Bubble Memory 6

intel

A PRIMER ON MAGNETIC BUBBLE MEMORY

Magnetic bubble memory is a solid-state technology with high reliability, ruggedness, small size, light weight, and limited power dissipation. It has applications in telecommunications, data acquisition, industrial control, terminals, and small business computers. Yet many potential users remain unsure of the nature of a bubble memory. This primer is intended to introduce these users to the technology.

What a Magnetic Bubble Memory Is

A magnetic bubble memory stores data in the form of cylindrical magnetic domains in a thin film of magnetic material. The presence of a domain (a bubble) is interpreted as a binary 1, and absence of a domain is a 0. Bubbles are created from electrical signals by a bubble generator within the memory, and reconverted to electrical signals by an internal detector. Externally the memory is TTL-compatible.

An external rotating magnetic field propels these cylindrical domain bubbles through the film. Metallic patterns or chevrons deposited on the film steer the domains in the desired directions. Transfer rates, once started, are in the tens of thousands of bits per second, but because the data circulates past a pickup point at which it becomes available to the outside world, there is a latency averaging tens of milliseconds before data transfer can begin. In these respects, magnetic bubble memories are serial high-density storage devices like electromechanical disk memories. However, in a disk, the stored bits are stationary on a moving medium, whereas in the magnetic bubble memory the medium is stationary and the bits move.

Advantages of Magnetic Bubble Memories

The principal advantage of magnetic bubble memories are their non-volatility—that is, if power fails, the stored data is retained. Products incorporating bubble memories therefore do not require battery backups. Magnetic bubble memories share this feature with read only memories (ROMs), erasable PROMs (EPROMS), and electrically erasable PROMs (E^2PROMS). However, unlike any of these technologies, magnetic bubble memories can have data written into them at any time, at speeds comparable to those at which data is read. Furthermore, unlike disk memories, bubble memories are quiet and very reliable, because they have no moving parts. They are compact, and they dissipate very little power. Their support circuits are compatible with microprocessor systems. With a million or more bits per device, a bubble memory can store 16 to 64 times the amount of data of alternative semiconductor memories, providing very high storage capability in a compact space. Bubble memory has a wide variety of applications, some of which are listed in Table 1.

Table 1. Bubble Memory Applications

Numerical control	Robotics
Process control	Oil exploration
Aircraft navigation	Data acquisition
Cable television	Portable instruments
Telecommunications terminals	Avionics
Point-of-sale terminals	Gasoline pumps
Private branch telephone exchanges	Personal computers
Word processors	Office equipment
Flight-line test equipment	Automatic test equipment
Data encryption	

How Bubbles are Formed

Magnetic domains are found in all kinds of magnetic materials—iron bars, the coating on magnetic tape, ferrite toroids (the most common form of computer memory in the 1960s). Each domain is a group of atoms with parallel magnetic orientations. When the material in bulk is unmagnetized, the domains are oriented at random in three dimensions. When the material is magnetically saturated, most of the domains have the same orientation. Magnetization to a level less than saturation orients some of the domains to a common direction, but leaves many of them randomly oriented. When a domain orientation changes, usually by imposing an external magnetic field, the domain itself does not physically move, but boundaries between domains that have different orientations move or disappear altogether.

In an extremely thin film, less than 0.001 inch thick, the domain orientations may be constrained to two dimensions. In some kinds of material (orthoferrites and garnets), with proper crystallographic orientation, the domain orientations are always perpendicular to the film. When these materials are not in a magnetic field, some domains are oriented upward and some downward (north magnetic poles of some domains are on top of the film, and those of other domains are on the bottom). In these materials, the magnetic domains tend to be long and snakelike in the absence of an external field (Figure 1). When a weak magnetic field is applied perpendicular to the film, the domains that are oriented opposite to the applied field become substantially narrower. As the applied field, called a *bias field*, is made stronger, the length continues to decrease, until it becomes approximately the same as the width. Each domain is now cylindrical, magnetized oppositely to the applied field, and immersed in a much larger domain that is magnetized in the same direction as the field.

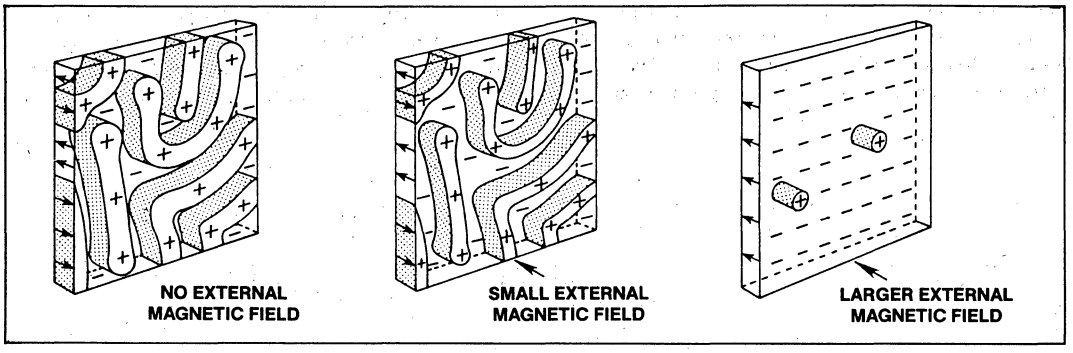


Figure 1. Magnetic Domains in Thin Film Under Increasing Magnetic Bias Field.

These small domains are the bubbles, generally less than 3 micrometers (1/ 10,000 inch) in diameter (Figure 2). When they are viewed from above, only the round shape is apparent, giving the domains the appearance of bubbles. If the bias field were to be made still stronger, all the bubbles would shrink and then disappear altogether; the entire film would be magnetized in the same direction as the bias field. The effect is reversible—that is, if the bias is removed, the domains return to a snakelike form.

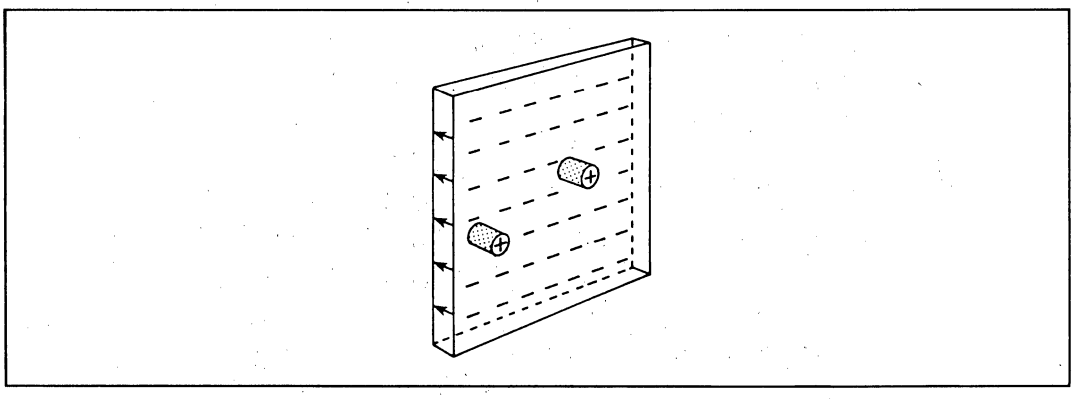


Figure 2. Magnetic Bubbles in a Thin Film

Why a Bubble Moves

Magnetic bubbles will move if they are in a magnetic field gradient. For instance, it will move from a region of lesser magnetic field strength to a region of greater strength. This is similar to the way a nail is pulled to the end of a bar magnet when it gets close the magnet.

In a bubble memory a magnetic film pattern is overlaid on the layer of bubbles. When this layer is magnetized it pulls the bubbles to the points of greatest field strength (or poles) as in Figure 3. The bubbles could then be moved if the pattern elements were moved.

A more easily controlled magnetic field is generated by two coils wrapped around the bubble layer and magnetic film pattern. With appropriate specification of the current in two coils positioned at right angles, the direction of the poles on the stationary elements can be changed in a controlled manner.

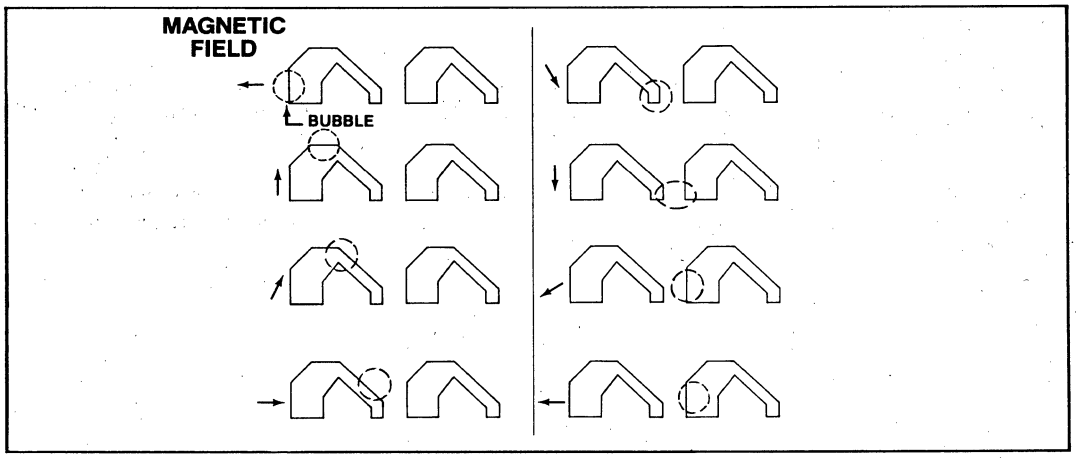


Figure 3. Bubble Propagation Under Asymmetric Chevrons

Various shapes for these metallic patterns have been used by different manufacturers to control the movement of the bubbles. At Intel asymmetric chevrons are used (Figure 3).

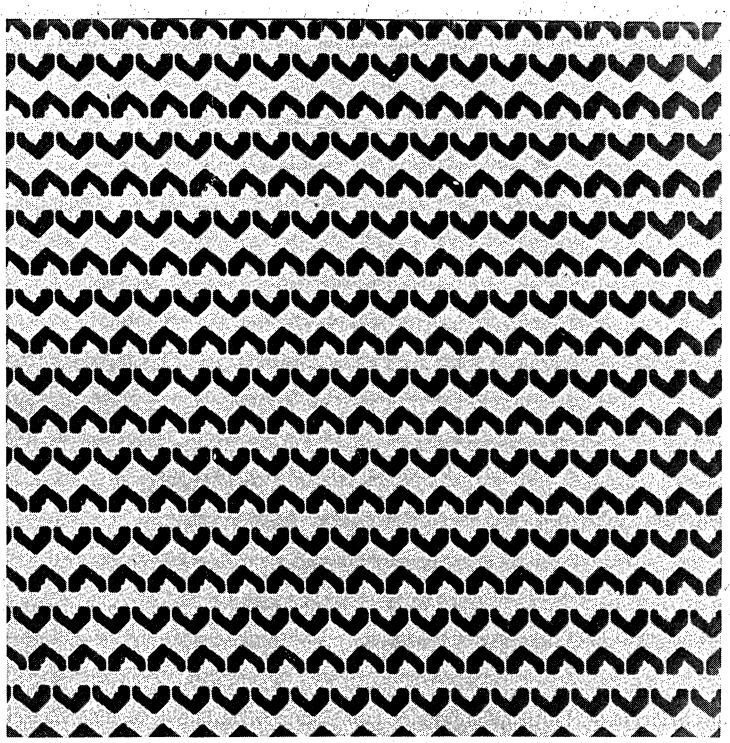


Photo 1. Asymmetric Chevrons Deposited on a Thin Film

Why Magnetic Bubbles are Non-Volatile

In a magnetic bubble memory system, the bias field in which the bubbles exist is generated by a pair of *permanent* magnets. The substrate bearing the thin film and its bubbles is mounted between these magnets and is therefore continuously subject to the bias field.

The rotating field that propels the bubbles through the film is generated by currents in two coils wrapped around the substrate at right angles to each other. These currents are generated by electronic circuits that are part of the magnetic bubble memory system. No mechanical motion is involved.

If power fails, the circuits stop operating, the rotating field disappears, and the bubbles stop moving. But the bias field, generated by the permanent magnet, is not affected. Therefore the bubbles and the data that they represent are maintained in the film. When power is restored the data is again accesible.

BUBBLE MEMORY MANUFACTURING TECHNOLOGY

Bubble memories are produced in a process that resembles semiconductor manufacturing in many ways (Figure 4). Manufacturing begins with a nonmagnetic garnet wafer on which a magnetic film is deposited, using conventional techniques. An ion implantation process alters the magnetization of the top surface of the film, discouraging the formation of abnormal bubbles with undesirable dynamic properties. Then nonmagnetic conductors, bubble-steering patterns of magnetic metal, insulation, passivation, and bonding pads are deposited in much the same way as successive layers on semiconductor integrated circuits. Patterns in each layer are defined photolithographically, just as with semiconductors.

Magnetic bubble technology differs from semiconductor technology in the materials used and in the complexity of the process. Semiconductor circuits use eight or more layers of silicon doped with various materials that affect its electrical characteristics, compared to about three layers of essentially pure metallic and insulating material in bubble technology. These materials are chosen for their magnetic rather than their electrical properties.

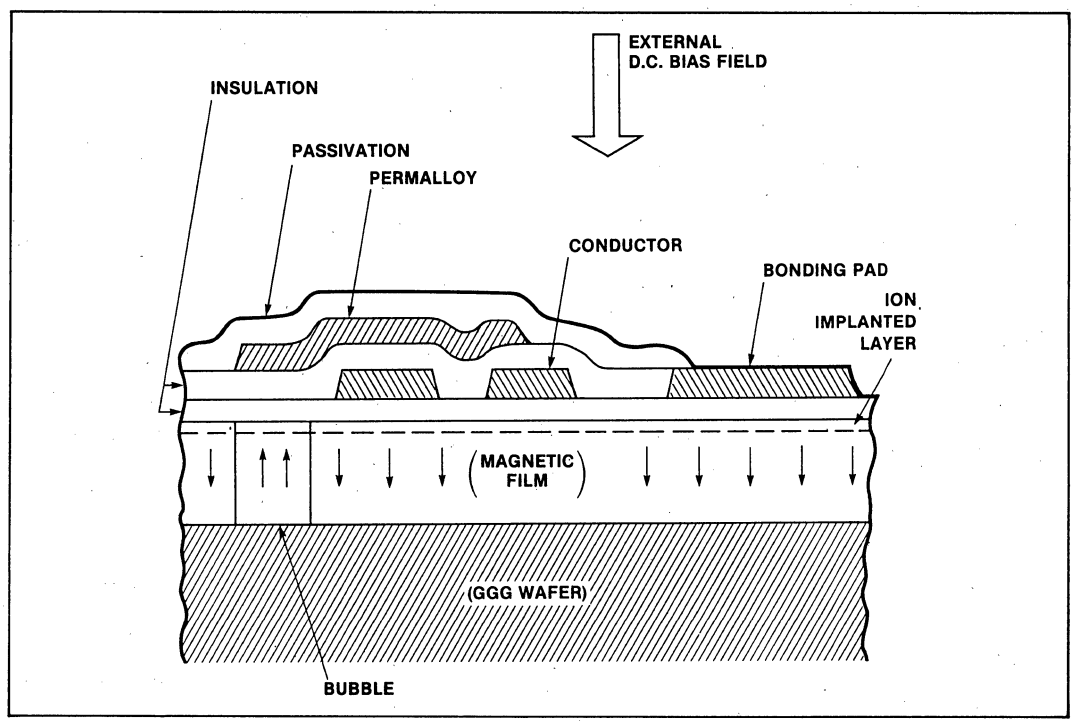


Figure 4. Magnetic Bubble Chip Cross Section

Bubble Memory Functional Description

The Intel 7110 magnetic bubble memory unit contains the bubble chip, the coils that generate the rotating field, two permanent magnets for the bias field, and a magnetic shield that prevents disturbances by external fields and forms a return path for the bias field around the bubble chip (Figure 5).

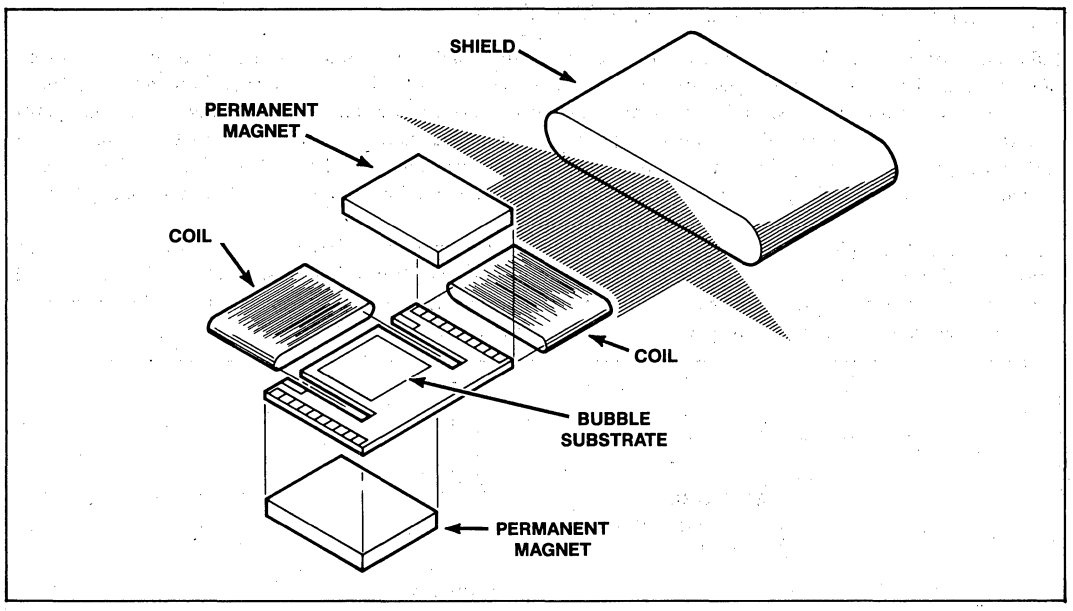


Figure 5. Magnetic Bubble Unit Assembly—Exploded View

Bubble Memory Architecture

Data is stored in the bubble memory unit with a block-replicate architecture (Figure 6). This architecture consists of a number of endless storage loops around which corresponding bits of successive pages continuously circulate, and two tracks, designated input and output, through which the controller writes and reads data in the storage loops. Exchange or replication of data between the tracks and the loops occurs in all loops simultaneously—the key idea in this architecture.

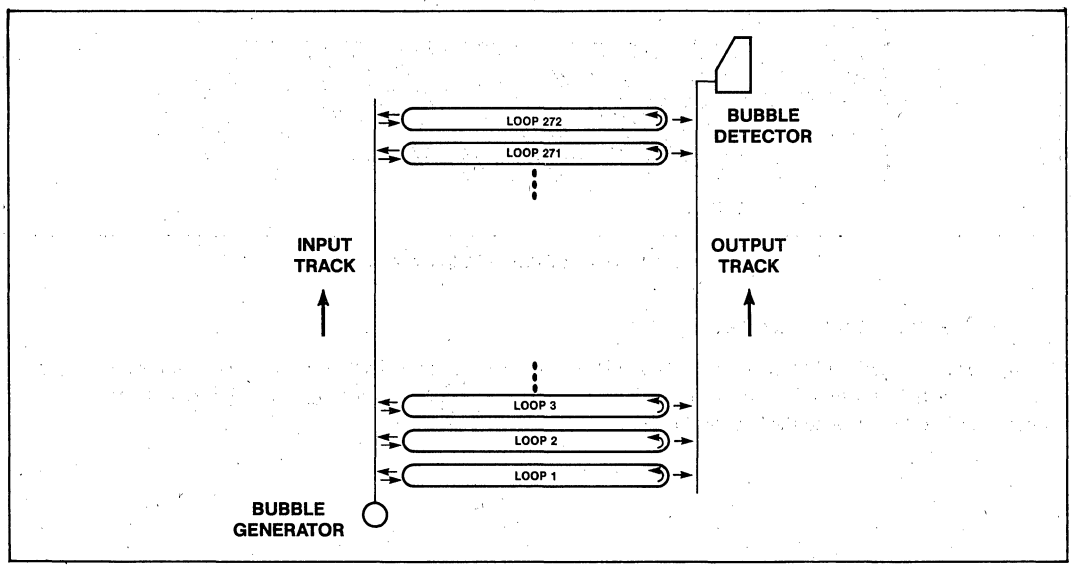


Figure 6. Block-Replicate Architecture

WRITING DATA INTO THE BUBBLE MEMORY

Seed Bubble

The seed bubble, at the beginning of the input track, is generated by an electric current pulse in a hairpin-shaped loop of conductive material. The pulse is strong enough to reverse the bias field locally and thus allow a bubble domain to be created. Once having been created, the seed bubble remains in existence as long as the external bias field is maintained.

The seed circulates under a permalloy patch, driven by the rotating field that propagates bubbles elsewhere in the memory. This bubble is constrained to a kidney shape by interaction of the bias and rotating field with the metal patch (Figure 7). The seed is split in two by a current pulse in the hairpin-shaped conductor. One of them remains under the patch as the seed, quickly regaining its original size; the other one, driven by the rotating field, is transferred to the input track section of the chip. The current pulse that splits the seed is generated to store a binary 1 in the memory; to store a 0, the pulse is omitted, and no bubble is generated.

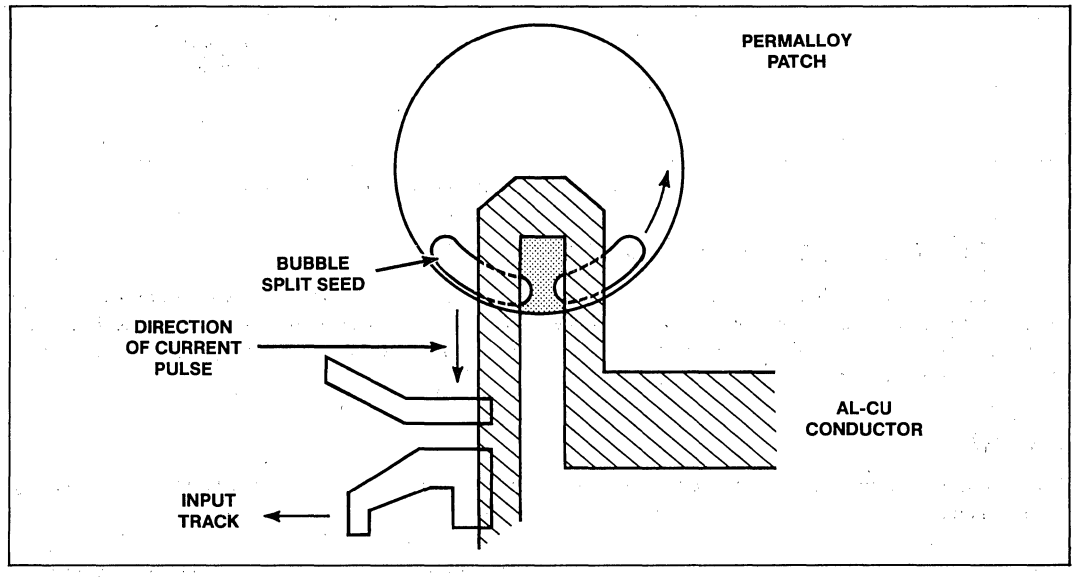


Figure 7. Seed Bubble and Bubble Generation

A seed bubble is maintained at one end of the input track. Bubbles corresponding to binary 1's in the input word are split from the seed and propagate along the input track. When the input track contains exactly one page (64 bytes) then the bubbles exchange places with old bubbles previously circulating in the loops. This is accomplished by an operation called swapping. Thereafter the new bubbles circulate, while the old bubbles now in the track propagate to the end and are destroyed.

Swapping

Transfer of data from the input track to a storage loop involves a swap, bringing the old data onto the input track for destruction at the end of the line, while the new data takes its place in the loop. This is done when a current pulse in an associated conductor under the chevrons causes a bubble to jump from the input track to the storage loop and vice versa. The swap pulse is essentially rectangular, preserving the bubble without cutting it in two.

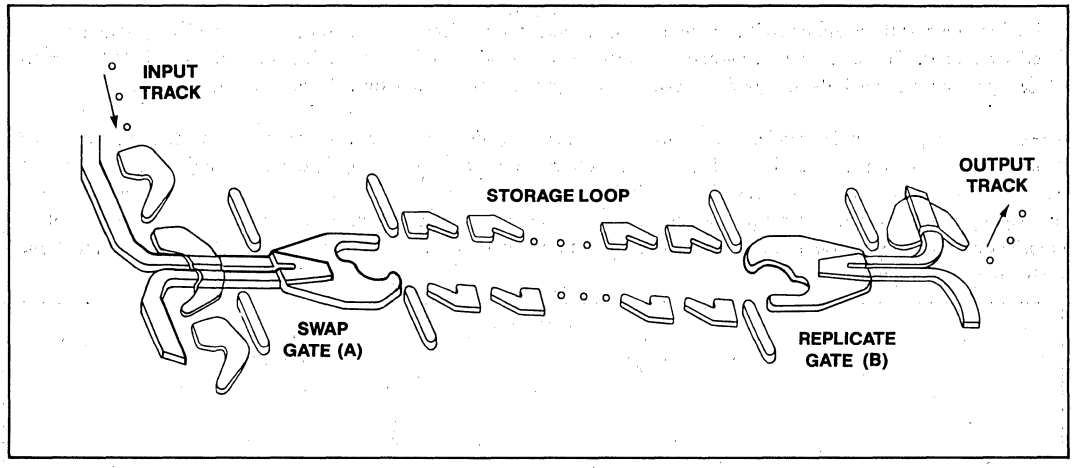


Figure 8. Swapping and Replication Configuration in Bubble Memory

READING DATA STORED IN THE BUBBLE MEMORY

To read the stored data, the circulating bubbles are replicated, one bubble or one unoccupied bubble site from each loop, onto the output track, after which they propagate to a bubble detector at its far end. After detection, these output bubbles are also destroyed. Meanwhile, the data in the loops continues to circulate, permitting a particular page to be read out repeatedly without regeneration, and protecting the stored data if power fails.

Replication

Data is transferred from the storage loop to the output track by replication, continuing to circulate in the loop after having been read out.

For replication, the bubble is propagated under a large element where it is stretched out. As it passes under a hairpin shaped conductor loop it is cut by a current pulse just as in bubble generation.
The replicating current pulse waveshape has a high, narrow leading spike for cutting the original bubble in two, and a lower and wider trailing portion during which the new bubble moves under the output track. The entire pulse lasts about one-quarter of a cycle of the rotating field. In this manner the data in the storage loops is replicated onto the output track, and yet retained in the storage loops in case of a sudden power failure.

Near the end of the output track is a bubble detector—essentially a magnetoresistive bridge formed by interconnecting the permalloy chevrons to make a continuous electrical path of maximum length (Figure 9). As bubbles pass under the bridge, the resistance changes slightly, modulating the currents through the bridge and creating an output voltage of several millivolts. Bubbles are stretched at right angles to the direction of propagation by adding parallel rows of chevrons; these stretched bubbles generate larger output signals at the detector. Beyond the detector, the output track runs the bubbles into the guard rail and destroys them.

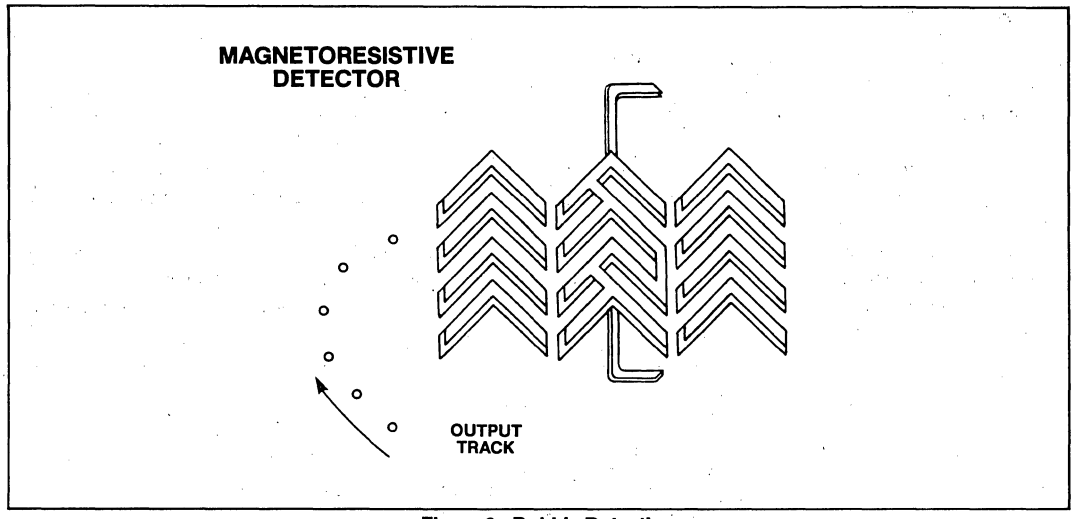


Figure 9. Bubble Detection

Redundancy

The Intel magnetic bubble memory unit physically stores data in 320 storage loops, with capacities of 4,096 bits each. Of the 320 loops, 272 are actually used (active) and 48 are spares (inactive); the boot loop records which loops are used.

Boot Loop

Some of the loops of an individual memory are set aside as spares. The decision as to which loops are to be used (active) and which are not to be used (inactive) is made after the memory unit has been assembled and is undergoing tests at the factory. The outcome of this decision is stored in an extra loop included in each memory chip, in the form of a 12 bit code for each "active" and "inactive" loop.

Whenever power is turned on in the memory system, the system must be initialized before it can be used. Part of the initialization process includes reading the contents of this extra loop, called the boot loop, and placing this information in a bootloop register in the formatter/sense amplifier. From then on, as long as power is on, this register identifies the "active" loops for both reading and writing; "inactive" loops are ignored. The formatter does not attempt to store data in "inactive" loops, and the sense amplifier ignores any data that appears from these loops.

Data Storage—External Appearance

Data is stored logically as 2,048 pages of 512 data bits each. 256 data bits plus 14 error-correction check bits and 2 unused bits are stored in each half of the bubble chip. If automatic error correction is not used, these 16 bits are available for data storage.

Error Correction

Error detection and correction can be performed in the formatter/sense amplifier, which includes a 14-bit cyclic redundancy code that corrects a single burst error of up to five bits in each 270-bit block including the code itself. These code bits are appended to the end of each 256-bit data block when writing into the cell, and checked when the block is read. The error correction feature can be used or not at the user's discretion, by properly setting a register in the bubble memory controller chip. If it is not used, the loops occupied by the code bits become available for additional data.

Access Time and Data Rate

Bubbles circulate at a rate of 50 kilohertz (the rotating field makes 50,000 complete revolutions per second). Average access time to the first bit of the first page is about 41 milliseconds—half the length of time required for a bubble to make one complete circuit of the loop, plus the time to shift a bubble along the length of the output track.

The 320 active and spare loops are actually in four "quads" of 80 loops each (Figure 10). This arrangement shortens the input and output tracks and thus reduces the read and write cycle times. The quads are separately addressable in pairs; in each pair the quads store odd-numbered and even-numbered bits of a word respectively. There are four seed bubbles and four input tracks, and four output tracks. The four output tracks share two detector bridges in such a way that there can never be bubbles from two tracks in a single detector simultaneously. By this means the four streams of output bubbles are interleaved into two bit streams that are stored in two registers in the sense amplifier. The data in these registers is interleaved again into a single stream transmitted serially to the controller.

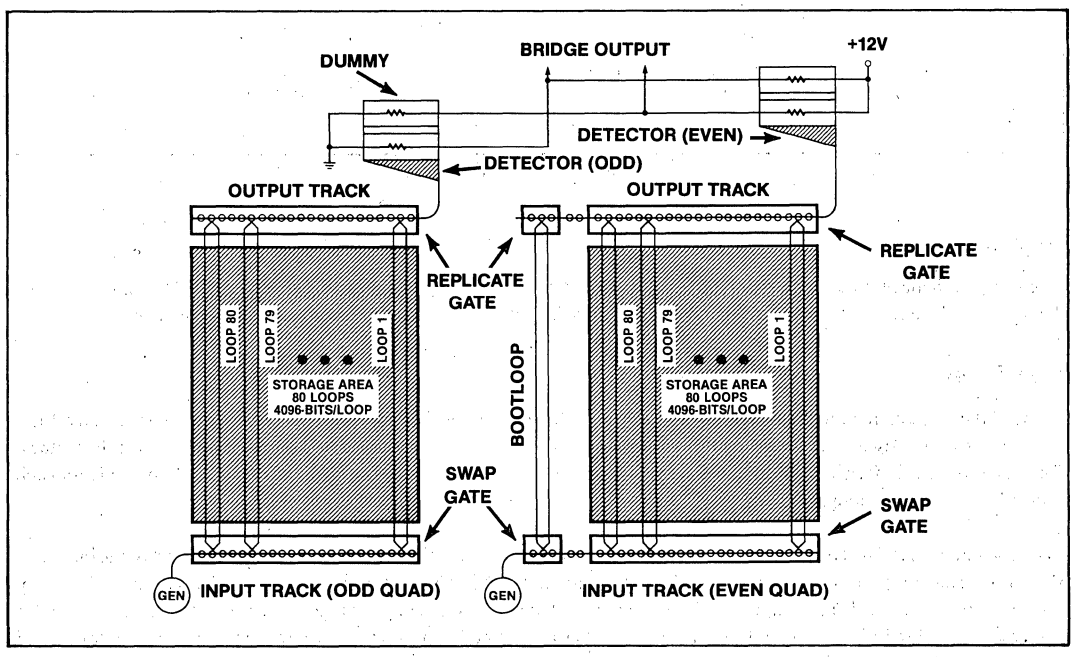


Figure 10. Organization of Bubble Memory (One-Half Chip)

SPECIFIC STRUCTURES OF A MAGNETIC BUBBLE MEMORY

A magnetic bubble memory system consists of a controller and up to eight 1-megabit magnetic bubble subsystems. A minimum system has a controller and one subsystem. The subsystem comprises one magnetic bubble unit in which the data is actually stored, and the peripheral units listed in Table 2 and diagrammed in Figure 11. These circuits are described later in this primer.

Table 2. Components of Intel Bubble Memory System

CONTROLLER	SUBSYSTEM
7220 Bubble Memory Controller (for 1 to 8 subsystems)	Memory 7110 Magnetic Bubble Unit Peripheral Units 7242 Formatter/Sense Amplifier 7230 Current Pulse Generator 7250 Coil Predriver 7254 Drive Transistor Assembly (2 required per subsystem)

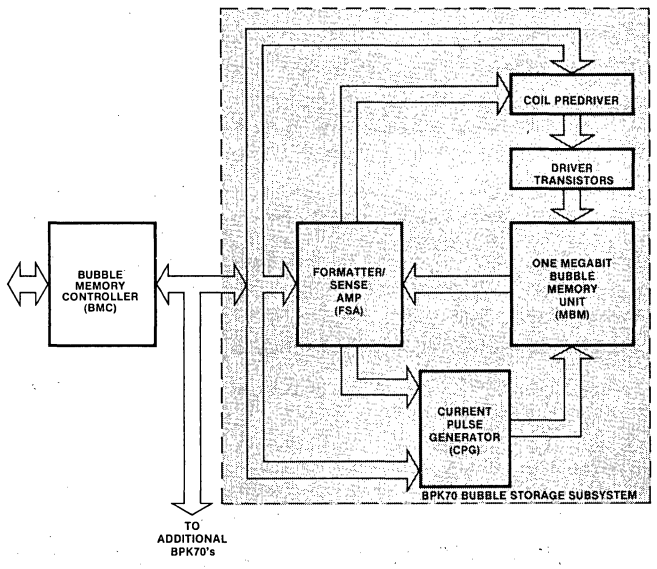


Figure 11. Minimum Magnetic Bubble Memory System, Shaded Portion is Bubble Subsystem

SUPPORT CHIPS

Five semiconductor integrated circuits are necessary to support each bubble chip. These components are described in some detail in the following paragraphs. In addition, each bubble memory system requires a controller, a separate integrated circuit described later.

Formatter/Sense Amplifier (FSA)

Serial data to be stored in or read from the bubble memory passes through the FSA. The FSA keeps track of which loops in the bubble memory are spares, executes the error correction coding and decoding if it is implemented, and shifts data to the bubble memory input tracks or from the output tracks, amplifying the output signals from the memory.

The FSA has a chip-select input, which is normally grounded (permanently enabled). However, each FSA drives the chip-select input of other circuits associated with the same bubble chip, so they are all enabled at the proper time.

Current Pulse Generator (CPG)

All signals except those that control the rotating field originate in the CPG. This device is the source of a current pulse that cuts a new bubble from the seed bubble whenever the FSA has a binary 1 to be stored. Later, when this bubble reaches the loop in which it is to reside, the CPG issues the signal that swaps it with the bubble or non-bubble previously stored in that location of the loop. When data is to be read, the bubble is replicated on the output track by still another signal from the CPG.

Coil Predriver (CPD)

Four digital signals (positive and negative versions of both X and Y waveforms) are sent to the CPD from the controller with appropriate durations and phases to control the rotating field that moves the bubbles in the memory. The CPD combines and inverts these to form eight pulsed outputs that are amplified in a separate transistor package to drive the coils surrounding the bubble chip with a triangular current waveform.

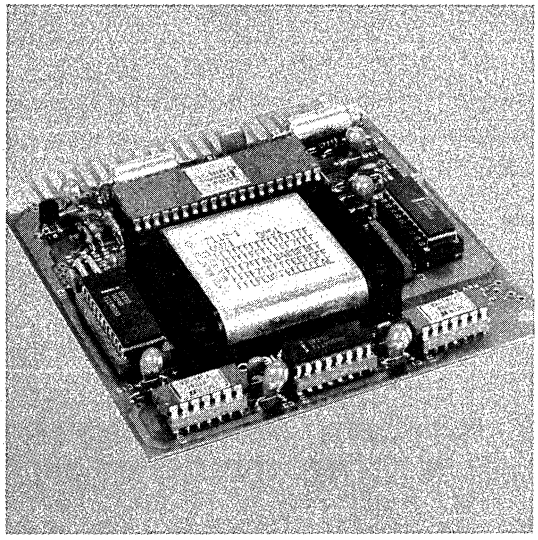


Photo 2. The Minimum Magnetic Bubble Memory System Including Controller

CONTROLLER

The bubble memory controller is the interface between the memory system and the equipment it serves. It converts serial data to parallel and parallel data to serial, and generates all timing signals required by the other support circuits in the bubble memory system. It can control up to eight bubble subsystems, for a total of a megabyte of memory.

Internal storage on the controller includes a first-in-first-out buffer with a capacity of 40 bytes. This buffer stores data to be sent serially to the FSA or just received from the FSA on one side, and data to or from the parallel bus served by the bubble memory on the other. It also serves as a speed matching device between the user at the parallel bus and the FSA which must transfer data to and from the bubble device at exactly the rotating field ratio in each channnel.

GLOSSARY

Bias field—a magnetic field perpendicular to a magnetic thin film that maintains conditions necessary to support formation of magnetic bubbles in the film.

Boot loop—in a magnetic bubble memory with serial/parallel/serial architecture and redundant loops, a special loop containing information that identifies which loops are active and which are inactive, as determined by factory test. This loop also contains the information necessary to synchronize the bubble memory page locations with the controller after power up.

Bubble, magnetic—a cylindrical magnetic domain in a thin film of orthoferrite or garnet. When viewed from above, the cylindrical shape appears spherical, hence the name "bubble." A bubble represents a binary 1 in most magnetic bubble memories.

Chevron—one of many possible shapes for a magnetic pattern deposited on a thin film to steer bubbles in a desired direction. Asymmetric chevrons are used in Intel memories.

Detector—a means of distinguishing bubbles from non-bubbles (1s from 0s) when a word is read from the bubble memory.

Domain, magnetic—a small region of a ferromagnetic substance that contains many similarly oriented atoms, so that the region as a whole is magnetized in that direction.

E²PROM—an acronym for electrically erasable programmable read-only memory, which is a memory component that, though nominally read-only, can accept changes to any work stored in it by electrical means, but at substantially slower speed than that at which stored words are read.

EPROM—an acronym for erasable programmable read-only memory, which is a memory component that, though nominally read-only, can be completely erased, usually by exposure to ultraviolet light, and then reloaded with new information, but at substantially slower speed than that at which stored words are read.

Ferrite—any of several compounds of iron, oxygen, and another metal, with magnetic properties that are useful in certain microwave applications and in computer memories.

Field, magnetic—a region of space in which a magnetic force exists and can be measured.

Garnet—a naturally occurring silicate mineral sometimes used in jewelry. Synthetic garnets with the same crystal structure can be made of oxides of iron and yttrium or one of the rare earths. Garnet is the preferred material for the thin magnetic film in a bubble memory.

Input track—a series of magnetic metal patterns that control the movement of bubbles in a thin film, and thereby lead them from a bubble generator toward one or more storage patterns.

Ion implantation—a process involving accelerators, similar to the machines used by nuclear physicists, for depositing dopants on and just below the surface of an electronic component; used to alter the physical properties of the material.

Latency—a delay between a request to read or write data in a memory and the actual beginning of the operation, imposed by a requirement for the address to move physically (but not necessarily mechanically) to a point where the data transfer can take place.

Magnetization vector—an expression of the magnitude and direction of a magnetic field at a point in space.

Magnetoresistance—a change in electrical resistance due to the presence of a magnetic field.

Major loop—in a magnetic bubble memory, an endless loop containing a bubble generator, a bubble detector, and/or a bubble annihilator, through which data is read or written, and which transfers bubbles to or from one or more minor loops (q.v.) in which they are stored. In some designs the major loop is not endless, and all bubbles not transferred out of it collapse when they reach the end. In these cases the major loop becomes an input or output track (q.v.).

Minor loop—in a magnetic bubble memory, an endless loop in which bubbles are stored, having been transferred into it from a major loop or input track (q.v.) and accessible by transfer into a major loop or output track (q.v.).

Non-Volatility—a property of some memory technologies that retains the integrity of stored data when power is turned off.

Orthoferrite—one of several oxides of iron and either yttrium or a rare earth. The molecular structure is simpler than that of garnet (q.v.). Orthoferrites were the first materials used for the thin magnetic film in experimental bubble memories, but have yielded to garnets, which have more desirable properties—notably ease of preparation as thin films with the necessary magnetic characteristics.

Output track—a series of magnetic metal patterns that control the movement of bubbles in a thin film, and thereby lead them from one or more storage patterns toward a bubble detector.

Permalloy—an easily magnetized and demagnetized alloy of nickel and iron.

PROM—acronym for programmable read-only memory—a read-only memory whose content is loaded by the user after delivery, as opposed to read-only memories whose content is fixed during manufacture. Once loaded, the data in a PROM is not alterable.

Pseudo-random access—a property of some memory technologies in which the time of access to blocks of stored data is largely (but not necessarily entirely) independent of the position of the block in the storage medium, but in which the time of access to bits, words or other entities depends on the position of that entity within the block.

Random access—a property of some memory technologies in which the time of access to any stored bit, word, or other entity is wholly independent of that entity's position in the storage medium.

Saturation—a state of magnetization of a material by a field such that, if the field is increased, the magnetization of the material does not increase and the magnetic flux density increases in proportion to the field (having increased much more rapidly in weaker fields).

Seed—a permanent bubble in a magnetic bubble memory, from which other bubbles are cut to represent stored binary 1s.

Serial access—a property of some memory technologies in which the time of access to any stored bit, word, or other entity depends strongly on that entity's position in the storage medium.

Thin film—any film of material deposited on a suitable substrate to take advantage of the material's special properties when dispersed as a film. Thickness ranges usually from about 10-9 to 10-6 meter, and occasionally to 10-5 meter or more, as in bubble memories.

T-I bar—one of several possible shapes for a magnetic pattern deposited on a thin film to steer bubbles in a desired direction, consisting of shapes like the letter T and the letter I alternately along a track. This pattern was used extensively in early bubble memory designs, but is no longer generally employed.

intel

APPLICATION NOTE

AP-119

December 1982

Microprocessor Interface for the BPK 72

Paul Wells
Application Engineer

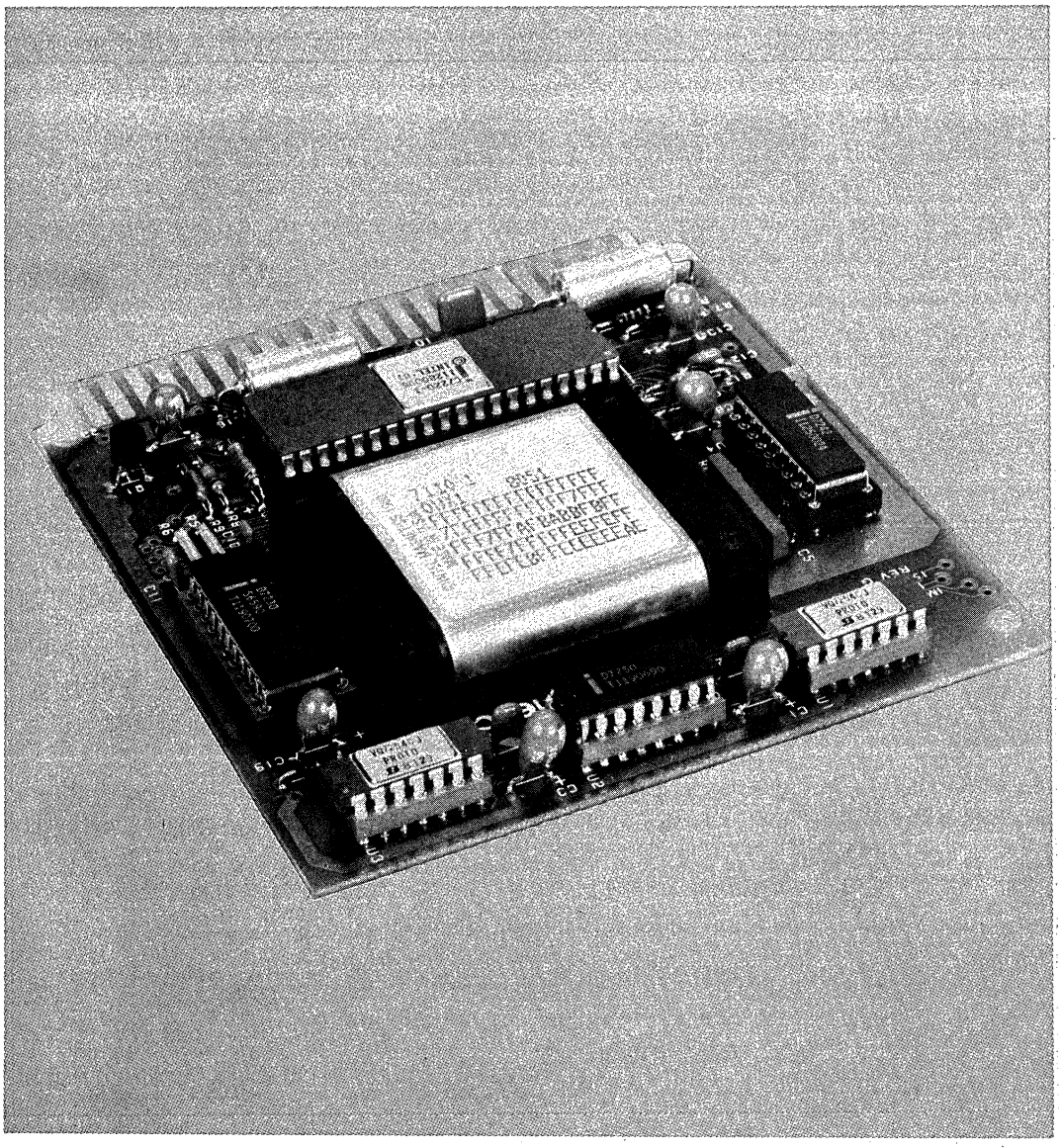
ORDER NUMBER: 210367-002

INTRODUCTION

To date, a major obstacle in the implementation of bubble memories in systems has been the inherently complex control requirements imposed by the bubble memory devices themselves. With the advent of Intel's BPK 72 bubble memory prototype kit, a design engineer can immediately realize the benefits of non-volatility, form factor, density and reliability without the complex control concerns. This application note provides additional background on the operating characteristics of the BPK 72 and is intended to further ease the design effort required in the implementation of bubble memory systems.

OVERVIEW

This application note provides an example of Bubble Memory system implementation using the BPK 72 and an Intel 8086 microprocessor. Before looking at this example, some explanation is necessary as to how this implementation was attained and how a user can take advantage of the principles involved.



As an introduction, the basic architecture of the BPK 72 is reviewed followed by an explanation of the operating characteristics of the BPK 72 kit as a whole and of the 7220 Bubble Memory Controller. Once the building blocks are in place, a detailed account of the implementation of a bubble memory kit is offered. The final section, which involves the actual implementation of the BPK 72 and an SDK-86, completes the application note.

BUBBLE SYSTEM OVERVIEW

A block diagram of the Intel Magnetics 128K-byte system is shown in Figure 1. The support circuitry used with one 7110 magnetic bubble memory (MBM) in the BPK 72 kit consists of the following 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 Bubble Memory Controller (BMC) completes the basic system.

The 7250 and the two 7254s supply the drive currents for the in-plane rotating magnetic field (X and Y coils) that move the magnetic bubbles within the MBM. The 7230 supplies the current pulses that generate the magnetic bubbles and transfer the bubbles into and out of the storage loops of the MBM.

The 7242 accepts signals from the bubble detectors in the MBM during read operations, buffers the signals and performs data formatting tasks that include the transparent handling of bootloop information. During write operations, the 7242 enables the current pulses of the 7230 that cause the bubbles to be generated in the 7110 MBM. Automatic error detection and correction of the data can be performed by the 7242.

The 7220 provides the 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 support circuitry.

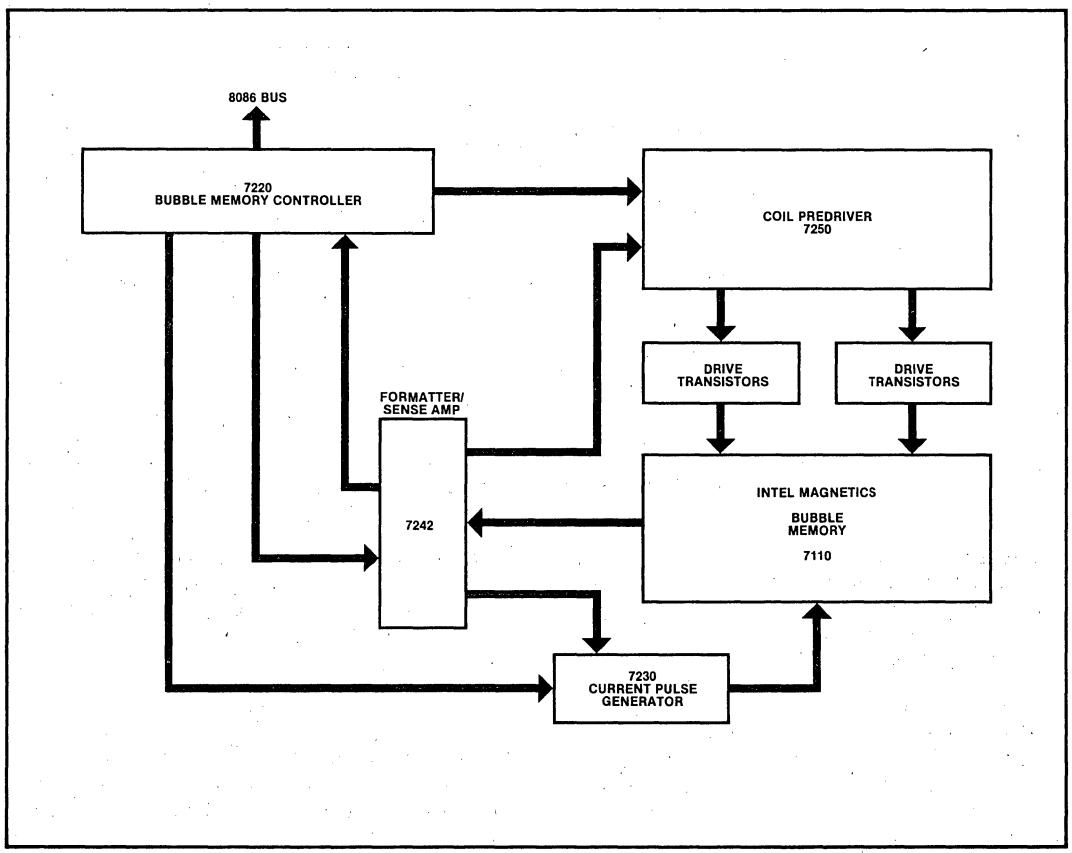


Figure 1. Block Diagram of the 128K Byte Magnetic Bubble Memory System

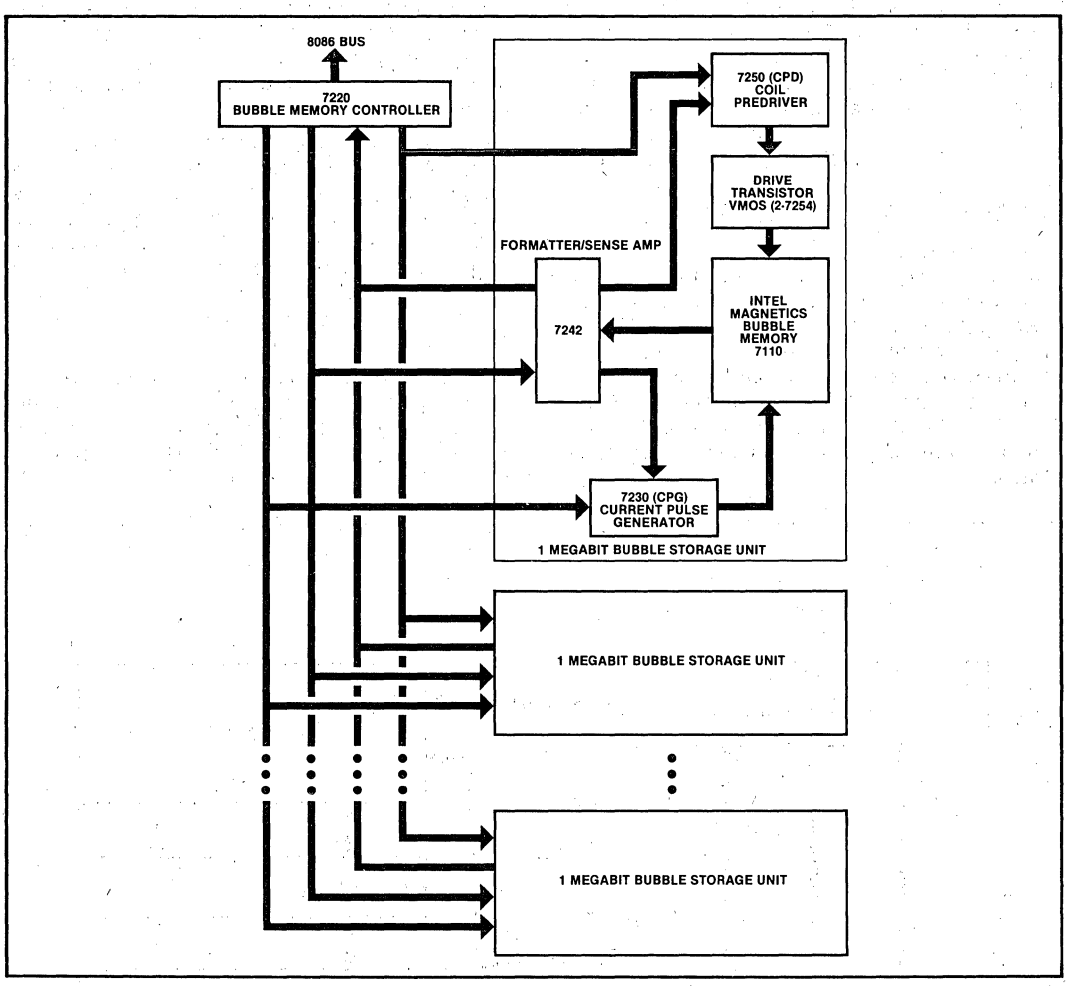


Figure 2. Bubble Memory System Expansion up to One Megabyte

Figure 2 shows how larger systems can be built from the basic components. A Bubble Storage Unit consists of one 128K-byte MBM and the five support chips shown. The components needed for one MBM cell are available as the BPK 70 kit. Larger systems can be constructed from the components supplied with one BPK 72 kit (which includes the 7220 controller) and one or more BPK 70 kits. For example, a one megabyte system can be assembled from one BPK 72 kit and seven BPK 70 kits. No additional TTL parts are required when building multibubble systems with up to eight MBMs.

