AN ABSTRACT OF THE THESIS OF

James Fryklund for the degree Master of Science

in Electrical and Computer Engineering presented on

Aug. 14 1975

Title: Hierarchial Digital Communication Processing in a Multi-Access Tele-Communication Environment

Redacted for Privacy

Abstract approved:

Donald L. Amort

This paper is concerned with the application of hierarchial and parallel computing structures to digital data communications processing in a multi-access tele-communications environment. Designs are analyzed and evaluated in terms of data collected from remote access tele-communications traffic and in terms of the application to several processing problems in the field of multi-access tele-communications.

Hierarchial Digital Communication Processing in a Multi-Access Tele-Communication Environment

by

James Fryklund

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

MASTER OF SCIENCE

June 1977

APPROVED:

Redacted for Privacy

Associate Professor of Electrical and Computer Engineering in charge of major

Redacted for Privacy

Head of Department of Electrical and Computer Engineering

Redacted for Privacy

Dean of Graduate School

Date thesis is presented August 14, 1975

ACKNOWLEDGEMENT

I would like to express my appreciation to Professor Donald L. Amort for his advice and direction.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES
(cont.)

LIST OF TABLES

GLOSSARY OF ABBREVIATIONS

HOST	Computer system resource that processes user procedures
I/O	Input/output
LIU	Line interface unit
CPS	Communications processing system
PLIU	programmable line interface unit
RJE	Remote job entry terminal
URT	Universal receiver/transmitter
stat's	Term used for set/reset, testable flip flops in micro programming environment
EIA	Electronic Industries Association
RAM	Random access memory
ROM	Read only memory
DMA	Direct memory access
LED	Light emitting diode
IOC	Input/Output command

HIERARCHIAL DIGITAL COMMUNICATION PROCESSING IN A MULTI-ACCESS TELE-COMMUNICATIONS ENVIRONMENT

I. INTRODUCTION

With the application of advanced techniques, digital data communications is the rapidly expanding field whereby large volumns of data are reliably and economically transmitted over various types of digital networks. The advanced techniques involve several fields, such as computer software, signal transmission, computing structures and computer firmware/hardware. This paper will be concerned with the latter two items, computing structures and firmware/hardware. These subjects will be dealt with in terms of the systems design, logic design, state diagrams and the application to digital data communications processing in a multi-access tele-communications environment.

The computing system[1] from which data for II and IV was derived is indicated in Figure 1, and functionally hierarchial processing is accomplished on the data stream as it passes bi-directionally through the individual computing elements of the system. In general, the functions accomplished in the various computing elements in Figure 1 are: 1) the Host computing system (Host) manages all resources and is the only computing resource that processes

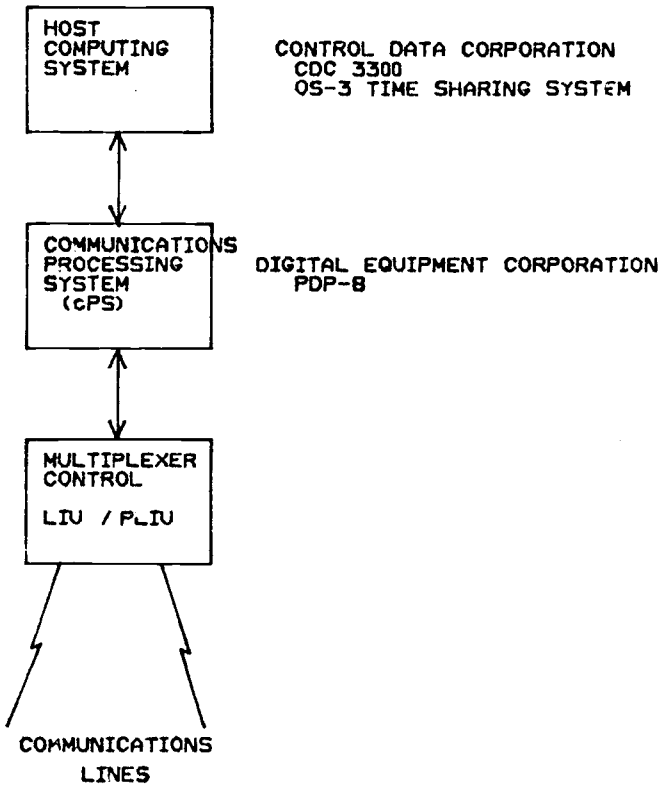


FIGURE 1: SINGLE SYSTEM

user procedures; 2) the communication processing system (CPS) provides data buffering functions, input/output (I/O) control functions, code translation, and line discipline dependent functions; 3) the Line Interface Units (LIU) perform bi-directional byte assembly/disassembly, generate and check parity, generate statuses and interrupts, edit certain of the data streams for control byte sequences and take appropriate action independently or in conjunction with the CPS.

Fail Soft Capability

The equipment configuration in Figure 2 is a simplified block diagram of the inconnection pattern that is utilized for providing greater communications processing continuity ("fail soft" capability). The term "fail soft" is used to define a computer processing system that can provide a continuum of computing resources at a degraded level when certain classes of failures occur within the system. It is the intention to provide such a continuum of communications processing resources, "fail soft" capability, via the processing structure indicated in Figure 2 in conjunction with supporting software. More precisely, a "fail soft" capability will be provided in a large percentage of cases when failures occur in the CPS or the CPS/HOST controllers. There is a set of failures that can occur in the CPS/HOST controllers and in the CPS bus controllers that would be fatal; however, the probability of this set of failures occurring is small with respect to the other failures that can occur in the above systems for which "fail soft" capability does cover.

Basically, "fail soft" capability will be implemented via having one of the CPS, CPS/HOST controller sets take over the functions of the other CPS set. There are three situations that may provoke the CPS's to degrade to a

"fail soft" configuration. These are: manual initiation, CPS set self imposed, and CPS set externally imposed. The manual case is the situation where it is desired to take one of the systems off-line for such purpose as maintenance, testing, etc. The second case is the situation where a CPS detects internal problems that impedes performing functions correctly. Problems of this type include failures in the CPS/HOST controllers, failures in the CPS is malfunctioning to such an extent that it is unaware that it is malfunctioning. This case is the most common. In this case, via a default system the other CPS set, e.g., CPS set B, will be interrupted. The interrupt status will define CPS set A as having failed. CPS set B will then commence an initialization procedure and will finally enable the operation of multiplexer A in conjunction with CPS set B, see Figure 2a.

A default system approach that has merit is one where CPS set A on a low priority interrupt driven basis must execute a time dependent chain of events within prescribed time windows, see Figure 2c. The code that is executed on this low priority basis will examine many internal parameters to detect system failures or degradation. From past experience and analysis this technique seems to offer a high probability of successfully detecting CPS set malfunctions.

Symetrical failure detection and enabling systems are implied such that system continuity remains through degradation if either CPS set A or CPS set B fails, Figures 2a and 2b.

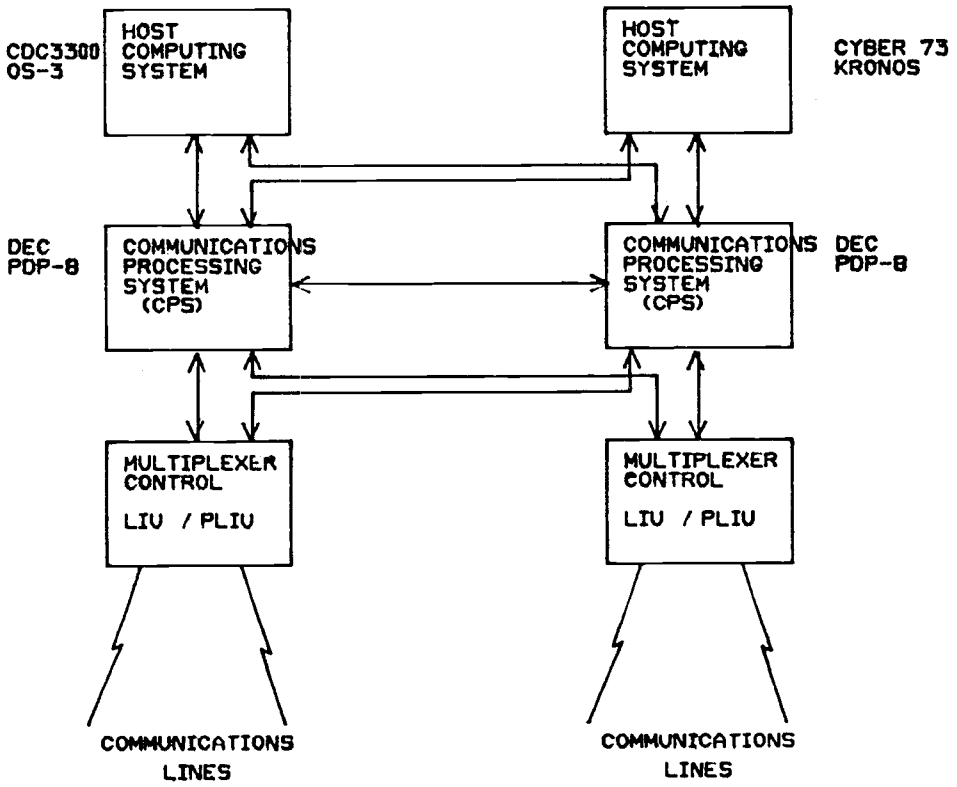


FIGURE 2: CROSS COUPLED SYSTEM

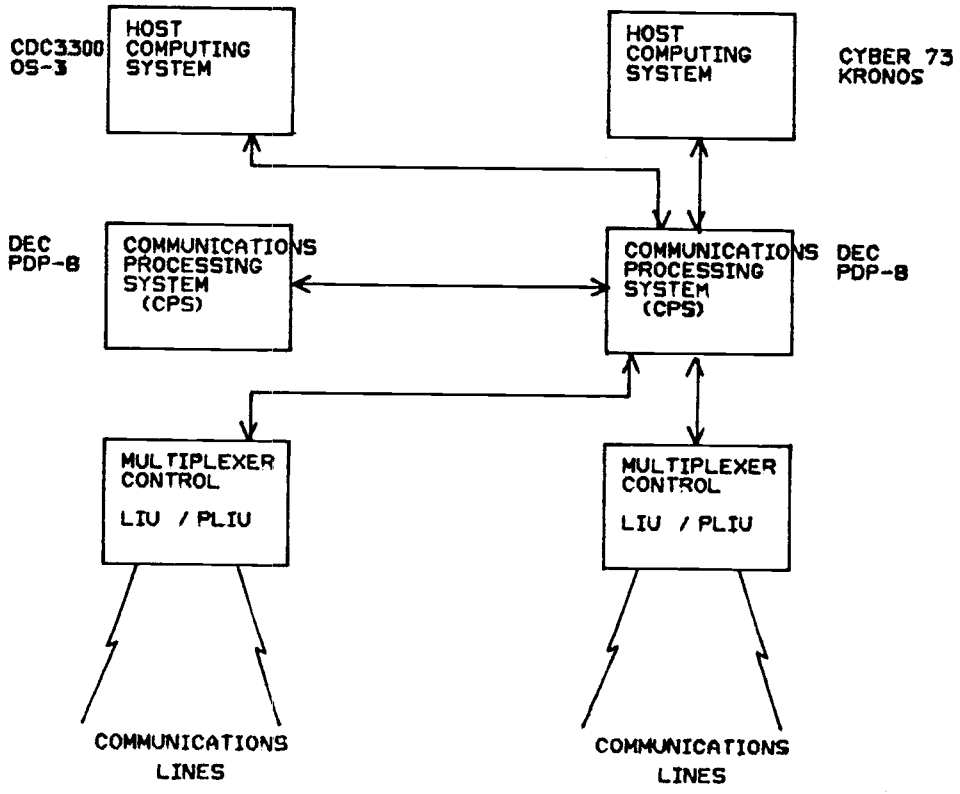


FIGURE 2A: FAIL SOFT SYSTEM CONFIGURATION

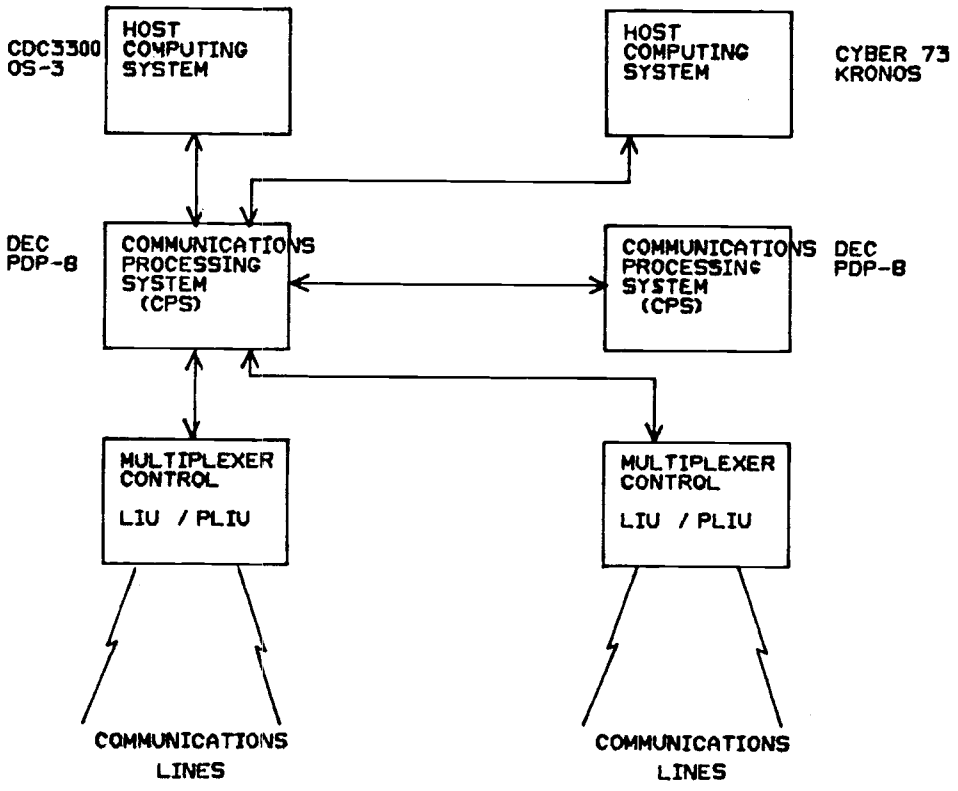
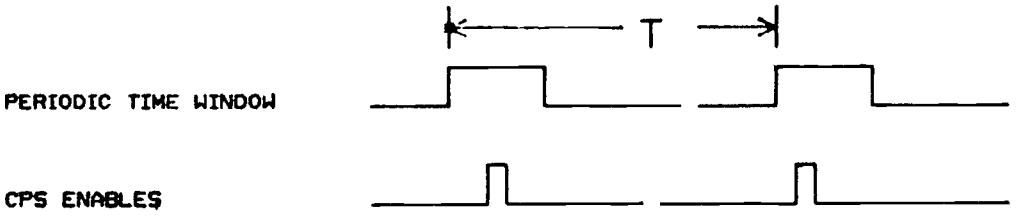


FIGURE 2B: FAIL SOFT SYSTEM CONFIGURATION

Referring to Figure 2c, the lack of occurrance of an enable signal within a timing window causes an interrupt providing the interrupt was selected. The status defines the cause of the interrupt. It has been suggested by others that n cycles of faults defines a CPS failure and m cycles of absence of faults defines a properly running CPS and a similar technique could be used for start up conditions.

CORRECT TIMING IS DEFINED AS FOLLOWS:



FAULTS ARE DEFINED AS FOLLOWS:

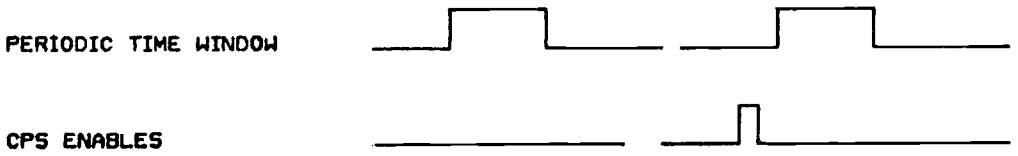


FIGURE 2C: FAULT TIMING DEFINITION

Distributed Processing

The general philosophy of hierarchial and parallel processing can effectively be applied to digital data communications. The set of functions to be accomplished on a data stream can be divided into a number of subsets. The functions of each subset should share the property that they can all be accomplished with a common, and as limited as possible, set of computing resources. The functions that have been grouped into subsets, with the previously defined properties, are then assigned to computing elements, such that the computing elements cover the functional requirements of the subsets assigned. The objective herein is to assign the functions as far from the top of the structure, the HOST, as possible, contingent upon remaining consistent with the previously defined criteria for the assignment of the functions to computing elements within the structure. The magnitude of the computational functions that can be effectively distributed increases as micro-electronics technology advances in cost/performance, i.e., as single chip computational complexities increase and cost per chip decreases. The technology determines whether it is effective to distribute functions

horizontally in a structure and also whether it is effective to distribute functions vertically as well.

Insofar as digital data communications is concerned, in those applications where the data stream has to be edited on a byte basis sophisticated direct memory access I/O system without distributed processing is ineffective. It is this bit by bit, byte by byte processing of the data stream that is particularly appropriate for the application of distributed (distributed as used herein implies a hierarchial/parallel structure) especially when the processing function resource requirement is well bounded. There are techniques that can be applied to attack the problem of functional processing requirements that are not well bounded, such as common and slave memory architectures.

It is with these concepts that an overview of a CPS with distributed PLIU's will be analyzed in terms of typical applications.

