BPK 72
BUBBLE MEMORY
PROTOTYPE KIT
USER'S MANUAL

intel® magnetics

BPK 72
BUBBLE MEMORY
PROTOTYPE KIT
USER'S MANUAL

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

Additional copies of this manual or other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

The information in this document is subject to change without notice.

Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this document. Intel Corporation makes no commitment to update nor to keep current the information contained in this document.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

The following are trademarks of Intel Corporation and its affiliates and may be used only to identify Intel products:

BXP	Intel	Library Manager	Plug-A-Bubble
CREDIT	int$_e$l	MCS	PROMPT
i	Intelevision	Megachassis	Promware
ICE	Intellec	Micromainframe	RMX/80
iCS	iRMX	Micromap	System 2000
im	iSBC	Multibus	UPI
Insite	iSBX	Multimodule	μScope

and the combination of ICE, iCS, iRMX, iSBC, iSBX, MCS, or RMX and a numerical suffix.

PREFACE

The BPK 72 User's Manual is written for the hardware and software designers of magnetic bubble memory systems using Intel's 1-megabit bubble memory devices and LSI support circuitry.

This manual is intended to be a practical guide, with emphasis primarily on "how to" information. Chapter 1 presents an overview in which the potential user can see the various possible system configurations and operating modes, and also the performance that can be expected from such systems. Chapter 2 provides information for the hardware designer of evaluation systems and prototype systems, and also provides guidelines helpful in designing boards for production. Chapter 3 describes how to operate the bubble memory system, and contains information primarily of use to the software designer.

Because the most complex of the bubble memory's operating requirements are automatically handled by LSI components, it is not strictly necessary for the system designer to know much about bubble memory chips or about the internal operation of the support circuits. However, Chapter 4 has been included for those readers who wish to know more. It also provides background information which may be helpful in establishing the proper context for the "how to" information in Chapters 2 and 3.

Chapter 5 provides service information for situations that may occasionally arise in the field.

CONTENTS

TABLES

ILLUSTRATIONS

CHAPTER 1
GENERAL INFORMATION

1.1 Introduction

The BPK 72 Bubble Memory Prototype Kit contains all parts and instructions needed to build the 1-megabit bubble memory system shown in figure 1-1. This small system (covering less than 16 square inches of board area) can be used as an evaluation system, by means of which the user can gain familiarity with the operation and performance characteristics of Intel bubble memories.

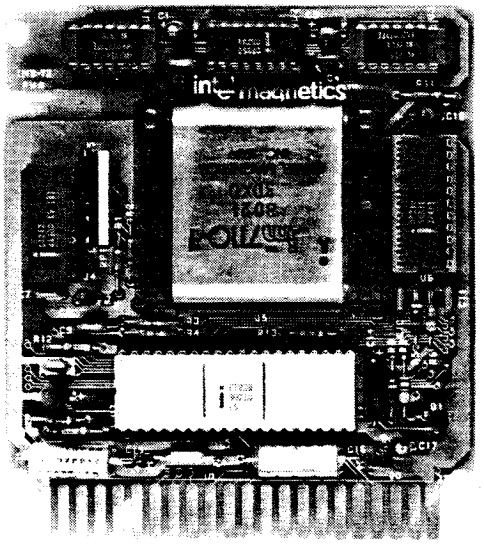


Figure 1-1. BPK 72 Bubble Memory Prototype Kit (Assembled) 121685-1

This manual serves as a user's guide for the BPK 72, and also for the BPK 70 1-Megabit Bubble Storage Units (BSUs). These are particularly useful in building larger bubble memory systems.

All of the bubble memory systems described in this manual are controlled by one or more Intel 7220 Bubble Memory Controllers. The user interface of the 7220 is directly compatible with the Intel microprocessor bus system. Thus the bubble memory system can be treated as a slave to an 8080, 8085, 8086, or 8088 host system.

1.2 Bubble System Configurations

The bubble memory systems specifically discussed in this manual range from 128K bytes to 1 megabyte in data storage capacity. Larger systems may also be built, following the simple guidelines presented.

The 128K-byte system is the one pictured in figure 1-1. Figure 1-2 is the corresponding block diagram. A single magnetic bubble memory (MBM) module is used. The immediate support circuitry for this 7110 MBM consists of five integrated circuit components—one 7250 Coil Predriver, two 7254 Quad VMOS Drive Transistor packs, one 7230 Current Pulse Generator, and one 7242 Formatter/Sense Amplifier. The 7220 controller completes this basic system.

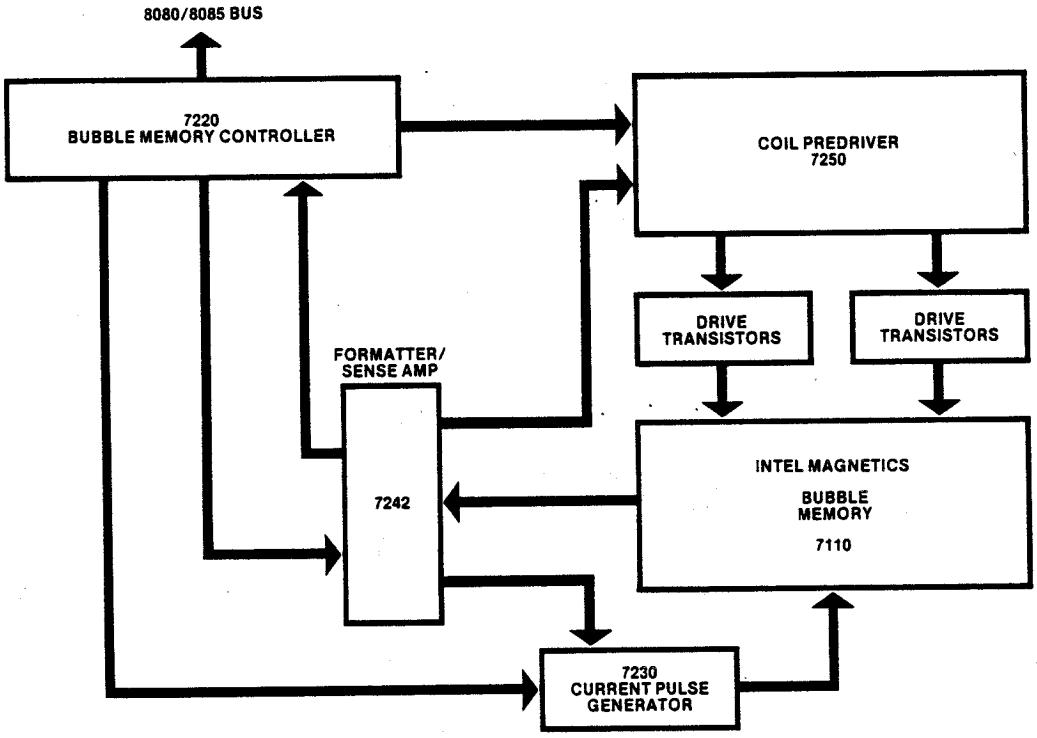


Figure 1-2. Block Diagram of the 128K-Byte Magnetic Bubble Memory System

The 7250 and the two 7254s supply the drive currents for the X and Y coils that move the magnetic bubbles within the MBM. The 7230 supplies the current pulses that generate the magnetic bubbles (GEN) and that transfer the bubbles into the storage loops of the MBM (SWAP) to complete a write operation or replicate them out of the storage loops (REP) for a read operation.

The 7242 reads the bubble signals (data) from the bubble detectors in the MBM. In addition, the 7242 performs data formatting tasks, including the transparent handling of boot loop information, and (if the user elects) automatic error detection and correction.

The 7220 provides a user interface, performs serial-to-parallel and parallel-to-serial data conversions, and generates all timing signals necessary for the proper operation of the MBM's immediate support circuitry.

The details of the above mentioned operations are given in Chapter 4.

Figure 1-3 shows how larger systems can be built up from basic components. A Bubble Storage Unit consists of one 128K-byte MBM and its five immediate IC support chips. The IC components needed for one MBM cell are available as the BPK 70 kit.

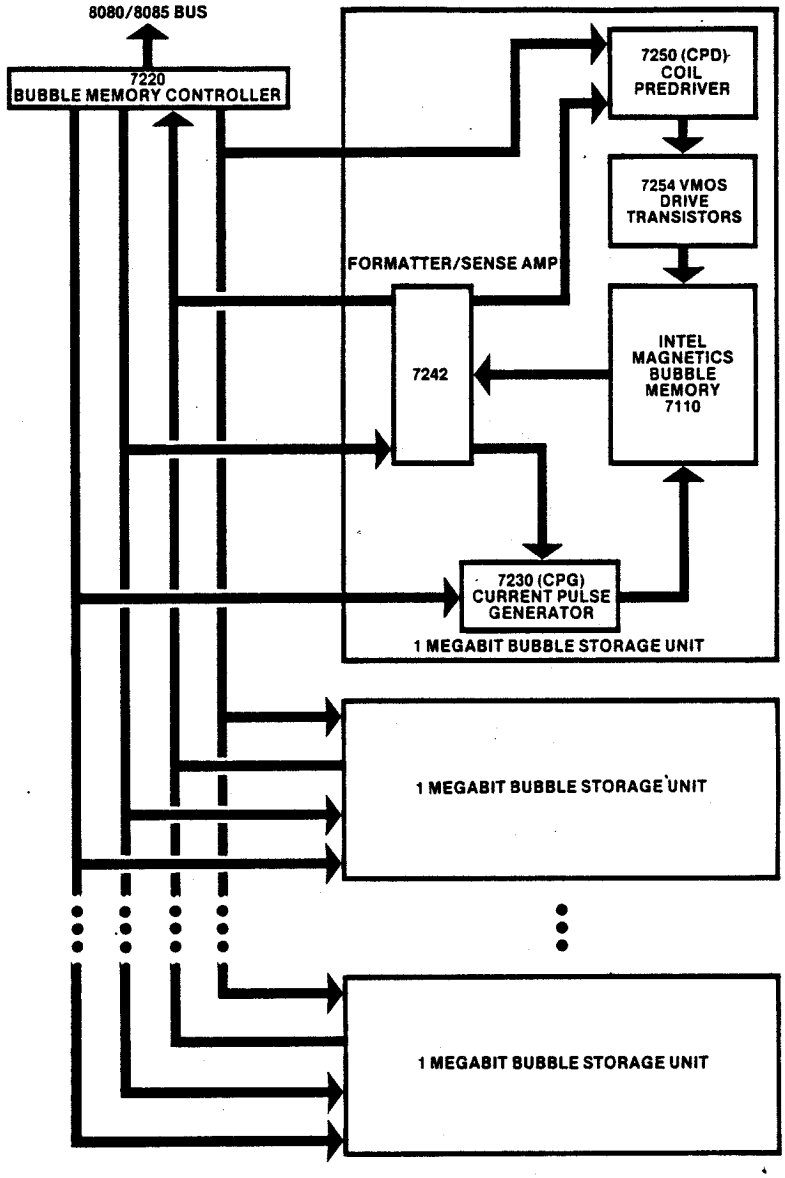


Figure 1-3. Bubble Memory System Expansion up to One Megabyte

The larger systems may be constructed from the parts supplied with one BPK 72 kit and one or more BPK 70 kits. For example, a one-megabyte system may be assembled from one BPK 72 kit and 7 BPK 70 kits.

The BPK 72 kit contains documentation and service aids. Once a user has purchased one BPK 72 kit, future systems may be built using a 7220 BMC and a BPK 70 kit in place of the BPK 72 kit. For example, a one-megabyte system may be assembled using one 7220 and eight BPK 70 kits.

1.3 Bubble System Software

Intel does not supply software as part of the BPK 72 Bubble Memory Kit. Software is available, however, from the INSITE User's Library. Moreover, Chapter 3 of this manual contains many software examples which can aid the software designer.

1.4 Equipment Supplied

The BPK 72 Bubble Memory Kit consists of the following major components:

1 7110 Magnetic Bubble Memory (MBM)
1 7220 Bubble Memory Controller (BMC)
1 7250 Coil Predriver (CPD)
2 7254 Quad VMOS Drive Transistor packs
1 7230 Current Pulse Generator (CPG)
1 7242 Formatter/Sense Amplifier (FSA)
1 Socket for the 7110
1 Dummy Module
1 Seed Module
1 Printed Wiring Board (IMB-72)
1 BPK 72 User's Manual
1 Heat Staking Tip for 7110 Socket Installation

The BPK 70 Bubble Memory Kit consists of the following major components:

1 7110 Magnetic Bubble Memory (MBM)
1 7250 Coil Predriver (CPD)
2 7254 Quad VMOS Drive Transistor packs
1 7230 Current Pulse Generator (CPG)
1 7242 Formatter/Sense Amplifier (FSA)
1 Socket for the 7110

The 7220 Bubble Memory Controller is available as a separate component for use with the BPK 70 kits.

1.5 Equipment Required

Power supplies: +5V and +12V (Specific requirements given in Section 2.5)
Clock: 4 MHz, TTL levels, 50% (±5%) duty cycle

1.6 Documentation Supplied

The documentation supplied with the BPK 72 Bubble Memory Kit consists of this manual, plus application notes available at the time of shipment.

1.7 Operating Characteristics

The characteristics of the user interface for the 7220 controller are given in detail in Section 2.5.

In addition to these specifications, there are some system characteristics which are configuration dependent—specifically data rates, power consumption, and required board area.

Under software control, bubble memory systems using more than one MBM may operate these MBMs in either parallel or multiplexed modes. The parallel mode of operation allows a higher data transfer rate, but also requires more power dissipation (both standby and maximum). How these modes are selected, and the full range of options available to the user are discussed in Section 3.2. The corresponding user data rate requirements are discussed in Section 2.5.

Table 1-1 lists the performance specifications for a few typical systems.

Table 1-1. Bubble Memory System Performance

Parameter	One MBM Unit	Four MBMs Operated in Parallel	Eight MBMs Operated in Parallel	Eight MBMs Multiplexed One at a Time
Capacity	128 kilobytes	512 kilobytes	1 megabyte	1 megabyte
Average Data Rate (kilobits/sec)	68	272	544	68
Maximum Data Rate (kilobits/sec) MFBTR=1*	100	400	800	100
Maximum Data Rate (kilobits/sec) MFBTR=0*	400	1600	3200	400

*See Enable Register in Section 3.2.2 for a discussion of the MFBTR bit.

CHAPTER 2
PREPARATION FOR USE

2.1 Introduction

This chapter describes how to build small bubble-memory systems using Intel's 1-megabit bubble memory devices and LSI support circuitry. Information is also given on the user interface requirements which need to be met in order to properly operate such systems.

2.2 Unpacking and Inspection

Inspect the shipping box visually for damage. If any is evident, contact the carrier to make a claim. Remove the kit from the package (as a precaution, use a ground strap before touching any of the components). Check the components against table 2-1.

2.3 Parts Identification

All the IC components are plainly marked (for example, the CPG is stamped "I7230"). All resistors use the standard color code. Capacitors use their manufacturer's code, but are nevertheless easily identified.

2.4 Fabrication of Hardware Prototypes

2.4.1 128K-Byte System

This single-MBM system can be assembled from the BPK 72 Bubble Memory Kit. The kit includes the IMB-72 printed circuit board on which the components supplied should be mounted. Assemble the board, referring to the assembly drawing (figure 2-1), the schematic (figure 2-2), and the parts list (table 2-1) for the IMB-72 board. Attach a suitable user interface connection, as detailed in Section 2.5.

2.4.2 One-Controller Systems up to One Megabyte

The 7220 Bubble Memory Controller can process data and supply timing and control signals for up to eight bubble storage units each capable of storing 128K bytes of user data. Typical bubble memory systems contain 1, 2, 4, or 8 bubble storage units. Each bubble storage unit contains the LSI components supplied in the BPK 70 Bubble Memory Kit (see figure 1-3). To build a bubble system larger than 128K bytes, the user should first determine how many bubble storage units are required. If *n* units are required, the user should order 1 BPK 72 kit and *n*-1 BPK 70 kits, or 1 7220 and *n* BPK 70 kits.

Table 2-1. BPK-72 Parts List

Reference	Description	Qtv
	PC Board, IMB-72	1
C1,3,8,13,17,19	Capacitor, Electrolytic 15 μF, 20V, ±20%	6
C2,4,7,9-12	Capacitor, Ceramic 0.1 μF, 50V, +80%, −20%	7
C5,6	Capacitor, Mica 120 pF, 100V, ±5%	2
C14	Capacitor, Electrolytic 2.2 μF, 20V, ±10%	1
C15,16	Capacitor, Electrolytic 47 μF, 20V, ±10%	2
C18	Capacitor, Ceramic 1 μF, 20V, ±10%	1
Q1(1)	Transistor VN0106N3 or VN10KM	1
R1,2	Not Used	
R3,4	Resistor, Carbon 5.1Ω, ¼W, 5%	2
R5,6	Bus Wire Jumper	2
R7	Resistor, Carbon 10Ω, ¼W, 5%	1
R8	Resistor, Carbon 12k, ¼W, 5%	1
R9(2)	Resistor, Metal Film 3.48k, ¼W, 1%	1
R10	Resistor, Carbon 33k, ¼W, 5%	1
R11	Resistor, Carbon 47Ω, ¼W, 5%	1
R12,13	Resistor, Carbon 5.1k, ¼W, 5%	2
RP1(3)	Resistor Pack 1k, 0.8W, ±2%	1
U1,3	Integrated Circuit, 7254 Quad VMOS Drive Transistors	2
U2	Integrated Circuit, 7250 Coil Pre-Driver	1
U4	Integrated Circuit, 7242 Dual Formatter/Sense Amplifier	1
U5	Integrated Circuit, 7110 1 Megabit Bubble Memory	1
U6	Integrated Circuit, 7230 Current Pulse Generator	1
U7	Integrated Circuit, 7220 Bubble Memory Controller	1

Table 2-1. BPK-72 Parts List (Cont'd.)

Reference	Description	Qty
XU1,3	Socket, 14 pin	2
XU2	Socket, 16 pin	1
XU4	Socket, 20 pin	1
XU5	Socket, 20 pin	1
XU6	Socket, 22 pin	1

(1) VN0106N3 manufactured by Supertex, 1225 Bordeau Drive, Sunnyvale, CA (408) 744-0100
VN10KM manufactured by Siliconix, 2201 Laurelwood Road, Santa Clara, CA (408) 988-8000

(2) Manufacturer Part Number:
CC-3481-F (Allen Bradley), MF1/10-3.48K-1% (Dale), NC55-3.48K-R (Corning)

(3) Manufacturer Part Number:
764-3-R1K (Beckman), 4308R-102-102 (Bourns), MSP08A-03-102G (Dale)

(4) The required mating connector (not supplied) for the IMB-72 board is a TRW Cinch 251-22-30-160 connector manufactured by TRW Cinch Connectors (1501 Morse Avenue, Elk Grove Village, Illinois, 60007). Equivalent connectors from other manufacturers are acceptable.