One 7220 is capable of controlling up to eight Bubble Storage Units simultaneously. Larger systems can be configured with multiple 7220's and additional Bubble Storage Units.

Functional Organization of the 7110 Bubble Memory

The Intel Magnetics 7110 Bubble Memory utilizes a "major track/minor loop" architecture. With this architecture, if a binary 1 is to be written, a "seed bubble," always present in the 7110, is split in two. One bubble remains at the generator as the

seed, and the other is propagated down the input (major) track. If a 0 is to be written, the seed bubble is not duplicated. The data generated is sent down the input track, in serial, until it is aligned with the "swap" gates at the minor loops of the device. The new data is then swapped into the minor loops in parallel at the same time the old data is swapped out to the major track.

To read data from the 7110, data is rotated in the minor loops until it is positioned at the "replicate" gates opposite the output track. On receipt of a replicate signal, the data in the minor loops is duplicated by splitting the bubbles. The original data remains in the minor loops, and the duplicate data is clocked down the output track where the detector elements of the bubble memory operate to transform the presence or absence of a bubble into small electrical signals that are converted into digital '1' and '0' signals in the 7242 FSA.

With the 7110, the process of reading data from the minor loops by simultaneously splitting all of the bubbles in a page is known as "block replicate." The advantage of the block replicate architecture is that the data currently stored in the minor loops is not compromised during a read operation; the data to be read never leaves the minor loops. This architecture can be contrasted with earlier architectures that required the data to leave the minor loops, be detected and then returned to the minor loops. In the event of a power failure, bubble systems not utilizing the block replicate architecture could suffer a loss of data during a read operation; the data being sensed would not be returned from the major loop to the minor loops.

With the 7110 MBM, there are 2048 positions for the data within a minor loop. To move the bubbles in the MBM, a magnetic field is induced and rotated in the plane of the 7110. As the field is rotated 360 degrees, every bubble is moved ahead one position, and all of the bubbles maintain the same position relative to one another. All of the bubbles in similar positions in the loops are referred to as a "page."

By way of illustration, suppose the bubble is made of five minor loops (a,b,c,d,e) capable of holding nine pages of data (Table 1). During four 360 degree "rotations" of the in-plane magnetic field, the nine pages of data shift four positions (1.1, 1.2, 1.3, 1.4).

Table 1. 7110 Loop Operation

abcde	abcde	abcde	abcde
00000	00011	00000	00000
00011	00000	00000	11111
00000	00000	11111	00000
00000	11111	00000	00000
11111	00000	00000	00000
00000	00000	00000	10110*
00000	00000	10110*	00000
00000	10110*	00000	00011
10110*	00000	00011	00000
1.1	1.2	1.3	1.4

* = page zero

The 7110 MBM actually contains 320 minor loops, of which 272 must be good. The additional 48 loops provide 15% redundancy. This redundancy factor allows some of the loops in the 7110 to be bad while maintaining a completely functional one megabit device. A map of the good and bad loops is placed on the label of the 7110 and is also

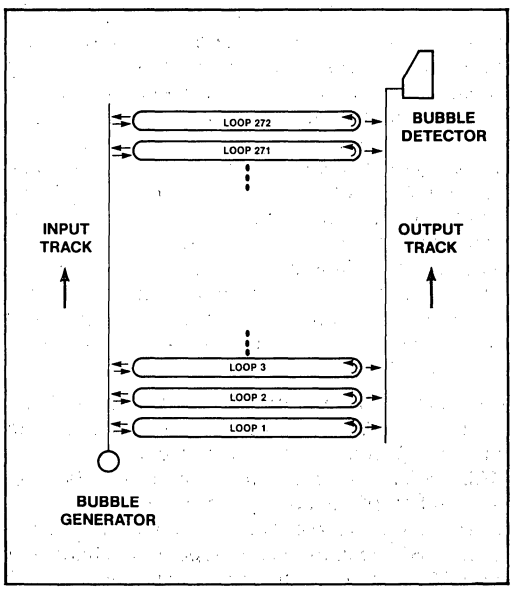


Figure 3. Functional Organization of the 7110

encoded and placed in the boot loop of the device as it is tested. This map, the bootloop, consists of forty bytes of data. Each good loop in the 7110 is represented by a one, each bad loop by a zero. When the system is initialized, the 7220 BMC reads the bootloop from the 7110 and decodes it. The bootloop is then automatically placed in the bootloop register of the 7242. The bootloop register serves as a working 'map' of the 7110 for read and write operations.

With the pages of data rotating around the minor loops, there must be a mechanism to orient the device and to assign a starting address to a page. The mechanism used to identify page zero involves the bootloop that resides on the 7110. Page zero (or address zero) is defined as the position of the 7110 after the bootloop has been read by the 7220 controller. Thus, each time the host CPU sends an "initialize" command, the bootloop is read by the 7220, and the 7110 is queued at page zero. From this point, any desired page in the bubble can be obtained by the controller.

Data Flow Within the Bubble Memory System

To better understand the relationship between the 7110 MBM and its support circuitry, the data flow within the bubble system during a read operation is examined. During the read operation, bubbles from the storage loops are replicated onto an output track and then moved to a detector within the MBM. All movements within the MBM occur under the influence of a rotating magnetic field; the number of rotations and the rotation timing are under the control of the 7220 BMC. 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 7242 Formatter/Sense Amplifier (FSA).

The data path between the 7110 MBM and the 7242 FSA consists of two channels (channel A and channel B) connected to the two halves of the MBM. When data is written, the bit stream is divided with half of the data going to each side of the MBM. During a read operation, data from each half 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 with the corresponding bit in the FSA bootloop register.

If an incoming data bit is found to be from a good loop (a corresponding "1" in the FSA bootloop register), it is stored in the FSA FIFO; otherwise, it is ignored. This process continues until both FSA FIFOs (channels A and B) are filled with 256 bits. Error detection and correction, if enabled by the user, is applied to each block of 256 bits at this point. If error correction is not enabled, 272 bits of data can be buffered in each FIFO.

As data leaves the 7242 FSA, the bit patterns buffered in each of the FSA FIFOs is interleaved and sent to the 7220 BMC in the form of a serial bit stream via a one-line bidirectional data bus (DIO line). In the 7220 BMC, the data undergoes a serial-to-parallel conversion and is assembled into bytes that are buffered in the 7220 FIFO. It is from this FIFO that the data is written onto the user interface.

COMMUNICATING WITH THE 7220

The CPU views the 7220 BMC as two input/output ports on the bus. When the least-significant bit of the address line is active (A0 = 1), the command/status port is selected. When the least-significant bit of the address line is inactive (A0 = 0), the bidirectional data port is selected. In order to define the operations on these ports, it is necessary to understand something of the internal organization of the 7220 Bubble Memory Controller.

For simplicity, the user need only view the 7220 as containing a 40-byte FIFO and a collection of 8-bit registers. The FIFO is a buffer through which data passes on its way from the 7242 Formatter/Sense Amplifier (FSA) 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 7220 controller and the controller interface to the FSAs.

The six 8-bit registers internal to the 7220 are loaded by the user prior to any operation of the bubble system and contain information regarding the operating mode of the 7220. Loading the 7220 registers before any commands are sent is similar to passing parameters to a subroutine prior to invocation, hence, the registers are often referred to as "parametric registers."

Data transferred between the CPU and the 7220 FIFO and parametric registers takes place over an 8-bit data port. The choice as to whether the data is destined for the FIFO or the parametric registers, however, is made through the command/status port. In one case, the actual commands that cause some operation to take place, such as a read or write, consist of a 4-bit code sent by the CPU to select one of 16 possible commands. This 4-bit code occupies the low-order nibble (bits 0, 1, 2, and 3) of the command byte. The command byte must also have bit 4 set to indicate to the 7220 that a command is being sent. In the

second case, another 4-bit code on the command port (bits 0, 1, 2, and 3) is used to select either one of the parametric registers or the 7220 FIFO. As shown in Table 2, if bit 4 of the command byte is set to zero, the value of the low-order nibble is taken to be a pointer value that specifies a parametric register or the 7220 FIFO. This pointer is referred to as the "Register Address Counter" (RAC).

Table 2. Command Port Function

FUNCTION	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Command	0	0	0	1	C	C	C	C
RAC	0	0	0	0	R	R	R	R

RAC values that may be sent out on the command port and the corresponding register names are illustrated in Table 3. The RAC points to, or selects, six unique registers and the 7220 FIFO. Once a RAC value is sent by the CPU to the 7220 via the command port, the next read or write operation to the data port transmits data to or receives data from the register addressed. Notice that the six registers have values that are in ascending order starting at 0AH and that the FIFO has a value of 0.

The reason for this ordering is due to the auto-incrementing feature of the RAC; once the first register is selected, each subsequent byte of data on the data port causes the RAC to be automatically incremented and to point to the next register in the sequence. Once the most-significant byte of the Address Register has been loaded, the RAC value automatically rolls over from 0FH to 0 and points to the 7220 FIFO. The system is now in position to transfer data to or from the FIFO without the user code explicitly pointing to the FIFO.

Table 3. Register Address Counter Assignments

Register Name	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Read/ Write
Utility Register	0	0	0	0	1	0	1	0	R/W
Block Length Register (LSB)	0	0	0	0	1	0	1	1	W
Block Length Register (MSB)	0	0	0	0	1	1	0	0	W
Enable Register	0	0	0	0	1	1	0	1	W
Address Register (LSB)	0	0	0	0	1	1	1	0	R/W
Address Register (MSB)	0	0	0	0	1	1	1	1	R/W
7220 FIFO	0	0	0	0	0	0	0	0	R/W

Once the FIFO has been selected, the RAC stops incrementing and continues to point to the FIFO until changed by the user software. This sequence minimizes the number of instructions necessary for a given transaction and aids in establishing a protocol to ensure that all of the necessary information is sent to the controller. The user, however, is not bound to follow this automatic sequence. Each parametric register may be selected and loaded in any order; specific registers may be updated where needed, but in each case, the host software must explicitly name the register to be loaded. Until a user is familiar with the bubble system, it is recommended that the auto-incrementing feature be used.

It is important to remember that once a command has been given to the 7220 BMC, the parametric registers must not be updated until the Status byte indicates that the operation is complete. The parametric registers are, in effect, working registers for the controller during the execution of a command. For example, during a Read or Write operation, the Block Length Register, which contains the terminal page count for the operation, is decremented by the 7220. Similarly, the Starting Address Register, which initially contains the starting page for an operation, is incremented by the controller as each page is transferred. Attempting to modify these registers during the operation of a command causes the block count and address to be incorrect.

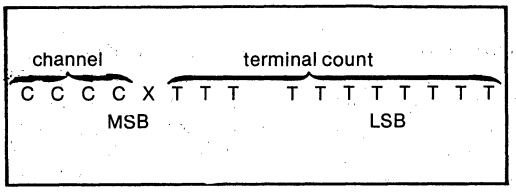
Addressing the Bubble Memory System

One of the interesting aspects of the Intel Bubble Memory System is its inherent addressing flexibility. The user may treat a 7220 BMC with eight

bubbles as a collection of 16K pages of 64 bytes each (addressing each bubble in turn) or as collection of 2K pages of 512 bytes each (addressing eight bubbles in parallel). Of course, there are a variety of configurations in between these two extremes, each dictated by the user's need for speed, power consumption, address space, and cost. Control over the configuration is achieved at run time via two of the parametric registers: the Block Length Register and the Starting Address Register.

The Block Length Register (BLR) is a 16-bit value divided into two fields: the "terminal count" field and the "channel" field. The bit configuration for the BLR is as follows:

Table 4. Block Length Register



The "terminal count" field ranges over eleven bits and defines the total number of pages requested for a read or write operation. With eleven bits in the field, a user may request from one to 2048 pages be transferred (eleven bits of zero indicate a 2048-page transfer). The width of the page is effectively defined in the "channel" field. This field specifies the number of FSA channels that are to be addressed. Recalling that each 7242 FSA has two channels to communicate with one 7110 bubble memory, the legal combinations in this field address one channel (one half of a 7110), two, four, eight, or 16 channels. These combinations translate into page sizes of 32, 64, 128, 256, or 512 bytes, respectively. (The one-channel mode of operation is usually reserved for diagnostic purposes, and examples of its use will be illustrated later.)

Table 5 shows the relationship between the "channel" field and the number of FSA channels selected. Notice that the channel field bits are encoded. A value of "0001" binary selects two FSA channels: 0 and 1.

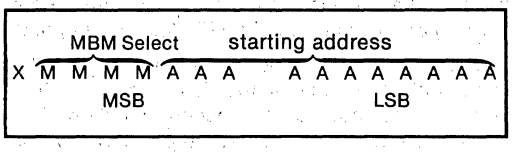
Table 5. FSA Channel Select

	Channel field (BLR MSB bits 7, 6, 5,4)				
	0000	0001	0010	0100	1000
Number of channels selected:	0	0,1	0,1,2,3	0 to 7	0 to F

Thus, a BLR value of "0001" in the high-order four bits selects one bubble through channels 0 and 1 Similarly, a BLR value of "0010" selects two bubbles in parallel with a page size of 128 bytes. This, however, is not the complete story. For example, a value of "0100" in the BLR selects four bubbles in parallel through channels 0 to 7. Suppose, that there are eight bubbles in the system and that the user desires to arrange the eight bubbles as two sets of four. The mechanism to communicate through channels 0 to 7 and channels 8 to F resides with the Address Register (AR).

The Address Register contains a 16-bit value divided into two fields: a "starting address" field of eleven bits and a "magnetic bubble memory (MBM) select" field of four bits.

Table 6. Starting Address Register



The eleven bits in the starting address field of the AR are set by the user to indicate to the 7220 BMC on which page of a bubble's 2048 pages the transfer is to start. For example, if a read operation is to start at page 1125 and is to continue for 16 pages, the starting address field contains 1125, and a value of 16 is placed in the terminal count field of the BLR. After each page is transferred, the starting address field is incremented and the terminal count is decremented by the controller.

Continuing with the example of two banks of four bubbles, notice in Table 7 that the MBM select field is needed to switch between the two banks. A value of "0000" in bits 3, 4, 5, and 6 of the high-order byte of the address register selects bank 0 or FSA channels 0 through 7; a value of "0001" selects bank 1 or FSA channels 8 through F. Each bank contains 2048 pages of 256 bytes.

To operate eight bubbles serially, a user needs only to specify a value of "0001" once in the channel field of the BLR and to begin with a value of "0000" in the MBM select field. As page 2048 is written in the first bubble, the AR, managed by the 7220 controller, rolls over to 0 and updates the MBM select field with no additional bit manipulation. In this case, the bubble system appears as 16K pages of 64 bytes each. Power consumption is one-eighth of that consumed by operating eight bubbles in parallel. However, the data rate is limited to the data rate of one bubble.

Table 7. FSA Channel Select/MBM Select

MBM SELECT AR MSB BITS (6, 5, 4, 3)	"CHANNEL FIELD" (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				

The Enable Register

The Enable register is the parametric register that defines the various modes of operation of the 7220 controller. The data transfer mode (polled, interrupt driven, or DMA operation) is selected by setting the appropriate bit in this register. Likewise, the type of error correction to be applied to the data is selected, based on the bits selected in this register.

While the function of each of the enable register fields is described in the BPK 72 manual, some of the finer points and implications are detailed here.

Note that it is possible to completely change the operating characteristics of the bubble system through software control. A system can go from the DMA mode with error correction enabled to a system operating in polled I/O with no error correction enabled by altering the value of the Enable register. Though most implementations will not take advantage of this degree of flexibility, there are cases where the Enable register is modified during system operation. For example, the normal interrupt and MFBTR bits can be modified between operations to change interrupt and read data rates, respectively. (If the error correction mode is changed, the CPU must issue an Initialize command to the 7220 controller).

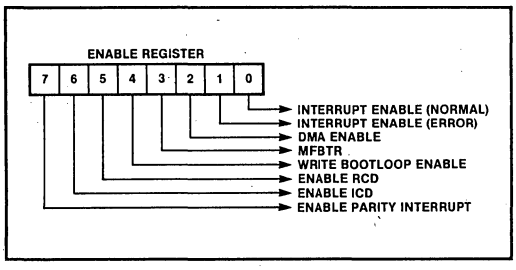


Figure 4. Enable Register Definition

The interrupt capabilties of the 7220 are reflected in the NORMAL, PARITY and ERROR INTERRUPT bits of the ENABLE register byte. The 7220 controller is capable of issuing interrupts to a CPU at the normal completion of an operation, if a parity error is encountered between the 7220 controller and the CPU, or if a data transfer error is found by the 7242 FSA. Any (or all) of these conditions are selected via the Enable register byte, and any resultant interrupts are sent to the CPU via a single INT line. At this point, the software must examine the status register to determine the cause of the interrupt. (An additional interrupt, the FIFO half-full interrupt, is issued on the DRQ pin and is not controlled by the Enable byte).

One of the more difficult aspects of the ENABLE register byte to understand is the operation of the ERROR INTERRUPT bit (bit 2). This bit normally is not used alone, but in conjunction with the ENABLE RCD and ENABLE ICD bits of this register. These three bits form combinations that gate selected 7242 error conditions to the CPU. For example, if, while operating under error correction, a user does not wish to be bothered by an interrupt that indicates an error has been corrected automatically by the system, a specific pattern of these three bits would be selected (100 or 010 from Table 8). If the user wishes to be notified of all errors, another pattern would be selected (011 or 101).

Table 8. Error Correction Combinations

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

The purpose of the ERROR INTERRUPT bit is not to enable or disable error interrupts, but rather to aid in selecting the type of error interrupt received by the CPU. If any type of error correction is selected, interrupts are enabled automatically.

The ENABLE RCD (read corrected data) bit causes the error correction algorithm to be applied to the data being transferred from the 7110 MBM in an almost transparent manner. The RCD bit allows the 7220 controller to send its own commands to the 7242 FSA. These commands cause the FSA to automatically correct and transfer to the controller, any data that is found to be in error and that is considered correctable.

With only the RCD bit on, no interrupt is generated if a correctable error is found. However, the user is informed that a correctable error was encountered and corrected during the data transfer via the 7220 status byte at the end of the operation. Uncorrectable and timing errors cause an interrupt to which the CPU must respond. With both the RCD bit and ERROR INTERRUPT bit on, the CPU is notified via an interrupt whenever a correctable, uncorrectable or timing error is encountered.

The RCD mode of operation is suitable for transfers where a GO/NO GO termination is sufficient. For example, when loading executable code from the bubble to RAM, it is necessary to know that the transfer was good (with errors corrected) or aborted due to an uncorrectable error.

A retry of an uncorrectable page of data is accomplished by sending another Read command without modifying the parametric registers. It may be the case that the errors encountered were soft (read) errors that may not be present on a retry. Thus, what may have been detected as an uncorrectable error, may become a correctable error (or simply vanish) on a subsequent read of the offending page. In this case, the error correction ability of the system corrects the errors automatically without additional user intervention.

The advantage of the RCD mode of operation is that error correction can be applied transparently to the CPU except for uncorrectable conditions. The disadvantage is that a page of uncorrectable data is passed to the controller before the interrupt is sent. The software must have the ability to clear the 7220 FIFO prior to rereading the offending page from the bubble.

If a given page continues to show up as having a correctable error after a number of retrys, it is up to the user's protocol to determine the action to be taken. One protocol suitable for handling errors involves "scrubbing" the data. Suppose a page appears with an error and, on retry, the error is still present. If the error is correctable, the data should be corrected and written back to the bubble and then read back into RAM. The probability of encountering an uncorrectable error after the first retry is 1 in 10^{16}. Data scrubbing after one retry maintains this level of reliability.

The ENABLE ICD (internally correct data) bit also enables the error correction capability of the bubble system, but allows a slightly different interaction between the 7220 controller and the 7242 FSA than defined for the RCD mode. Error interrupt conditions are the same as defined for RCD operation. With the ICD bit on, correctable errors are handled automatically, but the operation halts for uncorrectable or timing errors. With both the ICD and ERROR INTERRUPT bits on, the operation halts for correctable, uncorrectable or timing errors. The ICD mode differs from the RCD mode in that when an operation halts due to an error, the offending page is held in the 7242 FSA and is not automatically transferred to the 7220 FIFO. Though the difference is subtle, the ICD mode of operation allows more flexibility in error logging and recovery. With data held in the 7242, the number of the bad page can be read for logging purposes, and the data can be recycled through the error correction network or reread from the bubble repeatedly. When the CPU is interrupted due to an error in the ICD mode, the user must look at the 7220 status byte to determine the type of error encountered. If the error is correctable, the user's software sends a Read Corrected Data command (0CH) to the controller. This command causes the controller to issue it's own commands to the 7242 to correct the error and to transfer the data to the 7220 FIFO. (Recall this action is done automatically when the RCD mode is selected; uncorrectable errors can be handled as described above).

As an example of how the ICD mode can be utilized, suppose that during a data transfer in the RCD mode, a correctable error consistently occurs. The

error, of course, is automatically handled by the 7242, and the only indication that an error had been corrected is through the status byte at the end of the transfer. There is no information as to how many or in what page the error or errors appear. One way to diagnose the problem is to reread the entire data block in the ICD mode with the ERROR INTERRUPT bit on. The transfer stops at the appearance of any error, and the data remains in the 7242. The page number of the error can be found by reading the Address Register since this register is incremented automatically after each page is read if no error is detected.

The user should then issue an RCD command to the 7220 to allow the page to be corrected and transferred to the 7220. Once the transfer is complete, the enable register again is changed to disable all error correction, and the 7220 is reinitialized. The entire block is read again and compared with the corrected version. (Error correction bits are appended to the data and can be ignored.) If a bad loop is suspected, the bad loop location could be calculated and the bootloop modified.

It is unlikely that repeated correctable errors are sufficient motivation to modify the bootloop. Repeated uncorrectable errors, however, at the same location, might be sufficient reason. Note that modifying the bootloop is an extreme measure and should only be performed as a last resort and only if justified by test data.

The Status Register

The 7220's 8-bit Status register is accessed by reading the Command port (A0 = 1). This register provides information regarding error conditions, the termination of commands, and the readiness of the controller to transfer data or accept new commands.

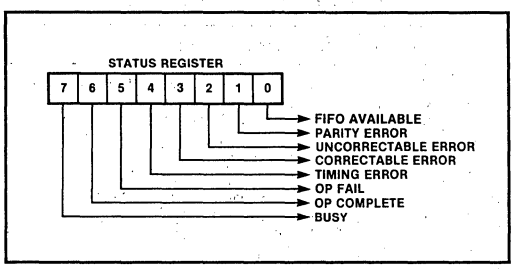


Figure 5. Status Register Definition

Values for the Uncorrectable Error and Correctable Error fields are generated when error correction is utilized as previously defined. The PARITY ERROR bit is set when a parity error is encountered on data sent to the controller on the D_0-D_7 lines. The TIMING ERROR bit is set for a number of conditions. The most frequent cause of a timing error is when the CPU fails to keep up with the rate at which the controller is filling or emptying the FIFO (an overflow or underflow condition). With one bubble in the system and the MFBTR bit of the Enable byte set to one, the controller moves data to or from the FIFO at a rate of about one byte every 80 microseconds. With eight bubbles operating in parallel, the rate is about one byte every 10 microseconds. (With the MFBTR bit set to 0, the data rate on a one page transfer or the last page of a multipage transfer is four times these rates.) Once a Read or Write command is issued, if the CPU cannot meet these transfer requirements, a timing error results.

Another way in which a timing error occurs is when the proper number of bits is not set in the bootloop register of the 7242 FSA. The 7242 must have 272 loops active to operate properly (270 with error correction enabled). If a mistake is made either when the bootloop of the 7110 is written or if the bootloop register is loaded incorrectly from RAM by the user, a timing error results. A timing error also occurs if the Write Bootloop command is issued to the 7220 controller and the WRITE BOOTLOOP ENABLE bit of the Enable byte is not on. Finally, a timing error is generated if the bootloop synch code is not found when a Read Bootloop or Initialize command is issued.

The OP FAIL and OP COMPLETE bits of the status register simply indicate the state of an operation after a command is executed. If an operation fails (OP FAIL = 1), the cause can be determined by looking at the other error bits of the status byte. When an operation (command) terminates successfully, the OP COMPLETE bit is set, and the status register shows a 40H.

The FIFO AVAILABLE bit of the status byte is more complex than the other bits since its meaning can change depending on the type of operation being performed as outlined below.

From an operational point of view, the FIFO AVAILABLE bit acts as a gate for the FIFO handling software. During a write operation, if the FIFO bit is set (1), there is room for more data; if the FIFO bit is clear (0), the FIFO is full. During a read operation, if the FIFO bit is set, data has been placed in the FIFO by the controller; if it is clear, the FIFO is empty.

Table 9. FIFO Available Bit Semantics

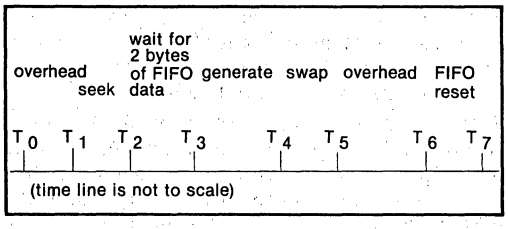
FIFO AVAIL BIT	BUSY = 1 & writing	BUSY = 1 & reading	BUSY = 0 & reading
1	room for data	data avail.	data avail.
0	no room for data	no data	no data

Note that it is possible to complete an operation with data still remaining in the FIFO (indicated by a 41H status value). This condition is quite legal; it is up to the software to remove the data or to issue a FIFO RESET command.

The BUSY bit indicates when the controller is in the process of executing a command. When a command is sent, the BUSY bit goes active within a few microseconds after the command is received and remains active until the operation either completes or fails. It is important to note that the BUSY bit remains active until all other bits in the status byte have been set. Thus it is possible to see logically-exclusive conditions such as BUSY and OP COMPLETE at the same time. The key to interpreting the status byte is to consider the status byte valid only after the BUSY bit returns to an inactive level. The single exception to this rule is the FIFO AVAILABLE bit.

The action of the controller during a write operation is one of the more complex sequences and serves as a good illustration of the behavior of the BUSY and FIFO AVAILABLE bits. Suppose a Write command is sent to transfer an arbitrary number of pages. Table 10 shows the activity of the controller at various steps in the sequence.

Table 10. Stages of a Write Command



Before the Write command is sent, the FIFO is in a general-purpose mode and remains in this mode until T_2. When the command is sent at T_0, the BUSY bit is low and, in fact, the BUSY bit must be low in order for the controller to accept a new command (except Abort). Sometime between T_0 and T_1, the BUSY bit goes high. Thus, between T_1 and T_2, the status byte will be 80H.

At T_2, the FIFO is internally placed in the "write mode," and FIFO AVAILABLE changes meaning from "FIFO has data" to "FIFO has room". For proper operation, the FIFO must be empty prior to issuing the WRITE command. This condition can be guaranteed by using the FIFO Reset command. Assuming the FIFO is empty, at T_2 the status byte changes from 80H to 81H. The status byte remains at 81H until T_6 (unless the CPU is able to fill the FIFO in which case, the FIFO AVAILABLE bit toggles between 0 and 1).

At T_7 (the completion of the command), the status byte should be 40H if the CPU did not load data between T_6 and T_7. If data was loaded during this interval, the status value is 41H.

Notice that if the FIFO contains data when the Write command is sent, the CPU can, by mistake, overflow the FIFO during the "seek" portion of the command. This condition results from the FIFO AVAILABLE bit being a "1" due to data present in the FIFO, not because there is room in the FIFO. While the following diagnostic routines take advantage of the "preloading" ability of the FIFO, the examples of operational software at the end of this application note do not preload the FIFO.

7220 Commands

The 7220 command set consists of 16 commands identified by a 4-bit command code. The function of most of the commands is obvious from the command name (e.g., Initialize, Abort, Read, Write). These commands are adequately described in the BPK 72 manual. There are, however, some commands and protocols that merit additional discussion (specific examples are covered later in this document).

Table 11. 7220 Commands

D3	D2	D2	D1	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

In general, all commands sent to the 7220 controller must be preceded by the setting of the parametric registers. While there are some exceptions as with the Abort command, it is usually necessary to supply operating information to the controller via the parametric registers prior to issuing any command. Since many initial problems stem from failing to load the registers prior to issuing commands, the user software should never assume that the regsiters contain valid data.

After the bubble system has been powered up, the 7220 controller inhibits (or ignores) all commands except an Initialize or Abort command. One of these commands must be sent prior to issuing any other command. Normally, the first command issued after loading the parametric registers is the Initialize command. This complex command reads and decodes the bootloop information from each bubble in the system and places this information in the bootloop register of the corresponding 7242 FSA. Pointers internal to the 7220 automatically are prepared for normal operation. As described later, the combination of the Abort, MBM Purge and Write Bootloop Register commands is functionally similar to the Initialize command. (The only time the MBM Purge command is used is in conjunction with the Abort command).

Once the system has been initialized, the remainder of the command set can be selected. Assuming, for example, that a Read command is to be executed, the user selects the page number and length of the transfer via the parametric regisiters and then issues the Read command. If the system uses the polled mode, the CPU reads the status register and waits for the BUSY bit to go active and then for the FIFO READY bit to indicate that data is being sent to the FIFO. Data can be taken from the FIFO until the FIFO READY bit goes inactive.

If the page selected for the read operation is not in position to be read (i.e., the page is not at the replicate gates), additional time is required to execute the Read command as the proper page is rotated into position. In systems where faster response is desired, the Read Seek command can be used to place the page into position in order to free the CPU to perform other tasks. Once the page is in position, approximately eight milliseconds are required before the data is available to the CPU. This latency only occurs on the first page of a multipage transfer. Similarly, when a page is not in a position to be written, Write Seek can be used to position the page at the swap gates.

If there is any doubt regarding the state of the FIFO prior to a read or write operation, the user should issue a FIFO Reset command in order to clear the 7220's FIFO counter before initiating the data transfer. If a prior transfer is stopped with data remaining in the FIFO or if the FIFO is partially filled, the 7220's internal FIFO counter is not zero, and there is a danger that the subsequent transfer count may be incorrect. If the FIFO is reset properly, execution of a FIFO Reset command is redundant.

Although the 7220 FIFO may be treated as a 40-byte RAM buffer, the temptation to "pre-load" the FIFO with 40 bytes of data and then to issue a Write command should be avoided due to the danger of overflowing the FIFO. Prior to issuing a Write command, a FIFO Reset command should be sent, and the parametric registers should be loaded. Following the Write command, the CPU should monitor the status byte and wait for the BUSY and FIFO AVAILABLE bits to go active. When this status condition occurs, the user software should then send the proper number of bytes to the 7220. The FIFO AVAILABLE bit of the status byte should be polled prior to sending each byte.

An exception to not preloading the FIFO is when a Write Bootloop, Write Bootloop Register, or Write Bootloop Register Masked command is used. Prior to issuing any of these commands, a FIFO Reset command must be sent before preloading the bootloop data into the FIFO. When one of the bootloop-related commands is issued, the 7220 controller immediately begins taking data from the FIFO. If the FIFO is not preloaded, incorrect data may be transferred. The operation of the normal Write command differs from the bootloop-related commands in that, after a Write command is issued, the 7220 waits for at least two bytes to be present in the FIFO before beginning to transfer data to the bubble.

If the FSA encounters an error condition during a read or write operation, the status of the FSA is reflected in the 7220 status byte. If the user system decodes the error and decides to continue, the error flags in the 7220 controller and FSA first must be cleared. To clear the status bytes, the software can issue an Initialize command. However, this command resets all of the current operating parameters in the 7220 controller. To continue processing without resetting the system, the software can use the Software Reset command. This command resets any error flags and clears the FIFO, but does not affect the parametric register fields that define the system configuration (e.g., number of FSA channels selected).

INSTALLING THE BPK 72 BUBBLE MEMORY KIT

This section examines the individual components of the Bubble Memory System and how each component can be analyzed. All elements of the bubble system need not be working before any meaningful diagnostics can be effected. In general, a user first establishes communication between the host CPU and the 7220 controller. Next, communication with the 7242 formatter/sense-amplifier is verified via the 7220 controller. Finally, the operation of the 7110 Bubble Memory is checked. The software that exercises each of these phases of implementation should be small, well-defined device drivers that can be controlled through a system monitor.

The procedures that follow are applicable to most startup problems. The procedures are organized in chronological fashion and address each step of the installation process as it would normally occur. Software drivers in 8086 assembly language are provided to illustrate the basic functions supported by the device drivers.

Powering Up for the First Time

With power removed from the IMB-72 board, insert all of the supporting integrated circuits with the exception of the 7110 Bubble Memory Module. Insert the "dummy module" included in the BPK 72 kit in place of the 7110. The dummy module is electrically equivalent to the 7110 module and allows the circuits of the BPK 72 kit to be tested without the possibility of damaging the bubble. With both the +5V and +12V power supplies turned off, insert the IMB 72 with the dummy module into the edge connector. As power is applied to the system, monitor the RESET.OUT/pin of the 7220 controller and verify that the signal goes from low to high after power is applied. The low-to-high transition indicates that the power-up sequence has been completed successfully.

Communicating With the 7220 Bubble Memory Controller

The first step in communicating with the 7220 is to write initial values to the parametric registers using the code sequence in Table 15. When the registers have been set, the code shown in Table 12 can be used to examine the 7220 status byte.

The status value returned in Table 12 should be 40H. The user should not continue until the proper status value can be obtained repeatedly after performing the power-up sequence. Reading back the correct status indicates that the host CPU and the 7220 are communicating and that the power-up sequence is being performed by the 7220.

Table 12. Reading 7220 Controller Status

```
RDSTAT:
  ; THIS PROGRAM READS THE 7220
  ; STATUS BYTE
  ; TO READ STATUS, THE HOST CPU MUST
  ; READ FROM THE 7220 WITH A0 = 1.
    IN      AL, 49H       ; COMMANDS/STATUS
                          ; PORT ADDRESS OF
                          ; 7220
    MOV     STATUS, AL    ; MOVE AL REGISTER
                          ; TO STATUS
    RET
```

Once the power-up sequence is complete and the 7220 status register has been read, the 7220 FIFO can be accessed. The software drivers that write and read the FIFO are shown in Tables 13 and 14. Notice that these code sequences do not send commmands to the 7220; only data is transferred to and from the controller. The purpose here is to test the bus interface and timing between the CPU and the 7220 controller. In this case, the 7220 FIFO is used as a general purpose RAM. Any data can be written to the FIFO, but it is best to use an easily indentifiable sequence (e.g., an incrementing pattern) for easy recognition.

Table 13. Writing the 7220 FIFO

```
WTFIFO:
  ; THIS PROGRAM WRITES 40 BYTES FOR
  ; MEMORY TO THE 7220 FIFO.
  ; DATA IS ASSUMED TO BE ATBUFADR.
    MOVE    SI,BUFADR     ; LOAD BUFFER
                          ; POINTER
    MOV     CX,40         ; LOAD COUNT
WRT1:
    LODSB                 ; PUT BYTE AT SI
                          ; INTO AL, AUTO INCR
                          ; SI
    OUT     48H, AL       ; OUTPUT BYTE TO
                          ; DATA PORT
    LOOP    WRT1          ; DECREMENT COUNT,
                          ; LOOP IF NOT 0
    RET
```

Once forty bytes have been written to the FIFO, the 7220 status byte should be read. The status value should be "41H" (indicating that data is in the FIFO). Other status values such as "parity error" can be ignored. While status values give some indication of the CPU-7220 interaction, the integrity of the data is more important here. If the data read back is not the same as the data sent, a fundamental timing and/or interface problem between the CPU and the 7220 is indicated.

To verify that data is being transmitted to the 7220, the code sequence shown in Table 14 can be used to read back the FIFO data into user RAM space for direct comparison with the original pattern.

Table 14. Reading the 7220 FIFO

```
RDFIFO:
  ; THE PROGRAM READS 40 BYTES FROM
  ; THE 7220 FIFO INTO MEMORY.
    MOV    DI, BUFADR  ; LOAD BUFFER AD-
                       ; DRESS INTO DI
    MOV    CX,40       ; LOAD COUNT INTO
                       ; CX
RD1:
    IN     AL,48H      ; INPUT FROM DATA
                       ; PORT
    STOSB              ; STORE AL AT ADDR
                       ; IN DI, AUTO INCR. DI
    LOOP   RD1         ; DECREMENT COUNT
                       ; IN CX, LOOP IF NOT 0
    RET
```

After reading the FIFO, the status byte should be read (a value of "40H" or "42H," indicating that the FIFO has no data, should be obtained). The user should not proceed until the FIFO can be written and read correctly and until the FIFO status indicates the amount of data in the FIFO (not empty or empty). These steps verify that the CPU can communicate with the 7220. Note that no data has been transferred to or from the 7242 Formatter/Sense Amplifier or the 7110 bubble device (or dummy module).

Communicating With the 7242 Formatter/Sense Amplifier

The next step in verifying the BPK 72 is to ensure that the 7220 is driving the 7242 Formatter/Sense Amplifier properly by first setting up the 7220 for interaction with the 7242 and then sending commands to the 7220 to exercise the 7242 functions that can be verified easily.

Under normal operating conditions an Initialize command is the second command sent to the system. However, the Initialize command assumes that the 7110 Bubble Memory is installed and attempts to read bootloop information. Since the dummy module is installed at this time, timing errors result from the attempted Initialize command. Although no harm results from using the Initialize command, an Abort command followed by an MBM-Purge command can be used in place of the Initialize command to eliminate timing errors. The Abort command is sent by executing the code sequence at label "CMND9" in Table 16. When Abort command execution is complete, the user should read the status byte and check for an op-complete indication (40H).

Table 15. Write Register Sequence for Two FSA Channels

```
WTREG2:;                        WRITE REGISTERS
; 2 FSA CHANNELS SELECTED.
; THIS IS USED FOR DEBUG TO WRITE/READ THE
; BOOTLOOP REGISTERS AND CHECK FOR MISSING SEEDS, ETC.
; THE FOLLOWING VALUES INTO THE 7220 REGISTERS
;       B = 01H     : 1 PAGE TRANSFER
;       C = 10H     : SELECT 2 CHANNELS (WHOLE BUBBLE)
;       D = 08H     : STANDARD TRANSFER RATE
;       E = 00H     : PAGE 0
;       F = 00H     : FIRST BUBBLE

    MOV     AL, 0BH         ; SELECT B REGISTER
    OUT     49H, AL
    MOV     AL, 01H         ; ONE PAGE TRANSFERS
    OUT     48H, AL
    MOV     AL, 10H         ; WHOLE BUBBLE (2 FSA CHANNELS)
    OUT     48H, AL
    MOV     AL, 08H         ; LOW FREQ
    OUT     48H, AL
    MOV     AL, 00H         ; START ADDRESS = 0000H
    OUT     48H, AL
    MOV     AL, 00H         ; FIRST BUBBLE
    OUT     48H, AL
    RET
```

Once the op-complete status is received, the MBM-Purge command is issued by executing the routine labeled "CMNDE" in Table 16. This command, as described in the BPK 72 manual, clears all of the controller registers, counters and address RAM (except the block length register), the NFC bits, the FSA present counter and the high-order four bits of the address register. After the command is complete, the user again should receive an operation complete indication on reading the status byte.

After the Abort and MBM-Purge commands are executed and is status verified, additional commands may be sent to the 7220 BMC. Since the purpose of this section is to verify the interaction of the 7242 and 7220, manually loading and reading the 7242 bootloop registers can be used for the verification. Two additional commands are required to load and read the bootloop registers: the Write Bootloop Register command and the Read Bootloop Register command. These commands transfer data between the 7242 bootloop registers and the 7220 FIFO. Since the ability to transfer data between user RAM and the 7220

Table 16. 7220 Controller Commands

```
CMNDS:          ; 7220 COMMANDS
    ; THESE 16 ROUTINES EACH SEND A SINGLE COMMAND TO THE 7220.
    ; FOR EXAMPLE, THE "INITIALIZE COMMAND" WILL WRITE 11H
    ; TO THE 7220 WITH A0 = 1. THESE ARE THE 7220 COMMANDS LISTED
    ; IN THE BPK-72 USERS MANUAL.
CMND0:
    MOV         AL, 10H         ; WRITE BOOTLOOP REGISTER MASKED COMMAND
    OUT         49H, AL
    RET
CMND1:
    MOV         AL, 11H         ; INITIALIZE COMMAND
    OUT         49H, AL
    RET
CMND2:
    MOV         AL, 12H         ; READ COMMAND
    OUT         49H, AL
    RET
CMND3:
    MOV         AL, 13H         ; WRITE COMMAND
    OUT         49H, AL
    RET
CMND4:
    MOV         AL, 14H         ; READ SEEK COMMAND.
    OUT         49H, AL
    RET
CMND5:
    MOV         AL, 15H         ; READ BOOTLOOP REGISTER COMMAND
    OUT         49H, AL
    RET
CMND6:
    MOV         AL, 16H         ; WRITE BOOTLOOP REGISTER COMMAND
    OUT         49H, AL
    RET
CMND7:
    MOV         AL, 17H         ; WRITE BOOTLOOP COMMAND
    OUT         49H, AL
    RET
CMND8:
    MOV         AL, 18H         ; READ FSA STATUS COMMAND
    OUT         49H, AL
    RET
CMND9:
    MOV         AL, 19H         ; ABORT COMMAND
    OUT         49H, AL
    RET
CMNDA:
    MOV         AL, 1AH         ; WRITE SEEK COMMAND.
    OUT         49H, AL
    RET
CMNDB:
```

Table 16. 7220 Controller Commands (cont.)

```
        MOV     AL, 1BH         ; READ BOOTLOOP COMMAND
        OUT     49H, AL
        RET
CMNDC:
        MOV     AL, 1CH         ; READ CORRECTED DATA COMMAND
        OUT     49H, AL
        RET
CMNDD:
        MOV     AL, 1DH         ; FIFO RESET COMMAND
        OUT     49H, AL
        RET
CMNDE:
        MOV     AL, 1EH         ; MBM PURGE COMMAND
        OUT     49H, AL
        RET
CMNDF:
        MOV     AL, 1FH         ; SOFTWARE RESET COMMAND
        OUT     49H, AL
        RET
```

FIFO has been verified previously, these two additional commands verify the system's ability to transfer between user RAM and the 7242 FSA.

The 7220 parametric registers must be loaded prior to sending the Write Bootloop Register command. The sequence of operations is important; loading the parametric registers destroys the first byte of data in the 7220 FIFO. If valid bootloop information is placed in the FIFO before the parametric registers are loaded, the first byte of bootloop register information is invalid. Accordingly, the sequence of operations must be as follows:

(1) load the 7220 parametric registers
(2) load bootloop data into the 7220 FIFO
(3) send the Write Bootloop Register command.

As a point of interest, if a user wishes to maintain the system bootloop in EPROM rather than to allow automatic handling by the system, the Initialize command would not be used and would be replaced by a sequence similar to the one described.

After the 7220 parametric registers are loaded, the CPU next must load the 7220 FIFO with 40 bytes of bootloop register data using the "write FIFO" sequence from Table 13. This sequence then is followed by the code sequence to issue the Write Bootloop Register command. The data pattern written to the bootloop register should be an easily identified sequence of bytes such as an incrementing pattern. Under operational conditions, the data written to the bootloop registers represents "loop map" information that is written on the label of the 7110 device. Under these test conditions, it only is necessary to ensure that the 40 bytes sent out are the same 40 bytes read back.

Once the Write Bootloop Register command has been sent, the status byte is read (when the BUSY bit goes low) and an operation-complete status is verified. Any parity error indication may be ignored. Valid status at this point indicates that communication with the 7242 has been established. To verify that the data has been transferred properly, the contents of the bootloop register are read into the 7220's FIFO. The CPU then must transfer the data to user RAM in order to compare the data with the original pattern. To read the bootloop register, it only is necessary to issue the Read Bootloop Register command. This command places the contents of the 7242's bootloop register into the 7220's FIFO. The user then must execute the "read FIFO" sequence from Table 14 in order to transfer the data from the 7220 FIFO to RAM. Comparing the loop map written into the bootloop register and the loop map read from the bootloop register should show the loop maps to be equal.

Installing the 7110 MBM

Reading and writing the 7110 bubble memory requires the application of specific control signals at the appropriate times within the read or write cycles. These control signals originate from the 7254 and 7230 integrated circuits and are generated under the control of the 7220 BMC. Prior to installing the 7110, the presence of the control signals should be verified. While it is unlikely that the 7110 can be seriously damaged, it is possible for the "seeds" and bootloop established at the factory to be lost if there are problems with the 7254 or 7330 control signals and, if lost, would require additional steps on the part of the user to regenerate the seeds and bootloop data. With the dummy module installed, the required control signals can be verified directly on the bubble socket, and the possibility of damaging the bubble can be avoided.

The first control signal waveform to check is the coil drive on pins 9, 10, 11, and 12 of the 7110 socket. The drive current can be verified by ensuring that the voltage waveform on these pins (or on pins 1 and 7 of the 7254) conforms to Figure 6A when the drive field is being rotated. To rotate the drive field, the following code sequence can be used:

1. Write the parametric registers.
2. Send the Read command.

Next, the "cut and transfer" pulses generated during a read operation should be checked. The waveforms on pins 2 and 3 of the 7110 socket (REPLICATE.A and REPLICATE.B), should appear as shown in Figure 6B.

The cut and transfer pulses that occur during a write operation should now be verified. The waveforms on pins 7 and 8 of the 7110 socket (GENERATE. A and GENERATE. B) should appear as shown in Figure 6C. Since a write operation is required, a new code sequence must be used for this test:

1. Write the parametric registers.
2. Write data (any patten) to the FIFO.
3. Send the Write command.

bootloop register of the 7242 first must be loaded to allow data to be written. A Write Bootloop Register Masked command can be used to write a bootloop register pattern of all ones; it is only necessary to write the bootloop register once.

Finally, the SWAP pin is tested for proper operation during a write operation. The waveforms on pins 13 and 14 of the 7110 (SWAP.A and SWAP.B) should appear as shown in Figure 6D. The code sequence described for a write operation may be used.

One additional check of the system should be made prior to installing the 7110 device to determine if valid status values are received after a Read or Write command is issued to the 7220 BMC. Since the bubble is not yet installed, no data actually is transferred; the system should, however, execute the Read or Write command, and valid status should be received. Since a new command cannot be issued to the 7220 while a command is in progress, an Abort command is sent to cancel any command that may be pending from the last test performed. Next, a FIFO Reset command is sent to clear any data remaining in the FIFO. The status byte received should indicate an OP-COMPLETE and FIFO AVAILABLE status condition. The 7220 now is ready to execute a Read or Write command.

First, the 7220 parametric registers are loaded using the modified "diagnostic" driver shown in Table 17. This routine selects one FSA channel (half of a bubble) and, with ECC disabled, requires the loading of only 34 bytes in the 7220 FIFO. By limiting the FIFO to less than 40 bytes, FIFO underflow/overflow conditions are eliminated, and timing errors are avoided in the status byte. After, the 7220 FIFO is preloaded with 34 bytes of data (any pattern), a Write command is issued to the 7220 BMC. The 7220 status value received following command execution should reflect OP-COMPLETE since the 7220 transferred the data from its FIFO to the 7242 and executed the Write command as though the bubble were in place.

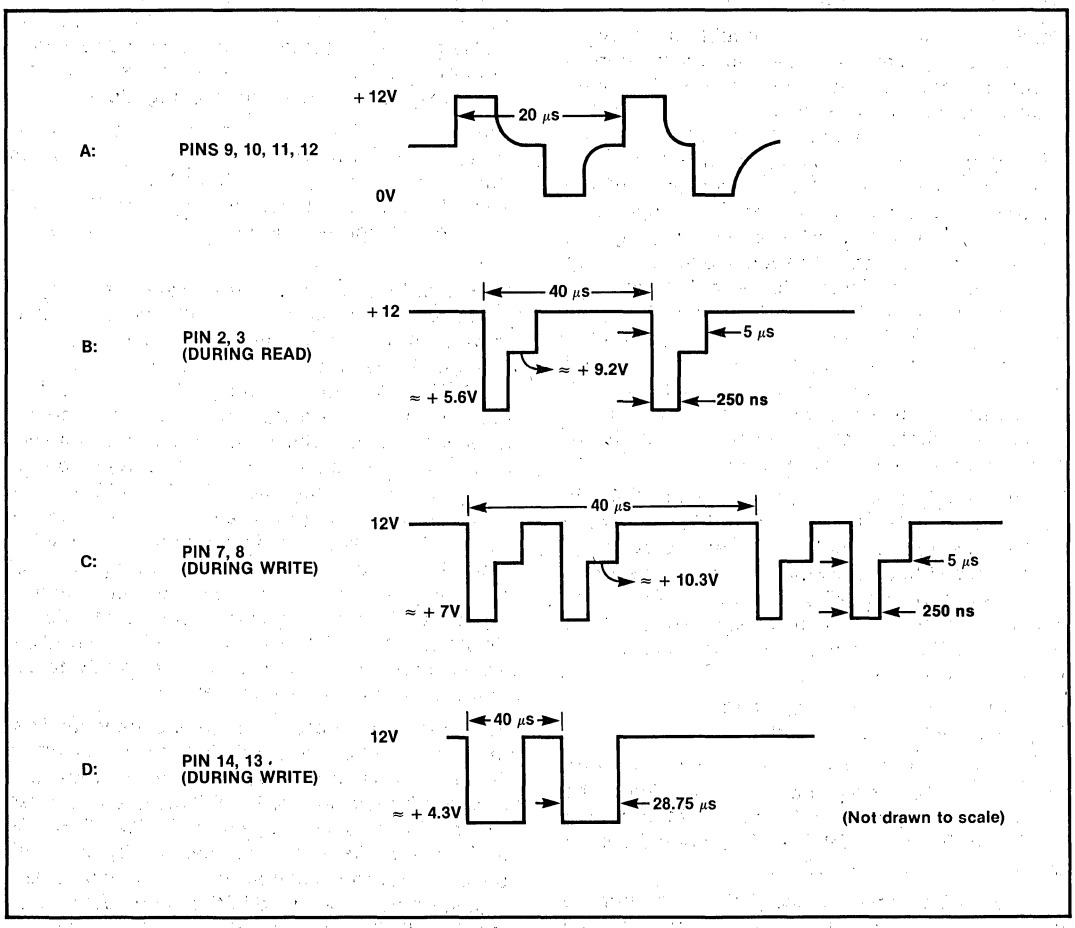


Figure 6. Control Signal Waveforms

To test the system in the read mode, the 7220 parametric registers are reloaded and a Read command is issued to the 7220. The user software must now read 34 bytes of "data" from the 7220's FIFO. Note that the data read will consist of all zeroes since no bubble is in place.

When the system completes all of the previous tests successfully, the 7110 bubble memory device may be inserted. Before proceeding, REMOVE POWER FROM THE SYSTEM.

Installing the 7110 is no different from installing any other device. Remove the dummy module in the 7110 socket and insert the 7110 Bubble Memory. Note that the 7110 is keyed to prevent the device from being inserted incorrectly. When power is applied, the system should execute its power-up sequence as described for the dummy module, and the 7220 status byte should return OP-COMPLETE after the parametric registers have been loaded.

Table 17. Write Register Sequence for One FSA Channel

```
WTREG1:;            WRITE REGISTERS (ONE HALF BUBBLE)
    ; THIS PROGRAM WRITES THE 7220 REGISTERS "B" THROUGH "F".
    ; DIAGNOSTIC ROUTINE WITH ONE FSA CHANNEL SELECTED
    ; THE FOLLOWING VALUES ARE WRITTEN TO THE 7220 REGISTERS.
    ;                B = 01H  :        1 PAGE TRANSFER
    ;                C = 00H  :        SELECT 1 CHANNEL (HALF BUBBLE)
    ;                D = 08H  :        LOW FREQ
    ;                E = 00H  :        PAGE 0
    ;                F = 00H  :        FIRST BUBBLE
      MOV     AL, 0BH       ; SET REGISTER ADDRESS COUNTER (RAC) TO B REGISTER
      OUT     49H, AL       ; PROT ADDRESS OF 7220 WITH A0 = 1
      MOV     AL, 01H       ; SET B REGISTER TO 01H (ONE PAGE TRANSFER)
      OUT     48H, AL       ; PORT ADDRESS OF 7220 WITH A0 = 0
      MOV     AL, 0H        ; SELECT HALF BUBBLE (1 FSA CHANNEL)
      OUT     48H, AL
      MOV     AL, 08H       ; SELECT LOW FREQ (NO ERROR CORRECTION)
      OUT     48H, AL
      MOV     AL, 0H        ; START ADDRESS = 000H
      OUT     48H, AL
      MOV     AL, 0H        ; SELECT THE FIRST BUBBLE
      OUT     48H, AL
      RET
```

Normal Read and Write Operations

Under normal operating conditions, a user sends an Initialize command and then proceeds to access the bubble. The Initialize command automatically purges the RAM area of the 7220, reads and decodes the bootloop on the 7110, fills the 7242 bootloop registers, and places the 7110 at page 0. This very important command is the next command to be tested before reading and writing data.

To verify the Initialize command, load the 7220 parametric registers to select both FSA channels for one bubble and then send the Initialize command. Status following execution of this command should be 40H, OP-COMPLETE. Once the 7220 is initialized, data can be transferred to and from the bubble. For a first attempt, it is recommeded that the operations be kept simple. That is, avoid error correction, DMA, or interrupts and only attempt single page transactions until reasonably familiar with the basic operations.

