p. 107

NASA Technical Memorandum 86411

BUBBLE MEMORY MODULE FOR SPACECRAFT APPLICATION

P. J. Hayes, Karen T. Looney, and C. D. Nichols

APRIL 1985

(NASA-TM-86411) BUBBLE MEMORY MODULE FOR SPACECRAFT APPLICATION (NASA) 107 p CSCL 09A
N87-20476
Unclas
G3/33 45321

Date for general release April 30, 1987

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

CONTENTS

BUBBLE MEMORY MODULE FOR SPACECRAFT

SUMMARY

Bubble domain technology offers an all-solid-state alternative for data storage in onboard data systems. A versatile modular bubble memory concept has been developed. The key module is the bubble memory module which contains all of the storage devices and circuitry for accessing these devices. This report describes the functional design of the bubble memory module, with particular emphasis on the dual bus interface. Hardware designs which could be used to demonstrate system performance are also discussed.

INTRODUCTION

Most NASA aerospace missions require nonvolatile, high-density, onboard memory to provide data and program storage capability. In the past, tape recorders have been the only alternative to provide this memory function. Tape recorders, however, have a history of reliability problems associated with moving parts and impose undesirable system functional restrictions.

An all-solid-state memory with no moving parts is needed to provide improvement in both reliability and functional performance. Anticipated benefits of higher reliability are reduction in expensive periodic maintenance, elimination of the occasional need to adapt to manufacturer's new tape formulations, and greatly reduced probability of catastrophic loss of the whole memory. Improved functional characteristics include first-in/first-out data sequencing, multifunctional fast block access and serial memory capability, simultaneous read/write, asynchronous data rates, fast command reconfiguration for memory setup, partitioning flexibility, and expandable capacity.

Recognizing the need for an all-solid-state alternative, NASA has been developing bubble domain technology. Significant progress has been realized in conventional permalloy gap devices (refs. 1-3), wide gap and ion implant devices (ref. 4), self-structured devices (refs. 5-6), and memory system development (refs. 7-10). A system concept and bubble memory module functional design which is potentially applicable to all of the device types has been developed. This report documents the bubble memory module design and preliminary hardware designs aimed at memory module functional demonstration with available commercial bubble devices.

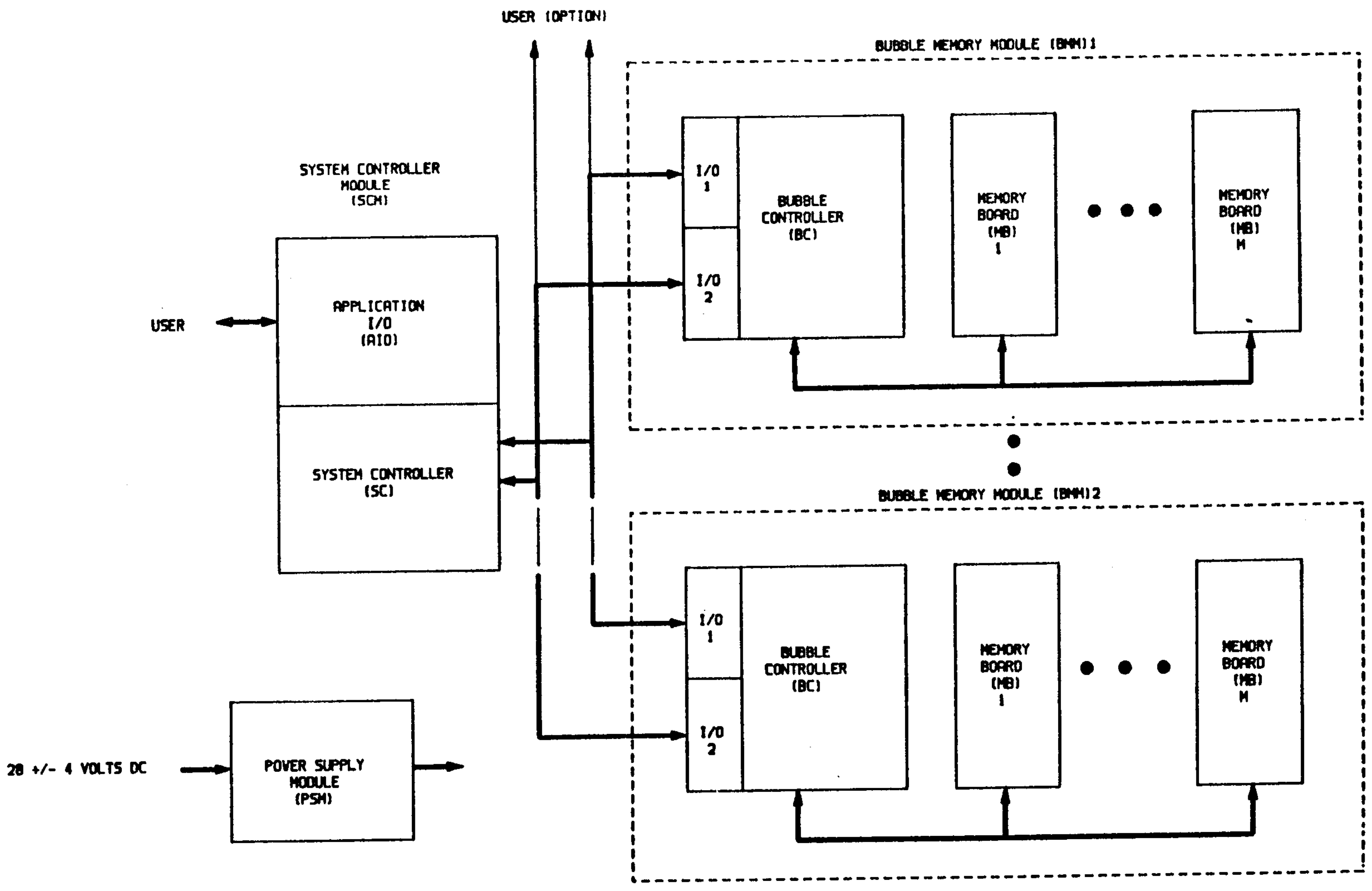
The system architecture provides simultaneous operation of bubble devices to attain high data rates. Banks of bubble devices are accessed by a given bubble controller to minimize controller parts. A power strobing technique is discussed which could minimize the average system power dissipation. A fast initialization method using EEPROM devices promotes fast access. Noise and crosstalk problems and implementations to minimize these are discussed.

The use of brand or trade names herein does not necessarily imply NASA endorsement.

MODULAR BUBBLE MEMORY SYSTEM CONCEPT

A modular bubble memory system organization which could satisfy most onboard applications is depicted by the block diagram of Figure 1. Major modules are the Bubble Memory Module (BMM), the System Controller Module(SCM), and the Power Supply Module (PSM). The minimum system consists of one PSM and one BMM. The need for the SCM is user-dependent and will not be required in some applications; however, in most large capacity memory systems an SCM will be required. The Application I/O (AIO) tailors the memory system to the user's specific interface and provides serial to parallel data conversion where needed. Different AIO designs and reprogramming of the SCM should provide unlimited flexibility for future applications, within the performance realm of the BMM. A dual bus interface (each having identical address, data, and control lines) provides access to the BMM's. This interface is microprocessor compatible, thus permitting a user with appropriate control electronics to integrate the BMM directly into an experiment without a specific SCM.

The dual bus interface enables two BMM's to be accessed simultaneously at twice the BMM data rate (assuming an adequately sized PSM) and provides a degree of redundancy in high reliability applications. Figure 2 shows some possible combinations of bubble controllers and memory boards possible in various system sizes. The solid lines represent constant system size and a fixed percentage of capacity lost should a memory board fail. The dashed lines indicate the number of bubble controllers (BC) per system and the corresponding percentage of system capacity lost should a bubble controller fail. The concept of graceful failure is illustrated in this Figure. Specific choices of the number of BC's and the number of MB's



BUBBLE MEMORY SYSTEM BLOCK DIAGRAM
FIGURE 1

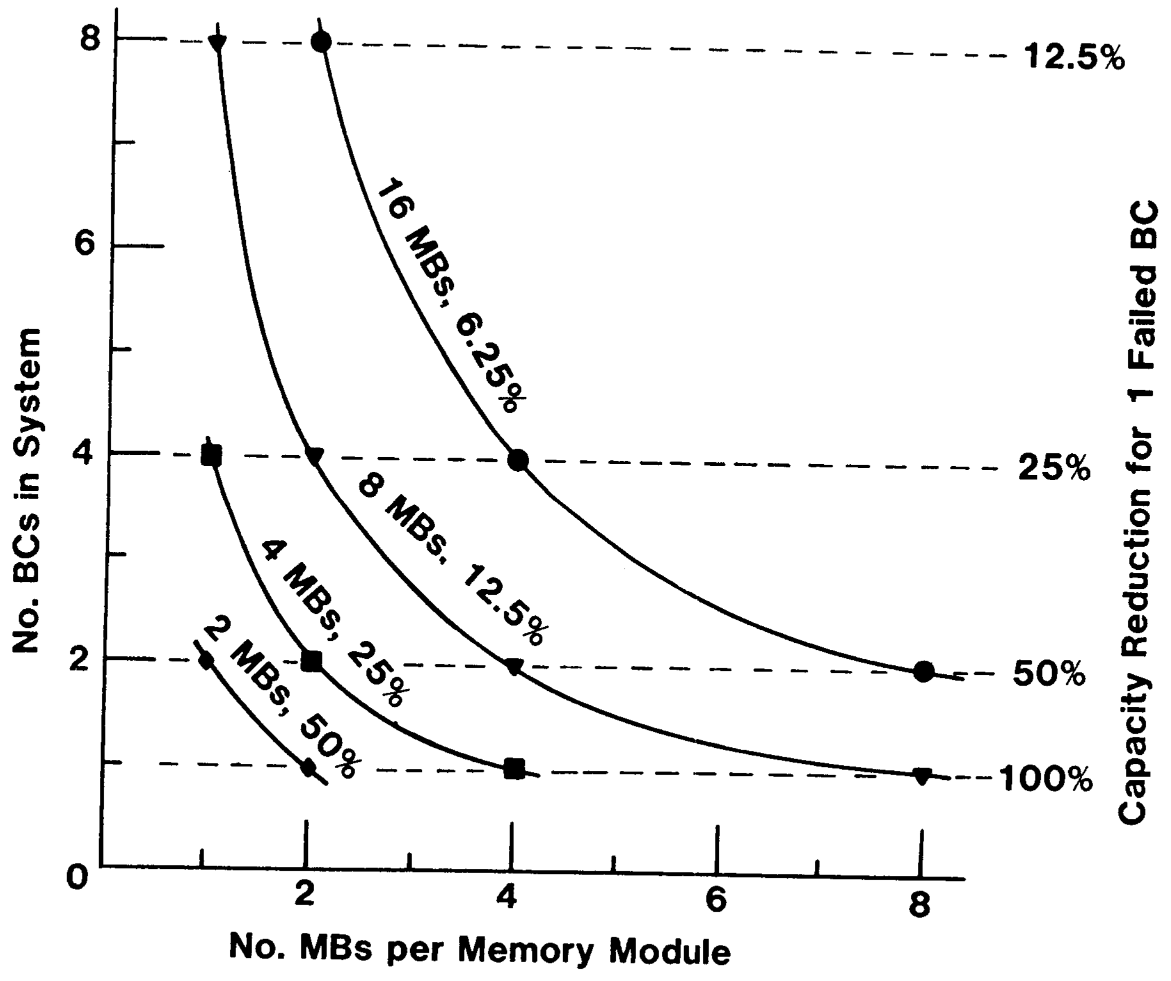


Illustration of various system configurations and graceful failure capability. Values on the curves indicate system size in number of memory boards (MB) and the percentage of system capacity lost for each failed MB. (BC = bubble controller)

FIGURE 2

per BMM enables substantially increased lifetime probability for applications such as those requiring redundant write-protected programs for significantly long missions.

Typically, only one BMM and one MB are powered on at any given time to conserve power. The SCM provides the switching for BMM selection and routes data to and from the BMM's. The SCM is programmable to organize the BMM or Memory Boards (MB) into user channels or blocks of memory. Since this system concept is intended for block and contiguous data access, any large buffers would be located within the SCM. Buffering within the BMM is held to a minimum to accommodate single block or multiple block access at the instantaneous maximum rate of the BMM. The PSM is intended to provide adequate power to operate the SCM and one BMM simultaneously at the maximum data rate.

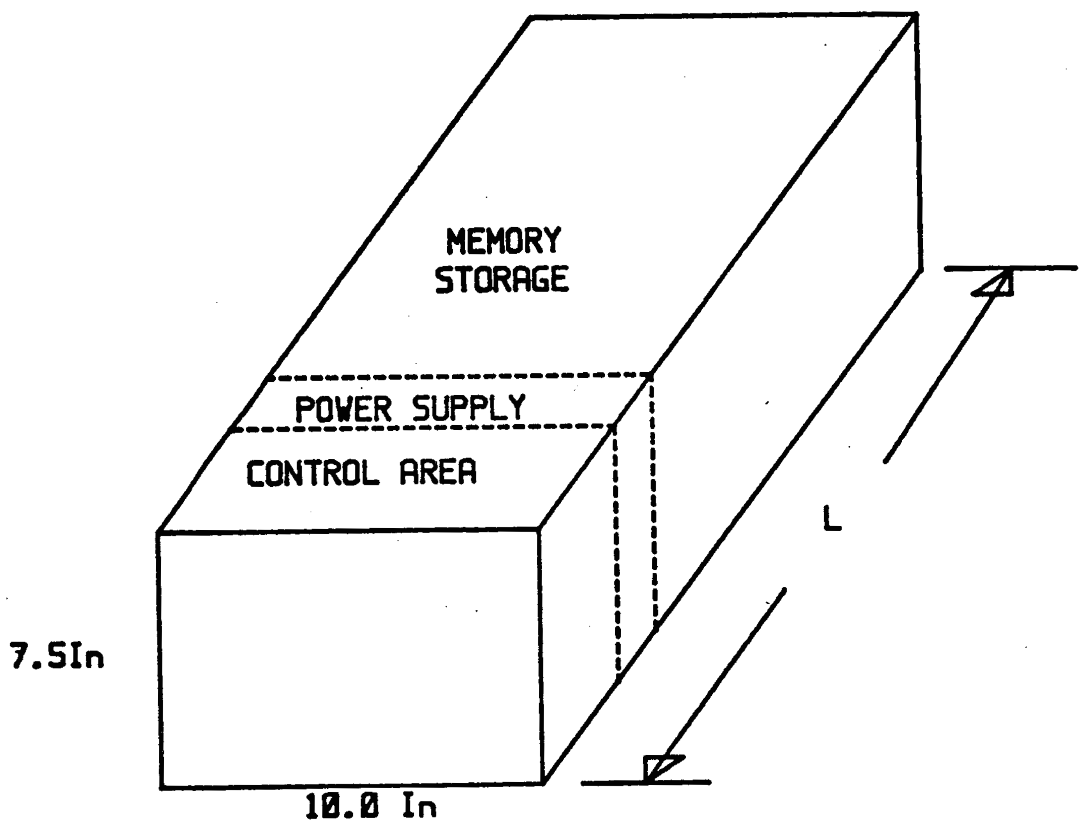
The Bubble Memory Module (BMM) is the principal subsystem of the modular memory system concept and its functional design is discussed in detail below. The BMM contains all memory device-related technology including not only the memory device but also all electronics required to store and retrieve data in the memory devices. Communication and data transfer to and from the BMM are by digital signals on appropriate bus lines compatible with microprocessor controllers.

The Power Supply Module (PSM) provides power to all other modules and enables the system to be compliant with the overall system power line requirements and restrictions. The PSM is sized for minimum power to operate the smallest number of memory devices to attain the maximum desired BMM data rate.

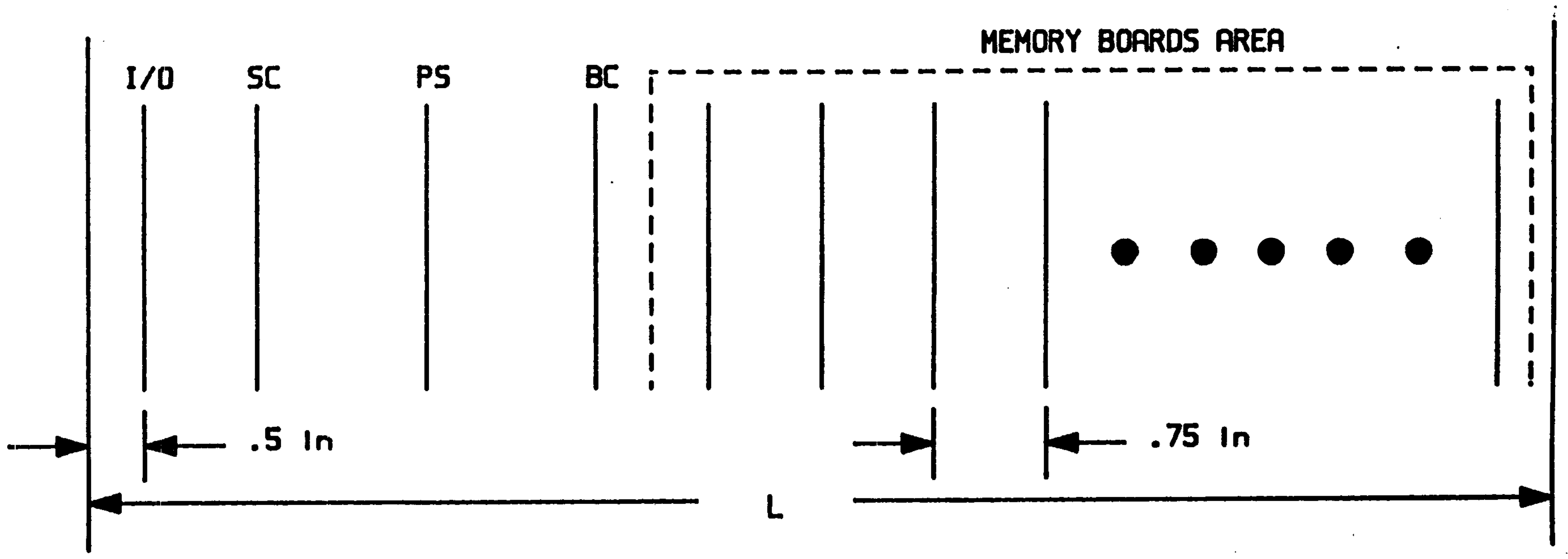
The System Controller Module (SCM) consists of one or more circuit cards providing system organization, housekeeping, and control. The division between the System Controller (SC) and Application I/O (AIO) could be designed to minimize redevelopment for potential future applications which are yet undefined.

Capacity estimates have been made for a memory system containing one PSM and one BC. The memory system geometry, shown in Figure 3, is consistent with previous onboard electronic systems. For this estimate, the spacing between circuit boards comprising the system is depicted in Figure 4. The circuit boards are 6 inches long by 9 inches wide. Each memory board is assumed to carry 8 bubble memory devices.

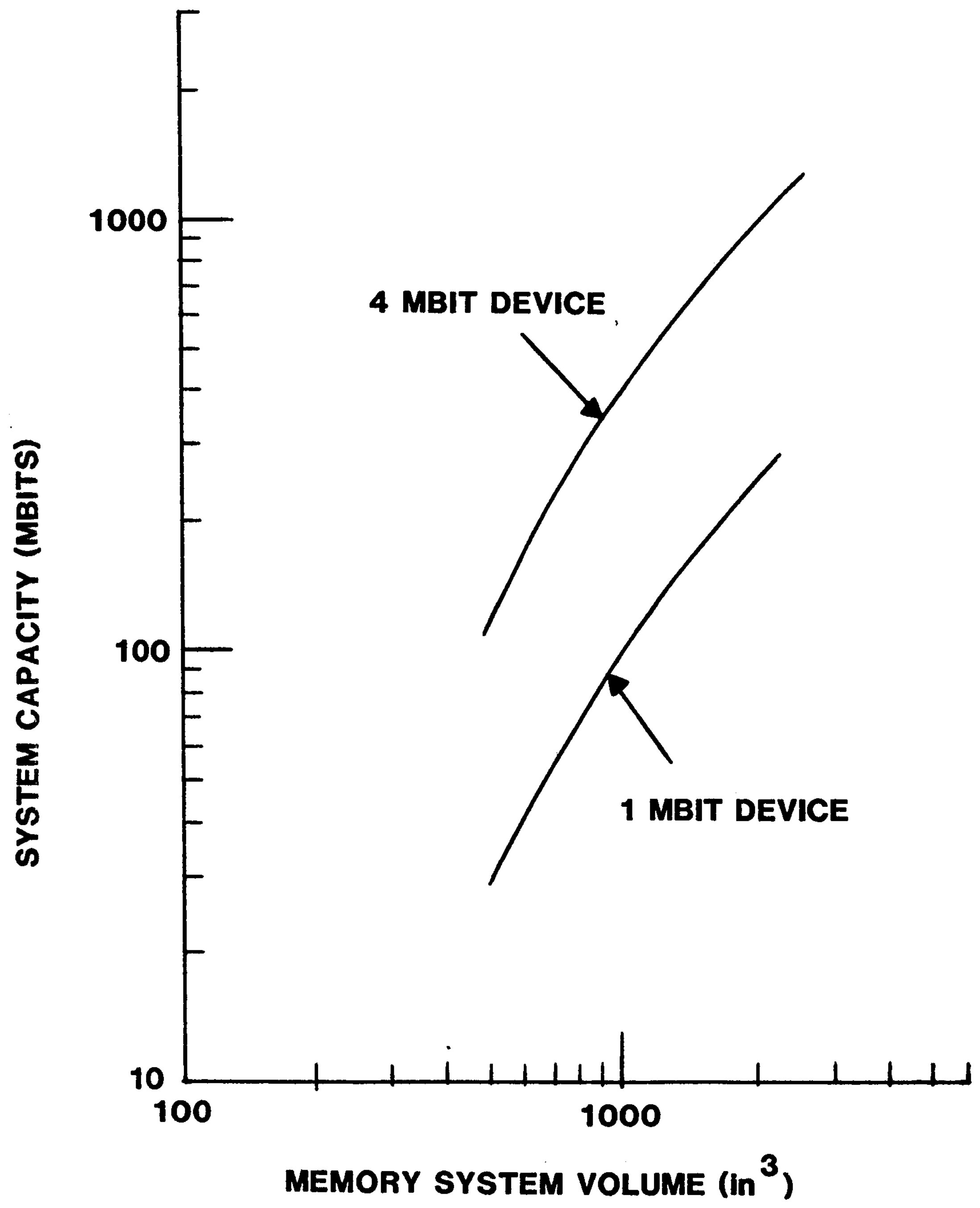
The resulting projections for onboard memory systems are plotted in Figure 5 for 1 Mbit and 4 Mbit devices. Devices of 1 Mbit capacity are commercially available and 4 Mbit commercial devices have been announced. Thus onboard systems in the 10^8 to 10^9 bit capacity ranges could be developed.



MEMORY SYSTEM GEOMETRY
FIGURE 3



CIRCUIT CARD SPACING ASSUMPTIONS
FIGURE 4



POTENTIAL SPACECRAFT BUBBLE MEMORY SYSTEMS
FIGURE 5

MEMORY MODULE DESIGN

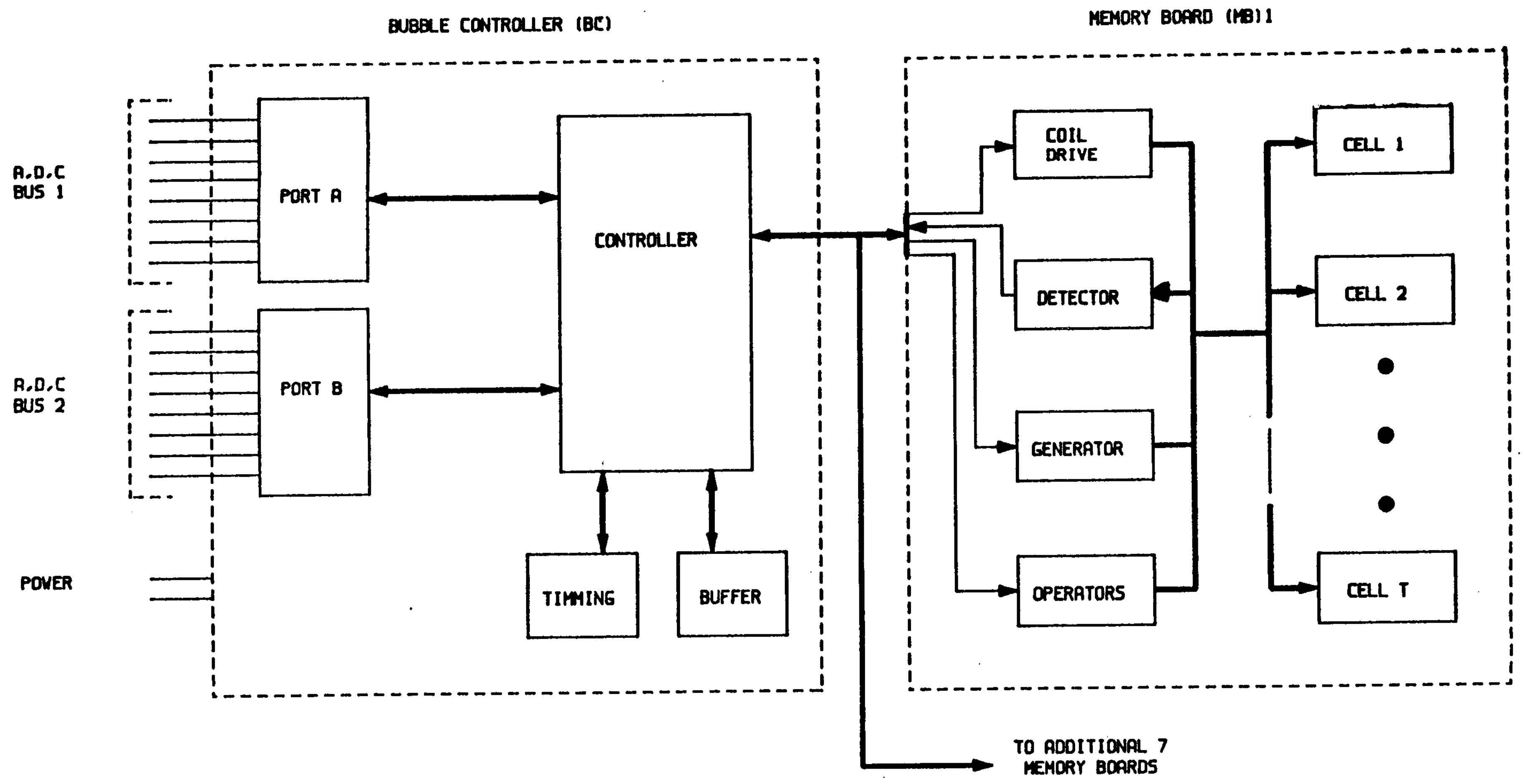
Functional Organization

The Bubble Memory Module (BMM) design is functionally organized as shown in Figure 6. Two circuit board designs are required, one for the Bubble Controller (BC) and one for the Memory Board (MB). The BMM is expandable from a minimum of one MB up to at least 8 MBs in parallel. A suitable internal bus scheme is included to permit the BC to energize one MB at a time to access data within a specific location.

The BMM provides block organization of data with the maximum block size determined by the number of minor loops in the bubble device and the number of parallel memory devices (N) required to operate simultaneously to attain the required maximum BMM data rate. The BMM could provide for accessing smaller blocks at lower data rates. The number of memory devices (T) on the MB (Figure 6) is an integral multiple of N to provide the maximum data rate while operating a single MB.

The Bubble Controller (BC) acts on address and control signals from the user (through the dual port I/O) and routes data to or from the appropriate blocks in storage. Operation of the BMM is exclusively via the BC and access to the BC is only through the dual port I/O. The BC performs the following:

1. Timing--supplies internal timing and coordination of all internal memory device timing functions and also generates appropriate user timing signals where necessary at the I/O.
2. Commands--accepts commands to perform all read, write, initialization, block organization, turn-on, and turn-off functions.

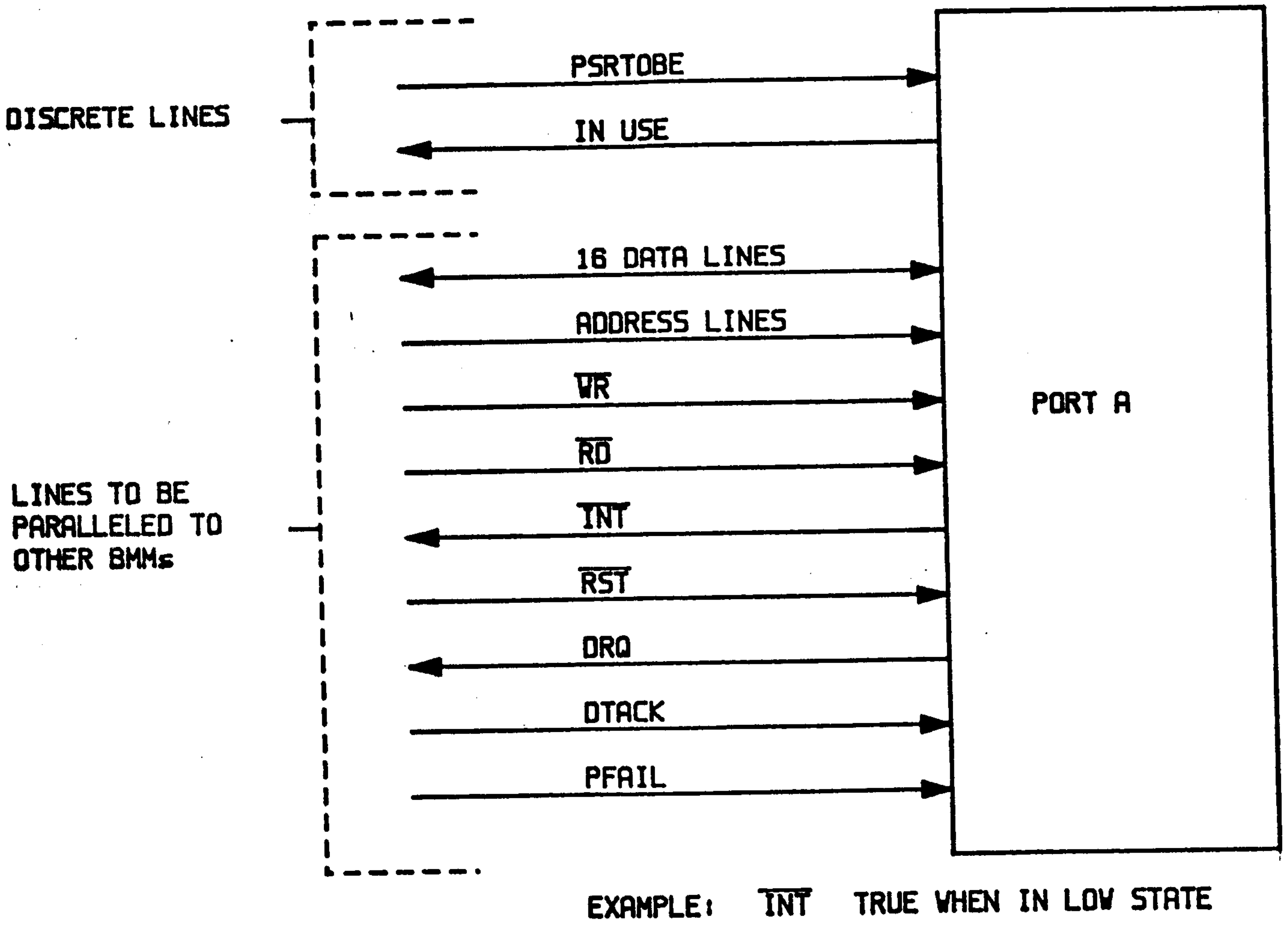


BUBBLE MEMORY MODULE FUNCTIONAL DIAGRAM
FIGURE 6

3. Status--contains status registers which reflect the current organization and operational status of the BMM.
4. Power protect--provides internal protection to protect the integrity of stored data in event of loss of power to the BMM.
5. Block access--enables read and write access to data in first-in-first-out block fashion. The minimum data accessed is one block. Groups of blocks are accessible by user designation of the first block and the number of blocks. It is intended that arbitrary sections of the BMM accommodate both random block access and contiguous block access.
6. Data buffering--data buffers are kept to a minimum and used only as needed to accommodate the maximum average BMM data rate.
7. Dual bus control--responds to commands at both BMM interfaces and excludes operation of one bus when the other is being used.
8. Diagnostics--provides an internal scheme for quick self-testing and identification of problem areas. Status registers indicate any non-functional memory blocks to provide in-service work arounds.