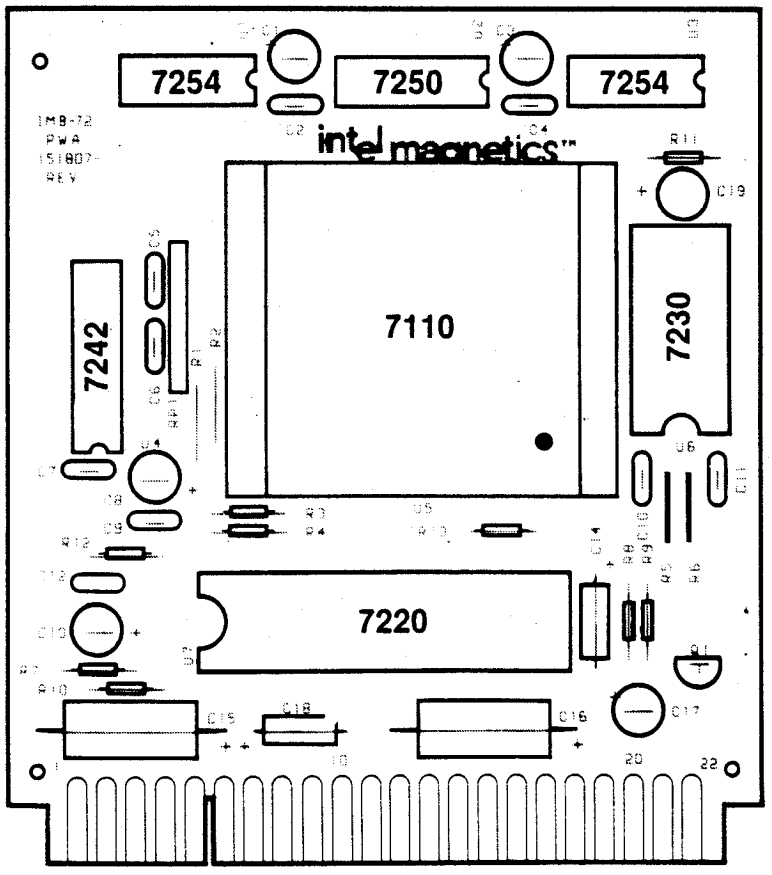


Figure 2-1. Assembly Drawing for the IMB-72 Board 121685-4

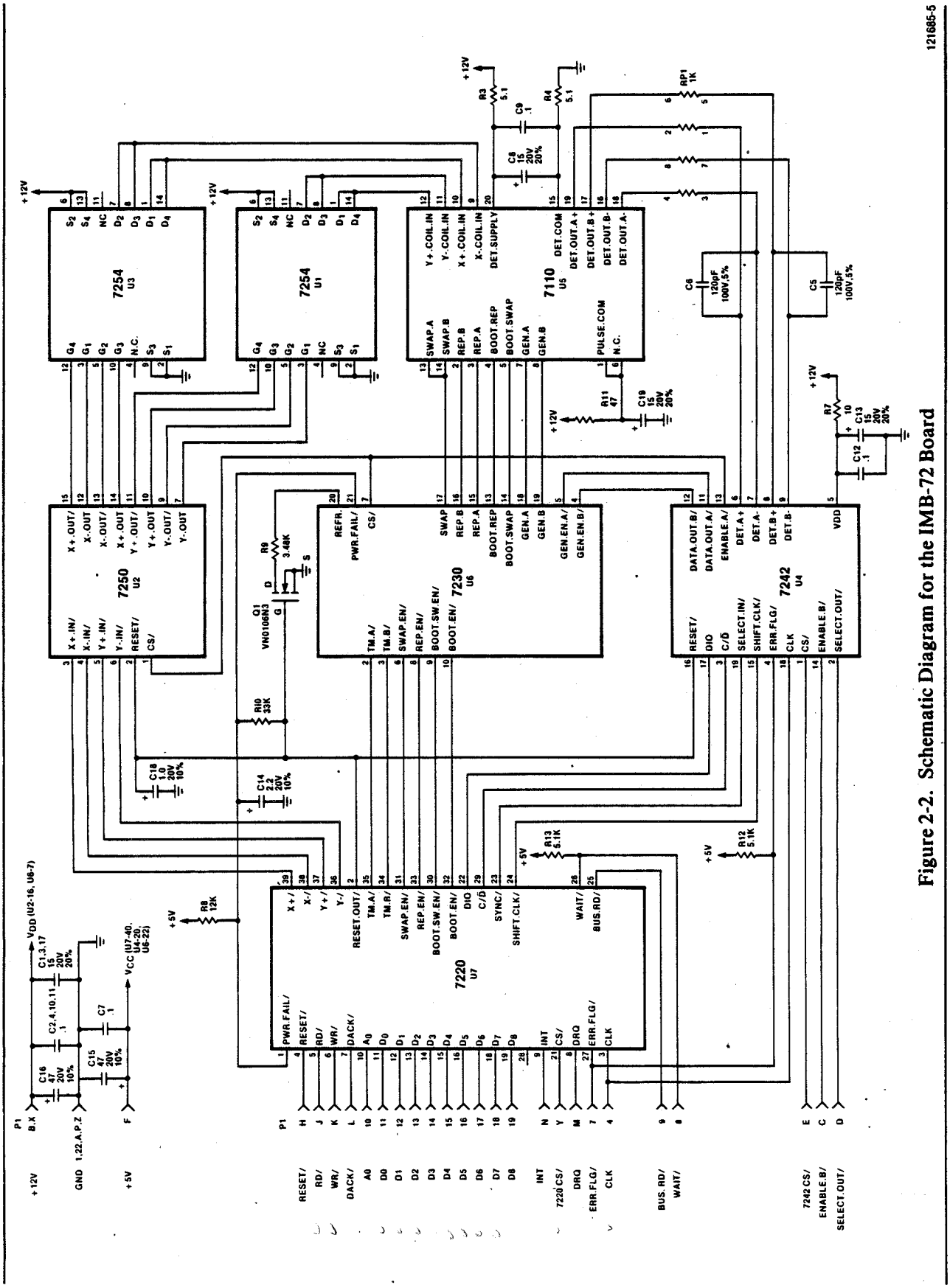


Figure 2-2. Schematic Diagram for the IMB-72 Board

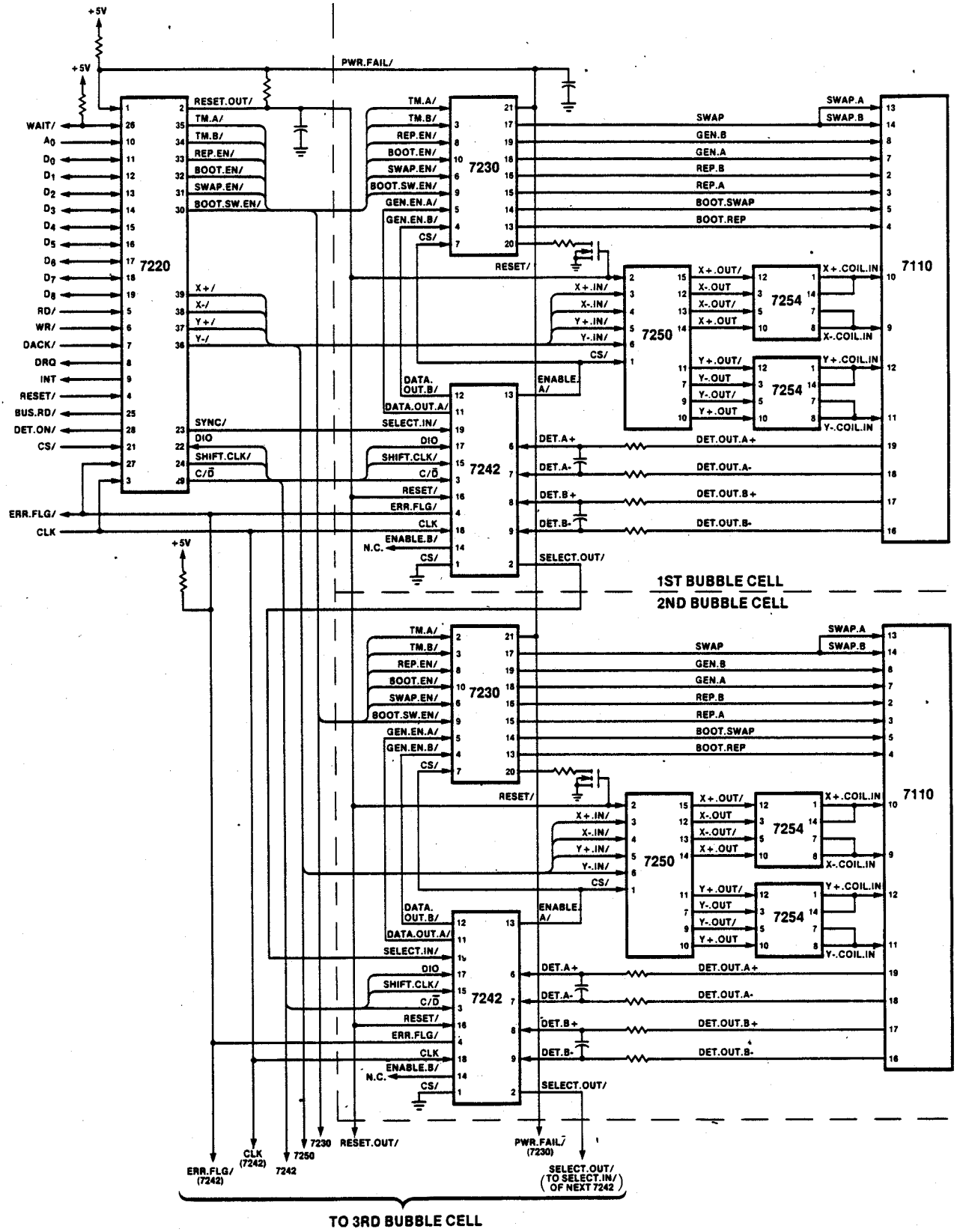


Figure 2-3. Schematic Diagram for a One-Megabyte System 121685-6

2.4.3 Multibank Systems

Systems larger than one megabyte with one controller may be built by utilizing the chip select (CS/) pins of the immediate support circuits in each bubble storage unit.

2.5 User Interface Requirements

This section describes the characteristics of the user interface needed to properly operate the bubble memory system. The requirements are such that they are easily satisfied by most currently available microprocessors. Thus, for example, the bubble memory systems described in this manual will interface easily to systems using 8080, 8085, 8086 or 8088 microprocessors.

2.5.1 User Interface Signals

The user interface signals can be seen at the left of figure 2-2. Referring to this figure for orientation, the functions of these signals are described below. A "/" after the signal name denotes an active low signal.

RD/ and WR/ are input signals that indicate to the BMC (bubble memory controller) that the user wishes, respectively, to read from or write to one of the user-assessible registers in the BMC.

A0 is an address bit that selects which registers shall be involved. If A0 is a logic "0" a parametric register is involved. If A0 is a logic "1" the command register, status register, or the register address counter is involved.

D0 to D7 are the eight bidirectional data lines used for carrying data between the user's system and the BMC. D0 is the least significant bit (LSB). D7 is the most significant bit (MSB).

D8 is the parity bit for the D0-D7 data lines. The BMC checks for odd parity during transfers from the user's system to bubble memory and generates an odd parity bit during transfers from the bubble memory to the user's system.

RESET/, when low, interrupts BMC sequencer activity, and initiates a RESET sequence. After the RESET sequence is concluded, a logic zero on RESET/ causes a logic zero on RESET.OUT/, a BMC output signal to the MBM support circuitry. After a RESET sequence has occurred, the next BMC command must be the Initialize command, or alternatively, the Abort command followed by the MBM Purge command.

PWR.FAIL/, when low, causes the BMC to begin a controlled stop sequence, similar to that described for RESET/.

DET.ON/, when low, indicates that an MBM is detecting. It is useful in connection with MBM power saving techniques.

DRQ is the data transfer request signal. In DMA mode, it indicates to the host system that the BMC is ready to transfer one byte of data to or from the host system. In other data transfer modes, DRQ indicates that 22 bytes can be transferred.

DACK/ is the DMA acknowledge signal. When DACK/ is low, the DMA controller is indicating to the BMC that the next memory cycle is available for data transfer. DACK/ should be active only when DMA (direct memory access) data transfer is desired, and then only when the DMA ENABLE bit has been set in the BMC's enable register. In general, 7220 CS/ should not be active during DMA transfers. However, CS/ may be activated to read the status register between DMA cycles. If DMA is not used, DACK/ must be pulled up to Vcc with a 5.1kΩ resistor.

7220 CS/ is the BMC's chip select signal. Except during DMA transfers, a logic "1" on 7220 CS/ disables the BMC from responding to any other user interface signals (except RESET/ or PWR.FAIL/). The BMC ignores all system bus activity and keeps all its bus outputs in the high impedance state.

7242 CS/ is the chip select signal for the FSA (formatter/sense amplifier). When 7242 CS/ is a logic "1" the FSA is completely disabled. 7242 CS/ is useful for selecting banks of FSAs. In single-bank systems, 7242 CS/ may be tied low.

CLK is the clock input for both the BMC and the FSAs. The clock frequency should be 4 MHz±0.1%, with a duty cycle of 50%±5%. This assures that the 7110 rotational field specification is met.

ERR.FLG/ is an error flag through which the FSAs can notify the BMC of an error condition within an FSA. When BMCs are operated in parallel, the ERR.FLG/ lines are tied together.

WAIT/ is a bidirectional signal. When BMCs are operated in parallel, the WAIT/ pins are tied together. When an error is detected in one of the BMCs, the WAIT/ signal indicates to the other BMCs that they should halt along with the BMC that has detected the error condition.

BUS.RD/, when high, indicates that the BMC's DIO line (over which data passes between the BMC and the FSAs) is in the input mode (data passes from an FSA to the BMC). When low, BUS.RD/ indicates that the DIO line is in the output mode (data passes from the BMC to an FSA).

INT is the interrupt line from the BMC to the host system. It notifies the user that the BMC status has changed, and that the BMC requires servicing by the host system.

2.5.2 Signal Timing Requirements

The timing requirements for the user interface are generally those of the MULTIBUS specification. For detailed timing requirements, refer to the 7220 BMC data sheet.

2.5.3 Data Rate Requirements

The data rate requirements for the user interface are a function of the number of MBMs in the system, as well as the exact multiplexing mode used for the FSAs. These requirements are listed in table 2-2. The user should study this table in relation to the type of data transfer mode (polled, interrupt driven, or DMA) he plans to use, to be sure that the host system response times will be adequate.

Table 2-2. User Data Transfer Rate Requirements

Number of MBMs Operating In Parallel	Maximum Data Transfer Rate Between BMC FIFO and the FSAs During Write Bubble Data Commands	Maximum Data Transfer Rate Between BMC FIFO and the FSAs During Read Bubble Data Commands	
		*MFBTR=0	*MFBTR=1
1	12.5 K bytes/second	50 K bytes/second	12.5 K bytes/second
2	25 K bytes/second	100 K bytes/second	25 K bytes/second
4	50 K bytes/second	200 K bytes/second	50 K bytes/second
8	100 K bytes/second	400 K bytes/second	100 K bytes/second

*See Section 3.2.2 for a discussion of the MFBTR bit.

2.5.4 Power Requirements

The bubble memory system runs on standard +5V and +12V DC power. The current requirements are a function of the number of MBMs in the system, as well as the exact multiplexing mode used for the FSAs. These current requirements are listed in table 2-3.

Table 2-3. Bubble Memory Power Requirements

Configuration		Capacity (Bytes)	Power (Watts)					
BPK-72	BPK-70		+5V (Maximum)	+12V (Maximum)	Total Active (Maximum)	Total Active (Typical)	Total Standby (Maximum)	Total Standby (Typical)
1	0	128K	1.92	4.80	6.72	3.90	3.03	1.55
1	1	256K	2.79	9.60	12.39	7.30	4.57	2.60
1	2	384K	3.65	14.40	18.05	10.70	6.11	3.65
1	3	512K	4.52	19.20	23.72	14.10	7.65	4.70
1	4	640K	5.38	24.00	29.38	17.50	9.19	5.75
1	5	768K	6.25	28.80	35.05	20.90	10.73	6.80
1	6	896K	7.11	33.60	40.71	24.30	12.27	7.85
1	7	1024K	7.98	38.40	46.38	27.70	13.81	8.90

Breakdown by Device	Power (Watts)					
	+5V (Maximum)	+12V (Maximum)	Total Active (Maximum)	Total Active (Typical)	Total Standby (Maximum)	Total Standby (Typical)
7110	0	1.740	1.740	1.480	0.440	0.290
7220	1.050	0	1.050	0.500	1.050	0.500
7230	0.235	0.440	0.675	0.390	0.475	0.225
7242	0.630	0.375	1.005	0.500	1.005	0.500
7250	0	0.945	0.945	0.480	0.060	0.030
7254(2)	0	1.300	1.300	0.550	0	0

2.6 Fabrication of Boards for Production

In the design of bubble memory boards, attention to a few simple guidelines can effectively preclude many of the problems that might otherwise appear in the bubble memory operation. Careful design is especially important on bubble memory boards, because of the presence of large drive signals, which must coexist in close proximity to small sense signals. The following guidelines are presented as an aid to the board designer. The IMB-72 board is used for purposes of illustration, but the principles involved apply to larger boards as well.

Power Supply/Ground Layout. Although multilayer PC boards with separate ground and voltage layers for each power supply are desirable, very good results can be achieved at much lower cost by constructing a "radial" power/ground system as in figures 2-4A and 2-4B. In this approach the ground system is configured in radial branches from a common source point (usually near the power supply feed point or connector). The ground branches should be as wide as practical and the power traces routed above, below or interleaved with the ground branch. It is especially important that the low-level signal circuits such as the 7242 do not share a branch with high-level drive circuits, such as the output lines of the 7250 coil drive or the 7230 current pulse generator. The net effect of this technique is to minimize power feed line impedance and isolate high level power supply currents from low level signal return areas.

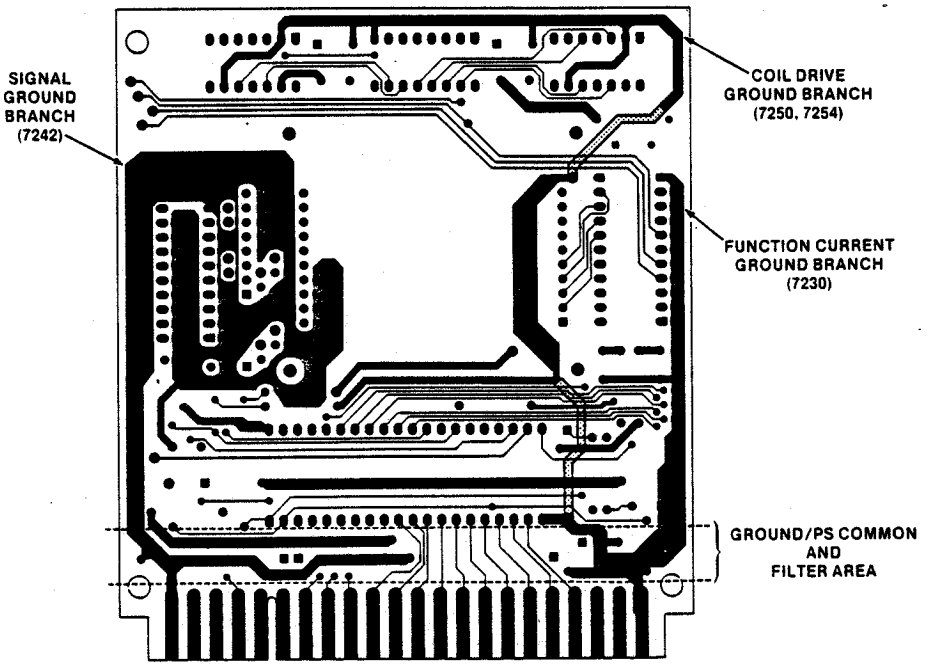


Figure 2-4A. IMB-72 Ground Branches 121685-7

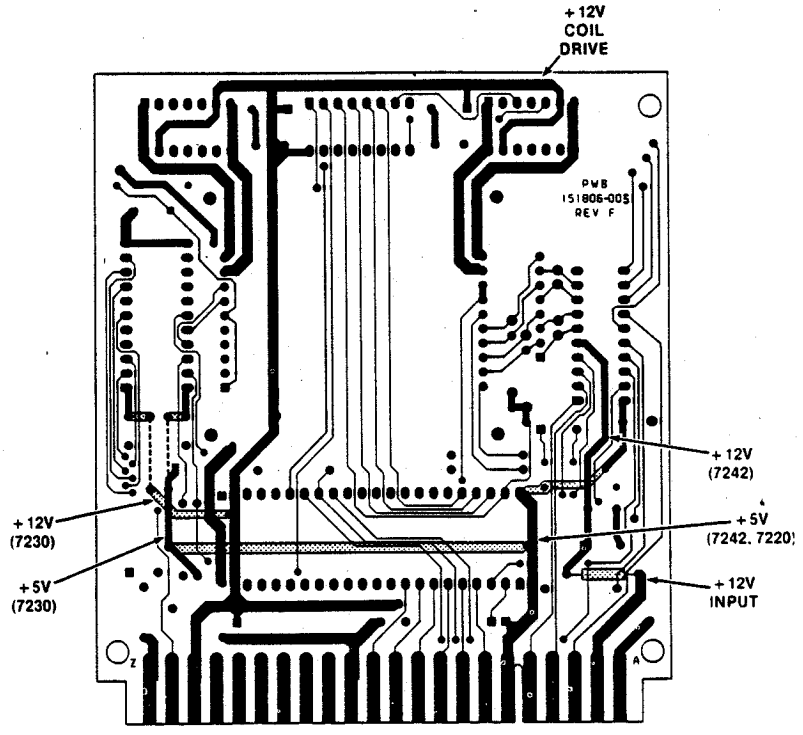


Figure 2-4B. IMB-72 Power Distribution 121685-8

Filter Components. All individual chip filter components should be located as close to chip pins as possible. Larger electrolytic capacitors on each power supply feed line should be located at the common point of the radial system.

Bubble Signal Trace Layout. One critical layout area for bubble signals exists between the 7110 detector terminals and the 7242 sense amplifier input terminals. These lines must be kept as short and direct as possible, including the path through the recommended signal filter. It is essential that no digital logic signal leads cross or come near the bubble signal leads (including those on the other side of the board). Separation of 0.5 inches and avoidance of logic traces parallel to bubble signal runs is usually sufficient. An example of the bubble signal trace layout is shown in figure 2-4C.

Coil Drive Feed to 7110. The coil drive feeder traces to the X and Y coils in the 7110 should be kept as short as possible by placing the 7254 VMOS drivers as close to the 7110 as practical. The 7254 connections to each coil should be wide (.1" to .2") and should form a tightly closed loop as in figure 2-4D.

Bubble Memory and Support Circuit Package Layout. The most important requirement for achieving a good circuit layout is a good IC package layout. The 7110 MBM and the Intel MBM support electronics have been designed to optimize board layout, through appropriate chip pin assignments, which also maximize circuit density and greatly ease the task of achieving a good circuit layout. The recommended MBM/support electronics cell layout is shown in figure 2-1. Adherence to these layout guidelines is strongly recommended, in order to achieve the specified performance levels of the MBM and support electronics.

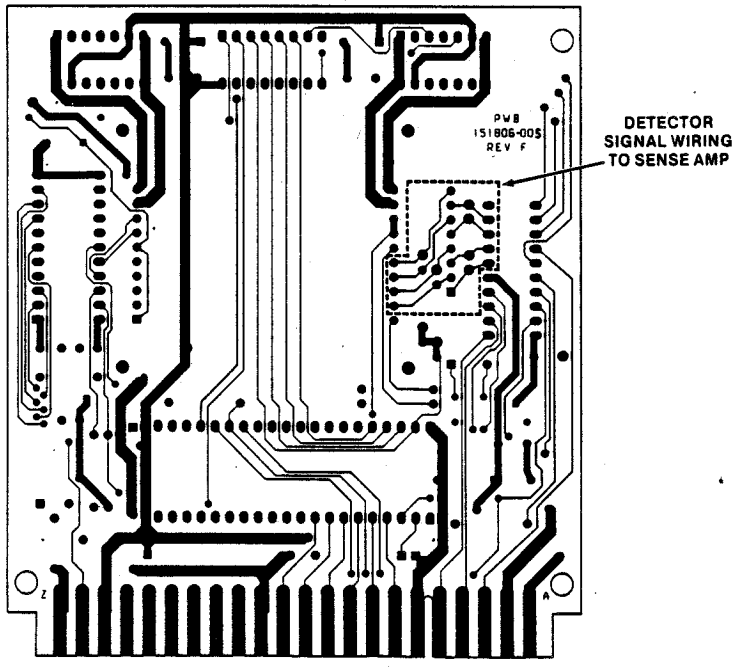


Figure 2-4C. IMB-72 Detector/Sense Amplifier Signal Routing 121685-9

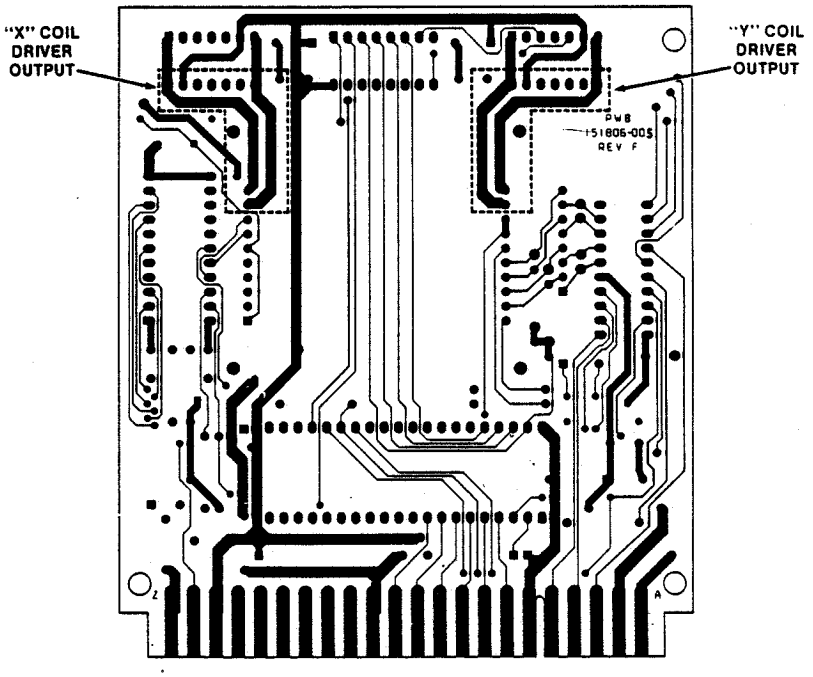


Figure 2-4D. IMB-72 Coil Driver Output Signal Routing 121685-10

CHAPTER 3
OPERATING INFORMATION

3.1 Introduction

This chapter contains information primarily of use to the bubble memory system software designer. It shows how to operate the bubble memory system from the user interface. Software examples (in ISIS-II 8080/8085 assembly language) are included for the operations described.

3.2 User-Accessible Registers

The user operates the bubble memory system by reading from or writing to specific registers within the bubble memory controller (BMC). This section identifies these registers and gives brief functional descriptions, including bit configurations and address assignments.

3.2.1 Register Addressing

Selection of the user-accessible registers depends on register address information sent from the user to the BMC. This address information is sent via a single address line (designated A0) and also via data lines D0 through D4. The following table shows how the information on these lines is used to select a specific register. CMDR refers to the command register, and RAC refers to the register address counter.

Address	BMC Read Request (RD/ = 0)	BMC Write Request (WR/ = 0)
A0=1	READ STATUS REGISTER	IF D4=1, LOAD CMDR IF D4=0, LOAD RAC
A0=0	READ REGISTER SPECIFIED BY RAC	LOAD REGISTER SPECIFIED BY RAC

Both CMDR (command register) and RAC (register address counter) are 4-bit registers which are loaded from D0-D3. By studying the above table, we can see that the status register is selected and read by a single read request, and that the command register is selected and loaded by a single write request. The remaining registers are accessed by a two-stage process, in which the desired register is first selected by placing its address in the RAC, and then read or written with a subsequent read or write request.

Table 3-1 gives a complete listing of the address assignments for the user-accessible registers. The registers are listed in two groups. The first group (STR, CMDR, RAC) consists of those registers that are selected and accessed in one operation. The second group (UR, BLR, ER, AR, FIFO) consists of those registers that are selected according to the contents of RAC.