II. SYSTEM OVERVIEW

This chapter contains a justification for the distributed architecture approach to communications processing based on data collected, Appendix B, on the system indicated in Figure 1. This data is then summarized in Table 1. The data in Table 1 is then extrapolated, Table 2, to satisfy the specifications for the system indicated in Figure 2 which utilize the same computing resources and implementation techniques of the system in Figure 1. The capacities for each CPS, Figure 2, are maximum of 256K (where K=1024) 12 bit words with 1.2 microsecond cycle time, 262 LIU's 144 bi-directional direct memory access channels under the control of 18 programmable controllers (PLIU's). Table 3 indicates the distributed architecture cycle loadings for the same I/O activity levels as indicated in Table 2.

Data Gathered

Data was gathered on the I/O activity for interactive non-buffered remote terminals from the system indicated in Figure 1. Remote terminal access to CPS ports is a system without contention limitations. This environment is capitalized on in the design of the PLIU in several ways to minimize the cost per channel. The general philosophy is to minimize the facilities which are dedicated to specific channels, because of the low duty cycles, and to share the majority of the logic through hierarchial control schemes. The total number of terminals which are available to users is approximately 200. Typical concurrent usage is indicated in Figure 2d. However, data gathered on the instantaneous I/O activity indicates an activity substantially less than the number of users concurrently using the system. Items that affect instantaneous I/O activity are user program resource demanding characteristics relative to the host computing system resource capacities. If user program resource requirements are significant with respect to the available resources from the host system then the host computing system may not be able to sustain user programs in an I/O activity. Hence, the instantaneous terminal I/O activity is only loosely correlated with the

number of concurrent users because the preceeding resource relationship is relevant in this case.

The data, Appendix B, was collected from the Oregon State University time-sharing system OS-3, Figure 1. The number of concurrent users associated with each of the data sets (1, 2, 3, 4, 5, 6, 7, 8, and 9) is 29, 32, 39, 43, 41, 32, 35, 44, 44 and 43, respectively. For each of the data sets there is a histogram. For the composite data set there is a frequency distribution, cumulative frequency distribution, histogram and a variable description table. The various histograms indicate the frequency of occurrence of instantaneously active terminals.

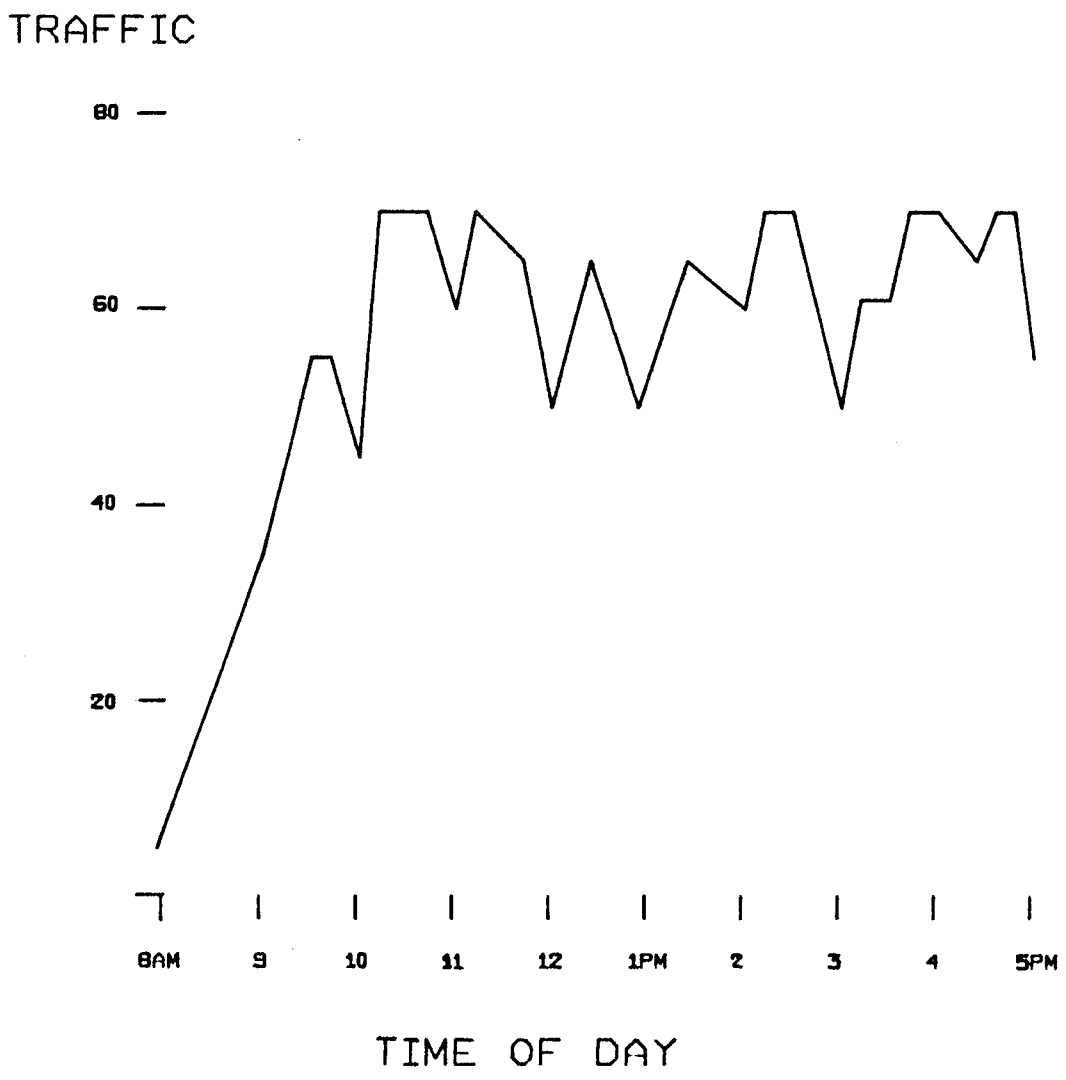


FIGURE 2D: CONCURRENT USERS

System Cycle Loading

Table 1 is derived from an analysis of the interrupt service times and architecture for the system indicated in Figure 1. The impact of the I/O activity is given as a precentage of available CPS cycles required to support the activity level for one second. The I/O activity in categories A and B of Tables 1, 2 and 3 have a low level of hardware support whereas the I/O activity in other categories have a higher level of support. According to the data collected, see Appendix B, the I/O activity indicated in category A and B, Table 1, is equivalent of approximately 38 logged on interactive terminals. The I/O activity in category C is equivalent to approximately 5 logged on interactive graphics terminals operating at a 20% duty cycle. And finally, the I/O activity in category D is that required to support one Remote Job Entry terminal (RJE) at 50% cuty cycle.

Table 2 and 3, I/O activity is extrapolated from Table 1 which is consistent with the channel capacities and bandwidths that are objectives for the system indicated in Figure 2: 1) category A and B, the interactive terminal I/O activity is equivalent to 114 logged on interactive terminals each at 110 bits per second; 2) category C,

25 interactive asynchronous high speed terminals each at 9600 bits per second and at a 20% duty cycle, and 3) category D, 10 sophisticated line discipline high speed devices each at 9600 bits per second, full duplex and at a 50% duty cycle operation.

The CPS cycle loading data that appears in Table 2 assumes the same implementations as that in Table 1. Table 3 has the same I/O loadings as does Table 2 but utilizes the distributed architecture of the system indicated in Figure 2. The lower cycle loading figures in Table 3 reflects the off loading of processing functions from the CPS processor to the distributed PLIU processors. As indicated in Table 3 the impact of the PLIU's appears only on category D. The PLIU supports category C also, however, the CPS overhead is nearly the same for this approach as in Table 1. The PLIU's can be used to off load the main CPS processor from other tasks if desired, this aspect is not addressed further herein.

It is apparent from Table 2 that the desired I/O activity loading cannot be supported by utilizing the same implementations and resources that existed in the Table 1, Figure 1 case, since the CPS cycles required per second exceeds the number available from the CPS. The required CPS cycles listed in Tables 1, 2 and 3 are not maximums

since the CPS has to perform several other overhead functions not included. Also, the CPS has to have a surplus of cycles above the typical requirement such that the probabilities of short duration data overruns (lost data) and other time critical functions are within desired design limits (very small). In other words, the cycle requirement is not constant but is a summation of synchronous and asynchronous events some of which are time critical and some of which are not time critical. The more sophisticated the hardware and software designs are in addressing this aspect of the design problem, the closer 100% cycle utilization can be approached. For the subject application, a utilization of 50% is maximum considering data completeness. The distributed processor approach, Figure 2 and Table 3, can satisfy the I/O activity requirements as well as the CPS cycle limits.

TABLE 1. CPS CYCLE LOADING WITHOUT DISTRIBUTED PROCESSORS

		TRANSFER ACTIVITY PER SECOND	CYCLES EACH	TOTAL CYCLES	% TOTAL AVAILABLE CYCLES
A.	ASYNCHRONOUS INPUT	10 BYTES	350	3500	0.45
B.	ASYNCHRONOUS OUTPUT	60 BYTES	450	27000	3.5
C.	ASYNCHRONOUS HIGH SPEED DIRECT MEMORY ACCESS	2 BLOCKS	600	1200	0.16
D.	SOPHISTICATED HIGH SPEED DIRECT MEMORY ACCESS	2.5 BLOCKS	10000	25000	3.2
E.	INTER COMPUTER READ	20 BLOCKS	700	1400	1.8
F.	INTER COMPUTER WRITE	20 BLOCKS	650	13000	1.7
					10.81%

TABLE 2. CPS CYCLE LOADING WITHOUT DISTRIBUTED PROCESSORS

		TRANSFER ACTIVITY PER SECOND	CYCLES EACH	TOTAL CYCLES	% TOTAL AVAILABLE CYCLES
A.	ASYNCHRONOUS INPUT	30 BYTES	350	10500	1.4
B.	ASYNCHRONOUS OUTPUT	180 BYTES	450	81000	10.5
C.	ASYNCHRONOUS HIGH SPEED DIRECT MEMORY ACCESS	20 BLOCKS	600	12000	1.6
D.	SOPHISTICATED HIGH SPEED DIRECT MEMORY ACCESS	100 BLOCKS	10000	10^6	130.
E.	INTER COMPUTER READ	90 BLOCKS	700	63000	8.2
F.	INTER COMPUTER WRITE	60 BLOCKS	650	36000	4.7
					156.4%

TABLE 3. CPS CYCLE LOADING WITH DISTRIBUTED PROCESSORS

		TRANSFER ACTIVITY PER SECOND	CYCLES EACH	TOTAL CYCLES	% TOTAL AVAILABLE CYCLES
A.	ASYNCHRONOUS INPUT	30 BYTES	350	10500	1.4
B.	ASYNCHRONOUS OUTPUT	180 BYTES	450	81000	10.5
C.	ASYNCHRONOUS HIGH SPEED DIRECT MEMORY ACCESS	20 BLOCKS	600	12000	1.6
D.	SOPHISTICATED HIGH SPEED DIRECT MEMORY ACCESS	100 BLOCKS	700	70000	9.1
E.	INTER COMPUTER READ	90 BLOCKS	700	63000	8.2
F.	INTER COMPUTER WRITE	60 BLOCKS	650	36000	4.7
					35.5%

III. LINE INTERFACE UNITS PROGRAMMABLE TO THE BYTE LEVEL

This section pertains to a line interface unit design programmable to the byte level, a design that utilizes universal receiver/transmitters (URT). This approach utilizes the URT processing capabilities to off-load certain processing requirements from the PLIU programmatic control. The URT's have a significant amount of processing capability, are economical mass produced MOS LSI, and are flexible because of programmatic features. The particular URT utilized in this design is the INTEL 8251. It performs double buffered bi-directional byte parallel/bit serial assembly/disassembly, detects synchronous idle input codes, and can be programmed for various types of byte parity generating and checking as well as for insertion of output synchronous idles. The utilization of the URT processing capabilities is an example of the partitioning of the total set of processing functions into subsets that can be accomplished on a hierarchial basis.

Systems Design of the PLIU to the Byte Level

The architecture of the PLIU is indicated in Figure 3 as well as each of the subsystems. As indicated in Figure 3, the subsystems are interconnected over an internal bi-directional bus. This bus is buffered from the bi-directional multiplexer bus for three reasons: 1) to minimize loading on the multiplexer bus; 2) to re-establish the signal to noise ratio, and 3) such that data transmission on the two buses can be occurring independently. In this manner, the performance of each PLIU system is determined by the communication lines with which it is communicating and not restricted unnecessarily by the transmission requirements of the multiplexer control to the other LIU's and PLIU's. Only when a PLIU has to access common memory over the dynamic time-multiplexed bus will throughput of a PLIU be affected by loading on the multiplexer refers to the technique of utilizing the bus for various functions on a dynamic time slot allocation basis. Time on the bus is divided into slots and functions are accomplished in these time slots on a basis determined by a priority resolving network.

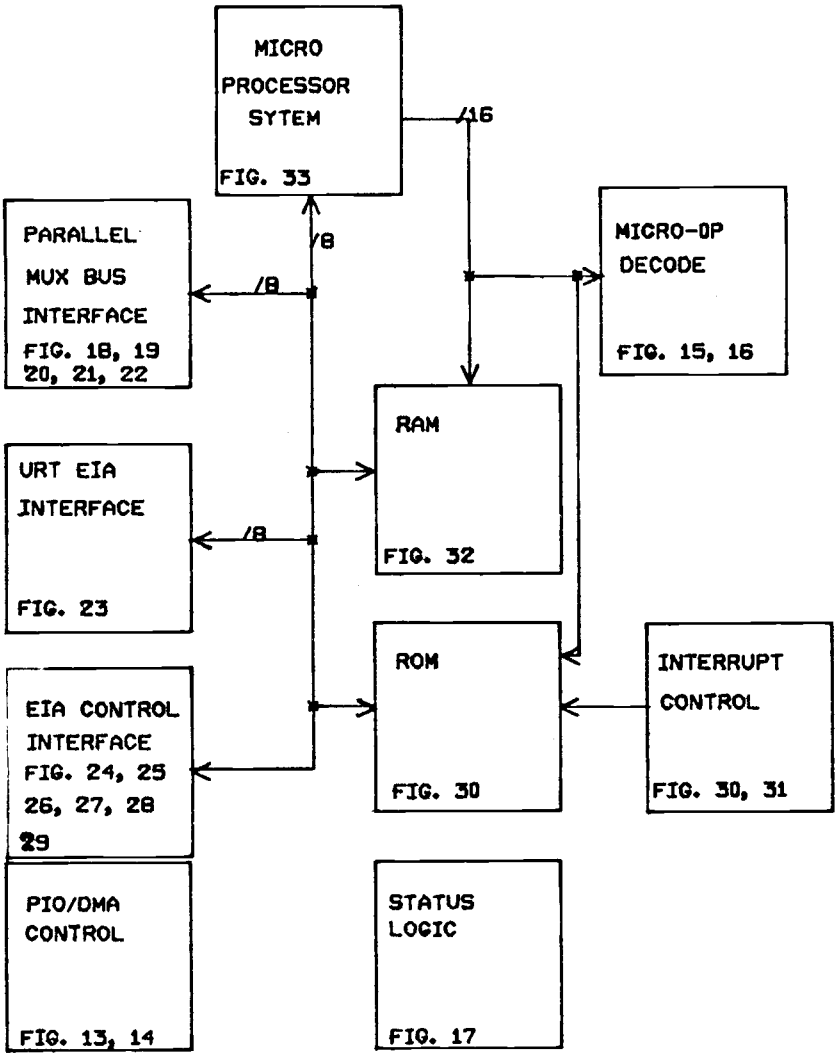


FIGURE 3. PLIU SUBSYSTEMS

An important aspect of the system's design is the allocation of the address space to various memory subsystems: ROM, local slave read/write RAM, common read/write RAM, and interlock read/write RAM (see Figure 4). A segment of the address space is also allocated to micro-operation decodes. The ROM is a memory space used for initialization, debug, restart and interrupt vectors. The slave RAM is used for nearly all program and data storage necessary to support the required communications processing. It is desirable to have most of the processor operators and operands in local memory from a performance standpoint. If the distributed processors had to contend with each other for access to operators and operands over the multiplexed bus to common memory the performance would be degraded. (The multiplexed bus can sustain 0.4 MHZ.) The address spaces in the distributed processors that overlaps with the other processors and with the CPS processor is called the common memory space. There are no restrictions on the use of this address space except for programmatic write access control. For optimum performance, usage should be limited to necessary interprocessor communication, and dump and restart routines. It can be used for seldomly executed or referenced common operator and operands also. The interlock memory

can be used to set logical interlocks such that a particular processor can restrict access by other processors while a subset of common storage is being manipulated. The basic interlock memory operations include:

1) conditional destructive read
    a) if interlock not set, set interlock and set I.D. and transfer interlock bit and I.D. to processor
    b) if interlock is set, transfer interlock bit and I.D. to processor
2) reset interlock, reset interlock and set I.D.

