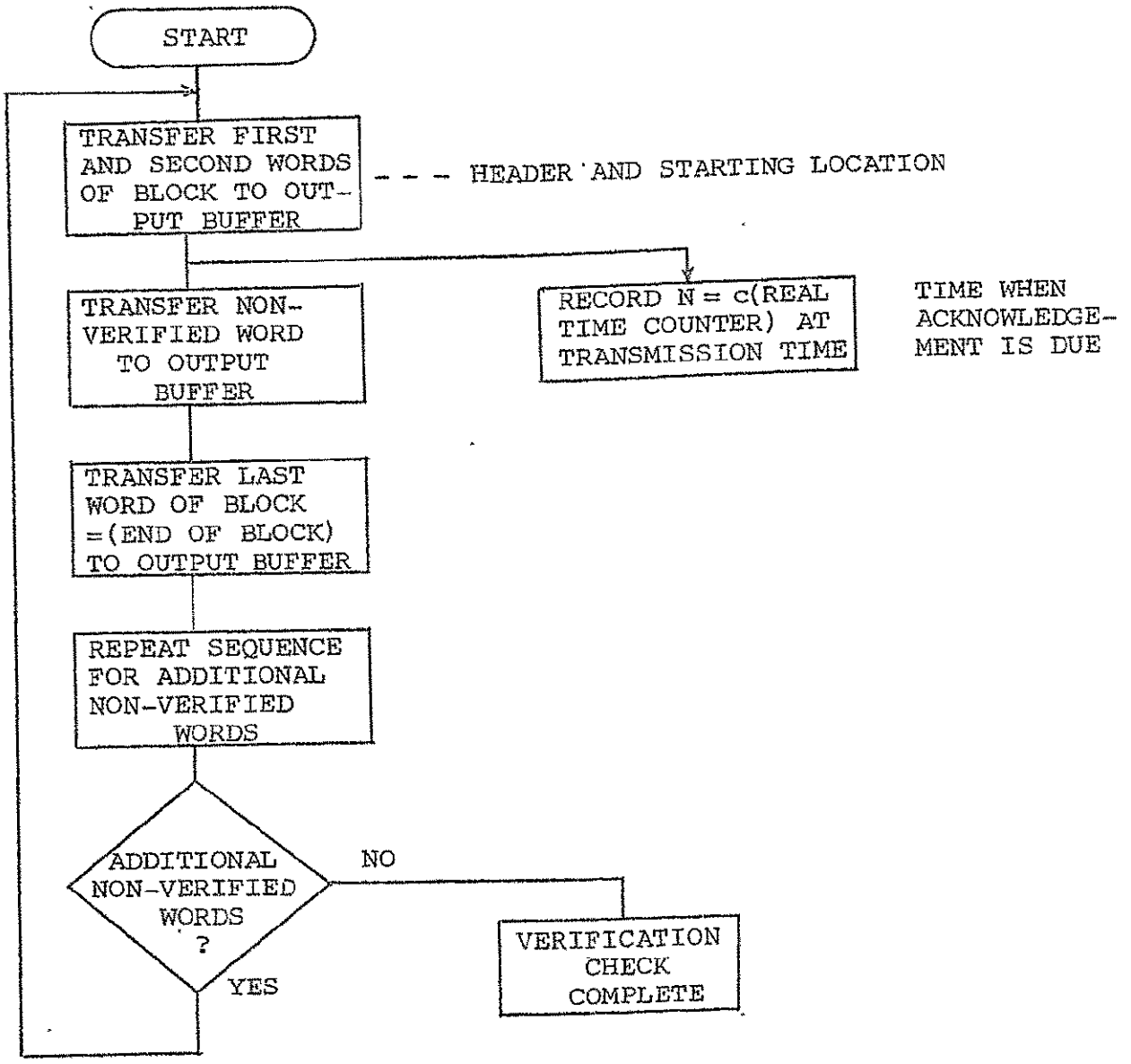


FIGURE 3.5.6

RETRANSMISSION OF NON-VERIFIED PRIORITY FOUR WORDS



3.6 DETAILED FLOW CHARTS OF CONTROLLER SEQUENCES

Figures 3.6.1 through 3.6.6 illustrate the detailed logic and arithmetic operations required to perform each sequence shown briefly in the previous flow charts.

3.6.1 Detailed Input Sequence (Figures 3.6.1 - 3.6.3)

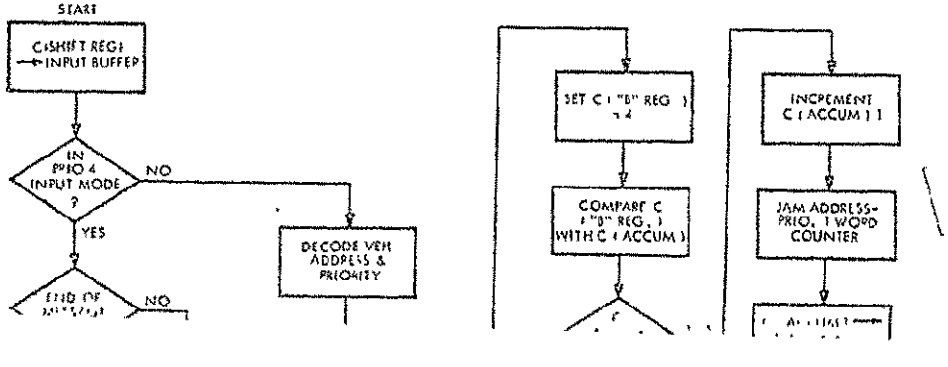
Each time a word is received for external transmission the word counter for the required section of buffer memory is checked to determine if storage space is available. If space is available, the input pointer is queried to determine where the word is to be stored. The address contained in the input pointer and the contents of the word counter are then both incremented, reflecting the addition of a word to memory.

3.6.2 Detailed Output Sequence (Figure 3.6.4)

The search for words in each section of buffer memory is performed by transferring the contents of the respective word counter to the arithmetic unit where it is tested for zero. If non-zero, the contents of the output pointer is transferred to the buffer memory address register and the word read from that location to the output buffer register. The contents of the output pointer is then incremented while the contents of the word counter is decremented.

FOLDOUT FRAME

FIGURE 3.6.1 DETAILED INPUT SEQUENCE



FOLDOUT FRAME 2

NOT REPRODUCIBLE

FIGURE 3.6.2 DETAILED INPUT SEQUENCE (CONT.)

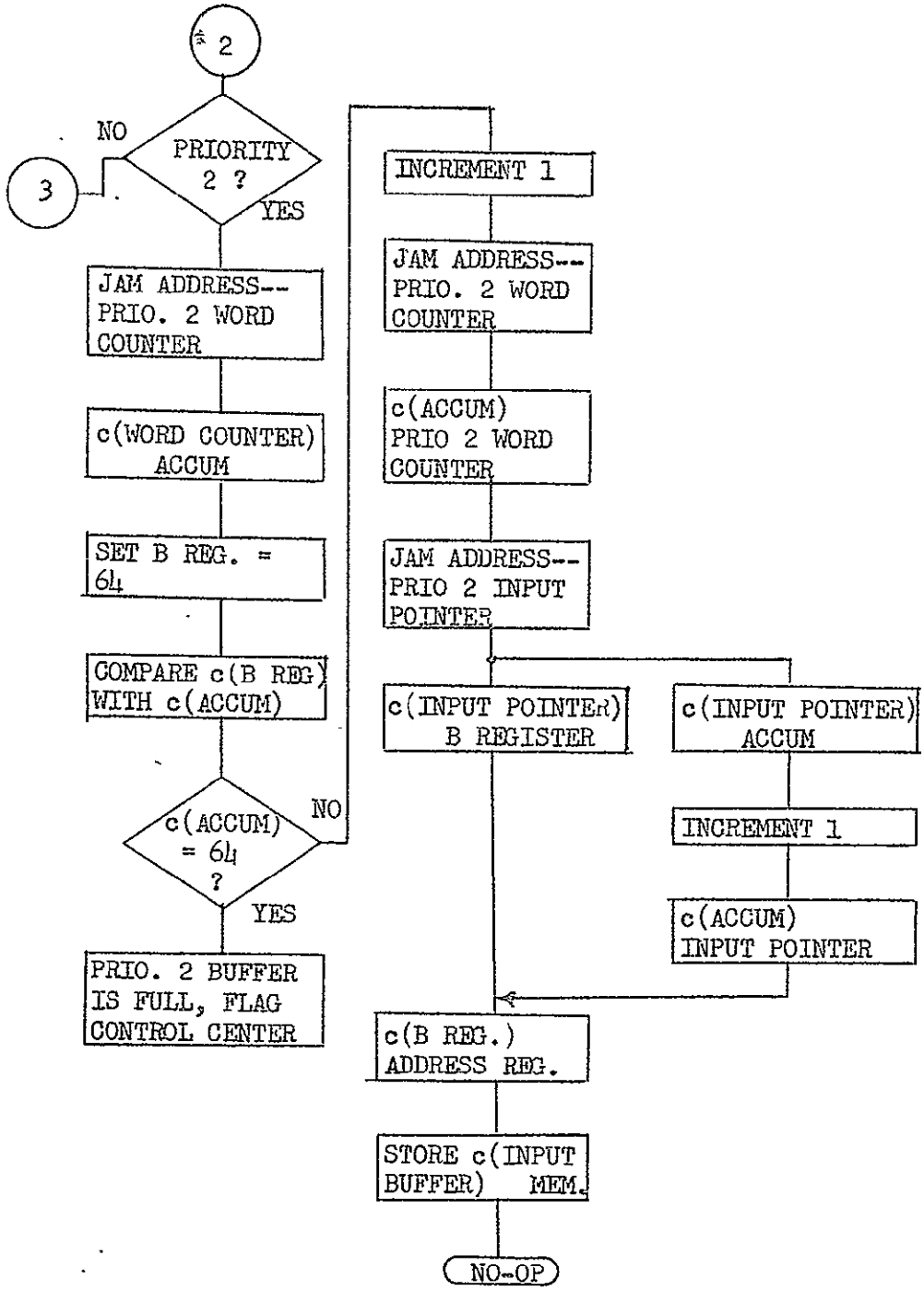


Figure 3.6.2 (continued)

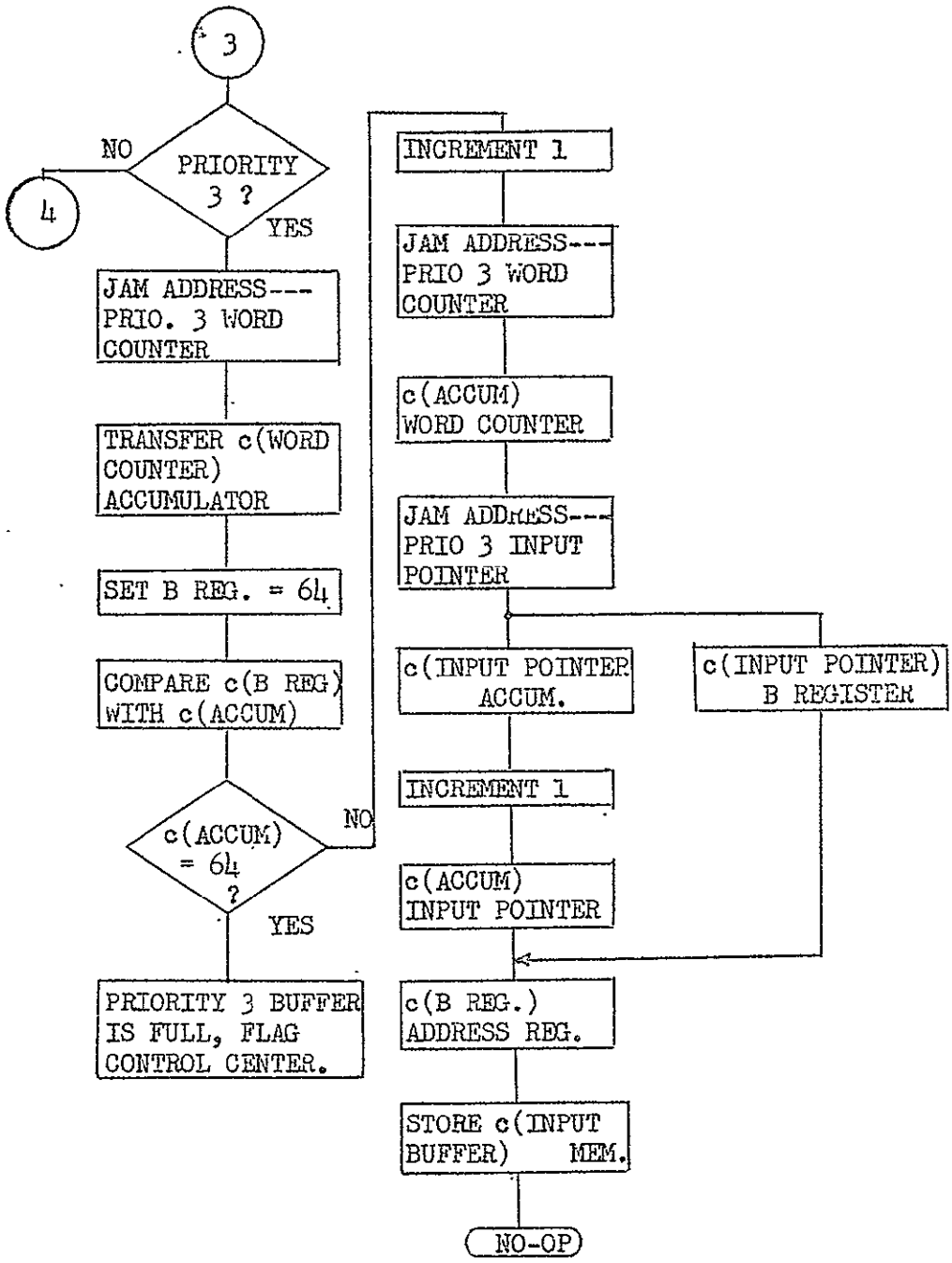
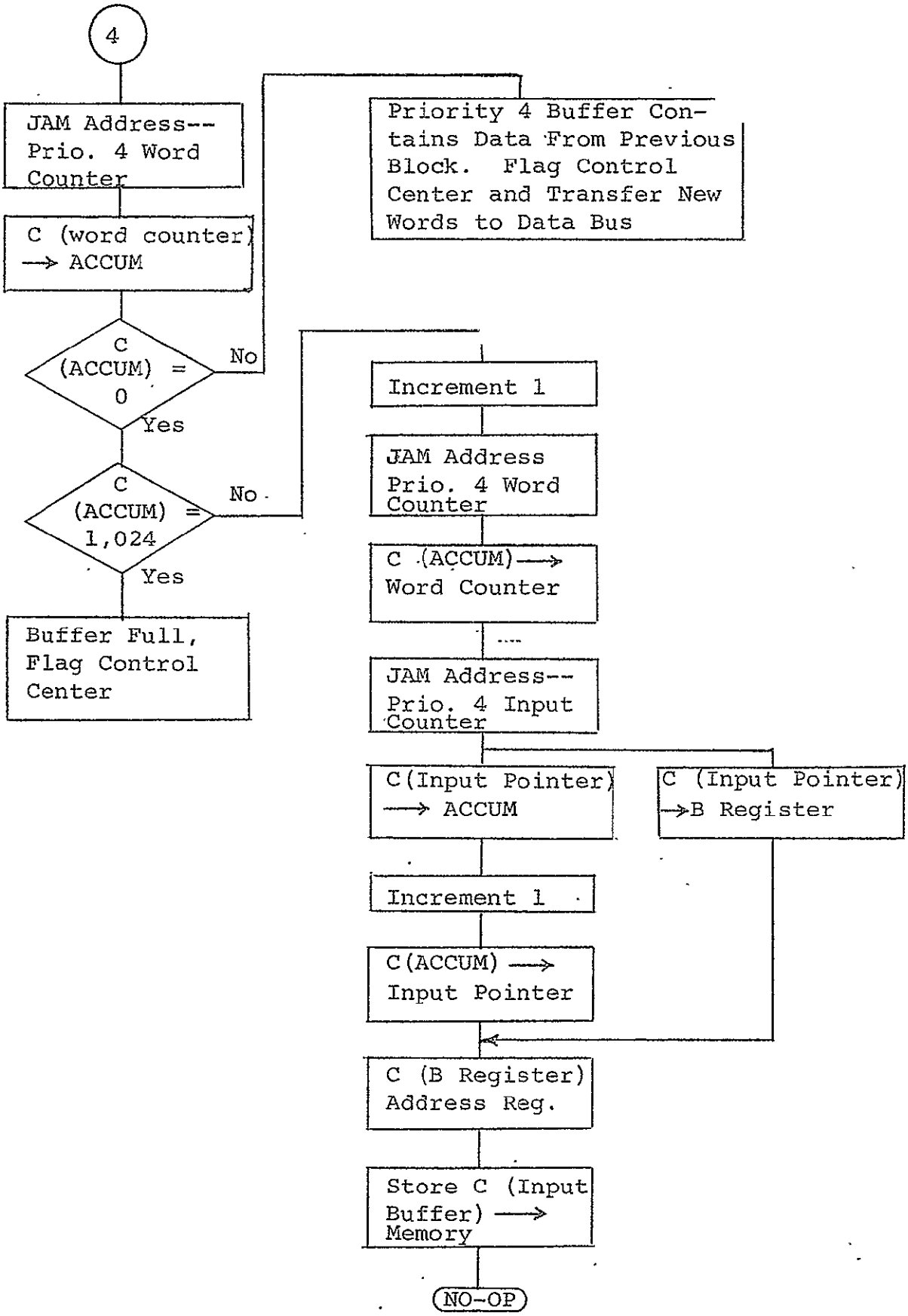


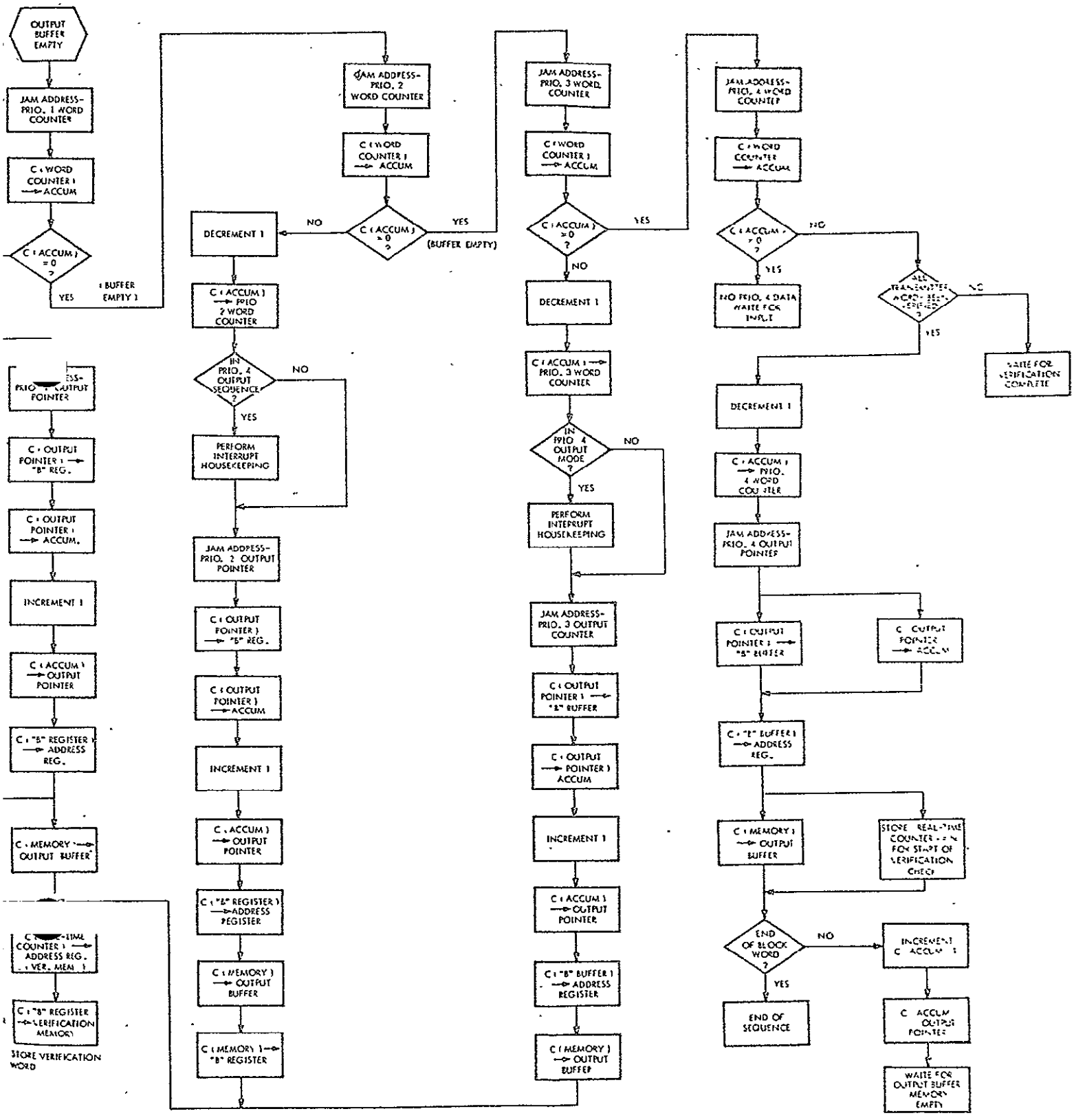
Figure 3.6.3

DETAILED INPUT SEQUENCE (con't)



FOLDOUT FRAME

IGURE 3.6.4 DETAILED OUTPUT SEQUENCE



POOR COPY 2

3.6.3 Detailed Acknowledgement Verification Sequence - Priority Four Data Words (Figure 3.6.5a)

This sequence shows how the verification pointer is used to determine when the acknowledgement is for the header or starting address words. If the content of the verification pointer is 1 or 2, the acknowledgement is for the header word or starting address, respectively. If either of these words is not acknowledged, the entire block of data is re-transmitted. The verification pointer is incremented after each acknowledgement is received until end of block is reached.

3.6.4 Detailed Verification Check - Priority Four (Figure 3.6.5)

The verification status of words in the priority four buffer section is started when the contents of the real-time counter is $N + (N - k_i) + 1$ (or $2N - k_i + 1$), where N = real-time count at the time the third word of the data block was transmitted, and k_i is the acknowledgement count for the addressed vehicle at that time. This quantity is incremented by 1 as the verification check progresses down the data block. The tag bit at each location is examined to determine whether or not the word was properly acknowledged. The word is retransmitted, along with the header, starting address and end of message words, when a "0" tag bit is detected.

A new block of priority level four data words is accepted only after a previous block of data has been transmitted, acknowledged and successfully verified.

FIGURE 3.6.5 (a)

PRIO. 4 DETAILED ACKNOWLEDGEMENT VERIFICATION SEQUENCE

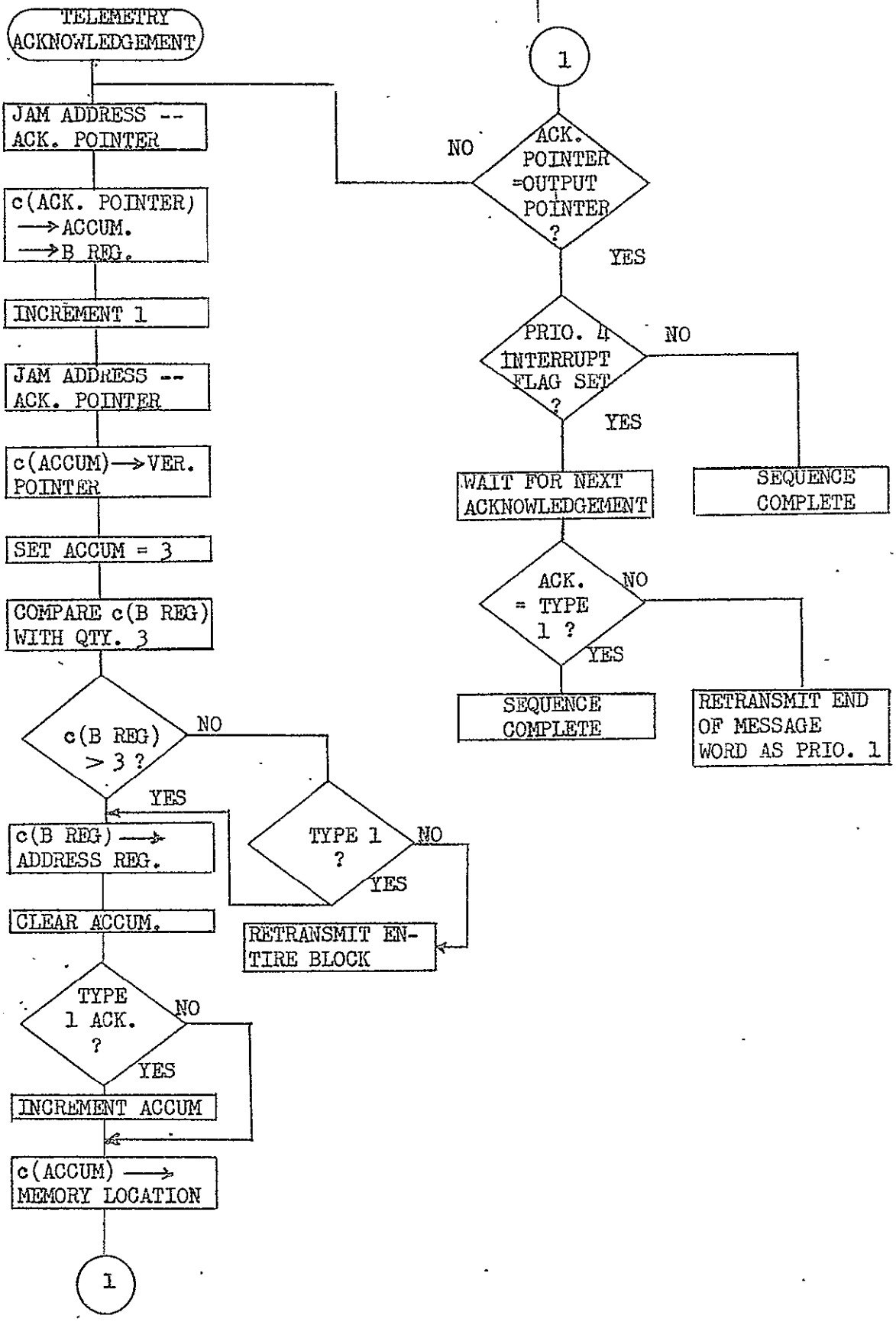
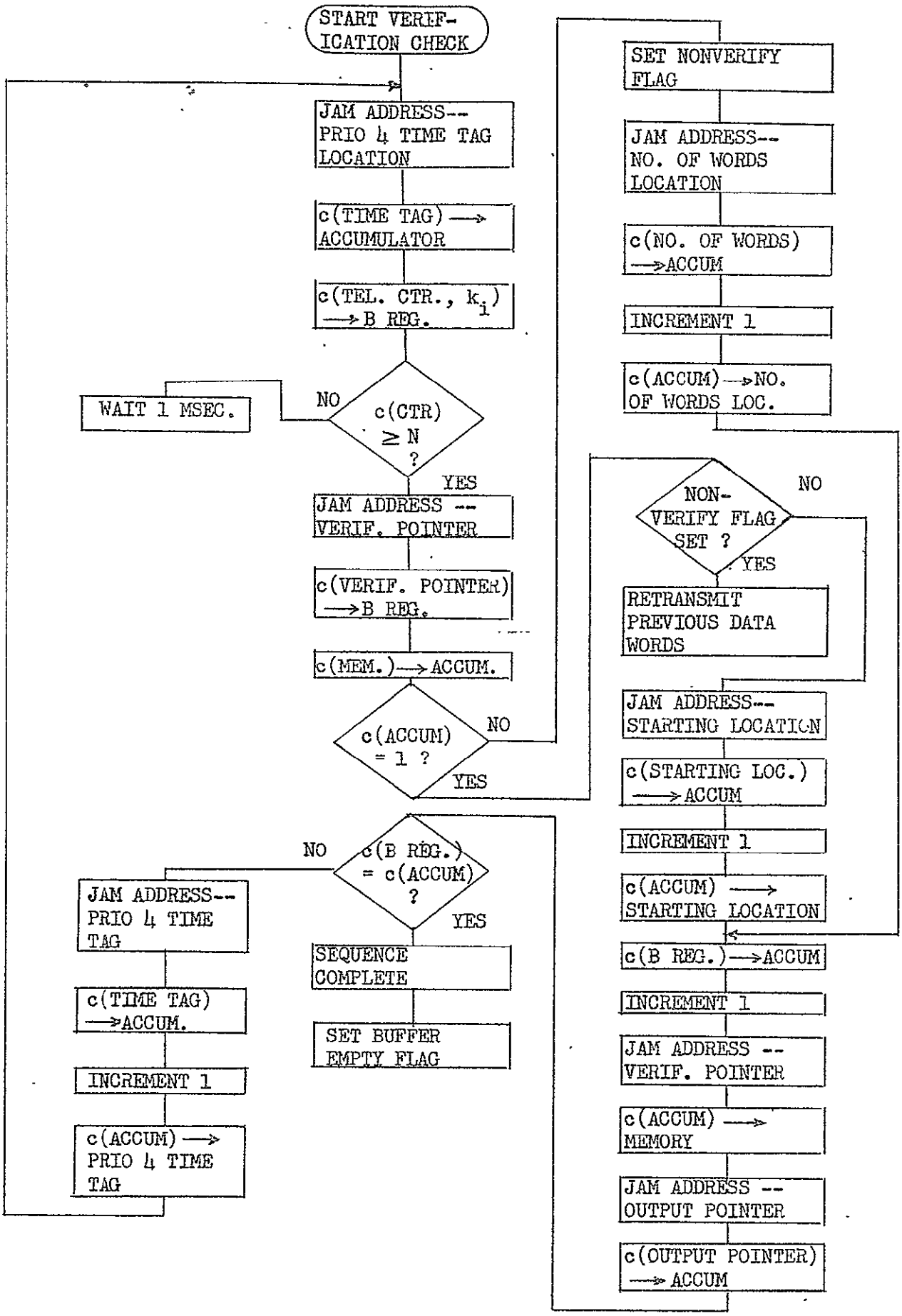


FIGURE 3.6.5

PRIORITY 4 VERIFICATION CHECK



3.6.5 Priority Four Retransmission Sequence (Figure 3.6.6)

This sequence shows how data transfer words which were not acknowledged are retransmitted.

3.7 PROGRAM TIMING FOR THE COMMAND CONTROLLER

This section illustrates the program timing for the control functions required to implement the detailed sequences which were flow-charted in Section 3.5.

Each program cycle has been divided into sixteen basic time intervals. The duration of each interval will depend, in the final analysis, upon the logic technology and memory systems selected. A time of one-half microsecond is consistent with the utilization of MOS technology in both the logic and memory sections. In this case the total program cycle time (16 interval) is 8 microseconds.

If additional operations are required to complete a sequence at the end of a program cycle, the program memory is read during the last time interval. The instruction register is incremented prior to this time if the next program address is sequentially located. If not, a new program address is loaded into the instruction register. When a sequence is completed, a no-operation (NO-OP) control function inhibits controller program operation until an interrupt is received.

FIGURE 3.6.6

PRIO. 4 RETRANSMISSION SEQUENCE

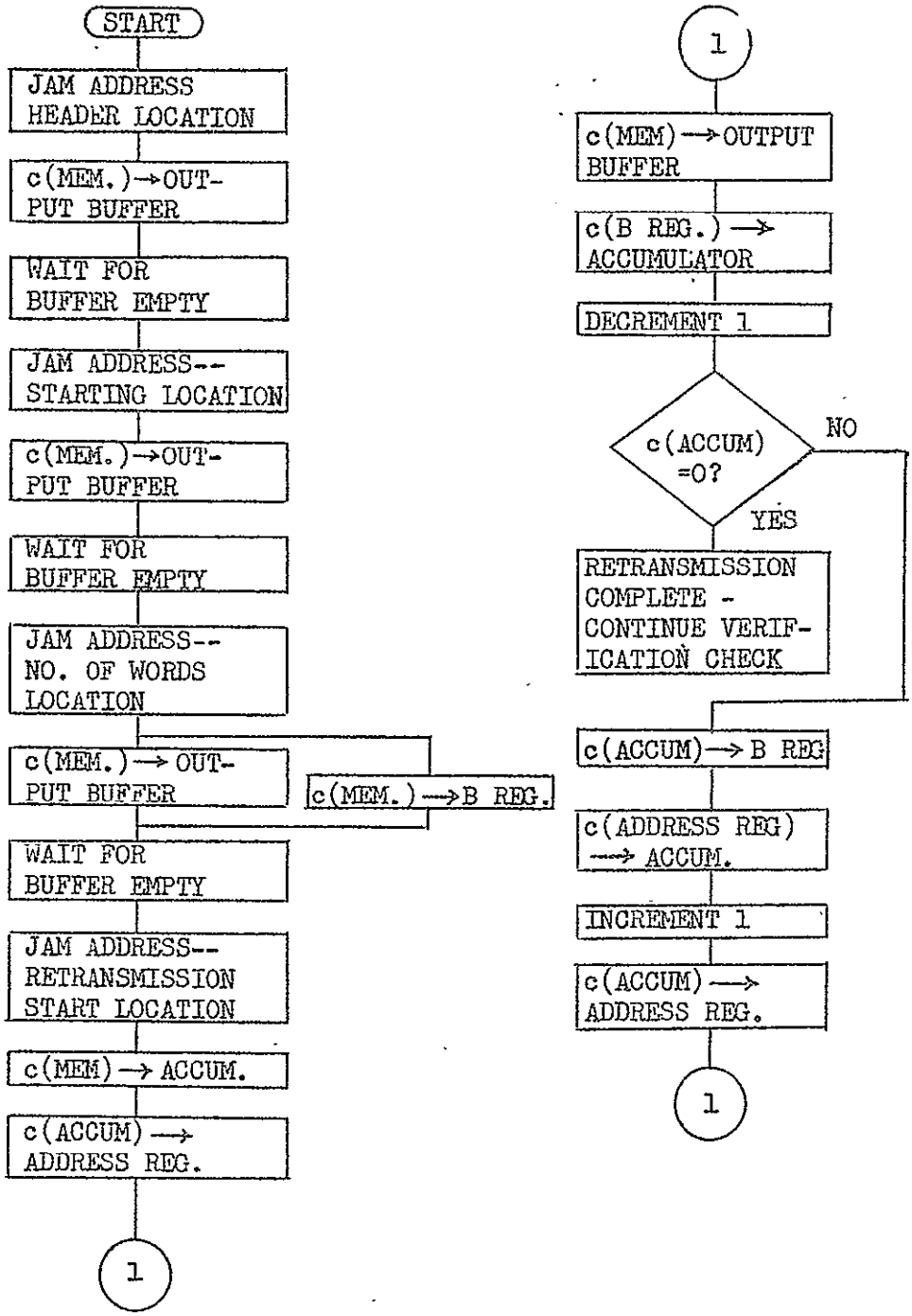


FIGURE 3.7.1

INPUT SEQUENCE

(Controller Not In Priority ④ Input Mode)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
C(Input Buffer)→Data Bus	X		[if	v.		address		=		space base]						
Set Priority 4 Input Flag	X		[if	space		base	∘	priority		4]						
Prog. Address Priority ① Input			[if	priority		1]										X
Prog. Address Priority ② Input			[if	priority		2]										X
Prog. Address Priority ③ Input			[if	priority		3]										X
C(Input Buffer)→Data Bus	X		[if	priority		4	∘	Buffer		Empty]						
Reset Buffer Empty Flag	X															
Set Priority 4 Input Flag	X		[if	priority		4]										
Prog. Address Priority ④ Input			[if	priority		4	∘	Buffer		Empty]						X

FIGURE 3.7.1.1

INPUT SEQUENCE - PRIORITY ①

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
JAM Adress Priority 1 Word Counter	X	X	X	X	X	X	X	X	X							
Read Memory	X	X	X													
C(Memory)→Accum.			X													
Set B Reg. = 4	X															
Compare C(B Reg.) with C(Accum)				X	X											
Flag Data Bus						X	[if	B	Reg.	=	Accum]					
Increment Accum 1						X										
Write Memory								X	X							
C(Accum.)→Word Counter							X	X	X							
Increment Inst. Reg.										X						
Read Control Memory																X

INPUT SEQUENCE - PRIORITY 1 (con't)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
JAM Address Priority 1 in-pointer	X	X	X	X												
Read Memory	X	X	X													
C(Input Pointer)⟶B Reg.			X													
C(Input Pointer)⟶Accum.			X													
Increment Accum. 1				X	X											
Write Memory							X	X								
C(Accum)⟶Memory						X	X	X								
C(B Reg.)⟶Address Reg.									X							
Write Memory													X	X		
C(Input Buffer)⟶Memory												X	X	X		
No-OP														—	X	

FIGURE 3.7.1.2 INPUT SEQUENCE - PRIORITY 2

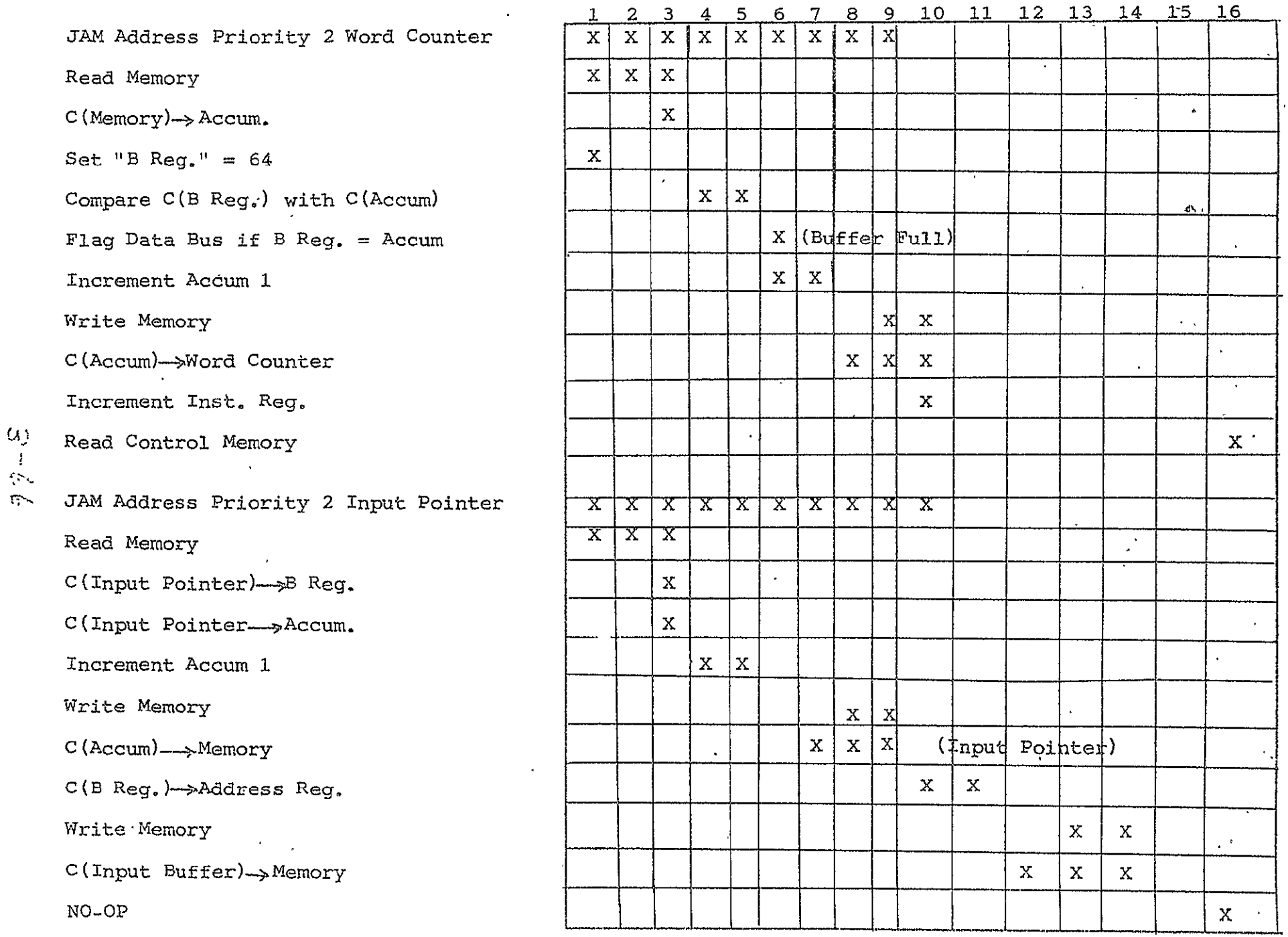


FIGURE 3.7.1.3

INPUT SEQUENCE - PRIORITY 3

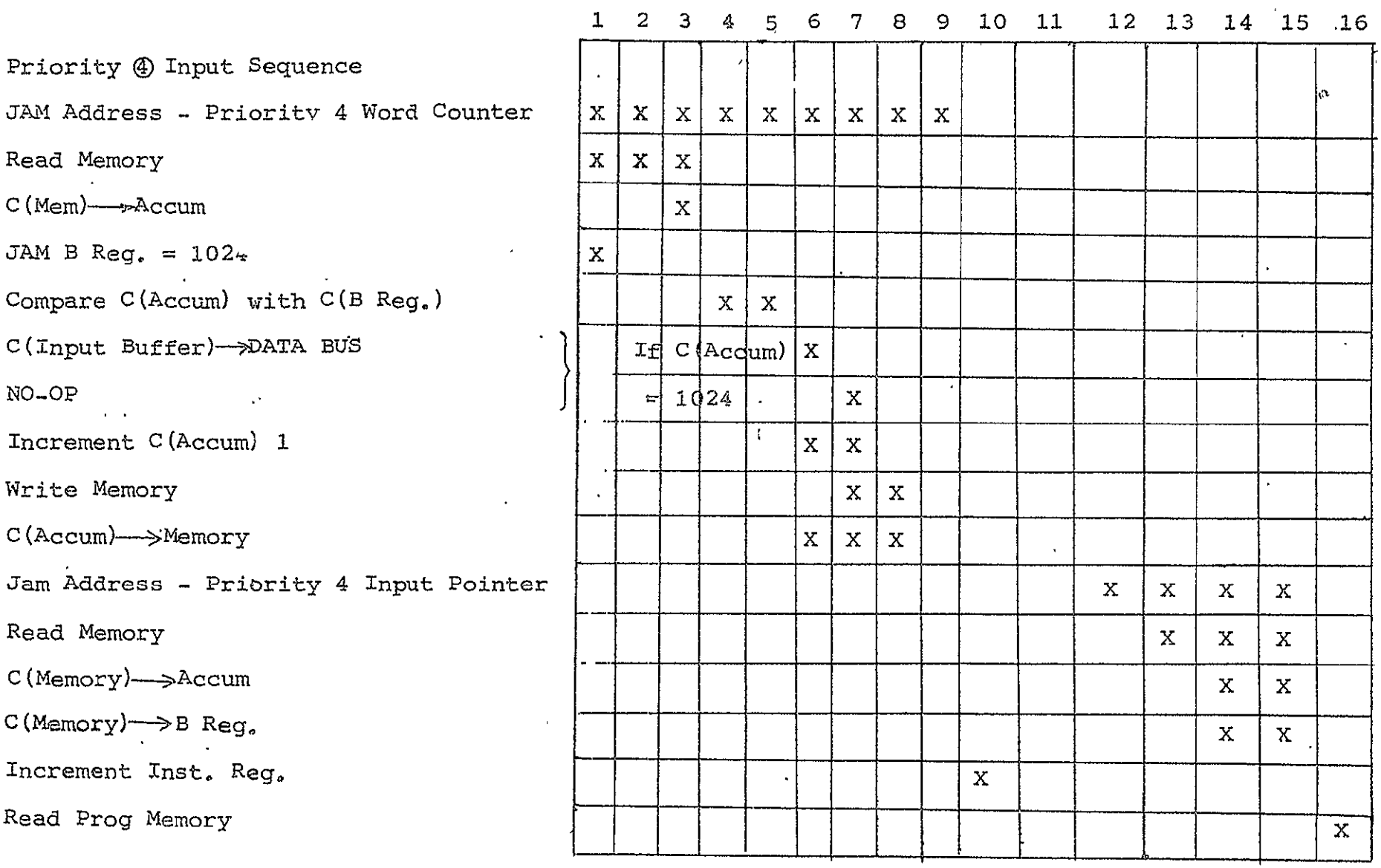
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
JAM Address Priority 3 Word Counter	X	X	X	X	X	X	X	X	X							
Read Memory	X	X	X													
C(Mem)→Accum			X													
Set B Reg. = 64	X															
Compare C(B Reg.) & (Accum)				X	X											
Flag Data Bus if Equal						X	(Buffer Full)									
Increment Accum 1						X	X									
Write Memory									X	X						
C(Accum)→Word Counter								X	X	X						
Increment Inst. Reg.										X						
Read Control Memory																X

INPUT SEQUENCE - PRIORITY 3 (con't)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
JAM Address Priority 3 Input Pointer	X	X	X	X	X	X	X	X	X							
Read Memory	X	X	X													
C(Input Pointer)→ B Reg.			X													
C(INput Pointer)→Accum			X													
Increment Accum 1				X	X											
Write Memory							X	X								
C(Accum)→ Memory						X	X	X	(Input Pointer)							
C(B Reg.)→ Address Reg.									X	X						
Write Memory													X	X		
C(Input Buffer)→ Memory												X	X	X		
No-Op												—			X	

FIGURE 3.7.1.4

PRIORITY 4 INPUT SEQUENCE



PRIORITY 4 INPUT SEQUENCE (con't)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Increment Accum.	X	X														
JAM Address - Priority 4 Input Pointer		X	X	X	X											
Write Memory				X	X											
C(Accum) → Memory			X	X	X											
Clear Address Reg.						X										
C(B Reg.) → Address Reg.							X									
Address Memory									X	X	X					
Write Memory										X	X					
C(Input Buffer) → Memory									X	X	X					
NO-OP												X				

FIGURE 3.7.2

INPUT SEQUENCE

(Controller in Priority 4 Mode)

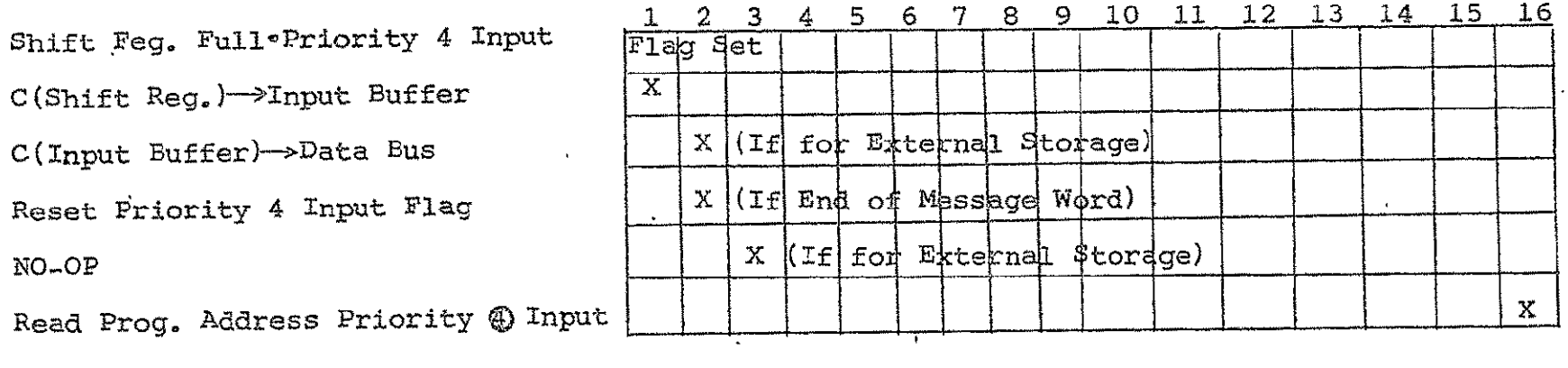
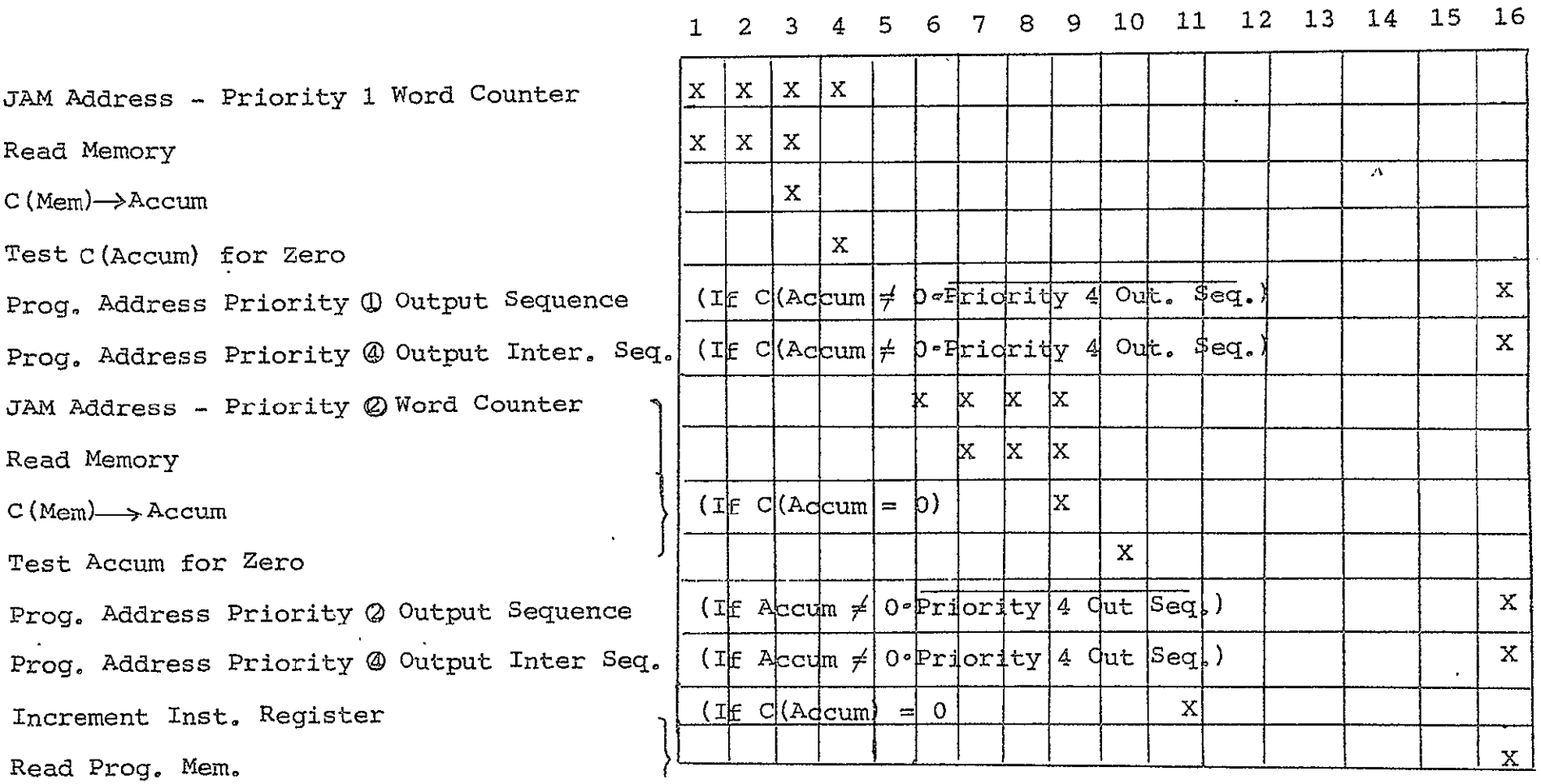


FIGURE 3.7.3

OUTPUT BUFFER EMPTY



OUTPUT BUFFER EMPTY (con't)

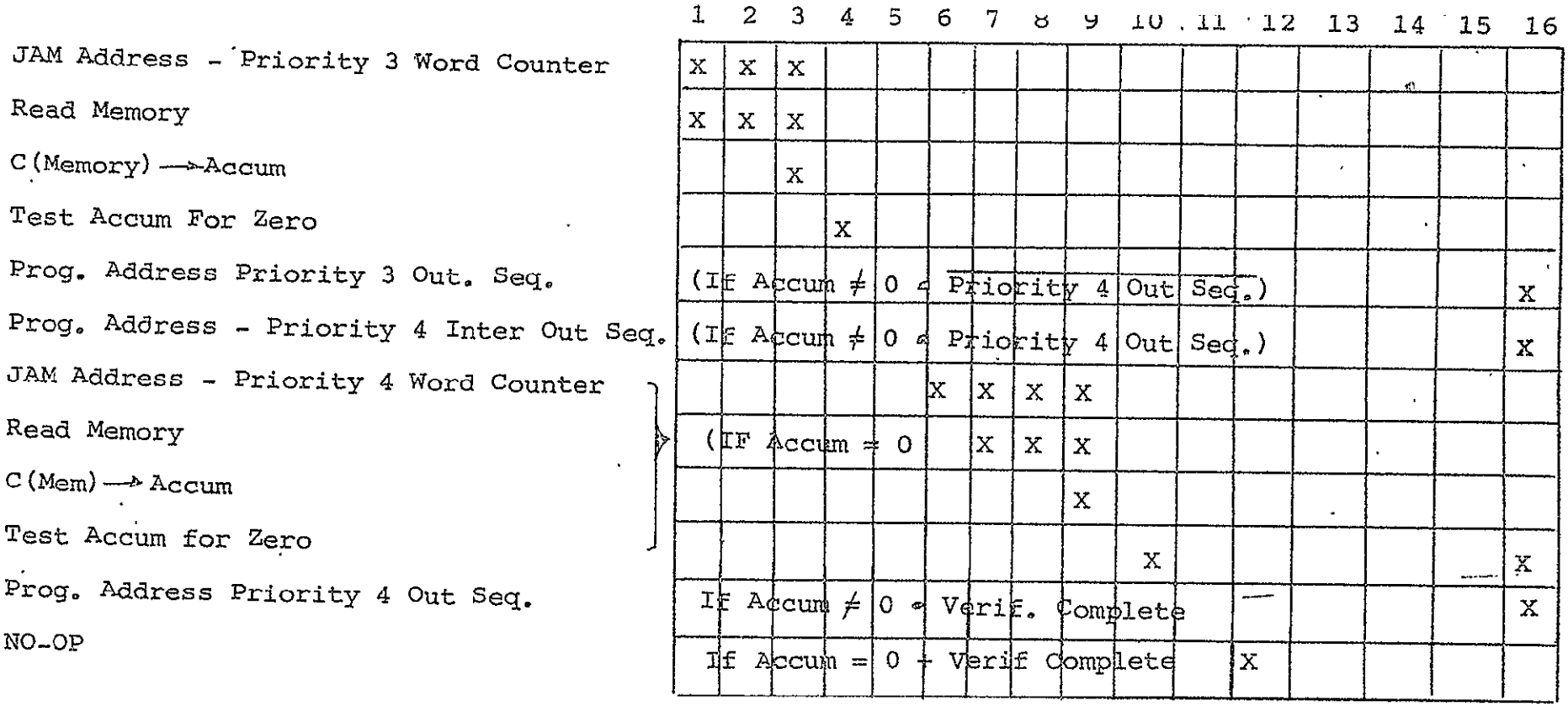


FIGURE 3.7.3.1

OUTPUT SEQUENCE PRIORITY 1

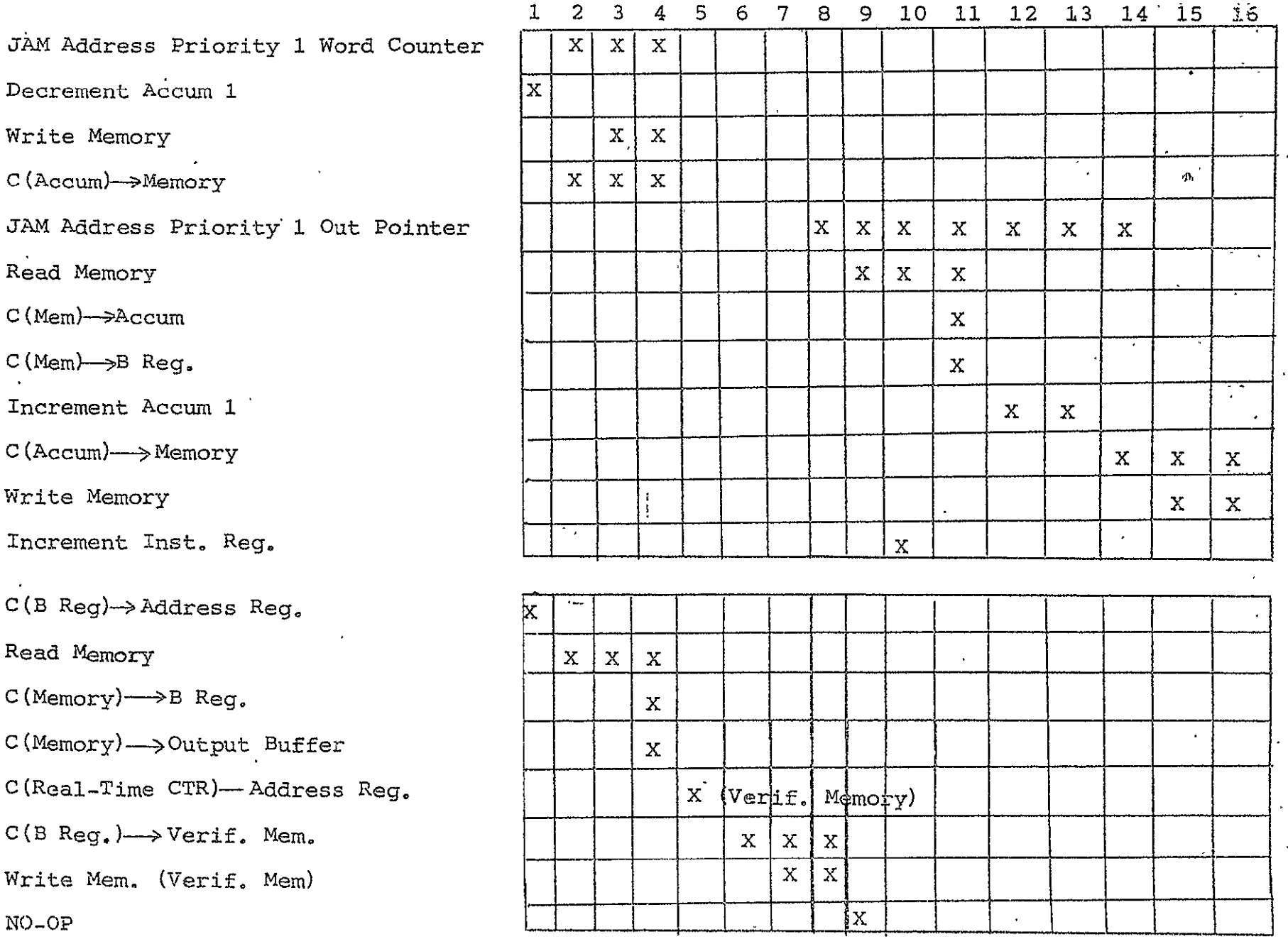


FIGURE 3.7.3.2

OUTPUT SEQUENCE - PRIORITY ②

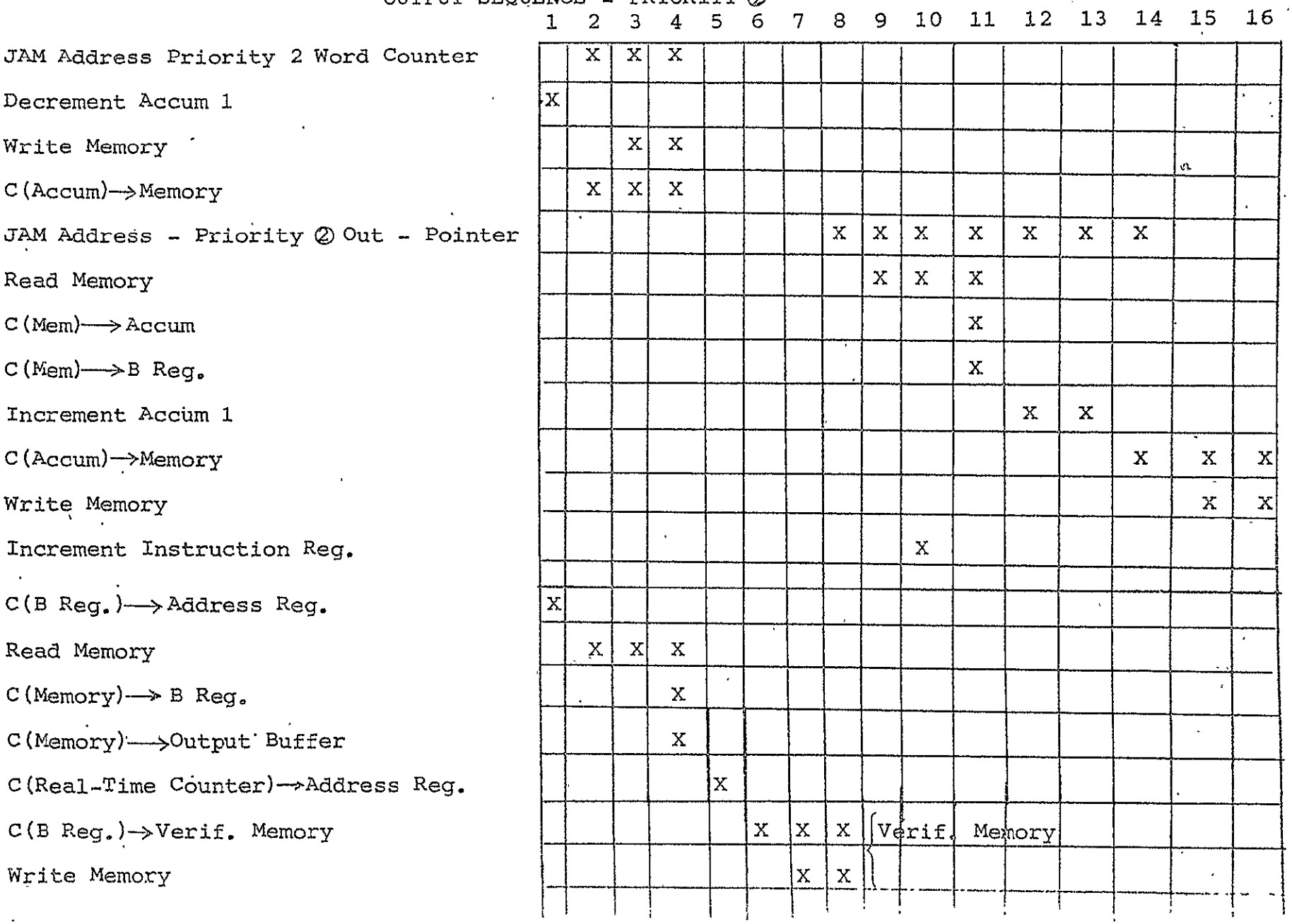


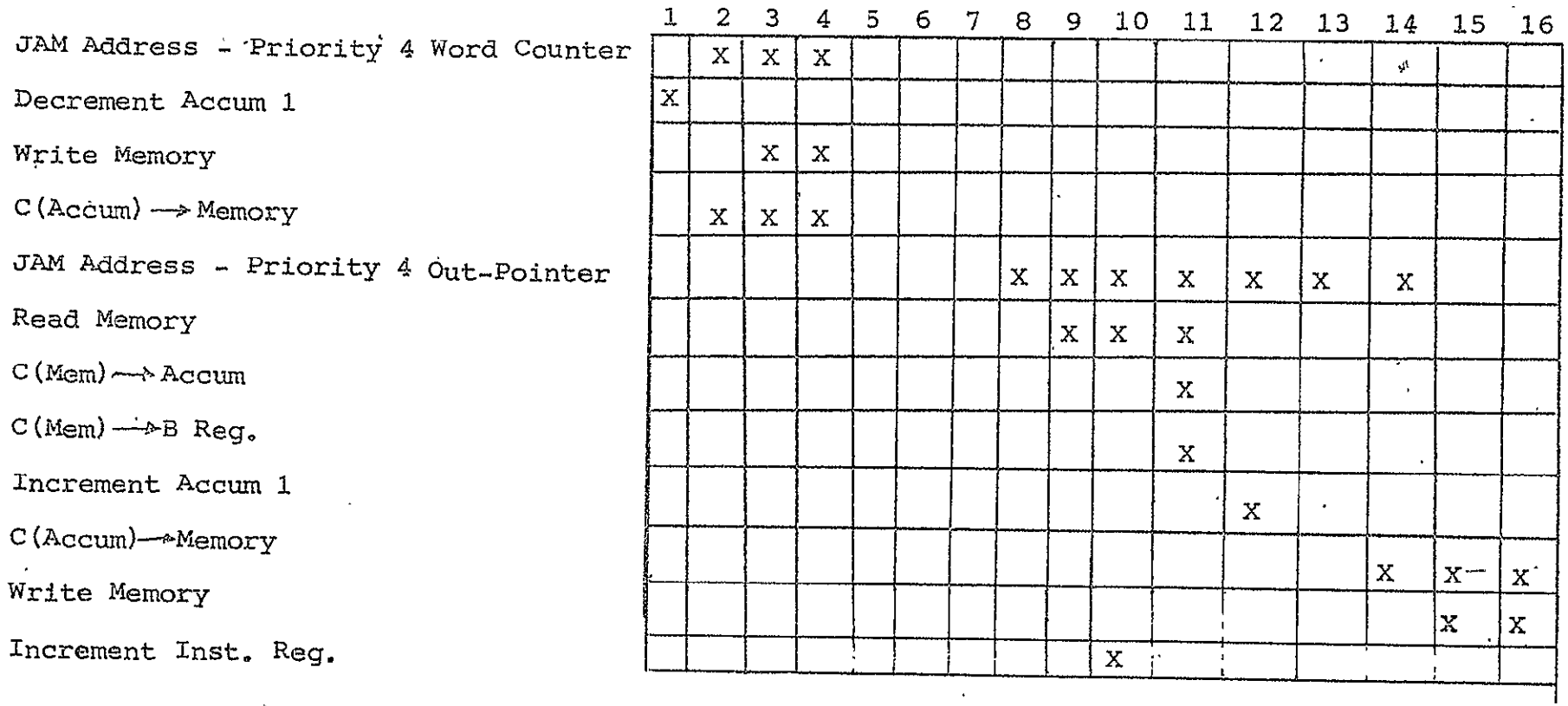
FIGURE 3.7.3.3

OUTPUT SEQUENCE - PRIORITY ③

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
JAM Address - Priority 3 Word Counter		X	X	X												
Decrement Accum 1	X															
Write Memory			X	X												
C(Accum)⟶ Memory		X	X	X												
JAM Address - Priority ③ Out-Pointer									X	X	X	X	X	X		
Read Memory									X	X	X					
C(Mem)⟶Accum											X					
C(Mem)⟶B Reg.											X					
Increment Accum 1												X	X			
C(Accum)⟶Memory														X	X	X
Write Memory															X	X
Increment Instruction Reg.										X						
C(B Reg.)→Address Reg.	X															
Read Memory		X	X	X												
C(Memory)⟶B Reg.				X												
C(Memory)⟶Output Buffer				X												
C(Real-Time Counter)⟶Address Reg.					X											
C(B Reg.)⟶Verif. Memory						X	X	X	Verif. Memory							
Write Memory (Verif. Memory)							X	X								
NO-OP									X							

FIGURE 3.7.3.4

OUTPUT SEQUENCE PRIORITY 4



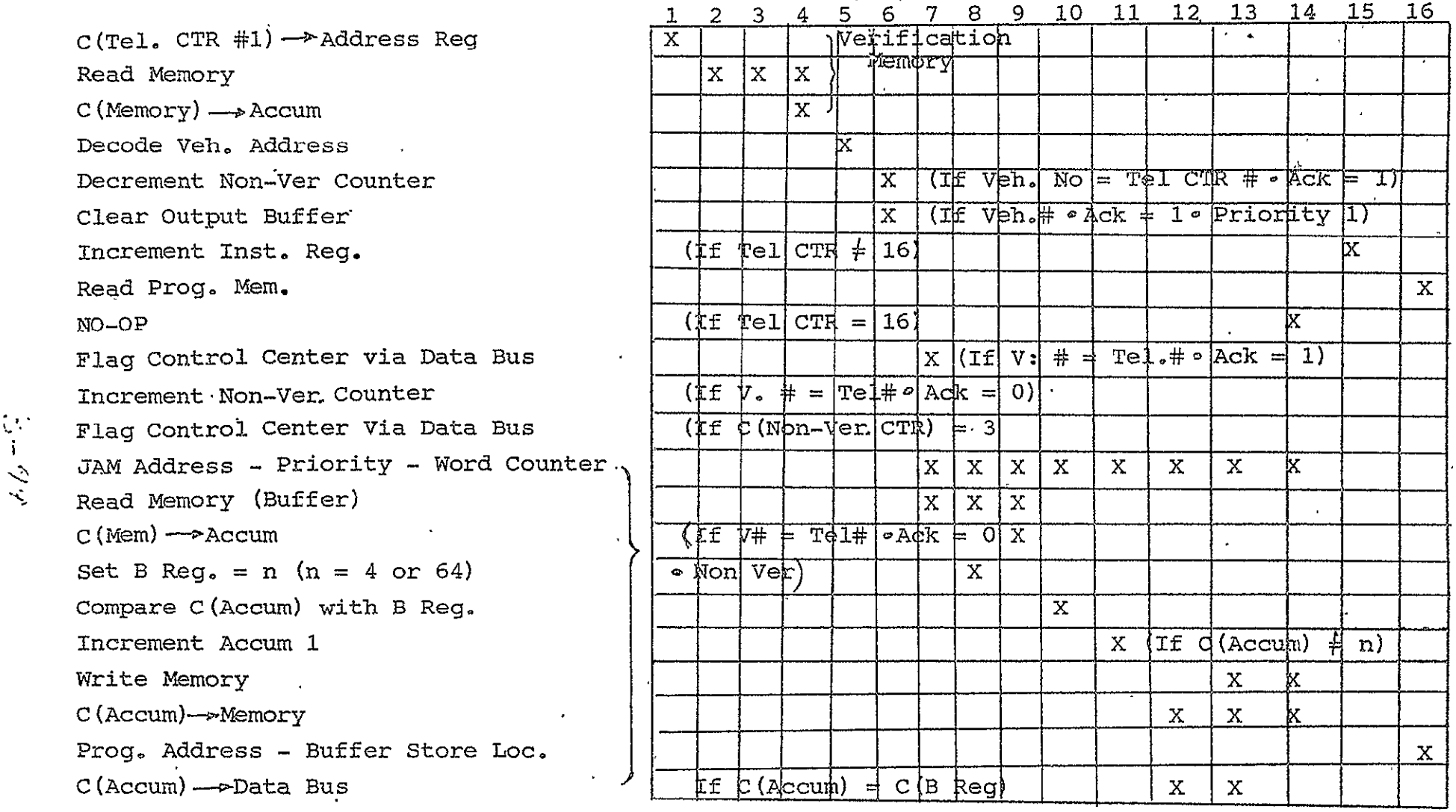
OUTPUT SEQUENCE - PRIORITY ④ (con't)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
C(B Reg.)⟶Address Reg.	X															
Read Memory		X	X	X												
C(Memory)⟶Output Buffer				X												
C(Real Time Counter)⟶Priority 4 Time Tag													X	X	X	
JAM Address - Priority 4 Time Tag Loc.		If	First		Word								X	X	X	
Write Memory														X	X	
NO-OP (If End of Message					X											
Increment Accum 1						X										
JAM Address - Output Pointer							X	X	X							
Write Memory								X	X							
C(Accum) Memory = (Output Pointer							X	X	X							
NO-OP																X

2-13

FIGURE 3.7.4

ACKNOWLEDGEMENT CHECK - PRIORITY 1, 2, & 3



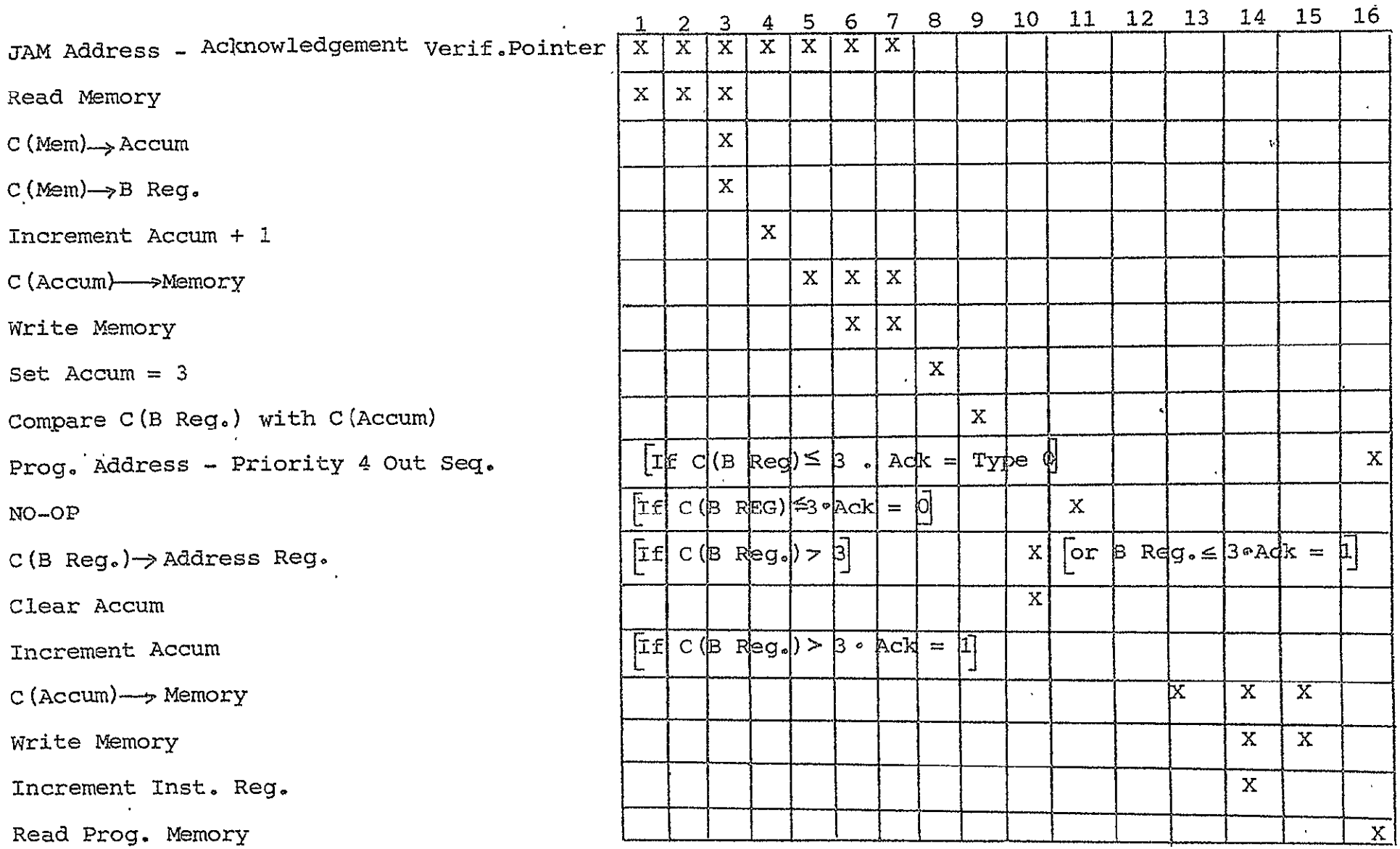
ACKNOWLEDGEMENT CHECK - PRIORITY 1, 2, & 3 (con't)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
JAM Address - Priority - Input Pointer	X	X	X	X	X	X	X	X								
Read Memory	X	X	X													
C(Mem) → Accum, B Reg			X													
Increment Accum 1				X												
Write Memory (Buffer)							X	X								
C(Accum) → Input Pointer						X	X	X								
C(B Reg.) → Address Reg.									X							
Read Verif. Memory								X	X	X						
C(Verif. Mem) → Accum										X						
Write Buffer Memory													X	X		
C(Accum) → Memory												X	X	X		
Prog. Address Seq. for next Tel. Counter																X

Seq. is repeated for each Tel Counter)

FIGURE 3.7.5

PRIORITY 4 ACKNOWLEDGEMENT CHECK



PRIORITY 4 ACKNOWLEDGMENT CHECK (con't)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
JAM Address - Priority 4 Output Pointer	X	X	X													
Read Memory	X	X	X													
C(Mem)→Accum.			X													
JAM Address - Acknow. Pointer						X	X	X								
Read Memory							X	X	X							
C(Memory)→B Reg.									X							
Compare C(Accum) ◦ C(B Reg.)										X						
NO-OP (Until Next Tel. Ack.)		(If	C(Accum)			≠	C(B	Reg)			X					
NO-OP - Sequence Complete		(If	C(Accum)			=	C(B	Reg)			X	◦Priority	4	Int.		Flag
Prog. Address - End of Message Ack		If	C(Accum)			=	C(B	Reg.)◦	Priority	4	Int.	Flag				X
(Next Acknowledgement)																

FIGURE 3.7.5.1

PRIORITY 4 ACKNOWLEDGEMENT CHECK - END-OF-MESSAGE WORD

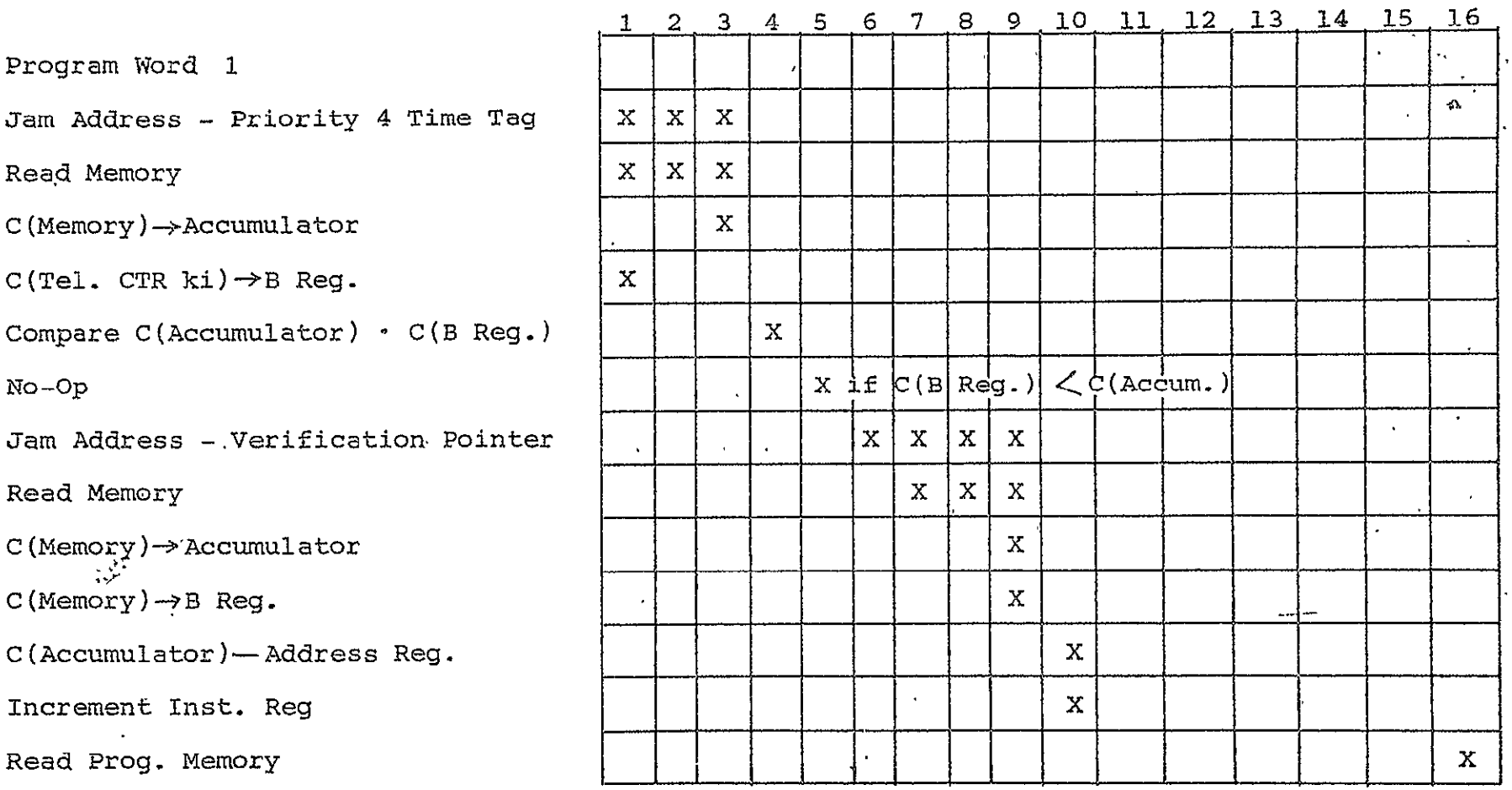
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
NO-OP	X	If	Ack.	=	Type	1										
JAM Address Priority 1 Word Counter	X	X	X	X	X	X	X	X								
Read Memory		X	X	X												
C(Memory) ⟶ Accum				X												
Set C(B Reg) = 4			X													
Compare C(Accum) . C(B Reg)					X											
End-of-Message→Data Bus						X	(If	(B	Reg.)	=	C(Accum)					
NO-OP							X	If	(B	Reg.)	=	(Accum)				
Increment Accum 1								X								
C(Accum) ⟶ Memory										X	X	X				
Write Memory											X	X				
Increment Inst. Reg.													X			
Read Prog. Memory																X

PRIORITY 4 ACKNOWLEDGEMENT CHECK - END-OF-MESSAGE WORD (con't)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
JAM Address - Priority 1 Input Pointer	X	X	X	X	X	X	X	X								
Read Memory	X	X	X													
C(Mem)→Accum			X													
C(Mem)→B Reg.			X													
Increment Accum 1				X												
C(Accum)→Memory					X	X	X									
Write Memory						X	X									
C(B Reg.)→Address Reg.								X								
End-of-Message Word→ Memory											X	X	X			
Write Memory												X	X			
NO-OP														X		

FIGURE 3.7.6

PRIORITY 4 VERIFICATION CHECK



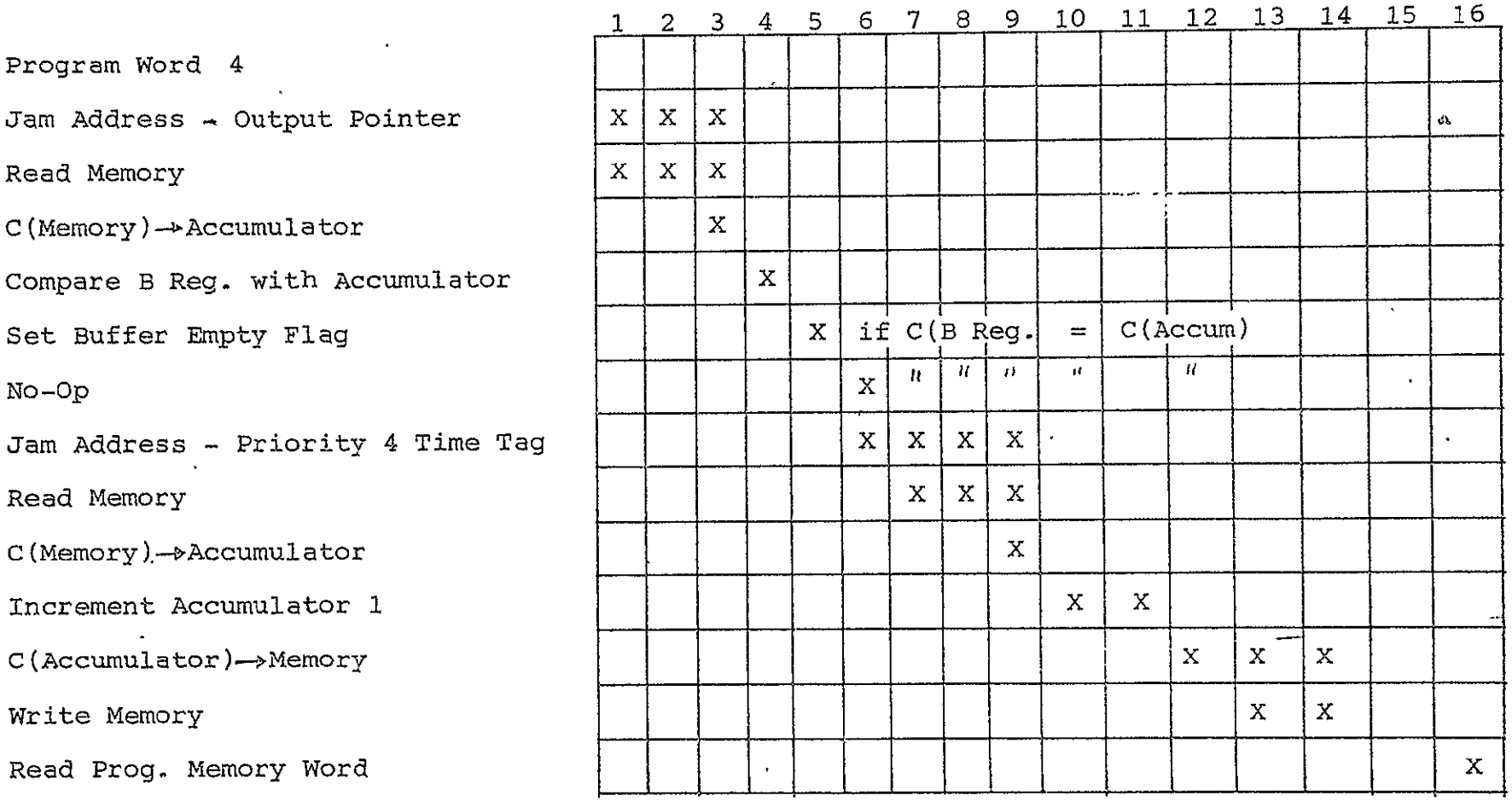
PRIORITY 4 VERIFICATION CHECK (con't)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Program Word 2																
Read Memory	X	X	X													
C(Memory)→Accumulator			X													
Check Accumulator Tag Bit for "1"				X												
Prog. Address Re-Trans. Sequence		if Tag Bit				=	1	· Non-Verify Flag								X
Increment Inst. Reg.	if Tag Bit				=	1	· $\overline{\text{Non-Verify Flag}}$									
Read Prog. Memory																X
Prog. Address - Prio. 4 Non-Verif. Seq.			if Tag Bit			=	0									X

PRIORITY 4 VERIFICATION CHECK (con't)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Program Word 3																
Jam Address - Starting Location	X	X	X	X	X	X	X	X								
Read Memory	X	X	X													
C(Memory)→Accumulator			X													
Increment 1				X	X											
C(Accumulator)→Memory						X	X	X								
Write Memory							X	X								
C(B Reg.)→Accumulator									X							
Increment Accumulator 1										X	X					
Jam Address - Verification Pointer												X	X	X		
Write Memory													X	X		
C(Accumulator)→Memory												X	X	X		
Increment Instruction Reg.										X						
Read Prog. Memory																X

PRIORITY 4 VERIFICATION CHECK (con't)



PRIORITY 4 VERIFICATION CHECK (con't)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Program Word 5																
Set Non-Verify Flag	X															
Jam Address - "No. of Words" Location	X	X	X	X	X	X	X	X	X							
Read Memory	X	X	X													
C(Memory)→Accumulator			X													
Increment Accumulator 1				X	X											
C(Accumulator)→Memory						X	X	X								
Write Memory							X	X								
C(B Reg.) →Accumulator									X							
Increment Accumulator 1										X	X					
Jam Address - Verification Pointer												X	X	X		
Write Memory													X	X		
C(Accumulator)→Memory												X	X	X		
Read Prog. Memory Word 4																X

3-87

FIGURE 3.7.6.1

PRIORITY 4 VERIFICATION CHECK

(RETRANSMISSION SEQUENCE)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Program Word 6 - a																
Jam Address - Header Location	X	X	X													
Read Memory	X	X	X													
C(Memory)→Output Buffer			X													
Increment Instruction Reg.										X						
No-Op											X					
Read Program Memory	(if Buffer Empty)															X

PRIORITY 4 VERIFICATION CHECK

(RETRANSMISSION SEQUENCE) (con't)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Program Word 6 - b																
Jam Address - Starting Address	X	X	X													
Read Memory	X	X	X													
C(Memory)→Output Buffer			X													
Increment Instruction Reg.										X						
No-Op											X					
Read Prog. Memory (if Buffer Empty)																X

PRIORITY 4 VERIFICATION CHECK

(RETRANSMISSION CHECK) (con't)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Program Word 6 - c																
Jam Address - No. of Words Location	X	X	X													
Read Memory	X	X	X													
C(Memory)→Output Buffer			X													
C(Memory)→B Reg.			X													
Increment Instruction Reg.										X						
Jam Address - Retrans. Start Location						X	X	X	X							
Read Memory							X	X	X							
C(Memory)→Accumulator									X							
C(Accumulator)→Address Reg.										X						
Increment Instruction Reg.										X						
No-Op											X					
Read Prog. Memory (if Buffer Empty)																X

2-92

PRIORITY 4 VERIFICATION CHECK

(RETRANSMISSION SEQUENCE)(con't)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Program Word 6. - d																
Read Memory	X	X	X													
C(Memory)→Output Buffer			X													
C(B Reg.)→Accumulator		X														
Decrement Accumulator			X	X												
Check Accumulator for Zero					X											
No-Op	}	If	C	(Accum)	=	0				X						
Read Program Word 1	}															X
C(Accumulator)→B Reg.						X										
C(Address Reg.)→Accumulator							X									
Increment Accumulator								X	X							
C(Accumulator)→Address Reg.										X						
No-Op											X					
Read Program Word 6 - d	(if Output Buffer Empty)															X

4.0 CODING TECHNIQUES - IMPLEMENTATION COMPLEXITY VS. PERFORMANCE

4.1 INTRODUCTION

After some preliminary screening, four basic coding techniques were selected for further, more detailed, consideration; vis., BCH codes with algebraic decoding; projective and Euclidean geometry codes with threshold decoding; concatenated algebraic codes; and convolutional codes with Viterbi decoding. The following sections summarize the results of these investigations.