Table 3-1. Address Assignments for the User-Accessible Registers

A0	D7	D6	D5	D4	D3	D2	D1	D0	Symbol	Name of Register	Read/Write
1	0	0	0	1	C	C	C	C	CMDR	Command Register	Write Only
1	0	0	M	0	B	B	B	B	RAC	Register Address Counter	Write Only
1	S	S	S	S	S	S	S	S	STR	Status Register	Read Only

A0	RAC B3	RAC B2	RAC B1	RAC B0	Symbol	Name of Register	Read/Write
0	1	0	1	0	UR	Utility Register	Read or Write
0	1	0	1	1	BLR LSB	Block Length Register LSB	Write Only
0	1	1	0	0	BLR MSB	Block Length Register MSB	Write Only
0	1	1	0	1	ER	Enable Register	Write Only
0	1	1	1	0	AR LSB	Address Register LSB	Read or Write
0	1	1	1	1	AR MSB	Address Register MSB	Read or Write
0	0	0	0	0	FIFO	FIFO Data Buffer	Read or Write

SSSSSSSS = 8-bit status information returned to the user from the STR
CCCC = 4-bit command code sent to the CMDR by the user.
BBBB = 4-bit register address sent to the RAC by the user.
B3B2B1B0 = 4-bit contents of RAC at the time the user makes a read or write request with A0=0.
LSB = Least Significant Byte
MSB = Most Significant Byte
M = Modifier (see Status Register in Section 3.2.2 for a discussion of the D5 bit when loading RAC).

Figure 3-1 shows the data flow for the user-accessible registers. The register file shown in this figure contains the registers with addresses 1010 through 1111. These registers are sometimes called parametric registers, because they contain flags and parameters that determine exactly how the BMC will respond to commands written to the CMDR. It is generally necessary to load the required parametric information into these registers before issuing commands to the BMC. A frequently used procedure is to first load the parametric registers in the order of increasing register address, and then issue a read or write command to the CMDR.

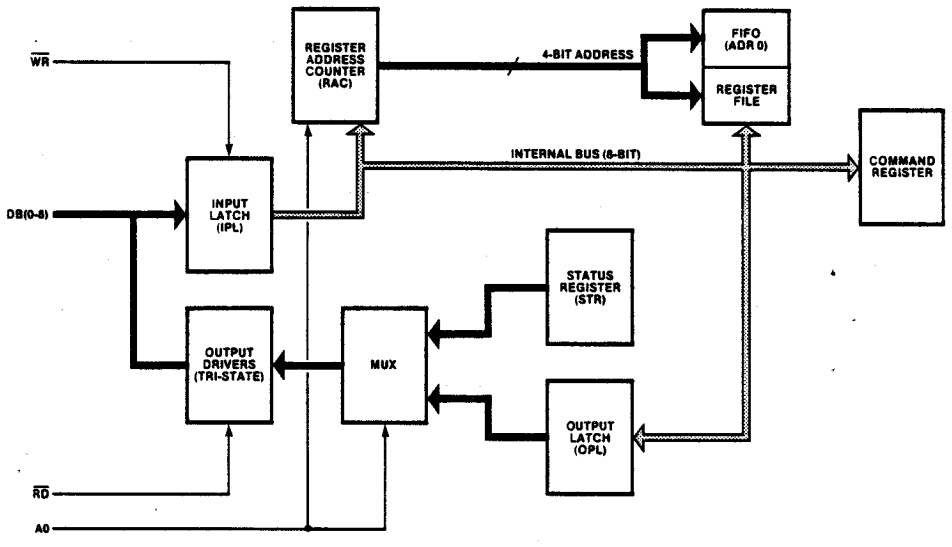


Figure 3-1. Data Flow for the User-Accessible Registers 121685-11

To facilitate such operations, the BMC automatically increments the RAC by one count after each transfer of data to or from a parametric register. This enables the user to access the next register without having to make another write reference to the RAC. This cuts down on the number of operations (and hence the time required) to load all the parametric registers.

The RAC increments from the initially loaded value through address 1111 and then on to 0000 (the FIFO address). When it has reached 0000, it no longer increments. All subsequent data transfers (with A0=0) will be to or from the FIFO until such time as the RAC is loaded with a different register address.

We can now proceed to discuss each of the user-accessible registers individually, examining its bit configuration and explaining its overall function.

3.2.2 Functional Descriptions

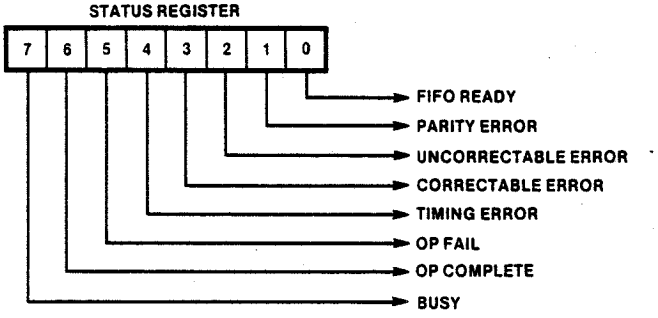
Command Register (CMDR) 4 Bits, Write Only

The user issues a command to the BMC by writing a 4-bit command code to the CMDR. Section 3.3 discusses the various commands that the BMC can execute. The command codes themselves are listed in Section 3.3.1.

The user must not write a command to the command register until all relevant parametric information has been loaded into the parametric registers.

Status Register (STR) 8 Bits, Read Only

The user reads the BMC status register in response to an interrupt signal, or as part of the polling process in a polled data transfer mode. The status register provides information about error conditions, completion or termination of commands, and about the BMC's readiness to transfer data or accept new commands. The individual bit descriptions are as follows:



BUSY, when active ("1"), indicates that the BMC is in the process of executing a command. During "non-read" operations (i.e., data transfer from BMC FIFO to FSA or execution of a non-transfer command), BUSY remains active only while the BMC actually is executing the command. During read operations (i.e., data transfer from FSA to BMC FIFO or during execution of an Initialize command), BUSY is active during command execution and remains active as long as there is "sufficient" data in the FIFO to hold the DRQ pin active. In the DMA mode, only one byte in the FIFO is sufficient to hold DRQ active; in the non-DMA mode, 22 bytes are required to hold DRQ active. When BUSY is inactive ("0"), the BMC is ready to receive a new command.

OP COMPLETE (when=1) indicates the successful completion of a command.

OP FAIL (when=1) indicates that the BMC was unable to successfully complete the current command.

TIMING ERROR (when=1) indicates that an FSA has reported a timing error to the BMC, or that the host system has failed to keep up with the BMC, thereby causing the BMC FIFO to overflow or to become empty. TIMING ERROR is also set if no bootloop sync code is found during the read boot loop phase of initialization, or if a Write Bootloop command is issued when the WRITE BOOTLOOP ENABLE bit in the enable register is equal to zero.

CORRECTABLE ERROR (when=1) indicates that an FSA has reported to the BMC that a correctable error has been detected in the last data block transferred.

UNCORRECTABLE ERROR (when=1) indicates that an FSA has reported to the BMC that an uncorrectable error has been detected in the last data block transferred.

PARITY ERROR (when=1) indicates that the BMC's parity check circuitry has detected a parity error on a data byte sent to the BMC by the user on the data lines D0-D8. The BMC implements odd parity checking on data sent to the BMC by the user and generates odd parity for data sent to the user by the BMC.

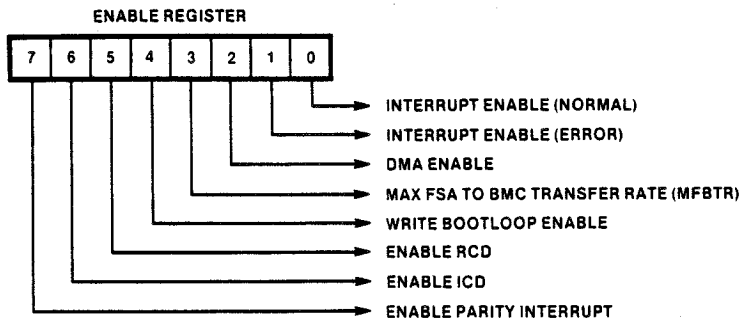
FIFO READY has two functions. When the BMC is not busy (BUSY=0), FIFO READY = 0 indicates that the FIFO and its input and output latches are all completely empty. When the BMC is busy (BUSY=1), FIFO READY = 1 indicates (during read operations) that the BMC output latch has data for the user, or (during write operations) that the BMC input latch is ready to receive data from the user.

During execution of write commands, FIFO READY is not valid on the first read of the status register in which BUSY = 0.

Bits 1 through 6 of the status register are valid only when BUSY (bit 7) = 0 and are reset when a new command is issued. These bits may also be reset by writing to the register address counter (RAC) with the modifier bit (bit D5) equal to one and bits D3 through D0 equal to zero. Resetting the status register bits also clears the INT (interrupt) signal on the user interface.

Enable Register (ER) 8 Bits, Write Only

The user sets various bits of the enable register or enable or disable various functions within the BMC or the FSAs. The individual bit descriptions are as follows:



In the above figure and in the text below, the following abbreviations are used:

ICD = INTERNALLY CORRECT DATA
RCD = READ CORRECTED DATA
UCE = UNCORRECTABLE ERROR
CE = CORRECTABLE ERROR
TE = TIMING ERROR

ENABLE PARITY INTERRUPT enables the BMC to interrupt the host system (via the INT line) when the BMC detects a parity error on the data bus lines D0-D7.

ENABLE ICD enables the BMC to give the Internally Correct Data command to an FSA when an error has been detected by the FSA's error detection and correction circuitry. The FSA responds to such a command by internally cycling the data through its error correction network. When finished, the FSA returns status to the BMC as to whether or not the error is correctable. The value of ENABLE ICD affects the action of INTERRUPT ENABLE (ERROR), as explained below.

ENABLE RCD enables the BMC to give the Read Corrected Data command to an FSA in which an error has been detected. This command causes the FSA to correct the error (if possible) and also to transfer the corrected data to the BMC. When the data transfer has been completed, the BMC reads FSA status to determine whether or not the error was correctable. In the case of an uncorrectable error, bad data would have been sent to the user. The value of ENABLE RCD affects the action of INTERRUPT ENABLE (ERROR), as explained below.

WRITE BOOTLOOP ENABLE (when=1) enables the bootloop to be written. If this bit is equal to zero, and a Write Bootloop command is received by the BMC, the command is aborted and the TIMING ERROR bit is set in the STR.

MFBTR (Maximum FSA to BMC Transfer Rate) controls the maximum burst transfer rate from the FSAs to the BMC FIFO, in relation to the last page of a multiple-page transfer (in a one-page transfer the last page is the only page). The following table shows the maximum required host interface data rate as a function of the MFBTR bit.

Number of MBMs Operated in Parallel	Maximum Required Host Interface Data Rate	MFBTR Bit	
		Read Command	Write Command
1	50K byte/sec	0	N/A
2	100K byte/sec	0	N/A
4	200K byte/sec	0	N/A
8	400K byte/sec	0	N/A
1	12.5K byte/sec	1	0
2	25K byte/sec	1	0
4	50K byte/sec	1	0*
8	100K byte/sec	1	0

NOTE: The MFBTR bit should always be set to "0" for all commands except "Read."

DMA ENABLE (when=1) enables the BMC to operate in DMA data transfer mode, using the DRQ and DACK/ signals in interaction with a DMA controller. When equal to zero, DMA ENABLE sets up the controller to support interrupt driven or polled data transfer.

INTERRUPT ENABLE (ERROR) selects error conditions under which the BMC stops command execution and interrupts the host processor (via the INT line). INTERRUPT ENABLE (ERROR) operates in conjunction with ENABLE ICD and ENABLE RCD, as shown in the following table:

Enable ICD	Enable RCD	Interrupt Enable (ERROR)	Interrupt Action
0	0	0	No interrupts due to errors
0	0	1	Interrupt on TE only
0	1	0	Interrupt on UCE or TE
0	1	1	Interrupt on UCE, CE, or TE
1	0	0	Interrupt on UCE or TE
1	0	1	Interrupt on UCE, CE, or TE
1	1	0	Not used
1	1	1	Not used

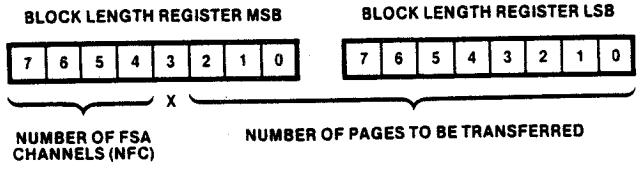
INTERRUPT ENABLE (NORMAL) (when=1) enables the BMC to interrupt the host system (via the INT line), when a command execution has been successfully completed (OP COMPLETE = 1 in the STR).

Utility Register (UR) 8 Bits, Read or Write

The utility register is a general purpose register available to the user in connection with bubble memory system operations. It has no direct effect on the BMC operation, but is provided as a convenience to the user.

Block Length Register (BLR) 16 Bits, Write Only

The contents of the block length register determine the system page size and also the number of pages to be transferred in response to a single bubble data read or write command. The bit configuration is as follows:



The system page size is proportional to the number of magnetic bubble memory modules (MBMs) operating in parallel during the data read or write operation. Each MBM requires two FSA channels. Bits 4 through 7 of BLR MSB actually specify the number of FSA channels to be accessed. Taking a 1-megabyte system (eight MBMs) as an example, the following table shows how these four bits specify the number of FSA channels, the system page size, the number of pages in the system, and the range of logical page addresses:

BLR MSB 7 6 5 4	NFC	System Page Size	Number of Pages	Address Range
0 0 0 1	2	512 BITS (544)	16384	0000-3FFF
0 0 1 0	4	1024 BITS (1088)	8192	0000-1FFF
0 1 0 0	8	2048 BITS (2176)	4096	0000-0FFF
1 0 0 0	16	4096 BITS (4352)	2048	0000-07FF
0 0 0 0	1	—	—	—

Note that two sets of values are given for the system page size. The numbers in parentheses represent system page size when error correction is not used. The numbers not in parentheses represent system page size when error correction is used.

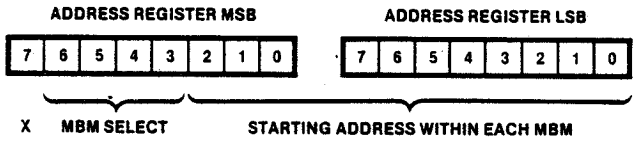
The last line of the table represents a process in which only one FSA channel is accessed. This option is not used during normal read or write operations.

The BLR LSB, together with the three least significant bits of the BLR MSB, specifies the number of pages to be transferred. Up to 2048 pages can be transferred in response to a single bubble data read or write command, hence the requirement for 11 bits. All 11 bits equal to zero specifies a 2048 page transfer.

BLR MSB Bit 3 is not used.

Address Register (AR) 16 Bits, Read or Write

The contents of the address register determine which MBM group is to be accessed, and, within that group, what starting address location shall be used in a data read or write operation. The bit configuration is as follows:



Within each MBM there are 2048 possible starting address locations for a data read or write operation, hence the requirement for 11 bits in the starting address.

The selection of the MBMs to be read or written is specified by AR MSB Bits 3-6. The BMCs interpretation of these bits depends on the number of MBMs in a group, which is specified by BLR MSB Bits 4-7, as explained above.

The following table shows which MBM groups are selected in response to given values for BLR MSB Bits 4-7 and AR MSB Bits 3-6. A 1-megabyte system (8 MBMs) is represented, with the FSA channels numbered 0 through F:

Table 3-2. Selection of FSA Channels

AR MSB BITS (6,5,4,3)	BLR MSB BITS (7,6,5,4)				
	0000	0001	0010	0100	1000
0 0 0 0	0	0,1	0,1,2,3	0 to 7	0 to F
0 0 0 1	1	2,3	4,5,6,7	8 to F	
0 0 1 0	2	4,5	8,9,A,B		
0 0 1 1	3	6,7	C,D,E,F		
0 1 0 0	4	8,9			
0 1 0 1	5	A,B			
0 1 1 0	6	C,D			
0 1 1 1	7	E,F			
1 0 0 0	8				
1 0 0 1	9				
1 0 1 0	A				
1 0 1 1	B				
1 1 0 0	C				
1 1 0 1	D				
1 1 1 0	E				
1 1 1 1	F				

As explained above, the accessing of single FSA channels is done only as part of diagnostic processes. AR MSB Bit 7 is not used.

FIFO Data Buffer (FIFO) 40 x 8 Bits, Read or Write

The BMC FIFO is a 40-byte buffer through which data passes on its way from the FSAs to the user, or from the user to the FSAs. The primary purpose of the FIFO is to reconcile differences in timing requirements between the user interface to the BMC and the BMC's interface to the FSAs. This allows the data transfer to proceed in a more asynchronous and flexible manner, and relaxes timing constraints, both to the FSAs and also to the user's equipment. The user's system must, however, meet the data rate requirements discussed in Section 2.5.3.

When the BMC is busy (executing a command) the FIFO functions as a data buffer. When the BMC is not busy, the FIFO is available to the user as a general purpose FIFO. For this reason, the BMC is said to be in the general purpose FIFO mode when it is not busy.

The FIFO is dual port; data can be written at one port while simultaneously data is being read at the other port. The FIFO has input and output latches providing additional buffering. A total of 43 bytes of data may be stored in the FIFO, the FIFO's input and output latches, and the BMC's input latch.

During execution of a command involving data transfer (between the user and the FSAs) the data passes through the FIFO. The status of the FIFO is indicated by the FIFO READY bit in the STR (status register). FIFO READY = 1 indicates that (during a write operation) there is room in the FIFO for more data or that (during a read operation) there is data available in the FIFO.

The DRQ line (on the user interface) also indicates FIFO status. In DMA data transfer mode (DMA ENABLE = 1 in the enable register) the DRQ line and the DACK/ line together provide a standard DMA data transfer capability; the BMC can handshake with an 8257 or 8237 DMA controller. In the polled and interrupt-driven data transfer modes (DMA ENABLE = 0 in the enable register) the DRQ line indicates 'FIFO HALF FULL' or 'FIFO HALF EMPTY'. This provides an interrupt capability. During write operations, DRQ activates when there is room in the FIFO for 22 more bytes of data. During read operations, DRQ activates when there are 22 bytes of data in the FIFO for the user to read. It is recommended that the user transfer data in blocks of 22 bytes, so that the user's processor doesn't have to spend a disproportionate amount of time servicing this type of interrupt.

In general purpose FIFO mode (BMC not busy), FIFO READY = 0 indicates that the entire FIFO, including its input and output latches, is empty. As mentioned above, when the BMC is not busy, the FIFO is available to the user as a general purpose FIFO. A consequence of using the FIFO as a general purpose buffer is that the user is responsible for making sure that the FIFO is reset prior to issuing a write command.

The FIFO is addressed automatically after the last parametric register has been written since the RAC (register address counter) is self-incrementing. Alternatively, the user can explicitly address the FIFO by writing address zero into the RAC. Since writing the RAC clears byte 0 of the FIFO, to prevent accidental loss of data, a user must never write an address (even the address of the FIFO itself) into the RAC while the FIFO contains any data.

After a Write Bubble Data command is issued, the BMC will not start the data transfer until there are at least two bytes of data in the FIFO. During the execution of read or write commands, it is the user's responsibility to keep up with the data transfer process, so that the FIFO does not overflow or underflow. If the FIFO does overflow or underflow before the specified data transfer is complete, the TIMING ERROR bit is set in the STR.

3.3 Commands

As mentioned above, the user operates the bubble memory system by reading from or writing to the user-accessible registers in the BMC. The user issues commands to the BMC by writing to the command register (CMDR). This is done by setting A0 = 1, (D7,D6,D5,D4) = (0,0,0,1), and sending the 4-bit command code via (D3,D2,D1,D0). Before issuing a command, the user should be sure that the proper operands and modifiers for the command are in the parametric registers (see Section 3.2 for details). Then the command itself is issued. While the BMC is executing the command, the user must not write to any BMC register except the FIFO. The user can read the STR at any time, to determine whether or not the BMC is busy.

Commands can be written to CMDR only when BUSY = 0 in the STR. When the BMC accepts the command, BUSY is set (=1). An Abort command, however, is accepted by the BMC even when BUSY = 1.

When a command has been completed (or terminated) BUSY is reset. An interrupt is generated if the corresponding interrupt enable bits are set in the enable register (ER).

3.3.1 Command Codes

The following table lists the 4-bit command codes used to issue the sixteen commands recognized by the BMC:

D3	D2	D1	D0	Command Name
0	0	0	0	Write Bootloop Register Masked
0	0	0	1	Initialize
0	0	1	0	Read Bubble Data
0	0	1	1	Write Bubble Data
0	1	0	0	Read Seek
0	1	0	1	Read Bootloop Register
0	1	1	0	Write Bootloop Register
0	1	1	1	Write Bootloop
1	0	0	0	Read FSA Status
1	0	0	1	Abort
1	0	1	0	Write Seek
1	0	1	1	Read Bootloop
1	1	0	0	Read Corrected Data
1	1	0	1	Reset FIFO
1	1	1	0	MBM Purge
1	1	1	1	Software Reset

The following software example shows the assembly language code that may be used to issue the above commands to the BMC:

```
CMDST EQU 49H       ;PORT ADDRESS OF 7220 WITH A0=1
DATREG EQU 48H      ;PORT ADDRESS OF 7220 WITH A0=0

INIT EQU 11H        ;INITIALIZE COMMAND
READ EQU 12H        ;READ COMMAND

CMND1:              ;SEND INITIALIZE COMMAND
     MVI A, INIT
     OUT CMDST
     RET

CMND2:              ;SEND READ COMMAND
     MVI A, READ
     OUT CMDST
     RET
```

3.3.2 Command Descriptions

This section gives brief descriptions of the BMC commands. More detailed information is presented in the succeeding sections, as specific operating procedures are discussed.

The most commonly used commands are:

Initialize
Read Bubble Data
Write Bubble Data

Other commands used in normal operation are:

Read Seek
Write Seek
Abort
MBM Purge
Read Corrected Data
Reset FIFO
Software Reset
Read FSA Status

Commands relating to the bootloop, and used only for diagnostic purposes are:

Read Bootloop Register
Write Bootloop Register
Write Bootloop Register Masked
Read Bootloop
Write Bootloop

Initialize

The BMC executes the Initialize command by first interrogating the bubble system to determine how many FSAs are present, then reading and decoding the bootloop from each MBM and storing the results in the corresponding FSA's bootloop register. The NFC bits in the BLR MSB must be set to 0001 before issuing the Initialize command. The MBM GROUP SELECT bits (in the AR) must select the last MBM in the system.

Read Bubble Data

The Read Bubble Data command causes data to be read from the MBMs into the BMC FIFO. The selection of the MBMs to be accessed is determined as indicated in table 3-2, and the starting address for the read operation is as specified in the address register (AR). The block length register (BLR) specifies the number of system pages to be read. All the parametric registers must be properly set up before issuing the Read Bubble Data command.

Write Bubble Data

The Write Bubble Data command causes data to be read from the BMC FIFO and written into the MBMs. The selection of the MBMs to be accessed is determined as indicated in table 3-2, and the starting address for the write operation is as specified in the address register (AR). The block length register (BLR) specifies the number of system pages to be written. All the parametric registers must be properly set up and the FIFO must be reset before issuing the Write Bubble Data command.

Read Seek

The Read Seek command rotates the selected MBMs to a designated address location with respect to the MBM's output track. No data transfer occurs. The positioning is such that the next data location available to be read is the specified (in AR) page address plus one. The Read Seek command may be used to reduce latency (access time) in cases where information is available for the user to predict the location of an impending read reference to the MBMs. After the Read Seek command is executed, the parametric registers must be rewritten.

A Read Seek command to address zero is made by writing all 1's in the 11 least significant bits of the AR. The MBM GROUP SELECT bits are set to specify the MBM group to be read (not one less).

Write Seek

The Write Seek command rotates the selected MBMs to a designated address location with respect to the MBM's input track. No data transfer occurs. The positioning is such that the next data location available to be written is the specified (in AR) page address plus one. The Write Seek command may be used to reduce latency (access time) in cases where information is available for the user to predict the location of an impending write reference to the MBMs. After the Write Seek command is executed, the parametric registers must be rewritten.

A Write Seek command to address zero is made by writing all 1's in the 11 least significant bits of the AR. The MBM GROUP SELECT bits are set to specify the MBM group to be written (not one less).