Dual Port Interface

The BMM input/output interface is a dual port interface configurable to an individual user's needs. Each of the two input/output ports of the BMM requires a number of signal lines for control of the BMM in addition to signal lines used for data transfer. These are shown in Figure 7 for Port A. Those for Port B are identical. All signal lines shown for Port A other than the two discrete lines PSTROBE and IN USE may be paralleled with other Port A signal lines to up to seven other memory modules. All signal lines for Port B other than PSTROBE and IN USE may be paralleled with other



SIGNAL LINES FOR ONE PORT
FIGURE 7

Port B signal lines similarly. Only the module having an active PSTROBE line may respond to signals on its interface. An inactive signal port presents a high impedence at all its signal lines when inactive.

In order to minimize the noise sensitivity of signal lines all lines are level sensitive rather than edge-sensitive. Each of the signal lines is described below:

1. BMM Power Strobe Line (PSTROBE)

   In order to establish a minimum power condition a power strobe line is provided. If the PSTROBE line is high then the bubble controller assumes a condition to provide "instant" response to signal and control lines. When the PSTROBE line is low then the module deenergizes, with the exception of the circuitry necessary to detect the power strobe becoming active. Response to the PSTROBE line may not be immediate when the memory module has "housekeeping chores" to perform before responding to either the PSTROBE line going active (access request) or to the PSTROBE line going inactive (end of access) at the end of a transaction. In parallel operation of multiple BMMs, it is not the responsibility of the memory module to check the status of the other BMM PSTROBE lines. The user ensures that no two Port A and Port B PSTROBE's occur simultaneously for access to the same memory module. If the A and B ports of the same module have simultaneously active PSTROBE signals, the first accessed port maintains constant communication until its PSTROBE signal goes inactive and, if necessary, until its housekeeping chores are also completed. During this period the

IN USE signal of the alternate port is maintained active. The PSTROBE line is not connected in parallel with any other signal line.

The PSTROBE line is not the power line to the BMM, but rather, is a logic line to activate the BMM through the selected port. Power lines are provided through additional lines.

2. Memory Modules "In Use" Line (IN USE)

This line indicates to the user that the alternate access port is being used, and that access to this port cannot be granted. This line indicates the condition of the alternate port regardless of the port power strobe condition. This line is not to be connected to any other port line.

3. Data Lines

The data lines are high impedance (3 state) bidirectional data transfer lines capable of parallel bussing with similar data transfer lines of other memory modules.

4. Address Lines

The address lines are high impedance (3 state) signal lines capable of parallel operation with other similar address lines of other memory modules.

5. Write Control Line ($\overline{WR}$)

The Write Control Line ($\overline{WR}$) controls the direction of data transfer. The $\overline{WR}$ line defines the direction of data transfer as being from the user to the Bubble Memory Module. This line also defines timing in that all address and data lines maintain stable information during the active period of this control line.

6. Read Control Line (RD)

   The Read Control Line (RD) controls the direction of data transfer. The RD line defines the direction of data transfer as being from the Bubble Memory Module to the user. This line also defines timing in that all address lines maintain stable information during the active period of this control line.

7. Memory Module Interrupt (INT)

   A low on the INT line indicates to the user that the memory module has a new status and requires servicing by the user. The INT line remains low until the module is serviced by the user.

8. Bubble Memory Module Reset Line (RST)

   A low level on the RST simultaneously with an active PSTROBE line for the same module forces the memory module to an inactive idle state.

9. Data Request Line (DRQ)

   When the appropriate protocol has been followed for a particular memory module this line may go high to indicate that the module is ready for immediate data transfer to or from the user. The DRQ line is used only in the Direct Data Transfer (DDT) mode.

10. Data Transfer Acknowledge Line (DTACK)

    The user sets this line active in response to the module DRQ signal line becoming active. Data transfer occurs immediately. The transaction must be completed by the time the DTACK line is returned to an inactive state. The DTACK line is used only in the DDT mode.

The primary control lines are the IN USE and PSTROBE lines. These two lines are unique to each port and may not be connected to other similar lines of other ports. The bubble controller (BC) resolves multiple user requests via these lines. The PSTROBE line identifies to the controller that a user is requesting access. Unless the PSTROBE line is TRUE a module port will not respond to any user request to access memory. If a user requires access it must initially do so by activating its PSTROBE line. It must not activate any other line, unless its IN USE line is not activated by the bubble controller. Once a user gains access to the memory port it has a monopoly of all operations to that memory module. The alternate port on that module is locked out until the user terminates access by deactivating the PSTROBE. When the port is able to respond to the PSTROBE it should activate the INT and wait for the user to specify the next operation by the condition of the address lines to that port. It should also activate the alternate port IN USE signal line.

A parallel arrangement of BMMs (Figure 8) provides increased system capacity, flexibility, and redundancy. All port A signal lines other than PSTROBE and IN USE are capable of being paralleled with the corresponding port B signal lines of the other memory modules. Similarly, all port B lines except PSTROBE and IN USE are capable of being paralleled with the corresponding port B signal lines of other memory modules. Access to any memory module can be obtained via Port A and Port B of that module, but not simultaneously. While access to one memory module is occurring through Port A (B) then simultaneous access to another memory module is possible through Port B (A). For example, in Figure 8, access to BMM1 can be accessed through Port A simultaneously with access to BMM2 through Port B. Appropriate phasing of the PSTROBE lines for Port A and Port B can enable the selected MM to maintain its controller active for alternate access through Ports A and B.

The BMM provides status registers which identify all required operational parameters for its control and operation. These registers are accessible to the user via data lines. In addition, the BMM will internally make measurements of parameters such as BMM power and MB temperature and store this information in registers which are similarly accessible to the user.

On a status request the BMM responds by placing a status word on the data lines at the time of the RD stroke. Each bit of the status word has a defined meaning for interpretation by the user.

A set of commands characterizes all BMM operations. To issue a command to the memory module the user sets the address lines to the appropriate condition and activates the WR line. The memory module responds by setting the INT false after accepting the command. The user will then remove the WR signal. The command set includes, as a minimum, the following generic commands:

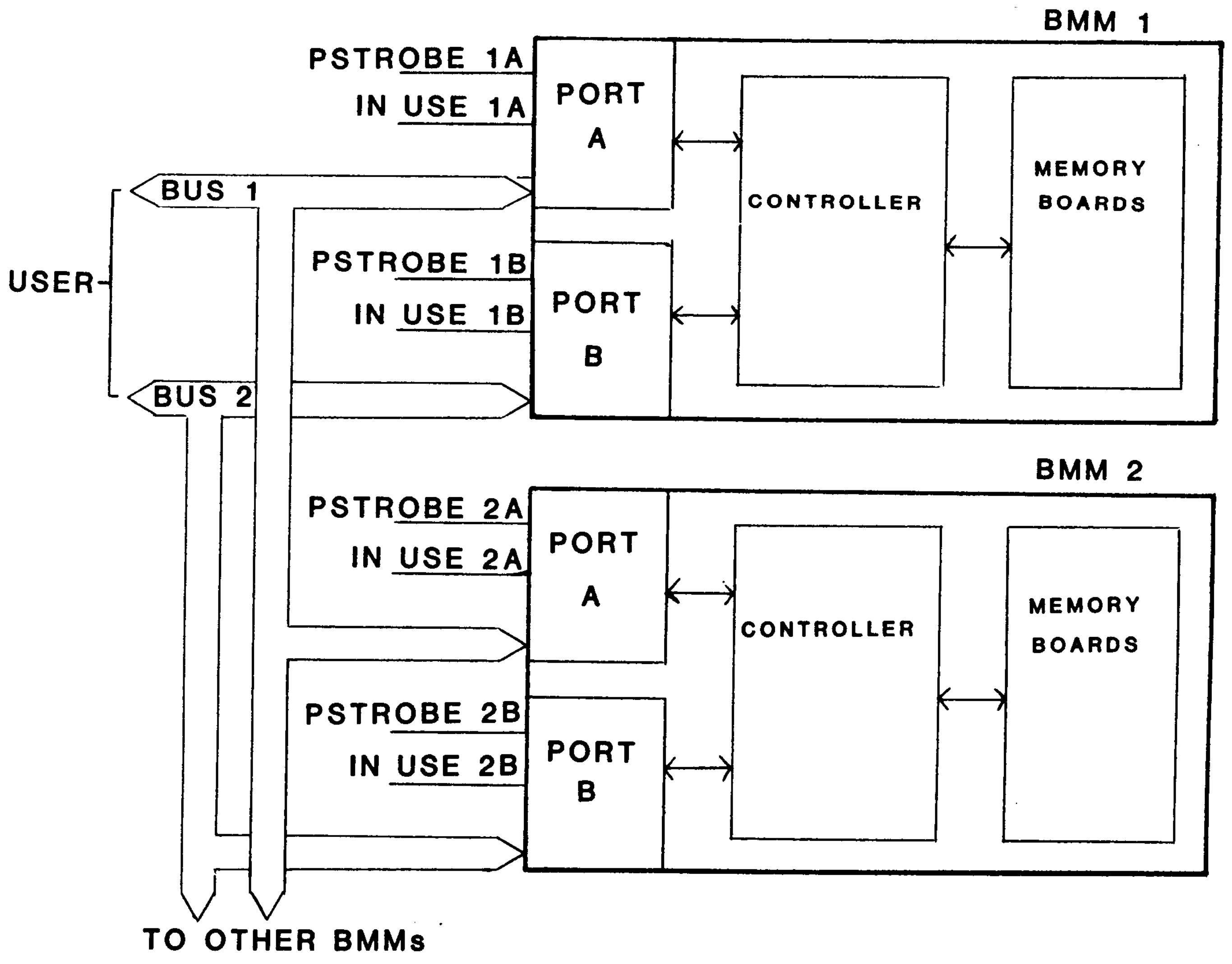


Figure 8.- Interface for multiple bubble memory modules.

1. NORMAL INITIALIZE

Either this command or fast initialize is required whenever the BMM is activated by the PSTROBE from an off condition. It is not required when both PSTROBEs of Port A (B) and Port B (A) are on simultaneously for a brief time and control is changing from one port to another. The memory module performs any internal housekeeping necessary prior to continued operation. This command is not a RESET, and is ignored if received during any other commanded operation (seek, transfer, etc.). This command causes bootloop information to be loaded into the appropriate controller register directly from the bubble devices. At the completion of this command the module is ready to receive further commands in a polled transfer mode. The block transfer number is equal to zero.

2. FAST INITIALIZE

This command is identical to that for NORMAL INITIALIZE with the exception that bootloop information is loaded into the appropriate controller register from EEPROM devices.

3. LOAD BOOTLOOP Specified

This command causes the BMM to transfer bootloop information from the specified memory board into EEPROM devices contained in the bubble controller.

4. SEEK BLOCK Specified

This command causes the memory module to perform all operations necessary prior to actual data transfer into or out of the memory storage. At the completion of the operation the INT line is used to notify the user of the completion of the operation.

5. WRITE TO BLOCK Specified

This command causes the memory module to start an actual write operation. If the seek block command has not been previously issued, the memory module automatically performs the seek operations necessary, and then immediately commences the write operation. Only one block is transferred, starting at the block specified by the "set block start" address (discussed below). Dependent upon previous commands, the $\overline{\text{INT}}$ or DRQ line is used to request data transfer operation for data storage.

6. READ BLOCK Specified

This command causes the memory module to start an actual read operation. If the seek block command has not been previously issued the memory module automatically performs the seek operations, and then immediately commences the read operation. Only one block is transferred starting at the block specified by the set block start address. Dependent upon previously issued commands the $\overline{\text{INT}}$ or DRQ line is used to request data transfer operations for data transfer to the user.

7. ABORT OPERATION

This command causes the memory module to abort any operation in progress including data transfer and return to a status to receive further commands including "INITIALIZE." This command does not cause any loss of stored data already in the memory devices.

8. MULTI-BLOCK WRITE

This command causes the memory module to start a write operation. If the seek block command has not been previously issued, the memory module automatically performs the seek operations, and then immediately commences the write operation. Writing continues sequentially until all the blocks specified by the "block transfer number" have been exhausted. Where the transfer crosses MB boundaries, the bubble controller anticipates and energizes the MB's in a manner which provides the average continuous useful BMM data rate and minimizes BMM power.

9. MULTI-BLOCK READ

This command causes the memory module to start a read operation. If the seek block command has not been previously issued the memory module automatically performs the seek operation, and then immediately commences the read operation. Reading continues sequentially until all the blocks specified by the block transfer number have been exhausted. Where transfer crosses MB boundaries, the similar characteristics as those for MULTI-BLOCK WRITE are used.

10. GO TO DDT

This command causes the BMM to transfer data using Direct Data Transfer (DDT). Direct Data Transfer is discussed below.

11. SET UP ECC

This command causes the BMM to organize itself for Error Correction Coding (ECC). The fact that the MM is set up for ECC is indicated in the status word and also retained in nonvolatile storage when the BMM is off. Once set up for ECC the BMM remains so, with or without power applied to the BMM, until the ABORT ECC command is received.

12. ABORT ECC

This command terminates the organization of the BMM established by the SET UP ECC command. It enables the BMM to use storage areas allocated for ECC for additional storage of data. This condition remains unchanged, with or without power applied to the BMM, until a SET UP ECC command is received.

Address lines as indicated in Figure 7, are used to specify the type of information transfer required. Typical functional address requirements are:

1. STATUS REQUEST

   The condition of the data lines establishes the present status of the addressed memory module.

2. MODULE COMMAND

   The condition of the data lines specifies the action or operation desired by the user.

3. DATA TRANSFER

   The information on the data lines is the information previously stored, or to be stored in the memory module.

4. SET BLOCK START

   The information on the data lines specifies the "set block address" which is the address of the next block to be accessed by the user. To issue a starting block address, the user sets the address lines to the appropriate condition and activates the WR line. The memory module responds by setting the INT line false after accepting the data on the data lines. The user then removes the WR signal.

5. BLOCK TRANSFER NUMBER

   The information on the data line specifies the number of blocks to be transferred in the next read or write operation.

Information Transfer and Timing

The BMM transfers information, whether storage data, address, commands, etc., by all of the following three transfer methods.

1. Polled Data Transfer

   The user continually polls the module for module status. Immediately after the status word indicates a "data available" status, the user requests a data word transfer read or write operation as previously set up by the parameter specifications.

2. Interrupt Data Transfer

   The user sets up the module parameters required for the operation and waits for the module $\overline{\text{INT}}$ signal line to indicate a change in status. The user then requests a module status word to verify that data is available. Immediately after the status indicates a "data available" status, the user requests a data transfer read or write as previously set up by the parameter specifications.

3. Direct Data Transfer (DDT)

   In the DDT mode, the DRQ line is used by the memory module to inform the user that data is available for immediate transfer. The user acknowledges the transfer of data using only the DTACK line. Due to the variable length of the bus lines, no bus clock is used. All transactions are handshake although timeouts may be used to minimize the effects of no response from one of the transaction partners.

   The timing parameters for data transfer, when <u>not</u> using the DDT method, are shown in Figures 9 and 10 for WRITE and READ, respectively. The information on the data lines may contain storage data, commands, module transfer parameters, status, etc. Only the selected

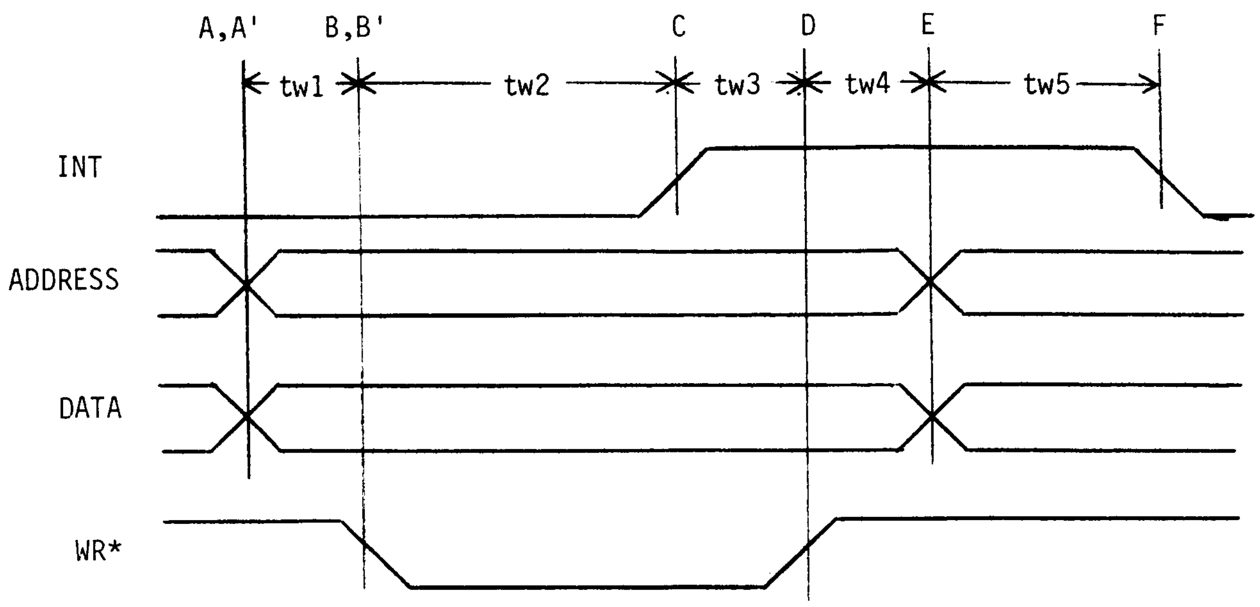


Figure 9. Memory write operation - timing.

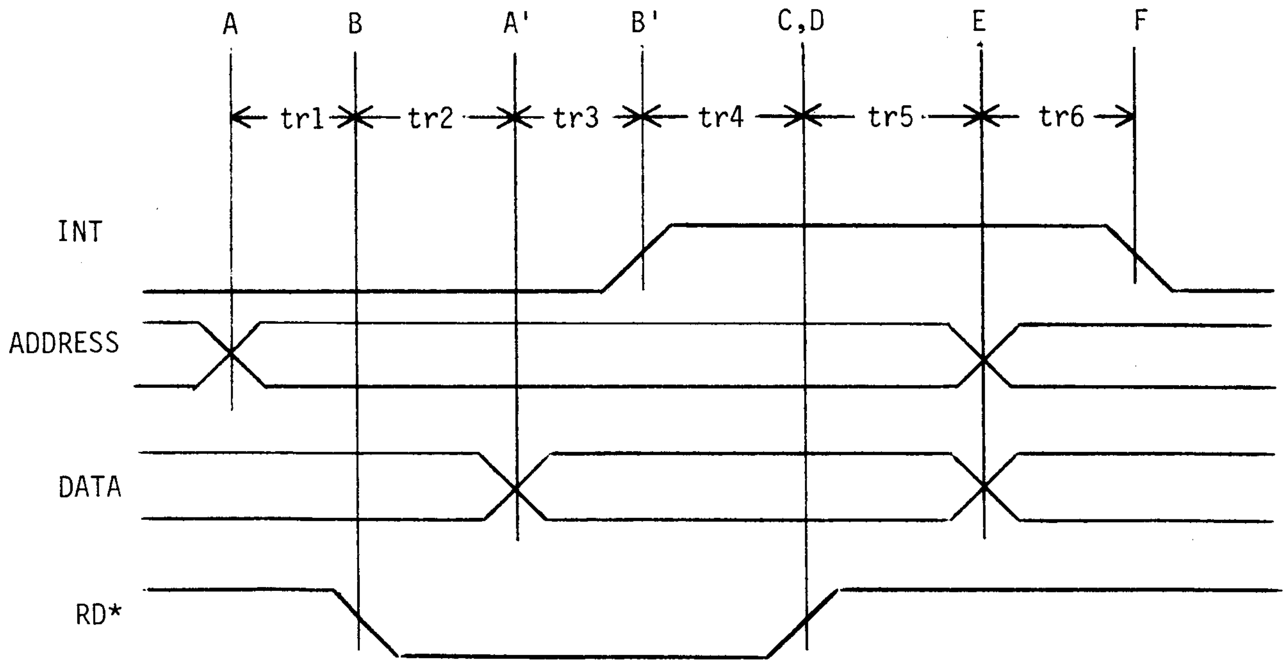


Figure 10.- Memory read operation - timing.

port with an active PSTROBE line and false IN USE line responds to transfer request. The handshake waveform meanings corresponding to Figures 9 and 10 are given in Table 1.

The timing parameters when using the DDT method for storage data are given in Figures 11 and 12 for WRITE and READ, respectively. The waveform meanings are described in Table 2. The response times indicated on the figures and tolerances are not specified but normally would be specified during or prior to system development. The interface design facilitates data transfer of as much as one block of data at an instantaneous rate of at least 10 Mbps.

The Direct Data Transfer operation bypasses use of the address and $\overline{RD}/\overline{WR}$ lines. In the DDT transfer the memory module issues a direct transfer request on the DRQ line. For this to happen the user must first conform to the DDT protocol to establish the transfer parameters with the memory module. When ready the memory module will inform the user via the DRQ line. The user responds by only activating the DTACK line.

If the transfer is a write operation the user must ensure that stable data are on the data lines before asserting the DTACK control line. The user must maintain the data stable until after the memory returns the DRQ line inactive to signal acceptance of the data. The user should then remove the DTACK signal and subsequently remove the data line signal.

If the transfer is a read operation then the user should respond by DTACK. The memory should then place stable data on the data lines and subsequently lower the DRQ line. The memory maintains stable data on the data lines until after the user returns the DTACK to an inactive level.

Table 1. Identification of waveform meanings for non-DDT transfers (See Figs. 9 and 10)

A	- User sets up functional address for transfer
A'	- Data set up on lines
B	- User is ready and requests transfer
B'	- Indicates data are stable on lines
C	- Data has been taken
D	- Acknowledge data transaction has been completed
E	- Address removed by user. Data removed from lines.
F	- Status change signaling readiness for next data transfer

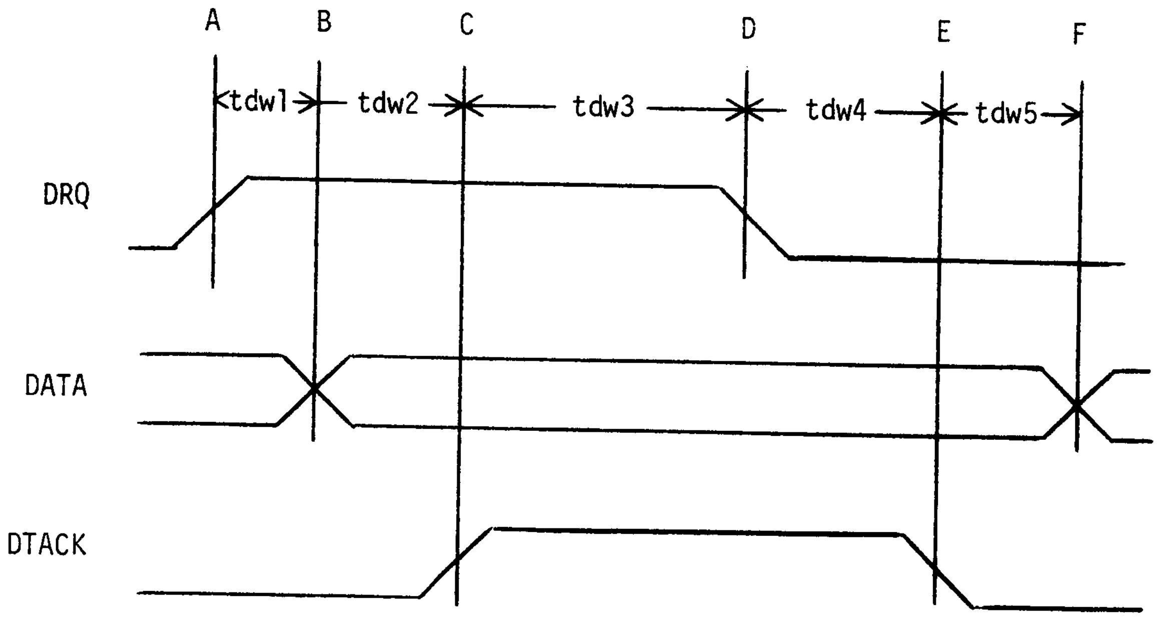


Figure 11. DDT write operation - timing.

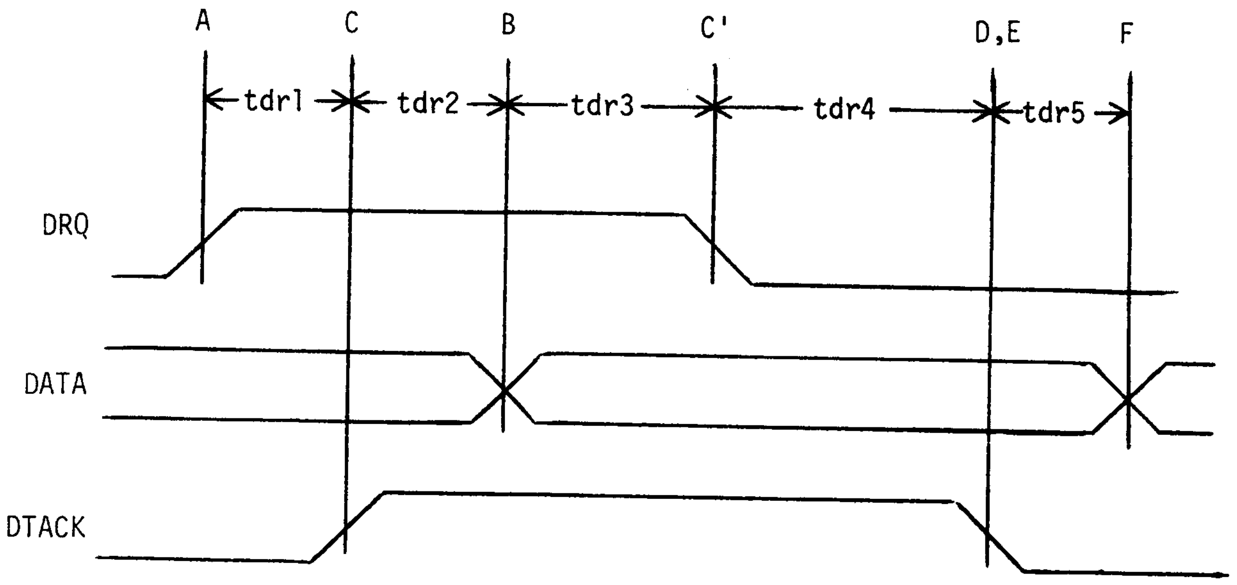


Figure 12.- DDT read operation - timing.

Table 2. Identification of waveform meanings for DDT transfers (See Figs. 11 and 12)

A	- BMM requests data transfer
B	- Data put on lines
C	- Acknowledge request and says "I'm ready for transfer." (Data are on lines if a WRITE.)
C'	- Data are stable on lines (READ only)
D	- Data has been taken
E	- Acknowledge transfer complete
F	- Remove data from lines

Miscellaneous Features

The Bubble Controller (BC) provides Error Correction Coding (ECC) of blocks of data stored in the memory device. The BC is capable of organizing, upon receiving appropriate user commands, the MM to either use or not use the ECC. When ECC is used, the BC provides user options for error detection without correction, error detection with automatic correction, and error detection with stop and wait for user instructions. When ECC is not used, the storage area normally used for error coding is converted to additional data storage sites.

The BMM provides internal protection of data stored in the memory devices in event of a power failure. This protection facilitates an orderly shut down of the BMM and ensures that data, ECC Status, and other status requirements are not lost. Data located in buffers within the BMM need not be saved.

The BMM design facilitates its being strobed on for brief intervals via the PSTROBE line to conserve power in aerospace onboard memory applications. In conjunction with this approach the BMM is also able to select and power-on the MB in which storage blocks have been addressed. Hence, fast turn-on is of prime importance to minimize power and simultaneously provide fast access. The following maximum access times can be provided by the BMM upon activation of the PSTROBE signal:

1. Time to be ready to receive commands = $\leq$ 20 μsec.

2. Time to receive appropriate commands and initialize controller $\leq$ 1 msec.

3. Time to receive appropriate commands and begin data transfer (READ or WRITE) < 5 μsec. (Includes energizing the appropriate MB.)

The Memory Board (MB) contains the memory devices and much of the circuitry required to access these devices. The MB is controlled exclusively by the BC and is capable of being connected in parallel with other similar MB's up to a total of 8 in parallel.

The BMM receives data and reproduces stored data asynchronously in block form. The minimum actual data transfer rate to memory chips will be higher than the average BMM input/output data rate to accommodate command, control, and address functions. The average useful data rate for contiguous blocks, which is available to the user at the BMM I/O, is 1.2 Mbps. The actual average data rate is dependent on the duty cycle at which blocks are requested.