Prior to issuing the Write command, a FIFO Reset command is sent and then the parametric registers are loaded to select the page address and number of FSA channels. After the Write command is sent, the data should be output to the 7220 FIFO. When the proper number of bytes have been transferred, the 7220 status byte should reflect OP-COMPLETE and FIFO AVAILABLE to indicate that the data has been written into the 7110 bubble memory and can now be read. To read back the data written, issue a FIFO Reset command and reload the parametric registers to select the same page address in which the data was written. Issue the Read command to move the data from the 7110 to the 7220 FIFO and then use the "read FIFO" routine to transfer the data to user RAM. As always, the 7220 status byte should be checked after the operation.

AN IMPLEMENTATION EXAMPLE

To illustrate the ease with which Intel's bubble memory solution may be implemented, an MCS®-86 System Design Kit (SDK-86) is used as a vehicle to control a single BPK 72 bubble memory kit.

The bus interface between the 8086 CPU and the 7220 bubble memory controller requires seven integrated circuits and consists of four sections: address decode, data bus decode and buffering, a clock circuit, and miscellaneous control logic. The system requires power supply voltages of +12V, +5V, and, if a CRT is used, −12V.

The 8086 bus is expanded through two 50-pin, wirewrap connectors, and the BPK 72 is connected to the SDK-86 by a flat cable into a 40-pin connector located on the SDK-86. The following interface diagram shows how the signals required by the bubble system are derived from the 8086. Detailed diagrams of the address, data, clock and control logic are in the appendix.

Either the SDK-86's Keypad or Serial monitor may be used to write and debug the necessary software drivers to control the BPK 72. There is, however, an EPROM-based monitor (BMDSDK) explicity designed for the BPK 72 and is available from the Intel Insite Library. Some of the bubble-specific portions of this monitor are discussed in the following text.

Monitor Software

The BMDSDK Bubble Monitor is a highly-modular program that is written in 8086 assembly language and that resides in two 2716 EPROMs. This monitor implements, at the console level, most of the standard SDK-86 monitor functions (display/change memory, etc.) and all of the 7220 commands. The current version of the monitor utilizes only polled I/O protocol; implementing an interrupt-driven system on the SDK-86 is possible using the principles outlined in this application note. The DMA mode of operation is not available with the hardware described.

The BPK 72 driver routines are confined to one module; a listing of this module is included in the appendix. To provide some feeling for the elements of "operational" software as opposed to the test drivers discussed earlier, the write function implemented in BMDSDK monitor is examined. The flow chart in Figure 9 shows how the routine is constructed on a functional basis. Note that the subroutine reflects a very "safe" approach in that the FIFO Reset command always is sent prior to issuing the Write command. While the FIFO Reset command is not mandatory, if there is any a doubt regarding the state of the FIFO prior to a read or write operation, resetting the FIFO is a good idea. Note also that a running byte count is maintained and that the routine exits when the count goes to zero. Such a counter is not actually necessary; the FIFO AVAILABLE bit alone can be used to gate the data to the 7220.

The calling program supplies the BMWRIT routine with the total number of bytes to be transferred in the CX register. The total number of bytes written is sent to the console at the end of the operation as a monitor function. BMWRIT also returns the value of the status byte to the calling program.

Note that at label WRIT01, the routine does not progress after the Write command is sent unless both the BUSY and FIFO AVAILABLE bits are set by the controller. Once these values are set, the code issues a byte of data to the controller only if the FIFO AVAILABLE bit indicates there is room. The remainder of the code in BMWRIT is concerned with processing special write requests for the bootloop and bootloop register commands.

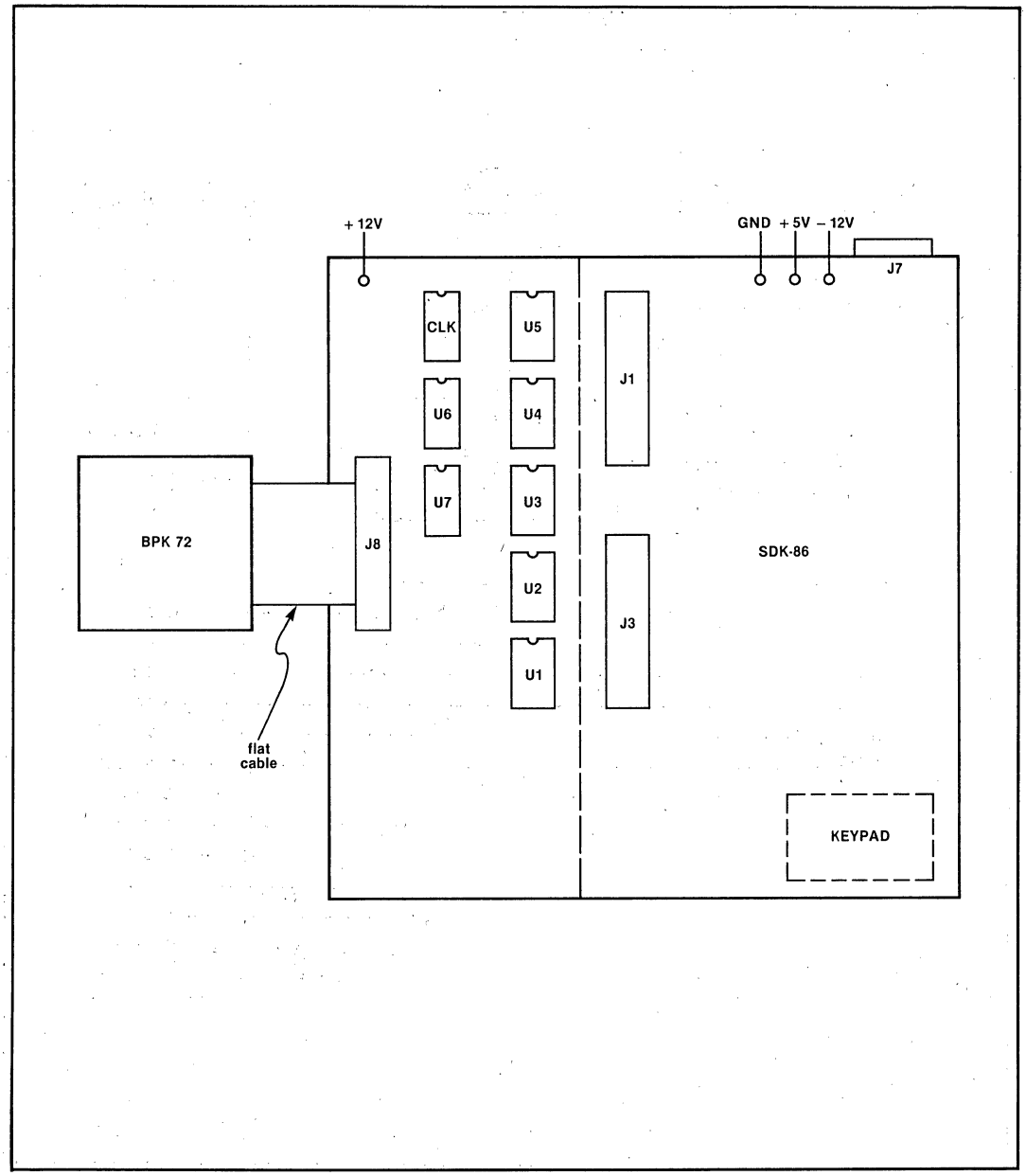


Figure 7. SDK-86/BPK 72 Implementation

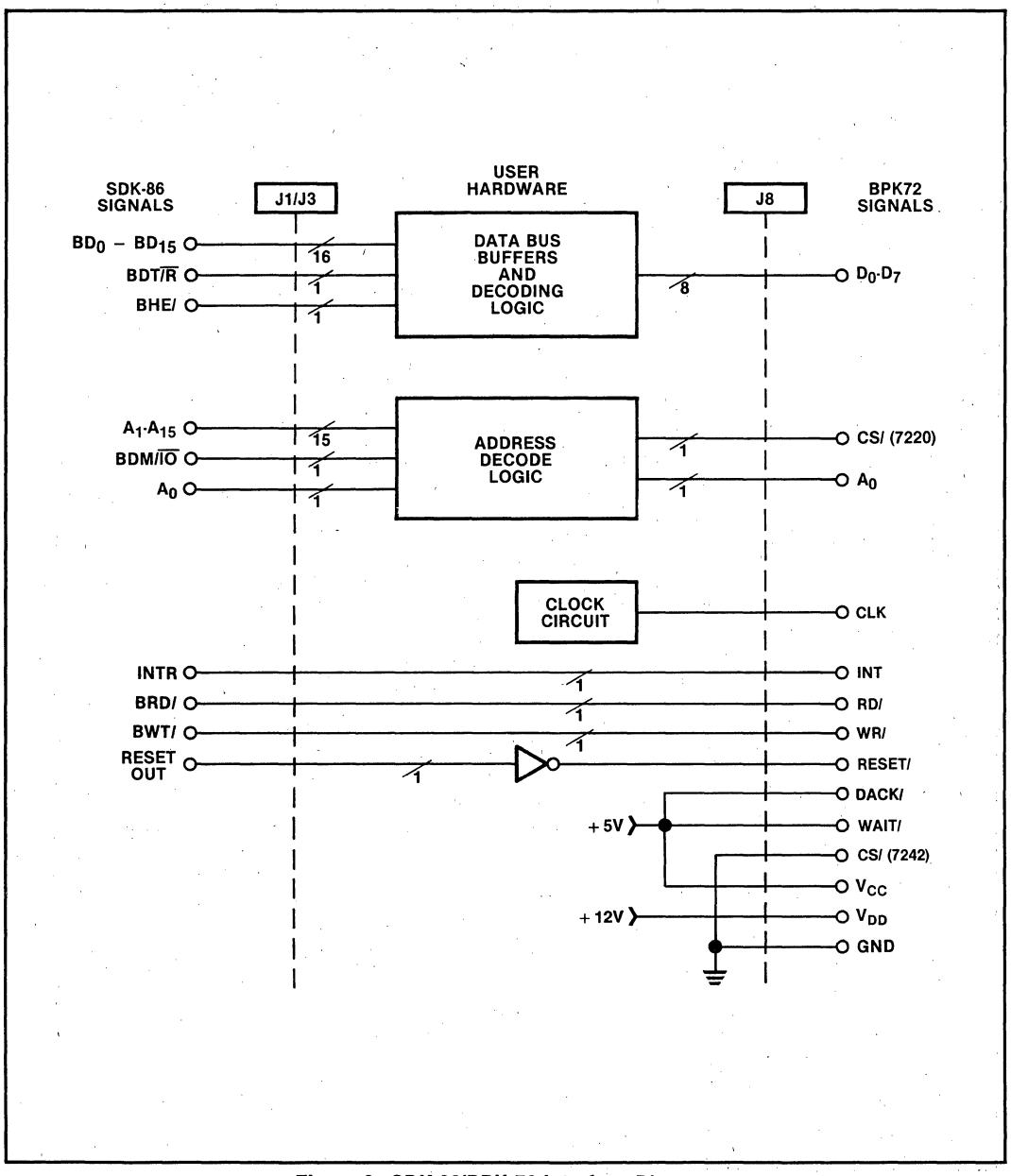


Figure 8. SDK-86/BPK 72 Interface Diagram

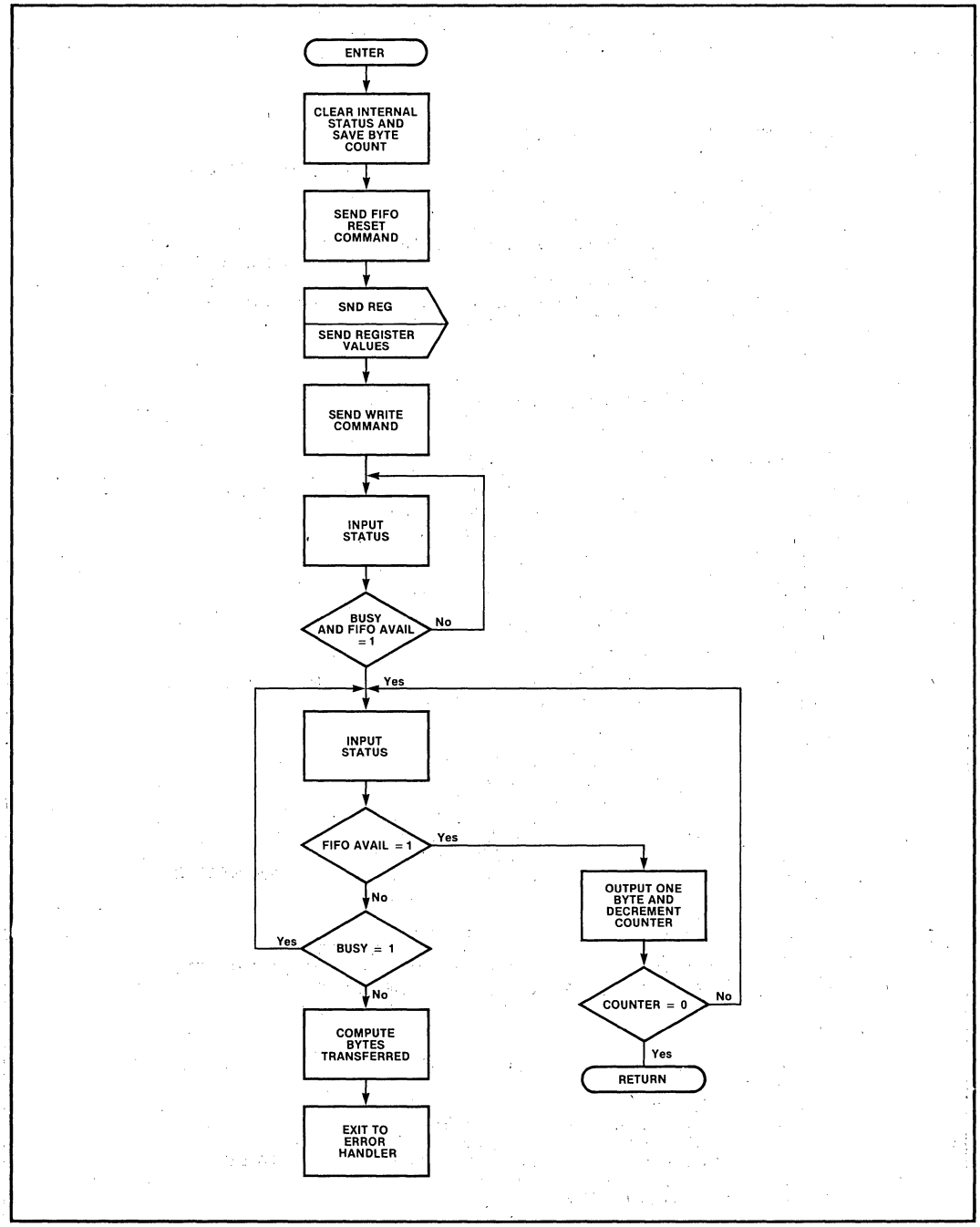


Figure 9. BMWRIT Flowchart

Table 18. BMWRIT Procedure for the SDK-86

```
;
;  FUNCTION: BMWRIT - WRITE BUBBLE MEMORY DATA.
;  INPUTS: CX = # OF BYTES TO WRITE.
;  OUTPUTS: A = STATUS: F/F(C=1: ERROR OCCURED) BX=# OF BYTES WRITTEN.
;  CALLS: SNDREG, BMWAIT.
;  DESTROYS: ALL.
;  DESCRIPTION:   THIS PROCEDURE PERFORMS A BUBBLE MEMORY WRITE OPERATION.
;                 AN ERROR WILL OCCUR IF THE NUMBER OF BYTES GIVEN FOR THE
;                 WRITE OPERATION EXCEED THE NUMBER THAT THE BMC EXPECTS
;                 (DERIVED FROM COMMAND, BLOCK LENGTH AND NUMBER OF FSA
;                  CHANNELS), OR IF THE NUMBER OF BYTES IS LESS THAN THAT
;                  WHICH THE BMC EXPECTS.
;
BMWRIT:
        XOR     AL, AL              ; A = 0
        MOV     STATUS, AL          ; CLEAR STATUS
        MOV     BX, CX
        MOV     AL, CFR
        OUT     BMSTAT, AL          ; FIFO RESET
        CALL    SNDREG              ; SEND REGISTERS TO BMC.
        MOV     SI, BUFADR          ; SET UP SRC BFR PTR (IN DATA SEG)
        MOV     AL, BMCMD           ; GET COMMAND
        OUT     BMSTAT, AL          ; ISSUE IT.
WRIT01:
        IN      AL, BMSTAT
        TEST    AL, BUSYBT          ; WAIT FOR BUSY. . .
        JZ      WRIT01
        TEST    AL, FIFOBT          ; AND FIFO READY
        JZ      WRIT01
;
;  KEEP STUFFING DATA INTO FIFO UNTIL DONE OR AN ERROR OCCURS.
;  (NOTE: BMC GOING NOT BUSY IS AN ERROR).
;
WRIT03:
        IN      AL, BMSTAT          ; GET STATUS
        TEST    AL, FIFOBT          ; FIFO READY?
        JZ      WRIT04              ; NO, WAIT FOR IT
        LODSB                       ; YES, GET DATA FOR IT
        OUT     BMDATA, AL          ; GIVE IT TO BMC
        LOOP    WRIT03              ; LOOP UNTIL DONE.
        JMP     BMWAIT              ; XFER DONE, WAIT FOR A GOOD STATUS
WRIT04:
        TEST    AL, BUSYBT
        JNZ     WRIT03              ; OK IF STILL BUSY
        SUB     BX, CX              ; BX:# OF BYTES XFERED
        JMP     CTRL99              ; ERROR IF NOT BUSY AND CX NOT ZERO
;  SPECIAL WRITE FOR BOOTLOOP AND BOOTLOOP REG CMNDS
;
BMWRTB:
        XOR     AL, AL              ; A = 0
        MOV     STATUS, AL          ; CLEAR STATUS
        MOV     BX, CX
        MOV     AL, CFR
        OUT     BMSTAT, AL          ; FIFO RESET
        CALL    SNDREG              ; SEND REGISTERS TO BMC.
        MOV     SI, BUFADR          ; SET UP SRC BFR PTR (IN DATA SEG)

   ; FILL FIFO WITH 20/40/41 BYTES
```

Table 19. BMWRIT Procedure for the SDK-86 (cont.)

```
;
;
;
WRTB01:
        LODSB
        OUT     BMDATA, AL      ; STICK IN FIFO.
        LOOP    WRTB01          ; LOOP UNTIL FILL COUNT = 0.
        IN      AL, BMSTAT      ; GET BMC STATUS
        TEST    AL, BUSYBT      ; CHECK BUSY BIT.
        JZ      SHORT WAITEX    ; NOT BUSY, ALREADY DONE.
        MOV     CX, OFFFFH      ; JUST IN CASE. . .
WAITPO:                         ; POLLED WAIT MODE
        IN      AL, BMSTAT      ; GET STATUS
        TEST    AL, BUSYBT      ; CHECK BUSY BIT
        LOOPNZ  WAITPO          ; LOOP IF STILL BUSY
        JCXZ    CTRL99          ; PROBABLY AN ERROR IF CX = 0
WAITE:
        MOV     STATUS, AL      ; A = STATUS
        RET
```

SUMMARY

The purpose of this application note is to provide a more clear understanding of the functions and characteristics of the BPK 72 one-megabit bubble memory kit. This kit has been designed specifically to relieve the user of the design effort that historically is associated with implementing a bubble memory system, and to provide a simple interface that is compatible with a broad range of microprocessor systems.

The BPK 72 is a subsystem in itself that should be viewed as simply one more component on the system bus. This component-level approach, plus the inherent flexibility of the kit, provides the user with maximum utility and functionality. By understanding how each of the subsystem parts fits together and by approaching the implementation of the kit in a methodical fashion as described in this note, the development of a working system is facilitated.

APPENDIX A

SDK-86/BPK 72
HARDWARE INTERFACE

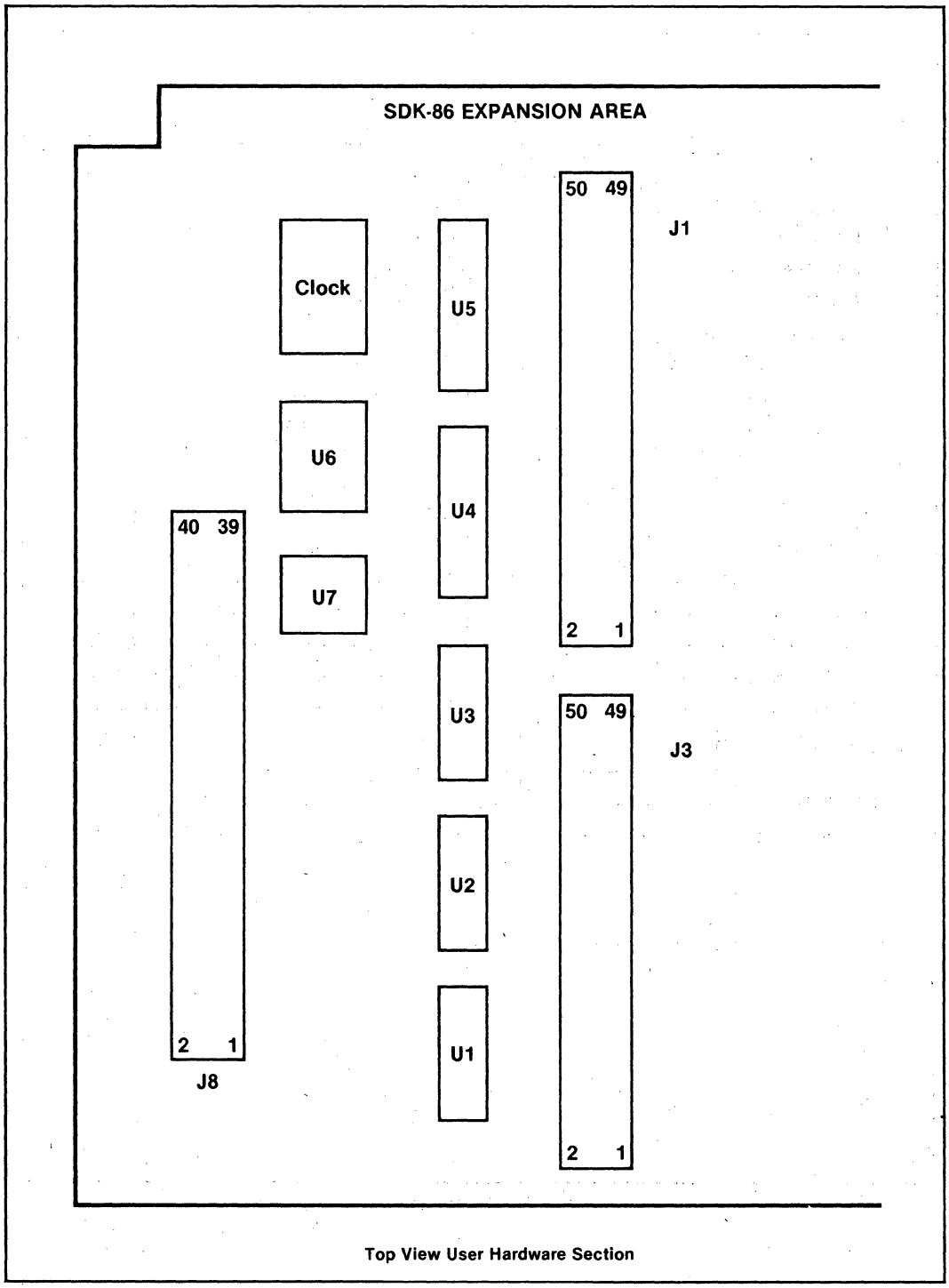


Figure 10. Parts Layout

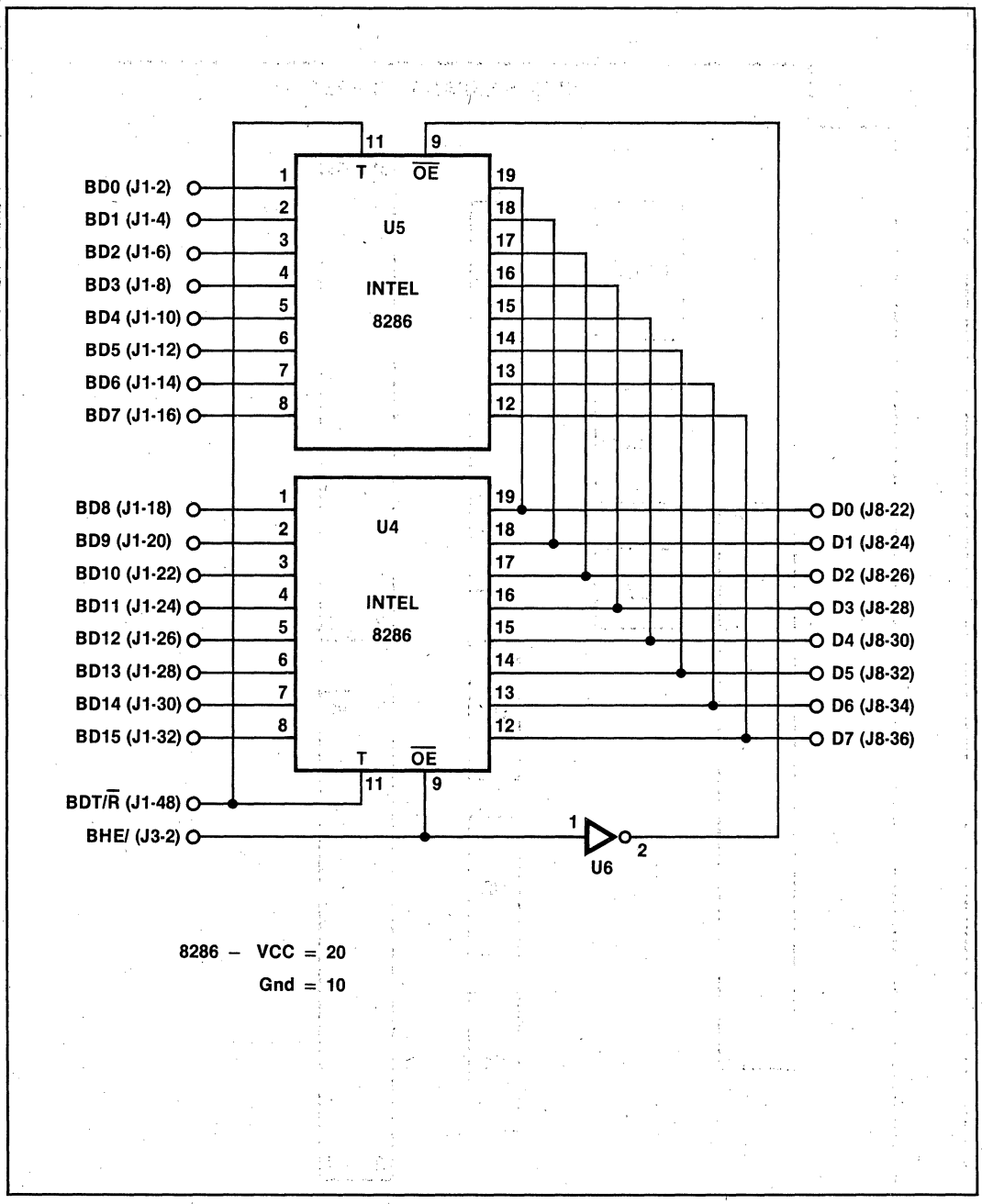


Figure 11. Data Bus Buffer and Decoding Logic

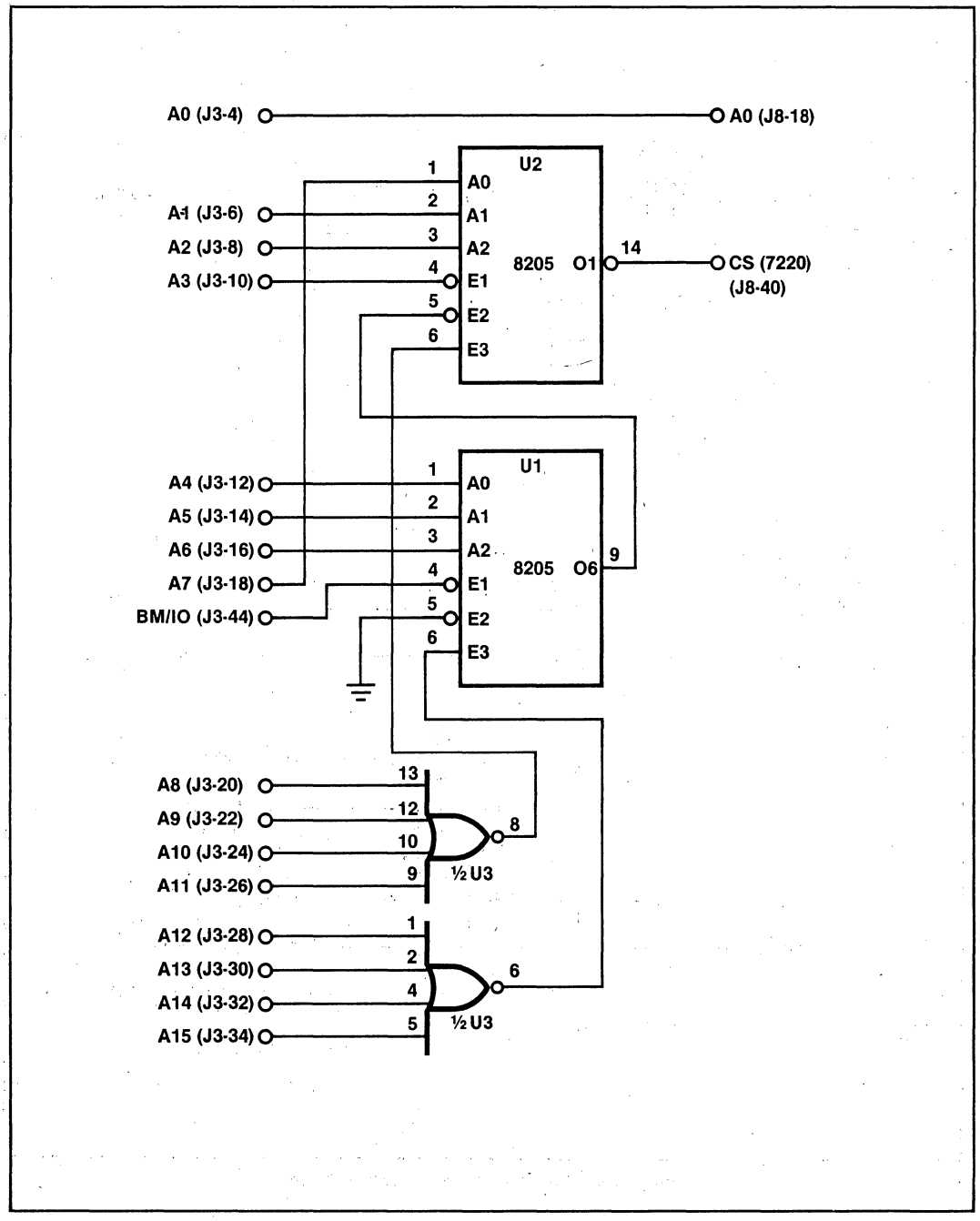


Figure 12. Address Decode Logic

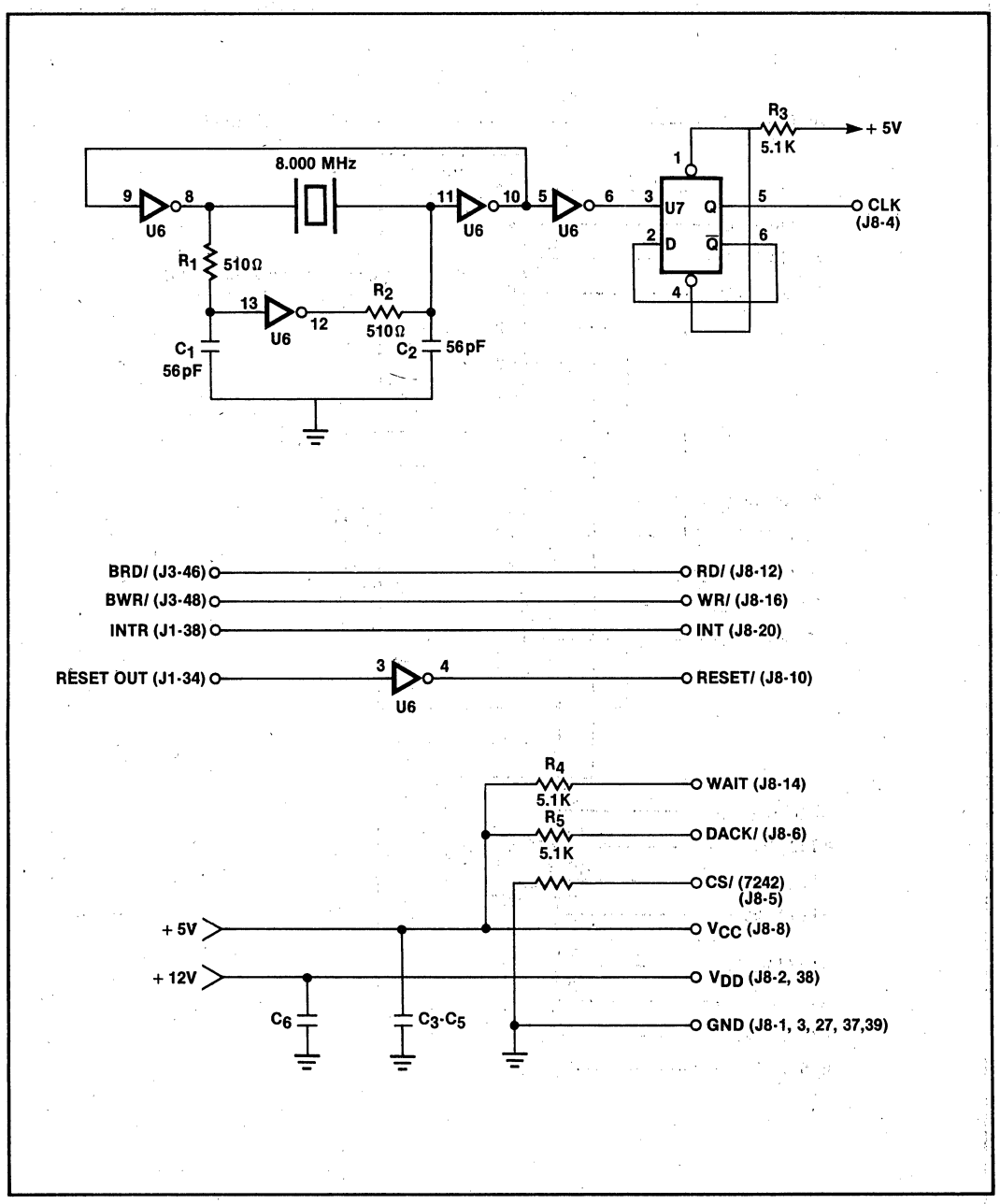


Figure 13. Clock Circuit and Control Signals

Table 20. SDK-86 Pinout

Pin	J1/J2	J3/J4	J5	J6
2	BD0	BHE/	P2C1	—
4	BD1	A0	P2C2	P1B3
6	BD2	A1	P2C3	P1B4
8	BD3	A2	P2B7	P1B2
10	BD4	A3	P2B0	P1B5
12	BD5	A4	P2B6	P1B1
14	BD6	A5	P2B3	P1B6
16	BD7	A6	P2B4	P1B0
18	BD8	A7	P2B2	P1B7
20	BD9	A8	P2B5	P1C3
22	BD10	A9	P2B1	P1C2
24	BD11	A10	P2C0	P1C1
26	BD12	A11	P2C4	P1C0
28	BD13	A12	P2C5	P1C4
30	BD14	A13	P2C6	P1C5
32	BD15	A14	P2C7	P1C6
34	RESET OUT	A15	P2A0	P1C7
36	PCLK/	A16	P2A7	P1A0
38	INTR	A17	P2A1	P1A7
40	TEST	A18	P2A6	P1A1
42	HOLD	A19	P2A2	P1A6
44	BHLDA	BM/IO/	P2A5	P1A2
46	BDEN/	BRD/	P2A3	P1A5
48	BDT/R/	BWR/	P2A4	P1A3
50	BALE	BINTA/	—	P1A4

All Odd Pins are Ground except as follows:

	J2
41	CSX/ (FD000-FDFFF)
43	CSY/ (FC000-FCFFF)
45	BS3
47	BS4
49	BS5

Table 21. SDK-86/BPK 72 Cable Wiring

Signal	J8	P1
+ 12v	2, 38	B, X
+ 5v	8	F
Ground	1, 3, 27, 37, 39	1, A, P, 22, Z
D0	22	11
D1	24	12
D2	26	13
D3	28	14
D4	30	15
D5	32	16
D6	34	17
D7	36	18
CS/ (7220)	40	Y
A0	18	10
RD/	12	J
WR/	16	K
INT	20	N
RESET/	10	H
CS/ (7242)	5	E
WAIT/	14	8
CLK	4	4
DACK/	6	L

Cable is standard 40 conductor Flat Cable.
All Odd Conductors are grounded at J8.

Table 22. SDK-86/BPK 72 Parts List

Item	Description	QT	Ref	
1	IC-8205 - Bindry Decoder	2	U1, U2	Intel (TI-74LS13)
2	IC-8286 - Octal Bus Tranciever	2	U4, U5	Intel
3	IC-746525 - Dual 4 Input M	1	U3	Any
4	IC-74H04 - Inverter	1	U6	Any
5	Resistor 510Q 1/4w	2	R1, R2	Any
6	Capacitor, 56pF 25V	2	C1,C2	Any
7	Capacitor, .1pF 25V	4	C3-C6	Any
8	Crystal, 8.000MHz Serie Res.	1	Y1	Any
9	Connector, 50 pin wirewrap	2	J1, J3	3M # 3433
10	Connector, 40 pin wirewrap	1	J8 (M)	3M # 3432
11	Connector, 40 pin	1	J8 (F)	3M # 3417
12	Connector, 44 pin Edge w/w	1	P1	Any
13	IC Socket, 20 pin w/w	2		Any (Augat)
14	IC Socket, 16 pin w/w	3		Any
15	IC Socket, 14 pin w/w	3		Any
16	Adapter Plug Assembly, 16 pin	1		Augat # 616-CE1
17	Flat Cable, 40 Conductor, 1 Ft.	1		3M # 3365
18	IC-74LS74 - Dual D Flip-Flop	1	07	Any
19	Resistor 5.1K 1/4W ±5%	3	R3, R4, R5 R5	Any
20	IC-74LS32 - OR Gate	1	U8	Any

APPENDIX B

SDK-86/BPK 72
SOFTWARE DRIVER

```
ISIS-II MCS-86 MACRO ASSEMBLER V2.1 ASSEMBLY OF MODULE DRIVER
OBJECT MODULE PLACED IN :F1:DRIVER.OBJ
ASSEMBLER INVOKED BY:  asm86 :f1:DRIVER.a86 xref print(:f1:DRIVER.lst) debug WORKFILES(:F0:.:F0:)

LOC  OBJ                  LINE     SOURCE

                             1     $TITLE(                    BPK-72 DRIVER ROUTINES.)
                             2             NAME    DRIVER
                             3 +1  $INCLUDE(:F1:RAMDEF.EXT)
                       =1    4     ;
                       =1    5     ;       publics from module RAMDEF, file RAMDEF.A86
                       =1    6     ;
----                   =1    7     STACK   SEGMENT STACK
                       =1    8             EXTRN   BMSTAK:NEAR
----                   =1    9     STACK   ENDS
                       =1   10     ;
----                   =1   11     DATA    SEGMENT PUBLIC
                       =1   12             EXTRN   RAM:BYTE,SCRBUF:BYTE,MYBUF:BYTE
                       =1   13             EXTRN   DEFADR:WORD,DEFBUB:BYTE,DEFNFC:BYTE,DEFENA:BYTE
                       =1   14             EXTRN   DEFMOD:BYTE,DEFPAG:WORD,DEFBLK:WORD
                       =1   15             EXTRN   BUFADR:WORD,BLKLEN:WORD,ENABLE:BYTE,PAGENO:WORD
                       =1   16             EXTRN   BBLNUM:BYTE,NFC:BYTE,MODE:BYTE,STATUS:BYTE,BMCMD:BYTE
                       =1   17             EXTRN   INBUF:BYTE,INBUFP:WORD,INBUFC:BYTE
                       =1   18             EXTRN   INBUFA:WORD,INBUFL:BYTE
                       =1   19             EXTRN   OUTBUF:BYTE,OUTBFP:WORD,OUTBFC:BYTE
                       =1   20             EXTRN   OUTBFA:WORD,OUTBFL:BYTE
                       =1   21             EXTRN   RDLEN:WORD,WRLEN:WORD
                       =1   22             EXTRN   PROMPT:BYTE,LEVMSK:BYTE
                       =1   23             EXTRN   BPADR:WORD,USERRG:WORD
                       =1   24             EXTRN   POPREGS:WORD,PUSHREGS:WORD
                       =1   25             EXTRN   USERBX:WORD,USERDS:WORD,USERBP:WORD,USERSS:WORD
                       =1   26             EXTRN   USERSP:WORD,USERIP:WORD,USERCS:WORD,USERFL:WORD
                       =1   27             EXTRN   USERPC:WORD
----                   =1   28     DATA    ENDS
                       =1   29     ;
                            30 +1  $INCLUDE(:F1:BMC.EQU)
                       =1   31     ;
                       =1   32     ; THESE ARE THE COMMAND EQUATES FOR BMDS
                       =1   33     ;
  0010                 =1   34     CWBRM   EQU     10H                     ; WRITE BOOTLOOP WITH MASK.
  0011                 =1   35     CIZ     EQU     11H                     ;INITIALIZE
  0012                 =1   36     CRD     EQU     12H                     ;READ
  0013                 =1   37     CWD     EQU     13H                     ;WRITE
  0014                 =1   38     CRS     EQU     14H                     ;READ SEEK
  0015                 =1   39     CRBR    EQU     15H                     ;READ BOOTLOOP REGISTER
  0016                 =1   40     CWBR    EQU     16H                     ;WRITE BOOTLOOP REGISTER
  0017                 =1   41     CWB     EQU     17H                     ;WRITE BOOTLOOP
  0018                 =1   42     CRFS    EQU     18H                     ;READ FIFO STATUS
  0019                 =1   43     CAB     EQU     19H                     ;ABORT
  001A                 =1   44     CWRS    EQU     1AH                     ; WRITE SEEK.
  001B                 =1   45     CRB     EQU     1BH                     ;READ BOOTLOOP
  001C                 =1   46     CRCD    EQU     1CH                     ;READ CORRECTED DATA
  001D                 =1   47     CFR     EQU     1DH                     ;FIFO RESET
  001E                 =1   48     CPURG   EQU     1EH                     ; MBM PURGE COMMAND.
  001F                 =1   49     CSR     EQU     1FH                     ;SOFTWARE RESET
                       =1   50     ;
```

```
LOC  OBJ                  LINE     SOURCE

                    =1      51     ; I/O PORT ADDRESSES.
                    =1      52     ;
  00E1              =1      53     BMSTAT  EQU     0E1H            ; BUBBLE MEMORY DEVICE STATUS PORT.
  00E0              =1      54     BMDATA  EQU     0E0H            ; BUBBLE MEMORY DEVICE DATA PORT.
                    =1      55     ;
                    =1      56     ; STATUS WORD BITS
                    =1      57     ;
  0001              =1      58     FIFOBT  EQU     01H             ; FIRST BIT IS FIFO STATUS
  0002              =1      59     PARERR  EQU     02H             ; SECOND BIT IS PARITY ERROR.
  0004              =1      60     UNCERR  EQU     04H             ; THIRD BIT IS UNCORRECTABLE ERROR BIT.
  0008              =1      61     CORERR  EQU     08H             ; FOURTH BIT IS CORRECTABLE ERROR BIT.
  0010              =1      62     TIMERR  EQU     10H             ; FIFTH BIT IS TIMING ERROR BIT.
  0020              =1      63     OPFAIL  EQU     20H             ; OPERATION FAIL BIT.
  0040              =1      64     OPDONE  EQU     40H             ; OPERATION COMPLETE BIT.
  0080              =1      65     BUSYBT  EQU     80H             ; BUSY BIT.
                    =1      66     ;
                    =1      67     ; ENABLE REG BITS
                    =1      68     ;
  0001              =1      69     INTENA  EQU     01H             ; INTERRUPT NORMAL
  0002              =1      70     IERENA  EQU     02H             ; INTERRUPT ERROR
  0004              =1      71     DMAENA  EQU     04H             ; DMA
  0008              =1      72     RSVD1   EQU     08H
  0010              =1      73     WBLENA  EQU     10H             ; WRITE BOOTLOOP
  0020              =1      74     RCDENA  EQU     20H             ; READ CORRECTED DATA
  0040              =1      75     ICDENA  EQU     40H             ; INTERNALLY CORRECTED DATA
  0080              =1      76     RSVD2   EQU     80H
                            77 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                            78     CODE    SEGMENT PUBLIC
                            79             ASSUME  DS:DATA,CS:CODE,SS:STACK
                            80     ;*************************************************************
                            81     ;                                                           *
                            82     ;                BPK72 DRIVER routines                      *
                            83     ;                ===== ====== ========                      *
                            84     ;                                                           *
                            85     ;  The routines in this module constitute the routines      *
                            86     ;  needed to directly drive the BPK72 bubble memory         *
                            87     ;  development board.  This module is designed to be self   *
                            88     ;  contained, and may be called by ANY user procedures.     *
                            89     ;                                                           *
                            90     ;          The procedures in this module are                *
                            91     ;                                                           *
                            92     ;  BMCTRL - Perform non-data transfer BMC operations.      *
                            93     ;  BMREAD - Perform data read BMC operations.              *
                            94     ;  BMWRIT - Perform data write BMC operations.             *
                            95     ;                                                           *
                            96     ;  ZAPREG - Set internal registers to an acceptable value  *
                            97     ;                                                           *
                            98     ;          Parameter passing                                *
                            99     ;          ========= =======                                *
                           100     ;                                                           *
                           101     ;  All parameters are passed to the BMC driver routines via *
                           102     ;  common (PUBLIC) variables.  These variables are         *
                           103     ;                                                           *
                           104     ;  BUFADR - The memory address of the input/output buffer  *
                           105     ;            to be used for data transfer operations.       *
                           106     ;  ENABLE - The enable byte to be passed to the BMC before *
                           107     ;            every operation.                               *
                           108     ;  PAGENO - The starting block number to be passed to the  *
                           109     ;            BMC before every operation.  (NOTE: This field *
                           110     ;            has no meaning for control operations).        *
                           111     ;  BLKLEN - The number of pages to be transfered by the BMC.*
                           112     ;            (NOTE: This field has no meaning for control   *
                           113     ;            operations).                                   *
                           114     ;  BBLNUM - The bubble select to be transfered to the BMC  *
                           115     ;            before every operation.(NOTE: This field has   *
                           116     ;            no meaning for SOME control operations).       *
                           117     ;  NFC    - The number of FSA channels passed to the BMC   *
                           118     ;            before every operation. (NOTE: This field has  *
                           119     ;            no meaning for SOME of the control operatioNs). *
                           120     ;                                                           *
                           121     ;  For a detailed definition of the ENABLE,PAGENO,BLKLEN,   *
                           122     ;   BBLNUM. and NFC fields, refer to the BPK-72 USER MANUAL *
                           123     ;   or the Bubble Memory Design Handbook.                  *
                           124     ;                                                           *
                           125     ;*************************************************************
                           126 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           127     ;*******************************
                           128     ;
                           129     ; ENTRY POINTS
                           130     ;
                           131             PUBLIC  ZAPREG,BMCTRL,BMWAIT,BMREAD,BMWRIT,BMWRTB
                           132     ;
                           133     ;***********************
                           134     ;
                           135     ; MISC EQUATES
                           136     ;
  000B                     137     REG1    EQU     0BH                     ; FIRST BMC REGISTER TO USE IS BLOCK LENGTh
  003C                     138     STATER  EQU     3CH                     ; STATUS WORD ERROR MASK
                           139                                             ; IGNORE PARITY ERR, REV D OF BMC
                           140 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           141     ;*******************************
                           142     ;
                           143     ;    MODE BYTE DEFINITION
                           144     ;    ==== ==== ==========
                           145     ;
                           146     ; The bits in the MODE BYTE specify the type of the data transmission
                           147     ; TO USE, AND WHETHER TO PRINT STATUS AFTER EACH OPERATION.
                           148     ; If interrupts are enabled in the MODE BYTE, they must also be selected
                           149     ; in the ENABLE BYTE for desired operation to occur.
                           150     ;
  0001                     151     INTMOD  EQU     01H             ; FIRST BIT IN MODE WORD IT INTERRUPT SELECT.
  0002                     152     DMAMOD  EQU     02H             ; SECOND BIT IN MODE WORD IS DMA SELECT.
  0080                     153     DBGMOD  EQU     80H             ; DEBUG BIT OF MODE WORD
                           154 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           155     ;*******************************
                           156     ;
                           157     ; FUNCTION: BMCTRL - PERFORM BMC CONTROL OPERATIONS (NON-DATA TRANSFER).
                           158     ; INPUTS: NONE
                           159     ; OUTPUTS: A=STATUS;F/F(C=1: AN ERROR OCCURED).
                           160     ; CALLS: SNDREG,BMWAIT
                           161     ; DESTROYS: ALL
                           162     ; DESCRIPTION: THIS PROCEDURE IS USED TO PERFORM NON-DATA TRANSFER
                           163     ;              BMC OPERATIONS.
                           164     ;
0000                       165     BMCTRL:
0000 E8D700                166             CALL    SNDREG                  ; LOAD BMC REGISTERS.
0003 A00000          E     167             MOV     AL,BMCMD                ; GET COMMAND.
0006 E6E1                  168             OUT     BMSTAT,AL               ; INITIATE COMMAND.
0008 E80E00                169             CALL    BMWAIT                  ; WAIT FOR COMPLETION.
000B 243C                  170             AND     AL,STATER               ; DO WE HAVE AN ERROR?
000D A00000          E     171             MOV     AL,STATUS               ; LOAD STATUS INTO 'A' FOR EXIT
0010 7502                  172             JNZ     SHORT CTRL99            ; ERROR, RETURN WITH FLAG SET.
0012 F8                    173             CLC                             ; CLEAR CARRY(ERROR FLAG)
0013 C3                    174             RET                             ; AND RETURN
                           175     ;
                           176     ; WE HAD AN ERROR, RETURN WITH ERROR FLAG(CARRY FLAG) SET.
                           177     ;          THIS IS THE GENERAL ERROR EXIT
                           178     ;
0014                       179     CTRL99:
0014 A20000          E     180             MOV     STATUS,AL
0017 F9                    181             STC                             ; SET ERROR FLAG (CARRY FLAG)
0018 C3                    182             RET                             ; AND RETURN.
                           183     ;*******************************
                           184     ;
                           185     ; FUNCTION: BMWAIT
                           186     ; INPUTS: NONE
                           187     ; OUTPUTS: STATUS IN A
                           188     ; CALLS: NOTHING
                           189     ; DESTROYS: A,F/F
                           190     ; DESCRIPTION: THIS PROCEDURE WILL WAIT UNTIL THE CURRENT BMC
                           191     ;              OPERATION COMPLETES.
                           192     ;
0019                       193     BMWAIT:
                           194     ;
                           195     ; CHECK CURRENT STATUS (GOOD ONLY IF RAC=0 AND BSY=0)
                           196     ;
0019 E4E1                  197             IN      AL,BMSTAT               ; GET BMC STATUS
001B A880                  198             TEST    AL,BUSYBT               ; CHECK BUSY BIT.
001D 740B                  199             JZ      SHORT WAITEX            ; NOT BUSY, ALREADY DONE.
001F B9FFFF                200             MOV     CX,0FFFFH               ; JUST IN CASE...
0022                       201     WAITPO:                                 ; POLLED WAIT MODE
0022 E4E1                  202             IN      AL,BMSTAT               ; GET STATUS
0024 A880                  203             TEST    AL,BUSYBT               ; CHECK BUSY BIT
0026 E0FA                  204             LOOPNZ  WAITPO                  ; LOOP IF STILL BUSY
0028 E3EA                  205             JCXZ    CTRL99                  ; PROBABLY AN ERROR IF CX=0
002A                       206     WAITEX:                                 ; CORRECT STATUS AND RETURN.
002A A20000          E     207             MOV     STATUS,AL               ; A = STATUS
002D C3                    208             RET
                           209 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           210     ;*****************************
                           211     ;
                           212     ; FUNCTION: BMREAD
                           213     ; INPUTS: CX = NUMBER OF BYTES TO READ, ES SET TO DS
                           214     ; OUTPUTS: A = STATUS; F/F(C=1: ERROR OCCURED)
                           215     ;          BX = NUMBER OF BYTES READ
                           216     ; CALLS: SNDREG
                           217     ; DESTROYS: ALL
                           218     ; DESCRIPTION: ALL PARAMETERS ARE PASSED THROUGH COMMON(PUBLIC)
                           219     ;              VARIABLES( SEE MODULE HEADER).
                           220     ;
002E                       221     BMREAD:
002E 32C0                  222             XOR     AL,AL                   ; A = 0
0030 A20000          E     223             MOV     STATUS,AL               ; CLEAR STATUS.
0033 8BD9                  224             MOV     BX,CX                   ; SAVE BYTE COUNT FOR LOOP
0035 E8A200                225             CALL    SNDREG                  ; SEND REGISTERS TO BMC.
0038 8B3E0000        E     226             MOV     DI,BUFADR               ; SET UP DEST BFR PTR (IN EXTRA SEG)
003C 8CD8                  227             MOV     AX,DS
003E 8EC0                  228             MOV     ES,AX                   ; SET EXTRA SEG FOR BYTE MOVE DEST
0040 A00000          E     229             MOV     AL,BMCMD                ; GET COMMAND
0043 E6E1                  230             OUT     BMSTAT,AL               ; ISSUE IT.
                           231     ;
0045 B9FFFF                232             MOV     CX,0FFFFH
0048                       233     BMRD1:
0048 E4E1                  234             IN      AL,BMSTAT
004A A880                  235             TEST    AL,BUSYBT
004C E1FA                  236             LOOPZ   BMRD1                   ; WAIT FOR BUSY, BUT NOT FOREVER
004E E3C4                  237             JCXZ    CTRL99                  ; CX=0 PROBABLY AN ERROR
0050 8BCB                  238             MOV     CX,BX
                           239     ;
                           240     ;        READ LOOP
                           241     ;        ==== ====
                           242     ;
0052                       243     BMRD2:
0052 E4E1                  244             IN      AL,BMSTAT               ; GET STATUS
0054 A801                  245             TEST    AL,FIFOBT               ; FIFO EMPTY?
0056 7407                  246             JZ      SHORT BMRD3             ; YEP, GO CHECK FOR BUSY.
0058 E4E0                  247             IN      AL,BMDATA               ; NOPE, GET DATA
005A AA                    248             STOSB                           ; STORE IT
005B E2F5                  249             LOOP    BMRD2                   ; AND GO FOR MORE.
005D EBBA                  250             JMP     BMWAIT                  ; XFER DONE, WAIT FOR A GOOD STATUS
005F                       251     BMRD3:                                  ; NOTHING IN FIFO, IS OP COMPLETE?
005F A880                  252             TEST    AL,BUSYBT               ; CHECK BUSY BIT
0061 75EF                  253             JNZ     BMRD2                   ; STILL BUSY, WAIT.
0063 2BD9                  254             SUB     BX,CX                   ; BX <- # OF BYTES XFERED
0065 EBAD                  255             JMP     CTRL99
                           256 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                          257      ;*******************************
                          258      ;
                          259      ; FUNCTION: BMWRIT - WRITE BUBBLE MEMORY DATA.
                          260      ; INPUTS: CX = # OF BYTES TO WRITE.
                          261      ; OUTPUTS: A = STATUS; F/F(C=1:ERROR OCCURED), BX=# OF BYTES WRITTEN.
                          262      ; CALLS: SNDREG,BMWAIT.
                          263      ; DESTROYS: ALL.
                          264      ; DESCRIPTION: THIS PROCEDURE PERFORMS A BUBBLE MEMORY WRITE OPERATION.
                          265      ;              AN ERROR WILL OCCUR IF THE NUMBER OF BYTES GIVEN FOR THE
                          266      ;              WRITE OPERATION EXCEED THE NUMBER THAT THE BMC EXPECTS
                          267      ;              (DERIVED FROM COMMAND, BLOCK LENGTH AND NUMBER OF FSA
                          268      ;               CHANNELS), OR IF THE NUMBER OF BYTES IS LESS THAN THAT
                          269      ;              WHICH THE BMC EXPECTS.
                          270      ;
0067                      271      BMWRIT:
0067 32C0                 272              XOR     AL,AL                   ; A = 0
0069 A20000         E     273              MOV     STATUS,AL               ; CLEAR STATUS
006C 8BD9                 274              MOV     BX,CX
006E B01D                 275              MOV     AL,CFR
0070 E6E1                 276              OUT     BMSTAT.AL               ; FIFO RESET
0072 E86500               277              CALL    SNDREG                  ; SEND REGISTERS TO BMC.
0075 8B360000       E     278              MOV     SI.BUFADR               ; SET UP SRC BFR PTR (IN DATA SEG)
0079 A00000         E     279              MOV     AL,BMCMD                ; GET COMMAND
007C E6E1                 280              OUT     BMSTAT,AL               ; ISSUE IT.
007E                      281      WRIT01:
007E E4E1                 282              IN      AL,BMSTAT
0080 A880                 283              TEST    AL,BUSYBT               ; WAIT FOR BUSY...
0082 74FA                 284              JZ      WRIT01
0084 A801                 285              TEST    AL,FIFOBT               ;   AND FIFO READY
0086 74F6                 286              JZ      WRIT01
                          287      ;
                          288      ; KEEP STUFFING DATA INTO FIFO UNTIL DONE OR AN ERROR OCCURS.
                          289      ; (NOTE: BMC GOING NOT BUSY IS AN ERROR).
                          290      ;
0088                      291      WRIT03:
0088 E4E1                 292              IN      AL,BMSTAT               ; GET STATUS
008A A801                 293              TEST    AL,FIFOBT               ; FIFO READY?
008C 7407                 294              JZ      WRIT04                  ; NO. WAIT FOR IT
008E AC                   295              LODSB                           ; YES, GET DATA FOR IT
008F E6E0                 296              OUT     BMDATA,AL               ; GIVE IT TO BMC
0091 E2F5                 297              LOOP    WRIT03                  ; LOOP UNTIL DONE.
0093 EB84                 298              JMP     BMWAIT                  ; XFER DONE, WAIT FOR A GOOD STATUS
0095                      299      WRIT04:
0095 A880                 300              TEST    AL,BUSYBT
0097 75EF                 301              JNZ     WRIT03                  ; OK IF STILL BUSY
0099 2BD9                 302              SUB     BX,CX                   ; BX <- # OF BYTES XFERED
009B E976FF               303              JMP     CTRL99                  ; ERROR IF NOT BUSY AND CX NOT ZERO
                          304      ;
                          305      ; SPECIAL WRITE FOR BOOTLOOP AND BOOTLOOP REG CMNDS
                          306      ;
009E                      307      BMWRTB:
009E 32C0                 308              XOR     AL,AL                   ; A = 0
00A0 A20000         E     309              MOV     STATUS,AL               ; CLEAR STATUS
00A3 8BD9                 310              MOV     BX,CX
00A5 B01D                 311              MOV     AL,CFR
```

```
LOC  OBJ                  LINE     SOURCE

00A7 E6E1                  312             OUT     BMSTAT,AL               ; FIFO RESET
00A9 E82E00                313             CALL    SNDREG                  ; SEND REGISTERS TO BMC.
00AC 8B360000        E     314             MOV     SI,BUFADR               ; SET UP SRC BFR PTR (IN DATA SEG)
                           315     ;
                           316     ; FILL FIFO WITH 20/40/41 BYTES
                           317     ;
00B0                       318     WRTB01:
00B0 AC                    319             LODSB
00B1 E6E0                  320             OUT     BMDATA,AL               ; STICK IN FIFO.
00B3 E2FB                  321             LOOP    WRTB01                  ; LOOP UNTIL FILL COUNT=0.
00B5 A00000          E     322             MOV     AL,BMCMD
00B8 E6E1                  323             OUT     BMSTAT,AL               ; SEND CMND
00BA E95CFF                324             JMP     BMWAIT
                           325 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           326     ;*******************************
                           327     ;
                           328     ; FUNCTION: ZAPREG - ZAP ALL INTERNAL REGISTERS.
                           329     ; INPUTS: NONE
                           330     ; OUTPUTS: NONE
                           331     ; CALLS: NOTHING
                           332     ; DESTROYS: NOTHING.
                           333     ; DESCRIPTION: SET ALL INTERNAL REGISTERS EXCEPT 'ENABLE' TO AN
                           334     ;              ACCEPTABLE VALUE.  NOTE:  AN ACCEPTABLE VALUE MAY
                           335     ;              OR MAY NOT BE THE ONE DESIRED AS A DEFAULT.
                           336     ;
00BD                       337     ZAPREG:
00BD 9C                    338             PUSHF                           ; SAVE FLAGS
00BE 50                    339             PUSH    AX                      ; SAVE REGISTERS
00BF 53                    340             PUSH    BX
00C0 BB0000                341             MOV     BX,0
00C3 891E0000         E    342             MOV     PAGENO,BX               ; STARTING PAGE NUMBER = 0
00C7 43                    343             INC     BX
00C8 891E0000         E    344             MOV     BLKLEN,BX               ; BLOCK LENGTH = 1
00CC 32C0                  345             XOR     AL,AL
00CE A20000           E    346             MOV     BBLNUM.AL               ; BUBBLE NUMBER = 0
00D1 FEC0                  347             INC     AL
00D3 A20000           E    348             MOV     NFC.AL                  ; # OF FSA CHANNELS = 1 (2 CHANNELS)
00D6 5B                    349             POP     BX                      ; RESTORE REGISTERS.
00D7 58                    350             POP     AX
00D8 9D                    351             POPF
00D9 C3                    352             RET
                           353 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           354     ;*******************************
                           355     ;
                           356     ; FUNCTION: SNDREG - FORMAT AND SEND INTERNAL REGISTERS TO BMC.
                           357     ; INPUTS: NONE
                           358     ; OUTPUTS: NONE
                           359     ; DESTROYS: NOTHING.
                           360     ; DESCRIPTION: FORMAT AND SEND ALL INTERNAL REGISTERS TO THE BMC.
                           361     ;
00DA                       362     SNDREG:
00DA 9C                    363             PUSHF
00DB 50                    364             PUSH    AX                      ; SAVE REGISTERS
00DC 53                    365             PUSH    BX
00DD 51                    366             PUSH    CX
00DE B00B                  367             MOV     AL,REG1                 ; GET FIRST REGISTER ADDRESS.
00E0 E6E1                  368             OUT     BMSTAT,AL               ; SELECT IT.
                           369     ;
                           370     ; CONSTRUCT AND SEND BLOCK LENGTH.
                           371     ;
00E2 8B1E0000        E     372             MOV     BX,BLKLEN               ; HL = BLOCK LENGTH
00E6 8AC3                  373             MOV     AL,BL                   ; A  = BLOCK LENGTH LSB
00E8 E6E0                  374             OUT     BMDATA,AL               ; GIVE IT TO BMC.
00EA A00000          E     375             MOV     AL,NFC                  ; A  = NUMBER OF FSA CHANNELS.
00ED B104                  376             MOV     CL,4
00EF D2E0                  377             SHL     AL,CL
00F1 0AC7                  378             OR      AL,BH                   ; MERGE INTO BLOCK MSB
00F3 E6E0                  379             OUT     BMDATA,AL               ; GIVE IT TO BMC.
                           380     ;
                           381     ; SEND ENABLE BYTE.
                           382     ;
00F5 A00000          E     383             MOV     AL,ENABLE               ; GET ENABLE BYTE
00F8 E6E0                  384             OUT     BMDATA,AL               ; GIVE IT TO BMC
                           385     ;
                           386     ; CONSTRUCT AND SEND ADDRESS REGISTER.
                           387     ;
00FA 8B1E0000        E     388             MOV     BX,PAGENO               ; HL = STARTING PAGE NUMBER
00FE 8AC3                  389             MOV     AL,BL                   ; A  = ADDRESS REGISTER LSB
0100 E6E0                  390             OUT     BMDATA,AL               ; GIVE IT TO BMC.
0102 A00000          E     391             MOV     AL,BBLNUM               ; A  = BUBBLE NUMBER
0105 B103                  392             MOV     CL,3
0107 D2E0                  393             SHL     AL.CL
0109 0AC7                  394             OR      AL,BH                   ; MERGE INTO PAGE NUMBER MSB.
010B E6E0                  395             OUT     BMDATA,AL               ; GIVE IT TO BMC.
                           396     ;
                           397     ; RESTORE REGISTERS AND RETURN.
                           398     ;
010D 59                    399             POP     CX
010E 5B                    400             POP     BX
010F 58                    401             POP     AX
0110 9D                    402             POPF
0111 C3                    403             RET
                           404 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           405     CODE    ENDS
                           406             END
```