4.2 ALGEBRAIC AND THRESHOLD DECODABLE CODES

4.2.1 Block Code Decoding Strategies

The most important techniques for decoding random-error-correcting block* codes can be divided into four basic categories:

(1) Maximum-likelihood decoding
(2) Algebraic decoding
(3) Threshold decoding
(4) Probabilistic decoding

The first of these techniques is optimum, i.e. minimize the probability of a decoding error, but is easily eliminated from further consideration for the present application on the basis of the complexity of the resulting decoder. The limited applicability of the fourth decoding method combined with its random decoding delay make it generally unsuitable for the command system application also. The other two methods, however, both merit further consideration. Both approaches lead to decoders which can be practically implemented, at least for some codes.

* The present discussion is limited to block code decoding methods. Convolutional coding schemes, and in particular the Viterbi decoding algorithm, will be examined subsequently.

The codes which are most compatible with the algebraic decoding method (in particular, the BCH codes) and those which are best adapted to majority logic decoding (the projective and Euclidean geometry codes) are too similar for practical block lengths to allow us to eliminate one method in favor of the other. (BCH codes generally have a greater minimum distance for a given word length and redundancy than do projective and Euclidean geometry codes. On the other hand, the threshold decoding algorithm, unlike most algebraic algorithms, provides some error correction capability beyond the minimum distance of the code, so these two effects tend to cancel each other.)

Since it does not seem possible to select in advance either the algebraic or the threshold decoding method over the other, therefore, both methods, and the codes associated with them, will be considered here. The approach will be to determine the performance attainable using two techniques, to select those codes associated with the two methods which best meet the system objectives (if such codes exist), and to compare the resulting decoders in terms of their complexity.

The more information supplied to the decoder concerning the identity of the successive channel symbols, the more reliably it should be able to decode. Ideally, the input to the decoder should be a sequence of real numbers, each number corresponding to the likelihood that a particular channel symbol is a "one" or a "zero". Generally, a decoder designed to exploit such information would be unacceptably complex. (Several exceptions to this statement will be discussed later.) A compromise between the desire to design the decoder to utilize more information about the received

symbols and the desire to keep its complexity to a minimum is afforded through the use of erasures. That is three rather than two demodulator outputs are recognized, a "one' a "zero", and an "erasure". The latter output occurs when the received symbol is not sufficiently like a "one" or a "zero" to be identified with either.

Both algebraic and threshold decoders are able to accommodate erasures with little increase in complexity. Moreover, it is possible, and potentially advantageous, to place thresholds (not to be confused with the internal threshold decoder thresholds) on the number of observed errors and erasures which are to be tolerated before a decoded word is rejected as too unreliable. To be more specific, suppose any two code words in an (n, k) code dictionary (n=word length, k=number of information bits per code word) are separated by a guaranteed minimum Hamming distance d. Errors -only decoding algorithms are based on the fact that if a received n-tuple is found to differ in n_e bit positions from a true code word, and if $n_e \leq \left\lfloor \frac{d-1}{2} \right\rfloor$ (the square brackets denoting the integer part of the enclosed fraction), then it must differ in at least $d-n_e > n_e$ positions from any other word. Since one code word is therefore uniquely closest to the received n-tuple it presumably represents the word most likely transmitted. Similarly, in the case of erasures-and-errors-decoding, any code word differing in $\left\lfloor \frac{d-n_r-1}{2} \right\rfloor$ or fewer positions from a recieved n-tuple containing n_r erasures must clearly be uniquely closest to that n-tuple. Since such a word again represents the best guess as to the transmitted code word, the decoder uses this as the basis for its decision. (If a received n-tuple contains d or more erasures, generally no attempt is made to decode it.)

An obvious generalization on this strategy, useful when repeat requests are possible, is to recognize the decoded n-tuple as a valid code word only if this code word is sufficiently "close"

to the original n-tuple. That is, let d_o be defined as the error-correction distance and d_1 as the erasure-correction distance. A decoded word is now to be accepted as valid only if the n-tuple which gave rise to it contains fewer than d_1 erasures and only if the number of unerased positions in which the two n-tuples (the received n-tuple and the code word closest to it) differ is $\left\lfloor \frac{d_o - n_r - 1}{2} \right\rfloor$ or less. When $d_o = d_1 = d$ this modified strategy is equivalent to the conventional erasure-and-error-correction decoding strategy. When $d_o = d_1 = 1$, either an erasure or an error results in rejected code word; i.e. the code is used as an error-and-erasure-detection code only. For intermediate values of d_o and d_1 some error and erasure patterns are corrected while others are considered too extensive for reliable correction and a repeat transmission is requested. This added flexibility in the decoding strategy is achieved at little cost in complexity provided a repeat request capability is already available. Yet when, as here, a code word accepted in error is immeasurably more serious than a rejected one, this additional degree of freedom in the decoding strategy is clearly well worth investigating.

Section 4.2.3 of this report outlines a procedure for bounding the performance of this generalized decoding strategy as a function of the parameters d_o and d_1. It is important to note that for this purpose the threshold and algebraic algorithms are essentially identical. Both can be designed to correct n_e errors and n_r erasures so long as $n_r < d_1$ and $n_e \leq \left\lfloor \frac{d_o - n_r - 1}{2} \right\rfloor$ with neither d_o nor d_1 exceeding the guaranteed minimum code distance d. The fact that threshold decoders can sometimes correct more than d errors will have neglible effect on the results to be derived there (which, in any event are only upper bounds on the actual error and rejection probabilities). Before these bounds can be developed however, it is necessary to discriminate between erasures and non-erasures. This is the subject of the next section.

4.2.2 Erasure Techniques

As mentioned in the preceeding paragraphs, an important parameter in examining the effectiveness of various coding schemes is the erasure threshold. If a received bit is too enmeshed in noise to be identified as either a zero or a one with any degree of reliability, it may be advantageous to treat it as completely unknown, that is, as an erasure. Most error-detecting and error correcting decoding algorithms can handle erasures with virtually no increase in complexity, a statement which does not generally apply it more than three output symbols are recognized (i. e. if the output of the bit detector is quantized into more than three levels.) Since increasing the number of quantization levels from two to three can be quite effective in reducing the probability of a rejected or erroneously accepted command, therefore, the technique is well worth considering.

The question consequently arises as to the most effective method for introducing erasures. Perhaps the most obvious method is, implied above, simply to define a threshold γ and test the bit detector output x against this threshold if $x > \gamma$ accept the received bit as a one; if $x < -\gamma$, accept it as a zero; and if $-\gamma \leq \leq \gamma$, call it an erasure. It will be noted however, that this method is quite sensitive to the signal level. If x is kept constant, for example, and the signal level drops, the percentage of erasures could rise precipitously even when the signal-to-noise ratio remains unaltered. For this reason, a second method was considered whose performance depends only on the signal-to-noise ratio, not on the individual signal or noise level. This method involves sampling the bit detector output several, say k, times during a bit period. If ℓ or more of these samples are larger than the preceeding sample (or greater than zero in the case of the first sample) the bit is called a one. If ℓ or more of the samples are less than their predecessors (or less than zero for the first sample) it is called a zero. Any other event is identified as an erasure. (This scheme, with k = 5, will be

recognized as the method used in the Apollo command link.)

After some consideration, it was concluded that the first of those two methods is preferable. The reason is that the major objection to the first approach, its signal-level dependence, is almost completely nullified if the receiver has an automatic gain control (or even a hard-limiter) as will presumably be the case. Moreover, the second method, in effectively converting each bit into a number of sub-bits, will result in significantly less reliable decisions than the first. The mathematical details supporting this conclusion follow.

4.2.2.1 The Threshold Method

Let μ and $-\mu$ be the expected outputs of the bit detector when the received bit is a one and a zero, respectively, let σ^2 be the variances of this output, and let γ be the threshold, as previously defined. Then, if the perturbing noise is Gaussian,

$$p = \text{pr (bit error)} = \text{pr(bit error} \mid \text{"one" transmitted)pr("one")} + \text{pr (bit error} \mid \text{"zero" transmitted)pr("zero")}$$

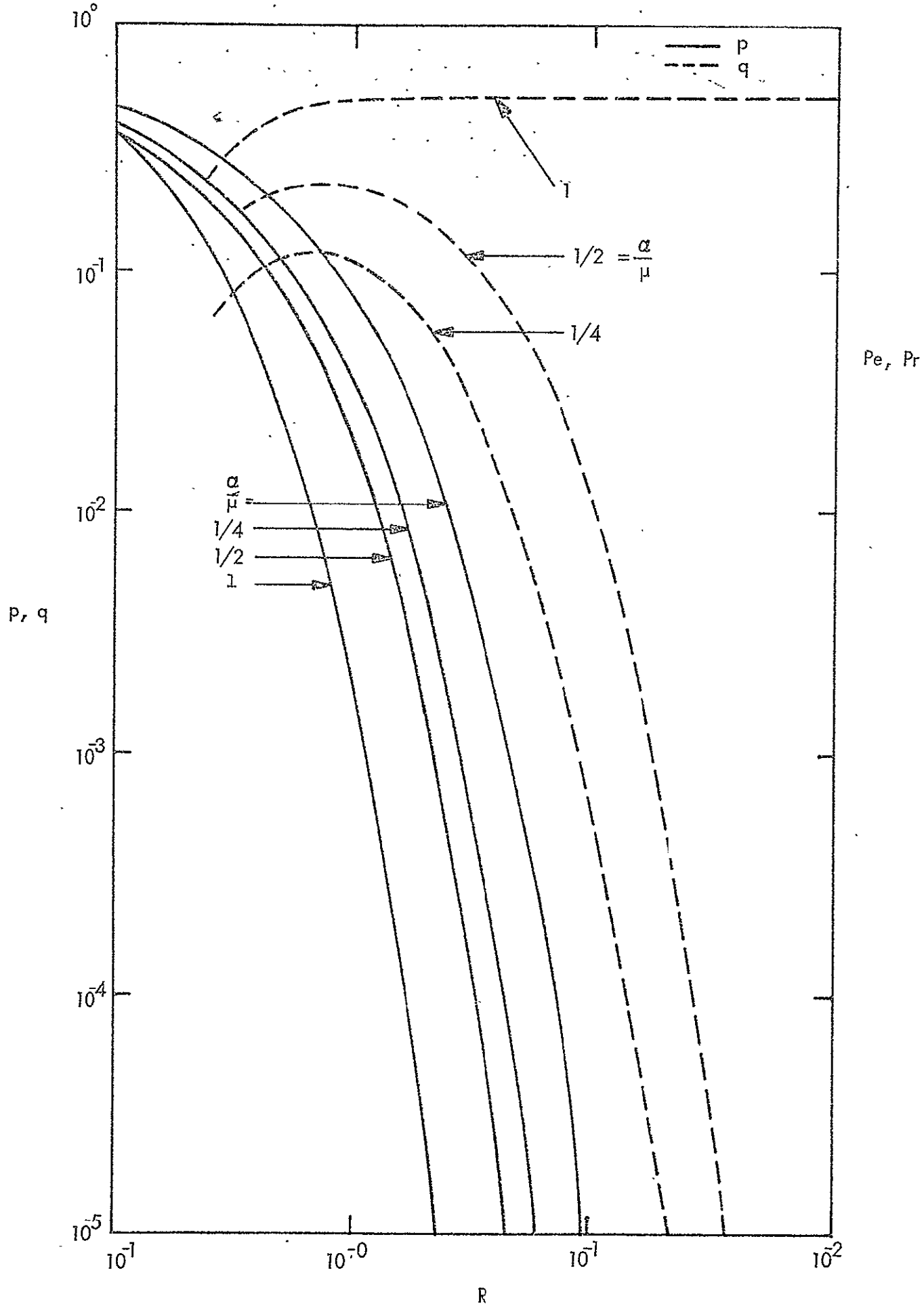
$$= \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{-\gamma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx = \tfrac{1}{2}\left[1-\text{erf}\left(\frac{\mu+\gamma}{\sqrt{2}\sigma}\right)\right] \text{ and} \qquad (1a)$$

$$q = \text{pr(bit erasure)} = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\gamma}^{\gamma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx = \tfrac{1}{2}\left\{\text{erf}\left(\frac{\mu+\gamma}{\sqrt{2}\sigma}\right)-\text{erf}\left(\frac{\mu-\gamma}{\sqrt{2}\sigma}\right)\right\} \qquad (1b)$$

Both p and q are plotted in Figure 2.1, for several values of γ/μ, as a function of $R = \mu^2/2\sigma^2 = E/N_0$ the ratio of the bit energy to the single-sided (white) noise spectral density. The signal level dependence mentioned earlier is quite apparent here. Suppose, for example γ/μ were originally equal to $\frac{1}{4}$, but that the receiver amplitude gain dropped by a factor of two. Then if γ were kept constant, γ/μ would double, and essentially half of all received bits would be erased at all signal-to-noise ratios of interest. Thus, most practical decoding algorithms would result in the rejection of nearly all received commands.

Ideally, γ/μ should be kept constant so that the fraction of erasures is not affected by a change in the signal level. This

Figure 4.1 Bit Error and Erasure Probabilities



would entail a procedure for estimating the current signal level and adjusting the threshold accordingly. While this is not difficult to do, it is in effect accomplished automatically if the receiver is equipped with an automatic gain control (AGC).

An ideal AGC receiver has the property that the output power, $\mu_o^2 + \sigma_o^2$, is kept constant regardless of the input signal and noise power levels, μ_i^2, and σ_i^2, respectively. Thus, the receiver power gain is $G = \left(\frac{c}{\mu_i^2 + \sigma_i^2}\right)$ with c a constant which, with no loss in generality, will be set equal to 1. Then if the input signal level is μ_i, the output level is

$$\mu = G^{\frac{1}{2}} \mu_i = \frac{1}{(1 + \sigma_i^2/\mu_i^2)^{\frac{1}{2}}}$$

and depends only on the signal-to-noise- ratio μ_i^2/σ_i^2, not on the signal itself. Thus, γ/μ will remain constant, independent of the input signal or noise levels so long as the signal-to-noise ratio $R = \mu_i/2\sigma_i^2$ stays fixed. The threshold will change slightly with the signal-to-noise ratio, of course, but this will have a minimal effect. The ratio γ/μ at $R = \infty$, for example, is smaller than that at R = 1 by a factor of only $(2/3)^{\frac{1}{2}}$. The effect of this change in threshold is that, as the signal-to-noise ratio decreases the erasure probability decreases slightly faster, and the error probability slightly slower, than they would otherwise. In many cases this will actually be a desirable situation, but in any event it is not very significant.

4.2.2.2 The Sub-bit Decision Method

If each bit is divided into k sub-bits of equal duration, the probability of an erroneous decision concerning any one of these sub-bits is

$r = \frac{1}{2}\left[1-\text{erf}\left(\frac{R}{k}\right)^{\frac{1}{2}}\right]$ with R as previously defined. If a bit is interpreted as an erasure unless at least $\ell > k/2$ sub-bits are in agreement, the probability of a bit error is

$$p = F(k,\ell,r) \triangleq \sum_{i=0}^{k-1} \binom{k}{i} (1-r)^i r^{k-i} \qquad (2a)$$

while the erasure probability is

$$q = 1-F(k,\ell,r)-F(k,\ell,1-r) \qquad (2b)$$

It is apparent that k would have to be moderately large if a sufficient error/erasure tradeoff is to exist. (e.g. If k = 3,

ℓ must equal either 2 or 3. In the first case erasures could never occur; in the second, they would occur with a generally unacceptably large probability.)

To assess the cost of making sub-bit decisions, consider the error probability when k is odd and $\ell = \frac{k+1}{2}$ (no erasures.) Then, when R is large

$$r \to \frac{e^{-r/k}}{2\sqrt{\pi r/k}} \qquad (3a)$$

and

$$p \to \binom{k}{\frac{k-1}{2}} r^{\frac{k+1}{2}} \to \frac{k+1}{k} \frac{1}{4} \binom{k}{\frac{k-1}{2}} \frac{e^{-(R/2)\frac{k+1}{k}}}{(4\pi R)^{\frac{k+1}{2}}} \qquad (3b)$$

Thus, even for k = 5, the error prabability behaves asymptotically as though the signal-to-noise ratio R were decreased by a factor of 3/5, and this factor is a decreasing function of k. This strongly suggests that a rather sizeable penalty would be paid, in terms of the effective signal-to-noise ratio reduction were the sub-bit decision procedure to be used. This fact, combined with the fact that the threshold method is considerably easier to implement, seems a sufficient reason for eliminating the sub-bit decision scheme from further consideration.

4.2.3 Decoding Error Probabilities and Some Bounds

Let p be the probability that a binary code symbol is received in error [i.e. p = Pr (1 received | 0 transmitted) = Pr (0 received | 1 transmitted)] and let q be the probability that a received symbol is identified as an erasure. Then, if d denotes the guaranteed minimum distance separating any two n-symbol code words, and d_o and d_1 the error and erasure correction distances, respectively, as defined in the previous section, the probability P_e of a decoding error is overbounded by

$$P_e \leq \sum_{i=0}^{d_1 - 1} \binom{n}{i} q^i (1-q)^{n-i} \left\{ 1 - \sum_{j=0}^{\left\lfloor \frac{d - d_o + i}{2} \right\rfloor} \binom{n-i}{j} r^j (1-r)^{n-i-j} \right\} \qquad (1)$$

where $r \triangleq p/(1-q)$ is the probability of an error given that the symbol in question was not received as an erasure and the square brackets denote the integer part of the enclosed quantity. This bound follows because an error is committed only

if the number i of erasures is less than d_1 and, at the same time, the number of non-erased symbol errors exceeds* $\left[d - \frac{d_o+i}{2}\right]$. An error may not result even if both of these events take place, so this is indeed an upper bound on P_e. Similarly, the probability $P_e + P_r$ of either a word error or a rejected word is

$$P_e + P_r = 1 - \sum_{i=o}^{d_1-1} \binom{n}{i} q^i (1-q)^{n-i} \sum_{j=o}^{\left[\frac{d_o-i-1}{2}\right]} \binom{n-i}{j} r^j (1-r)^{n-i-j} \qquad (2)$$

since a word is correctly received if and only if** the number i of erasures is less than d_1 and the number of errors does not exceed $\left[(d_o - i - 1)/2\right]$.

If bi-phase PSK modulation is used, as will be assumed here, p and q are as defined in equation (1) of the preceeding section For present purposes these probabilities are most conveniently expressed in the form

$$\begin{aligned} p &= \tfrac{1}{2}\left\{1 - \operatorname{erf}\left(\sqrt{R'}(1+\theta)\right)\right\} \\ q &= \tfrac{1}{2}\left\{\operatorname{erf}\left(\sqrt{R'}(1+\theta)\right) - \operatorname{erf}\left(\sqrt{R'}(1-\theta)\right)\right\} \end{aligned} \qquad (3)$$

where R denotes the ratio of the signal energy to the noise spectral density and θ the erasure threshold (with $\theta = 0$ corresponding to no erasures and $\theta = 1$ to an erasure probability of $\frac{1}{2}$.)

A computer program has been written to evaluate both Pe and Pr as a function of n, d, d_0, d_1, R'and θ. Since these probabilities are functions of so many parameters, however, it is clear that an evaluation of all possibilities would be a formidable task even for a computer. It is therefore of interest to develop bounds on both P_e and P_r which are analytically more tractable. With the aid of such bounds, the optimum values of

* If $i \geq d_o$, this number of non-erased symbol errors need only exceed d-i-1.

**The only if portion of this statement may not be true for some threshold decoding algorithms. In this case this expression provides an upper bound on $P_e + P_r$.

these various parameters can at least be estimated, thereby significantly reducing the number of cases to be investigated on the computer.

One method for bounding error probabilities which turns out to be effective here is to use the so called Chernoff bound. This bound is based on the simple observation that

$$\sum_{i=m}^{n} f_i \le \sum_{i=0}^{n} e^{s(i-m)} f_i \tag{4a}$$

for any set of non-negative summands f_i and for any m, $0 \le m \le n$, and any $s \ge 0$. Similarly, under the same conditions,

$$\sum_{i=0}^{m} f_i \le \sum_{i=0}^{m} e^{-t(i-m)} f_i \tag{4b}$$

for any $t \ge 0$.

To apply this bound to the present problem, let n_e denote the number of symbols of a code work received in error, and let n_r denote the number of erased symbols. Then

$$P_e \le \text{Prob}\left\{ 2n_e > 2d - d_o - n_r,\ n_r < d_1 \right\} \tag{5a}$$

and

$$P_e + P_r \le \text{Prob}\left\{ 2n_e \ge d_o - n_r \text{ or } n_r \ge d_1 \right\}$$

$$\le \begin{cases} \text{Prob}\left\{ 2n_e \ge d_o - n_r \right\} + \text{Prob}\left\{ n_r \ge d_1 \right\} & d_1 < d_o \\ \text{Prob}\left\{ 2n_e \ge d_o - n_r \right\} & d_1 \ge d_o \end{cases} \tag{5b}$$

Using the Chernoff bound then yields

$$P_e \le e^{t(d_1-1)} e^{-s(2d-(d_o-1))} \sum_{i=0}^{n} \sum_{j=0}^{n} e^{-tj} e^{s(2i+j)} \Pr(n_e=i, n_r=j) \tag{6}$$

But $\Pr(n_e=i, n_r=j) = \binom{n}{j} q^j (1-q)^{n-j} \binom{n-j}{i} r^i (1-r)^{n-j-i}$

with r as previously defined. Thus, it follows after some manipulation that

$$P_e \leq \min_{s\geq 0, t, \geq 0} \exp\left\{-n\left[2s\, d/n - s\left(\frac{d_o-1}{n}\right) - t\left(\frac{d_1-1}{n}\right) - \mu(s,t)\right]\right\} \qquad (7)$$

where $\mu(s,t) \triangleq \log_e (pe^{2s} + qe^{s-t} + 1-p-q)$

That P_e is bounded by the minimum over all non-negative s and t of the expression on the right is obviously true since it is overbounded by that expression for any specific non-negative s and t.

Similar manipulations establish that

$$P_e + P_r \leq P_o(d_o) + P_1(d_1, d_o) \qquad (8)$$

where

$$P_o(d_o) = \min_{s\geq 0} \exp\left\{-n\left[s\frac{d_o}{n} - \nu(s)\right]\right\}$$

with $\nu(s) = \log_e \left\{pe^{2s} + qe^{s} + 1-p-q\right\}$

and

$$P(d_1, d_o) = \begin{cases} \min\limits_{s\geq 0} \exp\left\{-n\left[s\frac{d}{n} - \eta(s)\right]\right\} & d_1 < d_o \\ 0 & d_1 \geq d_o \end{cases}$$

with

$$\eta(s) = \log_e \left[qe^{s} + 1-q\right]$$

The ultimate goal, of course, is to find the threshold θ and the error and erasure distances which, for given values of n, d, and R', minimize P_r subject to the constraint that $P_e \leq 10^{-18}$. Since the above bounds are considerably more tractable than the earlier expressions for P_e and P_r, the obvious suggestion is to find the values of θ, d_o, and d_1, which minimize the bound on P_r subject to the condition that the bound on P_e not exceed 10^{-18}. The Chernoff bounding method is known to yield very tight error probability bounds, especially when, as here, the quantities being bounded are small. Thus, the results obtained in this way should provide an excellent estimate of the true optimum values for θ, d_o, and d_1.

Several approaches are possible at this point. Since the bounds on P_e and P_r are to be minimized with respect to s, t, θ, d_o, and d_1, the minimization could be carried out in any of several orders. The generally most tractable approach seems to be first to minimize the bound on P_e with respect to s and t. Doing this yields

$$P_e \leq \exp\left\{-n\left[\left(1-\delta+\frac{\delta_o-\delta_1}{2}\right)\log_e\left\{\frac{1-\delta+\frac{\delta_o-\delta_1}{2}}{1-p-q}\right\}\right.\right.$$
$$\left.\left.+\left(\delta-\frac{\delta_o+\delta_1}{2}\right)\log_e\left\{\frac{\delta-\frac{\delta_o+\delta_1}{2}}{p}\right\}+\delta_1\log_e\frac{\delta_1}{q}\right]\right\} \qquad (9)$$

where $\delta = d/n$, $\delta_o = \frac{d_o-1}{n}$, $\delta_1 = \frac{d_1-1}{n}$, and the following inequalities are assumed satisfied:

$$\frac{\left(\delta-\frac{\delta_o+\delta_1}{2}\right)}{p} \geq \frac{1-\delta+\frac{\delta_o-\delta_1}{2}}{1-p-q} \qquad (10)$$

and

$$\frac{\delta-\frac{\delta_o+\delta_1}{2}}{p} \quad \frac{1-\delta+\frac{\delta_o-\delta_1}{2}}{1-p-q} \geq \left(\frac{\delta_1}{q}\right)^2$$

(As a consequence, the inequality

$$q \geq \frac{\delta_1}{\delta - \left(\frac{\delta_o + \delta_1}{2}\right)} p$$

must also be satisfied.)

If these inequalities are not all satisfied the bound is tightest either for $t = 0$ or for $s = 0$ (or both). Equating t to zero is equivalent to bounding P_e without taking into consideration the possibility that the number of erasures could equal or exceed the erasure correction distance d_1. In this case, the bound becomes

$$P_e \leq \min_{s \geq 0} \exp\left\{-n\left[s(2\delta/n - \frac{d_o - 1}{n}) - \nu(s)\right]\right\} \qquad (11)$$

with $\nu(s)$ as previously defined. In contrast, the bound resulting when $s = 0$ is that which would obtain if word errors were prevented only by the number of erasures equalling or exceeding d_1. This latter case is obviously not of interest. The best upper bound on P_e (eqn. 7) will clearly result either when the inequalities (10) are satisfied or when $t = 0$. A computer program has also been written to evaluate the exponent in equation (9) as a function of $\delta, \delta_o, \delta_1$, R' and θ, and to indicate the regions of these parameters for which the solution is valid. It will be noted that this approach requires considerably less computer time than does an evaluation of ther error probability expression (1) over the same range of parameters. Moreover, these results provide considerable insight into the optimum relationship between the various parameters of concern.

Even if the tightest bound on P_e does not occur when $t = 0$, the resulting bound (equation 11) is nevertheless a valid one. When $t = 0$ and $d_o \leq d_1$, the bounds of interest are

$$P_e \leq \min_{s \geq o} P\left[s, \left(\frac{2\delta - d_o + 1}{n}\right)\right] \qquad (12)$$

and $P_e + P_r \leq \min_{s \geq o} P\left(s, \frac{d_o}{n}\right)$

with $P(s,x) = \exp\left\{-n\left[sx - \nu(s)\right]\right\}$

and with $\nu(s)$ as previously defined, (cf. equns. (8) and (11).

Thus, under these conditions, both bounds are of the same form, and, infact, reduce to the form encountered for error- and erasure-correction-only decoding.* In contrast to the situation earlier, it is convenient to minimize these bounds first with respect to θ and then with respect to s. Minimizing $P(s,x)$ over θ is equivalent to minimizing $\nu(s)$, or more conveniently $e^{\nu(s)}$ over θ. But as is readily verified, $e^{\nu(s)}$ is minimized with respect to θ when $\theta = s/4R'$. Further, when both p and q are small, as they must be here, they can be closely approximated using the first two terms in the asymptotic expansion of the error function:

$$\operatorname{erf} x \tilde{=} 1 - \frac{e^{-x^2}}{x\sqrt{\pi}}$$

Using this approximation yields, * when $\theta = s/4R'$,

$$e^{\nu(s)} \tilde{=} \begin{cases} \sqrt{\frac{R}{\pi}} \frac{1}{(\frac{R-s^2}{4R'})} \exp -\left\{(\frac{R'-3s}{2} + \frac{s^2}{16R}\right\} & s > 4R'(3-2\sqrt{2}) \\ 1 & s < 4R'(3-2\sqrt{2}) \end{cases} \quad (14)$$

Accordingly, since $\nu(s)$ is constant for $s \leq 4R'(3-2\sqrt{2})$, and since its derivative with respect to s exceeds unity for $s > 4R'(3-2\sqrt{2})$, $P(s,x)$ is minimized for $x \leq 1$ when $s = 4R'(3-2\sqrt{2})$, in which case $P(s,x)$ assumes the especially convenient form

$$P(s_{opt},x) = e^{-nx\, 4R'(3-2\sqrt{2})} \cong e^{-.687k} \times R \tilde{=} P(x) \quad (15)$$

where $R \triangleq (n/k)\, R'$ is the ratio of the signal energy per information bit to the noise spectral density.

G. D. Forney, Jr., "Generalized Minimum Distance Decoding", IEEE Transactions on Information Theory, IT-12, p. 125, 1966

In summary, we have established the following convenient bounds:

$$P_e \leq P\left(\frac{2d}{n} - \frac{d_o-1}{n}\right) \qquad (16)$$

$$P_e + P_r \leq P\left(\frac{d_o}{n}\right)$$

where

$$P(x) = e^{-.687k \times R}$$

Since the two approximations used in arriving at this result, the Chernoff bound and the asymptotic expansion of the error function, are both known to be quite accurate over the range of interest here, these bounds should be quite tight provided that the effect of equating t to zero in equation (7) is not unduely restrictive. In any event, equation (16) provides good, and extremely convenient, bounds on the word error and rejection probabilities. These expressions will now be used to provide a preliminary screening of the various error-correcting codes which appear to merit further consideration.

4.2.4 Choice of codes and Their Performance

4.2.4.1 A Preliminary Screening

As mentioned earlier, two block code decoding techniques, algabraic decoding and threshold decoding, are being considered. The best known class of algebraically decodable codes are the BCH codes; the best known classes of threshold decodable codes are the projective and Euclidean geometry codes. Codes in both of these categories which have been selected for further consideration are listed in Tables 4.1 and 4.2. The selection was based on the following considerations: (1) Each code word should represent an integer multiple of 18-bit command message, or conversely, each command message should be encoded into an integral number of code words; i.e. the number k of information bits per code word should either be an integral multiple or an integral divisor of 18. (2) Codes with rates k/n approaching 1/10

impose unreasonable bandwidth requirements on the command link and should be considered only if the desired performance cannot be achieved using higher-rate codes. (3) Code word lengths n exceeding 127 would lead to overly complex decoders and should therefore also be excluded from consideration, again unless the desired performance cannot otherwise be attained.

No projective or Euclidean geometry codes having word lengths $n \leq 32$ are shown in Table 4.2, since all such codes are also BCH codes and are included in Table 4.1. If one of these codes were to be chosen for the command system, either of the two decoding algorithms might be used. The codes in these tables, incidentally are in most cases shortened versions of a longer code. That is, if there exists a code having the parameters n,k,d, there also exists a shortened code having the parameters $n-\ell$, $k-\ell$, d, for any integer ℓ. The purpose of shortening, of course, is to achieve a value of k satisfying condition (1).

TABLE 4.1 SELECTED (SHORTENED) BCH CODES

n	k	d	d_o	$-\log_{10} P_r$
3	1	3	x	x
3	2	2	x	x
6	3	3	x	x
7	3	4	x	x
13	9	3	x	x
13	3	7	x	x
14	6	5	x	x
19	9	6		x
23	18	3	x	x
24	18	4	x	x
28	18	5	x	x
29	9	11	x	x
31	6	15	x	x
32	6	16	x	x
62	6	31	x	x
62	9	27	x	x
63	18	21	12	6.9
63	36	11	7	7.9
123	18	47	35	10.5
124	54	23	23	<18
127	36	31	31	<18

TABLE 4.2 SELECTED (SHORTENED) PROJECTIVE AND EUCLIDEAN GEOMETRY CODES

n	k	d	d_o	L	$-\log_{10}Pr$
33	18	5	x	2	x
57	36	7	1	4	1.29
58	36	8	2	3	2.4
59	18	15	2	3	1.31
60	18	16	4	2	2.48
60	54	3	x	5	x
62	36	9	4	1	4.8
64	36	10	5	1	5.6
70	54	6	1	2	1.5
79	18	22	6	1	2.92
116	18	31	6	3	1.89
117	18	32	8	2	2.53
124	54	15	10	4	8.6
125	54	16	12	3	8.7

The other columns in Tables 4.1 and 4.2 indicate the maximum error correction distance d_o which still results in an error probability not exceeding 10^{-18} when R = 6.845 (so that the uncoded error probability is 10^{-4}), the base 10 logarithm of the rejection probability P_r for that value of d_o (cf. equation (16), Section 4.2.3) and a threshold decoding parameter L. An x in the d_o and $-\log_{10} P_r$ columns reflects the fact that in these cases, the acceptable error probability is exceeded regardless of the value of d_o. The parameter L denotes the number of threshold stages needed in the threshold decoder. Consequently, L is a measure of the decoder complexity, the number of logic elements being approximately exponentially proportional to L. (cf. Section 4.2.5.2)

Tables 4.1 and 4.2 immediately suggest several interesting conclusions: (1) An acceptable word-error probability may be maintained provided the code word length n is sufficiently long; possibly even for an n on the order of 60. (2) The attainability of the desired performance is amazingly independent of the ratio k/n, the code rate. (3) The rejection probability P_r actually decreases as the code rate k/n increases, provided the rate is not so high as to preclude the attainment of the required error rate. In short, acceptable performance seems to be relatively easily attained (under the conditions postulated here) provided the code word length is on the order of 60 or greater.

The question as to whether a block length of 60 is actually sufficient to achieve the desired end still remains unresolved, however. This is because the above conclusions were obtained using only approximations to the error and rejection probabilities.

Nevertheless, these results do provide a guide as to the more promising code parameters for further investigation. The results of this more detailed investigation follows.

4.2.4.2 Exact Results

Computer programs were written for evaluating the exact expressions for the probability of a word error and a word rejection (equations (1) and (2) Section 4.2.3) and for evaluating bounds on these values (equations (9) and (11) and (12) of the same section).
Selected portions of the print-out of these programs are included as an appendix to this report and the results of greatest interest are summarized in Table 4.3.

On the basis of these results one comes to the following conclusions:

(1) Although it is possible to achieve the goal of a 10^{-18} word error probability when the signal-to-noise ratio is such that the uncoded bit error probability p is 10^{-3}, the length n of a code word needed for this purpose must be on the order of 127 or greater. However, when $p = 10^{-3}$ and $n \approx 127$, the word rejection rate at the desired 10^{-18} word error probability will be unacceptably large regardless of the code and decision criterion used. Thus, satisfactory results are attainable when the uncoded bit error probability is 10^{-3} only when codes of a word length $n \approx 255$ are used, a word length which would probably entail unacceptably complex decoders.

(2) When the signal-to-noise ratio is such that $p = 1.6 \times 10^{-4}$ codes with word length $n \approx 63$ are sufficient to achieve a 10^{-18}

Table 4.3

SELECTED ERROR PROBABILITIES AND BOUNDS

($p = 1.6 \times 10^{-4}$)

n	k	d	$d_o=d_1$	Θ	$-\log_{10}P_e$	bound 1	bound 2	$-\log_{10}P_r$
63	18	21	5	1/8	17.66	15.91	15.30	0.46
63	36	11	3	1/8	17.73	15.94	14.98	1.46
				1/6	19.02	16.27	15.92	1.30
			5	1/4	17.48	13.09	14.04	1.97
64	36	10	3	1/4	18.73	12.67	15.09	0.80
127	36	31	11	1/8	17.72	16.17	15.60	0.71
			15	1/4	17.24	13.56	14.47	0.92
124	54	23	14	1/8	17.09	15.91	-	4.56
				1/6	17.39	16.03	-	4.32
				1/4	18.08	14.74	15.11	3.28

word error probability but again only when the word rejection probability is relatively large. (The (63,36) codes with an error-correcting distance $d_o = 5$ and erasure threshold $\theta = \frac{1}{4}$ achieves an error probability of nearly 10^{-18} with a word rejection probability of approximately 10^{-2}. Thus, only about one command in 100 would have to be retransmitted, a generally tolerable situation.)

(3) In general the codes having a high information ratio k/n outperform those having lower rates under the conditions postulated here. Thus, several commands (2 or 3) will have to be coded as one code word for best performance. The (127,54) BCH code, for example, achieves the 10^{-18} error probability with rejection probability of 10^{-3} or less.

(4) The word error probability bounds of equations (11) and (9) section 4.2.3 (bounds 1 and 2, respectively in Table 4.3) are sufficiently tight to be used for preliminary screening purposes. Although bound 2 is sometimes better than bound 1, the difference is never dramatic, and the relative simplicity of the latter bound, particularly when approximated as in equation (15), makes it generally preferable for this purpose. (The blanks in the bound (2) column of Table 4.3 correspond to cases in which the conditions of equation (10) are violated and hence in which the bound is invalid.)

It will be noted that the approximate results listed in tables 4.1 and 4.2 of the preceeding section are slightly optimistic.

4.2.5 Algebraic (BCH) and Threshold Decoding Algorithms

This section outlines the algebraic and threshold decoding algorithms in sufficient detail to allow an accurate estimate of the complexity of the required decoders. This information combined with the results of the previous section will then provide the desired performance versus complexity information needed to arrive at the most efficient solution to the problem at hand.

4.2.5.1 The BCH Decoding Algorithm and Its Implementation

The decoding algorithm presented here is the most efficient known for BCH codes. It is described in greater detail in the recent book by Berlekamp.*

Let $c_o, c_1, \ldots, c_{n-1}$ represent the transmitted binary BCH code word ($c_i = 0$ or 1 all i) and define the polynominal

$$c(Z) = \sum_{i=0}^{n-1} c_i Z^i$$

Then, by construction, there exists some Galois field element α such that $c(\alpha^j) = 0$ for all $j = 1, 2, \ldots, d-1$ (n, k and d are as previously defined). Further, let the received word be represented in the same way by the polynominal

$$r'(Z) = \sum_{i=0}^{n-1} r_i Z^i = \sum_{i=0}^{n-1} c_i Z^i + \sum_{i=0}^{n-1} e_i Z^i$$

where $e(Z) \triangleq \sum_{i=0}^{n-1} e_i Z^i$

represents the error vector. Then

$$r(\alpha^j) = 0 + \sum_{i=0}^{n-1} e_i \alpha^{ij} \triangleq \sum_{k=1}^{\frac{d-1}{2}} X_k^j \triangleq S_j$$

where $X_k \triangleq e_{i_k} \alpha^{i_k}$, with i_k the k^{th} integer i such that $e_i \neq 0$.

* E. Berlkamp, "Algebraic Coding Theory", McGraw-Hill, New York, N.Y., 1968

Finally, define

$$\sigma(Z) = \prod_{i=1}^{\frac{d-1}{2}} (1 + X_i Z) = \sum_{i=0}^{\frac{d-1}{2}} \sigma_i Z^i$$

where the σ_i are the so-called elementary symmetric functions:

$$\sigma_0 = 1$$

$$\sigma_1 = \sum X_i$$

$$\sigma_2 = \sum_{i>j} X_i X_j$$

$$\sigma_3 = \sum_{i>j>k} X_i X_j X_k$$

etc.

Then the error-correcting problem can be identified with the problem of finding the reciprocal roots X_i of the polynominal $\sigma(Z)$. (If $\sigma(\alpha^{-\ell}) = 0$, $X_i = \alpha^{\ell}$ for some i and $e_\ell = 1$ indicating an error in the ℓ^{th} received bit). This is the basis for the decoding algorithm described here.

The algorithm can be divided into three distinct modes:

(1) Determine the quantities $S_j = r(\alpha^j)$, $j = 1, 2, \ldots . d-1$.

(2) Evaluate the polynominal $\sigma(Z)$

(3) Find the reciprocal roots of $\sigma(Z)$.

The logical operations to be performed in each of these modes is discussed below.

Mode 1 Determine $S_j = r(\alpha^j)$

Step 1.1 The received bits are read serially into each of ℓ "R" registers of length $m = \log_2 (n + 1)$ and into one storage register of length n. (See Figure 4.1. If the code is a shortened or non-primitive code m is the smallest integer such that all the generator polynominal roots are in $GF(q^m)$.) The i^{th} R-register R_i, $i = 1, 2, d-1$, is wired so as to divide the received polynominal by $g_i(x)$, the minimal polynominal having the root α^i. Since α^i, α^{2i}, α^{4i}, α^{8i}... are all roots of the same minimal polynominal, ℓ, the number of distinct R - registers will generally be less than d-1. (With the (63, 18 BCH code, for example $\ell = 8, d - 1 = 20$.). The contents of R_i after the complete code word has been received is

$$r_i(x) = \sum_{j=0}^{m-1} r_j^i x^j$$

where $r(x) = g_i(x)\, q(x) + r_i(x)$ represents the received word. (i.e., the contents of the m storage cells of R^i are r^i_{m-1}, r^i_{m-2},..., r^i_0, respectively.)

Step 1.2 The contents of R_i are read into the i^{th}, $2i^{th}$, $4i^{th}$, $8i^{th}$ ....., m-stage S_a registers. The j^{th} S_a - register S_{aj} is so wired that its contents are multiplied by the field element α whenever they are shifted one position to the right. It is cycled j-1 times between each serial read-in from R_i. Thus, upon receiving its first input r^i_{m-1}, it contains r^i_{m-1}; after j-1 shifts it contains $r^i_{m-1}\alpha^{j-1}$; after the second input it contains $r^i_{m-1}\alpha^j + r^i_{m-2}$; after j-1 more shifts it contains $r^i_{m-1}\alpha^{2j-1} + r^i_{m-2}\alpha^{j-1}$; etc. After exactly $(m-1)j + 1$ clock cycles, therefore, S_{aj} contains a binary representation of the element $r_i(\alpha^j)$ where $j = \nu i$ for some integer $\nu \geq 1$. This completes mode 1.

Mode 2 Evaluate $\sigma(Z)$

Step 2.1 Parallel shift the contents of each S_a - register into the corresponding m-stage S_b-register , and from the σ_a

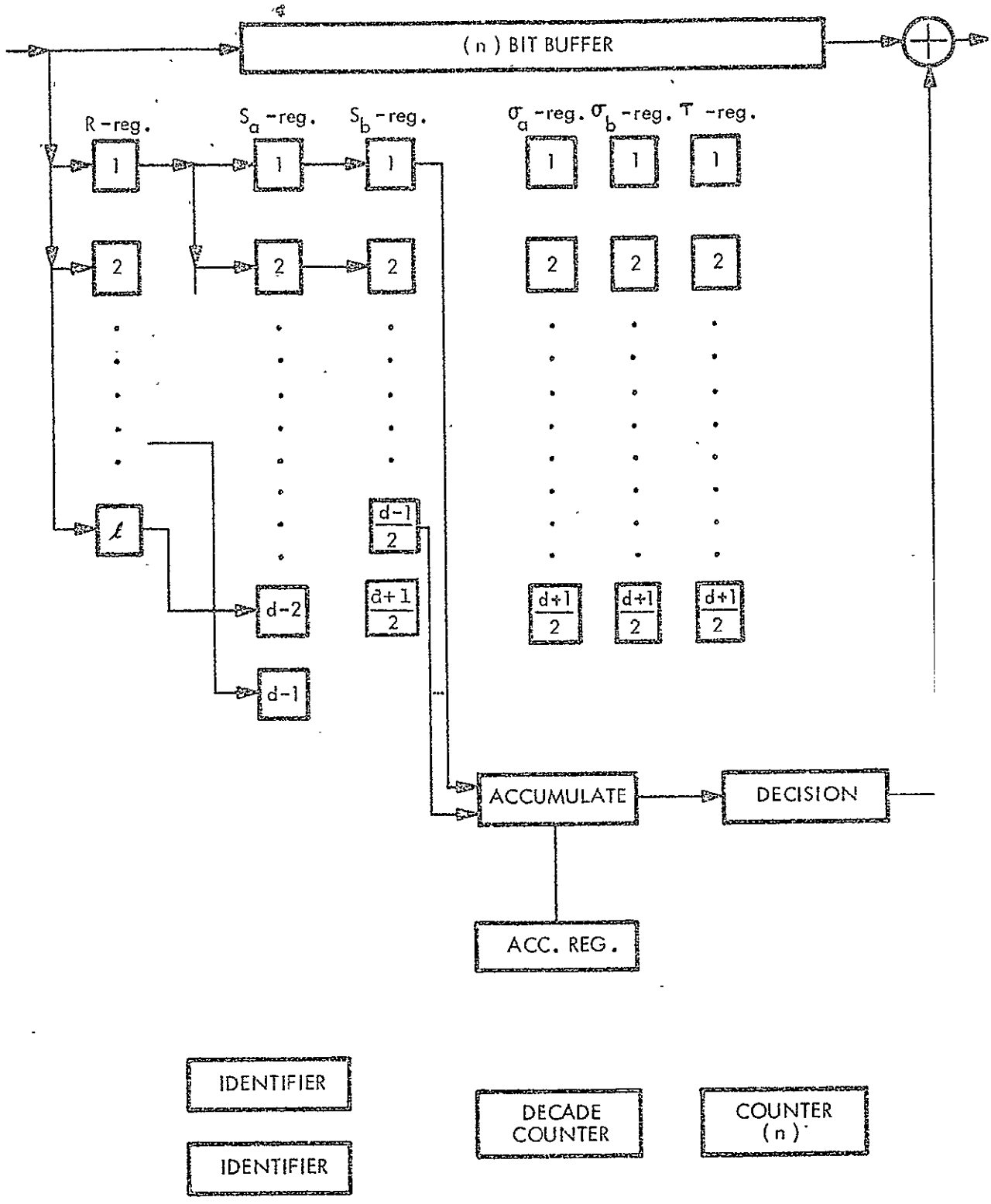


FIGURE 4.2a BLOCK DIAGRAM OF BCH DECODER

- register to the corresponding σ_b register. (Initially the $i^{\underline{th}}$ σ_a register σ_{a_i} is set to 1, 0, 0,..., $0 \triangleq 1$ for $i = 1$ and to 0, 0,..., $0 \triangleq \emptyset$ for $i \neq 1$.) There are, in general, $\left[\frac{d+1}{2}\right]$ each of the S_b, σ_a, and σ_b registers, each register having m storage cells.

Step 2.2 Cycle each S_b register and the corresponding σ_b register simultaneously, until each σ_b register contains 1. If a σ_b registe initially contains $\emptyset$, set the corresponding S_b register equal to $\emptyset$, without cycling either register. The S_b - registers are wired to count forward, and the σ_b - registers to count backward, in the field $GF(2^m)$. The result of this step, therefore, is to multiply the contents of S_b by the contents of the corresponding σ_b.

Step 2.3 Add the contents of the S_b registers (on a term-by-term modulo-two basis) and store in the accumulator. This is equivalent to forming the sum $\sum_i S_{b_i}$ over GF (2^m).

Step 2.4 Parallel shift from the $i^{\underline{th}}$ to the $(i+1)^{\underline{st}}$ τ register. Initially the $i^{\underline{th}}$ τ register is set to 1 for $i = 1$ and $\emptyset$ for $i > 1$. The element $\emptyset$ is read into the $1^{\underline{st}}$ τ register at the time of the shift. Also parallel shift the contents of each τ - register to the corresponding S_b register. Finally, shift the contents of the accumulator to the accumulator register.

Step 2.5 Cycle each S_b register and the accumulator register until the latter contains the element 1. If the accumulator register initially contains $\emptyset$, do not cycle but rather read $\emptyset$ into all the S_b registers. Since each S_b register counts up and the accumulator register counts down in GF (2^m), this step results in the formation of the products of the contents each of the S_b registers with the contents of the accumulator register.

Step 2.6 Add (term-by-term, modulo-two) the contents of each σ_a register to the contents of the corresponding S_b register and store the result in the S_b register.

Step 2.7 If the accumulator does not contain the element β and if, at the k^{th} iteration of this mode 2 sequence, the $k + 1^{st}$ σ_a register does contain β, proceed to step 2.9. If either of these conditions is violated, go to step 2.8.

Step 2.8 Shift the contents of the i^{th} to the $(i + 1)^{st}$ τ - register as in step 2.4. Then go directly to step 2.12.

Step 2.9 Parallel shift the contents of the i^{th} to the $(i + 1)^{st}$ σ_a register, and shift β into the 1^{st} σ_a register. Then parallel shift the contents of each register to the corresponding σ_b register. Finally, shift the contents of the accumulator to the accumulator register.

Step 2.10 Cycle the contents of each σ_b register and the accumulator register simultaneously until the latter contains the element 1. Since the accumulator register and the σ_b registers both count down in $GF(2^m)$ the contents of the latter registers are divided by the contents of the accumulator in this step.

Step 2.11 Parallel shift the contents of each σ_b - register into the corresponding τ register.

Step 2.12 Parallel shift the contents of each S_b - register to the corresponding σ_a register.

Step 2.13 Parallel shift from the $(i + 2)^{nd}$ to the i^{th} S_a register, all i. If this complete sequence (from step 2.1 to 2.13) has been completed $\left[\frac{d - 1}{2}\right]$ times, procede to step 2.14. Otherwise return to step 2.1.

Step 2.14 Shift the contents of each S_b register into the corresponding S_a register and procede to Mode 3.

This completes Mode 2. For proof that this procedure actually does result in the formulation of the polynomial $\sigma(Z)$, the coefficients σ_i, $i = 1, 2, \ldots, \left[\frac{d-1}{2}\right]$, of which are now stored in the first $\left[\frac{d-1}{2}\right]$ S_a - registers, see Berlekamp.* Since each multiplication or division over GF (2^m) requires at most $2^m - 1$ clock cycles, since each iteration requires a total of at most 3 multiplications and divisions, since there are $\left[\frac{d-1}{2}\right]$ iterations, and since the operations other than multiplication and division are relatively insignificant so far as their time consumption is concerned, the total time required in Mode 2 is proportional to approximately $3d\ 2^{m-1}$.

Mode 3 Find the reciprocal roots of $\sigma(Z)$.

Step 3.0 Skip to step 3.2.

Step 3.1 Cycle the j^{th} S_a register j steps. (Note: only the 1^{st} $\left[\frac{d-1}{2}\right]$ S_a registers are used here.)

Step 3.2 Add, (term-by-term , modulo-two) the contents of each of these registers and store in the accumulator.

Step 3.3 If this sum is 1 (= 1,0,0,...,0) after the i^{th} repetition of this step) add 1 (modulo-two) to the $(n - i + 1)^{st}$ received bit and read it cyclically back into the message storage register.

Step 3.4 Return to step 3.1. Stop after n iterations. The 1^{st} k bits in the message storage register, reading from right to left, constitute the decoded message.

This rationale here is easily explained. The i^{th} received bit is in error if and only if $\sigma(\alpha^{-i}) = 1 + \Sigma\ \sigma_j\ ^{-ij} = 1$. Cycling an S_a - register once is equivalent to multiplying the contents of the register by α. Thus, after the ℓ^{th} iteration, the content of the accumulator is $\Sigma_j\ \sigma_j\ \alpha^{j} = \Sigma_j\ \sigma_j\ \alpha^{-(n-\ell)j}$ as desired. The time required to complete this mode is approximately $n\ \frac{d-1}{2}$ clock cycles.

* ibid.

To detect as well as correct errors, it is only necessary to count the number of times the accumulator reads 1 in mode 3. If this number, (which equals the number of detected errors) exceeds $\frac{d_o-1}{2}$ where d_o is the error-correcting distance the message is rejected.

Erasures are detected, of course, simply by counting the number of times during the message reception that the detector output does not exceed the threshold. If this number r exceeds $d_1 - 1$ the message is rejected. If $r \leq d_1 - 1$ erasures as well as the errors will be corrected so long as the number of errors $e \leq e(d_o,r) = \left[\frac{d_o-1}{2} - r/2\right]$ and detected so long as $e(d_o,r) + 1 \leq e \leq d - 1 - r - e(d_o,r)$.

This can be accomplished with virtually no increase in decoder complexity simply by decoding the received message (at most) twice, the first time with the r erasures all replaced by ones, the second, by zeros. That this technique works follows from the observation that the number of errors under one of the two situations will be r/2 or less. Thus, if $e + r/2 \leq (d_o-1)/2$, the resulting errors including the erasure substitutions can be corrected. Similarly, if $e > e(d_o,r)$ and if $e + r \leq d-1-e(d_o,r)$, at least $e(d_o,r) + 1$ errors will be detected.

The decoding algorithm outlined above tacitly assumed that up to $\left[(d-1)/2\right]$ errors were to be corrected. When only $\left[(d_o-1)/2\right]$ errors are to be corrected, the number of S_b, σ_a, σ_b, and τ registers can all be reduced to $\left[(d_o + 1)/2\right]$. To detect whether or not more than d_o errors actually occurred, it is only necessary to read the corrected word back into the R registers and to see whether they all read Ø at the conclusion. If this situation does not obtain, the decoded word must be rejected. This procedure can be accomplished simply by recycling the corrected word back through the decoder input as it is being corrected. If the decoded word is error-free, it can then be retrieved from the decoder n-bit shift-register.

4.2.5.2 Majority Logic Decoding

A general threshold decoder for an L-step orthogonalizable code is shown in Figure 4.2b. The boxes labeled d/2 represent (d-1) input threshold devices; their outputs are zero if $\frac{d-1}{2}$ or fewer of their inputs are zero and one otherwise. The L-1 adding circuits determine the modolu-two sum of various subsets of their inputs; these sums then constitute the inputs to the succeeding threshold devices. Since each threshold device requires d-1 disjoint inputs which are themselves outputs, or combination of outputs, from the threshold devices at the preceeding step, it is evident that the number of threshold devices is an exponentially increasing function of L. Specifically, the total number N_T of such devices is

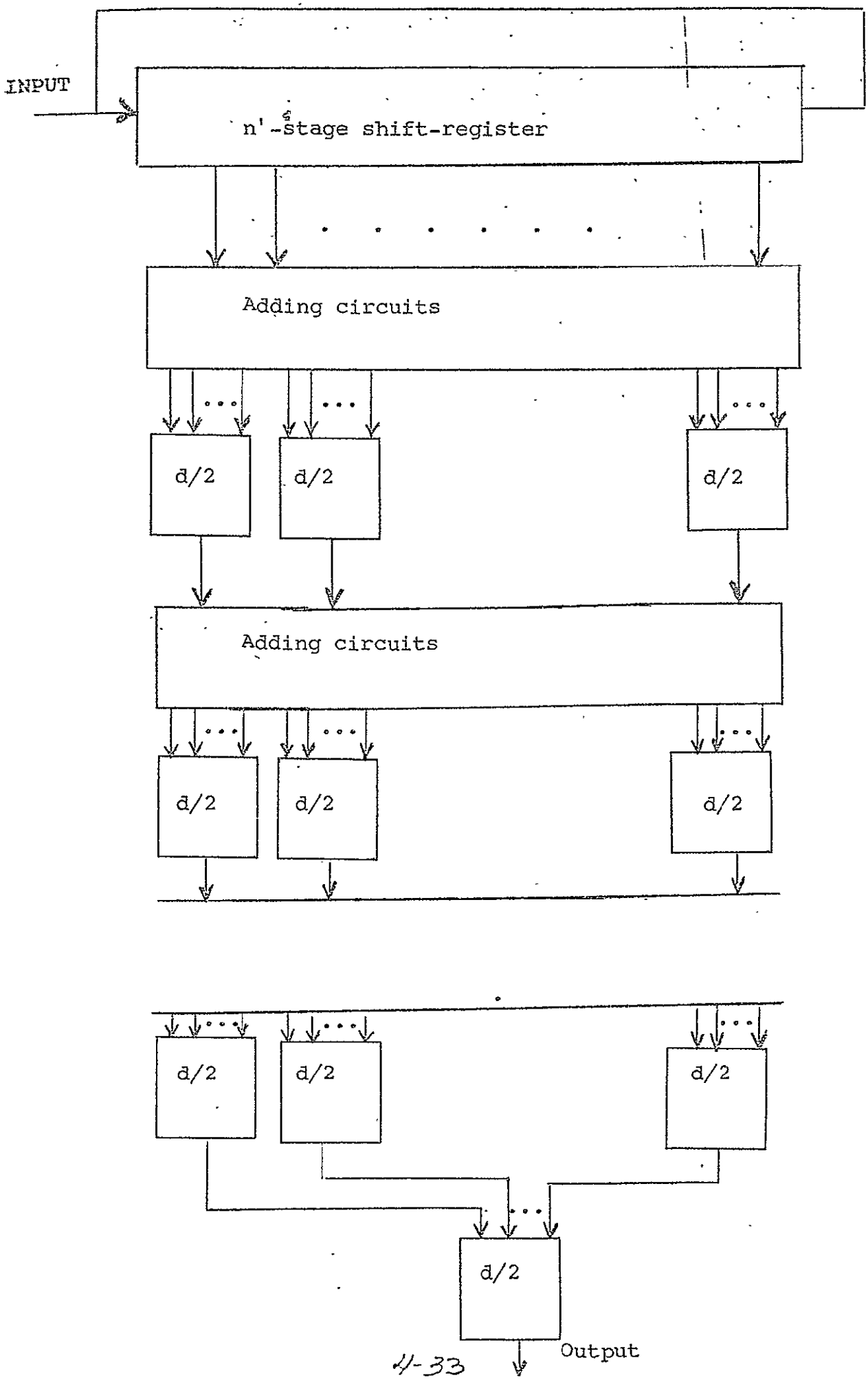
$$N_T = \sum_{i=0}^{L-1} (d-1)^i$$

This fact obviously limits consideration to threshold decodable codes which can be orthogonalized in a small number of steps.

Decoding is accomplished by reading the received word into the n'-stage shift register, allowing sufficient time for the threshold decisions to filter down through the various stages, and observing the output from the final threshold device. This output is the first information bit. The contents of the shift-register are then cyclically shifted one bit position to the right and the process repeated thereby producing the next information bit. After k shifts the word has been decoded (i.e. all k information bits have been determined).

The length of the shift register is indicated as n' since, if the code is shortened cyclic code, the shift register length will equal the word length of the original unshortened code and hence will exceed the actual word length. A modified version of this decoding algorithm, incidentally would replace this n'-stage shift register by two shift registers, one of length n-k and the second of length k. The (n-k)-stage shift register would be wired so as to divide the received code word by the code generator polynomial thereby producing its syndrome. The outputs of the last threshold device in this case

FIGURE 4.2b THRESHOLD DECODER



is not the successive information bits, but rather the error bits; (i.e. the output is a one if the corresponding information bit is in error and a zero otherwise. These error bits are then added modulo-two to the information bits stored in the k-stage shift register. Which of these two methods results in the simplest implementation depends on the particular code being used.

Conceptually, the threshold decoder approach is relatively simple. It must be cautioned, however, that threshold devices and the summing circuits can be rather complex, so the overall decoder can be more complex than it at first appears. If a code is one-step orthogonalizable, the resulting decoder will generally be less complex than any algebraic decoder. When the number of steps needed to orthogonalize the code is greater than one, however, the relative complexities tend to reverse. The relative merits of the two decoding schemes can therefore be assessed only after specific codes are selected.

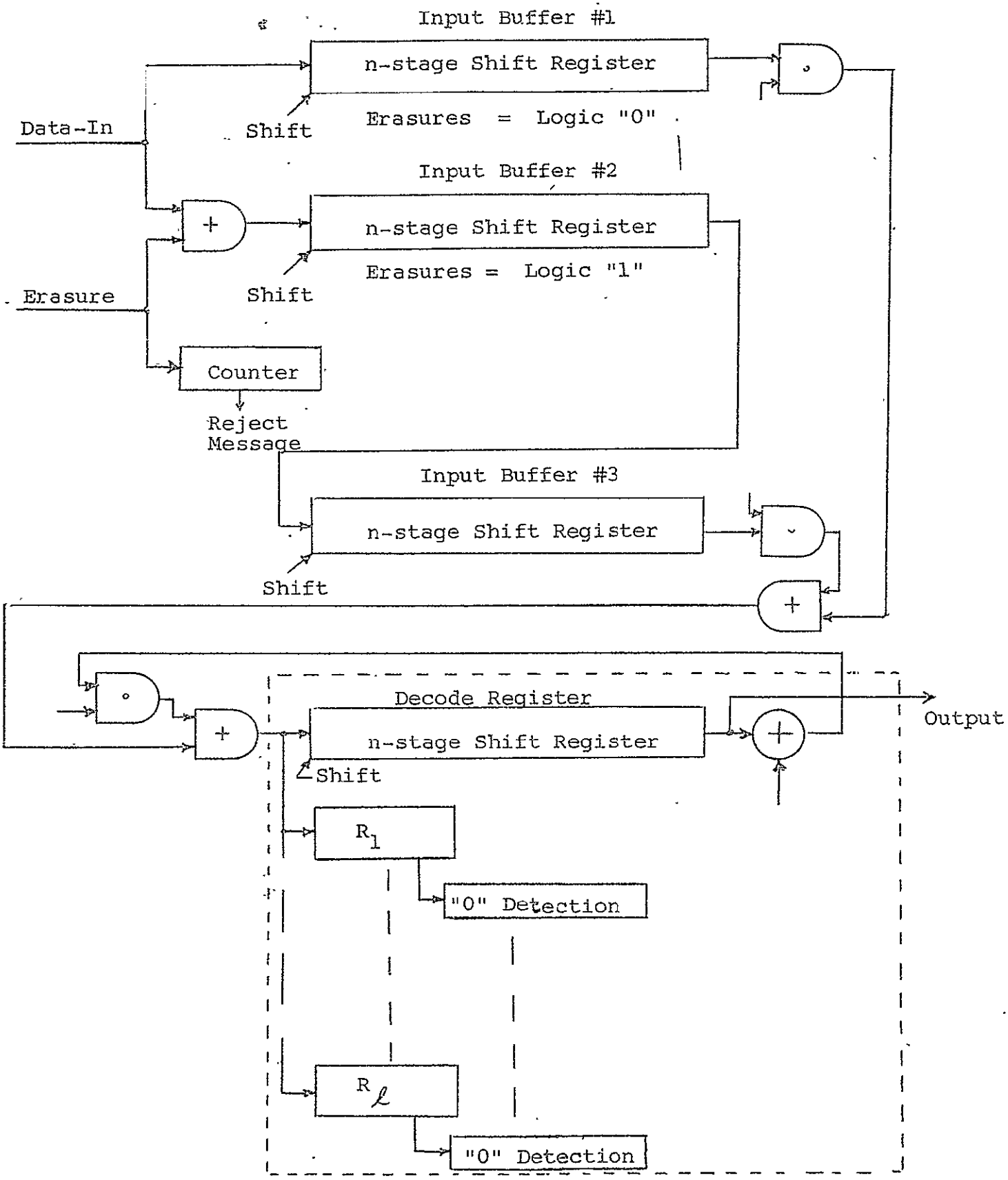
4.2.5.3 BCH Decoder Implementation and Package Count

Three n-stage input buffer shift registers (ref. Figure 4.2c) are required to implement the BCH decoder, in addition to logic functions shown in Figure 4.2a.

The first n-bit data word received is shifted into both input buffers #1 and #2. An erasure bit is stored in buffer #1 as a logic "0", while the same bit is stored in buffer #2 as a logic "1". The number of erasure indications are counted and the message is rejected when the count is $\geq d_o$.

FIGURE 4.2C

BCH DECODER INPUT BUFFER LOGIC



When the complete n-bit input word has been received, input buffers 1 and 2 must be emptied before the first bit of the following word is received. Input buffer 1 is shifted into the decode register, while input buffer #2 is shifted into input buffer #3.

The decode operations described in Section 4.2.5.1 are now performed on the contents of the decode register. If the received word with erasures represented by logic "0" is rejected during decode because of excessive errors, the contents of input buffer #3 is shifted into the decode register and the decode operation repeated.

The number of dual-in-line packages (D.I.P.'s) required to implement the BCH decoding algorithm discussed earlier has been estimated. These results are summarized below.

No package count was made for threshold decoders since apparently no satisfactory L = 1 codes exist for any block length $n < 123$ (cf. Tables 4.2 and 4.3). A preliminary investigation revealed that L = 2 and L = 3 threshold decoders are at least as complex as the analogous algebraic decoders.

TABLE 4.4

PACKAGE COUNTS FOR BCH DECODER CONFIGURATIONS

DECODER CONFIGURATION	PACKAGE COUNT (DIP's)*
BCH #1, (n = 63, k = 36)	136
BCH #2, (n = 127, k = 54)	243

*Approximately one-fourth of the packages are 24 pin DIP's which have nearly twice the length and width compared with the fourteen and sixteen-pin DIP's.

4.3 CONCATENATED CODES

4.3.1. The Concatenated Code Concept

Since it does not appear possible to achieve satisfactory results without going to relatively long codes an alternative algebraic coding schemes was explored. The main advantage of this scheme, called concatenated coding, is that the effective code length can be increased without a corresponding increase in the decoder complexity. Briefly, the scheme involves two (or more) stages of encoding; the first, or outer, code is (generally) a cyclic (n_2,k_2) code defined over some field $GF(2^{k_1})$ while the second, or inner, code is a binary (n_1,k_1) code. The data stream is first grouped into blocks of k_1 bits each, with each block defining a symbol in $GF(2^{k_1})$. These symbols are then grouped into blocks of k_2 2^{k_1}-ary symbols and encoded using the (n_2,k_2) code over $GF(2^{k_1})$. Finally, each of the symbols in the resulting code word is encoded using an (n_1,k_1) binary code. The composite code thus uses n_1n_2 code bits to represent k_1k_2 data bits. The advantage in doing this is that the decoding operation is separated into two stages, the first decoder decoding the (n_1,k_1) inner code and the second the (n_2,k_2) outer code. The two decoders will usually be significantly less complex than a single decoder operating on a code having word length n_1n_2. Moreover, if the codes are properly selected, the resulting performance can be comperable to that using an (n_1n_2,k_1k_2) code.

In order for the concatenated scheme to be effective, the inner code should be such that that probability that one of its words is erroneously decoded as some other word is (approximately) independent of the two words involved. (i.e. All code words should be roughly equal distance from all other code words.) When this is the case, the outer code can be evaluated in terms of the minimum Hamming distance separating any two of its code words.

After some preliminary screening, two codes, the (7,4) Hamming code having minimum distanced = 3 and the (8,4) biorthogonal code having minimum distance d = 4, were selected as possible inner codes, and the Reed-Solomon (15,5), d = 11, and (15,9), d = 7 codes were chosen for the outer codes. In order to minimize the resulting decoder complexity it was stipulated that the outer code decoder should not attempt to correct errors; rather it could only detect errors ($d_o = 1$) although it could both correct and detect erasures ($d_1 \geq 1$). The erasures, of course correspond to those inner code words rejected by the inner decoder. These erasures are induced as before by imposing thresholds on both the acceptable number of erasures and on the acceptable number of errors.

Of the four possible combinations, only the (8,4) combined with the (15,9) achieved the desired 10^{-18} word error probability P_e (with $p = 1.6 \times 10^{-4}$) under the imposed conditions (See the appendix to this report). The word rejection probability P_r in this case was 10^{-2} (more precisely, $P_e = 1.41 \times 10^{-19}$, $P_r = 1.006 \times 10^{-2}$). This was accomplished by setting the inner code erasure threshold θ to $\frac{1}{4}$, and both its error and erasure

correcting distances to 3. The outer code then is required to correct up to 2 erasures but to reject any word found to contain any errors or more than 2 erasures.

This coding scheme lends itself to particularly simple implementation. Several highly efficient techniques are known for decoding the (8,4) inner code, which, because it has such a short word length, is not difficult to decode in any event. In addition, since the (15,9) outer code is required only to detect errors and to correct only up to two erasures, its decoder can also be quite simple (cf. Section 4.3.3).

4.3.2 Detailed Error and Rejection Probability Estimates

4.3.2.1 Word Error and Rejection Probabilities for Concatenated Codes vs. Signal-to-Noise Ratio and Erasure Probability

The word error and rejection probabilities associated with the concatenated coding approach discussed in preceeding paragraphs (i.e. the concatenation of (15,9) Reed-Solomon code with an (8,4) biorthogonal code) are here examined in more detail. Table 4.5 shows the word error and rejection probabilities as a function of the ratio R of the received singal energy per information bit to the noise spectral density, and of the normalized erasure threshold $\theta = \frac{\gamma}{\mu}$, γ representing the actual threshold and μ the received signal level.

Note that as $R \rightarrow 0$, the (unnormalized) erasure threshold γ also approaches zero for fixed θ. If the receiver is equipped with an automatic gain control (AGC) as it presumable will be (see Section 4.2.2), the output signal level will actually be a function of the signal-to-noise ratio:

$$\mu = \frac{1}{(1 + \gamma/2)\,1/2} \qquad (1)$$

with λ the product of the AGC bandwidth and the bit period. Thus, θ will actually vary with R

$$\Theta = \gamma(1 + \gamma_{2R})^{1/2} \qquad (2)$$

Table 4.5 Concatenated Code Error and Rejection Probabilities

(θ fixed)

R	P_e	P_r
	FOR THETA =	2.000D-01
0.	7.7875742D-01	2.2124258D-01
8.45D-01	6.1926653D-03	9.9379963D-01
1.28D+00	1.7598963D-03	9.9775515D-01
2.65D+00	2.0546574D-06	8.5403840D-01
3.38D+00	1.6487636D-08	5.6736949D-01
4.81D+00	4.6740662D-13	1.1815874D-01
5.44D+00	3.2439517D-15	4.6230353D-02
6.84D+00	4.6396723D-20	4.5970328D-03
8.41D+00	1.4913632D-25	3.0602277D-04
	FOR THETA =	2.300D-01
0.	7.7875742D-01	2.2124258D-01
8.45D-01	4.5219465D-03	9.9547225D-01
1.28D+00	1.2455879D-03	9.9836821D-01
2.65D+00	1.3286621D-06	8.6770536D-01
3.38D+00	1.0043028D-08	5.9099132D-01
4.81D+00	2.5078926D-13	1.2995789D-01
5.44D+00	1.6409855D-15	5.1909816D-02
6.84D+00	2.0645064D-20	5.3086125D-03
8.41D+00	5.7931318D-26	3.4841177D-04
	FOR THETA =	2.500D-01
0.	7.7875742D-01	2.2124258D-01
8.45D-01	3.6638971D-03	9.9633128D-01
1.28D+00	9.8630019D-04	9.9868133D-01
2.65D+00	9.8821298D-07	8.7682300D-01
3.38D+00	7.2053755D-09	6.0841383D-01
4.81D+00	1.6804035D-13	1.4037375D-01
5.44D+00	1.0681519D-15	5.7329763D-02
6.84D+00	1.2682943D-20	6.1230523D-03
8.41D+00	3.3935247D-26	4.1335031D-04
	FOR THETA =	2.700D-01
0.	7.7875742D-01	2.2124258D-01
8.45D-01	2.9614021D-03	9.9703457D-01
1.28D+00	7.7712792D-04	9.9893692D-01
2.65D+00	7.2854216D-07	8.8606358D-01
3.38D+00	5.1344908D-09	6.2751917D-01
4.81D+00	1.1300139D-13	1.5349083D-01
5.44D+00	7.0252366D-16	6.4565215D-02
6.84D+00	8.0144836D-21	7.3574869D-03
8.41D+00	2.0556651D-26	5.2863697D-04
	FOR THETA =	3.000D-01
0.	7.7875742D-01	2.2124258D-01
8.45D-01	2.1354991D-03	9.9786142D-01
1.28D+00	5.3619576D-04	9.9923647D-01
2.65D+00	4.5031939D-07	9.0023565D-01
3.38D+00	3.0221471D-09	6.5971302D-01
4.81D+00	6.1892570D-14	1.7965303D-01
5.44D+00	3.7588927D-16	8.0083831D-02
6.84D+00	4.1425257D-21	1.0460263D-02
8.41D+00	1.0683486D-26	8.7767565D-04

and the erasure probability will increase more rapidly with decreasing R than it would were θ fixed. This effect is shown in Table 4.6a where the word error and rejection probabilities are again tabulated but this time as a function of γ with θ as given by equation (2) (and with $\lambda = 1/25$).

Table 4.6a thus takes into account the effect of the AGC on the error probability. But these results too must be qualified when the signal or noise levels change suddenly. This is because the AGC time constant will presumable be long compared to a word time. (Were this not true, the threshold would in fact vary during a word period). Consequently. if initially R = 6.845 (so that the uncoded error probability is 10^{-4}) and if the signal amplitude suddenly drops, the receiver gain will temporarily remain constant and θ will temporarily change

$$\text{from } \theta = \gamma \quad \text{to} \quad \theta = \gamma \left[\frac{6.845}{R'}\right]^{1/2} \tag{3}$$

This new threshold will gradually decrease back to the θ defined in equation (2) due to the action of the AGC. In the interim, however, the threshold will be essentially that defined in equation (3). The resulting error probability is tabulated in Table 4.6b, as a function of R = R' and γ.

It is seen upon comparing Tables 4.6a and 4.6b, that a sudden change in signal level is actually less deleterious then a gradual change. The possibility therefore suggests itself of forcing the threshold θ to behave as described in equation (3) even when the signal change is gradual. This scheme would probably be too difficult to implement, however, to make it practical. A more reasonable, and conventional, approach would be to place a threshold on the signal-to-noise ratio, and to cease any

attempt to decode a message received at a sub-threshold signal-to-noise ratio. A convenient device for estimating the signal-to-noise ratio is the AGC itself. Regardless of how this estimate is made, however, there will inevitably be a short time delay between any sudden signal level change and the reliable determination of this event. The results tabulated in Table 4.6b fortunately apply to precisely this situation. The few words received between the time the signal drops and the time the decoder is instructed to stop decoding, therefore, are much less likely to be erroneously decoded than would be expected from Tables 4.5 and 4.6a.

Table 4.6a Concatenated Code Error and Rejection Probabilities

(γ fixed)

R	P_e	P_r
	FOR GAMMA =	.150000
0.000000	7.0263396D-01	2.9736604D-01
.845000	1.0371950D-02	9.8961555D-01
1.280000	3.1151452D-03	9.9617006D-01
2.645000	4.2801394D-06	8.3013699D-01
3.380000	3.8884724D-08	5.3177568D-01
4.805000	1.4811282D-12	1.0490389D-01
5.445000	1.1938983D-14	4.0749537D-02
6.845000	2.4282347D-19	4.1936238D-03
8.405000	1.1879819D-24	3.2001767D-04
	FOR GAMMA =	.200000
0.000000	6.7992613D-01	3.2007387D-01
.845000	6.0414845D-03	9.9395098D-01
1.280000	1.7286613D-03	9.9779215D-01
2.645000	2.0322940D-06	8.5438535D-01
3.380000	1.6326362D-08	5.6781203D-01
4.805000	4.6326295D-13	1.1829734D-01
5.445000	3.2158296D-15	4.6286050D-02
6.845000	4.6006374D-20	4.6017934D-03
8.405000	1.4766195D-25	3.0617580D-04
	FOR GAMMA =	.250000
0.000000	6.5766521D-01	3.4233479D-01
.845000	3.5516477D-03	9.9644365D-01
1.280000	9.6393600D-04	9.9870851D-01
2.645000	9.7432473D-07	8.7725558D-01
3.380000	7.1168305D-09	6.0908795D-01
4.805000	1.6631355D-13	1.4067779D-01
5.445000	1.0578446D-15	5.7473457D-02
6.845000	1.2574415D-20	6.1413256D-03
8.405000	3.3704728D-26	4.1463312D-04
	FOR GAMMA =	.300000
0.000000	6.3536327D-01	3.6463673D-01
.845000	2.0533177D-03	9.9794370D-01
1.280000	5.2049964D-04	9.9925626D-01
2.645000	4.4192117D-07	9.0077743D-01
3.380000	2.9736946D-09	6.6073065D-01
4.805000	6.1110416D-14	1.8029359D-01
5.445000	3.7156760D-16	8.0433117D-02
6.845000	4.1032359D-21	1.0520865D-02
8.405000	1.0339322D-26	8.8378550D-04

Table 4.6b Concatenated Code Error and Rejection Probabilities

($\gamma' = \gamma(6.845/R)^{1/2}$)

R	P_e	P_r
	FOR GAMMA =	.150000
0.000000	4.99137330-07	9.99999500-01
.845000	4.45107540-04	9.99553930-01
1.280000	2.87088800-04	9.99557060-01
2.645000	1.12493200-06	8.72848100-01
3.380000	1.31924440-08	5.77655290-01
4.805000	7.42561940-13	1.11866560-01
5.445000	7.23448780-15	4.23327470-02
6.845000	2.44859700-19	4.19328320-03
8.405000	2.55025420-24	3.40876930-04
	FOR GAMMA =	.200000
0.000000	1.24801080-12	1.00000000+00
.845000	4.47249750-05	9.99955080-01
1.280000	4.42249250-05	9.99905170-01
2.645000	3.10638840-07	9.10702860-01
3.380000	3.98144440-09	6.42657750-01
4.805000	2.10476200-13	1.34200320-01
5.445000	1.86257300-15	5.06202540-02
6.845000	4.63967230-20	4.59703280-03
8.405000	3.08830050-25	3.01138020-04
	FOR GAMMA =	.250000
0.000000	9.83284120-20	1.00000000+00
.845000	2.18962040-06	9.99997790-01
1.280000	3.74653730-06	9.99985560-01
2.645000	6.31807440-08	9.48591970-01
3.380000	1.01490020-09	7.30575270-01
4.805000	6.39587770-14	1.78016320-01
5.445000	5.67016610-16	6.91795380-02
6.845000	1.25879430-20	5.12305230-03
8.405000	6.57027550-26	3.38801650-04
	FOR GAMMA =	.300000
0.000000	3.49727380-28	1.00000000+00
.845000	4.89475460-08	9.99999950-01
1.280000	1.59234580-07	9.99998550-01
2.645000	8.33752980-09	9.78007120-01
3.380000	1.89962410-10	8.31810860-01
4.805000	1.77093940-14	2.69497660-01
5.445000	1.71861440-16	1.09662790-01
6.845000	4.14252570-21	1.04602630-02
8.405000	2.01679550-26	5.34186160-04

4.3.2.2. Improved Error and Rejection Probability Estimates Based on Code Word Weight Distributions

Actually, the error probability figures listed here are undoubtedly quite conservative. It has been consistantly assumed in determining these probabilities that a word error would automatically result if there were more than some initial number n_c of symbol errors. In actuality, most error patterns' containing more than n_c errors will, in the situations of interest here, still result in non-decodable words which will be rejected.

This is particularly true in the case of the (15,9) Reed-Solomon code designed to correct only up to two erasures and only to detect errors. To demonstrate this, it is necessary to know the weight distribution of the words in the code in question. Fortunately, the weight distribution for Reed-Solomon codes is indeed known. Specifically*, the number of code words of weight $w \geq 7$ in the (15,9) Reed-Solomon code over $GF(2^4)$ is

$$A_w = 15\binom{15}{w}\sum_{i=0}^{w-7}(-1)^i\binom{w-1}{i}2^{4(w-7-i)} \quad (4)$$

---

* See, for example, Berlekamp, Algebraic Coding Theory, McGraw-Hill Book Co., New York, 1968, p. 429.

The decoding algorithm under consideration involves no error correction and at most two erasure corrections. Consequently, if there are no erasures, the error pattern must itself be a non-zero code word to cause a decoding error. Thus, of the

$$B_e = \binom{15}{e} 15^e \tag{5}$$

error patterns involving e errors, only the fraction

$$\frac{A_e}{B_e} = 15^{1-e} \sum_{i=0}^{e-7} (-1)^i \binom{e-1}{i} 2^{4(e-7-i)} \approx 15^{-6} \quad (\text{alle} \geq 7) \tag{6}$$

of these patterns will cause a decoding error.

Similarly, if an error/erasure pattern involving e errors and 1 erasure is to cause a decoding error, it must either be a code word of weight e with one erased zero or a code word of weight e + 1 with one non-zero symbol erased.

Thus, of the

$$C_e = \binom{15-e}{1} \binom{15}{e} 15^e \tag{7}$$

such patterns, only

$$D_e = \binom{15-e}{1} A_e + \binom{e+1}{1} A_{e+1} \tag{8}$$

can cause a decoding error. Finally, of the

$$E_e = \binom{15-e}{2} \binom{15}{e} 15^e \tag{9}$$

error/erasure patterns of weight e and containing two erasures, only

$$F_e = \binom{15-e}{2} A_e + \binom{15-e-1}{1} \binom{e+1}{1} A_{e+1} + \binom{e+2}{2} A_{e+2} \tag{10}$$

such patterns will cause a decoding failure.

Since

$$\frac{A_e}{B_e^*} \approx 15^{-6}, \quad \frac{C_e}{D_e} \approx 15^{-5}, \text{ and } \frac{E_e}{F_e} \approx 15^{-4}$$

regardless of the value of e ($e \geq 5$), we conclude that the conditional probability of a decoding error, given that the number of guaranteed correctable and detectable errors has been exceeded, is at most of the order of

$$15^{-4} \approx 2 \times 10^{-5}$$

Accordingly, a more reasonable estimate of the actual probability incurred using the concatenated coding approach results upon multiplying the figures quoted in Tables 4.5 and 4.6 the factor 2×10^{-5}. The fact that most error patterns of greater than the critical weight are still not code words does indeed make a significant difference in this particular case.