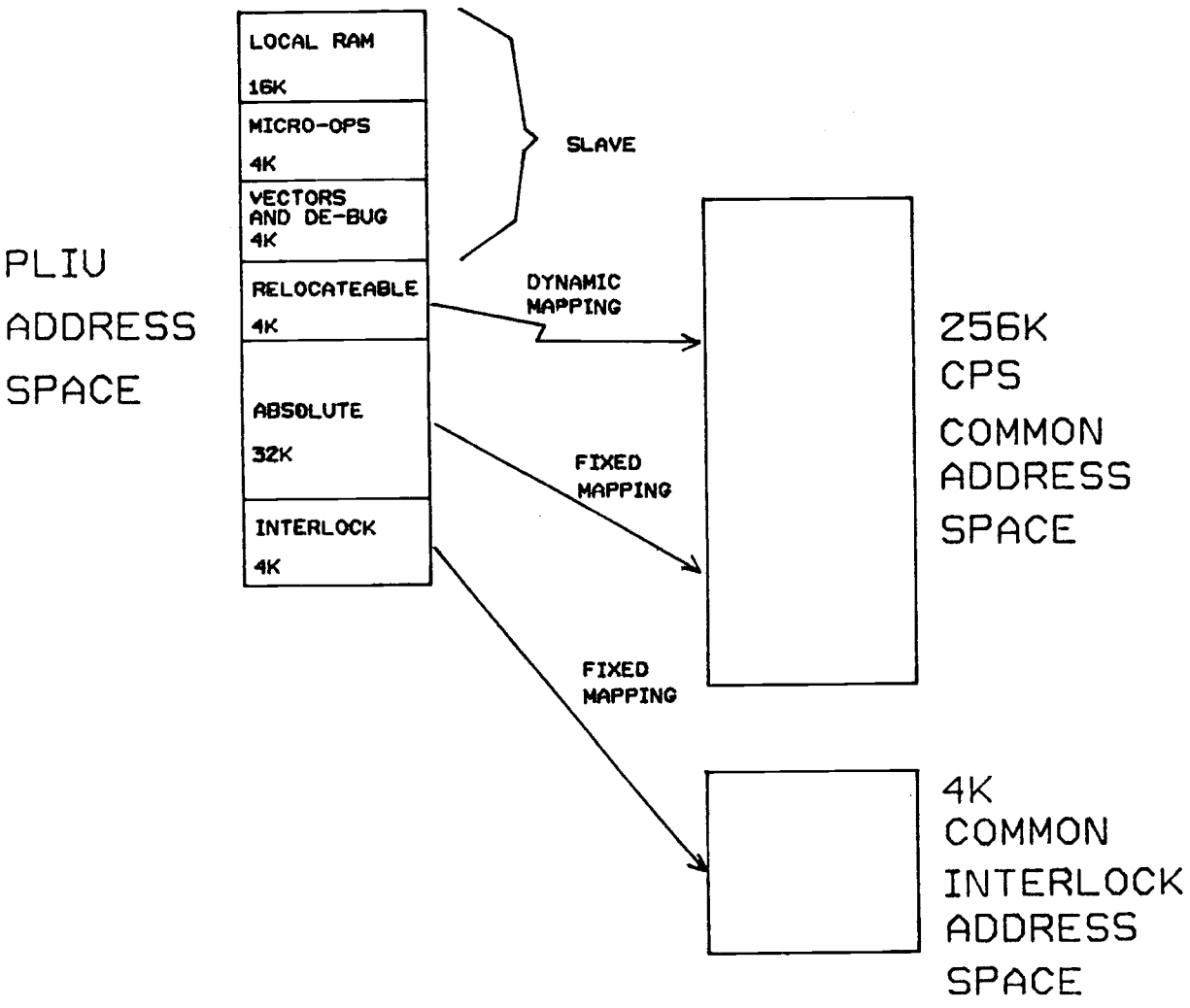


FIGURE 4. ADDRESS SPACE ALLOCATION

Discussion of Appendix A Schematics

Figure 13 details the logic for PDP-8 control over the PLIU's. The decoding of the various instructions is defined here. Figure 14 delineates the logic in the PLIU that detects Direct Memory Access (DMA) references to the PLIU and decodes the commands which in turn control the transmission of control and data information. Figure 15 details the PLIU micro-processor address decode (the micro operations). These micro operations are used for setting and resetting flops, testing, loading and reading registers, etc. It should be noted that micro operation decodes are grouped into three mutually independent sets all of which are enabled by the signal micro-op's.

Figure 16 contains the logic for PDP-8 enable of PLIU DMA activity, PLIU micro-processor address decode and DMA request setting logic. The DMA request enable must be set by the PDP-8 in order for the PLIU to access common memory. The PLIU micro processor address decode classifies the address into six sets: 32k absolute common memory reference, 16k local RAM reference, 4k micro-operations decode, 4k relocateable common memory references, 4k interlock memory reference, and 4k local ROM.

Figure 17 indicates the gating, testing and flip flops available for inter-processor interrupting, enabling, etc. Figure 18 indicates the tri-state buffers that are used to minimize loading on the bus and to isolate and regenerated signal-to-noise ratios. Figure 19 is the logic that implements the multiplexing of control and data information between the PLIU and the common memory system controller. This logic multiplexes the addresses for absolute references and for relocateable references, and control information. The control information is transmitted for every common memory reference. Figure 20 multiplexes the least significant address bits to the common memory controller, these bits are transmitted for every common memory reference.

Figure 21 is the logic for the high order 4 bits of the data that is transmitted for every common memory reference. Figure 22 is the logic necessary for the transmission of the lower 8 bits of data between the PLIU's and the common memory over the bi-directional multiplexer bus. Figure 23 consists of two of the URT's out of the 8 that are the serial/parallel line communication controllers. Figure 24 is the logic for generating the URT serial clocks for synchronous, asynchronous and loopback modes of operation. Figures 25, 26, 27, 28 and 29 are 4 bits out of the 8 bits

that interface control and data signals between the EIA RS-232 interface, LED displays, and URT's for synchronous, asynchronous and loopback modes.

Figures 30 and 31 are the interrupt support logic. Figure 32 is the 4k RAM local memory and support logic. The RAM's used are the AM 9140's. These are bussed together in the usual fashion to yield a 8 bit word and an address space of 16k words. Figure 33 is the microprocessor and support logic.

IV. APPLICATIONS

Input/Output Command Queuing

The PLIU's system design includes the facilities and can be programmed to utilize the technique of data block chaining. Data block chaining is the technique that allows the operating system to queue input and/or output commands (IOC) preceding I/O initiation or during I/O activity. Some implementations of the technique restrict queuing of IOC to occurring previous to initiation of the I/O activity. There are several applications for the technique, such as: 1) for accommodating computer system input activity with unknown message lengths; 2) for accomplishing continuous I/O message transmission across segmented or paged computer memory structures; 3) for accommodating variable length message I/O activity with CPS control of time dependency of I/O interrupt servicing, and 4) for transparent text transmission.

The manner in which data block chaining is used to control I/O interrupt service time will be discussed here. This technique is implemented in this design such that the CPS can queue IOC's for each concurrent activity beyond the currently utilized IOC. The CPS has control of the block size for each IOC. The PLIU's issue interrupts to

the CPS as IOC's are expended, and if at least one IOC has been queued by the CPS the PLIU starts utilizing it transparent to the CPS. The CPS can service PLIU issued I/O interrupts while the PLIU is utilizing the previously queued IOC. As long as the interrupt processing is completed previous to the PLIU expending the current IOC queue no timing, data over-run or lost data error conditions will occur. Since the CPS controls the block length of the IOC and since block length is directly related to the time required by the PLIU to expend the IOC, hence the I/O interrupt service time dependency is CPS controlled.

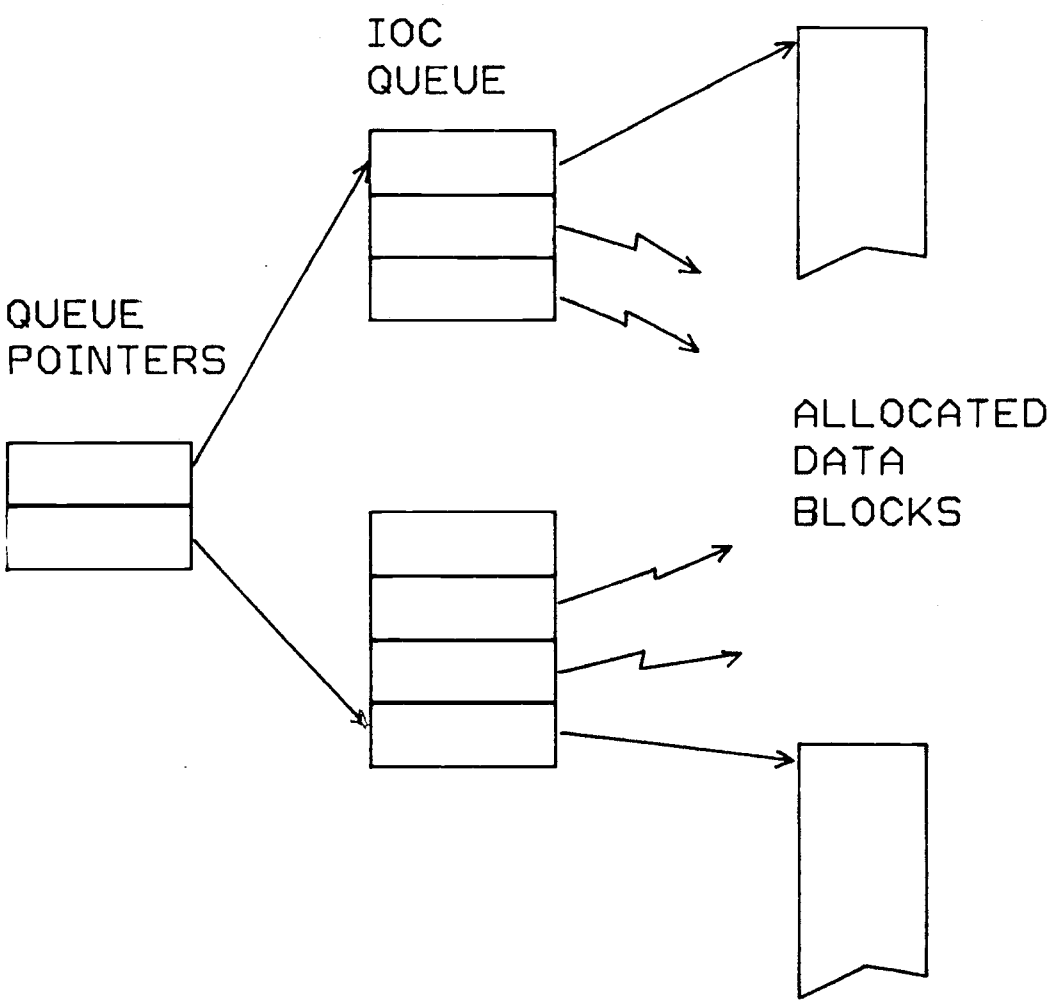


FIGURE 5. BLOCK CHAIN

Processing Synchronous Data:

The PLIU design has been evaluated to verify that its performance capabilities include the processing functions of the data streams indicated in figures 6 and 7. Their data streams are bi-directionally independent (full duplex) and are of the synchronous type. The objective in synchronous communications is to minimize the amount of control information that has to be transmitted along with the data. The synchronous type of line discipline has a sequence of control characters called the header which prcedes the message to be transmitted. The message is immediately followed by an end of message sequence - the trailer. The header is used via the receiving PLIU to determine the byte boundaries the message is assembled in and finally the trailer sequence is detected. Usually the latter contains a message checkword which is used to verify proper reception of the message by the receiving PLIU.

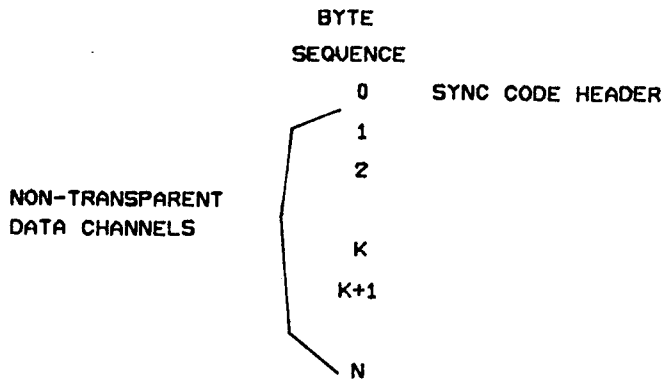


FIGURE 6. TIME-MULTIPLEXED DATA FORMAT

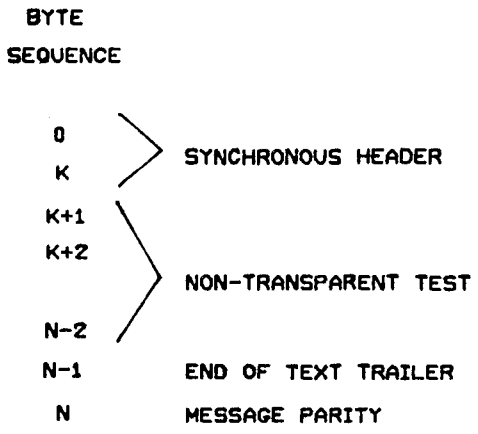


FIGURE 7. SYNCHRONOUS DATA FORMAT

Interlock Free Communications:

This discussion brings attention to the important problem of communication between the asynchronously executing processors. As indicated previously, it is implied here that the communication be accomplished with linked queue structures. Two closed structures are used, where each structure is: 1) a queue is stuffed at the tail by processor A and fetched at the head by processor B, 2) a second queue is stuffed at the tail by processor B and fetched at the head by processor A - see figure 8. Interlock free inter-processor communication for the asynchronously executing processors can be sustained by this scheme.

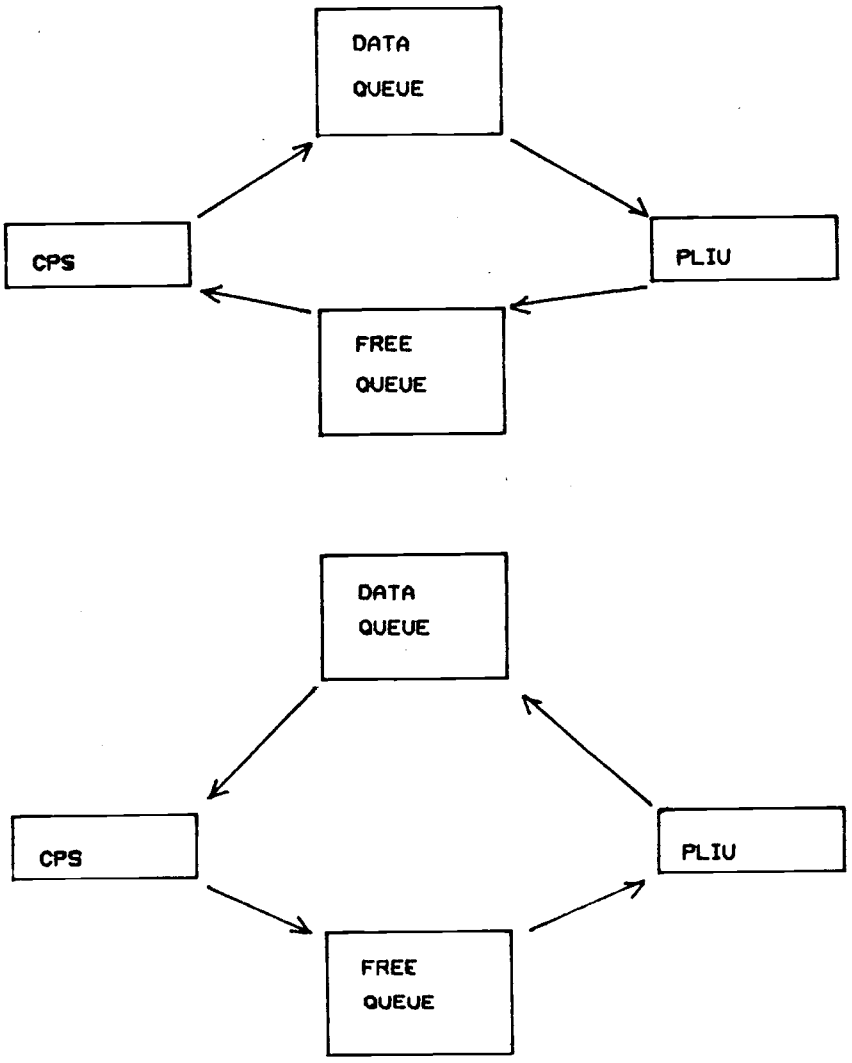


FIGURE 8. CPS/PLIU COMMUNICATION PHILOSOPHY

TABLE 4. TYPICAL CODE OCCURRENCE

CODE TYPE	PROBABILITY OF OCCURRENCE
IDLES	.84
DATA	.16
CONTROL	<.01

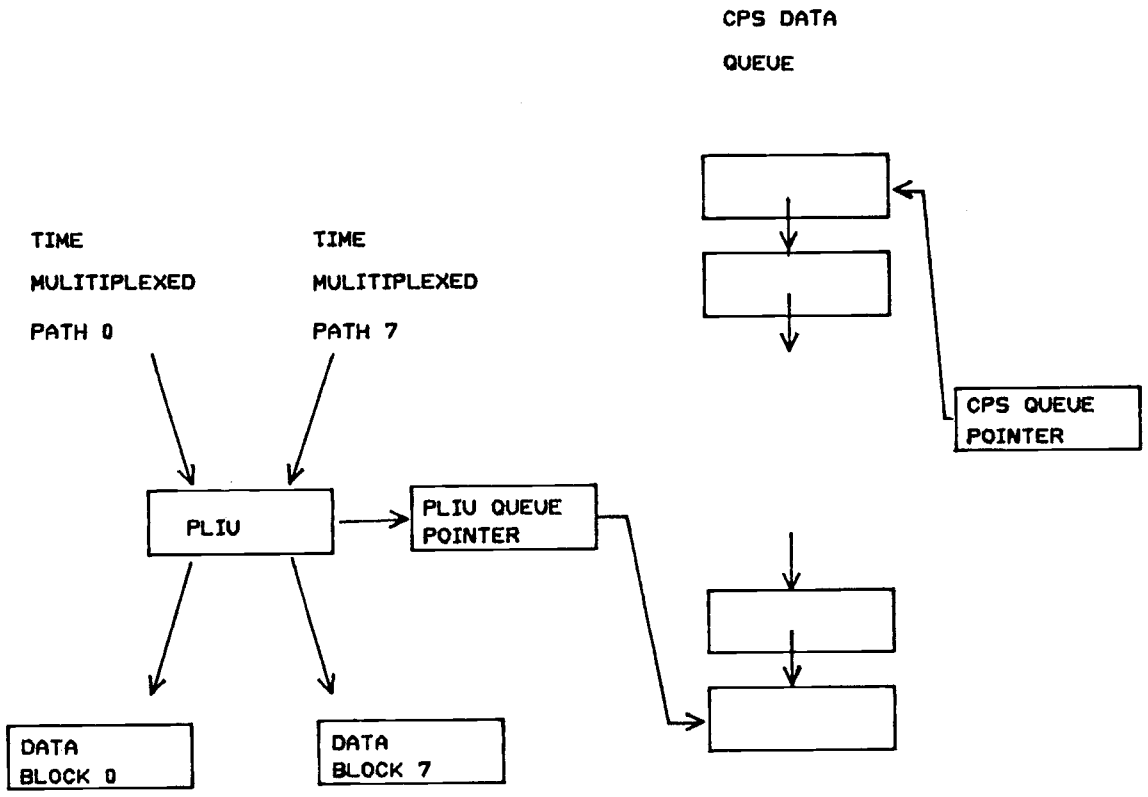


FIGURE 9. TIME-MULTIPLEXED INPUT DATA FLOW

Processing Time-Multiplexed Data

Figure 9 is a sketch of the data flow for time multiplexed input to the CPS. The period is 13 bytes long - one synchronous idle code and 12 channel of multiplexed data. The data in period one consists of several idle codes (233's), a break code (037), and control codes (235 and 267). This data stream is edited by the PLIU and stored in the two data blocks in the CPS memory as indicated. Time-multiplexed data from the remote terminals that pertains only to the HOST is placed in a separate block by the PLIU. This block can then be queued for transmission to the HOST without requiring any CPS editing. The block of control codes does require CPS analysis. This control information is stored in the separate block such that the CPS does not have the overhead of sifting through non-control information in order to get to the control information. In practice the probabilities of idles, data and control information is such that this processing is very effective.