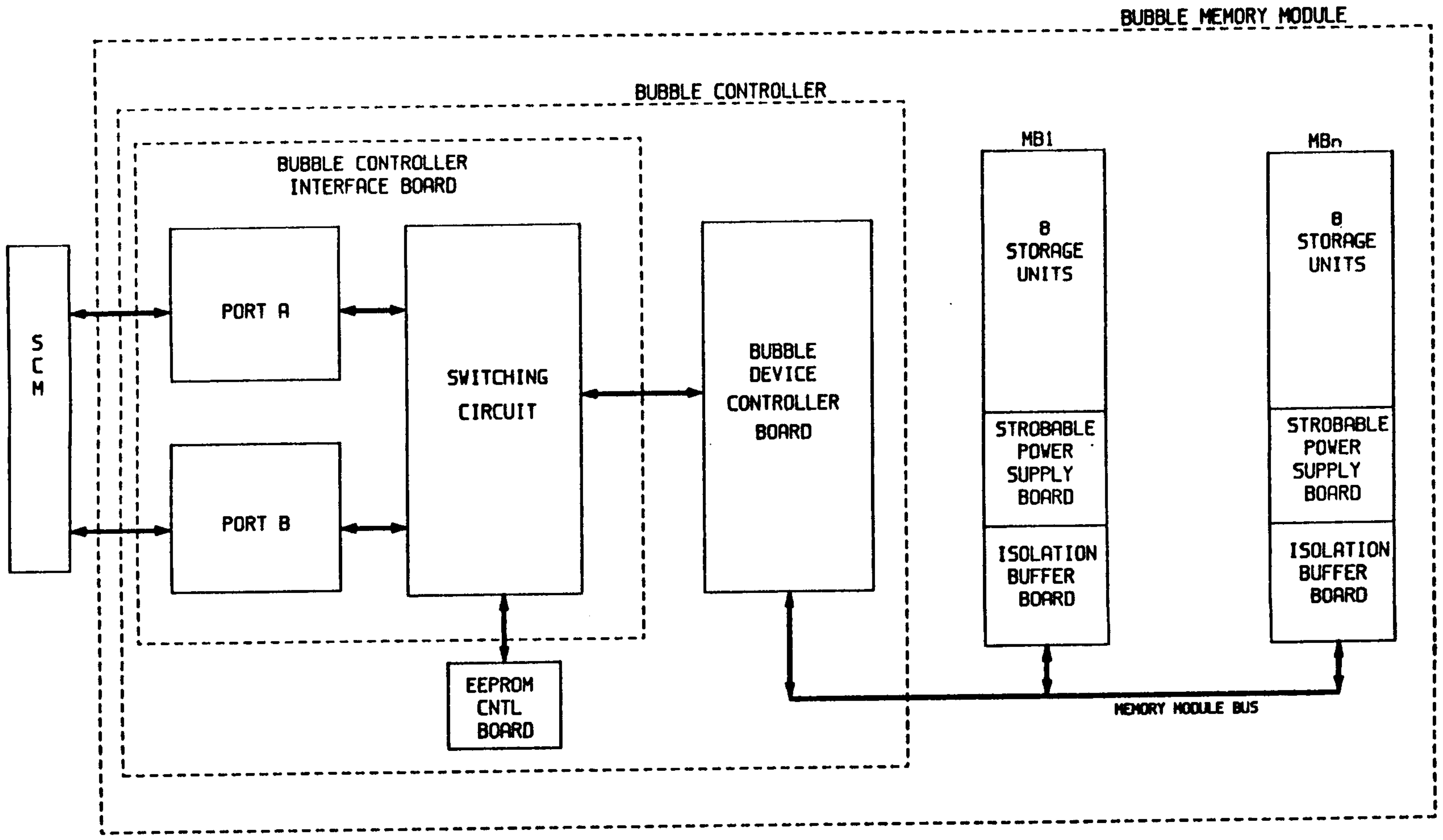
The average power required to operate the BMM is data rate dependent. The power goal for the BMM at 1.2 Mbps is $\leq$ 25 watts/Mbps. Circuit multiplexing and power switching to turn on only those circuits which are needed for data access should minimize power. The BMM requires zero power to maintain stored data integrity.

As a goal, the uncorrected error rate for the BMM is less than 10^{-8} errors/bit for all data patterns. When using Error Correction Coding (ECC) and detection the BMM should provide corrected data with less than 10^{-14} errors/bit.

MEMORY MODULE IMPLEMENTATION

Hardware Structural Organization

The prototype evaluation design of the modular bubble memory system concept is shown in the block diagram in Fig. 13. Major modules depicted in this figure are defined as in Fig. 1. The System Controller Module (SCM) translates high level user commands into simple digital signals for use by the Bubble Controller (BC). The BC takes these digital signals and outputs the specific current and voltage levels and timing characteristics that are required to drive the Memory Boards (MB). For testing purposes, the SCM is designed around a 16-bit single board computer with serial and parallel port capability. The SCM and the BC, minus the Bubble Device Controller (BDC), reside in a card cage (shown in Fig. 14). The memory module dual bus interface is designed to be generically compatible with any microprocessor. Only the Port A section has been constructed for the implementation, due to the duplication involved in the Port A and B designs. Each Memory Board (MB), for the purposes of functional evaluation, consists of a series of printed circuit cards arranged in a memory module card cage arrangement, as pictured in Fig. 14. This card cage contains 11 cards: 1 BDC Board, 1 Isolation Buffer Board (IBB), 8 Storage Units (SU), and 1 Strobable Power Supply Board (SPSB). The interface between the BC and the BDC, which is in the memory module card cage, is twisted pair cable. The isolation buffers insure that there is no garbling of data on the bus. A block diagram of the current set-up is shown in Fig. 15. Specific elements within each module will be discussed separately in greater detail below.



BLOCK DIAGRAM OF BUBBLE MEMORY MODULE
FIGURE 13

ORIGINAL PAGE IS
OF POOR QUALITY

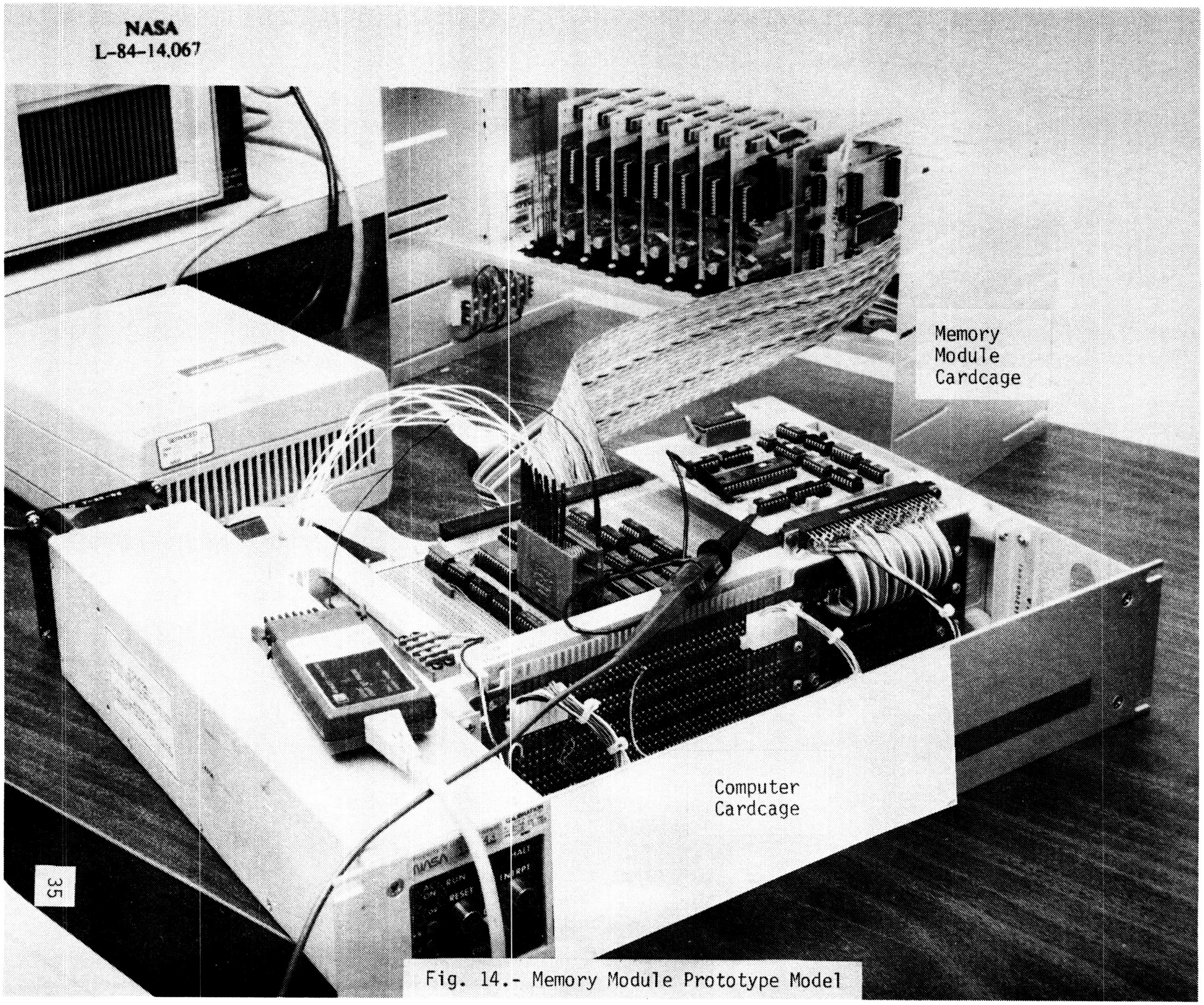
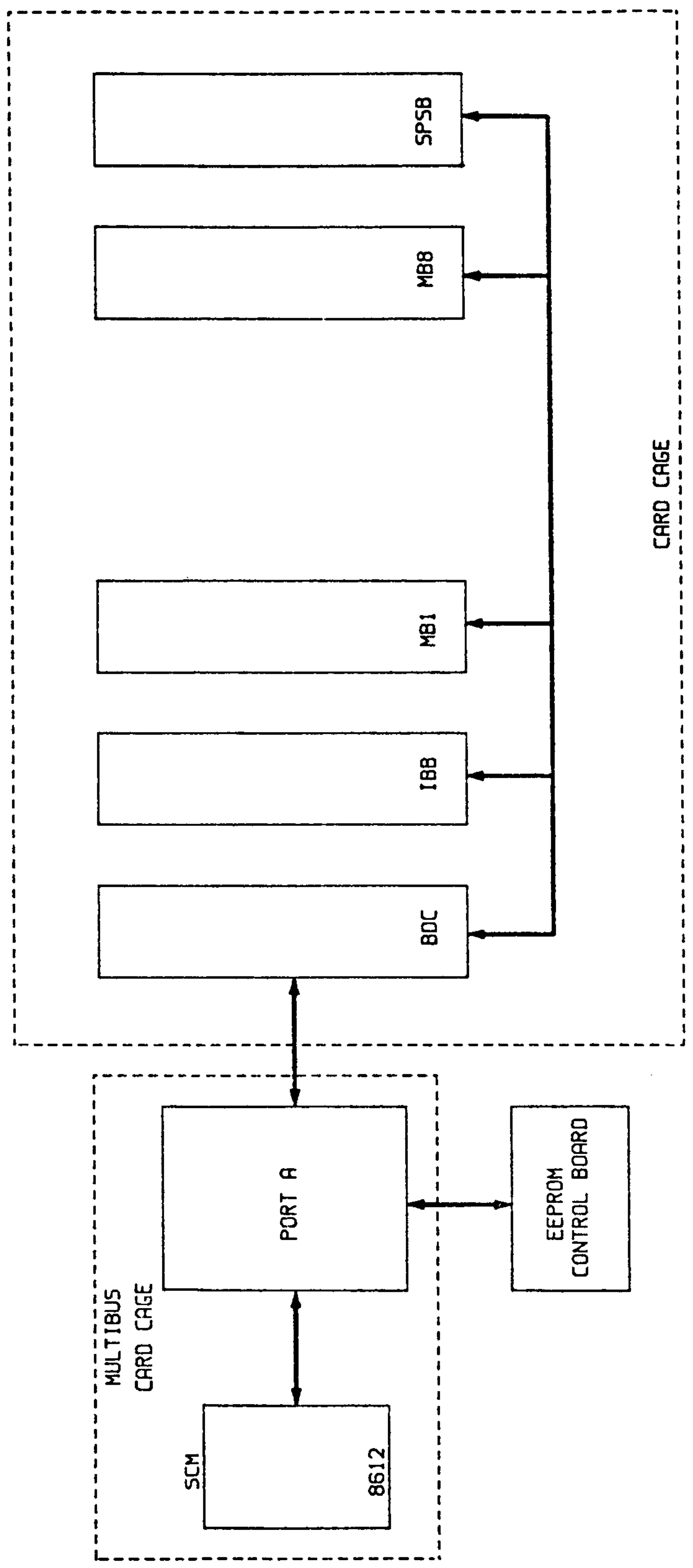


Fig. 14.- Memory Module Prototype Model

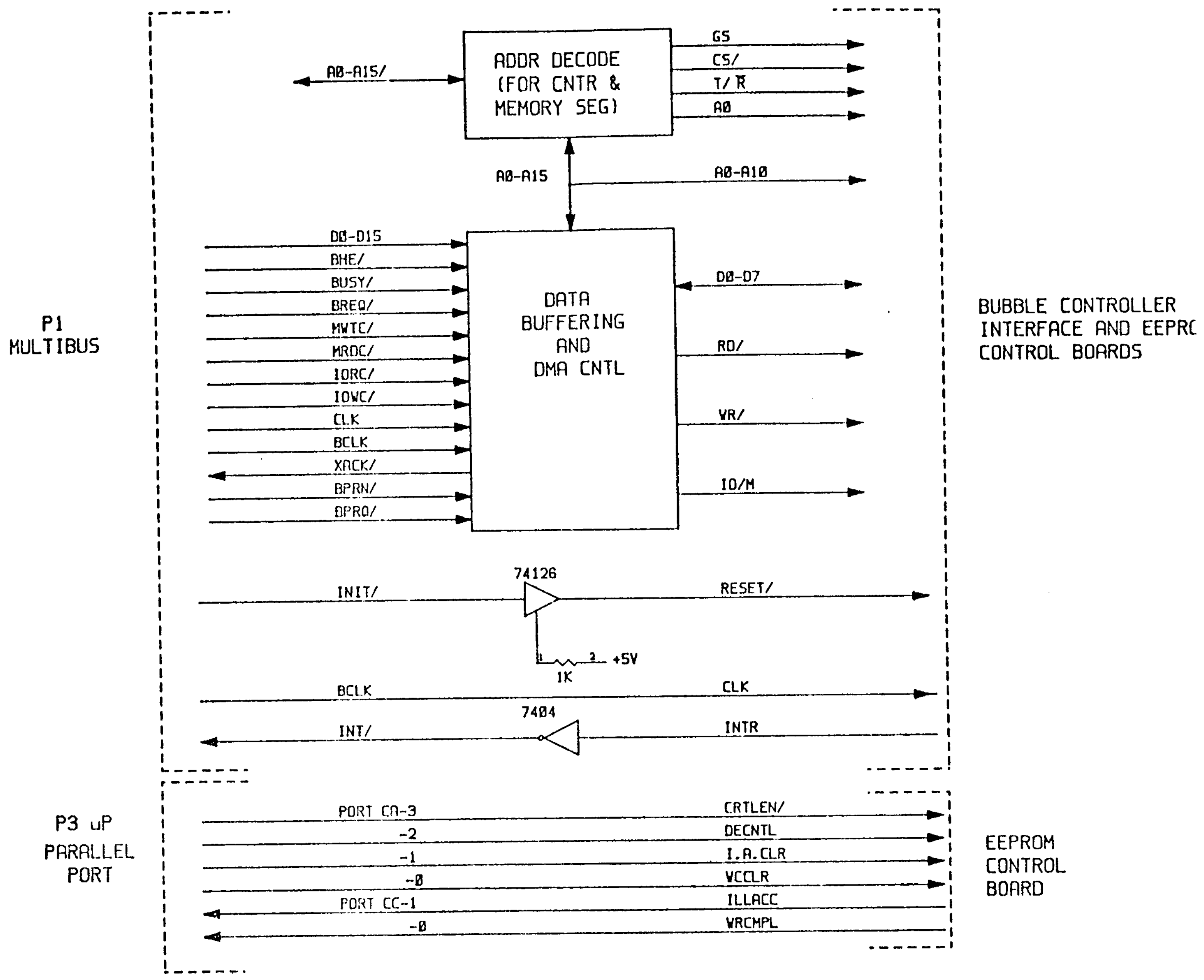


CURRENT PROTOTYPE SET-UP
FIGURE 15

Bubble Controller Interface Board

The bubble controller interface board (BCIB) consists of the circuitry necessary to interface the BDC to the single board computer, which is serving as the SCM for this experiment. Functionally, the BC acts on address and control signals from the user (in this case, the SCM), exclusively through the dual port I/O, to handle the routing of data to and from the appropriate storage units. A block diagram of the interface circuitry for Port A is shown in Fig. 16. The interface for Port B will be almost identical except for the circuitry that will be shared with Port A.

Table 3 shows the necessary hardware interface signals. A schematic of the actual prototype board is shown in Fig. 17a and 17b, and Fig. 17c shows the actual prototype board. This interface, as shown in Fig. 16, required the use of address buffers and decoding, data and control signal buffers, DMA control circuitry (for DDT operation), acknowledge decode, and EEPROM interface control circuitry. Buffering of all signals that cross the bus is necessary so that there is no confusion about who has control of the bus and to prevent the garbling of data. Address decoding is required for the selection of the bubble device controller when issuing commands or transmitting data, and for the selection of the proper storage location within the memory system. An acknowledge signal is needed from the BC to let the single board computer know the information has been received. Interface control circuitry for the EEPROM is needed to shuttle the bootloop information between the EEPROM board and the appropriate memory board upon the issuance of the fast initialization command (ref. 10). The DMA control circuitry is necessary for the implementation of the DDT mode. It is important to note here that everything possible must be



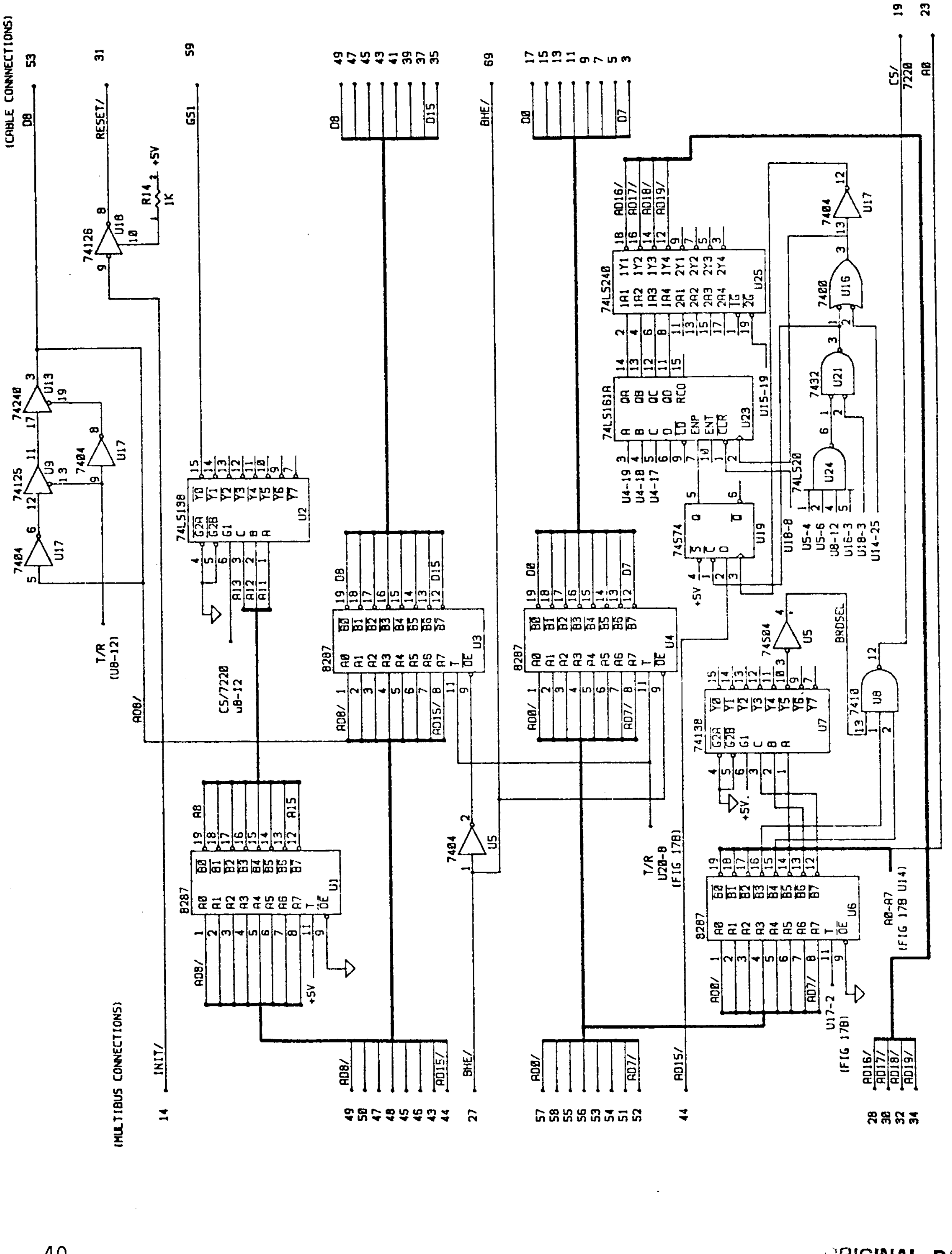
BLOCK DIAGRAM OF BUBBLE INTERFACE CONTROLLER
FIGURE 16

Table 3. Bubble Device Controller Interface Signals

SIGNAL	FUNCTION

A0	Address line, selects:
	A0=0 FIFO or parametric registers
	A1=1 Command/Status registers

D0-D7	Bidirectional data bus
D8	Optional odd parity bit
7220 CS/	BMC chip select input CS/=0 controller select: CS/=1 tri-state interface signals
RD/	Read 7220 registers or data FIFO
WR/	Write 7220 registers or data FIFO
DACK/	DMA acknowledge
WAIT/	Used when BMC's are operated in parallel--causes halt when an error is detected in a BMC--requires
CLK	4 MHz T^2L level clock clock period = 250 nm(0.25 ns tolerance) duty cycle = 50% (5% tolerance)
RESET/	A low on this pin forces the interruption of any 7220 activity, performs a controlled shutdown and initiates a reset sequence
7242 CS/	Should be tied to ground for single bank systems

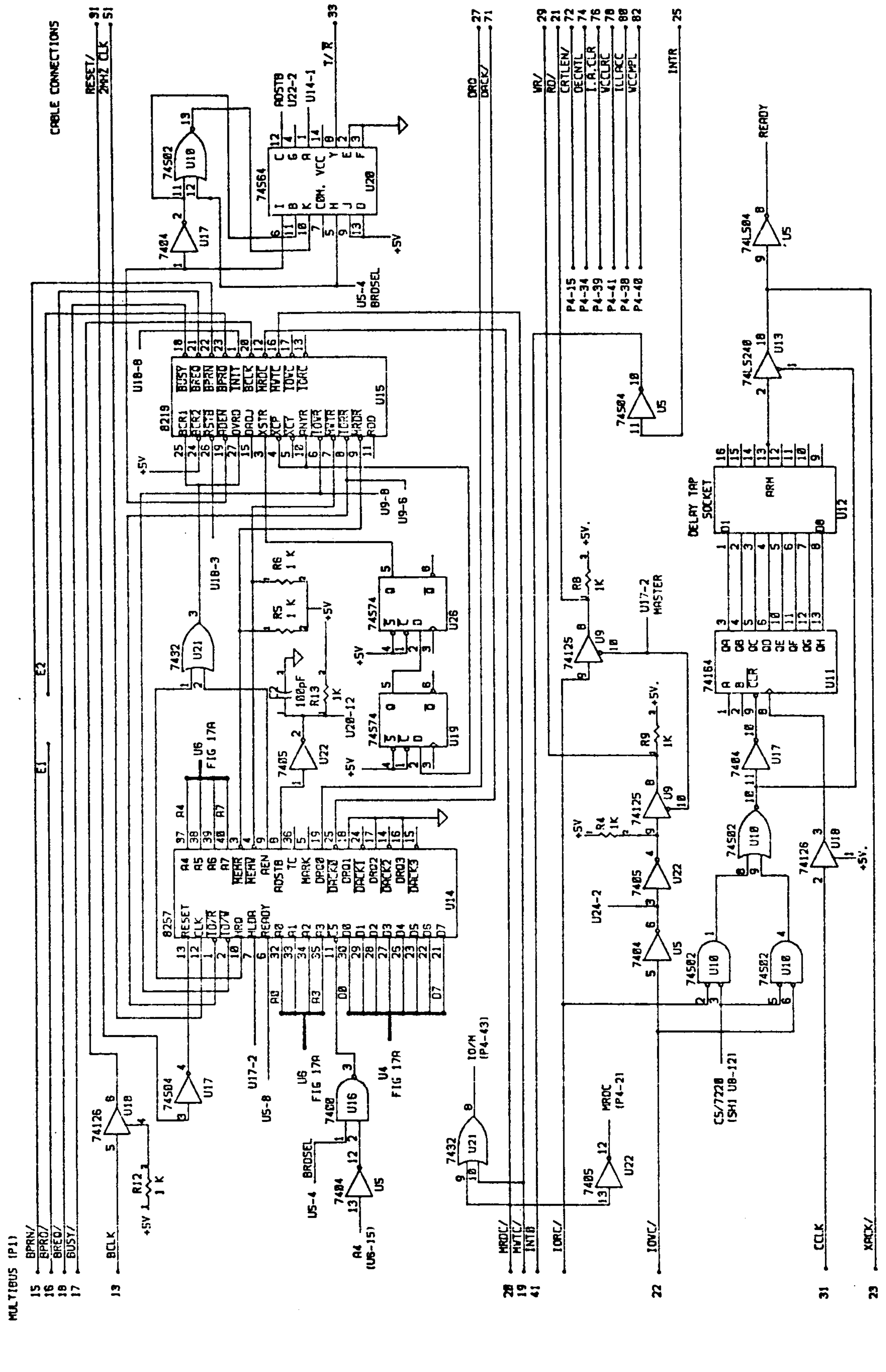


SCHEMATIC FOR THE BUBBLE CONTROL INTERFACE BOARD

FIGURE 17A

ORIGINAL PAGE IS OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY



SCHEMATIC FOR THE BUBBLE CONTROL INTERFACE BOARD
FIGURE 17B

NASA
L-84-14,062

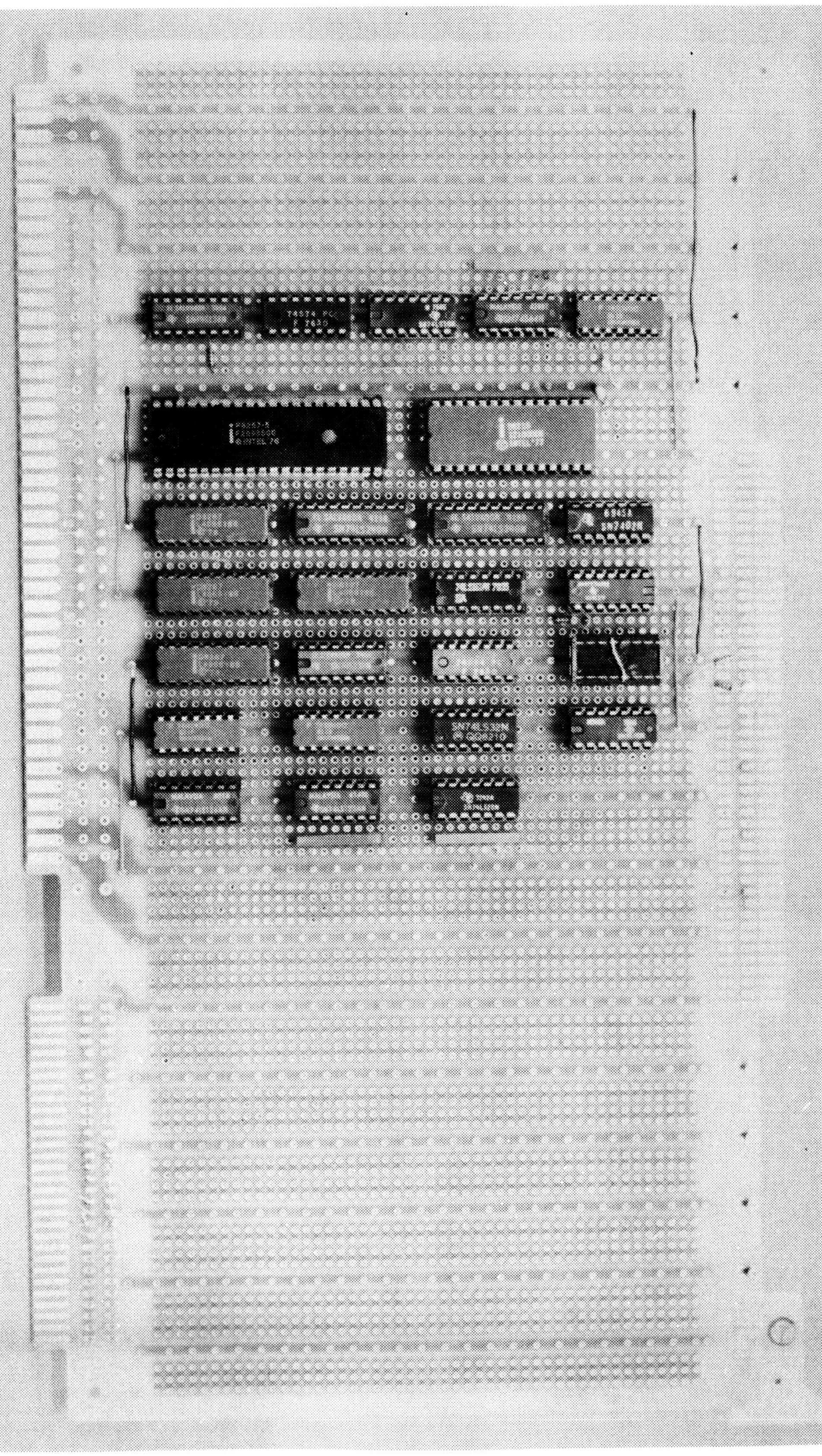
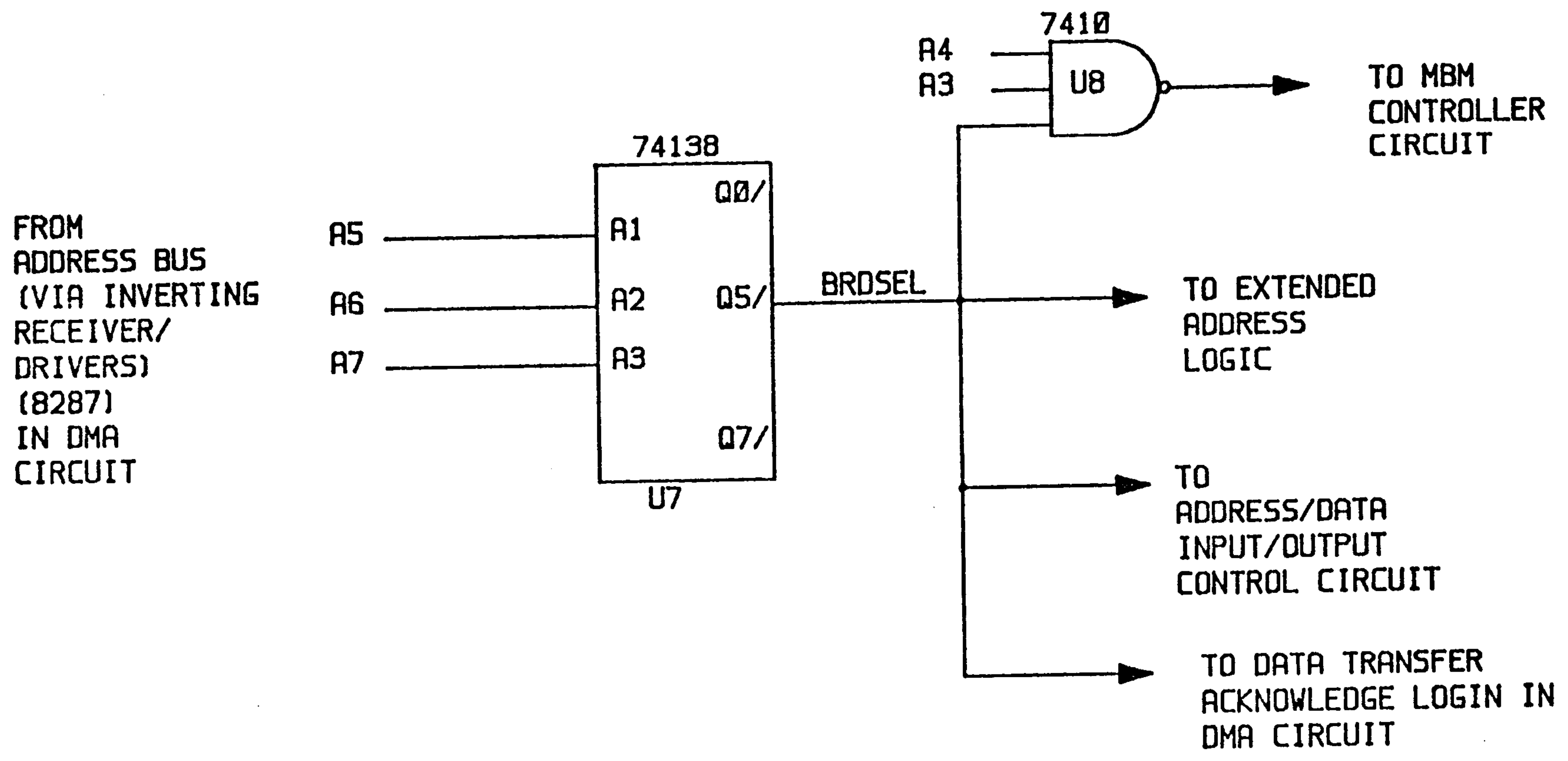