XREF SYMBOL TABLE LISTING

```
NAME      TYPE     VALUE  ATTRIBUTES, XREFS

??SEG . . SEGMENT        SIZE=0000H PARA PUBLIC
BBLNUM. . V BYTE   0000H  EXTRN  16# 346 391
BLKLEN. . V WORD   0000H  EXTRN  15# 344 372
BMCMD . . V BYTE   0000H  EXTRN  16# 167 229 279 322
BMCTRL. . L NEAR   0000H  CODE PUBLIC  131 165#
BMDATA. . NUMBER   00E0H    54# 247 296 320 374 379 384 390 395
BMRD1 . . L NEAR   0048H  CODE  233# 236
BMRD2 . . L NEAR   0052H  CODE  243# 249 253
BMRD3 . . L NEAR   005FH  CODE  246 251#
BMREAD. . L NEAR   002EH  CODE PUBLIC  131 221#
BMSTAK. . L NEAR   0000H  EXTRN  8#
BMSTAT. . NUMBER   00E1H    53# 168 197 202 230 234 244 276 280 282 292 312 323 368
BMWAIT. . L NEAR   0019H  CODE PUBLIC  131 169 193# 250 298 324
BMWRIT. . L NEAR   0067H  CODE PUBLIC  131 271#
BMWRTB. . L NEAR   009EH  CODE PUBLIC  131 307#
BPADR . . V WORD   0000H  EXTRN  23#
BUFADR. . V WORD   0000H  EXTRN  15# 226 278 314
BUSYBT. . NUMBER   0080H    65# 198 203 235 252 283 300
CAB . . . NUMBER   0019H    43#
CFR . . . NUMBER   001DH    47# 275 311
CIZ . . . NUMBER   0011H    35#
CODE. . . SEGMENT        SIZE=0112H PARA PUBLIC   78# 79 405
CORERR. . NUMBER   0008H    61#
CPURG . . NUMBER   001EH    48#
CRB . . . NUMBER   001BH    45#
CRBR. . . NUMBER   0015H    39#
CRCD. . . NUMBER   001CH    46#
CRD . . . NUMBER   0012H    36#
CRFS. . . NUMBER   0018H    42#
CRS . . . NUMBER   0014H    38#
CSR . . . NUMBER   001FH    49#
CTRL99. . L NEAR   0014H  CODE  172 179# 205 237 255 303
CWB . . . NUMBER   0017H    41#
CWBR. . . NUMBER   0016H    40#
CWBRM . . NUMBER   0010H    34#
CWD_. . . NUMBER   0013H    37#
CWRS. . . NUMBER   001AH    44#
DATA. . . SEGMENT        SIZE=0000H PARA PUBLIC   11# 28 79
DBGMOD. . NUMBER   0080H    153#
DEFADR. . V WORD   0000H  EXTRN  13#
DEFBLK. . V WORD   0000H  EXTRN  14#
DEFBUB. . V BYTE   0000H  EXTRN  13#
DEFENA. . V BYTE   0000H  EXTRN  13#
DEFMOD. . V BYTE   0000H  EXTRN  14#
DEFNFC. . V BYTE   0000H  EXTRN  13#
DEFPAG. . V WORD   0000H  EXTRN  14#
DMAENA. . NUMBER   0004H    71#
DMAMOD. . NUMBER   0002H    152#
ENABLE. . V BYTE   0000H  EXTRN  15# 383
FIFOBT. . NUMBER   0001H    58# 245 285 293
ICDENA. . NUMBER   0040H    75#
```

```
NAME      TYPE     VALUE  ATTRIBUTES, XREFS
IERENA. . NUMBER   0002H    70#
INBUF . . V BYTE   0000H  EXTRN  17#
INBUFA. . V WORD   0000H  EXTRN  18#
INBUFC. . V BYTE   0000H  EXTRN  17#
INBUFL. . V BYTE   0000H  EXTRN  18#
INBUFP. . V WORD   0000H  EXTRN  17#
INTENA. . NUMBER   0001H    69#
INTMOD. . NUMBER   0001H   151#
LEVMSK. . V BYTE   0000H  EXTRN  22#
MODE. . . V BYTE   0000H  EXTRN  16#
MYBUF . . V BYTE   0000H  EXTRN  12#
NFC . . . V BYTE   0000H  EXTRN  16# 348 375
OPDONE. . NUMBER   0040H    64#
OPFAIL. . NUMBER   0020H    63#
OUTBFA. . V WORD   0000H  EXTRN  20#
OUTBFC. . V BYTE   0000H  EXTRN  19#
OUTBFL. . V BYTE   0000H  EXTRN  20#
OUTBFP. . V WORD   0000H  EXTRN  19#
OUTBUF. . V BYTE   0000H  EXTRN  19#
PAGENO. . V WORD   0000H  EXTRN  15# 342 388
PARERR. . NUMBER   0002H    59#
POPREGS . V WORD   0000H  EXTRN  24#
PROMPT. . V BYTE   0000H  EXTRN  22#
PUSHREGS. V WORD   0000H  EXTRN  24#
RAM . . . V BYTE   0000H  EXTRN  12#
RCDENA. . NUMBER   0020H    74#
RDLEN . . V WORD   0000H  EXTRN  21#
REG1. . . NUMBER   000BH    137# 367
RSVD1 . . NUMBER   0008H    72#
RSVD2 . . NUMBER   0080H    76#
SCRBUF. . V BYTE   0000H  EXTRN  12#
SNDREG. . L NEAR   00DAH  CODE  166 225 277 313 362#
STACK . . SEGMENT         SIZE=0000H PARA STACK
STATER. . NUMBER   003CH    138# 170
STATUS. . V BYTE   0000H  EXTRN  16# 171 180 207 223 273 309
TIMERR. . NUMBER   0010H    62#
UNCERR. . NUMBER   0004H    60#
USERBP. . V WORD   0000H  EXTRN  25#
USERBX. . V WORD   0000H  EXTRN  25#
USERCS. . V WORD   0000H  EXTRN  26#
USERDS. . V WORD   0000H  EXTRN  25#
USERFL. . V WORD   0000H  EXTRN  26#
USERIP. . V WORD   0000H  EXTRN  26#
USERPC. . V WORD   0000H  EXTRN  27#
USERRG. . V WORD   0000H  EXTRN  23#
USERSP. . V WORD   0000H  EXTRN  26#
USERSS. . V WORD   0000H  EXTRN  25#
WAITEX. . L NEAR   002AH  CODE  199 206#
WAITPO. . L NEAR   0022H  CODE  201# 204
WBLENA. . NUMBER   0010H    73#
WRITO1. . L NEAR   007EH  CODE  281# 284 286
WRITO3. . L NEAR   0088H  CODE  291# 297 301
WRITO4. . L NEAR   0095H  CODE  294 299#
WRLEN . . V WORD   0000H  EXTRN  21#
WRTBO1. . L NEAR   00B0H  CODE  318# 321
```

```
NAME      TYPE     VALUE  ATTRIBUTES, XREFS
ZAPREG. . L NEAR   00BDH  CODE PUBLIC  131 337#

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

intel

APPLICATION NOTE

AP-127

December 1982

Powerfail Considerations for Magnetic Bubble Memories

Dick Pierce
Applications Engineer
Intel Corporation

December 1982
ORDER NUMBER: 210300-001

INTRODUCTION

Intel has developed a new, comprehensive power-fail circuit that is incorporated into all Intel Bubble Board Memory products: BPK 72 Bubble Memory Prototype Kit, iSBX™ 251 MULTIMODULE™ board, and the iSBC® 254 MULTIBUS® compatible board. The use of this circuit also is recommended for all customer-designed bubble memory boards. The overall performance enhancements offered by this circuit include improved noise immunity and a factor-of-four reduction in the time required to shut down the bubble system.

Scope and Organization

In an effort to focus on implementation details, this application note is organized so that a reader can obtain sufficient information to implement a bubble design without an intimate working knowledge of the powerfail circuitry. However, for those interested, a complete detailed explanation of the integrated powerfail circuitry and the additional external circuitry is included. Appendix A contains a technical discussion of the effects of power loss on a Magnetic Bubble Chip. In addition, the previous circuit versions (Revision 0 and Revision 1), along with the present circuit, are completely documented and compared in Appendix B.

Bubble Memory Operation and the Powerfail Function

The power-fail circuitry is partially integrated into two of the five MBM support components, and additional required circuitry is provided by external components. Historically, several evolutionary improvements have been made in the external circuitry (see Table 1) to further reduce the risk of data loss following an abrupt power failure.

An essential feature of the bubble memory (MBM) is non-volatile data storage. This non-volatility results from two permanent magnets within the bubble device that produce a magnetic field (bias field) that maintain the magnetic domains, or bubbles (representing data) in the chip even when power is removed. The bubbles remain stationary in fixed positions until the data is accessed. To move the bubbles, an in-plane rotating magnetic field is induced by pulsing two mutually-perpendicular coils surrounding the bubble chip. Special conductor lines on the bubble chips provide all the current related functions for reading and writing to the bubble device. A special support IC produces current pulses (swap, relicate, and generate) to perform these functions. A complete set of support circuits provides the necessary timing and waveforms to precisely maneuver the bubbles to their desired positions. To prevent bubbles from moving to undesired positions, certain precautions must be observed.

As power is applied or removed, the system must prevent any current transients in the coils or bubble function conductors. If power is removed with the coils operating, the system must ensure that the coil currents are shut down in an orderly fashion to guarantee that the magnetic bubbles come to rest in stable, known positions. The powerfail reset circuit ensures that the system is powered up in an orderly manner and serves to alert the system should power fail. Both the power-up and

Table 1. Powerfail Reset Circuit Product History

Product	Powerfail Circuit Revision Level		
	0	1	2
BPK 72	July 1979 thru August 1982 Rev. A thru Rev. G	N/A	September 1982
iSBX™-251 Board	N/A	September 1981 thru October 1982	November 1982
iSBX-251C Board	N/A	N/A	July 1982
iSBC®-254 Board	December 1980 thru July 1982	July 1982 thru November 1982	November 1982

power-down sequences require a finite period of time to complete their functions until the sequence is complete. To allow proper execution of a power down sequence, the system voltages (+5V DC, +12V DC) must not decay to a level that prevents operation of the powerfail circuitry and critical bubble memory functions. In most power supply designs, adequate energy storage is available to provide enough "hold time" to complete an orderly shutdown. However, if dc power decays too rapidly sufficient time may not exist for a proper shutdown and may cause data to be lost within the MBM.

System Description

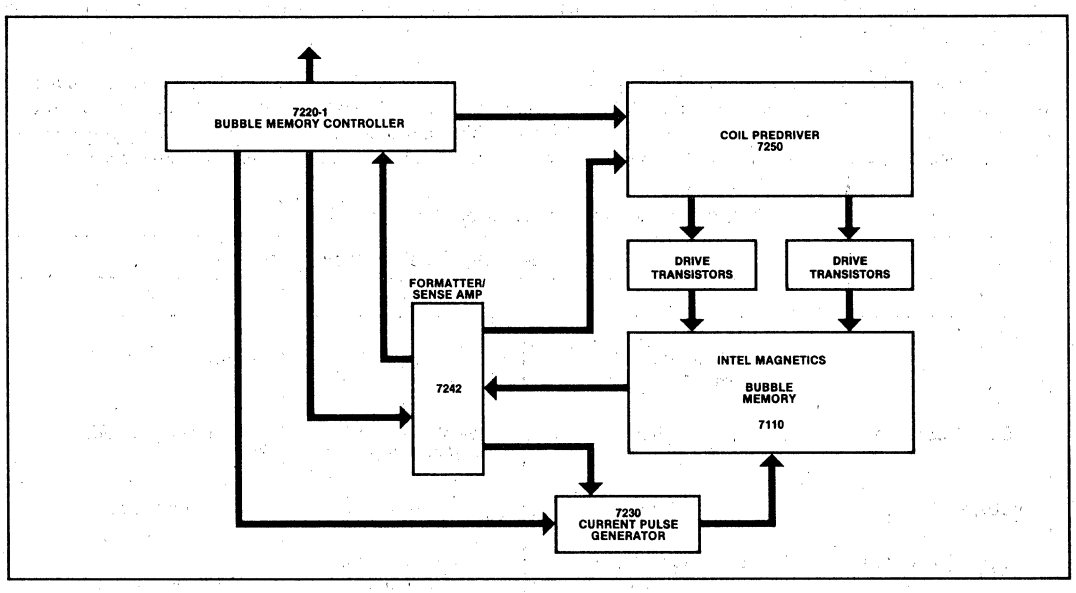
The basic Intel Bubble Memory system consists of one 7110 magnetic bubble memory and five integrated support components: a 7220-1 Bubble Memory Controller (BMC), a 7230 Current Pulse Generator (CPG), a 7242 Formatter-Sense Amplifier (FSA), a 7250 Coil Predriver (CPD), and two 7254 quad drive transistor packages. These support circuits are interfaced to the MBM as shown in Figure 1 to form the basic one megabit (128K byte) system. The support components provide all of the functions necessary for the storage and retrieval of data within the MBM. In addition, two of the support components, the 7220-1 BMC and the 7230 CPG, contain the integrated powerfail circuitry that facilitates proper power-up and power-down operations.

OVERVIEW — POWER UP/DOWN OPERATION

A block diagram of the power fail circuitry for the bubble memory system is shown in Figure 2. The following paragraphs provide an operational overview of the integrated powerfail circuit and the external circuit requirements.

During a power up sequence, the 7230 holds PWR.FAIL/* active (low) until both supplies are above the minimum required level. The 7230 contains power supply monitors (+5V and +12V) that determine when either supply falls below threshold level and activate PWR.FAIL/ signal accordingly. On power-up, the PWR.FAIL/ signal is delayed an additional 2 msec by an external RC network (time delay 1) to allow the 7220-1 substrate bias generator to fully charge. Following this delay, the positive-going transition on the 7220-1 PWR.FAIL/ input initiates a 7220-1 power-up sequence.

The RESET.OUT/ signal was designed to remain active during the power-up sequence and then to go inactive at the conclusion of the 50 μs power-up sequence. However, the RESET.OUT/ signal is indeterminate during execution of the 7220-1 power-up sequence. A second external RC network (time delay 2) derived from PWR.FAIL/ ensures that RESET.OUT/ is



*"/" denotes an inactive signal.

Figure 1. System Block Diagram

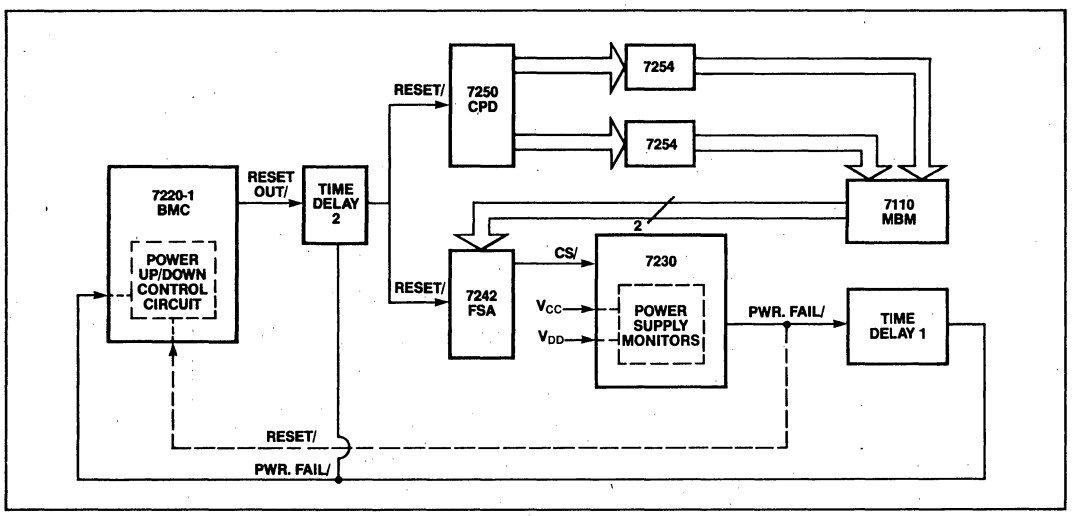


Figure 2. Block Diagram of Powerfail

held active (≤0.8V) during this time. The RESET.OUT/ signal occassionally will remain in its active state following a power-up sequence; accordingly the first command issued to the BMC during an initialization sequence must be an Abort command to ensure that RESET.OUT/ is deactivated.

The power-up sequence is designed to power the system up in an orderly fashion and to prevent any current transients from reaching the bubble device. The power-down sequence ensures that the coil drivers are shut down in the proper phase and that the support circuits are reset. When power fails, the 7230 notifies the 7220-1 by asserting the PWR.FAIL/ signal. The 7220-1 responds to a negative transition on either the PWR.FAIL/ input or the RESET/ input (external circuit revision level dependent) and initiates a power-down sequence. If the coils are active (i.e., bubbles propagating), the 7220-1 first terminates the coil drive control signals during the appropriate phase and then resets the support circuits by asserting the RESET.OUT/ signal. The two system supply voltages must not decay faster than the specified rates to ensure the RESET/ input to all the support circuits (excluding the 7220-1) reaches an active level (less than 0.8 volts).

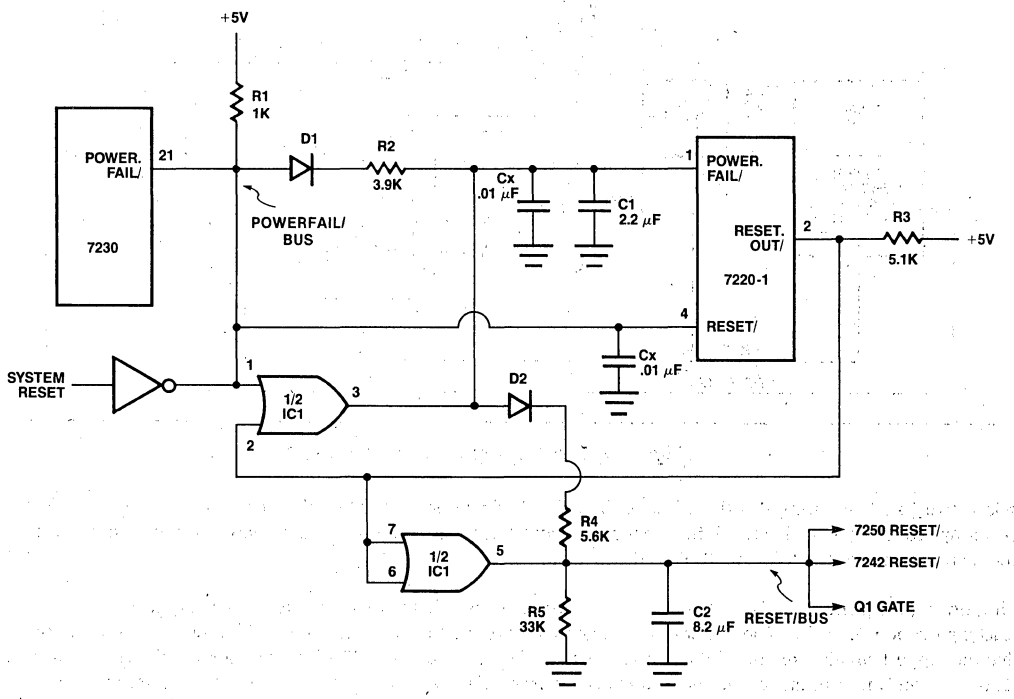
Powerfail Reset Circuit Solution

The external circuitry shown in Figure 3, in conjunction with the integrated circuitry contained in the 7230 and 7220-1, comprises the powerfail circuit (revision 2). This design contains six additional components compared to previous powerfail circuits and includes an 8-pin DIP IC (TI 75463).

This revised circuit has been fully developed and tested by Intel and currently is incorporated in many bubble products. Operational details are not required for the user to implement a custom design using the circuit in Figure 3. However, for any bubble memory designs that cannot conform to the recommended powerfail circuit, a reader must understand the system characteristics and requirements prior to choosing an alternative design.

The software implementation details to ensure correct powerfail circuit operation are shown in Figure 4. This routine should be implemented as a routine for cold start operation (application of power) and warm start operation (a RESET/ pulse applied to the 7220-1 BMC). The voltage decay rates shown in Table 2 also cannot be exceeded.

The power-up routine is based on the typical power-up timing shown in Figure 5. This timing does not assume that a system reset has been incorporated into the powerfail circuit. If the hardware reset line is used, the user must ensure that the 7220-1 RESET/ input is inactive before issuing the first Abort command. In addition, user software always must issue an Abort command every time the system is reset.



NOTES:
1. ALL RESISTORS 1/4 WATT, 5% TOL.
2. ALL CAPACITORS 10 VDC, 10% TOL.
3. D1,D2 ARE IN914 OR EQUIVALENT
4. IC1 IN75463
5. Cx IS OPTIONAL—RECOMMENDED FOR EXTREMELY NOISY POWER SUPPLY SITUATIONS
6. SYSTEM RESET MAY BE USED AS SHOWN PROVIDED OUTPUT OF INVERTER IS OPEN COLLECTOR.

Figure 3. External Powerfail Circuit Solution

Table 2. Power Supply Decay Rate Specifications During Power-down or Power Failure

Power Down/Powerfail Decay Rate			
V_{CC} (volts/msec)		V_{DD} (volts/msec)	
Min.	Max.	Min.	Max.
None	0.45	None	1.1

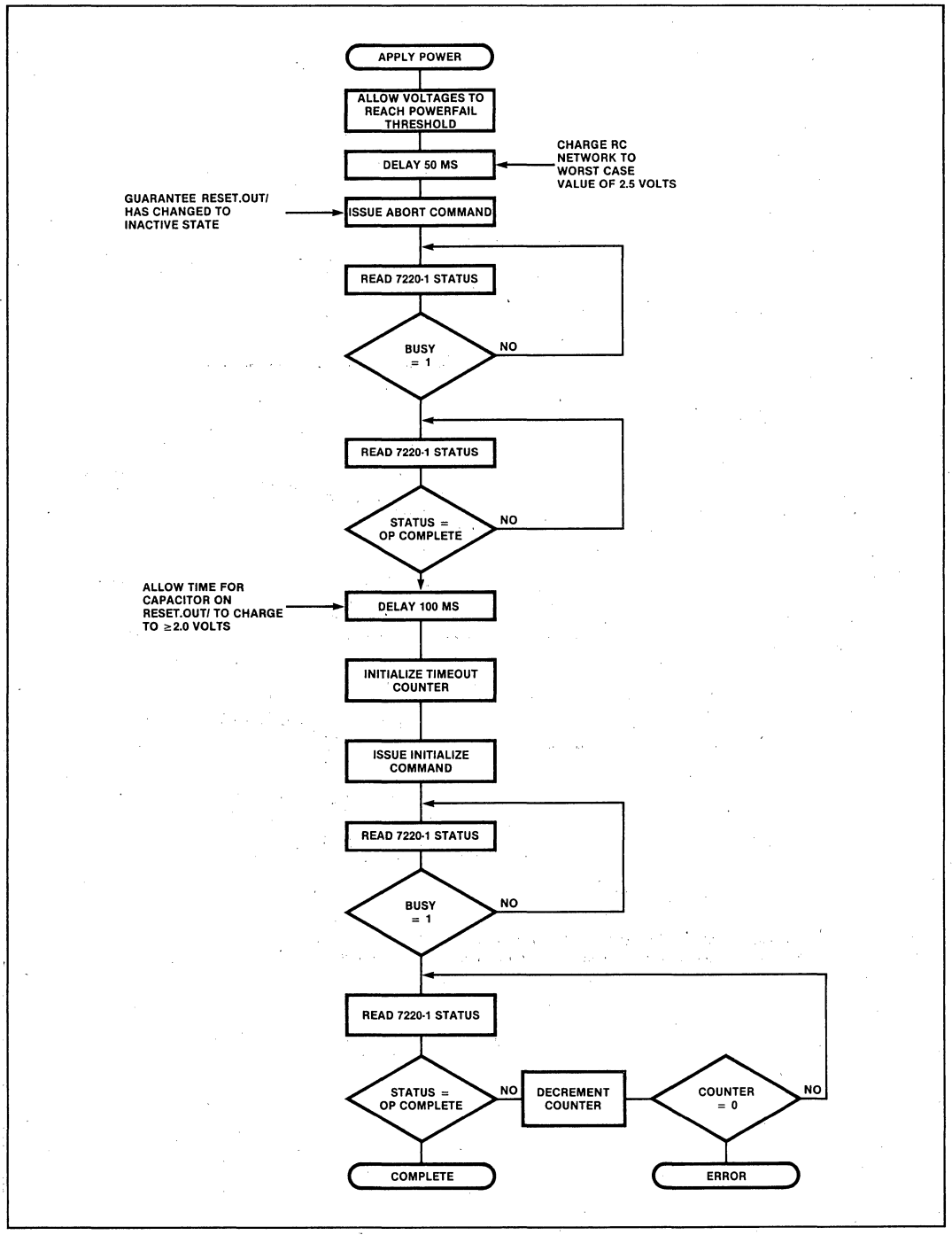
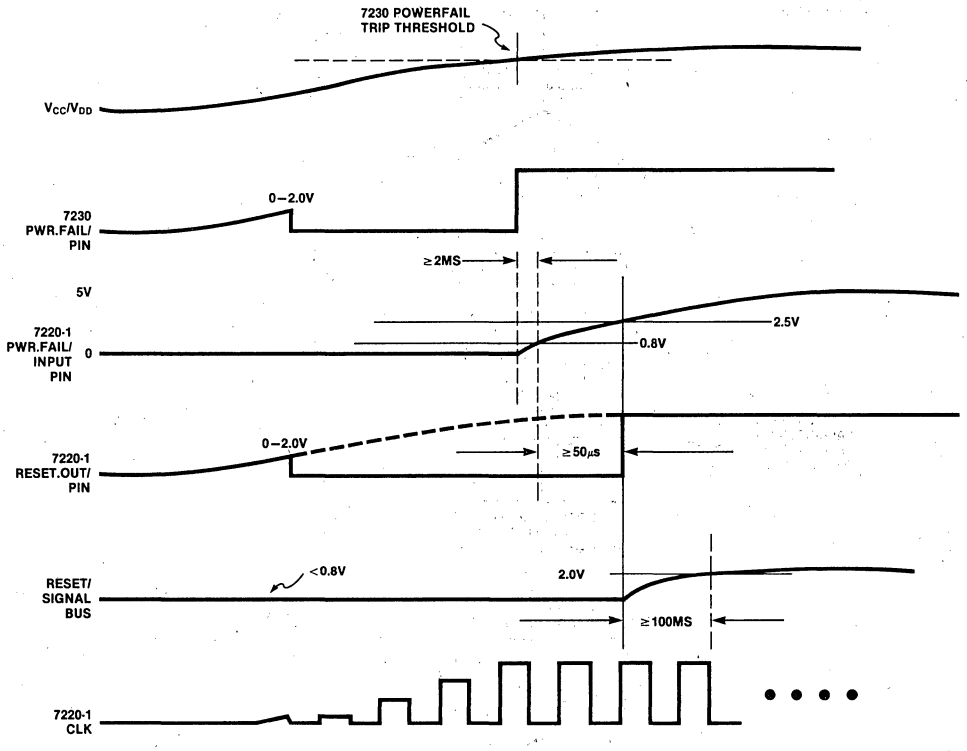


Figure 4. Power-up Flowchart



NOTES:
1. 7220-1 INPUTS SHOULD NOT LEAD V_{CC} BY GREATER THAN 0.5 VOLTS.
2. THE ABOVE SEQUENCE APPLIES ONLY IF THE SYSTEM RESET INPUT IS INACTIVE.

Figure 5. Power-up Timing for Powerfail Reset Circuit (Revision 2)

The worst case power-down timing sequence is also included in Figure 6. The total system power-down time varies according to whether the coils are active (i.e., rotating magnetic field is on) or inactive. The worst case power-down sequence is guaranteed to be completed provided that the above voltage decay rates are met.

INTEGRATED POWERFAIL CIRCUIT CHARACTERISTICS

Introduction

The following section provides an in-depth look at the input and output characteristics of the support circuits that contain the integrated powerfail circuitry. A complete understanding of these characteristics establishes the groundwork necessary for the detailed description of the overall powerfail circuit operation that follows.

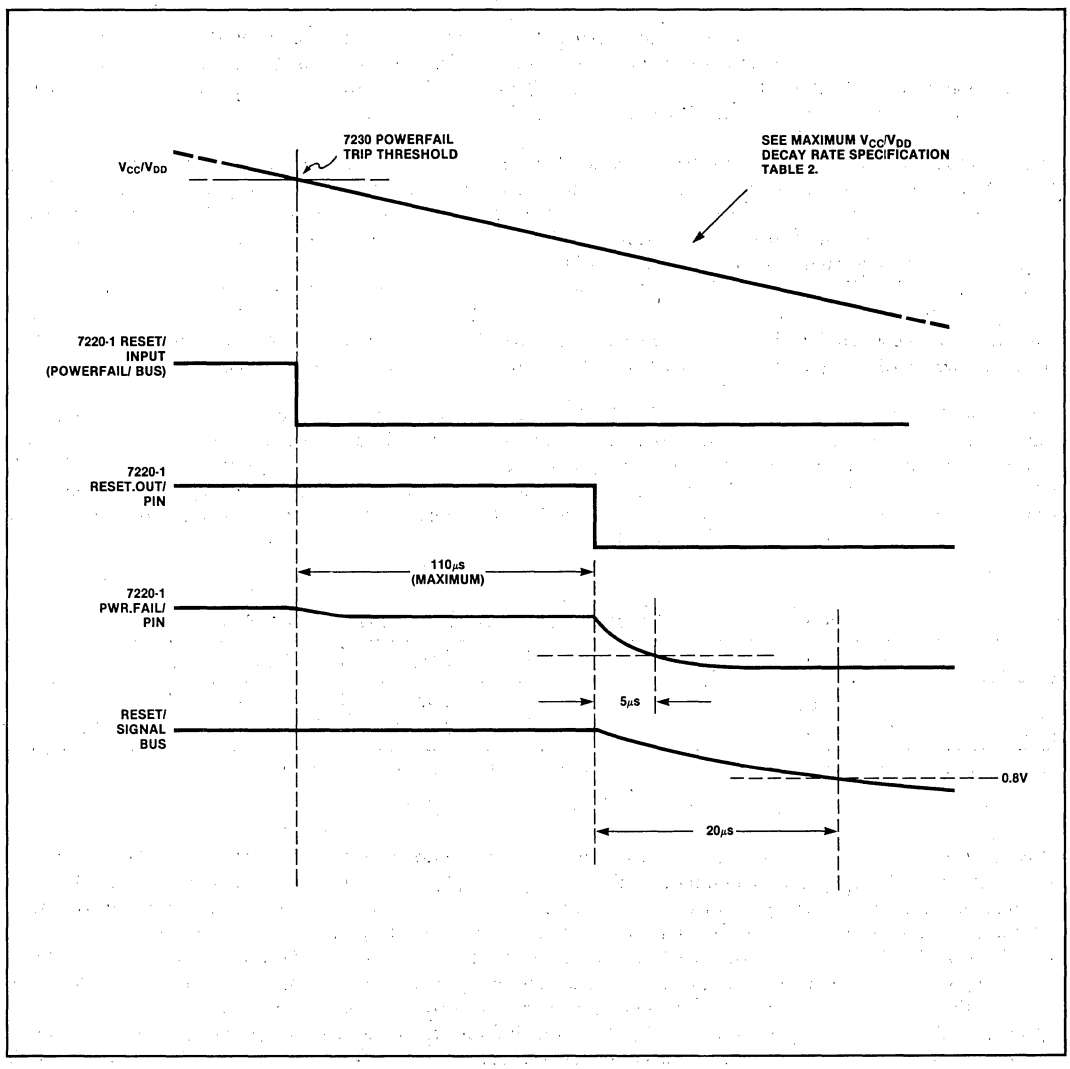


Figure 6. Power-down Timing for Powerfail Reset Circuit (Revision 2)

7230 PWR.FAIL/ OUTPUT

The 7230 Current Pulse Generator PWR.FAIL/ output is responsible for indicating when the system supply voltages (+5V, +12V) reach correct operating levels. During power up, normal operation, and power down, an internal zener reference comparator circuit within the 7230 senses both V_{CC} and V_{DD} and indicates when both levels are above approximately 92 percent of their nominal values. An active state on PWR.FAIL/ indicates one or both dc voltages are below this threshold. The PWR.FAIL/ output is an active-low, open-collector output requiring an external pullup resistor.

The PWR.FAIL/ output is asserted (active low) as power is applied until the +5V and +12V supplies both reach approximately their 92 percent levels at which point the 7230 output transistor switches off to allow the PWR.FAIL/ signal to rise to an inactive level governed by an external RC network. The RC networks on the PWR.FAIL/ line must hold the PWR.FAIL/ signal at an active level for at least 2.0 milliseconds to guarantee adequate time for the BMC to power up. The 7230 PWR.FAIL/ output then will remain inactive until one or both system voltages fall below the threshold.

The PWR.FAIL/ output is not an internally latched signal. In other words, the output responds immediately to any transition through the threshold (trip point). The disadvantage to this excellent response capability is that the output will toggle on transitions through the threshold. Systems should be designed to avoid an extremely noisy power supply or temporary power loss that could cause the PWR.FAIL/ signal to pulse for a very short duration.

During temporary power loss in Revision 0 and Revision 1 circuits, the PWR.FAIL/ input to the 7220-1 could pulse below V_{IH} (2.5 volts) and initiate a power down sequence. The 7220-1 PWR.FAIL/ input should remain active until the entire power down sequence is completed (maximum 110 μsec). As detailed later in the 7220-1 PWR.FAIL/ input description, if the 7220-1 PWR.FAIL/ input goes inactive during execution of a power down sequence, the sequence is immediately terminated. This type of termination can stop the drive field in the wrong phase and compromise bubble data. The solution is to use the 7220-1 RESET/ input to initiate a power down sequence rather than the 7220-1 PWR.FAIL/ input.

Two important considerations in properly designing a powerfail circuit are 1) the accuracy of threshold trip point of the 7230 PWR.FAIL/ output and 2) the behavior of this output at low voltages (below 2 volts).

The worst case threshold level that the 7230 PWR.FAIL/ output will trip must be above the worst case operating limits of the support circuits with an additional margin to allow for an adequate period of time to complete a power down sequence (worst case 110 microseconds for revision level 1 and 2 powerfail reset circuits). In the case of the 7230 CPG and the 7110 MBM, which both have a ±5% voltage specification for V_{CC} and/or V_{DD}, special powerfail characteristics are applicable. As shown below, (Table 3) only critical bubble memory functions are guaranteed at these supply values and not full memory operation.

Table 3. Powerfail Characteristics for 7230 Threshold Trip Point*

Symbol	Min.	Typ.	Max.
V_{CC} TH	4.43V	4.60V	4.70V
V_{DD} TH	10.75V	11.10V	11.28V

*Powerfail characteristics apply to 7110 bubble memory data integrity only and not to full memory operation.

Second, the 7230 PWR.FAIL/ output cannot be guaranteed active (low) until V_{CC} reaches about 2.0 volts since the output transistor is not operational until that point. As V_{CC} is applied, the output is not active and will track (follow within a few tenths of a volt) V_{CC} until V_{CC} reaches approximately 2.0 volts. At this point, the output transistor turns on and the output goes active (low) and remains low until the system voltages both reach the threshold trip point as described earlier. A similar response occurs as power is removed. The output transistor turns on and pulls the output active (low) at the threshold point and remains turned on until V_{CC} reaches approximately 2.0 volts where the output goes inactive (transistor not operating). This operation must be controlled on power-up and depends on the rate of rise of system voltages. This is because the PWR.FAIL/ output is indirectly connected to the RESET/ input of the support circuits (7250 and 7242 and Q1 reference current switch) through two RC networks in Rev. 0 and Rev. 1 power-fail circuits. These inputs can rise to as much as 1.5V before the 7230 PWR.FAIL/ output turns on, which is above V_{IL} max-

imum (0.8V) thus potentially enabling these circuits. This could result in current transients reaching the drive coils or bubble function conductors and move bubbles from their rest position resulting in data loss. Observing the rate of rise specifications protects against this possibility . The revision 2 powerfail circuit eliminates this problem and has no rate of rise limitation.

7220-1 PWR.FAIL/ INPUT

The 7220-1 PWR.FAIL/ input serves a dual function; a positive transition initiates a power-up sequence while a negative transition initiates a power-down sequence of the bubble memory system. In order for the 7220-1 to become fully functional an on chip back-bias generator must fully charge the 7220-1 substrate. Therefore, before any sequence can be executed, including the power-up sequence a time delay is required. An external RC delay on the PWR.FAIL/ input ensures this input is held low (<0.8V) at least 2.0 milliseconds after V_{CC} has reached the 7220-1 voltage specification range.

The power-up sequence is initiated once the RC network charges to a point where the 7220-1 recognizes a positive transition on the PWR.FAIL/ input. From a cold start (application of power), a positive transition must occur or the controller will not power-up correctly. Once the power-up sequence is completed, the RESET.OUT/ is designed to be released, however, two possible exceptions exist. First, if the 7220-1 RESET/ is held low during power-up, the 7220-1 internal power-up sequence will be completed however RESET.OUT/ will not be released until RESET/ is inactive. Second, the 7220-1's internal RESET.OUT/ output transistor may remain turned on dependent upon the power-up status of certain internal 7220-1 flip-flops. Because of this an ABORT command is always necessary to internally reset these flip-flops, in turn ensuring release of the RESET.OUT/ output.

If the 7220-1 BMC does not receive a positive transition on PWR.FAIL/ during power-up, a power-up sequence is not initiated. This leaves the controller in an unknown state. In this unknown state the controller cannot communicate properly with the data and control inputs. This can only occur as a result of:

1. **"Brown out"** — short duration of power failure in which power drops below specified levels.

2. **Power-up circuit failure** — The PWR.FAIL/ pin never reaches V_{IH} (minimum) of 2.5 volts.

The above conditions are resolved by ensuring a positive transition occurs on the PWR.FAIL/ input during power-up and after brownout. It is necessary to execute a power-up sequence even though power to the system is only interrupted momentarily in order to restore the 7220-1 to the required internal state.

Once the PWR.FAIL/ positive transition has occured, this input should remain in the inactive state (V_{IH} > 2.5V) as long as power is applied to the system. If power is removed, it is the negative transition of this input which intitiates the second function, power down. The function can also be initiated with the RESET/ input of the 7220-1.

An important consideration is how the 7220-1 PWR.FAIL/ input distinguishes between positive and negative transitions. On power up (positive transition), crossing the input threshold (typically 1.6V to 1.9V) a pulse is generated internally which resets the 7220-1 to a known state and initiates a power-up sequence. On power down (negative transition), crossing the input threshold (typically 1.35V to 1.6V with the designed- in hysteresis) the signal initiates a power-down sequence. If a power-down sequence has been initiated, a positive transition must not inadvertently occur on the 7220-1 PWR.FAIL/ input prior to the power-down sequence completion. A positive transition internally generates a reset pulse (to halt any current BMC activity) and initiates a power-up sequence effectively terminating a power down sequence. The result is a possibility of shutting the coil drives down in the improper phase resulting in data loss in the MBM.

The PWR.FAIL/ input has built in hysteresis to reduce the susceptibility to multiple threshold crossings or glitching. However, the values of hysteresis range from 50 mV to 400 mV. To improve noise and power fluctuation immunity, the use of PWR.FAIL/ input for initiating a power down sequence was abandoned in Revision 1 and Revision 2 circuit designs. The 7220-1 RESET/ input is used instead to initiate power down (see next section.)

7220-1 RESET/ INPUT

The 7220-1 RESET/ input, when asserted, will terminate any current BMC activity and initiate a RESET sequence (identical to the sequence initiated by the PWR.FAIL/ input going active). After the sequence is concluded, the RESET.OUT/ is activated to reset the MBM support circuitry. RESET.OUT/ will remain active until RESET/ is inactive.

The RESET/ input is a level sensitive latched input. This is a distinct advantage over the PWR.FAIL/ input; where any fluctuations of the input once the signal was recognized could possibly terminate the power down sequence. The RESET/ input is latched on the negative edge of the BMC clock and must be active low (< .8V) for at least one clock period (250ns) to guarantee recognition.

7220-1 RESET.OUT/

The RESET.OUT/ output has two functions: 1) to guarantee the bubble memory system is disabled during power-up and after power down of the bubble memory system and 2) to provide a pulse (reset) to the support circuits during normal operation. Since the RESET.OUT/ output is an active low open drain, it requires an external pullup resistor to V_{CC}.

The support circuits controlled by RESET.OUT/ are the 7250 Coil Predriver, the 7242 Formatter Sense Amplifier, and a VMOS transistor switch which enables a reference current for the 7230. These circuits must be disabled during the entire power-up sequence and immediately following the conclusion of a power-down sequence to prevent any current transients or extraneous enable pulses. Data loss is a possible consequence should the support circuits not remain disabled during power cycling.

During power up the RESET.OUT/ signal can not be guaranteed active (low) until the 7220-1 power-up sequence has executed. Therefore, external circuitry must assure RESET.OUT/ does not rise above V_{IL} maximum (.8V) until 50 μs after initiation of the power-up sequence. By ensuring the RESET.OUT/ is active during power-up it guarantees the support circuits are reset to a known state. The 7220-1 BMC is designed with the capability to reset the support circuits during normal operations by pulsing the RESET.OUT/ 750 μs (3 clock periods). This pulse can occur as the result of two user issued commands to the BMC: an INITIALIZE command and an MBM PURGE command.

The external RC network on the RESET.OUT/ signal prevents the RESET.OUT/ pulse from going active during its 750 μs duration. In spite of an inability to reset the support circuits by issuing the proper command, correct operation is guaranteed since the support circuits only require a one time reset signal at power-on.

Additional Bubble Memory Controller Inputs

The 7220-1 has several additional inputs that could indirectly affect power up operation. It is important that the user exercise caution and adhere to all requirements to ensure proper power-up operations. The following outlines those requirements.

CLK (CLOCK)

The CLK input of the 7220-1 must be present when the positive power up transition occurs at the 7220-1 PWR.FAIL/ input. This requirement allows the BMC to properly execute a power-up sequence. The input requirements are a precise 4MHz (±.1%) with a 50 percent duty cycle (±5%).

DACK/ (DATA ACKNOWLEDGE)

The DACK/ input is normally used in conjunction with an INTEL DMA controller chip (8257 or 8237) which automatically provides drive for this input. However, if DMA is not used a 5.1K pullup resistor to V_{CC} is required. This requirement prevents erratic BMC operation.

WAIT/

The WAIT/ input must also be guaranteed inactive through an external 5.1K pullup resistor. It is designed to be used in parallel controller applications to maintain synchronization between controllers should an error be detected in one during a data transfer.

CS/, RD/, WR/, A0, D0-D8

These inputs require no special considerations other than to observe the V_{IH} minimum specification. This specification prevents an incorrect power-up sequence execution.

ENERGY STORAGE REQUIREMENTS

The data integrity and non-volatility of the MBM during power down operations is guaranteed by design provided the voltage decay rates specifications for both V_{CC} and V_{DD} are observed. Most commercially available power supplies provide enough energy storage to fulfill these requirements. However, some applications may exist where the bubble memory could suddenly become disconnected from the dc supply; a case where the power supply energy storage is not of value. In these special applications, the local onboard capacitance must meet the hold up time requirement.

The worst case onboard capacitance values can be determined according to the following equation:

$$C = \frac{Q\ max}{V\ min} = \frac{I\ max\ \Delta T\ max}{\Delta V\ min}$$

A worst case calculation must include the following considerations: 1) If any additional circuitry exists on the pc board that uses the same power supplies, the additional current drain must be accounted for and 2) the worst case (minimum) threshold trip point of the 7230 is used.

The capacitance required on a pc board containing one / megabit bubble memory system is calculated as follows:

$$C_{5V} = \frac{366 \times 10^{-3}\ amp \times (110 \times 10^{-6}\ sec)}{0.01 \qquad x\ 5\ volts} = 805\ \mu F$$

$$C_{12V} = \frac{381 \times 10^{-3}\ amp \times (110 \times 10^{-6}\ sec)}{0.01 \qquad x\ 12\ volts} = 350\ \mu F$$

Supplemental Powerfail Sensing

In many systems, additional signals are available that provide advanced warning of an imminent power failure or the existence of an abnormal condition prior to actual loss of dc power (e.g., AC powerfail sensing, AC or DC overvoltage, ambient over/under temperature). These signals are easily incorporated into the powerfail circuit design via an open-collector gate or inverter connected to the PWR.FAIL/ signal bus.

The advantage of utilizing these signals is the bubble memory system can complete a power down sequence prior to losing dc power. However, local dc powerfail sensing is always required due to the possibility of local dc power loss without the loss of AC power.

Noise Effects of Powerfail Circuit Operation

The 7230's powerfail voltage monitoring function is implemented internally with two independent, logically-OR'ed voltage comparators. The comparators respond quickly to a sudden loss of V_{CC} or V_{DD} and therefore can respond to noise transients on the power supply lines that cross the comparator switching threshold. As much as 100 mV of noise

from coil drive switching is not uncommon. Note that the operating power supply tolerance for all INTEL Bubble Memory products is ±5% including up to 50 mV of noise on the power supply lines. This tolerance should not be confused with the operation of the powerfail circuit beyond the normal operating range during power-down operation.

To minimize "nuisance" activation of the PWR.FAIL/ signal bus, ample high frequency decoupling on the 7230's V_{CC} and V_{DD} pins should be provided. Typically, 0.01 μF to 0.1 μF ceramic disk or mica capacitors are sufficient. Another source of unwanted powerfail circuit activation is noise that is coupled directly onto the PWR.FAIL/ signal bus. This noise is minimized through good printed circuit layout practices and, if required, by the inclusion of a small capacitor directly on the PWR.FAIL/ bus. This capacitor slightly increases the power-down time and should be kept as small as possible (0.01 μF maximum).

APPENDIX A

TECHNICAL DISCUSSION OF POWER LOSS EFFECT ON 7110

The effects of power loss on an MBM are best understood by describing the way in which the device functions and the way in which it can lose data.

A magnetic bubble memory device (See Figure 7) consists of a bubble memory chip, two mutually-perpendicular coils, two permanent magnets, and a shield to provide protection from interference by external magnetic fields. The two permanent magnets produce an external magnetic field (bias field) that maintains the magnetic domains, or bubbles, in the chip even when power is removed. To move the bubbles, an in-plane rotating magnetic field is induced by pulsing the two mutually-perpendicular coils.

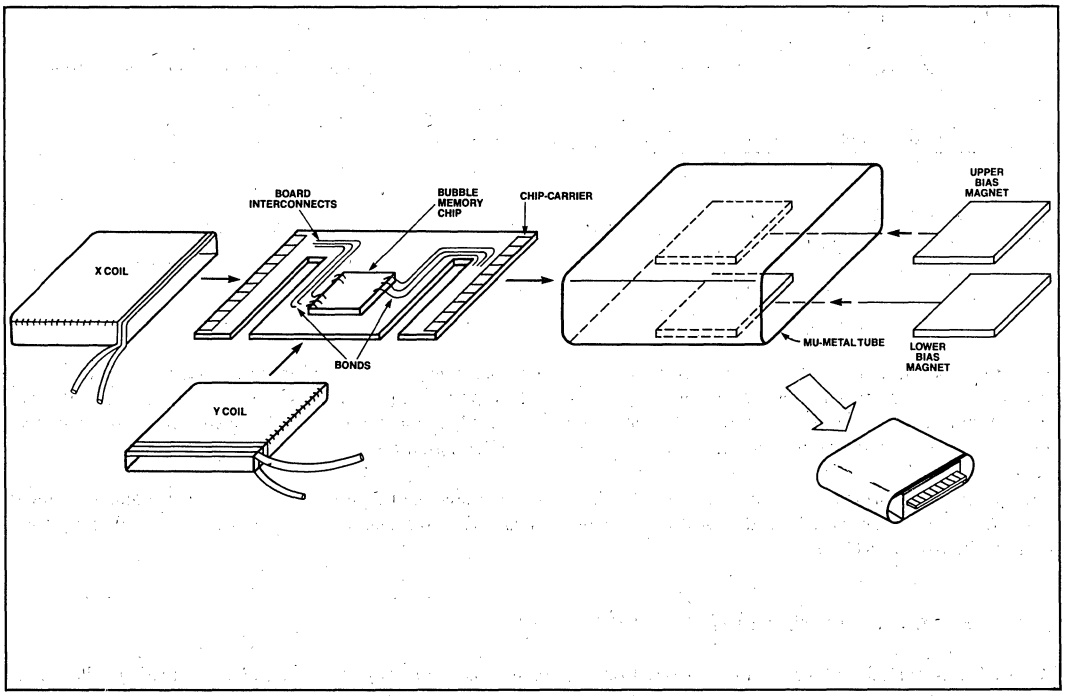


Figure 7. Device Break-down

The bubble memory chip itself consists of a thin magnetic garnet crystal film grown on a non-magnetic gadolinium-gallium-garnet substrate. This thin film possesses a property that magnetic moments associated with each atom in the single crystal structure have only two possible directions: an upward or downward direction perpendicular to the plane of the film. This constraint in direction results in only two conditions of magnetization (see Figure 8). These magnetic moments tend to group themselves together into magnetic domains. The size and shape of the domains are determined primarily by a balancing of several forces that minimize the sum of magnetic energy.

Without an external field, the film surface area of upward domains is equal to that of downward domains and there is no net magnetic field within the plane of film. Application of an external magnetic field perpendicular to the film causes domains to line up in the direction of the field. As the external field is increased, the downward domains enlarge while the opposing (upward) domains shrink until they finally are reduced to a cylindrical shape. This microscopic magnetized cylinder opposing the externally applied field is a magnetic bubble. Within the magnetic film, the presence of a magnetic bubble represents a binary one and the absence of a magnetic bubble represents a binary zero.

The memory function is provided by the bubble. However, an organized means is needed to propagate the bubbles along certain paths and to provide storage sites. A soft ferromagnetic material (permalloy) is deposited on the thin garnet film in C-shaped patterns. These patterns are arranged to form shift-register like loops that provide the means to store and move bubbles. Each pattern is magnetized according to the rotating magnetic field, and the polarity of each pattern changes instantaneously as the rotating magnetic field vector changes. The rotating field is generated by driving the X and Y coils with triangular-waveform currents, one lagging the other by 90° in phase. A magnetic bubble propagates from one storage site (permalloy pattern) to the next for every 360° of rotation of the rotating field. Each storage site has a preferred position (home) for the bubble to reside corresponding to zero degrees of the rotating magnetic field. All bubbles start, stop and are stored in this position.

In the event of power failure, it is important that the rotating magnetic field is shut down in the proper phase (i.e., 0°). If an orderly shut down is not complete, the rotating field may be shut down in an improper phase that causes bubbles to stop in an unstable position within the storage loops. When this type of stoppage occurs, the bubbles either will come to rest in another, but incorrect, stable position or will leave their original storage loop (possibly contaminating valid data in another storage loop).

As power is applied, it also is important that the rotating magnetic field does not move (i.e., current transients must be prevented from reaching the coils). This function also is provided via the powerfail circuitry. Thus, the purpose of the powerfail circuitry is twofold 1) to prevent any current transients from reaching the X-Y coils or bubble function generators and 2) to halt the coils in proper phase should power fail.

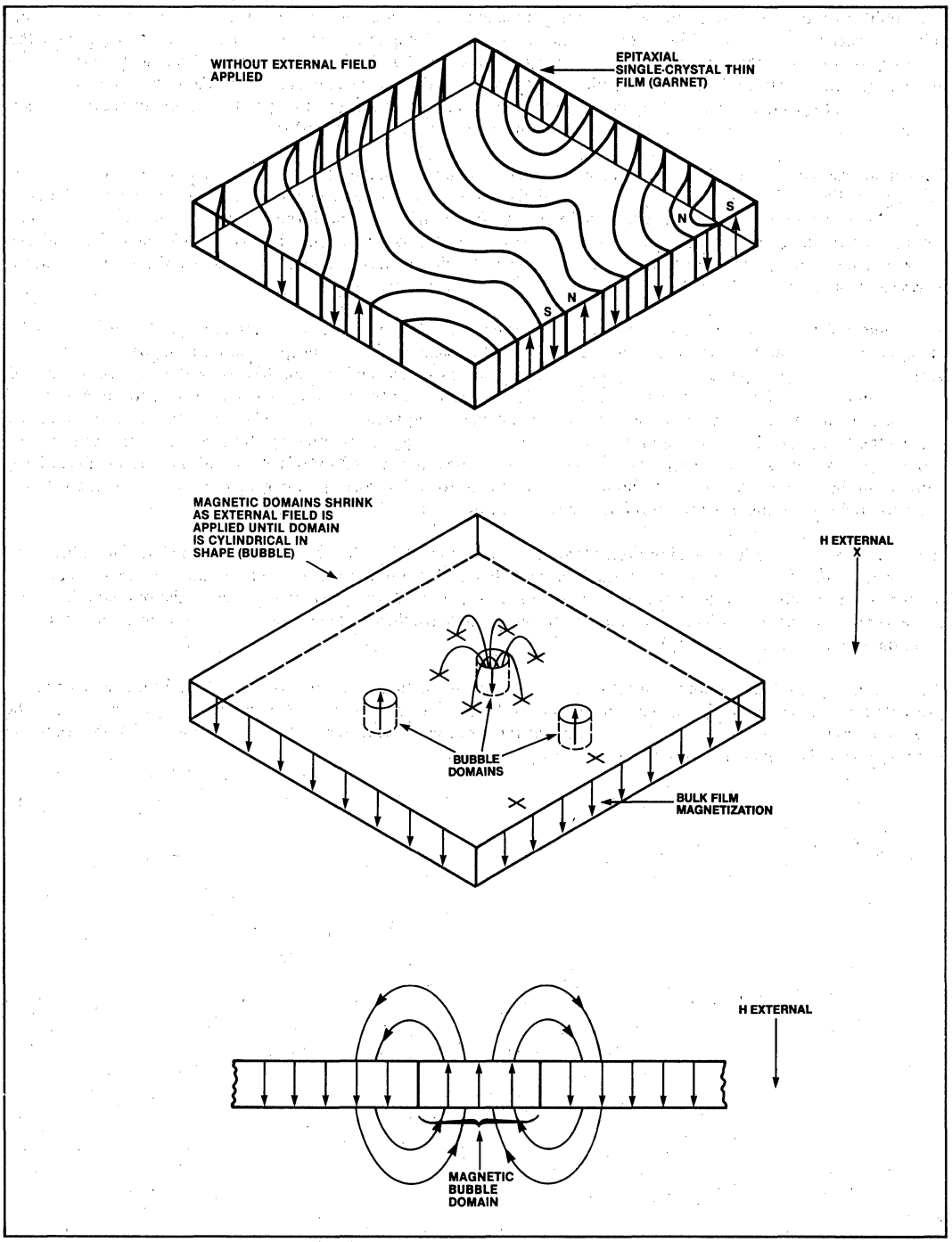


Figure 8. Device Magnetization

APPENDIX B

DETAIL POWER CIRCUIT DESCRIPTION

As discussed in the Introduction, the powerfail reset circuit actually consists of two portions — an integrated section and several additional external components. The degree to which external disturbances (noise, power fluctuations) influence system performance depends heavily on the system environment and configuration. Consequently, the reliable analysis of their effect on system performance is difficult and generally is best accomplished by measurement. In this Appendix, each revision level of the powerfail reset circuit is detailed. Several timing diagrams based on measurement and computer simulation also are included.

Powerfail Reset Circuit — Revision 0

Summary

The overall performance of the powerfail reset circuit (revision 0) is adequate provided that a specific set of conditions is observed. The requirements are summarized below (Table 4). Noise is also a concern. System generated noise is typically low level and can usually be neglected in portions of the circuit where the signal levels are high. Often, however, bubble systems generate significant levels of noise in a system where signal levels are low. Even low-level noise can degrade overall bubble memory system performance.

Table 4. Power Supply Requirements for Powerfail Reset Circuit (Revision 0)

	V_{CC} (volts/msec)		V_{DD} (volts/msec)	
	Min.	**Max.**	**Min.**	**Max.**
Power-Up Voltage Rate of Rise	0.11	None	None	None
Power-Down/Power Failure Decay Rate	None	0.70	None	.15

Noise, power fluctuations, and a rapid decay of voltage are the primary contributors to the incorrect operation of the first powerfail reset circuit (revision level 0). Since noise and power fluctuations are unavoidable in most practical systems, techniques for minimizing these effects were developed for subsequent circuits. Note that no bubble memory is immune to extremely abrupt removal of dc power. All bubble memory systems require a minimal amount of time to effect an orderly shutdown in order to maintain data integrity.

Subsequent circuit designs have been implemented to minimize system requirements by reducing the overhead required to power-down the bubble system.

The most serious fault of any powerfail reset circuit is where bubble memory data integrity is jeopardized. The first powerfail reset circuit design (revision 0) could not prevent data loss when:

1) Power was removed too rapidly for the system to ensure proper power-down.

2) Power was applied too slowly.

3) Multiple threshold crossings or "glitches" occured on the 7220-1 PWR.FAIL/ input while the coils were active.

The first two conditions can be easily prevented by following the requirements shown in Table 4. The third condition was difficult to reliably prevent and was the motivation for the revision of the circuit.

Power-up

When power initially is applied to the system (Figure 9), the PWR.FAIL/ signal is designed to be asserted by the 7230 CPG until both V_{CC} and V_{DD} reach approximately 92 percent of their nominal values. Referring to Figures 9 and 10, the 7230 internal PWR.FAIL/ output transistor cannot be guaranteed operational until V_{CC} reaches approximately 2.0 volts. During this indeterminate state of the output transistor, the floating output lags V_{CC} by approximately 0.7 volts. Therefore, the RC networks on the PWR.FAIL/ signal line (R1/C1 and R2/C2) begin charging immediately after power is applied. They continue to charge until the 7230 PWR.FAIL/ output transistor turns on. The 7230 PWR.FAIL/ output goes inactive (transistor off) when both supplies have reached the power-fail trip point. Since the RESET/ input of the 7242 FSA and the 7250 CPD are tied via the R1C1/R2C2 network to 7230 PWR.FAIL/ output, these support circuits potentially could be enabled if the 7230 PWR.FAIL/ output were allowed to rise above V_{IL} (0.8 volts). A current transient then could activate the MBM coils or bubble function conductors and cause bubbles to move to an unstable position. Note that a slow power-on ramp would be the only condition that could prematurely enable the support circuits.

Once V_{CC} reaches approximately 2.0 volts, the PWR.FAIL/ output transistor turns on to pull the PWR.FAIL/ signal low until both V_{CC} and V_{DD} reach the powerfail trip point. When the trip point is reached, the output transistor is turned-off and the PWR.FAIL/ signal is allowed to rise to the inactive level. The RC networks continue to hold the PWR.FAIL/ signal at an active level for at least 2.0 milliseconds after V_{CC} and V_{DD} have reached the trip point level. The RC delay ensures adequate time for the 7220-1 BMC's substrate bias generator to become fully operational and fully charge the 7220-1 substrate to its operational bias voltage. At some time before the PWR.FAIL/ signal reaches the 7220-1 V_{IH} (maximum) of 2.5 volts, the 7220-1 power-on initialization sequence starts. Up to this point, the 7220-1 is in an indeterminate state and the RESET.OUT/ signal, which is derived from the PWR.FAIL/ signal should be active. The behavior of the RESET.OUT/ signal, however, is similar to the 7230 PWR.FAIL/ output at low V_{CC} (below approximately 2.0 volts). As V_{CC} is slowly applied to the system, the RESET.OUT/ output transistor initially is inactive and the pullup resistor forces this output to follow 7220-1 PWR.FAIL input. Once V_{CC} reaches approximately 1.8 volts, the output transistor should turn on (RESET.OUT/ active) and remain active until completion of the power up sequence. During the inactive period, the RESET.OUT/ signal is capable of reaching the inactive level and potentially enabling the support circuits prematurely.

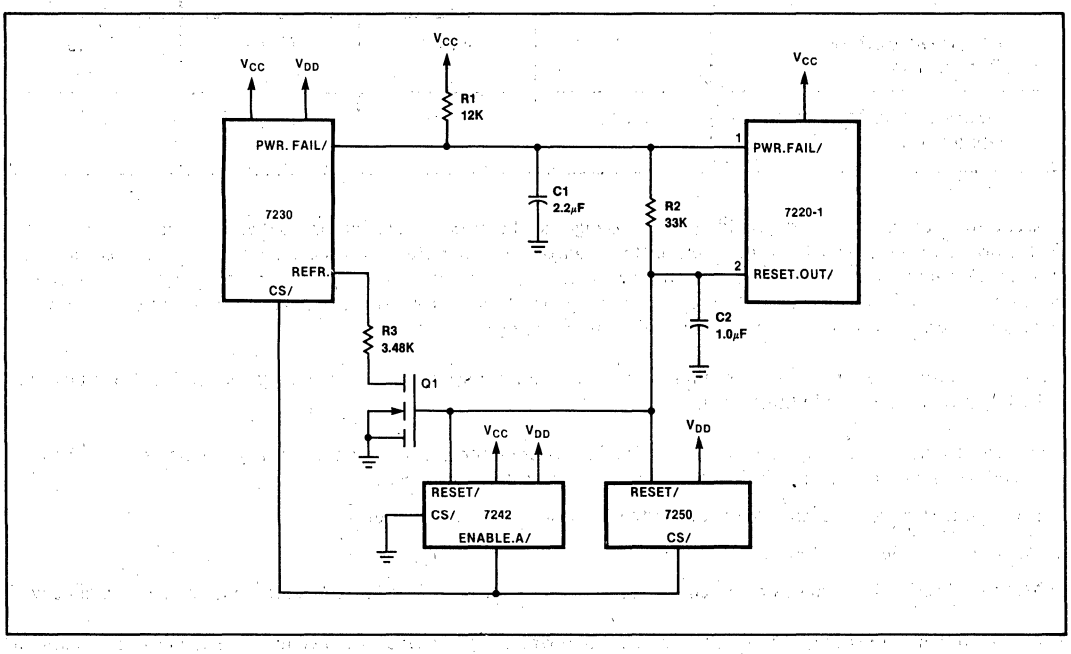


Figure 9. Revision 0 Circuit

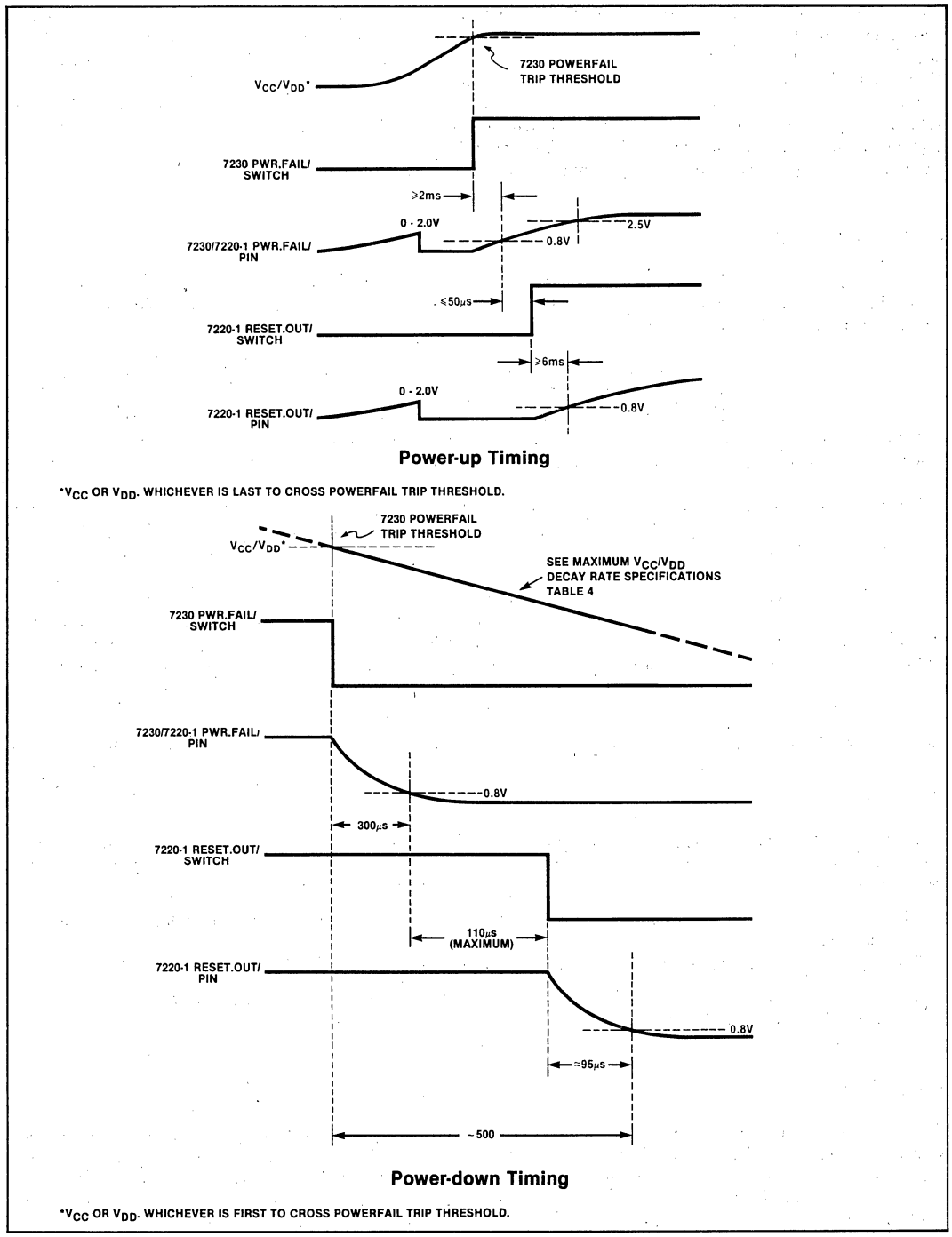


Figure 10. Power-up/Power-down Timing (Revision 0)

At the completion of the power-on initialization sequence, the 7220-1's internal RESET.OUT/ output transistor should be allowed to turn off. However, depending on the power-up state of certain internal 7220-1 flip-flops, this output may remain active. An Abort command is capable of internally resetting these flip-flops and releasing the RESET.OUT/ output to allow it to rise to the inactive level as determined by the R2/C2 delay network. When RESET.OUT/ reaches its inactive level, the 7242 FSA and 7250 CPD RESET/ lines are deactivated and 7230 current reference switch Q1 is turned on. The 7242 ENABLE.A/ line, which is controlled by the 7220-1, may now be activated; when active, this line enables the 7230 CS/ and 7250 CS/ (chip select) lines. The system now is fully operational and ready to execute an Initialize command (provided the Abort command had been issued).

Power-down Operation

If either V_{CC} or V_{DD} falls below the 7230 powerfail trip level, the internal PWR.FAIL/ signal in the 7230 is asserted immediately. However, due to the charge on capacitor C1 in the power-up delay network, the PWR.FAIL/ signal is prevented from reaching the active low level until C1 discharges to V_{IL} (maximum 0.8V).

When the PWR.FAIL/ signal level reaches the logic low-level threshold of the 7220-1's PWR.FAIL/ input, an internal power-down sequence is initiated within the 7220-1. As discussed earlier in the 7220-1 PWR.FAIL/ input description, the 7220-1 PWR.FAIL/ input cannot tolerate any positive threshold crossings during the power-down sequence. If a positive transition should occur; a power-up sequence will be initiated taking precedence over the power-down sequence currently in progress, and this unorderly shutdown could result in the loss of data.

The execution time of 7220-1 power-down sequence varies according to whether the coils are active (i.e., rotating magnetic field is on) or inactive. If the rotating field is off, the power down sequence is completed in approximately 10 microseconds. If the rotating field is on and a swap operation has not been initiated, the worst-case power-down time is increased to 26 microseconds; if a swap operation has been initiated, the power-down time sequence requires a maximum of 110 microseconds. The power-down time is shown in Figure 10. Note that the total system power-down time, since the operation is not complete until the RESET.OUT/ signal line is asserted, is the sum of the 7220-1's internal power-down sequence time and the discharge times for capacitors C1 and C2. To ensure proper operation of the bubble system for data integrity during power-down operations, the power supply maximum decay rates must be observed.