Since the (8,4) inner code is a more nearly "closed-packed" code than the (15,9) Reed-Solomon code, the change in the error probability estimate incurred by taking its weight distribution into account is considerably less dramatic. Nevertheless, since this effect is also easily determined, it is worthwhile to do so. Let $P_e(i,j)$ denote the probability of a decoding error (as opposed to an erasure) given that the received word contains i erroneous bits and j erased bits. In the preceding investigation it was assumed that $P_e(i,j) = 1$ for all $i \geq 3$, $j = 0, 1$, and for all $i \geq 2$, $j = 2$. But, as is easily verified, the actual values of $P_e(i,j)$ are as listed in the following matrix

j \ i	0	1	2	3	4	5	6	7	8
0	0	0	0	1	1/5	1	0	1	1
1	0	0	0	1	1/5	0	0	1	-
2	0	0	1/5	2/5	2/5	0	1	-	-

(Recall that the (8,4) code contains exactly one word of weight zero, 14 words of weight four, and one word of weight 8.)

The computer program was modified to use these values of P(i,j). The results are included in the Appendix and summarized in Table 4.6c. It will be noted that the rejection probabilities are actually somewhat decreased in most cases of interest when this refinement is taken into account, but, as expected, the change is minor. (In order to facilitate comparison with earlier results, Table 4.6c does not reflect the above-mentioned reductions in the error probability estimates resulting from consideration of the Reed-Solomon code structure. Thus, these error probabilities should actually be reduced by a factor of approximately 2×10^{-5}.)

P_e	P_γ	R
	$\theta = 0$	
2.427854800-02	9.7572145D-01	0.000000
2.201831600-02	9.7787772D-01	.845000
1.164720950-02	9.8469692D-01	1.280000
5.295560800-05	7.0225572D-01	2.645000
9.908123800-07	3.9336500D-01	3.380000
1.906532600-10	5.2109173D-02	4.805000
3.431999560-12	3.8006165D-02	5.445000
4.552623980-16	7.9381161D-03	6.845000
2.057165160-20	1.5622839D-03	8.405000
	$\theta = 0.15$	
2.427854800-02	9.7572145D-01	0.000000
1.607941180-03	9.9603037D-01	.845000
5.893447600-04	9.9855744D-01	1.280000
6.406495000-07	6.2136995D-01	2.645000
4.802819500-09	5.1961264D-01	3.380000
1.317283210-13	9.9224805D-02	4.805000
9.451277800-16	3.7846684D-02	5.445000
1.599100320-20	3.6497519D-03	6.845000
7.050624200-26	2.4557351D-04	8.405000
	$\theta = 0.20$	
2.427854800-02	9.7572145D-01	0.000000
6.764834800-04	9.9931319D-01	.845000
1.493143900-04	9.9972029D-01	1.280000
1.152953730-07	8.4399864D-01	2.645000
5.952204700-10	5.5502093D-01	3.380000
7.724408600-15	1.1718742D-01	4.805000
3.940030100-17	4.3126723D-02	5.445000
3.151135600-22	4.0336445D-03	6.845000
5.876519600-28	2.3411784D-04	8.405000
	$\theta = 0.25$	
2.427854800-02	9.7572145D-01	0.000000
2.642954000-04	9.9972941D-01	.845000
6.397046700-05	9.9953464D-01	1.280000
1.862194700-08	8.6977498D-01	2.645000
5.548421100-11	5.9711922D-01	3.380000
4.045749400-15	1.3477308D-01	4.805000
1.483201500-18	5.4437117D-02	5.445000
5.801339390-24	5.6514989D-03	6.845000
	$\theta = 0.30$	
2.427854800-02	9.7572145D-01	0.000000
9.953526100-05	9.9969628D-01	.845000
1.928967100-05	9.9970533D-01	1.280000
2.681344900-09	8.9475216D-01	2.645000
6.391112300-12	6.5206580D-01	3.380000
1.919900290-17	1.7503059D-01	4.805000
5.152000570-20	7.7093581D-02	5.445000
1.030699870-25	1.0048737D-02	6.845000

Table 4.6c Refined Concatenated Code Error and Rejection Probabilities ($d_o = 1$, $d_1 = 3$, θ fixed)

Also included in Table 4.6c (and the appendix) is the case $\theta = 0$; i.e. the case in which no erasures are produced at the demodulator output. While the decoder performance in this case is marginally acceptable, it is clearly inferior to that attainable using a non-zero threshold. This fact, combined with the advantage of having a threshold in the event of a sudden drop in signal level (see section 4.3.2.1) strongly suggest that a non-zero threshold is worth the slight increase in demodulator complexity. Neverthless, is is comforting to note that, should a threshold prove inconvenient (or should it for some reason malfunction) satisfactory performance could still be achieved.

4.3.3. Encoder and Decoder Descriptions

Detailed logic diagrams of the encoders and decoders for both the inner (8,4) and outer (15,9) codes are shown in Figures 4.3 through 4.6. The rationale for these designs is explained in this section.

4.3.3.1 The Outer Encoder

The outer code encoder (figure 4.3) encodes two 18-bit commands into one code word. Two code words are loaded in parallel into the 36 encoder shift-register. The contents of the register are then shifted as indicated at a rate of one shift every 2/15th millisecond. After each 15 shifts, a new pair of commands is again loaded into the shift-register and the process begins anew.

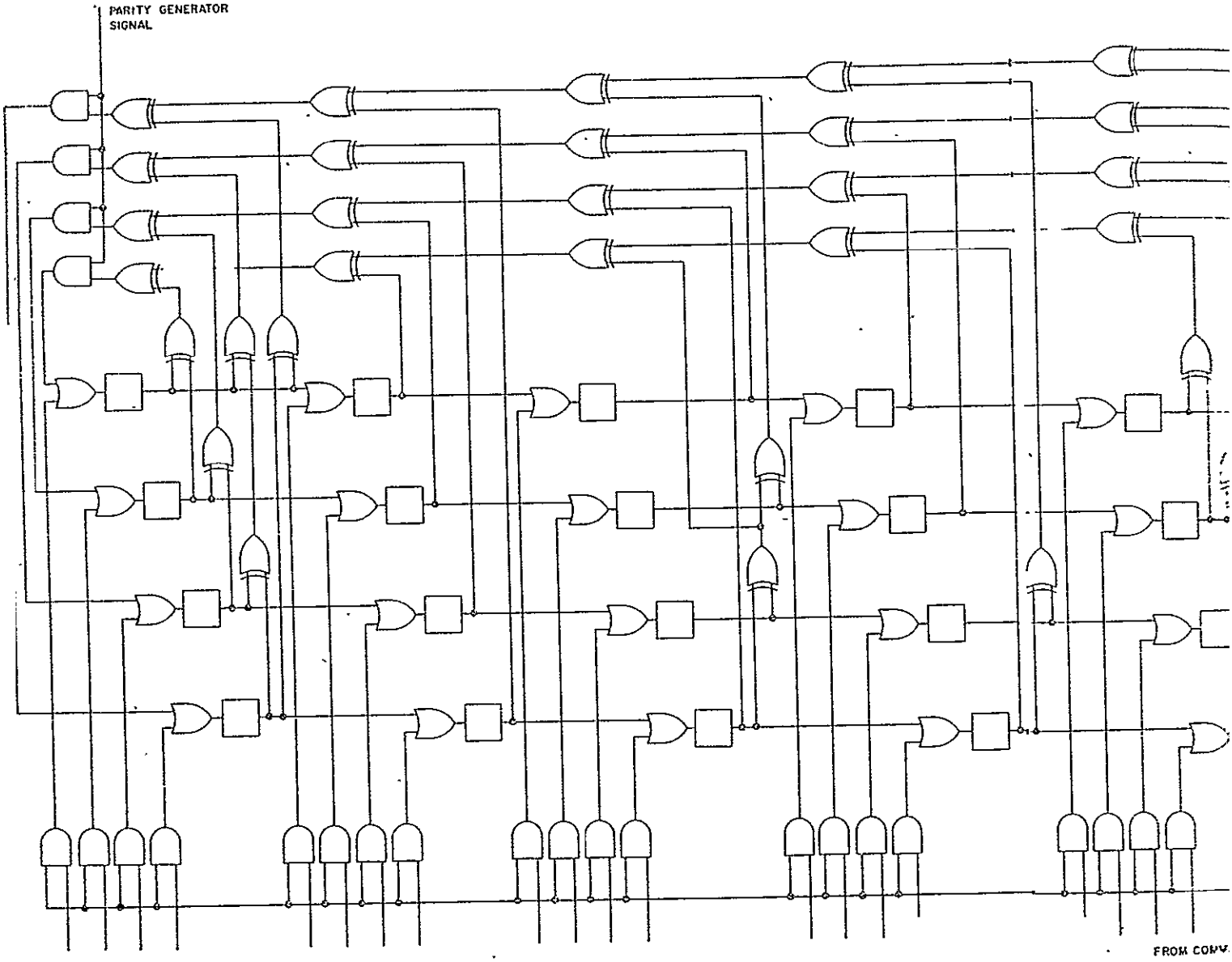
The feedback logic is determined by the Reed-Solomon generator polynomial

$$h(x) = \prod_{i=1}^{9} (x + \sigma^i) = x^9 + \sigma^6 x^8 + \sigma^5 x^7 + \sigma^2 x^6 + \sigma^{11} x^5 + \sigma x^4 + \sigma^2 x^3 + x^2 + \sigma^{11} x + 1 \qquad (1)$$

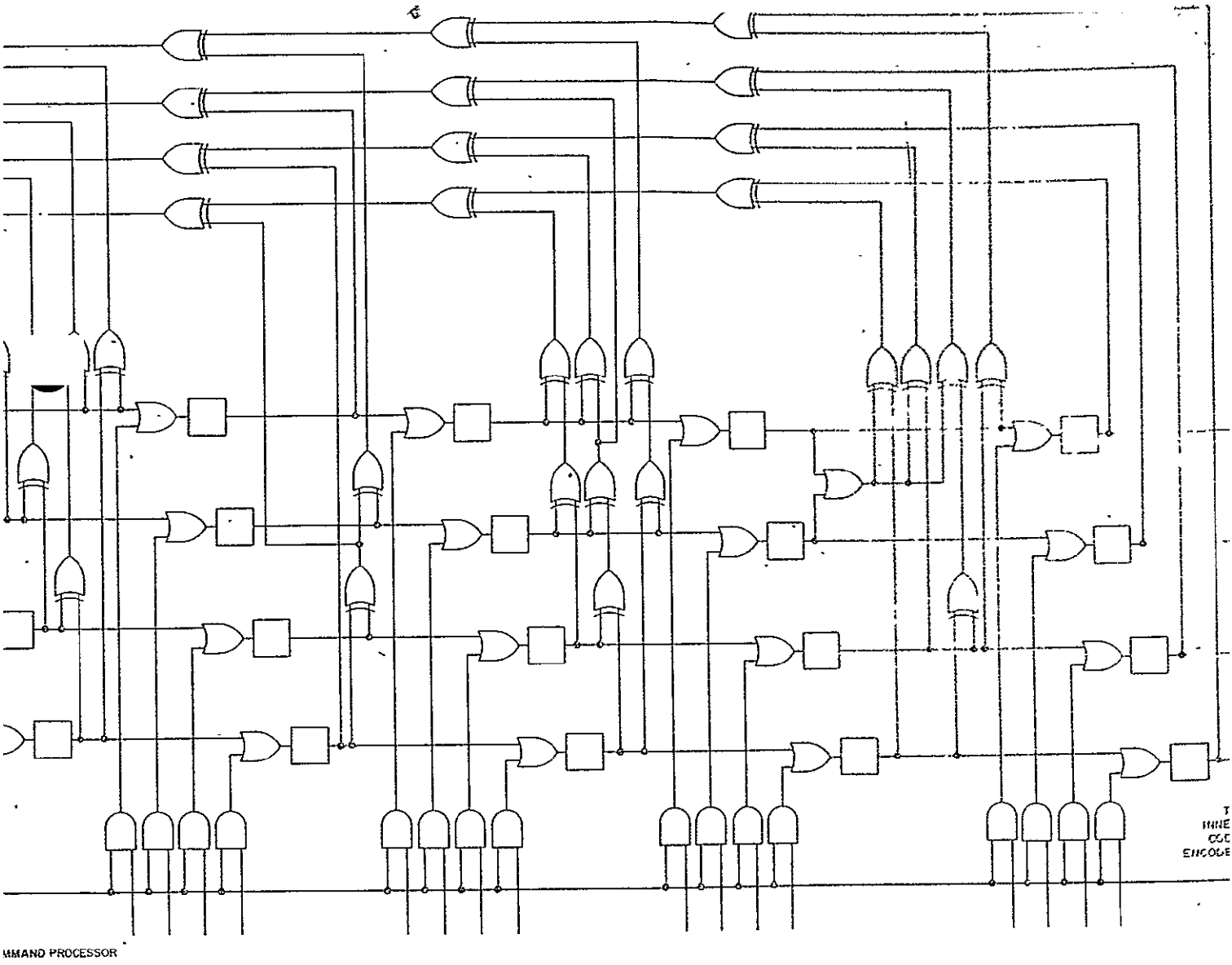
with σ a primitive element in $GF(2^4)$.*

Thus, the encoder is entirely analogous to the more familiar binary shift-register encoders. The only difference is that in this case the code "symbols" are elements in $GF(2^4)$ rather than in the binary field. Consequently, four binary digits are required to define each code symbol, and the arithmetic operations are performed on these groups of four digits rather than on the digits individually.

* See, for example, Berlekamp; op.cit.



FOLDOUT FRAME



FOLDOUT FRAME 2

Figure 4.3 Outer Code Encoder

4.3.3.2 The Inner Encoder

The heart of the inner code encoder (figure 4.4) is a ripple-counter operating at a 60 Khz rate; i.e. the outputs of the three ripple-counter stages are, respectively

```
. . . 0 1 0 1 0 1 0 1 . . .
. . . 0 0 1 1 0 0 1 1 . . .
. . . 0 0 0 0 1 1 1 1 . . .   (1)
```

with a change of state occuring every 1/60th of a millisecond. The 4 right-most cells of the outer code decoder serve as inputs to the inner code encoder, controlling the logical combination of the code generators, (1) used to generate a given word. After 8 shifts of the ripple-counter, the outer code shift-register is shifted once and the next set of four bits in encoded. Note that the four right-hand cells of the outer encoder shift-register are duplicated; this is so that the last four outer code bits can be encoded at the time that the next pair of commands is being read into the outer code shift-register. It will also be noted that the forth input to final "exclusive-or" consists of a logical combination of the code generators. An alternative and simpler method would be to add the forth outer code bit modulo-two to the exclusive-or output. The first method, however, results in a systematic code (the first, second, third, and fifth code bits can be identified uniquely with the bits which were uncoded) while the second does not. As a result the decoder is somewhat less complex using the first method.

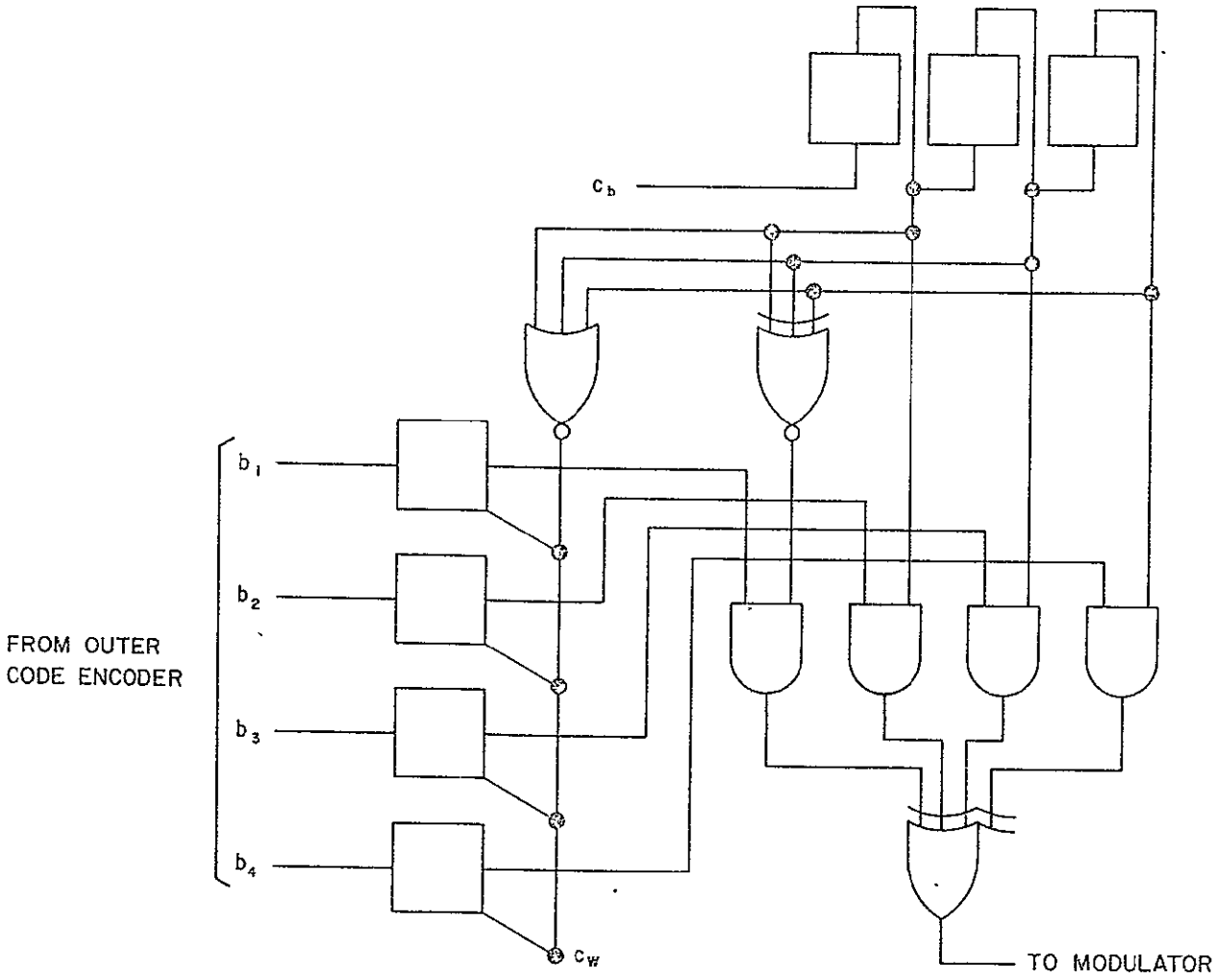


Figure 4.4 Inner Code Encoder

4.3.3.3 The Inner Decoder

The inner code decoder philosophy is based on the following observations:

(1) The (8,4) orthogonal code (as represented here) can be looked at as a 7,4 Hamming code augmented by an overall parity-check bit (the last bit in each word.) Thus, to correct one error, the decoder need only determine the three-bit Hamming code syndrome ($p_1p_2p_3$) from the first 7 code word bits. If the syndrome is the all zeros syndrome (000) either no errors have occurred, or, if the code word contains an odd number of "one", the overall parity bit is in error (excluding, of course, the possibility of three or more errors). In either even, the information bits (bits 1,2,3, and 5) can be assumed to be correct.

If the syndrome is other than the all zeros syndrome, it will uniquely identify the location of the error (again provided only one error has occurred). A logical combination of the syndrome bits can then be used as the error corrector. (If the error is not in an information bit, of course, no correction is required.)

If only one error has occurred, the code word must contain an odd number of "ones". Accordingly, if an error is detected and the parity is even, the word must contain at least two errors, and, under the proposed decoding rule, the code word will be rejected and the corresponding outer code symbol treated as an erasure.

(2)

If the word contains either one or two erasures these erasures are to be corrected, under the proposed decoding rule, unless the code is also found to contain an error. Thus, such a word can be decoded by first decoding it with the erasures represented by "zeros" and, then, if the first attempt is unsuccessful, decoding it with the erasures represented by "ones". If the code word is found to contain either no errors or one error under either decoding attempt, and in the latter case, if the observed error is in an erased bit, the corrected word satisfies the desired condiditons and the code word can be accepted as correct. Under any other set of conditions the code word must be rejected and the corresponding outer code symbol represented as an erasure.

(3) Finally, the code word is to be rejected if it contains three or more erasures.

The inner code decoder is shown in Figure 4.5. The received code bits produced at the output of the demodulator are read serially into the lower shift-register and the erasure indicators are read into the upper register (a "one" indicating an erasure). If a bit is erased, it is represented by a zero in the lower register. As the bits are read in their parity is determined and the number of erasures is counted. After the 8 bits comprising one code word have been read in the syndrome $(p_1 p_2 p_3)$ is determined and the approriate logical combination of this syndrome used to generate the proper correction bit. If an error is found, the correction is carried out unless the word parity was found to be even indicating an even number of errors. If either one or two erasures are observed, and if the first decoding was not successful, the erasure indicator bits are shifted into the corresponding erased bit positions and the word is again decoded. (If there is only one erasure, this changes the code word parity, so the state of the parity indicator is also changed under this condition.) Note that if there is an erasure the decoded word is accepted only if either no errors are observed of if the single observed error occurs in an erased bit.) If neither decoding is successful, of if the erasure counter counts three or more erasures, the decoder output is represented by the all-zeros word and a "one" is outputted from the erasure indicator. A total of 1/60th millisecond is available for carrying out the (at most) two steps needed to decode each of the inner code words.

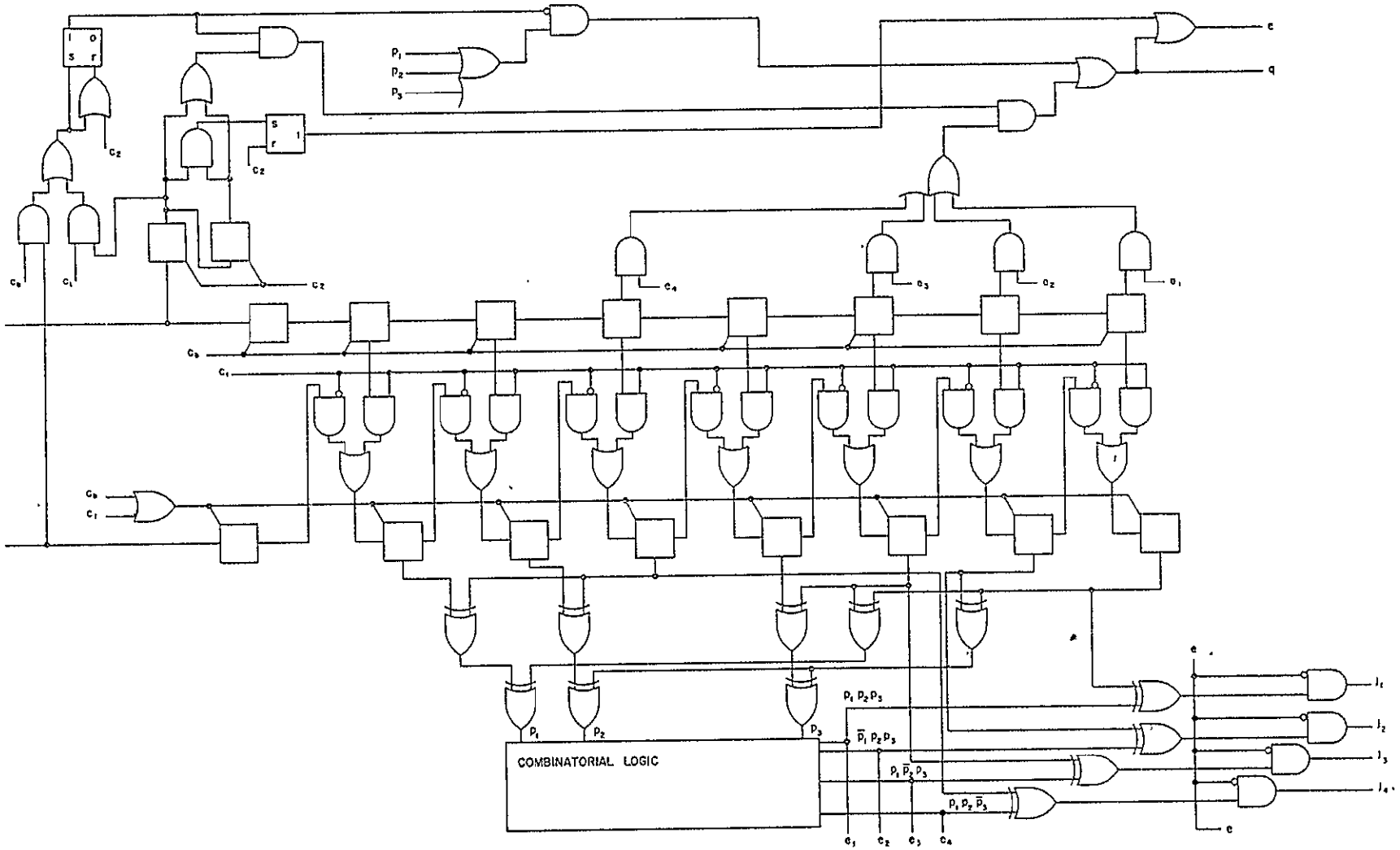


Figure 4.5a Inner Code Decoder

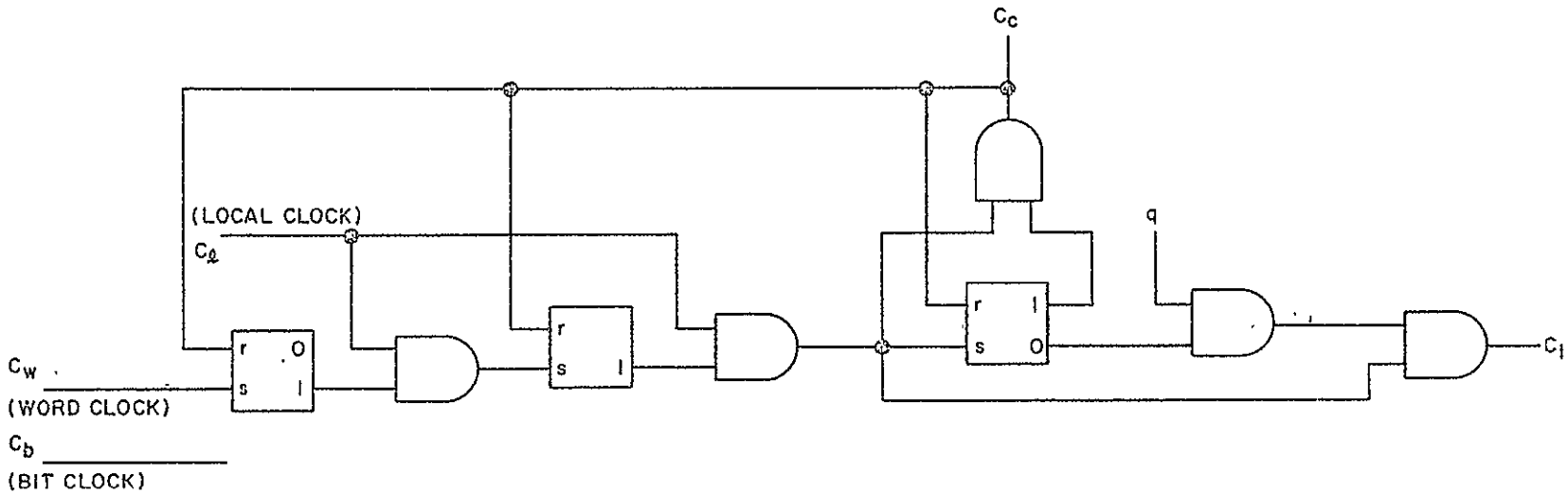


Figure 4.5b Timing Generator for Inner Code Decoder

These decoded words then constitute the four-bit "symbols" of the outer code. These symbols are shifted serially in groups of 15 to the outer code decoder shift-register.

4.3.3.4 The Outer Decoder

The generally most efficient decoding algorithm for Reed-Solomon codes is essentially that described earlier for binary BCH codes. (Since Reed-Solomon codes are defined over larger fields than the binary field, an additional step is required; after the errors are located, their value must also be determined.) In the situation of interest here, however, only erasures are to be corrected. Consequently, the decoder can be considerably simplified. This is especially true, when, as here, at most two erasures are to be corrected.

The key to the decoding algorithm is the observation that, when the code word contains at most two erasures, it is always possible to shift the code word cyclically until one erasure appears at either the 4th or the 9th symbol position and the second at the 10^{th}, 11^{th}, 12^{th}, 13^{th}, or 14^{th} position. In the absence of either erasures or errors, any nine consecutive symbols of a (15,9) Reed-Solomon code define the following symbol (see figure 4.3). If one of these nine symbols is an erasure, the tenth symbol then uniquely determines what these symbols must be. But this is precisely the situation when the code word is cyclically shifted as described above.

(The reason for requiring the one erased symbol to appear in either the fourth or the ninth symbol position is to simplify the decoding logic, as will become apparent shortly.) In particular, suppose the erased symbol appears at position four. Then from equation (1),

$$\sigma s_4 = s_{15} + \sigma^{11} s_1 + s_2 + \sigma^2 s_3 + \sigma^{11} s_5 + \sigma^2 s_6 + \sigma^5 s_7 + \sigma^6 s_8 + s_9 \triangleq q \qquad (3)$$

with s_i denoting the symbol in the i^{th} shift-register position. (Note that $s_{15} = s_0$ and that, since the symbols are defined over a field of characteristic 2, $s_i = -s_i$.) Thus, it is only necessary to shift $\sigma^{-1}q$ into the erased position to "correct" it. Similarly, if the erased symbol is in position nine, it can be corrected by reading $\sigma^0 q = q$ into that position. The second erasure can then be corrected by cycling the code until it appears at either position four or nine and again reading in the corrected symbols. If the code word contains errors as well as erasures, of course, these "corrections" may be in error. However, if the number of erasures plus errors is six or less, the distance properties of the code guarantee that the resulting corrected word will not be a code word and hence that some of the parity-checks must fail. This event can therefore be detected by shifting the code cyclically fifteen times (i.e. fifteen one-symbol shifts) and checking each time to see whether or not all the parity-check relationships are satisfied. (Actually, only six shifts

are required, but is is convenient to return the code to its original position.)

This is the algorithm implemented by the decoder shown in figure 4.6. The heart of the decoder is a shift-register essentially identical to that of the encoder (figure 4.6b). The erased positions are indicated by "ones" in a second shift-register. (figure 4.6d) The 15 code-word symbols from the inner code are first read into the decoder shift-register and the erased position indicators into the erasure shift-register. The local clock then replaces the symbol clock as the shift signal (figure 4.6e). At the same time, the decoder shift-register input is switched from the inner decoder to its own output so that the code word can be cycled (figure 4.6c). When the erased positions are properly located as described above, the corrected symbol is read into the erased position (see figure 4.6b and 4.6d)). (Note that the erased symbol is represented by $\theta = 0000$ so it does not affect the formulation of q.) The second erasure is similarly corrected (if there is a second erasure). Both erased symbols will thus certainly be replaced in this manner by the time two complete cycles (30 shifts) have been completed. After two complete cycles, then, the corrected word is shifted fifteen more times. If any of the parity-checks fail during this last step, (or if more than two erasures were observed; see figure 4.6d), the reject flip-flop (figure 4.6a) is set and the decoded word rejected. Otherwise it is accepted as a valid word. This completes the decoding operation.

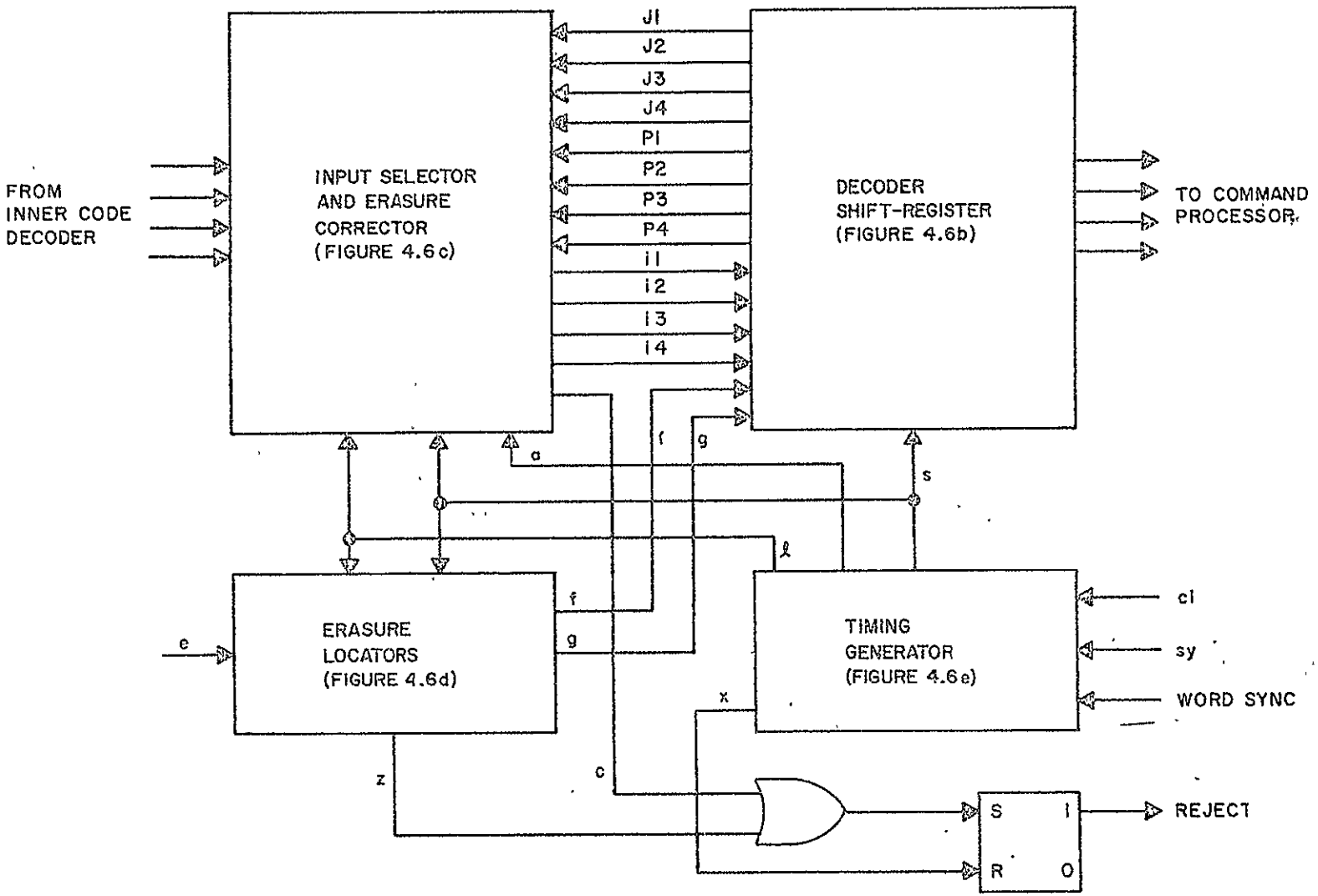
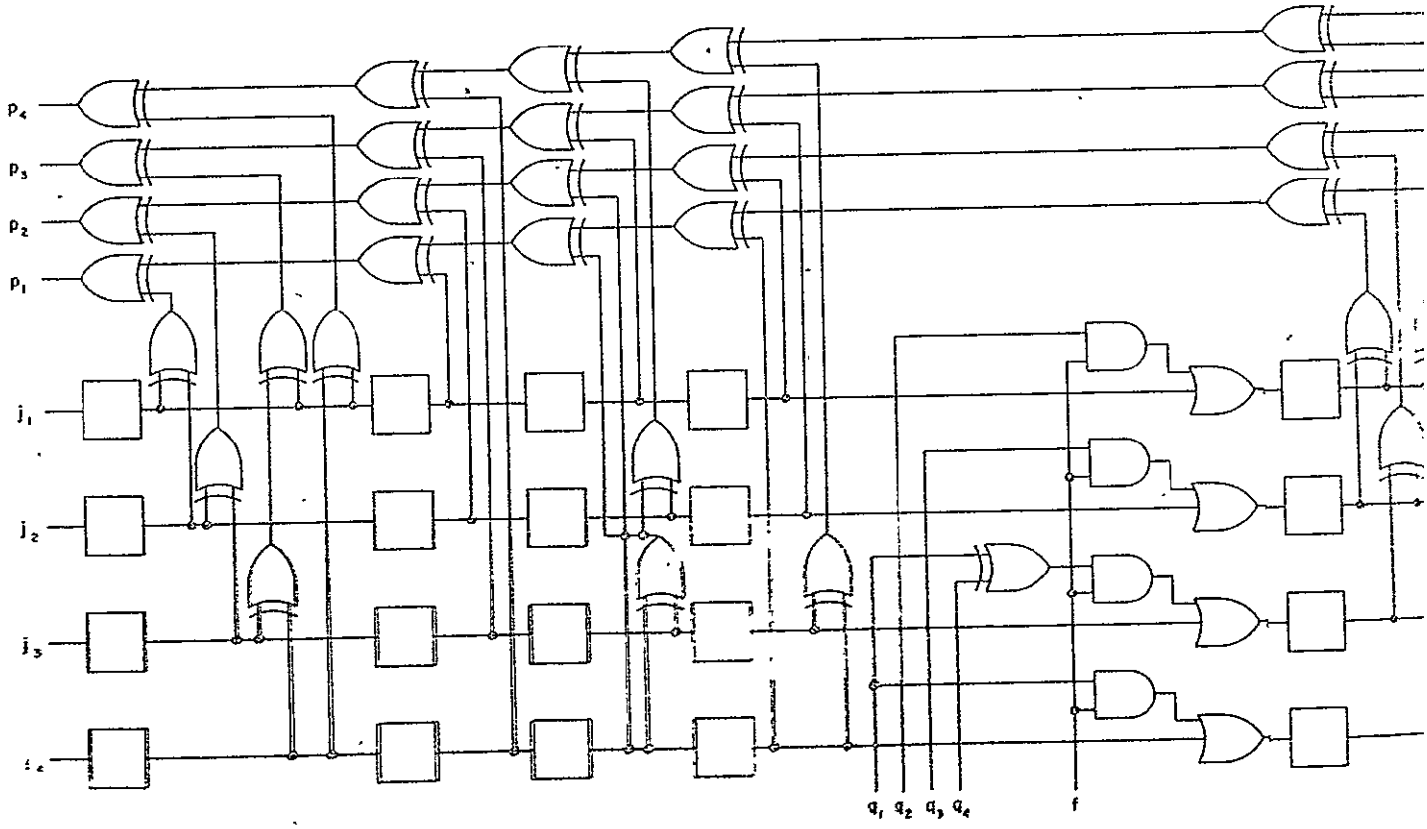


Figure 4.6a Outer Code Decoder



FOLDOUT FRAME

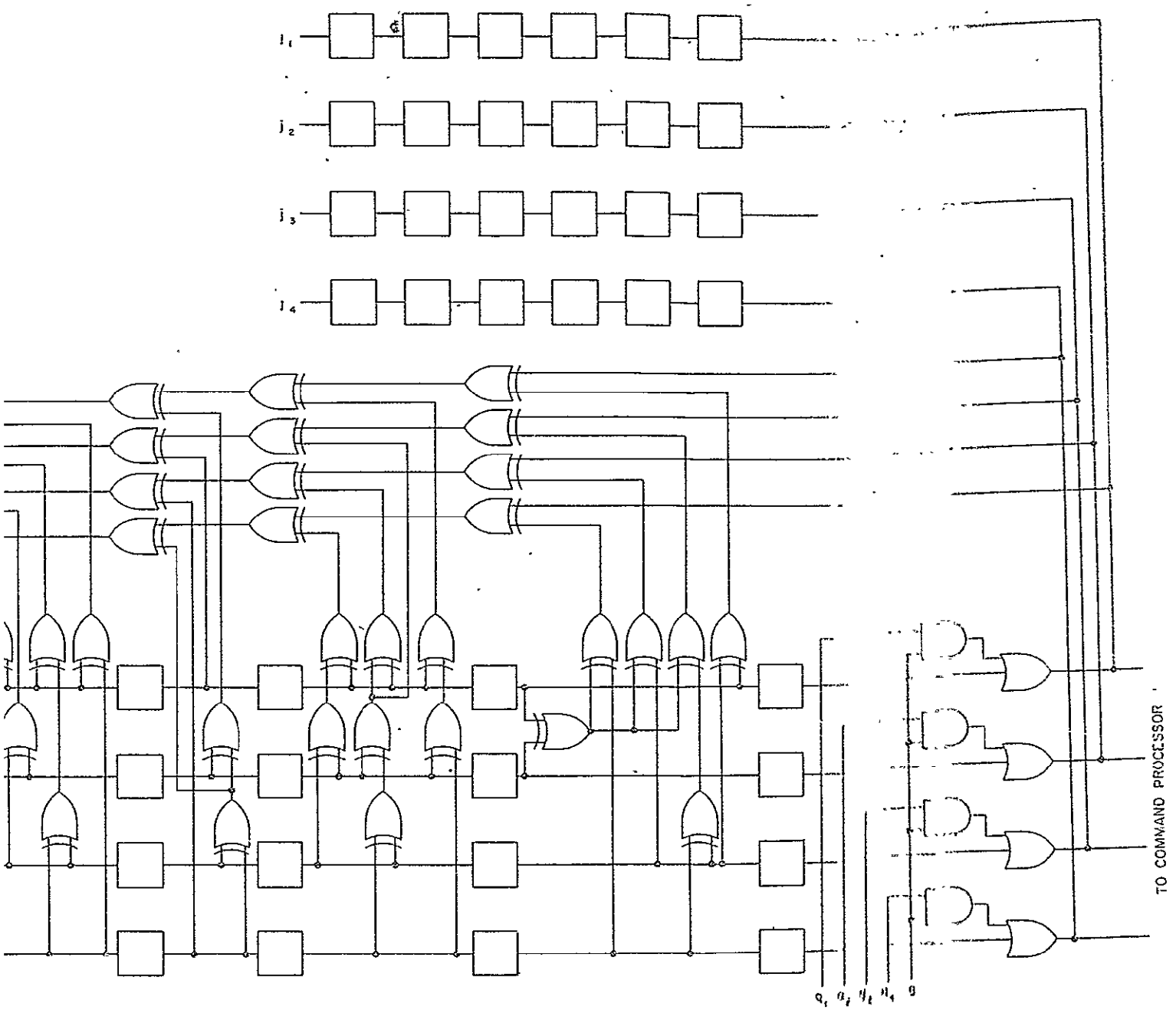


Figure 4.6b Decoder Shift-Register

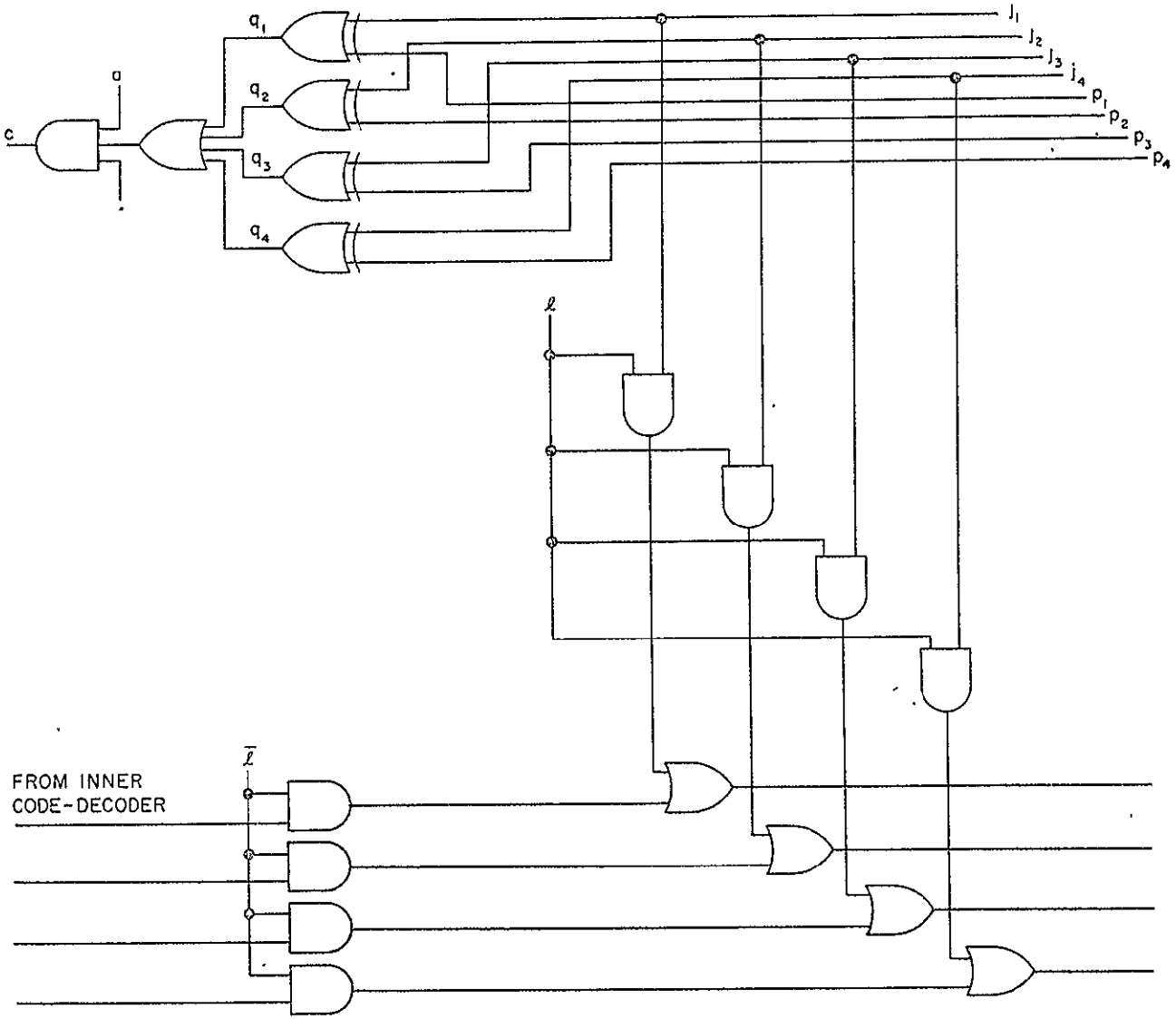


Figure 4.6c Input Selector and Erasure Corrector

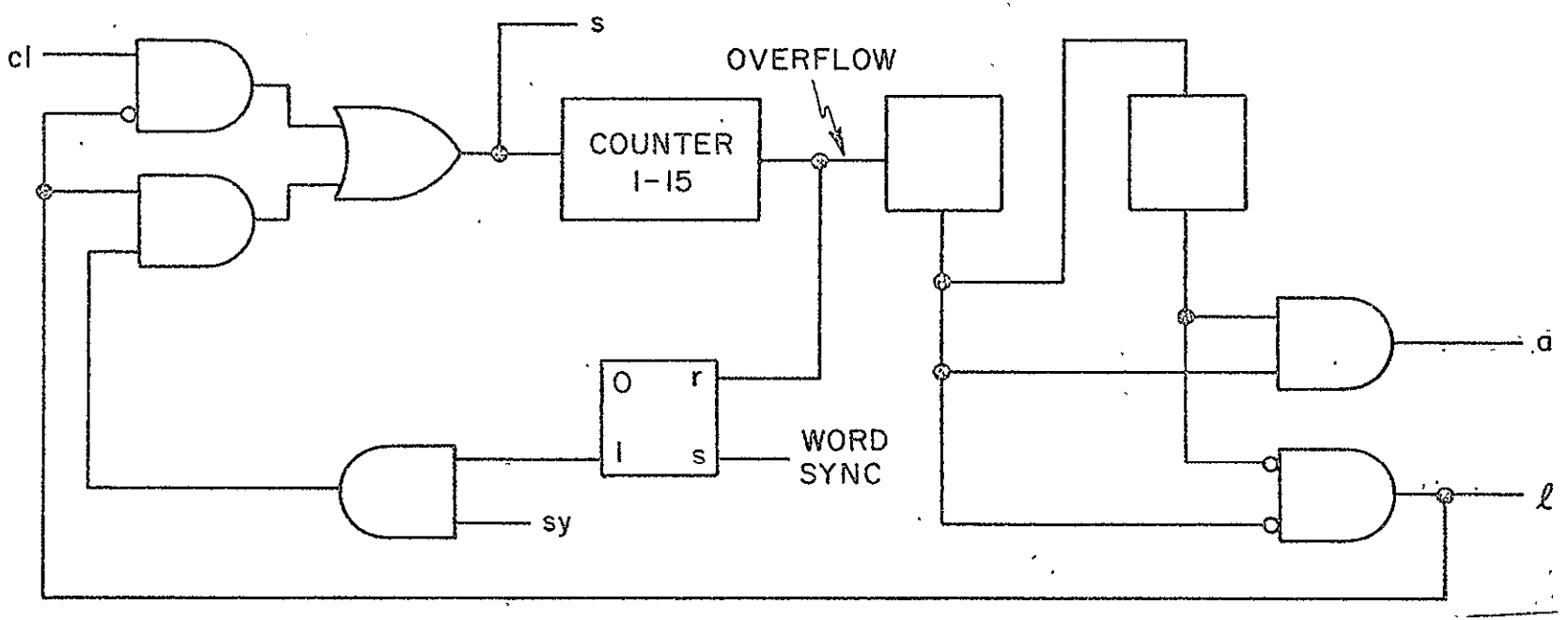


Figure 4.6e Timing Generator

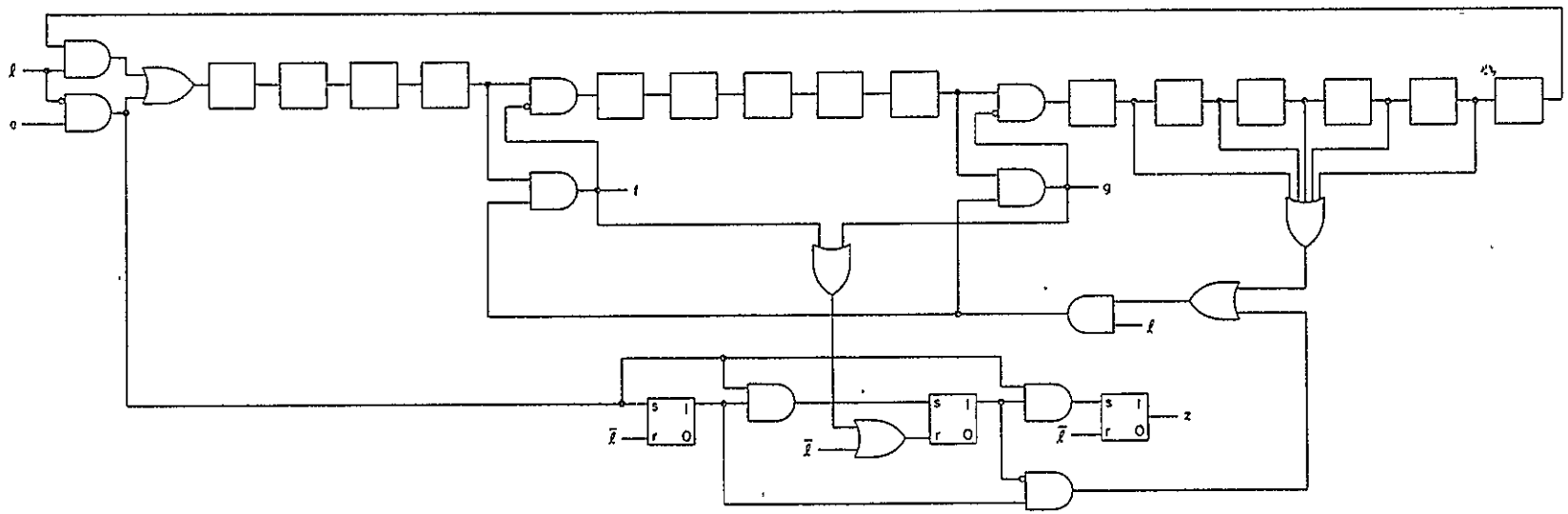


Figure 4.6d Erasure Locators

$\hat{a}$ = error check mode

c = reject (error detected)

ℓ = feedback mode ($\bar{\ell}$ = read-in mode)

z = reject (too many erasures)

cl = local clock

s = shift-register shift pulse

sy = symbol clock

e = erasure indication (from inner decoder)

f = correct erasure in fourth symbol

g = correct erasure in ninth symbol

FIGURE 4.6f GLOSSARY

It will be observed that the required 45 shifts needed to decode a word must be accomplished in the 2/15th millisecond separating the appearance of the last symbol of one code word from the first symbol of the next word. This leaves approximately 3 μsecs for each shift. Since such a shift-register can operate at microsecond rates, this does not appear to pose any problems. Once the decoding has been completed, the decoded pair of command words can be read out serially (or on a 4-bit byte basis) from the 9th "column" of the shift-register. This can be done while the first nine symbols of the next code word are being read in, leaving as much at 6/5 millisecond for this purpose.

4.3.3.5 Parts Count for Decoder, Encoder & Command Controller

Table 4.7 contain a parts count for the decoders and encoders described in the preceeding paragraphs. The parts count is in terms of Dual-in-Line Packages (DIPs) which could be used for a breadboard implementation.

Table 4.7 - Parts Count for Decoder,

Encoder & Command Controller

Unit	Figure #	No. of DIPs
Inner Code Decoder	4.5	20
Timing Generator for Inner Code Dec.	4.5b	4
Decoder Shift Register	4.6b	23
Input Selector & Erasure Corrector	4.6c	4
Erasure Locators	4.6d	9
Timing Generator	4.6e	5
Total,Decoder		65
Outer Code Encoder	4.3	15
Inner Code Encoder	4.4	4
Total, Encoder		19

4.3.4 Concatenated Coding Using a "Green Machine" Decoder

The concatenated coding scheme described in the preceding section used an (8,4) biorthogonal code as the inner code. This code was decoded algebraically following a bit-by-bit detector. An obviously superior method would be for the detector to detect on a word-by-word basis; that is, for the detector to determine which of the 16 possible words was the one most likely received. This in effect involves correlating each received noise-corrupted code word with all sixteen contenders and selecting the word corresponding the largest of these correlations.

The obvious method for implementing the task just outlined is generally much too complex to be seriously considered for the application of concern here. A technique is known, however, whereby the complexity of such detectors can be significantly reduced when the code in question is of a certain class (the Kroenecker product codes) which, fortuitously, includes the (8,4) code.

The resulting detector, which has come to be called the "Green Machine" after its inventor, R. R. Green, is described below. The following subsection discusses the error and rejection probabilities attainable using the same concatenated coding scheme discussed in the preceding section, but with the inner decoder replaced with a Green Machine. The last subsection then examines in detail the logic needed to implement an (8,4) Green Machine decoder.

4.3.4.1. The Green Machine

The discussion here will be limited to the Green Machine associated with the (8,4) biorthogonal code. Half of this code (with zeros represented by + and ones by -) is shown in Figure 4.7. The second half is identical to the first half but with the +'s interchanged.

```
+ + + + + + + +
+ + + + - - - -
+ + - - + + - -
+ + - - - - + +
+ - + - + - + -
+ - + - - + - +
+ - - + + - - +
+ - - + - + + -
```

Figure 4.7 Orthogonal half of an (8,4) Biorthogonal Code

Now let $(y_1, y_2, \ldots, y_8)$ represent the 8 matched filter outputs corresponding to one code word. (Note that the y_i's represent real numbers, not binary numbers.) The ideal detector forms the eight correlations

$$c_j = \sum_{i=1}^{8} s_i(j) y_i \qquad j = 1,2,\ldots,8 \qquad (1)$$

where $s_i(j)$ is the sign represented by the jith element in the matrix in Figure 4.7. If $|c_\nu| > |c_j|$ for all $j \neq \nu$, and if $c_\nu > 0$, it is concluded that the ν th word was transmitted; if $|c_\nu| > |c_j|$, all $j \neq \nu$, and if $c_\nu < 0$, it is concluded that the complement of the ν th word (i.e. the $(\nu + 8)$th word was transmitted. (Actually, since the outer code decoder is designed to recognize erasures, this decision scheme will be altered here so that a word is accepted only if $\max c_j > \theta$ for some threshold θ. If $\max |c_j| \leq \theta$, the received word will be treated as an erasure.

This has no effect on the following discussion, however.)

A device for accomplishing this, the Green Machine, is shown in Figure 4.8. Its operation is as follows: Upon receipt of the first input y_1 the switches labeled "1" in Figure 4.8 are in the "up" position. (The top and bottom of each pair of switches are always switched simultaneously.) Thus y_1 is read into the top of the pair of registers labeled 1. (Note that in practice the y_i will be quantitized and represented by an ℓ -bit binary number. Thus, the cell in the first register actually represents ℓ binary storage cells and, since the contents of the successive registers are sums of the contents of their predecessors, the cells in the second and third registers each represent $\ell + 1$ and $\ell + 2$ storage cells respectively.)

Next, switch 1 is switched and y_2 is read into the lower register. This switching continues with each y_i, i odd, read into the upper register and each y_i, i even, into the lower register.

In the meantime, the switches labeled with a 2 are switched, and the registers labeled 2 are strobed in such a way that $y_1 + y_2$, and $y_1 - y_2$ are successively read into the top register, and $y_3 + y_4$ and $y_3 - y_4$ into the bottom, etc.

Similarly, the switches and registers labeled with a 3 are switched and strobed so as to read, successively, $y_1 + y_2 + y_3 + y_4$, $y_1 + y_2 - y_3 - y_4$, $y_1 - y_2 + y_3 - y_4$, $y_1 - y_2 - y_3 + y_4$, into the top register and $y_5 + y_6 + y_7 + y_8$, $y_5 + y_6 - y_7 - y_8$, $y_5 - y_6 + y_7 - y_8$, $y_5 - y_6 - y_7 + y_8$ into

Figure 4.8 The Green Machine

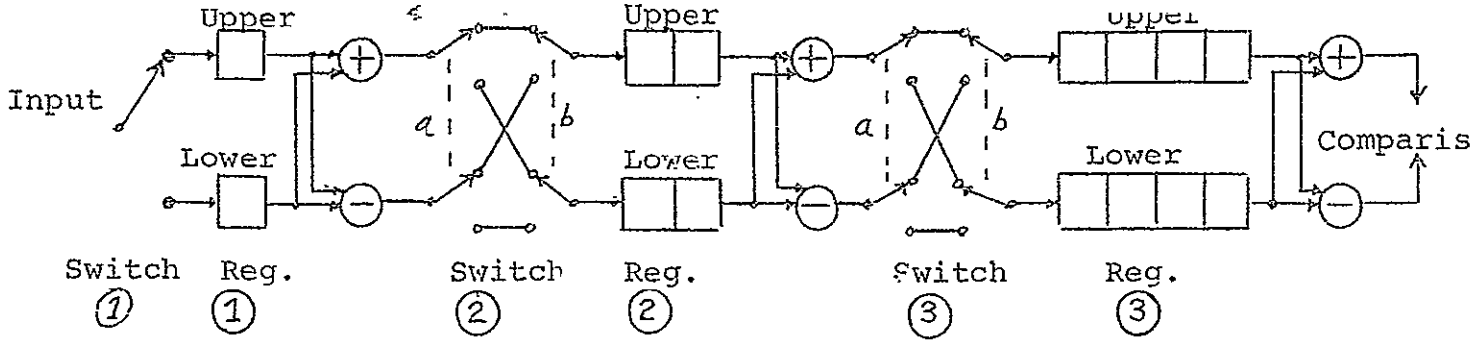
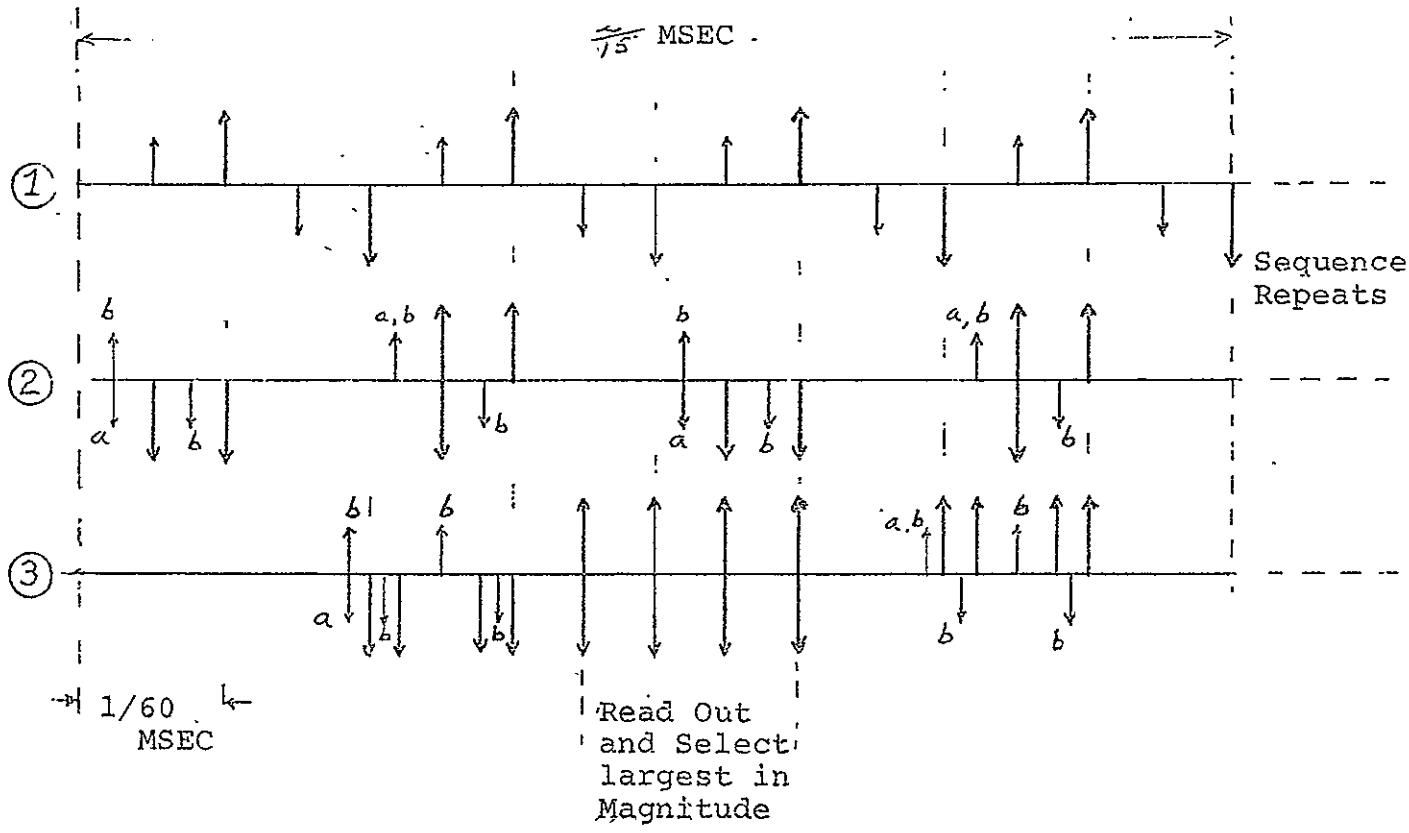


Figure 4.9 Timing Diagram for Green Machine



Note: Short Arrows refer to Switched

Long Arrows refer to A shift into Register.

the bottom register.

Finally, the contents of these last two registers are shifted out and the contents of each lower cell alternately added and subtracted from the contents of the corresponding stage in the upper register. Thus, the first such output is $y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8$, the second $y_1 + y_2 + y_3 + y_4 - y_5 - y_5 - y_6 - y_7 - y_8$, the third $y_1 + y_2 - y_3 - y_4 + y_5 + y_6 - y_7 - y_8$, etc.

It is easily verified that these successive outputs are, indeed, precisely the correlation coefficients of equation (1). Thus, it remains only to determine which of these outputs has the greatest magnitude, whether this magnitude exceeds some preset threshold, and, if so, what its sign is, to complete the decoding.

(As an aid in verifying that the desired correlation are in fact successfully formulated, see the timing diagram shown in Figure 4.9. A short arrow there represents an instant at which the switch in Figure 4.8 bearing the indicated label is switched, its direction indicating whether it is switched to the upper or lower position. Similarly, a long arrow represents a strobing signal to the indicated register, the direction of this arrow denoting whether the upper or lower register is to be strobed.)

4.3.4.2 Detailed Green Machine Implementation

Figures 4.10 through 4.17 illustrate the detailed logic required to implement a "Green Machine" inner decoder. The decoder accepts the demodulator output which is in the form of digitized information representing a received code bit. The logic design in this example is based on a two-bit digitized input (two bits represent one received undecoded message bit).

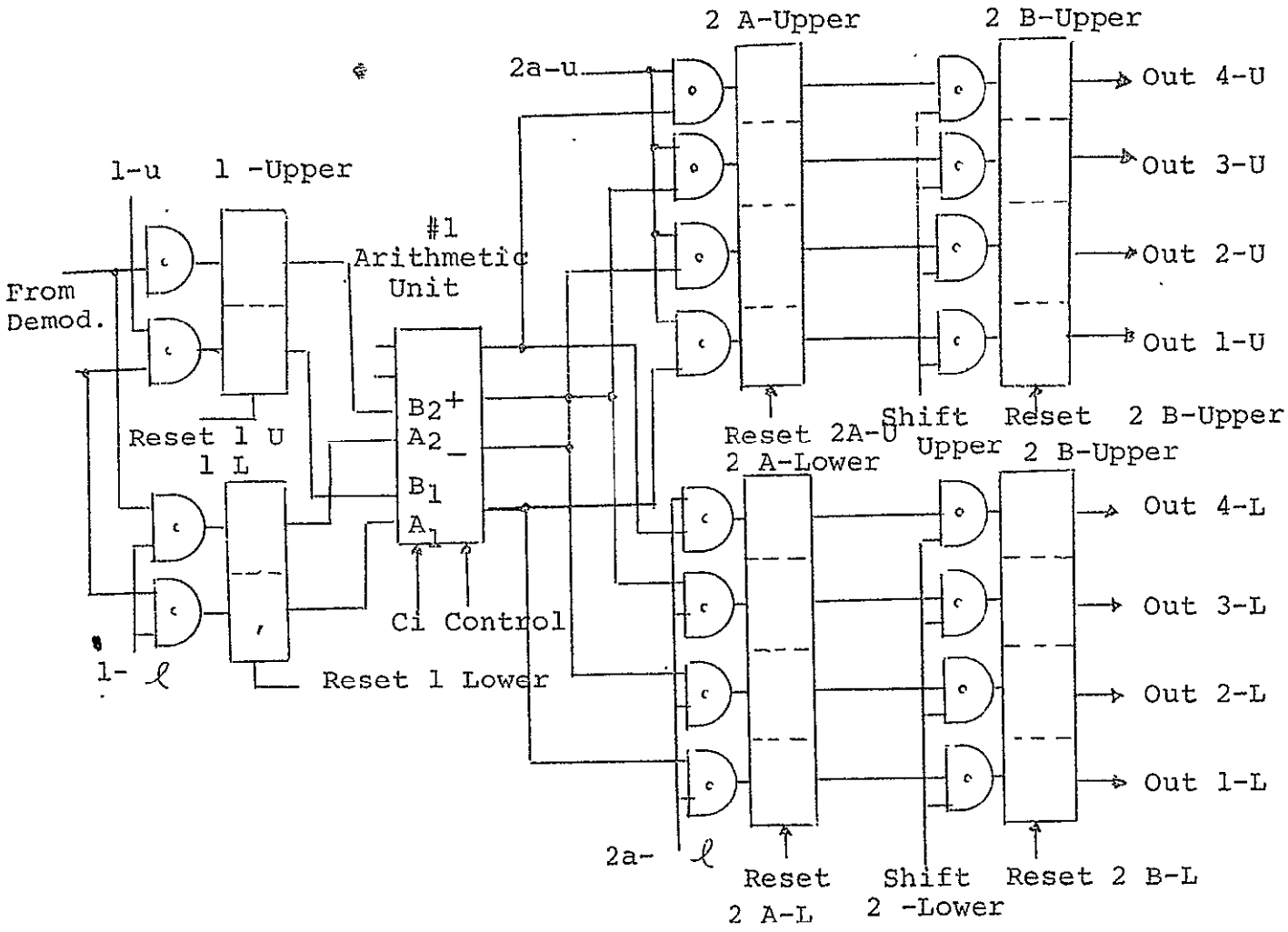
Table 4.8 defines the sequence of operations performed by the "Green Machine" in generating a four-bit output code "symbol" from eight received input bits. Registers specified in the sequences are illustrated in Figures 4.10 through 4.13.

Addition and subtraction operations are performed in two's complement representation of signed binary numbers. The arithmetic unit performs a subtraction by internally complementing the 1 lower output and adding a forced carry-in to the least significant bit. Sign and overflow indications are retained throughout the successive addition and subtraction operations by providing both a sign bit and overflow bit for the sum and difference. The most significant bit represents the sign, with a "1" indicating a negative number. The second most significant bit represents an overflow or underflow condition.

A "1" here for a positive word indicates overflow, while a "0" for a negative word indicates underflow. Input A_3 of arithmetic unit number 1 (Figure 4.10) is forced to a logic "1" when a subtraction is performed.

Figure 4.10

Input Registers 1 and 2



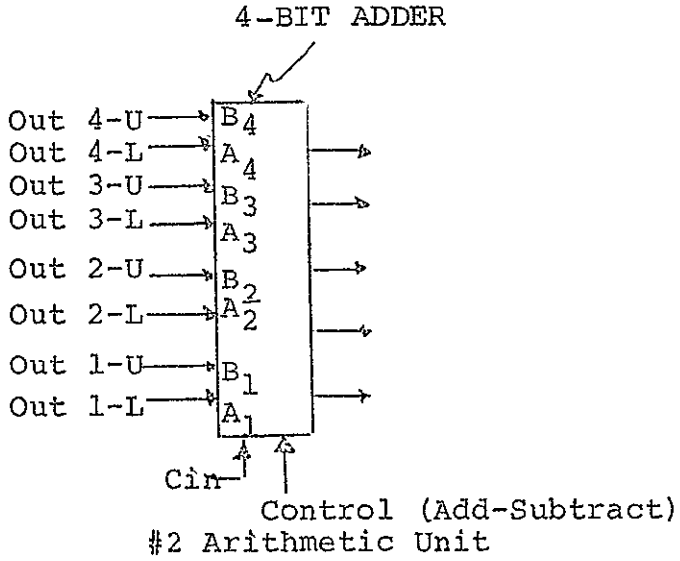


TABLE 4.8

QUENCE FOR GENERATION OF A 4-BIT OUTPUT CODE "SYMBOL" FROM 8 RECEIVED BITS

1. Bit 1 → (1) Upper
2. Bit 2 → (1) Lower
3. Add, (1) U + (1) L, Sum → (2) A - Upper = (1 + 2)
4. (a) (2) A - U → (2) B - U = (1 + 2)
   (b) Sub., (1) U - (1) L, Diff. → (2) A - U = (1 - 2)
5. Bit 3 → (1) U
6. Bit 4 → (1) L
7. Add, (1) U + (1) L, Sum → (2) A - L = (3 + 4)
8. (a) (2) A - L → (2) B - L = (3 + 4)
   (b) Sub., (1) U - (1) L, Diff. → (2) A - L = (3 - 4)
9. (a) Bit 5 → (1) U
   (b) Bit 6 → (1) L
10. Add, ((2) B - U + 2 (B) - L), Sum → (3) - 1U = (1 + 2) + (3 + 4)
11. (a) (3) - 1U → (3) - 2U = (1 + 2) + (3 + 4)
    (b) Sub., ((2) B - U) - ((2) B - L), Diff. → (3) - 1U = (1 + 2) - (3 + 4)
12. (a) (2) A - U → (2) B - U = (1 - 2)
    (b) (2) A - L → (2) B - L = (3 - 4)
13. Add, 1 U + 1 L, Sum → (2) A - U = (5 + 6)
14. (a) (3) - 2U → (3) - 3U = (1 + 2) + (3 + 4)
    (b) (3) - 1U → (3) - 2U = (1 + 2) - (3 + 4)
    (c) Add, ((2) B - U) + ((2) B - L), Sum → (3) - 1U = (1 - 2) + (3 + 4)
15. (a) (3) - 3U → (3) - 4U = (1 + 2) + (3 + 4)
    (b) (3) - 2U → (3) - 3U = (1 + 2) - (3 + 4)
    (c) (3) - 1U → (3) - 2U = (1 - 2) + (3 - 4)
    (d) Sub., ((2) B - U) - ((2) B - L), Diff. → (3) - 1U = (1 - 2) - (3 - 4)
16. (a) (2) A - U → (2) B - U = (5 + 6)
    (b) Sub., (1) U - (1) L, Diff. → (2) A - U = (5 - 6)

TABLE 4.8
(Cont.)

17. Bit 7 → (1) U

18. Bit 8 → (1) L

19. Add, (1) U + (1) L, Sum → (2) A - L = (7 + 8)

20. (a) (2) A - L → (2) B - L = (7 + 8)

    (b) Sub., (1) U - (1) L, Diff. → (2) A - L = (7 - 8)

21. Add, (2) B - U + (2) B - L, Sum → (3) - 1L = ⌈(5 + 6) + (7 + 8)⌉

22. (a) (3) - 1L → (3) - 2L = ⌈(5 + 6) + (7 + 8)⌉

    (b) Sub., ((2) B - U) - ((2) B - L), Diff. → (3) 1 - U = ⌈(5 + 6) - (7 + 8)⌉

23. (a) (2) A - U → (2) B - U = (5 - 6)

    (b) (2) A - L → (2) B - L = (7 - 8)

24. (a) (3) - 2L → (3) - 3L = ⌈(5 + 6) + (7 + 8)⌉

    (b) (3) - 1L → (3) - 2L = ⌈(5 + 6) - (7 + 8)⌉

(c) Add, ((2) B - U) + ((2) B - L), Sum → (3) - 1L = ⌈(5 - 6) + (7 - 8)⌉

25. (a) (3) - 3L → (3) - 4L = ⌈(5 + 6) + (7 + 8)⌉

    (b) (3) - 2L → (3) - 3L = ⌈(5 + 6) - (7 + 8)⌉

    (c) (3) - 1L → (3) - 2L = ⌈(5 - 6) + (7 - 8)⌉

    (d) Sub., ((2) B - U) - ((2) B - L), Diff. → (3) - 1L = ⌈(5 - 6) - (7 - 8)⌉

26. Add, ((3) - 4U) + ((3) - 4L), Sum = (1 + 2 + 3 + 4) + (5 + 6 + 7 + 8)

    Sub., ((3) - 4U) - ((3) - 4L), Diff. = (1 + 2 + 3 + 4) - (5 + 6 + 7 + 8)

27. Compare magnitude of sum with threshold (Ø) in B reg. (Ref. Fig. 4.12)

28. If sum is larger:

    (a) Transfer to B Reg., output code bits i_1, i_2, i_3 = 000.
    (i_4 depends upon sign bit). Ref. Fig. 4.13

    (b) Difference magnitude is compared with new contents of
    B Reg. (= Sum). If difference is > sum, transfer
    Diff → B Reg., output code now: i_1, i_2, i_3 = 001,
    (i_4 depends upon sign bit).

TABLE 4.8
(Cont.)

29. If sum (step 28 (a)) is less than θ, Diff. is compared with θ. If Diff. $> \theta$, Diff. is transferred to B.Reg. Thus, based on sum, Diff. and sign of each, output code could be one of following:

i_1	i_2	i_3	i_4
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1

30. ③ - 3U → ③ - 4U = ⌈(1 + 2) - (3 + 4)⌋
③ - 3L → ③ - 4L = ⌈(5 + 6) - (7 + 8)⌋
③ - 2U → ③ - 3U = ⌈(1 - 2) + (3 - 4)⌋
③ - 2L → ③ - 3L = ⌈(5 - 6) + (7 - 8)⌋
③ - 1U → ③ - 2U = ⌈(1 - 2) - (3 - 4)⌋
③ - 1L → ③ - 2L = ⌈(5 - 6) - (7 - 8)⌋

31. Repeat steps 27, 28, and 29 for new contents of ③ - 4U and ③ - 4L. Output code can now be one of following:

i_1	i_2	i_3	i_4
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1

32. ③ - 3U → ③ - 4U = ⌈(1 - 2) + (3 - 4)⌋
③ - 3L → ③ - 4L = ⌈(5 - 6) + (7 - 8)⌋
③ - 2U → ③ - 3U = ⌈(1 - 2) - (3 - 4)⌋
③ - 2L → ③ - 3L = ⌈(5 - 6) - (7 - 8)⌋

TABLE 4.8
(Cont.)

33. Repeat steps 27, 28, and 29 for new contents of ③ - 4U and ③ - 4L. Output code can now be one of the following:

i_1	i_2	i_3	i_4
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1

34. ③ - 3U → ③ - 4U
③ - 3L → ③ - 4L

35. Repeat steps 27, 28 and 29 for new contents of ③ - 4U and ③ - 4L. Output code can now be one of the following:

i_1	i_2	i_3	i_4
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

The greatest magnitude of sum or difference transferred to the B register thus determines the output code symbols, i_1, i_2 and i_3 while the sign determines i_4. If no sum or difference was found to be greater than threshold, θ, then the output code is held at 0000 and discrete signal, erasure (Fig. 4.12 remains at a logic "1".

This completes the sequence for generation of a 4-bit "symbol" code from 8 received input bits. The process is repeated for the next 8 received input bits.

Figure 4.11

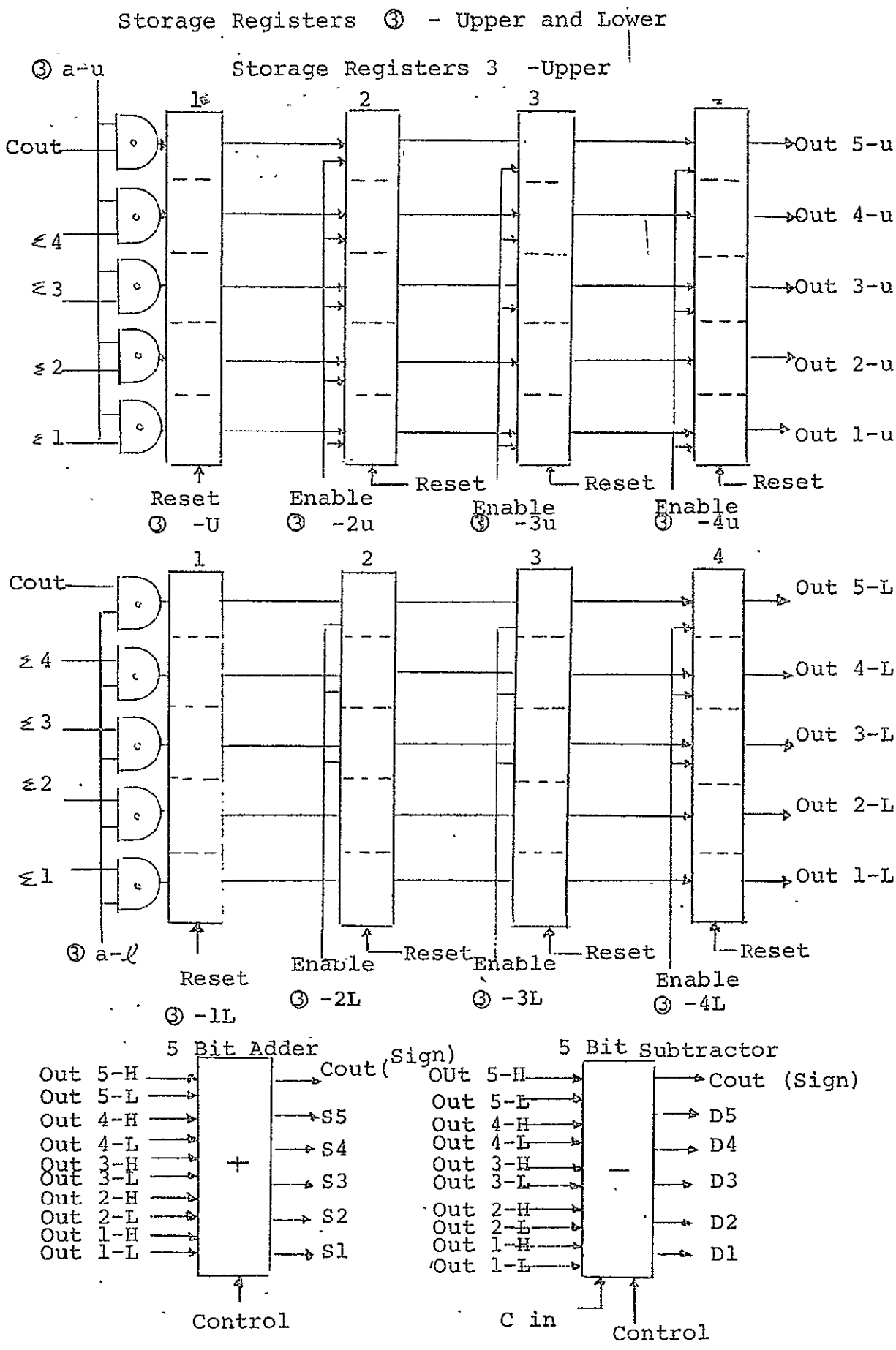

Storage Registers ③ - Upper and Lower
③ a-u
Storage Registers 3 -Upper
Cout
≤4
≤3
≤2
≤1
1
2
3
Out 5-u
Out 4-u
Out 3-u
Out 2-u
Out 1-u
Reset
Enable
Reset
Reset
Reset
③ -U
③ -2u
③ -3u
③ -4u
Cout
≤4
≤3
≤2
≤1
1
2
3
4
Out 5-L
Out 4-L
Out 3-L
Out 2-L
Out 1-L
③ a-ℓ
Reset
Enable
Reset
Enable
Reset
Reset
Enable
③ -1L
③ -2L
③ -3L
③ -4L
5 Bit Adder
Out 5-H
Out 5-L
Out 4-H
Out 4-L
Out 3-H
Out 3-L
Out 2-H
Out 2-L
Out 1-H
Out 1-L
Cout (Sign)
S5
S4
S3
S2
S1
Control
5 Bit Subtractor
OUt 5-H
Out 5-L
Out 4-H
Out 4-L
Out 3-H
Out 3-L
Out 2-H
Out 2-L
Out 1-H
Out 1-L
Cout (Sign)
D5
D4
D3
D2
D1
C in
Control

Figure 4.12

COMPARISON LOGIC

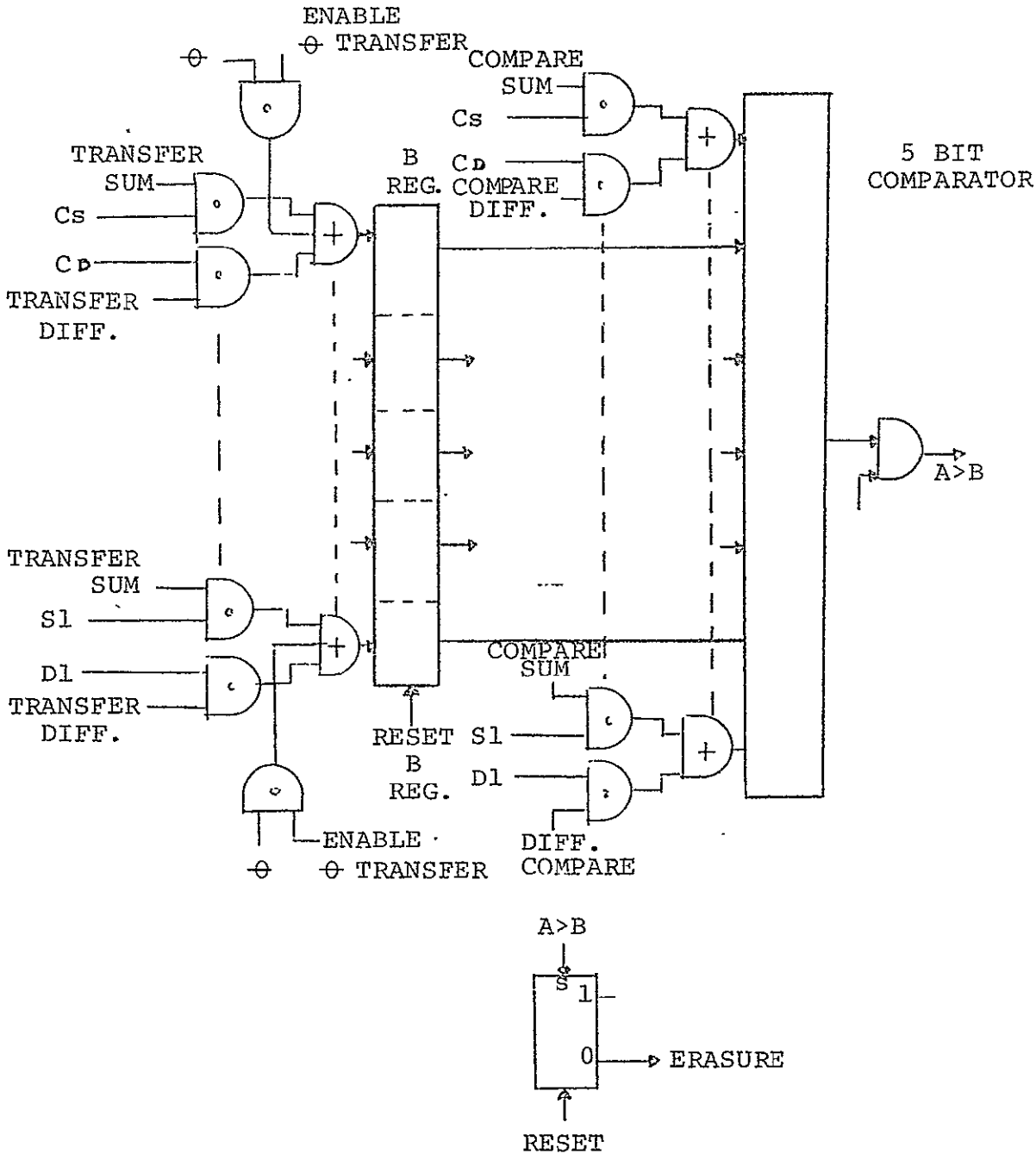
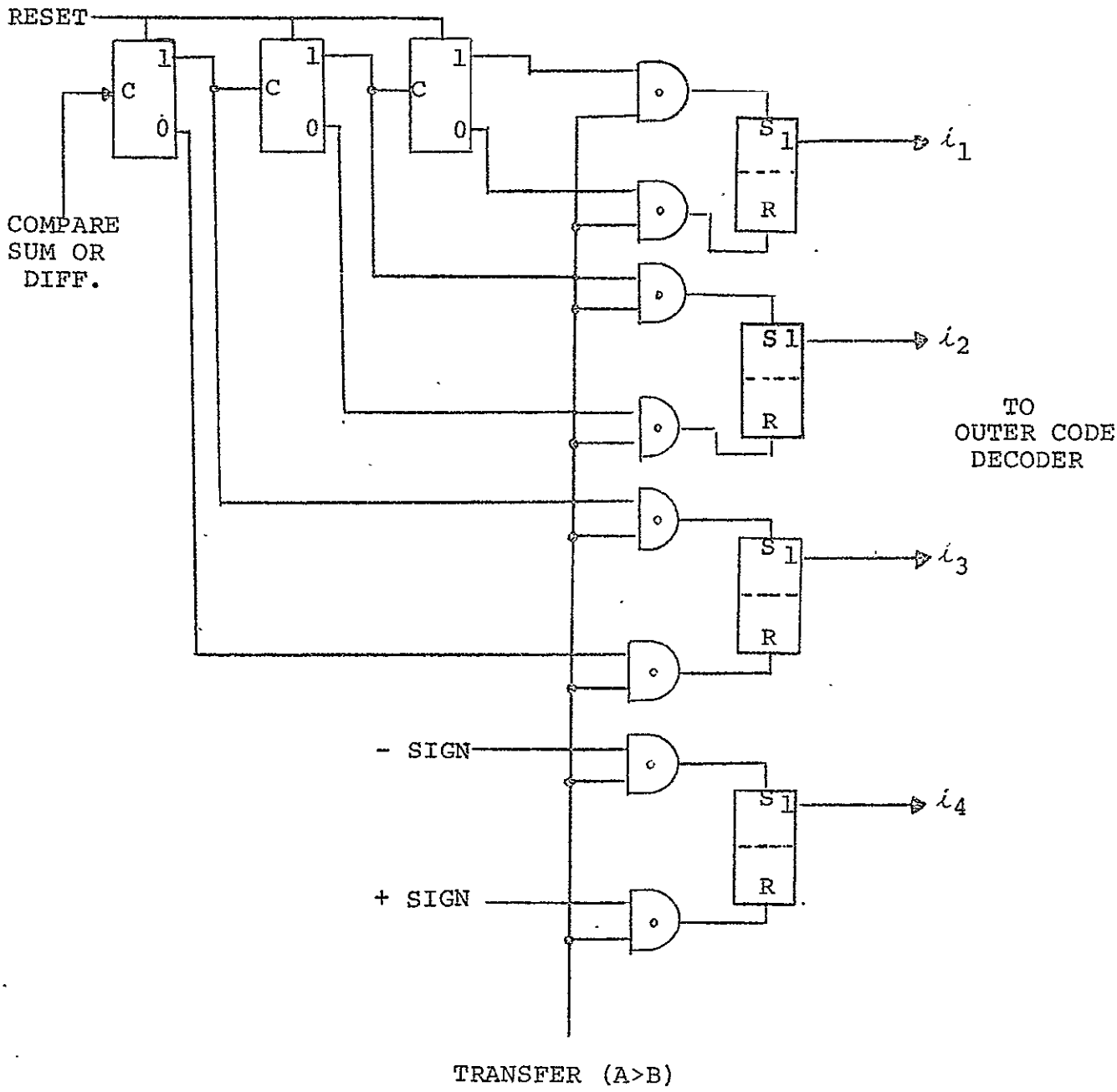


Figure 4.13

FOUR-BIT CODE "SYMBOL" GENERATOR



The two bits of original information received from the demodulator has expanded to six bits at the outputs of the 5 bit adder and subtractor (Figure 4.11.). An overflow bit is added at each arithmetic unit, while an additional bit for sign was also added at the first arithmetic stage. The magnitude comparisons are made only on five of the six bit sum and differences.