Table 4 lists a more typical occurrence of code types in the multiplexed data stream. From this tabulation, it is evident that PLIU preprocessing of the data stream can significantly off-load the CPS and HOST via editing the data stream.

Figure 10 is a sketch of the data flow for a time-multiplexed output data stream from the CPS. As indicated in Figure 10, on each multiplexed period bytes of information are fetched from the byte queues. In the majority of cases, as in the input situation discussed previously, the duty cycle of remote terminals in active I/O state is typically 16%. Various techniques can be used to provide a low CPS overhead method of supplying the idles to the PLIU. One such technique is to have the CPS fill the linked byte queues with data and the PLIU fetch the data off the other end of the linked byte queues. In this manner the CPS only references the linked byte queues if data (non-idles) are to be queued.

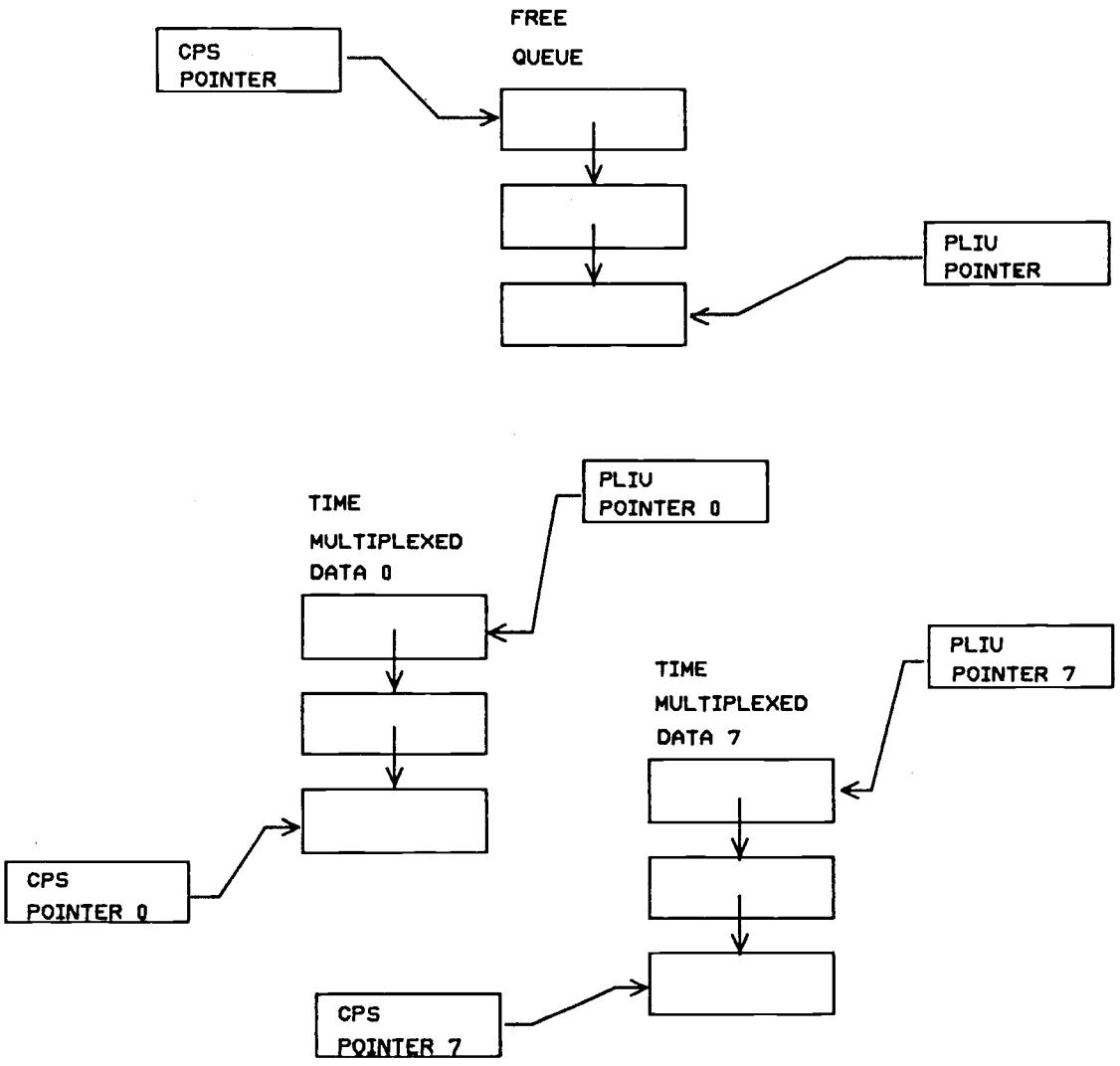


FIGURE 10. TIME-MULTIPLEXED OUTPUT DATA FLOW

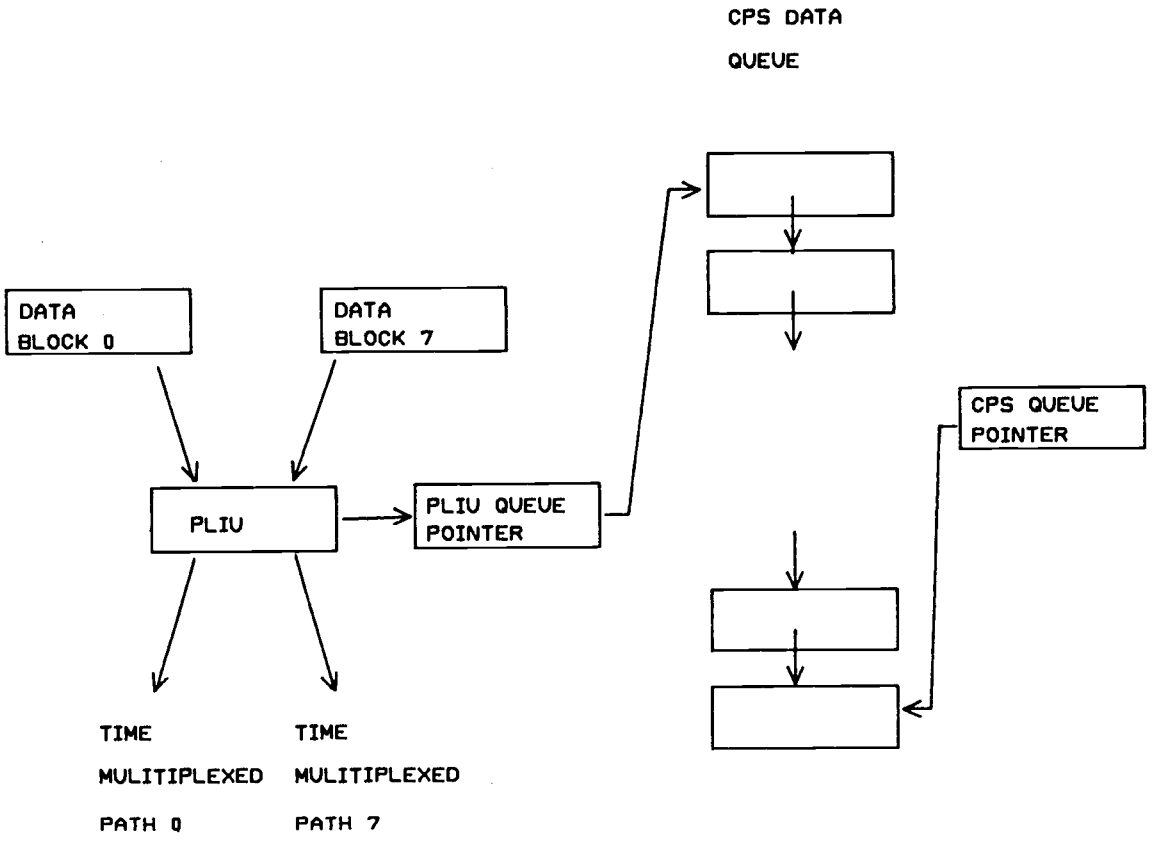


FIGURE 11. TIME-MULTIPLEXED OUTPUT DATA FLOW

Processing Transparent Text Data

Another telecommunication technique is transparent text transmission. Transparent text transmission capability presents some tedious processing overhead to the CPS, as in the case of the previously discussed time-multiplexed communications, since both input and output data byte streams have to be edited. The concept of transparent text transmission allows data codes to be transmitted which consists of all possible binary combinations with a minimum of control code overhead. The problem that arises is that at least one code is required for control purposes. A common format for transparent text transmission, see figure 12, involves a header, text body, and trailer. The header and trailer are non-transparent. The header consists of a sync sequence and addressing, and ends with a control sequence that indicates transparent text follows. Then transparent text is processed until a control sequence is detected that indicates non-transparent text mode exists and finally the trailer is processed. When in transparent text mode all occurrances of a predefined bit pattern are recoded into two such codes. The only existence of a single occurance of this code initiates an exit to non-transparent text mode in the PLIU receiver. The encoding and decoding

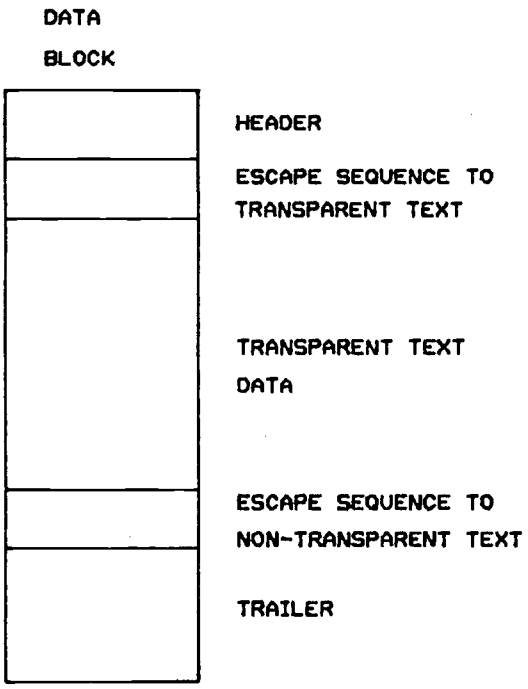


FIGURE 12. TRANSPARENT TEXT DATA FORMAT

required in transparent text transmission is another task that can readily be accomplished by the PLIU.

There are many other applications of the distributed PLIU concept in a CPS, such as communication description, protocal control, code compression/expansion, and code conversion. Quite often the situation exists that there is fixed and dynamic tabular data utilized in these and other applications. Of course, this requires a significant amount of memory space. This does not have to restrict these applications from the range of PLIU's. Techniques can be utilized such that the PLIU's have access to common memory. For example, the PLIU's can have access to the CPS main memory. In some applications, the PLIU's would require only read-only access. It is important for integrity purposes to restrict areas of access as well as write access if the applications allow.

V. SUMMARY AND CONCLUSIONS

Computing utilities can best be achieved by multiple computer structures because of the service continuity requirements as well as because of cost/performance considerations. The overall CPS structure discussed herein is one with a hierarchial and parallel structure that effectively accomplishes digital communications processing functions.