Powerfail Reset Circuit — Revision 1

Summary

The powerfail reset circuit (revision 1) was designed to reduce the requirements placed on the revision 0 powerfail reset circuit and to further reduce the risk of data loss during power-up/down operation. Specifically, the improvements realized were:

1. The possibility of data loss was eliminated provided that the circuit was operated within voltage decay rates specifications.
2. Power-down time was shortened to reduce the energy storage requirements.

The power supply requirements (shown in Table 5) were relaxed with this implementation, which reduces the system requirements and the possibility of data loss.

Power-up

The power-up operation of the circuit shown in Figure 11 is unchanged from the power-up operation of the revision 0 circuit. The characteristics associated with the operation of the powerfail reset circuit below approximately 2.0 volts were not resolved with this circuit solution. If the voltage rise time specifications were not observed, the support circuits could have been enabled prematurely and would allow current transients to reach the drive coils or bubble function conductors (resulting in data loss).

Table 5. Power Supply Requirements for Powerfail Reset Circuit (Revision 1)

	V_{CC} (volts/msec)		V_{DD} (volts/msec)	
	Min.	**Max.**	**Min.**	**Max.**
Power-Up Voltage Rate of Rise	0.12	None	None	None
Power-Down/Power Failure Decay Rate	None	0.45	None	1.1

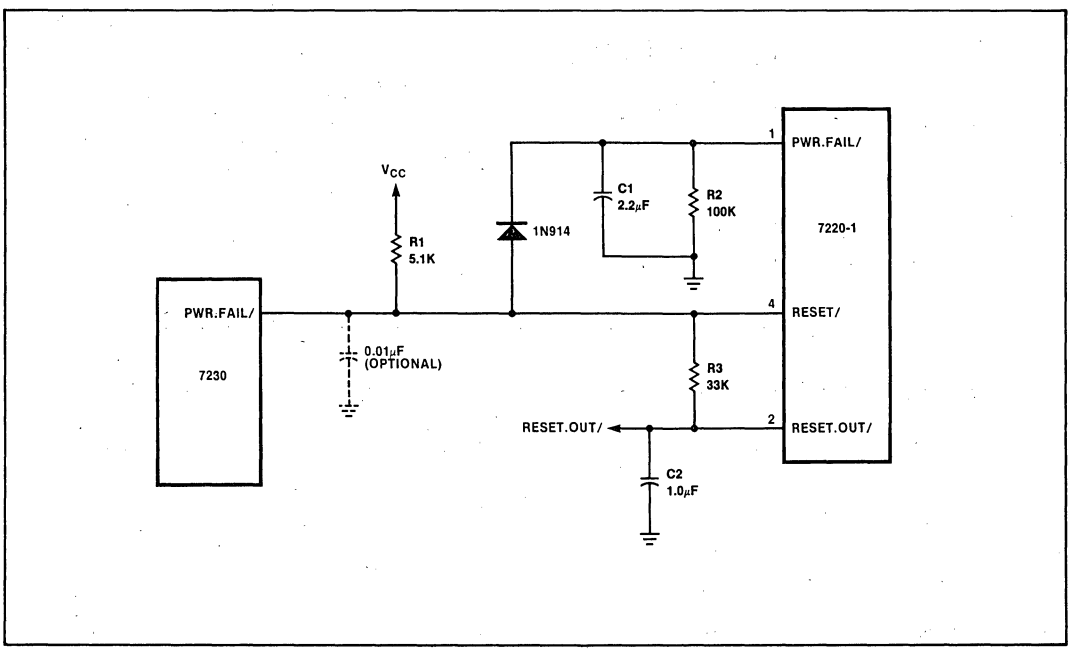


Figure 11. Revision 1 Circuit

Power-down

The simple modifications implemented in the external powerfail circuit (revision 1) greatly reduced the overall power-down operation timing (See Figure 12). This modification made use of the 7220-1 RESET/ input to initiate a power-down sequence instead of the 7220-1 PWR.FAIL/ input by effectively isolating the 7230 PWR.FAIL/ signal from delay capacitor C1 during power-down operations (eliminating an initial capacitor discharge delay). The 7220-1 BMC initiates an internal power-down sequence whenever its RESET/ input goes active, identical to the negative transition of the 7220-1 PWR.FAIL/ input. The difference between these two 7220-1 input signals is that the RESET/ input is latched and does not recognize a low-to-high transition and power-up therefore must be initiated by the positive transition of the 7220-1 PWR.FAIL/ input. With this circuit, the power-up operation timing was unaltered, and the power-down operation timing was reduced from approximately 500 microseconds in the revision 0 powerfail circuit to approximately 200 microseconds in the revision 1 powerfail circuit.

The primary reason for further refining this approach was the increased possibility for a "communication lockout" by the 7220-1. "Communication lockout" resulted when power was temporarily lost from the system. Specifically, the following two conditions were responsible for the "communication lockout":

1) The 7220-1 RESET/ input was activated low due to power loss (minimum pulse width must be 250 nanoseconds to ensure that it is latched) and initiated a power-down sequence.

2) The 7220-1 PWR.FAIL/ discharged but not below the inactive state (0.8 to 2.5 volts, typically 1.5 volts), before power was restored. A power-up sequence could not be initiated to reset the BMC to a known state and communication is "locked out."

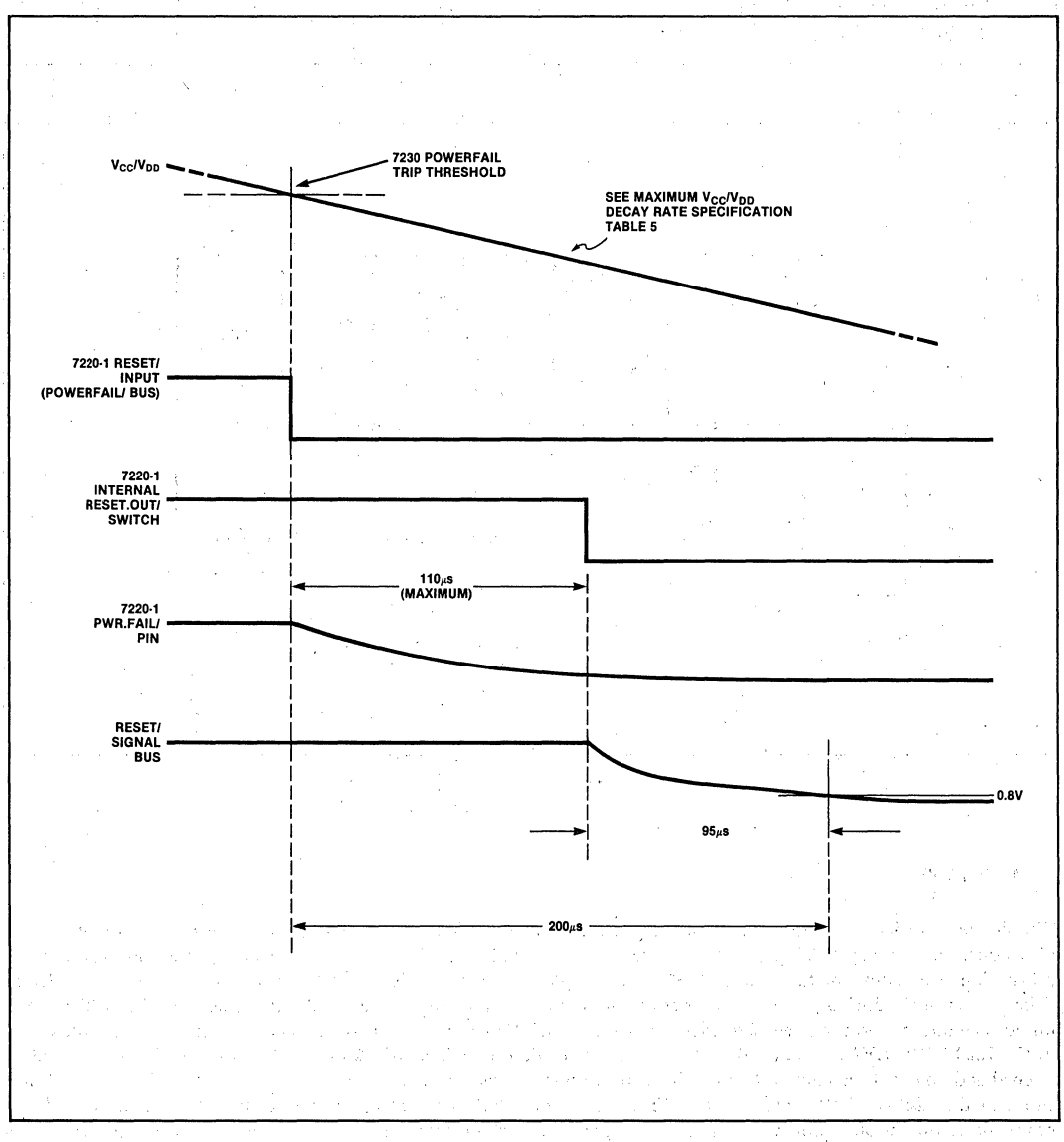


Figure 12. Power-down Timing (Revision 1)

Even if the circuit is operated within the voltage decay rate specifications, this inconvenience is still possible; the only solution is to pulse the 7220-1 PWR.FAIL/ input long enough to discharge C1 to a worst case value of 0.8 volt either by power cycling or external control. This user inconvenience and special system requirement led to the development of the next powerfail reset circuit.

Powerfail Reset Circuit — Revision 2

The powerfail reset circuit (revision 2) was developed to eliminate the possibility of data loss during power-up and power-down operation provided the power supply requirements are observed. The following paragraphs describe the principals of operation of the powerfail reset circuit. As power is applied or removed, several different signal value combinations are possible which complicate the analysis of this circuit. For the sake of simplicity, a general overview of a typical case is included rather than a detailed representation of each case. Throughout this discussion it is helpful for the reader to refer to the schematic diagram (Figure 3) and the timing diagrams (Figure 5 and 6).

Power-up

The overall circuit operation is complicated by the additional component, IC1. The power-up operation of the revision 2 circuit is very similar to previous circuits, however, the possibility of prematurely enabling the support components is eliminated. Diodes D1, D2 and resistor R5 serve to prevent capacitor C2 from charging beyond a level (0.8V) that could potentially deactivate the RESET/ signal bus to the 7242 FSA, the 7250 CPD and the VMOS transistor switch. Resistor R5 is chosen so that as V_{CC} is applied, diodes D1 and D2 will be forward biased and provide sufficient voltage drop to prevent capacitor C2 from charging above 0.8V.

Once the 7220-1 power-up sequence is complete or the first Abort command is received, the 7220-1 RESET.OUT/ is deactivated and capacitor C2 is allowed to fully charge. When the RESET/ signal bus reaches an inactive state the power-up sequence is complete and the system is prepared to accept an Initialize command (provided the Abort command has been issued).

Power-down

The power-down operation of the external powerfail reset circuit (revision 2) is very similar to revision 1. The fundamental difference is the ability to maintain a charge on capacitor C1 throughout the 7220-1 power-down sequence. This eliminates any glitch sensitivity or incorrect circuit operation during momentary power loss. The 7220-1 BMC initiates an internal power-down sequence whenever its RESET/ input goes active. The 7220-1 RESET.OUT/ signal is gated through IC1 and remains inactive during this time period preventing capacitor C1 from discharging. At the completion of the 7220-1 power-down sequence RESET.OUT/ signal is pulled low which causes both of the IC1 OR gate outputs to go low. The current sinking capability of these outputs act to quickly discharge capacitors C1 and C2 and complete the power-down sequence.

intel

APPLICATION NOTE

AP-150

October 1983

8085 To BPK 72 Interface

Ulmont Smith Jr.
Applications Engineer

OCTOBER 1983
ORDER NUMBER: 210849-002

8085 TO BPK 72 INTERFACE

INTRODUCTION

Bubble Memory is quickly emerging as the preferred high density storage medium for a variety of microprocessor applications. Considering their size and reliability, Bubble Memory allows the designer to utilize the advantages of microprocessors in environments that were not possible using other high density peripheral storage technologies. Aside from portable or rugged environmental applications, bubbles also open up new design possibilities for desk-top terminal applications. Some of the benefits that can be realized from the implementation of Bubble Storage are increased flexibility, reduced maintenance, and non-volatility.

In addition to a one megabit Bubble Memory, Intel magnetics also manufactures a complete family of integrated-support circuits that simplify the task of designing with Bubble Memory. The family of support circuits provides an easy-to-use microprocessor interface via a single VLSI component, the Bubble Memory Controller (BMC). The remaining support circuits are controlled by the Bubble Memory Controller allowing the designer total freedom from the control signals associated with Bubble Memory technology.

At the component level, the BPK 72 (Bubble Memory Prototype Kit) provides the best opportunity to discover the potential of bubble storage. The BPK 72 comes complete with all the hardware and documentation necessary to prototype a one megabit (128K-bytes) Bubble Memory System. The BPK 72 is completely assembled and tested leaving the designer with the simple task of interfacing to a host processor.

This application note demonstrates how little effort is required to interface a BPK 72 with an 8085 microprocessor. The first four sections, "Introduction, BPK 72 Overview, Constructing the Hardware Interface, Implementing the 8085/BPK 72 Software Driver," and Appendix A (software listing) provide all the information necessary to interface a BPK 72 with an 8085 microprocessor based system. The remaining chapters describe in detail the hardware and software considerations involved with designing and implementing a Bubble Memory Interface.

A set of generalized flowcharts describing the software driver may also be found in Appendix A to facilitate the task of interfacing with other microprocessors.

BPK 72 OVERVIEW

The BPK 72 consists of a completely assembled and tested 10cm x 10cm printed circuit board containing a one megabit Bubble Memory and the complete family of integrated support circuits.

A block diagram of the BPK 72 is presented in Figure 1. It illustrates the key components in a one megabit, 128K-byte Bubble Memory System.

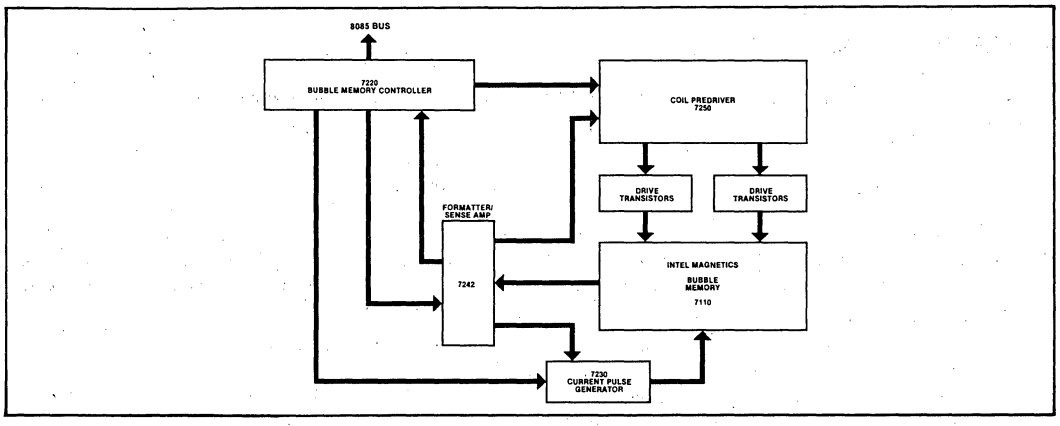


Figure 1. Block Diagram of the BPK 72

The 7110 Bubble Memory Module is supported by the following integrated circuits:

7220-1 Bubble Memory Controller (BMC)
The 7220-1 provides a convenient microprocessor interface and generates the timing signals necessary for the proper operation of the remaining support circuitry.

7242 Formatter Sense Amplifier (FSA)
The 7242 is responsible for detecting and enabling the generation of magnetic bubbles within the 7110. The 7242 also performs data formatting tasks and the option of automatic error detection and correction.

7250 Coil Predriver and 7254 Drive Transistors
The 7250 and two 7254s supply the drive currents for the rotating magnetic field that move the magnetic bubbles within the 7110 Bubble Memory Module.

7230 Current Pulse Generator (CPG)
The 7230 generates a set of waveforms necessary to input and output data from the 7110.

CONSTRUCTING THE HARDWARE INTERFACE

The hardware necessary to interface a BPK 72 with an 8085 microprocessor consists of a few simple connections to the system bus and the addition of only three integrated circuits; 7406—hex inverter (open collector), 7430—eight input nand gate, and an 8284A—Intel clock generator.

A schematic is presented in Figure 2 of the interface logic between a BPK 72 and the demultiplexed bus from an 8085 microprocessor.

The interface uses the eight input nand gate to enable chip-select on the BPK 72 when an I/O instruction is executed at ports 0FEH ("H" designates hexadecimal notation) or 0FFH. The address line A8 from the microprocessor bus is connected to A0 on the BPK 72 to select one of two internal ports. If the ports 0FEH and 0FFH are not available, simply connect A8 to the input of the nand gate and move a higher order address line (A9–A15) to A0 on the BPK 72. In the event that the I/O addresses are changed, the user must enter the new port locations into the software driver (see Appendix A). The I/O port locations are initialized as equates at the beginning of the program. All system dependent variables have been parameterized whenever possible.

The designer has the option of memory mapping the BPK 72 or utilizing 2 of the 256 I/O ports available on the 8085. The I/O ports were chosen for this interface to simplify the address decoding and to provide easy access to existing systems.

POWER SUPPLY REQUIREMENTS

The BPK 72 operates on standard +5V and +12V DC power within a 5% tolerance. The worst case power consumption is a follows:

+5VDC = 2 watts maximum
+12VDC = 5 watts maximum

When power is applied or removed from a Bubble Memory System, the rotating magnetic field within the 7110 Bubble Memory is held in the proper phase to insure non-volatility. This is accomplished through the use of a power fail reset circuit. The following power supply specifications must be observed to effectively support the power fail circuitry:

A. VDD = +12V, ±5% tolerance
   Power off/power fail voltage decay rate—less than 1.1 volts/millisecond
B. VCC = +5V, ±5% tolerance
   Power off/power fail voltage decay rate—less than 0.45 volts/millisecond
C. Voltage sequencing—no restrictions
D. Power on voltage rate of rise—no restrictions

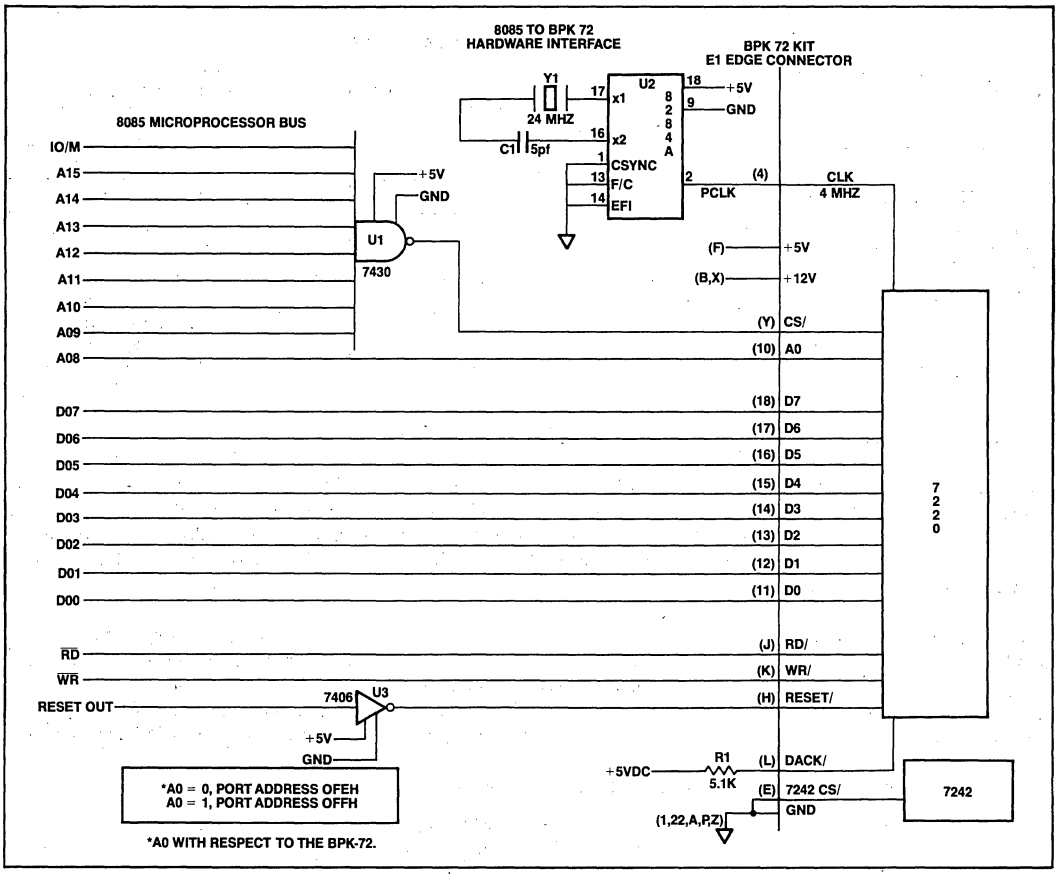


Figure 2. Hardware Interface

The interface designer should verify that the system power supply decay rates meets the specifications previously listed. To simulate worst case conditions, connect a 2 watt load on the +5 volt supply and a 5 watt load on the +12 volt supply. The power supply decay rates can be easily measured during the removal of power with a standard oscilloscope. No attempt should be made to use the BPK 72 until the power supply decay rates have been verified.

Table 1. 8085/BPK 72 Interface Parts List

Item	Description	Quantity	Reference	Manufacturer
1	IC-7430—8 input nand gate	1	U1	any
2	IC-8284A—clock generator	1	U2	Intel
3	IC-7406—hex inverter open collector	1	U3	any
4	Crystal—24.0000MHz fundamental mode, series resonant	1	Y1	any
5	Resistor—5.1Kohm, 1/4W, 5%	1	R1	any
6	Mica Capacitor—5pf, 100VDC, 5%	1	C1	any
7	Edge connector, 44 pin	1	E1	TRW, CINCH #50-44B-10

IMPLEMENTING THE 8085/BPK 72 SOFTWARE DRIVER

An 8085 to BPK 72 software driver program listing is presented in Appendix A. The driver consists of a set of subroutines that can be called to perform commonly used Bubble Memory commands. A detailed description and flowchart of each subroutine is provided with the program listing. The software driver is relocatable and may be linked with other programs. The name of the program is "BPK72." It begins at 0800H and requires less than 1K bytes of memory allocation.

The software driver is written in 8085 assembly language. It can be easily incorporated into existing systems as part of a utility program to transfer data between the BPK 72 and the 8085's addressable memory. The subroutines have been designed to eliminate the need for any further software development concerning the operation of the BPK 72. Assembly was chosen over higher level languages to provide the most efficient and portable code. With only minor modifications to the parameterized variables, the program, "BPK72," will run on almost any 8085 based system.

The following subroutines in the program "BPK72" will now be discussed:

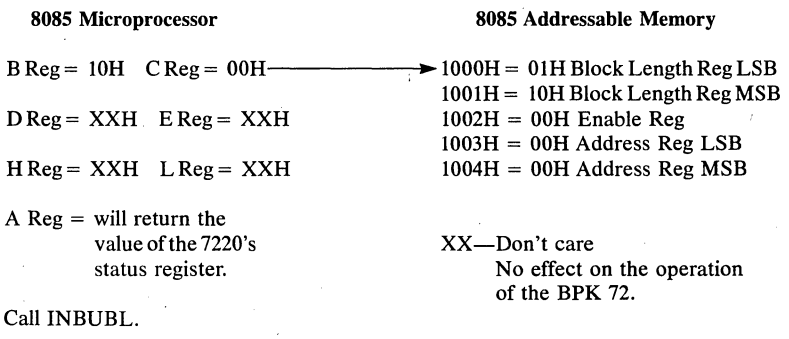
INBUBL—Initialize Bubble Memory
WRBUBL—Write Bubble Memory data
RDBUBL—Read Bubble Memory data
ABORT—Abort present command, reset BPK 72

INITIALIZING THE BUBBLE

After powering up, the BPK 72 must be initialized before any data transfers can begin. Initialization is needed to synchronize the 7220 Bubble Memory Controller with the data in the 7110 Bubble Memory storage loops and also because the 7110 employs redundancy. The 7110 Bubble Memory contains 320 storage loops. However, only 272 of the 320 loops are necessary for a 100% functional one megabit part. The additional 48 loops provide a 15% redundancy. Redundancy is used to significantly increase the yield of Bubble Memory modules during manufacture.

A map of the active and inactive loops is placed on a label attached to the case of the 7110. The same map is also placed in the 7110 during final test. When the system is initialized, the 7220 reads the map (boot loop) from the 7110 and decodes it. The boot loop is transferred from the 7220 into a pair of boot loop registers in the 7242 formatter sense amplifier. The boot loop registers are used to format data to insure that only functional loops are enabled during read or write operations.

Only one call to the initialization subroutine, INBUBL, is necessary to initialize a BPK 72. The following is an example of how to call INBUBL:



The example shown above demonstrates how to set up the B–C registers prior to calling the initialization subroutine, INBUBL. The B–C register pair must contain the address of the first of five consecutive locations within the 8085's addressable memory. In this example, the B–C registers are pointing to the first of five memory locations starting at 1000H. The data contained in 1000H through 1004H is a memory image of the parametric registers within the Bubble Memory Controller. The parametric registers contain a set of flags and parameters that determine exactly how the 7220 will respond to a software command.

Note the values used for the block length and address registers. These values must always be used during the initialization process with a one megabit Bubble Memory System. The enable register is shown with a 00H indicating the absence of error detection and correction. The 7220 and 7242 provide an optional error detection and correction feature to enhance data integrity. It is recommended that first time users begin without the use of error correction. Later on if error correction is desired, a 20H should be placed in the memory location designated as the enable register. A discussion concerning the use of error correction may be found in the section titled, "Communicating with the 7220."

Figure 3 illustrates the sequence of program flow necessary to initialize a Bubble Memory System using the subroutine INBUBL. Note that Figure 3 includes a test of the Bubble Memory Controller's status register. The status register is separate from the parametric registers and contains information about error conditions, completion or termination of commands, and the 7220's readiness to transfer data. To simplify the task of verifying a successful initialization, INBUBL returns the value of the 7220's status register to the calling routine through the 8085's "A" register. A successful initialization will return a 40H status. All other values indicate a BPK 72 system failure. Consult Appendix B in the unlikely event that the subroutine INBUBL fails to return a successful status.

READING AND WRITING

Only one call to the subroutine RDBUBL or WRBUBL is necessary to transfer data between the BPK 72 and the 8085's addressable memory.

Like many high density peripheral storage devices, Bubble Memory data is organized into pages rather than bytes. The 7220 Bubble Memory Controller partitions the one megabit Bubble Memory into 2048 pages of either 64 or 68 bytes in length. The page length is dependent upon the use of automatic error detection and correction—64 bytes with error correction and 68 bytes without. Data transfers are specified in terms of whole pages. Therefore the minimum amount of data that can be transferred from one read or write command is 64 or 68 bytes.

The parametric registers are used to communicate to the controller which page or pages will be transferred during a read or write command. The address register LSB and the first three bits of the address register MSB define the starting page address for read or write commands. The block length register determines how many pages will be transferred starting at the location defined by the address register. Theoretically, data transfers can range from 1 to 2048 pages in length. However, this application limits the maximum data transfer between the BPK 72 and the 8085's

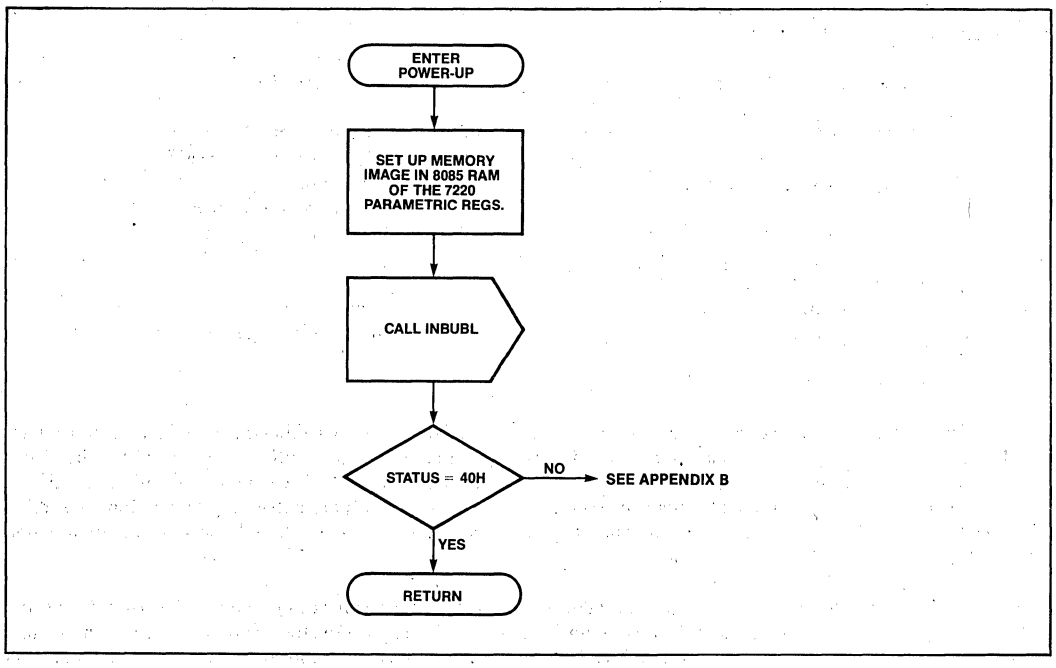


Figure 3. Initializing the BPK 72

memory to no more than 255 contiguous pages. This limitation results from the need to prevent data transfers that could exceed the addressable memory space of the 8085. The block length register LSB may be assigned any value between 1 and 255 depending on the size of the transfer. A detailed description of the parametric registers may be found in the section titled, "Communicating with the 7220."

The following is an example of how to use the Read Bubble Memory subroutine, RDBUBL, to transfer the first 16 pages (00H–0FH) of data from the BPK 72 to the 8085's addressable memory, starting at location 2000H:

8085 Microprocessor	**8085 Addressable Memory**
B Reg = 10H C Reg = 00H ——→	1000H = 10H Block Length Reg LSB
	1001H = 10H Block Length Reg MSB
D Reg = 20H E Reg = 00H ——→	1002H = 00H Enable Reg*
	1003H = 00H Address Reg LSB
H Reg = XXH L Reg = XXH	1004H = 00H Address Reg MSB
A Reg = will return the value of the 7220's status register.	2000H = start data transfer
	243FH = last data transfer (1088 byte transfer)
	XX—Don't care. No effect on the operation of the BPK 72.
Call RDBUBL.	*—Assumes that the BPK 72 was initialized without error correction.

The Write Bubble Memory subroutine, WRBUBL, can be substituted for the call to RDBUBL to transfer data from the 8085's addressable memory to the first 16 pages in the BPK 72.

The example shown above demonstrates how to set up the B–C and D–E registers prior to calling a read or write subroutine. Just as in the case of initialization, the B–C registers contain the address of the first of five consecutive memory locations within the 8085's addressable memory. The data contained in the memory addressed by the B–C registers is used to load the 7220's parametric registers. The D–E register pair contains the address of the first byte of data to be transferred to or from the 8085's addressable memory.

Figure 4 illustrates how the read and write subroutines, RDBUBL and WRBUBL, should be called from another routine. The flowchart includes a program path to handle errors in the unlikely event that the read or write subroutines fail to return a successful status. First time users can omit the additional program flow for preliminary evaluation. The next section, "Checking the Status," describes the appropriate status values necessary to verify a successful data transfer.

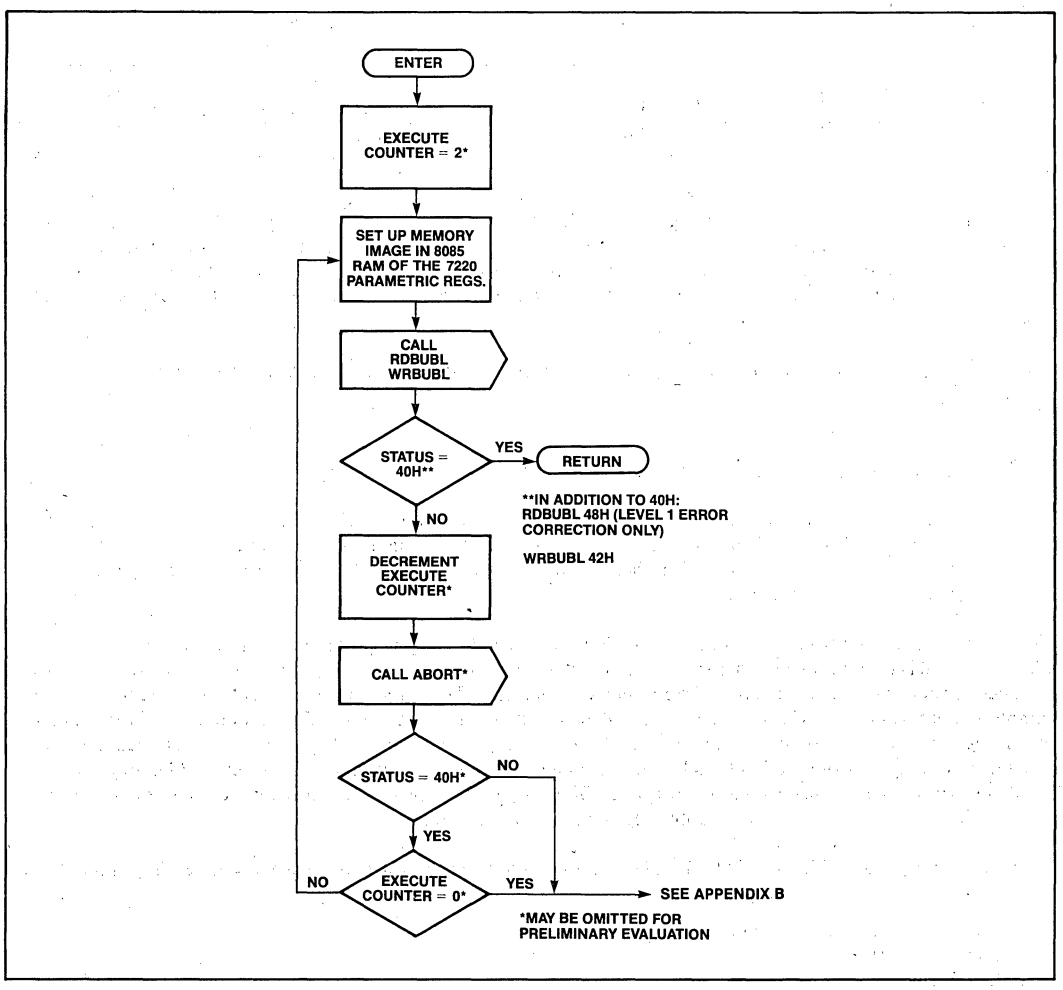


Figure 4. Reading and Writing to the BPK 72

CHECKING THE STATUS

After calling a subroutine to initialize, read, or write Bubble Memory data, the 7220's status register should be read to verify that the command was successfully executed. Note that flowcharts 1 and 2 include a test of the status register to detect for any errors. In order to facilitate the task of verification, each of the commonly used subroutines in the program "BPK72" return the contents of the 7220's status register to the calling routine through the 8085's "A" register. It is the responsibility of the calling routine to verify the success of each subroutine. A list of acceptable status register values for each of the subroutines in the program "BPK72" is presented in Table 2.

Table 2. Acceptable Status Register Values

Subroutine	Acceptable Status Register Value(s)	Comments
INBUBL	40H	OP-complete
WRBUBL	40H 42H	OP-complete OP-complete, parity error
RDBUBL	40H 48H	OP-complete OP-complete, correctable error*
ABORT	40H	OP-complete

*Level 1 error correction only

If any read errors are encountered during the transfer of data, they will almost always result from external noise interfering with the signal path between the 7110 Bubble Memory and the 7242 formatter sense amplifier. Since the data within the Bubble Memory is usually correct, a second attempt to transfer data should be successful. Figure 4 illustrates the use of the ABORT command to reset the Bubble Memory Controller before making another attempt to read or write Bubble Memory data.

Service information is presented in Appendix B in the unlikely event that any of the subroutines in Table 2 do not function properly.

7220 MICROPROCESSOR INTERFACE OVERVIEW

The key to any interface incorporating a BPK 72 is the Bubble Memory Controller. The controller provides a complete interface to a TTL level microprocessor bus that allows the designer total freedom from the intricate timing and waveforms necessary to support a Bubble Memory System. A block diagram of the 7220 Bubble Memory Controller is presented in Figure 5.

The 7220 interface circuitry consists of one 8-bit bidirectional port. The port provides access to internal registers. The address line A0 is used to select either the command/status or parametric/data registers. A command register is used to issue instructions such as read or write Bubble Memory data. The status register provides information about the completion or termination of commands and the 7220's readiness to transfer data. The parametric registers contain a set of flags and parameters that determine exactly how the 7220 will respond to a software command. The data register is actually a forty byte FIFO to buffer the timing differences between the 7110 Bubble Memory and a host processor. In order to transfer data to (from) the BPK 72, the host processor must load the parametric registers followed by issuing a read or write Bubble Memory data command.

To maintain design flexibility, the 7220 Bubble Memory Controller provides the user with three different modes of data transfer:

1. DMA, direct memory access
2. Interrupt-driven
3. Polled I/O

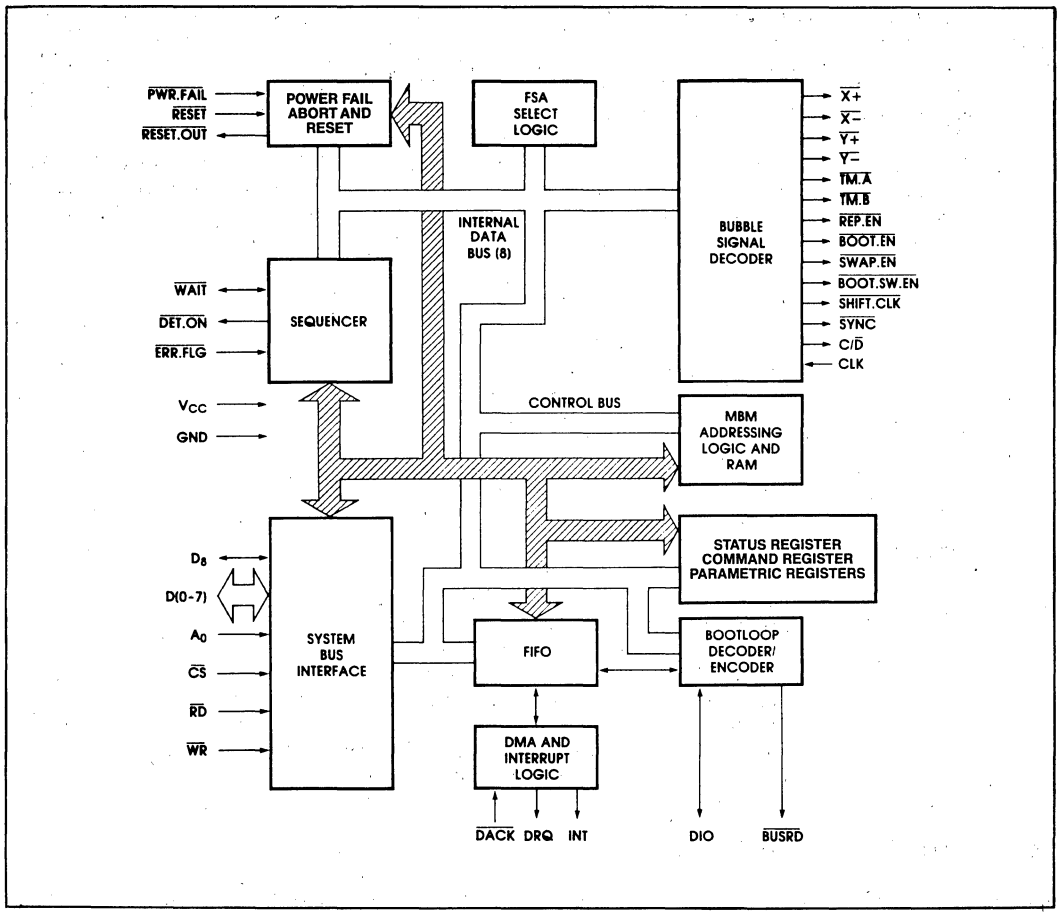


Figure 5. Block Diagram of the 7220 Bubble Memory Controller

In the DMA data transfer mode, the 7220 operates in conjunction with a DMA controller (such as Intel's 8257) using the DRQ (data request) and DACK (data acknowledge) lines for handshaking. With the help of a DMA controller, the 7220 transfers the data to (from) the host processor's memory. Once the data transfer begins, program intervention is not required until the entire data transfer has been completed.

In the interrupt mode, the 7220 along with an interrupt controller (such as Intel's 8259) uses the DRQ (data request) line to initiate a data transfer. The DRQ line becomes active when the 7220 is ready to send or receive a burst of data. A typical data burst is 22 contiguous bytes for an interrupt-driven interface. A set of software drivers are also necessary to service the interrupts to coordinate the transfer of data between the 7220 and the memory associated with a host processor. One advantage to the interrupt mode is multitasking. Since the host processor is only servicing the 7220 during data transfers, dead time between data transfers can be utilized for other processor tasks.

A polled mode interface reads the 7220 status register to determine when to transfer one byte of data. Of all the interface modes, polled I/O is the simplest configuration to implement. No special hardware or external controllers are necessary to interface the 7220 with a microprocessor. The major portion of a polled mode design is the software. Just as in the interrupt mode, a set of software drivers are required to read and write data to the 7220.

This application uses a polled mode configuration. The polled I/O data transfer mode was selected over DMA and interrupt-driven to simplify the interface design. A polled mode interface does not require the use of a DMA or interrupt controller. Furthermore, the polled mode interface provides the most flexibility for incorporating a BPK 72 into existing 8085 systems. Since the majority of a polled mode design consists of software, simple program modifications to accommodate existing systems can be easily entered into the software driver provided in Appendix A.

In terms of performance, the polled I/O transfer mode is the lowest compared to DMA or interrupt-driven. The DMA and interrupt modes offer the advantage of multitasking. However, the average access time and data transfer rate remain the same for each data transfer mode. The following formulas and examples demonstrate how to calculate the transfer time for a one megabit Bubble Memory System:

READ N-page transfer:
Transfer time = seek time + 8.7 ms + 7.5 ms (N−1)

WRITE N-page transfer:
Transfer time = seek time +7.5 ms (N)

Average seek time = 41 ms
Worst case seek time = 82 ms
Average data rate = 8.5 K-bytes/sec

For Example:

A. Time to read 1 page (assuming avg seek time):
Transfer time = 41 ms + 8.7 ms = 49.7 ms
B. Time to write 1 page (assuming avg seek time):
Transfer time = 41 ms + 7.5 ms = 48.5 ms
C. Time to read 10 contiguous pages (assuming avg seek time):
Transfer time = 41 ms + 8.7 ms + 7.5 ms (10–1) = 117.2 ms
D. Time to write 10 contiguous pages (assuming avg seek time):
Transfer time = 41 ms + 7.5 ms (10) = 116.0 ms

HARDWARE INTERFACE DESCRIPTION

To simplify the task of interfacing a BPK 72 with a microprocessor, the 7220 Bubble Memory Controller provides a convenient set of TTL signals that may be directly connected to a system bus. The interface signals on the BPK 72 necessary to implement a polled mode configuration are presented in Table 3.

PARITY BETWEEN THE 8085 AND BPK 72

The 7220 has the capability of generating and detecting odd parity using the bidirectional data line D8. The parity bit may be used to increase the reliability of the data path between the 7220 and a host processor. During data transfers, odd parity is generated for read operations and tested for write operations. The host processor may read the 7220 status register to determine if a parity error occurred during a write operation. Parity is typically implemented when a long transmission path exists between the host processor and the 7220. Since most systems utilize a simple edge connector backplane and a short transmission path (less than 18 inches), parity is not necessary. Parity is not implemented in this application to minimize the hardware complexity.

The parity bit, D8, is not stored within the 7110 Bubble Memory module. A separate and more effective error detection and correction feature is available as an option to increase the data integrity within the 7110. See the section titled, "Communicating with the 7220" for further details about the option of automatic error detection and correction.

Table 3. BPK 72 Polled Mode Interface Signals

Signal	Function
A0	Address line A0 = 0 Selects the FIFO data buffer or the parametric registers. A0 = 1 Selects command/status registers.
D0–D7	8 bit bidirectional data bus.
D8	Optional odd parity bit, not used in this application.
CS/	Chip select input. A logic high will tri-state the 7220 interface signals. (Slash, "/" designates a low active signal, system ground)
RD/	Read 7220 registers or data FIFO.
WR/	Write 7220 registers or data FIFO.
DACK/	DMA acknowledge. If DMA is not used, DACK/ requires an external pullup resistor to VCC (5.1 Kohm).
CLK	4 MHz TTL level clock. Clock period = 250 ns, 0.25 ns tolerance. Duty cycle = 50%, 5% tolerance.
RESET/	A low on this pin forces the interruption of any 7220 activity, performs a controlled shut-down, and initiates a reset sequence. The next instruction following RESET/ must be an abort command.
7242 CS/	7242 chip select signal is used to select banks of 7242s. 7242 CS/ must be tied low (system ground) for a single bank configuration.

4 MHZ CLOCK

The BPK 72 requires an external 4 MHz (may be asynchronous with respect to a host processor) TTL level clock. The specifications for the period and duty cycle are presented in Table 3. The 7220 uses the external clock to generate the timing signals that control the rotating magnetic field within the 7110 Bubble Memory. For reliable operation, the clock tolerances must be observed to assure that the rotating field is stable and accurate.

An Intel integrated circuit, 8284A clock driver, is used to generate the 4 MHz external clock. The 8284A along with a 24MHz series resonant crystal (fundamental mode) will provide a precise and accurate clock for any interface incorporating a BPK 72. The circuit configuration for the 8284A is illustrated in Figure 2. Other techniques of clock generation are acceptable as long as the duty cycle and period are within the specifications listed in Table 3.

SOFTWARE INTERFACE DESCRIPTION

The software driver presented in Appendix A contains the following subroutines that may be called from another routine:

* INBUBL —Initialize the BPK 72.
* RDBUBL —Read Bubble Memory data.
* WRBUBL —Write Bubble Memory data.
ABORT —Abort present command, reset BPK 72.
FIFORS —Reset 7220 FIFO data buffer.
WRFIFO —Write 7220 FIFO data buffer.
RDFIFO —Read 7220 FIFO data buffer.
WRBLRS —Write 7242 boot loop registers.
RDBLRS —Read 7242 boot loop registers.
MBMPRG —Bubble Memory purge command.
** RDBOOT —Read Bubble Memory boot loop.
** BOOTUP —Write Bubble Memory boot loop.

* Most commonly used commands.
** Diagnostic routines (see Appendix B).

Each of the subroutines listed above is described in further detail in Appendix A. Along with each subroutine is a generalized flowchart displaying the program flow. The user is encouraged to read the software driver to better understand the software interaction necessary to interface a BPK 72 with an 8085 microprocessor.

COMMUNICATING WITH THE 7220

Some additional background is necessary to understand the operation of the 7220 Bubble Memory Controller. Figure 6 illustrates the user-accessible registers that control and format the flow of data between the 7110 Bubble Memory and a host processor.

The address assignments for the user-accessible registers within the 7220 are presented in Table 4. The registers are listed in two groups. The first group (status, command, register address counter) consists of those registers that are selected and accessed in one operation. The second group contains the FIFO data buffer and the parametric registers (utility, block length, enable, address), they are selected according to the contents of the register address counter (RAC).

Table 4. 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	0	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

NOTES:
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

Table 5. Parametric Registers and FIFO Data Buffer

	RAC						
A0	B3	B2	B1	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

To successfully implement the hardware and software presented in this application, certain restrictions are placed on the contents of the user-accessible registers. Each of the user-accessible registers and any necessary restrictions will now be discussed in further detail.

COMMAND REGISTER

The 7220 command set consists of 16 commands identified by a 4 bit command code. A list of the commands is presented in Table 6.

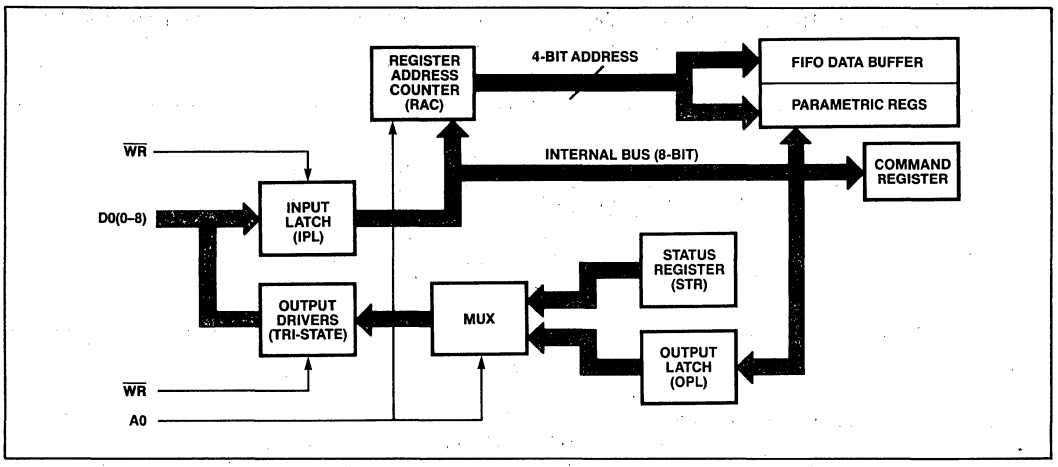


Figure 6. 7220 User Accessible Registers

Table 6. 7220 Commands

D3	D2	D2	D1	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 commands listed in Table 6 are provided for reference purposes only. The software driver in Appendix A consists of a series of subroutines that automatically issue the appropriate commands to perform a data transfer.

The function of each command is usually apparent from the command name (e.g., initialize, read bubble data, write bubble data). Additional detail concerning the function of each command may be found in the BPK 72 user's manual.

REGISTER ADDRESS COUNTER

The register address counter consists of a 4 bit address that points to one of the six parametric registers:

Utility register (UT)—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 operation of the 7220. It is provided as a convenience to the user.

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

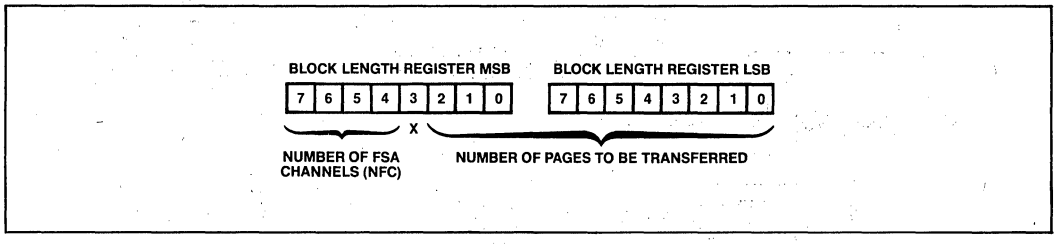


Figure 7. Block Length Registers

The 7220 has the capability of supporting up to eight 7110 Bubble Memory modules. Each 7110 contains two channels that are sensed by a 7242 formatter sense amplifier (FSA). In multiple Bubble Memory configurations, the BLR allows the user to select the page size. Since the BPK 72 consists of only one Bubble Memory module, the field specifying the number of FSA channels in the BLR MSB must contain −0001B ("B" designates a binary notation). After the FSA field is set, the page size is dependent upon the use of error detection and correction. Error correction will be discussed in the next section describing the function of the enable register.

The BLR LSB and the first 3 bits of the BLR MSB determine the number of pages to be transferred during a single read or write command. This application restricts the user to no more than 255 contiguous pages to prevent data transfers that could exceed the addressable memory space of the 8085.

For This Application

BLR MSB—10H at all times.
("H" designates a hexadecimal notation)

BLR LSB—Selectable from 01H to FFH (1 to 255 pages).

CAUTION: 00H in the BLR LSB will enable a 2048 page transfer resulting in a timing error.

Enable Register (ER)—The user sets the bits in the enable register to enable or disable various functions within the 7220. The individual bit descriptions are as follows:

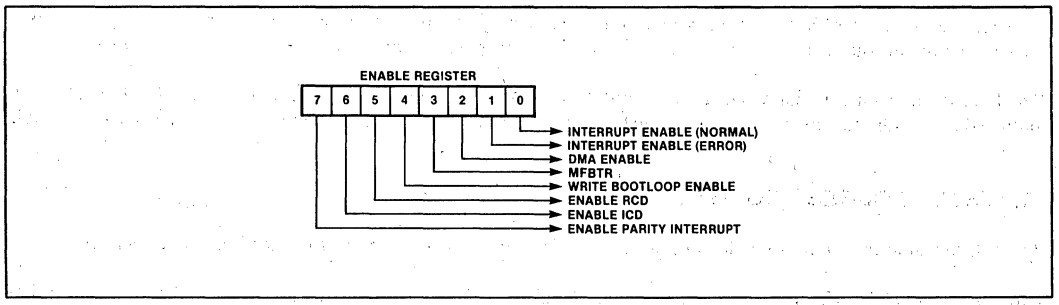


Figure 8. Enable Register

One of the most important functions concerning the enable register is the option of automatic error detection and correction. If error correction is enabled during a write operation, the 7242 formatter sense amplifier appends each 256 bit block of data with a 14 bit fire code. Both the data and the fire code are stored within the 7110 Bubble Memory module. During a read operation, the 7242 compares the data with the fire code to check for any errors. With 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 7220) or uncorrectable. Additional information about the fire code is available in the BPK 72 user's manual.

The enable register offers three levels of error correction. All three levels utilize the same error correction algorithm but differ in their interaction with a host processor. Table 6 defines the relevant register bits for the various levels of error correction.

Table 6. Error Correction Levels

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

Level 0 does not enable the error detection and correction algorithm. In this mode, the 7220 partitions one megabit systems into 2048 pages consisting of 68 bytes per page.

Level 1 is the most popular level of error correction. If an error is detected during a read operation, the 7242 automatically cycles the data through its error correction algorithm and transfers the data to the 7220. If the error was correctable, the 7220 will continue to function normally i.e., correctable errors in Level 1 are transparent to the host processor. If the error was uncorrectable, the 7220 will stop reading at the end of the page wherein the error was encountered. In the unlikely event that the 7220 stops because of an uncorrectable error, the host processor should try at least one more attempt to read the data. In most cases, errors result from random noise that can interfere with the signal path between the 7110 and 7242. Since the data is usually correct within the 7110, another attempt to read the data should yield a successful status.

Level 2 and Level 3 differ from Level 1 in that page-specific logging of uncorrectable errors is possible and the transfer of erroneous data can be prevented. Level 3 differs from Level 2 in that Level 3 also allows the logging of correctable errors.

Neither Level 2 nor Level 3 is supported by this application because the probability of an uncorrectable error is typically one in 10^{16} bits read. An error rate of this magnitude will produce few if any uncorrectable errors throughout the useful life of a Bubble Memory System.

It is recommended that Level 1 error correction be utilized to improve the integrity of the data within the 7110. In Level 1, the 7220 assigns 64 bytes to a page in one megabit Bubble Memory Systems.

Aside from error correction, the enable register performs many other functions.

Enable Parity Interrupt—If this bit is set, any parity errors between the host and the 7220 during write operations will generate an interrupt. Since parity and the interrupt mode are not used in this application, the enable parity interrupt bit should be reset to a logical zero.

Write Bootloop Enable—This bit must be reset to prevent accidental erasure of the boot loop within the 7110.

MFBTR—The MFBTR bit should always be reset to maximize the data transfer rate between the 7220 and 7242 during read operations.

DMA Enable—If this bit is set, the 7220 will attempt to transfer data in the DMA mode. Since this application utilizes a polled mode interface, this bit must be reset to a logical zero.

Interrupt Enable (Normal)—If this bit is set, an interrupt is sent to the host processor after the successful completion of a Bubble Memory command. Since this application uses a polled mode interface, this bit should be reset to a logical zero.

For This Application

Enable Reg—00H. No error correction.
—20H. Level 1 error correction.

Address Register (AR)—The contents of the address register determine which starting address locations will be used during a read or write command. For systems with a multiple Bubble Memory configuration, an additional magnetic Bubble Memory (MBM) select field is used to specify which Bubble Memory(s) will be selected. The bit configuration is as follows:

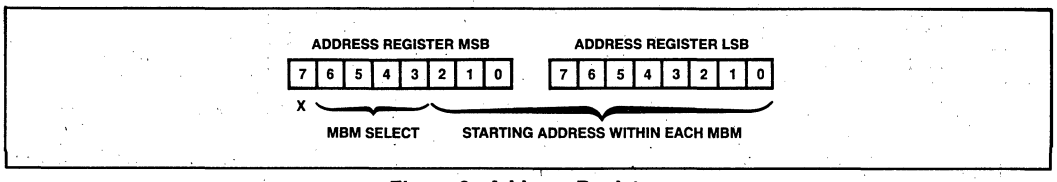


Figure 9. Address Registers

Since the BPK 72 consists of only one 7110 Bubble Memory module, the MBM select field must contain −0000B ("B" designates a binary notation).

For This Application

AR MSB—00000XXX

AR LSB—XXXXXXXX, X = user selectable page address
from 0 to 2047.

STATUS REGISTER

In a polled data transfer mode, the status register provides information about error conditions, completion or termination of commands, and the 7220's readiness to transfer data or accept new commands. The bit configuration for the status register is as follows:

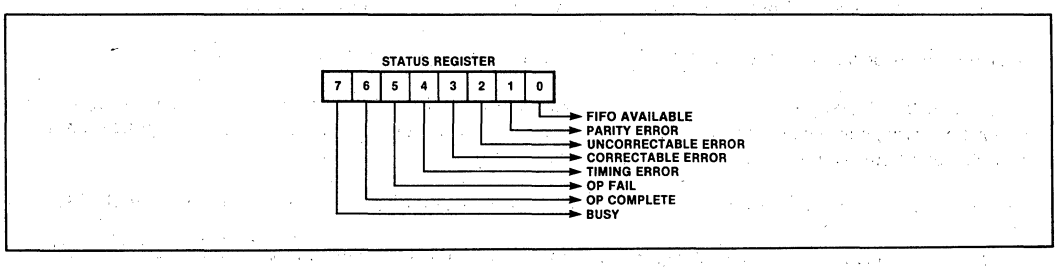


Figure 10. Status Register

Busy—When active (Logic 1), the Busy bit indicates that the 7220 is in the process of executing a command. Bits 1 through 6 of the status register are valid only when the busy bit is not active (Logic 0).

OP Complete—When active (Logic 1), the OP Complete bit indicates the successful completion of a command.

OP Fail—When active (Logic 1), the OP Fail bit indicates that the 7220 was unable to successfully complete the current command.

Timing Error—When active (Logic 1), the Timing Error bit indicates that an FSA has reported a timing error to the 7220, or that the host system has failed to keep up with the required data rate during a read or write operation.

Correctable Error—When active (Logic 1), the Correctable Error bit indicates that an FSA has detected a correctable error in the last block of data read from the 7110.

Uncorrectable Error—When active (Logic 1), the Uncorrectable Error bit indicates that an FSA has detected an uncorrectable error in the last block of data read from the 7110.

Parity Error—When active (Logic 1), the Parity Error bit indicates that a parity error was detected between the 7220 and the host processor. Parity errors are only detected by the 7220 during write operations. Since parity is not used in this application, ignore all parity errors.

FIFO Ready—When the 7220 is busy, an active FIFO Ready bit (Logic 1) indicates that the FIFO has data for reading or space for writing. When the 7220 is not busy, the FIFO Ready bit (Logic 0) indicates that the 40 byte FIFO and the input and output latches are completely empty.

SUMMARY

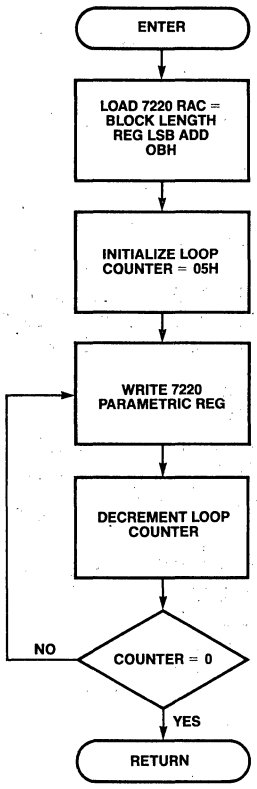
This application note is intended to eliminate almost all of the development effort necessary to interface an 8085 microprocessor with a BPK 72. With the addition of only a few IC's and the software driver presented in Appendix A, the designer is well on the way to incorporating the benefits of improved reliability, reduced maintenance, and non-volatility into any 8085 microprocessor based system.

APPENDIX A
8085 TO BPK-72 INTERFACE SOFTWARE DRIVER LISTING AND FLOWCHARTS