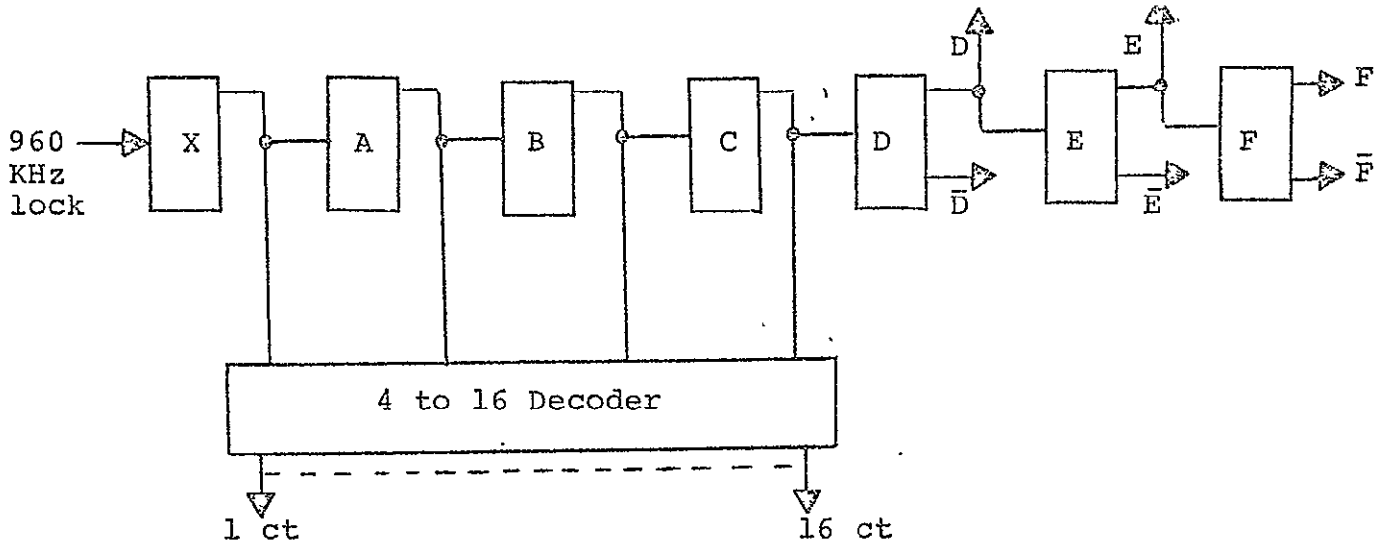
If the sign of the difference is negative (most significant bit = 1), the remaining five bits are complemented and incremented by one to restore the number from two's complement signed representation. The resulting number is then used for magnitude comparison.

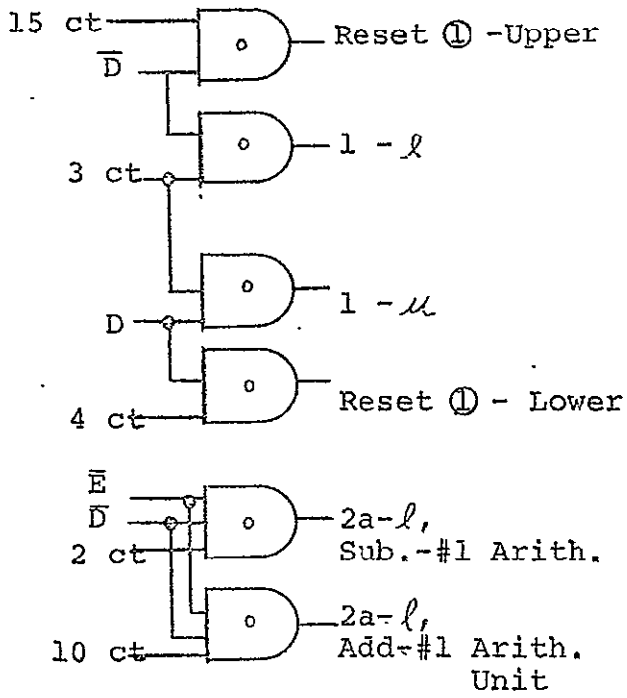
4.3.4.2.1. Timing and Control

A four bit output code "symbol" is generated for each group of eight received input code bits. This interval constitutes 2/15 of a millisecond time span. The interval between received input code bits (1/60th of a millisecond) is divided into 16 equal time pulses for basic logic operations (Figure 4.14). The combination of the time counter outputs to generate required control signals is illustrated in Figure 4.15 through 4.17. It should be pointed out that the decoder operation described in the preceeding paragraphs in approximately 42 microseconds. This allows 10 microseconds for a simultaneous addition and subtraction, then a compare and possible transfer of each to the B register. These operations can be performed with existing medium speed logic.

Figure 4.14

CONTROL SIGNAL GENERATION





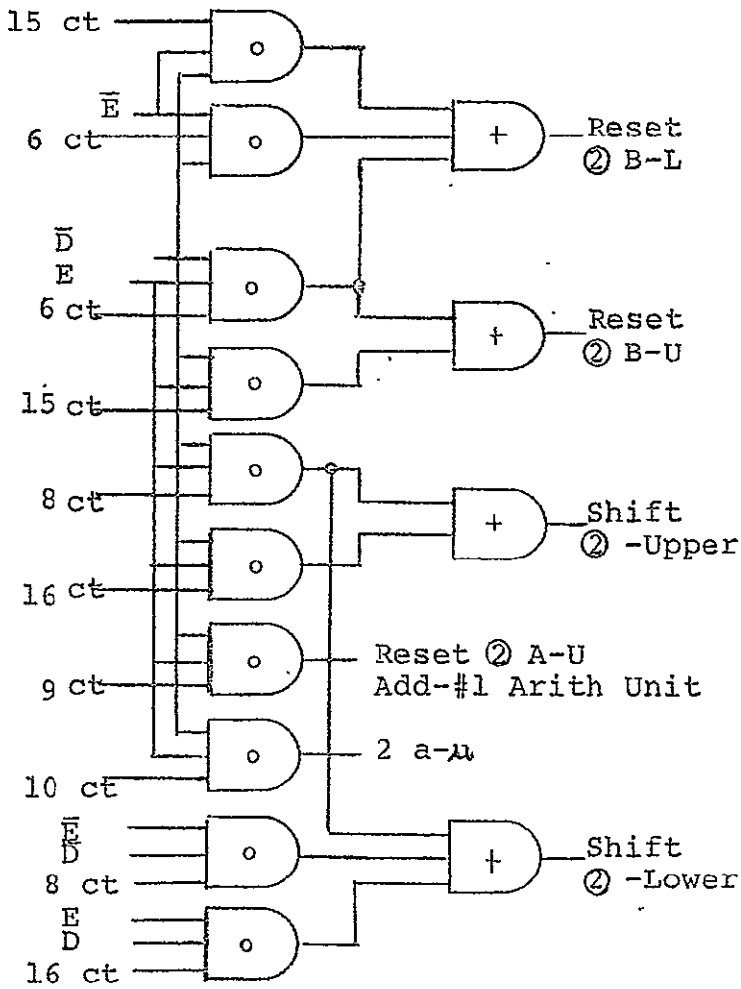


Figure 4.15

CONTROL SIGNAL GENERATION
(Cont)

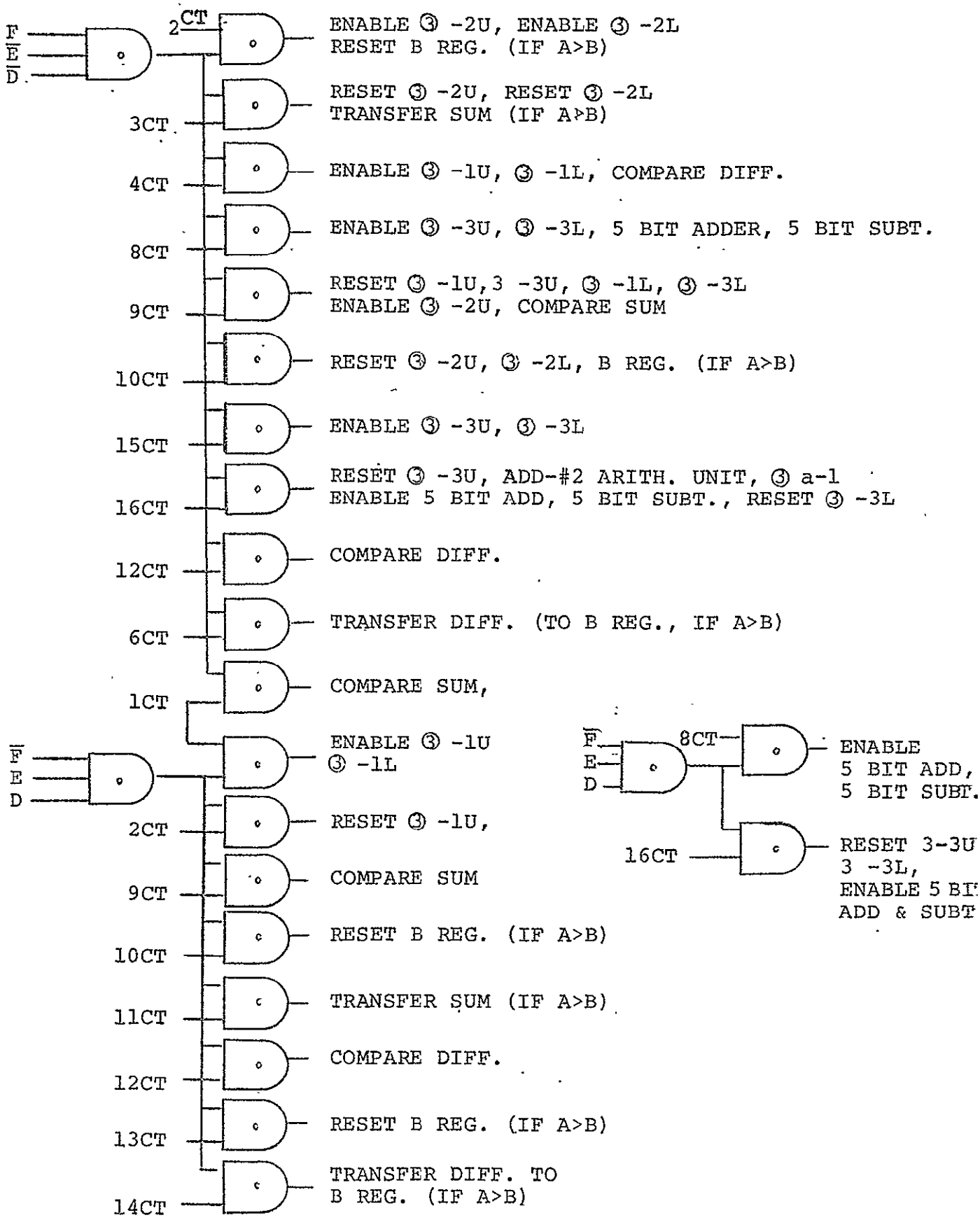


Figure 4.16

CONTROL SIGNAL GENERATION
(Cont)

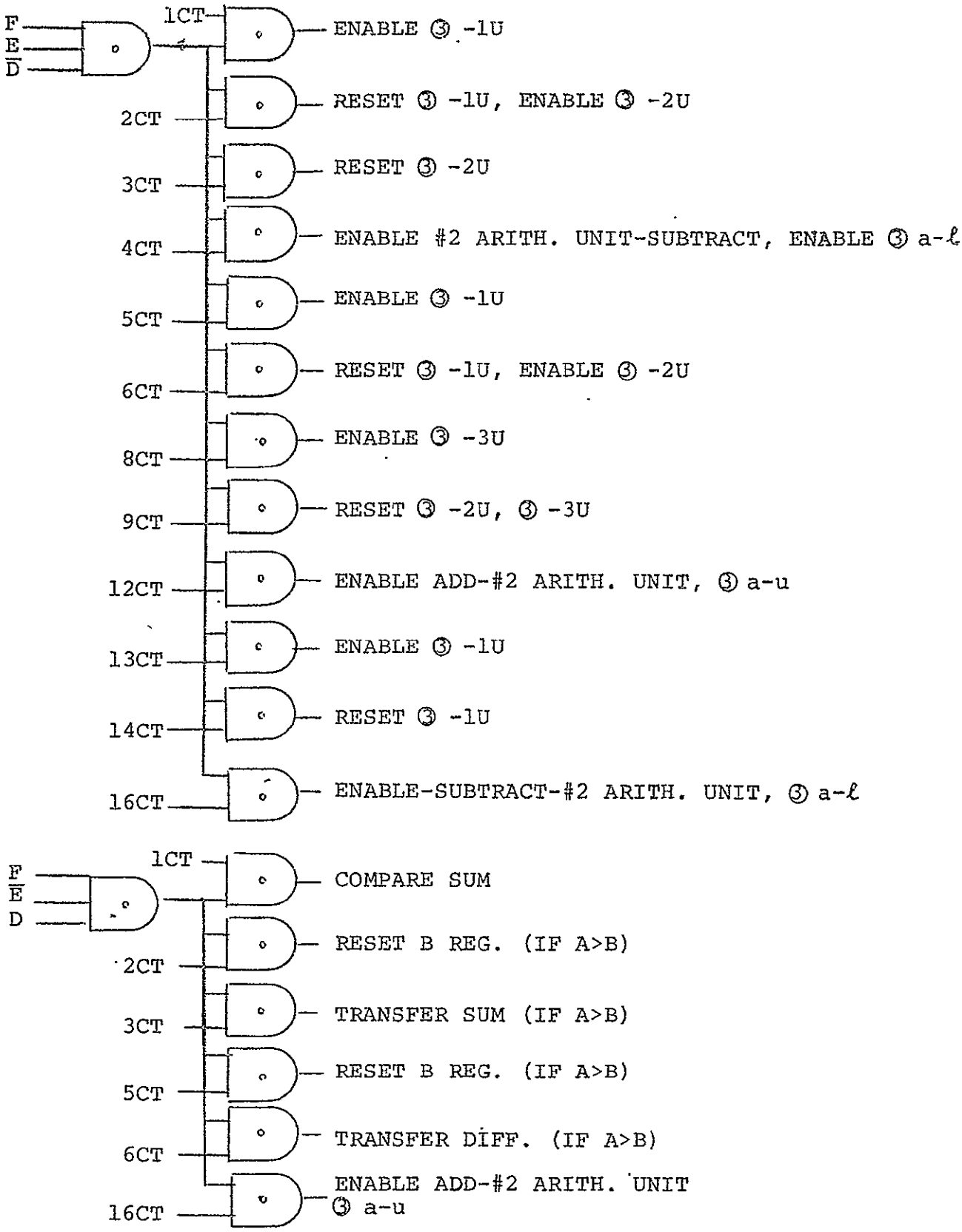
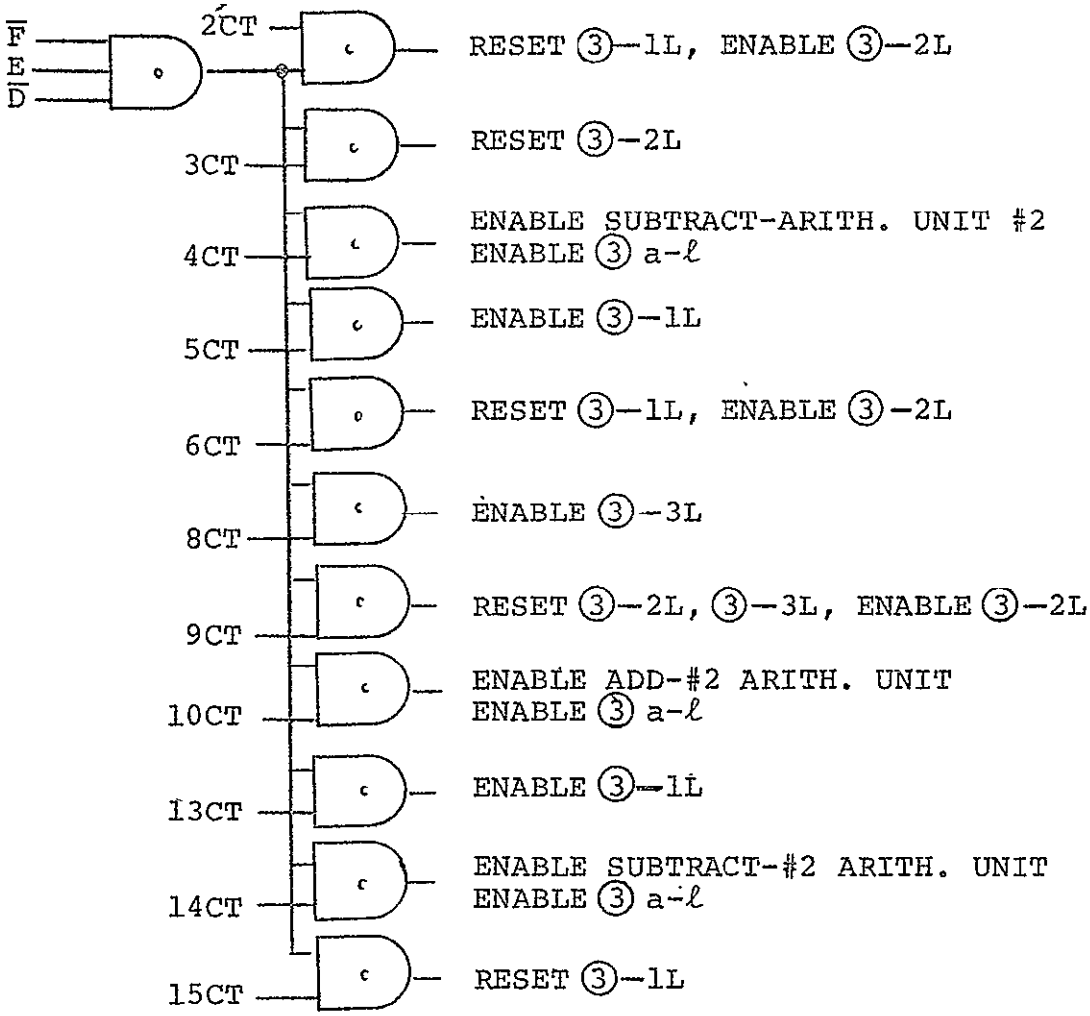


Figure 4.17

CONTROL SIGNAL GENERATION
(Cont)



4.3.4.2.2 Logic Packaging

The number of logic units required to implement the inner code decoder described has been determined to be approximately 51 dual-in packages. This compares with 24 packages required to implement the inner code decoder portion of the concatenated code decoder described in Section 4.3.3.3. The number of logic packages required to implement the "Green Machine" inner code decoder for received input code bits digitized to the number of bits, ℓ, equal 4, 6 and 8 have also been determined. The numbers are as follows:

ℓ = 2 bits -- 51 packages

ℓ = 4 bits -- 56 packages

ℓ = 6 bits -- 65 packages

ℓ = 8 bits -- 74 packages

Based on the packaging methods proposed for the breadboard version of the digital command controller and various types of decoders, the following table 4.9 has been prepared. This table compares the complete decoder packaging requirements for the different lengths, ℓ, of digitized input bits, versus the previous concatenated decoder. It also compares the logic board requirements for the same configurations.

TABLE 4.9

	Previous Decoder	Decoder with "Green Machine" inner Decoder ℓ = 2 Bits	ℓ = 4 Bits	ℓ = 6 Bits	ℓ = 8 Bits
1. No. of Packages	65	92	97	106	115
2. No. of Logic Boards	2	2	2	2	2

4.3.4.3. Error and Rejection Probabilities Using a Green Machine Decoder

In the following discussion it is assumed that the input samples to the Green Machine decoder are unquantitized (i.e. that the number 2^{ℓ} of quantization levels is infinite.) In practice, of course, ℓ will be finite, although experimental evidence indicates that for $\ell \geq 8$ the quantization effect is entirely negligable. In any event, the error probabilities derived below will provide a bound on the results actaully attainable.

Let R denote the ratio of the signal energy perinformation bit to the noise spectral density and let $R^* = \frac{k_{rs}}{n_{rs}} \log_2 m$, with k_{rs} and n_{rs} denoting the dimension and word length of the Reed-Solomon outer code and m the number of words in the biorthogonal inner code. (In the case of interest here, $n_{rs} = 15$, $k_{rs} = 9$, and $m = 16$.) Then, if a word is identified as an erasure unless the magnitude of the largest output of the Green Machine exceeds $\theta\sqrt{R^*}$, the erasure probability is

$$q = \left[\operatorname{erf}(\theta\sqrt{R^*})\right]^{\frac{m}{2}-1} \left\{ \tfrac{1}{2}\left[\operatorname{erf}\left((1+\theta)\sqrt{R^*}\right) - \operatorname{erf}\left((1-\theta)\sqrt{R^*}\right)\right]\right\} \quad (2)$$

as is readily verified. Similarly, and under the same conditions, the probability of an error is

$$p = 1-q-\frac{1}{\sqrt{\pi}}\int_{-(1-\theta)\sqrt{R^*}}^{\infty} e^{-\eta^2}\left[\operatorname{erf}(\eta+\sqrt{R^*})\right]^{\frac{m}{2}-1} d\eta \quad (3)$$

Equations (2) and (3) have been evaluated on a digital computer for various values of R and 0. The resulting

The results are presented in Tables 4.11 and 4.12. The first table lists the results when the Reed-Solomon outer decoder is designed only to detect errors and erasures; the second when it is designed, in addition, to correct up to two erasures.

It should be emphasized that these results do not take into account the effects of quantizing the Green Machine inputs to a finite number of levels. Past experience indicates that 4 to 6 bit quantization (16 to 64 levels) has a negligible effect on the decoder performance. Coarser quantization (2 to 4 bits) would undoubtedly have a deleterious effect on the system performance although, in terms of the probability of a correct decision, the Green Machine should perform at least as well as the algebraic inner decoder discussed previously. (Note, however, that the functions of the threshold θ is different in the two cases, so the two sets of results will not in general show the same dependence on this parameter.)

P_e	P_r	R
	$\theta = 0$	
9.9999995D-01	5.1446123D-08	0.000000
3.7573582D-01	6.2373584D-01	.845000
5.7352030D-02	9.2946027D-01	1.280000
5.8615859D-06	5.6516791D-01	2.645000
1.7033364D-08	2.9138791D-01	3.380000
1.1338748D-13	5.9174542D-02	4.805000
4.7680694D-16	2.7440058D-02	5.445000
2.7227862D-21	4.9351309D-03	6.845000
3.6607591D-27	7.1644036D-04	8.405000
	$\theta = 0.2$	
9.9999995D-01	5.1446123D-08	0.000000
3.7564115D-01	6.2383054D-01	.845000
5.7312544D-02	9.2950058D-01	1.280000
5.8478012D-06	5.6520870D-01	2.645000
1.6981938D-08	2.9143450D-01	3.380000
1.1298268D-13	5.9194062D-02	4.805000
4.7511830D-16	2.7450678D-02	5.445000
2.7144205D-21	4.9374105D-03	6.845000
3.6623400D-27	7.1677445D-04	8.405000
	$\theta = 0.25$	
9.9999995D-01	5.1446123D-08	0.000000
3.7519969D-01	6.2427209D-01	.845000
5.7137817D-02	9.2968083D-01	1.280000
5.7831921D-06	5.6546492D-01	2.645000
1.6729921D-08	2.9174627D-01	3.380000
1.1081027D-13	5.9346426D-02	4.805000
4.6565481D-16	2.7539835D-02	5.445000
2.6621836D-21	4.9597687D-03	6.845000
3.6543267D-27	7.2070714D-04	8.405000
	$\theta = 0.3$	
9.9999995D-01	5.1446123D-08	0.000000
3.7363353D-01	6.2583870D-01	.845000
5.6522661D-02	9.3031892D-01	1.280000
5.5586664D-06	5.6662256D-01	2.645000
1.5847869D-08	2.9322743D-01	3.380000
1.0292903D-13	6.0153613D-02	4.805000
4.3052192D-16	2.8039359D-02	5.445000
2.4552216D-21	5.1029625D-03	6.845000
3.2850505D-27	7.5045109D-04	8.405000

Table 4.10Erasure and Error Probabilities

(Erasure Detecting Outer Decoder; Green Machine Inner Decoder)

(Continued)

P_e	P_r	R
	$\theta = 0.4$	
9.9999995D-01	5.1446123D-08	0.000000
3.5915264D-01	6.4032473D-01	.845000
5.1256852D-02	9.3584744D-01	1.280000
3.9698777D-06	5.8061586D-01	2.645000
1.0039577D-08	3.1223276D-01	3.380000
5.4731830D-14	7.2373032D-02	4.805000
2.1709572D-16	3.6323303D-02	5.445000
1.1553999D-21	8.0951815D-03	6.845000
1.4590133D-27	1.5922400D-03	8.405000
	$\theta = 0.5$	
9.9999995D-01	5.1446123D-08	0.000000
3.0619465D-01	6.9330717D-01	.845000
3.5537110D-02	9.5275151D-01	1.280000
1.3172494D-06	6.4117532D-01	2.645000
2.3692144D-09	3.9938940D-01	3.380000
7.7142339D-15	1.4077209D-01	4.805000
2.5530346D-17	8.8214309D-02	5.445000
9.8874553D-23	3.2852844D-02	6.845000

Table 4.10 (Continued)

P_e	P_r	R
	$\theta = 0$	
9.9999995D-01	5.1446123D-08	0.000000
3.7573582D-01	6.2373584D-01	.845000
5.7352030D-02	9.2946027D-01	1.280000
5.8615859D-06	5.6516791D-01	2.645000
1.7033364D-08	2.9138791D-01	3.380000
1.1338748D-13	5.9174542D-02	4.805000
4.7680694D-16	2.7440058D-02	5.445000
2.7227862D-21	4.9351309D-03	6.845000
3.6607591D-27	7.1644036D-04	8.405000
	$\theta = 0.2$	
9.9999995D-01	5.1446123D-08	0.000000
3.7574179D-01	6.2372972D-01	-.845000
5.7355498D-02	9.2945092D-01	1.280000
5.8658277D-06	5.6504633D-01	2.645000
1.7055352D-08	2.9128286D-01	3.380000
1.1365608D-13	5.9145276D-02	4.805000
4.7811474D-16	2.7426360D-02	5.445000
2.7316199D-21	4.9329678D-03	6.845000
3.6826791D-27	7.1620554D-04	8.405000
	$\theta = 0.25$	
9.9999995D-01	5.1446123D-08	0.000000
3.7576277D-01	6.2370803D-01	.845000
5.7378727D-02	9.2940225D-01	1.280000
5.8925355D-06	5.6447681D-01	2.645000
1.7203138D-08	2.9076600D-01	3.380000
1.1577067D-13	5.8986866D-02	4.805000
4.8921034D-16	2.7348868D-02	5.445000
2.8195171D-21	4.9193336D-03	6.845000
3.8775048D-27	7.1452133D-04	8.405000
	$\theta = 0.3$	
9.9999995D-01	5.1446123D-08	0.000000
3.7586882D-01	6.2359940D-01	-.845000
5.7475408D-02	9.2921626D-01	1.280000
6.0144086D-06	5.6246418D-01	2.645000
1.7918132D-08	2.8890898D-01	3.380000
1.2744452D-13	5.8389680D-02	4.805000
5.5480937D-16	2.7049099D-02	5.445000
3.4316469D-21	4.8629712D-03	6.845000
4.9860285D-27	7.0691755D-04	8.405000

Table 4.11 Erasure and Error Probabilities

(Erasure Detecting and Two-erasure

Correcting Outer Decoder; Green Machine Inner Decoder)

(Continued)

P_e	P_r	R
	$\theta = 0.4$	
9.9999995D-01	5.1446123D-08	0.000000(
3.7707387D-01	6.2237097D-01	.845000(
5.8588612D-02	9.2729821D-01	1.280000(
7.5469196D-06	5.4575550D-01	2.645000
2.7853997D-08	2.7381744D-01	3.380000
3.4732477D-13	5.3535852D-02	4.805000
2.0358520D-15	2.4574537D-02	5.445000
2.5873934D-20	4.3685690D-03	6.845000
9.1941914D-26	6.3302807D-04	8.405000
	$\theta = 0.5$	
9.9999995D-01	5.1446123D-08	0.000000
3.8258685D-01	6.1675889D-01	.845000
6.3188640D-02	9.1957811D-01	1.280000(
1.2599135D-05	4.9572665D-01	2.64500(
6.0603995D-08	2.3229876D-01	3.38000(
1.2576967D-12	4.1114605D-02	4.80500(
9.3511133D-15	1.8283422D-02	5.44500(
2.0667350D-19	3.0862901D-03	6.04500(
1.4317704D-24	4.2965217D-04	8.40500(

Table 4.11 (Continued)

A comparison of the results tabulated in Tables 4.10 and 4.11 with previous results using an algebraic rather than a Green Machine decoder (cf. Table 4.5) leads to the following conclusions: * (1) The Green Machine decoder does generally result in superior performance measured in terms of the probability of a correct decision. (2) On the other hand, the algebraic inner decoder is more versatile in its ability to convert potential errors into rejections. At low signal-to-noise ratios, in fact, the error probability using the Green Machine is actually larger than that using the algebraic approach under the same conditions. (The Green Machine rejection probability is, of course, smaller than that of its competitor in this case, but that is small consolation.) (3) The performance of the Green Machine concatenated decoder is essentially independent of whether the outer decoder is designed to correct up to two erasures or whether it is designed only to detect erasures. (Compare Tables 4.10 and 4.11). (It is also quite insensitive to the threshold θ at least for $\theta < 1/2$.)

---

* Note: The refinement of the error probability estimates discussed in Section 4.3.2 has not been taken into account in the tabulations in Tables 4.10 and 4.11 in order to facilitate comparison with the earlier results. The actual error probability will be smaller than those listed in Table 4.10 by a factor of approximately 10^{-7} and smaller than those listed in Table 4.11 by a factor of approximately 2×10^{-5}.

This last observation obviously suggests simplifying the outer decoder so that it only detects erasures. This reduces its gate count somewhat, partially compensating for the increase in the complexity of the Green Machine vis-à-vis the algebraic inner decoder. It has been determined that the simplified outer decoder could be implemented with 21 fewer dual-in-line packages than needed for the outer decoder described in section 4.3.3.4. Taking this into account modifies slightly the comparison between the concatenated decoder complexities as shown in Table 4.13 (cf. Table 4.9).

Table 4.13

	Previous Decoder	Decoder with "Green Machine" decoder and erasure detecting outer decoder			
No. of Packages		$\ell = 2$	$\ell = 4$	$\ell = 6$	$\ell = 8$
	65	71	76	85	94
No. of Logic Boards	2	2	2	2	2

In conclusion, it appears that the advantage offered by the Green Machine approach is marginal, but then so is the increase in complexity of the associated equipment. One consideration which might tip the scales in favor of one of these two approaches is the relationship between the error probability and the signal-to-noise ratio R when the threshold θ is allowed to vary with R. This relationship has already been investigated when the inner code is algebraically decoded (cf. Section 4.3.2), Tables 4.6a and 4.6b). Analogous results using a Green Machine inner decoder are presented in Tables 4.14 and 4.15. Comparing these two sets of tables leads only to the conclusion that, in general, a larger value of γ is needed to achieve the same result using a Green Machine. On the basis of the evidence thus far developed, therefore, to it appears that the choice as to whether to use a Green Machine or an algebraic inner decoder can be made almost arbitrarily. Both offer essentially the same performance for the same complexity.

Table 4.14 Green Machine Error and Erasure Probabilities

$(\theta = \gamma(1 + (1/50\ R))1/2$

P_e	P_r	R
	$\gamma = .500000$	
9.99999160D-01	8.43151280D-07	0.000000
3.01296960D-01	6.98257460D-01	.845000
3.47172240D-02	9.53650170D-01	1.280000
1.27629700D-06	6.43023790D-01	2.645000
2.29515480D-09	4.01527920D-01	3.380000
7.48649060D-15	1.42073080D-01	4.805000
2.48020080D-17	8.91309290D-02	5.445000
9.02592070D-23	3.32430930D-02	6.845000
2.18056880D-28	1.15908490D-02	8.405000
	$\gamma = .700000$	
9.99988620D-01	1.13839130D-05	0.000000
7.38216630D-02	9.25895510D-01	.845000
2.29757890D-03	9.93355770D-01	1.280000
2.19512630D-09	9.16596050D-01	2.645000
8.35845310D-13	8.46731760D-01	3.380000
2.07901570D-19	6.94961610D-01	4.805000
2.39230810D-23	6.26012760D-01	5.445000
1.67847600D-29	4.85264140D-01	6.845000
	$\gamma = .800000$	
9.99967590D-01	3.24091680D-05	0.000000
1.50624150D-03	9.84883360D-01	.845000
1.45394210D-04	9.98540870D-01	1.280000
7.32961780D-15	9.86065080D-01	2.645000
7.58542740D-16	9.72883800D-01	3.380000
1.84835370D-23	9.39407270D-01	4.805000
7.86827740D-27	9.20158900D-01	5.445000
	$\gamma = .900000$	
9.99918700D-01	8.13018370D-05	0.000000
1.45764240D-03	9.98591940D-01	.845000
3.20422140D-06	9.99783360D-01	1.280000
3.96531740D-15	9.98932200D-01	2.645000
7.22308930D-25	9.98352310D-01	3.380000
8.45664830D-29	9.95839690D-01	4.805000
	$\gamma = 1.00000$	
9.99815690D-01	1.84308550D-04	0.000000
6.44114640D-05	9.99928360D-01	.845000
2.45834660D-08	9.99982310D-01	1.280000
3.31725780D-19	9.99974220D-01	2.645000
7.12562620D-25	9.99973580D-01	3.380000
5.44174804D-33	9.99972940D-01	4.805000

Table 4.15 Green Machine Error and Erasure Probalilities

$(\theta = \gamma(6.845/R)1/2$

P_e	P_r	R
	γ = .500000	
5.40907850D-23	1.00000000D+00	0.000000
9.2003423D-14	1.0000000D+00	.845000
2.9213360D-12	9.9999987D-01	1.280000
6.7150661D-12	9.8548530D-01	2.645000
4.7317427D-13	8.6479418D-01	3.380000
1.7214811D-16	3.3973847D-01	4.805000
2.4660019D-18	1.7561180D-01	5.445000
9.8874553D-23	3.2852844D-02	6.845000
3.2922211D-28	4.1172201D-03	8.405000
	γ = .700000	
1.6915790D-50	1.0000000D+00	0.000000
1.2011138D-32	1.0000000D+00	.845000
2.9158700D-36	1.0000000D+00	1.280000
2.7742046D-25	9.9999995D-01	2.645000
1.5691966D-24	9.9996335D-01	3.380000
3.9337131D-25	9.7333590D-01	4.805000
5.7331474D-26	8.8782297D-01	5.445000
1.1636530D-28	4.8[illegible]567D-01	6.845000
2.3379744D-28	1.3556306D-01	8.405000
	γ = .800000	
2.4143408D-67	1.0000000D+00	0.000000
5.7251051D-46	1.0000000D+00	.845000
6.0994793D-42	1.0000000D+00	1.280000
2.1105874D-34	1.0000000D+00	2.645000
7.7027962D-33	9.9999999D-01	3.380000
1.1133278D-31	9.9967134D-01	4.805000
5.8247670D-31	9.9517007D-01	5.445000
	γ = .900000	
2.9974067D-86	1.0000000D+00	0.000000
3.0519583D-56	1.0000000D+00	.845000
1.4094855D-57	1.0000000D+00	1.280000
1.0350451D-41	1.0000000D+00	2.645000
6.2953339D-42	1.0000000D+00	3.380000
5.7140427D-35	9.9999977D-01	4.805000
	γ = 1.000000	
3.1400131-107	1.0000000D+00	0.000000
1.9883997D-64	1.0000000D+00	.845000
1.2878630D-57	1.0000000D+00	1.280000
7.9001356D-47	1.0000000D+00	2.645000
1.5702916D-42	1.0000000D+00	3.380000
1.1627456D-38	1.0000000D+00	4.805000

4.4 CONVOLUTIONAL CODES

A number of algorithms for decoding convolutional codes have been explored. Until recently, the various sequential decoding algorithms have aroused the greatest interest; lately, the maximum-likelihood, or Viterbi decoding algorithm has received considerable attention. Sequential decoders are generally too complex and, because of their variable decoding rate, too difficult to interface with to be considered for the application of interest here. Viterbi decoders, however, are potentially applicable if acceptable error probabilities can be attained with moderately short constraint length codes. Investigation of this possibility is the subject of the next several sections of this report. The next section briefly describes the decoding algorithm is translated into hardware and the number of packages needed to implement such a decoder is determined as a function of the constraint length, the information rate, and the number of receiver quantization levels. Finally, in Section 4.4.3 the error probabilities attainable using this approach are estimated as a function of these same parameters.

4.4.1 The Viterbi Algorithm

The principle of the Viterbi decoding algorithm is to test the received sequence against all possible code sequences, rejecting a particular sequence only when its associated likelihood is guaranteed to be, and to remain, at least as small as some still contending sequence. It is easily verified that this can be done by retaining only 2^{k-1} sequences in contention at any time, where K is the code

sequences in contention at any time, where K is the code constraint length measured in information bits.

A flow diagram of this algorithm is shown in Figure 4.18. The ΔS_1 terms are monotone functions of the correlations between the V successive channel symbols representing an information bit (code rate = 1/V) and each of the 2^K binary V-tuples which could possibly have been transmitted. (e.g. $\Delta S_i = X_1 + \overline{X}_2 + \overline{X}_3 + \ldots + X_V$ with X_i a quantized ℓ-tuple representing a channel symbol and $\overline{X}_i$ its complement.) The terms SR_j and A_j represent the $j^{\underline{th}}$ 5K bit shift-register and the $j^{\underline{th}}$ accumulator, respectively, and B_j and S_j their respective contents. The expressions B_j0 and B_j1 indicate the contents of B_j augmented by adjoining, respectively, a zero and a one on the right of B_j. Finally, ℓ represents the number of bits used to represent each channel symbol; i.e. the detector output is quantized to 2^ℓ levels.

With these definitions, the flow diagram of figure 4.18 should be self-explanatory. (The parenthetical word "later" included at various steps in the algorithm refers to the fact that the new contents of a shift register or an accumulator are not to be read in until the present contents are no longer needed. This delayed read-in is accomplished in practice by duplicating all or part of the associated registers. In any event, the terms B_j and S_j always refer to the contents of the associated registers at the beginning of the current iteration.)

It is not difficult to verify that the algorithm thus represented does indeed reject only those sequences determined to be less likely than at least one retained reference.

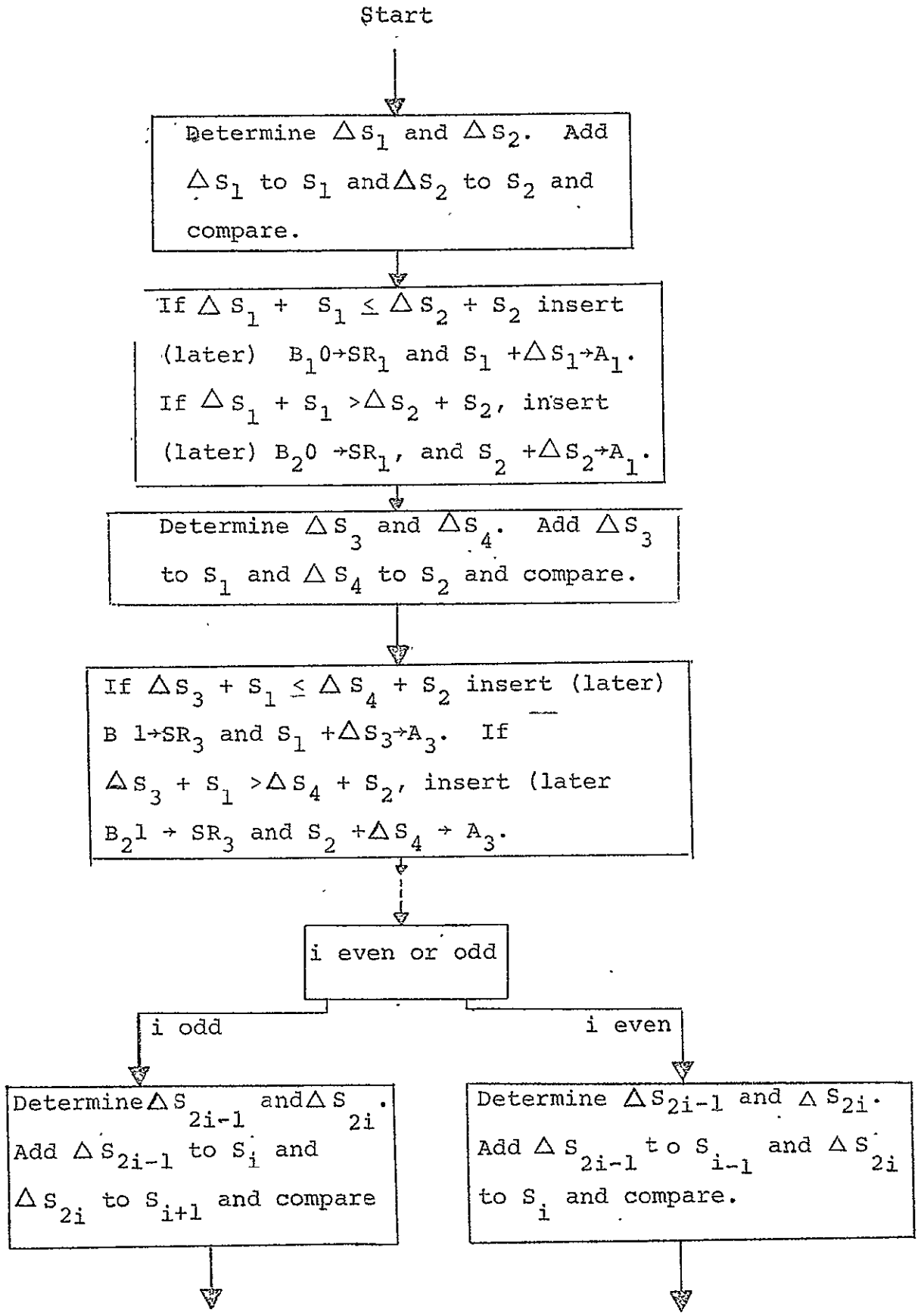


Figure 4.8. Maximum-likelihood Decoding Algorithm Flow Chart

(continued)

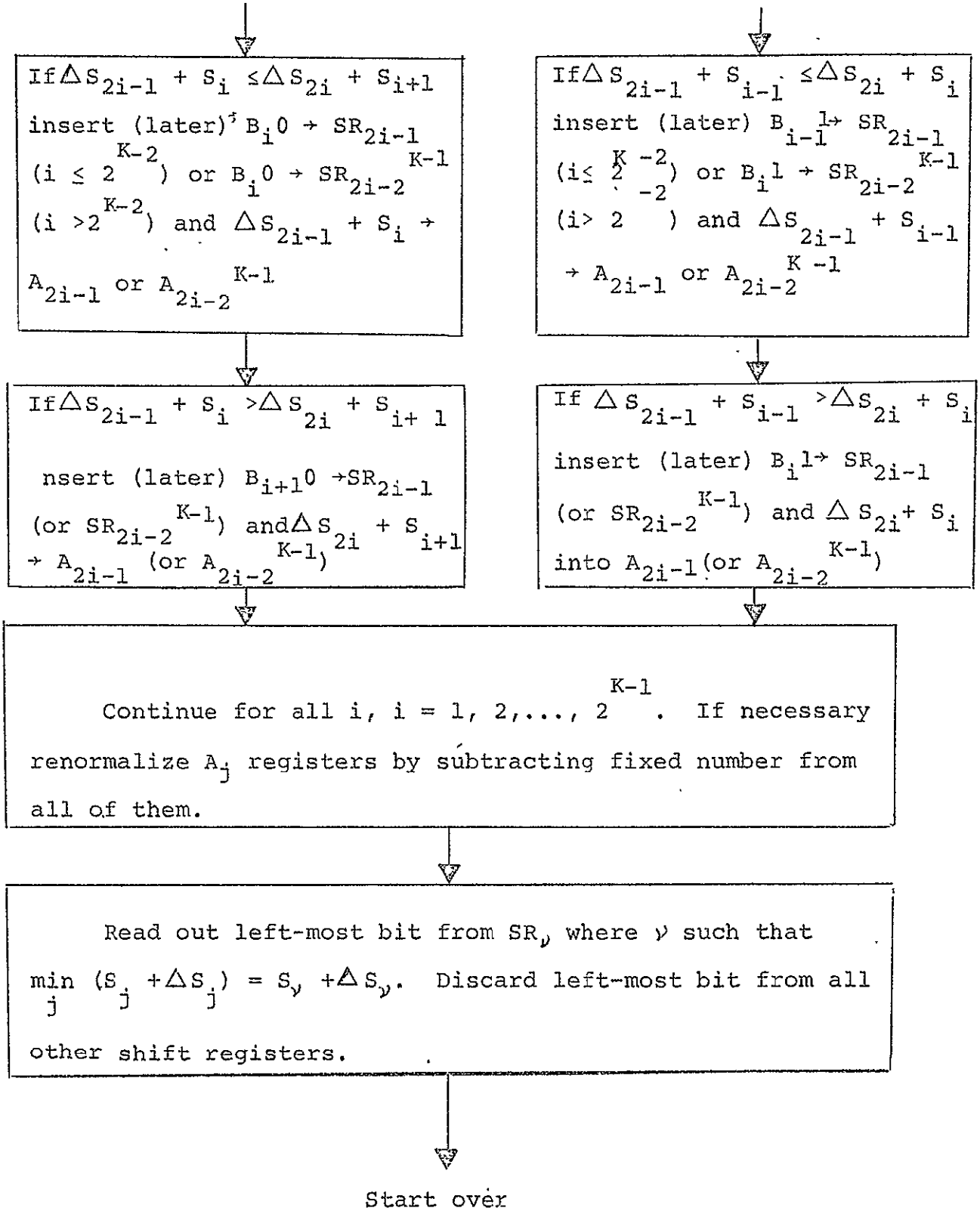


Figure 4.8 Maximum-likelihood Decoding Algorithm Flow Chart

4.4.2 Implementation of the Maximum Likelihood Concolutional Decoding Algorithm

4.4.2.1 General Description

Figure 4.19 is the block diagram for the maximum liklihood convolutional decoding algorithm implementation. The decoder input consists of $V\ell$-bit binary numbers representing a received code bit from the quantizer where the term V refers to the number of input channels.

The ΔS generator receives the quantized information bits and converts them to 2^K arithmetic-sum increments, ΔS (cf section 4.4.1). Starting with i = 1, the ΔS_{2i-1} and ΔS_{2i} outputs are presented to adders 1 and 2, respectfully. These numbers are added to sums S_i and $S_i + 1$ which are stored in accumulation memories Ax-odd and even. Accumulation memories Ax-odd and even and Ay-odd and even are four separate memories with individual address selection. Dotted lines indicate that memory utilization alternates with each cycle of decoder operation, a cycle ending with the generation of an information bit ($i = 2^{K-1}$). Memories Ax-odd and Ay-odd contain odd locations one through i - 1, while memories Ax-even and Ay-even contain even locations two through i; max $i = 2^{K-1}$.

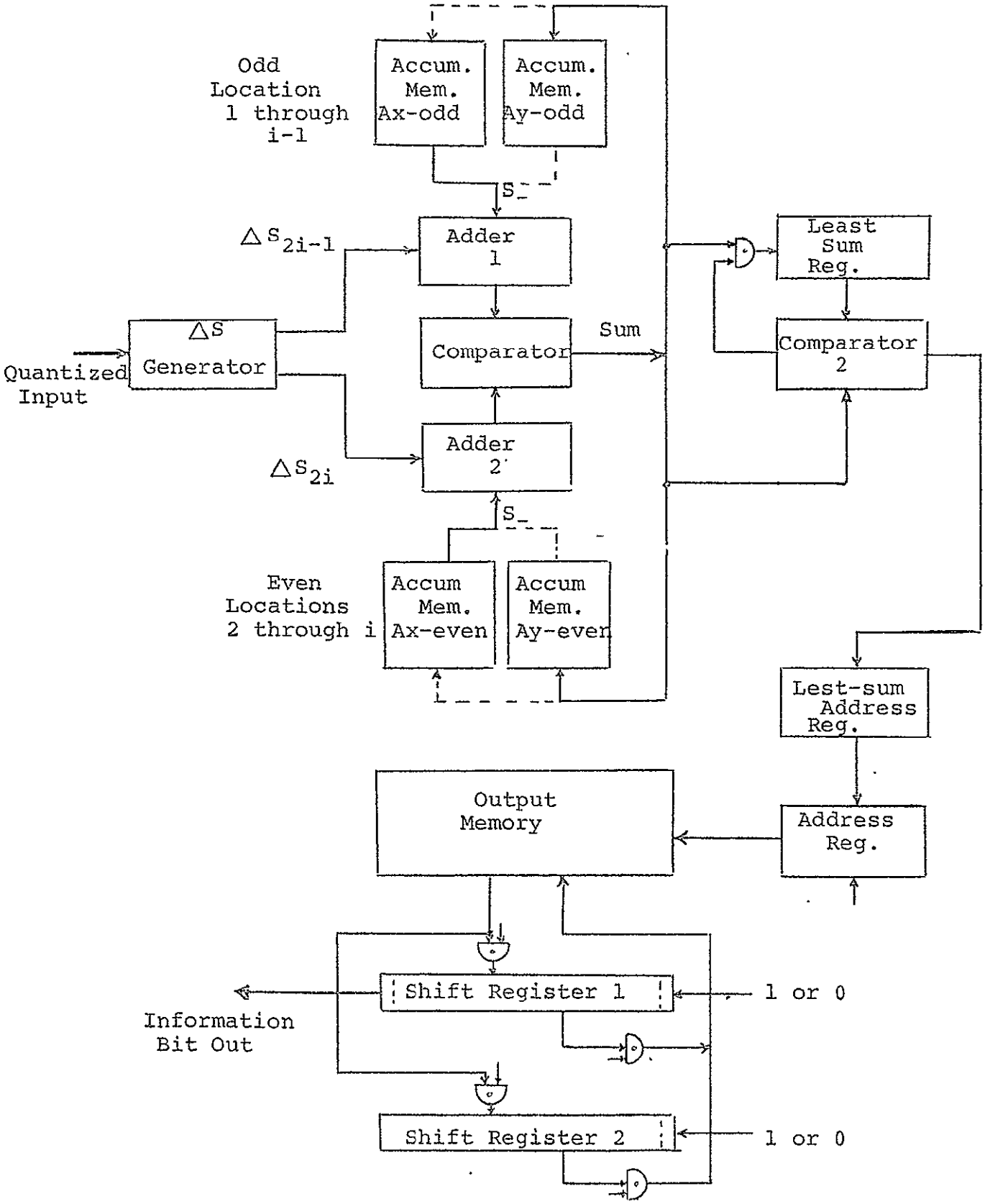
A comparison in performed on the outputs of the two adders and the lesser sum is stored in memory Ay-odd, location 2i - 1, and the least sum register (L.S.R.).

At this time the contents of output memory locations. B_i and B_{i+1} are read into shift registers 1 and 2

Figure 4.19

Block Diagram Maximum Likelihood

Convolutional Decoder



respectfully, and shifted one place to the left. If the $\Delta S_{2i-1} + S_i$ sum is lesser, a logic "0" is loaded into the least significant bit position, and the contents returned to location B_i.

If the $\Delta S_{2i} + S_{i+1}$ sum is lesser, the contents of shift register 2 are shifted left one place, a logic 0 loaded into the least significant bit position, and the contents written into location B_i.

The sequence now repeats for i = 2. A ΔS_{2i-1} term is added to accumulated sum S_{i-1}, while ΔS_{2i} is added to S_i and the resulting sums compared. The lesser sum is stored in memory Ay-odd location 2i - 1, and is compared with the contents of the L.S.R. If the sum is less than the contents of the L.S.R., the sum is loaded into the L.S.R. Contents of output word memory location B_{2i-1} are read into shift register 2 if $S_{2i-1} + S_{i-1}$ is lesser sum. A logic "1" is loaded into the least significant bit of shift register 1 and the contents written into location B_{2i-1}.

When $\Delta S_{2i} + S_i$ is the lesser sum, the contents of B_{2i-1} are read into shift register 1. A logic "1" is loaded into the least significant bit position of shift register 2 and the contents written into location B_{2i-1}.

4.4.2.1.1 Output Bit Determination

The above sequence is repeated until i = 2^{K-1} at which time the output word memory locations corresponding to the least sum is read into shift register 1. The most significant bit of the register represents the information bit.

A new cycle is now started for generation of the next information bit. During this cycle, the accumulated sum terms, S, are read from memories Ay-odd and even, and the new sums stored in memories Ax-odd and even. It should be pointed out that new sums are always stored in odd locations (one through i - 1) first, then even locations (two through i).

4.4.2.1.2 Accumulated Sum Normalization

To prevent overflow during the successive addition operations, the new sums are normalized when the most significant bit in each sum generated during a cylce is a logic 1. When this condition is detected, the most significant bit if each S term read from Ax or Ay during the following cycle is set to a logic "0".

4.4.2.2 Detailed Sequence Operation

Table 4.16 defines the detailed sequence of operations required to generate one bit of information from quantized input bits, with K = 3 (cf. figure 4.19.)

Table · 4.16

Detailed Sequence of Operation, K = 3

i = 1

1. (a) Form ΔS_1 and ΔS_2 (Ref. Fig. 4.18)

   (b) Read C(Memory Ax-odd location, 1) = S_1

   (c) Read C(Memory Ax-even, location 2) = S_2

2. (a) Add: $\Delta S_1 + S_1$

   (b) Add: $\Delta S_2 + S_2$

   (c) Read C(Output Memory location 1) = $B_1 \rightarrow$ Shift Register 1

   (d) Set Shift Register 1 LSB = 0

   (e) Read C(Output Memory, location 2) = $B_2 \rightarrow$ Shift Register 2

   (f) Set Shift Register 2 LSB = 0

3. Compare: $\Delta S_1 + S_1$ and $\Delta S_2 + S_2$ --

   (a) If $\Delta S_1 + S_1 \leq \Delta S_2 + S_2$

   $\Delta S_1 + S_1 \longrightarrow$ Ay-odd, location 1

   $\longrightarrow$ L.S. Register

   Increment Most Significant Bit Counter (MSBCTR) if MSB = 1

   c(Shift Register 1) = B1 0→Output Memory location 1

   Set Least Sum Address (L.S.A.) = 1

   (b) If $\Delta S_2 + S_2 < \Delta S_1 + S_1$

   $\Delta S_2 + S_2 \longrightarrow$ Ay-odd, location 1

   $\longrightarrow$ L.S. Register

   Increment MSB CTR if MSB = 1

   c(Shift REgister 2) = B_2 0→Output Memory location 1

   Set L.S.A. = 1

---

i = 2

4. (a) Form $\triangle$S3 and $\triangle$S4

(b) Read c(Ax-odd,location 1) = S1

(c) Read c(Ax-even, location 2) = S2

5. (a) Add: $\triangle$S3 + S1

(b) Add: $\triangle$S4 + S2

6. Compare $\triangle$S3 + S1 and $\triangle$S4 + S2;

(a) If S3 + S1 $\leq$ S4 + S2;

$\triangle$S3 + S1 $\longrightarrow$ Ay-odd, location 3

Increment MSBCTR if MSB = 1

c(Output Memory Location 3) = B3 $\longrightarrow$ Shift Register 2

Set Shift Register 1 L.S.B. = 1

c(Shift Register 1) = B 1 $\longrightarrow$ Output Memory Location 3

1. Compare $\triangle$S3 + S1 and c(L.S.Register)

If $\triangle$S3 + S1 $\leq$ c(L.S.Register); $\triangle S_3 + S_1 \longrightarrow$ L.S. Register

Set L.S.A. = 3

(b) If $\triangle S_4 + S_2 < \triangle S_3 + S_1$

$\triangle S_4 + S_2 \longrightarrow$ Ay-odd,,location 3

Increment MSBCTR if MSB = 1.

c(Output Memory Location 3) = B3 $\longrightarrow$ Shift Register 1

c(Shift Register 2) = B21 $\longrightarrow$ Output Memory Location 3

1. Compare $\triangle S_4 + S_2$ and c(LS Reg.)

If $\triangle S_4 + S_2 \leq$ c(L.S. Register); $\triangle S_4 + S_2 \longrightarrow$ L.S.Reg.

Set L.S.A. = 3

7. (a) Form ΔS_5 and ΔS_6

(b) Read c(Ax-odd, location 3) = S_3

(c) Read c(Ax-even, location 4) = S_4

8. (a) Add: $\Delta S_5 + S_3$

(b) Add: $\Delta S_6 + S_4$

(c) c(Output Memory Location 4) = B4 $\longrightarrow$ Shift Register 2

9. Compare $\Delta S_5 + S_3$ with $\Delta S_6 + S_4$

(a) If $\Delta S_5 + S_3 \leq \Delta S_6 + S_4$

$\Delta S_5 + S_3 \longrightarrow$ Ay-even, location 2

Increment MSBCTR if MSB = 1

Set Shift Register 1 L.S.B. = 0

c(Shift Register 1) = B3 0 $\longrightarrow$ Output Memory, Location 2

1. Compare $\Delta S_5 + S_3$ and c(L.S.Register)

If $\Delta S_5 + S_3 \leq$ c(L.S. Register); $\Delta S_5 + S_3 \longrightarrow$ L.S. Reg.

Set L.S.A. = 2

(b) If $\Delta S_6 + S_4 \leq \Delta S_5 + S_3$

$\Delta S_6 + S_4 \longrightarrow$ Ay-even, location 2

Increment MSBCTR if MSB = 1

Set Shift Register 2 L.S.B. = 0

c(Shift Register 2) = B_4 0 $\longrightarrow$ Output Memory, Location 2

1. Compare $\Delta S_6 + S_4$ and c(L.S.Register)

If $\Delta S_6 + S_4 \leq$ c(L.S.Register); $\Delta S_6 + S_4 \longrightarrow$ L.S. Register

Set L.S.A. = 2

i = 4

10. (a) Form ΔS_7 and ΔS_8

(b) Read c(Ax-odd, location 3) = S_3

(c) Read c(Ax-even, location 4) = S_4

11. (a) Add: $\Delta S_7 + S_3$

(b) Add: $\Delta S_8 + S_4$

12. Compare $\Delta S_7 + S_3$ and $\Delta S_8 + S_4$

(a) If $\Delta S_7 + S_3 \leq \Delta S_8 + S_4$

$\Delta S_7 + S_3 \longrightarrow$ Ay-even, Location 4

Increment MSBCTR if MSB = 1

Set Shift Register 1 LSB = 1

c(Shift Register 1) = $B_3 1 \longrightarrow$ Output Memory, Location 4

1. Compare $\Delta S_7 + S_3$ and c(L.S. Register)

If $\Delta S_7 + S_3 \leq$ c(L.S. Register); $\Delta S_7 + S_3 \longrightarrow$ L.S.Register

Set LSA = 4

(b) If $\Delta S_8 + S_4 < \Delta S_7 + S_3$

$\Delta S_8 + S_4 \longrightarrow$ Ay-even, location 4

Increment MSBCTR if MSB = 1

Set Shift Register 2 LSB = 1

c(Shift Register 2) = $B_4 1 \longrightarrow$ Output Memory, Location 4

1. Compare $\Delta S_8 + S_4$ and c(L.S. Register)

If $\Delta S_8 + S_4 \leq$ c(L.S.Register); $\Delta S_8 + S_4 \rightarrow$ L.S. Register

Set L.S.A = 4

13.

c(Output Memory, Location = L.S.A.) ⟶ Shift Register 1

Output MSB of Shift Register 1 = Information Bit

14. Read c(MSBCTR)

If c(MSBCTR) = 4; Set Normalize Flag.

(Next cycle, MSB of S terms from Ay-odd and even will be set to logic "0" as they are read out of memory.)

4.4.2.3 Logic Implementation

The proposed implementation of the algorithm is essentially a serial configuration, with a small degree of parallel hardware for certain functions. A serial implementation greatly reduces the total logic requirements compared with a totally parallel configuration, but reduces the overall decoder speed of operation.

The incremental sums, ΔS, are generated two at a time to allow two new sums to be formed at the same time. This operation also requires duplicate adders and two memories, Ax-odd and Ax-even, for the S terms. An additional set of memories, Ay-odd and Ay-even, are utilized to allow the storage of new sums without buffering or serialized read-write operations. The use of semiconductor memories which provide up to 256 bits of storage in one dual-in-line package is an important factor in reducing the total logic requirements.

The configuration illustrated in the block diagram of Figure 4.19 can be utilized for values of K up to and including 9. The output memory is paralleled for the K = 10 configuration to save time required for storing contents of some locations in "spare" locations for later use.

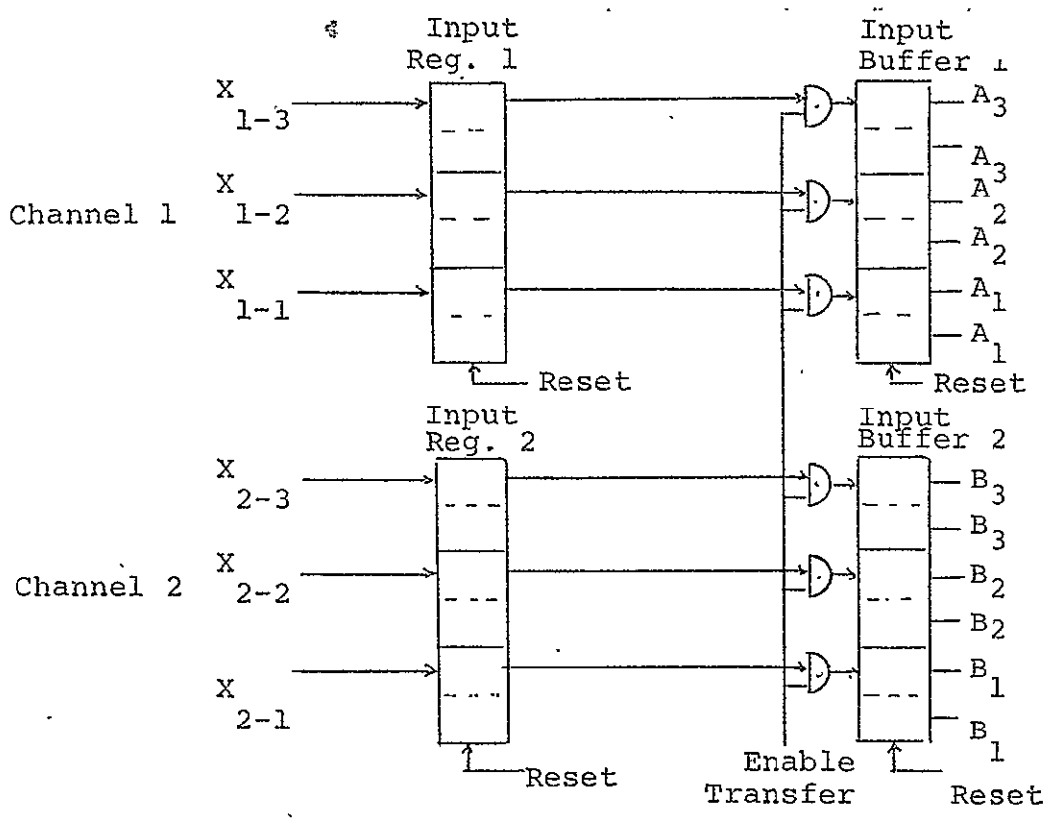
The logic which will be described here was selected to implement a decoder for illustrative purposes with K = 3, V = 2, and ℓ = 3.

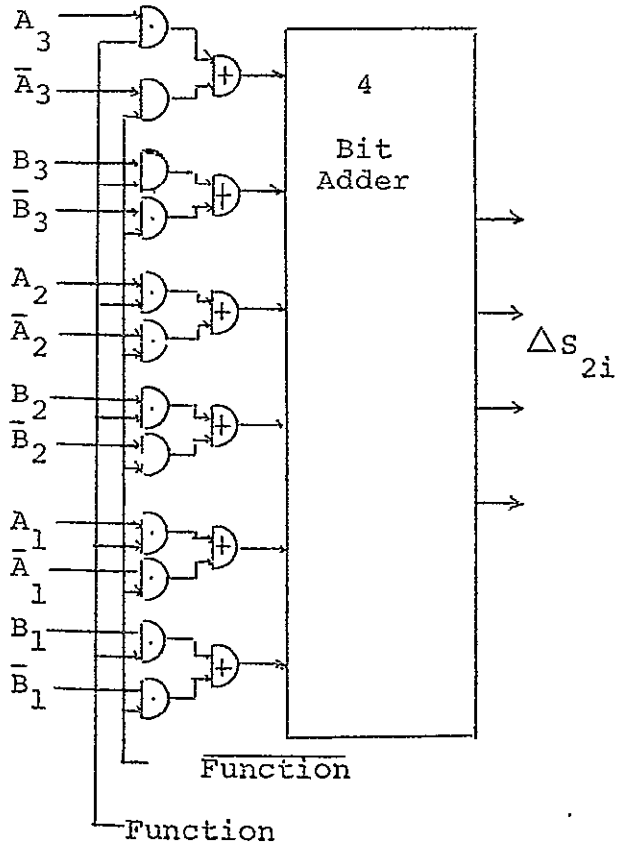
4.4.2.3.1 ΔS Generator

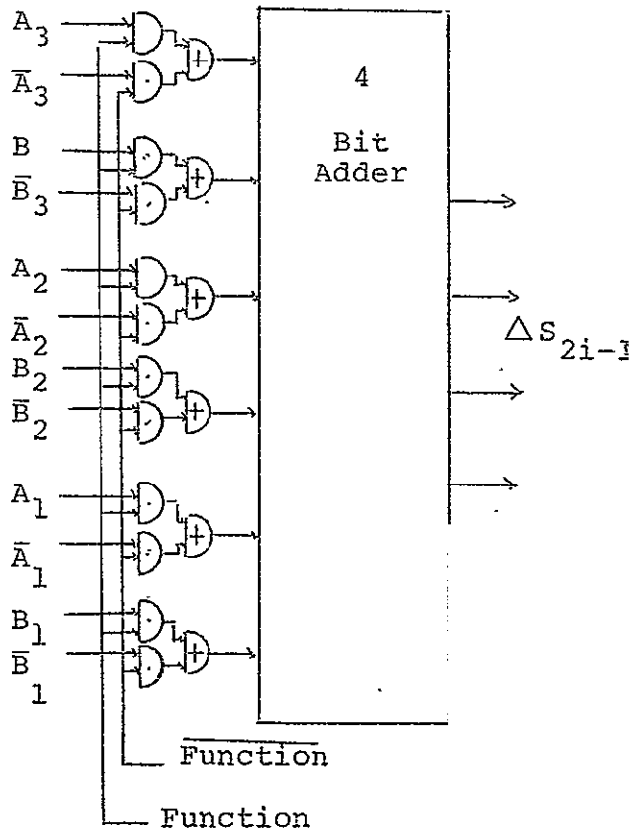
Figure 4.20 is an example of the logic required to implement the two terms ΔS_{2i-1} and ΔS_{2i} for a two

Figure 4.20

ΔS Generators







channel input (V = 2) with three bits per channel (ℓ = 3). Channel 1 and 2 information is received in input registers 1 and 2 respectively, then transferred to the input buffer from which the ΔS terms are formed. Input buffers are required since the channel 1 and channel 2 inputs are supplied sequentially rather together; both words must be present for generation of the ΔS terms.

The true and inverted functions are both supplied to the 4-bit adders to allow the generation of A + B, A + B', A' + B or A' + B' type terms, as required by the algorithm described in Section 2 of this report.

4.2.3.2 S + S Adders and Comparator

The ΔS + S adders are comprised of a 4-bit adder and a 2-bit adder connected together to form a 6-bit adder (Figure 4.21). It can be shown that a decoder implemented for K = 3, V = 2 and l = 3 requires 6 stages for the ΔS + S sum to ensure no overflow before normalization. The normalization procedure, as mentioned in 4.4.2.1 consists of a check on the most significant bit (MSB) of each stored sum. When the MSB of each sum (ΔS + S) stored during a cycle is a logic "1", then each MSB is changed to a logic "0" during the next cycle prior to addition with a ΔS term.

4.4.2.3.3 Accumulation Memories (Reference Figure 4.22)

Four separate memories are utilized for storing the accumulated sums (S) during the decoding process.

Figure 4.21

ΔS + S Adders and Comparator

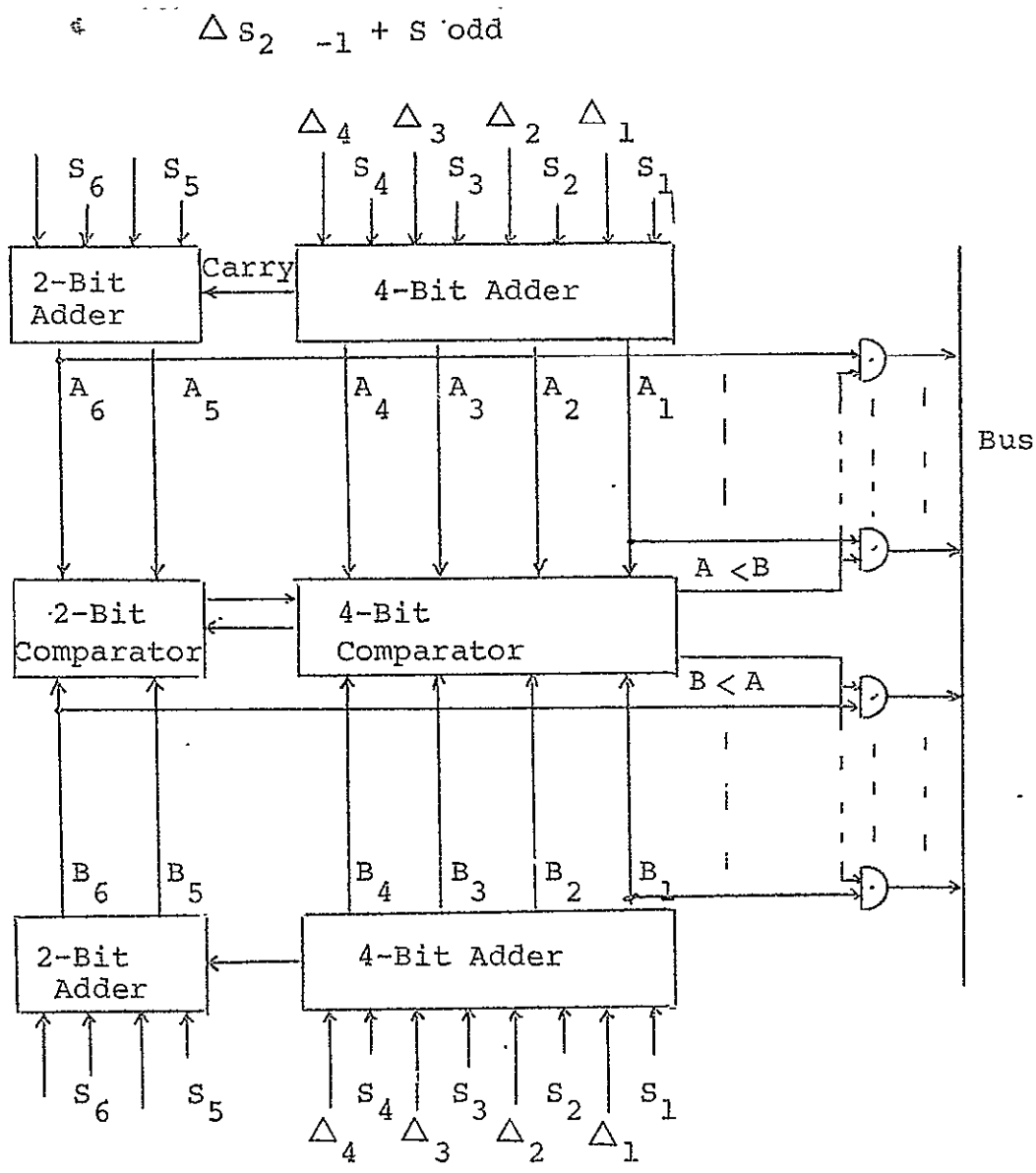
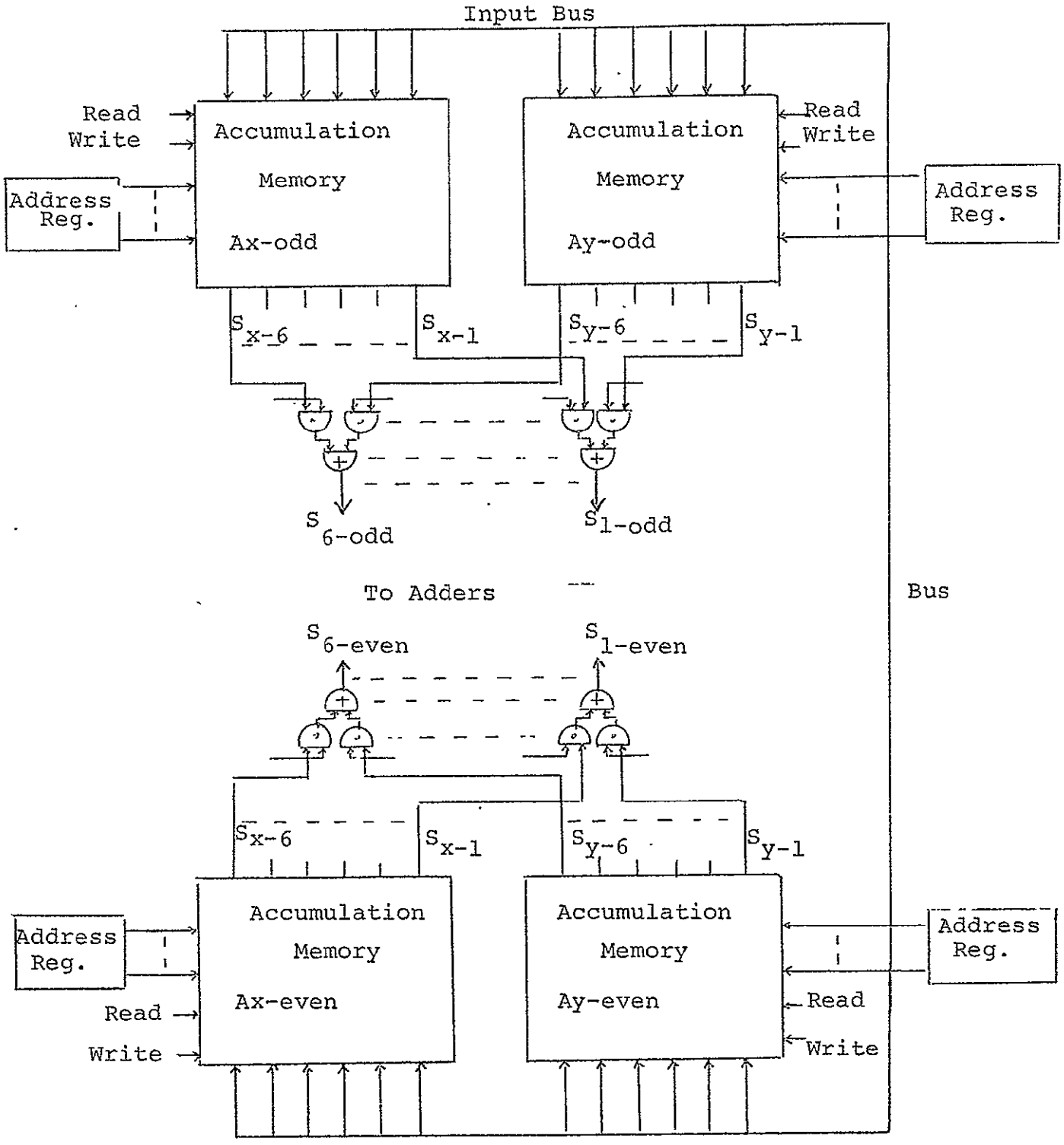


Figure 4.22

Accumulation Memories



The memories, Ax-odd, Ax-even, Ay-odd and Ay-even each consist of 2 word x 6 bit units. For the general case the memory size required is $2^{K-1}/2$ words by $\ell + 3$ bits. The savings in logic over the use of registers becomes appreciable as the value of K increases. As an example, for K = 8, 8 memory D.I.P.s (each 64 words x 4 bits) are sufficient to implement the accumulation memories. Using registers (8 bits per D.I.P.) the requirement would be 2^{K-1}, or 128 D.I.P.s.

A memory access time of approximately 10 nanoseconds is required for the decoder configuration with K = 10. Memory packages are available in this speed range using current mode logic for address logic and storage cells. These units are available only in the 16 word x 4 bit configuration, however, requiring a relatively greater number of packages.

4.4.2.3.4 Least Sum Address Generation

The least sum address logic (Figure 4.23) provides a means of recording the accumulated sum address which contains the smallest sum. This same address is then used to select the location in output memory from which the output bit (most significant bit) is read.

As each sum is stored and compared with the contents of the least sum register (LS Reg.), the inputs to the encoder are sequentially enabled (via control). The encoder generates a 3-bit code which is applied to the

Figure 4.23

Least Sum Address Generation

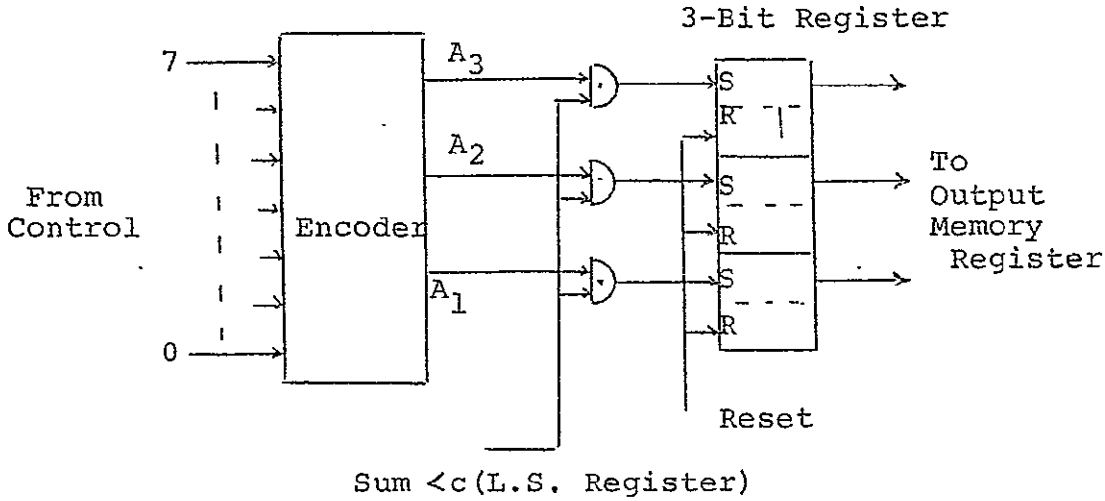
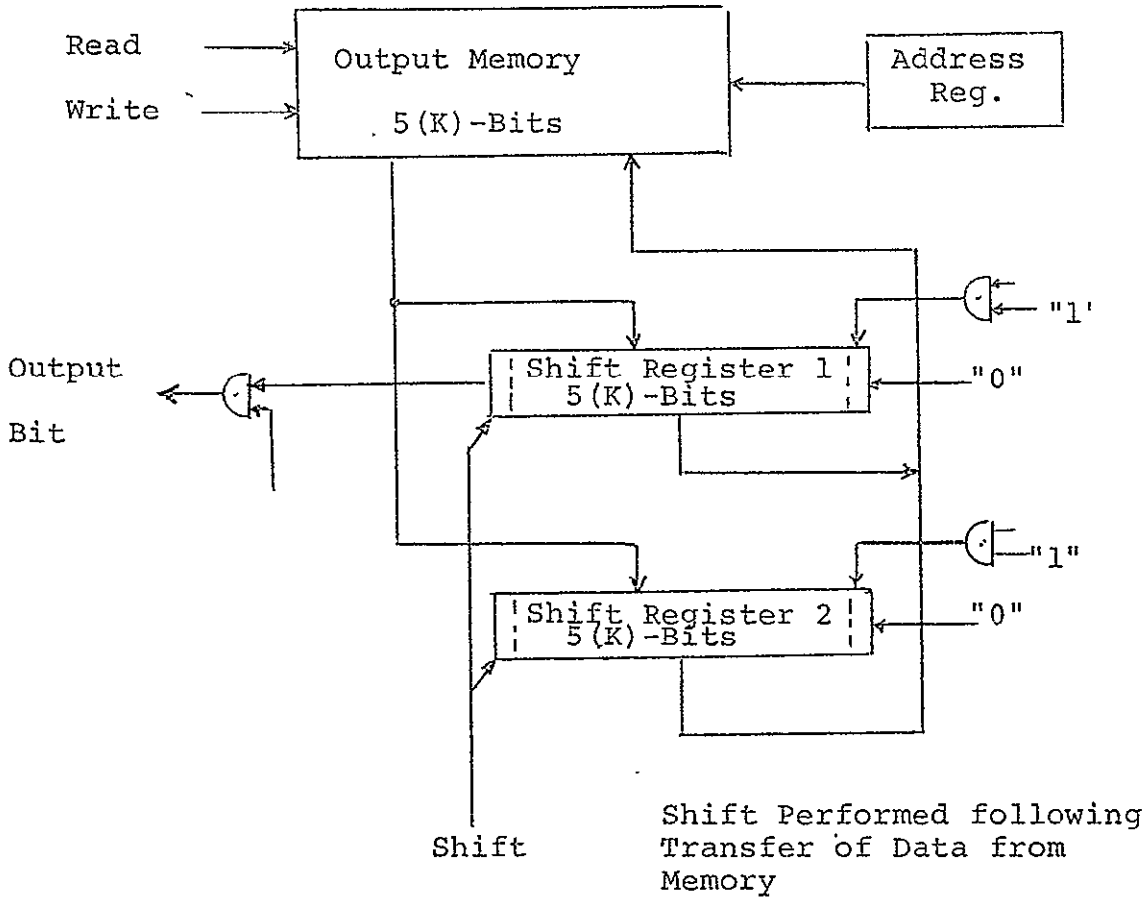


Figure 4.24
Output Memory



3-bit storage register only if the signal SUM $<$ c(LS Reg.) is present. Thus the last least-sum address is stored in the register from which it can be used to address the output memory.