BIBLIOGRAPHY

1. J. W. Meeker, et al, OS-3: The Oregon State Open Shop Operating System, AFIPS Proceedings, Vol. 34, 1969, pp 241-248.

2. Intel Corporation Universal Synchronous Receiver/ Transmitter, 8251, specifications.

3. Texas Instruments Inc. TTL Data Book

4. Electronic Industries Association RS-232C Standard.

5. Motorola M6800 Microprocessor Applications Manual.

APPENDIX A

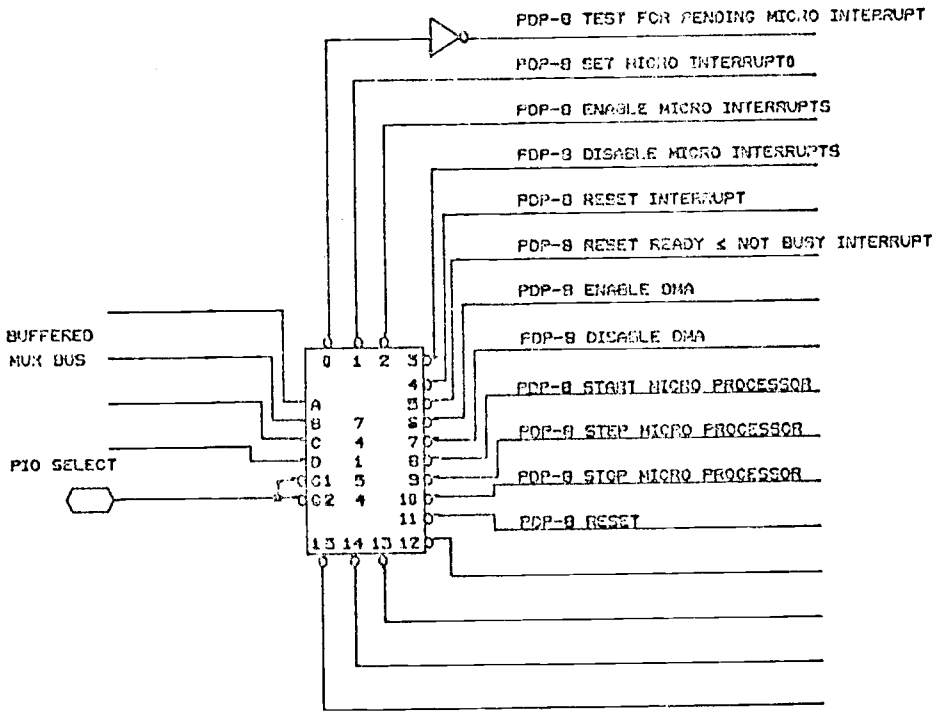


FIGURE 13. PDP-8 PIO DECODE

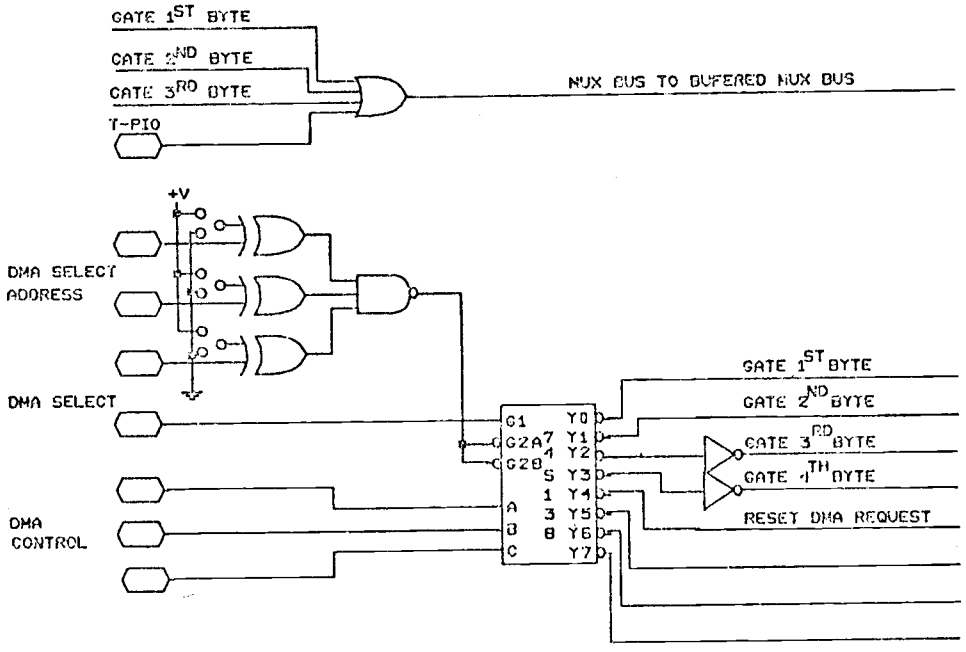


FIGURE 14. DMA DECODE

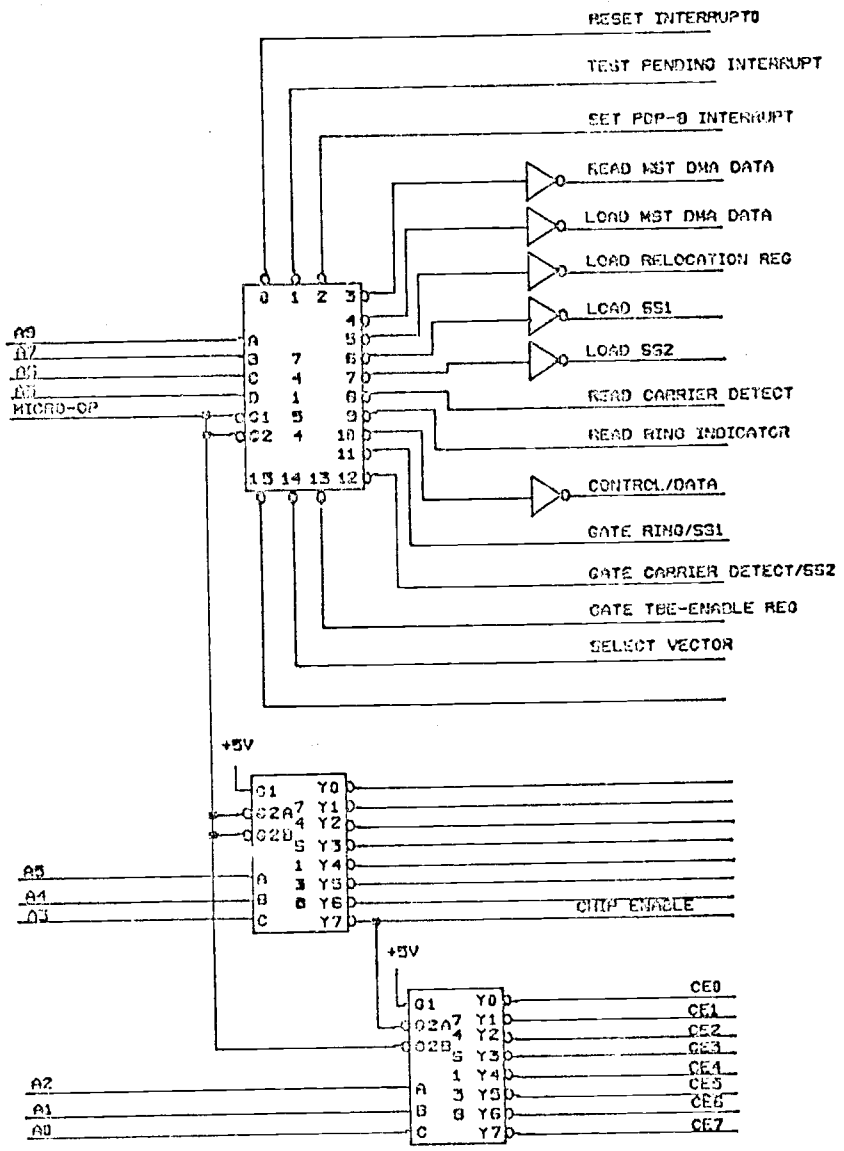


FIGURE 15. MICRO OPERATION DECODE

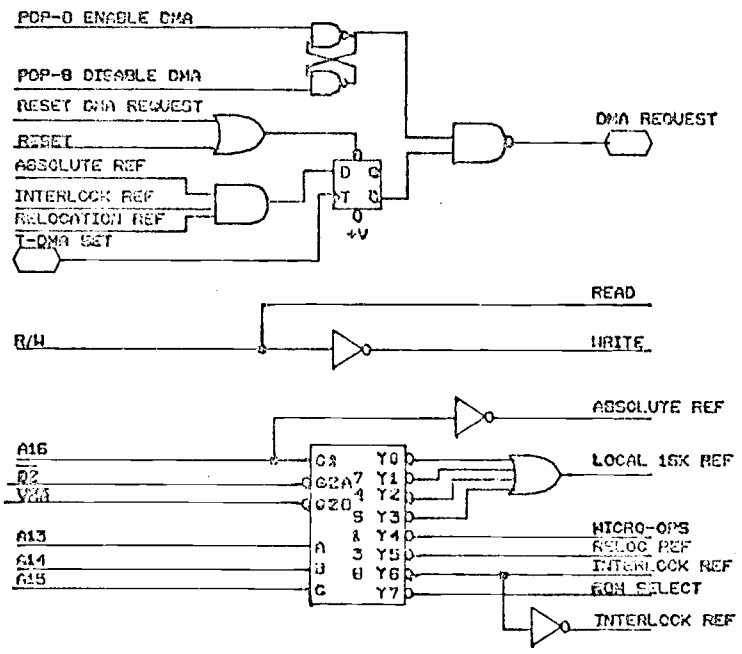