Abort

The Abort command causes the termination of the command currently being executed by the BMC. The MBMs are stopped in a controlled manner, such that no bubble data is lost. The Abort command will be accepted by the BMC (and is typically issued) when the BMC is busy. If an Abort command is issued when the BMC is not busy, it functions as a FIFO reset.

An Abort command issued when the BMC is busy must be followed by an Initialize command or an MBM Purge command.

MBM Purge

The MBM Purge command clears all BMC registers, counters and the MBM address RAM, except it does not clear the block length register, the NFC bits, the FSA present counter, or the four high-level bits of the address register. The MBM Purge command is useful in systems where the user stores the boot loop information in an external PROM.

Read Corrected Data

The Read Corrected Data command causes the BMC to read into the BMC FIFO a 256-bit block of data from the FIFO of each selected FSA channel after a data error has been detected. The data cycles through the error correction network of the FSA. After the data has been read into the BMC FIFO, the FSA reports to the BMC whether or not the error was correctable. The Read Corrected Data command is used only when the system is in error correction mode (ENABLE ICD or ENABLE RCD set in the ER).

Reset FIFO

The Reset FIFO command clears the BMC FIFO, its input and output latches, and the BMC's input and output latches.

Software Reset

The Software Reset command clears the BMC FIFO and all registers, except those containing initialization parameters. It also causes the BMC to send the Software Reset command to every FSA in the system. No re-initialization is needed after this command.

Read FSA Status

The Read FSA Status command causes the BMC to read the 8-bit status register of all FSA channels and to store this information in the BMC FIFO. For example, in a 1-megabyte system (eight MBMs), the status of all FSA channels is read and stored in the first 16 bytes of the BMC FIFO. The Read FSA Status command is independent of all parametric registers.

Read Boot Loop Register

The Read Boot Loop Register command causes the BMC to read the boot loop register of the selected FSA channels and to store this information in the BMC FIFO. Twenty bytes are transferred for each FSA channel selected. When two FSA channels are selected, the boot loop register data from each register is interleaved (i.e., the first bit is read from the "A" channel, the second bit from the "B" channel, and so on). Reading the boot loop registers from more than two FSA channels at a time is possible, but not recommended since the user must begin to read the data from the BMC FIFO to avoid an overflow condition.

Write Boot Loop Register

The Write Boot Loop Register command causes the BMC to write the contents of the BMC FIFO into the boot loop registers of the selected FSA channels. The boot loop register data must be loaded into the FIFO prior to issuing the Write Boot Loop Register command. Twenty bytes (160 bits) are required for each FSA channel selected. When two FSA channels are selected, the host must interleave the data on a bit-by-bit basis. Writing of more than two FSA channels at a time is possible, but not recommended.

Proper operation of an FSA during the transfer of data to or from an MBM requires that each FSA boot loop register contains either 135 (error correction implemented) or 136 (error correction not implemented) logic 1's to represent the 135 or 136 minor loops of each MBM half in which the data actually is stored. Since an MBM boot loop may indicate that more than 135 or 136 loops are usable, in writing the FSA boot loop registers, the user must be sure that exactly 135 or 136 logic 1's are written into each register.

Write Boot Loop Register Masked

The Write Boot Loop Register Masked command can be used in place of the Write Boot Loop Register command. This command counts the number of logic 1's written into the boot loop registers and masks out (zeroes) any additional 1's after the proper count is reached.

Read Boot Loop

The Read Boot Loop command causes the BMC to read the boot loop from the selected MBM, and to store the decoded boot loop information in the BMC FIFO. The NFC bits in the BLR MSB must be set to 0001 before issuing the Read Boot Loop command. The Read Boot Loop command must be immediately preceded by a Reset FIFO command.

Write Boot Loop

The Write Boot Loop command causes the existing contents of the selected MBM's boot loop to be replaced by new boot loop data based on 40 bytes of information stored in the FIFO (the user must actually write 41 bytes, where the 41st byte is all 0's). Encoding of the boot loop data is done by the BMC hardware. Prior to issuing the Write Boot Loop command, the FIFO must first be reset and then the data must be loaded into the FIFO. More details on procedures for writing the boot loop are given in Chapter 5.

The NFC bits in the BLR MSB should be set to 0001 before issuing the Write Boot Loop command. The Write Boot Loop command will not execute unless the WRITE BOOT LOOP ENABLE bit has been set equal to 1 in the enable register.

3.4 Start-up Procedures

3.4.1 Power Up

To power up the bubble memory system, turn on the power while holding both PWR.FAIL/ and RESET/ low (active). The following two conditions should be satisfied:

1. PWR.FAIL/ is held low for at least two milliseconds after the power supplies have all reached approximately 94% of their respective operating voltages.
2. RESET.OUT/ is held low for at least 200 clock periods after PWR.FAIL/ goes high (inactive).

These two conditions can be satisfied through the use of a simple circuit consisting of two resistors and two capacitors. Such a circuit is described in Section 4.4, as shown in figure 2-2 as R8, R10, C14 and C18.

Condition 1 above assures that the PWR.FAIL/ signal remains active until the power supply voltage levels have been above the threshold levels of the power fail detect circuitry for 2 milliseconds. This interval allows time for the BMC to power up.

When PWR.FAIL/ goes inactive, the BMC executes a power up sequence. When this sequence is complete, the OP COMPLETE bit is set in the status register (STR). Note that in order to read the STR, the parametric registers first must be written. After writing the parametric registers, the user can check for successful power up by reading the STR. The following software example shows how to read the status register.

```
RDST:               ;READ STATUS FROM 7220 STR

     IN CMDST       ;PUT 7220 STATUS INTO ACCUMULATOR
     RET
```

3.4.2. Initialization

After successful power up the user must initialize the bubble memory system. This is done as follows:

1. Write address 1011 to the register address counter (RAC). This selects the Block Length Register Least Significant Byte (BLR LSB).
2. Send 5 bytes of data via D0-D7 with A0 = 0. This will load the parametric registers. Be sure that the NFC bits in the BLR MSB are set to 0001 and that the MBM GROUP SELECT bits (in the AR) select the last MBM in the system. If error correction is desired, set the appropriate bits in the enable register (error correction mode changes require re-initialization).
3. Send an Initialize command to the BMC (see Section 3.3 for details).

The following software example shows how to carry out the first two steps listed above:

```
;
;NAME:WTREGI
;FUNCTION:WRITES THE 7220 REGISTERS WITH INITIALIZE
VALUES
;INPUTS:NONE
;OUTPUTS:7220 B - F REGISTERS
;CALLS:NONE
;DESTROYS:A REG
;
  WTREGI:

     MVI A, 0BH    ;SELECT B - REGISTER
     OUT CMDST     ;PORT ADDRESS OF 7220 WITH A0=1

     MVI A, 01H    ;WRITE B - REGISTER
     OUT DATREG    ;PORT ADDRESS OF 7220 WITH A0=0

     MVI A, 10H    ;WRITE C - REGISTER, SELECTS 2 FSA CHANNELS
     OUT DATREG

     MVI A, 00H    ;WRITE D - REGISTER, ERROR CORRECTION NOT ENABLED
     OUT DATREG

     MVI A, 00H    ;WRITE E - REGISTER
     OUT DATREG

     MVI A, 18H    ;WRITE F - REGISTER, 4 MBM'S IN SYSTEM
     OUT DATREG

     RET
```

The user can now check to see whether or not the initialization was successful. The exact manner in which the BMC relates this information depends on what the user has written into the enable register (ER).

If an initialization problem occurs, TIMING ERROR is set in the STR. When the operation is complete, OP COMPLETE is set. If the operation fails, OP FAIL is set. During command execution, BUSY is set.

If INTERRUPT ENABLE (NORMAL) is set in the ER, then upon completion of the command execution the INT line will activate on the user interface. If INTERRUPT ENABLE (ERROR) is set in the ER, the INT line will activate when an error condition occurs (in this case TIMING ERROR).

Thus the user can obtain the required information either by polling the status register or through the use of interrupt routines.

3.5 Normal Operation

3.5.1 Addressing Methods

The user has a choice among several configurations in the address space (as seen at the user interface to the BMC). The address field may be increased with a decrease in system page size (number of bits per page) or vice versa.

The product of system page size and the number of pages (address field size) is determined by the number of MBMs in the system. For example, if there are eight MBMs, a total of one megabyte of storage capacity is available. This may be configured in four different ways, as indicated in the discussion of the block length register (BLR). The full story on how the address space is set up is contained within the discussions of the BLR and the AR (address register) given earlier.

The choice of system page size affects the data transfer rate, since the latter is proportional to the number of MBMs operating in parallel.

3.5.2 Data Transfer Modes

Three data transfer modes may be used for transferring data across the user interface between the user's system and the BMC:

1. DMA (Direct Memory Access) Data Transfer
2. Interrupt-Driven Data Transfer
3. Polled I/O

DMA mode is the highest performance system. In DMA data transfer mode, the BMC operates in conjunction with a DMA controller (such as the 8257 or the 9517/8237), using the DRQ and DACK/ lines for coordination (handshaking). With the help of the DMA controller, the BMC transfers the data to (from) the user's memory. Once the data transfer begins, program intervention is not required until the entire data transfer has been completed (or aborted because of an error condition).

The user loads the DMA controller with the starting memory address to (from) which the data is to be transferred and the number of bytes expected, then activates the DMA channel to which the BMC is connected. The user also writes compatible parameters to the user-accessible registers in the BMC, and then issues a command to start the data transfer.

When the BMC is ready to transfer a byte of data to (from) the user's memory, it activates the DRQ signal. The DMA controller, after capturing the system bus, activates the DACK/ signal to the BMC, and then makes a read (write) request to the FIFO. The BMC responds by placing a byte of data on (inputting a byte of data from) data lines D0-D7. DRQ remains active as long as the BMC has more data bytes to transfer (expects more bytes from the user's memory) and the FIFO contains at least one byte of data (one empty location) for data transfer, the same condition that sets FIFO READY = 1 in the STR. After the last byte has been transferred, the BMC deactivates the DRQ signal. When the command as a whole has been completely executed, BUSY is reset in the STR. If INTERRUPT ENABLE (NORMAL) = 1 in the ER, an interrupt is generated.

To activate the DMA mode in the BMC the user sets DMA ENABLE = 1 in the ER.

The following software example shows how to set up a DMA data transfer:

```
                              ;DMA
          ;READ FROM BUBBLE, WRITE TO SYSTEM RAM

DMA0 EQU 50H        ;DMA CONTROLLER ADDRESS
DMA1 EQU 51H        ;DMA CONTROLLER ADDRESS
DMA8 EQU 58H        ;DMA CONTROLLER ADDRESS

DMA:
     CALL WTREG     ;WRITE THE REGISTERS

     MVI C, 41H     ;DMA COMMAND
     LXI D, 4021H   ;TERMINAL COUNT FOR READ BUBBLE
     LXI H, 8000H   ;DMA DATA ADDRESS

     MVI A, 00H     ;RESET DMA MODE-SET REGISTER
     OUT DMA8
```

```
MOV A, L        ;WRITE DMA ADDRESS REGISTER
OUT DMA0
MOV A, H
OUT DMA0
MOV A, E        ;WRITE DMA TERMINAL COUNT
OUT DMA1
MOV A, D
OUT DMA1

MOV A, C        ;WRITE DMA MODE-SET
OUT DMA8

MVI A, READ     ;SEND 7220 READ COMMAND
OUT CMDST

RET
```

Interrupt-driven data transfer mode makes use of the fact that the DRQ line doubles as a 'FIFO HALF FULL/HALF EMPTY' indicator when the BMC is not in DMA mode. The user then connects the DRQ line as an interrupt. When the interrupt occurs, the user transfers a burst of data, typically 22 bytes.

Polled I/O is the lowest performance system. In polled I/O the user simply polls the FIFO READY bit in the STR to determine whether the FIFO has data (room) available. Alternatively, the user can periodically check the DRQ line to determine whether 22 bytes can be transferred.

In all three data transfer modes, the user's system (taking priority and overhead into account) must meet the data rate requirements listed in Chapter 1, in order to avoid timing errors.

3.5.3 Reading/Writing Parametric Registers

The operations involved in reading (writing) data from (to) the parametric registers in the BMC are explained in Section 3.2. The following software example shows how all the parametric registers can be written using only one write reference to the register address counter (RAC):

```
;
;NAME:WTREG
;FUNCTION:WRITES THE 7220 REGISTERS WITH USER DEFINED VALUES
;INPUTS:NONE
;OUTPUT:7220 B - F REGISTERS
;CALLS:
;DESTROYS:A-REG
;

  WTREG:

     MVI A, 0BH     ;SELECT B - REGISTER
     OUT CMDST

     LDA BREG       ;WRITE B - REGISTER
     OUT DATREG

     LDA CREG       ;WRITE C - REGISTER
     OUT DATREG
```

```
LDA DREG        ;WRITE D - REGISTER
OUT DATREG

LDA EREG        ;WRITE E - REGISTER
OUT DATREG

LDA FREG        ;WRITE F - REGISTER
OUT DATREG

RET
```

3.5.4 Reading/Writing Bubble Data

After the parametric registers have all been properly written, the user starts the data transfer simply by sending a Read Bubble Data or Write Bubble Data command to the CMDR.

Once the data transfer has begun, the user must keep up, as explained in the discussion of the FIFO data buffer in Section 3.2.2, in order to avoid a timing error.

In the case of a read operation, the set of possible errors is larger than in a write operation. This is because of the error detection and correction circuitry, which picks up the errors during the read operation, and also because most data errors do, in fact occur during a data read operation. More details on the error detection and correction process are given in Chapter 4 and in the appendices.

Data errors may result in interrupts to the user's system, according to the setting of bits in the enable register.

3.5.5 Interrupts

There are three general circumstances that can result in interrupts to the user's system:

1. Data request (DRQ)
2. Completion of a command execution (INT)
3. Error Conditions (INT)

The data request interrupts function as described in the section on data transfer modes.

The interrupt upon completion of command execution occurs if the INTERRUPT ENABLE (NORMAL) bit is set in the enable register.

Interrupts are cleared whenever a new command is issued or when the register address counter (RAC) is written with the modifier bit (bit D5) equal to one and bits D3 through D0 equal to zero. Interrupts are not cleared by reading status.

Error interrupts are reported according to the settings of three bits in the enable register, as explained in Section 3.2.2.

3.6 Shut-down Procedures

3.6.1 Normal

A normal power down procedure is one in which the user voluntarily powers down the bubble memory system when the BMC is not busy. No special steps need to be taken if the two resistors and capacitors mentioned in Section 3.4.1 are included.

3.6.2 Power Fail

The 7230 Current Pulse Generator contains a special power fail detection circuit. The purpose of this circuit is to detect when the power supplies fall below threshold levels. Such an event automatically initiates an orderly shutdown of the MBMs, in such a manner that no MBM data will be lost or invalidated, even if the power failure occurs during a command execution.

Details on the operation of the overall power fail protection circuitry are given in Section 4.4.

3.7 Diagnostic Procedures

The bubble memory system contains a number of built-in features designed to detect and correct errors before data is sent to the user. The most common kind of data error is a 'soft' error, which occurs during a read operation because of noise in the sense circuitry. The error detection circuitry in the FSAs will (within the limits of the Fire code) pick up this kind of error, correct it (if error correction is enabled) and (when error interrupts are enabled) notify the user. Uncorrectable errors have a very low probability, as explained in Section 4.5. If the error is not correctable, the block of data containing the error can be reread, and, in most cases, the error will not reoccur. If the error turns out to be a 'hard' error (error in the MBM data itself) it may be necessary to rewrite that block of data into the MBM.

Sometimes an error condition occurs because of troubles related to the boot loop information. An error could occur during the initialization process, when the boot loop is being read, decoded, and written into the FSA's boot loop registers. Whatever the cause, boot loop-related problems can be tracked down and remedied through the use of the five boot loop-related commands.

3.8 Error Correction Options

To understand the BMC's error correction options, the types of data errors that could occur within the system must be defined. In respect to the MBM, data errors are considered either permanent ("hard") or transient ("soft"). A hard error is an error that is repeatable (i.e., the error occurs each time the page is read), and a soft error is an error that usually does not recur when the page is reread. In respect to the FSA, errors are either correctable (the FSA is able to reconstruct the data using an error correction algorithm before transferring the data to the BMC) or uncorrectable, irrespective of the error's permanency.

The use of the bubble system's error correction capabilities depends on the host system's requirements relative to error logging and data recovery. Error logging, the process through which the host system compiles a list of bubble memory pages in

which errors occur, is used to isolate permanent errors or other repetitive problems. Data recovery depends on both the nature of the error and the error correction option selected within the BMC.

A typical bubble memory system runs its entire useful life without ever producing an uncorrectable error (the probability of an uncorrectable error is approximately one in 10^{14} bits read). Accordingly, in many applications, a user may not elect to provide error logging or data recovery capabilities in system software; other users may elect to include some level of error correction as a test of system integrity.

To accommodate the different host system requirements, the bubble memory system provides several levels of error correction; the desired level is selected by setting the appropriate bit or bits in the BMC's enable register (ER). Table 3-3 defines the relevant enable register bits for the various levels of error correction.

Table 3-3. Error Correction Levels

Error Correction Level	Enable Register Bit		
	Bit 6 ICD	Bit 5 RCD	Bit 1 Interrupt Enable (Error)
Level 1	0	1	0
Level 2	1	0	0
Level 3	1	0	1

3.8.1 Level 1

Level 1 is the minimum level of error correction and is used when the host system is not concerned with correctable errors or error logging. During level 1 operation, if a read error is detected, the BMC automatically issues a read corrected data (RCD) command to the FSA. The FSA, in response to the RCD command, cycles the data through its error correction circuitry, updates its status register and transfers the "corrected" data to the BMC. The BMC reads the FSA's status register and updates its status register. If the read error was correctable, valid data would have been transferred to the BMC, the BMC would increment the address register to the next page to allow the operation to continue, and an interrupt would not be generated (i.e., the entire operation would be transparent to the host system). However, if the FSA is unable to correct the erroneous data (i.e., if the data is uncorrectable), invalid data would have been transferred to the BMC. The BMC, after reading the "uncorrectable error" status from the FSA, stops command execution and interrupts the host system, but does not increment the address register. To recover from an uncorrectable error, the host system may retry the operation. However, since the erroneous page of data was transferred to the host, it is recommended that the host system reload the parametric registers and repeat the entire read operation in order to reread the erroneous page.

3.8.2 Level 2

Level 2 differs from level 1 in that page-specific logging of uncorrectable errors is possible and the transfer of erroneous data to the BMC is prevented. As with level 1, correctable errors are transparent to the host system.

During level 2 operation, when a read error is detected, the BMC automatically issues an internally correct data (ICD) command to the FSA. In response to this command, the FSA cycles the data through its error correction circuitry and updates its status register (correctable or uncorrectable error status), but does not transfer

the data to the BMC. The BMC reads the FSA's status register and updates its own status register. If the error is correctable, the BMC automatically issues an RCD command to the FSA (to transfer the corrected data) and increments the address register, and the read operation continues with the next page. If the data is uncorrectable, command execution is stopped, and the BMC interrupts the host system. When interrupted, the host system can read the address register to determine the address of the page containing the error and can issue a subsequent read command (without reloading the parametric registers) to retry the page. The subsequent read command causes the page to be reread and, if the uncorrectable error does not recur (i.e., if the uncorrectable error is a "soft" error), command execution continues with the next page. If successive retrys are not successful (hard uncorrectable error), the host system can examine the erroneous page by issuing an RCD command to the BMC to transfer the uncorrectable data from the FSA; the BMC will interrupt the host system due to the uncorrectable error and will not increment the address register.

3.8.3 Level 3

Level 3 is the most complete form of error handling. This level causes an interrupt to be generated to the host system whenever either a correctable or uncorrectable error is encountered. Error logging may be performed on correctable as well as uncorrectable errors, and an unlimited number of retries may be performed on an erroneous page. Similar to Level 2, the transfer of erroneous data to the BMC can be prevented.

During level 3 operation, when a read error is detected, the BMC issues an ICD command to the FSA. The FSA cycles the data through its error correction circuitry and updates its status register, but does not transfer the data to the BMC. The BMC, in turn, reads the FSA's status register, updates its own status register and interrupts the host system. When interrupted, the host system must examine the BMC's status register to determine if the error is correctable or uncorrectable and may log the address of the page in error by reading the address register. If the error is correctable, the host system can either issue an RCD command to transfer the corrected data from the FSA to the BMC or can retry the page by issuing subsequent read commands. If the error is uncorrectable, the host system would only retry the page by issuing subsequent read commands. Note that it is possible for the host system to issue an RCD command in response to an uncorrectable error; the operation would be accepted, but the data transferred for the page in error would be the uncorrectable page of data.

Since command execution is stopped for correctable errors, the host system can perform bit-by-bit comparisons of the corrected data with the data read from the same page without error correction for diagnostic analysis (reinitialization of the bubble system to the non-error correction mode would be required).

Over a period of time, if a consistent pattern emerges, it may then be possible to identify a storage loop responsible for the error, and the boot loop could be rewritten to remove the defective loop. (The probability of a defective loop is extremely low.)

3.8.4 Status Register

When using error correction, the host system can read the BMC's status register (STR) at the time of the interrupt or at the completion of command execution to ascertain additional information relating to the error condition. The relevant bits of the STR are bits 6, 5, 3 and 2; the remaining bits are irrelevant to error correction.

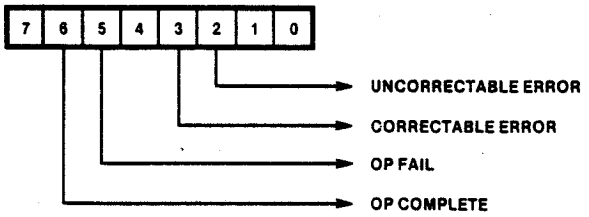


Table 3-4 lists the possible status indications that could occur when using error correction.

Table 3-4. Error Correction Status Indications

STR Bit Set	Error Correction Level	Indication
3	3	Correctable error detected, address of page containing the error is in the AR.
5 and 2	All	Uncorrectable error detected, address of page containing the error is in the AR.
6	All	Operation complete, no errors detected.
6 and 3	1 and 2	Operation complete, one or more correctable errors detected and corrected.
6 and 2	1	Operation complete, uncorrectable error detected on last page of multipage transfer.
5, 3 and 2*	1 and 2**	Very Rare. During a multipage transfer, one or more correctable errors detected and corrected on one or more pages, and a subsequent uncorrectable error detected on a page other than the last page of the specified transfer.
6, 3 and 2*	1	Extremely rare. During a multipage transfer, one or more correctable errors detected and corrected on one or more pages, and a subsequent uncorrectable error detected on the last page of the specified transfer.

*These status indications may occur on single-page transfers in Multi-MBM systems when one MBM has a correctable error and another MBM has an uncorrectable error.

**This status indication may occur in level 3 in Multi-MBM systems when more than one MBM has an error on the same page and at least one of the errors is correctable and one of the errors is uncorrectable.

CHAPTER 4
PRINCIPLES OF OPERATION

4.1 Introduction

This chapter contains valuable background information on the functional characteristics of the various components of the bubble memory system. The IC components used in the bubble memory systems have been designed with transparency in mind—that is, a maximum number of operations are handled by the hardware and firmware of these components. Chapter 3 of this manual has focused on the operations performed by the user of the bubble memory system. Chapter 4 examines the internal workings of the bubble memory system.

Each 1-Megabit Bubble Storage Unit (BSU) operates in its own domain, and is unaffected by the number of BSUs in the system. Thus the roles played by the MBM's immediate support circuitry (FSA, CPG, CPD, and coil drivers) can be described as if the system contained only one MBM module.

The connections between the FSAs and the controller, and the operation of the controller itself are somewhat system configuration dependent. The way that this affects the data flow within the bubble memory system is discussed in Section 4.2. The effect on controller operation has already been discussed at some length in Chapter 3.