The particular configuration shown is capable of addressing 0-7 locations. Duplicating the logic provides addressing for 0-64 locations.

4.4.2.3.5 Output Memory and Shift Registers (Figure 4.24)

A 5(K) bit semiconductor memory, in connection with two 5(K) bit shift registers, is utilized for the information output-bit generation. The specific word size for K = 3 is 4 words. The general case requires a memory word size of $2^{K-1} + 2^{K-3}$, the latter term supplying intermediate storage locations necessary when K is $>$ 3. During each cycle the contents of certain locations must be retained for later utilization.

Data is read from memory into either shift register where it is shifted one pulse to the left. A logic 0 or 1, according to the algorithm, is loaded into the least significant bit position before the data is written back into memory.

The time available for reading data from locations and placing in temporary storage is insufficient when K = 10. For this configuration, two output memories are required. Data is read from one memory and stored in the other during a given cycle; the operation being reversed during the following cycle.

4.4.2.3.6 Timing and Control

The decoder cycle repeats every 1/18th millisec (≈56 usec) which is the time between information bits for a transmission rate of one 18-bit message each millisecond. Figure 4.25 shows a 1.14 MHz clock rate counted down by stages A through F to provide the 56 usec interval. An additional stage, G, is required to select the Ax, Ay memory read-write functions which alternate each cycle.

TABLE 4.17

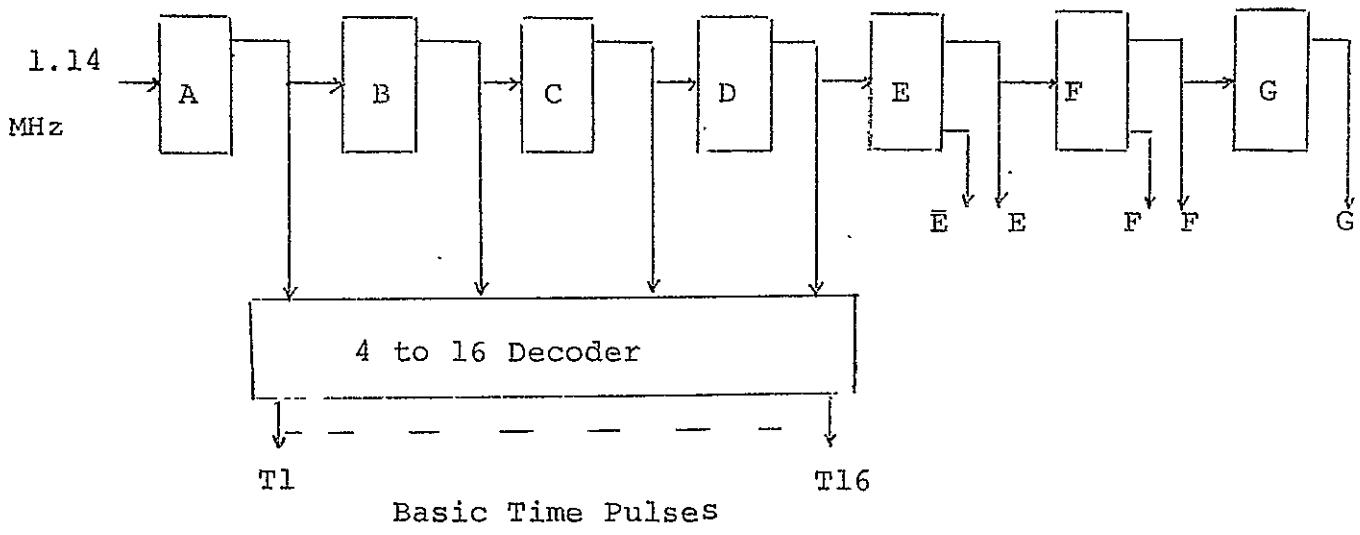
TIMING REQUIREMENTS

	K								
	2	3	4	5	6	7	8	9	10
CLOCK FREQ. (MHz)	.570	1.14	2.28	4.56	9.12	18.24	36.48	72.96	114
BASIC TIME INTERVAL = 1/f (nano-sec.)	1750	875	437	218	109	55	28	14	7

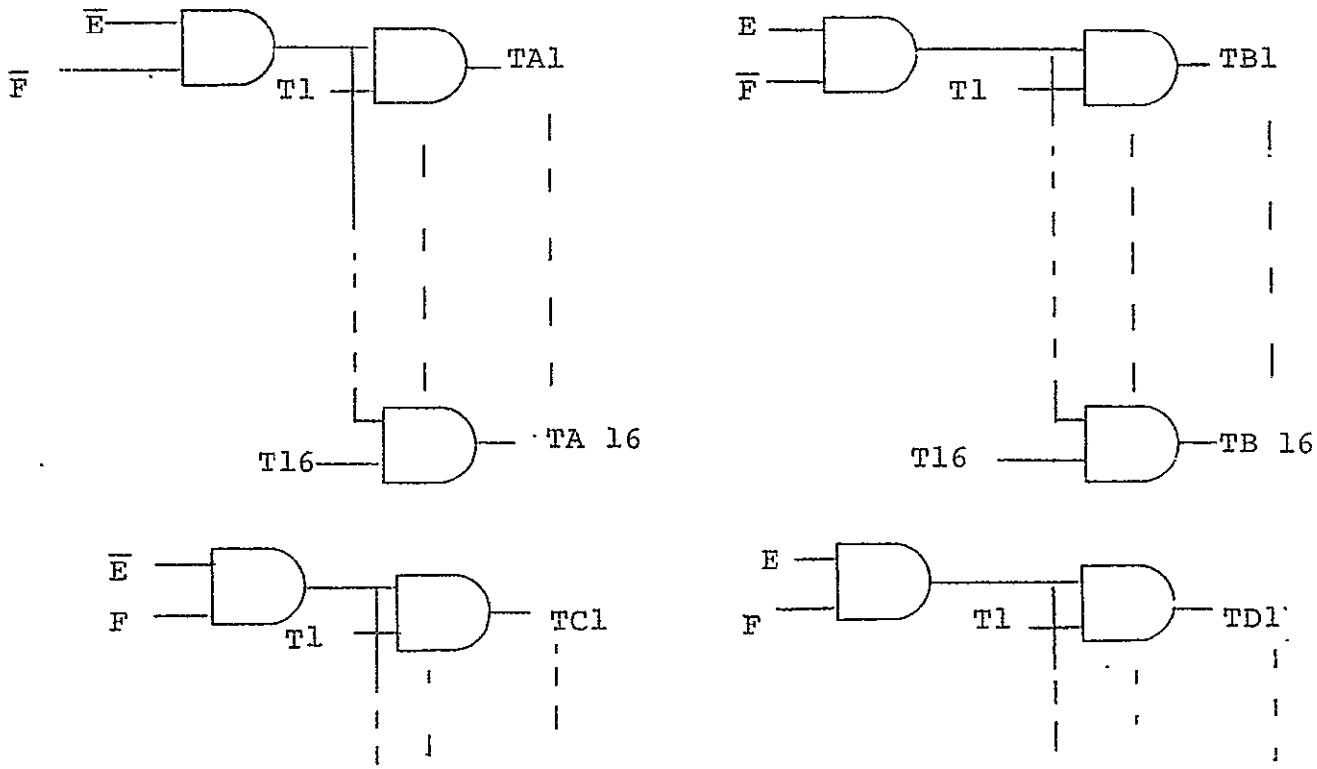
A 4 to 16 line decoder provides the basic time pulses T1 through T16 for control functions. Table 4.17 defines the clock rates and basic time pulse duration for values of K from 2 through 10. High speed TTL logic is required for time-pulse generation with K = 7 and 8. For K = 9 and 10, current mode logic, such as MECL II, is required for parts of the time-pulse generator. Figure 4.26 shows a typical timing diagram for K = 9.

Figure 4.25

Time Pulse Generation for Decoder



Typical Control-Pulse Generation Logic



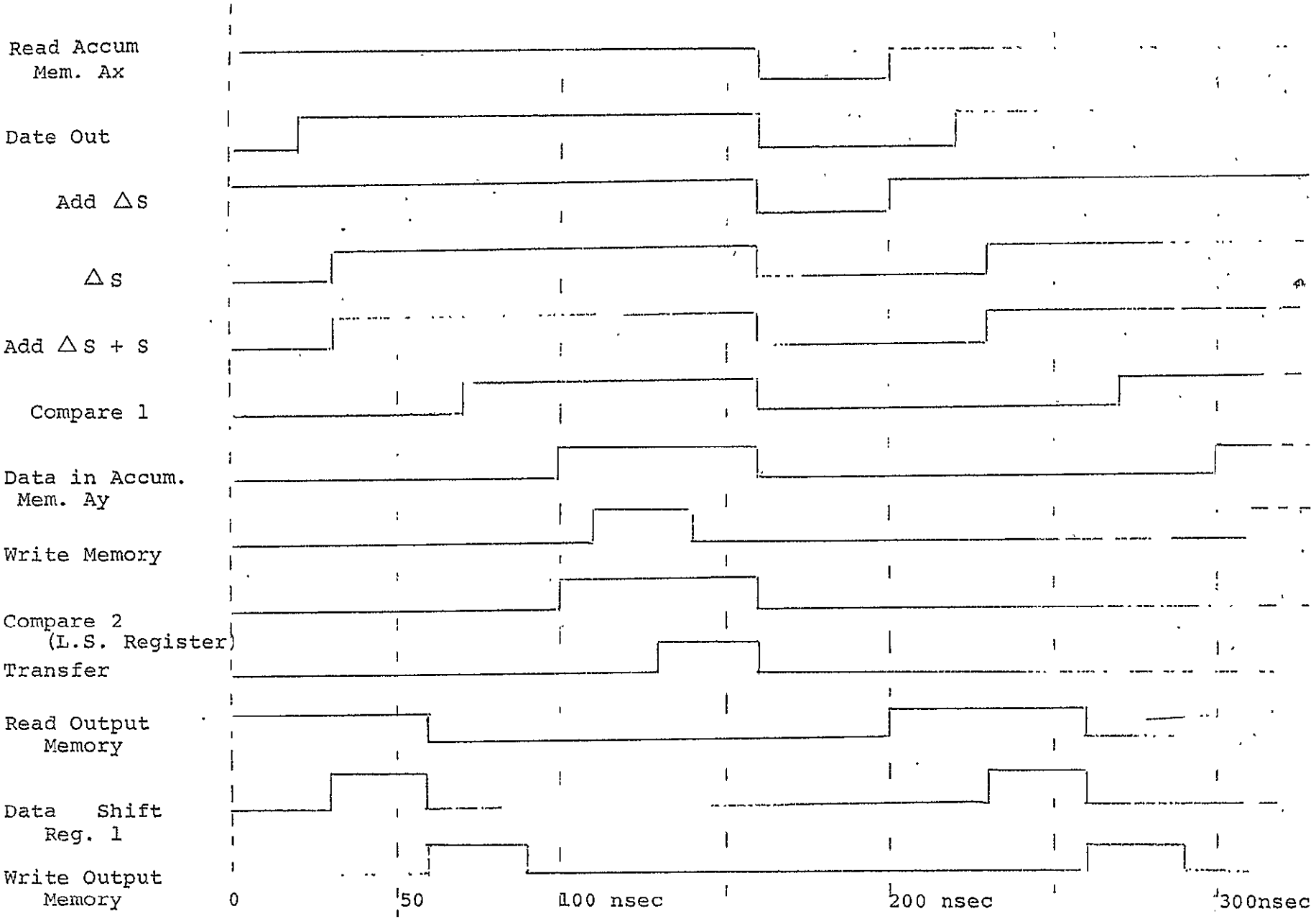


Figure 4.26 Typical Timing Diagram K = 9

4.4.2.3.7 Packaging Requirements

Table 4.18 lists the number of logic packages (D.I.P.s) required to implement the decoder for various values of K, V and ℓ. This is the type of breadboard implementation which was proposed for the command controller and the previous decoders.

TABLE 4.18

PACKAGE REQUIREMENTS FOR VARIOUS DECODER CONFIGURATIONS

		K								
	V	2	3	4	5	6	7	8	9	10
1	2	40	43	50	56	61	77	94	136	285
	3	42	45	52	58	63	79	96	138	287
	4	44	47	54	60	65	81	98	140	289
2	2	41	44	51	57	62	78	98	144	300
	3	44	47	54	60	65	81	101	147	303
	4	46	49	56	62	67	83	103	149	305
3	2	46	49	56	62	67	83	103	149	305
	3	50	53	60	66	71	87	107	153	309
	4	52	55	62	68	73	89	109	155	311
4	2	46	49	56	62	67	83	103	149	305
	3	52	55	62	68	73	89	109	155	311
	4	56	59	66	72	77	93	113	159	315

4.4.3 Error and Rejection Probabilities for Maximum Likelihood Decoding of Convolutional Codes

Most of the known results concerning the probability of an error in decoding convolutional codes using the Viterbi algorithm have been obtained by simulation. While these results are universally concerned with error probabilities considerably larger than those of interest here, they are easily extrapolated down to the range of interest. Doing so, one immediately concludes that ordinary Viterbi decoding does not achieve the desired error probability at the signal-to-noise ratios operative here for any constraint length $K \leq 10$. Yet the complexity of equipment needed to decode codes having constraint lengths greater than 10 clearly precludes their further consideration, at least for spaceborne applications (see Section 4 4.2).

It is possible to decrease the probability of an error, however, by rejecting those decoded bits which do not appear to be sufficiently reliable. (This, of course, was the technique used to reduce the word error probability associated with the block codes studied earlier.) The Viterbi algorithm for decoding convolutional codes involves the calculation of a number for each of the 2^{K-1} currently most likely code sequences. This number is a monotonically decreasing function of this likelihood of its associated sequence. An obvious rejection technique in this case, therefore, is to compare the two smallest of these numbers.

If these numbers are sufficiently different, it can be concluded that the sequence corresponding to the smallest number is significantly more likely than any other sequence and hence can safely be accepted as the correct sequence. As soon as this difference drops below some threshold value, however, this conclusion no longer holds, and no sequence should be accepted as correct. In this case, the decoded bits will be rejected, and will continue to be rejected until this difference again exceeds the rejection threshold. (Another advantage of this rejection approach over some of its potential competitors is that it represents only a small increment in the amount of hardware needed to implement the decoder; an increase by about four DIP's in the package counts of the previous section should be sufficient.)

To judge the effectiveness of incoporating a rejection criterion in the decoding algorithm consider the following situation. Suppose the decoded sequence is the correct sequence up until some time t_o and that the (eventual) decoded sequence and the correct sequence diverge at that point. Then at time $t_o + 5KT_b$ (with T_b the information bit period) the decoder output will be in error (assuming the decoder registers are 5K bits long) unless it is rejected by the rejection criterion. The sequence of bits received over the interval $(t_o, t_o + 5KT_b)$ can consequently be regarded as a word from a block code of word length $n = 5Kv$.

Thus the probability of an error and of a rejection can be estimated using the techniques of section 4.2.3

To apply those results to the present situation consider first the case in which $\ell = 1$ (the notation is as in the sections of 4.4.1 and 4.4.2) and in which a decoded bit is accepted if, and only if, the difference (e_2-e_1) (with e_2 the Hamming distance separating the received n-bit ($n = 5KV$) sequence from the nearest competitor to the decoded sequence, and with e_1 the distance separating the received and decoded sequences) is less than or equal to $d-(d_o-1)$. Here d is the minimum Hamming separating any two n-bit "words", and d_o is an integer in the range $1 \leq d_o \leq d$. A correct acceptance of a decoded bit will certainly be made, using the rejection criterion described, if $e_1 \leq (d_o-1)/2$ since then $e_2 \geq e_1$, must exceed the specified threshold. Similarly, e_1 must exceed $d-(d_o/2)$ in order to cause an error. Thus, with P_e and P_r the error and rejection probabilities respectively, we have

$$P_e + P_r \approx Pr \left\{ e_1 > \frac{d_o-1}{2} \;\middle|\; \ell = 1 \right\} \tag{1a}$$

$$P_e \approx Pr \left\{ e_1 > \frac{2d-d_o}{2} \;\middle|\; \ell = 1 \right\} \tag{1b}$$

Equations (1a) and (1b) can be approximated using the approach detailed in section 4.2.3 leading to the bounds

$$P_e \leq P\left(\frac{2d-d_o+1}{n}\right) \tag{2a}$$

and

$$P_e + P_r \leq P\left(\frac{d}{n}o\right) \tag{2b}$$

where $P(x) = e^{-(nRx)/2}$

Here R denotes the ratio of the signal energy per channel symbol to the noise spectral density and n is as previously defined. The constant .687 in the exponent of the P(x) defined in the section 4.2.3 is replaced here by 1/2 since, when $\ell = 1$, erasures are not acknowledged. A retracing of the argument presented in that section quickly establishes that the correct constant should indeed be 1/2 when no erasures are allowed.

At the other extreme, when $\ell = \infty$, a similar argument leads to the conclusion that *

$$P_e \approx P\left(\frac{4d-2d_o+2}{n}\right) \qquad \ell = \infty \qquad (3a)$$

and

$$P_e + P_r \approx P\left(\frac{2d_o}{n}\right) \qquad (3b)$$

with P(x) as previously defined.

It remains only to find appropriate values of d to complete the desired bounds. But this is easily accomplished upon observing that, for convolutional codes,

$$d \leq \min_h d\left[(h + K-1)V, h\right] \qquad (4)$$

* cf. G. D. Forney, Jr., "Generalized Minimum Distance Decoding," IEEE Transactions on Information Theory, IT-12, p. 125, 1966.

where $d(n,K)$ is the maximum minimum distance attainable with an (n,k) block code and h ranges over all non-negative integers. (This bound follows immediately upon noting that the minimum weight of a convolutional code word generated by an information sequence, the first and last bits of which are separated by a distance of $h + K - 1$ or less, is thus bounded, and that convolutional codes are linear.) Using the Griesmer bound on $d(n,K)$ then establishes the bounds of Table 4.19.

Table 4.19

Bounds on the Maximum Minimum (free) Distance d Attainable with Constraint Length K, Information Rate 1/V, Convolutional Codes

K \ V	2	3	4
2	4	6	8
4	6	10	12
6	8	13	18
8	11	16	22
10	12	20	27

While these results are bounds on d, they seem to be attainable, or nearly so, for all values of K and V of interest here.

Table 4.20 reflects the bounds of equations (2a) (2b), (3a) and (3b) when RV = 6.845 (corresponding to an uncoded error probability of 10^{-4}) and when the distances d are as listed in Table 4.19. The numbers listed show the bounds on the rejection probability P_r when d_o is chosen to yield an error probability bound of 10^{-18}. The bound denoted "upper" refers to the bound of equation (2b) with d_o so chosen that the bound of equation (2a) is at most 10^{-18}. Similarly, the "lower" bound refers to the bound of equation (3b) with d_o so chosen that the bound of equation (3a) is at most 10^{-18}. The dashed entries indicate values of K and V for which even the lower bound on P_e exceeds 10^{-18} regardless of the value of d_o.

Not surprisingly, the differences between the upper and lower bounds indicated is Table 4.19are large. Nevertheless, it does seem apparent from these results that a V of 3 or 4 and a K of at least 6 and more likely of 8 or 10 is needed to achieve the desired error probability with a sufficiently small rejection probability. Furthermore, these bounds refer to the probability of the <u>first</u> error and <u>first</u> rejection. Clearly, when a bit error or rejection does occur it will with high probability be followed by a sequences of errors and rejections. Since a data word will have to be rejected whenever any one of its bits is rejected (and is in error whenever one or more of its bits are in error), each "first"

Table 4.20

Rejection Probability Bounds

K \ V	2		3		4	
	$-\text{Log}_{10} P_r$		$-\text{Log}_{10} P_r$		$-\text{Log}_{10} P_r$	
	Lower	Upper	Lower	Upper	Lower	Upper
2	-	-	-	-	-	-
4	-	-	1.96	0	-	-
6	5.88	0	7.84	0	8.82	0
8	14.7	0	13.72	0	14.7	0
10	17.64	0	21.56	2.94	22.05	2.35

error or rejection will generally give rise to a sequence of several data word errors or rejections. Thus, a K as large as 10 may indeed be required to achieve the desired performance.

4.5 WORD SYNCHRONIZATION

The most straightforward, and, in the case of the command system, the most appropriate method for initially establishing synchronization between the transmitter and receivers is to transmit a synchronization prefix. This would consist of a pure tone of sufficient duration to bring the receiver phase-locked loop into lock, followed by a sequence of phase reversals to allow the receiver to establish bit synchronization, followed in turn by a known bit sequence for establishing word synchronization. The demodulated signal would be routed to the decoder only after the word sync sequence has been positively identified. Finally, no output from the decoder would be recognized as a code word until a predetermined command is received and decoded. This event would then initiate an acknowledgement establishing the coherent acknowledgement link needed for the command system to operate properly.

Clearly, such an acquisition procedure can be made to be extremely reliable. Decoding begins only when all sync devices, phase-locked loop, bit synchronizer, and word synchronizer, have identified specific events, and then only after a prescribed command is received. A potentially much more vulnerable situation arises, however, if an initially synchronized system should somehow loose synchronism without detecting this event. Since the received signal may still be strong, the demodulator could continue to operate and the decoder would receive and attempt to decode a signal which appears to be valid but when decoded bears little resemblance to the transmitted message.

enabling the decoder to recognize that a loss of synchronism has indeed occurred.

The $(8,4)_2$ inner code is rendered reasonably invulnerable to mis-synchronism by adding almost any random binary 8-tuple to each code word. One good choice is the 8-tuple 00001101. (This sequence can be generated using only two dual-in-line packages. Thus neither the encoder nor the decoder complexity is significantly affected by this modification.) It has been shown* that any overlap of two words from the resulting code is quite unlikely to be a true code word. More specifically, suppose the decoder is out-of-sync by one bit (i.e. the decoder begins decoding either one bit early or one bit late). Then, in the absence of transmission errors, it will never see a true code word. Half the words it sees, on the average, will in fact differ from the nearest code word bit one bit and half by two bits. An identical statement holds when the decoder is out-of-sync by two bits. When it is three bits out-of-sync, an overlap word can be identical to a code word, but this happens only $1/16^{th}$ of the time on the average. Half the time the overlap word will differ from the nearest true word by one bit, and $7/16^{th}$ of the time by two bits. Fianlly, when the decoder is four bits out-of-sync, all overlap words differ by exactly two bits from the nearest code word.

It is therefore clear that when a sync error does occur the inner decoder will see a bit sequence which bears little resemblance to a sequence of true code words, and will hence

---

* F. D. Natali, "Synchronization Properties of a Near-Comma-Free (8,4) Code", IEEE Transactions on Communication Technology, Vol. COM-17, p. 500, Aug. 1969.

reject a large percentage of these overlap words as undecodable The outer decoder will then with high probability see more than two erasures in a given word, and, consequently, it too will reject the sequence as undecodable. Accordingly, virtually all words transmitted to the mis-synchronized decoder will be rejected. The command system will be informed of this fact via the acknowledgement link and will be forced to re-enter the acquition mode in order to bring the errant decoder back into synchronism.

To be more quantitive about the probabilities involved, suppose the decoder is one bit out of synchronism and, for the moment, that the received signal-to-noise ratio is infinite. Then, with probability 1/2, the inner decoder will see a word containing two errrors and hence will treat it as an erasure. With probability $p_1 \approx 1\text{-}3 \times 10^{-3}$, the outer decoder will see more than two erasures in any 15-symbol word and will therefore not attempt to decode it. Even if the input to the outer decoder contains fewer than three erasures, the non-erasured symbols will in effect appear to be a complexity random symbol sequence and will with probability $p_2 \approx 1\text{-}1/6 \times 10^{-4}$ or greater differ by at least one symbol from any Reed-Solomon code word. Thus, the probability of such a mis-synchronized word not being rejected is on the order of 5×10^{-8} or less. At finite signal-to-noise ratios when the probability of an erasure is no longer zero, the rejection probability p_1 should actually increase considerably. This is because even words containing one error are rejected, using the prescribed decoding algorithm, if such a word also contains one or more erasures. Hence, the probability that the outer decoder finds its input sequence decodable should actually

be considerably less than 5×10^{-8}. In any event, to arrive at the probability of an error due to mis-synchronization this probability must be multiplied by the probability of an undetected loss of synchronism. The undetected loss-of-synchronism probability will be extremely small at the signal-to-noise ratios of interest here so long as the bit synchronizer is properly designed. Thus, the possibility of an error due to mis-synchronism does not seem to pose a serious threat here if adequate measures are taken in the system design.

4.6 SUMMARY AND CONCLUSIONS

Four different coding schemes have been examined in some detail: BCH coding and algebraic decoding; projective and Euclidean geometry codes and threshold decoding; Reed-Solomon outer, biorthogonal inner codes, and concatenated decoding, and convolutional codes with Viterbi decoding. Investigation of the second of these schemes did not proceed as far as a detailed decoder package count since, for the error probabilities of interest here, the resulting decoders were almost certain to be at least as complex as those of the first scheme. The complexities (measured in DIP's) of the decoders required to achieve minimum performance (defined here as $P_e \leq 10^{-18}$, $P_r \lesssim 10^{-2}$) are listed in Table 4.21. The associated error and rejection probabilities at R = 6.845 so that p(uncoded) = 10^{-4} are also summarized in Table 4.21.

Table 4.21

Decoder Package Requirements vs. P_e and P_r

Type of Decoder	Package Count	P_e	P_r	References
BCH* (63,16)	136	3×10^{-17}	10^{-2}	Tables 4.3, 4.4.A
BCH* (127,54)	243	2.5×10^{-17}	2×10^{-4}	Tables 4.3, 4.4A
Concatenated w. Algebraic Decoder	65	4.6×10^{-20}	4.5×10^{-3}	Tables 4.5, 4.6
Concatenated w. Green Machine	76 ($\ell = 4$)	2.7×10^{-21}	4.9×10^{-3}	Tables 4.11, 4.12
Convolutional † (K = 8, V = 3) w. Viterbi Decoder	111 ($\ell = 3$)	10^{-18}	10^{-3}	Tables 4.18, 4.19

* The BCH error and rejection probabilities correspond to an R of 6.55 ($p = 1.6 \times 10^{-4}$)

† The convolutional code error and rejection probabilities are rough estimates. The package count of Table 4.19 is here increased by four to provide the necessary bit rejection capability (cf. section 4.4.3)

It should be recalled that the P_e figures listed in this table were all more or less based on the assumption that an error pattern involving more than $d-(d_o-1)/2$ symbol errors automatically causes a decoding error. Actually this is true only for close-packed codes, and then only if $d_o=d$. In all cases of interest here d_o is considerably less than d. The result is that many events identified as errors in the preceeding analyses would actually result in rejections, not errors. This statement was supported quantitatively in section 4.3.2 for the concatenated code case; it also applies (although not to the same degree) to the other coding schemes as well.

It might be argued that as a result the coding schemes investigated here are more powerful than they really need to be to achieve the desired performance. Actually, this added margin is extremely valuable to have, for example, when the possibility of deep fades is taken into account. For then the probability of an error pattern of greater than critical weight can become significantly larger than the design value of 10^{-18} (cf. Tables 4.7, 4.8). The fact that only one out of, say, 10^7 such patterns actually result in decoding error is extremely desirable for precisely these eventualities. In any event, the advantage of the concatenated approach vis-a-vis the other methods seems apparant from Table 4.21. The choice between the Green Machine and algebraic inner code decoders, however, is less obvious; the latter is recommended here, for lack of a more compelling reason, simply on the basis of its slightly smaller package count.

5.0 MICRO-ELECTRONIC DEVELOPMENT PROGRESS

The complexity of the MSI and LSI logic arrays which can be successfully produced and interconnected continues to grow. Activity in the micro-electronic field at Raytheon is concentrated on the development and utilization of devices in the gold beam lead configuration.

5.1 BEAM LEAD SYSTEM

The beam lead system is a packaging technique which totally eliminates wire bonding along with the cost and reliability problems associated with wire bonding. This is accomplished by making the lead an integral part of the chip. These beam shaped contacts are electroformed on the device and extend over the periphery for ease in mounting (Figure 5.1). The beams also serve as mechanical and electrcial interconnections. This approach offers many advantages over conventional wire bond and wire weld presently used to attach the circuit element in hybrid applications. The conventional approach has been to use gold wire in conjunction with thermal-compression bonding techniques (Figure 5.1), resulting in as many as thirty-two bonds for sixteen lead devices. Sixteen tail-pull operations are also required to make electrical and mechanical interconnections between the chip and hybrid substrate patterns. This results in a total of 48 operations in order to insert a sixteen lead integrated circuit into a hybrid circuit for each such element used. Each such bond is a potential area of failure due to misplaced bonds, shorting of the ball bonds placed too

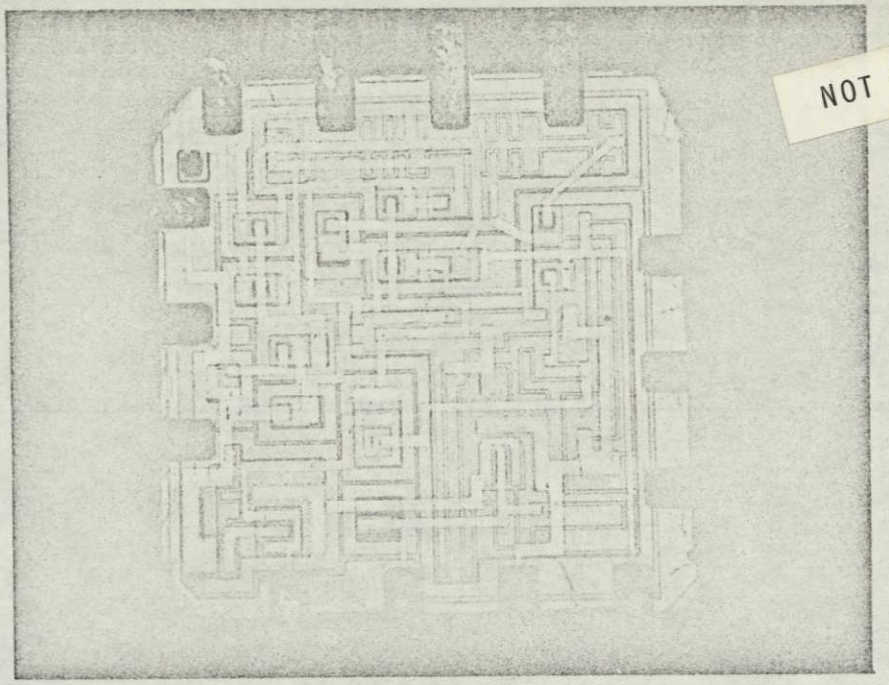


FIGURE 5.1 BEAM LEAD INTEGRATED CIRCUIT FABRICATED AT RAYTHEON SEMICONDUCTOR OPERATION

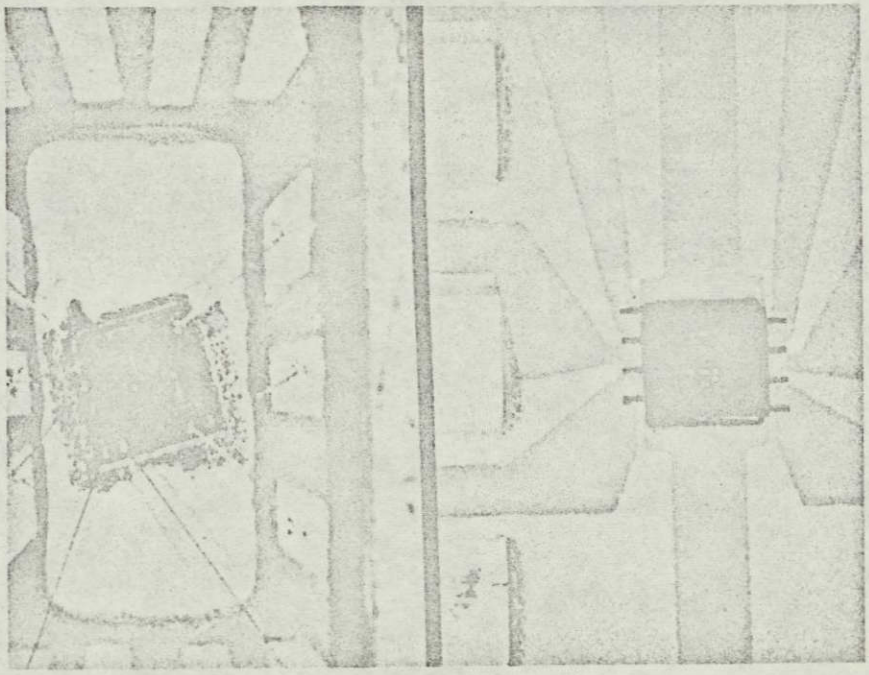


FIGURE 5.2 COMPARISON OF BEAM LEAD BONDING AND WIRE BONDING TECHNIQUES. BEAM LEAD BONDING DEVELOPED TO ACHIEVE BETTER RELIABILITY AND REDUCED COST.

close together, incomplete welds due to insufficient bonding pressure or too low a bonding temperature, and failure due to weak bonds tearing llose during tail pull. It is advisable for hybird circuit applications that the amount of bonds necessary to insert any circuit element be kept to a minimum in order to increase yields and final reliability of the finished hybrid system.

The beam leads an as integral part of the integrated circuit are ideally suited for hybrid applications. All leads lie in the same place and are welded in a single operation, reducing failure modes which results in increased reliability of the hybrid circuit. The semiconductor chip is encapsulated in its own hermetic package with only the gold beams exposed. This encapsulation protects the metal interconnects from corrosion and degradation due to scratching from handling.

It can be stated then, in conclusion, that beam lead chips will greatly alter electronic systems over the next few years. They presently offer the only practical approach yet devised for putting a large number of leads on a relatively small chip and successfully bonding these leads to a substrate without damage to the chip.

5.2 EXISTING DEVICES

5.2.1 Logic Arrays

Three families of NAND Gate Arrays are available. These are the 24, 40, and 60-Gate Arrays. All are fabricated on the standard 101 x 144 mil chip which

employs a single layer of metallization for interconnections. Crossovers of the interconnect pattern are accomplished by using conducting pads that were diffused into the semiconductor during fabrication of the array. In 1971 these underpasses in the semiconductor will be discontinued and two layer metallization will be used on the chip. The NAND Gate Arrays are available in a 50 pin format in a gold-beam lead configuration.

A total of 7 NAND Gate Arrays are available. These are:

1) RA103 -- 24-Gate Array
   Contains 24 high level gates

2) RA104 -- 40-Gate Array -- Low Power
   Contains 40 high level gates

3) RA105 -- 40-Gate Array -- Medium Speed/Powe
   Contains 40 high level gates

4) RA106 -- 40-Gate Array -- High Speed
   Contains 40 high level gates

5) RA107 -- 60-Gate Array -- Low Power
   Contains 20 high level gates and
   40 expander gates

6) RA108 -- 60-Gate Array -- Medium Speed/Powe
   Contains 20 high level gates and
   40 expander gates

7) RA109 -- 60-Gate Array -- High Speed
   Contains 20 high level gates and
   40 expander gates

5.2.2 Universal 24-Gate Array RA103

The 24-Gate Array is a universal logic array consisting of 24 four-input high level TTL NAND gates. The 24-Gate Array is a high speed, low density array

that is suitable for implementing irregular logic circuits that require gate-to-pin ratio of 0.2 to 0.5. The gates of this array were designed using RAY II design techniques in order to obtain high speed TTL logic. Since the various components of the gate circuit, or cell, are not connected until the metallization layer is applied, this circuit provides the designer with the capability of wire-OR'ing gates or increasing the Fan-in.

Up to 6 gates can be wire-OR'ed with a speed degradation of less than 1 ns per wire-OR. Similarly, the Fan-in can be increased to 8 inputs. All metallization which would interconnect unused components is omitted.

5.2.3 Universal 40-Gate Array, RA104, 105, and 106

The 40-Gate Array is a universal logic array consisting of 40 five-input high level TTL NAND gates. The 40-Gate Array is a family of medium density that is suitable for circuits that require a gate-to-pin ratio of 0.5 to 1.0. The gate structure employs Ray III tolerances (0.1 mil). These tight tolerances were necessary to achieve the desired density without any sacrifice in speed. The three members of this family utilize identical cell arrangement and I/O contract placement. They differ only in cell circuitry and structure to provide a choice in the speed/power trade

off. Wire-OR'ing up to 6 gates and increasing the Fan-in to 10 inputs are implemented as for the 24-Gate Array. Figure 5.3 is a photograph of a completed 40-Gate Array with beam leads.

5.2.4 Universal 60-Gate Array, RA107, 108 and 109

The 60-Gate Array is a universal logic array consisting of 20 five-input high level TTL NAND gates and 40 four-input TTL expander NAND gates. A total of 40 gates are provided primarily for use as a wire-OR extender or to increase the Fan-in of a high level NAND gate. These gates may also be used as NAND gates, but the fan-out is limited to 2. Since two of these gates can be accommodated in the same space necessary for one active pull up gate, a considerable increase in overall gate density is obtained. The 60-Gate Array is a family of high density that is suitable for circuits that require a gate-to-pin ratio of more than 1. The gate structure was designed with RAY III tolerances (0.1 mil). These tight tolerances allow the desired density with no sacrifice of speed. As with the 40-Gate Array, the members of this family employ slightly differing cell circuitry and structure, but identical I/O contacts which allows the designer freedom of choice in the speed/power trade off.

5.3 DEVICES IN DEVELOPMENT STAGE

The Bedford Microelectronics Department has presently under development a low power, low speed bipolar array. Power

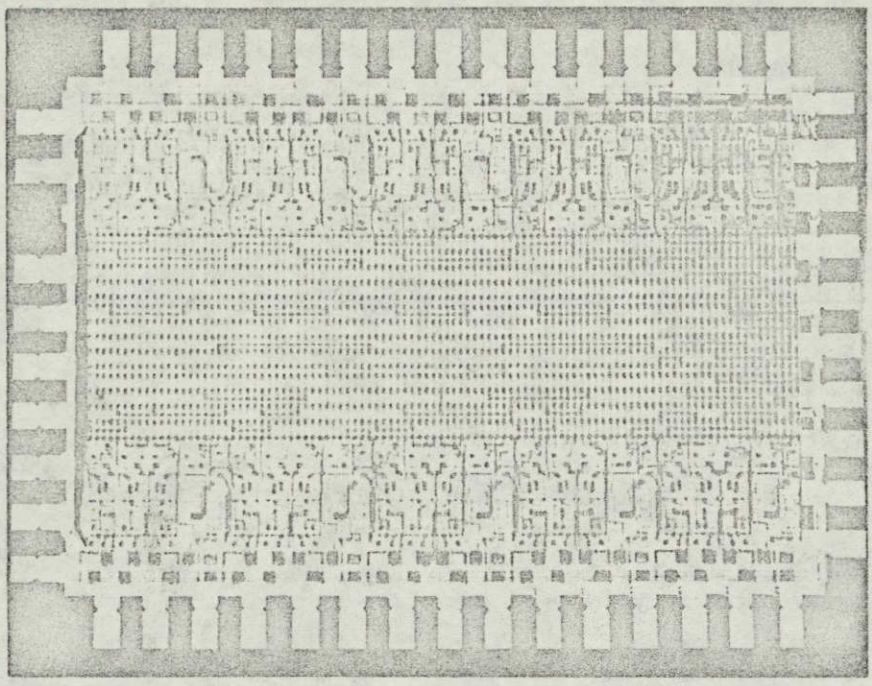


FIGURE 5.3

FORTY-GATE LOGIC CHIP WITH 50 BEAM LEADS DEVELOPED BY RAYTHEON MICROELECTRONICS PROGRAM

dissipation per gate is less than 2 milliwatts and propagation delay time approximates 50 nanoseconds. A proprietary method of fabricating diodes on the chip promises gate densities of from 160 to 180 per chip (144 mils square). This array appears attractive for use in the digital command system.

Standard LSI logic arrays of both bipolar and MOS technology will soon be available on the market. Functions expected to be available first are 8-bit look-ahead/carry adders, a dual 6-bit shift register and a 5-stage Johnson counter, all of approximately 100 gate complexity. A standard/custom flexibility will be offered by storing wafers containing only diffused components and no metalization. Mask sets for a customized array are then made up according to requirements and final metallization applied.

5.3.1 Complimentary MOS (C-MOS) Technology

C-MOS logic with very low power consumption, good noise immunity and medium speed capabilities make it a strong contender for future digital systems. Other features of the C-MOS logic are high fan-out capabilities and the ability to accept a wide range of power supply voltages.

Complimentary circuits differ from their polarity counterparts (P-MOS or N-MOS) in a variety of ways. In the more common P-channel transistor, the P+ source and drain regions are diffused into a crystal of N-type silicon. In complimentary MOS the basic building block is not a single transistor,

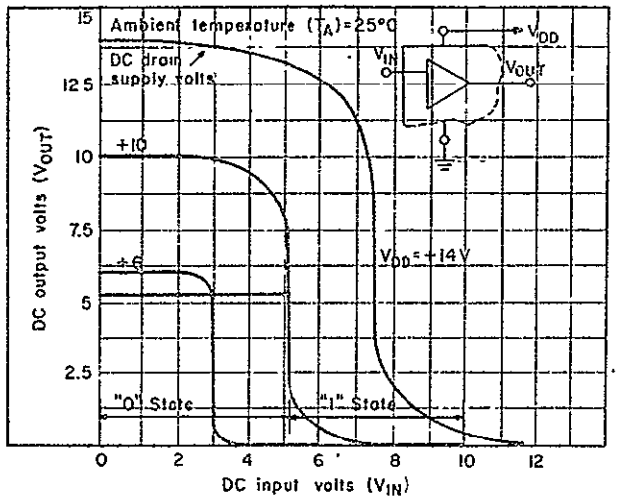
but two transistors in series; one P-channel and one N-channel. Because they have two transistors in series complimentary circuits consume very little quiescent power. In either logic state, one transistor or the other is off, therefore, the only power dissipated is leakage current. Power required during switching operations is somewhat greater since load capacities must be charged through the P-channel transistor as the output switches high. This energy must later be dissipated through the N-channel transistor during the transition to a low signal level. Power dissipation during switching is equal to $C_o V_{DD}^2 f$, where C_o is the output and load capacitance, V_{DD} is the supply voltage, and f is operating frequency in Hertz. For a sample power dissipation calculation, a load capacity C_o of 10 pf (equivalent to an input loading of 20 C-MOS gates on an array), V_{DD} of 10 volts, and fequency of 1 Megahertz is assumed. $P = 10 \times 10^{-12} \times 10^2 \times 10^6 =$ 1 milliwatt. The static (non-switching) power is in the region of 1×10^{-6} watts.

Another characteristic of C-MOS circuits is their ability to operate with unregulated power supplies. The circuits can maintain a logical 0 of 0V and a logical 1 of V_{DD} volts with a supply voltage ranging from 6 to 15V.

Figure 5.3.1 shows the voltage transfer characteristics for a typical inverter using three different power supplies. The noise immunity shown on the curve (about

5V for both the 1 and 0 state) for a 10V supply voltage illustrates the high noise immunity. It should be noted that neither logic level begins to change until the change in input voltage equals the threshold voltage of the off transistor.

Packaging desities attainable with C-MOS technology are approximately 25% greater than for low power bipolar. A function containing 775 MOS transistors in a C-MOS configuration on a 146 x 155 mil chip has been successfully manufactured.



TYPICAL VOLTAGE TRANSFER CHARACTERISTICS FOR THE BASIC C-MOS INVERTER

FIGURE 5.3.1

RADIATION SUSCEPTABILITY

C-MOS transistor configurations exhibit a change in threshold voltage when subjected to a strong gamma radiation environment, a change not evident with bipolar circuits. The response to radiation over an extended

period of time is still under investigation. It is known, however, that a dose rate of from 1000 to 2000 rads/sec (a rate much greater than humans can withstand) causes a threshold voltage shift of approximately one-tenth volt.

The radiation effect on the digital command system will depend upon the radiation environment and normal shielding provided by structural metal within the space base.

5.4 HYBRID INTERCONNECTION TECHNIQUES

The Bedford Microelectronics Department develops the technologies required to build a bridge between device capabilities and system requirements. This includes the development of thick and thin film multi-chip interconnection modules. Typical devices developed so far are an 8-chip thin film module, and a 6-chip thick film module (see Figures 5.4.1 and 5.4.2). The interconnection module shown in Figure 5.4.2 called a Raypak, is a 1.15 inch square ceramic substrate with 5 mil interconnect lines on 10 mil centers. The module I/O leads, 20 per side, are on 50 mil centers. A hermetically sealed protective cover completes the package (Figure 5.4.3).

The square package configuration with leads extending from two sides allows heat sinks and interconnecting lines to be run under the other two sides. A .75 inch square "mini" Raypak of similar construction is also being developed at the Bedford Microelectronics Department.

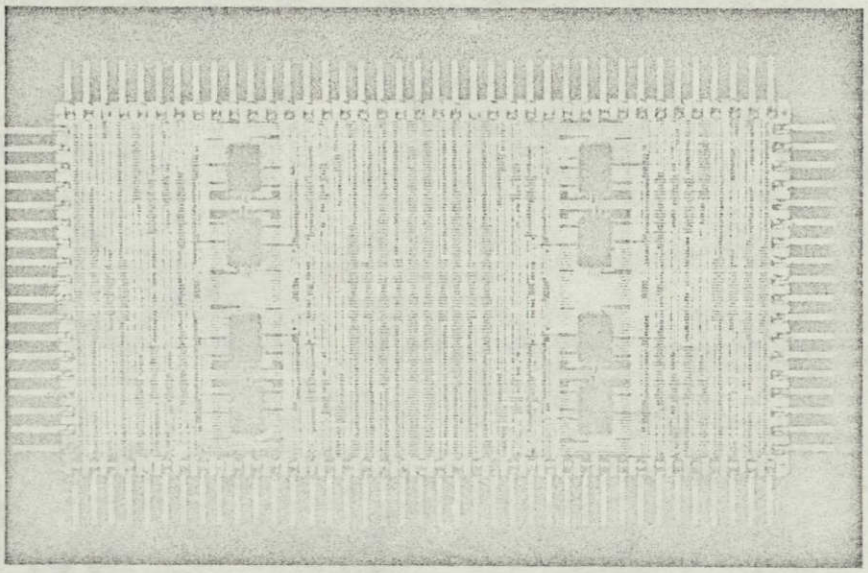


FIGURE 5.4.1 TYPICAL RAYTHEON THIN FILM MODULE CONTAINING 8 40-GATE ARRAYS

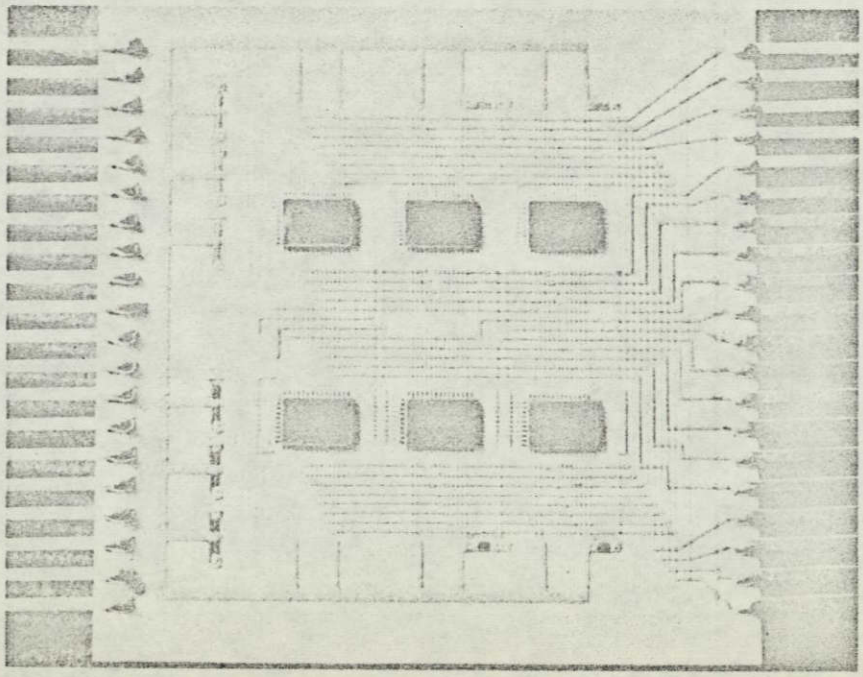


FIGURE 5.4.2 RAYTHEON THICK FILM INTERCONNECTION MODULE CONTAINING 6 40-GATE ARRAYS

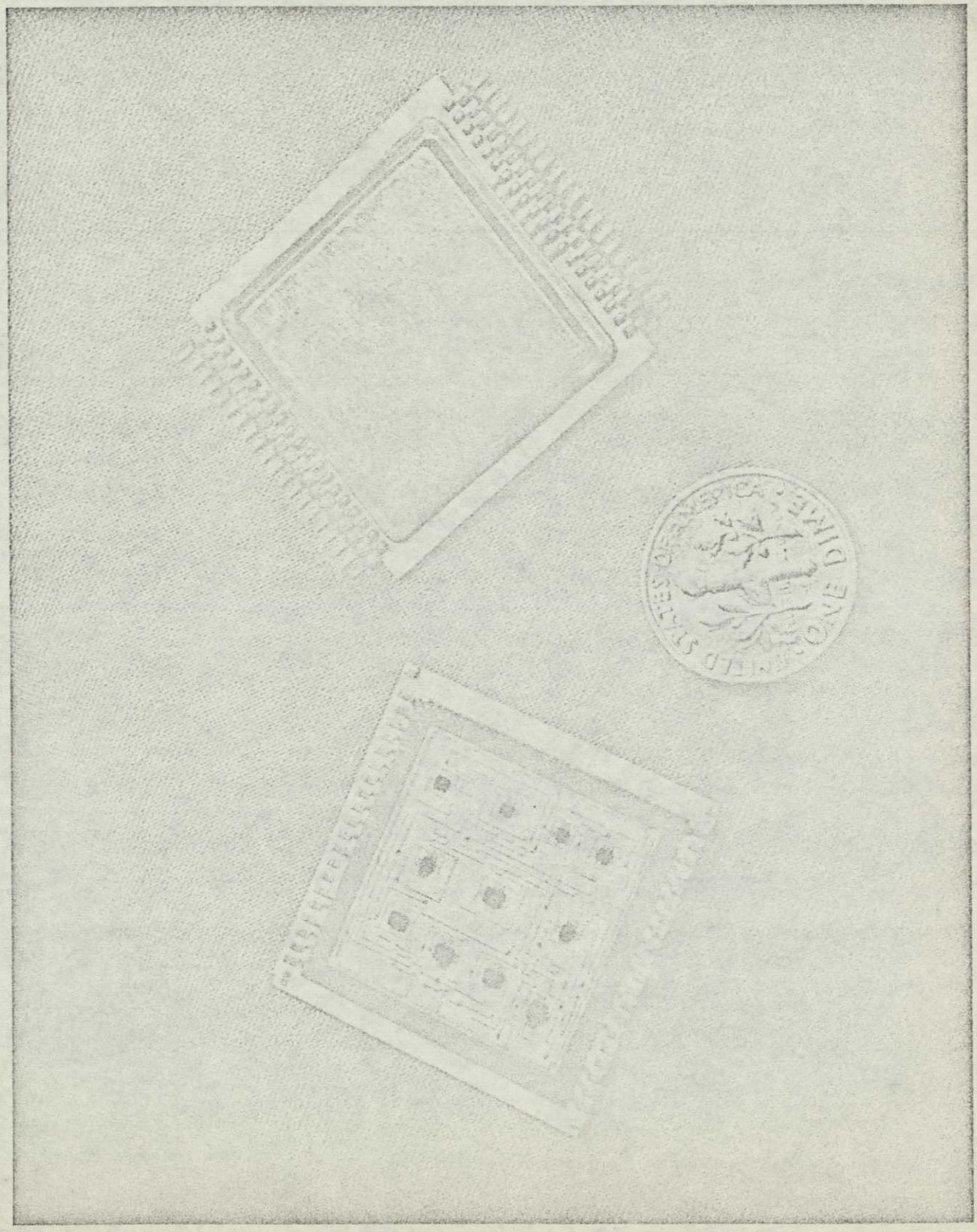


FIGURE 5.4.3 STANDARD 40 PIN RAYPAK HYBRID LSI PACKAGE

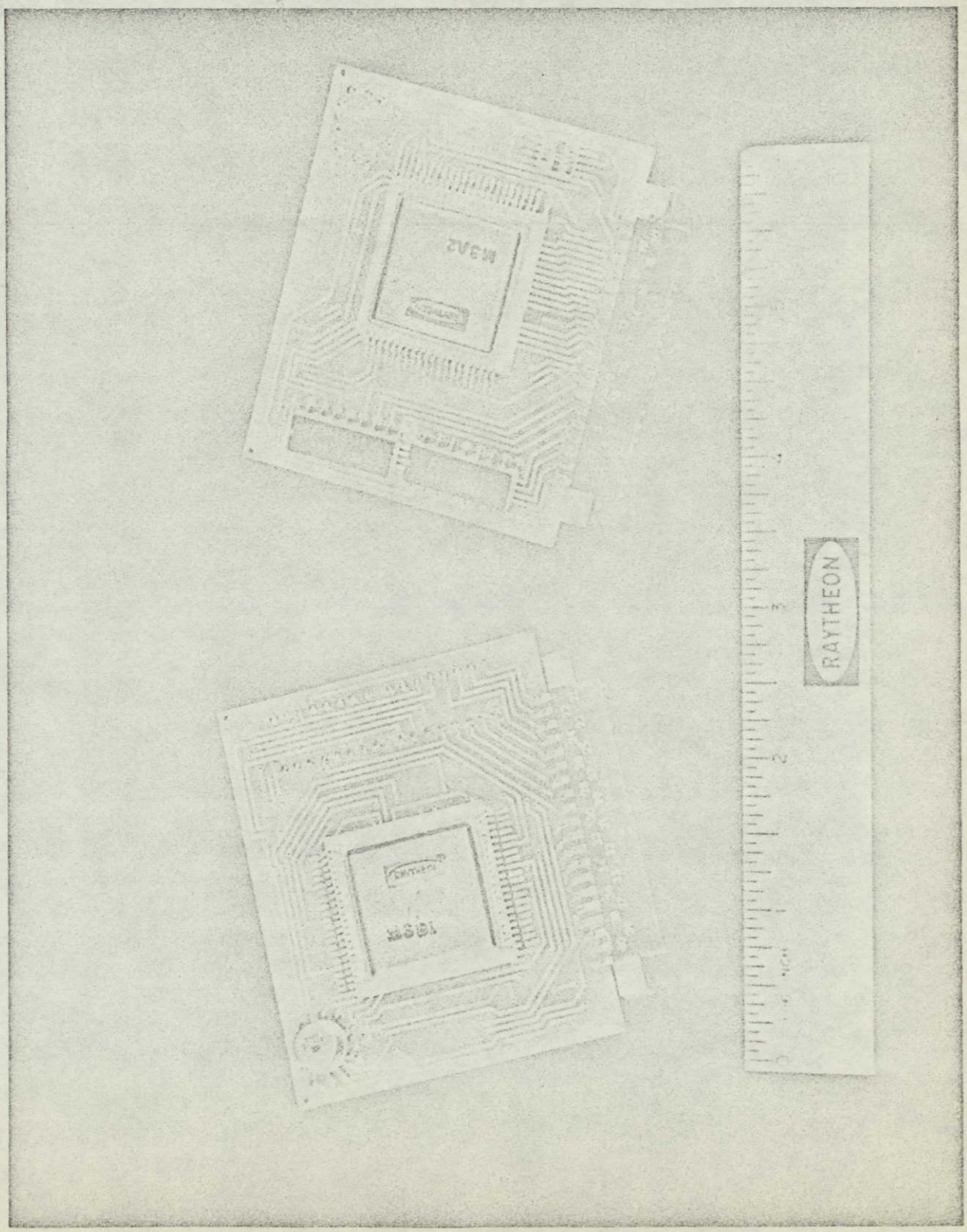


FIGURE 5.4.4 MODULE WITH 2 RAYPAK LSI PACKAGES

A wide variety of semiconductor devices, both bipolar and MOS, can be interconnected on the Raypaks. The equivalent of 300 logic gates have been successfully interconnected on one standard Raypak. MOS memory chips are presently available with 256 bits on a 140 x 140 mil chip. Eight of these chips, along with one bipolar chip for address decode, can be mounted on a Raypak to produce a 2048 bit random access memory package.

Another hybrid technique being developed at Raytheon involves the use of a silicon inner substrate. Logic chips are mounted on the silicon substrate which provides interconnections of two layer metallization. I/O connections are made from the inner substrate to the outer substrate (plastic or ceramic). Two advantages of this hybrid technique are the compatibility of the logic chip and silicon inner substrate, and the greater density of interconnections possible using thin film metallization methods.

5.5 LSI MODULE DEVELOPMENT

Two LSI module configurations are presently being developed at Raytheon. One, the Advanced Functional (AF) LSI module, carries four mini Raypaks, while the other, the type "B" LSI module, carries six standard Raypaks.

5.5.1 Advanced Functional LSI Module (Reference Figure 5.5)

The AF is an 80 pin module with overall dimensions of approximately 2.6 inches by 2.1 inches by 0.5 inches. This particular module is designed for air cooling, having heat fins across the top length.

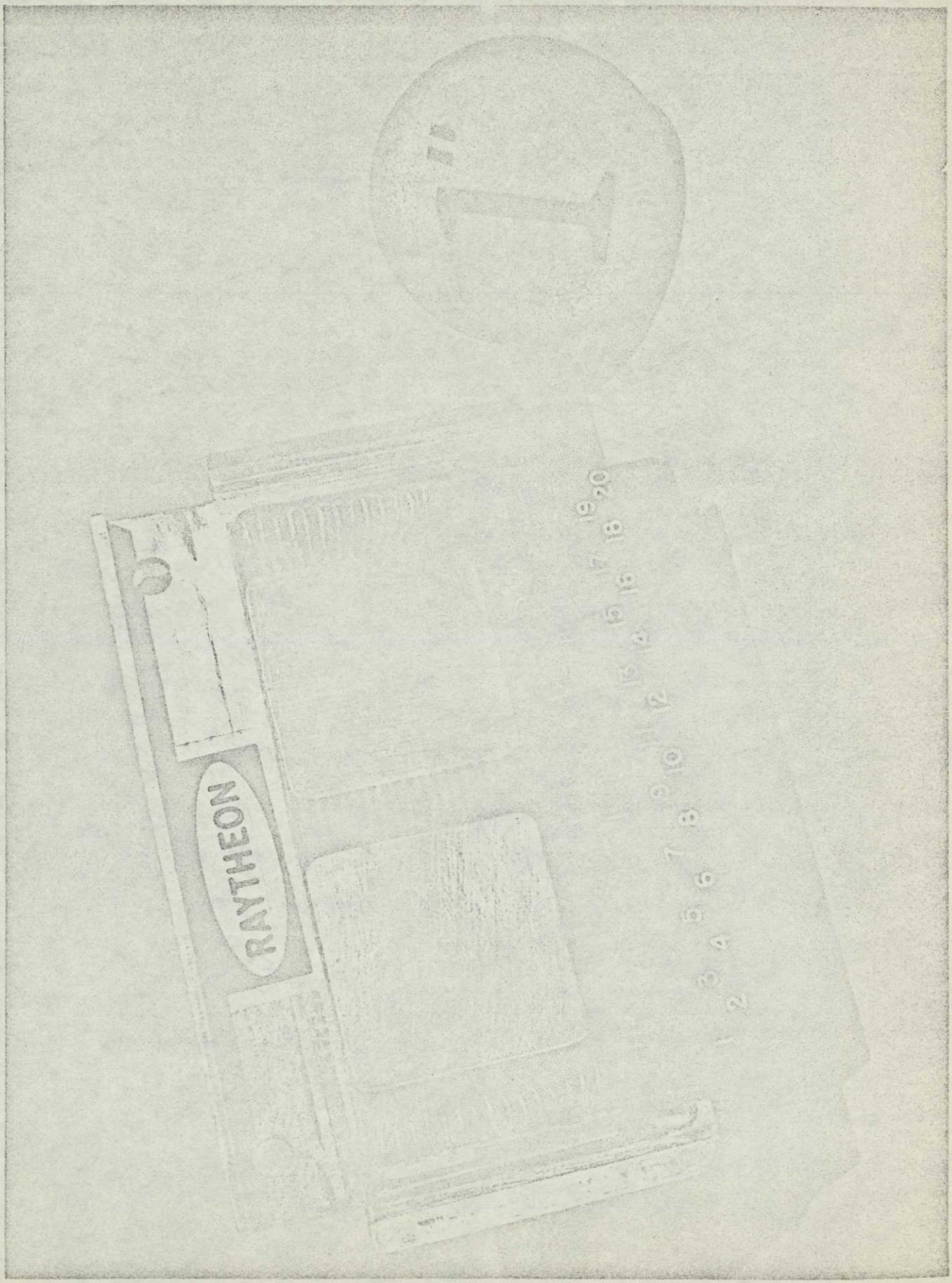


FIGURE 5.5 ADVANCED FUNCTIONAL LSI MODULE

The height of the module could be reduced to 1.6 inches by removing these fins and allowing heat conduction through each end to a liquid-cooled surface.

The module carries four mini Raypaks of 0.75 inches by 0.75 inches mounted on two substrates. Two 40-pin connectors are soldered to the substrates using a PC board and feed-thru to connect the back row of pins to the front of the substrates. The assemblies are bonded to a cast or machined header and a flex strip soldered to each end of the substrates to provide inter substrate connections (reference Figure 5.5).

A thermal analysis shows that the module if capable of dissipating 4 to 5 watts of power while maintaining normal semiconductor operating temperatures in the Raypaks.

5.5.2 Type B LSI Module (Reference Figure 5.6)

The type B LSI module configuration measures 6.4 inches wide by 3.6 inches high by 0.3 inches thick and utilizes a one hundred and twelve pin interface connector. Up to six standard Raypak logic units (reference Section 5.4), can be mounted on the module, with a maximum power dissipation of fifteen watts. Logic units are mounted on a multilayer board which consists of a ground plane, power plane, two signal planes and a top-pad plane.

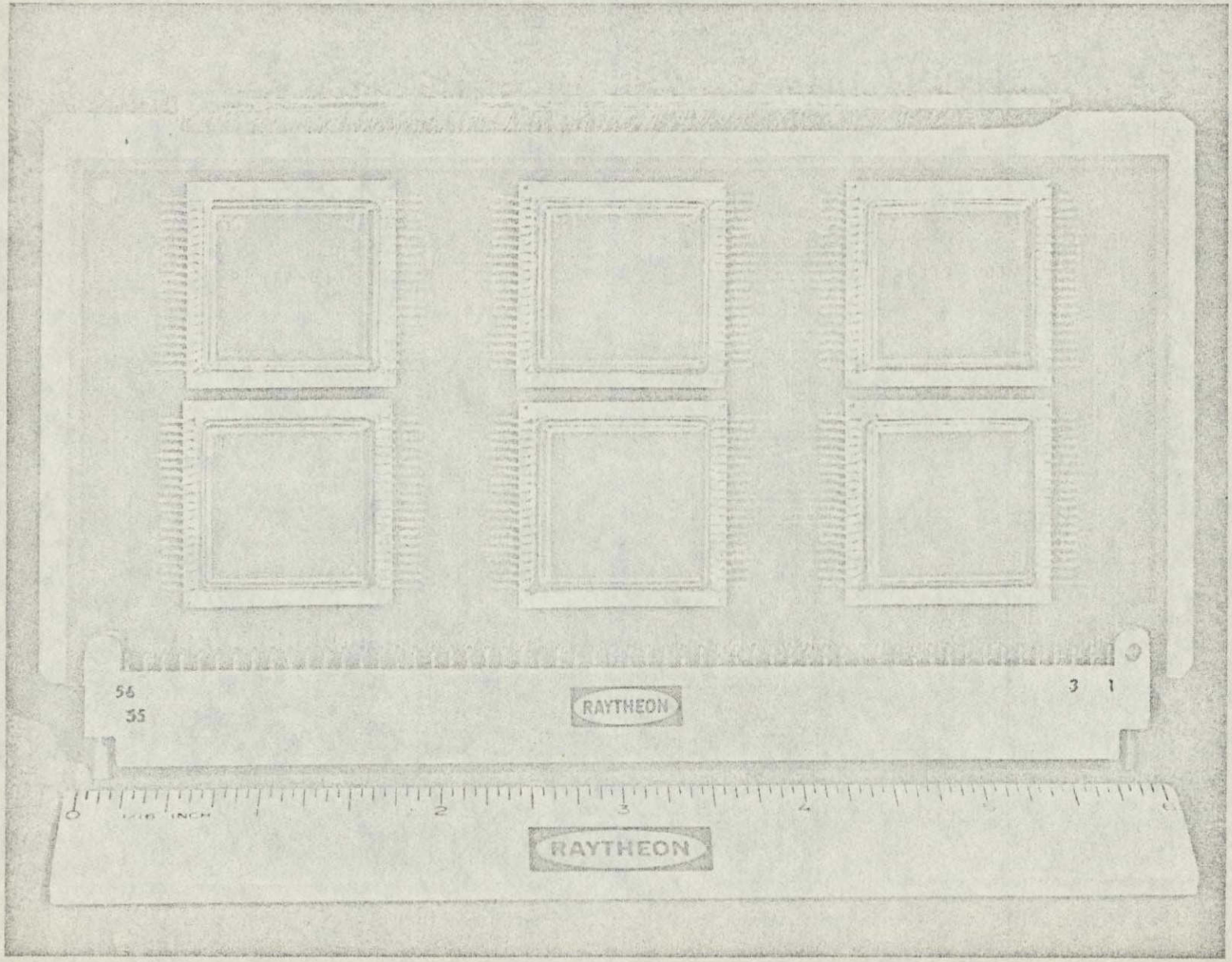


FIGURE 5.6

TYPE "B" LSI MODULE

5.5.3 Gate Density Per Module

The number of gates per module is a variable which depends on pin quantity, heat dissipation and electrical function desired (interconnect area). Of these, the heat dissipation is the nearest fixed. Assuming the use of low power bipolar chips with an average gate dissipation of 2 milliwatts, then the number of gates available is approximately 2000.

5.5.4 Interconnect Area

As previously noted in Section 2.4, a logic function comprised of 300 gates has been interconnected on a 1.15 inch square substrate, therefore, 200 gates should be possible on a .75 inch square Raypak. Since there are four Raypaks per AF module, a total gate density of 800 on a module is feasible.

5.6 COMMAND CONTROLLER LOGIC PACKAGING

The logic gate requirements for implementing the proposed command controller is estimated at 2500, less memory. Using the figure of 800 gates per module estimated in Section 5.5.1, the number of modules required is 4. Since one of the modules is only partially utilized, the read-only program memory could be mounted on it. The program memory capacity, including a possible self-test, is estimated at 75 words by 48 bits, for a total of 3600 bits. This can be packaged in two of the mini-Raypaks.

STORAGE MEMORY

In Section 5.4 the proposed memory packaging density was 2048 bits per Raypak. The estimated requirement for the command controller storage memory is 2048 words by 48 bits, at most, or 40,960 bits. A total of 20 Raypaks, or 5 modules are, therefore, required to implement the memory section.

TOTAL MODULE REQUIREMENT

The total command controller module requirement is 4 + 5 = 9 LSI modules, requiring a package of approximately 4.75 inches by 2 inches by 3 inches.

5.6.1 Power Requirements

The power requirements for the command controller are estimated for implementation with low power bipolar and C-MOS technology.

Bipolar Implementation

Using low power bipolar gates of 2 milliwatts dissipation, the logic power dissipation is equal to 2500 x 2 = 5000 milliwatts or 5 watts.

C-MOS Implementation

The power dissipation of C-MOS logic gates even when switching at maximum rate is approximately 1 milliwatt (reference Section 5.3.1). Therefore, the power dissipation of the equivalent C-MOS logic is one-half that for bipolar logic, equaling 2.5 watts.

Memory Power Dissipation

C-MOS technology only will be considered for the command controller storage memory since bipolar memory cells being produced or in development are of high speed and relatively high power types. Sufficient access speeds are obtainable with C-MOS memory cells and a low-power bipolar chip (1 per Raypak) for address decode. The power dissipation is determined from available figures for a 10^6 bit C-MOS memory which requires 50 watts. A memory capacity of 40,000 bits, then, dissipates approximately

$$\frac{4 \times 10^4}{10^6} \times 50 = 2 \text{ watts.}$$

Total Command Controller Power Dissipation

1. Bipolar Logic and C-MOS memory = 5 + 2 = 7 watts
2. C-MOS Logic and C-MOS memory = 2.5 + 2 = 4.5 watts

The power, volume and weight requirements for the command controllers are summarized in Table 5.6.

TABLE 5.6

POWER, VOLUME AND WEIGHT SUMMARY

	CONFIGURATION	POWER (watts)	VOLUME (In^3)	WEIGHT (lbs.)
1.	Bipolar (Low Power) Logic and C-MOS Memory	7	28.5	3.6
2.	C-MOS Logic and C-MOS Memory	4.5	28.5	3.6

5.7 ENCODER LSI LOGIC PACKAGING

5.7.1 Concatenated Encoder (Ref. Sections 4.3.3.1 and 4.3.3.2).

Module Requirements

Number of logic gates = 285

Number of Modules @ 800 gates/module = 1

Power Requirements

1. Bipolar (Low Power) Implementation

   = 285 gates x 2 milliwatts/gate = 0.57 watts

2. C-MOS Implementation

   = 285 x 1 milliwatt/gate = 0.29 watts

Volume

Packaging volume for one module = 0.5 x 2 x 3 = $3in^3$

Weight

Weight (one module) = approximately 0.4 pounds including supporting hardware.

5.7.2 BCH Encoder #1 (63, 36, 11)

Module Requirement

Number of logic gates = 205

Number of modules @ 800 gates/module = 1

Power Requirements

1. Bipolar (Low Power) Implementation

   = 205 gates x 2 milliwatts/gate = 0.41 watts

2. C-MOS Implementation

   = 205 x 1 milliwatt/gate = 0.2 watts

Volume

Packaging volume (one module) = $3in^3$

Weight

Weight (one module) = 0.4 pounds

5.7.3 BCH Encoder #2 (127, 54, 23)

Module Requirement

Number of gates = 290

Number of modules @ 800/module = 1

Power Requirements

1. Bipolar (Low Power) Implementation

= 290 gates x 2 milliwatts/gate = 0.58 watts

2. C-MOS Implementation

= 290 x 1 milliwatt/gate = 0.29 watts

Volume

Packaging volume (one module) = $3in^3$

Weight

Weight (one module) = 0.4 pounds

5.7.4 Convolutional Encoder (K = 8, V = 3)

Includes an 18-bit input buffer.

Module Requirement

Number of logic gates = 200

Number of modules @ 800 gates/module = 1

Power Requirements

1. Bipolar (Low Power) Implementation

= 200 gates x 2 milliwatts/gate = 0.4 watts

2. C-MOS Implementation

= 200 x 1 milliwatt/gate = 0.2 watts

Volume

Packaging volume (one module) = $3\ in^3$

Weight

Weight (one module) = 0.4 pounds

5.8 DECODER LSI LOGIC PACKAGING

5.8.1 Standard Concatenated Decoder (Ref. Table 4.7)

Module Requirement

Number of gates = 975

Number of modules @ 800 gates/module = 2

Power Requirements

1. Bipolar (Low Power) Implementation

   = 975 gates x 2 milliwatts/gate = 1.95 watts

2. C-MOS Implementation

   = 975 x 1 milliwatt/gate = 0.98 watts

Volume

Packaging volume for two modules = 2 x $3in^3$/module = $6in^3$.

Weight

Weight (two modules) = 2 x 0.4 lbs/module = 0.8 pounds.

5.8.2 Concatenated Decoder with "Green Machine" Inner Decoder (ℓ = 4) Ref. Table 4.14

Module Requirement

Number of logic gates = 1140

Number of modules @ 800 gates/module = 2

Power Requirements

1. Bipolar (Low Power) Implementation

   = 1140 gates x 2 milliwatts/gate = 2.28 watts

2. C-MOS Implementation

   = 1140 x 1 milliwatt/gate = 1.14 watts.

Volume

Packaging volume for two modules = $6in^3$

Weight

Weight for two modules = 0.8 pounds

5.8.3 BCH Decoder #1 (63, 36, 11) Ref. Table 4.4

Module Requirement

Number of logic gates = 2720

Number of modules @ 800 gates/module = 4

Power Requirements

1. Bipolar (Low Power) Implementation

= 2720 gates x 2 milliwatts/gate = 5.4 watts

2. C-MOS Implementation

= 2720 x 1 milliwatt/gate = 2.7 watts

Volume

Packaging volume for four modules =

4 x $3in^3$/module = $12in^3$

Weight

Weight (four modules) = 4 x 0.4 lbs/module =

1.6 pounds

5.8.4 BCH Decoder #2 (127, 54, 23) Ref. Table 4.4

Module Requirement

Number of gates = 4860

Number of modules @ 800 gates/module = 7

Power Requirements

1. Bipolar (Low Power) Implementation

= 4860 gates x 2 milliwatts/gate = 9.7 watts

2. C-MOS Implementation

ϕ = 4860 x 1 milliwatt/gate = 4.9 watts

Volume

Packaging volume for seven modules =

7 x $3in^3$/module = $21in^3$

Weight

Weight (seven modules) = 7 x 0.4 lbs/module =

2.8 pounds

5.8.5 Viterbi Decoder

Table 5.8.1 compares the estimated module and volume requirements of decoders implemented with LSI logic for K = 2 through 10. Packaging densities of 800 gates per module and 2048 memory bits per Raypak (4 Raypaks/module) are obtained from Section 5.5.1 and 5.4, respectively.

The module requirements for K = 7, 8, 9, and 10 are based on a maximum power dissipation of 5 watts/module. This limit is exceeded when requirements are based on logic, due to the greater power consumption of the high speed logic, particularly in the memory area. Volume is based on use of the 80 pin LSI module described in Section 5.5.

TABLE 5.8.1

K (V = 3, ℓ = 2)

	2	3	4	5	6	7	8	9	10
NUMBER OF MODULES	2	2	2	2	3	*4	*6	*9	*16
VOLUME, In^3	6	6	6	6	9	12	18	27	48

*Based on maximum power dissipation of 5 watts/module.

Power Requirements

Power requirements for the decoder range from very low for the configurations which do not require high speed logic, to relatively high for values of K which require implementation with high speed TTL and MECL II logic. Table 5.8.2 lists the estimated powe requirements for the different configurations, based on the following logic dissipation:

LOGIC TYPE		POWER DISSIPATION, MILLIWATTS/PACKAGE
CMOS	(MSI)	10
LOW-POWER TTL	(MSI)	20
STANDARD TTL	(MSI)	200
MECL II	(MSI)	250
MECL II	(SSI)	100
HIGH-SPEED TTL	(SSI)	80
STANDARD TTL	(SSI)	40
LOW-POWER TTL	(SSI)	4

MEMORY TYPE	POWER, MILLIWATTS/BIT
CMOS	0.5
HYBRID; BIPOLAR - MOS	1.0
BIPOLAR	2.0
CURRENT MODE LOGIC	5.0

(For Accumulation Memories, K = 10,

Either CMOS or low power TTL logic can be utilized for K = 2 through 5, standard TTL for K = 6, 7 and 8, while a combination of high-speed TTL and MECL logic is required for K = 9 and 10.

TABLE 5.8.2

POWER REQUIREMENTS, WATTS

LOGIC TYPE	K = 2	3	4	5	6	7	8	9	10
CMOS	0.6	0.65	.75	.87	*6.8	*7.5	N/A	N/A	N/A
LOW-POWER TTL	1.3	1.4	1.5	1.7	**7.6	**9.1	N/A	N/A	N/A
STANDARD TTL	9.2	9.7	10.6	12.0	14.0	18.0	26.0	N/A	N/A
HIGH SPEED TTL & MECL II	---	---	---	---	---	---	---	44	70

*Combination of CMOS (Output Memory, 1/2 of Logic) and Standard TTL.

**Combination of Low Power TTL (Output Memory, 1/2 of Logic) and Standard TTL.

5.8.6 Summary of Logic Packaging for Various Decoder Configurations

The power, volume and weight requirements for two configurations each of the BCH, concatenated, and viterbi decoders are summarized in Table 5.8.3. Logic density is based on the estimated 800 gates per module figure from Section 5.5.1, except for the viterbi decoder where power dissipation requires a lower gate density.

TABLE 5.8.3

DECODER POWER, VOLUME AND WEIGHT SUMMARY

SYSTEM CONFIGURATION	POWER (WATTS) BIPOLAR	POWER (WATTS) C-MOS	VOLUME (In^3)	WEIGHT* (lbs.)
BCH #1 (63, 36, 11)	5.4	2.7	12	1.6
BCH #2 (127, 54, 23)	9.7	4.9	21	2.8
CONCATENATED:				
1. Standard	1.95	0.97	6	0.8
2. Green Machine ($\ell = 4$)	2.28	1.14	6	0.8
VITERBI: (V = 3, $\ell = 3$)				
1. K = 8	26.0	N/A	18	2.4
2. K = 10	70.0	N/A	48	6.4

*Based on estimated module weight (with supporting hardware) of 0.4 pounds.

6.0 RELIABILITY CONSIDERATIONS

This section is concerned with techniques for increasing the reliability of the command system and, in particular, for satisfying the postulated fail-operational/fail-safe requirement. To this end, section 6.1 examines, from a quite general point of view, the potential gain in reliability to be realized from various redundant system configurations. These considerations are then specialized to the space base command system in section 6.2 Finally, as a point of reference, the mean-time-to-failure of the LSI implemented (non-redundant) command controller and that of the concatencated coder/decoder are estimated in section 6.3.

6.1 RELATIVE EFFECTIVENESS OF VARIOUS RELIABILITY CONFIGURATION

Most techniques for increasing the reliability of a system fall into one of four categories:

(1) Error masking techniques (e.g. triple modular redundancy)

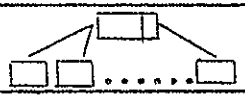
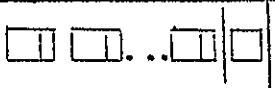
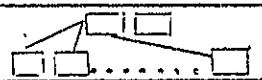
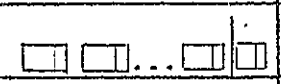
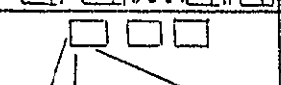
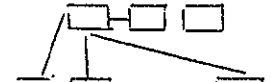
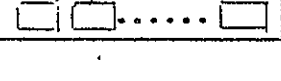
(2) External testing (e.g. supervisory computers)

(3) Self-testing hardware

(4) Self-testing software

Table 6.1 was derived as a first step in assessing the relative effectiveness of these various techniques both singly and in combination. The basic system configuration postulated consists of m identical units, all of which must operate if the system as a whole is to operate; configuration 1 represents this basic system. The remaining configurations considered are as follows:

Table 6.1

	Configuration	Mode	Probability System Operating at Time t	MTF	Prob. of a Failure in First t seconds t<< MTF
1			$e^{-m\alpha t}$	$\frac{1}{m\alpha} \triangleq T$	$m\alpha t \triangleq p$
2		FS	$e^{-(m+1)\alpha t}$	$\frac{m}{m+1} T$	$\frac{m+1}{m} p$
3		FS	$e^{-m\beta t}$	$\alpha/\beta \; T$	$(\beta/\alpha)p$
4		FO	$e^{-m\alpha t} + me^{-[(m-1)\alpha+\beta]t} - me^{-[m\alpha+\beta]t}$	$\left(1+\frac{m^2\alpha^2}{[(m-1)\alpha+\beta][m\alpha+\beta]}\right)T$	$\frac{1}{2}\left[\frac{m-1}{m}+\frac{2}{m}\beta/\alpha\right]p^2$
5		FO	$e^{-m\beta t} + \frac{m\beta}{\beta-\alpha}e^{-[(m-1)\beta+\alpha]t} - \frac{m\beta}{\beta-\alpha}e^{-m\beta t}$	$\frac{\alpha}{\beta}\left[1+\frac{m\beta}{(m-1)\beta+\alpha}\right]T$	$\frac{1}{2}(\beta/\alpha)^2\left[1-\frac{\beta-\alpha}{m\beta}\right]p^2$
6		FO/FS	$e^{-(m+1)\alpha t}[1+(m+1)\alpha t]$	$\frac{2m}{m+1}T$	$\frac{1}{2}\frac{(m+1)^2}{m^2}p^2$
7		FO/FS	$(m+2)e^{-(m+1)\alpha t} - (m+1)e^{-(m+2)\alpha t}$	$\frac{m(2m+3)}{(m+1)(m+2)}T$	$\frac{1}{2}\frac{(m+1)(m+2)}{m^2}p^2$
8		FO/FS	$e^{-m\beta t}(1+m\beta t)$	$2(\alpha/\beta)T$	$\frac{1}{2}(\beta/\alpha)^2 p^2$
9		FO/FS	$(m+1)e^{-m\beta t} - me^{-(m+1)\beta t}$	$\frac{2m+1}{m+1}\alpha/\beta \; T$	$\frac{1}{2}\left(\frac{m+1}{m}\right)(\beta/\alpha)^2 p^2$
10		FO/FO/FS	$[(m+1)^2+1]e^{-(m+1)\alpha t} + (m+1)\alpha t e^{-(m+1)\alpha t} - (m+1)^2(1+\alpha t)e^{-(m+2)\alpha t}$	$\frac{m(3m^2+10m+9)T}{(m+1)(m+2)^2}$	$(1/6)\frac{(m+1)^2(m+3)}{m^3}p^3$
11		FO/FO/FS	$(m+2)^2 e^{-(m+1)\alpha t} - [(m+2)^2-1]e^{-(m+2)\alpha t} - (m+1)(m+2)\alpha t e^{-(m+2)\alpha t}$	$\frac{(3m+4)m}{(m+1)(m+2)}T$	$(1/6)\frac{(m+1)(m+2)^2}{m^3}p^3$
12		FO/FO/FS	$\left[1+m\beta t+\frac{m^2\beta^2t^2}{2}\right]e^{-m\beta t}$	$3\alpha/\beta \; T$	$(1/6)(\beta/\alpha)^3 p^3$
13		FO/FO/FS	$\frac{(m+1)(m+2)}{2}e^{-m\beta t} - m(m+2)e^{-(m+1)\beta t} + \frac{m(m+1)}{2}e^{-(m+2)\beta t}$	$\frac{3m^2+6m+2}{(m+1)(m+2)}\left(\frac{\alpha}{\beta}\right)T$	$\frac{(m+1)(m+2)}{m^2}\frac{1}{6}\left(\frac{\beta}{\alpha}\right)^3 p^3$

2- (m + 1) identical units, one of which sequentially monitors each of the others - The system shuts down whenever the monitor and the unit being tested disagree.

3- m identical units, each of which has been augmented to provide a self-test capability - The system shuts down whenever one unit fails. (The equipment complexity has increased in the process by a factor of β/α. Typically, β/α will be on the order of 1.1 or less (e.g. only two additional bits are needed to detect a single error in an 18 bit adder, so, in this case, $\beta/\alpha = \frac{10}{9} \approx 1.11$).

4- m units plus one self-testing monitor - Whenever the monitor detects a failure, either within itself or in one of the units, that unit is replaced. The system continues to operate without further monitoring.

5- m self-testing units plus a non-self-testing spare. The first unit to fail is replaced by a spare.

6- (m + 2) identical units, one of which serves as a monitor and the second as a spare. When a first failure is detected the spare is used to determine whether it is in the monitor or in the unit being tested; it then replaces the defective unit. The system then continues to operate as in configuration 2.

7*- Same as 6, except that the spare also helps monitor the other units and hence is kept running.

8- (m + 1) identical self-testing units, one of which serves as a spare.

9*- Same as 8 except that the spare is kept operating even when it is not needed.

10- (m + 3) identical units, including one monitor and two spares. Operates as in mode 6, except that now two replacements are possible, so three failures are needed to shut down the system.

11- Same as 10 except one of the spares is also kept in operation.

12- (m + 2) identical self-testing units including two spares. First two detected failures result in a * replacement, the third in a shutdown.

13- Same as 12 except that both spares are kept operating.

It will be noted that configurations 2 and 3 are fail-safe (FS), 4 and 5 fail operational (FO), 6 through 9 fail operational/ fail safe (FO/FS), and 10 through 13 fail operation/ fail operational/fail safe (FO/FO/FS). The remainder of the columns in Table 6.1 are self-explanatory. The results were derived under the usual assumption of exponentially time-dependent failure probabilities.

Note that for purposes of an anlysis of this sort, the four categories list above degenerate, into only two distinct categories. Categories (1) and (2) differ only in the action taken when an error is detected. In category (1) the erroneous unit is ignored because of the masking logic; in category (2), it is switched out of the system. (The relative merits of the two approaches will be discussed in greater detail presently.) Similarly, categories (3) and (4) differ only in that one method requires additional equipment to be associated with each subunit while the second requires more time. (In the latter case, β/α is essentially unity.)