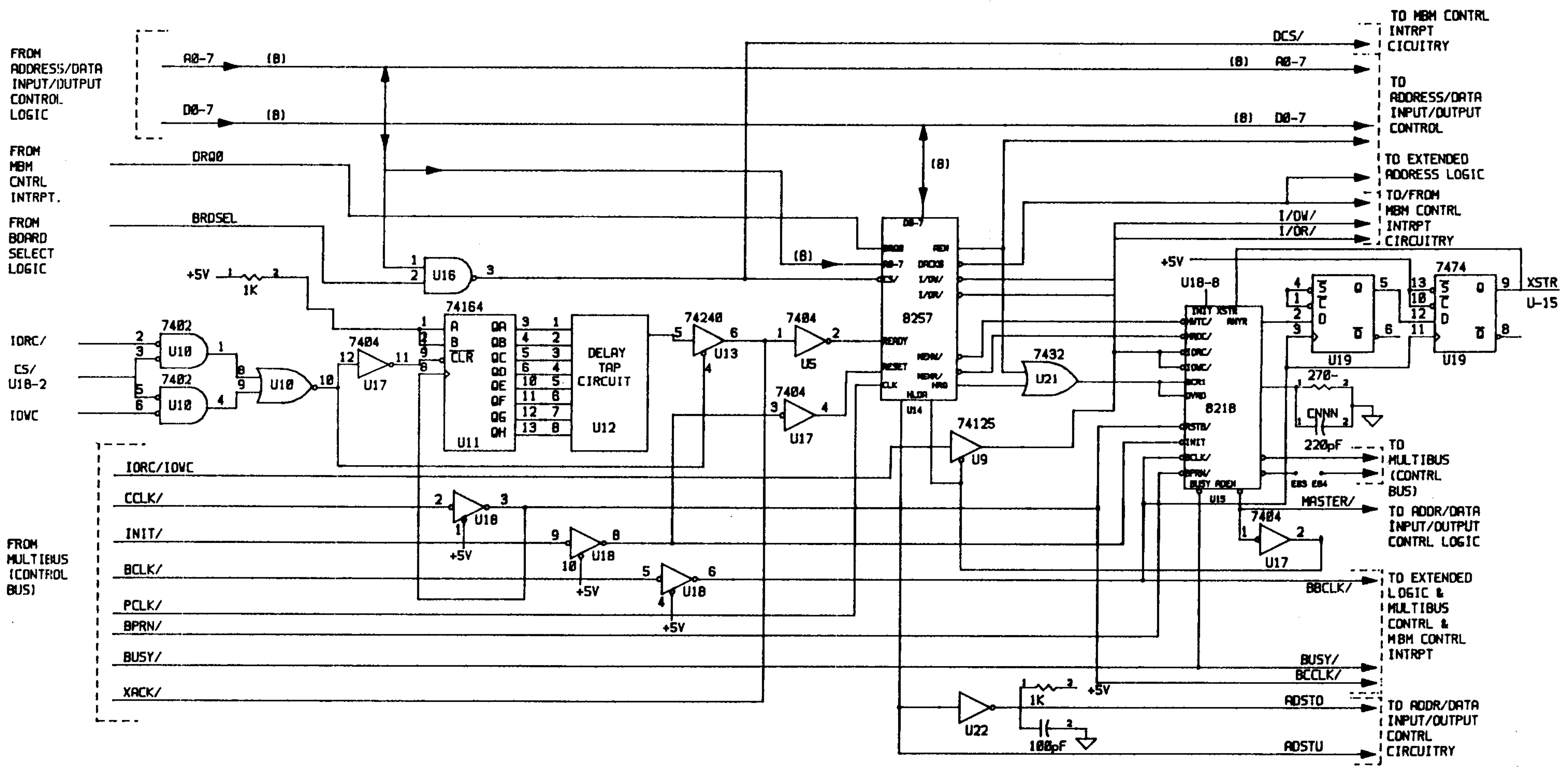
Fig. 17c.- Bubble Controller Interface Board

done to insure that the system is completely free of spurious signals on the data and control lines that interface directly to the BDC: decoupling capacitors should be used on all integrated circuits, all line lengths should be as short as possible, and the data lines to the BDC should be in twisted pairs. The system will not function properly if any spurious signal activity is present.

The address decode logic is necessary for the selection of the bubble device controller for the issuance of commands, the DMA circuitry for the transmission of data, and the proper storage location within the memory system. Selection of the BCIB is achieved using the board select logic shown in Fig. 18. This logic sets up the base address for all of the devices that are to be addressed on this interface board. Selection of the bubble device controller is done by setting A3(=ADR3/) and A4(=ADR4/) true and combining with the BRDSEL signal, as depicted in Fig. 18, to yield a hexadecimal address of BE for the BDC data port and BF for the BDC command/status port. To select the DMA controller, the address bit A4(=ADR4/) is set true and combined with the BRDSEL signal from the board select circuitry via the logic shown in Fig. 19. The resulting addresses for the DMA controller ports are: channel 0 of the address port is A0H, channel 0 for the Terminal Count port is A1H, and the mode/status port is A8H. Further details for the specific operation of the DMA circuitry can be found in reference 14. Also shown in Fig. 19 is the data transfer acknowledge logic, U11 and associated circuitry, that is necessary to provide the proper handshake timing between the BDC and the DMA circuitry. Selection of a particular memory location within the system is done via the Group Select circuitry shown in Fig. 20.

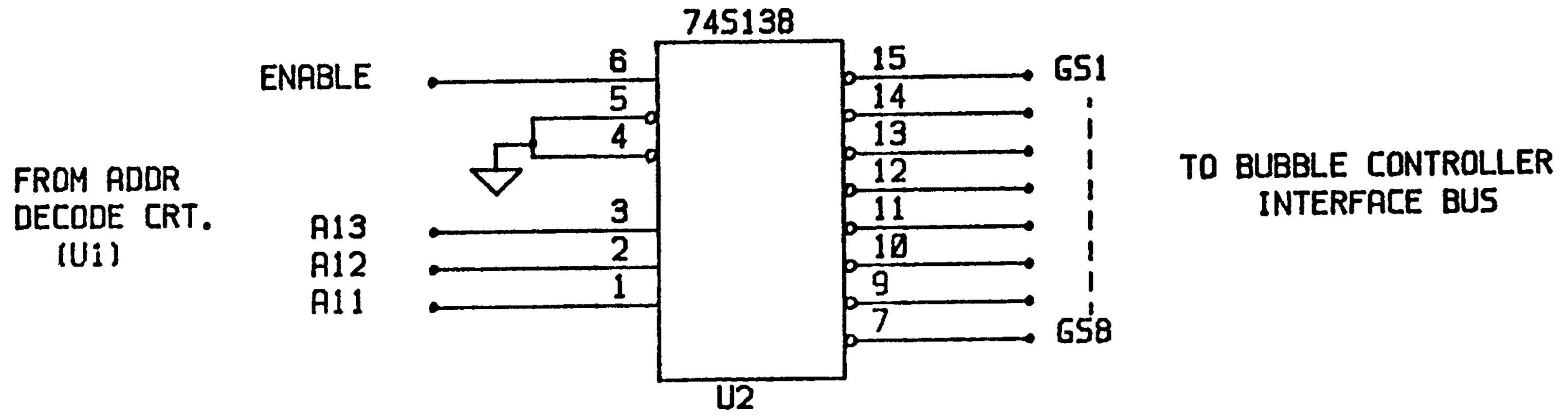


BOARD SELECT LOGIC
FIGURE 18



FROM ADDRESS/DATA INPUT/OUTPUT CONTROL LOGIC
A0-7
D0-7
(8)
FROM MBM CNTRL INTRPT.
DRQ0
FROM BOARD SELECT LOGIC
BRDSEL
+5V
1K
IORC/
CS/ U18-2
IOVC
7402
U10
U16
7404
U17
74164
U11
DELAY TAP CIRCUIT
U12
74240
U13
U5
8257
U14
7432
U21
74125
U9
8218
U15
U18-8
7474
U19
XSTR
U-15
270-
CNNN
220pF
U18
FROM MULTIBUS (CONTROL BUS)
IORC/IOVC
CCLK/
INIT/
BCLK/
PCLK/
BPRN/
BUSY/
XACK/
U22
100pF
DCS/
TO MBM CONTRL INTRPT CICUITRY
TO ADDRESS/DATA INPUT/OUTPUT CONTROL
TO EXTENDED ADDRESS LOGIC
TO/FROM MBM CONTRL INTRPT CIRCUITRY
I/OW/
I/OR/
TO MULTIBUS (CONTRL BUS)
MASTER/
TO ADDR/DATA INPUT/OUTPUT CONTRL LOGIC
BBCLK/
TO EXTENDED LOGIC & MULTIBUS CONTRL & MBM CONTRL INTRPT
BCCLK/
ADSTO
ADSTU
TO ADDR/DATA INPUT/OUTPUT CONTRL CIRCUITRY

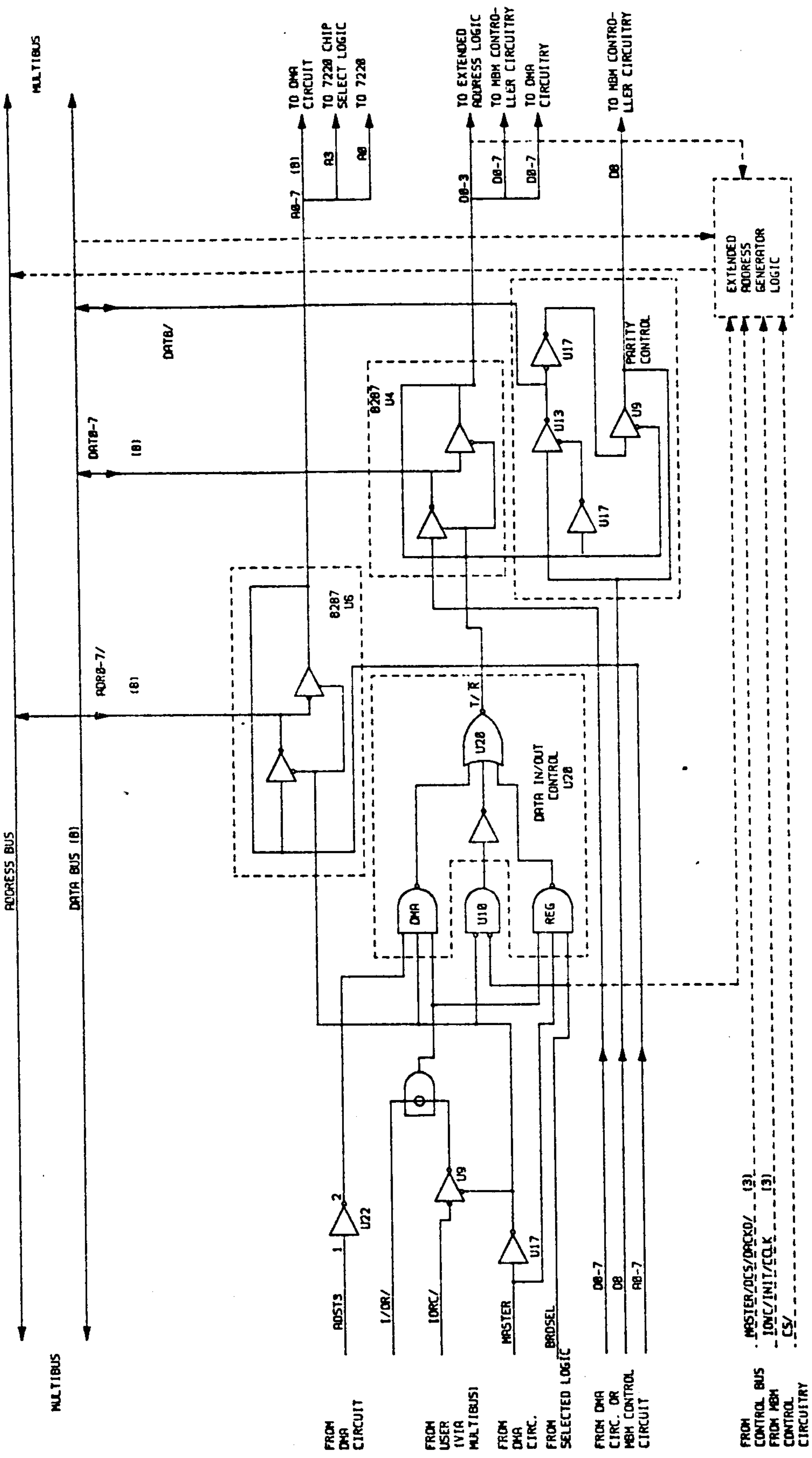
DMA CONTROL CIRCUITRY
FIGURE 19



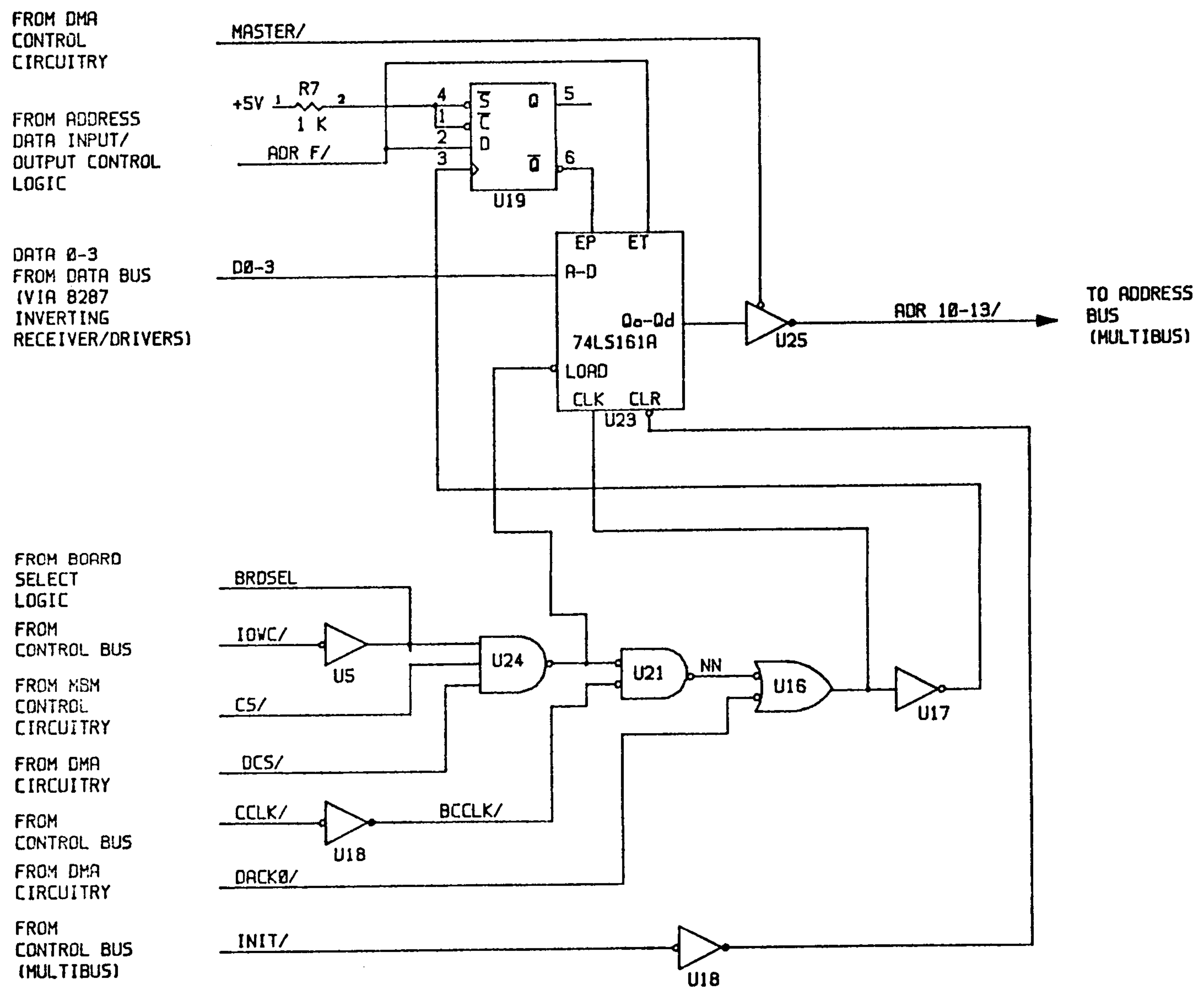
GROUP SELECTION CIRCIUT
FIGURE 20

The data/address input/output control logic, as shown in Fig. 21 and 22, consists of drivers and receivers which are controlled through logic (U20) and the extended address circuitry (see fig. 22) which is needed for use in conjunction with 16 bit processors. The data type is determined by the desired mode of operation and controlled by the data/address input/output control logic. Operation of the circuit in non-DMA mode causes the address bits to be distributed directly to the various circuits on the BCIB. However, during DMA transfer, the addresses are generated by the DMA and extended address circuitry, and placed onto the Intel MULTIBUS. The extended address logic handles the generation of the additional four higher-order address bits that are necessary for interfacing with the 20-bit addressing that is utilized by the Intel 8086 16-bit processor. Details on the specific functionality of the extended address logic can be found in reference 14. Control of the drivers for data transfers falls into several categories: data being transferred (in the DMA mode) from the user to the bubble memory and vice versa, reading/writing to the registers of the BDC, and reading/writing to the bubble memories when not in DMA mode. The data in/out logic in U20 control the drivers in U4 for reading/writing (DMA) to the bubble memory or the BDC registers. The drivers in U4 are disabled for data transfers in non-DMA mode when the board is de-selected. Data transfers from the bubble memory to the user are accompanied by a parity bit (D8) which is generated by the BDC. Conditioning the output of U20 to enable the drivers in U4 causes the driver in U13 to be enabled so the D8/ is driven to the processor (SCM) along with the other data lines, D0/-D7/.

The interface control circuitry for the EEPROM consists mainly of routing signals from the single board computer bus and parallel port, and



ADDRESS/DATA INPUT/OUTPUT CONTROL LOGIC
FIGURE 21

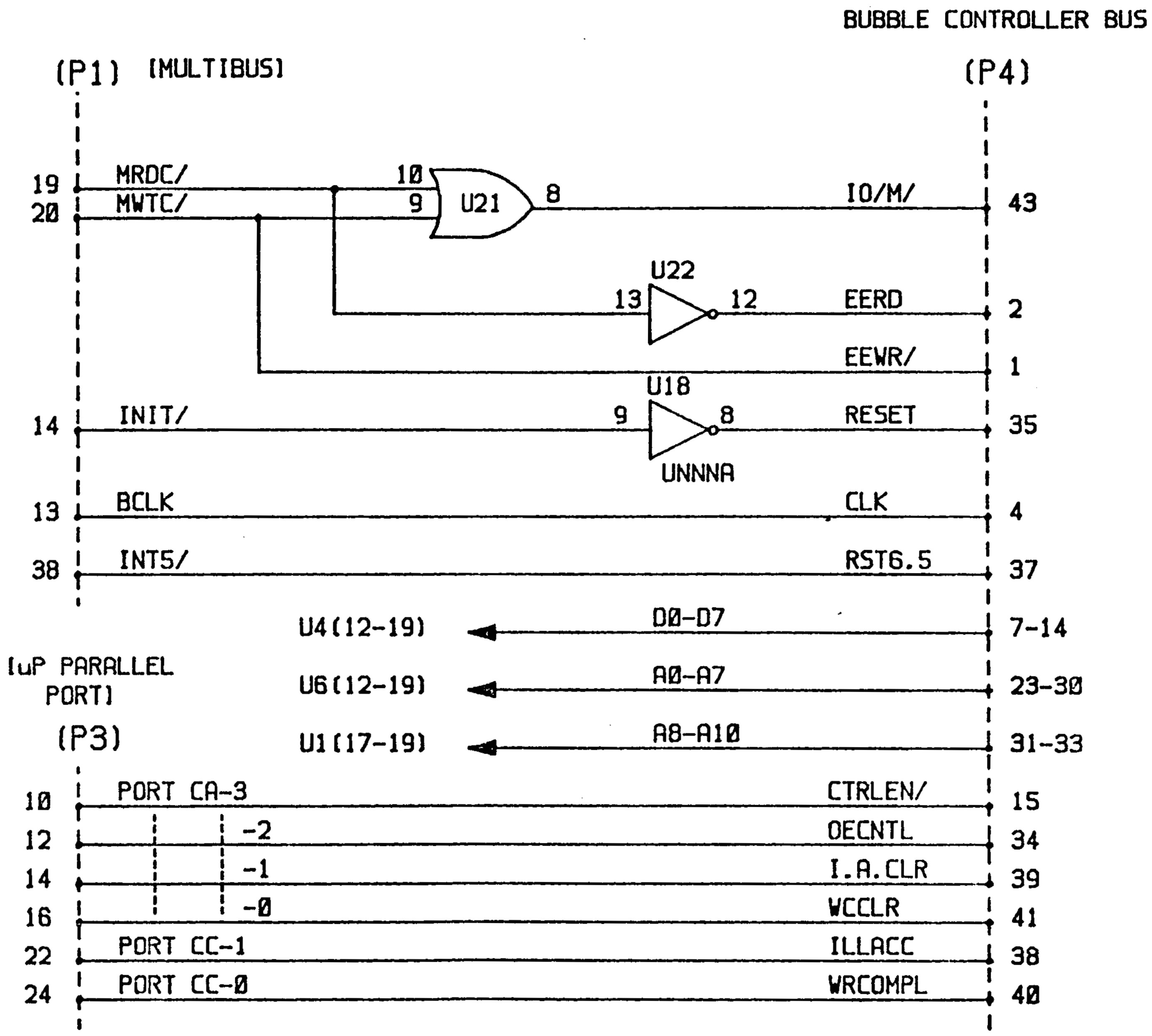


EXTENDED ADDRESS LOGIC
FIGURE 22 [REF14]

the appropriate decode/buffer circuitry, to the EEPROM port on the BCIB. Some additional logic is also incorporated to generate the proper signals for interfacing with the EEPROM and its control circuitry. Fig. 23 shows a diagram of the logic and data/address routing that is on the BCIB. A parallel port on the single board computer is used to generate most of the handshaking signals that the EEPROM control circuitry requires. These signals (CRTLEN/, OECNTL, I.A.CLR, WCCLR, ILLACC, WRCMPL) are software interpreted and controlled by the controlling software for the Memory Module system. Other control signals are routed directly from the single board computer bus or from the address/data buffers on the BCIB.

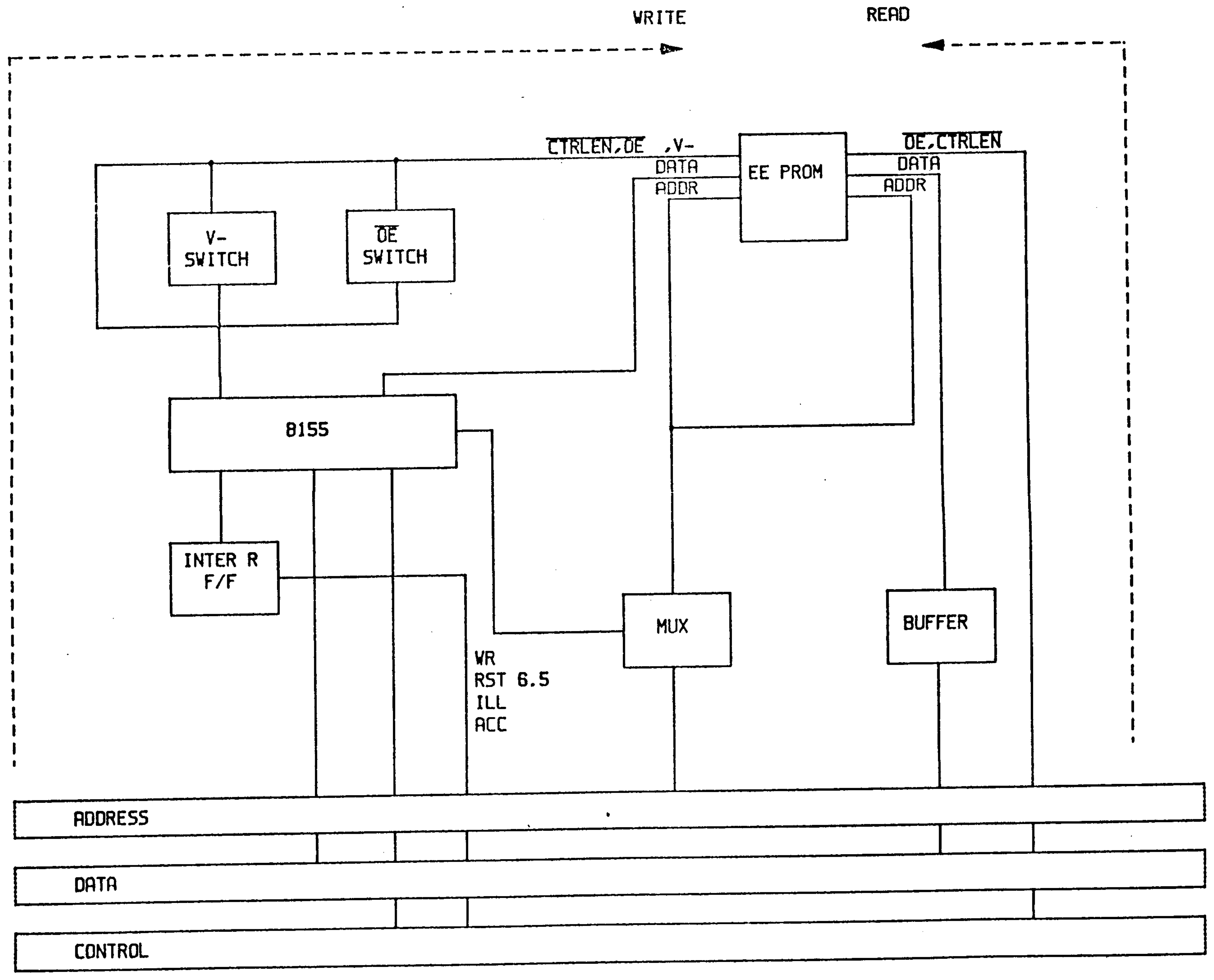
EEPROM Control Board

A faster initialization technique is needed for bubble memory systems when power strobing of arbitrary memory segments is utilized to minimize the power requirements of the system. In the context of this development, a separate commercially available EEPROM board was used to test the concept of fast initialization. Further discussion of the fast initialization technique is presented in a later section of this paper. The EEPROM board used in this experiment is mounted on the computer card cage (see fig. 14). A pin-for-pin connection is established between connector P4 on the BCIB and the EEPROM board connector. Figs. 24a, 24b and 24c, show a block diagram, aschematic and a photograph, respectively, of the EEPROM controller design. This design is chosen for its software flexibility, minimal hardware overhead, real-time processing capability, and high read access speeds for the EEPROM device. Reference 12 provides a more complete description of the design and functionality of the EEPROM controller board.