Throughout the discussions of this chapter it is helpful for the reader to refer to the schematic diagram for the IMB-72 board (figure 2-2).

4.2 Data Flow Within the Bubble Memory System

4.2.1 One-MBM System

During a read operation, data flows as follows. Bubbles from corresponding positions in each of the circular data storage loops (in the MBM) are replicated onto an output track. They then move (under the influence of a rotating magnetic field) to a detector. The detector outputs a differential voltage according to whether a bubble is present or absent in the detector at any given time. This voltage is fed to the detector input of the Formatter/Sense Amplifier (FSA). Data from each channel (A channel or B channel) of the MBM goes to the corresponding channel of the FSA. In the FSA, the sense amplifier performs a sample-and-hold function on the detector input data, and produces a digital 0 or 1. The resulting data bit is then paired up with the corresponding bit in the FSA's boot loop register to determine whether it represents data from a 'good' loop. If it does, the data bit is stored in the FSA FIFO. Error detection and correction (if enabled by the user) is applied to each block of 256 data bits. If correction is not enabled, 272 bits per FSA FIFO are read directly from the MBM.

From the FSA FIFO, data is sent to the bubble memory controller (BMC) in the form of a serial bit stream, via a one-line bidirectional data bus (DIO). The data is multiplexed onto the DIO line, with data bits coming alternately from the A and B channels of the FSA. The BMC outputs a SYNC/ pulse to the SELECT.IN/ input of the FSA. The FSA responds by placing a data bit from the A channel FIFO on the DIO line. One clock cycle later, a data bit from the B channel FIFO is placed on the DIO line. The BMC continues to output SYNC/ pulses, once every 20 clock cycles, each time receiving two data bits in return.

In the BMC, the data undergoes serial-to-parallel conversion, and is assembled into bytes, which are then placed in the BMC FIFO, which can hold 40 bytes of data (not counting its input and output latches). From this FIFO, the bytes of data are written onto the user interface.

During a write operation, the data flow consists of the corresponding complementary operations in the reverse order. Data bytes from the user interface are placed in the BMC FIFO. The BMC then performs parallel-to-serial conversion on the FIFO output data and places it on the DIO line.

The BMC outputs a SYNC/ pulse, followed one clock cycle later by the first data bit, followed one cycle later by the second data bit. The FSA receives the SYNC/ pulse on its SELECT.IN/ input line, and responds by inputting the first data bit to its A channel FIFO and the second data bit to its B channel FIFO. Data transfer continues, with the BMC outputting a SYNC/ pulse (and two data bits) every 20 clock cycles.

The data is loaded into each FIFO (if error detection and correction are enabled by the user, 14 ECC code bits are appended to the end of each 256-bit block of data). The data bits are taken out of the FSA FIFO according to the contents of the corresponding boot loop register. If the boot loop register shows a one in the position corresponding to the upcoming data bit, that data bit is inverted and placed on the DATA.OUT/ line to the current pulse generator (CPG). If the boot loop register shows a zero in the position, the FSA inserts a zero into the serial bit stream by disabling DATA.OUT/. In this manner, it is assured that the data is written only into good loops in the MBM, and that zeroes are written into all bad loops.

The operations described in the above paragraph occur simultaneously in each FSA channel. For simplicity we shall follow the data in the A channel. The DATA.OUT.A/ signal from the FSA goes to the GEN.EN.A/ input to the CPG. This input (when carrying a data bit equal to one) enables the CPG to output the proper pulses on the GEN.A output pin to generate a new bubble in the A channel of the MBM. The timing of these pulses is determined by the BMC, and depends on whether the bubble is to be generated in the odd or even quad of the A channel of the MBM. The bubble is generated by replicating a constant existing seed bubble. The bubbles which are generated in this manner, are moved along the input track until they are positioned next to the corresponding circular data storage loops. The bubbles are moved by a rotating magnetic field, which runs at a 50 KHz rate, moving one bubble position in each field cycle. This rate is the same as the SHIFT.CLK/ rate. SHIFT.CLK/ is generated by the controller and is used by the FSA to enable the DATA.OUT.A/ signal.

When the bubbles on the input track have been positioned opposite their respective desired storage positions on the circular loops, a SWAP pulse is sent to the MBM by the CPG. This causes the new data on the input track to be exchanged for whatever old data may be present in the adjacent bubble positions on the circular storage loops. This completes the write operation for those bits. The swapped out bits cause no difficulty; they are later propagated into a guard rail which functions as a bubble annihilator.

The above description gives an overview. More detail on the individual operations can be found in the functional descriptions of each component (Section 4.3).

4.2.2 Multiple-MBM Systems

The data flow in a multiple-MBM system is in most respects similar to that which occurs in a one-MBM system. The difference is in the time-division multiplexing that occurs on the DIO bus line between the BMC and the FSAs.

For normal data transfer operations, the BMC may exchange data with as few as two FSA channels (one MBM) or as many as 16 FSA channels (8 MBMs). The first step in setting up a data transfer is to select which FSA channels are to be involved. The user makes this selection by writing the appropriate high-order bits of the BLR and AR registers (see table 3-2). The BMC conveys the selection information to the FSAs by outputting the proper signals on the C/D/, SYNC/, and DIO lines.

The SYNC/ output of the BMC is connected to the SELECT.IN/ input of the first FSA. The SELECT.OUT/ output of each FSA is connected to the SELECT.IN/ input of the next FSA. Thus a daisy chain is created. Each FSA propagates the SELECT.IN/ signal to the SELECT.OUT/ pin, but with a delay of two clock cycles. Thus the SYNC/ signal from the BMC moves through the entire set of FSA channels, advancing one FSA channel per clock cycle.

To select the FSAs for data transfer, the BMC raises C/D/, indicating that a command is coming, and simultaneously outputs a SYNC/ pulse (the SYNC/ pulse is propagated down the FSA chain, in shift register fashion, as described above). The BMC then outputs a serial bit stream on the DIO line, beginning in the clock period following the SYNC/ pulse. Each bit position in this bit stream corresponds to a particular FSA channel. Each FSA monitors the DIO bus and responds to what is on the bus at the clock period following its receipt of the propagated SYNC/ pulse. In this manner, by suitably arranging the configuration of bits, the BMC is able to alert the chosen FSAs that the command about to be issued is for them.

Twenty clock periods after the first SYNC/ pulse, the BMC drops C/D/ and sends a four-bit command on the DIO bus line. In this case, the command would be to read or write data from or to the MBMs. In the case of a read operation, the BMC must wait until there is data available in the FSA FIFO. In the case of a write operation, the BMC may send a second SYNC/ pulse (indicating data about to be sent) as soon as 40 clock pulses after the first SYNC/.

Whether the data is being read from the FSA FIFOs or written into them, the pattern of accumulation or distribution is that of time-division multiplexing among the selected FSA channels. Each FSA outputs or inputs data to or from the DIO bus line during its characteristic time slot in relation to the previous SYNC/ pulse. The overall data transfer rate is proportional to the number of FSA channels selected.

The above description gives an overview. More detail can be found in the functional description of the FSA (Section 4.3.3).

4.3 Component Functional Descriptions

The basic components of the bubble memory system are described briefly in Section 1.2. The overall layout of a bubble memory system can be seen by referring to figures 1-2 and 1-3, or to the schematic diagrams in figures 2-2 and 2-3. The following sections concentrate on the internal descriptions of each functional component.

4.3.1 7110 1-Megabit Magnetic Bubble Memory (MBM)

The Intel Magnetics 7110 MBM is a very high density 1-megabit non-volatile solid state memory using the magnetic bubble technology. One megabit in this context means that the memory will store 1,048,576 data bits, along with their ECC bits, if error correction is used. In describing this MBM, it is useful for the reader to refer to figure 4-1.

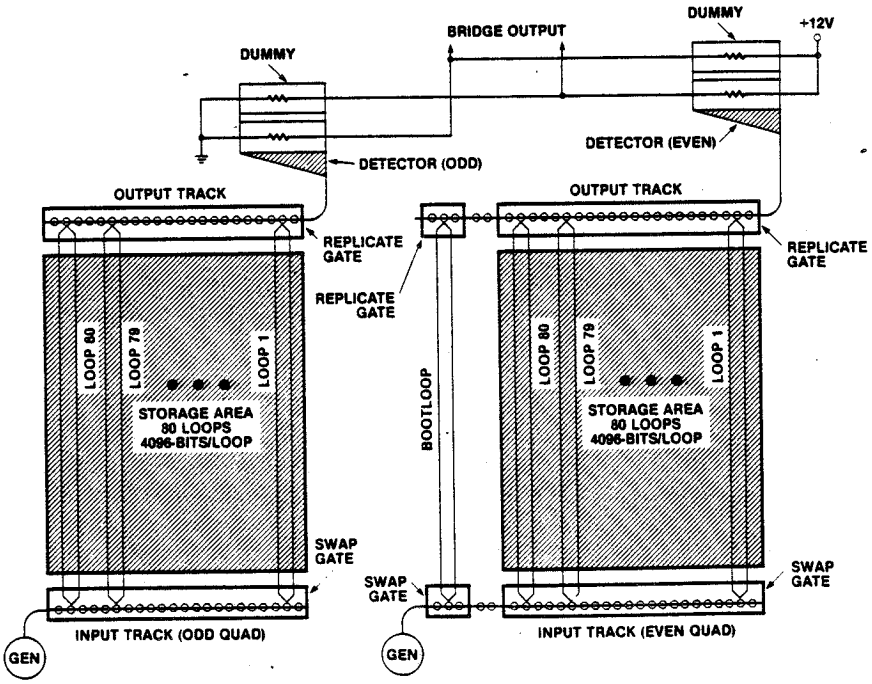


Figure 4-1. 7110 Bubble Memory Architecture (One Half) 121685-12

Figure 4-1 shows the architecture for one half of the 1-megabit module. The two halves are identical. Each half operates in connection with one channel of the FSA. The architecture is major track-minor loop, which means simply that the bubbles propagate into and out of the memory on tracks and are stored in circular loops. The 1-megabit storage capacity is provided by 256 loops of 4096 data storage positions each. An additional 14 loops are provided for an ECC code, making a total of 270 loops, or 135 loops in each half.

Referring to figure 4-1, we see that each half module contains 160 storage loops—25 more than required. This redundancy factor is provided so that the design will be tolerant of defects in some loops. This greatly increases the yield of bubble memory modules during manufacture, and consequently decreases the cost. Up to 25 storage loops can be bad (contain defects) and still leave the required 135 loops per half module for the data bits plus the ECC code. In actual practice, screening is done to a maximum of 24 defective loops. This enables the device to provide 16 bits for a utility field for users who do not elect to use the FSA's error correction facilities.

The 160 loops in each half module are organized as two quads of 80 loops each. Each quad contains an 81st loop, called the boot loop. During manufacture, the boot loops are tested, and one of the two boot loops, which is known to be good, is bonded (hooked up) by connecting its boot swap and boot replicate gates to the corresponding pins of the MBM package.

The purpose of the boot loop is to provide a map showing which storage loops are good and which are bad. The boot loop also contains information which is used in positioning the circular loops (providing a physical address reference). The two functioning boot loops (one in each half module) are connected in parallel, in the sense that their boot swap and boot replicate gates connect to a single set of pins on the MBM package. Thus they are designed to be read or written simultaneously. In actual practice, both boot loops are written with the same information, which contains the loop map information for the entire module. This provides a redundancy factor which effectively increases the integrity of the boot loop information.

The boot loops are written at the factory, and typically do not require rewriting during the entire useful life of the module. However, for additional protection, the boot loop map is also written on a label attached to the MBM. Section 5.4 explains how to rewrite the boot loops. Because the boot loops are written with identical information, it is common to speak of "the boot loop" in reference to these two boot loops.

The data flow into and out of the storage area is as follows. Referring to figure 4-1, bubbles are generated at the GEN elements. A bubble is generated by replicating a seed bubble, whenever the corresponding data bit is a one. The bubbles propagate along the input track until they are positioned opposite the desired storage locations in the circular loops (the bubbles are moved along the major tracks by the same rotating magnetic field that moves the bubbles in the circular loops). When the bubbles have reached the desired positions, a SWAP pulse is sent to the swap gate. This causes the set of bubbles on the input track to be exchanged bit-for-bit with the bubbles already residing at the adjacent storage locations in the circular loops.

To read this same data out of the MBM, it is first necessary to move the bubbles along the circular loops until they are opposite the output track. Then a REP pulse is sent to the replicate gate. This causes the bubbles to be duplicated onto the output track. The stored data in the circular loops is not affected. Next, the bubbles propagate along the output track until they reach the detector element, which produces an output signal indicating the presence or absence of a bubble.

The above two paragraphs provide a general picture of the data flow within the MBM. We shall now fill in some details of the operation at each stage.

The GEN elements are shown as two separate elements in figure 4-1, yet the MBM provides only one GENERATE input for each channel (half module). The resolution of this seeming contradiction is that the two GEN elements are connected in series, but are operable at two different times in the field cycle. Depending on which timing is used, one GEN element or the other will be effective in producing a bubble. In practice the GEN elements in the odd and even quads are activated during alternate field rotation cycles. The bubbles move along the input track one position for each field cycle. Thus, on a given input track, every second position is a potential bubble location (a data one is indicated by the presence of a bubble, while the absence of a bubble indicates a data zero).

A bubble is generated by replicating a seed bubble which is always present at the GEN element. Thus the generator is basically a replicate gate. This method of bubble generation is more reliable over a wide temperature range than the more conventional nucleating type of bubble generation. The seed bubble is created at the factory in each generator, and typically requires no maintenance. However, if the seed bubble is ever lost, the user can regenerate it by using a special 'seed module' (supplied as part of the BPK 72 kit). Instructions on the use of this seed module are given in Section 5.3.

Referring once more to figure 4-1, it can be seen that the minor loops are spaced four bubble positions apart on the input track. Yet we have mentioned above that the data is spaced two positions apart. Thus in order to get all the data into the minor loops it is necessary to use two SWAP pulses. The first swap pulse transfers half the data, and the second SWAP pulse, occurring two field cycles later, transfers the remaining data from the input track to the minor loops.

One bubble is generated each field cycle, yet the bubbles in the odd and even quads are transferred into the minor loops by the same two SWAP pulses. This is made possible by constructing the input track for the odd quad to be one bubble position longer than the input track for the even quad. Specifically, the even quad has 21 bubble positions between the GEN element and the boot loop location, while the odd quad has 22.

As mentioned earlier, the swap gate serves a dual function. The data in the adjacent (to the input track) locations of the minor loops is transferred to the input track at the same time that the data from the input track is transferred into the minor loops. This effectively erases the old data from the minor loops before the new data is transferred in. The old data then exists on the input track, where it exists harmlessly until it is later shifted to the end of the input track into a guard rail, which functions as a bubble annihilator.

The fact that two SWAP pulses are used means that the data is stored in two bit positions on each minor loop. This effectively doubles the MBM page size. It means that each half module can accept 256 data bits (plus the ECC bits or the utility field), and that the module as a whole can accept 512 data bits (per pair of SWAP pulses). It also means that, for each page to be written, 256 data bits pass through each FSA channel and thus (if error detection and correction is enabled) the 14 ECC bits associated with those 256 data bits can all be stored in the same channel (half module) of the MBM.

The above discussion explains why it is said that the 7110 magnetic bubble memory has a binary page organization of 512 bits/page and 2048 pages, even though the module itself has 320 minor loops of 4096 bubble locations each.

The relationship between loops and tracks is quite similar for data output (read) operations. The minor loops effectively intersect the output track every four bubble positions. The bubbles are replicated onto the output track by a REP pulse applied to the block replicate gates. This causes a non-destructive readout of the bubble locations adjacent to the output track. One data bit (presence or absence of bubble) from each loop is transferred to the corresponding position on the output track. There is no differentiation between good and bad loops at this time.

Two field cycles later, a second REP pulse replicates the data from the minor loops (which have rotated two positions) onto the output track, in such a manner that the second set of data fall midway between the positions of the first set on the output track. This effectively reproduces the original configuration of data bits as it was written on the input track for that physical page.

The block replicate gates for the odd and even quads operate simultaneously. After replication, the bubbles propagate along the output tracks to the detectors. The output track for the even quad is one bubble position longer than the output track for the odd quad. Specifically, the odd quad has 40 bubble positions between the end of the block replicate gate and the detector, while the even quad has 41. This assures that the bubbles will arrive at the respective detectors in an interleaved fashion, and that the detector output represents the same sequence of bubbles originally created at the GEN elements.

The detector elements operate on a magneto-resistance principle. The cylindrical bubble domains are stretched into long strip domains by a chevron expander, and are then propagated to the active portion of the detector. The detector consists of a stack of interconnected chevrons, through which a current is passed. As the strip domain propagates through the stack, its magnetic flux causes a change in stack resistance, which produces an output signal on the order of a few millivolts. Following detection, the strip domain is propagated into a bubble guard rail. A dummy detector stack is located in the immediate vicinity of the active detector. The active and dummy detectors are connected in a bridge configuration, such that common mode pickup (originating predominantly from the in-plane drive field) is greatly attenuated at the bridge output.

The BMC keeps track of the physical position of the bubbles within the minor loops, and also implements the proper mapping function between logical (system) page addresses and physical page addresses (relative positions on the MBM storage loops). Figure 4-2 shows some of the relationships between the physical addresses and the logical addresses. Note that each page address (whether physical or logical) represents two locations on the loop.

The boot loop contains a reference field (242 zeroes followed by a 14-bit SYNC pattern) which is used to locate the 'zero' physical page. Before data can be read from or written to the MBM, the user must issue an Initialize command. This causes the boot loop information to be read from the boot loop and stored in the two boot loop registers of the FSA. It also causes the minor loops of the MBM to be rotated to a standard position, which is defined as page zero (with respect to the output track). A counter in the BMC then keeps track of the current logical page address by adding a certain constant 'N' to the previous logical address each time the first bit of a new page is rotated to the block replicate gate. The value of this constant is determined by the considerations used in choosing the mapping of physical to logical page addresses. Two criteria must be met:

1. The mapping must be one-to-one. This requires that the constant N be relatively prime with respect to 2048 (satisfied by any odd number).

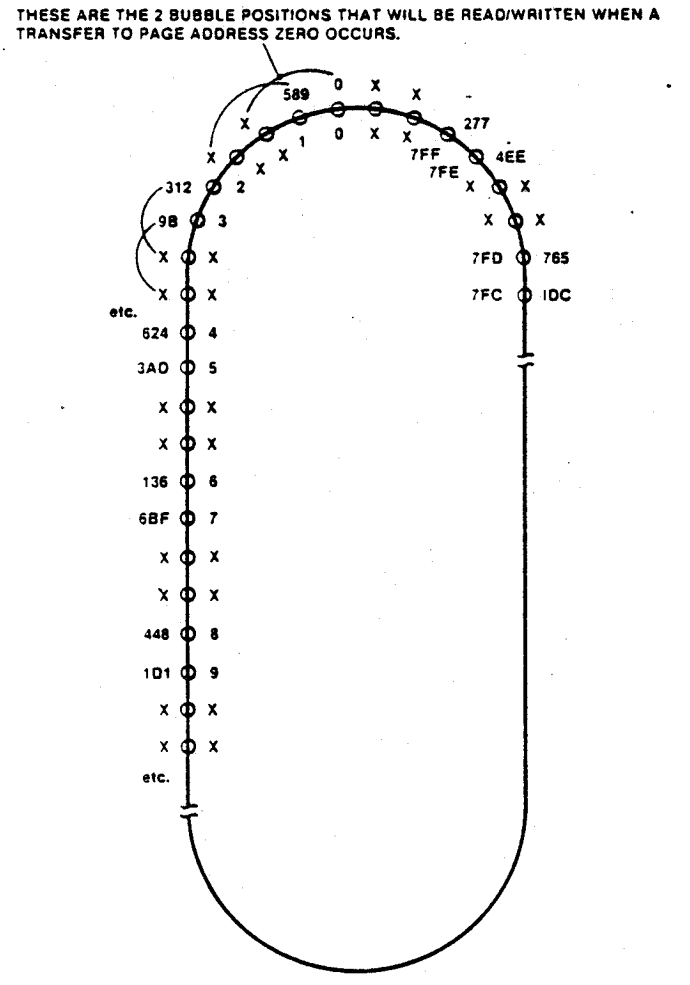


Figure 4-2. 7110 Storage Loop (Showing Logical and Physical Addresses)

2. The mapping must be efficient with respect to access time for multiple page data transfers. For example, during a read operation, 357 field rotation cycles are required after the second replicate pulse before the last bit on that page (from loop 80 in the even quad) reaches the detector. It is desirable to wait until the last bit has been detected before replicating the next page. For efficient operation, then, the physical spacing (on the minor loops) between consecutive logical addresses must be greater than 359 bubble positions, but not much greater, lest the access times should suffer.

In practice, the spacing is 369 bubble positions between an even logical page address and the next (odd) address, and 371 bubble positions between an odd logical page address and the next (even) address.

The controller locates a desired page (for reading data) by rotating the minor loops and adding the constant N (to the counter described above) until a match occurs with the logical address in the user-accessible register AR (representing the next desired logical page address). Then the rotation stops, and (the first half of) the desired data is opposite the block replicate gate.

In the case of a write operation, the process is quite similar, except that the match must occur at the appropriate time for the first bubble of the new page to be generated. The bits are then written onto the input tracks and propagated into position for swapping. The desired physical page position on the minor loops reaches the swap gate just as the bubbles reach the corresponding positions on the input track. This condition is fulfilled simply by adding a special 'write constant' to the current address in the counter before comparing it with the contents of AR.

4.3.2 7220 Bubble Memory Controller (BMC)

The Intel 7220 BMC is designed to provide all the control and timing functions needed to operate Intel 7110 MBMs and their immediate support circuitry. The 7220 provides a MULTIBUS compatible user interface, and also provides suitable interfaces to the 7242 FSA, the 7230 CPG, and the 7250 CPD.

Figure 4-3 is a block diagram of the 7220 BMC. Four of the blocks shown—the SYSTEM BUS INTERFACE, the user-accessible REGISTER FILE, the FIFO, and the DMA AND INTERRUPT LOGIC—have already been discussed at length in Chapter 3. The remaining blocks are discussed below.

The MBM ADDRESSING LOGIC AND RAM block contains two more user- accessible registers—the BLR and the AR, plus an adder and the MBM address RAM. The MBM address RAM stores the next available logical page (read) address for each MBM. This is the address that is used to keep track of the position of the MBM's storage loops, and to implement the logical-to-physical address mapping function, as described at the end of Section 4.3.1.

The DIO BOOT LOOP DECODER/ENCODER block performs parallel-to- serial and serial-to-parallel conversions between the FIFO data and the serial bit stream on the DIO line. This block also generates the BUS.RD/ signal, which indicates the direction of data transfer on the DIO line (this is useful in situations which require external buffering on the DIO line). This block also contains the circuitry which decodes the boot loop data during a Read Boot loop (or Initialize) operation, and encodes the boot loop data during a Write Boot loop operation (including the 'zero field' and 14-bit 'sync' code).