Table 6.1 immediately suggests the following conclusions: If the system is to tolerate ℓ failures with the last failure leading to a fail-safe condition (a situation requiring either ℓ additional non-self-testing or $(\ell-1)$ additional self-testing units) the mean-time-to-failure will be roughly ℓ times that associated with the basic system, and the short-term failure probability proportional to the ℓ^{th} power of the basic system failure probability.

* Configurations 7,9,11 and 13 were included, in part, to guage the advantage in keeping some of the units shut down until they are actually needed.

(If the last failure leads to a fail-operational mode, the mean-time-to-failure and the short-term failure probability are altered as before, but with ℓ replaced by $\ell + 1$.) These general trends, however, are modified by the parameter m, the number of identical units in the basic system. Perhaps the most surprising observation is that the mean-time-to-failure and the short-term failure probability are, respectively, in nearly every configuration, monotonically increasing and monotonically decreasing functions of m. Moreover, the percentage of the total system complexity represented by monotoring or self-test equipment is obviously a monotonically decreasing function of m. Evidently, whenever it is possible to subdivide a system into a number of essentially equivalent subunits, it is advantageous to do so before applying any of the above-mentioned reliability techniques.*

(This statement would be obvious were the added hardware kept constant, independent of the number of subunits as, for example, when each subunit is triplicated. It is less obvious, however, when, as here, the amount of peripheral monotoring equipment decreases as the number of units is increased.)

Table 6.2 illustrates the effect of m on the mean-time-to-failure, the short-term failure probability, and the percentage increase in equipment complexity associated with the various configurations of Table 6.1. Scanning this table, one concludes that when m is large a self-test capability is of questionable merit. Essentially the same performance results with less increase in equipment complexity without this capability. (Compare, for example, configurations 6 and 8 or 10 and 12. When m is small, however, the advantages of a self-test capability are readily apparent. (Again, compare configurations 6 and 8 or 10 and 12, this time with $m = 1$).

* This conclusion does require qualification, however; see below.

Table 6.2

Comparison of Parameters when m = 1 and when m>>1.

Configuration	MTF		Short Term Failure Probability		Incremental Equipment Complexity	
	m = 1	m>>1	m = 1	m>>1	m = 1	m>>1
1	T	T	p	p	0	0
2	$\frac{1}{2}$T	T	2p	p	100%	0
3	α/β T	(α/β) T	(β/α) p	(β/α) p	$\frac{\beta-\alpha}{\alpha} \times 100\%$	$\frac{\beta-\alpha}{\alpha} \times 100\%$
4	$\left[1+\frac{\alpha^2}{\beta(\alpha+\beta)}\right]$ T	2T	$(\beta/\alpha)p^2$	$\frac{1}{2}p^2$	$(\beta/\alpha) \times 100\%$	0
5	$(1 + \alpha/\beta)$T	$2(\alpha/\beta)$T	$\frac{1}{2}(\beta/\alpha)\ p^2$	$\frac{1}{2}(\beta/\alpha)^2 p^2$	$(\beta/\alpha) \times 100\%$	$\frac{\beta-\alpha}{\alpha} \times 100\%$
6	T	2T	$2p^2$	$\frac{1}{2}p^2$	200%	0
7	5/6 T	2T	$3p^2$	$\frac{1}{2}p^2$	200%	0
8	$2(\alpha/\beta)$T	$2(\alpha/\beta)$T	$\frac{1}{2}(\beta/\alpha)^2 p^2$	$\frac{1}{2}(\beta/\alpha)^2 p^2$	$[2(\beta/\alpha)-1] \times 100\%$	$(\frac{\beta-\alpha}{\alpha})100\%$
9	$(3/2)(\alpha/\beta)$T	$2(\alpha/\beta)$T	$(\beta/\alpha)^2 p^2$	$\frac{1}{2}(\beta/\alpha)^2 p^2$	$[2(\beta/\alpha)-1] \times 100\%$	$(\frac{\beta-\alpha}{\alpha})100\%$
10	(11/ 9) T	3T	$(8/3)\ p^3$	$(1/6)\ p^3$	300%	0
11	(7/6) T	3T	$3p^3$	$(1/6)\ p^3$	300%	0
12	$3(\alpha/\beta)$T	$3(\alpha/\beta)$T	$(1/6)(\beta/\alpha)^3 p^3$	$(1/6)(\beta/\alpha)^3 p^3$	$[3(\beta/\alpha)-1 \times 100\%]$	$(\frac{\beta-\alpha}{\alpha})100\%$
13	$(11/6)(\alpha/\beta)$T	$3(\alpha/\beta)$T	$(\beta/\alpha)^3 p^3$	$(1/6)(\beta/\alpha)^3 p^3$	$[3(\beta/\alpha)-1 \times 100\%]$	$(\frac{\beta-\alpha}{\alpha})100\%$

In some cases (e.g. configuration 7) the added redundancy actually decreases the mean-time-to-failure (MTF) when m is small. (It is interesting to note that when m = 1 configuration 7 is, so far as the parameters in Table 1 are concerned, identical to a triple-modular-redundant (TMR) configuration. The only difference is that the TMR system, while avoiding the switching problem, can fail in a non-fail-safe condition.) This decrease in MTF is less significant than it might first appear. The short-term-failure probability is still much smaller in configuration 7 than in configuration 1, even when m = 1. Moreover, when several configuration 7 subsystems are concatenated, the MTF of the overall system will actually be greater than that of the corresponding non-redundant system, even though the MTF of the individual subsystems is less. Specifically, the ratio of the MTF of the redundant (configuration 7) system to the corresponding non-redundant systems is

$$n \sum_{i=0}^{n} \binom{n}{i} (-1)^i \frac{3^{n-i}\, 2^i}{(2n+i)}$$

where n denotes the number of concatenated subsystem in the overall system. Thus, even for n = 2, the MTF of the overall system is increased by a factor of 31/30.

One final comment is in order regarding the results summarized in Tables 6.1 and 6.2. It was assumed, in deriving these results, that when a failure occurred it was detected, and the proper switching accomplished with probability one. Moreover, the switching needed for the monitor to operate properly was also assumed to be immune to error. Obviously these assumptions are acceptable only if the switches and associated logic are at least an order-of-magnitude more reliable than the units themselves. This, in turn, imposes an upper limit on the optimum value of m; as m increases, the associated switching logic necessarily becomes more complicated while each individual unit becomes less complex. The assumption that the switches are much more reliable than the units clearly eventually ceases to be valid.

(The switching circuitry can, of course, also be redundant thereby mitigating this problem, but eventually a decision must be made as to which of these circuits should actually be used.)

6.2 COMMAND SYSTEM RELIABILITY CONSIDERATIONS

The command system must tolerate a single failure and remain in an operational condition, while a second failure must lead to a fail-safe condition. Some means must, therefore, be provided for detecting failures and performing required action to assure a fail-operational/fail-safe (FO/FS) configuration.

Various system reliability and error detection techniques and their effectiveness were discussed in the preceding section. It was noted there that system redundancy can be implemented in essentially two ways, by pure hardware redundancy and by hardware redundancy combined with a self-test or self-monitoring capability.

6.2.1 Pure Hardware Redundancy

High reliability on a real-time basis can be provided by a switchable triple redundancy configuration (Figure 6.1) In this configuration a switchable voting element is used to detect a difference between the three channel inputs to the voter M, and to identify the failed channel, as well as performing the basic voting functions. Upon detection of a disagreement in input signals, the voter switches out the failed channel. The system continues in operation (fail operational) with the voter now comparing the remaining two channel inputs.

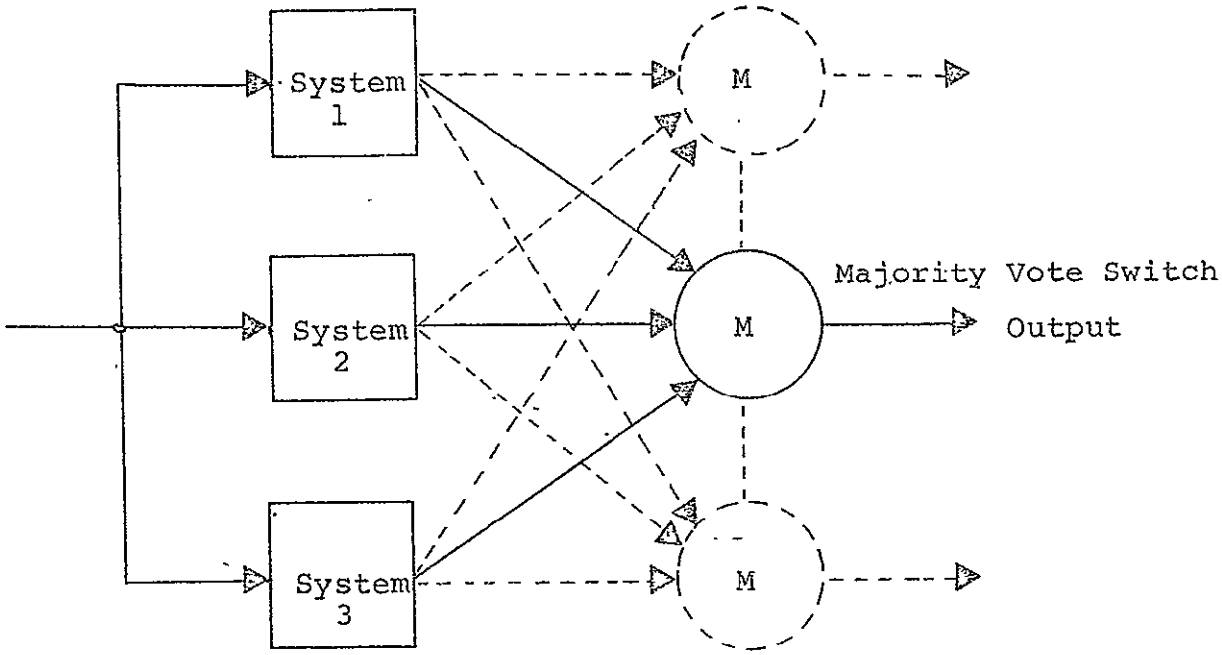


Figure 6.1 Switchable Triple Redundancy

at System Level

Upon a disagreement between the remaining two channels, both are shut down (fail safe). The critical element in this configuration is the switchable voting element. The voter logic can be implemented in triplicate to guard against such failures (Figure 6.1)

The effectiveness of this technique can be carried even further by the partitioning of each system into subunits, or modules. The overall configuration would then appear as in Figure 6.2.

An error resulting from a failure in module 1 of system 3 passes only as far as voter 1.

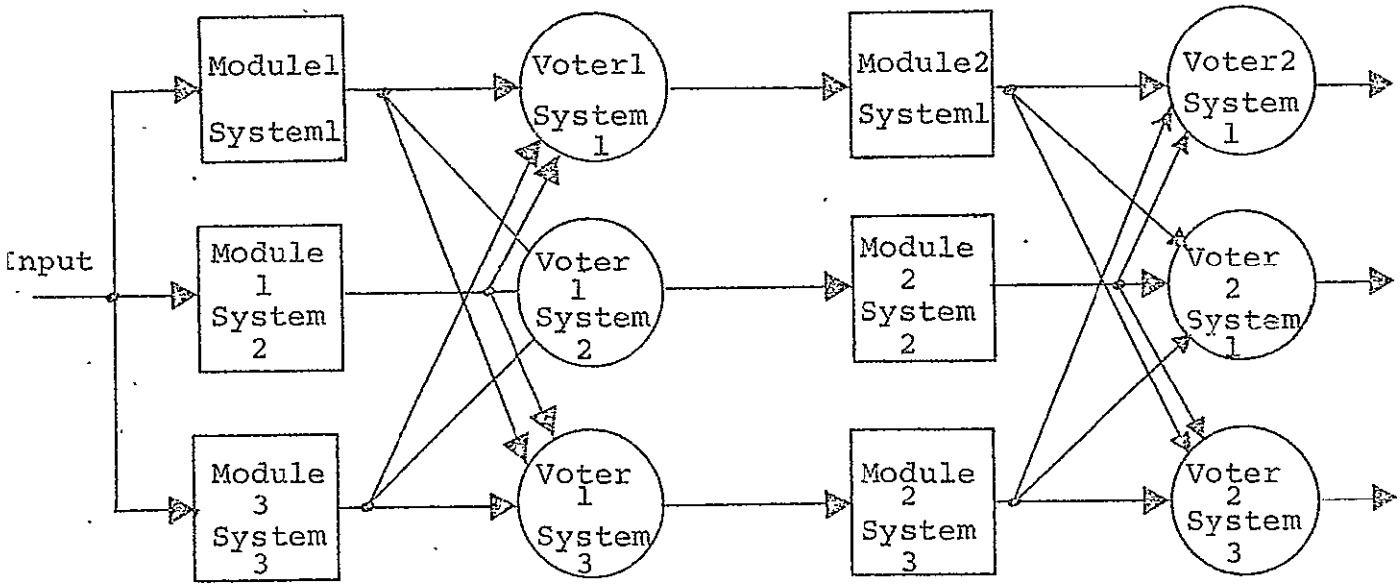


Figure 6.2 Switchable Triple Redundancy at Subsystem Level

Each section of the voter has a good input signal from module 1 of system 1 and module 1 of system 2, and a faulty input from module 1 of system 3. The bad input is outvoted by the two good inputs and the voter output to all three following systems is good. Module 1 of system 3 is switched out and inputs from module 1 of system 1 and module 1 of system 2 compared. A disagreement between these two causes the system to be shut down.

In the command system most of the interfaces between modules as discussed for Figure 6.2 would be data transfer buses of at least 18 bit size. The logic required to implement the voters in triplicate would approach fifty percent of that required for a single system and is considered impractical for this application.

Some of the desirable features of the configuration (i.e, that illustrated in Figure 6.1) are:

1. Automatic failure detection and fault isolation is achieved by means of the switchable disagreement detector network without need for special test routine or diagnostics.
2. Permits the manual replacement of a failed unit without interrupting system operation.
3. Provides a means for detecting and compensating for logic failures before the resulting errors can be propagated to the output of the system.

The main disadvantage to this configuration is the amount of hardware required, i.e., a triplicated command system plus voter-switching logic. The voter-switching operation in this case could be performed on a serial basis, keeping the logic requirements for this item relatively small.

6.2.2 Hardware Redundancy Combined With Self-Monitoring

An FO/FS configuration involving duplication rather than triplication is possible if some means of self-monitoring is also provided (cf. Table 6.1, configures 6 and 8). The most efficacious method of applying this self-monitoring capability, of course, depends to a great extent on the failure modes expected.

The expected failure mode can be determined to a great extent by the manner in which the logic is partitioned on a LSI chip. Registers can be organized on a chip on a bit basis or on a word basis. With bit organization only one bit of a register is located on one chip. With word organization more than one bit, or even the entire word, can be located on one chip. The failure of a chip in the latter case will cause more than one bit per word to fail complicating somewhat this monitoring task. With bit organization, in contrast, although a chip failure can cause the loss of a bit in many different words, only one bit per word is lost, and this can be detected, for example, with a single-parity error check.

Semiconductor memories can also be organized on a bit or word basis. Organizing the memory such that one bit of many words is located on one chip provides bit organization. With word organization more than one bit of a word or words are located on a chip. Here again, the disadvantage of the memory organized on a word basis is the problem of detecting multiple bit errors. In addition, the word-organized memory chip requires more I/O connections, making the system inherently less reliable.

If logic partitioning requires that more than one bit of a register or memory word be placed on a chip, the addition of several parity bits would be required for monitoring purposes. Even this requirement need not complicate the hardware excessively, however. Parity generator/checkers with twelve inputs, for example, are presently available in a single dual-in-line package.

Although the use of simple parity techniques detects errors incurred in the transfer of a command word from one location to another, the possibility still remains that a failure in the control section will cause a more basic malfunction of the command system. While it is unlikely that such a failure could alter the content of a command word, it could cause words to be transmitted in an incorrect priority sequence, or prevent the transmission of some words. The detection of failures in the control section requires a self-test operation of some form.

6.2.2.1 Self-Testing Hardware

The self-test program proposed here is contained in the control memory along with normal program sequences. Anytime the command controller goes into an idle mode of operation with no input or output activity, the self-test program is initiated. The controller returns to the idle mode upon successful completion of the test; however, the program is repeated after a prescribed length of time should the controller remain in idle mode for an extended period of time. The principle steps of the self-test program are as follows.

Step 1. Load a priority 1 word into input buffer, causing an input interrupt.

Step 2. Controller performs normal input sequence for a priority 1 word.

Step 3. Controller now goes into an output mode and continually outputs the command word until a simulated telemetry acknowledgements (type 0 & 1) are generated. The output word is compared with word loaded into the input buffer. The type 0 acknowledgement allows the output transmission to continue, while a type 1 acknowledgement stops the output. If the input and output words compare, and the acknowledgement logic functions properly, the test continues.

Step 4. The self-test program repeats steps 1 and 2 for a priority 2 command. The output mode is initiated and the command transmitted. A simulated type 0 acknowledgement causes the word to be retransmitted, while a type 1 advances the test if the acknowledgement logic is functioning properly.

This procedure is repeated for priority 3 and 4 data transfer words, the entire test requiring approximately 50 millisec. The additional logic required to implement this self-test is a small percentage of the command controller logic and the extra program control memory storage roughly 10% of that for the functional program.

6.2.2.2 Dual Redundancy Switching

In order to provide an operational configuration following a first failure, a second command system must operate in parallel (Figure 6.3) to prevent loss of any data stored in the failed units memory. The detection of a failure in the first system would initiate a switching "in" of the back-up system, and switching "out" of the failed unit. A unit would be switched out if one or both of the following two conditions occurred:

1. A self-test failure in that unit.
2. A parity error in that unit.

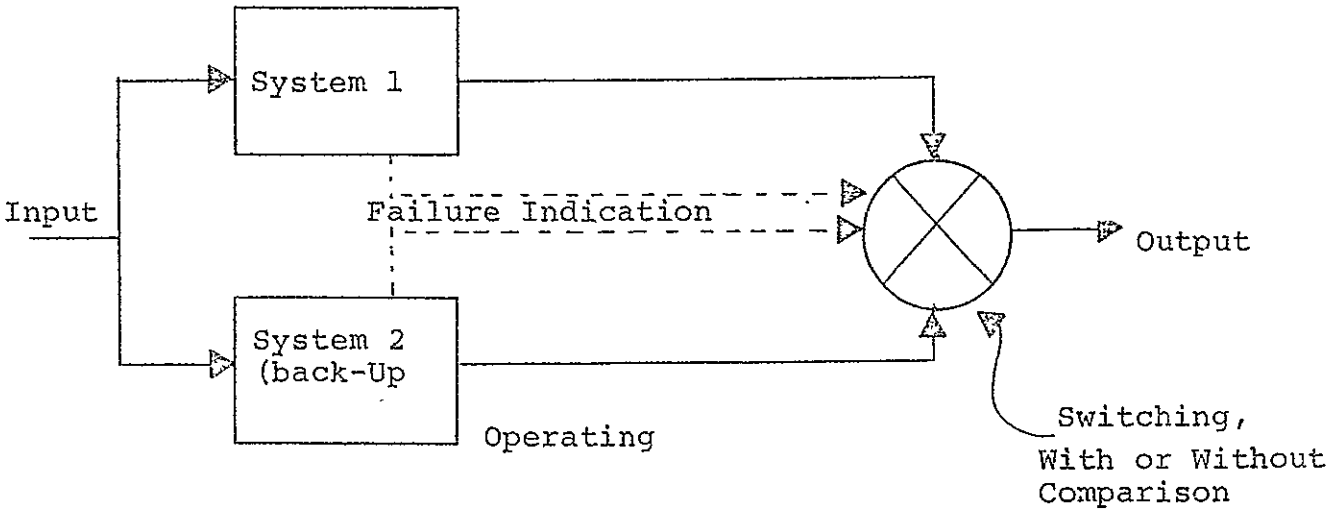


Figure 6.3

Operating Spare Unit at System Level

(If both units operate simultaneously, a disagreement in their outputs could be used to initiate the self-test program.) After one unit has been switched off, the remaining unit continues operation until a parity error or self-test error occurs, causing that unit to be shut down (fail safe). This technique can also be configured as shown in Figure 6.4, providing increased reliability at a cost of additional switching logic. Presumably, the command system would be separated into at least two subunits, the controller subunit and the coder/decoder subunit, for most efficient operation. Most likely, it would be further subdivided, but the exact nature of this subdivision will depend in part upon the way in which the system is partitioned into LSI modules.

A desirable feature of the combination signal redundancy and self-testing configuration is that a system which has been shut down because of a transient error can be returned to operation following a successful self-test.

6.3 NON-REDUNDANT COMMAND SYSTEM MTF

6.3.1 Controller MTF

The command controller can be divided into two major functional units for purposes of reliability analysis, a central processor unit (CPU) and a memory unit. The CPU

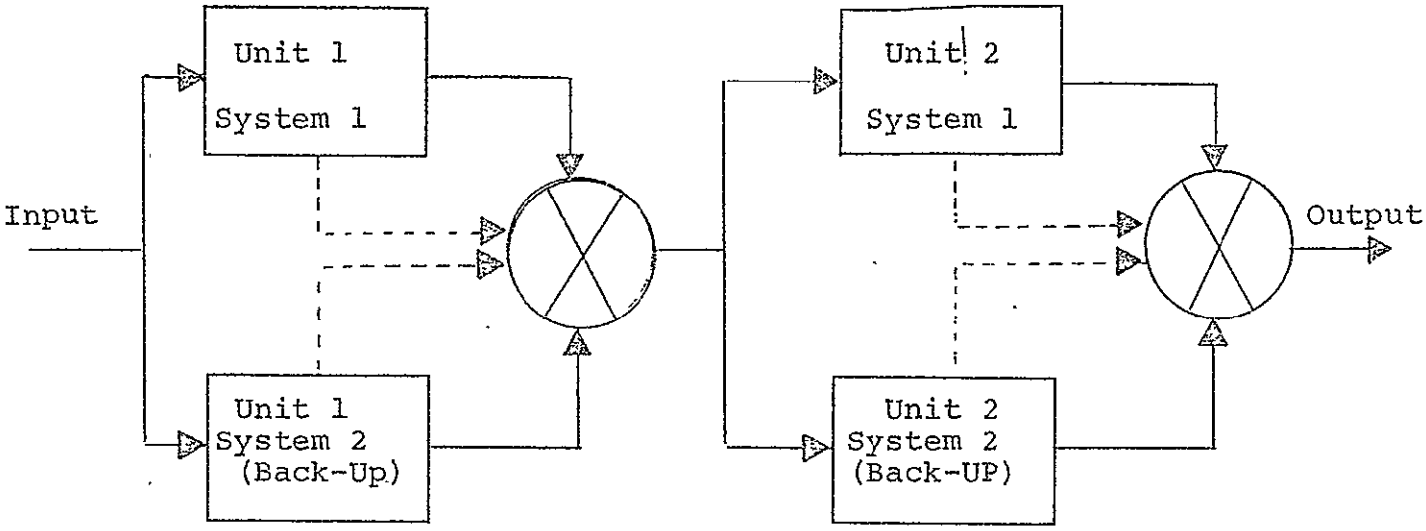
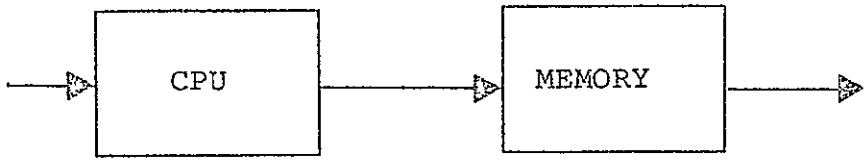


Figure 6.4

Operating Spare Units at Subsystem Level

section contains telemetry counters, arithmetic logic, buffer registers and the control section. The reliability configuration for the controller appears then as follows:



Assuming a LSI packaging technique such as the RAYPAK (described in section 5 of this report)for both CPU and memory, the reliability for each RAYPAK works out as follows:

Element		$\lambda(10^{-6})$
8 chips/RAYPAK	$8 \times .03 \times 10^{-6}$	.24
50 leads/chips	$8 \times 50 \times .00001 \times 10^{-6}$ /bond	.004
40 solder leads/RAYPAK	$40 \times .0001 \times 10^{-6}$ /solder joint	.004

$$\Sigma \lambda = .248 \times 10^{-6} \text{/hour.}$$

The RAYPAK requirement for each functional unit (ref. section 5.6) is:

CPU	12 RAYPAKS	$\frac{\lambda\ (10^{-6})}{12 \times .248 = 2.976}$
Memory	16 RAYPAKS	$16 \times .248 = 3.968$

MTF = 144,000 Hrs. Total $\lambda = 6.944 \times 10^{-6}$

If a power supply is included in the reliability analysis the MTF might be considerably less. A possibility exists of powering down the system during periods of inactivity, reducing the operational duty cycle and thereby increasing the MTF.

6.3.2 Concatenated Encoder/Decoder MTF

6.3.2.1 Concatenated Encoder

Two Raypak type logic units are required to implement the proposed encoder design with LSI. The MTF for the encoder can then be calculated as follows:

Using the failure rate calculated in Section 6.3.1

for a Raypak, $= .248 \times 10^{-6}$/hour

the total encoder failure rate $= 2 \times .248 \times 10^{-6}$

$= .496 \times 10^{-6}$/hour

MTF = 2,016,000 hours.

6.3.2.2 Concatenated Decoder (with Standard Inner Decoder)

The concatenated decoder with standard inner decoder requires six Raypaks. The total failure rate $= 6 \times .248 \times 10^{-6}$/hour

$= 1.488 \times 10^{-6}$/hour.

MTF = 670,000 hours.

7.0 CONCLUDING REMARKS

The controller design (Ref. Section 3) selected for the digital command system consists basically of random access memory for command word buffer-storage (approximately 1200 words), 512 words for verification word storage, and a control section utilizing a read-only program memory. Investigation of the program timing required for control sequences (Section 3) indicates that C-MOS logic is suitable for implementing controller logic. The power, weight and volume requirements for the proposed command controller utilizing LSI logic are listed below:

Power Dissipation	=	4.5 Watts
Weight	=	3.6 Lbs.
Volume	=	28.5 In^3

After a detailed examination of several possible coding strategies, it was concluded that the most attractive scheme for the present application involves a concatenated approach, combining a (15, 9) Reed-Solomon Outer Code with an (8, 4) biorthogonal inner code. The rationale for this decision can be found in Section 4.5. This conclusion was reinforced in Section 5 when the LSI implementation of the contending decoders was investigated. As shown there, the complete concatenated code decoder could be implemented using 6 RAYPAKS while the complete encoder requires only 2 more RAYPAKS.

The complete space base command system then could be packed in a volume of 37.5 in^3 weighing only 4.8 lbs. and requiring only 5.5 watts of power (C-MOS configuration). Since only the decoder is required on the subsidiary vehicles, these same figures become, in this case, 6.0 in^3, 0.8 lbs., and 1 watt, respectively. The command system proposed here should therefore be compatable with even the simplest of subsidiary vehicles likely to be associated with a space base.

Some remarks are in order concerning the extent to which the conclusions of this report are tied to the 10^{-18} error probability requirement. If this requirement were relaxed somewhat would a significantly less complex decoder suffice? Or, alternatively, could a decoder of comparable complexity achieve a significantly lower rejection rate if the error probability were allowed to be larger?

Obviously, the answers to these questions are highly dependent on the amount by which the acceptable error probability (P_e) is increased. It seems clear, however, that P_e must be increased considerably to achieve a major reduction in the decoder complexity. The recommended decoder is about as simple as a 36-bit constraint length decoder can be; i.e. its major component is a 36-bit shift-register with a relatively small amount of peripheral logic. Since the basic command word length is 18, the decoder must at the minimum include an 18-bit register.

The decoder complexity could of course be halved by reducing the constraint length from 36 to 18, but both the error and rejection probability exponents would be essentially halved in the process (i.e. the error probability would increase to on the order of 10^{-9} and the rejection probability to nearly 10%.) Thus, there seems to be little chance of significantly reducing the decoder complexity without a major increase in the error probability.

It is, of course, possible to decrease the word rejection rate at the expense of an increase in the erro probability. But to do so using algebraic decoding techniques necessitates the use of more sophisticated error and erasure correction algorithms. Such algorithms inevitably result in a sizable increase in decoder complexity. One rather attractive possibility for decreasing the rejection probability, for example, is simply to modify the concatenated decoding algorithm recommended here by increasing the erasure correction distance from $d_1 = 3$ to $d_1 = 5$. This would reduce P_r from approximately 3×10^{-3} to approximately 10^{-4} with only a modest increase in error probability (see the appendix to this report). Even this change, however, would necessitate a major change in the decoding algorithm and hence a considerably more complex decoder. Another possibility would be to use a convolution code with a

constraint length of, say, K = 8, and to decode it with a Viterbi decoder, but with a less stringent rejection criterion than that needed to achieve a 10^{-18} error probability. At sufficiently large error probabilities (say on the order of 10^{-12} this method would probably out perform any algebraic decoder of comperable complexity. Even so, however, the resulting decoder would be significantly more complex, and, because of the faster logic required, would consume significantly more power, than the decoder recommended in this report.

It would thus appear that the rejection probability cannot be significantly decreased without a large increase in the error probability, or in the complexity of the decoder, or both. Moreover, the advantage in reducing the rejection rate from the 0.3% attainable using the approach recommended here to, say, 0.01% or less, is questionable. As long as any rejections are allowed, the system must store all messages until they are acknowledged. Consequently, the complexity of the overall system (excluding the decoder) is little affected by a reduction in the rejection rate. And at a 0.3% rejection rate only about 0.3% of the messages which must be transmitted are repeat messages, so the capacity of the command link cannot be noticably increased by a reduction in P^r. In view of the cost of such a reduction measured in terms of the error probability or of the decoder complexity, the exchange seems decidedly unfavorable, at least under the constraints operative here.

APPENDIX

SELECTED COMPUTER PRINT-OUTS

This appendix is divided into three parts. The first part shows the computer print-out for various codes and code parameters discussed in Section 4. The second shows the error and erasure probabilities, p and q respectively, for the (7, 4) and (8, 4) binary codes, and the (15, 9) and (15, 11) Reed-Solomon codes when the symbol error and erasure probabilities are the previously determined values of p and q. Part three is similar to part two but with the inner code error and rejection probabilities calculated using the exact expressions derived in Section 4.3.2.2.

As a point of reference, the ratio R (defined as the ratio of the signal energy per information bit to the noise spectral density) needed to attain a bit error probability p of 10^{-2} without coding is R = 2.71; an R of 4.74 corresponds to $p = 10^{-3}$, and R of 6.55 to $p = 1.6 \times 10^{-4}$, and an R of 6.845 to $p = 10^{-4}$. Note that in every case the symbol signal-to-noise ratio R* has been reduced by the factor $k/n \frac{k_1 k_2}{n_1 n_2}$ in the concatenated code case to reflect the fact that n code bits must be sent in the time allocated to k information bits.

PART I

INPUT R*(36/63)

ALL R VARYING THETA

N= 63 D= 11 D0= 1 D1= 1

R*	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
2.709D+00						
	1.250000D-01	2.62197042950-15	7.34810281900-01	2.65189718100-01	-1.45813722110+01	-1.33824775420-01
	1.666670D-01	8.20115525430-17	8.12443047100-01	1.87556952900-01	-1.60861249660+01	-9.02070736680-02
	2.500000D-01	4.98531919950-20	9.25774289550-01	7.42257104480-02	-1.93023070300+01	-3.34948845390-02
1.549D+00						
	1.250000D-01	3.57612158350-08	9.82020110370-01	1.79798538720-02	-7.44658772420+00	-7.87961839230-03
	1.666670D-01	3.02189758780-09	9.90500961470-01	9.49903550530-03	-8.51972025800+00	-4.14509855840-03
	2.500000D-01	1.51429344200-11	9.97929044180-01	2.07095580640-03	-1.08197899580+01	-9.00337287070-04
3.743D+00						
	1.250000D-01	6.29232307500-22	4.09724766260-01	5.90275233740-01	-2.12011889870+01	-3.87507783850-01
	1.666670D-01	7.48069124250-24	5.11391738490-01	4.88608261510-01	-2.31260582700+01	-2.91246292260-01
	2.500000D-01	3.37560793940-28	7.21466350300-01	2.78533649700-01	-2.74716480010+01	-1.41783919900-01

ALL THETA VARYING R

N= 63 D= 11 D0= 1 D1= 1

THETA	R*	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.250D-01						
	4.740000D+00	2.62197042950-15	7.34810281900-01	2.65189718100-01	-1.45813722110+01	-1.33824775420-01
	2.710000D+00	3.57612158350-08	9.82020110370-01	1.79798538720-02	-7.44658772420+00	-7.87961839230-03
	6.550000D+00	6.29232307500-22	4.09724766260-01	5.90275233740-01	-2.12011889870+01	-3.87507783850-01
1.667D-01						
	4.740000D+00	8.20115525430-17	8.12443047100-01	1.87556952900-01	-1.60861249660+01	-9.02070736680-02
	2.710000D+00	3.02189758780-09	9.90500961470-01	9.49903550530-03	-8.51972025800+00	-4.14509855840-03
	6.550000D+00	7.48069124250-24	5.11391738490-01	4.88608261510-01	-2.31260582700+01	-2.91246292260-01
2.500D-01						
	4.740000D+00	4.98531919950-20	9.25774289550-01	7.42257104480-02	-1.93023070300+01	-3.34948845390-02
	2.710000D+00	1.51429344200-11	9.97929044180-01	2.07095580640-03	-1.08197899580+01	-9.00337287070-04
	6.550000D+00	3.37560793940-28	7.21466350300-01	2.78533649700-01	-2.74716480010+01	-1.41783919900-01

INPUT R*(36/63)

ALL R VARYING THETA

N= 63 D= 11 D0= 3 D1= 3

R*	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
2.709D+00						
	1.250D-01	2.23185055690-12	2.33244028790-01	7.66755971200-01	-1.16513348890+01	-6.32189465680-01
	1.666670D-01	2.27949699740-13	2.91638214260-01	7.08361785740-01	-1.26421609760+01	-5.35155569670-01
	2.500D-01	1.26683237060-15	4.93773116980-01	5.06226883020-01	-1.48972808480+01	-3.06472558520-01
1.549D+00						
	1.250D-01	6.02092611470-06	8.50002801640-01	1.49991177430-01	-5.22033670220+00	-7.05796428320-02
	1.666670D-01	1.22447618350-06	8.88165792320-01	1.11832983210-01	-5.91204965760+00	-5.15059576800-02
	2.500D-01	2.87746403090-08	9.53359664170-01	4.66403070600-02	-7.54099009650+00	-2.07432266490-02
3.743D+00						
	1.250D-01	1.86879729430-18	3.49326688080-02	9.65067331190-01	-1.77284378030+01	-1.45676823360+00
	1.666670D-01	9.60440773200-20	4.98262020040-02	9.50173798000-01	-1.90175294110+01	-1.30254221560+00
	2.500D-01	1.34342825820-22	1.40348090760-01	8.59651909240-01	-2.18717855210+01	-8.52793490960-01

ALL THETA VARYING R

N= 63 D= 11 D0= 3 D1= 3

THETA	R	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.250D-01						
	4.740D+00	2.23185055690-12	2.33244028790-01	7.66755971200-01	-1.16513348890+01	-6.32189465680-01
	2.710D+00	6.02092611470-06	8.50002801640-01	1.49991177430-01	-5.22033670220+00	-7.05796428320-02
	6.550D+00	1.86879729430-18	3.49326688080-02	9.65067331190-01	-1.77284378030+01	-1.45676823360+00
1.667D-01						
	4.740D+00	2.27949699740-13	2.91638214260-01	7.08361785740-01	-1.26421609760+01	-5.35155569670-01
	2.710D+00	1.22447618350-06	8.88165792320-01	1.11832983210-01	-5.91204965760+00	-5.15059576800-02
	6.550D+00	9.60440773200-20	4.98262020040-02	9.50173798000-01	-1.90175294110+01	-1.30254221560+00
2.500D-01						
	4.740D+00	1.26683237060-15	4.93773116980-01	5.06226883020-01	-1.48972808480+01	-3.06472558520-01
	2.710D+00	2.87746403090-08	9.53359664170-01	4.66403070600-02	-7.54099009650+00	-2.07432266490-02
	6.550D+00	1.34342825820-22	1.40348090760-01	8.59651909240-01	-2.18717855210+01	-8.52793490960-01

7-8

INPUT R*(36/63)

ALL R VARYING THETA

N= 63 D= 11 D0= 5 D1= 5

R*	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
2.709D+00						
	1.250000D-01	2.74894745210-10	4.18707552230-02	9.5812924450D-01	-9.560833561900+00	-1.3780892057D+00
	1.6666700D-01	7.8527581174D-11	5.5650292100D-02	9.4434970782D-01	-1.01049777800+01	-1.25453255180+00
	2.5000000D-01	3.6143280672D-12	1.3576400353D-01	8.6423599646D-01	-1.44197243000+01	-8.67215363620-01
1.549D+00						
	1.250000D-01	1.46434023240-04	5.90066712810-01	4.09786853170-01	-3.83435800530+00	-2.2909888434D-01
	1.6666700-01	6.3406562929D-05	6.4551225160D-01	3.5442434184D-01	-4.19786578790+00	-1.9009551056D-01
	2.5000000-01	6.3326936705D-06	7.83831145960-01	2.1616252135D-01	-5.19841151950+00	-1.0577748359D-01
3.743D+00						
	1.250000D-01	8.3946696100D-16	1.5125100363D-03	9.9848748996D-01	-1.50759963920+01	-2.82030173490+00
	1.6666700-01	1.5227661690-16	2.2095318860D-03	9.9779046811D-01	-1.58173638010+01	-2.6556997267D+00
	2.5000000-01	3.2927904666D-18	1.0609813772D-02	9.8939018623D-01	-1.74824359040+01	-1.9742922390D+00

ALL THETA VARYING R

N= 63 D= 11 D0= 5 D1= 5

THETA	R*	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.250D-01						
	4.740000D+00	2.74894745210-10	4.18707552230-02	9.5812924450D-01	-9.560833561900+00	-1.3780892057D+00
	2.710000D+00	1.46434023240-04	5.90066712810-01	4.09786853170-01	-3.83435800530+00	-2.2909888434D-01
	6.550000D+00	8.3946696100D-16	1.5125100363D-03	9.9848748996D-01	-1.50759963920+01	-2.82030173490+00
1.667D-01						
	4.740000D+00	7.8527581174D-11	5.5650292100D-02	9.4434970782D-01	-1.01049777800+01	-1.25453255180+00
	2.710000D+00	6.3406562929D-05	6.4551225160D-01	3.5442434184D-01	-4.19786578790+00	-1.9009551056D-01
	6.550000D+00	1.5227661690-16	2.2095318860D-03	9.9779046811D-01	-1.58173638010+01	-2.6556997267D+00
2.500D-01						
	4.740000D+00	3.6143280672D-12	1.3576400353D-01	8.6423599646D-01	-1.44197243000+01	-8.67215363620-01
	2.710000D+00	6.3326936705D-06	7.83831145960-01	2.1616252135D-01	-5.19841151950+00	-1.0577748359D-01
	6.550000D+00	3.2927904666D-18	1.0609813772D-02	9.8939018623D-01	-1.74824359040+01	-1.9742922390D+00

INPUT R*(36/63)

ALL R VARYING THETA

N= 63 K*	U= 11 THETA	U6= 7 PE	U1= 7 1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
2.7090+00	1.25[illegible]0D-01	1.2668756710D-08	4.99858376490-03	9.9500140357D-01	-7.8972660039D+00	-2.3011530257D+00
	1.6666[illegible]0D-01	7.9782755540D-09	6.71405653450-03	9.9328593549D-01	-8.0986909968[illegible]D+00	-2.1730150062D+00
	2.5000000D-01	2.5458455662D-09	2.16139154500-02	9.7838608201D-01	-8.5942959056D+00	-1.6652665518D+00
1.5490+00	1.2500000D-01	1.3176760111D-03	3.1924135010D-01	6.7944157389D-01	-2.8803891603D+00	-4.9588086118D-01
	1.6666700D-01	1.0109109312D-03	3.6306069032D-01	6.3592839875D-01	-2.9952871073D+00	-4.4002077092D-01
	2.5000000D-01	3.8369184288D-04	5.0773272301D-01	4.9190358515D-01	-3.4392664395D+00	-2.9436484572D-01
3.7430+00	1.2500000D-01	1.5058854081D-13	4.2228512270D-05	9.9995777149D-01	-1.2822217593D+01	-4.3743942187D+00
	1.6666700D-01	7.0947567838D-14	5.9008067523D-05	9.9994099193D-01	-1.3149062488D+01	-4.2290886080D+00
	2.5000000D-01	1.9504147050D-14	4.1585640243D-04	9.9958414360D-01	-1.3709873037D+01	-3.3610566078D+00

ALL THETA VARYING R

N= 63 THETA	U= 11 R	U6= 7 PE	U1= 7 1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.2570-01	4.7451[illegible]0D+00	1.2668756710D-08	4.99858376490-03	9.9500140357D-01	-7.8972660039D+00	-2.3011530257D+00
	2.7100[illegible]0D+00	1.3176760111D-03	3.1924135010D-01	6.7944157389D-01	-2.8803891603D+00	-4.9568086118D-01
	5.5500[illegible]0D+00	3.5058452406[illegible]D-13	4.2228512270D-05	9.9995777149D-01	-1.2822217593D+01	-4.3743942187D+00
1.6670-01	4.7451[illegible]0D+00	7.9782755540D-09	6.71405653450-03	9.9328593549D-01	-8.0986909968[illegible]D+00	-2.1730150062D+00
	2.7100[illegible]0D+00	1.0109109312D-03	3.6306069032D-01	6.3592839875D-01	-2.9952871073D+00	-4.4002077092D-01
	5.5500[illegible]0D+00	7.0947567838D-14	5.9008067523D-05	9.9994099193D-01	-1.3149062488D+01	-4.2290886080D+00
2.5000-01	4.7451[illegible]0D+00	2.5458455662D-09	2.16139154500-02	9.7838608201D-01	-8.5942959056D+00	-1.6652665518D+00
	2.7100[illegible]0D+00	3.8369184288D-04	5.0773272301D-01	4.9190358515D-01	-3.4392664395D+00	-2.9436484572D-01
	5.5500[illegible]0D+00	1.9504147050D-14	4.1585640243D-04	9.9958414360D-01	-1.3709873037D+01	-3.3610566078D+00

INPUT R*(36/63)

ALL R VARYING THETA

N= 63 R*	D= 11 THETA	D0= 9 PC	D1= 9 1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
2.7090D+00	1.2500000D-01	3.4220059301D-07	4.3521661022D-04	9.9956444119D-01	-6.4657192422D+00	-3.3612945380D+00
	1.6666700D-01	3.3973968043D-07	5.6893326539D-04	9.9943072649D-01	-6.4688537261D+00	-3.2449386724D+00
	2.5000000D-01	5.2298210853D-07	2.2217259464D-03	9.9777775107D-01	-6.2815131683D+00	-2.6533095130D+00
1.5490D+00	1.2500000D-01	6.7828237492D-03	1.3098800505D-01	8.6222917121D-01	-2.1685894679D+00	-8.8276847214D-01
	1.6666700D-01	7.1381556137D-03	1.5287921939D-01	8.3998262500D-01	-2.1464139885D+00	-8.1565154349D-01
	2.5000000D-01	6.8266979594D-03	2.4577862449D-01	7.4739527755D-01	-2.1658274836D+00	-6.0945589065D-01
3.7430D+00	1.2500000D-01	1.7018522994D-11	8.5069911330D-07	9.9999914928D-01	-1.0769078134D+01	-6.0702240199D+00
	1.6666700D-01	1.3869210630D-11	1.0848267534D-06	9.9999891516D-01	-1.0857948256D+01	-5.9646396130D+00
	2.5000000D-01	3.2122192046D-11	9.9324445934D-06	9.9999006752D-01	-1.0493194826D+01	-5.0029438489D+00

ALL THETA VARYING R

N= 63 THETA	D= 11 R*	D0= 9 PE	D1= 9 1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.2500D-01	4.7400000D+00	3.4220059301D-07	4.3521661022D-04	9.9956444119D-01	-6.4657192422D+00	-3.3612945380D+00
	2.7100000D+00	6.7828237492D-03	1.3098800505D-01	8.6222917121D-01	-2.1685894679D+00	-8.8276847214D-01
	6.5500000D+00	1.7018522994D-11	8.5069911330D-07	9.9999914928D-01	-1.0769078134D+01	-6.0702240199D+00
1.6670D-01	4.7400000D+00	3.3973968043D-07	5.6893326539D-04	9.9943072649D-01	-6.4688537261D+00	-3.2449386724D+00
	2.7100000D+00	7.1381556137D-03	1.5287921939D-01	8.3998262500D-01	-2.1464139885D+00	-8.1565154349D-01
	6.5500000D+00	1.3869210630D-11	1.0848267534D-06	9.9999891516D-01	-1.0857948256D+01	-5.9646396130D+00
2.5000D-01	4.7400000D+00	5.2298210853D-07	2.2217259464D-03	9.9777775107D-01	-6.2815131683D+00	-2.6533095130D+00
	2.7100000D+00	6.8266979594D-03	2.4577862449D-01	7.4739527755D-01	-2.1658274836D+00	-6.0945589065D-01
	6.5500000D+00	3.2122192046D-11	9.9324445934D-06	9.9999006752D-01	-1.0493194826D+01	-5.0029438489D+00

INPUT R*(36/63)

ALL R VARYING THETA

N= 63 D= 11 D0= 11 D1= 11

R*	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
2.7040D+00						
	1.2500D-01	7.0173013982D-06	2.2354679609D-05	9.9997062802D-01	-5.1538298698D+00	-4.6506315501D+00
	1.6666D-01	8.3051337398D-06	2.7973626349D-05	9.9996372124D-01	-5.0806533696D+00	-4.5532512304D+00
	2.5000D-01	3.1284658981D-05	1.2833249663D-04	9.9984038284D-01	-4.5046685747D+00	-3.8916633567D+00
1.5490D+00						
	1.2500D-01	2.6230410880D-02	2.1652298786D-02	9.5211729033D-01	-1.5811949065D+00	-1.6644959886D+00
	1.6666D-01	3.0352745546D-02	2.5611997653D-02	9.4403525680D-01	-1.5178020189D+00	-1.5915565467D+00
	2.5000D-01	4.7663903906D-02	4.9439855273D-02	9.0289675442D-01	-1.3218150694D+00	-1.3059228092D+00
3.7430D+00						
	1.2500D-01	1.5120491211D-09	1.1688120676D-08	9.9999998680D-01	-8.8204340999D+00	-7.9322553131D+00
	1.6666D-01	1.5720991096D-09	1.3261391729D-08	9.9999998517D-01	-8.8035200783D+00	-7.8774108961D+00
	2.5000D-01	1.3432442704D-08	1.4616051531D-07	9.9999984041D-01	-7.8718450032D+00	-6.8351699345D+00

ALL THETA VARYING R

N= 63 D= 11 D0= 11 D1= 11

THETA	R*	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.2500D-01						
	2.7040D+00	7.0173013982D-06	2.2354679609D-05	9.9997062802D-01	-5.1538298698D+00	-4.6506315501D+00
	1.5490D+00	2.6230410880D-02	2.1652298786D-02	9.5211729033D-01	-1.5811949065D+00	-1.6644959886D+00
	3.7430D+00	1.5120491211D-09	1.1688120676D-08	9.9999998680D-01	-8.8204340999D+00	-7.9322553131D+00
1.6667D-01						
	2.7040D+00	8.3051337398D-06	2.7973626349D-05	9.9996372124D-01	-5.0806533696D+00	-4.5532512304D+00
	1.5490D+00	3.0352745546D-02	2.5611997653D-02	9.4403525680D-01	-1.5178020189D+00	-1.5915565467D+00
	3.7430D+00	1.5720991096D-09	1.3261391729D-08	9.9999998517D-01	-8.8035200783D+00	-7.8774108961D+00
2.5000D-01						
	2.7040D+00	3.1284658981D-05	1.2833249663D-04	9.9984038284D-01	-4.5046685747D+00	-3.8916633567D+00
	1.5490D+00	4.7663903906D-02	4.9439855273D-02	9.0289675442D-01	-1.3218150694D+00	-1.3059228092D+00
	3.7430D+00	1.3432442704D-08	1.4616051531D-07	9.9999984041D-01	-7.8718450032D+00	-6.8351699345D+00

INPUT P*(18/79)

ALL R VARYING THETA

N= 79 D= 22 U0= 2 U1= 2

R	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.6800+00						
	1.2500000-01	2.2387945446D-11	9.9859845305D-01	1.4015469279D-03	-1.0649985760D+01	-6.0911105529D-04
	1.6666700-01	9.6597135296D-13	9.9932152015D-01	6.7847984945D-04	-1.2015035753D+01	-2.9476006081D-04
	2.5000000-01	1.0379939059D-15	9.9989252414D-01	1.0747586033D-04	-1.4983805196D+01	-4.6678681542D-05
6.1750-01						
	1.2500000-01	4.3036603327D-06	9.9999153676D-01	4.1593813067D-06	-5.3661418307D+00	-3.6755546966D-06
	1.6666700-01	4.4332203129D-07	9.9999779628D-01	1.7604018016D-06	-6.3532806855D+00	-9.5706615480D-07
	2.5000000-01	3.1571262603D-09	9.9999977022D-01	2.2662457170D-07	-8.5007080494D+00	-9.9792934933D-08
1.492D+00						
	1.2500000-01	2.0313553943D-16	9.7916144154D-01	2.0838558460D-02	-1.5692214089D+01	-9.1456969717D-03
	1.6666700-01	4.2115696613D-18	9.8827904472D-01	1.1720955281D-02	-1.7375556011D+01	-5.1204132351D-03
	2.5000000-01	9.3867499711D-22	9.9753181642D-01	2.4681835774D-03	-2.1027484750D+01	-1.0732435345D-03

ALL THETA VARYING R

N= 79 D= 22 U0= 2 U1= 2

THETA	R	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.250D-01						
	4.7400000+00	2.2387945446D-11	9.9859845305D-01	1.4015469279D-03	-1.0649985760D+01	-6.0911105529D-04
	2.7100000+00	4.3036603327D-06	9.9999153676D-01	4.1593813067D-06	-5.3661418307D+00	-3.6755546966D-06
	6.5500000+00	2.0313553943D-16	9.7916144154D-01	2.0838558460D-02	-1.5692214089D+01	-9.1456969717D-03
1.667D-01						
	4.7400000+00	9.6597135296D-13	9.9932152015D-01	6.7847984945D-04	-1.2015035753D+01	-2.9476006081D-04
	2.7100000+00	4.4332203129D-07	9.9999779628D-01	1.7604018016D-06	-6.3532806855D+00	-9.5706615480D-07
	6.5500000+00	4.2115696613D-18	9.8827904472D-01	1.1720955281D-02	-1.7375556011D+01	-5.1204132351D-03
2.500D-01						
	4.7400000+00	1.0379939059D-15	9.9989252414D-01	1.0747586033D-04	-1.4983805196D+01	-4.6678681542D-05
	2.7100000+00	3.1571262603D-09	9.9999977022D-01	2.2662457170D-07	-8.5007080494D+00	-9.9792934933D-08
	6.5500000+00	9.3867499711D-22	9.9753181642D-01	2.4681835774D-03	-2.1027484750D+01	-1.0732435345D-03

7-13

58876

INPUT R*(18/79)

ALL R VARYING THETA

N= 79 D= 22 D0= 6 D1= 6

R*	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.080D+00						
	1.2500000-01	4.8617193230D-08	9.4212671306D-01	5.7873238327D-02	-7.3132101175D+00	-2.5890682048D-02
	1.6666700-01	9.2148246878D-09	9.5816654516D-01	4.1833445622D-02	-8.0355129228D+00	-1.8558996810D-02
	2.5000000-01	1.3000172446D-10	9.8363377252D-01	1.6366227348D-02	-9.8860508868D+00	-7.1665684164D-03
6.175D-01						
	1.2500000-01	1.1582459385D-03	9.9824506093D-01	5.9669313308D-04	-2.9361992140D+00	-7.6282991082D-04
	1.6666700-01	4.0502427543D-04	9.9923012516D-01	3.6485056216D-04	-3.3925189462D+00	-3.3448116456D-04
	2.5000000-01	2.2804146772D-05	9.9987114203D-01	1.0605382357D-04	-4.6419861725D+00	-5.5965911374D-05
1.492D+00						
	1.2500000-01	1.9448883078D-12	6.7731695405D-01	3.2268304595D-01	-1.1711105335D+01	-1.6920805335D-01
	1.6666700-01	2.1812448030D-13	7.3339366150D-01	2.6660633850D-01	-1.2661295590D+01	-1.3466284784D-01
	2.5000000-01	9.9439951039D-16	8.5955647228D-01	1.4044352772D-01	-1.5002439098D+01	-6.5725851830D-02

ALL THETA VARYING R

N= 79 D= 22 D0= 6 D1= 6

THETA	R	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.250D-01						
	4.7400000+00	4.8617193230D-08	9.4212671306D-01	5.7873238327D-02	-7.3132101175D+00	-2.5890682048D-02
	2.7100000+00	1.1582459385D-03	9.9824506093D-01	5.9669313308D-04	-2.9361992140D+00	-7.6282991082D-04
	6.5500000+00	1.9448883078D-12	6.7731695405D-01	3.2268304595D-01	-1.1711105335D+01	-1.6920805335D-01
1.667D-01						
	4.7400000+00	9.2148246878D-09	9.5816654516D-01	4.1833445622D-02	-8.0355129228D+00	-1.8558996810D-02
	2.7100000+00	4.0502427543D-04	9.9923012516D-01	3.6485056216D-04	-3.3925189462D+00	-3.3448116456D-04
	6.5500000+00	2.1812448030D-13	7.3339366150D-01	2.6660633850D-01	-1.2661295590D+01	-1.3466284784D-01
2.500D-01						
	4.7400000+00	1.3000172446D-10	9.8363377252D-01	1.6366227348D-02	-9.8860508868D+00	-7.1665684164D-03
	2.7100000+00	2.2804146772D-05	9.9987114203D-01	1.0605382357D-04	-4.6419861725D+00	-5.5965911374D-05
	6.5500000+00	9.9439951039D-16	8.5955647228D-01	1.4044352772D-01	-1.5002439098D+01	-6.5725851830D-02

INPUT R*(18/79)

ALL R VARYING THETA

N= 79 O= 22 D0= 10 D1= 10

R°	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.0800+00						
	1.2500000D-01	3.8092349038D-06	6.8424333794D-01	3.1575285282D-01	-5.4191622449D+00	-1.6478942236D-01
	1.6666700D-01	2.1894669158D-06	7.2516843785D-01	2.7482937268D-01	-5.6596616129D+00	-1.3956110635D-01
	2.5000000D-01	3.0653781418D-07	8.2498993385D-01	1.7500975961D-01	-6.5135159437D+00	-8.3551350478D-02
6.1750D-01						
	1.2500000D-01	1.4030242668D-02	9.7467343248D-01	1.2963248573D-02	-1.8529348173D+00	-1.1140871718D-02
	1.6666700D-01	1.1064370631D-02	9.8049633024D-01	8.4392991293D-03	-1.9560732848D+00	-8.5540274524D-03
	2.5000000D-01	3.5064270485D-03	9.9258190999D-01	3.9116629602D-03	-2.4551351922D+00	-3.2336441725D-03
1.4920+00						
	1.2500000D-01	6.2771191870D-10	2.2982421810D-01	7.7017578127D-01	-9.2022396251D+00	-6.3860420875D-01
	1.6666700D-01	2.6287001330D-10	2.6732929310D-01	7.3267070664D-01	-9.5802589529D+00	-5.7295345002D-01
	2.5000000D-01	1.8883963733D-11	4.0491477390D-01	5.9508522608D-01	-1.0723906842D+01	-3.9263637708D-01

ALL THETA VARYING R

N= 79 O= 22 D0= 10 D1= 10

THETA	R	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.2500D-01						
	4.7400000D+00	3.8092349038D-06	6.8424333794D-01	3.1575285282D-01	-5.4191622449D+00	-1.6478942236D-01
	2.7100000D+00	1.4030242668D-02	9.7467343248D-01	1.2963248573D-02	-1.8529348173D+00	-1.1140871718D-02
	6.5500000D+00	6.2771191870D-10	2.2982421810D-01	7.7017578127D-01	-9.2022396251D+00	-6.3860420875D-01
1.6670D-01						
	4.7400000D+00	2.1894669158D-06	7.2516843785D-01	2.7482937268D-01	-5.6596616129D+00	-1.3956110635D-01
	2.7100000D+00	1.1064370631D-02	9.8049633024D-01	8.4392991293D-03	-1.9560732848D+00	-8.5540274524D-03
	6.5500000D+00	2.6287001330D-10	2.6732929310D-01	7.3267070664D-01	-9.5802589529D+00	-5.7295345002D-01
2.5000D-01						
	4.7400000D+00	3.0653781418D-07	8.2498993385D-01	1.7500975961D-01	-6.5135159437D+00	-8.3551350478D-02
	2.7100000D+00	3.5064270485D-03	9.9258190999D-01	3.9116629602D-03	-2.4551351922D+00	-3.2336441725D-03
	6.5500000D+00	1.8883963733D-11	4.0491477390D-01	5.9508522608D-01	-1.0723906842D+01	-3.9263637708D-01

7-15

5987

INPUT R*(18/79)

ALL R VARYING THETA

N= 79 D= 22 D0= 14 D1= 14

R	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.080D+00						
	1.2500000D-01	8.22347592300D-05	3.19251135560D-01	6.80666629680D-01	-4.08494457480D+00	-4.95867549300D-01
	1.6666700D-01	7.72215330940D-05	3.51474658950D-01	6.48448119510D-01	-4.11226158050D+00	-4.54105981810D-01
	2.5000000D-01	5.68672932820D-05	4.58922641910D-01	5.41020490790D-01	-4.24513744240D+00	-3.38260514950D-01
8.175D-01						
	1.2500000D-01	6.09261430760D-02	8.63638504270D-01	7.54353526530D-02	-1.21519631400+00	-6.36680034160-02
	1.6666700D-01	6.20759510130D-02	8.74049350780D-01	6.38746982060D-02	-1.20707661830+00	-5.84640454350-02
	2.5000000D-01	5.50282311090D-02	9.04759554680D-01	4.02122142060D-02	-1.25941444740+00	-4.34668218430-02
1.492D+00						
	1.2500000D-01	5.11276750970D-08	3.81409346850D-02	9.61859014190D-01	-7.29134395530+00	-1.41860866830+00
	1.6666700D-01	4.13671909890D-08	4.55609359400D-02	9.54439022690D-01	-7.38334396860+00	-1.34140736300+00
	2.5000000D-01	2.53818631270D-08	8.46550730450D-02	9.15344901570D-01	-7.59547650220+00	-1.07234701120+00

ALL THETA VARYING R

N= 79 D= 22 D0= 14 D1= 14

THETA	R	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.250D-01						
	4.7400000D+00	8.22347592300D-05	3.19251135560D-01	6.80666629680D-01	-4.08494457480D+00	-4.95867549300D-01
	2.7100000D+00	6.09261430760D-02	8.63638504270D-01	7.54353526530D-02	-1.21519631400+00	-6.36680034160-02
	6.5500000D+00	5.11276750970D-08	3.81409346850D-02	9.61859014190D-01	-7.29134395530+00	-1.41860866830+00
1.667D-01						
	4.7400000D+00	7.72215330940D-05	3.51474658950D-01	6.48448119510D-01	-4.11226158050+00	-4.54105981810-01
	2.7100000D+00	6.20759510130D-02	8.74049350780D-01	6.38746982060D-02	-1.20707661830+00	-5.84640454350-02
	6.5500000D+00	4.13671909890D-08	4.55609359400D-02	9.54439022690D-01	-7.38334396860+00	-1.34140736300+00
2.500D-01						
	4.7400000D+00	5.68672932820D-05	4.58922641910D-01	5.41020490790D-01	-4.24513744240D+00	-3.38260514950D-01
	2.7100000D+00	5.50282311090D-02	9.04759554680D-01	4.02122142060D-02	-1.25941444740+00	-4.34668218430-02
	6.5500000D+00	2.53818631270D-08	8.46550730450D-02	9.15344901570D-01	-7.59547650220+00	-1.07234701120+00

7-16

INPUT R*(18/79)

ALL R VARYING THETA

N= 79 U= 22 U0= 18 U1= 18

R*	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.0800+00						
	1.2500000-01	1.1032825557D-03	9.0674314528D-02	9.0822240292D-01	-2.9573132485D+00	-1.0425157189D+00
	1.6666700-01	1.1561560393D-03	1.0138543808D-01	8.9745840588D-01	-2.9369835479D+00	-9.9402441795D-01
	2.5000000-01	1.6512449318D-03	1.4744658433D-01	8.5090217074D-01	-2.7821885024D+00	-8.3136528364D-01
6.1750-01						
	1.2500000-01	1.8354076374D-01	5.6348918742D-01	2.5297004884D-01	-7.3626746545D-01	-2.4911441305D-01
	1.6666700-01	1.9201791198D-01	5.7609320026D-01	2.3188888776D-01	-7.1665825718D-01	-2.3950725080D-01
	2.5000000-01	2.2022576512D-01	6.0069981558D-01	1.7907441929D-01	-6.5713187248D-01	-2.2134250138D-01
1.4920+00						
	1.2500000-01	2.4859354870D-06	3.4227137419D-03	9.9657480032D-01	-5.6045101460D+00	-2.4656294215D+00
	1.6666700-01	2.4349159655D-06	4.0092970269D-03	9.9598826806D-01	-5.6135160227D+00	-2.3969317680D+00
	2.5000000-01	4.1999205809D-06	8.3164813097D-03	9.9167931877D-01	-5.3767589219D+00	-2.0800603842D+00

ALL THETA VARYING R

N= 79 U= 22 U0= 18 U1= 18

THETA	R*	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.2500-01						
	4.7400000+00	1.1032825557D-03	9.0674314528D-02	9.0822240292D-01	-2.9573132485D+00	-1.0425157189D+00
	2.7100000+00	1.8354076374D-01	5.6348918742D-01	2.5297004884D-01	-7.3626746545D-01	-2.4911441305D-01
	6.5500000+00	2.4859354870D-06	3.4227137419D-03	9.9657480032D-01	-5.6045101460D+00	-2.4656294215D+00
1.6670-01						
	4.7400000+00	1.1561560393D-03	1.0138543808D-01	8.9745840588D-01	-2.9369835479D+00	-9.9402441795D-01
	2.7100000+00	1.9201791198D-01	5.7609320026D-01	2.3188888776D-01	-7.1665825718D-01	-2.3950725080D-01
	6.5500000+00	2.4349159655D-06	4.0092970269D-03	9.9598826806D-01	-5.6135160227D+00	-2.3969317680D+00
2.5000-01						
	4.7400000+00	1.6512449318D-03	1.4744658433D-01	8.5090217074D-01	-2.7821885024D+00	-8.3136528364D-01
	2.7100000+00	2.2022576512D-01	6.0069981558D-01	1.7907441929D-01	-6.5713187248D-01	-2.2134250138D-01
	6.5500000+00	4.1999205809D-06	8.3164813097D-03	9.9167931877D-01	-5.3767589219D+00	-2.0800603842D+00

INPUT R*(18/79)

ALL R VARYING THETA

N= 79 U= 22 U0= 22 U1= 22

R*	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.0800+00						
	1.2500000-01	1.0178553437D-02	6.49171419260-03	9.83329732370-01	-1.99231393900+00	-2.18764060890+00
	1.6666700-01	1.1751190343D-02	7.2439394140D-03	9.8160487024D-01	-1.9526787710D+00	-2.1400251906D+00
	2.5000000-01	1.7174567745D-02	1.1044474367D-02	9.7178095789D-01	-1.7651141846D+00	-1.9568549484D+00
8.1750-01						
	1.2500000-01	4.0957263845D-01	6.8495263817D-02	5.2193209773D-01	-3.8766906416D-01	-1.1643394573D+00
	1.6666700-01	4.2777654802D-01	7.0453935959D-02	5.0176951602D-01	-3.6878302842D-01	-1.1520947397D+00
	2.5000000-01	4.8434083225D-01	7.4028299272D-02	4.4158086848D-01	-3.1480408506D-01	-1.1306022200D+00
1.4920+00						
	1.2500000-01	8.1468380407D-05	1.0096216640D-04	9.9981756945D-01	-4.0890109174D+00	-3.9958413391D+00
	1.6666700-01	8.8179801632D-05	1.1327525979D-04	9.9979854494D-01	-4.0546308825D+00	-3.9458649331D+00
	2.5000000-01	1.8478126336D-04	2.4118802900D-04	9.9957403071D-01	-3.7333420679D+00	-3.6176442515D+00

ALL THETA VARYING R

N= 79 U= 22 U0= 22 U1= 22

THETA	R*	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.2500-01						
	4.7400000+00	1.0178553437D-02	6.49171419260-03	9.83329732370-01	-1.99231393900+00	-2.18764060890+00
	2.7100000+00	4.0957263845D-01	6.8495263817D-02	5.2193209773D-01	-3.8766906416D-01	-1.1643394573D+00
	6.5500000+00	8.1468380407D-05	1.0096216640D-04	9.9981756945D-01	-4.0890109174D+00	-3.9958413391D+00
1.6670-01						
	4.7400000+00	1.1751190343D-02	7.2439394140D-03	9.8160487024D-01	-1.9526787710D+00	-2.1400251906D+00
	2.7100000+00	4.2777654802D-01	7.0453935959D-02	5.0176951602D-01	-3.6878302842D-01	-1.1520947397D+00
	6.5500000+00	8.8179801632D-05	1.1327525979D-04	9.9979854494D-01	-4.0546308825D+00	-3.9458649331D+00
2.5000-01						
	4.7400000+00	1.7174567745D-02	1.1044474367D-02	9.7178095789D-01	-1.7651141846D+00	-1.9568549484D+00
	2.7100000+00	4.8434083225D-01	7.4028299272D-02	4.4158086848D-01	-3.1480408506D-01	-1.1306022280D+00
	6.5500000+00	1.8478126336D-04	2.4118802900D-04	9.9957403071D-01	-3.7333420679D+00	-3.6176442515D+00

81-7

INPUT R=(36/64)

ALL R VARYING THETA

N= 64 D= 10 D0= 3 D1= 3

R	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
2.666D+00						
	1.250000D-01	1.63254482200-10	2.55548749330-01	7.44451250510-01	-9.78713488620+00	-5.92526240170-01
	1.666670D-01	2.23834843010-11	3.17387570630-01	6.82612429350-01	-1.06500723090+01	-4.98410084870-01
	2.500000D-01	2.29921083850-13	5.24172049740-01	4.75827950260-01	-1.26384212020+01	-2.80526140520-01
1.524D+00						
	1.250000D-01	5.93678911940-05	8.66805246200-01	1.33135385910-01	-4.22644837740+00	-6.20784688470-02
	1.666670D-01	1.41183905850-05	9.02115673190-01	9.78702084220-02	-4.85021480750+00	-4.47377717640-02
	2.500000D-01	4.61086377940-07	9.60578205200-01	3.94213337180-02	-6.33621770810+00	-1.74672714040-02
3.684D+00						
	1.250000D-01	7.32613395350-16	4.05319743570-02	9.59468025640-01	-1.51351251450+01	-1.39220224080+00
	1.666670D-01	5.86603922070-17	5.75873976990-02	9.42412602300-01	-1.62451901490+01	-1.23967254620+00
	2.500000D-01	1.86141496550-19	1.57460498260-01	8.42539501740-01	-1.87255154060+01	-8.02828378630-01

ALL THETA VARYING R

N= 64 D= 10 D0= 3 D1= 3

THETA	R	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.250D-01						
	4.740000D+00	1.63254482200-10	2.55548749330-01	7.44451250510-01	-9.78713488620+00	-5.92526240170-01
	2.710000D+00	5.93678911940-05	8.66805246200-01	1.33135385910-01	-4.22644837740+00	-6.20784688470-02
	0.550000D+00	7.32613395350-16	4.05319743570-02	9.59468025640-01	-1.51351251450+01	-1.39220224080+00
1.667D-01						
	4.740000D+00	2.23834843010-11	3.17387570630-01	6.82612429350-01	-1.06500723090+01	-4.98410084870-01
	2.710000D+00	1.41183905850-05	9.02115673190-01	9.78702084220-02	-4.85021480750+00	-4.47377717640-02
	0.550000D+00	5.86603922070-17	5.75873976990-02	9.42412602300-01	-1.62451901490+01	-1.23967254620+00
2.500D-01						
	4.740000D+00	2.29921083850-13	5.24172049740-01	4.75827950260-01	-1.26384212020+01	-2.80526140520-01
	2.710000D+00	4.61086377940-07	9.60578205200-01	3.94213337180-02	-6.33621770810+00	-1.74672714040-02
	0.550000D+00	1.86141496550-19	1.57460498260-01	8.42539501740-01	-1.87255154060+01	-8.02828378630-01

INPUT R*(36/64)

ALL R VARYING THETA

N= 64 D= 11 D0= 5 D1= 5

R*	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
2.666D+00						
	1.250000D-01	1.538169732?0-08	4.94672660670-02	9.5053271855D-01	-7.8129957387D+00	-1.30568209130+00
	1.666670D-01	5.738422915?0-09	6.55002429320-02	9.3449975133D-01	-8.2412074479D+00	-1.1837570693D+00
	2.500000D-01	4.692590457?0-10	1.55141380790-01	8.4485861874D-01	-9.3285873469D+00	-8.09272347560-01
1.524D+00						
	1.250000D-01	1.139453138?0-03	6.20390245660-01	3.7847030120D-01	-2.9433035311D+00	-2.07335039180-01
	1.666670D-01	5.646913053?0-04	6.7553205708D-01	3.2390325161D-01	-3.2481888991D+00	-1.70354036890-01
	2.500000D-01	7.598572571?0-05	8.0806589406D-01	1.9185812021D-01	-4.1192679844D+00	-9.25532230610-02
3.684D+00						
	1.250000D-01	2.476646304?0-13	1.94323965500-03	9.9805676034D-01	-1.2606136012D+01	-2.71147363560+00
	1.666670D-01	6.556240791?0-14	2.8441930152D-03	9.9715580698D-01	-1.3183345104D+01	-2.5460409345D+00
	2.500000D-01	3.190288463?0-15	1.32022027880-02	9.8679779721D-01	-1.4496170047D+01	-1.8793536007D+00

ALL THETA VARYING R

N= 64 D= 11 D0= 5 D1= 5

THETA	R	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.250D-01						
	4.740000D+00	1.538169732?0-08	4.94672660670-02	9.5053271855D-01	-7.8129957387D+00	-1.30568209130+00
	2.710000D+00	1.139453138?0-03	6.20390245660-01	3.7847030120D-01	-2.9433035311D+00	-2.07335039180-01
	6.550000D+00	2.476646304?0-13	1.94323965500-03	9.9805676034D-01	-1.2606136012D+01	-2.71147363560+00
1.667D-01						
	4.740000D+00	5.738422915?0-09	6.55002429320-02	9.3449975133D-01	-8.2412074479D+00	-1.1837570693D+00
	2.710000D+00	5.646913053?0-04	6.7553205708D-01	3.2390325161D-01	-3.2481888991D+00	-1.70354036890-01
	6.550000D+00	6.556240791?0-14	2.8441930152D-03	9.9715580698D-01	-1.3183345104D+01	-2.5460409345D+00
2.500D-01						
	4.740000D+00	4.692590457?0-10	1.55141380790-01	8.4485861874D-01	-9.3285873469D+00	-8.09272347560-01
	2.710000D+00	7.598572571?0-05	8.0806589406D-01	1.9185812021D-01	-4.1192679844D+00	-9.25532230610-02
	6.550000D+00	3.190288463?0-15	1.32022027880-02	9.8679779721D-01	-1.4496170047D+01	-1.8793536007D+00

INPUT R*(36/64)

ALL R VARYING THETA

N= 64 D= 10 D0= 7 D1= 7

R*	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
2.666D+00						
	1.250000D-01	5.3448691492D-07	6.3874282303D-03	9.9361203728D-01	-6.2720629225D+00	-2.1946739666D+00
	1.666670D-01	4.1745134050D-07	8.5827372659D-03	9.9141684528D-01	-6.3793941400D+00	-2.0663741819D+00
	2.500000D-01	2.2043799017D-07	2.6033044354D-02	9.7316673521D-01	-6.6567135573D+00	-1.5713300515D+00
1.524D+00						
	1.250000D-01	8.0006780733D-03	3.4408051826D-01	6.4791880367D-01	-2.0968732041D+00	-4.6333991631D-01
	1.666670D-01	6.7668711953D-03	3.9106270194D-01	6.0217042687D-01	-2.1696120901D+00	-4.0775360342D-01
	2.500000D-01	3.1004633490D-03	5.4135234469D-01	4.5554719196D-01	-2.5085733981D+00	-2.6651997768D-01
3.684D+00						
	1.250000D-01	3.3028221058D-11	6.0118650041D-05	9.9993988132D-01	-1.0481114817D+01	-4.2209907800D+00
	1.666670D-01	2.1318844820D-11	8.4699169155D-05	9.9991530081D-01	-1.0671236332D+01	-4.0721208498D+00
	2.500000D-01	1.2118652251D-11	5.7694604822D-04	9.9942305394D-01	-1.0916545677D+01	-3.2388647970D+00

ALL THETA VARYING R

N= 64 D= 10 D0= 7 D1= 7

THETA	R	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.250D-01						
	4.740000D+00	5.3448691492D-07	6.3874282303D-03	9.9361203728D-01	-6.2720629225D+00	-2.1946739666D+00
	2.710000D+00	8.0006780733D-03	3.4408051826D-01	6.4791880367D-01	-2.0968732041D+00	-4.6333991631D-01
	6.550000D+00	3.3028221058D-11	6.0118650041D-05	9.9993988132D-01	-1.0481114817D+01	-4.2209907800D+00
1.667D-01						
	4.740000D+00	4.1745134050D-07	8.5827372659D-03	9.9141684528D-01	-6.3793941400D+00	-2.0663741819D+00
	2.710000D+00	6.7668711953D-03	3.9106270194D-01	6.0217042687D-01	-2.1696120901D+00	-4.0775360342D-01
	6.550000D+00	2.1318844820D-11	8.4699169155D-05	9.9991530081D-01	-1.0671236332D+01	-4.0721208498D+00
2.500D-01						
	4.740000D+00	2.2043799017D-07	2.6033044354D-02	9.7316673521D-01	-6.6567135573D+00	-1.5713300515D+00
	2.710000D+00	3.1004633490D-03	5.4135234469D-01	4.5554719196D-01	-2.5085733981D+00	-2.6651997768D-01
	6.550000D+00	1.2118652251D-11	5.7694604822D-04	9.9942305394D-01	-1.0916545677D+01	-3.2388647970D+00

59884

INPUT R*(36/64)

ALL R VARYING THETA

N= 64 D= 10 D0= 9 D1= 9

R°	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
2.666D+00						
	1.250000D-01	1.09014342860-05	5.92020425670-04	9.99397078140-01	-4.96251635880+00	-3.22766330920+00
	1.666670D-01	1.24391773360-05	7.79443547750-04	9.99208117270-01	-4.90520834070+00	-3.10821333370+00
	2.500000D-01	2.70476020840-05	2.98330372370-03	9.96989648670-01	-4.56787123140+00	-2.52530252980+00
1.524D+00						
	1.250000D-01	3.23242949130-02	1.26422633210-01	8.41253071870-01	-1.49047093960+00	-8.98175168140-01
	1.666670D-01	3.55390616850-02	1.48186718710-01	8.16274219610-01	-1.44929404280+00	-8.29190718420-01
	2.500000D-01	3.87673837710-02	2.45516681850-01	7.15715934380-01	-1.41153350650+00	-6.09918994010-01
3.684D+00						
	1.250000D-01	2.77705741120-09	1.33952725010-06	9.99998657700-01	-8.55641514180+00	-5.87304844710+00
	1.666670D-01	2.83841480800-09	1.73424148810-06	9.99998262920-01	-8.54692413610+00	-5.76089042840+00
	2.500000D-01	1.12112359840-08	1.53955263650-05	9.99984593260-01	-7.95034650590+00	-4.81260545820+00

ALL THETA VARYING R

N= 64 D= 10 D0= 9 D1= 9

THETA	R	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.250D-01						
	4.740000D+00	1.09014342860-05	5.92020425670-04	9.99397078140-01	-4.96251635880+00	-3.22766330920+00
	2.710000D+00	3.23242949130-02	1.26422633210-01	8.41253071870-01	-1.49047093960+00	-8.98175168140-01
	6.550000D+00	2.77705741120-09	1.33952725010-06	9.99998657700-01	-8.55641514180+00	-5.87304844710+00
1.667D-01						
	4.740000D+00	1.24391773360-05	7.79443547750-04	9.99208117270-01	-4.90520834070+00	-3.10821333370+00
	2.710000D+00	3.55390616850-02	1.48186718710-01	8.16274219610-01	-1.44929404280+00	-8.29190718420-01
	6.550000D+00	2.83841480800-09	1.73424148810-06	9.99998262920-01	-8.54692413610+00	-5.76089042840+00
2.500D-01						
	4.740000D+00	2.70476020840-05	2.98330372370-03	9.96989648670-01	-4.56787123140+00	-2.52530252980+00
	2.710000D+00	3.87673837710-02	2.45516681850-01	7.15715934380-01	-1.41153350650+00	-6.09918994010-01
	6.550000D+00	1.12112359840-08	1.53955263650-05	9.99984593260-01	-7.95034650590+00	-4.81260545820+00

INPUT R*(54/124)

ALL R VARYING THETA

N=124 D= 23 U0= 10 U1= 10

R*	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
2.0640+00						
	1.2500000-01	8.21207045520-13	1.23517084380-01	8.76482915620-01	-1.20855473330+01	-9.08272968390-01
	1.6666700-01	1.62148014810-13	1.66908427280-01	8.33091572720-01	-1.27900883640+01	-7.77521735050-01
	2.5000000-01	1.53781249600-15	3.62350658280-01	6.37649341720-01	-1.48130966140+01	-4.40870945370-01
1.1400+00						
	1.2500000-01	1.11273299750-04	9.23255169640-01	7.66335570610-02	-3.95360903300+00	-3.46782518990-02
	1.6666700-01	3.97168204360-05	9.47148632900-01	5.28116502780-02	-4.40102552660+00	-2.35818632520-02
	2.5000000-01	1.26229522940-06	9.83225677330-01	1.67730603710-02	-5.89883905910+00	-7.34678819860-03
2.8520+00						
	1.2500000-01	3.87634473300-21	2.40926065230-03	9.97590739350-01	-2.04115776070+01	-2.61811621220+00
	1.6666700-01	4.14340297520-22	4.11436430800-03	9.95885635690-01	-2.13826428260+01	-2.38569725620+00
	2.5000000-01	1.36239702110-24	2.48099349020-02	9.75190065100-01	-2.38656963150+01	-1.60537437530+00

ALL THETA VARYING R

N=124 D= 23 U0= 10 U1= 10

THETA	R	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.2500-01						
	4.7400000+00	8.21207045520-13	1.23517084380-01	8.76482915620-01	-1.20855473330+01	-9.08272968390-01
	2.7100000+00	1.11273299750-04	9.23255169640-01	7.66335570610-02	-3.95360903300+00	-3.46782518990-02
	6.5500000+00	3.87634473300-21	2.40926065230-03	9.97590739350-01	-2.04115776070+01	-2.61811621220+00
1.6670-01						
	4.7400000+00	1.62148014810-13	1.66908427280-01	8.33091572720-01	-1.27900883640+01	-7.77521735050-01
	2.7100000+00	3.97168204360-05	9.47148632900-01	5.28116502780-02	-4.40102552660+00	-2.35818632520-02
	6.5500000+00	4.14340297520-22	4.11436430800-03	9.95885635690-01	-2.13826428260+01	-2.38569725620+00
2.5000-01						
	4.7400000+00	1.53781249600-15	3.62350658280-01	6.37649341720-01	-1.48130966140+01	-4.40870945370-01
	2.7100000+00	1.26229522940-06	9.83225677330-01	1.67730603710-02	-5.89883905910+00	-7.34678819860-03
	6.5500000+00	1.36239702110-24	2.48099349020-02	9.75190065100-01	-2.38656963150+01	-1.60537437530+00

INPUT R*(54/124)

ALL R VARYING THETA

N=124 D= 23 D0= 14 D1= 14

R*	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
2.664D+00						
	1.250000D-01	1.88893156800-10	1.27354910170-02	9.87264508790-01	-9.72378377540+00	-1.89498430610+00
	1.666670D-01	1.35149998820-10	1.86182673400-02	9.81381732520-01	-9.86916395380+00	-1.73006073790+00
	2.500000D-01	3.21911468000-11	6.31683851910-02	9.36831614780-01	-1.04922635510+01	-1.19950022510+00
1.180D+00						
	1.250000D-01	1.68117796380-03	6.98726233790-01	2.99592588240-01	-2.77438631120+00	-1.55692950780-01
	1.666670D-01	1.49036470740-03	7.51523140810-01	2.46986494480-01	-2.82670744230+00	-1.24057642130-01
	2.500000D-01	4.36561383770-04	8.71459344550-01	1.28104094070-01	-3.35995468260+00	-5.97528688530-02
2.852D+00						
	1.250000D-01	8.13202814560-18	2.74042824380-05	9.99972595720-01	-1.70898011270+01	-4.56218156510+00
	1.666670D-01	4.09038568890-18	4.75247734790-05	9.99952475230-01	-1.73882357400+01	-4.32307994450+00
	2.500000D-01	8.30923954300-19	5.20599999790-04	9.99479400000-01	-1.80804387210+01	-3.28349583640+00

ALL THETA VARYING R

N=124 D= 23 D0= 14 D1= 14

THETA	R*	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.250D-01						
	4.740000D+00	1.88893156800-10	1.27354910170-02	9.87264508790-01	-9.72378377540+00	-1.89498430610+00
	2.710000D+00	1.68117796380-03	6.98726233790-01	2.99592588240-01	-2.77438631120+00	-1.55692950780-01
	6.550000D+00	8.13202814560-18	2.74042824380-05	9.99972595720-01	-1.70898011270+01	-4.56218156510+00
1.667D-01						
	4.740000D+00	1.35149998820-10	1.86182673400-02	9.81381732520-01	-9.86916395380+00	-1.73006073790+00
	2.710000D+00	1.49036470740-03	7.51523140810-01	2.46986494480-01	-2.82670744230+00	-1.24057642130-01
	6.550000D+00	4.09038568890-18	4.75247734790-05	9.99952475230-01	-1.73882357400+01	-4.32307994450+00
2.500D-01						
	4.740000D+00	3.21911468000-11	6.31683851910-02	9.36831614780-01	-1.04922635510+01	-1.19950022510+00
	2.710000D+00	4.36561383770-04	8.71459344550-01	1.28104094070-01	-3.35995468260+00	-5.97528688530-02
	6.550000D+00	8.30923954300-19	5.20599999790-04	9.99479400000-01	-1.80804387210+01	-3.28349583640+00

INPUT R*(54/124)

ALL R VARYING THETA

N=124 D=.23 D0=.18 D1=.18

R*	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
2.0640D+00						
	1.2500D-01	1.8718812788D-08	6.9005186133D-04	9.9930992942D-01	-7.7277216992D+00	-3.1611182683D+00
	1.6666D-01	2.2435358117D-08	1.0264580633D-03	9.9897351950D-01	-7.6490669938D+00	-2.9886587893D+00
	2.5000D-01	5.7937154151D-08	4.8678663360D-03	9.9513207573D-01	-7.2370428410D+00	-2.3126613553D+00
1.1800D+00						
	1.2500D-01	1.1710312237D-02	3.7378459703D-01	6.1450509073D-01	-1.9314315250D+00	-4.2737859906D-01
	1.6666D-01	1.3787095657D-02	4.2277222190D-01	5.6344068244D-01	-1.8605272112D+00	-3.7389355559D-01
	2.5000D-01	1.6795292181D-02	5.7860538886D-01	4.0459931896D-01	-1.7748124365D+00	-2.3761752585D-01
2.8520D+00						
	1.2500D-01	7.8854039952D-15	1.5577396660D-07	9.9999984423D-01	-1.4103176051D+01	-6.8075051211D+00
	1.6666D-01	7.7153483051D-15	2.5231577048D-07	9.9999974768D-01	-1.4112644463D+01	-6.5980556039D+00
	2.5000D-01	3.9259831081D-14	4.1497319071D-06	9.9999585027D-01	-1.3406051573D+01	-5.3819799599D+00

ALL THETA VARYING R

N=124 D=.23 D0=.18 D1=.18

THETA	R	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.2500D-01						
	4.7400000D+00	1.8718812788D-08	6.9005186133D-04	9.9930992942D-01	-7.7277216992D+00	-3.1611182683D+00
	2.7100000D+00	1.1710312237D-02	3.7378459703D-01	6.1450509073D-01	-1.9314315250D+00	-4.2737859906D-01
	6.5500000D+00	7.8854039952D-15	1.5577396660D-07	9.9999984423D-01	-1.4103176051D+01	-6.8075051211D+00
1.6667D-01						
	4.7400000D+00	2.2435358117D-08	1.0264580633D-03	9.9897351950D-01	-7.6490669938D+00	-2.9886587893D+00
	2.7100000D+00	1.3787095657D-02	4.2277222190D-01	5.6344068244D-01	-1.8605272112D+00	-3.7389355559D-01
	6.5500000D+00	7.7153483051D-15	2.5231577048D-07	9.9999974768D-01	-1.4112644463D+01	-6.5980556039D+00
2.5000D-01						
	4.7400000D+00	5.7937154151D-08	4.8678663360D-03	9.9513207573D-01	-7.2370428410D+00	-2.3126613553D+00
	2.7100000D+00	1.6795292181D-02	5.7860538886D-01	4.0459931896D-01	-1.7748124365D+00	-2.3761752585D-01
	6.5500000D+00	3.9259831081D-14	4.1497319071D-06	9.9999585027D-01	-1.3406051573D+01	-5.3819799599D+00