EEPROM INTERFACE CONTROL CIRCUITRY

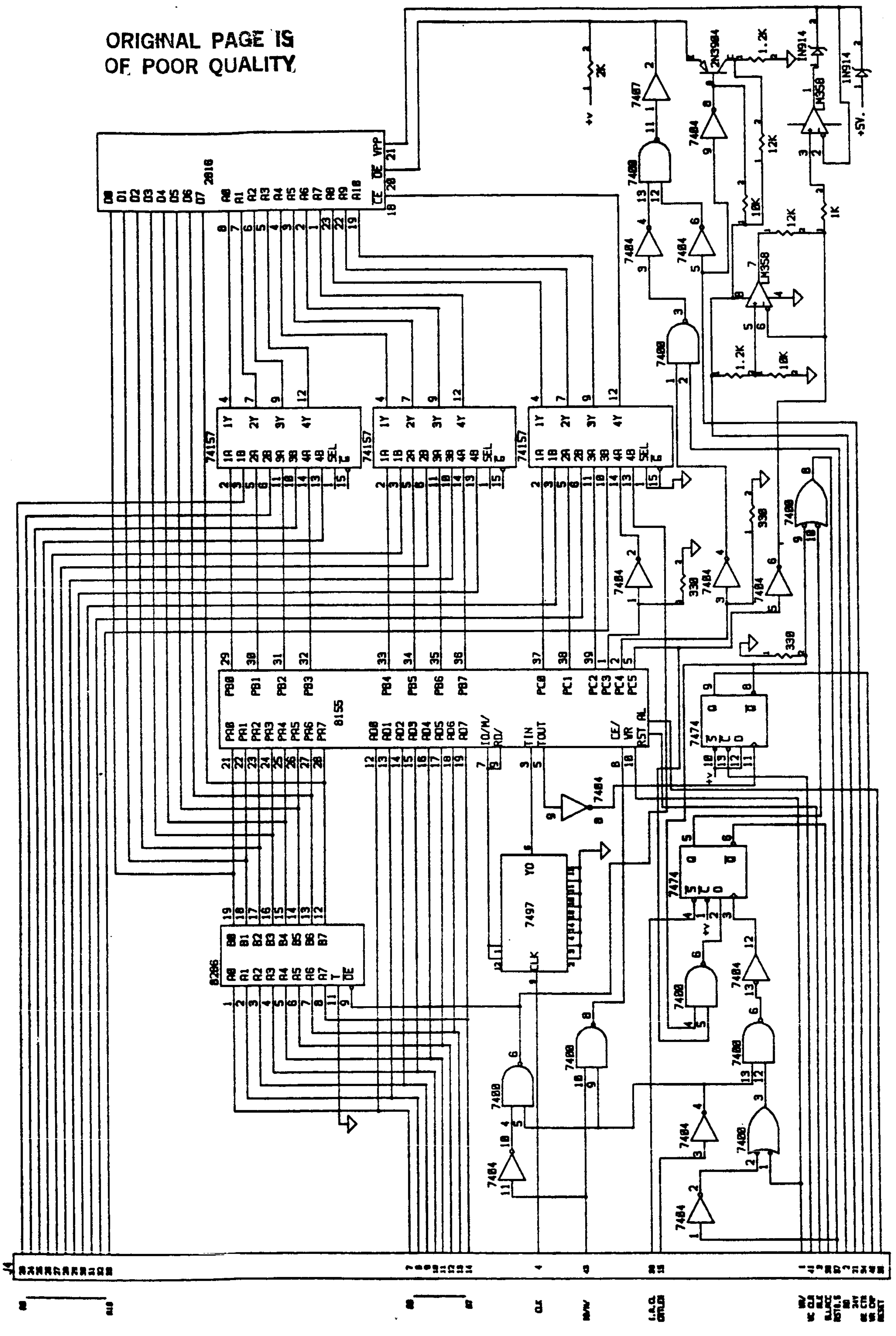
FIGURE 23



EE PROM CONTROLLER BLOCK DIAGRAM

FIGURE 24a [Ref12]

ORIGINAL PAGE IS
OF POOR QUALITY



EEPROM CONTROLLER SCHEMATIC
FIGURE 24b (Ref12)

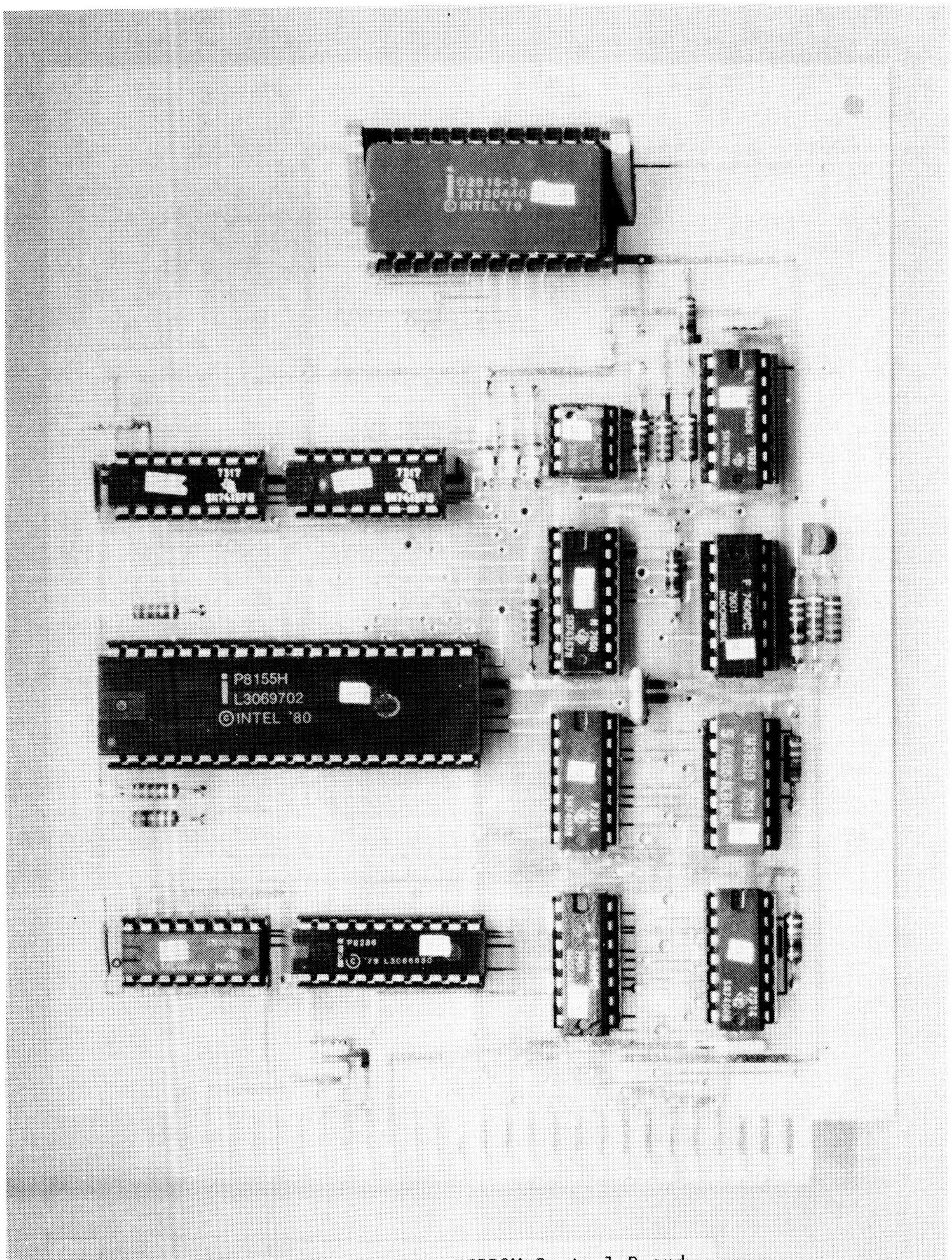


Figure 24c. EEPROM Control Board.

ORIGINAL PAGE IS
OF POOR QUALITY

Bubble Device Controller Board

The Bubble Device Controller Board (BDCD) is located in the first card slot in the card cage. It is a two-sided printed wire board having active and passive components located on the front side, while interconnections (plated through holes and wire routing) are interspersed on both sides, as shown in Fig. 25. A schematic of the board is shown in Fig. 26. The board has two electronic ports. The one located at the top region of the board is a ribbon type connector adaptable to flat ribbon connections. This provides signal routing for the microprocessor interface which is located in close proximity to the card cage, but a considerably long distance for parallel routing of high speed data. Cable length is maintained at 15 inches. In an earlier design when the cable length was selected at 24 inches, considerable noise problems were incurred. Inductive and capacitive coupling introduced many electrical paths for transient glitches, formed by high speed logic functions, to be coupled to adjacent signals. This, in turn, introduced confusion into the active bubble device controller chip. Two approaches were considered to reduce these anomolies. One, employing twisted-pair flat ribbon cable, as opposed to a flat type, and secondly, maintaining cable length as short as reasonable without restricting the test and evaluation conditions. To maintain shielding, each twisted-pair has one lead connected to the power supply ground and is referenced to common only on the controller board side.

The second port is a connector (56-pin edge type) located on the opposite end of the board. When connected into the mating connector (backplane) all output functions from the active device (large scale

ORIGINAL PAGE IS
OF POOR QUALITY

NASA
L-84-14,063

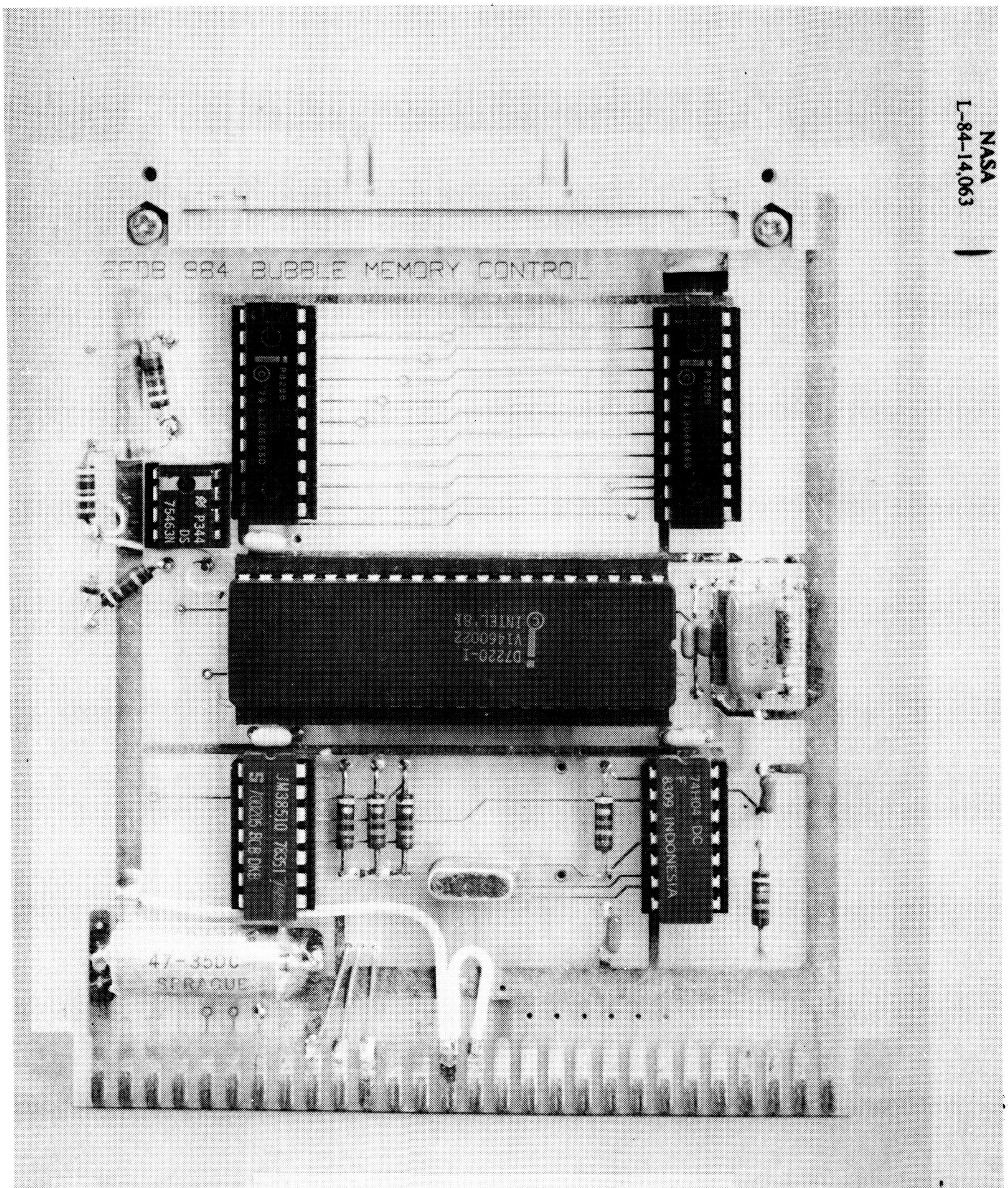
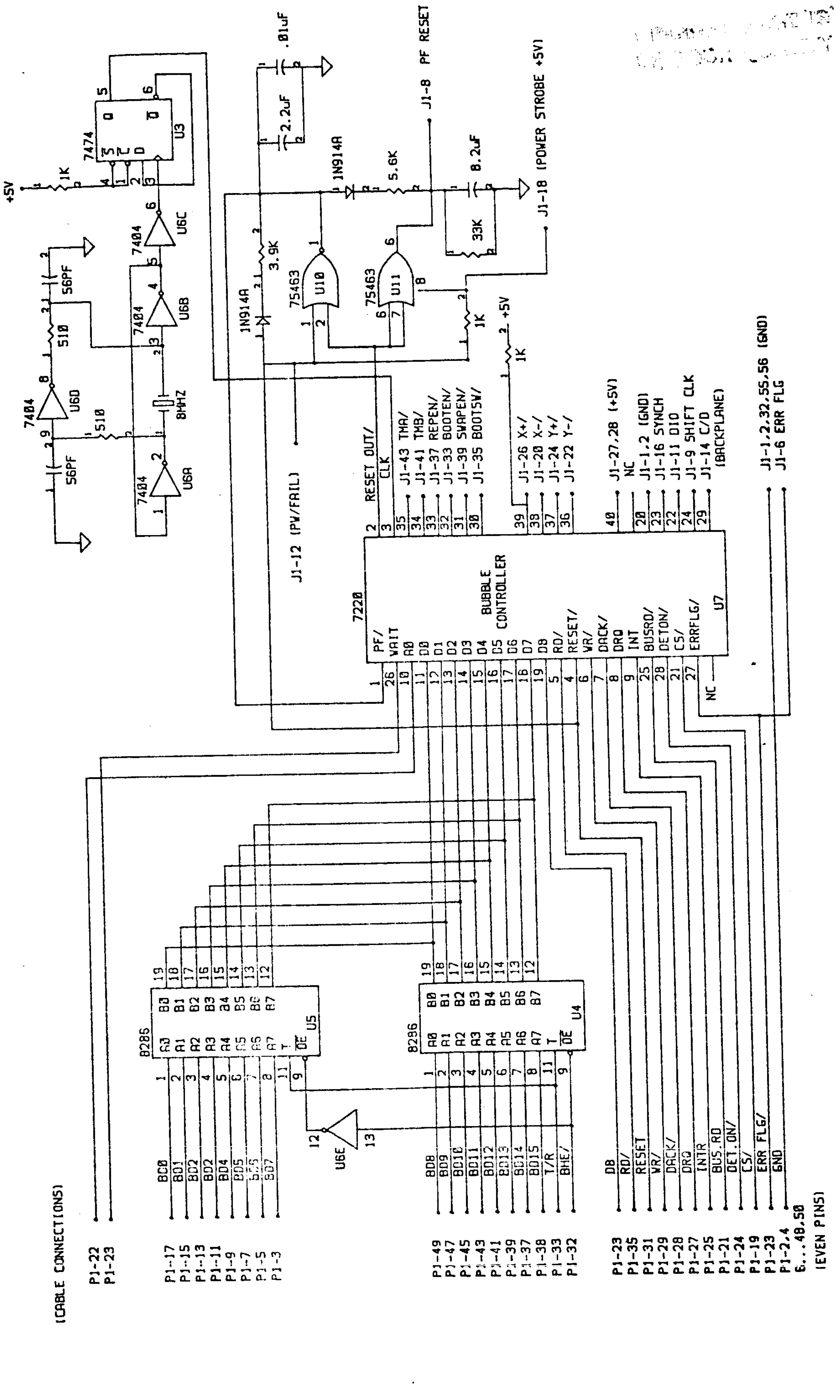


Fig. 25.- Bubble Device Controller Board



BUBBLE DEVICE CONTROLLER
FIGURE 26

integrated circuit) have commonality with the storage units and power strobe circuitry. The controller board embodiment is populated with the following functions: 1) tri-state logic switches which separate address and data functions, while maintaining signal integrity which was degraded as a result of the microprocessor interface, and 2) a master clock which establishes timing and synchronization for the controller and all associated circuitry. Originally, this clock was located on the microprocessor/buffer stage and was found to create cross talk and glitches which coupled to adjacent control and data lines.

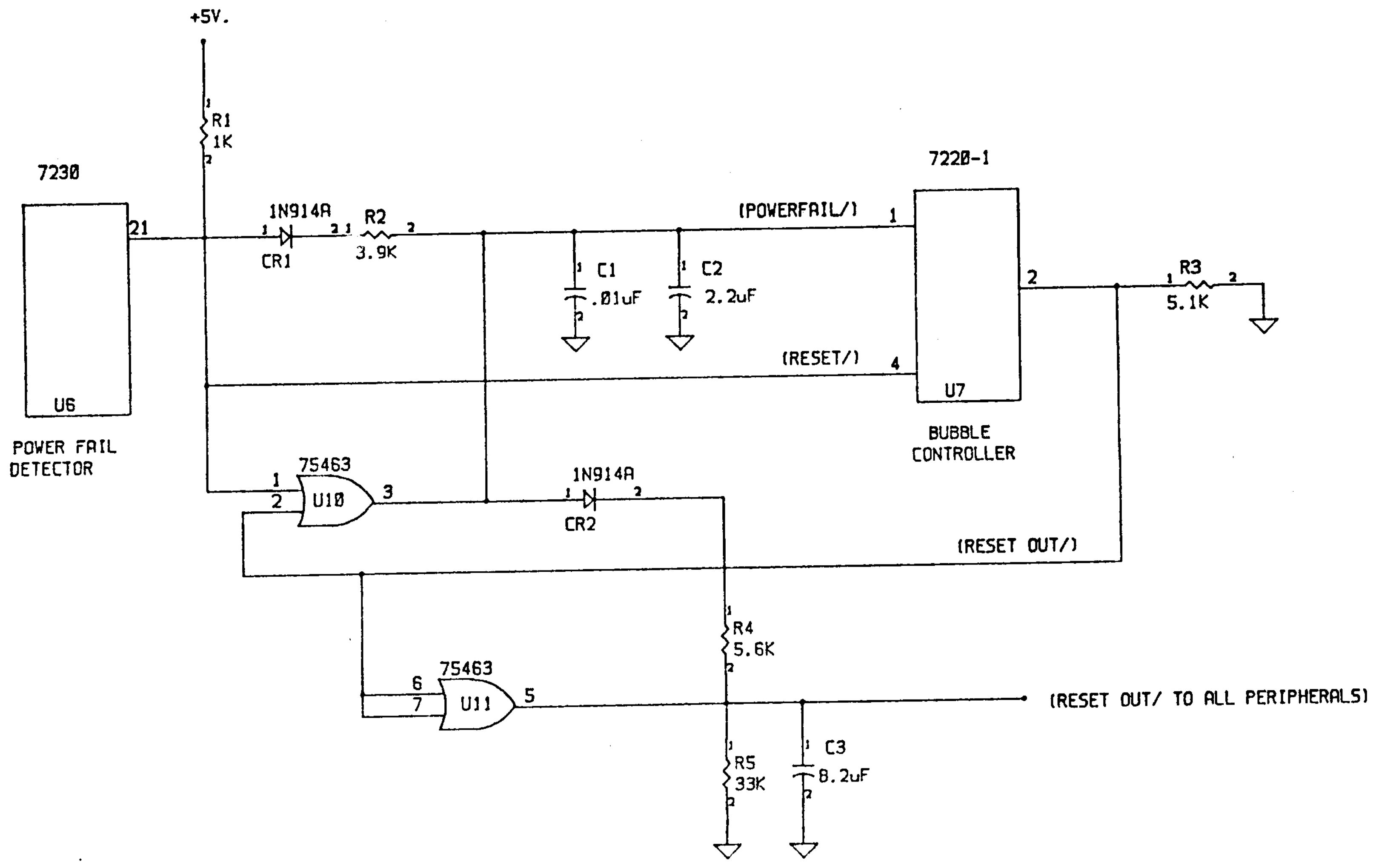
The heart of the bubble device controller is an integrated circuit performing all the tasks sent from the microprocessor control system as shown previously in Fig. 13. In the memory control application, the BDC is run at a clock rate of 4 Megahertz. The input/output signals (commands, data, status) share commonality on the BDC lines D0-D7 by microprocessor multiplexing. The remaining functions (write, read, INTR, etc.) are individual lines as shown in the circuit diagram of Fig. 26.

The final circuit on the board is a power detection circuitry as shown in Fig. 27. This is needed to preserve the data storage integrity. Data transfer in a bubble memory system requires the memory chip to be located in the center of two orthogonal rotating magnetic fields. When a write/ read command is initiated several events occur simultaneously. The controller generates timing phases for the magnetic field coil drivers, and replicate and transfer functions which guide the bubble along the storage track locations. When in a write, up to eight generators are activated to code data accordingly. In all cases, synchronization must be maintained if the data is to be preserved. Each movement of the data from one track position to the next occurs by rotating two magnetic fields periodically.

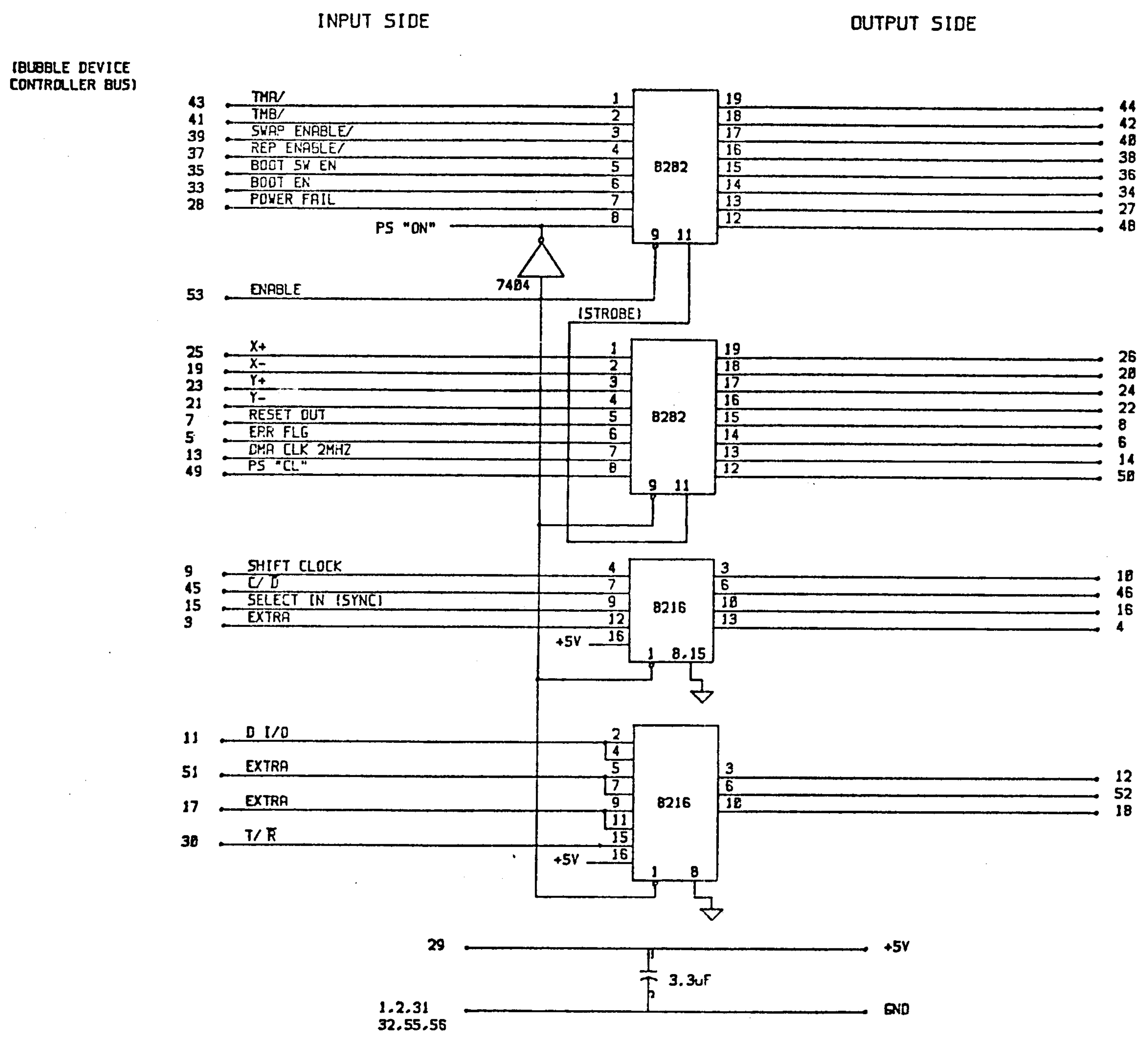
In the case of a write function, a bubble is generated whenever the write generator is activated at a precise time for each field rotation. For the read function, a precise time is selected whereby the magnetic field of the bubble is detected electrically for data retrieval. In both read/write, the field coils cannot be turned off before reaching a designated stop position. If turned off too soon, the bubble data is scrambled and is not accurately retrievable. A power fail detector reduces the risk associated with power failure while fields are rotating. To operate the card cage, power is applied (+12, +5 volts) to the memory controller chip and the power fail circuitry which is interspersed between the BDCB and individual SU's. The power fail detector initiates control of the active chip to insure that no field rotations occur without specific instructions applied to the memory storage peripherals. If, during a write/read operation where the fields are rotating, a power failure should occur, the detector responds accordingly, it sends anon-maskable interrupt to the controller which responds immediately by removing its current operating instruction, but allows timing signals to continue until both fields have reached the designated stop positions. When this occurs, the controller removes all logic functions and powers down in an orderly manner.

Isolation Buffer Board

A schematic of the Isolation Buffer Board (IBB) is shown in Fig. 28a and a photograph of the actual board is shown in 28b. The purpose of this board is to isolate each memory board from the other memory boards that are connected to the bus. Only the appropriate signals are allowed to filter



POWER DETECTION CIRCUITRY
FIGURE 27



ISOLATION BUFFER BOARD
FIGURE 28a

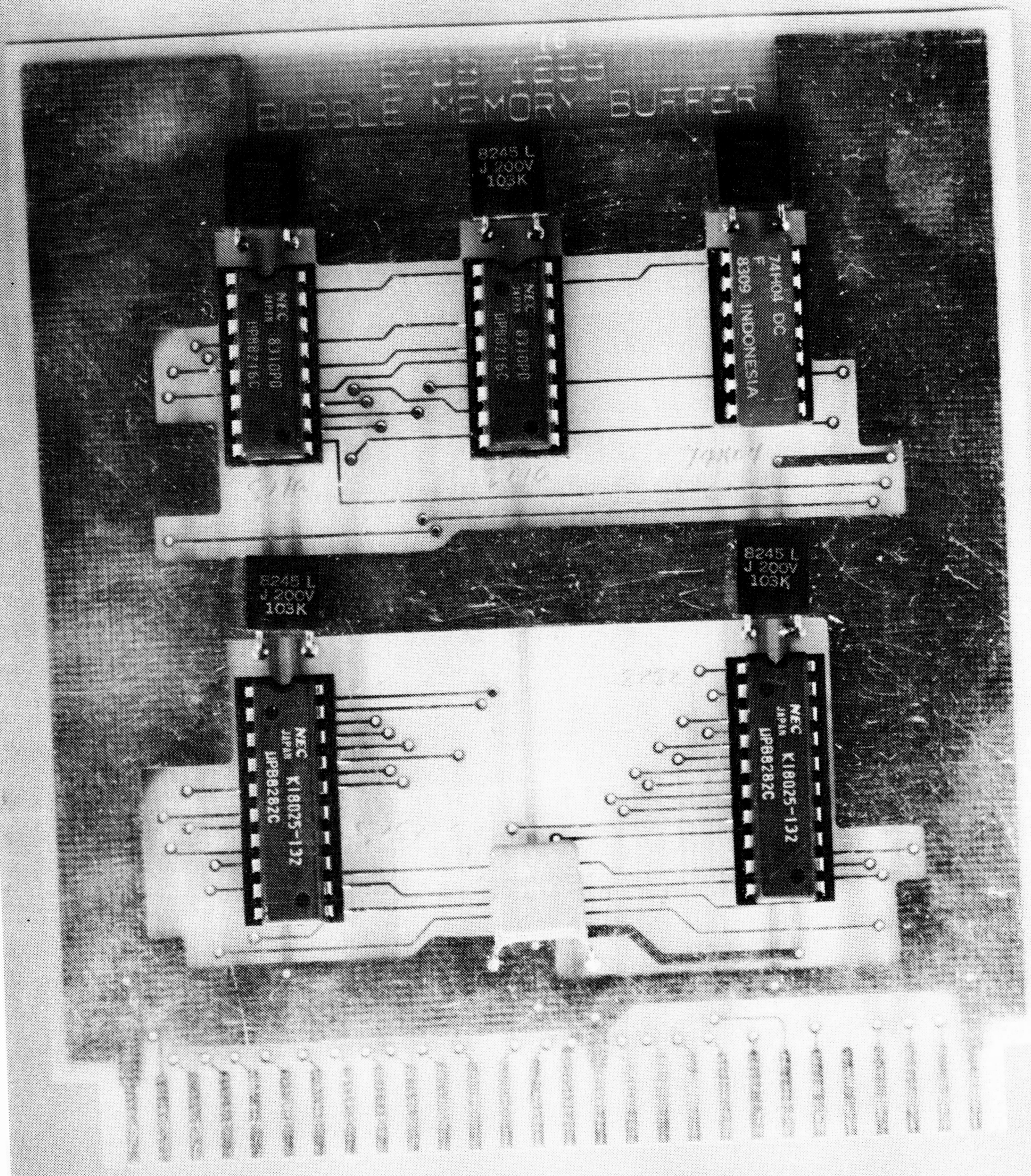

BUBBLE MEMORY BUFFER
8245 L J 200V 103K
NEC 8310PD JAPAN UPB8216C
74H04 DC F 8309 INDONESIA
NEC K18025-132 JAPAN UPB8282C

Fig. 28b.- Isolation Buffer Board

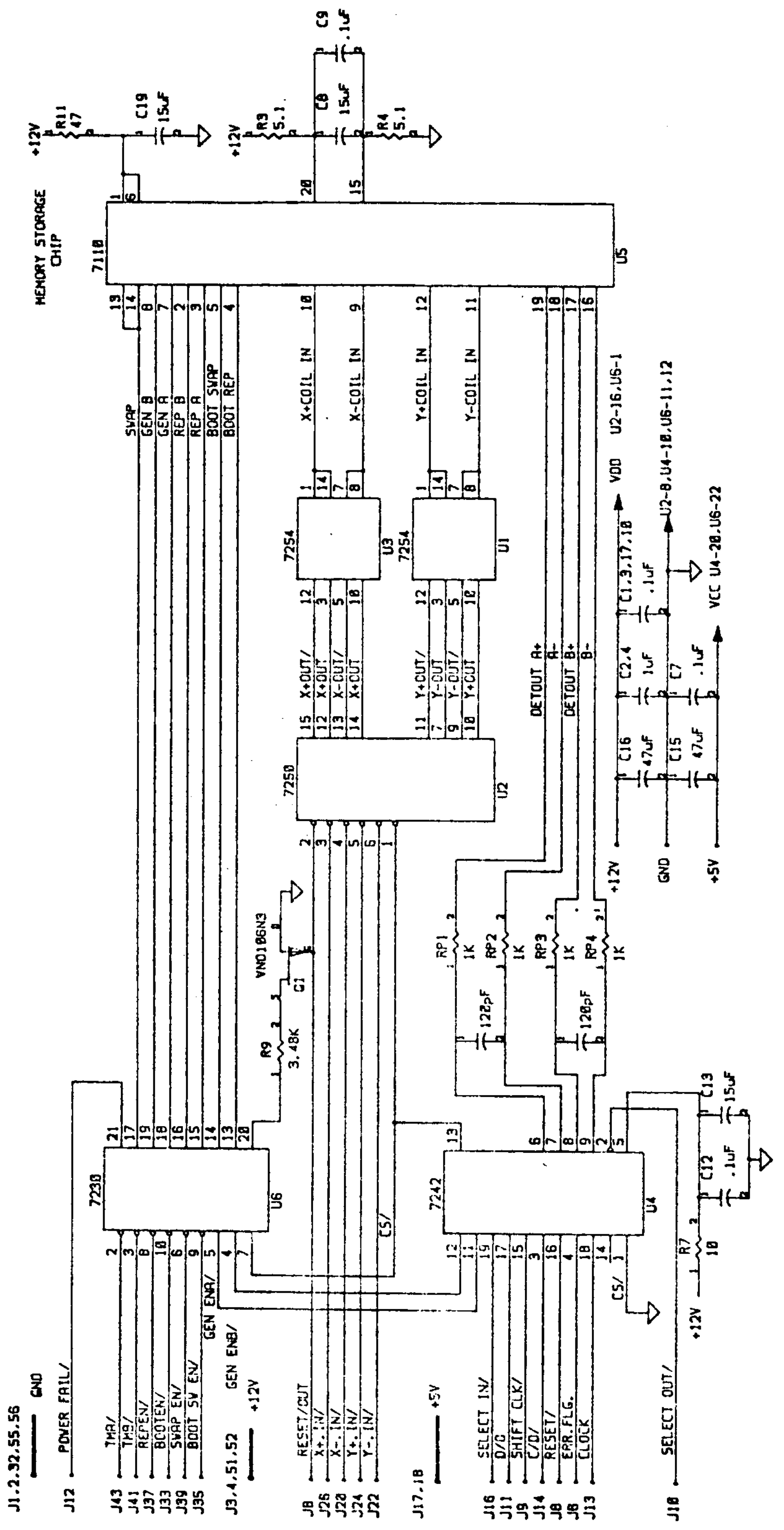
through to the selected board, protecting the other boards from receiving any spurious signal activity that may be present on the bus. The IBB also keeps the memory boards from generating any spurious activity that could interfere with the functioning of the selected memory segment.