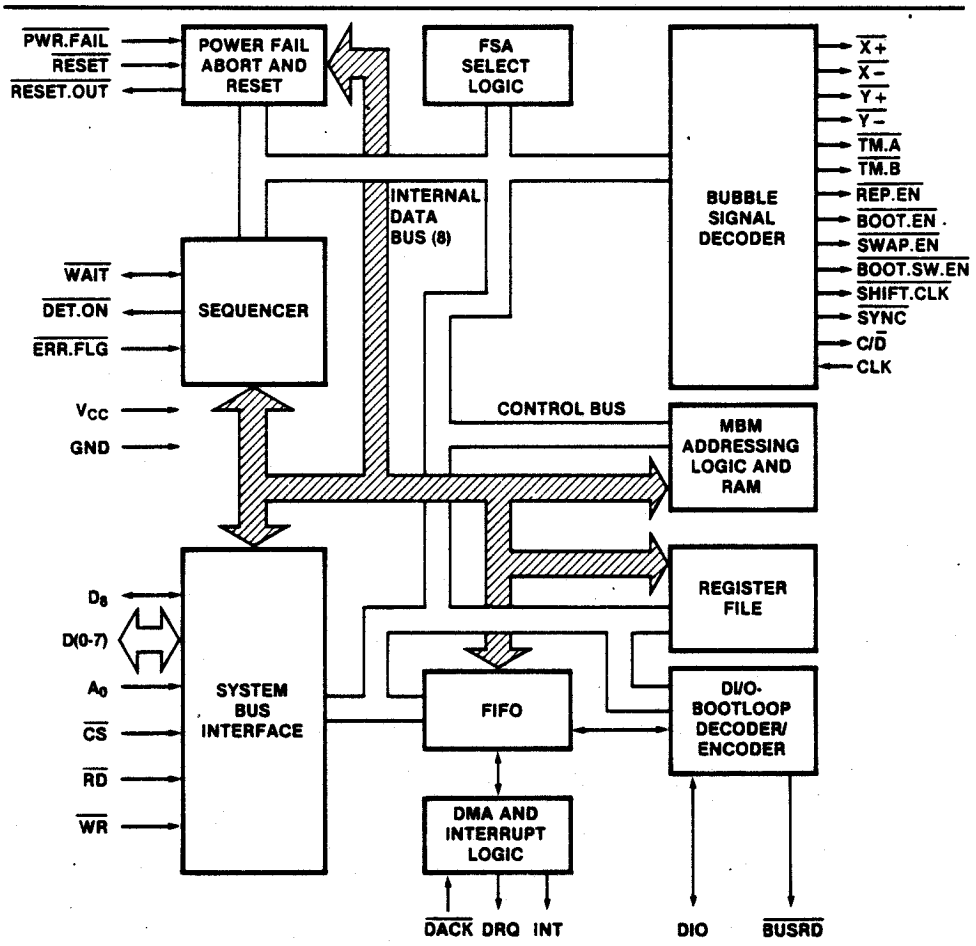


Figure 4-3. 7220 BMC Block Diagram 121685-14

The SEQUENCER block controls the execution of commands by decoding the contents of its own internal ROM in which the BMC firmware is located. This block also sets and resets flags and status bits, and controls actions in other parts of the BMC.

The POWER FAIL AND RESET block provides a means of resetting the bubble system in an orderly manner, when activated by the PWR.FAIL/ signal, the RESET/ signal, or the Abort command.

The FSA SELECT LOGIC block contains the logic which controls the timing of the interaction between the BMC and the FSAs. This is explained partly in Section 4.2.2 (describing data flow within a multiple-MBM system) and partly in Section 4.3.3 (dealing with the FSA). The FSA selection is determined by the four high-order bits in the BLR and the four high-order bits in the AR, both set by the user (as described in Section 3.2.2).

The BUBBLE SIGNAL DECODER block contains the logic for creating all the MBM timing signals. This block contains a three-stage counter, a decoder, and synchronous latches. The first stage of the counter is the divide-by-four counter that is always enabled. The second stage is a divide-by-twenty counter which produces the basic field rotation frequency from the output of the first stage. Thus, using the required 4 MHz clock input to the BMC, the first stage output is at a 1 MHz rate, and the second stage output is at a 50 KHz rate. Any of the 40 clock edges occurring

within each complete cycle of the divide-by-twenty counter can be used to set or reset the MBM signal latches. These set and reset functions are enabled by signals from the SEQUENCER block.

Figure 4-4 shows the general appearance of the bubble timing signals generated by the BMC. The purpose of these signals (shown as outputs from the BUBBLE SIGNAL DECODER block) is explained briefly below.

X+/, X−/, Y+/, and Y−/ go to the 7250 CPDs, and are used to enable the coil drive currents in the MBMs.

TM.A/ and TM.B/ go to the 7230 CPGs, and are used to determine, respectively, the pulse widths for the CUT and TRANSFER functions used in replicating the bubbles.

SWAP.EN/, REP.EN/, BOOT.SW.EN/, and BOOT.EN/ all go to the 7230 CPG. They are used to enable, respectively, the data swap, data replicate, boot swap, and boot replicate functions within the MBMs.

SHIFT.CLK/ goes to the FSAs. It is used to control the timing of events at the interface between each FSA and its corresponding MBM. More details are given in Section 4.3.3.

SYNC/ and C/D/ control the serial communications between the BMC and the FSAs (on the DIO line). More details are given in Section 4.3.3.

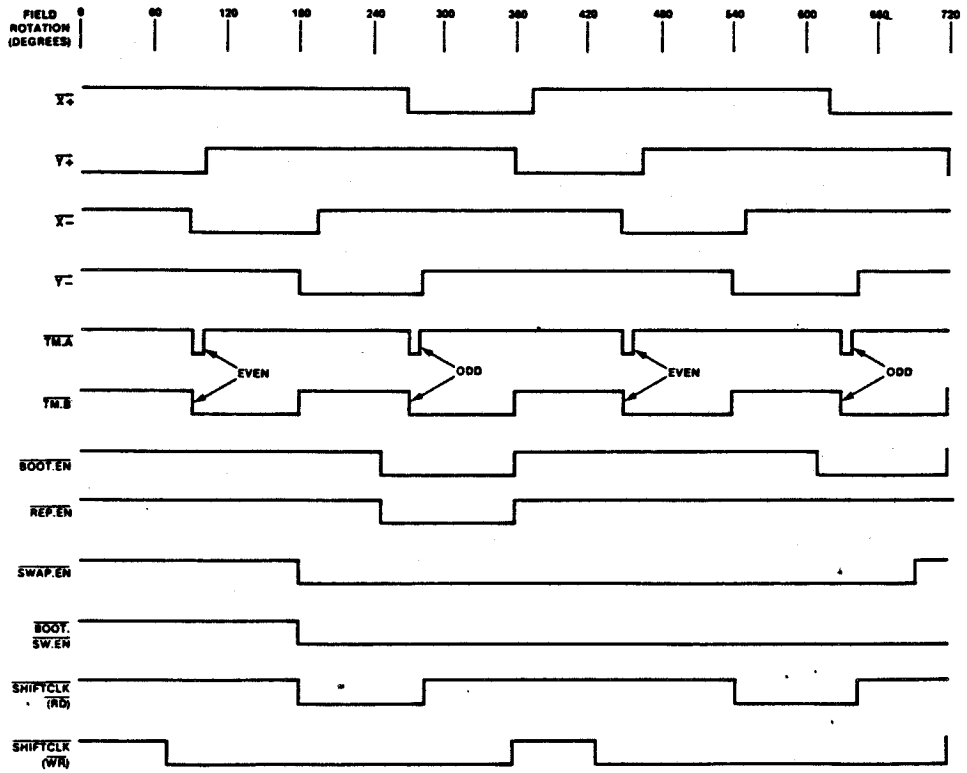


Figure 4-4. 7220 BMC Timing Diagram 121685-15

4.3.3 7242 Formatter/Sense Amplifier (FSA)

The Intel 7242 FSA is a dual formatter/sense amplifier that interfaces directly with the Intel Magnetics 7110 MBM, its coil predriver (7250 CPD) and current pulse generator (7230 CPG), and also with the 7220 bubble memory controller (BMC).

Figure 4-5 is a block diagram of the 7242 FSA. As shown in the figure, the 7242 contains two identical FSA channels. These can be operated independently, but in normal circumstances they are used to control the corresponding two channels (half modules) of the 7110 MBM. The following paragraphs give brief functional descriptions for the blocks, buses, and interface signals of the 7242 FSA.

The SERIAL COMMUNICATIONS block contains the control circuitry for the FSA's interface with the BMC. This block is shared by both channels of the FSA. The specific operations occurring on this interface are described later in this section.

The COMMAND DECODER block interprets the command codes sent to the FSA by the BMC, and generates the appropriate enable and reset signals to operate the rest of the FSA. This block also contains the status registers for both FSA channels.

The INTERNAL DATA BUS carries the information that passes through the FSA between the BMC and the MBM. The information changes format as it passes through the FSA, under the influence of the error correction circuitry (if enabled) and the boot loop register.

The I/O LATCHES and FLAG blocks work in conjunction with the BUS CONTROL block to maintain an orderly data transfer process on the INTERNAL DATA BUS.

The FIFO is a first-in-first-out data buffer for the serial data passing through the FSA. The effective length of the FIFO is 270 bits if error correction is enabled, or 272 bits if error correction is not enabled.

The BOOT LOOP REGISTER is a 160-bit register which contains information detailing the configuration of good and bad loops in the corresponding channel of the MBM. Each bit of the boot loop register corresponds to a specific minor loop in the MBM. As the data passes through the BUBBLE I/O LATCHES the contents of the boot loop register are used (during data read operations) to remove the bits corresponding to bad loops, or (during data write operations) insert zeroes in those bit positions corresponding to bad loops. When error correction is enabled, the boot loop register will contain exactly 135 ones (corresponding to the first 135 good loops in that channel of the MBM). When error correction is not enabled, the boot loop register will contain 136 ones.

The ERROR CORRECTION block implements the error detection and correction process, using a 14-bit Fire code. If error correction has been enabled (by the user), the error correction circuitry appends the 14-bit code to the end of each 256-bit block of data passing through the FIFO during a data write operation. During a data read operation, this circuitry, when enabled, checks the data block and notifies the BMC (via the ERR.FLG/ line) when an error has been detected. More details are given as part of the FSA command descriptions (below).

The BUBBLE I/O LATCHES and FLAG blocks buffer the data passing between the INTERNAL DATA BUS and the BUBBLE INTERFACE.

The BUBBLE INTERFACE block contains a sense amplifier and an output driver. The sense amplifier consists of a chopper-stabilized differential comparator and a sample-and-hold circuit.

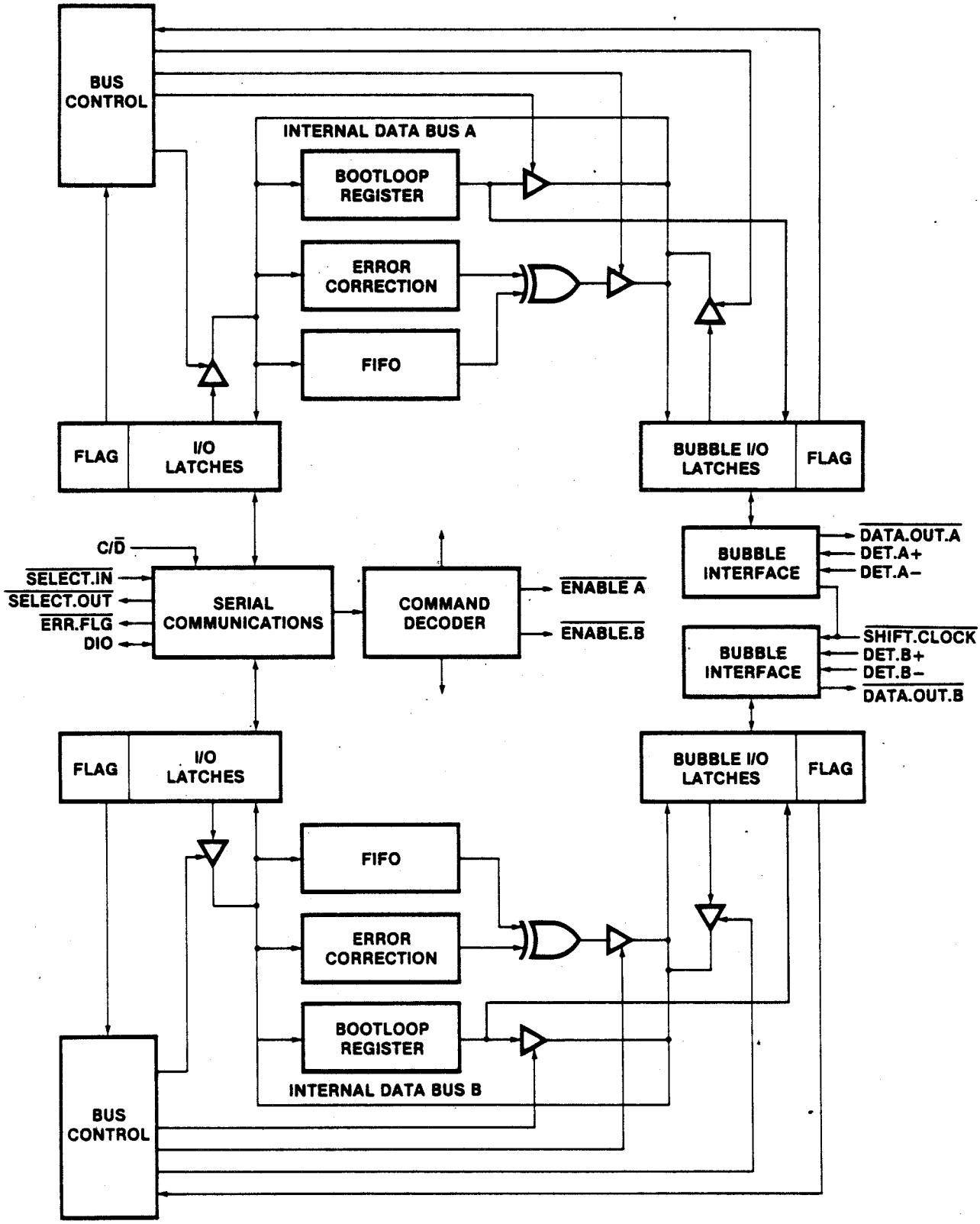


Figure 4-5. 7242 FSA Block Diagram 121685-16

The BMC interface signals consist of the DIO, C/D/, SELECT IN/, SHIFT.CLK/, ERR.FLG/, and RESET/ lines.

The DIO line is a bidirectional bus line that carries the serial data between the BMC and the FSA. It carries command and control information for the FSA, and also carries the data passing between the BMC and the FSA.

C/D/ is the command/data signal. This signal tells the FSA whether to interpret the ensuing bits on the DIO line as an FSA command or as data bits. If C/D/ goes high during a command execution, the current command is aborted.

SELECT IN/ comes from the BMC, or from the previous FSA in the FSA daisy chain. It is used in the FSA selection process, and also to control data multiplexing among the FSAs.

SHIFT.CLK/ is used to clock data out of the bubble interface output latch onto the DATA.OUT.A/ and DATA.OUT.B/ lines during MBM write operations. SHIFT.CLK/ is also used as a timing reference to generate internal gating signals for proper sense amplifier operation and data transfer into the bubble interface input latch during MBM read operations.

The ERR.FLG/ line is an output from the FSA to the BMC, which interrupts the BMC when an error condition is detected in the FSA. It is an open drain signal. The ERR.FLG/ lines from all the FSAs are tied together and connected to the ERR.FLG/ input to the BMC.

RESET/ resets all flags and pointers in the FSA, and also disables the FSA (equivalent to a high on CS/).

CS/ is the chip select signal for the FSA. When CS/ is high, the FSA is deselected and disabled. CS/ can be used as the bank select function in multibank systems. In single-bank systems CS/ is generally tied low.

The CLK input is the same (4 MHz) clock that drives the BMC.

The SELECT.OUT/ signal is connected to the next FSA in the FSA daisy chain, where it functions as the SELECT.IN/ signal for the FSA.

The ENABLE.A/ and ENABLE.B/ signals go to the CPG and CPD and serve as chip select signals for these devices. For systems within the scope of this manual ENABLE.A/ and ENABLE.B/ rise and fall together and may be considered interchangeable. The CPG and CPD are typically deselected with the ENABLE A/ signal.

The bubble interface signals are DET.A+, DET.A−, DET.B+. DET.B−, DATA.OUT.A/, and DATA.OUT.B/.

DET.A+, DET.A−, DET.B+, and DET.B− are the pairs of differential output signals from the detectors in (respectively) the A and B channels of the MBM.

DATA.OUT.A/ and DATA.OUT.B/ go (respectively) to the GEN.EN.A/ and GEN.EN.B/ inputs to the CPG. These signals (when low) enable the current pulses which cause the bubbles to be generated in the MBM during write operations.

The FSA operates under a command and control discipline imposed by the BMC. The BMC sends a four-bit command code via the serial DIO bus line. In the paragraphs to follow, we shall first list the commands executed by the FSA, and then explain how the commands and/or data are transferred over the serial interface between the BMC and the FSAs.

The following is a list of the command codes, and the names of the corresponding commands:

0000	No Operation	1000	Write Boot Loop Register
0001	Reserved	1001	Read Boot Loop Register
0010	Software Reset	1010	Reserved
0011	Initialize	1011	Reserved
0100	Write MBM Data	1100	Set Enable Bit
0101	Read MBM Data	1101	Read ERR.FLG/ Status
0110	Internally Correct Data	1110	Set Correction Enable Bit
0111	Read Corrected Data	1111	Read Status Register

The No Operation command resets the FSA FIFO and boot loop pointers, deactivates the ENABLE.A/ and ENABLE.B/ signals, and prevents further internal activity in the FSA. The FSA does, however, remain responsive to new commands.

The Software Reset command resets the FIFO and boot loop pointers, status flags, and the error correction enable bit. It also deactivates the ENABLE.A/ and ENABLE.B/ lines.

The Initialize command causes the FSA to read the MBM boot loop and pass it through to the BMC. The boot loop register is disabled during this process, so that it does not interfere with the data flow. The error correction logic is also disabled.

The Write MBM Data command causes the data from the BMC to be written into the good loops of the MBM. The SHIFT.CLK/ signal controls the output of bits from the FSA to the CPG. If error correction is enabled, the FSA automatically appends 14 ECC bits to the end of each 256-bit data block.

The Read MBM Data command causes the FSA to read data from the MBM. The data is sensed by the sense amplifiers and screened by the boot loop registers, so that only data from good loops is written into the FSA FIFOs. If error correction is enabled, the data is read in buffered mode—that is, a full block (270 bits) of data is read into the FIFO before any bits are read out of the FIFO. This enables the error correction circuitry to detect any errors and interrupt the BMC. If no error is detected, the 270 bits of data are read from the FIFO while simultaneously the next block of data is read in from the MBM. (The 14-bit error correction code is not placed in the BMC FIFO.) If an error is detected, the BMC may ignore it, send a Read Corrected Data command, or send an Internally Correct Data command (depending on which bits the user has set in the BMC's enable register).

The Internally Correct Data command causes the FSA to internally cycle the data through the error correction network without sending any data to the BMC. This operation takes about 1400 clock cycles. At the end of the operation, the FSA sets the CORRERR or UNCORRERR bit in the FSA status register (depending on whether the error was found to be correctable or uncorrectable). If the error is correctable, the BMC may then issue a Read Corrected Data command.

The Read Corrected Data command cycles the data through the error correction network as it is being read by the BMC. After all 270 bits have been transferred to the BMC, the FSA status register indicates whether the error was found to be correctable or uncorrectable. The Read Corrected Data command is used even when the data has been previously corrected by the Internally Correct Data command (in that case the second correction process does not change the data).

In both the Internally Correct Data command and the Read Corrected Data command, the ERR.FLG/ line remains inactive during command execution, but becomes active again at the completion of the process. This lets the BMC know when it may read the FSA status register.

The Write Boot Loop Register command causes the FSA's boot loop registers to be written with data from the BMC FIFO (typically, the BMC has first read and decoded the MBM boot loop, and has placed the decoded boot loop data into its own FIFO). Under normal conditions the boot loop registers of both FSA channels are written simultaneously, with interleaved data from the BMC.

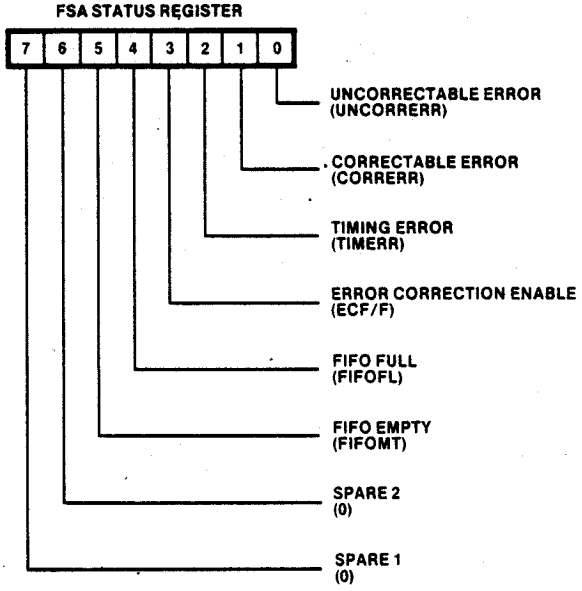
The Read Boot Loop Register command causes the FSA to transfer the contents of its boot loop registers to the BMC, where they are placed in the BMC FIFO.

The Set Enable Bit command resets the FSA FIFO and boot loop pointers, and activates the ENABLE.A/ and ENABLE.B/ lines.

The Read ERR.FLG/ Status command reads the composite error status for each addressed FSA channel. The composite error status is the logic OR of the CORRERR, UNCORRERR, AND TIMERR bits in the status register (the ERR.FLG/ FSA output signal is the logic NOR of the composite error status for the two channels within a particular FSA). When the BMC receives an ERR.FLG/ interrupt, it uses this command to isolate the FSA channel containing the error.

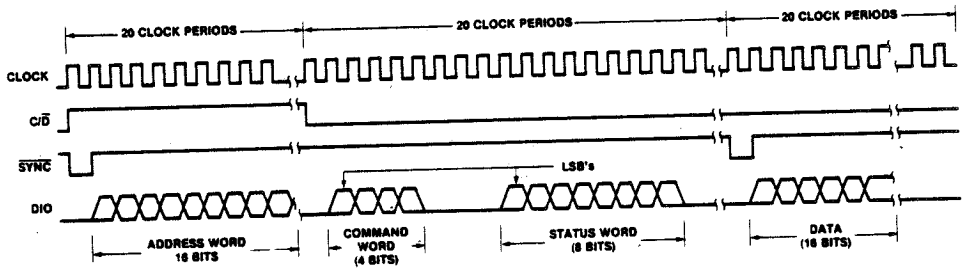
The Set (Error) Correction Enable Bit command enables the error correction circuitry. When this bit is set the FIFO becomes a 270-bit FIFO; when this bit is reset, the FIFO becomes a 272-bit FIFO.

The Read Status Register command causes the 8-bit status word from the addressed FSA channel to be sent to the BMC. The BMC sends this command to only one FSA channel at a time, thus avoiding bus contention on the DIO line. The format of bits in the FSA status register is as follows:



All the above commands are issued to the FSAs by the BMC. The following paragraphs describe the specific sequence of events involved (refer to figure 4-6A).

The BMC raises the C/D/ line, indicating that a command is coming, and simultaneously outputs a SYNC/ pulse. This SYNC/ pulse propagates down the FSA daisy chain in shift register fashion via the SELECT.OUT/ and SELECT.IN/ lines. The BMC outputs a serial bit stream on the DIO line, beginning in the clock period following SYNC/. Each bit in the stream is like an address bit for a particular FSA channel (up to 16 channels). Each FSA, upon receiving SELECT.IN/, looks



NOTE: STATUS INFO IS ONLY PRESENT ON BUS FOR THE READ STATUS COMMAND SEQUENCE.

Figure 4-6A. Command Sequence Timing Diagram 121685-17

for the presence or absence of a logic one on the DIO line during the following clock period. A logic one indicates that the FSA shall accept the coming command (the two channels of a given FSA are alerted independently—the A channel in the clock period immediately following receipt of SELECT.IN/, and the B channel in the clock period immediately after that). SELECT.OUT/ is activated two clock periods after SELECT.IN/ is received.

Twenty clock periods after SYNC/, the BMC drops C/D/ and sends a four-bit command code via the DIO line. Each addressed FSA channel responds to the command.

If the command is Read Status, the addressed FSA channel returns 8 bits of status information, starting four clock periods after the receipt of the last bit of command code. To avoid bus contention, only one FSA channel at a time is addressed.

If the command requires further data transfer, more SYNC/ pulses will be sent by the BMC. These will occur at integral multiples of 20 clock periods, starting no sooner than 40 clock periods after the SYNC/ pulse that preceded the command.