```
ASM80 :F1:BPKHDR


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72    PAGE    1


  LOC  OBJ        LINE       SOURCE STATEMENT

                    1 ;*******************************************************************************
                    2 ;
                    3 ;
                    4 ;                       PROGRAM: 8085 TO BPK72 SOFTWARE DRIVER V1.0
                    5 ;                       ULMONT S. SMITH JR.
                    6 ;                       INTEL CORPORATION
                    7 ;                       3065 BOWERS AVENUE
                    8 ;                       SANTA CLARA, CALIFORNIA 95051
                    9 ;
                   10 ;
                   11 ;*******************************************************************************
                   12 ;
                   13 ;
                   14 ;
                   15 ; ABSTRACT:
                   16 ;
                   17 ;       THIS PROGRAM  CONSISTS OF A SET OF  BUBBLE MEMORY SOFTWARE DRIVERS
                   18 ;       THAT SUPPORT A POLLED MODE INTERFACE BETWEEN A BPK72, 1MBIT BUBBLE
                   19 ;       MEMORY PROTOTYPE KIT,  AND  A STANDARD  8085  MICROPROCESSOR.  THE
                   20 ;       PROGRAM  UTILIZES A SET  OF PUBLIC  DIRECTIVES THAT  CAN BE CALLED
                   21 ;       TO PERFORM A BUBBLE MEMORY INITIALIZATION, READ, WRITE,  AND OTHER
                   22 ;       COMMONLY USED COMMANDS.  IN THE UNLIKELY EVENT THAT THE 7110 BUBBLE
                   23 ;       MEMORY BOOT LOOP IS LOST, TWO ROUTINES ARE PROVIDED TO EXAMINE AND
                   24 ;       REWRITE THE BOOT LOOP CODE.
                   25 ;
                   26 ;
                   27 ;
                   28 ; PROGRAM ORGANIZATION:
                   29 ;
                   30 ;       FUNCTIONS:
                   31 ;                         INTPAR
                   32 ;                         FIFORS
                   33 ;                         BYTCNT
                   34 ;                         WRITE
                   35 ;                         READ
                   36 ;                         ABORT
                   37 ;                         WRBUBL
                   38 ;                         RDBUBL
                   39 ;                         INBUBL
                   40 ;                         BOOTUP
                   41 ;                         RDBOOT
                   42 ;                         WRFIFO
                   43 ;                         RDFIFO
                   44 ;                         WRBLRS
                   45 ;                         RDBLRS
                   46 ;                         MBMPRG
                   47 ;
                   48 ;
                   49 ; EXTERNAL DECLARATIONS: NONE
                   50 ;
                   51 ;
                   52 $EJECT
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72   PAGE   2


 LOC  OBJ        LINE       SOURCE STATEMENT

                   53 ;
                   54 ; PUBLIC SYMBOLS:
                   55 ;
                   56 ;              FIFORS - RESET 7220 FIFO DATA BUFFER
                   57 ;              ABORT  - ABORT PRESENT COMMAND, RESET BPK72
                   58 ;              WRBUBL - WRITE BUBBLE MEMORY DATA
                   59 ;              RDBUBL - READ BUBBLE MEMORY DATA
                   60 ;              INBUBL - INITIALIZE THE BPK72
                   61 ;              BOOTUP - WRITE BUBBLE MEMORY BOOT LOOP
                   62 ;              RDBOOT - READ BUBBLE MEMORY BOOT LOOP
                   63 ;              WRFIFO - WRITE 7220 FIFO DATA BUFFER
                   64 ;              RDFIFO - READ 7220 FIFO DATA BUFFER
                   65 ;              WRBLRS - WRITE 7242 BOOT LOOP REGISTERS
                   66 ;              RDBLRS - READ 7242 BOOT LOOP REGISTERS
                   67 ;              MBMPRG - BUBBLE MEMORY PURGE COMMAND
                   68 ;
                   69 ;
                   70 ;*******************************************************************************
                   71 ;
                   72                        NAME BPK72
                   73 ;
                   74 ;*******************************************************************************
                   75 ;
                   76 ;
                   77 ;*******************************************************************************
                   78 ;
 0800              79                        ORG     0800H
                   80 ;
                   81 ;*******************************************************************************
                   82 ;
                   83 ;
                   84 ;*******************************************************************************
                   85 ;
                   86 ;                      PROGRAM EQUATES
                   87 ;
                   88 ;*******************************************************************************
                   89 ;
                   90 ;
 00FE              91 PRTA00  EQU     0FEH    ; A POLLED  MODE  INTERFACE  REQUIRES  ONLY  TWO  I/O
 00FF              92 PRTA01  EQU     0FFH    ; PORTS DESIGNATED BY THE A0 LINE ON THE BPK72 BOARD.
                   93                         ; THIS APPLICATION USES:
                   94                         ;
                   95                         ;       0FEH - A0=0 FOR PRTA00 (PORT A0= 0)
                   96                         ;              RD/WR BUBBLE MEMORY DATA AND REGS
                   97                         ;
                   98                         ;       0FFH - A0=1 FOR PRTA01 (PORT A0= 1)
                   99                         ;              RD STATUS REG
                  100                         ;              WR BUBBLE MEMORY COMMANDS
                  101 ;
                  102 $EJECT
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72    PAGE    3


 LOC  OBJ        LINE       SOURCE STATEMENT

                  103 ;*******************************************************************************
                  104 ;
                  105 ; FUNCTION: INTPAR
                  106 ; INPUTS:   B-C REGS, STARTING ADDRESS OF PARAMETRIC REGS IN RAM
                  107 ; OUTPUTS:  7220 PARAMETRIC REGS
                  108 ; CALLS:    NONE
                  109 ; DESTROYS: A, F/FS
                  110 ;
                  111 ; DESCRIPTION: LOAD THE 7220 PARAMETRIC REGS
                  112 ;       THE B-C REGS  CONTAIN THE ADDRESS TO THE FIRST OF FIVE CONTIGUOUS
                  113 ;       MEMORY LOCATIONS IN RAM.  THE DATA  ADDRESSED BY THE  B-C REGS IS
                  114 ;       USED TO LOAD  THE PARAMETRIC  REGISTERS IN THE 7220 BUBBLE MEMORY
                  115 ;       CONTROLLER.  INTPAR COPIES THE DATA IN RAM TO THE PARAMETRIC REGS.
                  116 ;
 0800 C5          117 INTPAR: PUSH    B       ; SAVE B-C REGS
 0801 D5          118         PUSH    D       ; SAVE D-E REGS
 0802 3E0B        119         MVI     A,0BH   ; LOAD A REG WITH BLR LSB ADDRESS
 0804 D3FF        120         OUT     PRTA01  ; LOAD 7220 RAC WITH BLR LSB ADDRESS
 0806 1E05        121         MVI     E,05H   ; INITIALIZE LOOP COUNTER
 0808 0A          122 LOAD:   LDAX    B       ; LOAD A REG FROM B-C REG ADDRESS
 0809 D3FE        123         OUT     PRTA00  ; WRITE PARAMETRIC REG
 080B 03          124         INX     B       ; INCREMENT B-C REGS TO THE NEXT ADDRESS IN RAM
 080C 1D          125         DCR     E       ; DECREMENT LOOP COUNTER
 080D C20808      126         JNZ     LOAD    ; IF NOT ZERO, JMP LOAD
 0810 D1          127         POP     D       ; RESTORE D-E REGS
 0811 C1          128         POP     B       ; RESTORE B-C REGS
 0812 C9          129         RET             ; RETURN TO CALL
                  130 ;
                  131 ;
                  132 ;
                  133 $EJECT
```



COMMENTS: THE UTILITY REGISTER IS NOT USED. THE RAC IS AUTOMATICALLY INCREMENTED AFTER EACH WRITE (WR/) IS EXECUTED. THE RAC WILL NOT INCREMENT BEYOND 00H.

Figure 11. INTPAR

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72    PAGE    4


 LOC  OBJ        LINE       SOURCE STATEMENT

                  134 ;*******************************************************************************
                  135 ;
                  136 ; FUNCTION: FIFORS
                  137 ; INPUTS:   BPK72 STATUS REG
                  138 ; OUTPUTS:  ISSUE FIFO RESET COMMAND TO BPK72
                  139 ;           A REG= BPK72 STATUS REG
                  140 ; CALLS:    NONE
                  141 ; DESTROYS: A, F/FS
                  142 ;
                  143 ; DESCRIPTION: RESET 7220 FIFO DATA BUFFER
                  144 ;       A FIFO RESET COMMAND IS ISSUED TO THE BPK72.  AFTER ISSUING THE
                  145 ;       COMMAND,  THE BPK72 STATUS REG IS POLLED UNTIL AN OP-COMPLETE,
                  146 ;       40H,  HAS BEEN READ OR THE TIME OUT LOOP COUNTER DECREMENTS TO
                  147 ;       ZERO.  FIFORS RETURNS THE VALUE OF THE BPK72 STATUS REG TO  THE
                  148 ;       CALLING ROUTINE  VIA THE 8085'S A REG.  ONLY  A  STATUS OF 40H
                  149 ;       INDICATES A SUCCESSFUL EXECUTION OF THE FUNCTION FIFORS.
                  150 ;
                  151         PUBLIC  FIFORS  ; DECLARE PUBLIC FUNCTION
 0813 D5          152 FIFORS: PUSH    D       ; SAVE D-E REGS
 0814 C5          153         PUSH    B       ; SAVE B-C REGS
 0815 0640        154         MVI     B,40H   ; LOAD B REG= 40H, OP-COMPLETE
 0817 11FFFF      155         LXI     D,0FFFFH; INTIALIZE TIME OUT LOOP COUNTER
 081A 3E1D        156         MVI     A,1DH   ; LOAD A REG= FIFO RESET COMMAND
 081C D3FF        157         OUT     PRTA01  ; WRITE FIFO RESET COMMAND
 081E DBFF        158 BUSYFR: IN      PRTA01  ; READ STATUS REG
 0820 07          159         RLC             ; TEST BUSY BIT= 1
 0821 DA2E08      160         JC      POLLFR  ; IF BUSY= 1, POLL STATUS REG FOR 40H
 0824 1B          161         DCX     D       ; DECREMENT TIME OUT LOOP COUNTER
 0825 AF          162         XRA     A       ; CLEAR A REG
 0826 B2          163         ORA     D       ; TEST D REG= 00H
 0827 B3          164         ORA     E       ; TEST E REG= 00H
 0828 C21E08      165         JNZ     BUSYFR  ; IF NOT ZERO, CONTINUE POLLING FIFO RESET COMMAND
 082B C33B08      166         JMP     RETFR   ; TIME OUT ERROR, RETURN
 082E DBFF        167 POLLFR: IN      PRTA01  ; READ STATUS REG
 0830 A8          168         XRA     B       ; TEST STATUS= 40H, OP-COMPLETE
 0831 CA3B08      169         JZ      RETFR   ; IF OP-COMPLETE, JMP RETFR
 0834 1B          170         DCX     D       ; DECREMENT TIME OUT LOOP COUNTER
 0835 AF          171         XRA     A       ; CLEAR A REG
 0836 B2          172         ORA     D       ; TEST D REG= 00H
 0837 B3          173         ORA     E       ; TEST E REG= 00H
 0838 C22E08      174         JNZ     POLLFR  ; IF NOT ZERO, CONTINUE POLLING FIFO RESET COMMAND
 083B C1          175 RETFR:  POP     B       ; RESTORE B-C REGS
 083C D1          176         POP     D       ; RESTORE D-E REGS
 083D DBFF        177         IN      PRTA01  ; READ STATUS REG
 083F C9          178         RET             ; RETURN TO CALL
                  179 ;
                  180 ;
                  181 ;
                  182 $EJECT
```

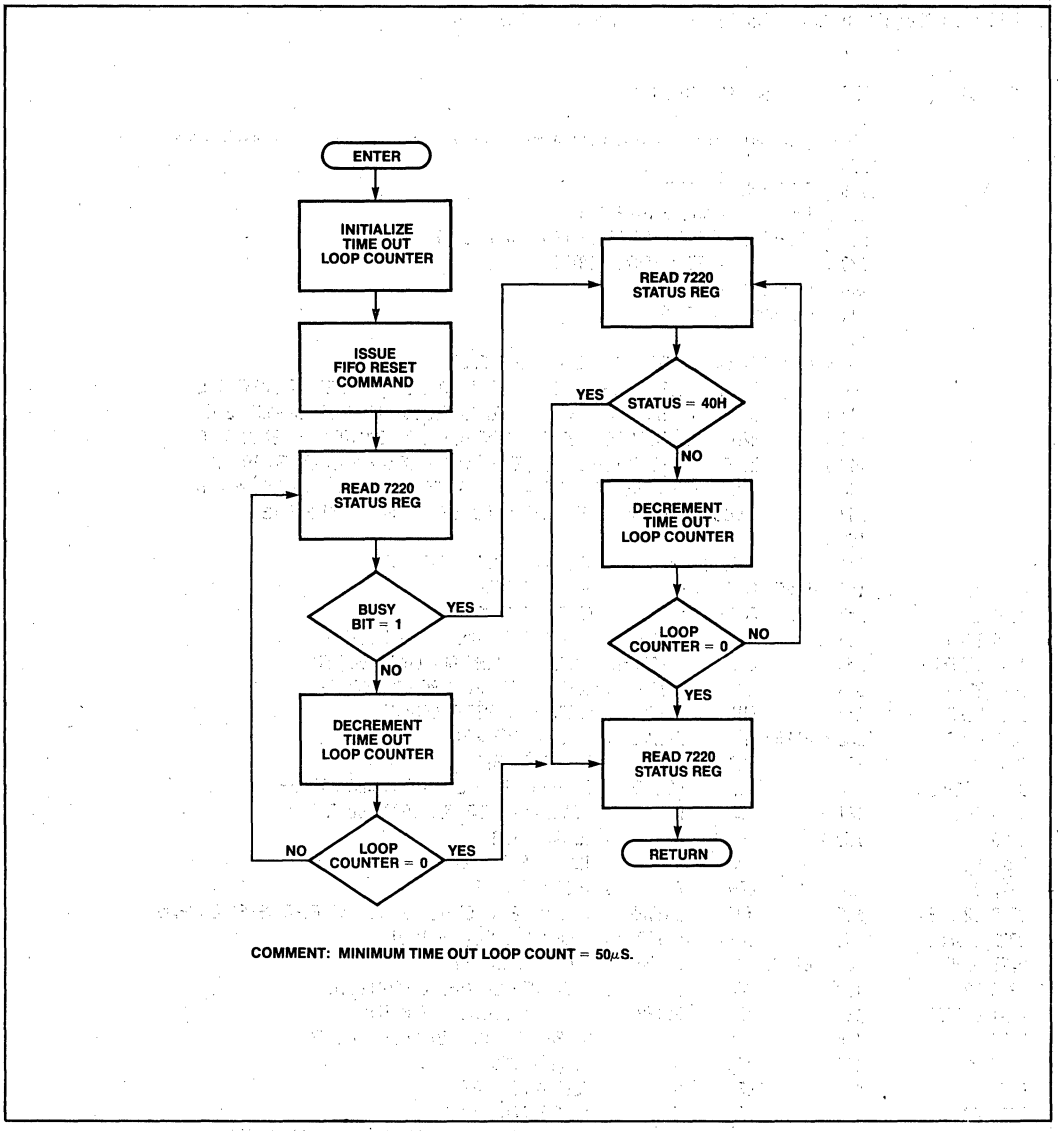
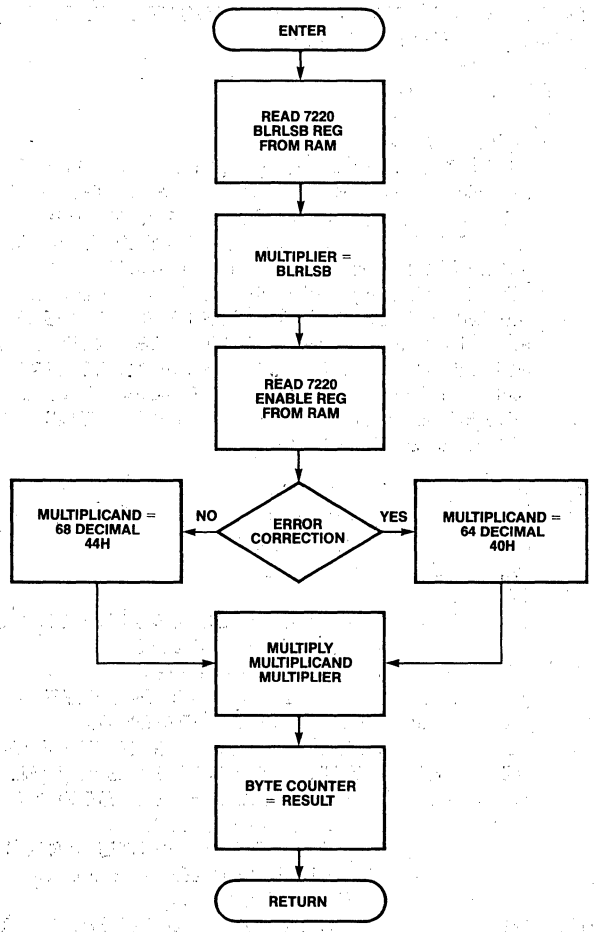


Figure 12. FIFORS

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72   PAGE   5


 LOC  OBJ        LINE      SOURCE STATEMENT

                  183 ;*****************************************************************************
                  184 ; FUNCTION: BYTCNT
                  185 ; INPUTS:   B-C REGS, STARTING ADDRESS OF PARAMETRIC REGS IN RAM
                  186 ; OUTPUTS:  H-L REGS= BYTE COUNTER
                  187 ; CALLS:    NONE
                  188 ; DESTROYS: A, H, L, F/FS
                  189 ;
                  190 ; DESCRIPTION: BYTE COUNTER
                  191 ;       THE B-C REGS CONTAIN THE ADDRESS TO THE FIRST OF FIVE CONTIGUOUS MEMORY
                  192 ;       LOCATIONS IN RAM.  THE DATA  ADDRESSED BY  THE B-C REGS IS USED TO LOAD
                  193 ;       THE PARAMETRIC REGS IN  THE 7220  BUBBLE MEMORY CONTROLLER.  THE ENABLE
                  194 ;       REG IS READ FROM RAM TO DETERMINE IF ERROR CORRECTION HAS BEEN ENABLED.
                  195 ;       THE USE OF ERROR CORRECTION REQUIRES A 64 BYTE TRANSFER/PAGE  - 68 BYTE
                  196 ;       TRANSFER/PAGE WITHOUT ERROR CORRECTION.  THE  BLOCK  LENGTH  REG LSB IS
                  197 ;       ALSO READ FROM RAM TO  DETERMINE THE NUMBER  OF PAGES TO BE TRANSFERRED
                  198 ;       DURING THE NEXT READ OR WRITE COMMAND.  THE  NUMBER  OF  BYTES PER PAGE
                  199 ;       MULTIPLIED BY THE NUMBER OF PAGES IS COMPUTED AND PASSED TO THE CALLING
                  200 ;       ROUTINE VIA THE 8085'S H-L REGS.  DATA TRANSFERS ARE LIMITED TO  16,320
                  201 ;       BYTES WITH ERROR CORRECTION AND  17,340 BYTES WITHOUT.  ONLY THE BLRLSB
                  202 ;       IS USED TO GENERATE THE BYTE COUNTER.
                  203 ;
0840 C5           204 BYTCNT: PUSH    B       ; SAVE B-C REGS
0841 D5           205         PUSH    D       ; SAVE D-E REGS
0842 0A           206         LDAX    B       ; LOAD A REG WITH BLRLSB
0843 6F           207         MOV     L,A     ; MOVE BLRLSB TO L REG
0844 03           208         INX     B       ;
0845 03           209         INX     B       ; INCREMENT B-C REGS TO ADDRESS THE ENABLE REG IN RAM
0846 0A           210         LDAX    B       ; LOAD A REG WITH ENABLE REG
0847 67           211         MOV     H,A     ; MOVE ENABLE REG TO H REG
0848 1640         212         MVI     D,40H   ; INITIALIZE D REG 64 BYTES/PAGE XFER, 40H
084A 3E60         213         MVI     A,60H   ; ERROR CORRECTION DETECTION MASK
084C A4           214         ANA     H       ; LOGICAL AND MASK WITH H REG, TEST FOR ERROR CORRECTION
084D C25208       215         JNZ     MULT    ; IF ZERO, ERROR CORRECTION IS NOT ENABLED
0850 1644         216         MVI     D,44H   ; NO ERROR CORRECTION, 68 BYTES/PAGE XFER, 44H
                  217                         ; MULTIPLY (D REG) X (L REG)
                  218                         ; 64 OR 68 BYTES X NO. OF PAGES IN BLRLSB
                  219                         ; RESULT WILL BE PLACED IN THE H-L REGS
                  220                         ; BEGIN MULTIPLY ROUTINE
0852 2600         221 MULT:   MVI     H,0H    ; INITIALIZE MOST SIGNIFICANT BYTE OF RESULT
0854 1E09         222         MVI     E,09H   ; INITIALIZE BIT COUNTER
0856 7D           223 MULTO:  MOV     A,L     ; MOVE LOW ORDER BYTE INTO A REG
0857 1F           224         RAR             ; ROTATE LEAST SIGNIFICANT BIT OF MULTIPLIER
0858 6F           225         MOV     L,A     ; MOVE LOW ORDER BYTE OF RESULT INTO L REG
0859 1D           226         DCR     E       ; DECREMENT BIT COUNTER
085A CA6708       227         JZ      DONE    ; EXIT IF COMPLETE
085D 7C           228         MOV     A,H     ; MOVE HIGH ORDER BYTE INTO A REG
085E D26208       229         JNC     MULT1   ; IF CARRY= 0, JMP MULTI
0861 82           230         ADD     D       ; ADD D REG TO A REG
0862 1F           231 MULT1:  RAR             ; CARRY= 0, SHIFT HIGH ORDER BYTE OF RESULT
0863 67           232         MOV     H,A     ; MOVE HIGH ORDER RESULT INTO H REG
0864 C35608       233         JMP     MULTO   ; CONTINUE LOOPING
0867 D1           234 DONE:   POP     D       ; RESTORE D-E REGS
0868 C1           235         POP     B       ; RESTORE B-C REGS
0869 C9           236         RET             ; RETURN TO CALL
                  237 ;
```



COMMENTS: THE PARAMETRIC REGS—BLRLSB AND THE ENABLE REG CAN NOT BE READ FROM THE 7220. THEY MUST BE READ FROM A MEMORY IMAGE IN RAM. SINCE ONLY THE BLRLSB IS USED TO COMPUTE THE BYTE COUNTER, DATA TRANSFERS ARE LIMITED TO 255 PAGES.

Figure 13. BYTCNT

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72    PAGE    6


 LOC  OBJ        LINE       SOURCE STATEMENT

                  238 ;*******************************************************************************
                  239 ;
                  240 ; FUNCTION: WRITE
                  241 ; INPUTS:   D-E REGS, STARTING ADDRESS OF DATA IN RAM
                  242 ;           H-L REGS, BYTE COUNTER
                  243 ;           BPK72 STATUS REG
                  244 ; OUTPUTS:  WRITE DATA TO BUBBLE MEMORY
                  245 ; CALLS:    NONE
                  246 ; DESTROYS: A, H, L, F/FS
                  247 ;
                  248 ; DESCRIPTION: TRANSFER DATA FROM RAM TO BUBBLE MEMORY
                  249 ;       THE  D-E REGS  CONTAIN THE STARTING  ADDRESS  IN  RAM OF  DATA
                  250 ;       TO BE WRITTEN  INTO  THE  BUBBLE  MEMORY.  THE  H-L  REGS MUST
                  251 ;       CONTAIN  A BYTE COUNTER  INDICATING  THE  NUMBER OF DATA BYTES
                  252 ;       TO BE TRANSFERRED.  THIS FUNCTION  BEGINS BY ISSUING THE WRITE
                  253 ;       BUBBLE MEMORY  DATA COMMAND FOLLOWED BY POLLING THE STATUS REG
                  254 ;       TO  DETERMINE IF THE 7220 FIFO DATA BUFFER IS READY TO RECEIVE
                  255 ;       DATA.  DATA IS  TRANSFERRED  UNTIL  THE  BYTE  COUNTER OR TIME
                  256 ;       OUT LOOP COUNTER DECREMENTS TO ZERO.  THE PARAMETRIC REGISTERS
                  257 ;       MUST BE LOADED  WITH THE DESIRED VALUES  PRIOR TO CALLING THIS
                  258 ;       FUNCTION.
                  259 ;
 086A D5          260 WRITE:  PUSH    D       ; SAVE D-E REGS
 086B C5          261         PUSH    B       ; SAVE B-C REGS
 086C 01FFFF      262         LXI     B,0FFFFH; INITIALIZE TIME OUT LOOP COUNTER
 086F 3E13        263         MVI     A,13H   ; LOAD A REG= WRITE BUBBLE MEMORY DATA COMMAND
 0871 D3FF        264         OUT     PRTA01  ; WRITE, WRITE BUBBLE MEMORY DATA COMMAND
 0873 0B          265 BUSYWR: DCX     B       ; DECREMENT TIME OUT LOOP COUNTER
 0874 AF          266         XRA     A       ; CLEAR A REG
 0875 B0          267         ORA     B       ; TEST B REG= 00H
 0876 B1          268         ORA     C       ; TEST C REG= 00H
 0877 CA8108      269         JZ      FINSHW  ; IF ZERO, TIME OUT ERROR, JMP FINSHW
 087A DBFF        270         IN      PRTA01  ; READ STATUS REG
 087C 07          271         RLC             ; TEST BUSY BIT= 1
 087D D27308      272         JNC     BUSYWR  ; IF ZERO, CONTINUE POLLING BUSY BIT
                  273                         ; CONTINUED ON NEXT PAGE
                  274 $EJECT
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72   PAGE   7


 LOC  OBJ        LINE       SOURCE STATEMENT

 0880 DBFF       275 POLLWR: IN     PRTA01  ; READ STATUS REG
 0882 0F         276         RRC            ; TEST FIFO READY BIT= 1
 0883 DA9608     277         JC     WFIFO   ; IF FIFO READY= 1, JMP WFIFO
 0886 DBFF       278         IN     PRTA01  ; READ STATUS REG
 0888 07         279         RLC            ; TEST BUSY BIT= 1
 0889 D2A108     280         JNC    FINSHW  ; IF ZERO, ERROR, JMP FINSHW
 088C 0B         281         DCX    B       ; DECREMENT TIME OUT LOOP COUNTER
 088D AF         282         XRA    A       ; CLEAR A REG
 088E B0         283         ORA    B       ; TEST B REG= 00H
 088F B1         284         ORA    C       ; TEST C REG= 00H
 0890 CAA108     285         JZ     FINSHW  ; IF ZERO, TIME OUT ERROR, JMP FINSHW
 0893 C38008     286         JMP    POLLWR  ; CONTINUE POLLING FIFO READY BIT
 0896 1A         287 WFIFO:  LDAX   D       ; LOAD A REG FROM D-E REG ADDRESS
 0897 D3FE       288         OUT    PRTA00  ; WRITE A REG TO 7220 FIFO DATA BUFFER
 0899 13         289         INX    D       ; INCREMENT D-E REGS TO NEXT ADDRESS IN RAM
 089A 2B         290         DCX    H       ; DECREMENT BYTE COUNTER
 089B AF         291         XRA    A       ; CLEAR A REG
 089C B4         292         ORA    H       ; TEST H REG= 00H
 089D B5         293         ORA    L       ; TEST L REG= 00H
 089E C28008     294         JNZ    POLLWR  ; IF BYTE COUNTER NOT ZERO, JMP POLLWR
 08A1 C1         295 FINSHW: POP    B       ; RESTORE B-C REGS
 08A2 D1         296         POP    D       ; RESTORE D-E REGS
 08A3 C9         297         RET            ; RETURN TO CALL
                 298 ;
                 299 ;
                 300 ;
                 301 $EJECT
```