Memory Storage Unit

Movement of data to and from a memory site at bit rates that can exceed one megabit per second, while employing a serial flow type of device that is relatively slow in accepting the data, requires a number of these devices to be configured in a parallel flow operation. Each storage unit (1 megabit) accommodates one bit of each data byte to achieve this process. Eight memory storage boards, previously shown in Fig. 14, are strapped together by a common bus rail (backplane of card cage) that is common with an active controller (BDC) capable of driving all units simultaneously.

The bubble memory chip and its field coil frame embodiment are located at the approximate center section of each printed circuit card. The printed circuit card has same physical dimensions as previously described in the memory controller section. All boards have identical electrical components and printed wire fabrication techniques. Fig. 29 and 30 show a schematic and a photograph, respectively, of a memory storage board.

One of the major concerns in board layout is the noise associated with magnetic field rotations. In conventional bubble domain detection methods, minute data signals are retrieved from a composite envelope containing many noise interferences. Two primary sources of noise are Faraday pickup ($d\phi/dT$), and magnetoresistor imbalance. Faraday pickup is proportional to



MEMORY STORAGE BOARD
FIGURE 29

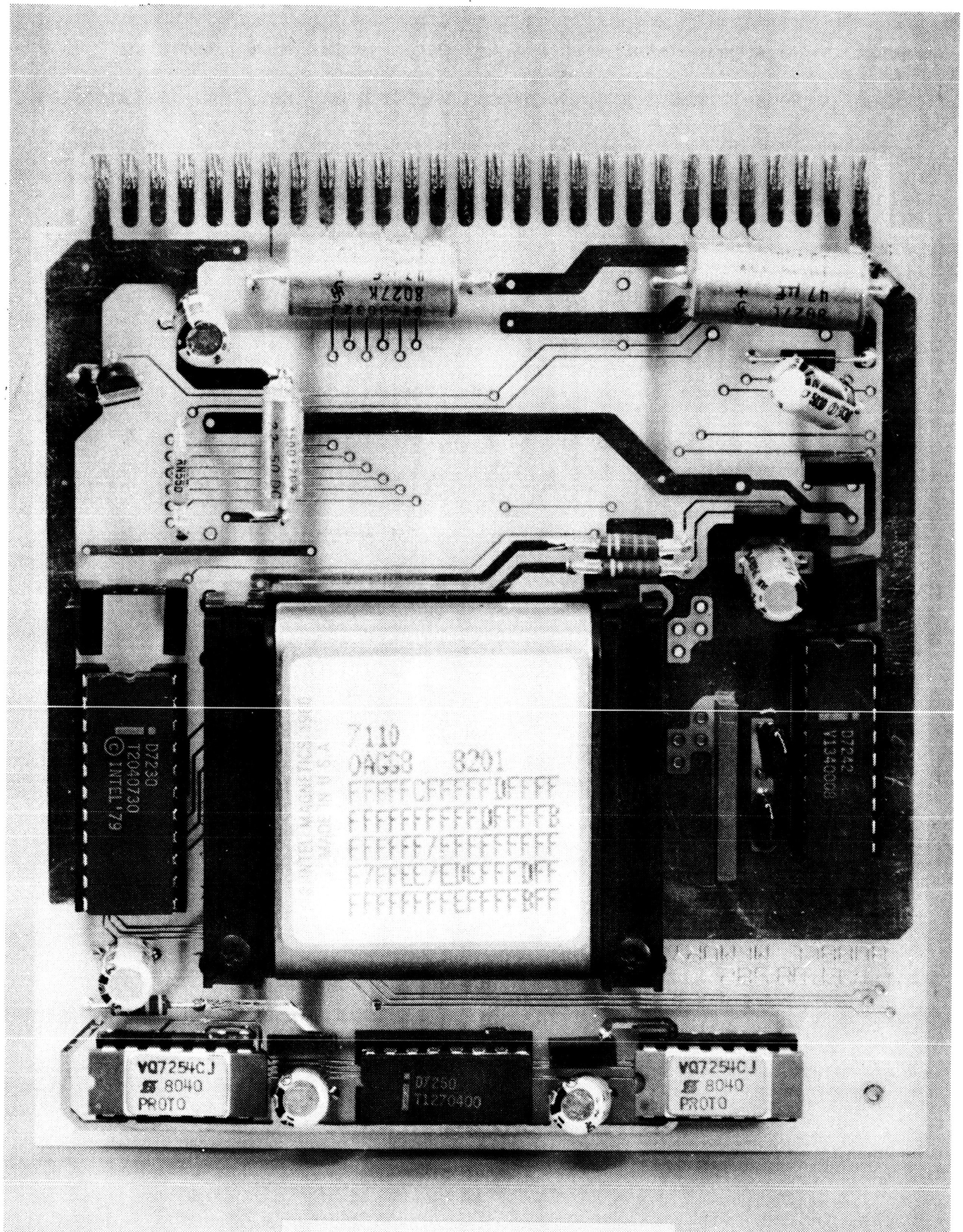


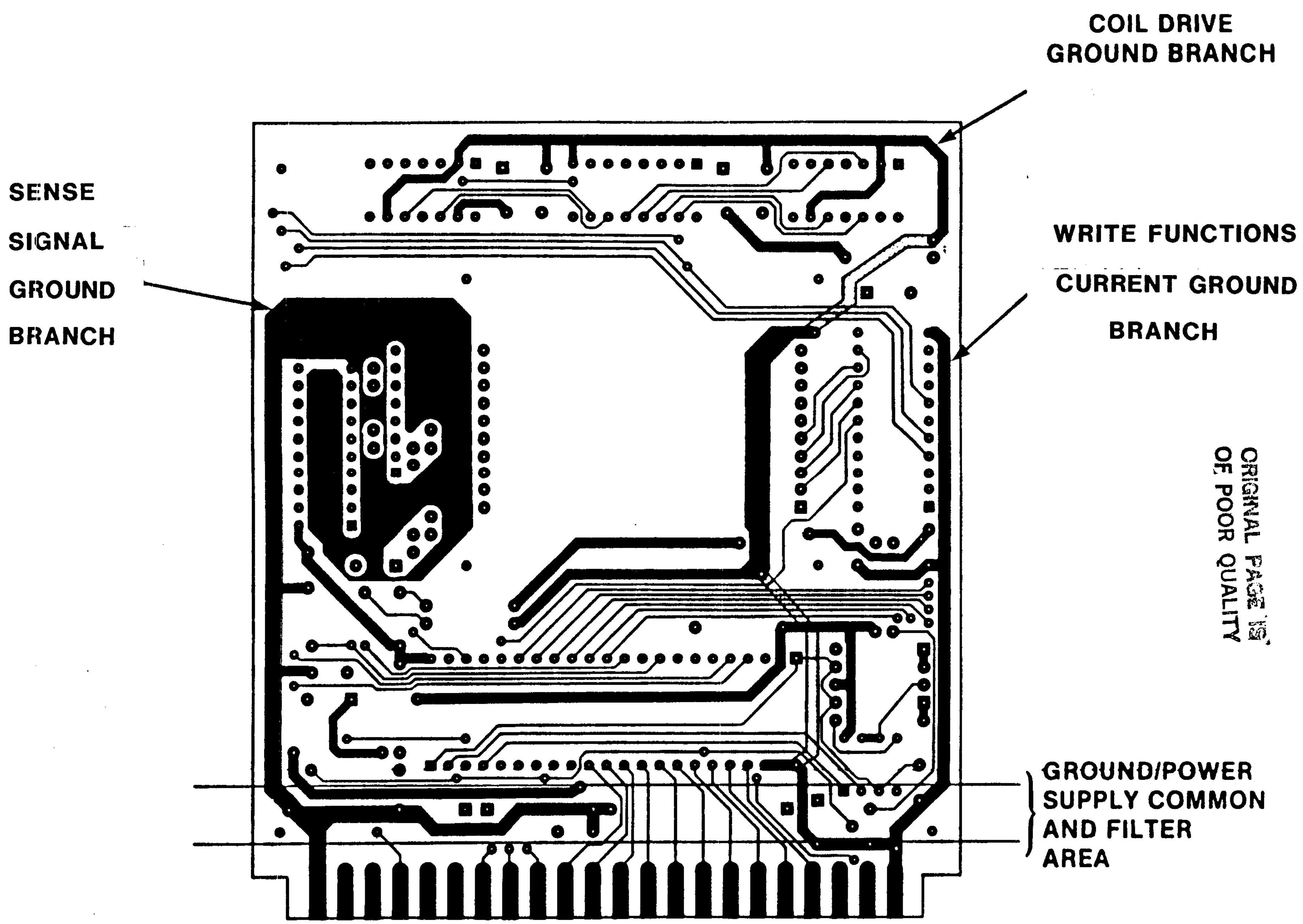
Fig. 30.- Memory Storage Board

frequency and intensity of the field rotation, and also the location and orientation of signal leads within the field. To minimize the noise pickup, the sense circuitry is located some distance (see fig. 31a and 31b) from the field coil frame, and is completely surrounded with a Faraday shield (ground plane) that is referenced to the power supply ground separate of coil currents. The sense lines are balanced bridge resistors and the lead length is held to a minimum value. This insures that transient noises are balanced as opposed to unbalanced (where the common rejection of the sense detector does not provide adequate discrimination).

When the magnetic fields are rotated, bipolar (active) switches periodically turn on/off for each half cycle of operation. This generates a step function (di/dt) followed by an inductive ringing of power supply/ground rails. To insure adequate isolation, both the field coil circuitry and the detector circuitry have separate power rails.

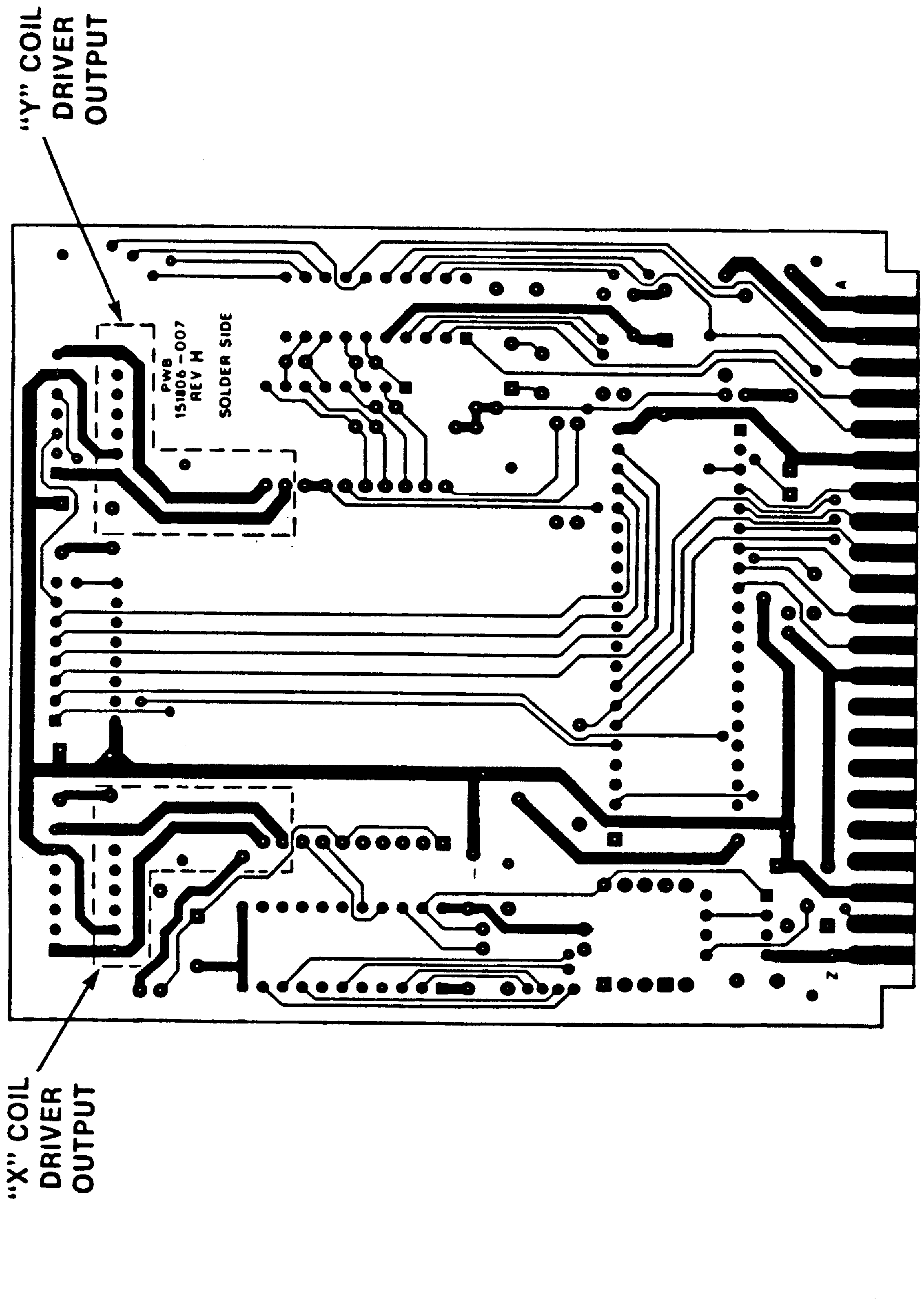
Power Strobe Circuitry

In many electronic systems, it is necessary to operate within a minimum power constraint. Two examples are spacecraft electronics where the power available is limited and a closed or encapsulated unit where heat dissipation is a concern. Often, it is possible to take advantage of the fact that only a small fraction of the electronic circuit is performing a useful function at any given instant. In this case, power is applied only to the part of the circuit needed at that instant. This can be accomplished in a straightforward manner by using active transistor switches in the individual voltage lines. This approach is limited since the V_{CE} saturation resistance of the transistor acts as a series resistance in the power lines and degrades the voltage regulation.


COIL DRIVE GROUND BRANCH
SENSE SIGNAL GROUND BRANCH
WRITE FUNCTIONS CURRENT GROUND BRANCH
GROUND/POWER SUPPLY COMMON AND FILTER AREA

ORIGINAL PAGE IS OF POOR QUALITY

SENSE/SIGNAL GROUND BRANCHES
FIGURE 31a



COIL DRIVER OUTPUT SIGNAL ROUTING
FIGURE 31b

An alternate approach which will maintain precision voltages under varying loads (typically .1% for load regulation) is the use of a voltage regulator that can be turned on and off under logic control. If one employs a three terminal adjustable type (ref. 16), the regulated output V_0 is determined by two resistors, R_1 and R_2, as shown in Fig. 32.

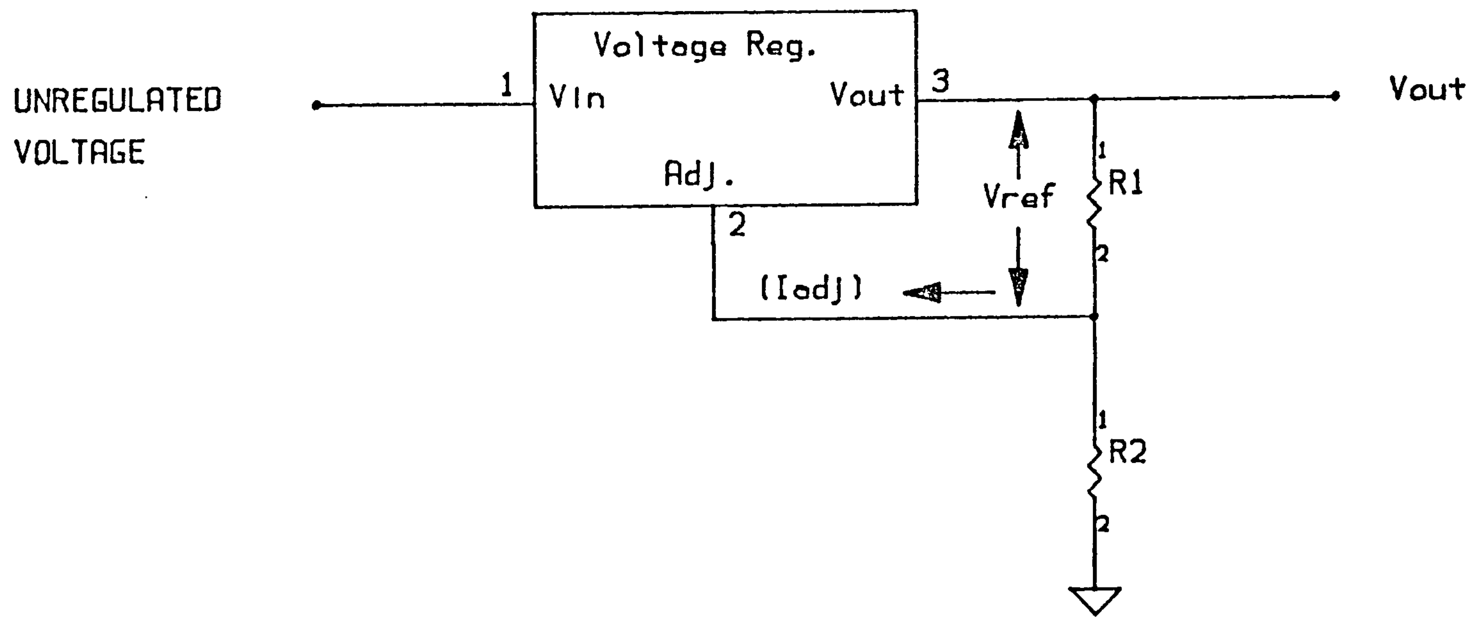
A voltage reference of +1.25 v (V_{ref}) is established between the output side and the adjust terminal. The reference is impressed across R_1 and since a constant I_1 then flows through R_2 the output is determined by

$$Y_{out} = V_{ref}\left(1 + R_2/R_1\right) + I_{adj} R_2$$

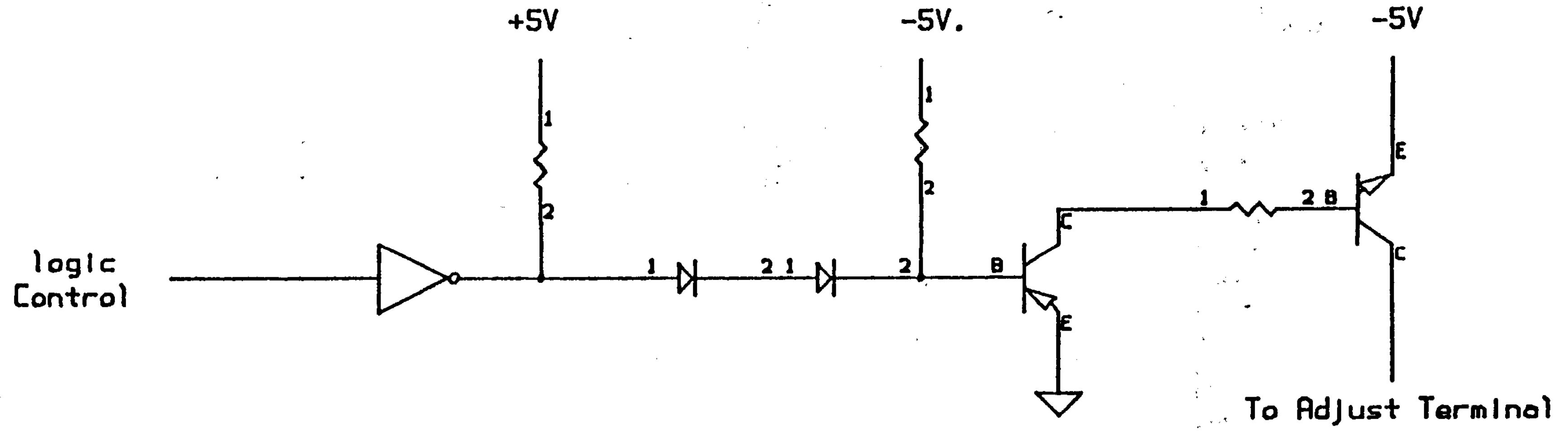
I_{adj} is typically 100 A and represents a small error term for varying line voltages and load changes.

If a negative potential (-1.25 v) is applied through a selected resistor to the adjust terminal, the regulator will turn off. To provide power strobing, the negative potential can be controlled with an NPN source transistor, pnp bias switch and an open collector logic inverter as shown in Fig. 33.

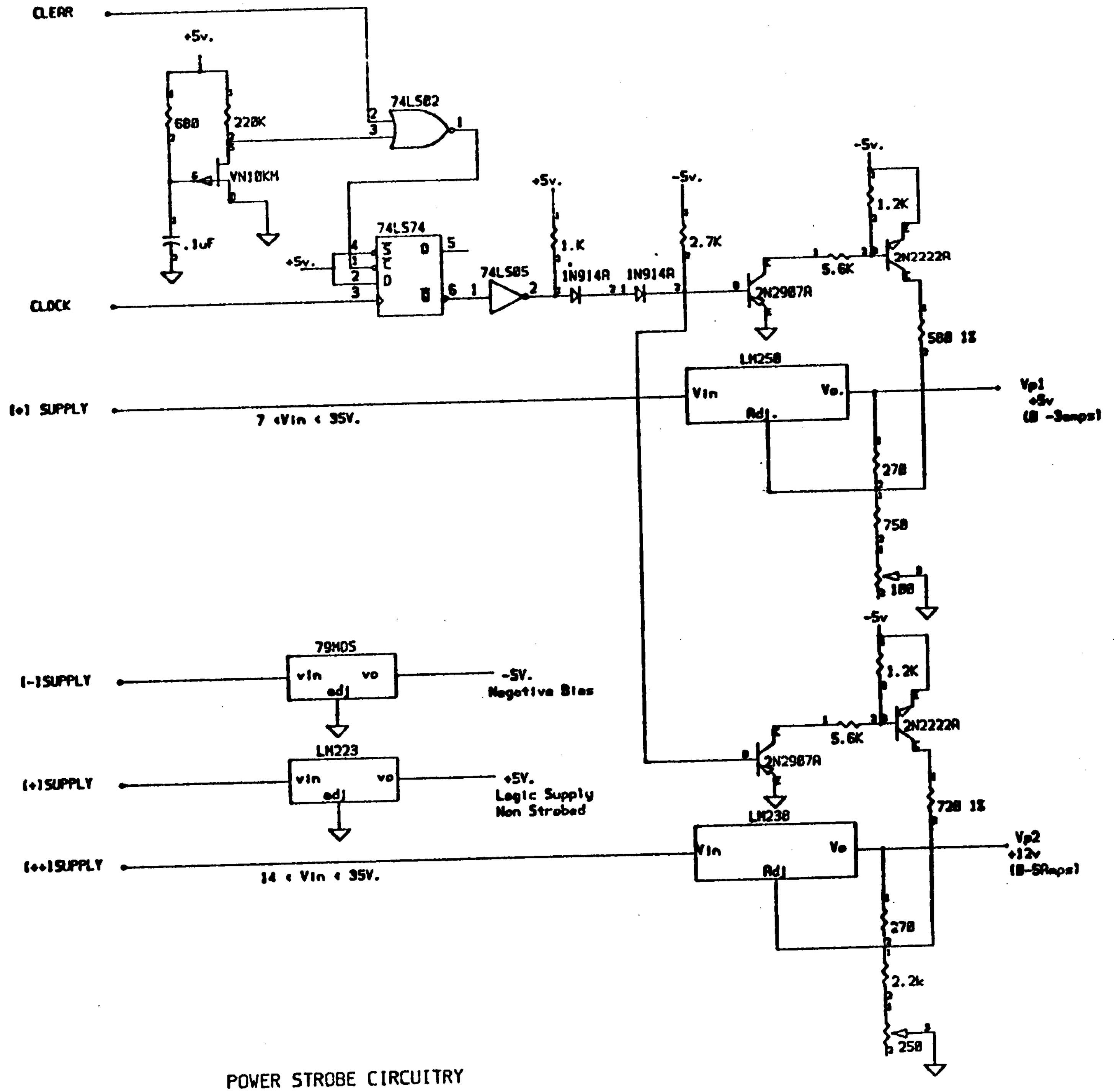
A typical application for this circuit would be the partitioning required in a bubble domain memory storage system. A single partition requires two voltage levels, namely +5 volts ±5% with current demand from 0 to 2 amps and +12 volts ±5% with current demand from 0 to 3.5 amps. Figure 34 shows the final circuitry using low power Schottky TTL as logic control. To initialize each partitioning, a "power-on" pulse developed from a field effect transistor and capacitor-resistor network, insures each flip-flop is properly initialized. Whenever a logic clock activates the



VOLTAGE SELECTION
FIGURE 32



STROBE CONTROL
FIGURE 33



POWER STROBE CIRCUITRY
FIGURE 34

flip-flop, turn on time is typically <10 sec. A logic low to the flip-flop's clear terminal will turn off the regulator. The turn off time varies according to the capacitance connected across V_{out}. A photograph of the actual prototype board is shown in Fig. 35.

Fast Initialization

What It Is and Why It Is Needed

Consideration of power minimization in high capacity bubble memory systems necessitates the activation of only the desired portions of the memory. Power strobing of arbitrary memory segments, for the conservation of available energy resources, requires a capability of fast turn-on. Bubble device architectures, which provide a redundant loop coding, limit the initialization speed. Each time a module is turned on it must be initialized to code the redundant loop information of the selected bubble devices into the BDC.

The functional organization of all bubble devices available commercially today is a major track/minor loop architecture similar to that shown in figure 36 (ref. 10). This architecture is desirable because it provides a shorter access time than the previous serial designs and improves manufacturing yields in high density devices. Each minor loop is for the storage of data, while the major tracks provide the input and output circuitry. Up to 15% of these loops are redundant to allow for processing defects. The location of the defects will vary from device to device, but post-fabrication testing can determine which loops are available for use. The code used to identify the good loops is currently stored in a

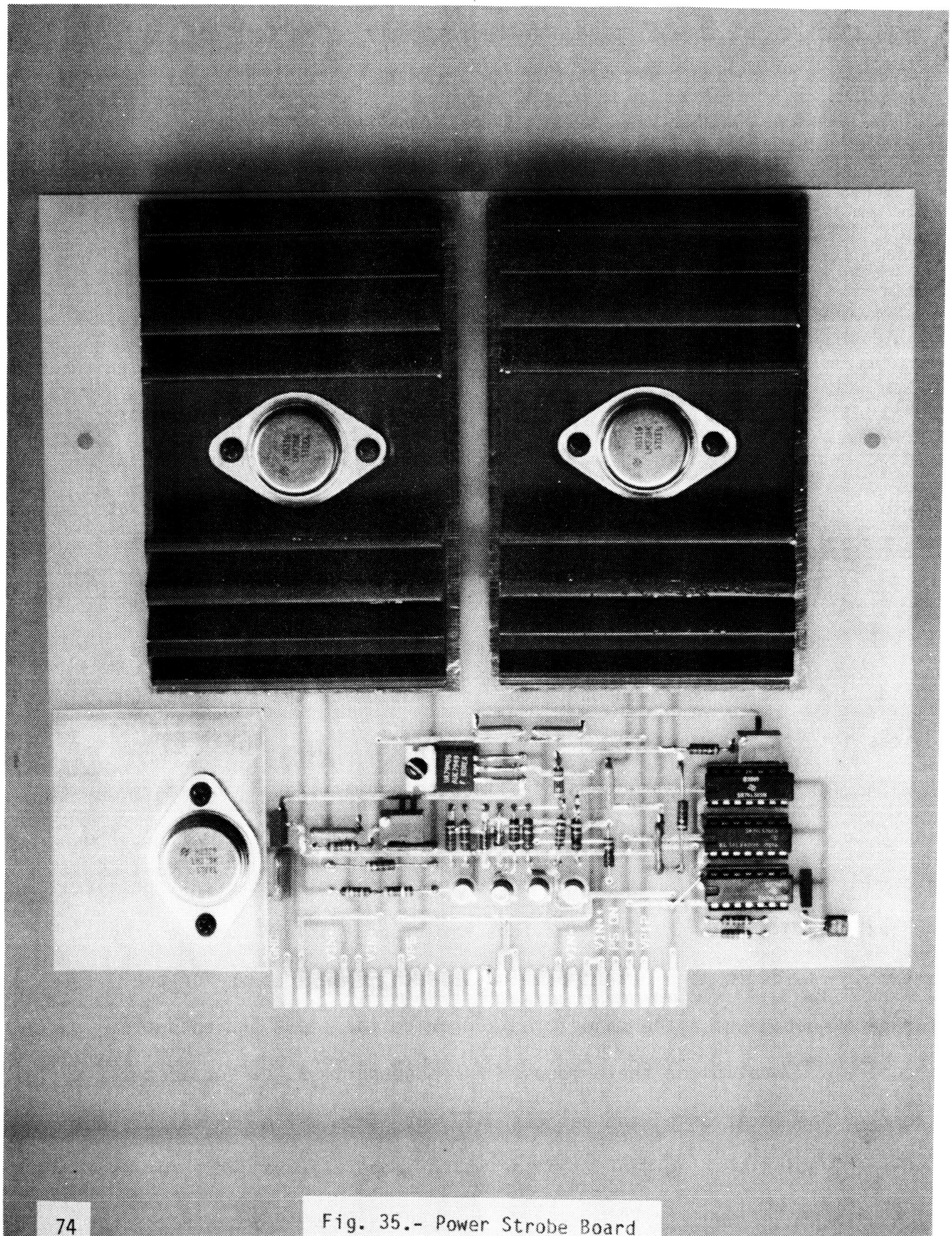
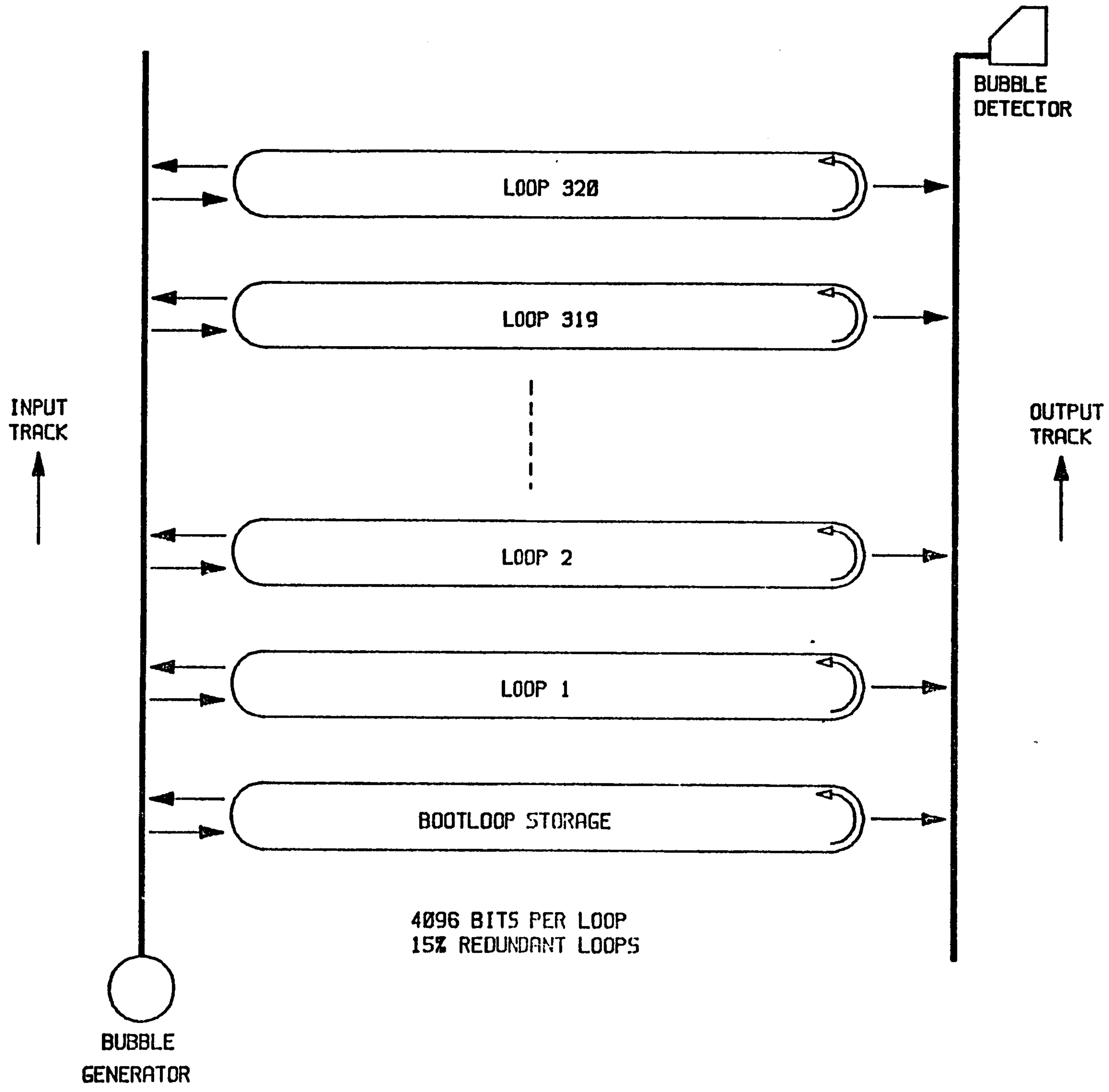


Fig. 35.- Power Strobe Board



FUNCTIONAL ORGANIZATION OF A BUBBLE CHIP
FIGURE 36 [REF 15]

separately accessible bootloop on the device. Initialization of most commercially available bubble devices requires the readout of the entire bootloop to a register in the control circuitry. The time required for this operation is determined by the rotation time of the field coils and the length of the minor loop.