FIGURE 16. ADDRESS DECODE

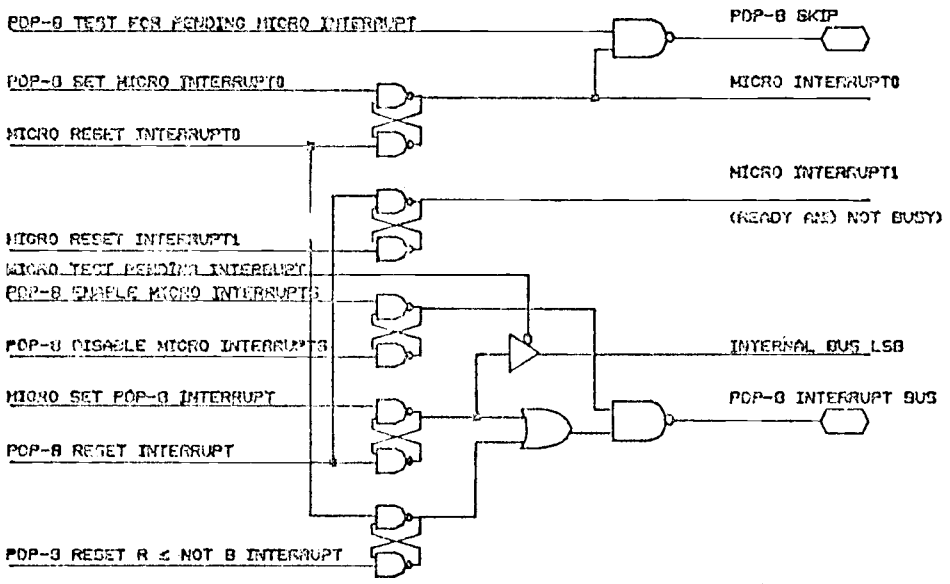


FIGURE 17. STATUS BITS

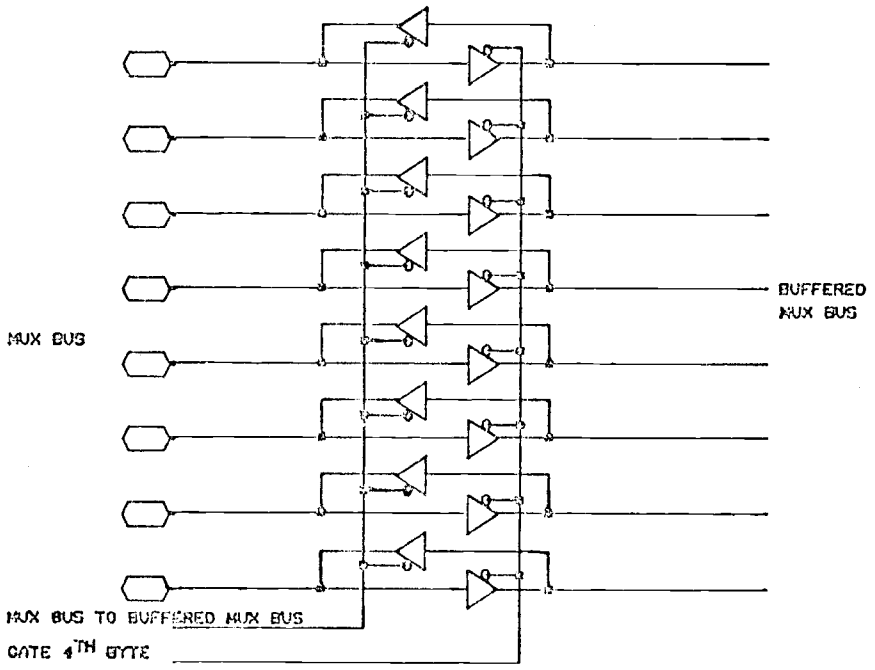


FIGURE 18. BUS BUFFER

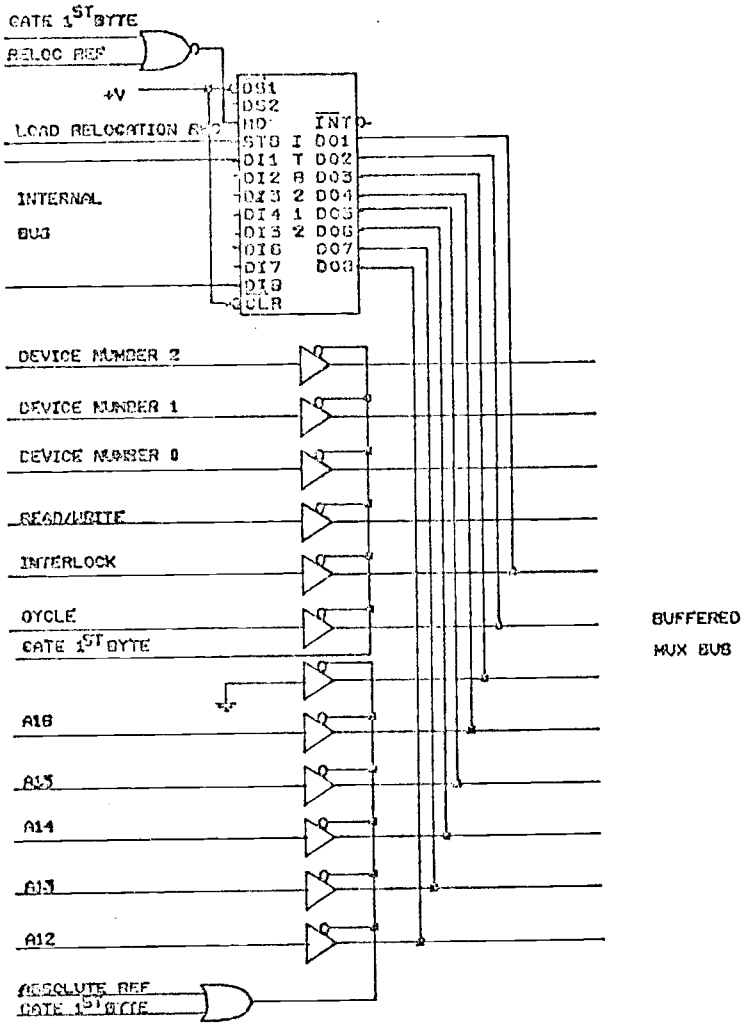


FIGURE 19. DMA CONTROL MULTIPLEXER

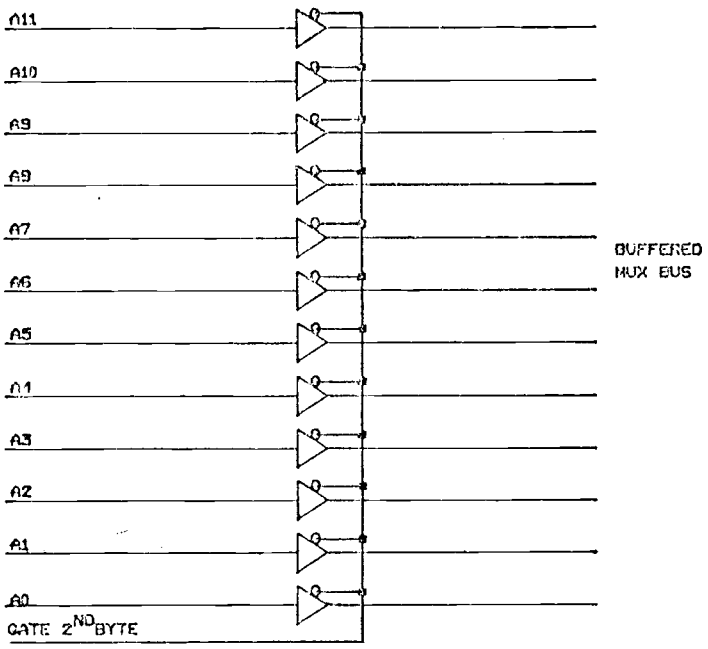


FIGURE 20. DMA ADDRESS CONTROL MULTIPLEXER

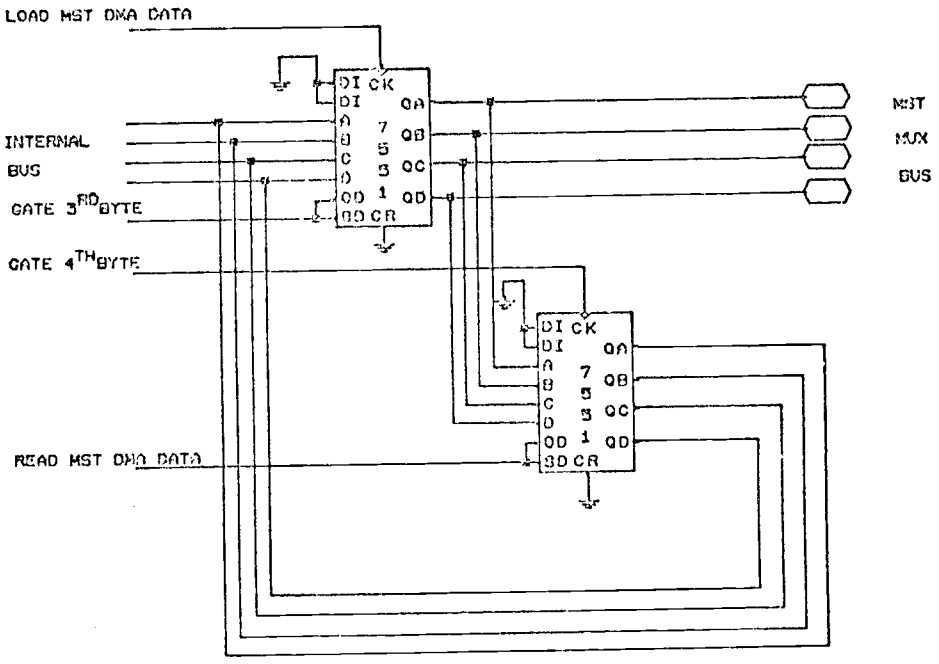


FIGURE 21. DMA DATA BUFFER

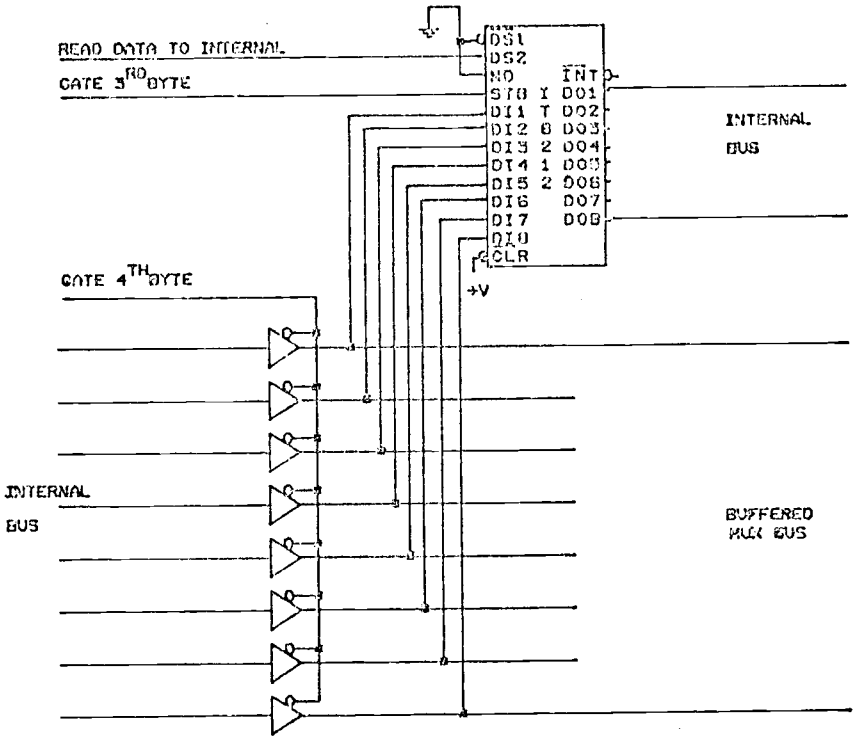


FIGURE 22. DMA DATA BUFFER

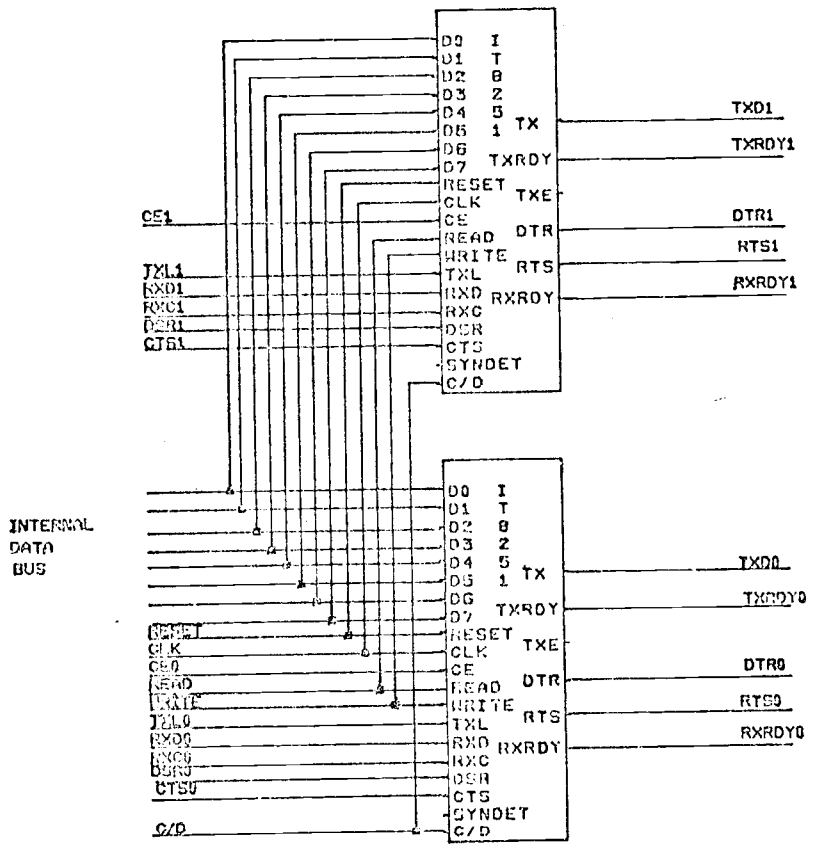


FIGURE 23. UNIVERSAL RECEIVER TRANSMITTER

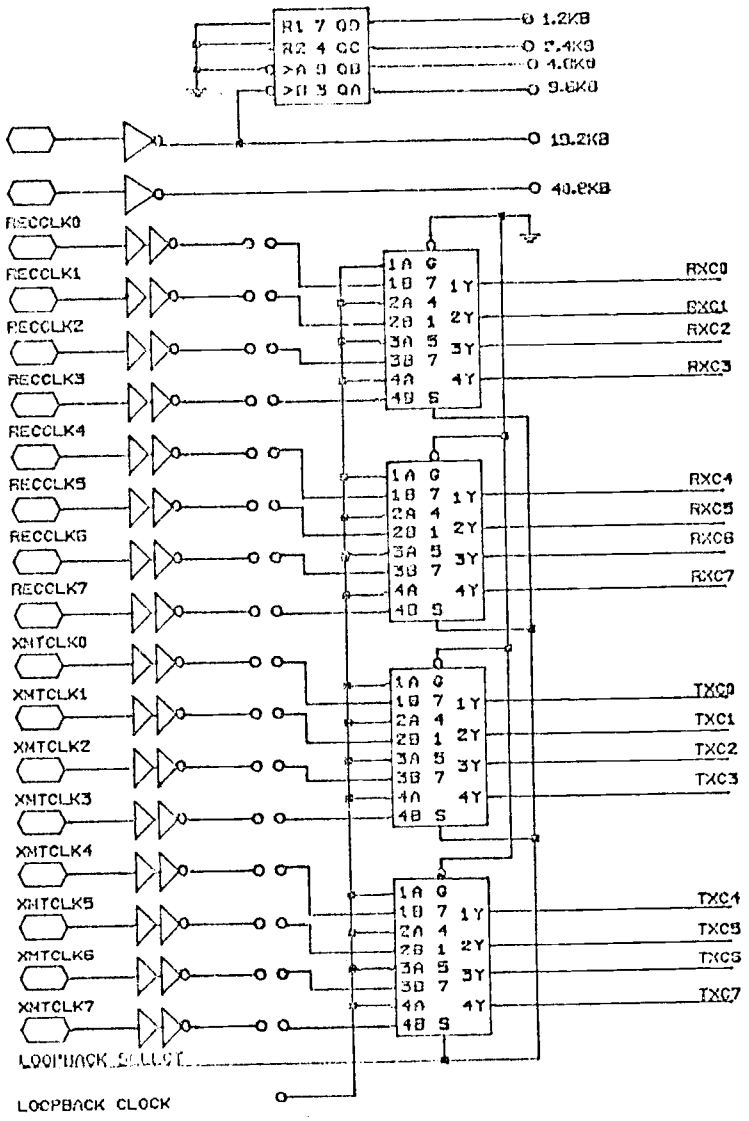


FIGURE 24. URT CLOCK LOGIC

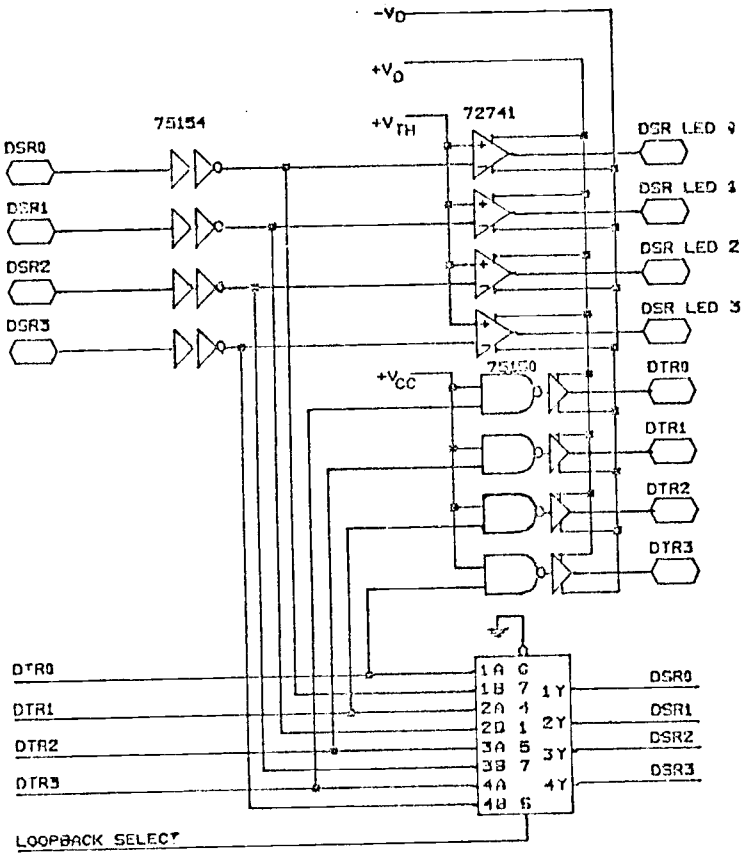