5988

INPUT R*(54/124)

ALL R VARYING THETA

N=124 D= 23 D0= 23 D1= 23

R	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
2.1640+00						
	1.2500000-01	3.3125666609B0-06	5.3443877953D-06	9.9999134304D-01	-5.4798351179D+00	-5.2721020365D+00
	1.6666700-01	4.6134330368D-06	7.6566000904D-06	9.9998772991D-01	-5.3359757787D+00	-5.1159606325D+00
	2.5000000-01	2.8936751239D-05	4.6666144200D-05	9.9992439710D-01	-4.5385502292D+00	-4.3309980813D+00
1.1800+00						
	1.2500000-01	7.9641771561D-02	2.9949993226D-02	8.9040823521D-01	-1.0988590880D+00	-1.5236032715D+00
	1.6666700-01	9.4894406713D-02	3.4995975001D-02	8.7010961829D-01	-1.0227593851D+00	-1.4559819024D+00
	2.5000000-01	1.6226648540D-01	5.4625233179D-02	7.8310828142D-01	-7.8977117030D-01	-1.2626066961D+00
2.8520+00						
	1.2500000-01	2.4471322755D-11	8.8809828022D-11	9.9999999890D-01	-1.0611342555D+01	-1.0051538971D+01
	1.6666700-01	3.1944871930D-11	1.2255680751D-10	9.9999999985D-01	-1.0495598849D+01	-9.9116625605D+00
	2.5000000-01	7.6207102424D-10	2.6797714949D-09	9.9999999662D-01	-9.1536189507D+00	-8.5719022368D+00

ALL THETA VARYING R

N=124 D= 23 D0= 23 D1= 23

THETA	R	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.2500-01						
	4.7400000+00	3.3125666609B0-06	5.3443877953D-06	9.9999134304D-01	-5.4798351179D+00	-5.2721020365D+00
	2.7100000+00	7.9641771561D-02	2.9949993226D-02	8.9040823521D-01	-1.0988590880D+00	-1.5236032715D+00
	6.5500000+00	2.4471322755D-11	8.8809828022D-11	9.9999999890D-01	-1.0611342555D+01	-1.0051538971D+01
1.6670-01						
	4.7400000+00	4.6134330368D-06	7.6566000904D-06	9.9998772991D-01	-5.3359757787D+00	-5.1159606325D+00
	2.7100000+00	9.4894406713D-02	3.4995975001D-02	8.7010961829D-01	-1.0227593851D+00	-1.4559819024D+00
	6.5500000+00	3.1944871930D-11	1.2255680751D-10	9.9999999985D-01	-1.0495598849D+01	-9.9116625605D+00
2.5000-01						
	4.7400000+00	2.8936751239D-05	4.6666144200D-05	9.9992439710D-01	-4.5385502292D+00	-4.3309980813D+00
	2.7100000+00	1.6226648540D-01	5.4625233179D-02	7.8310828142D-01	-7.8977117030D-01	-1.2626066961D+00
	6.5500000+00	7.9207102424D-10	2.6797714949D-09	9.9999999662D-01	-9.1536189507D+00	-8.5719022368D+00

LAST CASE COMPLETED

PART II

INPUT R*(12/63)

ALL R VARYING THETA

N= 7 D= 3 D0= 3 DI= 3

R*	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
9.0290-01						
	0.	1.2433087586D-01	3.5527136788D-15	8.7566912414D-01	-9.0542100692D-01	-1.4449439792D+01
	1.2500000D-01	8.3984199683D-02	8.3816153758D-02	8.3219964716D-01	-1.0758024151D+00	-1.0766722724D+00
	1.6666700-01	7.7332997700-02	9.9403085601D-02	8.2326391666D-01	-1.1116351545D+00	-1.0026001343D+00
	2.5000000-01	6.8389372372D-02	1.2514680463D-01	8.0646382300D-01	-1.1650113819D+00	-9.0258023474D-01
5.1620-01						
	0.	2.9685638725D-01	3.5527136788D-15	7.0314361275D-01	-5.2745360223D-01	-1.4449439792D+01
	1.2500000-01	2.3933311424D-01	1.2058600279D-01	6.4008088296D-01	-6.2099720804D-01	-9.1870310067D-01
	1.6666700-01	2.2781716644D-01	1.4638793176D-01	6.2579490179D-01	-6.4241355413D-01	-8.3449472508D-01
	2.5000000-01	2.0915693317D-01	1.8935522140D-01	6.0148784544D-01	-6.7952773478D-01	-7.2272271487D-01
1.2480+00						
	0.	5.6495195523D-02	2.2204604693D-15	9.4350480448D-01	-1.2479884840D+00	-1.4653559775D+01
	1.2500000-01	3.2218804559D-02	4.9782605157D-02	9.1799859028D-01	-1.4918905776D+00	-1.3029223802D+00
	1.6666700-01	2.8746313622D-02	5.8088002525D-02	9.1316568385D-01	-1.5414178406D+00	-1.2359135574D+00
	2.5000000-01	2.4754312006D-02	7.2530704462D-02	9.0271498353D-01	-1.6063491395D+00	-1.1394781044D+00

ALL THETA VARYING R*

N= 7 D= 3 D0= 3 DI= 3

THETA	R	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
0.						
	9.7400000+00	1.2433087586D-01	3.5527136788D-15	8.7566912414D-01	-9.0542100692D-01	-1.4449439792D+01
	2.7100000+00	2.9685638725D-01	3.5527136788D-15	7.0314361275D-01	-5.2745360223D-01	-1.4449439792D+01
	6.5500000+00	5.6495195523D-02	2.2204604693D-15	9.4350480448D-01	-1.2479884840D+00	-1.4653559775D+01
1.2500-01						
	9.7400000+00	8.3984199683D-02	8.3816153758D-02	8.3219964716D-01	-1.0758024151D+00	-1.0766722724D+00
	2.7100000+00	2.3933311424D-01	1.2058600279D-01	6.4008088296D-01	-6.2099720804D-01	-9.1870310067D-01
	6.5500000+00	3.2218804559D-02	4.9782605157D-02	9.1799859028D-01	-1.4918905776D+00	-1.3029223802D+00
1.6670-01						
	9.7400000+00	7.7332997700-02	9.9403085601D-02	8.2326391666D-01	-1.1116351545D+00	-1.0026001343D+00
	2.7100000+00	2.2781716644D-01	1.4638793176D-01	6.2579490179D-01	-6.4241355413D-01	-8.3449472508D-01
	6.5500000+00	2.8746313622D-02	5.8088002525D-02	9.1316568385D-01	-1.5414178406D+00	-1.2359135574D+00
2.5000-01						
	9.7400000+00	6.8389372372D-02	1.2514680463D-01	8.0646382300D-01	-1.1650113819D+00	-9.0258023474D-01
	2.7100000+00	2.0915693317D-01	1.8935522140D-01	6.0148784544D-01	-6.7952773478D-01	-7.2272271487D-01
	6.5500000+00	2.4754312006D-02	7.2530704462D-02	9.0271498353D-01	-1.6063491395D+00	-1.1394781044D+00

INPUT R*(12/35)

ALL R VARYING THETA

N= 7 D= 3 D0= 3 D1= 3

R*	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.625D+00						
	0.	2.3751976607D-02	1.1102230246D-16	9.7624802339D-01	-1.6243002431D+00	-1.5954589770D+01
	1.2500000D-01	1.1138603039D-02	2.5457317727D-02	9.6340465188D-01	-1.9531916015D+00	-1.5941873571D+00
	1.6666700D-01	9.5685271336D-03	2.9235668841D-02	9.6119580403D-01	-2.0191549073D+00	-1.5340869661D+00
	2.5000000D-01	8.0316127340D-03	3.6577656938D-02	9.5539073033D-01	-2.0951972404D+00	-1.4367841177D+00
9.291D-01						
	0.	1.1710349462D-01	2.2204460493D-15	8.8209650538D-01	-9.3143014445D-01	-1.4653559775D+01
	1.2500000D-01	7.8115811362D-02	8.0921438986D-02	8.4096274965D-01	-1.1072610520D+00	-1.0919364030D+00
	1.6666700D-01	7.1762765174D-02	9.5841535939D-02	8.3239569888D-01	-1.1441008353D+00	-1.0184462350D+00
	2.5000000D-01	6.3324541906D-02	1.2053543201D-01	8.1614002609D-01	-1.1984279432D+00	-9.1888527124D-01
2.246D+00						
	0.	5.7551239474D-03	3.8857805862D-16	9.9424487605D-01	-2.2399453186D+00	-1.5410521726D+01
	1.2500000D-01	1.9224990371D-03	7.5305845292D-03	9.9054691643D-01	-2.7161338690D+00	-2.1231713122D+00
	1.6666700D-01	1.5391064773D-03	8.4369200319D-03	9.9002397349D-01	-2.8127313341D+00	-2.0738160673D+00
	2.5000000D-01	1.2447195517D-03	1.0786457186D-02	9.8796882326D-01	-2.9049284886D+00	-1.9671211760D+00

ALL THETA VARYING R

N= 7 D= 3 D0= 3 D1= 3

THETA	R*	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
0.						
	4.7400000D+00	2.3751976607D-02	1.1102230246D-16	9.7624802339D-01	-1.6243002431D+00	-1.5954589770D+01
	2.7100000D+00	1.1710349462D-01	2.2204460493D-15	8.8289650538D-01	-9.3143014445D-01	-1.4653559775D+01
	6.5500000D+00	5.7551239474D-03	3.8857805862D-16	9.9424487605D-01	-2.2399453186D+00	-1.5410521726D+01
1.250D-01						
	4.7400000D+00	1.1138603039D-02	2.5457317727D-02	9.6340465188D-01	-1.9531916015D+00	-1.5941873571D+00
	2.7100000D+00	7.8115811362D-02	8.0921438986D-02	8.4096274965D-01	-1.1072610520D+00	-1.0919364030D+00
	6.5500000D+00	1.9224990371D-03	7.5305845292D-03	9.9054691643D-01	-2.7161338690D+00	-2.1231713122D+00
1.667D-01						
	4.7400000D+00	9.5685271336D-03	2.9235668841D-02	9.6119580403D-01	-2.0191549073D+00	-1.5340869661D+00
	2.7100000D+00	7.1762765174D-02	9.5841535939D-02	8.3239569888D-01	-1.1441008353D+00	-1.0184462350D+00
	6.5500000D+00	1.5391064773D-03	8.4369200319D-03	9.9002397349D-01	-2.8127313341D+00	-2.0738160673D+00
2.500D-01						
	4.7400000D+00	8.0316127340D-03	3.6577656938D-02	9.5539073033D-01	-2.0951972404D+00	-1.4367841177D+00
	2.7100000D+00	6.3324541906D-02	1.2053543201D-01	8.1614002609D-01	-1.1984279432D+00	-9.1888527124D-01
	6.5500000D+00	1.2447195517D-03	1.0786457186D-02	9.8796882326D-01	-2.9049284886D+00	-1.9671211760D+00

INPUT R*(1/6)

ALL R VARYING THETA

N= 8 D= 4 D0= 3 D1= 3

R*	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
7.900D-01						
	0.	4.2574671330D-02	1.5744801589D-01	7.9997731278D-01	-1.3708486960D+00	-8.0286280783D-01
	1.2500000-01	2.4844714246D-02	2.3465342872D-01	7.4050255703D-01	-1.6047782304D+00	-6.2957309556D-01
	1.6666700-01	2.1698302900D-02	2.5039939247D-01	7.2770230463D-01	-1.6595895413D+00	-6.0136672917D-01
	2.5000000-01	1.7401244640D-02	2.7913283196D-01	7.0346592340D-01	-1.7594196873D+00	-5.5418907824D-01
4.517D-01						
	0.	1.4297203134D-01	2.6568705872D-01	5.9134090993D-01	-8.4474891234D-01	-5.7562959900D-01
	1.25000D-01	1.0523494955D-01	3.7350725955D-01	5.2125779090D-01	-9.7784000282D-01	-4.2770095275D-01
	1.6666700-01	9.7690554718D-02	3.9717912236D-01	5.0513032292D-01	-1.0101474243D+00	-4.0101358818D-01
	2.5000000-01	8.4830552724D-02	4.3771967944D-01	4.7744976784D-01	-1.0714477033D+00	-3.5880392751D-01
1.192D+00						
	0.	1.4559312064D-02	8.8288166413D-02	8.9715252152D-01	-1.8368591452D+00	-1.0540975026D+00
	1.25000D-01	6.8255426487D-03	1.3681298720D-01	8.5636147016D-01	-2.1658628154D+00	-8.6387267455D-01
	1.6666700-01	5.7298570525D-03	1.4626846350D-01	8.4800167944D-01	-2.2418562126D+00	-8.3484930069D-01
	2.5000000-01	4.2171718538D-03	1.6570822569D-01	8.3007460245D-01	-2.3749787007D+00	-7.8065593283D-01

ALL THETA VARYING R

N= 8 D= 4 D0= 3 D1= 3

THETA	R*	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
0.						
	4.7400000+00	4.2574671330D-02	1.5744801589D-01	7.9997731278D-01	-1.3708486960D+00	-8.0286280783D-01
	2.7100000+00	1.4297203134D-01	2.6568705872D-01	5.9134090993D-01	-8.4474891234D-01	-5.7562959900D-01
	6.5500000+00	1.4559312064D-02	8.8288166413D-02	8.9715252152D-01	-1.8368591452D+00	-1.0540975026D+00
1.250D-01						
	4.7400000+00	2.4844014246D-02	2.3465342872D-01	7.4050255703D-01	-1.6047782304D+00	-6.2957309556D-01
	2.7100000+00	1.0523494955D-01	3.7350725955D-01	5.2125779090D-01	-9.7784000282D-01	-4.2770095275D-01
	6.5500000+00	6.8255426487D-03	1.3681298720D-01	8.5636147016D-01	-2.1658628154D+00	-8.6387267455D-01
1.667D-01						
	4.7400000+00	2.1898302900D-02	2.5039939247D-01	7.2770230463D-01	-1.6595895413D+00	-6.0136672917D-01
	2.7100000+00	9.7690554718D-02	3.9717912236D-01	5.0513032292D-01	-1.0101474243D+00	-4.0101358818D-01
	6.5500000+00	5.7298570525D-03	1.4626846350D-01	8.4800167944D-01	-2.2418562126D+00	-8.3484930069D-01
2.500D-01						
	4.7400000+00	1.7401244640D-02	2.7913283196D-01	7.0346592340D-01	-1.7594196873D+00	-5.5418907824D-01
	2.7100000+00	8.4830552724D-02	4.3771967944D-01	4.7744976784D-01	-1.0714477033D+00	-3.5880392751D-01
	6.5500000+00	4.2171718538D-03	1.6570822569D-01	8.3007460245D-01	-2.3749787007D+00	-7.8065593283D-01

INPUT R*(3/10)

ALL R VARYING THETA

N= 8 D= 4 D0= 3 D1= 3

R*	THETA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
1.4220+00						
	..	4.53732168950-03	4.44286807010-02	9.51033997610-01	-2.34320042880+00	-1.35233658280+00
	1.2500000-01	1.65465327300-03	7.12386725020-02	9.27106674230-01	-2.78129299730+00	-1.14728418200+00
	1.6666700-01	1.31241767850-03	7.62223872950-02	9.22465195030-01	-2.88192792820+00	-1.11791745320+00
	2.5000000-01	8.89250608100-04	8.83549568570-02	9.10755792530-01	-3.05097582920+00	-1.05376908080+00
8.1300-01						
	..	3.92225129840-02	1.51039796520-01	8.09739690490-01	-1.40648673030+00	-8.20908608120-01
	1.2500000-01	2.25165479650-02	2.25847576360-01	7.51635875680-01	-1.64749819080+00	-6.46184565630-01
	1.6666700-01	1.97750400250-02	2.41046026020-01	7.39178933950-01	-1.70388262880+00	-6.17900024070-01
	2.5000000-01	1.56215956210-02	2.68987021120-01	7.15391383260-01	-1.80627460850+00	-5.70268674610-01
1.9650+00						
	..	6.83021824900-04	1.36363142330-02	9.85680663940-01	-3.16556541890+00	-1.86530299950+00
	1.2500000-01	1.61421502720-04	2.27863617640-02	9.77052216730-01	-3.79203861400+00	-1.64232501190+00
	1.6666700-01	1.15754386290-04	2.43350761180-02	9.75549169500-01	-3.93646254320+00	-1.61376729100+00
	2.5000000-01	6.86136110170-05	2.95629171490-02	9.70368469240-01	-4.16358972390+00	-1.52925271370+00

ALL THETA VARYING R

N= 8 D= 4 D0= 3 D1= 3

THETA	R*	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)
0.						
	4.74000000+00	4.53732168950-03	4.44286807010-02	9.51033997610-01	-2.34320042880+00	-1.35233658280+00
	2.71000000+00	3.92225129840-02	1.51039796520-01	8.09739690490-01	-1.40648673030+00	-8.20908608120-01
	6.55000000+00	6.83021824900-04	1.36363142330-02	9.85680663940-01	-3.16556541890+00	-1.86530299950+00
1.2500-01						
	4.74000000+00	1.65465327300-03	7.12386725020-02	9.27106674230-01	-2.78129299730+00	-1.14728418200+00
	2.71000000+00	2.25165479650-02	2.25847576360-01	7.51635875680-01	-1.64749819080+00	-6.46184565630-01
	6.55000000+00	1.61421502720-04	2.27863617640-02	9.77052216730-01	-3.79203861400+00	-1.64232501190+00
1.6670-01						
	4.74000000+00	1.31241767850-03	7.62223872950-02	9.22465195030-01	-2.88192792820+00	-1.11791745320+00
	2.71000000+00	1.97750400250-02	2.41046026020-01	7.39178933950-01	-1.70388262880+00	-6.17900024070-01
	6.55000000+00	1.15754386290-04	2.43350761180-02	9.75549169500-01	-3.93646254320+00	-1.61376729100+00
2.5000-01						
	4.74000000+00	8.89250608100-04	8.83549568570-02	9.10755792530-01	-3.05097582920+00	-1.05376908080+00
	2.71000000+00	1.56215956210-02	2.68987021120-01	7.15391383260-01	-1.80627460850+00	-5.70268674610-01
	6.55000000+00	6.86136110170-05	2.95629171490-02	9.70368469240-01	-4.16358972390+00	-1.52925271370+00

N=15, D=7, D0=1, D1=1,3,5,7

N	D	D0	D1	P	Q	PC	PE	1-PC-PE	LOG10(PC)	LOG10(1-PC-PE)
15	7	1	1	8.89000000D-04	8.84000000D-02	2.45871810-01	1.33761030-18	7.54128190-01	-1.78736700+01	-1.22554820-01
15	7	1	1	6.86000000D-05	2.96000000D-02	6.36504210-01	3.61575960-26	3.63495790-01	-2.54418000+01	-4.39500620-01
15	7	1	1	6.83000000D-04	1.36000000D-02	8.05904720-01	3.97935270-19	1.94095280-01	-1.84001880+01	-7.11985020-01
15	7	1	1	8.03000000D-03	3.66000000D-02	5.04166510-01	9.69591930-12	4.95833490-01	-1.10134110+01	-3.04664140-01
15	7	1	1	1.24000000D-03	1.08000000D-02	8.33854780-01	2.63609170-17	1.66145220-01	-1.65790390+01	-7.79512150-01
15	7	1	1	5.76000000D-03	0.	9.16998180-01	1.30003750-12	8.30018200-02	-1.18860440+01	-1.08091240+00
NEXT SET										
15	7	1	3	8.89000000D-04	8.84000000D-02	8.47106040-01	2.78770290-13	1.52893960-01	-1.25547540+01	-8.15609660-01
15	7	1	3	6.86000000D-05	2.96000000D-02	9.89944930-01	1.41484530-19	1.00550740-02	-1.88492910+01	-1.99761470+00
15	7	1	3	6.83000000D-04	1.36000000D-02	9.88799670-01	3.36999150-15	1.12003280-02	-1.44723710+01	-1.95076930+00
15	7	1	3	8.03000000D-03	3.66000000D-02	8.71577090-01	4.56132880-09	1.28422910-01	-8.34090860+00	-8.91357500-01
15	7	1	3	1.24000000D-03	1.08000000D-02	9.81048300-01	4.36450000-14	1.89516990-02	-1.33600650+01	-1.72235190+00
15	7	1	3	5.76000000D-03	0.	9.16998180-01	1.30003750-12	8.30018200-02	-1.18860440+01	-1.08091240+00
NEXT SET										
15	7	1	5	8.89000000D-04	8.84000000D-02	9.79214020-01	4.62800670-09	2.07859800-02	-8.33460600+00	-1.68222950+00
15	7	1	5	6.86000000D-05	2.96000000D-02	9.98918330-01	4.39728480-14	1.08166800-03	-1.33568150+01	-2.96590600+00
15	7	1	5	6.83000000D-04	1.36000000D-02	9.89802600-01	2.30443610-12	1.01974020-02	-1.16374350+01	-1.99151050+00
15	7	1	5	8.03000000D-03	3.66000000D-02	8.85957160-01	1.84559360-07	1.14042650-01	-6.73386390+00	-9.42932690-01
15	7	1	5	1.24000000D-03	1.08000000D-02	9.81560190-01	5.96965750-12	1.84398140-02	-1.12240510+01	-1.73424350+00
15	7	1	5	5.76000000D-03	0.	9.16998180-01	1.30003750-12	8.30018200-02	-1.18860440+01	-1.08091240+00
NEXT SET										
15	7	1	7	8.89000000D-04	8.84000000D-02	9.86605740-01	9.35642640-06	1.33849040-02	-5.02889000+00	-1.87338470+00
15	7	1	7	6.86000000D-05	2.96000000D-02	9.98971390-01	1.64522340-09	1.02860820-03	-8.78377510+00	-2.98775000+00
15	7	1	7	6.83000000D-04	1.36000000D-02	9.89803040-01	2.02483130-10	1.01961640-02	-9.69361120+00	-1.99156320+00
15	7	1	7	8.03000000D-03	3.66000000D-02	8.86090070-01	1.21487710-06	1.13908710-01	-5.91546770+00	-9.43443050-01
15	7	1	7	1.24000000D-03	1.08000000D-02	9.81560580-01	1.14628160-10	1.84394160-02	-9.94070870+00	-1.73425280+00
15	7	1	7	5.76000000D-03	0.	9.16998180-01	1.30003750-12	8.30018200-02	-1.18860440+01	-1.08091240+00
NEXT SET										

N=15, D=11, D0=1, D1=3,7,11

N	D	D0	D1	P	Q	PC	PE	1-PC-PE	LOG10(PE)	LOG10(1-PC-PE)
15	11	1	3	1.7400000D-02	2.7900000D-01	1.2027556D-01	2.1412284D-13	8.7972444D-01	-1.2669337D+01	-5.5653342D-02
15	11	1	3	4.2600000D-02	1.5700000D-01	2.8296296D-01	3.7839660D-10	7.1703704D-01	-9.4220528D+00	-1.4445841D-01
15	11	1	3	4.2200000D-03	1.6600000D-01	4.9936530D-01	4.1924356D-19	5.0063470D-01	-1.8377534D+01	-3.0047905D-01
15	11	1	3	1.4600000D-02	8.8300000D-02	6.8532348D-01	1.1888204D-14	3.1467652D-01	-1.3924884D+01	-5.0213566D-01
15	11	1	3	6.8400000D-02	1.2500000D-01	2.3267626D-01	1.8765465D-08	7.6732372D-01	-7.7266407D+00	-1.1502138D-01
15	11	1	3	2.4800000D-02	7.7500000D-02	6.0978219D-01	1.1198781D-12	3.9021781D-01	-1.1950829D+01	-4.0869291D-01
15	11	1	3	5.6500000D-02	0.	4.1795408D-01	2.0669752D-11	5.8204592D-01	-1.0684665D+01	-2.3504275D-01
NEXT SET										
15	11	1	7	1.7400000D-02	2.7900000D-01	6.8947896D-01	1.2589417D-07	3.1052091D-01	-6.8999944D+00	-5.0790915D-01
15	11	1	7	4.2600000D-02	1.5700000D-01	5.1734215D-01	7.4941336D-07	4.8265710D-01	-6.1252786D+00	-3.1636130D-01
15	11	1	7	4.2200000D-03	1.6600000D-01	9.3233065D-01	8.6179806D-12	6.7669349D-02	-1.1064594D+01	-1.1696080D+00
15	11	1	7	1.4600000D-02	8.8300000D-02	8.0190085D-01	1.5401645D-10	1.9809915D-01	-9.8124329D+00	-7.0311739D-01
15	11	1	7	6.8400000D-02	1.2500000D-01	3.4484907D-01	2.9416646D-06	6.5514798D-01	-5.5314068D+00	-1.8366059D-01
15	11	1	7	2.4800000D-02	7.7500000D-02	6.8607860D-01	1.1913440D-09	3.1392140D-01	-8.9239628D+00	-5.0317908D-01
15	11	1	7	5.6500000D-02	0.	4.1795408D-01	2.0669752D-11	5.8204592D-01	-1.0684665D+01	-2.3504275D-01
NEXT SET										
15	11	1	11	1.7400000D-02	2.7900000D-01	7.6820791D-01	2.6366909D-04	2.3152842D-01	-3.5789408D+00	-6.3539569D-01
15	11	1	11	4.2600000D-02	1.5700000D-01	5.2047591D-01	1.1163436D-05	4.7951239D-01	-4.9522021D+00	-3.1919968D-01
15	11	1	11	4.2200000D-03	1.6600000D-01	9.3853429D-01	5.4813031D-07	6.1465161D-02	-6.2611162D+00	-1.2113710D+00
15	11	1	11	1.4600000D-02	8.8300000D-02	8.0202585D-01	1.0104539D-08	1.9797414D-01	-7.9954835D+00	-7.0339154D-01
15	11	1	11	6.8400000D-02	1.2500000D-01	3.4549527D-01	8.7741605D-06	6.5449595D-01	-5.0567944D+00	-1.8409303D-01
15	11	1	11	2.4800000D-02	7.7500000D-02	6.8612839D-01	1.1211897D-08	3.1387160D-01	-7.9503209D+00	-5.0324798D-01
15	11	1	11	5.6500000D-02	0.	4.1795408D-01	2.0669752D-11	5.8204592D-01	-1.0684665D+01	-2.3504275D-01
NEXT SET										

LAST CASE COMPLETED

PART III

DEC. 22,1970 CASE 1, PE EVALUATED BY EXPANSION, R* = (3/10)R, D1 = 3

ALL GAMMA VARYING R

N= 8 U= 4 D0= 3 D1= 3

GAMMA	R	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)	THETA
0.000000							
	0.000000	5.58593750000D-01	4.06250000000D-01	3.51562500000D-02	-2.52903927850-01	-3.91206626010-01	0.000000
	.845000	2.29508936780-01	3.73475141890-01	3.97115921330-01	-6.39200398930-01	-4.27854599350-01	0.000000
	1.280000	1.50123798140-01	3.18103205600-01	5.31774996270-01	-8.23550456460-01	-4.97431954390-01	0.000000
	2.045000	3.78212487450-02	1.60685961460-01	8.01492788980-01	-1.42226413610+00	-7.94022064290-01	0.000000
	3.360000	1.77315741320-02	1.04518191540-01	8.77750234330-01	-1.75125270790+00	-9.80808113410-01	0.000000
	4.805000	4.04962295210-03	4.28065315630-02	9.53143845490-01	-2.39258541070+00	-1.36848995980+00	0.000000
	5.445000	2.08697198730-03	2.82602402290-02	9.69652787780-01	-2.68048338030+00	-1.54882415070+00	0.000000
	6.845000	4.92340296780-04	1.12265590890-02	9.88281101610-01	-3.30773461690+00	-1.94975337210+00	0.000000
	8.405000	9.97590858410-05	3.97295473550-03	9.95927286180-01	-4.00104753920+00	-2.40088638290+00	0.000000
.150000							
	0.000000	5.58593750000D-01	4.06250000000D-01	3.51562500000D-02	-2.52903927850-01	-3.91206626010-01	.150000
	.845000	1.69441535620-01	4.98882810100-01	3.31675654280-01	-7.70980121290-01	-3.02001460190-01	.150000
	1.280000	9.62245406530-02	4.49796014580-01	4.53979438760-01	-1.01671412630+00	-3.46984396950-01	.150000
	2.045000	1.53506868260-02	2.50658893240-01	7.34390323940-01	-1.81387230150+00	-6.81610306880-01	.150000
	3.360000	5.63379048640-03	1.67920355450-01	8.26445854660-01	-2.24919930790+00	-7.74896655120-01	.150000
	4.805000	8.07543015850-04	7.18713434750-02	9.27321143510-01	-3.09283433460+00	-1.14344423690+00	.150000
	5.445000	3.38951220990-04	4.81714033390-02	9.51489585440-01	-3.46986279730+00	-1.31721016060+00	.150000
	6.845000	5.14333154860-05	1.96625772780-02	9.80325989410-01	-4.28875547940+00	-1.70726395190+00	.150000
	8.405000	8.43909869650-06	7.06086557770-03	9.92932695320-01	-5.19117491810+00	-2.15114205640+00	.150000
.200000							
	0.000000	5.58593750000D-01	4.06250000000D-01	3.51562500000D-02	-2.52903927850-01	-3.91206626010-01	.200000
	.845000	1.50652515970-01	5.32964050610-01	3.16383443200-01	-8.22023610950-01	-2.73302083770-01	.200000
	1.280000	8.10758631380-02	4.82413397090-01	4.36510771530-01	-1.09110858920+00	-3.16580639920-01	.200000
	2.045000	1.09549260280-02	2.69069150360-01	7.19975923620-01	-1.96039055060+00	-5.70136092640-01	.200000
	3.360000	3.68545998930-03	1.80866274690-01	8.15451793070-01	-2.43350829930+00	-7.42650877390-01	.200000
	4.805000	4.49101868030-04	7.78508731750-02	9.21721025160-01	-3.34862343940+00	-1.10884809720+00	.200000
	5.445000	1.74930906950-04	5.23111547280-02	9.47513914300-01	-3.75713330330+00	-1.28140569320+00	.200000
	6.845000	2.27048904790-05	2.14363873700-02	9.78540907740-01	-4.64388059870+00	-1.66884840360+00	.200000
	8.405000	2.39494394700-06	7.75915286260-03	9.92236452170-01	-5.62070464670+00	-2.11018569090+00	.200000
.250000							
	0.000000	5.58593750000D-01	4.06250000000D-01	3.51562500000D-02	-2.52903927850-01	-3.91206626010-01	.250000
	.845000	1.32858293040-01	5.64621455931D-01	3.02926492640-01	-8.76609174220-01	-2.48555715100-01	.250000
	1.280000	6.75439207170-02	5.11597979110-01	4.20858100180-01	-1.17041373350+00	-2.91071179720-01	.250000
	2.045000	7.67797626740-03	2.87014727880-01	7.05307297850-01	-2.11475334770+00	-5.42095817300-01	.250000
	3.360000	2.36129786610-03	1.94433355560-01	8.03205346570-01	-2.62684922530+00	-7.11229228640-01	.250000
	4.805000	2.42384653660-04	8.56039897010-02	9.14155626480-01	-3.61549488060+00	-1.06750599710+00	.250000
	5.445000	8.78295462880-05	5.82785755900-02	9.41633598860-01	-4.05637914020+00	-1.23449107150+00	.250000
	6.845000	9.70553630820-06	2.46723496650-02	9.75317944800-01	-5.01298046140+00	-1.60778948860+00	.250000
	8.405000	8.57867723360-07	9.32250845520-03	9.90676633680-01	-6.06657967190+00	-2.03046721410+00	.250000
.300000							
	0.000000	5.58593750000D-01	4.06250000000D-01	3.51562500000D-02	-2.52903927850-01	-3.91206626010-01	.300000
	.845000	1.16221780580-01	5.93133270090-01	2.90644949320-01	-9.34712475210-01	-2.26847714790-01	.300000
	1.280000	5.56450891160-02	5.38418432170-01	4.05936478710-01	-1.25457315770+00	-2.68880080930-01	.300000
	2.045000	5.28557063860-03	3.06663394950-01	6.88451035310-01	-2.27690811870+00	-5.13904908880-01	.300000
	3.360000	1.48184922950-03	2.10899361420-01	7.87627789350-01	-2.82919598130+00	-6.75943268890-01	.300000
	4.805000	1.27798425840-04	9.72081442620-02	9.02666057480-01	-3.89347507670+00	-1.01229734760+00	.300000
	5.445000	4.28897295680-05	6.79048905530-02	9.32052219720-01	-4.36764673260+00	-1.16809894640+00	.300000
	6.845000	4.01605116760-06	3.07107472170-02	9.69285236130-01	-5.39613588360+00	-1.51270961670+00	.300000
	8.405000	2.95865482470-07	1.26700332510-02	9.87329670880-01	-6.52890569940+00	-1.89722224530+00	.300000

DEC. 22,1970 CASE 1, PE EVALUATED BY EXPANSION, R* = (3/10)R, D1 = 3

ALL R VARYING GAMMA

N= 8 D= 4 D0= 3 D1= 3

R*	GAMMA	PE	1-PC-PE	PC	LOG10(PE)	LOG10(1-PC-PE)	THETA
0.000000							
	0.000000	5.585937500D-01	4.062500000000D-01	3.515625000D-02	-2.52903927850-01	-3.91206626010-01	0.000000
	.150000	5.585937500D-01	4.062500000000D-01	3.515625000D-02	-2.52903927850-01	-3.91206626010-01	.150000
	.200000	5.585937500D-01	4.062500000000D-01	3.515625000D-02	-2.52903927850-01	-3.91206626010-01	.200000
	.250000	5.585937500D-01	4.062500000000D-01	3.515625000D-02	-2.52903927850-01	-3.91206626010-01	.250000
	.300000	5.585937500D-01	4.062500000000D-01	3.515625000D-02	-2.52903927850-01	-3.91206626010-01	.300000
.253500							
	0.000000	2.235089367600-01	3.733751410900-01	3.971159213300-01	-6.39200398930-01	-4.27654599350-01	0.000000
	.150000	1.694415356200-01	4.988828101100-01	3.316756542800-01	-7.70980121290-01	-3.02001460190-01	.150000
	.200000	1.506652515990-01	5.329640508100-01	3.163834332000-01	-8.22023610950-01	-2.73302083770-01	.200000
	.250000	1.328586953040-01	5.642145543100-01	3.029264926400-01	-8.76609174220-01	-2.48555715100-01	.250000
	.300000	1.162421780580-01	5.931332700900-01	2.906449493200-01	-9.34712475210-01	-2.26847714790-01	.300000
.384000							
	0.000000	1.501237998140-01	3.181032056000-01	5.317749962700-01	-8.23550456460-01	-4.97431954390-01	0.000000
	.150000	1.622454566530-02	4.497960145800-01	4.539794387600-01	-1.78971412630+00	-3.46984396950-01	.150000
	.200000	6.107560313830-02	4.824133970900-01	4.365107715300-01	-1.09110858920+00	-3.16580639920-01	.200000
	.250000	6.758439207170-02	5.115974741100-01	4.208581301800-01	-1.17041373350+00	-2.91071179720-01	.250000
	.300000	5.564500691160-02	5.384184321700-01	4.059360478710-01	-1.25457315770+00	-2.68880080930-01	.300000
.793500							
	0.000000	3.782124567450-02	1.600859614600-01	8.014927894800-01	-1.42226413610+00	-7.94022064290-01	0.000000
	.150000	1.535409826260-02	2.500258973240-01	7.349903239400-01	-1.81387230150+00	-6.01610306080-01	.150000
	.200000	1.095492602800-02	2.690691503600-01	7.199759236200-01	-1.96039055060+00	-5.70136092640-01	.200000
	.250000	7.617974266740-03	2.871014767880-01	7.053072978500-01	-2.11475334770+00	-5.42095817300-01	.250000
	.300000	5.285570063860-03	3.064633940500-01	6.884510365100-01	-2.27690811870+00	-5.13904908860-01	.300000
1.014000							
	0.000000	1.773157413200-02	1.045181915400-01	8.777502343300-01	-1.75125270790+00	-9.80808113410-01	0.000000
	.150000	5.633790448640-03	1.867920455450-01	8.264458540600-01	-2.24919930790+00	-7.74896655120-01	.150000
	.200000	3.668545998930-03	1.808627469400-01	8.154517930700-01	-2.43350829930+00	-7.42650877390-01	.200000
	.250000	2.361297866100-03	1.944333555600-01	8.032053665700-01	-2.62684922530+00	-7.11229228660-01	.250000
	.300000	1.481849229500-03	2.108903614200-01	7.876277893500-01	-2.82919598130+00	-6.75943268890-01	.300000
1.441500							
	0.000000	4.049622952100-03	4.280653156300-02	9.531438454900-01	-2.39258541070+00	-1.36848995980+00	0.000000
	.150000	8.075430159500-04	7.187134347500-02	9.273211135100-01	-3.09283433460+00	-1.14344423690+00	.150000
	.200000	4.481016680300-04	7.788087317500-02	9.217210251600-01	-3.34862343940+00	-1.10884809720+00	.200000
	.250000	2.623646536600-04	8.560398970100-02	9.141536262800-01	-3.61549488060+00	-1.06750599710+00	.250000
	.300000	1.277982548400-04	9.720814426200-02	9.026640574800-01	-3.89347507670+00	-1.01229734760+00	.300000
1.633500							
	0.000000	2.086971987300-03	2.826024022900-02	9.696547877800-01	-2.68048338030+00	-1.54882415070+00	0.000000
	.150000	3.369512209900-04	4.817146434390-02	9.518495854400-01	-3.46986279730+00	-1.31721016060+00	.150000
	.200000	1.749309669500-04	5.231115472800-02	9.475199143000-01	-3.75713330330+00	-1.28140569320+00	.200000
	.250000	8.782554922880-05	5.827857559000-02	9.416345598860-01	-4.05637914020+00	-1.23449107150+00	.250000
	.300000	4.288972556800-05	6.790489055530-02	9.320542219720-01	-4.36764673260+00	-1.16809894640+00	.300000
2.053500							
	0.000000	4.923402967800-04	1.122655608900-02	9.882811016100-01	-3.30773461690+00	-1.94975337210+00	0.000000
	.150000	5.143331548600-05	1.962257727800-02	9.803259894100-01	-4.28875547940+00	-1.70724395190+00	.150000
	.200000	2.270469047900-05	2.143638737000-02	9.785409077400-01	-4.64388058870+00	-1.66884840360+00	.200000
	.250000	9.705536430820-06	2.467234766500-02	9.753179344800-01	-5.01298046160+00	-1.60778948860+00	.250000
	.300000	4.016651167600-06	3.071074721700-02	9.692852361300-01	-5.39613588360+00	-1.51270961670+00	.300000
2.521500							
	0.000000	9.975908588410-05	3.972954735500-03	9.959272861800-01	-4.00104753920+00	-2.40088638290+00	0.000000
	.150000	6.439024869650-06	7.068885577700-03	9.929326953200-01	-5.19117491810+00	-2.15114205640+00	.150000
	.200000	2.394943947000-06	7.759152882600-03	9.922384521700-01	-5.62070466670+00	-2.11018569090+00	.200000
	.250000	9.578677233600-07	9.422508495520-03	9.905796337680-01	-6.06657967190+00	-2.03046721410+00	.250000
	.300000	3.984654834700-07	1.267003342510-02	9.873296708800-01	-6.52090569940+00	-1.89722224530+00	.300000

19-37

DUPW P,W FOLLOW UP RUN N=15, D=7, D0=D1=1,3,5,7

N	D	D0	D1	P	Q	PC	PE	1-PC-PE	LOG10(PE)	LOG10(1-PC-PE)	R
						FOR GAMMA =	0.000000				
15	7	1	1	5.5859375D-01	4.0625000D-01	1.5489527D-22	4.0184107D-04	9.9959816D-01	-3.3959457D+00	-1.7455243D-04	0.000000
15	7	1	1	2.2951894D-01	3.7337514D-01	9.6329541D-07	2.6122090D-04	9.9973782D-01	-3.5829921D+00	-1.1388008D-04	.845000
15	7	1	1	1.5012380D-01	3.1810321D-01	7.6894309D-05	9.5907862D-05	9.9982720D-01	-4.0181458D+00	-7.5053514D-05	1.280000
15	7	1	1	3.7821249D-02	1.6068596D-01	3.6182144D-02	1.2725060D-07	9.6381773D-01	-6.8953402D+00	-1.6005089D-02	2.645000
15	7	1	1	1.7731574D-02	1.0451819D-01	1.4143742D-01	1.2751750D-09	8.5856258D-01	-8.8944302D+00	-6.6228042D-02	3.380000
15	7	1	1	4.0496230D-03	4.2806532D-02	4.8682431D-01	7.8625899D-14	5.1317069D-01	-1.3104434D+01	-2.8973816D-01	4.805000
15	7	1	1	2.0869720D-03	2.8260240D-02	6.2985960D-01	8.6902081D-16	3.7014040D-01	-1.5060970D+01	-4.3163352D-01	5.445000
15	7	1	1	4.9254030D-04	1.1226556D-02	8.3792927D-01	4.1083416D-20	1.6207073D-01	-1.9386333D+01	-7.9029540D-01	6.845000
15	7	1	1	9.9759066D-05	3.9729547D-03	9.4062056D-01	6.1243504D-25	5.9379437D-02	-2.4212940D+01	-1.2263639D+00	8.405000
						FOR GAMMA =	.150000				
15	7	1	1	5.5859375D-01	4.0625000D-01	1.5489527D-22	4.0184107D-04	9.9959816D-01	-3.3959457D+00	-1.7455243D-04	0.000000
15	7	1	1	1.6944154D-01	4.9888281D-01	6.4670157D-08	6.7771550D-06	9.9999316D-01	-5.1689526D+00	-2.9713771D-06	.845000
15	7	1	1	2.5224547D-02	4.4979601D-01	7.1703896D-06	1.1101344D-06	9.9999172D-01	-5.9546244D+00	-3.5962008D-06	1.280000
15	7	1	1	1.5354603D-02	2.5025899D-01	9.7477457D-03	1.1168342D-10	9.9025225D-01	-9.9520113D+00	-4.2541603D-03	2.645000
15	7	1	1	2.6337905D-03	1.6792036D-01	5.7308109D-02	2.5400063D-13	9.4269189D-01	-1.2595165D+01	-2.5630229D-02	3.380000
15	7	1	1	6.0754302D-04	7.1671343D-02	3.2244246D-01	7.8872877D-19	6.7755754D-01	-1.8103072D+01	-1.6905381D-01	4.805000
15	7	1	1	3.3875122D-04	4.8171452D-02	4.7430815D-01	2.2227776D-21	5.2569185D-01	-2.0653104D+01	-2.7926875D-01	5.445000
15	7	1	1	5.1433315D-05	1.9622577D-02	7.4226289D-01	5.4341202D-27	2.5773711D-01	-2.6264871D+01	-5.8882304D-01	6.845000
15	7	1	1	9.4470497D-06	7.0608559D-03	8.9907759D-01	1.6763454D-29	1.0092241D-01	-2.8775637D+01	-9.9601237D-01	8.405000
						FOR GAMMA =	.200000				
15	7	1	1	5.5859375D-01	4.0625000D-01	1.5489527D-22	4.0184107D-04	9.9959816D-01	-3.3959457D+00	-1.7455243D-04	0.000000
15	7	1	1	1.5092552D-01	5.3296405D-01	3.1857084D-08	1.9522991D-06	9.9999802D-01	-5.7094537D+00	-8.6170893D-07	.845000
15	7	1	1	5.1075831D-02	4.8241340D-01	3.9003794D-06	2.3747607D-07	9.9999578D-01	-6.6243801D+00	-1.8317952D-06	1.280000
15	7	1	1	1.0954920D-02	2.6906915D-01	7.2405174D-03	8.9329425D-12	9.9275948D-01	-1.1049005D+01	-3.1559560D-03	2.645000
15	7	1	1	3.0854600D-03	1.8086275D-01	4.6878573D-02	1.1671914D-14	9.5312143D-01	-1.3932858D+01	-2.0851767D-02	3.380000
15	7	1	1	4.4811570D-04	7.7830473D-02	2.9443702D-01	1.2167226D-20	7.0556298D-01	-1.9914808D+01	-1.5146421D-01	4.805000
15	7	1	1	1.7493097D-04	5.2311155D-02	4.4543460D-01	2.0959545D-23	5.5456540D-01	-2.2678618D+01	-2.5604723D-01	5.445000
15	7	1	1	2.2734690D-05	2.1436387D-02	7.2224541D-01	1.6799831D-28	2.7775459D-01	-2.7774695D+01	-5.5633875D-01	6.845000
						FOR GAMMA =	.250000				
15	7	1	1	5.5859375D-01	4.0625000D-01	1.5489527D-22	4.0184107D-04	9.9959816D-01	-3.3959457D+00	-1.7455243D-04	0.000000
15	7	1	1	1.1228849D-01	5.6421455D-01	1.6598118D-08	5.4569203D-07	9.9999946D-01	-6.2630524D+00	-2.4419958D-07	.845000
15	7	1	1	6.7543421D-02	5.1159796D-01	2.3016292D-06	4.8037079D-08	9.9999765D-01	-7.3184234D+00	-1.0204483D-06	1.280000
15	7	1	1	7.6779743D-03	2.8701473D-01	5.3171102D-03	6.2667760D-13	9.9468289D-01	-1.2202956D+01	-2.3153526D-03	2.645000
15	7	1	1	2.3612979D-03	1.9443336D-01	3.7359311D-02	4.5760894D-16	9.6264069D-01	-1.5339505D+01	-1.6535786D-02	3.380000
15	7	1	1	2.4234465D-04	8.5603989D-02	2.6018832D-01	1.5429543D-22	7.3981168D-01	-2.1811647D+01	-1.3087882D-01	4.805000
15	7	1	1	8.7625546D-05	5.8278576D-02	4.0572260D-01	1.6032979D-25	5.9427740D-01	-2.4794986D+01	-2.2601078D-01	5.445000
15	7	1	1	9.7022363D-06	2.4672350D-02	6.8737419D-01	1.2533718D-28	3.1262581D-01	-2.7901920D+01	-5.0497517D-01	6.845000
						FOR GAMMA =	.300000				
15	7	1	1	5.5859375D-01	4.0625000D-01	1.5489527D-22	4.0184107D-04	9.9959816D-01	-3.3959457D+00	-1.7455243D-04	0.000000
15	7	1	1	1.1622176D-01	5.9313327D-01	8.9215786D-09	1.4644384D-07	9.9999984D-01	-6.8343289D+00	-6.7474347D-08	.845000
15	7	1	1	5.5645089D-02	5.3841843D-01	1.3392794D-06	9.0369201D-09	9.9999865D-01	-8.0439796D+00	-5.8556675D-07	1.280000
15	7	1	1	5.2855706D-03	3.0626339D-01	3.6990982D-03	3.7715301D-14	9.9630090D-01	-1.3423482D+01	-1.6094766D-03	2.645000
15	7	1	1	1.4818492D-03	2.1089036D-01	2.7849375D-02	1.4981746D-17	9.7215062D-01	-1.6824438D+01	-1.2266440D-02	3.380000
15	7	1	1	1.2779825D-04	9.7208144D-02	2.1522477D-01	1.5793771D-24	7.8477523D-01	-2.3801514D+01	-1.0525471D-01	4.005000
15	7	1	1	4.2889726D-05	6.7904891D-02	3.4801958D-01	1.0000925D-27	6.5198042D-01	-2.6999960D+01	-1.8576545D-01	5.445000

FOR GAMMA = 0.000000

15	7	3	1	5.58593750D-01	4.06250000D-01	3.70716020-20	4.01841090-04	9.99598160D-01	-3.3959457D+00	-1.7455244D-04	0.000000
15	7	3	1	2.2955994D-01	3.73375140-01	9.31419090-06	4.40883380-04	9.99549800-01	-3.35567630+00	-1.9556234D-04	.845000
15	7	3	1	1.50173600-01	3.18103210-01	4.02512550-04	2.90729800-04	9.99306760-01	-3.53651040+00	-3.01175730-04	1.280000
15	7	3	1	3.76217490-02	1.60685990-01	6.17923650-02	2.12672500-06	9.38205010-01	-5.67228870+00	-2.77022530-02	2.645000
15	7	3	1	1.77315740-02	1.04518190-01	1.84295420-01	4.93841850-08	8.15704530-01	-7.30641210+00	-8.84671230-02	3.380000
15	7	3	1	4.84902300-03	4.28065320-02	5.17835190-01	1.44107840-11	4.82144810-01	-1.08413060+01	-3.16822510-01	4.805000
15	7	3	1	2.08697200-03	2.82602400-02	6.50194190-01	3.14233710-13	3.49805810-01	-1.25027470+01	-4.56172980-01	5.445000
15	7	3	1	4.92340300-04	1.12265590-02	8.44130840-01	6.41503480-17	1.55809160-01	-1.61928010+01	-8.07407020-01	6.845000
15	7	3	1	9.97590860-05	3.97295470-03	9.42033850-01	4.75579860-21	5.79661500-02	-2.03227770+01	-1.23682550+00	8.405000

FOR GAMMA = .150000

15	7	3	1	5.58593750-01	4.06250000-01	3.70716020-20	4.01841090-04	9.99598160-01	-3.39594570+00	-1.74552440-04	0.000000
15	7	3	1	1.09441540-01	4.98882810-01	5.00236110-07	1.25308350-05	9.99986910-01	-4.90202000+00	-5.68541720-06	.845000
15	7	3	1	9.02245470-02	4.49796010-01	2.99677050-05	4.36433990-06	9.99965670-01	-5.36008140+00	-1.49104740-05	1.280000
15	7	3	1	1.55006030-02	2.50258990-01	1.28040470-02	4.18091860-09	9.87195950-01	-8.37872830+00	-5.59663570-03	2.645000
15	7	3	1	5.03579050-03	1.67920350-01	6.31080550-02	2.90371220-11	9.36831940-01	-1.05370460+01	-2.83383090-02	3.380000
15	7	3	1	3.07543020-04	7.18713430-02	3.26654370-01	7.04621370-16	6.73345630-01	-1.51520440+01	-1.71761950-01	4.805000
15	7	3	1	3.30951220-04	4.81714630-02	4.76842610-01	4.85358440-18	5.23157390-01	-1.73139370+01	-2.81367630-01	5.445000
15	7	3	1	5.14333150-05	1.96225770-02	7.42847030-01	7.74883320-23	2.57152970-01	-2.21107640+01	-5.69808460-01	6.845000
15	7	3	1	5.43949870-06	7.06086500-03	8.99165040-01	3.51445850-28	1.00834960-01	-2.74541420+01	-9.96388880-01	8.405000

FOR GAMMA = .200000

15	7	3	1	5.58593750-01	4.06250000-01	3.70716020-20	4.01841090-04	9.99598160-01	-3.39594570+00	-1.74552440-04	0.000000
15	7	3	1	1.50222520-01	5.32964050-01	2.59398220-07	3.81089950-06	9.99995930-01	-5.41897250+00	-1.76771140-06	.845000
15	7	3	1	5.10756310-02	4.82413400-01	1.50678850-05	1.05538930-06	9.99983870-01	-5.97658730+00	-7.00317410-06	1.280000
15	7	3	1	1.09549260-02	2.69069120-01	8.87312880-03	4.58632580-10	9.91106940-01	-9.33853510+00	-3.87948250-03	2.645000
15	7	3	1	3.68546000-03	1.80862750-01	5.00506100-02	2.01125250-12	9.49943390-01	-1.16965330+01	-2.23022750-02	3.380000
15	7	3	1	4.48115070-04	7.78308730-02	2.96584160-01	1.94684060-17	7.03415340-01	-1.67106700+01	-1.52787860-01	4.805000
15	7	3	1	1.74930970-04	5.23111550-02	4.46658140-01	8.83034610-20	5.53331860-01	-1.90540220+01	-2.57014330-01	5.445000
15	7	3	1	2.27048800-05	2.14353370-02	7.22446780-01	5.64235950-25	2.77503220-01	-2.42485390+01	-5.56731970-01	6.845000

FOR GAMMA = .250000

15	7	3	1	5.58593750-01	4.06250000-01	3.70716020-20	4.01841090-04	9.99598160-01	-3.39594570+00	-1.74552440-04	0.000000
15	7	3	1	1.32023950-01	5.64214550-01	1.25793350-07	1.13696000-06	9.99998740-01	-5.94425480+00	-5.48407150-07	.845000
15	7	3	1	5.75439210-02	5.11597950-01	7.84248940-06	2.44890470-07	9.99991910-01	-6.61102810+00	-3.51231860-06	1.280000
15	7	3	1	7.67797430-03	2.87014730-01	6.18534100-03	4.49150170-11	9.93814660-01	-1.03476080+01	-2.69460160-03	2.645000
15	7	3	1	2.35124790-03	1.94433360-01	3.90067690-02	1.21168940-13	9.60993230-01	-1.29166090+01	-1.72796710-02	3.380000
15	7	3	1	2.42384630-04	8.56039890-02	2.61223140-01	4.52642720-19	7.38776860-01	-1.83442440+01	-1.31486720-01	4.805000
15	7	3	1	8.78255460-05	5.82785700-02	4.06290220-01	1.33696130-21	5.93709780-01	-2.08738810+01	-2.26425800-01	5.445000
15	7	3	1	7.65593630-06	2.46723550-02	6.87476800-01	3.46603540-27	3.12523200-01	-2.64601670+01	-5.05117730-01	6.845000

FOR GAMMA = .300000

15	7	3	1	5.58593750-01	4.06250000-01	3.70716020-20	4.01841090-04	9.99598160-01	-3.39594570+00	-1.74552440-04	0.000000
15	7	3	1	1.16221780-01	5.93133270-01	6.24343830-08	3.28999050-07	9.99999610-01	-6.48280530+00	-1.69997420-07	.845000
15	7	3	1	5.56450890-02	5.38418430-01	4.09307200-06	5.35090430-08	9.99995850-01	-7.27157280+00	-1.80084100-06	1.280000
15	7	3	1	5.28557000-03	3.06263390-01	4.12509460-03	3.82923120-12	9.95874910-01	-1.14168880+01	-1.79521110-03	2.645000
15	7	3	1	1.40184920-03	2.10890350-01	2.86353160-02	6.19681760-15	9.71364680-01	-1.42078310+01	-1.26176900-02	3.380000
15	7	3	1	1.27798250-04	9.72081440-02	2.15681840-01	8.67696870-21	7.84318160-01	-2.00616320+01	-1.05507730-01	4.805000
15	7	3	1	4.28897260-05	6.79048910-02	3.48259800-01	1.65391110-23	6.51740200-01	-2.27814880+01	-1.85925490-01	5.445000

FOR GAMMA = 0.000000

15	7	5	1	5.58579375D-01	4.06250000D-01	4.14302970D-18	4.01841090D-04	9.99598160D-01	-3.39594570+00	-1.74552440-04	0.000000
15	7	5	1	2.27952889D-01	3.73375140D-01	4.30788720D-05	6.27400178D-04	9.99432950D-01	-3.20245290+00	-2.91293050-04	.845000
15	7	5	1	1.30123880D-01	3.18103210D-01	1.04594851D-03	7.04792250-04	9.98249220-01	-3.15193890+00	-7.61019320-04	1.280000
15	7	5	1	3.76212490D-02	1.60085900D-01	7.02525870D-02	2.75499620-05	9.29719860-01	-4.55987900+00	-3.16478900-02	2.645000
15	7	5	1	1.77315740D-02	1.04518140D-01	1.90355890D-01	1.47082825D-06	8.09642640-01	-5.83024260+00	-9.17066300-02	3.380000
15	7	5	1	4.04790230D-03	4.28006530D-02	5.18777930D-01	2.03842150-09	4.81222070-01	-8.69070600+00	-3.17654470-01	4.805000
15	7	5	1	2.06007200D-03	2.08260240D-02	6.50500550D-01	8.76718820-11	3.49499450-01	-1.00571400+01	-4.56553510-01	5.445000
15	7	5	1	9.52340430D-04	1.12265580D-02	8.44212680D-01	7.72765890-14	1.55787320-01	-1.31119520+01	-8.07467890-01	6.845000
15	7	5	1	4.97594060D-05	3.97295470-03	9.42034040-01	2.84882950-17	5.79651590-02	-1.65453340+01	-1.23683300+00	8.405000

FOR GAMMA = .150000

15	7	5	1	5.58579375D-01	4.06250000D-01	4.14302970D-18	4.01841090D-04	9.99598160D-01	-3.39594570+00	-1.74552440-04	0.000000
15	7	5	1	1.69441540D-01	4.49868290D-01	2.33240780D-06	1.92884070-05	9.99978380-01	-4.71470360+00	-9.38990200-06	.845000
15	7	5	1	5.62245470D-02	4.49796010D-01	6.37922090D-05	1.35761830-05	9.99922630-01	-4.86722230+00	-3.36019650-05	1.280000
15	7	5	1	1.53509880D-02	2.50258990D-01	1.32512400D-02	1.20986420-07	9.86748640-01	-6.91726340+00	-5.79346410-03	2.645000
15	7	5	1	3.99379250D-03	1.67920360D-01	6.34476820D-02	2.56243180-09	9.36552320-01	-8.59134770+00	-2.84679580-02	3.380000
15	7	5	1	9.07343020D-04	7.18713430D-02	3.26660040D-01	4.85641420-13	6.73319960-01	-1.23136840+01	-1.71778510-01	4.805000
15	7	5	1	3.38951220D-04	4.81714630D-02	4.76848430D-01	8.17597640-15	5.23151070-01	-1.40874600+01	-2.81372880-01	5.445000
15	7	5	1	9.14333150D-05	1.96225770D-02	7.42847250D-01	8.86179230-19	2.57152750-01	-1.80524790+01	-5.89808820-01	6.845000
15	7	5	1	8.43279870D-06	7.06086650D-03	8.99105350D-01	3.09659060-23	1.00834950-01	-2.25091160+01	-9.96388900-01	8.405000

FOR GAMMA = .200000

15	7	5	1	5.58579375D-01	4.06250000D-01	4.14302970D-18	4.01841090D-04	9.99598160D-01	-3.39594570+00	-1.74552440-04	0.000000
15	7	5	1	1.51893450D-01	5.32964030D-01	1.01703740D-06	6.15283330-06	9.99992830-01	-5.21092480+00	-3.11419380-06	.845000
15	7	5	1	5.10758310D-02	4.82413400D-01	2.94879400D-05	3.69756710-06	9.99966810-01	-5.43208390+00	-1.44125220-05	1.280000
15	7	5	1	1.70954920D-02	2.69069150D-01	9.06507060D-03	1.88191640-08	9.90430910-01	-7.74012820+00	-3.95662400-03	2.645000
15	7	5	1	5.68546000D-03	1.80862730D-01	5.01571530D-02	2.67469830-10	9.49842850-01	-9.57272520+00	-2.23482440-02	3.380000
15	7	5	1	4.48161670D-04	7.78308730D-02	2.96591470D-01	2.40317410-14	7.03408530-01	-1.36192150+01	-1.52792370-01	4.805000
15	7	5	1	1.74933970D-04	5.23111550D-02	4.46669740D-01	2.86997700-16	5.53330760-01	-1.55421220+01	-2.57015580-01	5.445000
15	7	5	1	2.22744980D-05	2.14363870D-02	7.22426820D-01	1.65867670-20	2.77503180-01	-1.98360410+01	-5.56732040-01	6.845000
15	7	5	1	2.39494340D-06	7.75915290D-03	8.89726490D-01	2.18867450-25	1.10273510-01	-2.46598190+01	-9.57528810-01	8.405000

FOR GAMMA = .250000

15	7	5	1	5.58579375D-01	4.06250000D-01	4.14302970D-18	4.01841090D-04	9.99598160D-01	-3.39594570+00	-1.74552440-04	0.000000
15	7	5	1	1.32203450D-01	5.64214500-01	4.61032930-07	1.94583610-06	9.99997590-01	-5.71089370+00	-1.04529120-06	.845000
15	7	5	1	6.75949210D-02	5.11597900-01	1.43672950-05	9.80832430-07	9.99984450-01	-6.00840520+00	-6.53536790-06	1.280000
15	7	5	1	2.67797430D-03	2.87014730-01	6.25150200-03	2.48594080-09	9.93748500-01	-8.60450920+00	-2.72351580-03	2.645000
15	7	5	1	2.30129790D-03	1.94433360-01	3.90400720-02	2.47574480-11	9.60959330-01	-1.06062940+01	-1.72949930-02	3.380000
15	7	5	1	2.42304650D-04	6.56039890-02	2.61225000-01	1.02438860-15	7.38774940-01	-1.49895350+01	-1.31487840-01	4.805000
15	7	5	1	8.78259460D-05	5.82785700-02	4.06290590-01	8.60095370-18	5.93709410-01	-1.70654530+01	-2.26426070-01	5.445000
15	7	5	1	9.70053630D-06	2.46723500-02	6.87476800-01	2.01429400-22	3.12523200-01	-2.16958770+01	-5.05117740-01	6.845000
15	7	5	1	9.57007720D-07	9.32259500-03	8.66942420-01	1.01560970-27	1.31070710-01	-2.69932730+01	-8.82494350-01	8.405000

FOR GAMMA = .300000

15	7	5	1	5.58579375D-01	4.06250000D-01	4.14302970D-18	4.01841090D-04	9.99598160D-01	-3.39594570+00	-1.74552440-04	0.000000
15	7	5	1	1.16221780D-01	5.93133270-01	2.12223590-07	6.02917170-07	9.99999180-01	-6.21974230+00	-3.54011280-07	.845000
15	7	5	1	5.56450840D-02	5.38418430-01	6.73546870-06	2.48166190-07	9.99993020-01	-6.60525740+00	-3.03296470-06	1.280000
15	7	5	1	2.48553960D-02	3.06626330-01	4.13798870-03	3.00138550-10	9.95852010-01	-9.52267820+00	-1.80519530-03	2.645000
15	7	5	1	1.48164920D-03	2.10890360-01	2.06450670-02	1.99776470-12	9.71354330-01	-1.17038510+01	-1.26223180-02	3.380000
15	7	5	1	1.47778650D-04	9.72081140-02	2.15662290-01	3.67742590-17	7.84317710-01	-1.64344560+01	-1.05507980-01	4.805000
15	7	5	1	4.28897200D-05	6.79048910-02	3.48259870-01	2.15653960-19	6.51740130-01	-1.86662430+01	-1.85925540-01	5.445000
15	7	5	1	4.01005120D-06	3.07107470-02	6.26326750-01	2.29815690-24	3.73673250-01	-2.36386200+01	-4.27507990-01	6.845000

7-39

15	7	7	1	5.5835375D-01	4.0625000D-01	2.8684585D-16	4.0184109D-04	9.9959816D-01	-3.3959457D+00	-1.7455244D-04	0.000000
15	7	7	1	2.2953894D-01	3.7337514D-01	1.2779761D-04	7.7410082D-04	9.9990819D-01	-3.1112025D+00	-3.9182713D-04	.845000
15	7	7	1	1.5011738D-01	3.1810321D-01	1.8331675D-03	1.3714761D-03	9.9907953D-01	-2.8628118D+00	-1.3933938D-03	1.280000
15	7	7	1	3.7841249D-02	1.6068596D-01	7.1982460D-02	2.7244045D-04	9.2774510D-01	-3.5647284D+00	-3.2571331D-02	2.645000
15	7	7	1	1.7731574D-02	1.0451481D-01	1.7908864D-01	3.3629862D-05	8.0907996D-01	-4.4732749D+00	-9.2008555D-02	3.380000
15	7	7	1	4.0496230D-03	4.2806532D-02	5.1077442D-01	2.1857633D-07	4.8120486D-01	-6.6603969D+00	-3.1766999D-01	4.805000
15	7	7	1	2.0866722D-03	2.8260240D-02	6.5056341D-01	1.8536887D-08	3.4949657D-01	-7.7319632D+00	-4.5655708D-01	5.445000
15	7	7	1	4.9234403D-04	1.1226556D-02	8.4421272D-01	7.0527047D-11	1.5578728D-01	-1.0151644D+01	-8.0746802D-01	6.845000
15	7	7	1	9.9752705D-05	3.9729547D-03	9.4203446D-01	1.2928322D-13	5.7965158D-02	-1.2888458D+01	-1.2368330D+00	8.405000

FOR GAMMA = .150000

15	7	7	1	5.5835375D-01	4.0625000D-01	2.8684585D-16	4.0184109D-04	9.9959816D-01	-3.3959457D+00	-1.7455244D-04	0.000000
15	7	7	1	1.6944154D-01	4.9888281D-01	6.2655510D-06	2.5300998D-05	9.9996844D-01	-4.5968623D+00	-1.3705052D-05	.845000
15	7	7	1	9.6224547D-02	4.4979601D-01	9.4859488D-05	3.3331053D-05	9.9987181D-01	-4.4771510D+00	-5.5676013D-05	1.280000
15	7	7	1	1.5350869D-02	2.6925892D-01	1.3249174D-04	2.6610223D-06	9.8670559D-01	-5.5749515D+00	-5.8124101D-03	2.645000
15	7	7	1	3.6337905D-03	1.6794036D-01	6.3455942D-02	1.7148741D-07	9.3654389D-01	-6.7657678D+00	-2.8471867D-02	3.380000
15	7	7	1	8.9763392D-04	7.1871344D-02	3.2003014D-01	2.5360592D-10	6.7391986D-01	-9.5958406D+00	-1.7177857D-01	4.805000
15	7	7	1	3.3895122D-04	4.8171455D-02	4.7664489D-01	1.0434372D-11	5.2315106D-01	-1.0981534D+01	-2.8137289D-01	5.445000
15	7	7	1	2.1433315D-05	1.9622677D-02	7.4284725D-01	7.2777956D-15	2.5715275D-01	-1.4114763D+01	-5.8980882D-01	6.845000
15	7	7	1	8.4397967D-06	7.0608656D-03	8.9916055D-01	2.1704866D-18	1.0083495D-01	-1.7663443D+01	-9.9638890D-01	8.405000

FOR GAMMA = .200000

15	7	7	1	5.5835375D-01	4.0625000D-01	2.8684585D-16	4.0184109D-04	9.9959816D-01	-3.3959457D+00	-1.7455244D-04	0.000000
15	7	7	1	1.3008525D-01	5.3296403D-01	2.5626049D-06	8.3884085D-06	9.9998903D-01	-5.0763204D+00	-4.7647636D-06	.845000
15	7	7	1	8.1137534D-02	4.9241348D-01	4.1092490D-05	1.0163674D-05	9.9994874D-01	-4.9929493D+00	-2.2260804D-05	1.280000
15	7	7	1	1.9854920D-02	2.6906915D-01	9.0806759D-03	5.4793821D-07	9.9918780D-01	-6.2612684D+00	-3.9619424D-03	2.645000
15	7	7	1	3.0882460D-03	1.8086627D-01	5.0153122D-02	2.6965589D-08	9.4384085D-01	-7.5691901D+00	-2.2349156D-02	3.380000
15	7	7	1	4.4811110D-04	7.7830873D-02	2.6059148D-01	2.2474910D-11	7.0340852D-01	-1.0648302D+01	-1.5279238D-01	4.805000
15	7	7	1	1.7493077D-04	3.2311152D-02	4.4066077D-01	7.0667123D-13	5.5332026D-01	-1.2150783D+01	-2.5701558D-01	5.445000
15	7	7	1	2.2743485D-05	2.1436387D-02	7.2249682D-01	2.0575992D-16	2.7750318D-01	-1.5543999D+01	-5.5673204D-01	6.845000
15	7	7	1	6.3954433D-06	7.7591529D-03	8.8972644D-01	4.1214573D-20	1.1027351D-01	-1.9384907D+01	-9.5752821D-01	8.405000

FOR GAMMA = .250000

15	7	7	1	5.5835375D-01	4.0625000D-01	2.8684585D-16	4.0184109D-04	9.9959816D-01	-3.3959457D+00	-1.7455244D-04	0.000000
15	7	7	1	1.1426278D-01	5.0421465D-01	1.0981672D-06	2.7841483D-06	9.9999612D-01	-5.5553076D+00	-1.6860715D-06	.845000
15	7	7	1	6.7548421D-02	5.1159780D-01	1.8396404D-05	3.0651808D-06	9.9997854D-01	-5.5135439D+00	-9.3207478D-06	1.280000
15	7	7	1	1.6779743D-02	2.8701473D-01	6.2534622D-03	1.0441110D-07	9.9374527D-01	-6.9812533D+00	-2.7249243D-03	2.645000
15	7	7	1	2.3612497D-03	1.9443336D-01	3.9041104D-02	3.8339090D-09	9.6095889D-01	-8.4163582D+00	-1.7295190D-02	3.380000
15	7	7	1	2.4238465D-04	8.5603989D-02	2.8122256D-01	1.7563762D-12	7.3877494D-01	-1.1755382D+01	-1.3148785D-01	4.805000
15	7	7	1	8.7625540D-05	5.8278576D-02	4.0629059D-01	4.1918586D-14	5.9370941D-01	-1.3377593D+01	-2.2642607D-01	5.445000
15	7	7	1	7.1055436D-06	2.4672350D-02	6.8747680D-01	9.2000665D-18	3.1252320D-01	-1.7036171D+01	-5.0511774D-01	6.845000
15	7	7	1	6.5786772D-07	9.3225085D-03	8.8892429D-01	6.6690772D-22	1.1107071D-01	-2.1175934D+01	-8.8249435D-01	8.405000

FOR GAMMA = .300000

15	7	7	1	5.5835375D-01	4.0625000D-01	2.8684585D-16	4.0184109D-04	9.9959816D-01	-3.3959457D+00	-1.7455244D-04	0.000000
15	7	7	1	1.1622178D-01	5.9313327D-01	4.7177735D-07	9.1428471D-07	9.9999861D-01	-6.0389185D+00	-6.0195952D-07	.845000
15	7	7	1	5.5045089D-02	5.3841843D-01	8.3050684D-06	8.9364041D-07	9.9999080D-01	-6.0488372D+00	-3.9949668D-06	1.280000
15	7	7	1	2.2855700D-03	3.0626339D-01	4.1487503D-03	1.7843136D-08	9.9505123D-01	-7.7485288D+00	-1.8055351D-03	2.645000
15	7	7	1	1.4816442D-03	2.1089037D-01	2.8042751D-02	4.7827708D-10	9.7145425D-01	-9.3203204D+00	-1.2622356D-02	3.380000
15	7	7	1	1.2779825D-04	9.7208144D-02	2.1566229D-01	1.1807427D-13	7.8431771D-01	-1.2927845D+01	-1.0550798D-01	4.805000
15	7	7	1	4.2489726D-05	6.7904891D-02	3.4825987D-01	2.1302589D-15	6.5174013D-01	-1.4671568D+01	-1.8592554D-01	5.445000
15	7	7	1	1.0100512D-05	3.0710747D-02	6.2032675D-01	2.5209292D-19	3.7367325D-01	-1.8598439D+01	-4.2750799D-01	6.845000
15	7	7	1	2.9280954D-07	1.2670034D-02	8.2591491D-01	9.0904601D-24	1.7408609D-01	-2.3041414D+01	-7.5923592D-01	8.405000

7-46

FOR GAMMA = 0.000000

15	7	1	3	3.58[illegible]37[illegible]-01	4.06250000-01	2.19874990-18	2.42785480-02	9.75721450-01	-1.61477730+00	-1.06741470-02	0.000000
15	7	1	3	2.29[illegible]8940-01	3.73375140-01	1.03962790-04	2.20183160-02	9.77877720-01	-1.65721590+00	-9.71544840-03	.845000
15	7	1	3	1.5[illegible]38[illegible]-01	3.18103210-01	3.65[illegible]2[illegible]-03	1.16470950-02	9.84696920-01	-1.93378240+00	-6.69742230-03	1.280000
15	7	1	3	8.78212490-02	1.60685900-01	2.97691330-01	5.29558080-05	7.02255720-01	-4.27608640+00	-1.53504720-01	2.645000
15	7	1	3	1.77[illegible]157[illegible]-02	1.04[illegible]8[illegible]-01	6.04[illegible]2020-01	9.90812[illegible]-07	3.65[illegible]6690-01	-6.00[illegible]8600+00	-4.02999600-01	3.380000
15	7	1	3	4.04[illegible]2300-03	4.28065320-02	9.17[illegible]083[illegible]-01	1.90653260-10	8.21091730-02	-9.71975580+00	-1.08560830+00	4.805000
15	7	1	3	2.[illegible]8[illegible]720[illegible]-03	2.[illegible]2[illegible]240[illegible]-02	9.[illegible]1[illegible]184[illegible]-01	3.43199560-12	3.[illegible]8[illegible]165[illegible]-02	-1.[illegible]4[illegible]453[illegible]+01	-1.[illegible]1332[illegible]+00	5.445000
15	7	1	3	4.92[illegible]3[illegible]-04	1.12265560-02	9.92061880-01	4.55262380-16	7.93811610-03	-1.53417380+01	-2.10028260+00	6.845000
15	7	1	3	2.[illegible]7[illegible]8[illegible]-05	3.[illegible]2[illegible]-03	9.[illegible]8[illegible]-01	2.[illegible]1[illegible]-20	1.52283950-03	-1.96067310+01	-2.81734590+00	8.405000

FOR GAMMA = .150000

15	7	1	3	3.58[illegible]3750-01	4.06250000-01	2.19874990-18	2.42785480-02	9.75721450-01	-1.61477730+00	-1.06741470-02	0.000000
15	7	1	3	1.[illegible]4[illegible]-01	4.[illegible]-01	1.[illegible]-05	1.67941180-03	9.98403700-01	-2.77484280+00	-7.37318440-06	.845000
15	7	1	3	9.62245470-02	4.49796010-01	8.52014160-04	5.89344760-04	9.98557840-01	-3.22963060+00	-6.26773720-04	1.280000
15	7	1	3	1.53[illegible]-02	2.[illegible]-01	1.78429410-01	6.49064950-07	8.21569950-01	-6.19337950+00	-8.53554560-02	2.645000
15	7	1	3	3.63[illegible]-03	1.67920360-01	4.80387360-01	4.80281950-09	5.19612640-01	-8.31850370+00	-2.84320290-01	3.380000
15	7	1	3	3.[illegible]-04	7.[illegible]-02	9.[illegible]-01	1.31728420-13	9.93248050-02	-1.28803210+01	-1.00294230+00	4.805000
15	7	1	3	3.[illegible]-04	4.81714690-02	9.62[illegible]-01	9.45127780-16	3.78466840-02	-1.50245090+01	-1.42197220+00	5.445000
15	7	1	3	2.[illegible]-05	1.[illegible]-02	9.[illegible]-01	1.[illegible]-20	3.64975190-03	-1.97961240+01	-2.43773670+00	6.845000
15	7	1	3	5.43[illegible]-06	7.[illegible]-03	9.99753130-01	7.05062420-26	2.46873510-04	-2.51517720+01	-3.60752550+00	8.405000

FOR GAMMA = .200000

15	7	1	3	3.58[illegible]3750-01	4.06250000-01	2.19874990-18	2.42785480-02	9.75721450-01	-1.61477730+00	-1.06741470-02	0.000000
15	7	1	3	1.[illegible]-01	5.32964050-01	1.03269600-03	6.76483480-04	9.99313190-01	-3.16974280+00	-2.98381330-04	.845000
15	7	1	3	9.[illegible]-02	4.[illegible]-01	5.[illegible]-04	1.99314390-04	9.99223260-01	-3.70046130+00	-3.38768870-04	1.280000
15	7	1	3	1.[illegible]-02	2.69069150-01	1.34011340-01	1.15295730-07	8.45988540-01	-6.93818680+00	-7.26355180-02	2.645000
15	7	1	3	3.[illegible]-03	1.[illegible]-01	4.[illegible]-01	5.95220470-10	5.55020930-01	-9.22532210+00	-2.55690640-01	3.380000
15	7	1	3	4.48[illegible]-04	7.[illegible]-02	8.[illegible]-01	7.72440860-15	1.12187420-01	-1.41121350+01	-9.50[illegible]820-01	4.805000
15	7	1	3	1.[illegible]-06	5.[illegible]-02	9.[illegible]-01	3.94406300-17	4.31287230-02	-1.64[illegible]+01	-1.36523340+00	5.445000
15	7	1	3	2.27[illegible]-05	2.14363870-02	9.95966360-01	3.15113500-22	4.03364450-03	-2.15015330+01	-2.39430240+00	6.845000
15	7	1	3	2.39[illegible]-06	7.[illegible]-03	9.[illegible]-01	5.87[illegible]-28	2.34117840-04	-2.72308730+01	-3.63056550+00	8.405000

FOR GAMMA = .250000

15	7	1	3	3.58[illegible]3750-01	4.06250000-01	2.19874990-18	2.42785480-02	9.75721450-01	-1.61477730+00	-1.06741470-02	0.000000
15	7	1	3	1.[illegible]-01	5.64214530-01	6.52[illegible]-06	2.64295400-04	9.99729180-01	-3.57791040+00	-1.17632270-04	.845000
15	7	1	3	5.[illegible]-02	5.11597900-01	4.01387090-04	6.39704670-05	9.99534640-01	-4.19402050+00	-2.02149260-04	1.280000
15	7	1	3	1.[illegible]-03	2.[illegible]-01	1.[illegible]-01	1.86219470-08	8.69774980-01	-7.72997490+00	-6.05930870-02	2.645000
15	7	1	3	2.[illegible]-03	1.94433300-01	4.02880780-01	6.54842110-11	5.97119220-01	-1.01838630+01	-2.23938950-01	3.380000
15	7	1	3	2.[illegible]-04	8.[illegible]-02	8.[illegible]-01	4.04571430-16	1.34774030-01	-1.53930020+01	-8.70397010-01	4.805000
15	7	1	3	8.[illegible]-05	5.82785780-02	9.45502880-01	1.48320150-18	5.44371170-02	-1.78287820+01	-1.26410490+00	5.445000
15	7	1	3	3.[illegible]-05	2.[illegible]-02	9.[illegible]-01	5.80139390-24	5.61498910-03	-2.32366680+01	-2.25065110+00	6.845000

FOR GAMMA = .300000

15	7	1	3	3.58[illegible]3750-01	4.06250000-01	2.19874990-18	2.42785480-02	9.75721450-01	-1.61477730+00	-1.06741470-02	0.000000
15	7	1	3	1.[illegible]-01	5.93133270-01	4.[illegible]-06	9.95352410-05	9.99096280-01	-4.00202320+00	-4.50467300-05	.845000
15	7	1	3	5.58[illegible]-02	5.38418430-01	2.75375970-04	1.92896710-05	9.99705330-01	-4.71467520+00	-1.27990520-04	1.280000
15	7	1	3	2.[illegible]-03	3.06203390-01	1.05247840-01	2.68134490-09	8.94752160-01	-8.57164730+00	-4.82972460-02	2.645000
15	7	1	3	1.48[illegible]-03	2.10690360-01	3.49342000-01	6.39111230-12	6.50658000-01	-1.11944240+01	-1.86647230-01	3.380000
15	7	1	3	1.27[illegible]-04	9.72081440-02	8.[illegible]-01	1.91990290-17	1.75030590-01	-1.67167210+01	-7.56886030-01	4.805000
15	7	1	3	4.[illegible]-05	6.79048910-02	9.22306420-01	5.15200570-20	7.76935810-02	-1.92880240+01	-1.10961490+00	5.445000
15	7	1	3	4.01[illegible]-06	3.07107470-02	9.[illegible]-01	1.03699870-25	1.00487370-02	-2.49842220+01	-1.99788850+00	6.845000

FOR GAMMA = 0.000000

15	7	3	3	5.585937D-01	4.062500D-01	2.235066D-18	2.427854D-02	9.757214D-01	-1.614777D+00	-1.067414D-02	0.000000
15	7	3	3	2.295089D-01	3.737514D-01	1.123136D-04	2.219797D-02	9.776897D-01	-1.653686D+00	-9.798956D-03	.845000
15	7	3	3	1.501263D-01	3.181032D-01	3.981607D-03	1.184191D-02	9.841764D-01	-1.926578D+00	-6.927019D-03	1.280000
15	7	3	3	3.782124D-02	1.606859D-01	3.233020D-01	5.495528D-05	6.766430D-01	-4.259990D+00	-1.696404D-01	2.645000
15	7	3	3	1.773157D-02	1.045181D-01	6.474900D-01	1.038921D-06	3.525089D-01	-5.983417D+00	-4.528298D-01	3.380000
15	7	3	3	4.049623D-03	4.280653D-02	9.489167D-01	2.049856D-10	5.108329D-02	-9.688276D+00	-1.291721D+00	4.805000
15	7	3	3	2.086972D-03	2.826024D-02	9.817264D-01	3.745360D-12	1.827357D-02	-1.142650D+01	-1.738176D+00	5.445000
15	7	3	3	4.923403D-04	1.122655D-02	9.953234D-01	5.193716D-16	1.676542D-03	-1.528452D+01	-2.775585D+00	6.845000
15	7	3	3	2.975908D-05	3.972954D-03	9.998704D-01	2.532683D-20	1.095518D-04	-1.959641D+01	-3.960380D+00	8.405000

FOR GAMMA = .150000

15	7	3	3	5.585937D-01	4.062500D-01	2.235066D-18	2.427854D-02	9.757214D-01	-1.614777D+00	-1.067414D-02	0.000000
15	7	3	3	1.694415D-01	4.988628D-01	1.738160D-05	1.685165D-03	9.982874D-01	-2.773357D+00	-7.400370D-04	.845000
15	7	3	3	7.622454D-02	4.497960D-01	8.756114D-04	5.925989D-04	9.985317D-01	-3.227239D+00	-6.381042D-04	1.280000
15	7	3	3	1.533066D-02	2.650258D-01	1.814857D-01	6.447187D-07	8.183136D-01	-6.190629D+00	-8.697407D-02	2.645000
15	7	3	3	5.633790D-03	1.679203D-01	4.802473D-01	4.831604D-09	5.137526D-01	-8.315908D+00	-2.899245D-01	3.380000
15	7	3	3	8.073436D-04	7.187134D-02	9.046871D-01	1.324321D-13	9.511289D-02	-1.287800D+01	-1.021760D+00	4.805000
15	7	3	3	3.389512D-04	4.817146D-02	9.648687D-01	9.499791D-16	3.531222D-02	-1.502228D+01	-1.452074D+00	5.445000
15	7	3	3	2.143331D-05	1.982257D-02	9.969344D-01	1.006851D-20	3.065603D-03	-1.979402D+01	-2.513484D+00	6.845000
15	7	3	3	8.437098D-06	7.068865D-03	9.998405D-01	7.084092D-26	1.594166D-04	-2.514971D+01	-3.797466D+00	8.405000

FOR GAMMA = .200000

15	7	3	3	5.585937D-01	4.062500D-01	2.235066D-18	2.427854D-02	9.757214D-01	-1.614777D+00	-1.067414D-02	0.000000
15	7	3	3	1.505252D-01	5.329643D-01	1.055650D-05	6.783420D-04	9.993111D-01	-3.168551D+00	-2.992879D-04	.845000
15	7	3	3	8.107369D-02	4.824134D-01	5.915151D-04	2.001342D-04	9.992083D-01	-3.698682D+00	-3.439442D-04	1.280000
15	7	3	3	1.093492D-02	2.690691D-01	1.558003D-01	1.157454D-07	6.443360D-01	-6.936496D+00	-7.348469D-02	2.645000
15	7	3	3	3.683469D-03	1.808627D-01	4.481571D-01	5.972200D-10	5.516428D-01	-9.223865D+00	-2.581845D-01	3.380000
15	7	3	3	4.461016D-04	7.783087D-02	8.899597D-01	7.743864D-15	1.100402D-01	-1.411104D+01	-9.584483D-01	4.805000
15	7	3	3	1.749369D-04	5.231115D-02	9.581046D-01	3.949458D-17	4.184751D-02	-1.640346D+01	-1.377836D+00	5.445000
15	7	3	3	2.271486D-05	2.143638D-02	9.906177D-01	3.156775D-22	3.782272D-03	-2.150075D+01	-2.422247D+00	6.845000
15	7	3	3	2.394943D-06	7.759152D-03	9.997860D-01	5.885419D-28	2.017063D-04	-2.723022D+01	-3.694850D+00	8.405000

FOR GAMMA = .250000

15	7	3	3	5.585937D-01	4.062500D-01	2.235066D-18	2.427854D-02	9.757214D-01	-1.614777D+00	-1.067414D-02	0.000000
15	7	3	3	1.325849D-01	5.646145D-01	6.635432D-06	2.648886D-04	9.995284D-01	-3.576939D+00	-1.179365D-04	.845000
15	7	3	3	8.754392D-02	5.115979D-01	4.069279D-04	6.415732D-05	9.995289D-01	-4.192686D+00	-2.046422D-04	1.280000
15	7	3	3	1.677974D-03	2.870147D-01	1.310424D-01	1.866023D-08	8.689067D-01	-7.728943D+00	-6.102682D-02	2.645000
15	7	3	3	2.361297D-03	1.944333D-01	4.045284D-01	6.560492D-11	5.954717D-01	-1.018306D+01	-2.251388D-01	3.380000
15	7	3	3	2.462945D-04	8.560396D-02	8.662617D-01	4.050268D-16	1.337382D-01	-1.539251D+01	-8.737445D-01	4.805000
15	7	3	3	6.782554D-05	5.827857D-02	9.401305D-01	1.484596D-18	5.386949D-02	-1.782839D+01	-1.268657D+00	5.445000
15	7	3	3	9.710536D-06	2.467235D-02	9.944870D-01	5.807736D-24	5.512386D-03	-2.323621D+01	-2.258660D+00	6.845000