The achievement of higher memory data rates requires the paralleling of the bubble devices. Table 4 shows the number of devices required to produce a desired average useful data rate and the initialization times needed for each configuration using the Intel 1 Mbit bubble device. The initialization times are similar for other manufacturers' devices. The architecture of the BDC dictates that each bubble device be initialized in series, even though the devices can operate in parallel. The initialization time required for a 1 megabit data rate is on the order of 2 seconds, which is a very high time price to pay every time a device needs to be accessed. It is more desirable to get the initialization time down into the millisecond or microsecond range so that a memory segment can be accessed quickly to accommodate complicated mission scenarios. This normal initialization scheme could limit the applicability of bubbles in spacecraft to relatively slow-access recording situations. Since the present initialization scheme is dependent on the internal architecture of the device, the bootloop data must be stored externally in order to provide a faster initialization.

Table 4. Initialization Times

# OF DEVICES	AVERAGE USEFUL DATA RATE	INITIALIZATION TIME	
		AVERAGE	MAXIMUM
1	68 kb/s	80 ms	160 ms
8	544 kb/s	640 ms	1280 ms
16	1088 kb/s	1280 ms	2560 ms

Specific Implementation

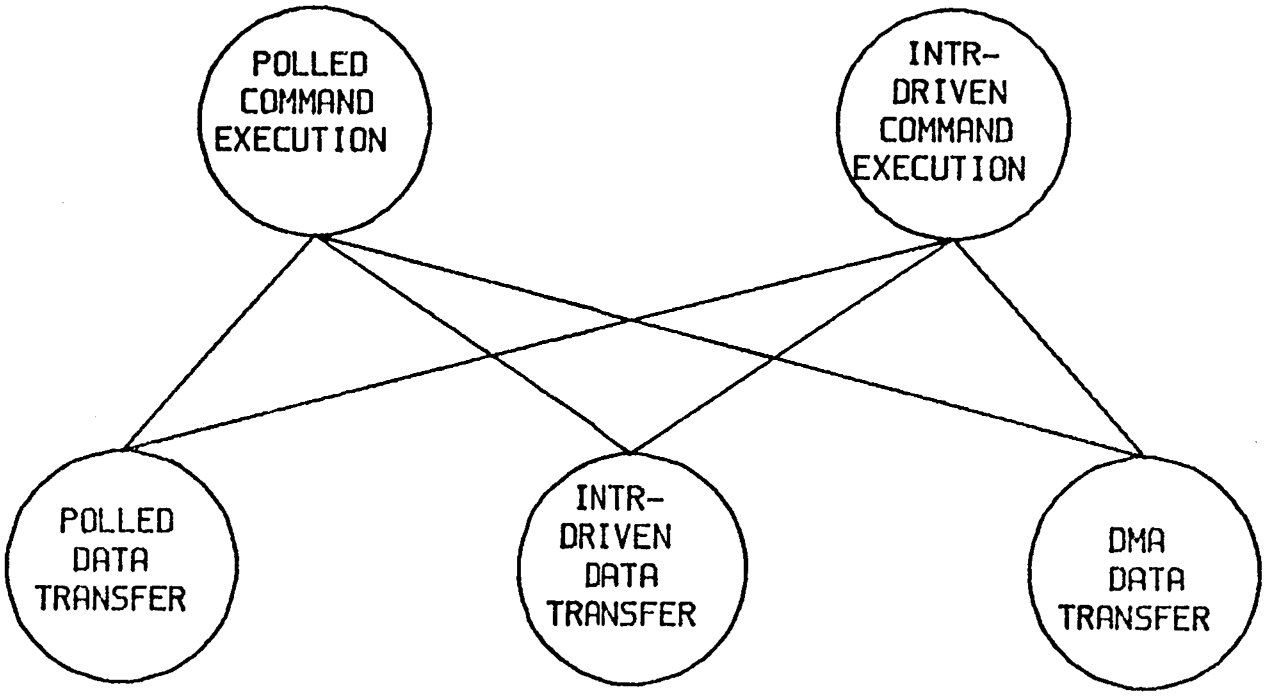
An alternate initialization technique using an EPROM for a small amount of external storage has been demonstrated (ref. 10). The implementation described in this paper utilizes an EEPROM as the external storage medium for the bootloop code. An EEPROM is chosen because it provides a high degree of RAM-like flexibility, while retaining the non-volatility of the ROM. The added feature of an in-system rewrite capability, not available in the EPROM, makes this implementation more easily adaptable to memory board replacements and future system modifications. The actual process of Fast Initialization is similar to the methodology outlined in the application using EPROM. For an explanation of the actual hardware used to implement the external storage of the bootloop information in the EEPROM, see the previous section entitled "EEPROM Controller Board."

SOFTWARE INTERFACE

The software interface required to control the Memory Module and interface with the user consists of command execution, data transfer (DMA), and EEPROM control software. Command execution software is responsible for the implementation of the bubble device controller commands. This process entails the issuance of the command to the bubble device controller, verification of command acceptance, and determination of command success or failure. The data transfer software handles the transference of raw data between the user and the memory storage devices, and internal system data between various system components. Use of the programmable DMA controller provides the flexibility of allowing the DMA to occur under program control. Control software for the EEPROM is necessary to initialize the EEPROM control circuitry and to efficiently read/write data from/to the EEPROM. Fig. 37 shows the various operating modes that are available to drive the memory module system. This paper presents a discussion of a driver design implemenation for polled command execution and DMA data transfer.

Command Execution

Command execution can be separated into two distinct phases: a command phase and a result phase. It is during the command phase that the software driver handles the loading of the parametric registers in the BDC, issues the requested command, and verifies the acceptance of the command by the BDC. During the result phase, the driver determines the success or failure of the command. An assessment of the success or failure of an issued command is ascertained by polling the BDC's status register, to verify the appropriate bit assignments shown in Table 5, when the system is being operated in the polled command execution mode. The status bits associated with command execution are Busy, Op complete, and Op fail. Additional bits in the status register (timing error, parity error) may also be set, in the case of an Op fail, as an indication of the nature of the failure.

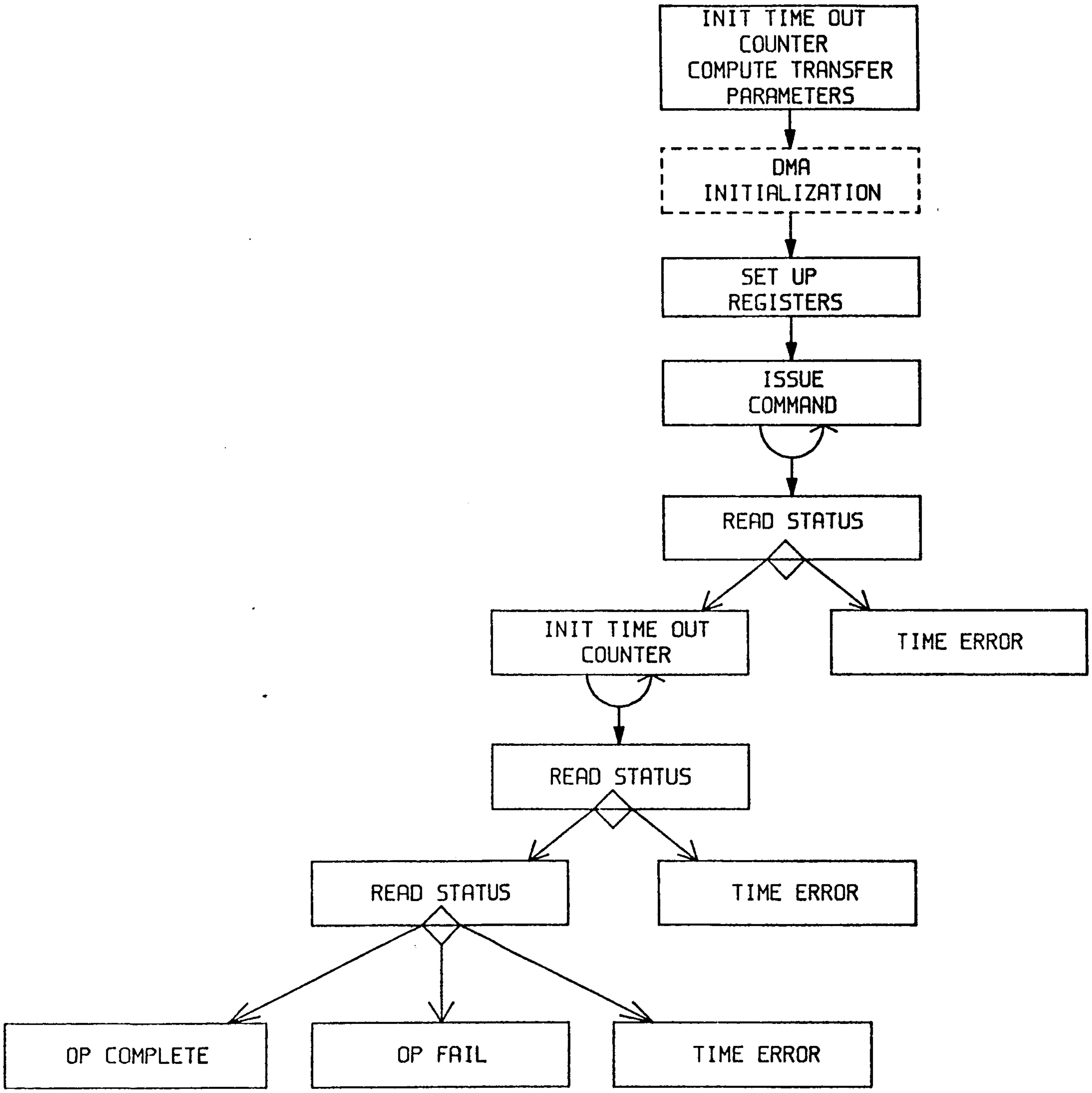


BUBBLE SYSTEM OPERATING MODES
FIGURE 37 [REF. 11]

Table 5. Status Register

BIT	FUNCTION
0	FIFO Ready
1	Parity Error
2	Uncorrectable Error
3	Correctable Error
4	Timing Error
5	Op Fail
6	Op Complete
7	Busy

Polled mode execution of commands consists of issuing the command and continually polling the status register in the BDC to access when the operation has been completed and whether or not it was successful. Before a command can be sent to the BDC, the busy bit must be clear. Acceptance and instigation of a command is indicated by the busy bit being set. The busy bit will continue to be set as long as the command is being executed. Data transfer operations will cause activity on the FIFO Ready bit throughout the execution of the command. Termination of the command is indicated by the clearing of the busy bit. An unsuccessful completion of the command will result in an Op Fail, with a possible indication of the reason for the failure. Successful completion of the command will be indicated by the setting of the Op Complete bit in the status register. Whether the command was completed successfully or not, the busy bit is now clear and the BDC is ready to accept another command. Fig. 38 shows a

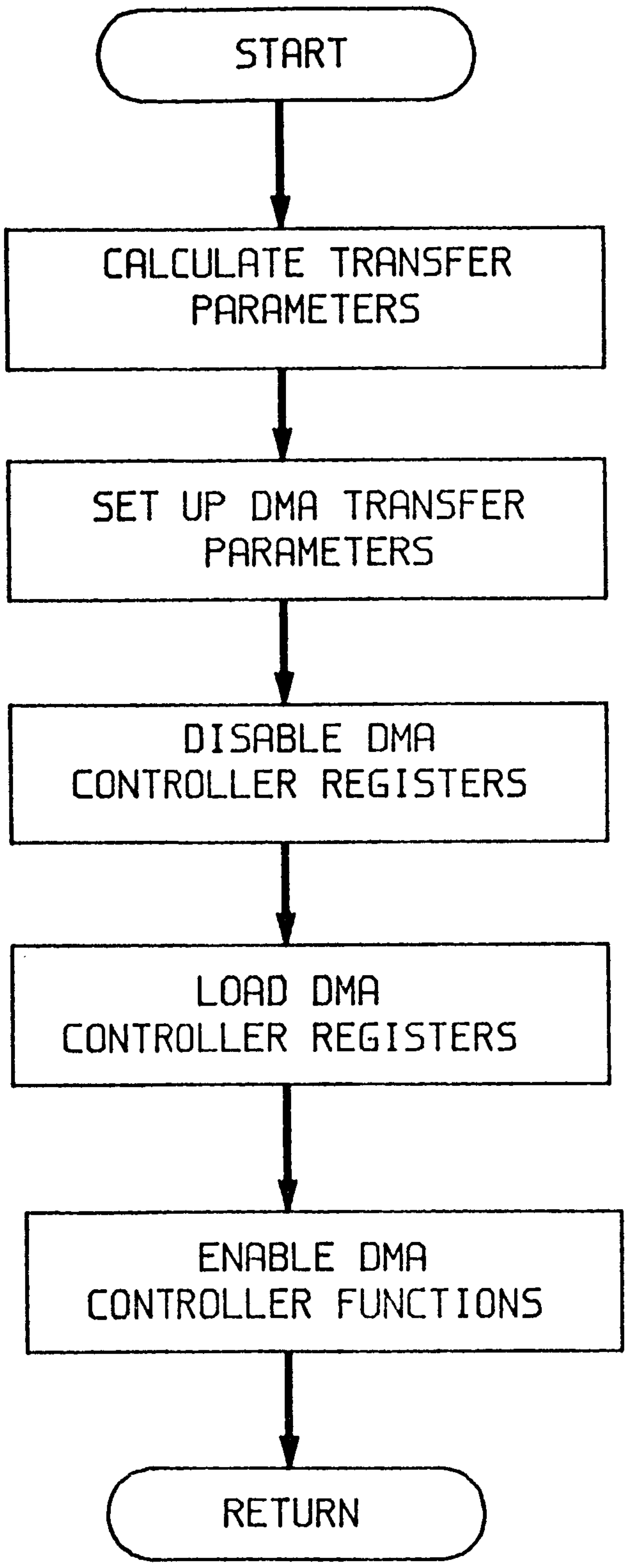


POLLED COMMAND EXECUTION ROUTINE
FIGURE 38

structure diagram for the polled command execution routine. Examples of non-data transfer command operations are abort, normal initialization, purge, and FIFO reset.

Data Transfer

The basic operation of the BDC is the same regardless of which data transfer mode is used (polled, interrupt, DMA). Pages of data are transferred to/from the bubble device continuously until the page count in the Block Length Register (ref. 11) is satisfied. During the read/write process in either polled or interrupt modes, the host processor keeps up with the BDC by continually emptying/filling the BDC FIFO until all of the data have been transmitted. However, operation of the data transfer process in DMA mode frees the host processor from having to participate in the actual transfer of data. This implementation uses the programable DMA controller and a bipolar microcomputer bus controller to control the data transfer process (see fig. 19). Elimination of the involvement of the host processor with the actual data transfer operation greatly decreases the software overhead of the system. Before a DMA operation can occur, the DMA Controller must be initialized to perform the requested operation (read/write bubble memory, read/write FIFO). Fig. 39 shows a flow chart of the DMA hardware initialization routine. Reference 14 provides further information about the programming requirements of the DMA Controller.


START
CALCULATE TRANSFER PARAMETERS
SET UP DMA TRANSFER PARAMETERS
DISABLE DMA CONTROLLER REGISTERS
LOAD DMA CONTROLLER REGISTERS
ENABLE DMA CONTROLLER FUNCTIONS
RETURN

DMA HARDWARE INITIALIZATION ROUTINE
FIGURE 39 [REF. 11]

EEPROM Control Software

The control software for the EEPROM handles the implementation of read/write commands for the transference of data from/to the EEPROM device. An EEPROM read will be implemented to obtain the bootloop data during the fast initialization of the bubble device controller. Reading the EEPROM device is no different than reading from any other area of memory within the single board computer system since the EEPROM device was simply assigned to off-board address space. Hence, the software needed to implement an EEPROM read simply consists of an assembly language command to shift data from one area of memory to another. The control software for the EEPROM also handles the necessary write/erase cycle timing. It will only be necessary to write to the EEPROM device when a new bubble storage unit is added to the memory module system. A write/erase cycle is composed of sending address/data information to the ports and instructing the EEPROM timer to time out for the full 10 ms that is necessary for the stable completion of the write/erase cycle. The write/erase routine also prevents the situation of an illegal information request to the EEPROM during the write/erase cycle through an illegal access interrupt structure. Completion of the write/erase cycle is indicated to the microprocessor via an interrupt after the EEPROM is instated to the read mode.

CONCLUSION

A functional organization of a bubble memory module has been developed. The system architecture provides simultaneous operation of bubble devices to attain high data rates. Banks of bubble devices are accessed by a given bubble controller to minimize controller parts. A power strobing technique is discussed which could minimize the average system power dissipation. A fast initialization method using EEPROM devices promotes fast access. The bubble memory module design and the associated memory system concept are particularly applicable to onboard spacecraft use because of techniques incorporated for power strobing and fast access to arbitrary data. Portions of an experimental embodiment of the memory module were evaluated. The power strobe and fast initialization techniques were demonstrated. Flight memory systems which incorporate the concepts and techniques of this work could now be developed for application.

ACKNOWLEDGMENT

The authors wish to express appreciation to Ron Gorton for his assistance in developing the bubble memory module interface description, and to Tony Brooks for his assistance in fabricating and testing some of the circuit cards and operating the computer test station. Thanks are also expressed to Dick Pierce of Intel for answering questions about bubble controller operation, microprocessor interface problems, power fail circuit operation, and EEPROM interfacing, and also for providing useful software routines.

REFERENCES

1. Besser, P. J., et al.: "Development of a High Capacity Bubble Domain Memory Element and Related Epitaxial Garnet Materials for Application in Spacecraft Data Recorders," NASA CR-144960, June 1976.

2. Becker, F. J., et al.: "An 8x10^5 Bit Bubble Memory Cell for Spacecraft Applications," IEEE Trans. Magn., Vol. MAG-16, 770, 1980.

3. Bohning, O. D., Adler, S. L. and Nichols, C. D.: "A Bubble Detection Circuit for Spacecraft Applications," IEEE Trans. Magn., Vol. MAG-16, 776, 1980.

4. Bonyhard, P. I., et al.: "Magnetic Bubble Technology for Military Applications," Interim Report AFWAL-TR-83-1121, August 1983.

5. Stermer, R. L., Jr., et al.: "A Current Access, Self-Structured, Multilayered Bubble Domain Memory," IEEE Trans. Magn., Vol. MAG-16, 1047, 1980.

6. Torok, E. J., Kamin, M., and Toman, C. H.: "Investigation of Multilayer Magnetic Domain Lattice File," NASA CR-165908, August 1982.

7. Murray, G. W., et al.: "10^8 Bit Solid State Spacecraft Data Recorder," NASA CR-3182, October 1979.

8. Bohning, O. D. and Becker, F. J.: "Bubble Memory Module," NASA CR-3380, December 1980.

9. Hayes, P. J., and Stermer, R. L., Jr.: "Bubble Domain Technology for Spacecraft Onboard Memory," Proc. SPIE Conf. for Advanced Remote Sensing," Vol. 363, August 26-27, 1982.

10. Looney, K. T., Nichols, C. D., and Hayes, P. J.: "Investigation of Fast Initialization of Spacecraft Bubble Memory Systems," NASA TM-85832, June 1984.

11. Intel Corp.: "BPK-72 Bubble Memory Prototype Kit User's Manual," 1983.

12. Intel Corp.: "E^2PROM Family Appl. Hdbk.," Book II, AP-102, November 1981.

13. Intel Corp.: "Memory Comp. Hdbk.," 1983.

14. Intel Corp.: "ISBC 254 Technical Manual," February 1981.

15. Intel Corp.: "Bubble Memory System Design Workshop: Student Study Guide," Version 1.0, November 1981.

16. National Semiconductor Corp., "Linear Application Handbook," 1980.

APPENDIX A

BUBBLE MEMORY TEST STATION

BUBBLE MEMORY TEST STATION

The bubble memory test station is configured as shown in Figure A-1. The tester is a commercially available 32-bit computer system. Peripherals pertinent to operation were the operator's console or CRT, disc drive, papertape reader, line printer, and an 8-line communications multiplexer. The communication multiplexer is used to interface the memory system under test with the computer system. Figure A-2 shows the functional block diagram of the multiplexer. A maximum of 8 memory systems can be interfaced via the multiplexer. The system conforms to the RS 232C interface and can be programmed for a variety of baud rates and character formats. Data transfers between the systems under test and the multiplexer are bit serial at a baud rate selected under program control. The multiplexer contains circuits to generate and detect the control signals required to set up, take down, supervise the data communications channel, and provide proper status and interrupt information to the processor. The multiplexer includes 8 universal asynchronous receiver/transmitter (UART) devices. To transmit data to the memory system under test, the UART accepts data from the baud rate generator and serializes the data to the RS 232C driver and on to the memory system. To receive data, the UART accepts data from the memory system via the RS 232C receiver, deserializes the data, and sends it to the processor.

The test station operational software is a machine-language program provided by the manufacturer called "BMMTEST" (ref. A-1). This software enables the system operator to test memory systems on the multiplexer bus. This program provides a set of routines that allows a variety of functions to be performed such as reading, writing, and error calculation. Each

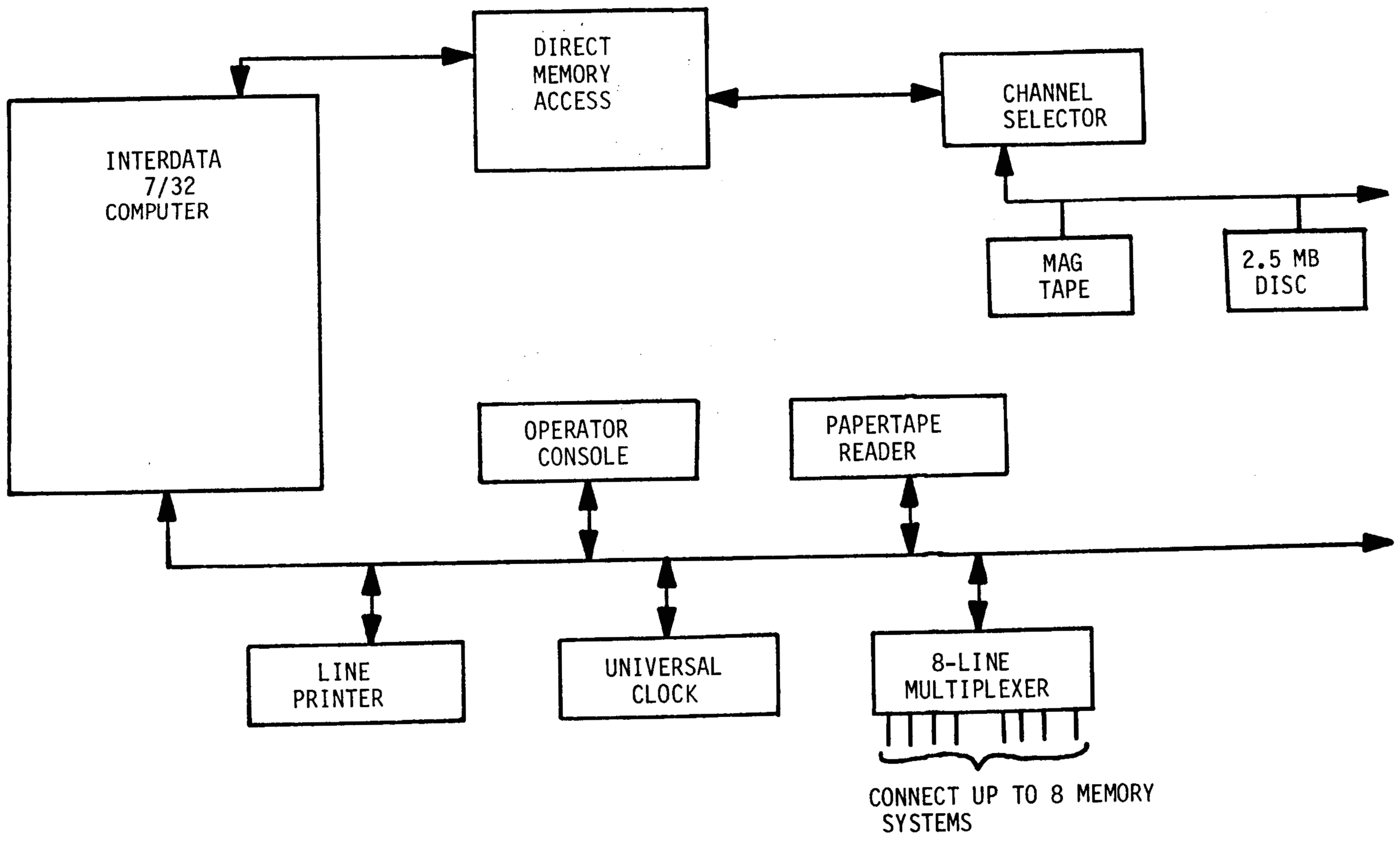
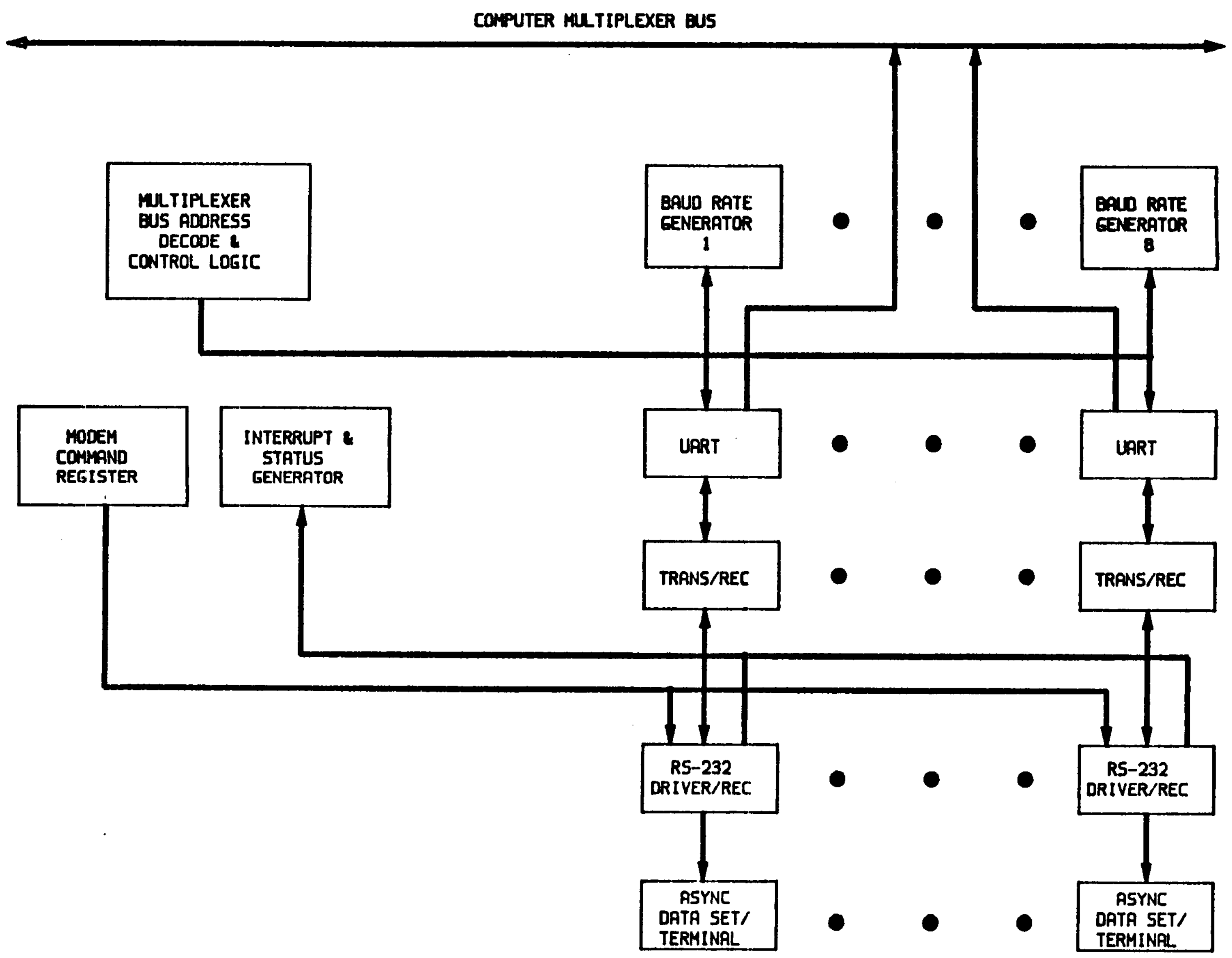


FIGURE A-1. TEST STATION BLOCK DIAGRAM



EIGHT-LINE COMMUNICATION MULTIPLEXER
FIGURE A-2

routine is designated by a mnemonic which, when combined with the proper parameter, provides the operator with the capability of specifying a wide variety of testing procedures. For a complete listing of the mnemonics, routines and required parameter (see ref. A-1).