FIGURE 25. DATA SET READY AND DATA TERMINAL READY

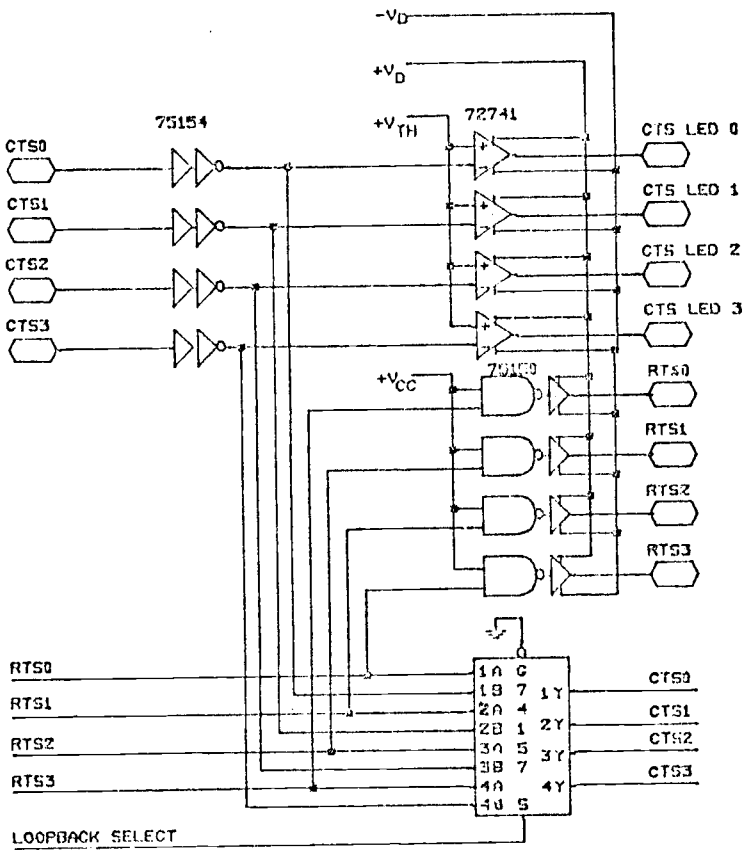


FIGURE 26. CLEAR AND REQUEST TO SEND

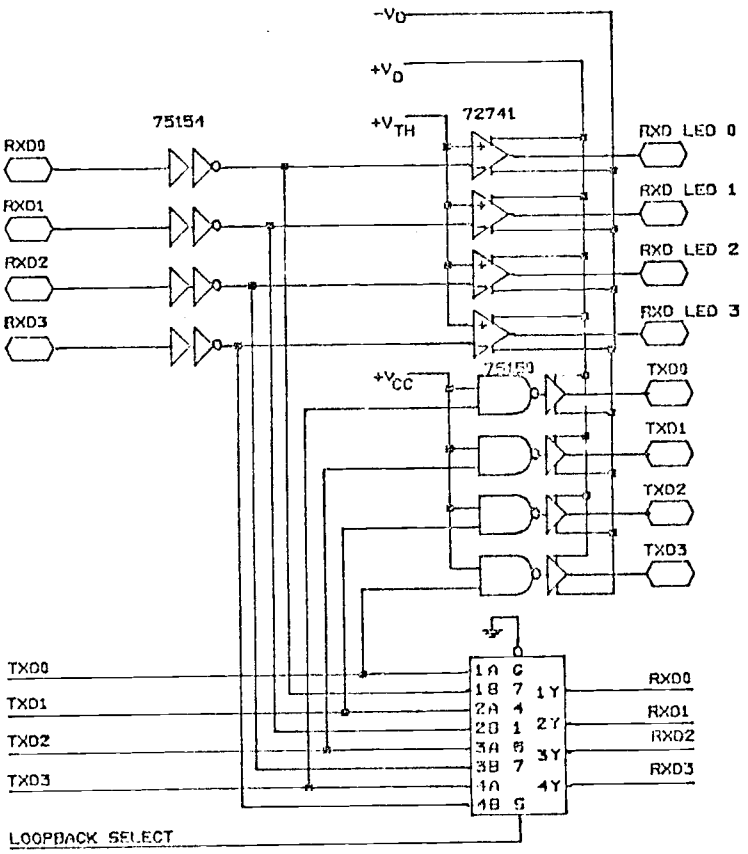


FIGURE 27. RECEIVE AND TRANSMIT DATA

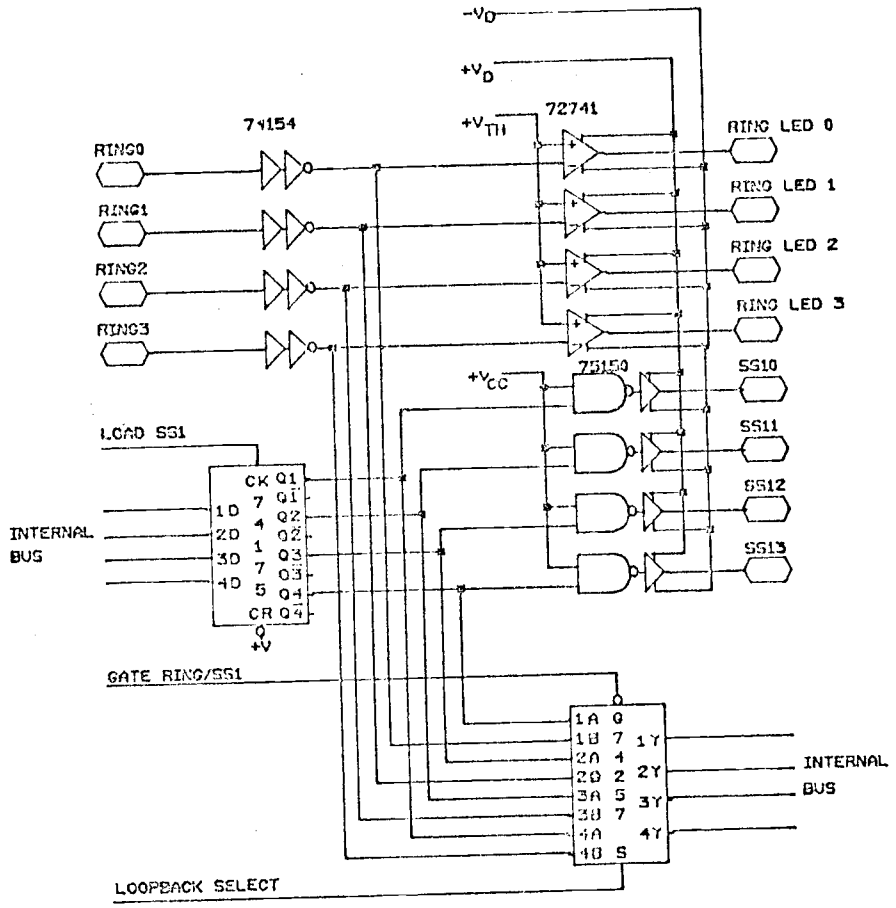


FIGURE 28. RING AND SUPERVISORY

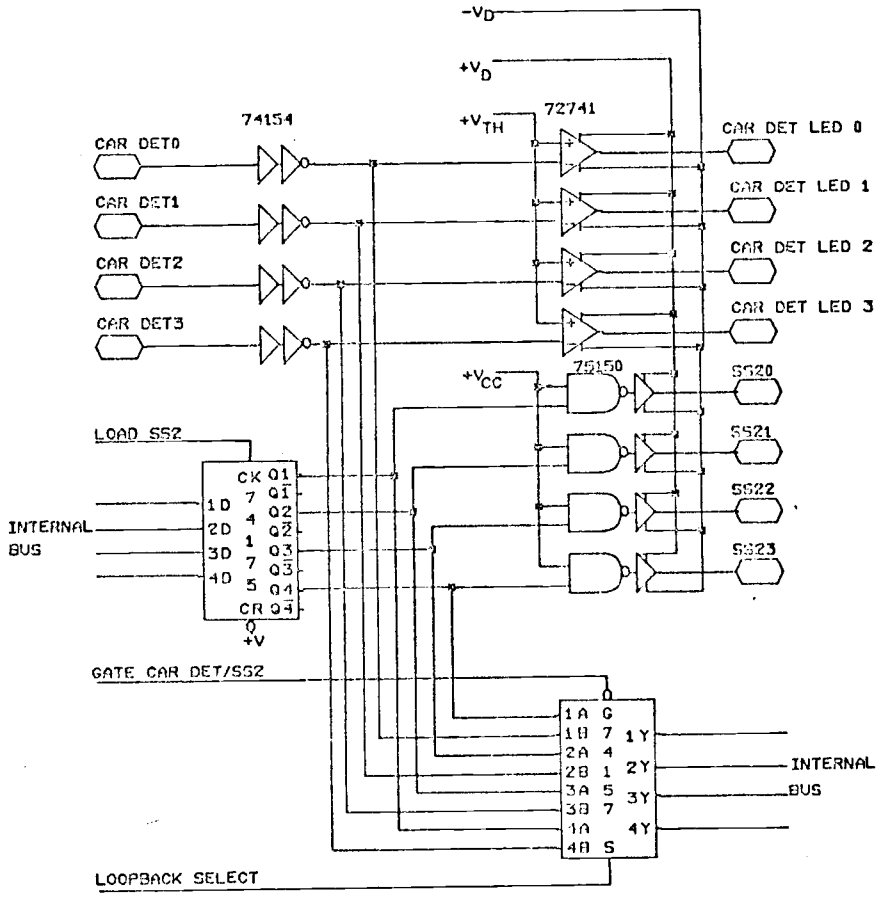


FIGURE 29. CARRIER AND SUPERVISORY

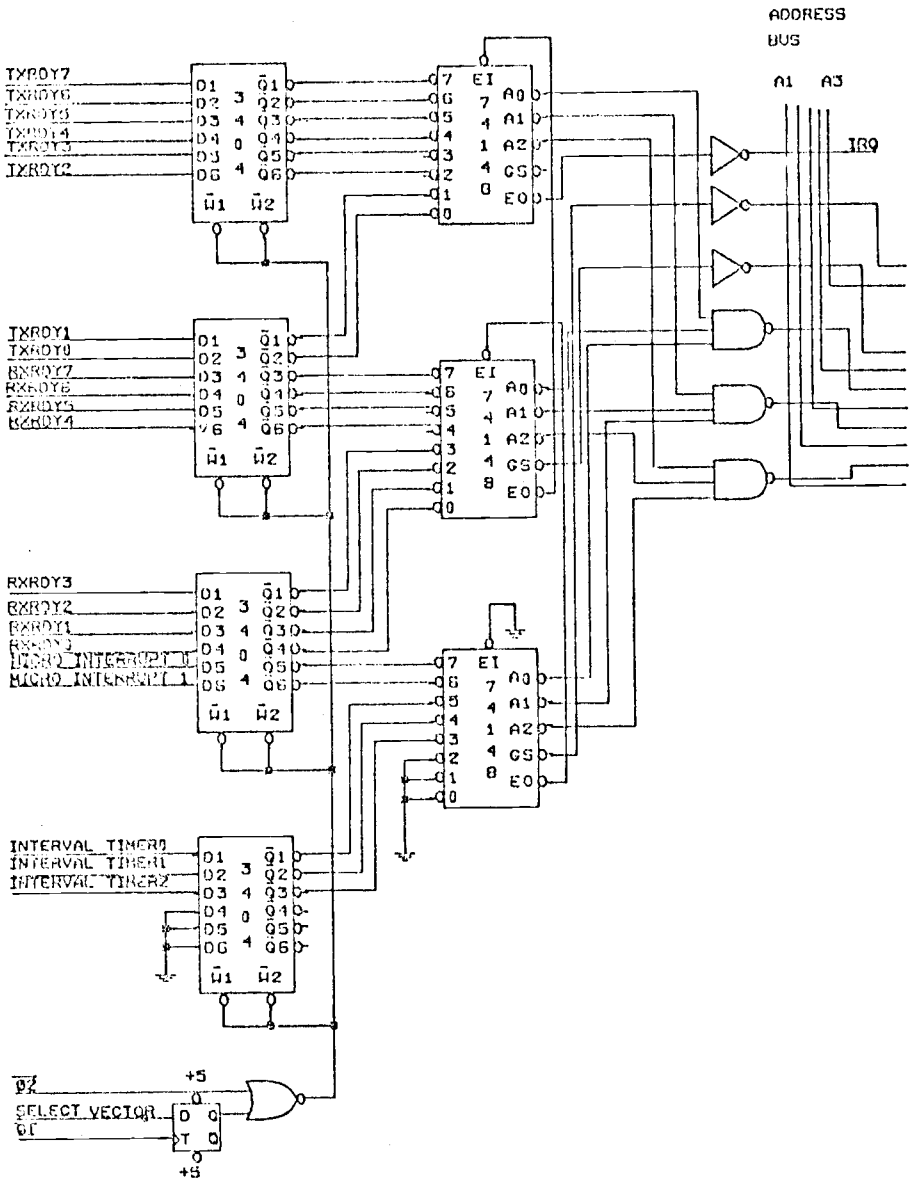


FIGURE 30. INTERRUPT LOGIC

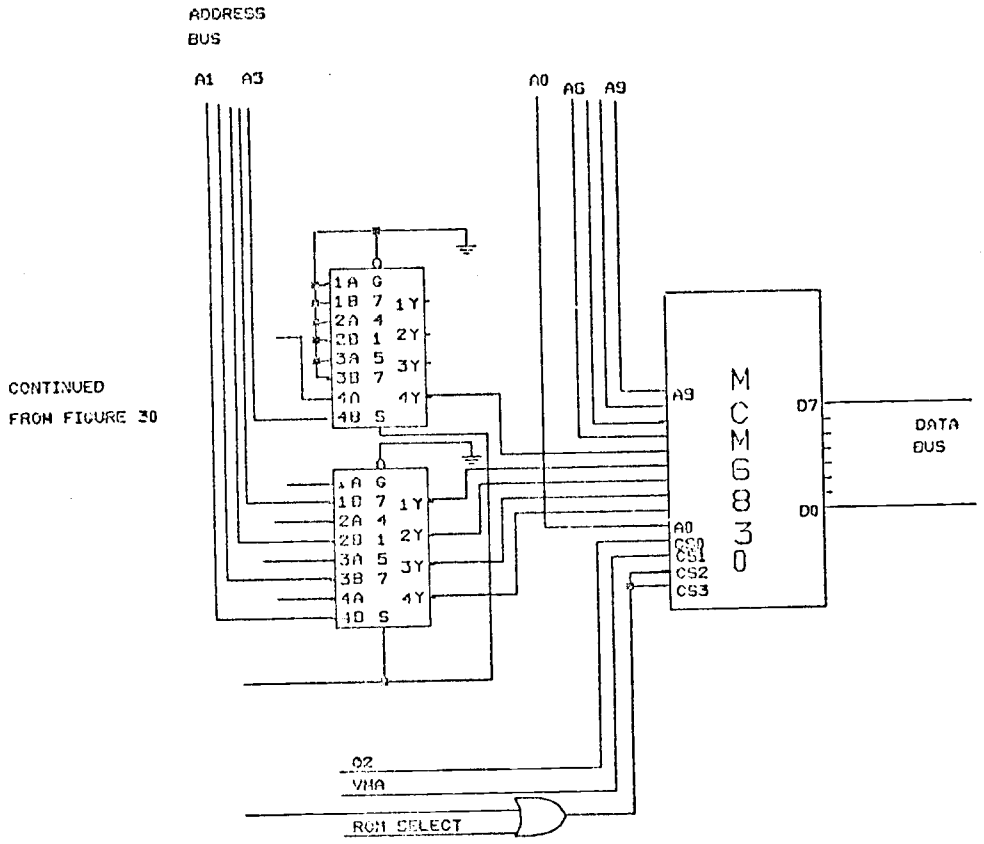


FIGURE 31. INTERRUPT LOGIC

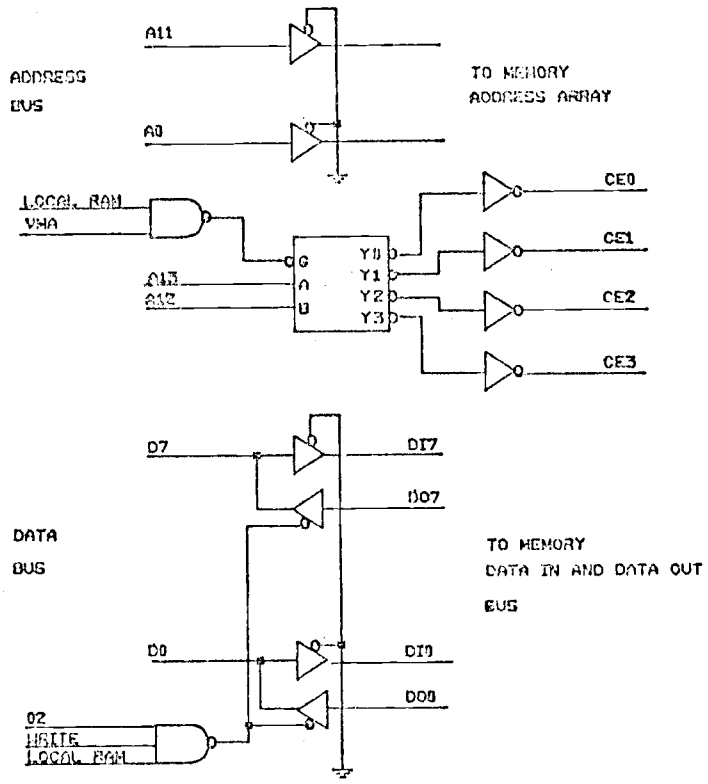


FIGURE 32. LOCAL MEMORY LOGIC

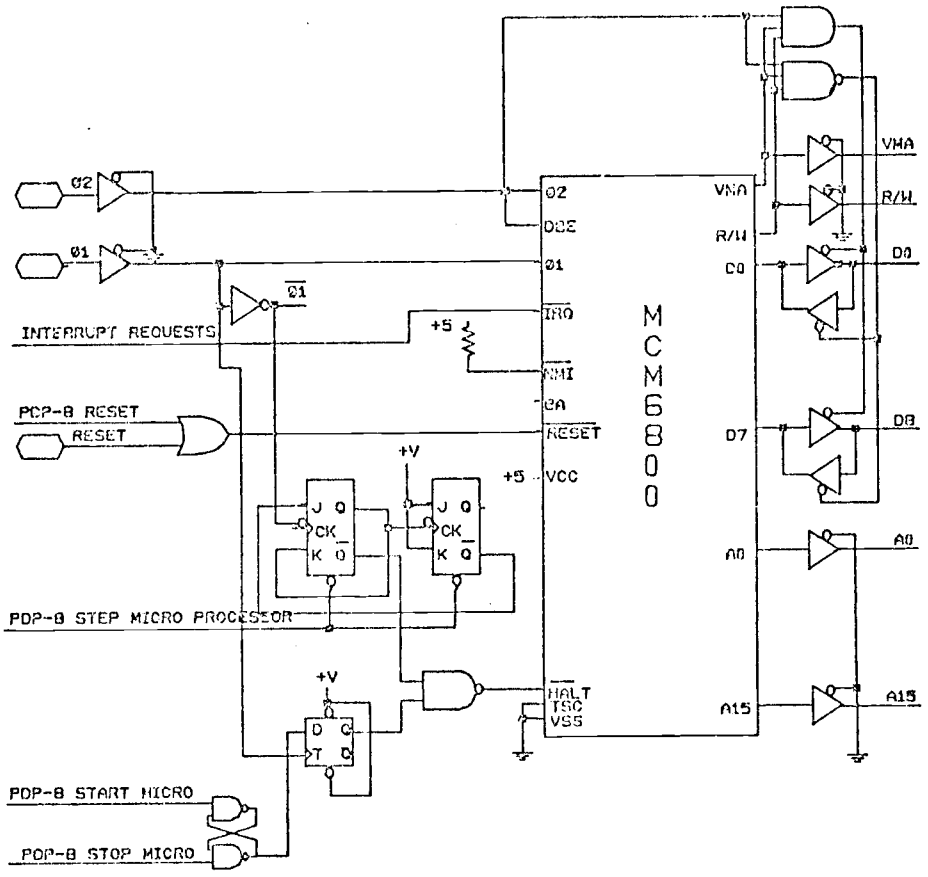
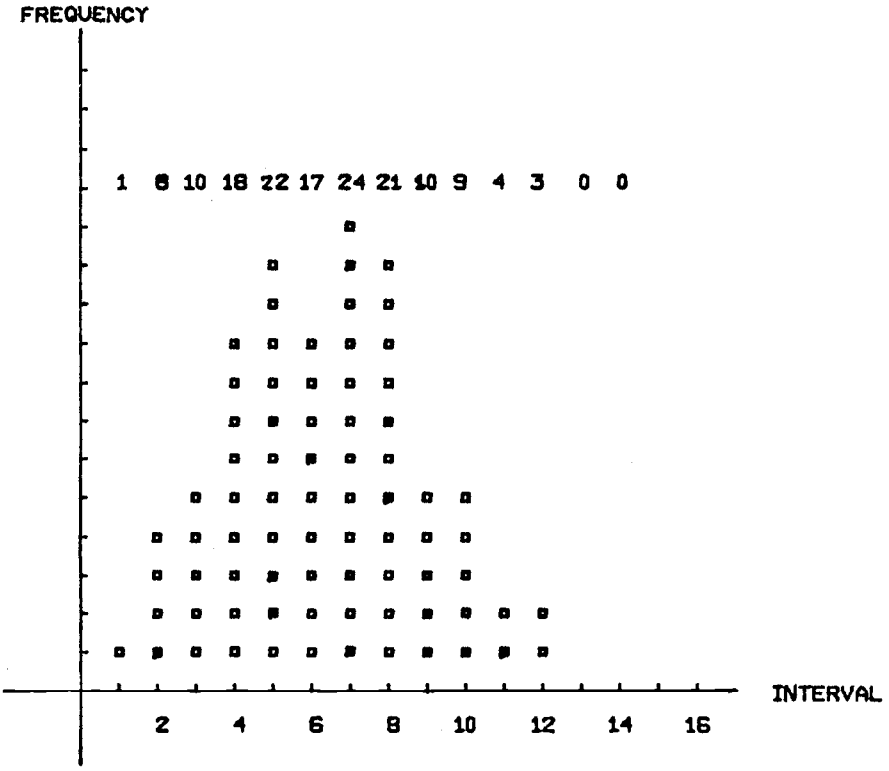