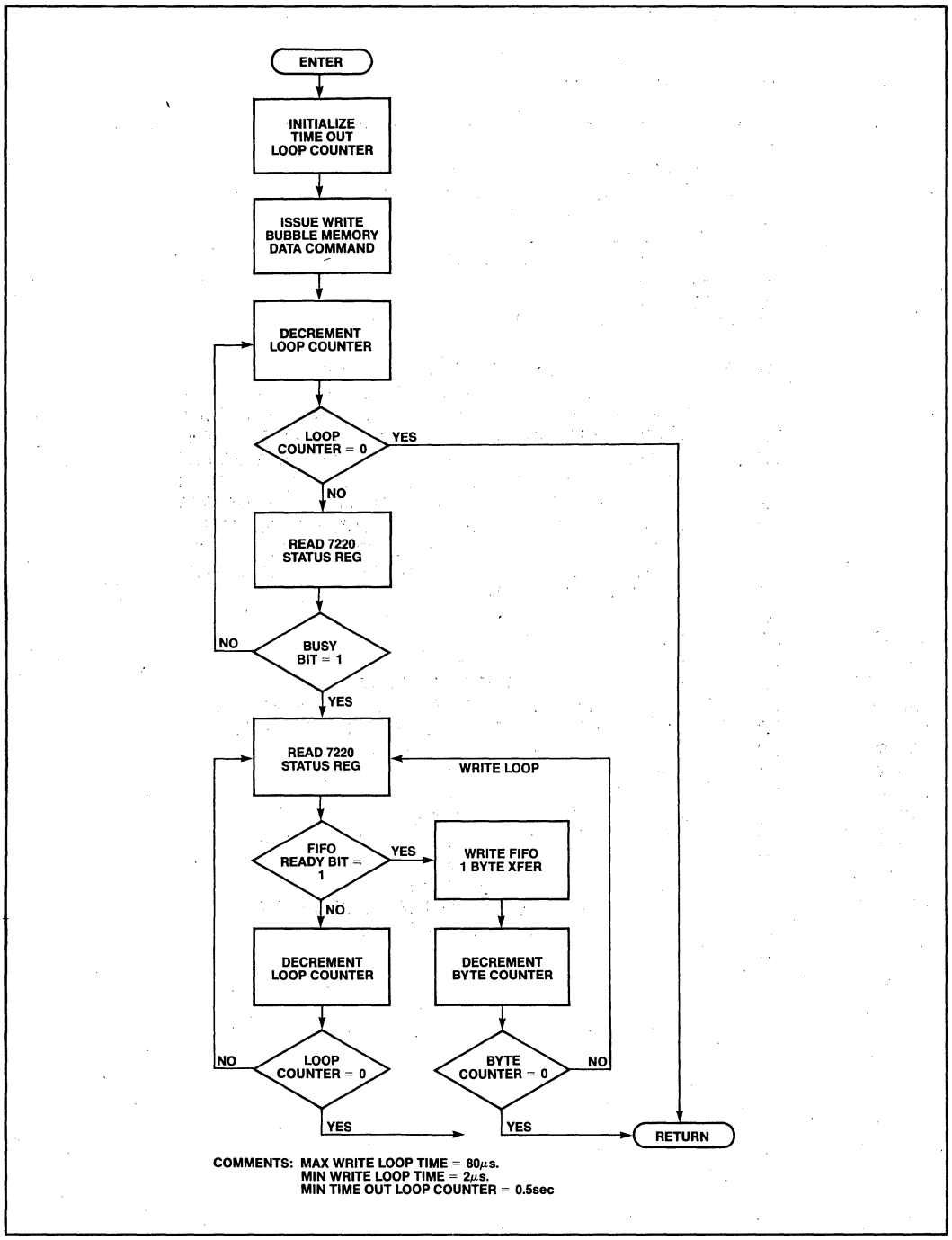


Figure 14. WRITE

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72    PAGE    8


 LOC  OBJ        LINE       SOURCE STATEMENT

                  302 ;*******************************************************************************
                  303 ;
                  304 ; FUNCTION:'READ
                  305 ; INPUTS:   D-E REGS, STARTING ADDRESS IN RAM
                  306 ;           H-L REGS, BYTE COUNTER
                  307 ;           BPK72 STATUS REG
                  308 ;           READ DATA FROM BUBBLE MEMORY
                  309 ; OUTPUTS:  WRITE DATA TO RAM
                  310 ; CALLS:    NONE
                  311 ; DESTROYS: A, H, L, F/FS
                  312 ;
                  313 ; DESCRIPTION: TRANSFER DATA FROM BUBBLE MEMORY TO RAM
                  314 ;       THE D-E REGS CONTAIN THE STARTING ADDRESS IN RAM USED TO STORE
                  315 ;       DATA READ FROM THE  BUBBLE MEMORY.  THE H-L  REGS MUST CONTAIN
                  316 ;       A BYTE  COUNTER  INDICATING  THE  NUMBER  OF DATA  BYTES TO BE
                  317 ;       TRANSFERRED.  THIS FUNCTION BEGINS  BY ISSUING THE READ BUBBLE
                  318 ;       MEMORY  DATA  COMMAND  FOLLOWED  BY  POLLING  THE  STATUS  REG
                  319 ;       TO DETERMINE  IF  THE  7220 FIFO  DATA  BUFFER  CONTAINS  DATA
                  320 ;       AVAILABLE FOR READING.  DATA  IS  TRANSFERRED  UNTIL  THE BYTE
                  321 ;       COUNTER  OR  TIME  OUT  LOOP COUNTER  DECREMENTS TO ZERO.  THE
                  322 ;       PARAMETRIC REGS MUST BE LOADED  WITH THE  DESIRED VALUES PRIOR
                  323 ;       TO CALLING THIS FUNCTION.
                  324 ;
 08A4 D5          325 READ:   PUSH    D       ; SAVE D-E REGS
 08A5 C5          326         PUSH    B       ; SAVE B-C REGS
 08A6 01FFFF      327         LXI     B,0FFFFH; INITIALIZE TIME OUT LOOP COUNTER
 08A9 3E12        328         MVI     A,12H   ; LOAD A REG= READ BUBBLE MEMORY DATA COMMAND
 08AB D3FF        329         OUT     PRTA01  ; WRITE, READ BUBBLE MEMORY DATA COMMAND
 08AD 0B          330 BUSYRD: DCX     B       ; DECREMENT TIME OUT LOOP COUNTER
 08AE AF          331         XRA     A       ; CLEAR A REG
 08AF B0          332         ORA     B       ; TEST B REG= 00H
 08B0 B1          333         ORA     C       ; TEST C REG= 00H
 08B1 CADB08      334         JZ      FINSHR  ; IF ZERO, TIME OUT ERROR, JMP FINSHR
 08B4 DBFF        335         IN      PRTA01  ; READ STATUS REG
 08B6 07          336         RLC             ; TEST BUSY BIT= 1
 08B7 D2AD08      337         JNC     BUSYRD  ; IF ZERO, CONTINUE POLLING BUSY BIT
                  338                         ; CONTINUED ON NEXT PAGE
                  339 $EJECT
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72    PAGE    9


 LOC  OBJ        LINE       SOURCE STATEMENT

 08BA DBFF        340 POLLRD: IN      PRTA01  ; READ STATUS REG
 08BC 0F          341         RRC             ; TEST FIFO READY BIT= 1
 08BD DAD008      342         JC      RFIFO   ; IF FIFO READY= 1, JMP RFIFO
 08C0 DBFF        343         IN      PRTA01  ; READ STATUS REG
 08C2 07          344         RLC             ; TEST BUSY BIT= 1
 08C3 D2DB08      345         JNC     FINSHR  ; IF ZERO, ERROR, JMP FINSHR
 08C6 0B          346         DCX     B       ; DECREMENT TIME OUT LOOP COUNTER
 08C7 AF          347         XRA     A       ; CLEAR A REG
 08C8 B0          348         ORA     B       ; TEST B REG= 00H
 08C9 B1          349         ORA     C       ; TEST C REG= 00H
 08CA CADB08      350         JZ      FINSHR  ; IF ZERO, TIME OUT ERROR, JMP FINSHR
 08CD C3BA08      351         JMP     POLLRD  ; CONTINUE POLLING FIFO READY BIT
 08D0 DBFE        352 RFIFO:  IN      PRTA00  ; LOAD A REG WITH ONE BYTE FROM FIFO DATA BUFFER
 08D2 12          353         STAX    D       ; STORE A REG IN REG D-E ADDRESS
 08D3 13          354         INX     D       ; INCREMENT D-E REGS TO NEXT ADDRESS IN RAM
 08D4 2B          355         DCX     H       ; DECREMENT BYTE COUNTER
 08D5 AF          356         XRA     A       ; CLEAR A REG
 08D6 B4          357         ORA     H       ; TEST H REG= 00H
 08D7 B5          358         ORA     L       ; TEST L REG= 00H
 08D8 C2BA08      359         JNZ     POLLRD  ; IF BYTE COUNTER NOT ZERO, JMP POLLRD
 08DB C1          360 FINSHR: POP     B       ; RESTORE B-C REGS
 08DC D1          361         POP     D       ; RESTORE D-E REGS
 08DD C9          362         RET             ; RETURN TO CALL
                  363 ;
                  364 ;
                  365 ;
                  366 $EJECT
```

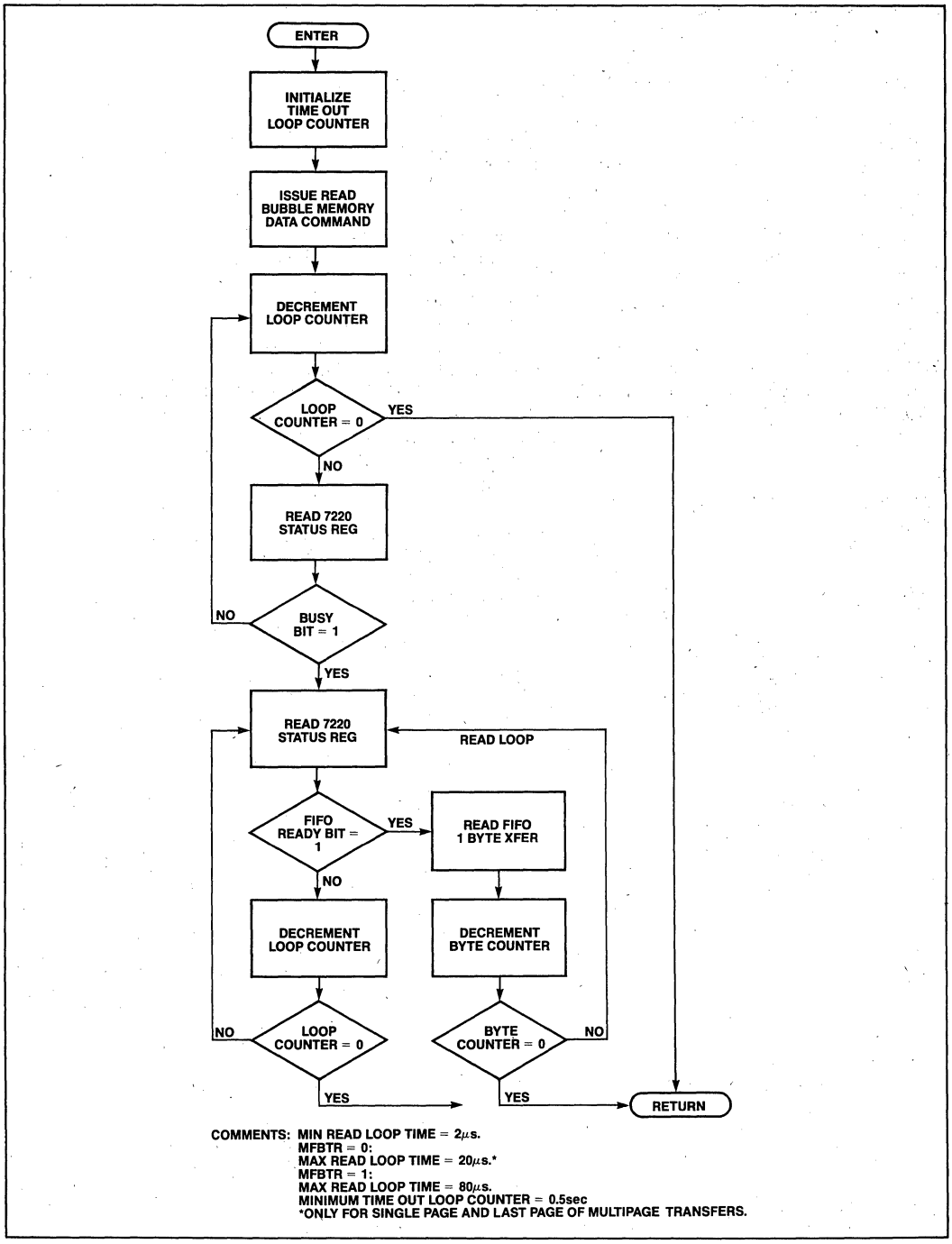


Figure 15. READ

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72    PAGE   10


 LOC  OBJ        LINE       SOURCE STATEMENT

                  367 ;*******************************************************************************
                  368 ;
                  369 ; FUNCTION: ABORT
                  370 ; INPUTS:   BPK72 STATUS REG
                  371 ; OUTPUTS:  ISSUE ABORT COMMAND TO BPK72
                  372 ;           A REG= BPK72 STATUS REG
                  373 ; CALLS:    NONE
                  374 ; DESTROYS: A, F/FS
                  375 ;
                  376 ; DESCRIPTION: ABORT PRESENT COMMAND, RESET BPK72
                  377 ;       AN ABORT  COMMAND IS  ISSUED  TO THE BPK72.  AFTER  ISSUING THE
                  378 ;       COMMAND,  THE  BPK72 STATUS REG IS POLLED UNTIL AN OP-COMPLETE,
                  379 ;       40H,  HAS BEEN  READ  OR THE  TIME OUT  LOOP COUNTER DECREMENTS
                  380 ;       TO ZERO.  THE ABORT FUNCTION  RETURNS  THE  VALUE  OF THE BPK72
                  381 ;       STATUS REG TO THE CALLING ROUTINE VIA THE 8085'S A REG.  ONLY A
                  382 ;       STATUS  OF 40H INDICATES A  SUCCESSFUL  EXECUTION  OF THE ABORT
                  383 ;       FUNCTION.
                  384 ;
                  385         PUBLIC  ABORT   ; DECLARE PUBLIC FUNCTION
 08DE D5          386 ABORT:  PUSH    D       ; SAVE D-E REGS
 08DF C5          387         PUSH    B       ; SAVE B-C REGS
 08E0 11FFFF      388         LXI     D,0FFFFH; INITIALIZE TIME OUT LOOP COUNTER
 08E3 0640        389         MVI     B,40H   ; LOAD B REG= 40H, OP-COMPLETE
 08E5 3E19        390         MVI     A,19H   ; LOAD A REG= ABORT COMMAND
 08E7 D3FF        391         OUT     PRTA01  ; WRITE ABORT COMMAND
 08E9 DBFF        392 BUSYA:  IN      PRTA01  ; READ STATUS REG
 08EB 07          393         RLC             ; TEST BUSY BIT= 1
 08EC DAF908      394         JC      POLLA   ; IF BUSY= 1, POLL STATUS REG FOR 40H
 08EF 1B          395         DCX     D       ; DECREMENT TIME OUT LOOP COUNTER
 08F0 AF          396         XRA     A       ; CLEAR A REG
 08F1 B2          397         ORA     D       ; TEST D REG= 00H
 08F2 B3          398         ORA     E       ; TEST E REG= 00H
 08F3 C2E908      399         JNZ     BUSYA   ; IF NOT ZERO, CONTINUE POLLING ABORT COMMAND
 08F6 C30609      400         JMP     RETA    ; TIME OUT ERROR, RETURN
 08F9 DBFF        401 POLLA:  IN      PRTA01  ; READ STATUS REG
 08FB A8          402         XRA     B       ; TEST STATUS= 40H, OP-COMPLETE
 08FC CA0609      403         JZ      RETA    ; IF OP-COMPLETE, JMP RETA
 08FF 1B          404         DCX     D       ; DECREMENT TIME OUT LOOP COUNTER
 0900 AF          405         XRA     A       ; CLEAR A REG
 0901 B2          406         ORA     D       ; TEST D REG= 00H
 0902 B3          407         ORA     E       ; TEST E REG= 00H
 0903 C2F908      408         JNZ     POLLA   ; IF NOT ZERO, CONTINUE POLLING ABORT COMMAND
 0906 C1          409 RETA:   POP     B       ; RESTORE B-C REGS
 0907 D1          410         POP     D       ; RESTORE D-E REGS
 0908 DBFF        411         IN      PRTA01  ; READ STATUS REG
 090A C9          412         RET             ; RETURN TO CALL
                  413 ;
                  414 ;
                  415 ;
                  416 $EJECT
```

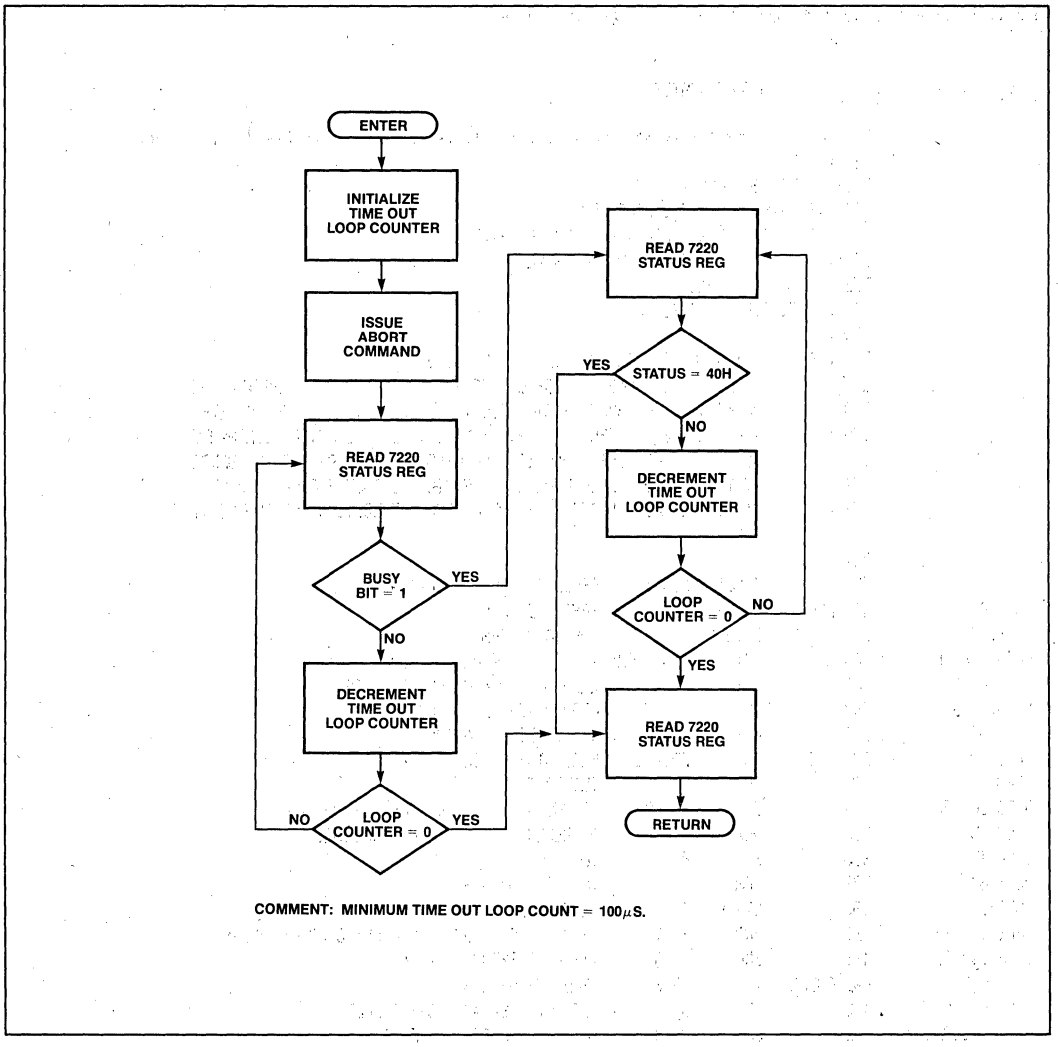


Figure 16. ABORT

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72    PAGE   11

 LOC  OBJ        LINE       SOURCE STATEMENT

                  417 ;*******************************************************************************
                  418 ;
                  419 ; FUNCTION: WRBUBL
                  420 ; INPUTS:   B-C REGS, STARTING ADDRESS OF PARAMETRIC REGS IN RAM
                  421 ;           D-E REGS, STARTING ADDRESS OF DATA IN RAM
                  422 ;           BPK72 STATUS REG
                  423 ; OUTPUTS:  WRITE DATA TO BUBBLE MEMORY
                  424 ;           A REG= BPK72 STATUS REG
                  425 ; CALLS:    FIFORS
                  426 ;           INTPAR
                  427 ;           BYTCNT
                  428 ;           WRITE
                  429 ; DESTROYS: A, F/FS
                  430 ;
                  431 ; DESCRIPTION: WRITE BUBBLE MEMORY DATA
                  432 ;       THE B-C  REGS CONTAIN THE ADDRESS  TO THE  FIRST  OF FIVE
                  433 ;       CONTIGUOUS  MEMORY  LOCATIONS IN RAM.  THE DATA ADDRESSED
                  434 ;       BY THE  B-C  REGS IS USED  TO LOAD THE  PARAMETRIC  REGS.
                  435 ;       THE D-E  REGS  CONTAIN  THE STARTING ADDRESS  IN  RAM  OF
                  436 ;       DATA TO BE WRITTEN INTO THE BUBBLE MEMORY. GIVEN THE DATA
                  437 ;       IN RAM  USED TO  LOAD THE PARAMETRIC REGS,  THIS FUNCTION
                  438 ;       WILL  RESET  THE 7220 FIFO,  LOAD  THE  PARAMETRIC  REGS,
                  439 ;       COMPUTE THE BYTE COUNTER, AND COPY THE DATA FROM RAM INTO
                  440 ;       THE BUBBLE MEMORY.  WRBUBL RETURNS THE VALUE OF THE BPK72
                  441 ;       STATUS  REG TO THE CALLING ROUTINE VIA  THE 8085'S A REG.
                  442 ;       ONLY  A  STATUS  OF  40H OR 42H  INDICATES  A  SUCCESSFUL
                  443 ;       EXECUTION OF WRBUBL.
                  444 ;
                  445         PUBLIC  WRBUBL  ; DECLARE PUBLIC FUNCTION
 090B E5          446 WRBUBL: PUSH    H       ; SAVE H-L REGS
 090C C5          447         PUSH    B       ; SAVE B-C REGS
 090D 0640        448         MVI     B,40H   ; LOAD B REG= 40H, OP-COMPLETE
 090F CD1308      449         CALL    FIFORS  ; CALL FIFORS, WRITE FIFO RESET COMMAND
 0912 A8          450         XRA     B       ; TEST FOR STATUS= 40H, OP-COMPLETE
 0913 C23109      451         JNZ     RETWR   ; IF NOT ZERO, FIFO ERROR, JMP RETWR
 0916 C1          452         POP     B       ; RESTORE B-C REGS
 0917 CD0008      453         CALL    INTPAR  ; CALL INTPAR, LOAD PARAMETRIC REGS
 091A CD4008      454         CALL    BYTCNT  ; CALL BYTCNT, COMPUTE BYTE COUNTER
 091D CD6A08      455         CALL    WRITE   ; CALL WRITE,  WRITE BUBBLE DATA
 0920 C5          456         PUSH    B       ; SAVE B-C REGS
                  457                         ; CONTINUED ON NEXT PAGE
                  458 $EJECT
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72    PAGE   12


 LOC  OBJ        LINE        SOURCE STATEMENT

 0921 21FFFF      459         LXI     H,0FFFFH; INITIALIZE TIME OUT LOOP COUNTER
 0924 DBFF        460 LOOPWR: IN      PRTA01  ; READ STATUS REG
 0926 07          461         RLC             ; TEST FOR BUSY BIT= 1
 0927 D23109      462         JNC     RETWR   ; IF ZERO, NOT BUSY, JMP RETWR
 092A 2B          463         DCX     H       ; DECREMENT TIME OUT LOOP COUNTER
 092B AF          464         XRA     A       ; CLEAR A REG
 092C B4          465         ORA     H       ; TEST H REG= 00H
 092D B5          466         ORA     L       ; TEST L REG= 00H
 092E C22409      467         JNZ     LOOPWR  ; CONTINUE POLLING STATUS REG
 0931 C1          468 RETWR:  POP     B       ; RESTORE B-C REGS
 0932 E1          469         POP     H       ; RESTORE H-L REGS
 0933 DBFF        470         IN      PRTA01  ; READ STATUS REG
 0935 C9          471         RET             ; RETURN TO CALL
                  472 ;
                  473 ;
                  474 ;
                  475 $EJECT
```

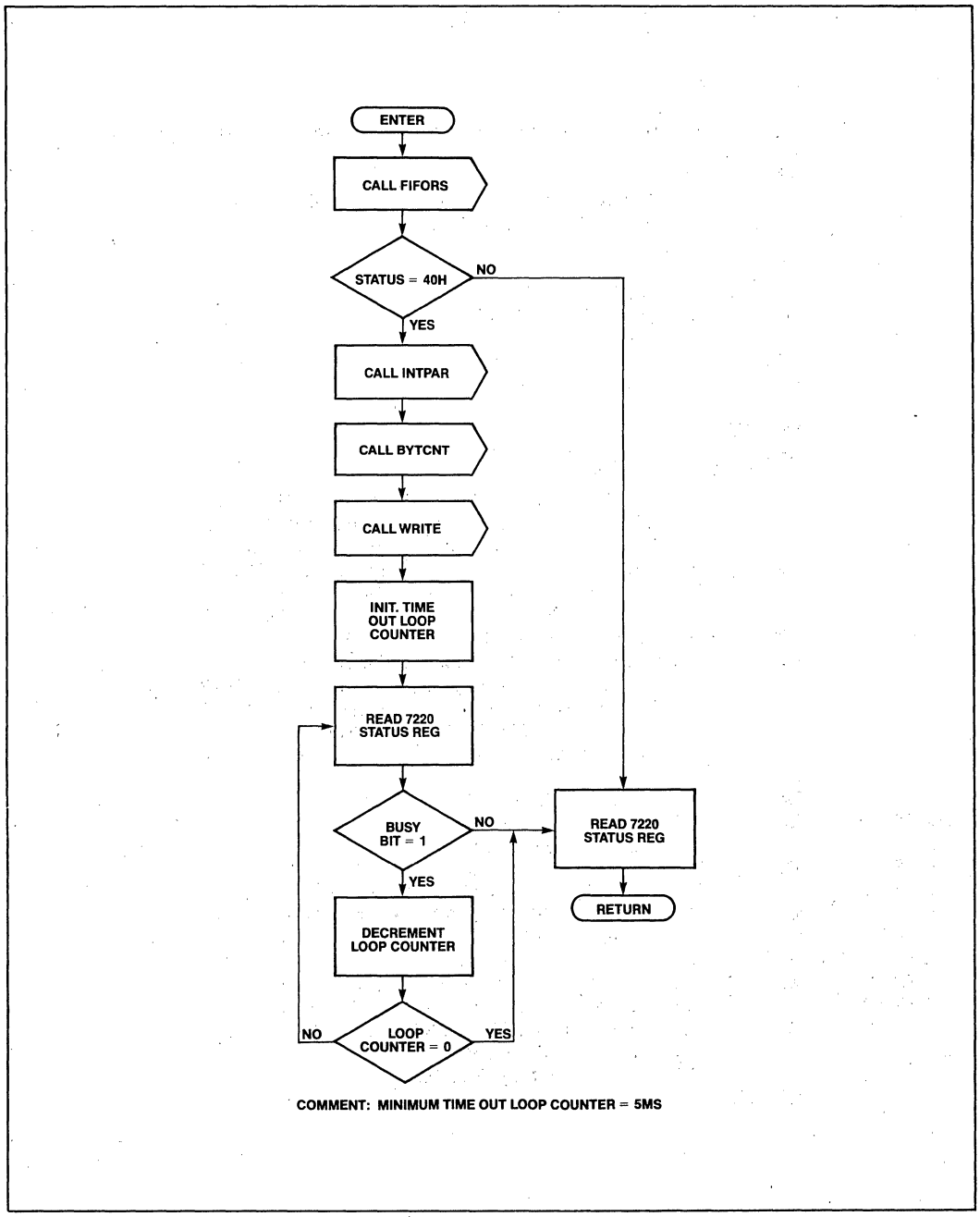


Figure 17. WRBUBL

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72    PAGE   13


 LOC  OBJ        LINE       SOURCE STATEMENT

                  476 ;*******************************************************************************
                  477 ;
                  478 ; FUNCTION: RDBUBL
                  479 ; INPUTS:   B-C REGS, STARTING ADDRESS OF PARAMETRIC REGS IN RAM
                  480 ;           D-E REGS, STARTING ADDRESS IN RAM
                  481 ;           BPK72 STATUS REG
                  482 ;           READ DATA FROM BUBBLE MEMORY
                  483 ; OUTPUTS:  WRITE DATA TO RAM
                  484 ;           A REG= BPK72 STATUS REG
                  485 ; CALLS:    FIFORS
                  486 ;           INTPAR
                  487 ;           BYTCNT
                  488 ;           READ
                  489 ; DESTROYS: A, F/FS
                  490 ;
                  491 ; DESCRIPTION: READ BUBBLE MEMORY DATA
                  492 ;       THE B-C REGS CONTAIN THE ADDRESS TO THE FIRST OF FIVE
                  493 ;       CONTIGUOUS MEMORY LOCATIONS IN RAM. THE DATA ADDRESSED
                  494 ;       BY THE B-C REGS IS USED TO LOAD THE PARAMETRIC REGS. THE D-E
                  495 ;       REGS CONTAIN THE STARTING ADDRESS IN RAM USED TO STORE
                  496 ;       DATA READ FROM THE BUBBLE MEMORY. GIVEN THE DATA IN RAM
                  497 ;       USED TO LOAD THE PARAMETRIC REGS, THIS FUNCTION WILL RESET
                  498 ;       THE 7220 FIFO, LOAD THE PARAMETRIC REGS, COMPUTE THE
                  499 ;       BYTE COUNTER, AND COPY THE DATA FROM THE BUBBLE MEMORY INTO
                  500 ;       RAM. RDBUBL RETURNS THE VALUE OF THE BPK72 STATUS REGISTER
                  501 ;       TO THE CALLING ROUTINE VIA THE 8085'S A REG. ONLY A STATUS
                  502 ;       OF 40H OR 48H WITH ERROR CORRECTION INDICATES A SUCCESSFUL
                  503 ;       EXECUTION OF RDBUBL.
                  504 ;
                  505        PUBLIC  RDBUBL  ; DECLARE PUBLIC FUNCTION
 0936 E5          506 RDBUBL: PUSH    H      ; SAVE H-L REGS
 0937 C5          507        PUSH    B       ; SAVE B-C REGS
 0938 0640        508        MVI     B,40H   ; LOAD B REG= 40H, OP-COMPLETE
 093A CD1308      509        CALL    FIFORS  ; CALL FIFORS, WRITE FIFO RESET COMMAND
 093D A8          510        XRA     B       ; TEST FOR STATUS= 40H, OP-COMPLETE
 093E C25C09      511        JNZ     RETRD   ; IF NOT ZERO, FIFO ERROR, JMP RETRD
 0941 C1          512        POP     B       ; RESTORE B-C REGS
 0942 CD0008      513        CALL    INTPAR  ; CALL INTPAR, LOAD PARAMETRIC REGS
 0945 CD4008      514        CALL    BYTCNT  ; CALL BYTCNT, COMPUTE BYTE COUNTER
 0948 CDA408      515        CALL    READ    ; CALL READ, READ BUBBLE DATA
 094B C5          516        PUSH    B       ; SAVE B-C REGS
                  517                        ; CONTINUED ON NEXT PAGE
                  518 $EJECT
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72    PAGE   14


 LOC  OBJ        LINE       SOURCE STATEMENT

 094C 21FFFF      519        LXI     H,0FFFFH; INITIALIZE TIME OUT LOOP COUNTER
 094F DBFF        520 LOOPRD: IN     PRTA01  ; READ STATUS REG
 0951 07          521        RLC             ; TEST FOR BUSY BIT= 1
 0952 D25C09      522        JNC     RETRD   ; IF ZERO, NOT BUSY, JMP RETRD
 0955 2B          523        DCX     H       ; DECREMENT TIME OUT LOOP COUNTER
 0956 AF          524        XRA     A       ; CLEAR A REG
 0957 B4          525        ORA     H       ; TEST H REG= 00H
 0958 B5          526        ORA     L       ; TEST L REG= 00H
 0959 C24F09      527        JNZ     LOOPRD  ; CONTINUE POLLING STATUS REG
 095C C1          528 RETRD:  POP     B       ; RESTORE B-C REGS
 095D E1          529        POP     H       ; RESTORE H-L REGS
 095E DBFF        530        IN      PRTA01  ; READ STATUS REG
 0960 C9          531        RET             ; RETURN TO CALL
                  532 ;
                  533 ;
                  534 ;
                  535 $EJECT
```

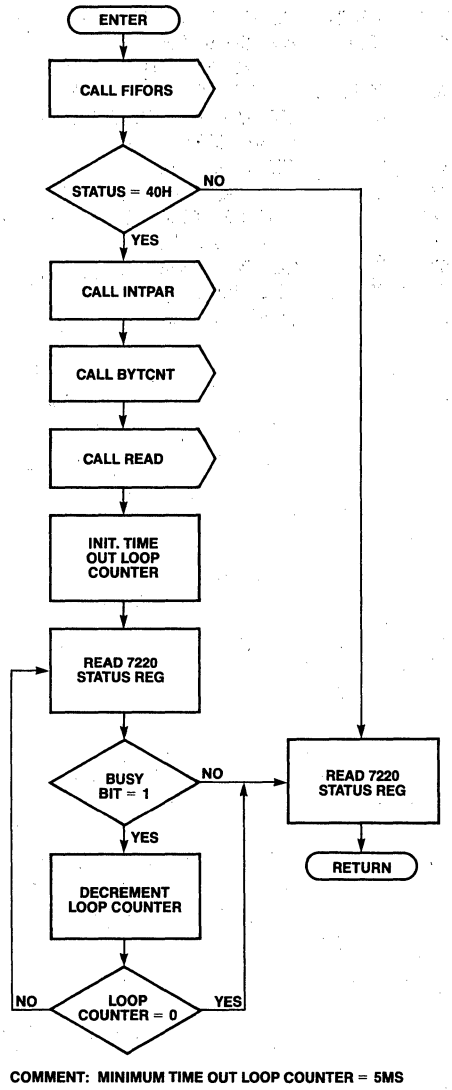


Figure 18. RDBUBL

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72    PAGE   15


 LOC  OBJ        LINE       SOURCE STATEMENT

                  536 ;*******************************************************************************
                  537 ;
                  538 ; FUNCTION: INBUBL
                  539 ; INPUTS:   B-C REGS, STARTING ADDRESS OF PARAMETRIC REGS IN RAM
                  540 ;           BPK72 STATUS REG
                  541 ; OUTPUTS:  A REG= BPK72 STATUS REG
                  542 ; CALLS:    ABORT
                  543 ;           INTPAR
                  544 ; DESTROYS: A, F/FS
                  545 ;
                  546 ; DESCRIPTION: INITIALIZE THE BPK72
                  547 ;       THE B-C REGS CONTAIN THE ADDRESS TO THE FIRST OF  FIVE CONTIGUOUS
                  548 ;       MEMORY LOCATIONS IN RAM.  THE  DATA ADDRESSED  BY  THE  B-C  REGS
                  549 ;       IS  USED TO  LOAD THE  PARAMETRIC REGS.  THIS FUNCTION WILL WRITE
                  550 ;       THE  PARAMETRIC  REGS  FOLLOWED  BY  ISSUING  A  BUBBLE  MEMORY
                  551 ;       INITIALIZATION  COMMAND.  AFTER  ISSUING  THE COMMAND,  THE BPK72
                  552 ;       STATUS REG IS POLLED UNTIL AN OP-COMPLETE,  40H,  IS  READ OR THE
                  553 ;       TIME  OUT  LOOP  COUNTER  DECREMENTS TO ZERO.  THIS  COMMAND MUST
                  554 ;       PRECEED ALL  OTHER  COMMANDS AFTER POWERING UP THE BPK72.  INBUBL
                  555 ;       RETURNS THE VALUE OF THE BPK72 STATUS  REG TO THE CALLING ROUTINE
                  556 ;       VIA THE 8085'S A REG.  ONLY A STATUS OF 40H INDICATES A SUCCESSFUL
                  557 ;       EXECUTION OF INBUBL.
                  558 ;
                  559         PUBLIC  INBUBL  ; DECLARE PUBLIC FUNCTION
 0961 D5          560 INBUBL: PUSH    D       ; SAVE D-E REGS
 0962 C5          561         PUSH    B       ; SAVE B-C REGS
 0963 0640        562         MVI     B,40H   ; LOAD B REG= 40H, OP-COMPLETE
 0965 CDDE08      563         CALL    ABORT   ; CALL ABORT COMMAND
 0968 A8          564         XRA     B       ; TEST STATUS= 40H, OP-COMPLETE
 0969 C29709      565         JNZ     RETIN   ; IF ZERO, OP-COMPLETE, CONTINUE
 096C C1          566         POP     B       ; PARAMETRIC REGS STARTING ADDRESS IN REG B
 096D CD0008      567         CALL    INTPAR  ; CALL INTPAR, LOAD PARAMETRIC REGS
 0970 C5          568         PUSH    B       ; SAVE B-C REGS
 0971 0640        569         MVI     B,40H   ; LOAD B REG= 40H, OP-COMPLETE
 0973 11FFFF      570         LXI     D,0FFFFH; INITIALIZE TIME OUT LOOP COUNTER
 0976 3E11        571         MVI     A,11H   ; LOAD A REG= INITIALIZE COMMAND
 0978 D3FF        572         OUT     PRTA01  ; WRITE INITIALIZE COMMAND
                  573                         ; CONTINUED ON NEXT PAGE
                  574 $EJECT
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72    PAGE   16

 LOC  OBJ        LINE        SOURCE STATEMENT

 097A DBFF        575 BUSYIN: IN      PRTA01  ; READ STATUS REG
 097C 07          576         RLC             ; TEST BUSY BIT= 1
 097D DA8A09      577         JC      POLLIN  ; IF BUSY= 1, POLL STATUS REG FOR 40H
 0980 1B          578         DCX     D       ; DECREMENT TIME OUT LOOP COUNTER
 0981 AF          579         XRA     A       ; CLEAR A REG
 0982 B2          580         ORA     D       ; TEST D REG= 00H
 0983 B3          581         ORA     E       ; TEST E REG= 00H
 0984 C27A09      582         JNZ     BUSYIN  ; IF NOT ZERO, CONTINUE POLLING THE INITIALIZE COMMAND
 0987 C39709      583         JMP     RETIN   ; TIME OUT ERROR, RETURN
 098A DBFF        584 POLLIN: IN      PRTA01  ; READ STATUS REG
 098C A8          585         XRA     B       ; TEST STATUS= 40H, OP-COMPLETE
 098D CA9709      586         JZ      RETIN   ; IF OP-COMPLETE, JMP RETIN
 0990 1B          587         DCX     D       ; DECREMENT TIME OUT LOOP COUNTER
 0991 AF          588         XRA     A       ; CLEAR A REG
 0992 B2          589         ORA     D       ; TEST D REG= 00H
 0993 B3          590         ORA     E       ; TEST E REG= 00H
 0994 C28A09      591         JNZ     POLLIN  ; IF NOT ZERO, CONTINUE POLLING INITIALIZE COMMAND
 0997 C1          592 RETIN:  POP     B       ; RESTORE B-C REGS
 0998 D1          593         POP     D       ; RESTORE D-E REGS
 0999 DBFF        594         IN      PRTA01  ; READ STATUS REG
 099B C9          595         RET             ; RETURN TO CALL
                  596 ;
                  597 ;
                  598 $EJECT
```

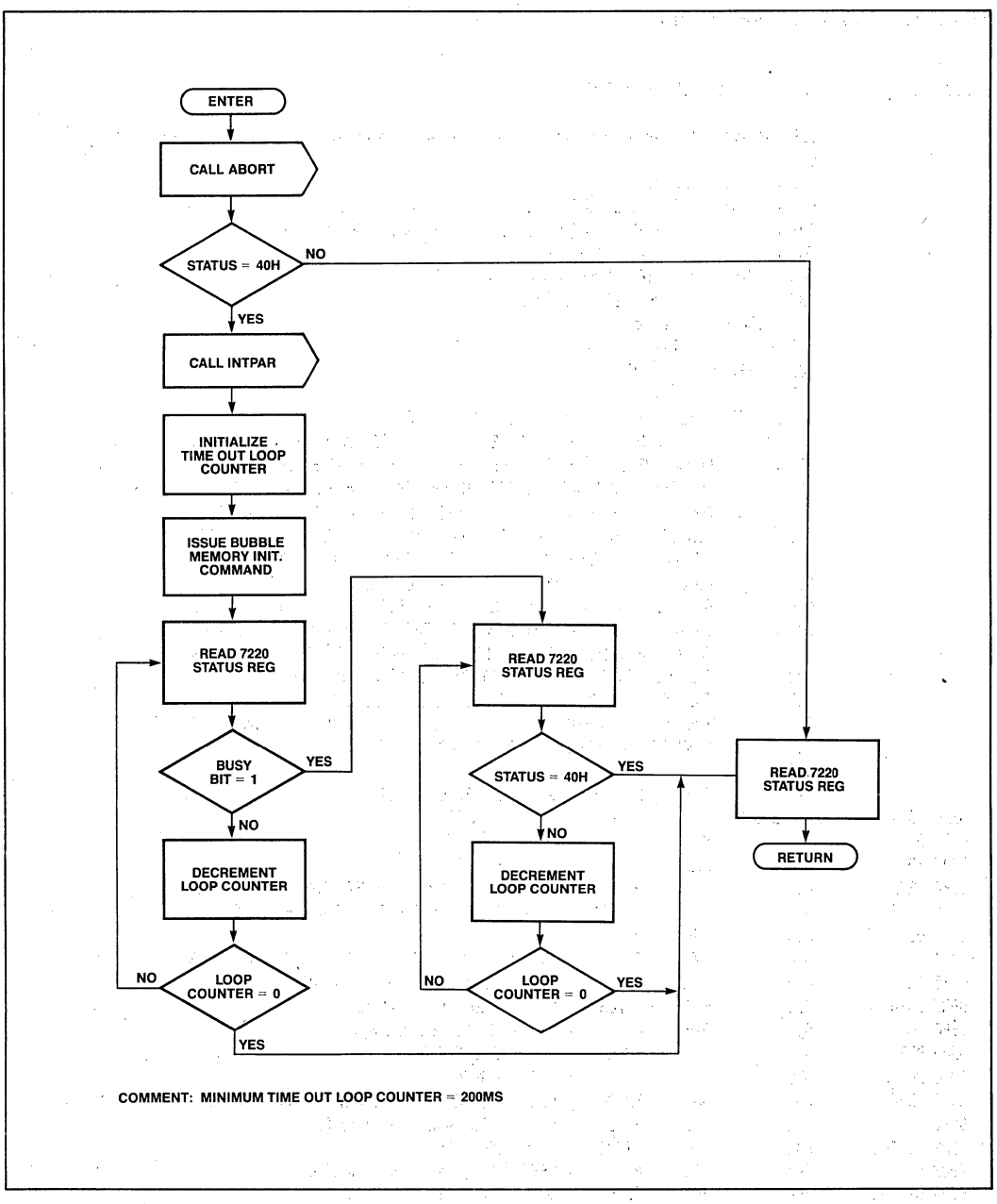


Figure 19. INBUBL

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72   PAGE  17


 LOC  OBJ        LINE       SOURCE STATEMENT

                  599 ;*******************************************************************************
                  600 ;
                  601 ; FUNCTION: BOOTUP
                  602 ; INPUTS:   B-C REGS, STARTING ADDRESS OF PARAMETRIC REGS IN RAM
                  603 ;           D-E REGS, STARTING ADDRESS OF BOOT LOOP CODE IN RAM
                  604 ;           BPK72 STATUS REG
                  605 ; OUTPUTS:  WRITE BUBBLE MEMORY BOOT LOOP
                  606 ;           A REG= BPK72 STATUS REG
                  607 ; CALLS:    FIFORS
                  608 ;           INTPAR
                  609 ; DESTROYS: A, F/FS
                  610 ;
                  611 ; DESCRIPTION: WRITE BUBBLE MEMORY BOOT LOOP
                  612 ;       THIS  FUNCTION WILL  WRITE THE  BOOT LOOP CODE  INTO THE 7110
                  613 ;       BUBBLE MEMORY.  THE D-E  REGS PROVIDE THE ADDRESS TO THE FIRST
                  614 ;       OF FORTY  CONTIGUOUS BYTES  IN RAM THAT CONTAIN THE BOOT LOOP
                  615 ;       CODE.  THE B-C REGS  CONTAIN  THE ADDRESS TO THE FIRST OF FIVE
                  616 ;       CONTIGUOUS MEMORY  LOCATIONS ALSO IN RAM.  THE DATA ADDRESSED
                  617 ;       BY THE  B-C  REGS  IS  USED  TO  LOAD  THE  PARAMETRIC  REGS.
                  618 ;       NOTE  THAT  THE  PARAMETRIC  ENABLE  REG  WRITE  BOOT  LOOP
                  619 ;       BIT  IS  AUTOMATICALLY  SET  AND A  FORTY-FIRST BYTE  OF ZERO
                  620 ;       IS WRITTEN  TO THE  FIFO DATA BUFFER TO AVOID A TIMING ERROR.
                  621 ;       BEFORE  A RETURN IS EXECUTED,  THE PARAMETRIC  ENABLE  REG IS
                  622 ;       RESTORED TO ITS VALUE PRIOR TO CALLING BOOTUP.  BOOTUP RETURNS
                  623 ;       THE VALUE OF THE BPK72  STATUS REG TO THE CALLING ROUTINE VIA
                  624 ;       THE 8085'S  A REG.  ONLY  A STATUS  OF  40H OR 42H  INDICATES
                  625 ;       A SUCCESSFUL EXECUTION OF BOOTUP.
                  626 ;
                  627         PUBLIC  BOOTUP  ; DECLARE PUBLIC FUNCTION
 099C C5          628 BOOTUP: PUSH    B       ; SAVE B-C REGS
 099D D5          629         PUSH    D       ; SAVE D-E REGS
 099E E5          630         PUSH    H       ; SAVE H-L REGS
 099F 3E0D        631         MVI     A,0DH   ; LOAD A REG= 0DH, 7220 RAC ENABLE REG ADDRESS
 09A1 D3FF        632         OUT     PRTA01  ; WRITE 7220 RAC WITH ENABLE REG ADDRESS
 09A3 03          633         INX     B       ;
 09A4 03          634         INX     B       ; INCREMENT B-C REGS TO ENABLE REG RAM ADDRESS
 09A5 0A          635         LDAX    B       ; LOAD A REG= ENABLE REG FROM RAM
 09A6 0610        636         MVI     B,10H   ; LOAD B REG= BOOT LOOP ENABLE MASK
 09A8 B0          637         ORA     B       ; SET BOOT LOOP ENABLE BIT
 09A9 D3FE        638         OUT     PRTA00  ; WRITE ENABLE REG
 09AB AF          639         XRA     A       ; CLEAR A REG
 09AC D3FF        640         OUT     PRTA01  ; LOAD 7220 RAC WITH FIFO DATA BUFFER ADDRESS
 09AE 0640        641         MVI     B,40H   ; LOAD B REG= 40H, OP-COMPLETE
 09B0 CD1308      642         CALL    FIFORS  ; CALL FIFO RESET
 09B3 A8          643         XRA     B       ; TEST STATUS= 40H, OP-COMPLETE
 09B4 C2230A      644         JNZ     RETBT   ; IF NOT ZERO, ERROR, JMP RETBT
                  645                         ; CONTINUED ON NEXT PAGE
                  646 $EJECT
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72   PAGE  18

 LOC  OBJ        LINE        SOURCE STATEMENT

 09B7 0E28        647         MVI     C,28H   ; LOAD C REG= 28H, BYTE COUNTER= 40 DECIMAL
 09B9 3EFF        648         MVI     A,0FFH  ; LOAD A REG= FFH
 09BB D3FE        649 ALLFFS: OUT     PRTA00  ; WRITE A REG INTO FIFO DATA BUFFER
 09BD 0D          650         DCR     C       ; DECREMENT BYTE COUNTER
 09BE C2BB09      651         JNZ     ALLFFS  ; IF BYTE COUNTER= ZERO, CONTINUE
 09C1 21FFFF      652         LXI     H,0FFFFH; INITIALIZE TIME OUT LOOP COUNTER
 09C4 3E16        653         MVI     A,16H   ; LOAD A REG= WRITE BOOT LOOP REG COMMAND
 09C6 D3FF        654         OUT     PRTA01  ; WRITE, WRITE BOOT LOOP REG COMMAND
 09C8 DBFF        655 BUSYB:  IN      PRTA01  ; READ STATUS REG
 09CA 07          656         RLC             ; TEST BUSY BIT= 1
 09CB DAD809      657         JC      POLLBR  ; IF BUSY= 1, POLL STATUS REG FOR 40H
 09CE 2B          658         DCX     H       ; DECREMENT TIME OUT LOOP COUNTER
 09CF AF          659         XRA     A       ; CLEAR A REG
 09D0 B4          660         ORA     H       ; TEST H REG= 00H
 09D1 B5          661         ORA     L       ; TEST L REG= 00H
 09D2 C2C809      662         JNZ     BUSYB   ; IF NOT ZERO, CONTINUE POLLING WRBLRS COMMAND
 09D5 C3230A      663         JMP     RETBT   ; TIME OUT ERROR, RETURN
 09D8 DBFF        664 POLLBR: IN      PRTA01  ; READ STATUS REG
 09DA A8          665         XRA     B       ; TEST STATUS= 40H
 09DB CAE809      666         JZ      CONT    ; IF ZERO, CONTINUE, OP-COMPLETE
 09DE 2B          667         DCX     H       ; DECREMENT TIME OUT LOOP COUNTER
 09DF AF          668         XRA     A       ; CLEAR A REG
 09E0 B4          669         ORA     H       ; TEST H REG= 00H
 09E1 B5          670         ORA     L       ; TEST L REG= 00H
 09E2 CA230A      671         JZ      RETBT   ; IF ZERO, TIME OUT, ERROR
 09E5 C3D809      672         JMP     POLLBR  ; CONTINUE POLLING WRBLRS COMMAND
                  673                         ; CONTINUED ON NEXT PAGE
                  674 $EJECT
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72   PAGE   19


 LOC  OBJ        LINE        SOURCE STATEMENT

 09E8 CD1308      675 CONT:   CALL    FIFORS  ; CALL FIFO RESET
 09EB A8          676         XRA     B       ; TEST STATUS= 40H
 09EC C2230A      677         JNZ     RETBT   ; IF NOT ZERO, ERROR, JMP RETBT
 09EF 0E28        678         MVI     C,28H   ; LOAD C REG= 28H, BYTE COUNTER= 40 DECIMAL
 09F1 1A          679 BLCODE: LDAX    D       ; LOAD A REG FROM D REG ADDRESS
 09F2 13          680         INX     D       ; INCREMENT D REG TO THE NEXT ADDRESS
 09F3 D3FE        681         OUT     PRTA00  ; WRITE BOOT LOOP CODE INTO FIFO DATA BUFFER
 09F5 0D          682         DCR     C       ; DECREMENT BYTE COUNTER
 09F6 C2F109      683         JNZ     BLCODE  ; IF NOT ZERO, JMP BLCODE
 09F9 AF          684         XRA     A       ; CLEAR A REG
 09FA D3FE        685         OUT     PRTA00  ; WRITE 415T BYTE OF ZERO INTO FIFO DATA BUFFER
 09FC 21FFFF      686         LXI     H,0FFFFH; LOAD TIME OUT LOOP COUNTER
 09FF 0EFD        687         MVI     C,0FDH  ; MASK, MASK OUT PARITY BIT
 0A01 3E17        688         MVI     A,17H   ; LOAD A REG= WRITE BOOT LOOP COMMAND
 0A03 D3FF        689         OUT     PRTA01  ; WRITE, WRITE BOOT LOOP COMMAND
 0A05 DBFF        690 BUSYBL: IN      PRTA01  ; READ STATUS REG
 0A07 07          691         RLC             ; TEST BUSY BIT= 1
 0A08 DA150A      692         JC      POLLBL  ; IF BUSY=1, POLL STATUS REG FOR OP-COMPLETE
 0A0B 2B          693         DCX     H       ; DECREMENT TIME OUT LOOP COUNTER
 0A0C AF          694         XRA     A       ; CLEAR A REG
 0A0D B4          695         ORA     H       ; TEST H REG= 00H
 0A0E B5          696         ORA     L       ; TEST L REG= 00H
 0A0F C2050A      697         JNZ     BUSYBL  ; IF NOT ZERO, CONTINUE POLLING THE WRBL COMMAND
 0A12 C3230A      698         JMP     RETBT   ; TIME OUT ERROR, RETURN
 0A15 DBFF        699 POLLBL: IN      PRTA01  ; READ STATUS REG
 0A17 A1          700         ANA     C       ; RESET BIT 1, PARITY BIT
 0A18 A8          701         XRA     B       ; TEST STATUS= 40H OR 42H, OP-COMPLETE
 0A19 CA230A      702         JZ      RETBT   ; IF ZERO, CONTINUE, OP-COMPLETE
 0A1C 2B          703         DCX     H       ; DECREMENT TIME OUT LOOP COUNTER
 0A1D AF          704         XRA     A       ; CLEAR A REG
 0A1E B4          705         ORA     H       ; TEST H REG= 00H
 0A1F B5          706         ORA     L       ; TEST L REG= 00H
 0A20 C2150A      707         JNZ     POLLBL  ; CONTINUE POLLING WRITE BOOT LOOP COMMAND
 0A23 E1          708 RETBT:  POP     H       ; RESTORE H-L REGS
 0A24 D1          709         POP     D       ; RESTORE D-E REGS
 0A25 C1          710         POP     B       ; RESTORE B-C REGS
 0A26 CD0008      711         CALL    INTPAR  ; CALL INTPAR, LOAD THE PARAMETRIC REGS
 0A29 DBFF        712         IN      PRTA01  ; READ STATUS REG
 0A2B C9          713         RET             ;
                  714 ;
                  715 ;
                  716 ;
                  717 $EJECT
```