Testing of the memory systems can begin after the program "BMMTEST" becomes operational in the system (ref. A-2). System operation is depicted in the flow diagram of Figure A-3. After the program becomes operational, no action occurs until routines are called by inputting the proper mnemonic code. The mnemonics can be entered from either the operator's console or from a file that has been stored on mag tape or disc drive. As each mnemonic is read, it is tested for validity; if valid, the mnemonic is decoded. The decode process translates the mnemonic and parameter, if applicable, into machine language and runs the associated routine. This process continues, one mnemonic at a time, until either a stop or an invalid code is read. When stop or invalid code is read, the BMMTEST is automatically cancelled and the processor is returned to its normal operational state (ref. A-2), the BMMTEST program must be reloaded in order for testing to continue.

Figure A-4 is an example of a mnemonic file that was used to test or program a memory system. The file labeled TESTA.CMD, was set up to perform one write, one read, and record errors. Figure A-5a is the flow diagram of TESTA.CMD. This procedure begins by initializing the processor and enabling the appropriate interrupts (ref. A-3). The byte error counter, counter 0(CTR0), is cleared and the read and write buffers, (RBUF, WBUF), are set up to an arbitrary size of 256 bytes. These buffers can range from 8 bytes to 2K bytes. The data pattern is then set up with an eight byte hexadecimal pattern that will be written to and read from the system under

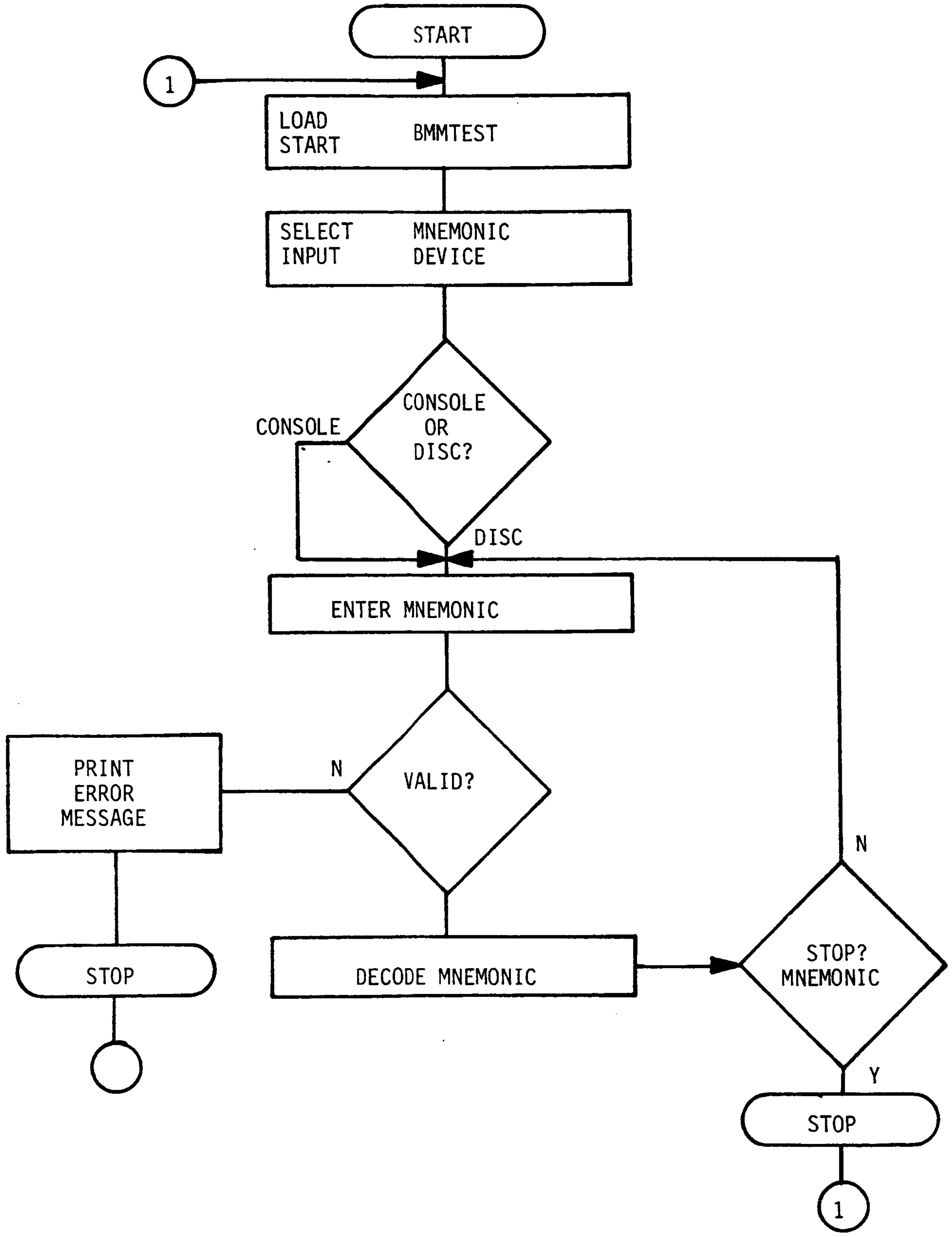


FIGURE A-3. SYSTEM OPERATION FLOW DIAGRAM

```
I                                                       ;INITIALIZE
UDL                                                     ;USER DEDICATED LOCATIONS
ICTR                                                    ;INITILIZE COUNTERS
SVC1                                                    ;SETUP INTERRUPT
VCTR0,0                                                 ;COUNTER 0 = 0
RBUF 256                                                ;READ BUFFER = 1 TO 2K BYTES
WBUF 256
DSMG TESTA NOW IN PROGRESS
PATTERN FF,FF,FF,FF,FF,FF,FF,FF                         ;SET DATA PATTERN
DMSG TESTA PATTERN FF,FF,FF,FF,FF,FF,FF,FF
IERT                                                    ;SET ERROR RATE TO 0
MATRIX                                                  ;DISPLAY DATA MATRIX
I
1 DMSG REPEAT OF TEST1                                  ;REPEAT TEST1 SEQUENCE
2 VCTR2,1                                               ;SET READ LOOP COUNTER = 1
3 VCTR4,1                                               ;SET WRITE LOOP COUNTER = 1
4 VCTR8,1                                               ;SET SEQ LOOP CUONTER = 1
5 VCTR1,1                                               ;READ/WRITE FLAG 0=READ/1=WRITE
6 TCTR1<1,12                                            ;IF COUNTER 1<1,GO TO STEP 12
7 SCTR4,1                                               ;CTR4 = CTR4 - 1
8 WRITE;                                                FILL WRITE BUFFER (WBUF) WITH DATA PATTERN
9 TCTR4>0,6
 10 SCTR1,1                                             ;END OF WRITE LOOP
 11 GO 6                                                ;GO STEP 6
 12 READ                                                ;FILL READ BUFFER (RBUF) WITH DATA
 13 CMP WBUF,RBUF                                       ;COMPARE BUFFERS FOR ERRORS
 14 MATRIX
 15 SCTR2,1
 16 TCTR2>0,11                                          ;IF COUNTER 2 > 0 THEN GO TO STEP 11
 17 SCTR8,1
 18 TCTR8>0,1
 19 DMSG 'END OF TEST1'
 20 DMSG'TOTAL BYTES IN ERROR THIS TEST
 21 DCTR0                                               ;DISPLAY BYTE ERROR COUNT(COUNTER 0)
 22 DMSG'TOTAL CHIP ERRORS'
 23 DCTR16,17,18,19,20,21,22,23
 24 DMSG'TOTAL POSITION ERROR'
 25 DCTR24,25,26,27,28,29,30,31
 26 FINI
 27 TCTR3<1,30                                          ;RUN TEST FLAG
 28 DMSG 'START OF TEST1'                               ;START TEST1 LOOP
 29 GO 2                                                ;GO TO STEP 2
 30 DMSG 'TEST1 NOT RUN'
FINI
VCTR3,1
ERAT1                                                   ;CALCULATE ERROR RATE
```

FIGURE A-4. MNEMONIC FILE TESTA.CMD

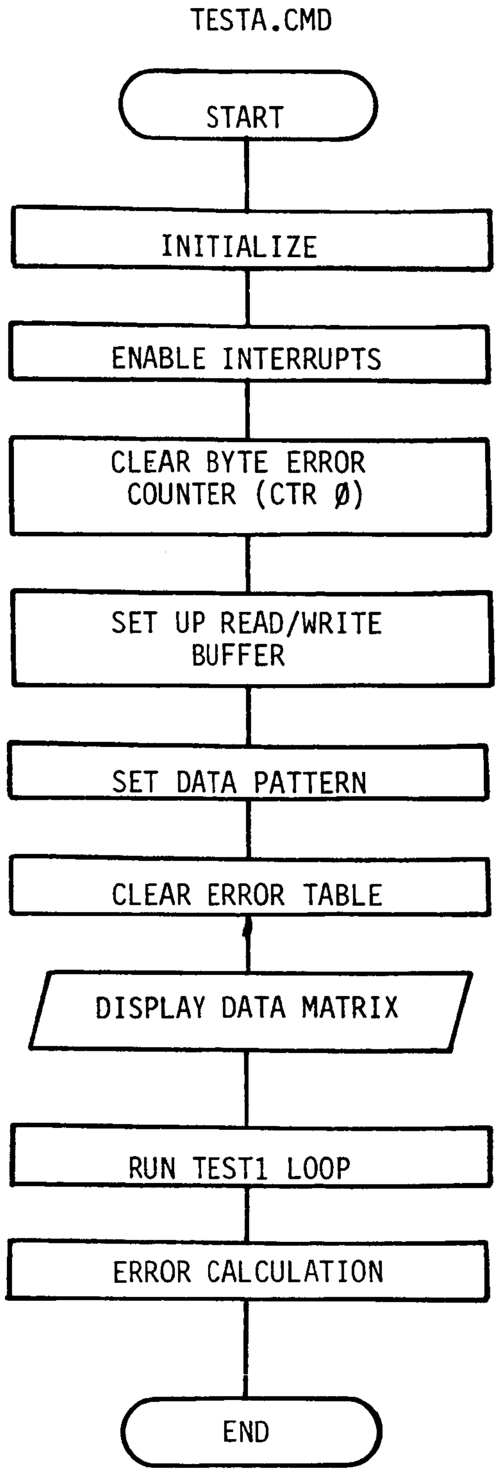


FIGURE A-5a. TESTA.CMD MAIN PROGRAM

test. The error table is cleared (IERT). The MATRIX mnemonic causes a data matrix to be listed to the line printer as shown in Figure A-6 part A. Next, the a subprogram, TEST1, Figure A-4, steps 14-43, is run, producing the read/write and error calculations. TEST1 is one of a number of subprograms which the operator can specify. Finally, the error calculations are made and listed to the printer as shown in Figure A-6, part B.

Figure A-5b is a flow diagram of subprogram TEST1. The subprogram begins by setting up four counters. Counter 2 (CTR2) is the read sequence loop counter. Counter 4 (CTR4) is for the write sequence. Counter 8 (CTR8) is for the TEST1 sequence loop.

Counter 1 (CTR1) is used as a read/write sequence flag where 1 = a write sequence and 0 = a read sequence. For this program, each counter is initially set to 1 (i.e. VCTR 2,1). The next step (TCTR1 < 1, 12) directs the program to a write operation. A 1 is then subtracted from the write sequence loop counter with the mnemonic SCTR4. WRITE initiates the write sequence, the write buffer is filled with the data pattern and then sent to the memory system under test. Counter 4 is tested (TCT R4 > D, 6), if greater than 0 the write sequence is repeated, if not, CTR1 is set to 0. The READ mnemonic initializes the read sequence. The RBUF is filled with data receiving the data which had previously been written on the system under test. CMP WBUF, RBUF compares the read and write buffers for errors. Errors, if any, are recorded and a listing is output to the printer as shown in Figure A-6, part B.

In this test procedure, the errors are recorded as shown in Figure A-6. Part A shows the error table after system initiation and prior to the write/read operation. Part B is a display of the matrix after the read/

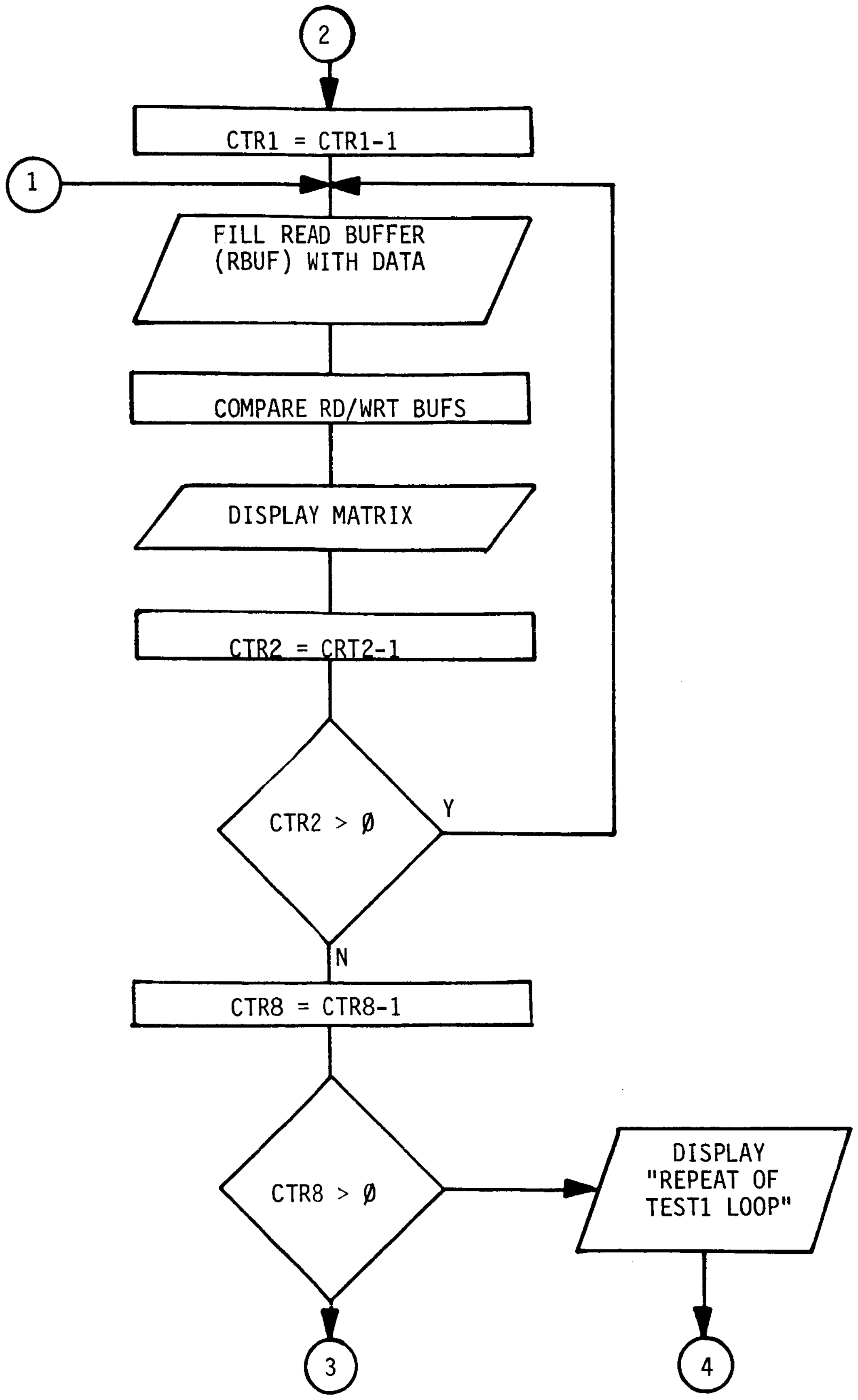


FIGURE A-5b. TEST1 (CONT)

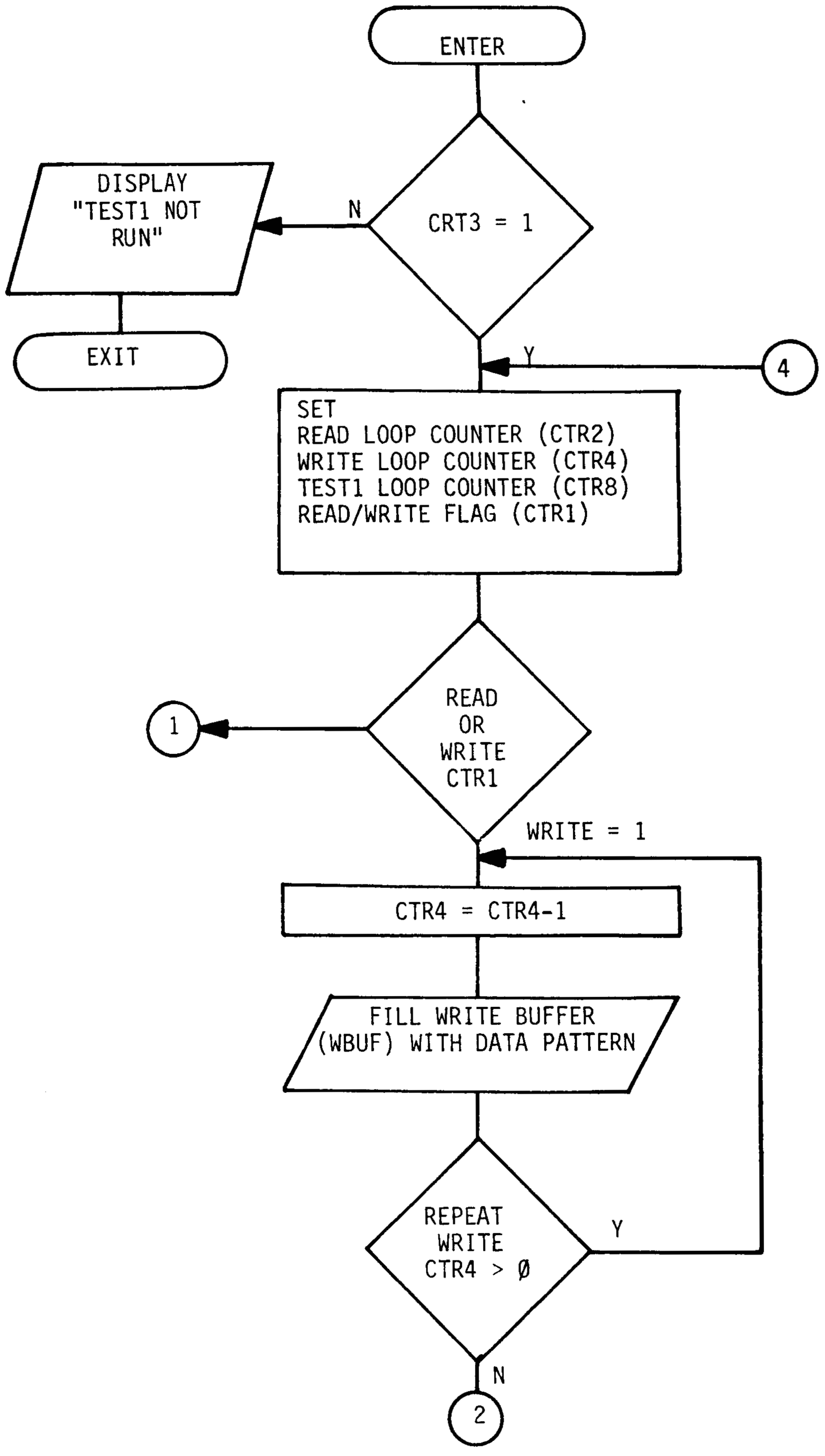


FIGURE A-5b. TEST1

C-2

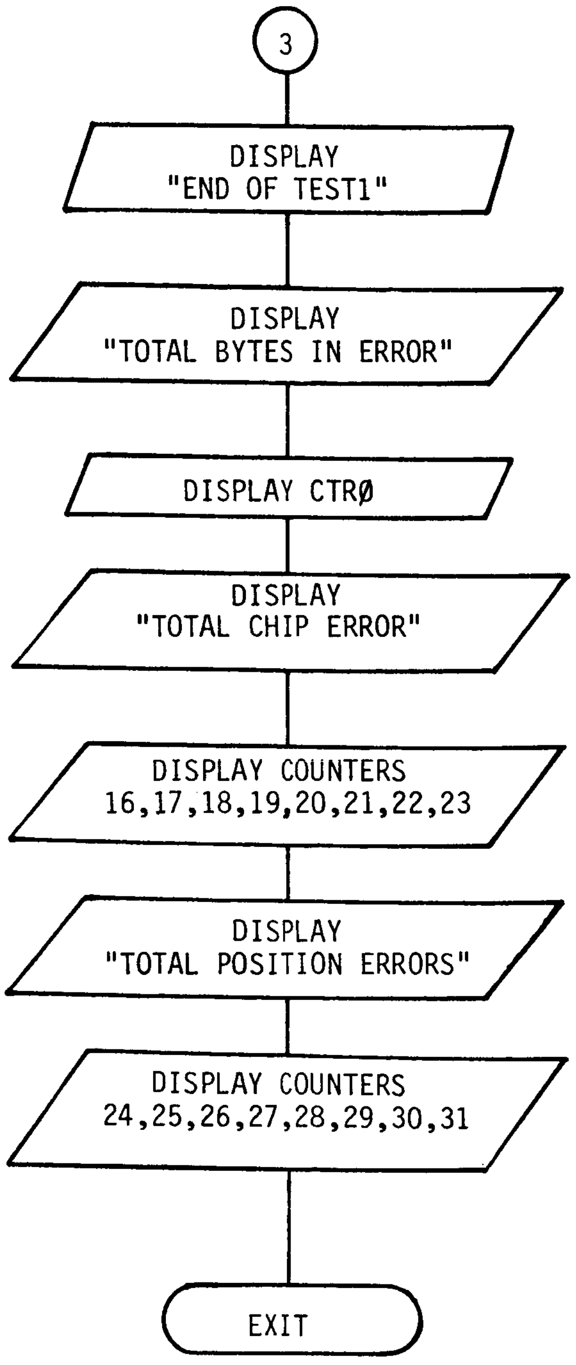

3
DISPLAY "END OF TEST1"
DISPLAY "TOTAL BYTES IN ERROR"
DISPLAY CTRØ
DISPLAY "TOTAL CHIP ERROR"
DISPLAY COUNTERS 16,17,18,19,20,21,22,23
DISPLAY "TOTAL POSITION ERRORS"
DISPLAY COUNTERS 24,25,26,27,28,29,30,31
EXIT

FIGURE A-5b. TEST1 (CONT)

```
08/29/84   10:37:36
  TESTA                  NOW IN PROGRESS

 TESTA    PATTERN 00,00,00,00,00,00,00,FF
 TESTA          CHANNEL      A
                             POS1       POS2       POS3       POS4       POS5       POS6       POS7       POS8        ACCUM  TOTAL
  0 0 0 0 0 0 0 1                   0          0          0          0          0          0          0          0             0
  0 0 0 0 0 0 0 1                   0          0          0          0          0          0          0          0             0
  0 0 0 0 0 0 0 1                   0          0          0          0          0          0          0          0             0
  0 0 0 0 0 0 0 1                   0          0          0          0          0          0          0          0             0   A
  0 0 0 0 0 0 0 1                   0          0          0          0          0          0          0          0             0
  0 0 0 0 0 0 0 1                   0          0          0          0          0          0          0          0             0
  0 0 0 0 0 0 0 1                   0          0          0          0          0          0          0          0             0
  0 0 0 0 0 0 0 1                   0          0          0          0          0          0          0          0             0

 00000000000000FF                   0          0          0          0          0          0          0          0             0

   TOTAL   ERRORS           CELL 0     CELL 1     CELL 2     CELL 3     CELL 4     CELL 5     CELL 6     CELL 7               ALL
 00000000000000FF                   0          0          0          0          0          0          0          0             0
                   START OF TEST #1    STD   WRITE CELLS 1,,,,


                             POS1       POS2       POS3       POS4       POS5       POS6       POS7       POS8        ACCUM  TOTAL
  0 0 0 0 0 0 0 1                   0          0          0          0          0          0          0          1             1
  0 0 0 0 0 0 0 1                   0          0          0          0          0          0          0          1             1
  0 0 0 0 0 0 0 1                   0          0          0          0          0          0          0          1             1
  0 0 0 0 0 0 0 1                   0          0          0          0          0          0          0          1             1   B
  0 0 0 0 0 0 0 1                   0          0          0          0          0          0          0          1             1
  0 0 0 0 0 0 0 1                   0          0          0          0          0          0          0          1             1
  0 0 0 0 0 0 0 1                   0          0          0          0          0          0          0          1             1
  0 0 0 0 0 0 0 1                   0          0          0          0          0          0          0          1             1

 00000000000000FF                   0          0          0          0          0          0          0          8             1

   TOTAL   ERRORS           CELL 0     CELL 1     CELL 2     CELL 3     CELL 4     CELL 5     CELL 6     CELL 7               ALL
 00000000000000FF                   0          8          0          0          0          0          0          0             8
END OF       TEST1
TOTAL BYTES IN ERROR THIS TEST (TOTAL STD )
COUNTER     0 =          1
TOTAL CHIP ERRORS TEST STD
COUNTER    16 =          1
COUNTER    17 =          1
COUNTER    18 =          1
COUNTER    19 =          1                                                                                                      C
COUNTER    20 =          1
COUNTER    21 =          1
COUNTER    22 =          1
COUNTER    23 =          1
             TOTAL POSITION ERRORS TEST STD
COUNTER    24 =          0
COUNTER    25 =          0
COUNTER    26 =          0
COUNTER    27 =          0                                                                                                      D
COUNTER    28 =          0
COUNTER    29 =          0
COUNTER    30 =          0
COUNTER    31 =          8
                END OF STD TEST1
       STOP
        BIT     TOTALS    WRITTEN      READ    ERRORS    ERRORS /WRITE      ERRORS /READ     ERRORS/(READ+WRITE)
   TYPE                      3200      3200         8      2300  E-  6       2300  E-  6          1230  E-  6
   TYPE   1                  3200      3200         8      2300  E-  6       2300  E-  6          1230  E-  6
```

FIGURE A-6. DATA MATRIX ERROR RECORDING FORMAT

write buffers have been compared and displays the number of errors detected according to bit position. Parts C and D are displays of counters that keep a running total of errors encountered by chip and bit position respectively during TESTA.CMD. Three error calculations are displayed:
1) READ ERROR RATE = number of READ BIT ERRORS/number of READ BITS,
2) WRITE ERROR RATE = number of WRITE BIT ERRORS/number of WRITE BITS, and
3) R/W ERROR RATE = number of R/W BIT ERRORS/total number of R/W BITS.

REFERENCES

A-1 Bubble Memory Module Test Program (BMMTEST) Perkin-Elmer Program #03-092.

A-2 OS/32 Operator Reference Manual P-E #29-640.

A-3 OS/32 Programmer Reference Manual P-E #29-613.

The reader may want to refer to the following PERKIN-ELMER publications for more detailed information:

Model 7/32 Reference Manual P-E #29-399

Model 7/32 Processor User's Manual P-E #29-405

OS/32 System Planning and Configuration Guide P-E #29-614

Assembly Language Programming Manual P-E #29-640

8-LINE Communications Multiplexers P-E #29-650

APPENDIX B

LIST OF ACRONYMS

LIST OF ACRONYMS

AIO	Application input/output interface
BCIB	Bubble controller interface board
BDC	Bubble device controller
BMM	Bubble memory module
BMMTEST	Bubble memory module test program
BRDSEL	Board select signal line
DDT	Direct data transfer
DMA	Direct memory access
DRQ	Data request line
DTACK	Data transfer acknowledge
ECC	Error correction coding
EEPROM	Electrically erasable programmable read only memory
IBB	Isolation buffer board
INT	Memory module interrupt line
MB	Memory board
PSM	Power supply module
PSTROBE	Power strobe line
SC	System controller
SCM	System controller module
SPSB	Strobable power supply board
SU	Storage unit
RAM	Random access memory
RD	Read control line
ROM	Read only memory
RST	Reset

TTL	Transistor-transistor logic
UART	Universal asynchronous receiver/transmitter
WR	Write control line

#65655 (MG)

1. Report No. NASA TM-86411	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle Bubble Memory Module for Spacecraft Application	5. Report Date April 1985
	6. Performing Organization Code 506-58-13-03
7. Author(s) P. J. Hayes, K. T. Looney, and C. D. Nichols	8. Performing Organization Report No.
	10. Work Unit No.
9. Performing Organization Name and Address NASA Langley Research Center Hampton, VA 23665	11. Contract or Grant No.
	13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

Bubble domain technology offers an all-solid-state alternative for data storage in onboard data systems. A versatile modular bubble memory concept ~~has~~ was ~~been~~ developed. The key module is the bubble memory module which contains all of the storage devices and circuitry for accessing these devices. This report documents the bubble memory module design and preliminary hardware designs aimed at memory module functional demonstration with available commercial bubble devices. The system architecture provides simultaneous operation of bubble devices to attain high data rates. Banks of bubble devices are accessed by a given bubble controller to minimize controller parts. A power strobing technique is discussed which could minimize the average system power dissipation. A fast initialization method using EEPROM devices promotes fast access. Noise and crosstalk problems and implementations to minimize these are discussed. Flight memory systems which incorporate the concepts and techniques of this work could now be developed for application.

17. Key Words (Suggested by Author(s)) Bubble Memory, Spacecraft Memory System, Initialization, EEPROM ?	18. Distribution Statement [redacted] Subject Category 33

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 106	22. Price

Available: NASA's Industrial Applications Centers