In the case of data transfers between the MBMs and the BMC, the data is time-division-multiplexed (as explained in Section 4.2.2). Figure 4-6B illustrates the timing relationships involved.

At the MBM end of the FSA, the interface consists of the DATA.OUT/ line and a pair of differential input signals. During read operations, the BMC outputs SHIFT.CLK/. This signal causes the FSA to sample the MBM output signal while SHIFT.CLK/ is low and to hold the signal after the trailing edge of SHIFT.CLK/. The data is latched in the bubble input latch and will subsequently be loaded into the FIFO.

SHIFT.CLK/, during write operations, causes a bit of data to be gated out of the FSA via the DATA.OUT/ line. The trailing edge of SHIFT.CLK/ forces DATA.OUT/ high again. During the SHIFT.CLK/ pulse (while SHIFT.CLK/ is low), the BMC outputs the TM.A/ and TM.B/ signals to the 7230 CPG. These signals control the timing of the cut and transfer pulses generated by the CPG (if the data bit is a one) that cause the seed bubble to be replicated by the GEN element in the MBM.

The SYNC/ pulse is synchronous with the beginning of a bubble memory field rotation. There are some system relationships between the timing at the two interfaces (MBM and BMC) of the FSA, as illustrated in figure 4-6C. Data read from the MBM is not available to the BMC until 40 clock cycles after SHIFT.CLK/. Data can not be written into the MBM until 40 clock cycles after SYNC/.

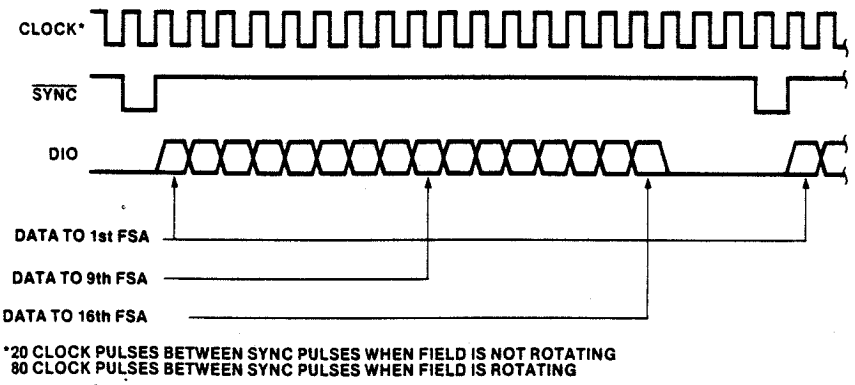


Figure 4-6B. Data Transfer Sequence Timing Diagram 121685-18

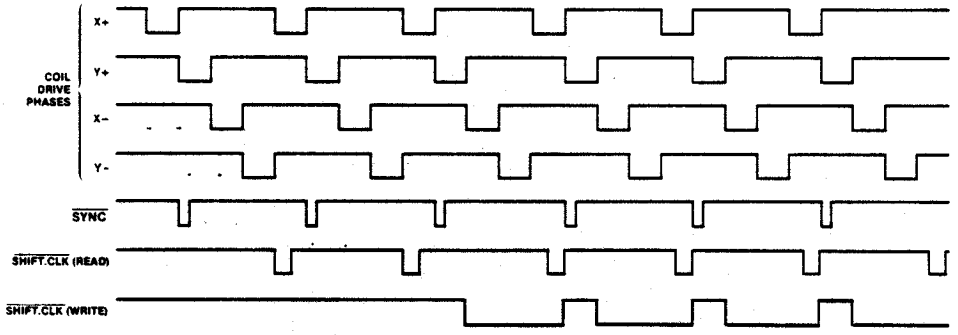


Figure 4-6C. System Timing 121685-21

4.3.4 7230 Current Pulse Generator (CPG)

The Intel 7230 CPG is a Schottky bipolar, TTL input-compatible device designed to drive Intel Magnetics bubble memories. Figure 4-7 is a logic diagram of the 7230 CPG.

Most of the inputs for the CPG come from the 7220 BMC, which supplies all timing signals. The 7242 FSA, however, supplies the chip select signal (CS/) and the generate enable signals (GEN.EN.A/ and GEN.EN.B/ come from the DATA.OUT.A/ and DATA.OUT.B/ outputs, respectively, of the FSA).

The reference current generator establishes a reference to which all the output currents are proportional. The individual current sinks supply the proper currents when enabled by the respective TTL inputs (active low), as shown in figure 4-7. Note that the 'swap' type output signals have one current sink each, while the 'rep' type output signals (including GEN.A and GEN.B, which replicate seed bubbles) have two current sinks each. This is because the replicate pulse is a two-level pulse, with a higher 'cut' current existing for 0.25 microseconds followed by a lower 'transfer' current existing for an additional 4.75 microseconds. The TM.A/ and TM.B/ inputs from the BMC enable the 'cut' and 'transfer' pulses, respectively.

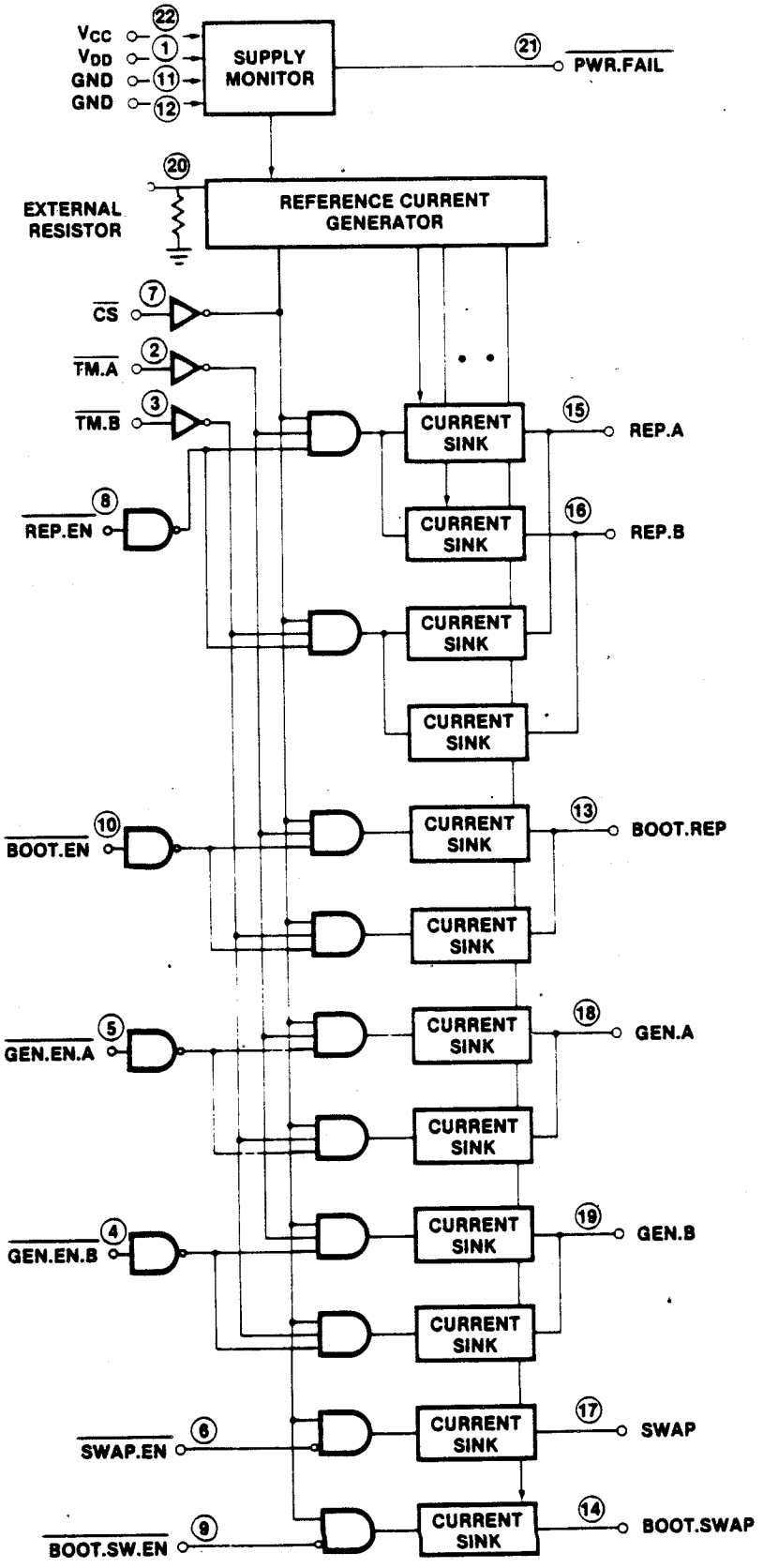


Figure 4-7. 7230 CPG Logic Diagram 121685-22

4.3.5 7250 Coil Pre-Driver (CPD)

The Intel 7250 CPD is a CMOS, TTL input compatible device designed for use with Intel Magnetics bubble memories. The outputs of the 7250 CPD are designed to drive the Intel 7254 VMOS drive transistors. Figure 4-8 is a logic diagram of the 7250 CPD.

The chip select input for the CPD comes from the 7242 FSA. All other inputs come from the 7220 BMC. The 7250 CPD generates high current outputs and their complements, as enabled by the corresponding TTL inputs, as shown in figure 4-8.

4.3.6 7254 Quad VMOS Drive Transistors

The Intel 7254 VMOS drive transistors are designed to supply drive currents for the X and Y coils of Intel Magnetics bubble memories. In a typical system, one 7254 is dedicated to driving the X coil, and another 7254 drives the Y coil. Figure 4-9 is a circuit diagram of the 7254.

In a typical X-coil circuit, S_1 and S_3 are grounded, S_2 and S_4 are tied to +12V, and the gate inputs G_1, G_2, G_3, and G_4 are driven, respectively, by the X−.OUT, X−.OUT/, X+.OUT, and X+.OUT/ signals from the 7250 CPD. The outputs D1 and D4 are tied together and run to the X+.COIL.IN pin of the 7110 MBM. D2 and D3 are tied together and go to the X−.COIL.IN pin of the 7110. The Y-coil circuit is exactly the same, with Y replacing X in the above description.

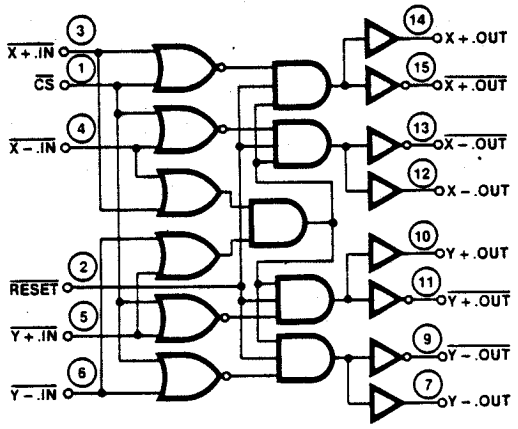


Figure 4-8. 7250 CPD Logic Diagram 121685-23

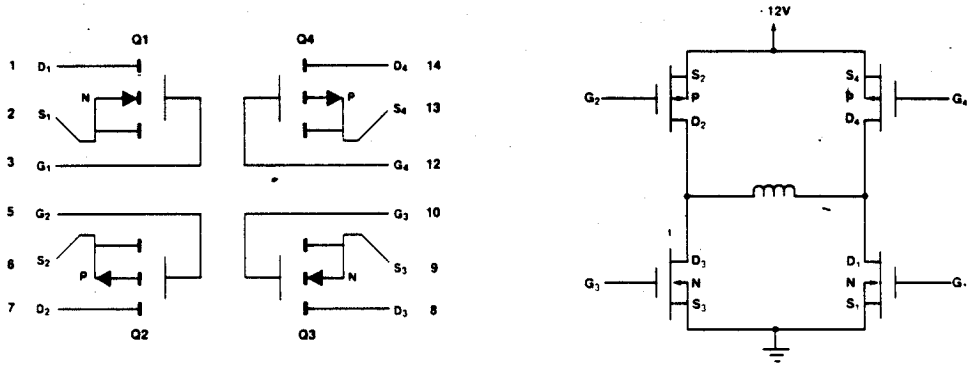


Figure 4-9. 7254 Circuit Diagram 121685-24

4.4 Power Fail Circuitry

The bubble memory is accessed by passing currents of the proper magnitude and phase through two coils inside the memory package. These currents must always be of the proper amplitude and phase or data can be lost. In addition, transients in the coils during power sequencing must be avoided. The purpose of the power fail circuitry is to prevent these transients, and to monitor the system voltages in order to stop the coil currents in the proper phase should power fail.

Figure 4-10A is a block diagram of the power fail circuitry in the bubble memory system. The power fail circuitry involves the 7230, 7220, 7250 and two time delays. The basic operation of the circuitry is as follows. The 7230 contains power supply monitors (+5V and +12V) which detect when either power supply is below the threshold levels for activating the shutdown. When a power failure is detected, the 7230 sends a signal (PWR.FAIL/) to the 7220. The 7220 determines when the coil drivers are in the proper phase and shuts them down. The 7220 then sends a signal (RESET.OUT/) to the 7250. This signal keeps the 7250 in such a state that the 7254 drive transistors are held off as power dies away. During a power up sequence, the 7230 holds PWR.FAIL/ active until both supplies are above the minimum required level. Time delay 1 holds the signal active an additional 2 msec to allow the substrate bias generator on the 7220 to become active. Since time delay 2 is derived from PWR.FAIL/, the RESET.OUT/ signal has been active during this time, holding the 7250 reset and preventing coil transients. When PWR.FAIL/ goes active, the 7220 enters a reset sequence and puts all of its control outputs in the proper state, including RESET.OUT/. Time delay 2 prevents RESET.OUT/ from going inactive until the 7220 has had time to complete its sequence.

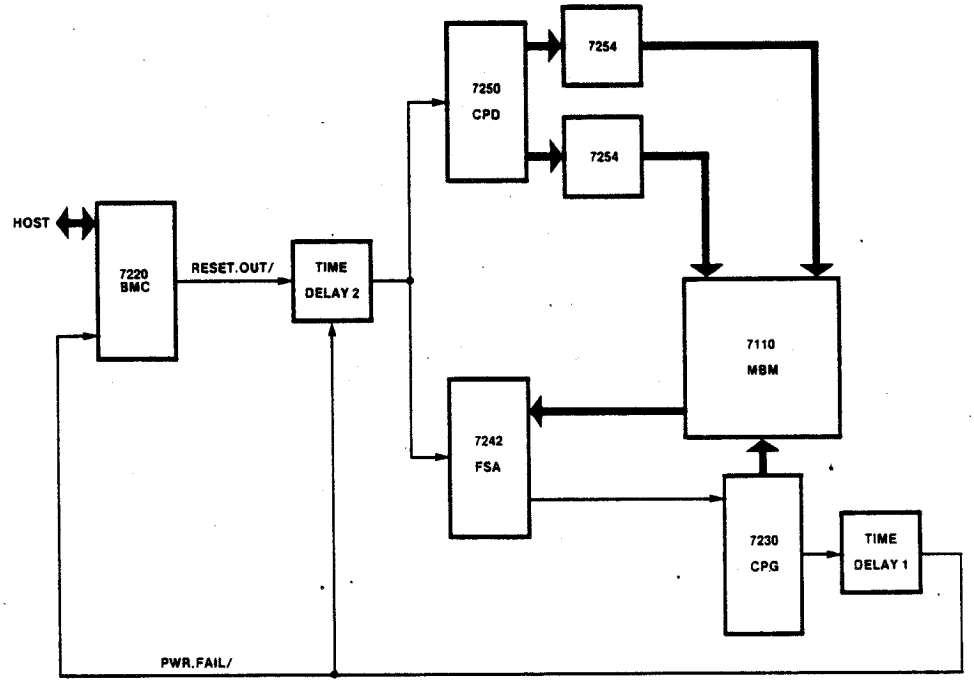


Figure 4-10A. Power Fail Block Diagram 121685-25

The special features built into the system to accommodate power failure are:

1. Power supply monitors built into the 7230. These monitors trip at about −6% of nominal.
2. Power fail input to the 7220. This input causes the 7220 to stop the drive field and issue a RESET.OUT/.
3. Specially designed outputs for the 7250. As long as RESET.OUT/ is applied, these outputs hold the 7254's off, even as power falls to zero.
4. Open collector PWR.FAIL/ and RESET.OUT/. These outputs allow use of RC networks to create time delays 1 and 2.
5. Circuit operation specified to −10% power input.

If the memory coil drive is active, the 7220 will require a maximum of about 470 μsec to shut down the coil drive and issue a RESET.OUT/. Therefore, the user must guarantee that there is enough capacitance in the system so that the power does not fall from −6% to −10% in less than 470 μsec.

Figure 4-10B is a schematic of the power fail circuitry. Time delay 1 is created by R_1 and C_1, while time delay 2 is created by R_2 and C_2. Figure 4-10C illustrates the timing requirements for a power up sequence, while figure 4-10D illustrates a power down sequence. Figure 4-10E is a simplified representation of the time delay networks.

The timing requirements are for minimum delays; there are no maximums. The remainder of this analysis assumes that all values are worst case for power failure.

The equations for PWR.FAIL/ and RESET.OUT/ are of the form $V=V_{IN}(1-Ae^{-at}-Be^{bt})$, where A,B,a, and b are functions of R_1,R_2,C_1, and C_2. However, for the purpose of this analysis, the form $V=V_{IN}(1-e^{-t/RC})$ can be used, with the resulting error (because of the simplification) being on the conservative, or safe side.

The delay requirements during a power-up sequence are such that PWR.FAIL/ must not reach 0.8V (a worst case zero level) until 2 msec after power is up, and it must reach 2.0V (worst case one level) before RESET.OUT/ reaches 0.8V.

Since the effect of R_2 and C_2 is to increase delay 1 (which helps ensure the delay requirements are met), they can be ignored and not affect the validity of the result. The voltage at PWR.FAIL/ (given the earlier simplifications) is given by $V=4.85(1-e^{-t/RC})$. Solving for t = 2 msec and V=0.4V (the difference between 0.4 and 0.8V) yields R_1C_1=23 msec. Using this value of RC and solving for V=1.6V yields 9.3 msec. This value can then be used to calculate R_2C_2 by using a similar set of equations. The solution yields R_2C_2 = 32 msec. These values are approximate (because of the simplifications made). However, they are larger than the minimums, therefore they are safe values. Since R_2C_2 is large (relative to the 50 μsec power down requirement), the minimum power down requirement is satisfied.

The individual values of R and C can vary widely as long as the RC product meets the minimum time constants. The maximum value of R is constrained by the leakage currents of the 7230 and 7220 inputs and outputs as well as capacitor leakages. In addition, temperature effects and minimum power supply voltages must be considered. Suggested values for R_1 and C_1 are 12 kΩ and 2.2 μf, while R_2 and C_2 are 33 kΩ and 1.0 μf.

In addition to the coil drive protection circuitry, there is a VMOS FET transistor (Q1) in the power fail circuit. This transistor is turned off as long as RESET.OUT/ is active. The purpose of the transistor is to shut off the reference current to the 7230 CPG during power sequencing. Without a reference current, all outputs of the 7230 are off, thus preventing transient currents in the 7110 gate elements during power sequencing. The requirements on this transistor are that it can be turned on and off

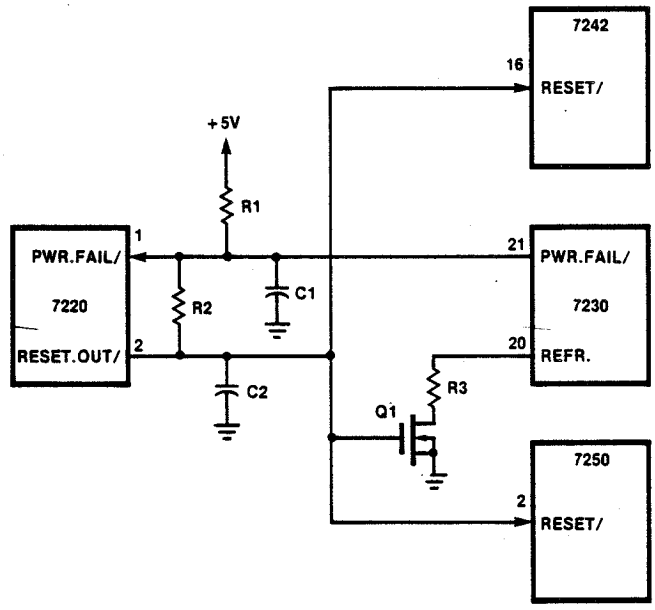


Figure 4-10B. Power Fail Schematic Diagram 121685-26

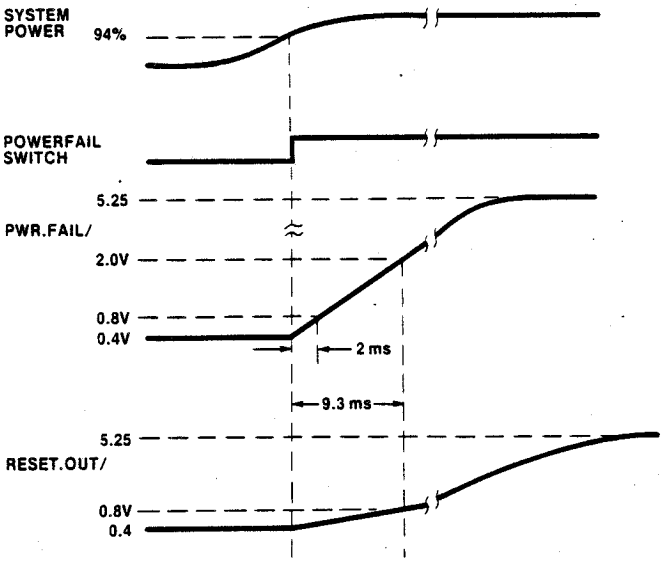


Figure 4-10C. Power-Up Timing 121685-27

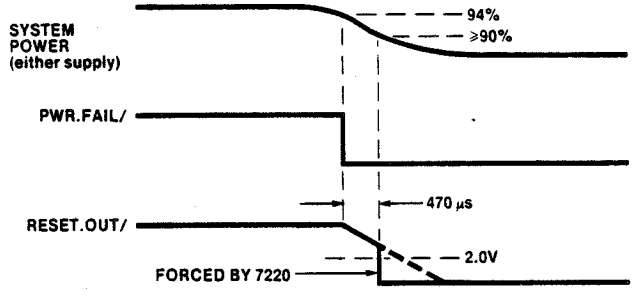


Figure 4-10D. Power-Down Timing 121685-28

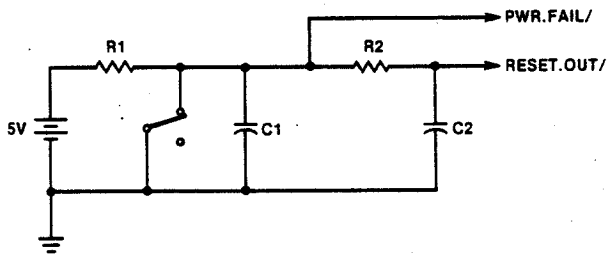


Figure 4-10E. Time Delay Networks 121685-29

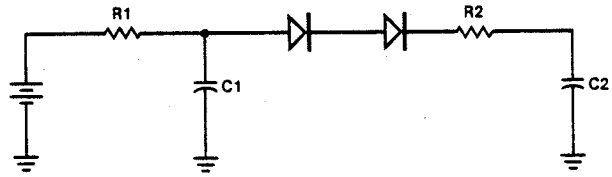


Figure 4-10F. Alternative Power Fail Circuit 121685-30

by TTL level signals and that its "on" resistance is small relative to the 3.48 k-ohm reference resistor. There are several commercially available transistors that meet these requirements.

An alternative circuit that could be used is shown in figure 4-10F. The values for R_1 and C_1 remain the same, while R_2 and C_2 can be much smaller. They can be smaller (they must still meet the 50 μsec requirement) because the two diodes ensure the PWR.FAIL/ reaches 2.0V before RESET.OUT/ reaches 0.8V.