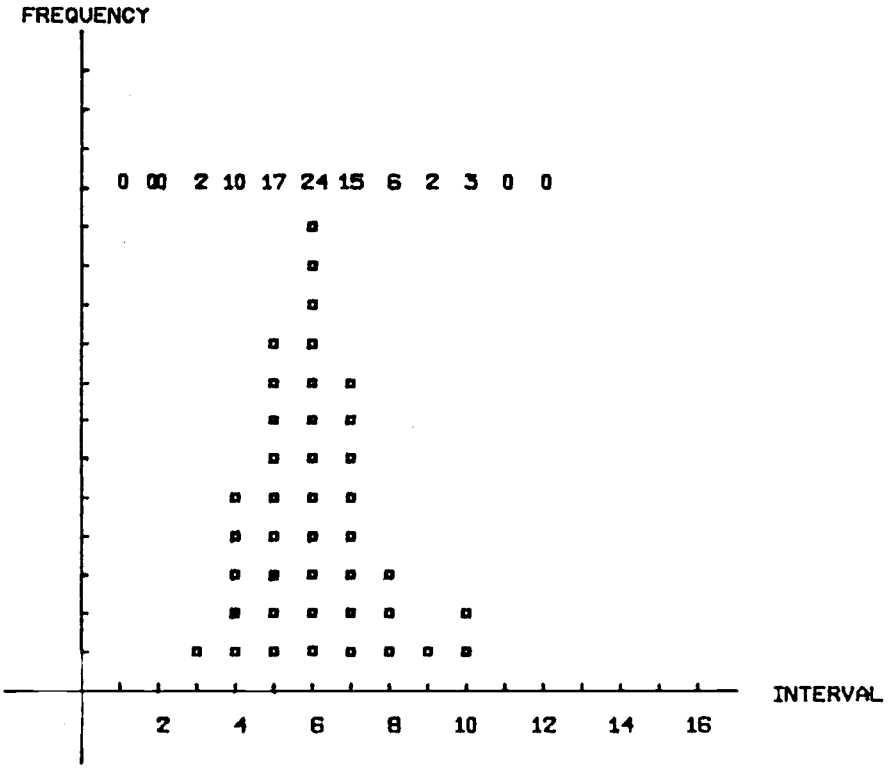
FIGURE 33. MICROPROCESSOR LOGIC

APPENDIX B

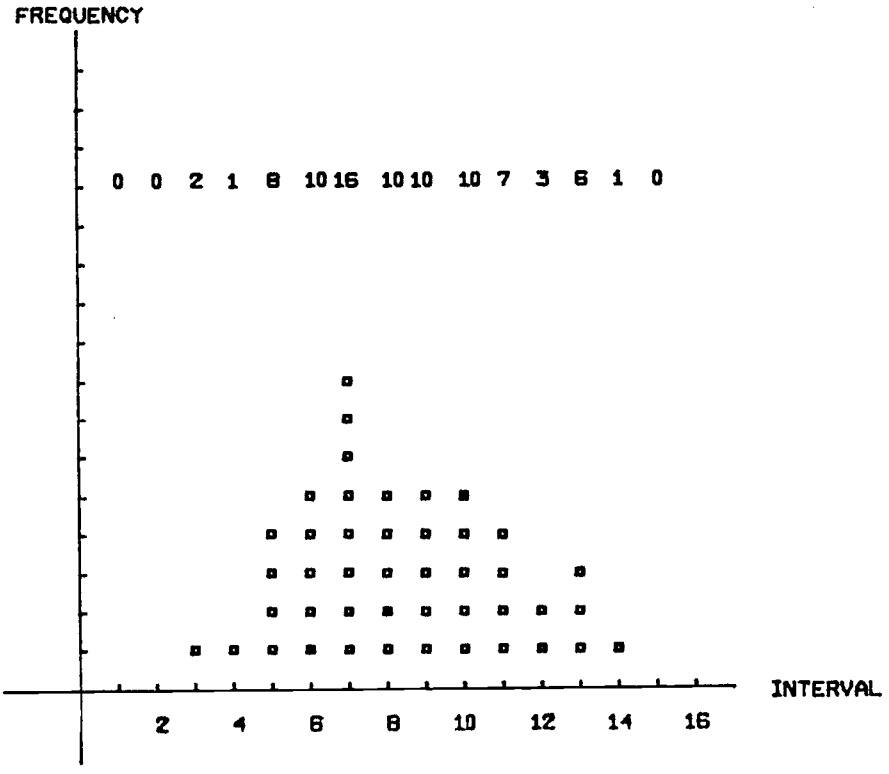
HISTOGRAM


FREQUENCY
1 6 10 18 22 17 24 21 10 9 4 3 0 0
INTERVAL
2 4 6 8 10 12 14 16

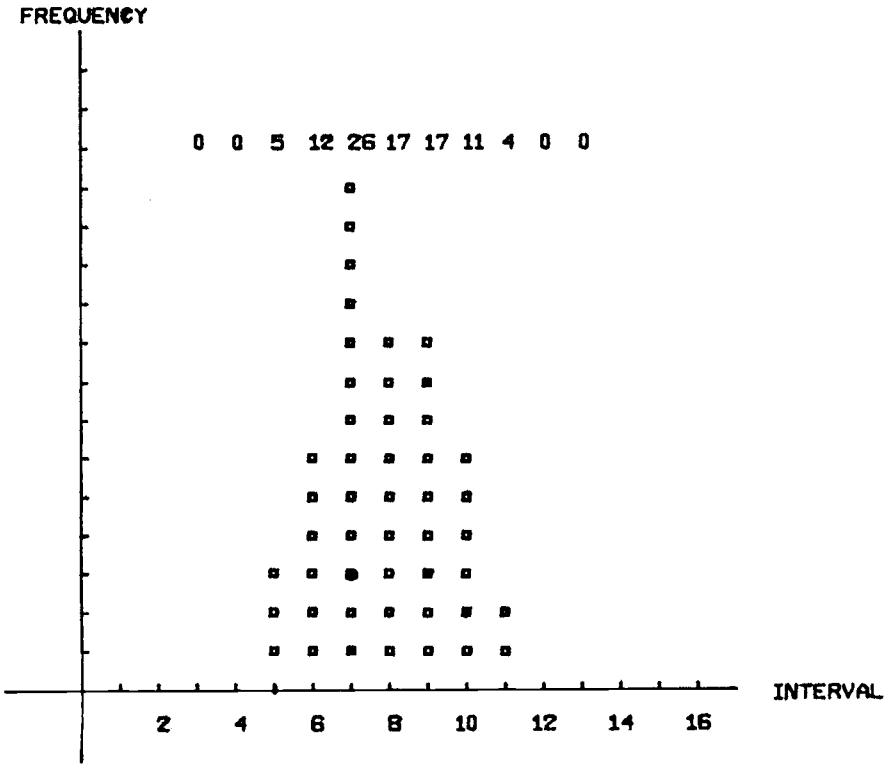
HISTOGRAM



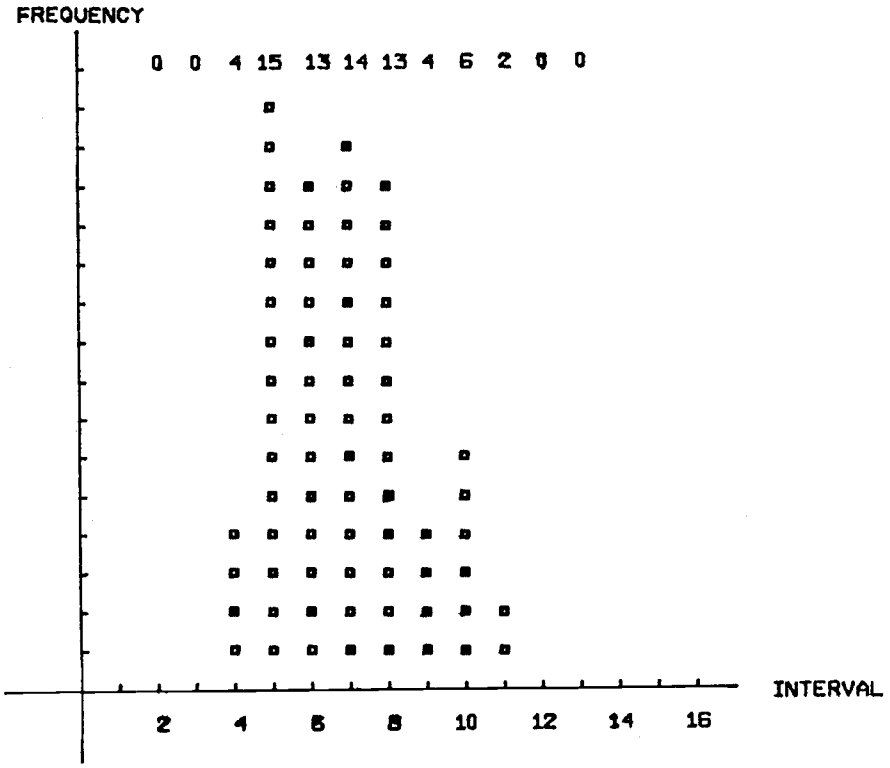
HISTOGRAM



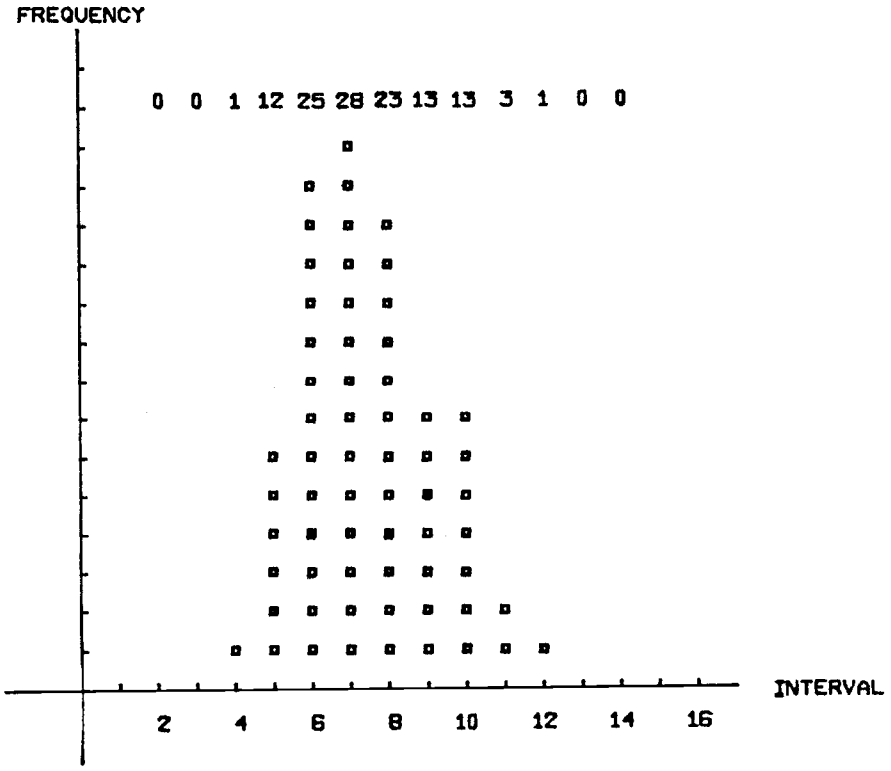
HISTOGRAM



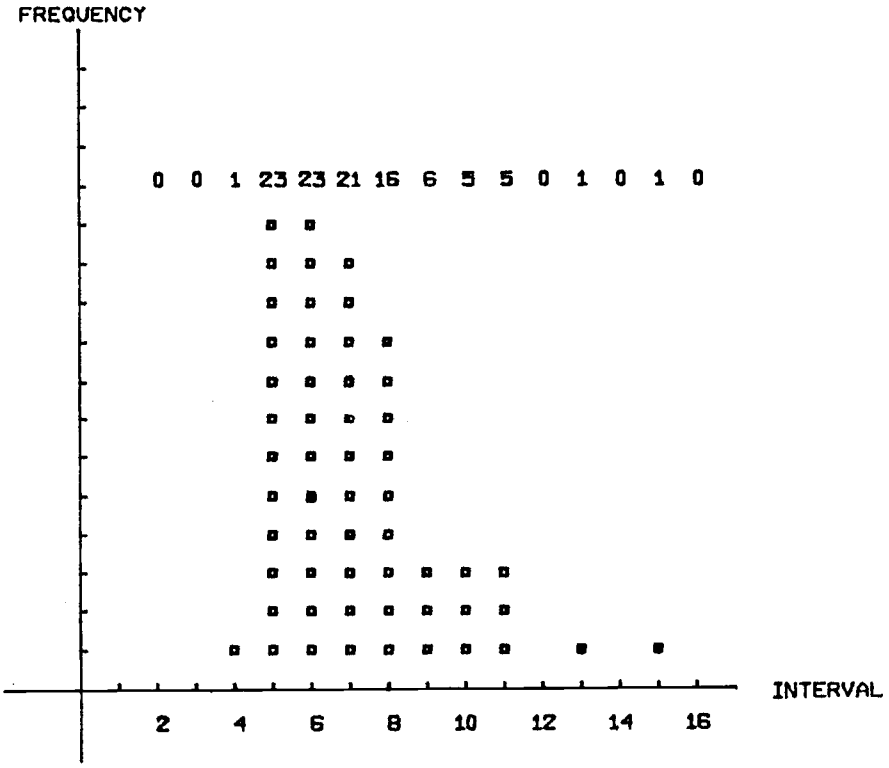
HISTOGRAM



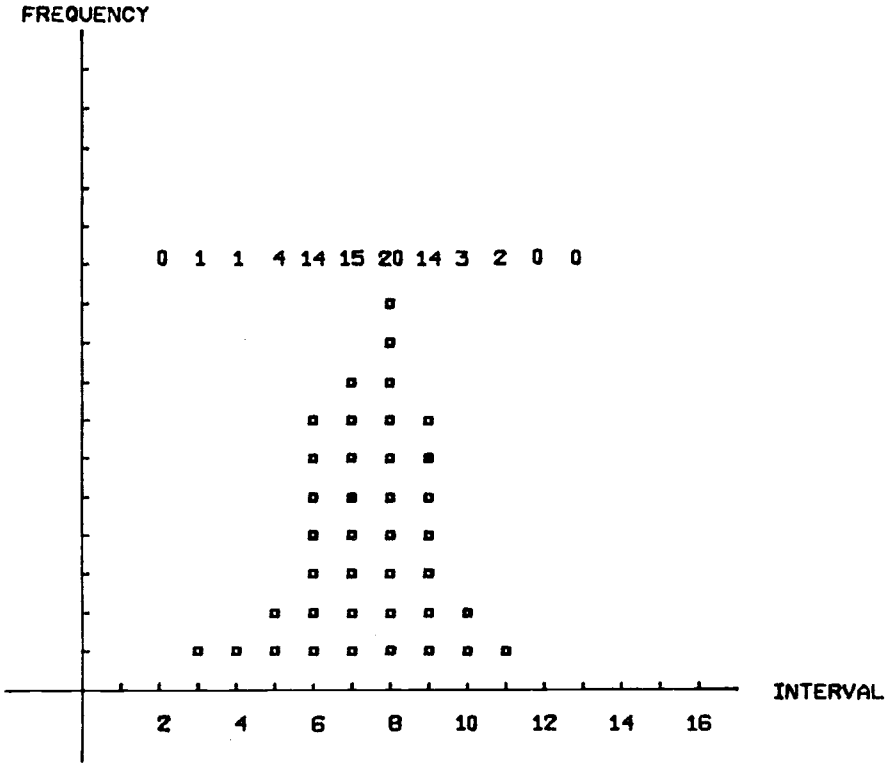
HISTOGRAM



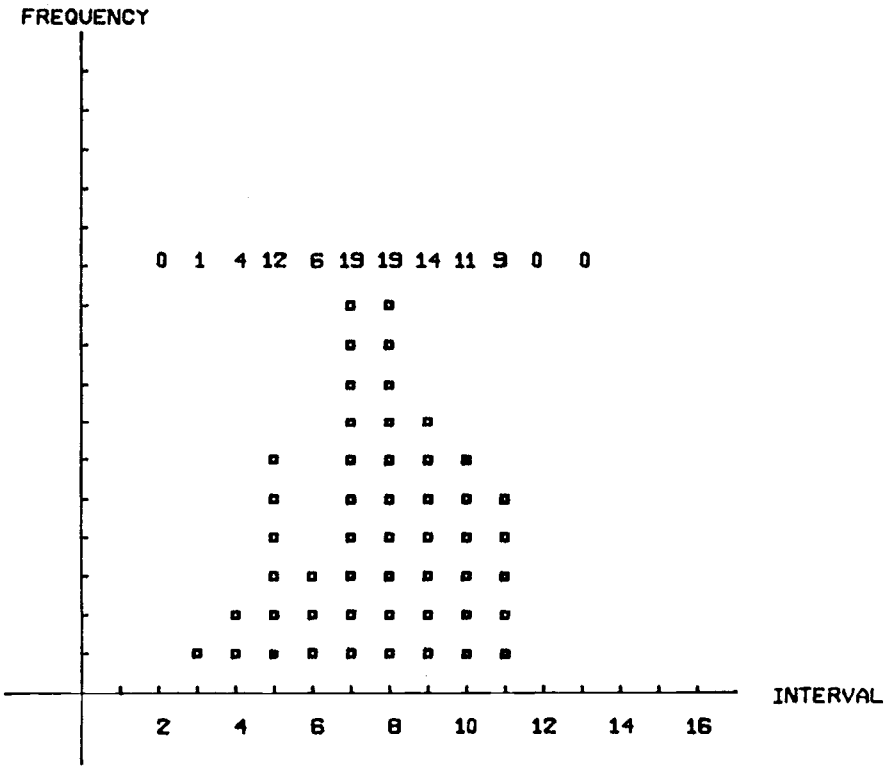
HISTOGRAM



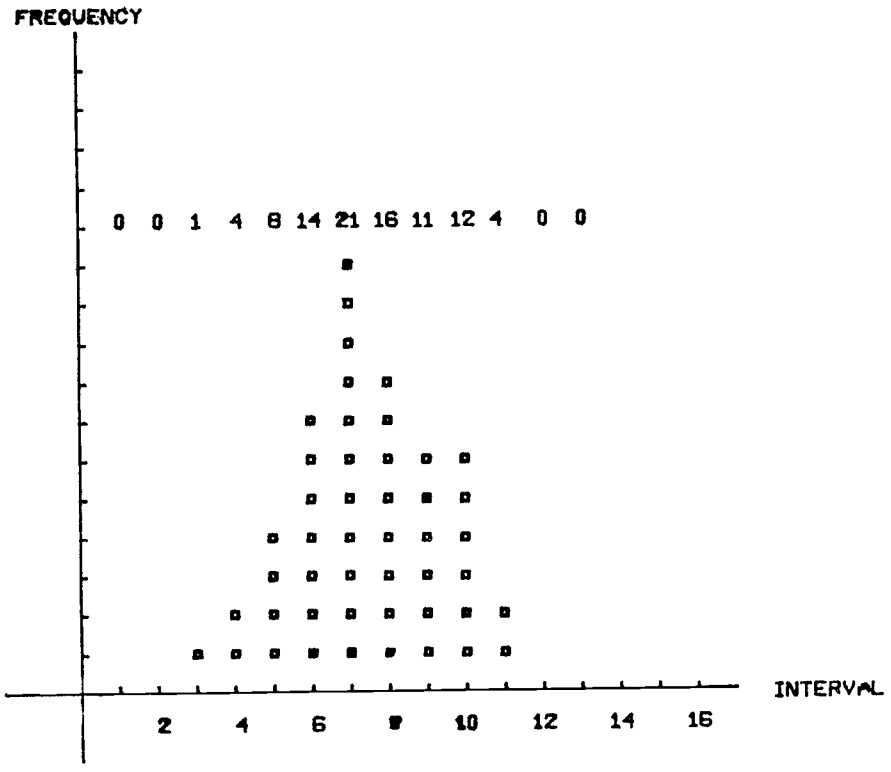
HISTOGRAM



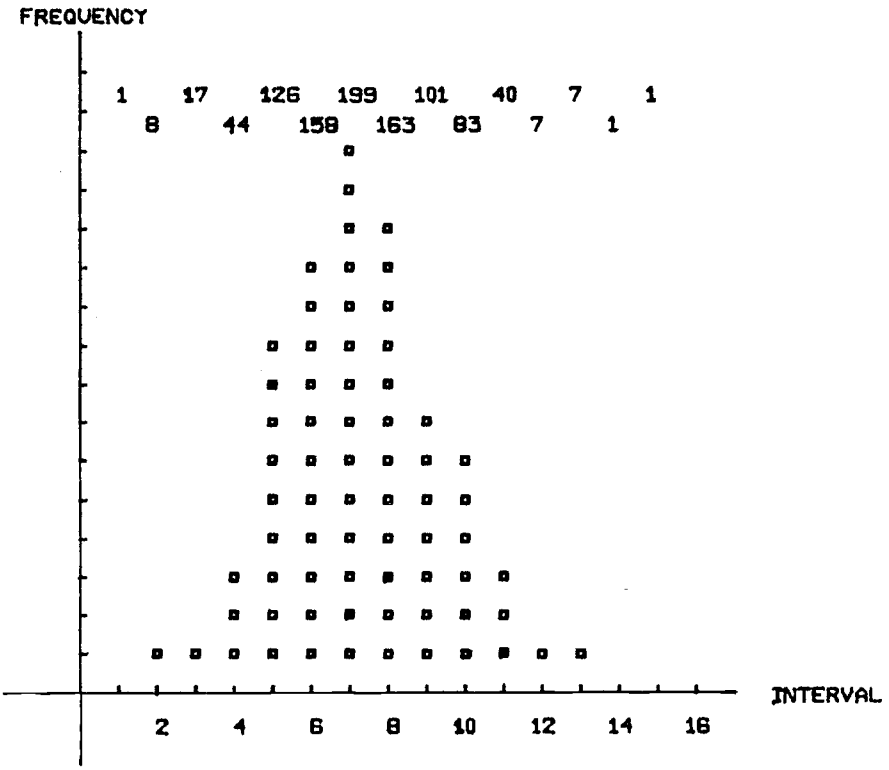
HISTOGRAM



HISTOGRAM



COMPOSITE HISTOGRAM



VARIABLE DESCRIPTION TABLE

Sample Size	956
Sum	5946.00000
Raw Sum of Squares	40996.00000
Corrected Sum of Squares	4013.87030
Average	6.21967
Standard Error of Mean	.06631
Median	6.00000
Maximum Value	14.00000
Minimum Value	0
Sample Variance	4.20301
Sample Standard Deviation	2.05012
Coefficient of Variation	.32962
Range	14.00000
Skewness	.20679
Kurtosis	3.11009