FOR GAMMA = .300000

15	7	3	3	5.585937D-01	4.062500D-01	2.235066D-18	2.427854D-02	9.757214D-01	-1.614777D+00	-1.067414D-02	0.000000
15	7	3	3	1.162617D-01	5.931332D-01	4.238084D-06	9.971777D-05	9.998960D-01	-4.001227D+00	-4.514926D-05	.845000
15	7	3	3	5.564508D-02	5.384184D-01	2.781297D-04	1.933414D-05	9.997025D-01	-4.713675D+00	-1.292061D-04	1.280000
15	7	3	3	5.285570D-03	3.062633D-01	1.056738D-01	2.685136D-09	8.943261D-01	-8.571033D+00	-4.850406D-02	2.645000
15	7	3	3	1.481594D-03	2.108903D-01	3.501279D-01	6.397294D-12	6.498720D-01	-1.119400D+01	-1.871721D-01	3.380000
15	7	3	3	1.277942D-04	9.720814D-02	8.254204D-01	1.920770D-17	1.745735D-01	-1.671652D+01	-7.580216D-01	4.805000
15	7	3	3	4.268976D-05	6.790489D-02	9.225406D-01	5.153695D-20	7.745336D-02	-1.928788D+01	-1.110959D+00	5.445000
15	7	3	3	4.010651D-06	3.071074D-02	9.892970D-01	1.037157D-25	1.000980D-02	-2.498415D+01	-1.999574D+00	6.845000

FOR GAMMA = 0.000000

15	7	5	3	2.5859375D-01	4.0625000D-01	4.6000022D-16	2.4278548D-02	9.7572145D-01	-1.6147773D+00	-1.0674147D-02	0.000000
15	7	5	3	2.2950894D-01	3.7337514D-01	9.2780655D-04	3.1270671D-02	9.6780152D-01	-1.5048628D+00	-1.4213699D-02	.845000
15	7	5	3	1.9012380D-01	3.1810321D-01	1.7955158D-02	2.7853850D-02	9.5419099D-01	-1.5551148D+00	-2.0364688D-02	1.280000
15	7	5	3	3.7821249D-02	1.6066596D-01	4.9731926D-01	6.7271421D-04	5.0200803D-01	-3.1721694D+00	-2.9928934D-01	2.645000
15	7	5	3	1.7735740D-02	1.0451817D-01	7.8012957D-01	2.9242315D-05	2.1967505D-01	-4.5339883D+00	-6.5821927D-01	3.380000
15	7	5	3	4.0490230D-03	4.2806532D-02	9.7504171D-01	2.7159480D-08	2.4958260D-02	-7.5660786D+00	-1.6027857D+00	4.805000
15	7	5	3	2.0869720D-03	2.8260240D-02	9.9190162D-01	9.7846887D-10	8.0983823D-03	-9.0094530D+00	-2.0916017D+00	5.445000
15	7	5	3	4.9234030D-04	1.1226558D-02	9.9941463D-01	5.8650810D-13	5.8536506D-04	-1.2231726D+01	-3.2325732D+00	6.845000
15	7	5	3	2.3759088D-05	3.8972954D-03	9.9997282D-01	1.4277599D-16	2.7583774D-05	-1.5046345D+01	-4.5593463D+00	8.405000

FOR GAMMA = .150000

15	7	5	3	2.5859375D-01	4.0625000D-01	4.6000022D-16	2.4278548D-02	9.7572145D-01	-1.6147773D+00	-1.0674147D-02	0.000000
15	7	5	3	1.6944159D-01	4.9688829D-01	1.3161600D-04	2.5417253D-03	9.9732660D-01	-2.5948714D+00	-1.1625720D-03	.845000
15	7	5	3	9.0224547D-02	4.4979601D-01	3.2621550D-03	1.7596059D-03	9.9497818D-01	-2.7545698D+00	-2.1864437D-03	1.280000
15	7	5	3	1.5352660D-02	2.5025892D-01	2.2881096D-01	1.7382447D-05	7.7117165D-01	-4.7598891D+00	-1.1284894D-01	2.645000
15	7	5	3	2.0537490D-03	1.6792030D-01	5.2521006D-01	3.9607345D-07	4.7478893D-01	-6.4022243D+00	-3.2349942D-01	3.380000
15	7	5	3	8.0754302D-04	7.1871344D-02	9.1176534D-01	8.4619127D-11	8.8214696D-02	-1.0072531D+01	-1.0544591D+00	4.805000
15	7	5	3	3.3695122D-04	4.8171463D-02	9.6708163D-01	1.4830645D-12	3.2918374D-02	-1.1828840D+01	-1.4825616D+00	5.445000
15	7	5	3	2.1433512D-05	1.9822377D-02	9.9771160D-01	1.7024829D-16	2.8803960D-03	-1.5766917D+01	-2.5405478D+00	6.845000
15	7	5	3	8.4390787D-06	7.0608656D-03	9.9964970D-01	6.0698031D-21	1.5030344D-04	-2.0216825D+01	-3.8230311D+00	8.405000

FOR GAMMA = .200000

15	7	5	3	2.5859375D-01	4.0625000D-01	4.6000022D-16	2.4278548D-02	9.7572145D-01	-1.6147773D+00	-1.0674147D-02	0.000000
15	7	5	3	1.5066526D-01	5.3296403D-01	7.5439596D-05	1.0670507D-03	9.9865551D-01	-2.9710017D+00	-4.9733044D-04	.845000
15	7	5	3	8.1072331D-02	4.8824134D-01	2.0100252D-03	8.6643329D-04	9.9723606D-01	-3.1763046D+00	-1.1639020D-03	1.280000
15	7	5	3	1.0934926D-02	2.6906915D-01	1.0548932D-01	4.2690783D-06	8.1450641D-01	-5.3696659D+00	-8.9105493D-02	2.645000
15	7	5	3	3.0804881D-03	1.8086627D-01	4.7235248D-01	7.3686009D-08	5.2764794D-01	-7.1326150D+00	-2.7656160D-01	3.380000
15	7	5	3	4.4810107D-04	7.7830873D-02	8.9389849D-01	8.8564540D-12	1.0610151D-01	-9.1052740D+01	-9.7427845D-01	4.805000
15	7	5	3	1.7491477D-04	5.2311155D-02	9.5944261D-01	1.1840373D-13	4.0597492D-02	-1.2924805D+01	-1.4914956D+00	5.445000
15	7	5	3	2.2704690D-05	2.1436397D-02	9.9630564D-01	7.5601063D-18	3.6941611D-03	-1.7121472D+01	-2.4324842D+00	6.845000
15	7	5	3	2.3545439D-06	7.7591527D-03	9.9924189D-01	1.3886422D-22	1.9820007D-04	-2.1857410D+01	-3.7028962D+00	8.405000

FOR GAMMA = .250000

15	7	5	3	2.5859375D-01	4.0625000D-01	4.6000022D-16	2.4278548D-02	9.7572145D-01	-1.6147773D+00	-1.0674147D-02	0.000000
15	7	5	3	1.3284565D-01	5.6842145D-01	6.4926424D-05	4.4071060D-04	9.9935150D-01	-3.3554665D+00	-2.1068393D-04	.845000
15	7	5	3	5.7543921D-02	5.1159796D-01	1.2525332D-03	2.4325171D-04	9.9850422D-01	-3.6139441D+00	-6.5009745D-04	1.280000
15	7	5	3	1.6774430D-03	2.8701147D-01	1.4710241D-01	9.5935640D-07	8.5080963D-01	-6.0180200D+00	-7.0167605D-02	2.645000
15	7	5	3	2.3612976D-03	1.9443336D-01	4.1893043D-01	1.2426512D-08	5.8106955D-01	-7.9056507D+00	-2.3577188D-01	3.380000
15	7	5	3	2.4238465D-04	8.5603995D-02	8.9684501D-01	8.4889843D-13	1.3155387D-01	-1.2071100D+01	-8.8069636D-01	4.805000
15	7	5	3	8.7825546D-05	3.8278567D-02	9.4682056D-01	8.8448668D-15	5.3179437D-02	-1.4053309D+01	-1.2742563D+00	5.445000
15	7	5	3	9.7023363D-06	2.4867235D-02	9.9452993D-01	3.2408824D-18	5.4700676D-03	-1.8484343D+01	-2.2620073D+00	6.845000
15	7	5	3	8.5766772D-07	9.3225085D-03	9.9960105D-01	3.2492078D-24	3.3845230D-04	-2.3488223D+01	-3.4698614D+00	8.405000

FOR GAMMA = .300000

15	7	5	3	2.5859375D-01	4.0625000D-01	4.6000022D-16	2.4278548D-02	9.7572145D-01	-1.6147773D+00	-1.0674147D-02	0.000000
15	7	5	3	1.1662178D-01	5.9313327D-01	2.6195991D-05	1.7690228D-04	9.9979690D-01	-3.7522666D+00	-8.8213418D-05	.845000
15	7	5	3	2.5645308D-02	5.3841843D-01	7.7276302D-04	8.4565377D-05	9.9914257D-01	-4.0728074D+00	-3.7249269D-04	1.280000
15	7	5	3	3.2855706D-03	3.0626339D-01	1.1602154D-01	1.9524560D-07	8.8397826D-01	-6.7094187D+00	-5.3558414D-02	2.645000
15	7	5	3	1.4819492D-03	2.1089036D-01	3.5821190D-01	1.8917277D-09	6.4178810D-01	-8.7231414D+00	-1.9260834D-01	3.380000
15	7	5	3	1.2779825D-04	9.7208144D-02	8.2659840D-01	7.5385287D-14	1.7340160D-01	-1.0312271D+01	-7.6094690D-01	4.805000
15	7	5	3	4.2864726D-05	9.7046871D-02	9.2270776D-01	6.2225578D-16	7.7092237D-02	-1.5206021D+01	-1.1298940D+00	5.445000
15	7	5	3	4.9166512D-06	3.0710747D-02	9.9001102D-01	1.3921775D-20	9.9889825D-03	-1.9856305D+01	-2.0004787D+00	6.845000
15	7	5	3	2.9560540D-07	1.2676933D-02	9.9917441D-01	8.2133077D-26	8.2558587D-04	-2.5085482D+01	-3.0832377D+00	8.405000

EXT SET

7-43

NOT REPRODUCIBLE

15	7	7	3	5.56D-01	4.06D-01	4.41D-14	2.42D-02	9.75721450-01	-1.61477730+00	-1.06741470-02	0.000000
15	7	7	3	2.29D-01	3.73D-01	3.75D-03	3.78D-02	9.58374090-01	-1.42169260+00	-1.84649380-02	.845000
15	7	7	3	1.51D-01	3.18D-01	4.17D-02	5.14D-02	9.06795210-01	-1.28820620+00	-4.24907850-02	1.280000
15	7	7	3	3.78D-02	1.60D-01	5.47D-01	6.04D-03	4.46D-01	-2.21848640+00	-3.50340280-01	2.645000
15	7	7	3	1.77D-02	1.04D-01	7.96D-01	5.99D-04	2.02489740-01	-3.22186760+00	-6.93596980-01	3.380000
15	7	7	3	4.04D-03	4.28D-02	9.75D-01	2.61D-06	2.42D-02	-5.58299220+00	-1.61520260+00	4.805000
15	7	7	3	2.08D-03	2.82D-02	9.92D-01	1.85D-07	7.95D-03	-6.73159420+00	-2.09914320+00	5.445000
15	7	7	3	4.92D-04	1.14D-02	9.99D-01	4.81D-10	5.81D-04	-9.31757850+00	-3.23517170+00	6.845000
15	7	7	3	7.97D-05	3.97D-03	9.99D-01	5.86D-13	2.75D-05	-1.22314820+01	-4.56018240+00	8.405000
						FOR GAMMA =	.150000				
15	7	7	3	5.58D-01	4.06D-01	4.41D-14	2.42D-02	9.75721450-01	-1.61477730+00	-1.06741470-02	0.000000
15	7	7	3	1.89D-01	4.98D-01	4.82D-04	3.23D-03	9.96284060-01	-2.49039200+00	-1.61681770-03	.845000
15	7	7	3	9.62D-02	4.49D-01	6.31D-03	4.00D-03	9.89673650-01	-2.39712140+00	-4.50799130-03	1.280000
15	7	7	3	1.53D-02	2.50D-01	2.34D-01	3.39D-04	7.64777000-01	-3.46866000+00	-1.16465180-01	2.645000
15	7	7	3	5.63D-03	1.67D-01	5.26D-01	2.34D-05	4.73118580-01	-4.62987460+00	-3.25029990-01	3.380000
15	7	7	3	8.07D-04	7.19D-02	9.11D-01	3.89D-08	8.81766610-02	-7.40932090+00	-1.05464640+00	4.805000
15	7	7	3	3.38D-04	4.81D-02	9.67D-01	1.66D-09	3.29129390-02	-8.77777600+00	-1.48263330+00	5.445000
15	7	7	3	5.14D-05	1.99D-02	9.97D-01	1.29D-12	2.88033340-03	-1.18862940+01	-2.54055720+00	6.845000
15	7	7	3	6.43D-06	7.06D-03	9.99D-01	3.74D-16	1.50303060-04	-1.54263270+01	-3.82303220+00	8.405000
						FOR GAMMA =	.200000				
15	7	7	3	5.58D-01	4.06D-01	4.41D-14	2.42D-02	9.75721450-01	-1.61477730+00	-1.06741470-02	0.000000
15	7	7	3	1.56D-01	5.32D-01	2.01D-04	1.40D-03	9.98332700-01	-2.85207250+00	-7.24703770-04	.845000
15	7	7	3	8.10D-02	4.82D-01	3.60D-03	1.68D-03	9.94710770-01	-2.77292190+00	-2.30317970-03	1.280000
15	7	7	3	1.09D-02	2.69D-01	1.88D-01	1.13D-04	8.11612520-01	-3.94327130+00	-9.06512600-02	2.645000
15	7	7	3	3.68D-03	1.80D-01	4.73D-01	6.55D-06	5.26963310-01	-5.18318740+00	-2.78219620-01	3.380000
15	7	7	3	4.48D-04	7.78D-02	8.93D-01	7.29D-09	1.06089400-01	-8.13690030+00	-9.74328010-01	4.805000
15	7	7	3	1.64D-04	5.23D-02	9.59D-01	2.57D-10	4.05964680-02	-9.58868200+00	-1.39151180+00	5.445000
15	7	7	3	2.27D-05	2.14D-02	9.96D-01	1.30D-13	3.69414790-03	-1.28847850+01	-2.43248570+00	6.845000
15	7	7	3	2.39D-06	7.75D-03	9.99D-01	2.30D-17	1.98200010-04	-1.66379210+01	-3.70289630+00	8.405000
						FOR GAMMA =	.250000				
15	7	7	3	5.56D-01	4.06D-01	4.41D-14	2.42D-02	9.75721450-01	-1.61477730+00	-1.06741470-02	0.000000
15	7	7	3	1.32D-01	5.64D-01	1.43D-04	6.04D-04	9.99251640-01	-3.21853040+00	-3.25129230-04	.845000
15	7	7	3	6.75D-02	5.11D-01	2.07D-03	6.95D-04	9.97231880-01	-3.15775440+00	-1.20384820-03	1.280000
15	7	7	3	7.67D-03	2.87D-01	1.50D-01	3.56D-05	8.49567260-01	-4.44820880+00	-7.08022300-02	2.645000
15	7	7	3	2.36D-03	1.94D-01	4.19D-01	1.69D-06	5.80805790-01	-5.77038010+00	-2.35969070-01	3.380000
15	7	7	3	2.42D-04	8.56D-02	8.68D-01	1.28D-09	1.31550220-01	-8.89225160+00	-8.80908430-01	4.805000
15	7	7	3	8.78D-05	5.82D-02	9.46D-01	3.79D-11	5.31790280-02	-1.04208360+01	-1.27425960+00	5.445000
15	7	7	3	7.79D-06	2.46D-02	9.94D-01	1.30D-14	5.47D-04	-1.38851000+01	-2.26200750+00	6.845000
15	7	7	3	8.57D-07	9.32D-03	9.99D-01	1.50D-18	3.38952290-04	-1.78235850+01	-3.46986140+00	8.405000
						FOR GAMMA =	.300000				
15	7	7	3	5.56D-01	4.06D-01	4.41D-14	2.42D-02	9.75721450-01	-1.61477730+00	-1.06741470-02	0.000000
15	7	7	3	1.16D-01	5.93D-01	7.70D-05	2.55D-04	9.99665370-01	-3.59253070+00	-1.45353810-04	.845000
15	7	7	3	5.50D-02	5.38D-01	1.18D-03	2.76D-04	9.98540640-01	-3.55771830+00	-6.34254360-04	1.280000
15	7	7	3	5.26D-03	3.06D-01	1.16D-01	1.02D-05	8.83481650-01	-4.98940330+00	-5.38024650-02	2.645000
15	7	7	3	1.40D-03	2.10D-01	3.50D-01	4.03D-07	6.41693710-01	-6.39402670+00	-1.92672220-01	3.380000
15	7	7	3	1.27D-04	9.72D-02	8.26D-01	2.13D-10	1.73400550-01	-9.67152320+00	-7.60949520-01	4.805000
15	7	7	3	4.26D-05	6.79D-02	9.22D-01	5.40D-12	7.70921310-02	-1.12668220+01	-1.11298990+00	5.445000
15	7	7	3	4.01D-06	3.07D-02	9.90D-01	1.34D-15	9.98698200-03	-1.48716410+01	-2.00047880+00	6.845000
15	7	7	3	4.95D-07	1.28D-02	9.99D-01	1.09D-19	8.25585870-04	-1.89569430+01	-3.06323780+00	8.405000

EXT SET

FOR GAMMA = 0.000000

15	7	1	5	5.88593750D-01	4.06250000D-01	3.88088643D-15	2.03054620D-01	7.96945380D-01	-6.92387130D-01	-9.85714410D-02	0.000000
15	7	1	5	2.29578940D-01	3.73375140D-01	1.49581320D-03	2.11158790D-01	7.87345390D-01	-6.75390830D-01	-1.03834710D-01	.845000
15	7	1	5	1.50162380D-01	3.18103210D-01	2.45849450D-02	1.43424560D-01	8.31990490D-01	-8.43376470D-01	-7.98816350D-02	1.280000
15	7	1	5	3.78212490D-02	1.60685960D-01	5.10140010D-01	1.92750950D-03	4.87932480D-01	-2.71500350D+00	-3.11640270D-01	2.645000
15	7	1	5	1.77315740D-02	1.04518140D-01	7.52097530D-01	6.52336850D-05	2.47837240D-01	-4.18552810D+00	-6.05833440D-01	3.380000
15	7	1	5	4.04962300D-03	4.28065320D-02	9.40059430D-01	3.79528690D-08	5.93405300D-02	-7.42075540D+00	-1.22664860D+00	4.805000
15	7	1	5	2.08697200D-03	2.82602400D-02	9.69106850D-01	1.10392720D-09	3.08931470D-02	-8.95705960D+00	-1.51013790D+00	5.445000
15	7	1	5	4.92340300D-04	1.12265580D-02	9.92039810D-01	4.06442610D-13	7.36019190D-03	-1.23910010D+01	-2.13311090D+00	6.845000
15	7	1	5	9.97599860D-05	3.97295470D-03	9.98504660D-01	5.53058450D-17	1.49534470D-03	-1.62572320D+01	-2.82525870D+00	8.405000

FOR GAMMA = .150000

15	7	1	5	5.88593750D-01	4.06250000D-01	3.88088643D-15	2.03054620D-01	7.96945380D-01	-6.92387130D-01	-9.85714410D-02	0.000000
15	7	1	5	1.69441540D-01	4.98882810D-01	5.60884900D-04	4.43056970D-02	9.56125450D-01	-1.36345500D+00	-1.94851200D-02	.845000
15	7	1	5	7.62245470D-02	4.49796010D-01	1.34577450D-02	2.85832400D-02	9.57959020D-01	-1.54388850D+00	-1.86530710D-02	1.280000
15	7	1	5	1.53306630D-02	2.50256990D-01	5.33367220D-01	2.99814710D-04	4.66332960D-01	-3.52314710D+00	-3.31303890D-01	2.645000
15	7	1	5	5.63379050D-03	1.67920360D-01	8.32433100D-01	7.30572320D-06	1.67559600D-01	-5.13633680D+00	-7.75830690D-01	3.380000
15	7	1	5	8.07543020D-04	7.18713430D-02	9.84859650D-01	1.75314540D-09	1.51403490D-02	-8.75618210D+00	-1.81986410D+00	4.805000
15	7	1	5	3.38951220D-04	4.81714630D-02	9.94411320D-01	3.19732580D-11	5.58868130D-03	-1.04952130D+01	-2.25269070D+00	5.445000
15	7	1	5	5.14333150D-05	1.96225770D-02	9.99221370D-01	3.88617780D-15	7.78629340D-04	-1.44104770D+01	-3.10866920D+00	6.845000
15	7	1	5	8.43909870D-06	7.00086560D-03	9.99903370D-01	1.41346610D-19	9.66318100D-05	-1.88497160D+01	-4.01487990D+00	8.405000

FOR GAMMA = .200000

15	7	1	5	5.88593750D-01	4.06250000D-01	3.88088643D-15	2.03054620D-01	7.96945380D-01	-6.92387130D-01	-9.85714410D-02	0.000000
15	7	1	5	1.50052520D-01	5.32964050D-01	4.29780010D-04	2.37925510D-02	9.75777660D-01	-1.62355900D+00	-1.06491280D-02	.845000
15	7	1	5	8.10758310D-02	4.82413490D-01	1.11300640D-02	1.49327650D-02	9.73937210D-01	-1.82586090D+00	-1.14690410D-02	1.280000
15	7	1	5	1.09549260D-02	2.69069130D-01	5.18758580D-01	1.20332960D-04	4.81121090D-01	-3.91961540D+00	-3.17745610D-01	2.645000
15	7	1	5	3.68594000D-03	1.80566270D-01	8.32551210D-01	2.42917420D-06	1.67446360D-01	-5.61454130D+00	-7.76124280D-01	3.380000
15	7	1	5	4.48101670D-04	7.78308730D-02	9.88906220D-01	3.90000802D-10	1.10937800D-02	-9.40892650D+00	-1.95492040D+00	4.805000
15	7	1	5	1.74430970D-04	5.23111500D-02	9.96625280D-01	5.88696880D-12	3.37471680D-03	-1.12301080D+01	-2.47176270D+00	5.445000
15	7	1	5	2.27048900D-05	2.14363870D-02	9.99648130D-01	4.68482420D-16	3.51872410D-04	-1.53293070D+01	-3.45361480D+00	6.845000
15	7	1	5	2.39494390D-06	7.75915240D-03	9.99956000D-01	1.05396080D-20	3.60027100D-05	-1.99771760D+01	-4.44366680D+00	8.405000

FOR GAMMA = .250000

15	7	1	5	5.88593750D-01	4.06250000D-01	3.88088643D-15	2.03054620D-01	7.96945380D-01	-6.92387130D-01	-9.85714410D-02	0.000000
15	7	1	5	1.32685950D-01	5.64214530D-01	3.27981890D-04	1.26844950D-02	9.86987020D-01	-1.89670970D+00	-5.68855730D-03	.845000
15	7	1	5	6.75439210D-02	5.11597900D-01	9.14280010D-03	7.45418640D-03	9.83403010D-01	-2.12759980D+00	-7.26846870D-03	1.280000
15	7	1	5	7.67179430D-03	2.87014730D-01	4.92220060D-01	4.44728020D-05	5.07674890D-01	-4.35190550D+00	-2.94414320D-01	2.645000
15	7	1	5	2.36129790D-03	1.94433300D-01	8.19115000D-01	7.47473300D-07	1.80884250D-01	-6.12640430D+00	-7.42599240D-01	3.380000
15	7	1	5	2.42384550D-04	8.56039850D-02	9.89749690D-01	8.42794550D-11	1.02503110D-02	-1.00742780D+01	-1.98926290D+00	4.805000
15	7	1	5	8.78255460D-05	5.82785760D-02	9.97453210D-01	1.08969770D-12	2.54679280D-03	-1.19626940D+01	-2.59400640D+00	5.445000
15	7	1	5	9.70553630D-06	2.46723500D-02	9.99863211D-01	6.24980850D-17	1.67887840D-04	-1.62041340D+01	-3.77498080D+00	6.845000
15	7	1	5	8.57866770D-07	9.32250850D-03	9.99968940D-01	9.96562800D-22	1.30635430D-05	-2.10014950D+01	-4.88393900D+00	8.405000

FOR GAMMA = .300000

15	7	1	5	5.88593750D-01	4.06250000D-01	3.88088643D-15	2.03054620D-01	7.96945380D-01	-6.92387130D-01	-9.85714410D-02	0.000000
15	7	1	5	1.16241780D-01	5.93133270D-01	2.49970223D-04	6.55115900D-03	9.93198940D-01	-2.18368190D+00	-2.96375320D-03	.845000
15	7	1	5	5.55495890D-02	5.38418430D-01	7.35512600D-04	3.54164440D-03	9.89103240D-01	-2.45079510D+00	-4.75837740D-03	1.280000
15	7	1	5	5.28857060D-03	3.06263370D-01	4.51172370D-01	1.52556590D-05	5.46812370D-01	-4.81656900D+00	-2.60576110D-01	2.645000
15	7	1	5	1.48184920D-03	2.10889030D-01	7.80794002D-01	2.17025460D-07	2.12003350D-01	-6.66348930D+00	-6.73595520D-01	3.380000
15	7	1	5	1.27778420D-04	9.72081440D-02	9.86783460D-01	1.85312690D-11	1.32165360D-02	-1.07320950D+01	-1.87888230D+00	4.805000
15	7	1	5	9.28897260D-05	8.79048910D-02	9.90972470D-01	2.15336110D-13	3.07509050D-03	-1.26668830D+01	-2.51214210D+00	5.445000
15	7	1	5	4.61065170D-06	3.07107470D-02	9.99670400D-01	1.01168120D-17	1.23600370D-04	-1.69949560D+01	-3.90796760D+00	6.845000
15	7	1	5	6.95006540D-07	1.26700330D-02	9.99949090D-01	1.35743100D-22	5.31985120D-06	-2.18672820D+01	-5.27410050D+00	8.405000

XT SET

FOR GAMMA = 0.000000

15	7	3	5	3.58953750D-01	4.06250000D-01	3.88092120D-15	2.03054620D-01	7.96945380D-01	-6.92387130D-01	-9.85714410D-02	0.000000
15	7	3	5	2.24504940D-01	3.73375140D-01	1.50416410D-03	2.11338460D-01	7.87157380D-01	-6.75021470D-01	-1.03938430D-01	.845000
15	7	3	5	1.50124380D-01	3.18103210D-01	2.47110360D-02	1.43619480D-01	8.31470050D-01	-8.42786950D-01	-8.01533870D-02	1.280000
15	7	3	5	3.78612490D-02	1.60688960D-01	5.35750730D-01	1.92950900D-03	4.62319760D-01	-2.71455320D+00	-3.35057540D-01	2.645000
15	7	3	5	1.77315740D-02	1.04518190D-01	7.94925530D-01	6.52817940D-05	2.04979190D-01	-4.18520790D+00	-6.88290230D-01	3.380000
15	7	3	5	4.04962300D-03	4.28065320D-02	9.71605310D-01	3.79672010D-08	2.83146490D-02	-7.42059140D+00	-1.54798880D+00	4.805000
15	7	3	5	2.08697200D-03	2.82602400D-02	9.89944140D-01	1.10424050D-09	1.05585580D-02	-8.95693630D+00	-1.97639540D+00	5.445000
15	7	3	5	4.92340300D-04	1.12265500D-02	9.98901380D-01	4.06506720D-13	1.09661800D-03	-1.23909320D+01	-2.95915330D+00	6.845000
15	7	3	5	2.97559080D-05	3.97259470D-03	9.99911940D-01	5.53102050D-17	8.20570370D-05	-1.62571950D+01	-4.08588420D+00	8.405000

FOR GAMMA = .150000

15	7	3	5	3.58953750D-01	4.06250000D-01	3.88092120D-15	2.03054620D-01	7.96945380D-01	-6.92387130D-01	-9.85714410D-02	0.000000
15	7	3	5	1.89441540D-01	4.98682810D-01	5.69344570D-04	4.33114510D-02	9.56119200D-01	-1.36339730D+00	-1.94879590D-02	.845000
15	7	3	5	9.62245470D-02	4.49796010D-01	1.34805420D-02	2.85864940D-02	9.57932960D-01	-1.54383910D+00	-1.86648820D-02	1.280000
15	7	3	5	1.53915830D-02	2.50258990D-01	5.36423520D-01	2.99818780D-04	4.63276660D-01	-3.52314120D+00	-3.34159580D-01	2.645000
15	7	3	5	3.63379050D-03	1.67920360D-01	8.38293040D-01	7.30575200D-06	1.61699650D-01	-5.13633510D+00	-7.91290920D-01	3.380000
15	7	3	5	8.07543020D-04	7.18713430D-02	9.89071560D-01	1.75314610D-09	1.09284390D-02	-8.75618190D+00	-1.96144190D+00	4.805000
15	7	3	5	3.38951220D-04	4.81714630D-02	9.96945780D-01	3.19732630D-11	3.05422380D-03	-1.04952130D+01	-2.51509910D+00	5.445000
15	7	3	5	2.14333150D-05	1.96225770D-02	9.99800550D-01	3.88817790D-15	1.94481190D-04	-1.44104770D+01	-3.71112240D+00	6.845000
15	7	3	5	6.43599870D-06	7.06086560D-03	9.99990830D-01	1.41346100D-19	9.17498620D-06	-1.88497160D+01	-5.03739460D+00	8.405000

FOR GAMMA = .200000

15	7	3	5	3.58953750D-01	4.06250000D-01	3.88092120D-15	2.03054620D-01	7.96945380D-01	-6.92387130D-01	-9.85714410D-02	0.000000
15	7	3	5	1.30052520D-01	5.32964030D-01	4.30013550D-04	2.37944090D-02	9.75775580D-01	-1.62352510D+00	-1.06500560D-02	.845000
15	7	3	5	9.10758310D-02	4.82413400D-01	1.11411540D-02	1.49335430D-02	9.73925300D-01	-1.82583710D+00	-1.14743510D-02	1.280000
15	7	3	5	1.09549260D-02	2.69069150D-01	5.20411120D-01	1.20333410D-04	4.79468540D-01	-3.91961380D+00	-3.19239880D-01	2.645000
15	7	3	5	3.68546000D-03	1.80862750D-01	8.35729250D-01	2.42917620D-06	1.64268320D-01	-5.01454100D+00	-7.84446170D-01	3.380000
15	7	3	5	4.48101670D-04	7.78308730D-02	9.91053360D-01	3.90008040D-10	8.94663860D-03	-9.40892640D+00	-2.04834010D+00	4.805000
15	7	3	5	1.74930970D-04	5.23111550D-02	9.97258830D-01	5.88696890D-12	2.14116810D-03	-1.12301080D+01	-2.66934920D+00	5.445000
15	7	3	5	2.27048900D-05	2.14363870D-02	9.99869500D-01	4.68482420D-16	1.00500650D-04	-1.53293070D+01	-3.99783110D+00	6.845000
15	7	3	5	2.39494390D-06	7.75915290D-03	9.99996210D-01	1.05396080D-20	3.79117970D-06	-1.99771760D+01	-5.42122560D+00	8.405000

FOR GAMMA = .250000

15	7	3	5	3.58953750D-01	4.06250000D-01	3.88092120D-15	2.03054620D-01	7.96945380D-01	-6.92387130D-01	-9.85714410D-02	0.000000
15	7	3	5	1.32458950D-01	5.64214550D-01	3.28091090D-04	1.26855860D-02	9.86986320D-01	-1.89668950D+00	-5.68886550D-03	.845000
15	7	3	5	6.75439210D-02	5.11597980D-01	9.14834890D-03	7.45438320D-03	9.83397270D-01	-2.12758830D+00	-7.27100260D-03	1.280000
15	7	3	5	1.67797440D-03	2.87014740D-01	4.93148870D-01	4.44728470D-05	5.06806660D-01	-4.35190510D+00	-2.95157690D-01	2.645000
15	7	3	5	2.30129740D-03	1.94433360D-01	8.20762460D-01	7.47473420D-07	1.79236790D-01	-6.12640420D+00	-7.46572830D-01	3.380000
15	7	3	5	2.42384650D-04	8.56039890D-02	9.90784510D-01	8.42794500D-11	9.21549100D-03	-1.00074270D+01	-2.03548150D+00	4.805000
15	7	3	5	8.78255460D-05	5.82705760D-02	9.98020830D-01	1.08969770D-12	1.97917060D-03	-1.19626940D+01	-2.70351680D+00	5.445000
15	7	3	5	7.70253630D-06	2.46723500D-02	9.99934710D-01	6.24980500D-17	6.52653700D-05	-1.62041340D+01	-4.18518410D+00	6.845000
15	7	3	5	8.57867720D-07	9.32250850D-03	9.99998220D-01	9.96562800D-22	1.77706450D-06	-2.10014950D+01	-5.75029680D+00	8.405000

FOR GAMMA = .300000

15	7	3	5	3.58953750D-01	4.06250000D-01	3.88092120D-15	2.03054620D-01	7.96945380D-01	-6.92387130D-01	-9.85714410D-02	0.000000
15	7	3	5	1.16271780D-01	5.93133270D-01	2.49955750D-04	6.55134160D-03	9.93198700D-01	-2.18366980D+00	-2.96385640D-03	.845000
15	7	3	5	5.56453080D-02	5.38418490D-01	1.35767380D-03	3.54168680D-03	9.89100440D-01	-2.45078960D+00	-4.75960610D-03	1.280000
15	7	3	5	2.45357900D-03	3.06260330D-01	4.51398370D-01	1.52556630D-05	5.48386370D-01	-4.81656890D+00	-2.60913340D-01	2.645000
15	7	3	5	1.48104920D-03	2.10690360D-01	7.88750220D-01	2.17025470D-07	2.11247560D-01	-6.66348930D+00	-6.75208300D-01	3.380000
15	7	3	5	1.27798230D-04	9.72081440D-02	9.87240550D-01	1.85312690D-11	1.27594670D-02	-1.07320950D+01	-1.89416750D+00	4.805000
15	7	3	5	9.20697200D-05	6.79088910D-02	9.97165130D-01	2.15336110D-13	2.83487120D-03	-1.26668830D+01	-2.54746670D+00	5.445000
15	7	3	5	4.01005120D-06	3.07107470D-02	9.99915330D-01	1.01168120D-17	8.46745640D-05	-1.69949560D+01	-4.07224700D+00	6.845000
15	7	3	5	2.95005460D-07	1.26700330D-02	9.99998490D-01	1.35743100D-22	1.60743890D-06	-2.18672820D+01	-5.79386550D+00	8.405000

XT SET

FOR GAMMA = 0.000000

15	7	5	5	5.58893750-01	4.00625000-01	4.33958580-15	2.03054620-01	7.96945380-01	-6.92387130-01	-9.85714410-02	0.000000
15	7	5	5	2.25[illegible]4940-01	3.73375140-01	2.31965700-03	2.20411150-01	7.77269200-01	-6.56766440-01	-1.09248540-01	.845000
15	7	5	5	1.50[illegible]360-01	3.18103210-01	3.88641140-02	1.59631420-01	8.01484570-01	-7.96881910-01	-9.61048330-02	1.280000
15	7	5	5	3.78212490-02	1.60086960-01	7.69767940-01	2.54726790-03	2.87684790-01	-2.59392540+00	-5.41063090-01	2.645000
15	7	5	5	1.77[illegible]740-02	1.04518170-01	9.27701220-01	9.34851880-05	7.21452980-02	-4.02925720+00	-1.14179200+00	3.380000
15	7	5	5	4.04962300-03	4.28006530-02	9.97610320-01	6.49216950-08	2.18961650-03	-7.18761010+00	-2.65963190+00	4.805000
15	7	5	5	2.[illegible]4[illegible]00-03	2.82[illegible]400-02	9.99[illegible]400-01	2.07896400-09	3.83363990-04	-8.68215300+00	-3.41638870+00	5.445000
15	7	5	5	4.92340300-04	1.12265500-02	9.99992560-01	9.92495450-13	7.44082920-06	-1.20032710+01	-5.12837870+00	6.845000
15	7	5	5	2.97[illegible]0[illegible]60-05	3.97[illegible]540-03	9.99999910-01	1.98060870-16	8.89619010-08	-1.57032010+01	-7.05069830+00	8.405000

FOR GAMMA = .150000

15	7	5	5	5.58893750-01	4.00625000-01	4.33958580-15	2.03054620-01	7.96945380-01	-6.92387130-01	-9.85714410-02	0.000000
15	7	5	5	1.[illegible]9441540-01	4.98882810-01	6.83578710-04	4.41680110-02	9.55148410-01	-1.35489220+00	-1.99291430-02	.845000
15	7	5	5	9.62245470-02	4.49796010-01	1.56670860-02	2.97535610-02	9.54379350-01	-1.52646100+00	-2.02789650-02	1.280000
15	7	5	5	1.[illegible]0[illegible]40-02	2.50[illegible]8900-01	5.83748770-01	3.16556510-04	4.15634670-01	-3.49954880+00	-3.80974880-01	2.645000
15	7	5	5	5.63374050-03	1.67920360-01	8.77256420-01	7.69699390-06	1.22735890-01	-5.11367890+00	-9.11028430-01	3.380000
15	7	5	5	3.07[illegible]4[illegible]20-04	7.18713430-02	9.95569760-01	1.83763280-09	4.03023960-03	-8.73574130+00	-2.39466910+00	4.805000
15	7	5	5	3.38[illegible]1220-04	4.81714630-02	9.95339630-01	3.34555770-11	6.60370820-04	-1.04755340+01	-3.18021210+00	5.445000
15	7	5	5	3.[illegible]3[illegible]30-05	1.96225770-02	9.99[illegible]30-01	4.05641010-15	9.27337990-06	-1.43918580+01	-5.03276190+00	6.845000
15	7	5	5	8.43[illegible]9870-06	7.06086560-03	9.99999940-01	1.47415830-19	6.17406560-08	-1.88314560+01	-7.20942880+00	8.405000

FOR GAMMA = .200000

15	7	5	5	5.58893750-01	4.00625000-01	4.33958580-15	2.03054620-01	7.96945380-01	-6.92387130-01	-9.85714410-02	0.000000
15	7	5	5	1.50052520-01	5.32964050-01	4.94896620-04	2.41851180-02	9.75319990-01	-1.61645180+00	-1.08528760-02	.845000
15	7	5	5	8.11[illegible]310-02	4.82413400-01	1.25596950-02	1.53997500-02	9.72040560-01	-1.61248630+00	-1.23156150-02	1.280000
15	7	5	5	1.09549260-02	2.69069150-01	5.50236560-01	1.24486750-04	4.49638950-01	-3.90487690+00	-3.47136070-01	2.645000
15	7	5	5	3.88546000-03	1.80862750-01	8.59924620-01	2.50226500-06	1.40072880-01	-5.60166670+00	-8.53645950-01	3.380000
15	7	5	5	4.48101670-04	7.78308730-02	9.94492140-01	3.98865750-10	5.00786290-03	-9.39918310+00	-2.30034760+00	4.805000
15	7	5	5	1.74933470-04	5.23111550-02	9.99[illegible]010-01	6.00583310-12	8.43985760-04	-1.12214270+01	-3.07366490+00	5.445000
15	7	5	5	2.27048900-05	2.14363870-02	9.99967610-01	4.76042210-16	1.23688490-05	-1.53223550+01	-4.90696550+00	6.845000
15	7	5	5	2.39[illegible]390-06	7.75915270-03	9.99[illegible]920-01	1.06784720-20	8.49402340-08	-1.99714910+01	-7.07088650+00	8.405000

FOR GAMMA = .250000

15	7	5	5	5.58893750-01	4.00625000-01	4.33958580-15	2.03054620-01	7.96945380-01	-6.92387130-01	-9.85714410-02	0.000000
15	7	5	5	1.32[illegible]850-01	5.64214550-01	3.65745080-04	1.28614100-02	9.86772850-01	-1.89071140+00	-5.78281040-03	.845000
15	7	5	5	6.75[illegible]2100-02	5.11597980-01	9.99395420-03	7.63346760-03	9.82372580-01	-2.11727810+00	-7.72376890-03	1.280000
15	7	5	5	1.67[illegible]0[illegible]30-03	2.87014730-01	5.11245080-01	4.54135370-05	4.88709530-01	-4.34281470+00	-3.10949190-01	2.645000
15	7	5	5	2.36124790-03	1.94433360-01	8.35164650-01	7.59834330-07	1.64833590-01	-6.11928110+00	-7.82951650-01	3.380000
15	7	5	5	2.42[illegible]4650-04	8.56039870-02	9.92968840-01	8.51280690-11	7.03115620-03	-1.00699270+01	-2.15297330+00	4.805000
15	7	5	5	8.78[illegible]0[illegible]05	5.82786760-02	9.98710890-01	1.09854110-12	1.28911220-03	-1.19591840+01	-2.88970930+00	5.445000
15	7	5	5	9.70[illegible]3630-06	2.46723500-02	9.99477030-01	6.28221870-17	2.29662730-05	-1.62018870+01	-4.63890950+00	6.845000
15	7	5	5	8.57867720-07	9.32256850-03	9.99999800-01	9.99812690-22	1.99127090-07	-2.10000820+01	-6.70086970+00	8.405000

FOR GAMMA = .300000

15	7	5	5	5.58893750-01	4.00625000-01	4.33958580-15	2.03054620-01	7.96945380-01	-6.92387130-01	-9.85714410-02	0.000000
15	7	5	5	1.16221780-01	5.93133270-01	2.71914900-04	6.62852610-03	9.93099560-01	-2.17858300+00	-3.00721100-03	.845000
15	7	5	5	3.56[illegible]070-02	5.38918430-01	7.85250710-03	3.60692010-03	9.88540570-01	-2.44286350+00	-5.00550120-03	1.280000
15	7	5	5	2.28[illegible]0[illegible]03	3.06263390-01	4.61946070-01	1.54482230-05	5.38038480-01	-4.81112150+00	-2.69186670-01	2.645000
15	7	5	5	1.48[illegible]920-03	2.10890360-01	7.96846180-01	2.18910800-07	2.03163600-01	-6.65973280+00	-6.92154090-01	3.380000
15	7	5	5	1.27798250-04	9.72081440-02	9.88412460-01	1.86066350-11	1.15875410-02	-1.07303320+01	-1.93600870+00	4.805000
15	7	5	5	4.28[illegible]720-05	6.79048910-02	9.97526250-01	2.15958310-13	2.47376430-03	-1.26656300+01	-2.60666480+00	5.445000
15	7	5	5	4.01[illegible]5120-06	3.07107470-02	9.99946150-01	1.01307330-17	6.38490480-05	-1.69943590+01	-4.19484560+00	6.845000
15	7	5	5	2.95[illegible]0[illegible]80-07	1.26700330-02	9.99994120-01	1.35825440-22	8.84638610-07	-2.18670190+01	-6.05313590+00	8.405000

FOR GAMMA = 0.000000

15	7	7	5	3.5859375D-01	4.0625000D-01	7.2764025D-13	2.0305462D-01	7.9694538D-01	-6.9238713D-01	-9.8571441D-02	0.000000
15	7	7	5	2.2950894D-01	3.7337514D-01	1.4205711D-02	2.6135971D-01	7.2443458D-01	-5.8276137D-01	-1.4000083D-01	.845000
15	7	7	5	1.5812380D-01	3.1810321D-01	1.2974213D-01	2.7925765D-01	5.9100002D-01	-5.5399461D-01	-2.2841250D-01	1.280000
15	7	7	5	3.7821249D-02	1.6068596D-01	8.7600485D-01	2.0759711D-02	1.0263544D-01	-1.6827787D+00	-9.8870264D-01	2.645000
15	7	7	5	1.7731574D-02	1.0451819D-01	9.7933941D-01	1.7324294D-03	1.8928156D-02	-2.7613445D+00	-1.7228917D+00	3.380000
15	7	7	5	4.0476230D-03	4.2806532D-02	9.9966057D-01	5.6725798D-06	3.3376014D-04	-5.2462194D+00	-3.4765655D+00	4.805000
15	7	7	5	2.0869720D-03	2.8260240D-02	9.9995374D-01	3.6023274D-07	4.5852166D-05	-6.4434168D+00	-4.3386401D+00	5.445000
15	7	7	5	4.9234030D-04	1.1226550D-02	9.9999950D-01	7.5689227D-10	5.0312770D-07	-9.1209659D+00	-6.2983218D+00	6.845000
15	7	7	5	9.9759060D-05	3.9729547D-03	1.0000000D+00	7.7233425D-13	2.9141736D-09	-1.2112195D+01	-8.5354846D+00	8.405000

FOR GAMMA = .150000

15	7	7	5	3.5859375D-01	4.0625000D-01	7.2764025D-13	2.0305462D-01	7.9694538D-01	-6.9238713D-01	-9.8571441D-02	0.000000
15	7	7	5	1.6944154D-01	4.9888281D-01	4.1878021D-03	5.4281976D-02	9.4153022D-01	-1.2653443D+00	-2.6165735D-02	.845000
15	7	7	5	9.6224547D-02	4.4979601D-01	4.8985174D-02	6.1954355D-02	8.8906047D-01	-1.2079282D+00	-5.1068699D-02	1.280000
15	7	7	5	1.5354983D-02	2.5025899D-01	6.7509999D-01	5.3034370D-03	3.1959658D-01	-2.2754426D+00	-4.9539788D-01	2.645000
15	7	7	5	5.0337905D-03	1.6792030D-01	9.0679315D-01	3.8674622D-04	9.2820107D-02	-3.4125739D+00	-1.0323579D+00	3.380000
15	7	7	5	8.0759392D-04	7.1671343D-02	9.9687365D-01	7.1304314D-07	3.1256342D-03	-6.1468842D+00	-2.5050619D+00	4.805000
15	7	7	5	3.3895122D-04	4.8171463D-02	9.9948144D-01	3.1665538D-08	5.1857550D-04	-7.4994131D+00	-3.2852299D+00	5.445000
15	7	7	5	5.1443315D-05	1.9622577D-02	9.9999259D-01	2.6012357D-11	7.4116901D-06	-1.0584820D+01	-5.1300828D+00	6.845000
15	7	7	5	5.4376987D-06	7.0608656D-03	9.9999995D-01	7.6423032D-15	4.9686536D-08	-1.4116776D+01	-7.3037613D+00	8.405000

FOR GAMMA = .200000

15	7	7	5	3.5859375D-01	4.0625000D-01	7.2764025D-13	2.0305462D-01	7.9694538D-01	-6.9238713D-01	-9.8571441D-02	0.000000
15	7	7	5	1.5065252D-01	5.3296405D-01	2.9110033D-03	3.0534494D-02	9.6655450D-01	-1.5152093D+00	-1.4773652D-02	.845000
15	7	7	5	6.1075831D-02	4.8241340D-01	3.6160006D-02	3.5244683D-02	9.2859531D-01	-1.4529064D+00	-3.2173513D-02	1.280000
15	7	7	5	1.0954926D-02	2.6906915D-01	6.1668587D-01	2.8223241D-03	3.8049181D-01	-2.5493931D+00	-4.1965469D-01	2.645000
15	7	7	5	3.6854600D-03	1.6806275D-01	8.6092216D-01	1.8767791D-04	1.1869016D-01	-3.7265868D+00	-9.2465408D-01	3.380000
15	7	7	5	4.4810107D-04	7.7830873D-02	9.9558407D-01	2.7533958D-07	4.4156521D-03	-6.5601314D+00	-2.3550052D+00	4.805000
15	7	7	5	1.7493097D-04	5.2311155D-02	9.9924457D-01	1.0900445D-08	7.5542020D-04	-7.9625558D+00	-3.1218114D+00	5.445000
15	7	7	5	2.2704890D-05	2.1436387D-02	9.9998864D-01	6.8631628D-12	1.1355922D-05	-1.1163476D+01	-4.9447776D+00	6.845000
15	7	7	5	2.3949439D-06	7.7591529D-03	9.9999992D-01	1.4787109D-15	7.9155032D-08	-1.4830117D+01	-7.1015215D+00	8.405000

FOR GAMMA = .250000

15	7	7	5	3.5859375D-01	4.0625000D-01	7.2764025D-13	2.0305462D-01	7.9694538D-01	-6.9238713D-01	-9.8571441D-02	0.000000
15	7	7	5	1.3205895D-01	5.6421455D-01	2.0374540D-03	1.6815607D-02	9.6114694D-01	-1.7742875D+00	-8.2659466D-03	.845000
15	7	7	5	6.7543921D-02	5.1159798D-01	2.6548147D-02	1.9515053D-02	9.5393680D-01	-1.7096303D+00	-2.0480397D-02	1.280000
15	7	7	5	7.6779743D-03	2.8701473D-01	5.5758219D-01	1.4235628D-03	4.4099425D-01	-2.8466234D+00	-3.5556708D-01	2.645000
15	7	7	5	2.3612979D-03	1.9443336D-01	8.4959590D-01	8.7029555D-05	1.5031707D-01	-4.0603332D+00	-8.2299170D-01	3.380000
15	7	7	5	2.4236465D-04	8.5603989D-02	9.9336146D-01	1.0736218D-07	6.6384348D-03	-6.9691487D+00	-2.1779343D+00	4.805000
15	7	7	5	9.7625546D-05	5.8278576D-02	9.9876862D-01	3.9347045D-09	1.2313798D-03	-8.4050879D+00	-2.9096080D+00	5.445000
15	7	7	5	9.7055363D-06	2.4672350D-02	9.9997768D-01	2.1068185D-12	2.2316928D-05	-1.1676373D+01	-4.6513656D+00	6.845000
15	7	7	5	8.5786772D-07	9.3225085D-03	9.9999980D-01	3.8512782D-16	1.9560621D-07	-1.5414395D+01	-6.7086174D+00	8.405000

FOR GAMMA = .300000

15	7	7	5	3.5859375D-01	4.0625000D-01	7.2764025D-13	2.0305462D-01	7.9694538D-01	-6.9238713D-01	-9.8571441D-02	0.000000
15	7	7	5	1.1622178D-01	5.9313327D-01	1.4194286D-03	9.0500464D-03	9.8953052D-01	-2.0433492D+00	-4.5708042D-03	.845000
15	7	7	5	5.5645089D-02	5.3841843D-01	1.9132414D-02	1.0452219D-02	9.7041537D-01	-1.9807915D+00	-1.3042335D-02	1.280000
15	7	7	5	5.2855700D-03	3.0626339D-01	4.9276436D-01	6.8133905D-04	5.0653430D-01	-3.1666367D+00	-2.9539114D-01	2.645000
15	7	7	5	1.4818492D-03	2.1089030D-01	8.0646535D-01	3.9014623D-05	1.9349564D-01	-4.4087726D+00	-7.1332882D-01	3.380000
15	7	7	5	1.2779825D-04	9.7208144D-02	9.8868282D-01	4.3868265D-08	1.1317133D-02	-7.3578495D+00	-1.9462636D+00	4.805000
15	7	7	5	4.2889726D-05	6.7904891D-02	9.9756695D-01	1.5656584D-09	2.4330512D-03	-8.8053030D+00	-2.6138488D+00	5.445000
15	7	7	5	4.0166512D-06	3.0710747D-02	9.9993664D-01	8.1516810D-13	6.3358515D-05	-1.2088753D+01	-4.1981950D+00	6.845000
15	7	7	5	2.9506548D-07	1.2670033D-02	9.9999912D-01	1.5110700D-16	8.8188079D-07	-1.5820715D+01	-6.0545901D+00	8.405000

FOR GAMMA = 0.000000

15	7	1	1	3.5035437D-01	4.0625000D-01	1.9455320D-12	5.9035097D-01	4.0964903D-01	-2.2888972D-01	-3.8758807D-01	0.000000
15	7	1	7	2.2950894D-01	3.7337514D-01	6.9519593D-03	5.9881932D-01	3.9422872D-01	-2.2270419D-01	-4.0425174D-01	.845000
15	7	1	7	1.2016439D-01	3.1810321D-01	5.9906036D-02	4.3608316D-01	5.0401081D-01	-3.6043069D-01	-2.9756015D-01	1.280000
15	7	1	7	3.7821249D-02	1.6068590D-01	5.5709073D-01	1.1984523D-02	4.3092475D-01	-1.9213792D+00	-3.6559856D-01	2.645000
15	7	1	7	1.7315740D-02	1.0451819D-01	7.6428314D-01	6.6033193D-04	2.3505653D-01	-3.1802377D+00	-6.2882768D-01	3.380000
15	7	1	7	4.0496230D-03	4.2806532D-02	9.4094654D-01	1.0350499D-06	5.9052429D-02	-5.9850387D+00	-1.2267622D+00	4.805000
15	7	1	7	2.0869720D-03	2.8260246D-02	9.6914856D-01	4.7190406D-08	3.0851395D-02	-7.3261463D+00	-1.5107252D+00	5.445000
15	7	1	7	4.9234030D-04	1.1226558D-02	9.9264029D-01	4.6215058D-11	7.3597069D-03	-1.0335216D+01	-2.1331395D+00	6.845000
15	7	1	7	9.9759086D-05	3.9729547D-03	9.9850466D-01	1.8455100D-14	1.4953418D-03	-1.3733884D+01	-2.8252595D+00	8.405000

FOR GAMMA = .150000

15	7	1	7	3.5859375D-01	4.0625000D-01	1.9455320D-12	5.9035097D-01	4.0964903D-01	-2.2886972D-01	-3.8758807D-01	0.000000
15	7	1	7	1.6944154D-01	4.9688281D-01	5.8121524D-03	2.7685661D-01	7.1733124D-01	-5.5774510D-01	-1.4428026D-01	.845000
15	7	1	7	7.6224547D-02	4.4979601D-01	6.7965136D-02	2.6258173D-01	6.6945313D-01	-5.8073550D-01	-1.7427982D-01	1.280000
15	7	1	7	1.5361088D-02	2.5025897D-01	7.4462150D-01	1.8736055D-02	2.3698645D-01	-1.7274146D+00	-6.2527649D-01	2.645000
15	7	1	7	5.6337905D-03	1.6792036D-01	9.1221048D-01	1.4067035D-03	8.6362820D-02	-2.8517974D+00	-1.0635726D+00	3.380000
15	7	1	7	8.0754302D-04	7.1871343D-02	9.8791736D-01	2.8454457D-06	1.2079794D-02	-5.5458497D+00	-1.9179405D+00	4.805000
15	7	1	7	3.3895122D-04	4.8171463D-02	9.9492504D-01	1.3108345D-07	5.0748308D-03	-6.8824521D+00	-2.2945784D+00	5.445000
15	7	1	7	2.1433319D-05	1.9622577D-02	9.9922877D-01	1.1373626D-10	7.7122819D-04	-9.9441011D+00	-3.1128171D+00	6.845000
15	7	1	7	8.4370987D-06	7.0608665D-03	9.9990342D-01	3.4054728D-14	9.6582132D-05	-1.3467823D+01	-4.4151032D+00	8.405000

FOR GAMMA = .200000

15	7	1	7	3.5859375D-01	4.0625000D-01	1.9455320D-12	5.9035097D-01	4.0964903D-01	-2.2888972D-01	-3.8758807D-01	0.000000
15	7	1	7	1.5006252D-01	5.3296405D-01	5.3709797D-03	1.9769942D-01	7.9629460D-01	-7.0399460D-01	-9.8580043D-02	.845000
15	7	1	7	8.1075831D-02	4.8241340D-01	6.7133565D-02	1.9876870D-01	7.3409773D-01	-7.0165199D-01	-1.3424612D-01	1.280000
15	7	1	7	1.0959492D-02	2.6906915D-01	7.7599766D-01	1.6213521D-02	2.0778862D-01	-1.7901227D+00	-6.8237825D-01	2.645000
15	7	1	7	3.0654870D-03	1.8086273D-01	9.3604061D-01	1.2295887D-03	6.2729805D-02	-2.9102401D+00	-1.2025261D+00	3.380000
15	7	1	7	4.4810167D-04	7.7830873D-02	9.9323628D-01	2.3827510D-06	6.7613329D-03	-5.6229213D+00	-2.1699677D+00	4.805000
15	7	1	7	1.7493097D-04	5.2311155D-02	9.9737451D-01	1.0603345D-07	2.6253874D-03	-6.9745571D+00	-2.5808066D+00	5.445000
15	7	1	7	2.2734890D-05	2.1436387D-02	9.9965947D-01	8.3700482D-11	3.4053065D-04	-1.0077272D+01	-3.4678438D+00	6.845000
15	7	1	7	2.9494439D-06	7.7591527D-03	9.9996408D-01	2.2139234D-14	3.5926357D-05	-1.3654837D+01	-4.4462052D+00	8.405000

FOR GAMMA = .250000

15	7	1	7	3.5859375D-01	4.0625000D-01	1.9455320D-12	5.9035097D-01	4.0964903D-01	-2.2888972D-01	-3.8758807D-01	0.000000
15	7	1	7	1.3228589D-01	5.6421455D-01	4.9134369D-03	1.3713469D-01	8.5795187D-01	-8.6285266D-01	-6.6537075D-02	.845000
15	7	1	7	6.7543921D-02	5.1159790D-01	6.4659964D-02	1.4566784D-01	7.8967223D-01	-8.3663631D-01	-1.0255313D-01	1.280000
15	7	1	7	1.0771743D-02	2.8701473D-01	7.7130772D-01	1.3391468D-02	1.9830082D-01	-1.8731718D+00	-7.0929594D-01	2.645000
15	7	1	7	2.3612979D-03	1.9443336D-01	9.4999490D-01	1.0445742D-03	4.8960522D-02	-2.9810607D+00	-1.3101540D+00	3.380000
15	7	1	7	4.4238462D-04	8.5603988D-02	9.9662941D-01	2.1155309D-06	3.7488760D-03	-5.6745806D+00	-2.4266785D+00	4.805000
15	7	1	7	8.7825548D-05	3.8276570D-02	9.9867375D-01	9.6272513D-08	1.3201492D-03	-7.0164977D+00	-2.8774076D+00	5.445000
15	7	1	7	1.7125364D-05	2.4072356D-02	9.9980344D-01	8.0840456D-11	1.4560315D-04	-1.0092371D+01	-3.8368292D+00	6.845000
15	7	1	7	3.5700772D-06	9.3225085D-03	9.9996713D-01	2.3541839D-14	1.2867975D-05	-1.3628160D+01	-4.8904898D+00	8.405000

FOR GAMMA = .300000

15	7	1	7	3.5859375D-01	4.0625000D-01	1.9455320D-12	5.9035097D-01	4.0964903D-01	-2.2888972D-01	-3.8758807D-01	0.000000
15	7	1	7	1.1622176D-01	5.9313327D-01	4.4235940D-03	9.2577696D-02	9.0299871D-01	-1.0334936D+00	-4.4312870D-02	.845000
15	7	1	7	5.5045089D-02	5.3841843D-01	6.0360777D-02	1.0353269D-01	8.3610653D-01	-9.8492248D-01	-7.7738386D-02	1.280000
15	7	1	7	5.2853704D-03	3.4062633D-01	7.6820167D-01	1.0764280D-02	2.0103405D-01	-1.9680150D+00	-6.9673037D-01	2.645000
15	7	1	7	1.4618449D-03	2.1089036D-01	9.5442042D-01	8.4948958D-04	4.4664688D-02	-3.0482278D+00	-1.3498413D+00	3.380000
15	7	1	7	1.2779826D-04	9.7208144D-02	9.9782481D-01	2.1505796D-06	2.1730384D-03	-5.6674445D+00	-2.6629326D+00	4.805000
15	7	1	7	2.2694726D-05	6.7904891D-02	9.9933057D-01	1.0810133D-07	6.6932690D-04	-6.9661690D+00	-3.1743617D+00	5.445000
15	7	1	7	4.0166512D-06	3.0710747D-02	9.9993962D-01	1.1831598D-10	6.0381331D-05	-9.9269566D+00	-4.2190973D+00	6.845000
15	7	1	7	7.9556548D-07	1.2670033D-02	9.9999556D-01	4.9784283D-14	4.4382815D-06	-1.3302864D+01	-5.3527852D+00	8.405000

EXT SET

NOT REPRODUCIBLE

FOR GAMMA = 0.000000

15	7	3	7	5.5859375D-01	4.0625000D-01	1.9455321D-12	5.9035097D-01	4.0964903D-01	-2.2888972D-01	-3.8758807D-01	0.000000
15	7	3	7	2.2950894D-01	3.7337514D-01	6.9603102D-03	5.9899899D-01	3.9404070D-01	-2.2257391D-01	-4.0445891D-01	.845000
15	7	3	7	1.5012360D-01	3.1810321D-01	6.0231654D-02	4.3627798D-01	5.0349037D-01	-3.6023671D-01	-2.9800883D-01	1.280000
15	7	3	7	3.7821749D-02	1.6068596D-01	5.8270145D-01	1.1986523D-02	4.0531203D-01	-1.9213068D+00	-3.9221051D-01	2.645000
15	7	3	7	1.7731574D-02	1.0451819D-01	8.0714114D-01	6.6038003D-04	1.9219848D-01	-3.1802061D+00	-7.1625005D-01	3.380000
15	7	3	7	4.0496230D-03	4.2806532D-02	9.7197242D-01	1.0350642D-06	2.8026546D-02	-5.9850327D+00	-1.5524304D+00	4.805000
15	7	3	7	2.0869720D-03	2.8260240D-02	9.8948315D-01	4.7190719D-08	1.0516806D-02	-7.3261434D+00	-1.9781161D+00	5.445000
15	7	3	7	4.9234030D-04	1.1226556D-02	9.9890187D-01	4.6215122D-11	1.0981330D-03	-1.0335216D+01	-2.9593451D+00	6.845000
15	7	3	7	9.9759086D-05	3.9729547D-03	9.9991795D-01	1.8455104D-14	8.2054165D-05	-1.3733883D+01	-4.0858994D+00	8.405000

FOR GAMMA = .150000

15	7	3	7	5.5859375D-01	4.0625000D-01	1.9455321D-12	5.9035097D-01	4.0964903D-01	-2.2888972D-01	-3.8758807D-01	0.000000
15	7	3	7	1.5944154D-01	4.9888281D-01	5.8126480D-02	2.7686236D-01	7.1732499D-01	-5.5773608D-01	-1.4428404D-01	.845000
15	7	3	7	9.6224547D-02	4.4979601D-01	6.7987936D-02	2.6258498D-01	6.6942708D-01	-5.8073012D-01	-1.7429672D-01	1.280000
15	7	3	7	1.5330683D-02	2.5025899D-01	7.4733780D-01	1.8732059D-02	2.3393014D-01	-1.7274145D+00	-6.3091381D-01	2.645000
15	7	3	7	5.6337905D-03	1.6792036D-01	9.1807042D-01	1.4067035D-03	8.0522874D-02	-2.8517974D+00	-1.0940807D+00	3.380000
15	7	3	7	8.0754302D-04	7.1871343D-02	9.9212927D-01	2.8454457D-06	7.0678849D-03	-5.5458497D+00	-2.1041420D+00	4.805000
15	7	3	7	3.3895122D-04	4.8171463D-02	9.9745950D-01	1.3108345D-07	2.5403733D-03	-6.8824521D+00	-2.5951025D+00	5.445000
15	7	3	7	5.1433315D-05	1.9622577D-02	9.9981292D-01	1.1373626D-10	1.8708003D-04	-9.9441011D+00	-3.7279726D+00	6.845000
15	7	3	7	6.4390987D-06	7.0608656D-03	9.9999087D-01	3.4054728D-14	9.1253087D-06	-1.3467823D+01	-5.0397524D+00	8.405000

FOR GAMMA = .200000

15	7	3	7	5.5859375D-01	4.0625000D-01	1.9455321D-12	5.9035097D-01	4.0964903D-01	-2.2888972D-01	-3.8758807D-01	0.000000
15	7	3	7	1.5005252D-01	5.3296405D-01	5.3712072D-03	1.9770128D-01	7.9692751D-01	-7.0399052D-01	-9.8581180D-02	.845000
15	7	3	7	6.1075831D-02	4.8241340D-01	6.7144654D-02	1.9876952D-01	7.3408582D-01	-7.0165021D-01	-1.3425316D-01	1.280000
15	7	3	7	1.0954926D-02	2.6906915D-01	7.7765041D-01	1.6213521D-02	2.0613607D-01	-1.7901227D+00	-6.8584600D-01	2.645000
15	7	3	7	3.0854600D-03	1.8086275D-01	9.3921864D-01	1.2295887D-03	5.9551767D-02	-2.9102401D+00	-1.2251053D+00	3.380000
15	7	3	7	4.4810167D-04	7.7830873D-02	9.9538343D-01	2.3827510D-06	4.6141910D-03	-5.6229213D+00	-2.3359044D+00	4.805000
15	7	3	7	1.7493097D-04	5.2311155D-02	9.9860606D-01	1.0603345D-07	1.3918387D-03	-6.9745571D+00	-2.8564111D+00	5.445000
15	7	3	7	2.2754896D-05	2.1436387D-02	9.9991064D-01	8.3700482D-11	8.9158900D-05	-1.0077272D+01	-4.0498353D+00	6.845000
15	7	3	7	2.3949439D-06	7.7591524D-03	9.9997625D-01	2.2139234D-14	3.7160370D-06	-1.3656837D+01	-5.4303877D+00	8.405000

FOR GAMMA = .250000

15	7	3	7	5.5859375D-01	4.0625000D-01	1.9455321D-12	5.9035097D-01	4.0964903D-01	-2.2888972D-01	-3.8758807D-01	0.000000
15	7	3	7	1.3285895D-01	5.6421455D-01	4.9135461D-03	1.3713528D-01	8.5295117D-01	-8.6285079D-01	-6.6537429D-02	.845000
15	7	3	7	6.7543921D-02	5.1159796D-01	6.4605464D-02	1.4566804D-01	7.8966649D-01	-8.3663572D-01	-1.0255629D-01	1.280000
15	7	3	7	7.6727743D-03	2.8701473D-01	7.9217595D-01	1.3391468D-02	1.9443259D-01	-1.8731718D+00	-7.1123095D-01	2.645000
15	7	3	7	2.3012979D-03	1.9443336D-01	9.5164236D-01	1.0445742D-03	4.7313065D-02	-2.9810607D+00	-1.3250189D+00	3.380000
15	7	3	7	2.4218465D-04	8.5603988D-02	9.9725883D-01	2.1155309D-06	2.7090556D-03	-5.6765806D+00	-2.5671821D+00	4.605000
15	7	3	7	8.7825546D-05	5.8278576D-02	9.9924138D-01	9.6272513D-08	7.5852699D-04	-7.0164977D+00	-3.1200290D+00	5.445000
15	7	3	7	9.7055363D-06	2.4672356D-02	9.9995700D-01	6.0840456D-11	4.3000684D-05	-1.0092371D+01	-4.3665246D+00	6.845000
15	7	3	7	8.5786772D-07	9.3225085D-03	9.9999842D-01	2.3541839D-14	1.5814968D-06	-1.3628160D+01	-5.8009317D+00	8.405000

FOR GAMMA = .300000

15	7	3	7	5.5859375D-01	4.0625000D-01	1.9455321D-12	5.9035097D-01	4.0964903D-01	-2.2888972D-01	-3.8758807D-01	0.000000
15	7	3	7	1.1622178D-01	5.9313327D-01	4.4236475D-03	9.2577878D-02	9.0299847D-01	-1.0334928D+00	-4.4312983D-02	.845000
15	7	3	7	5.5645089D-02	5.3841845D-01	6.0363531D-02	1.0353274D-01	8.3610373D-01	-9.8492230D-01	-7.7739639D-02	1.280000
15	7	3	7	5.2855706D-03	3.0626339D-01	7.8862766D-01	1.0764280D-02	2.0060806D-01	-1.9680150D+00	-6.9765163D-01	2.645000
15	7	3	7	1.4818492D-03	2.1089036D-01	9.5526636D-01	6.9489528D-04	4.3898747D-02	-3.0482278D+00	-1.3575479D+00	3.380000
15	7	3	7	1.2779825D-04	9.7208144D-02	9.9628188D-01	2.1505796D-06	1.7159689D-03	-5.6674445D+00	-2.7654906D+00	4.805000
15	7	3	7	4.2889726D-05	6.7904891D-02	9.9957078D-01	1.0810133D-07	4.2916756D-04	-6.9661690D+00	-3.3674338D+00	5.445000
15	7	3	7	4.0106512D-06	3.0710747D-02	9.9997655D-01	1.1831598D-10	2.1451976D-05	-9.9269566D+00	-4.6685337D+00	6.845000
15	7	3	7	2.9560540D-07	1.2670039D-02	9.9999427D-01	4.9769283D-14	7.2506926D-07	-1.3302864D+01	-6.1391416D+00	8.405000

:XT SET

FOR GAMMA = 0.000000

15	7	5	7	5.585937D-01	4.062500D-01	1.945997D-12	5.903509D-01	4.096490D-01	-2.288897D-01	-3.875880D-01	0.000000
15	7	5	7	2.295068D-01	3.733751D-01	7.775803D-03	6.080716D-01	3.841525D-01	-2.160452D-01	-4.154963D-01	.845000
15	7	5	7	1.501238D-01	3.181032D-01	7.420520D-02	4.522899D-01	4.735048D-01	-3.445631D-01	-3.246755D-01	1.280000
15	7	5	7	3.782124D-02	1.606859D-01	7.567186D-01	1.260428D-02	2.306770D-01	-1.899481D+00	-6.369955D-01	2.645000
15	7	5	7	1.773157D-02	1.045161D-01	9.399568D-01	6.805883D-04	5.936458D-02	-3.162043D+00	-1.226472D+00	3.380000
15	7	5	7	4.049023D-03	4.280653D-02	9.980974D-01	1.062018D-06	1.901515D-03	-5.973867D+00	-2.720900D+00	4.805000
15	7	5	7	2.086972D-03	2.826024D-02	9.996583D-01	4.816544D-08	3.416121D-04	-7.317264D+00	-3.466466D+00	5.445000
15	7	5	7	4.923406D-04	1.122659D-02	9.999930D-01	4.680111D-11	6.955786D-06	-1.032974D+01	-5.157653D+00	6.845000
15	7	5	7	9.975090D-05	3.972954D-03	9.999979D-01	1.859785D-14	8.610926D-08	-1.373053D+01	-7.064950D+00	8.405000

FOR GAMMA = .150000

15	7	5	7	5.585937D-01	4.062500D-01	1.945997D-12	5.903509D-01	4.096490D-01	-2.288897D-01	-3.875880D-01	0.000000
15	7	5	7	1.694415D-01	4.988828D-01	5.926682D-03	2.777189D-01	7.163541D-01	-5.563945D-01	-1.448721D-01	.845000
15	7	5	7	9.622454D-02	4.497960D-01	7.037446D-02	2.637520D-01	6.658734D-01	-5.788041D-01	-1.766082D-01	1.280000
15	7	5	7	1.535466D-02	2.502589D-01	7.946630D-01	1.874877D-02	1.865881D-01	-1.727026D+00	-7.291159D-01	2.645000
15	7	5	7	5.633790D-03	1.679203D-01	9.570338D-01	1.407094D-03	4.155911D-02	-2.851676D+00	-1.381333D+00	3.380000
15	7	5	7	3.075430D-04	7.187134D-02	9.990274D-01	2.845530D-06	9.696851D-04	-5.545836D+00	-3.013369D+00	4.805000
15	7	5	7	3.389122D-04	4.817146D-02	9.998533D-01	1.310849D-07	1.465202D-04	-6.882447D+00	-3.834102D+00	5.445000
15	7	5	7	5.143331D-05	1.962257D-02	9.999981D-01	1.137364D-10	1.872227D-06	-9.944100D+00	-5.727641D+00	6.845000
15	7	5	7	8.439098D-06	7.060865D-03	9.999999D-01	3.405473D-14	1.206308D-08	-1.346782D+01	-7.918541D+00	8.405000

FOR GAMMA = .200000

15	7	5	7	5.585937D-01	4.062500D-01	1.945997D-12	5.903509D-01	4.096490D-01	-2.288897D-01	-3.875880D-01	0.000000
15	7	5	7	1.500525D-01	5.329640D-01	5.436090D-03	1.980919D-01	7.964719D-01	-7.031309D-01	-9.882953D-02	.845000
15	7	5	7	6.107583D-02	4.824134D-01	6.856319D-02	1.992357D-01	7.322010D-01	-7.006327D-01	-1.353696D-01	1.280000
15	7	5	7	1.095492D-02	2.690691D-01	8.074758D-01	1.621767D-02	1.763064D-01	-1.790011D+00	-7.537317D-01	2.645000
15	7	5	7	3.685460D-03	1.808827D-01	9.634140D-01	1.229661D-03	3.535632D-02	-2.910214D+00	-1.451532D+00	3.380000
15	7	5	7	9.481016D-04	7.783087D-02	9.993222D-01	2.382759D-06	6.754153D-04	-5.622919D+00	-3.170429D+00	4.805000
15	7	5	7	1.749309D-04	5.231115D-02	9.999052D-01	1.060335D-07	9.465632D-05	-6.974556D+00	-4.023850D+00	5.445000
15	7	5	7	2.670489D-05	2.143638D-02	9.999989D-01	8.370049D-11	1.047195D-06	-1.007727D+01	-5.979972D+00	6.845000
15	7	5	7	2.394943D-06	7.759152D-03	9.999999D-01	2.213923D-14	5.797479D-09	-1.365603D+01	-8.236760D+00	8.405000

FOR GAMMA = .250000

15	7	5	7	5.585937D-01	4.062500D-01	1.945997D-12	5.903509D-01	4.096490D-01	-2.288897D-01	-3.875880D-01	0.000000
15	7	5	7	1.320589D-01	5.642145D-01	4.951200D-03	1.973111D-01	8.572376D-01	-8.622943D-01	-6.664550D-02	.845000
15	7	5	7	6.754392D-02	5.115979D-01	6.551107D-02	1.458471D-01	7.886418D-01	-8.361021D-01	-1.031202D-01	1.280000
15	7	5	7	1.677974D-03	2.870147D-01	8.102721D-01	1.339240D-02	1.763354D-01	-1.873141D+00	-7.536603D-01	2.645000
15	7	5	7	2.361297D-03	1.944336D-01	9.660445D-01	1.044586D-03	3.291086D-02	-2.981055D+00	-1.482660D+00	3.380000
15	7	5	7	2.423846D-04	8.560398D-02	9.994731D-01	2.115531D-06	5.247207D-04	-5.674580D+00	-3.280071D+00	4.805000
15	7	5	7	8.782554D-05	5.827857D-02	9.999314D-01	9.627252D-08	6.846860D-05	-7.016497D+00	-4.164508D+00	5.445000
15	7	5	7	9.705536D-06	2.467235D-02	9.999993D-01	8.084045D-11	6.815870D-07	-1.009237D+01	-6.166478D+00	6.845000
15	7	5	7	8.578677D-07	9.322508D-03	1.000000D+00	2.354183D-14	3.559440D-09	-1.362816D+01	-8.448618D+00	8.405000

FOR GAMMA = .300000

15	7	5	7	5.585937D-01	4.062500D-01	1.945997D-12	5.903509D-01	4.096490D-01	-2.288897D-01	-3.875880D-01	0.000000
15	7	5	7	1.162217D-01	5.931332D-01	4.445606D-03	9.265506D-02	9.028993D-01	-1.033130D+00	-4.436066D-02	.845000
15	7	5	7	5.564508D-02	5.384184D-01	6.085816D-02	1.035979D-01	8.355438D-01	-9.866487D-01	-7.803074D-02	1.280000
15	7	5	7	5.285570D-03	3.062633D-01	7.989753D-01	1.076447D-02	1.902601D-01	-1.968007D+00	-7.206521D-01	2.645000
15	7	5	7	1.481849D-03	2.108903D-01	9.463240D-01	8.948971D-04	3.581479D-02	-3.048226D+00	-1.465937D+00	3.380000
15	7	5	7	1.277982D-04	9.720814D-02	9.994538D-01	2.150579D-06	5.440426D-04	-5.667445D+00	-3.264367D+00	4.805000
15	7	5	7	4.280972D-05	6.790489D-02	9.999319D-01	1.081013D-07	6.798267D-05	-6.966169D+00	-4.167601D+00	5.445000
15	7	5	7	4.016651D-06	3.071074D-02	9.999993D-01	1.183159D-10	6.264116D-07	-9.926956D+00	-6.203140D+00	6.845000
15	7	5	7	2.459654D-07	1.267003D-02	1.000000D+00	4.978926D-14	3.268931D-09	-1.330286D+01	-8.485594D+00	8.405000

EXT SET

FOR GAMMA = 0.000000

15	7	7	7	3.583933750-01	4.062500000-01	2.669291400-12	5.903509700-01	4.096490300-01	-2.288897200-01	-3.875880700-01	0.000000
15	7	7	7	2.295088940-01	3.733751400-01	1.966185700-02	6.490202400-01	3.313179100-01	-1.877417600-01	-4.797550900-01	.845000
15	7	7	7	1.501238800-01	3.181032100-01	1.652632200-01	5.719164400-01	2.630203400-01	-2.426076200-01	-5.800196700-01	1.280000
15	7	7	7	3.782124900-02	1.606859000-01	9.235555700-01	3.081672500-02	4.562770900-02	-1.511213500+00	-1.340771300+00	2.645000
15	7	7	7	1.773157400-02	1.045181900-01	9.915250300-01	2.327527600-03	6.147446600-03	-2.633105200+00	-2.211305200+00	3.380000
15	7	7	7	4.049623000-03	4.280653200-02	9.999476700-01	6.669676800-06	4.566955400-05	-5.175895200+00	-4.340466300+00	4.805000
15	7	7	7	2.066972000-03	2.826024000-02	9.999959000-01	4.063194200-07	4.100287100-06	-6.391132600+00	-5.387185700+00	5.445000
15	7	7	7	4.923403000-04	1.122655000-02	9.999999800-01	8.027008900-10	1.808504000-08	-9.095446300+00	-7.742680500+00	6.085000
15	7	7	7	9.973590860-05	3.972954700-03	1.000000000+00	7.907340400-13	4.153863900-11	-1.210197000+01	-1.038154800+01	8.405000

FOR GAMMA = .150000

15	7	7	7	3.583933750-01	4.062500000-01	2.669291400-12	5.903509700-01	4.096490300-01	-2.288897200-01	-3.875880700-01	0.000000
15	7	7	7	1.692941540-01	4.998892210-01	9.431105500-03	2.878325900-01	7.027360100-01	-5.408595800-01	-1.532077900-01	.845000
15	7	7	7	9.622454700-02	4.497760100-01	1.034925700-01	2.959526400-01	6.005545900-01	-5.287774800-01	-2.214475100-01	1.280000
15	7	7	7	1.535160830-02	2.581458920-01	8.866114600-01	2.373566770-02	9.025006200-02	-1.624598400+00	-1.044552500+00	2.645000
15	7	7	7	5.633790500-03	1.679203600-01	9.865705500-01	1.796144000-03	1.164333000-02	-2.748083500+00	-1.933922800+00	3.380000
15	7	7	7	2.075930200-04	7.187344400-02	9.974[illegible]	3.556735600-06	6.507962700-05	-5.668964800+00	-4.186555900+00	4.805000
15	7	7	7	3.389512200-04	4.817146300-02	9.999951600-01	6.527170100-07	4.674973900-06	-6.788567000+00	-5.330220800+00	5.445000
15	7	7	7	5.144331200-05	1.962267700-02	9.999999900-01	1.[illegible]44730-10	1.053806600-08	-9.854664600+00	-7.977239100+00	6.085000
15	7	7	7	8.437[illegible]987[illegible]	7.060865600-03	1.000000000+00	4.169[illegible]-14	8.959969400-12	-1.337989600+01	-1.104769300+01	8.405000

FOR GAMMA = .200000

15	7	7	7	3.583933750-01	4.062500000-01	2.669291400-12	5.903509700-01	4.096490300-01	-2.288897200-01	-3.875880700-01	0.000000
15	7	7	7	1.506925200-01	5.329640500-01	7.852197000-03	2.044414700-01	7.096490300-01	-2.288897200-01	-3.875880700-01	0.000000
15	7	7	7	8.107393100-02	4.824134000-01	9.216350600-02	2.190806600-01	6.809064400-01	-6.894312300-01	-1.036356100-01	.845000
15	7	7	7	1.095492600-02	2.690691500-01	8.739265100-01	1.891551200-02	1.071543300-01	-6.593959600-01	-1.619347100-01	1.280000
15	7	7	7	3.885466000-03	1.808652700-01	9.844115600-01	1.414863740-03	1.417360000-02	-1.723181900+00	-9.699699700-01	2.645000
15	7	7	7	9.481010700-04	7.783087300-02	9.999141400-01	2.065770600-06	8.320451500-05	-2.869292300+00	-1.848519700+00	3.380000
15	7	7	7	1.749330970-04	5.[illegible]111550-02	9.999937900-01	1.169289010-07	6.090771400-06	-4.575494000+00	-2.979853100+00	4.805000
15	7	7	7	2.270489000-05	2.143638700-02	9.999999900-01	9.056317600-11	1.416791400-08	-6.[illegible]20815000+00	-4.[illegible]7,846694[illegible]+00	5.445000
15	7	7	7	2.398943900-06	7.759152700-03	1.000000000+00	2.361792400-14	1.227848400-11	-1.[illegible]0468D+01	-1.[illegible]80+01 -1.091085500+01	8.405000

FOR GAMMA = .250000

15	7	7	7	3.583933750-01	4.062500000-01	2.669291400-12	5.903509700-01	4.096490300-01	-2.288897200-01	-3.875880700-01	0.000000
15	7	7	7	1.322858450-01	5.642145500-01	6.622940900-03	1.412653300-01	8.521117900-01	-8.499664400-01	-[illegible]950342800-02	.845000
15	7	7	7	6.754392100-02	5.115979800-01	8.206526300-02	1.577287100-01	7.602060300-01	-8.020892500-01	-1.[illegible]8690-01	1.280000
15	7	7	7	1.[illegible]7974300-03	2.[illegible]701474[illegible]-01	8.566609270-01	1.477055800-02	1.286201700-01	-1.830603100+00	-8.90[illegible]00-01	2.645000
15	7	7	7	2.361297900-03	1.944333600-01	9.804758000-01	1.130856300-03	1.839334100-02	-2.946592600+00	-1.735339[illegible]00	3.380000
15	7	7	7	2.[illegible]238465500-04	8.560398700-02	9.[illegible]8567800-01	2.[illegible]2280880[illegible]-06	1.319993800-04	-5.653097900+00	-3.879428100+00	4.805000
15	7	7	7	8.786554600-05	5.827857600-02	9.999969100-01	1.002061300-07	1.073620200-05	-6.999105700+00	-4.969149300+00	5.445000
15	7	7	7	2.[illegible]025363[illegible]-06	2.[illegible]67235[illegible]-02	9.999999700-01	8.[illegible]2847212[illegible]-11	3.[illegible]2241880[illegible]-08	-1.[illegible]0081196[illegible]+01	-7.[illegible]6915786[illegible]+00	6.[illegible]5000
15	7	7	7	8.578677200-07	9.322508500-03	1.000000000+00	2.392696600-14	3.856733700-11	-1.362111200+01	-1.041378000+01	8.405000

FOR GAMMA = .300000

15	7	7	7	3.583933750-01	4.062500000-01	2.669291400-12	5.903509700-01	4.096490300-01	-2.288897200-01	-3.875880700-01	0.000000
15	7	7	7	1.162217800-01	5.931332700-01	5.593120400-03	9.507658300-02	8.993303000-01	-1.021926400+00	-4.608077600-02	.845000
15	7	7	7	5.564550800-02	5.384184300-01	7.[illegible]1380710[illegible]-02	1.[illegible]0443270[illegible]-01	8.174184600-01	-9.568607500-01	-8.755564300-02	1.280000
15	7	7	7	3.285570600-03	3.062633900-01	8.298136600-01	1.143036400-02	1.587559800-01	-1.941939900+00	-7.992699100-01	2.645000
15	7	7	7	1.[illegible]818492[illegible]-03	2.[illegible]089030[illegible]-01	9.729194800-01	9.336928800-04	2.614668250-02	-3.029796000+00	-1.582581000+00	3.380000
15	7	7	7	1.279782500-04	9.720814400-02	9.997241700-01	2.194429300-06	2.736350800-04	-5.658678400+00	-3.562028200+00	4.805000
15	7	7	7	7.[illegible]2889726[illegible]-05	6.[illegible]7904891[illegible]-02	9.[illegible]9997260[illegible]-01	1.[illegible]0906677[illegible]-07	2.[illegible]7267520[illegible]-05	-6.959924900+00	-4.566035900+00	5.445000
15	7	7	7	[illegible].[illegible]1865[illegible]20-06	3.[illegible]0710747[illegible]-02	5.[illegible]9999860[illegible]-01	1.191311400-10	1.358785200-07	-9.923974700+00	-6.866069200+00	6.085000
15	7	7	7	4.[illegible]5885543[illegible]-07	1.[illegible]7[illegible]-02	1.000000000+00	[illegible]-14	3.[illegible]1111090[illegible]-10	-1.[illegible]3001540[illegible]+01	-8.[illegible]5070836[illegible]+00	8.405000

NOT REPRODUCIBLE