A third approach is to not use the PWR.FAIL/ circuit at all, but to use an external circuit that monitors the input AC line. The PWR.FAIL/ input to the 7220 would be connected to this external signal rather than the 7230. The 2 msec and 50 μsec requirements would still have to be met.

4.5 Error Correction Code

The error correction code used by the 7242 FSA is a 14-bit Fire code (appended to each 256-bit block of data). This code is capable of correcting all single error bursts up to and including five bits in length. The generating polynomial is

$$G(x) = 1 + x^2 + x^5 + x^9 + x^{11} + x^{14}$$

When an error is detected, the FSA activates the ERR.FLG/ line to the BMC, and sets bits in the FSA status register according to whether or not the error is correctable. Uncorrectable errors are extremely rare — in a typical 1-megabyte bubble memory system, the entire useful life of the equipment will pass without a single uncorrectable error.

Use of the Dummy Module

The 7110 Dummy Module is an electrical equivalent of the 7110 bubble memory. This module may be plugged into the test board for debugging, to prevent the 7110 bubble memory from accidental burnout. Since the module uses 1k ohm resistors and the 7110 bubble memory uses coils, the coil drive signals produced during testing will not be the same. Figure 5-1 is a schematic diagram of the dummy module. All resistance values are ±5%. Table 5-1 is a parts list for the module.

Before installing the module, make sure that all power is removed from the board. Install the module in the socket, making sure that pin 1 of the module corresponds to pin 1 of the socket. Resistors should be visible after installation. Next, measure the resistances listed in table 5-2.

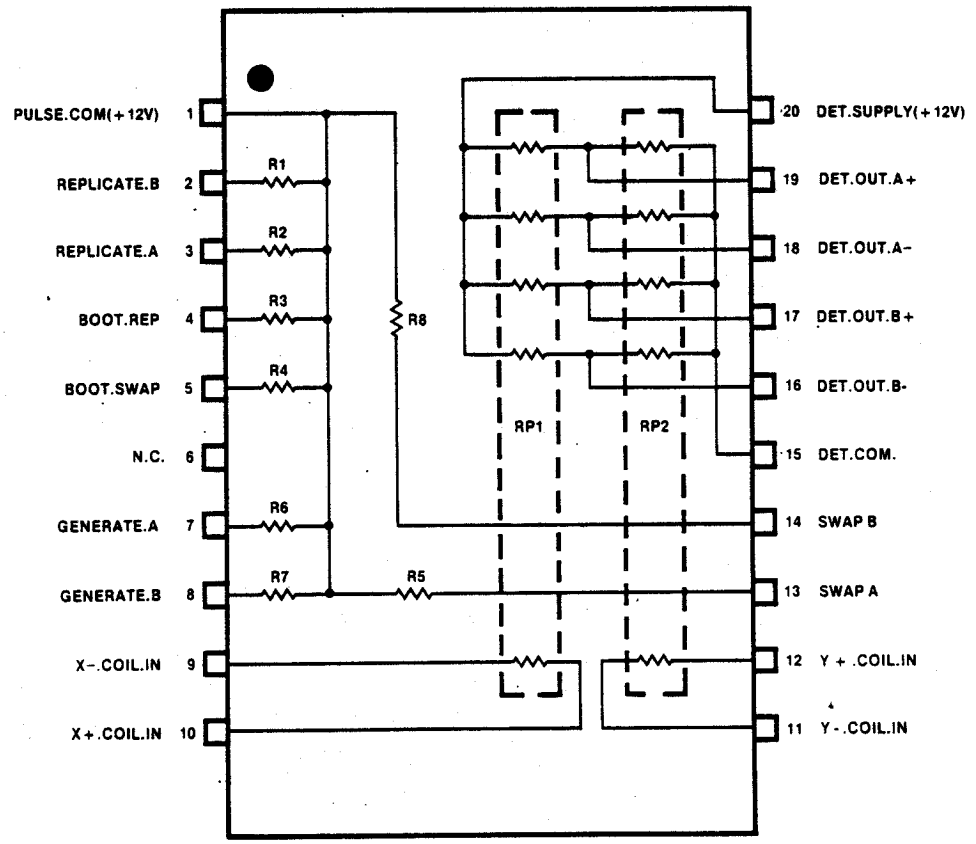


Figure 5-1. Dummy Module Schematic Diagram 121685-31

Table 5-1. 7110 Dummy Module Parts List

Item No.	Part No.	Description	Reference Designation	Quantity
1		PWB 7110 Dummy		1
2				
3	Bourns 4310R-102-102	Resistor Pack; 1K	RP1, RP2	2
4				
5	20, 1/4W, 5%	Replicate (B)	R1	1
6	20	Replicate (A)	R2	1
7	10	Boot Replicate	R3	1
8	20	Boot Swap	R4	1
9	110	Swap (A)	R5	1
10	33	Generate (A)	R6	1
11	33	Generate (B)	R7	1
12	110, 1/4W, 5%	Swap (B)	R8	1

Notes: 1. ASSY and P/L are tracking documents unless otherwise specified.

Table 5-2. Pin to Pin Resistances on the Dummy Module

Pin	To	Pin	Resistance (ohms)
1		2	20
1		3	20
1		4	10
1		5	20
1		7	33
1		8	33
9		10	1K
11		12	1K
1		13	55
1		14	55
15		16	625
15		17	625
15		18	625
15		19	625
15		20	500

Apply power and reset the board. The following D.C. voltages should be measured:

Pin	Voltage + 10%
1,2,3,4,5,6,7,8,13,14,20	+12
16,17,18,19	+6

When exercising the board, verify that the waveforms are as shown in figure 5-2.

After installing the bubble device and before applying power, measure the resistances between pins of the socket as above. The resistance between pins 9 and 10 should be approximately 5.2 ohms and between 11 and 12 should be approximately 2.3 ohms. It is important to measure the resistance and ensure the values are correct before the bubble device is exercised. Measure the resistance of the socket terminals to ensure that there is good contact with the bubble device.

Timing diagrams are given in figure 5-3. Use these diagrams to check the timing of the 7110 device. Figure 5-3 shows the relation of signals to coil drive pulses. All transitions occur on the positive edge of 'SYSCK'. Not all signals occur together but depend on the function performed. Functions reflected in the diagram are: start, stop, run, swap on, swap off, replicate, generate, and detect. Each function requires 64 clock cycles (one field rotation).

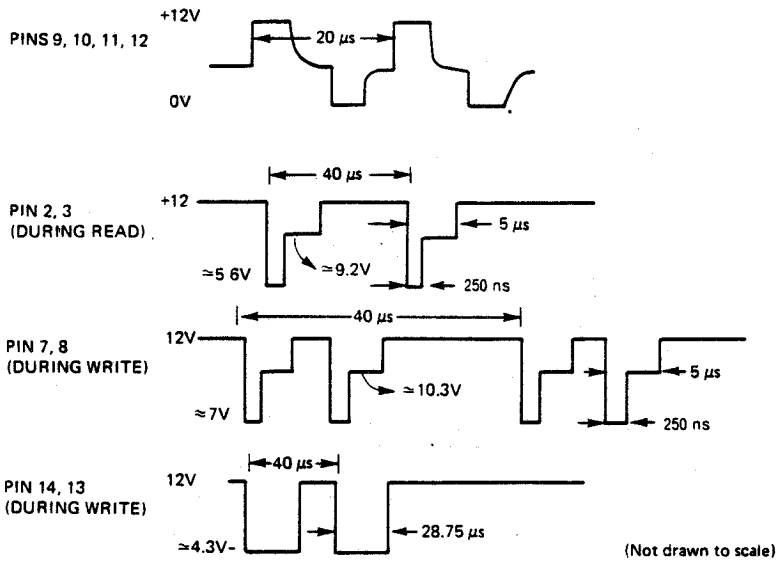


Figure 5-2. 7110 Dummy Module Waveforms 121685-32

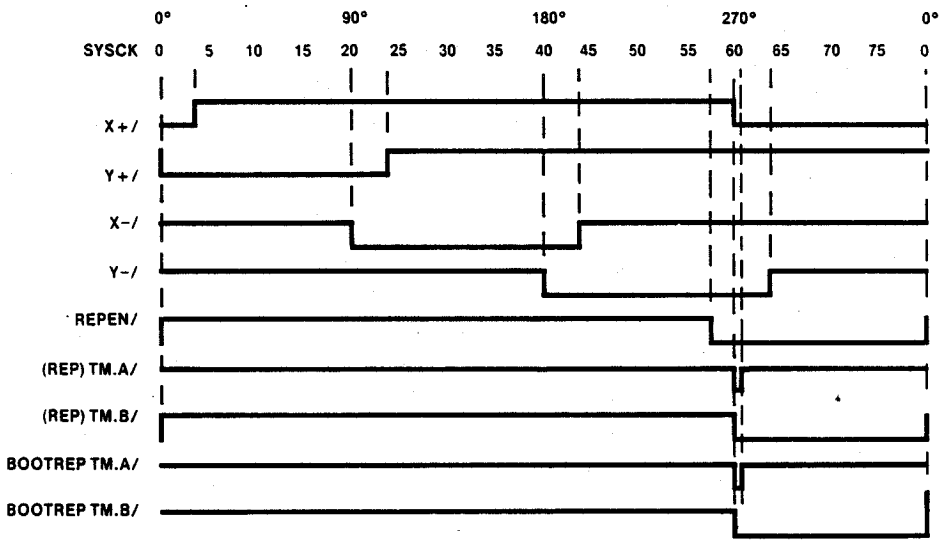


Figure 5-3A. Composite Bubble Timing Diagram 121685-33

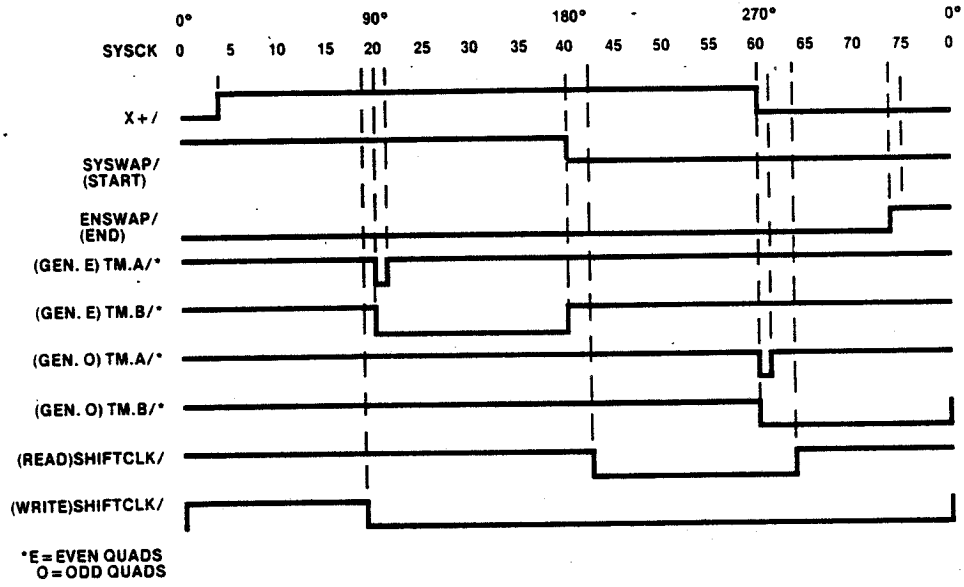


Figure 5-3B. Composite Bubble Timing Diagram 121685-34

5.2 Use of the Seed Module

To facilitate re-generation of the seed bubbles in the 7110, a special module has been provided. Figure 5-4 is a schematic of the seed module. Figure 5-5 is the corresponding assembly drawing. Table 5-3 is a parts list for the seed module. Should a seed bubble be inadvertently destroyed (by component failure, socket contact problems, or timing problems), the following seed procedure should be used:

1. Remove power from circuit.
2. Remove the 7230 current pulse generator from its socket, and install the 7230 in the socket provided on the seed module. Be careful to note the orientation of pin 1.
3. Install the seed module (with the 7230 installed) in the 7230 socket.
4. The 7110 has a 47Ω resistor (R11) between pin #1 and +12V. This resistor must be disabled for seed generation. Jumper 7110 pin #1 directly to +12V.

CAUTION

Failing to disable resistor R11 can cause permanent damage to the MBM when power is applied.

5. Apply power to circuit.
6. Send an Abort command.
7. Send MBM Purge command.
8. Write BMC registers for one page write.
9. Place two bytes of data in FIFO.
10. Execute a single page write. The data pattern is not important, since the TM.A/ signal, which does not depend on data, is used (some data must be loaded into the FIFO, however, to allow the controller to proceed with the write operation).

11. Remove power from circuit.
12. Remove the jumper installed in step 4.
13. Remove seed module from 7230 socket.
14. Remove the 7230 from the seed module and reinstall the 7230 in its socket.

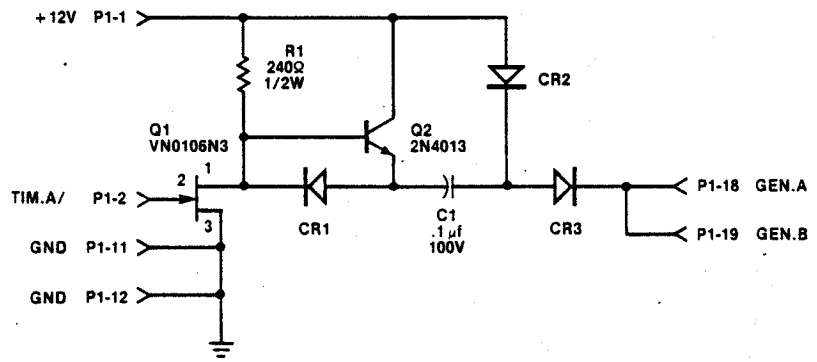


Figure 5-4. 7110 Seed Module Schematic 121685-35

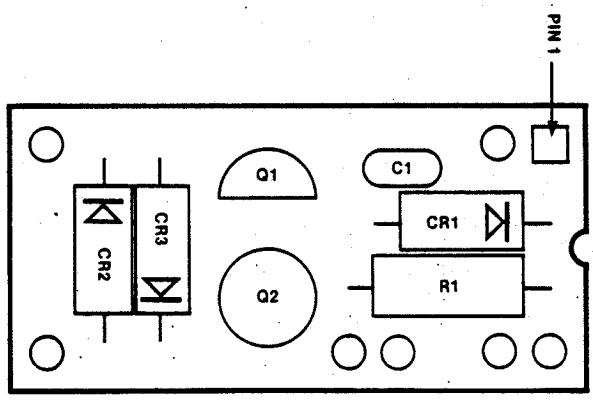


Figure 5-5. 7110 Seed Module Assembly 121685-36

Table 5-3. Seed Module Parts List

Reference	Description	Intel Part No.	Qty.
	PC Board, Seed Module	151838-001	1
C1	Capacitor, Ceramic 0.1μF, 100V	101754-049	1
CR1, 2, 3	Diode, 1N5818	101358-001	3
Q1	Transistor, VN0106N3	103919-001	1
Q2	Transistor, 2N4013	101428-001	1
	Socket, Augat GX151-80-161		1

To verify the presence of a seed bubble, proceed as follows:

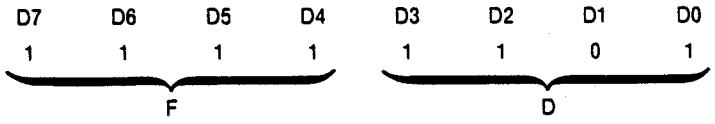
1. Apply power to circuit.
2. Send an Initialize command.
3. If boot loop is suspect or the Initialize command returns with a timing error, the boot loop register must be filled with all ones (FF's).
   a) Fill the FIFO with 40 bytes of ones (FF's).
   b) Send the Write Boot Loop Register command.
4. Execute a single page write. Data pattern is 68 bytes of FF's (34 bytes if block length MSB=0). Do not use error correction for seed verification.
5. Execute a single page read. Data pattern should be solid FF's if the boot loop was correct and Initialize was successful. Otherwise the data pattern should be mostly FF's. Holes in the pattern indicate bad loops.
6. If the data is not mostly FF's, repeat the Seed Procedure from step 1. If the seed still does not verify, contact your field sales office to arrange for return and replacement of the entire BPK 72 kit.

5.3 Writing the Boot Loop

The boot loop for each MBM is written at the factory, and typically requires no attention during the entire useful life of the bubble memory module. As an added precaution, however, the boot loop information is printed on the label attached to the MBM. The following example shows the format used:

```
7110-1
9GV4          8027
FDFBFFDFBFFDFFFF
9FFFFBFF7FFBFFFB
BFFFBF2FFFBFFDFF
7FF7BFFEFCEFFFFF
FFDFBFFBEFFFDF3F
```

The first line is the part number of the MBM; the second line contains the serial number and date code. Beginning on the third line, the 40 bytes making up the boot loop information are listed. Each byte is represented by two hex digits, and is written exactly as it should be sent to the BMC FIFO. For example, the first byte is FD. F is the most significant digit, and D is the least significant digit. A "1" represents a good loop, and a "0" represents a bad loop. The byte FD is sent to the BMC via data lines D0-D8, as follows:



The complete procedure for writing the boot loop is as follows:

1. Send the Abort command to the BMC.
2. Send the MBM Purge command.
3. Send the FIFO Reset command.
4. Set the NFC bits in the BLR (block length register MSB, bits 4-7) to 0001 (indicating two FSA channels).
5. Set ER (enable register) bit 3 equal to one (WRITE BOOT LOOP ENABLE).
6. Set the upper four bits of the AR (address register MSB, bits 3-6) to indicate the MBM whose boot loop is to be written.
7. Write the selected boot loop register with all ones (write 40 bytes of ones to the BMC FIFO, then send the Write Boot Loop Register command).
8. Write the boot loop itself (write the 40 bytes shown on the label to the BMC FIFO, followed by a 41st byte of zeroes, then send the Write Boot Loop command).

APPENDIX A
7110 BUBBLE MEMORY SOCKET INSTALLATION

The 151709-Rev. 1 socket for the 7110 Bubble Memory provides mechanical as well as electrical connection to a printed circuit board. Electrical connection is made with the socket's 20 solder pins; mechanical connection is made by means of four staking pins that also provide strain relief for the solder pins. The four staking pins are an integral part of the molded socket housing and, at board assembly time, these pins are staked (riveted) so that the socket becomes an integral part of the printed circuit board (see figure A-1).

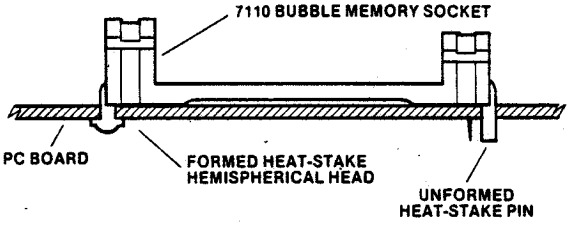


Figure A-1. Socket Installation 121685-37

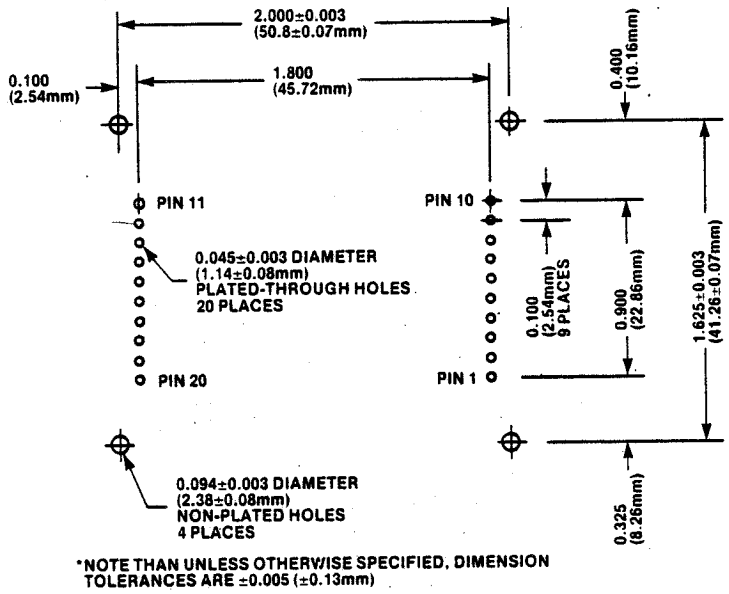
Riveting is accomplished by applying heat and pressure with the special soldering-pencil tip (part number 151875) that has been included with the BPK 72 kit. This tip is used to form the required hemispherical heads on the four heat-staking pins and is intended to be used with a Weller® EC2000 Electronic Soldering System (or equivalent).

The following procedure outlines bubble memory socket installation. Note that when possible, the socket should be attached to its pc board prior to the installation of the other components.

1. Install special heat-stake tip on the EC1201 soldering pencil and set the temperature control of the EC2002 Power Unit for between 400°F and 405°F (190°C — 193°C).

2. Insert the 7110 Bubble Memory Socket on the component side of the pc board as shown in figure A-2. Be sure to align pin 1 of the socket with pin 1 on the pc board.

3. With the pc board firmly supported (component side down on a workbench), form the hemispherical heads by pressing the special heat-stake tip against the ends of the four staking pins protruding through the board. Hold the soldering pencil at a 90° angle to the pc board, center the tip over the heat-stake pin and apply sufficient pressure until the head forms and fits tightly against the pc board (approximately ten seconds are required to form the head).



121685-38

Figure A-2. Printed Circuit Board Socket Mounting Dimensions

The special tip will not damage the pc board if the correct temperature is used. Care should be taken to hold the soldering pencil at a right angle to the pc board in order to prevent slipping and the possibility of gouging the board. Note that the socket should be firmly seated against the board and that the hemispherical heads should appear shiny; a dull black appearance indicates that the temperature of the tip is too high.

For low-volume production, a fixture that clamps the socket to the board and provides alignment for the staking tool is available. Contact Intel Magnetics Inc. for additional information.

Heat stake sockets in place on the pc board are not damaged by the wave-soldering process as long as no undue pressure is applied to the socket.

REQUEST FOR READER'S COMMENTS

The Microcomputer Division Technical Publications Department attempts to provide documents that meet the needs of all Intel product users. This form lets you participate directly in the documentation process.

Please restrict your comments to the usability, accuracy, readability, organization, and completeness of this document.

1. Please specify by page any errors you found in this manual.

2. Does the document cover the information you expected or required? Please make suggestions for improvement.

3. Is this the right type of document for your needs? Is it at the right level? What other types of documents are needed?

4. Did you have any difficulty understanding descriptions or wording? Where?

5. Please rate this document on a scale of 1 to 10 with 10 being the best rating. _____

NAME _____ DATE _____

TITLE _____

COMPANY NAME/DEPARTMENT _____

ADDRESS _____

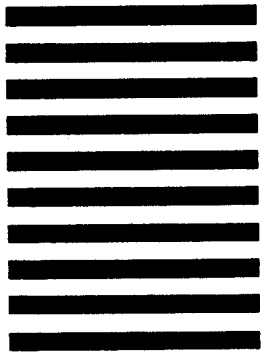
CITY _____ STATE _____ ZIP CODE _____

Please check here if you require a written reply. ☐

WE'D LIKE YOUR COMMENTS ...

This document is one of a series describing Intel products. Your comments on the back of this form will help us produce better manuals. Each reply will be carefully reviewed by the responsible person. All comments and suggestions become the property of Intel Corporation.

NO POSTAGE
NECESSARY
IF MAILED
IN U.S.A.



BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1040 SANTA CLARA, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Intel Corporation
Attn: Technical Publications M/S 6-2000
3065 Bowers Avenue
Santa Clara, CA 95051

intel® magnetics

3000 OAKMEAD VILLAGE DRIVE, SANTA CLARA, CALIFORNIA 95051 • (408) 987-7700

Printed in U.S.A./Q144/0980/5K/NCG