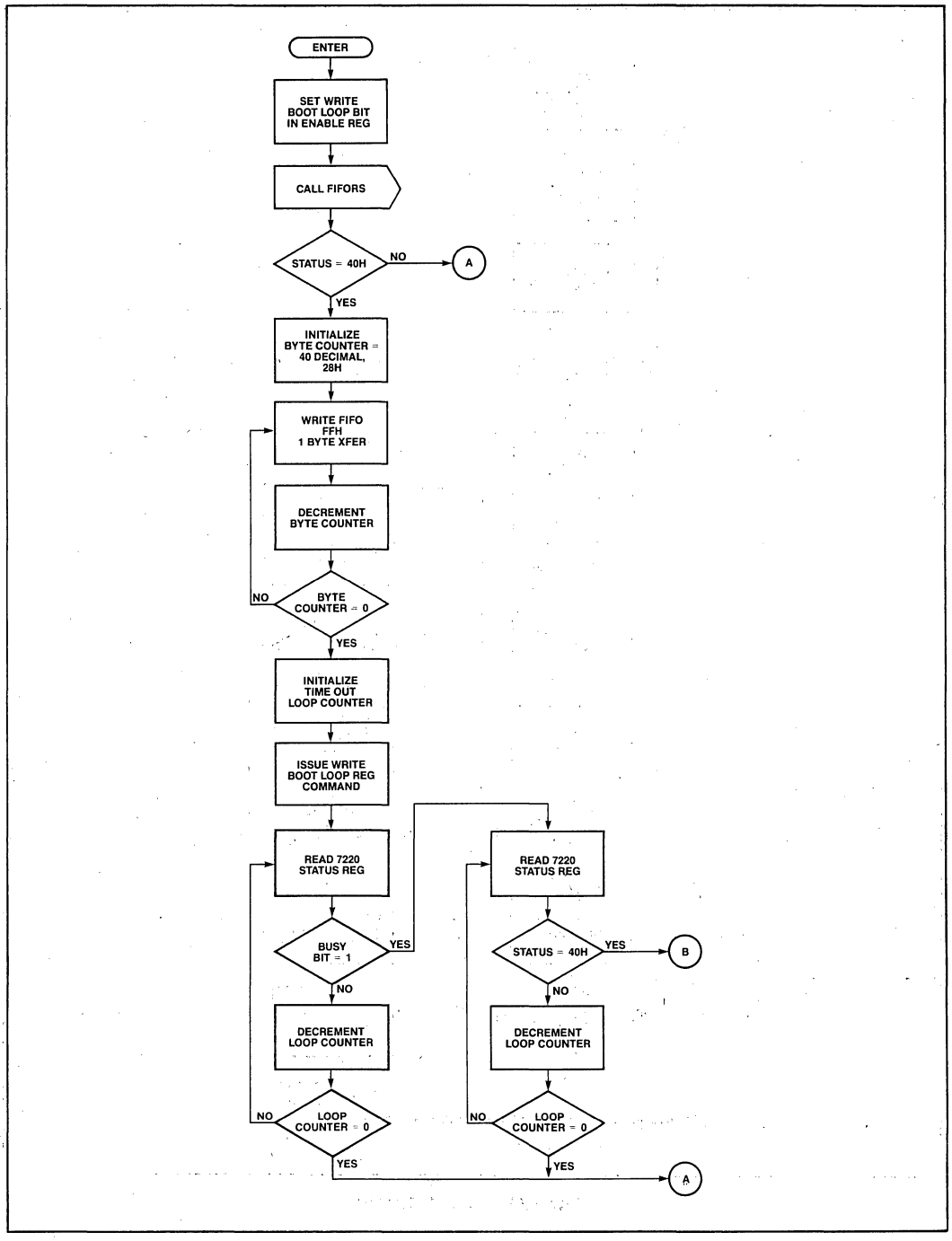


Figure 20. BOOTUP

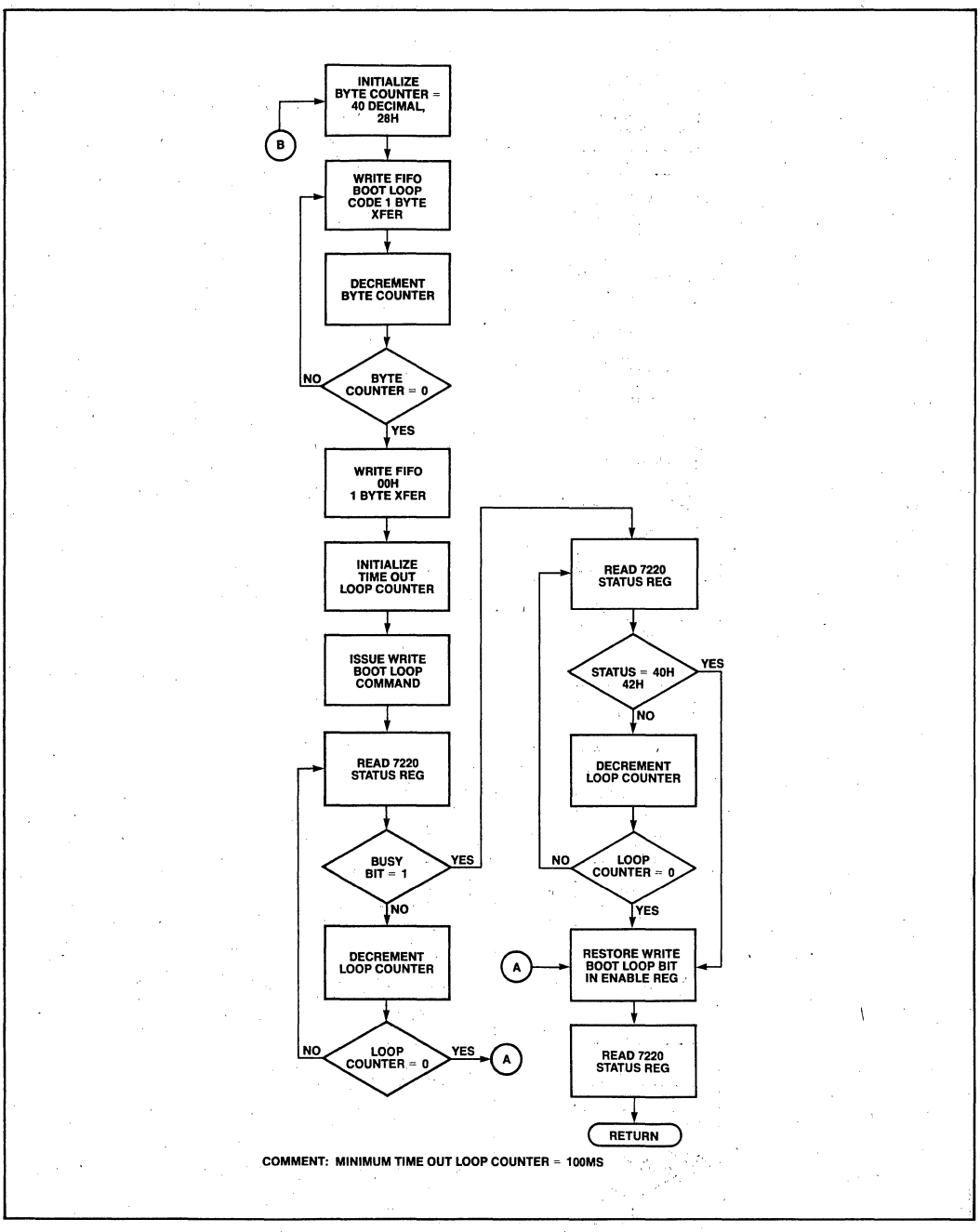


Figure 20 (Con't). BOOTUP

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72    PAGE   20


 LOC  OBJ        LINE       SOURCE STATEMENT

                  718 ;*******************************************************************************
                  719 ;
                  720 ; FUNCTION: RDBOOT
                  721 ; INPUTS:   D-E REGS, STARTING ADDRESS IN RAM
                  722 ;           BPK72 STATUS REG
                  723 ;           READ BUBBLE MEMORY BOOT LOOP
                  724 ; OUTPUTS:  COPY BUBBLE MEMORY BOOT LOOP TO RAM
                  725 ;           A REG= BPK72 STATUS REG
                  726 ; CALLS:    FIFORS
                  727 ; DESTROYS: A, F/FS
                  728 ;
                  729 ; DESCRIPTION: READ BUBBLE MEMORY BOOT LOOP
                  730 ;       THE  D-E REGS CONTAIN THE STARTING  ADDRESS  TO THE  FIRST OF 40
                  731 ;       CONTIGUOUS  MEMORY  LOCATIONS  IN  RAM  THAT WILL BE LOADED WITH
                  732 ;       A COPY OF THE BOOT LOOP CODE.  RDBOOT  RETURNS  THE VALUE OF THE
                  733 ;       BPK72  STATUS  REG TO  THE CALLING ROUTINE VIA THE 8085'S A REG.
                  734 ;       ONLY A STATUS OF 40H INDICATES A SUCCESSFUL EXECUTION OF RDBOOT.
                  735 ;
                  736         PUBLIC  RDBOOT  ; DECLARE PUBLIC FUNCTION
 0A2C C5          737 RDBOOT: PUSH    B       ; SAVE B-C REGS
 0A2D D5          738         PUSH    D       ; SAVE D-E REGS
 0A2E E5          739         PUSH    H       ; SAVE H-L REGS
 0A2F 0640        740         MVI     B,40H   ; LOAD B REG= 40H, OP-COMPLETE
 0A31 0E28        741         MVI     C,28H   ; LOAD C REG= 28H, BYTE COUNTER= 40 DECIMAL
 0A33 CD1308      742         CALL    FIFORS  ; CALL FIFO RESET
 0A36 A8          743         XRA     B       ; TEST STATUS= 40H, OP-COMPLETE
 0A37 C26A0A      744         JNZ     RETRDB  ; IF NOT ZERO, ERROR, JMP RETRDB
 0A3A 04          745         INR     B       ; B REG= 41H, OP-COMPLETE, FIFO FULL
 0A3B 21FFFF      746         LXI     H,0FFFFH; INITIALIZE TIME OUT LOOP COUNTER
 0A3E 3E1B        747         MVI     A,1BH   ; LOAD A REG= READ BOOT LOOP COMMAND
 0A40 D3FF        748         OUT     PRTA01  ; WRITE, READ BOOT LOOP COMMAND
 0A42 DBFF        749 BUSYRB: IN      PRTA01  ; READ STATUS REG
 0A44 07          750         RLC             ; TEST BUSY BIT= 1
 0A45 DA520A      751         JC      BTLPRD  ; IF BUSY= 1, POLL STATUS REG FOR 41H
 0A48 2B          752         DCX     H       ; DECREMENT TIME OUT LOOP COUNTER
 0A49 AF          753         XRA     A       ; CLEAR A REG
 0A4A B4          754         ORA     H       ; TEST H REG= 00H
 0A4B B5          755         ORA     L       ; TEST L REG= 00H
 0A4C C2420A      756         JNZ     BUSYRB  ; IF NOT ZERO, CONTINUE POLLING RDBL COMMAND
 0A4F C36A0A      757         JMP     RETRDB  ; TIME OUT ERROR, RETURN
                  758                         ; CONTINUED ON NEXT PAGE
                  759 $EJECT
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72   PAGE  21


 LOC  OBJ        LINE       SOURCE STATEMENT

 0A52 DBFF        760 BTLPRD: IN      PRTA01  ; READ STATUS REG
 0A54 A8          761         XRA     B       ; TEST STATUS= 41H, OP-COMPLETE, FIFO FULL
 0A55 CA620A      762         JZ      FIFORD  ; IF ZERO, JMP TO FIFO READ
 0A58 2B          763         DCX     H       ; DECREMENT TIME OUT LOOP COUNTER
 0A59 AF          764         XRA     A       ; CLEAR A REG
 0A5A B4          765         ORA     H       ; TEST H REG= 00H
 0A5B B5          766         ORA     L       ; TEST L REG= 00H
 0A5C CA6A0A      767         JZ      RETRDB  ; IF ZERO, TIME OUT, ERROR
 0A5F C3520A      768         JMP     BTLPRD  ; CONTINUE POLLING RDBL COMMAND
 0A62 DBFE        769 FIFORD: IN      PRTA00  ; READ FIFO DATA BUFFER
 0A64 12          770         STAX    D       ; WRITE RAM AT ADDRESS IN D REG
 0A65 13          771         INX     D       ; INCREMENT D REG TO NEXT RAM ADDRESS
 0A66 0D          772         DCR     C       ; DECREMENT BYTE COUNTER
 0A67 C2620A      773         JNZ     FIFORD  ; IF NOT ZERO, JMP FIFO READ
 0A6A DBFF        774 RETRDB: IN      PRTA01  ; READ STATUS REG
 0A6C E1          775         POP     H       ; RESTORE H-L REGS
 0A6D D1          776         POP     D       ; RESTORE D-E REGS
 0A6E C1          777         POP     B       ; RESTORE B-C REGS
 0A6F C9          778         RET             ; RETURN TO CALL
                  779 ;
                  780 $EJECT
```

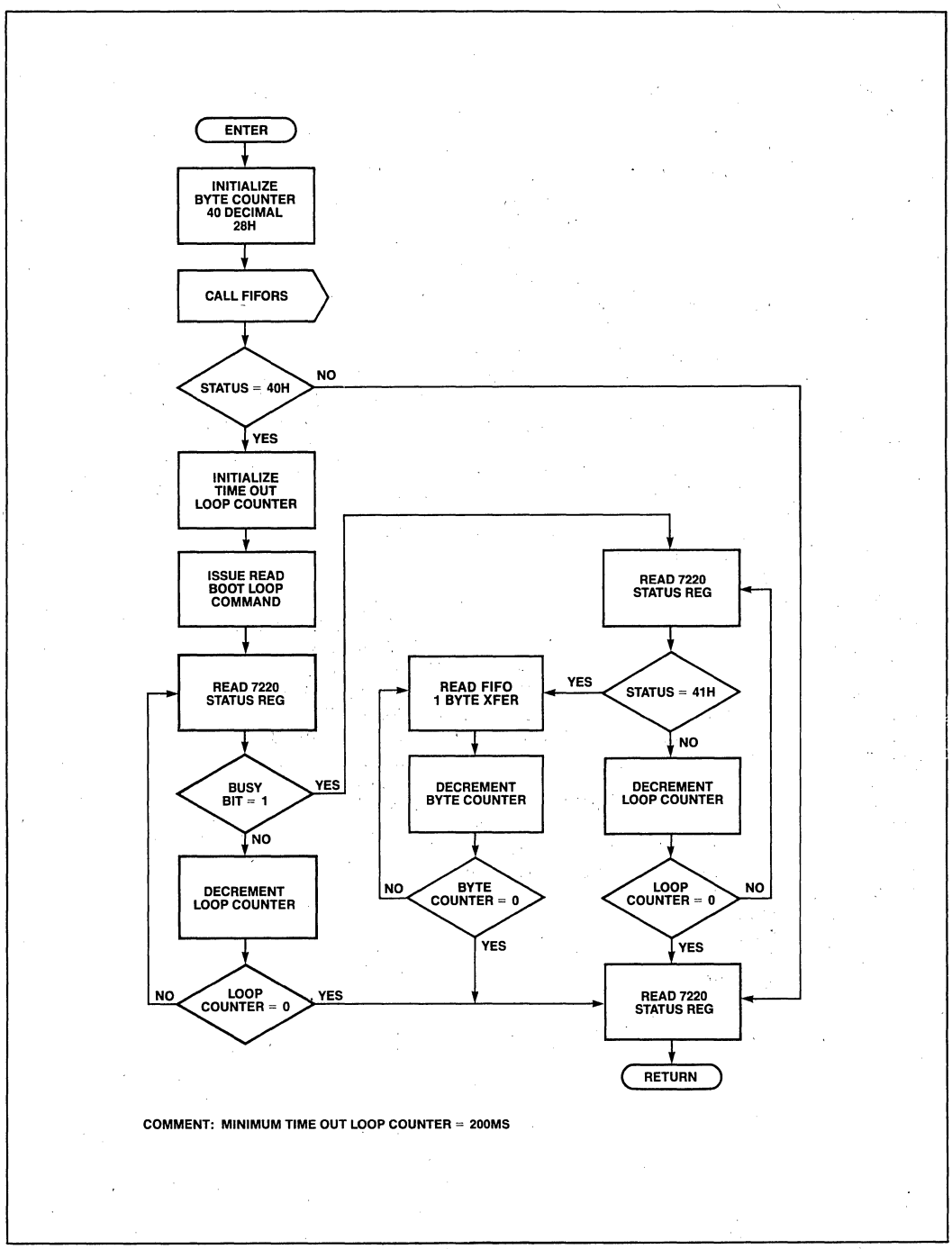


Figure 21. RDBOOT

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72    PAGE   22


 LOC  OBJ        LINE       SOURCE STATEMENT

                  781 ;*****************************************************************************
                  782 ;
                  783 ; FUNCTION: WRFIFO
                  784 ; INPUTS:   D-E REGS, STARTING ADDRESS OF DATA IN RAM
                  785 ;           BPK72 STATUS REG
                  786 ; OUTPUTS:  WRITE 40 BYTES IN THE BPK72 FIFO DATA BUFFER
                  787 ;           A REG= BPK72 STATUS REG
                  788 ; CALLS:    FIFORS
                  789 ; DESTROYS: A, F/FS
                  790 ;
                  791 ; DESCRIPTION: WRITE 7220 FIFO DATA BUFFER
                  792 ;       THE  D-E REGS PROVIDE THE ADDRESS TO THE FIRST OF 40 CONTIGUOUS
                  793 ;       BYTES IN RAM THAT CONTAIN DATA TO BE LOADED INTO THE BPK72 FIFO
                  794 ;       DATA BUFFER.  WRFIFO WILL TRANSFER THE DATA FROM RAM TO THE FIFO
                  795 ;       DATA BUFFER.  WRFIFO RETURNS  THE VALUE OF THE BPK72 STATUS REG
                  796 ;       TO THE CALLING ROUTINE VIA THE 8085'S A REG.  ONLY  A STATUS OF
                  797 ;       41H OR 43H INDICATES A SUCCESSFUL EXECUTION OF WRFIFO.
                  798 ;
                  799         PUBLIC  WRFIFO  ; DECLARE PUBLIC FUNCTION
 0A70 C5          800 WRFIFO: PUSH    B       ; SAVE B-C REGS
 0A71 D5          801         PUSH    D       ; SAVE D-E REGS
 0A72 0640        802         MVI     B,40H   ; LOAD B REG= 40H, OP-COMPLETE
 0A74 0E28        803         MVI     C,28H   ; LOAD C REG= 28H, INITIALIZE LOOP COUNTER
 0A76 CD1308      804         CALL    FIFORS  ; CALL FIFORS, WRITE FIFO RESET COMMAND
 0A79 A8          805         XRA     B       ; TEST FOR STATUS REG= 40H, OP-COMPLETE
 0A7A C2850A      806         JNZ     RETWF   ; IF NOT ZERO, FIFO ERROR, JMP RETWF
 0A7D 1A          807 INFIFO: LDAX    D       ; LOAD A REG FROM D-E REG ADDRESS
 0A7E D3FE        808         OUT     PRTA00  ; WRITE A REG TO 7220 FIFO DATA BUFFER
 0A80 13          809         INX     D       ; INCREMENT D-E REGS TO NEXT ADDRESS IN RAM
 0A81 0D          810         DCR     C       ; DECREMENT LOOP COUNTER
 0A82 C27D0A      811         JNZ     INFIFO  ; IF LOOP COUNTER NOT ZERO, JMP INFIFO
 0A85 D1          812 RETWF:  POP     D       ; RESTORE D-E REGS
 0A86 C1          813         POP     B       ; RESTORE B-C REGS
 0A87 DBFF        814         IN      PRTA01  ; READ STATUS REG
 0A89 C9          815         RET             ; RETURN TO CALL
                  816 ;
                  817 ;
                  818 ;
                  819 $EJECT
```

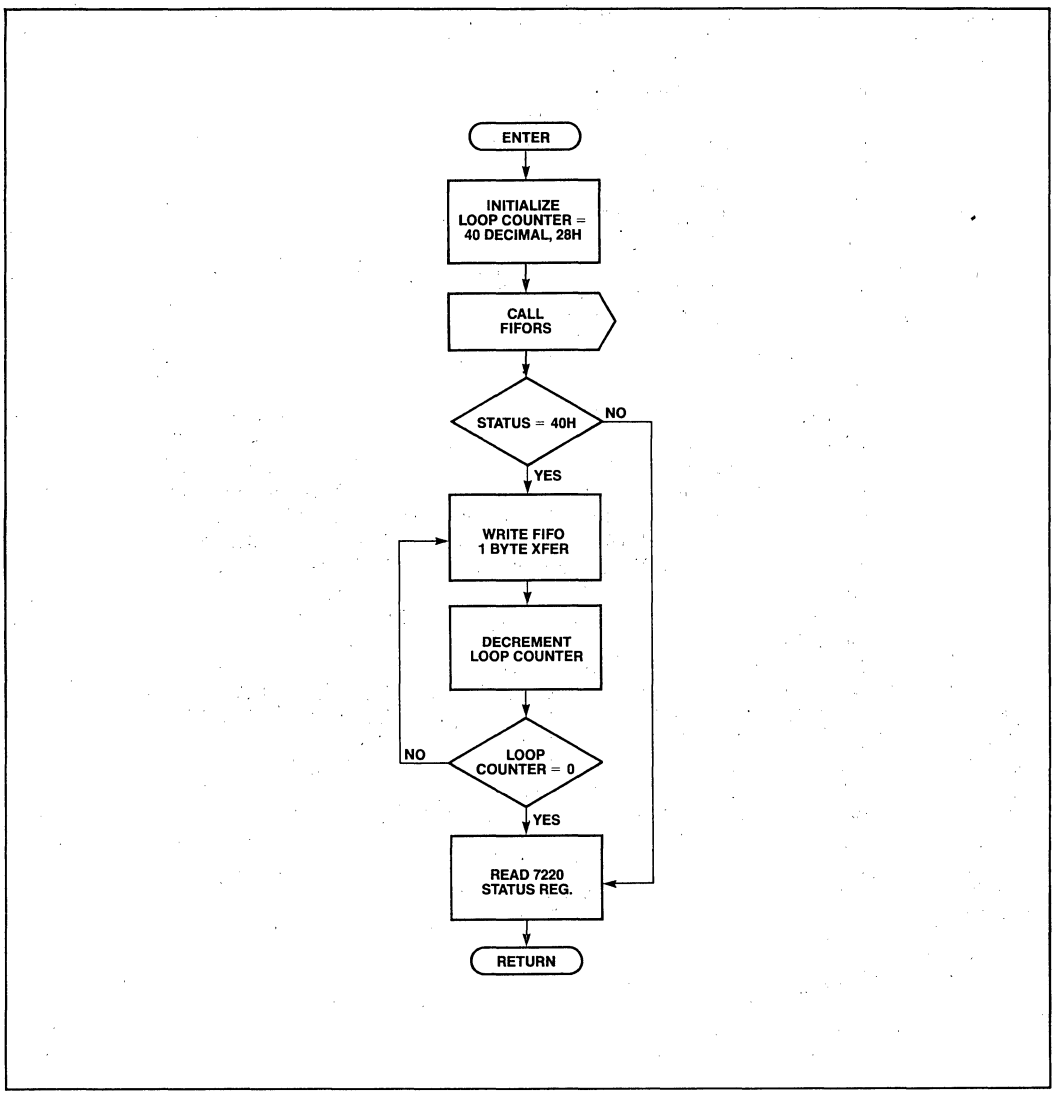


Figure 22. WRFIFO

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72    PAGE   23


 LOC  OBJ        LINE       SOURCE STATEMENT

                  820 ;*************************************************************************
                  821 ;
                  822 ; FUNCTION: RDFIFO
                  823 ; INPUTS:   D-E REGS STARTING ADDRESS IN RAM
                  824 ;           BPK72 STATUS REG
                  825 ;           READ 40 BYTES OF DATA FROM BPK72 FIFO DATA BUFFER
                  826 ; OUTPUTS:  TRANSFER FIFO DATA BUFFER TO RAM
                  827 ;           A REG= BPK72 STATUS REG
                  828 ; CALLS:    NONE
                  829 ; DESTROYS: A, F/FS
                  830 ;
                  831 ; DESCRIPTION: READ 7220 FIFO DATA BUFFER
                  832 ;       THE  D-E  REGS  CONTAIN THE ADDRESS TO THE  FIRST OF 40 CONTIGUOUS
                  833 ;       BYTES IN RAM THAT  WILL BE LOADED  WITH THE CONTENTS OF THE  BPK72
                  834 ;       FIFO DATA BUFFER. RDFIFO WILL TRANSFER THE DATA FROM THE FIFO DATA
                  835 ;       BUFFER TO RAM.  RDFIFO RETURNS THE VALUE OF  THE BPK72 STATUS  REG
                  836 ;       TO THE CALLING ROUTINE VIA THE 8085'S A REG.  ONLY A STATUS OF 40H
                  837 ;       OR 42H INDICATES A SUCCESSFUL EXECUTION OF RDFIFO.
                  838 ;
                  839         PUBLIC  RDFIFO  ; DECLARE PUBLIC FUNCTION
 0A8A C5          840 RDFIFO: PUSH    B       ; SAVE B-C REGS
 0A8B D5          841         PUSH    D       ; SAVE D-E REGS
 0A8C 0E28        842         MVI     C,28H   ; LOAD C REG= 28H, INITIALIZE LOOP COUNTER
 0A8E DBFE        843 OUTFIF: IN      PRTA00  ; LOAD A REG WITH ONE BYTE FROM FIFO DATA BUFFER
 0A90 12          844         STAX    D       ; LOAD A REG IN D-E REG ADDRESS
 0A91 13          845         INX     D       ; INCREMENT D-E REGS TO NEXT ADDRESS
 0A92 0D          846         DCR     C       ; DECREMENT LOOP COUNTER
 0A93 C28E0A      847         JNZ     OUTFIF  ; IF LOOP COUNTER NOT ZERO, JMP OUTFIF
 0A96 D1          848         POP     D       ; RESTORE D-E REGS
 0A97 C1          849         POP     B       ; RESTORE B-C REGS
 0A98 DBFF        850         IN      PRTA01  ; READ STATUS REG
 0A9A C9          851         RET             ; RETURN TO CALL
                  852 ;
                  853 ;
                  854 $EJECT
```

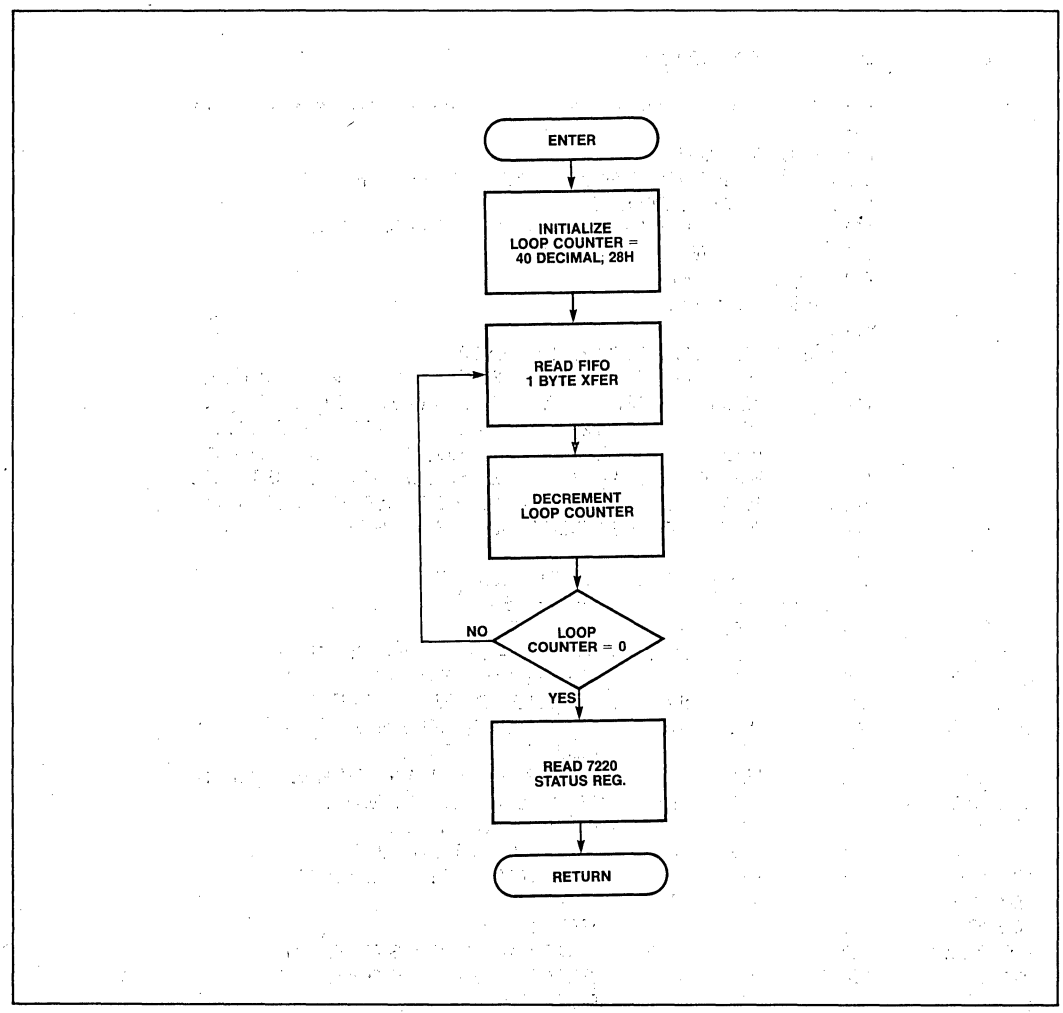


Figure 23. RDFIFO

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72    PAGE   24


 LOC  OBJ        LINE       SOURCE STATEMENT

                  855 ;*****************************************************************
                  856 ;
                  857 ; FUNCTION: WRBLRS
                  858 ; INPUTS:   D-E REGS, STARTING ADDRESS OF DATA IN RAM
                  859 ;           BPK72 STATUS REG
                  860 ; OUTPUTS:  WRITE BUBBLE MEMORY BOOT LOOP REGISTERS COMMAND
                  861 ;           A REG= BPK72 STATUS REG
                  862 ; CALLS:    WRFIFO
                  863 ; DESTROYS: A, F/FS
                  864 ;
                  865 ; DESCRIPTION: WRITE 7242 BOOT LOOP REGISTERS
                  866 ;       THE D-E REGS PROVIDE THE ADDRESS TO THE FIRST OF 40 CONTIGUOUS
                  867 ;       MEMORY  LOCATIONS  IN RAM THAT  CONTAIN DATA TO BE LOADED INTO
                  868 ;       THE 7242,  FORMATTER  SENSE  AMPLIFIER,  BOOT  LOOP REGISTERS.
                  869 ;       WRBLRS WILL  TRANSFER  THE DATA  FROM  RAM  TO  THE  BOOT LOOP
                  870 ;       REGISTERS.  WRBLRS  RETURNS THE VALUE OF THE BPK72  STATUS REG
                  871 ;       TO THE CALLING  ROUTINE VIA THE 8085'S A REG.  ONLY A STATUS OF
                  872 ;       40H INDICATES A SUCCESSFUL EXECUTION OF WRBLRS.
                  873 ;
                  874         PUBLIC  WRBLRS  ; DECLARE PUBLIC FUNCTION
 0A9B C5          875 WRBLRS: PUSH    B       ; SAVE B-C REGS
 0A9C E5          876         PUSH    H       ; SAVE H-L REGS
 0A9D 0641        877         MVI     B,41H   ; LOAD B REG= 41H, OP-COMPLETE, FIFO FULL
 0A9F 0EFD        878         MVI     C,0FDH  ; MASK, MASK OUT PARITY BIT
 0AA1 21FFFF      879         LXI     H,0FFFFH; INITIALIZE TIME OUT LOOP COUNTER
 0AA4 CD700A      880         CALL    WRFIFO  ; CALL WRITE FIFO DATA BUFFER
 0AA7 A1          881         ANA     C       ; RESET BIT 1, PARITY BIT
 0AA8 A8          882         XRA     B       ; TEST STATUS= 41H OR 43H, OP-COMPLETE, FIFO FULL
 0AA9 C2CE0A      883         JNZ     RETWBL  ; IF NOT ZERO, ERROR, JMP RETWBL
 0AAC 05          884         DCR     B       ; B REG= 40H, OP-COMPLETE
 0AAD 3E16        885         MVI     A,16H   ; LOAD A REG= WRITE BOOT LOOP REG COMMAND
 0AAF D3FF        886         OUT     PRTA01  ; WRITE, WRITE BOOT LOOP REG COMMAND
 0AB1 DBFF        887 BSYWBL: IN      PRTA01  ; READ STATUS REG
 0AB3 07          888         RLC             ; TEST BUSY BIT= 1
 0AB4 DAC10A      889         JC      POLWBL  ; IF BUSY= 1, POLL STATUS REG FOR 40H
 0AB7 2B          890         DCX     H       ; DECREMENT TIME OUT LOOP COUNTER
 0AB8 AF          891         XRA     A       ; CLEAR A REG
 0AB9 B4          892         ORA     H       ; TEST H REG= 00H
 0ABA B5          893         ORA     L       ; TEST L REG= 00H
 0ABB C2B10A      894         JNZ     BSYWBL  ; IF NOT ZERO, CONTINUE POLLING WRBLR COMMAND
 0ABE C3CE0A      895         JMP     RETWBL  ; TIME OUT ERROR, RETURN
 0AC1 DBFF        896 POLWBL: IN      PRTA01  ; READ STATUS REG
 0AC3 A8          897         XRA     B       ; TEST STATUS REG= 40H, OP-COMPLETE
 0AC4 CACE0A      898         JZ      RETWBL  ; IF ZERO, OP-COMPLETE, JMP RETWBL
 0AC7 2B          899         DCX     H       ; DECREMENT TIME OUT LOOP COUNTER
 0AC8 AF          900         XRA     A       ; CLEAR A REG
 0AC9 B4          901         ORA     H       ; TEST H REG= 00H
 0ACA B5          902         ORA     L       ; TEST L REG= 00H
 0ACB C2C10A      903         JNZ     POLWBL  ; IF NOT ZERO, CONTINUE POLLING WRBLR COMMAND
 0ACE E1          904 RETWBL: POP     H       ; RESTORE H-L REGS
 0ACF C1          905         POP     B       ; RESTORE B-C REGS
 0AD0 DBFF        906         IN      PRTA01  ; READ STATUS REG
 0AD2 C9          907         RET             ; RETURN TO CALL
                  908 $EJECT
```

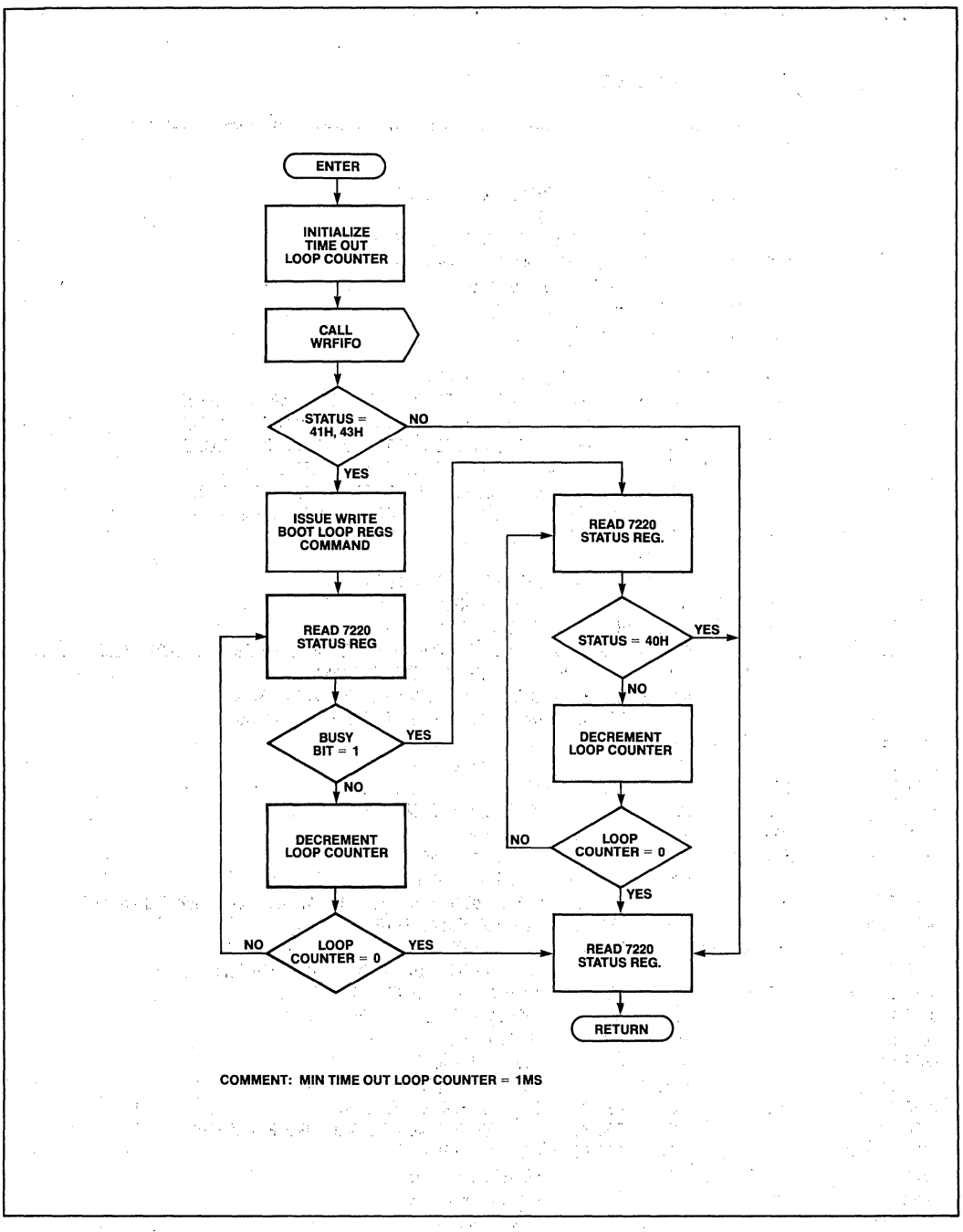


Figure 24. WRBLRS

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72    PAGE   25


 LOC  OBJ        LINE       SOURCE STATEMENT

                  909 ;*************************************************************************
                  910 ;
                  911 ; FUNCTION: RDBLRS
                  912 ; INPUTS:   D-E REGS, STARTING ADDRESS IN RAM
                  913 ;           BPK72 STATUS REG
                  914 ;           READ DATA FROM 7242 BOOT LOOP REGISTERS
                  915 ; OUTPUTS:  TRANSFER BOOT LOOP REGISTER DATA TO RAM
                  916 ;           A REG= BPK72 STATUS REG
                  917 ; CALLS:    RDFIFO
                  918 ; DESTROYS: A, F/FS
                  919 ;
                  920 ; DESCRIPTION: READ 7242 BOOT LOOP REGISTERS
                  921 ;       THE D-E REGS CONTAIN THE ADDRESS TO THE FIRST OF 40 CONTIGUOUS
                  922 ;       MEMORY LOCATIONS IN RAM TO BE  LOADED WITH THE CONTENTS OF THE
                  923 ;       7242, FORMATTER  SENSE AMPLIFIER, BOOT LOOP REGISTERS.  RDBLRS
                  924 ;       WILL COPY  THE CONTENTS  OF THE  BOOT  LOOP  REGISTERS TO RAM.
                  925 ;       RDBLRS  RETURNS THE  VALUE OF  THE  BPK72  STATUS  REG  TO THE
                  926 ;       CALLING ROUTINE  VIA THE 8085'S  A REG.  ONLY A  STATUS OF 40H
                  927 ;       INDICATES A SUCCESSFUL EXECUTION OF RDBLRS.
                  928 ;
                  929         PUBLIC  RDBLRS  ; DECLARE PUBLIC FUNCTION
 0AD3 C5          930 RDBLRS: PUSH    B       ; SAVE B-C REGS
 0AD4 E5          931         PUSH    H       ; SAVE H-L REGS
 0AD5 06C1        932         MVI     B,0C1H  ; LOAD B REG= C1H, OP-COMPLETE, FIFO FULL >22 BYTES (BUSY BIT=1)
 0AD7 21FFFF      933         LXI     H,0FFFFH; INTIALIZE TIME OUT LOOP COUNTER
 0ADA 3E15        934         MVI     A,15H   ; LOAD A REG= READ BOOT LOOP REGS COMMAND
 0ADC D3FF        935         OUT     PRTA01  ; WRITE THE READ BOOT LOOP REGS COMMAND
 0ADE DBFF        936 BSYRBL: IN      PRTA01  ; READ STATUS REG
 0AE0 07          937         RLC             ; TEST BUSY BIT= 1
 0AE1 DAEE0A      938         JC      POLRBL  ; IF BUSY= 1, POLL STATUS REG FOR C1H
 0AE4 2B          939         DCX     H       ; DECREMENT TIME OUT LOOP COUNTER
 0AE5 AF          940         XRA     A       ; CLEAR A REG
 0AE6 B4          941         ORA     H       ; TEST H REG= 00H
 0AE7 B5          942         ORA     L       ; TEST L REG= 00H
 0AE8 C2DE0A      943         JNZ     BSYRBL  ; IF NOT ZERO, CONTINUE POLLING READ BOOT LOOP REG COMMAND
 0AEB C3010B      944         JMP     RETRBL  ; TIME OUT ERROR, RETURN
 0AEE DBFF        945 POLRBL: IN      PRTA01  ; READ STATUS REG
 0AF0 A8          946         XRA     B       ; TEST STATUS= C1H, OP-COMPLETE, FIFO FULL
 0AF1 CAFE0A      947         JZ      CALLRD  ; IF ZERO, OP-COMPLETE, JMP CALLRD
 0AF4 2B          948         DCX     H       ; DECREMENT TIME OUT LOOP COUNTER
 0AF5 AF          949         XRA     A       ; CLEAR A REG
 0AF6 B4          950         ORA     H       ; TEST H REG= 00H
 0AF7 B5          951         ORA     L       ; TEST L REG= 00H
 0AF8 CA010B      952         JZ      RETRBL  ; IF ZERO, ERROR, JMP RETRBL
 0AFB C3EE0A      953         JMP     POLRBL  ; CONTINUE POLLING READ BOOT LOOP REG COMMAND
 0AFE CD8A0A      954 CALLRD: CALL    RDFIFO  ; CALL READ FIFO
 0B01 E1          955 RETRBL: POP     H       ; RESTORE H-L REGS
 0B02 C1          956         POP     B       ; RESTORE B-C REGS
 0B03 DBFF        957         IN      PRTA01  ; READ STATUS REG
 0B05 C9          958         RET             ; RETURN TO CALL
                  959 $EJECT
```

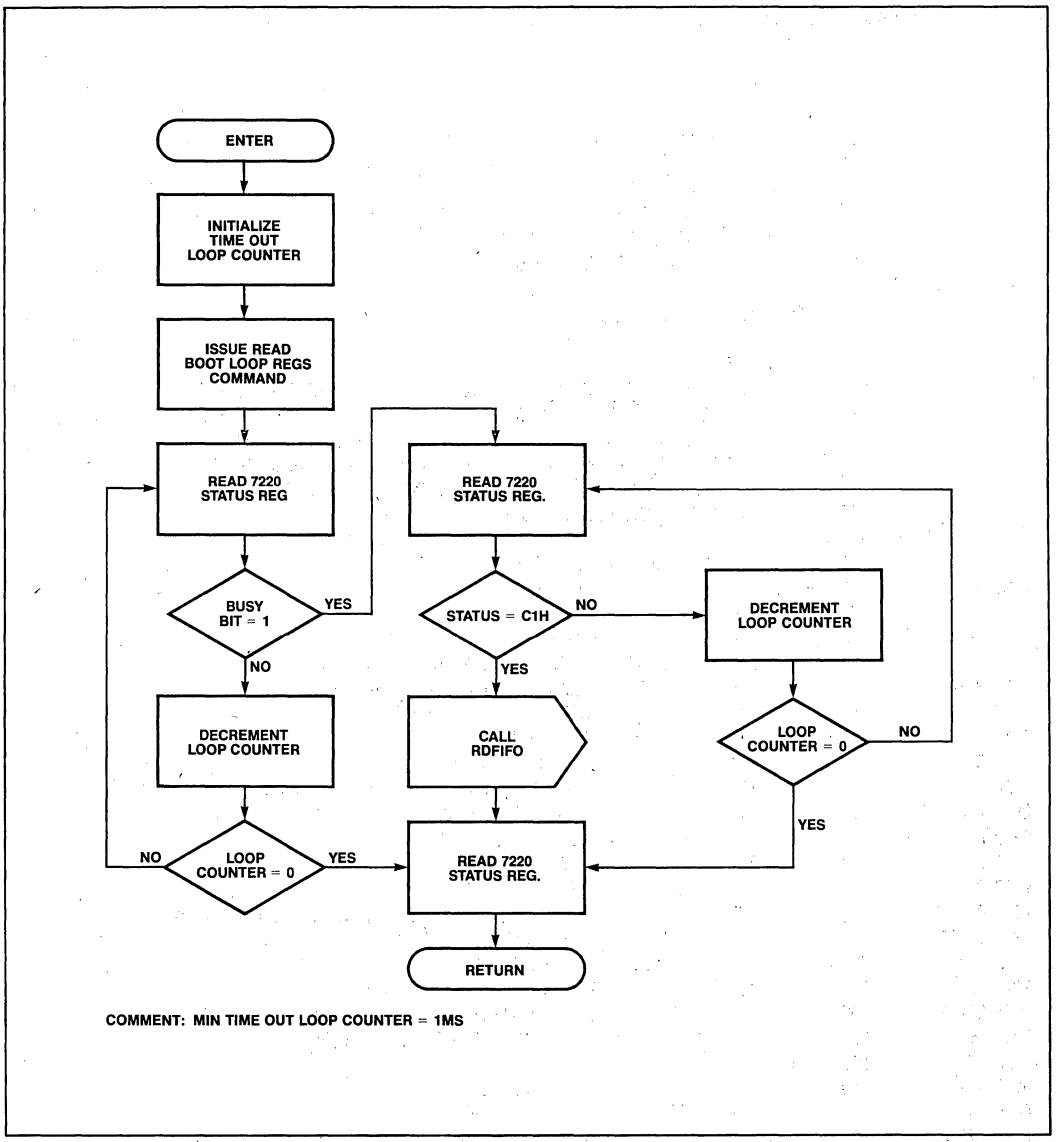
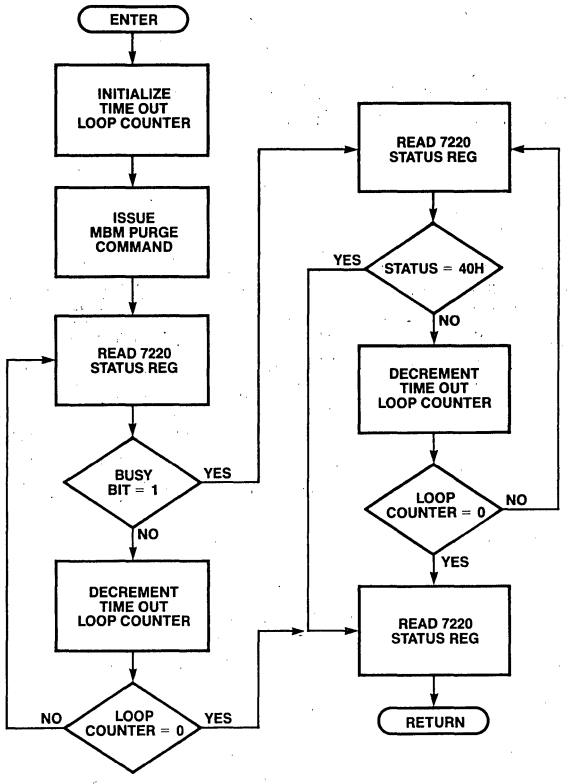


Figure 25. RDBLRS

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72    PAGE   26


 LOC  OBJ        LINE       SOURCE STATEMENT

                  960 ;*************************************************************************
                  961 ;
                  962 ; FUNCTION: MBMPRG
                  963 ; INPUTS:   BPK72 STATUS REG
                  964 ; OUTPUTS:  ISSUE MBM PURGE COMMAND
                  965 ;           A REG= BPK72 STATUS REG
                  966 ; CALLS:    NONE
                  967 ; DESTROYS: A, F/FS
                  968 ;
                  969 ; DESCRIPTION: MBM PURGE COMMAND
                  970 ;       AN MBM PURGE COMMAND IS ISSUED TO THE BPK72.  AFTER ISSUING THE
                  971 ;       COMMAND,  THE BPK72  STATUS REG IS POLLED UNTIL AN OP-COMPLETE
                  972 ;       40H,  HAS BEEN  READ OR THE  TIME OUT LOOP  COUNTER DECREMENTS
                  973 ;       TO ZERO.  MBMPRG  RETURNS THE VALUE OF THE BPK72 STATUS REG TO
                  974 ;       THE CALLING ROUTINE VIA THE 8085'S A REG.  ONLY A STATUS OF 40H
                  975 ;       INDICATES A SUCCESSFUL EXECUTION OF MBMPRG.
                  976 ;
                  977         PUBLIC  MBMPRG  ; DECLARE PUBLIC FUNCTION
 0B06 D5          978 MBMPRG: PUSH    D       ; SAVE D-E REGS
 0B07 C5          979         PUSH    B       ; SAVE B-C REGS
 0B08 0640        980         MVI     B,40H   ; LOAD B REG= 40H, OP-COMPLETE
 0B0A 11FFFF      981         LXI     D,0FFFFH; INITIALIZE TIME OUT LOOP COUNTER
 0B0D 3E1E        982         MVI     A,1EH   ; LOAD A REG= MBM PURGE COMMAND
 0B0F D3FF        983         OUT     PRTA01  ; WRITE MBM PURGE COMMAND
 0B11 DBFF        984 BSYMBM: IN      PRTA01  ; READ STATUS REG
 0B13 07          985         RLC             ; TEST BUSY BIT= 1
 0B14 DA210B      986         JC      POLMBM  ; IF BUSY= 1, POLL STATUS REG FOR 40H
 0B17 1B          987         DCX     D       ; DECREMENT TIME OUT LOOP COUNTER
 0B18 AF          988         XRA     A       ; CLEAR A REG
 0B19 B2          989         ORA     D       ; TEST D REG= 00H
 0B1A B3          990         ORA     E       ; TEST E REG= 00H
 0B1B C2110B      991         JNZ     BSYMBM  ; IF NOT ZERO, CONTINUE POLLING THE MBMPRG COMMAND
 0B1E C32E0B      992         JMP     RETMBM  ; TIME OUT ERROR, RETURN
 0B21 DBFF        993 POLMBM: IN      PRTA01  ; READ STATUS REG
 0B23 A8          994         XRA     B       ; TEST STATUS= 40H, OP-COMPLETE
 0B24 CA2E0B      995         JZ      RETMBM  ; IF OP-COMPLETE, JMP RETMBM
 0B27 1B          996         DCX     D       ; DECREMENT TIME OUT LOOP COUNTER
 0B28 AF          997         XRA     A       ; CLEAR A REG
 0B29 B2          998         ORA     D       ; TEST D REG= 00H
 0B2A B3          999         ORA     E       ; TEST E REG= 00H
 0B2B C2210B     1000         JNZ     POLMBM  ; IF NOT ZERO, CONTINUE POLLING MBM PURGE COMMAND
 0B2E C1         1001 RETMBM: POP     B       ; RESTORE B-C REGS
 0B2F D1         1002         POP     D       ; RESTORE D-E REGS
 0B30 DBFF       1003         IN      PRTA01  ; READ STATUS REG
 0B32 C9         1004         RET             ; RETURN TO CALL
                 1005 $EJECT
```



COMMENT: MINIMUM TIME OUT LOOP COUNTER = 150μS.

Figure 26. MBMPRG

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        BPK72    PAGE   27


 LOC  OBJ        LINE        SOURCE STATEMENT

                 1006 ;
                 1007        END


PUBLIC SYMBOLS
ABORT  A 08DE    BOOTUP A 099C    FIFORS A 0813    INBUBL A 0961    MBMPRG A 0B06    RDBLRS A 0AD3    RDBOOT A 0A2C
RDBUBL A 0936    RDFIFO A 0A8A    WRBLRS A 0A9B    WRBUBL A 090B    WRFIFO A 0A70

EXTERNAL SYMBOLS


USER SYMBOLS
ABORT  A 08DE    ALLFFS A 09BB    BLCODE A 09F1    BOOTUP A 099C    BSYMBM A 0B11    BSYRBL A 0ADE    BSYWBL A 0AB1
BTLPRD A 0A52    BUSYA  A 08E9    BUSYB  A 09C8    BUSYBL A 0A05    BUSYFR A 081E    BUSYIN A 097A    BUSYRB A 0A42
BUSYRD A 08AD    BUSYWR A 0873    BYTCNT A 0840    CALLRD A 0AFE    CONT   A 09E8    DONE   A 0867    FIFORD A 0A62
FIFORS A 0813    FINSHR A 08DB    FINSHW A 08A1    INBUBL A 0961    INFIFO A 0A7D    INTPAR A 0800    LOAD   A 0808
LOOPRD A 094F    LOOPWR A 0924    MBMPRG A 0B06    MULT   A 0852    MULT1  A 0862    MULTO  A 0856    OUTFIF A 0A8E
POLLA  A 08F9    POLLBL A 0A15    POLLBR A 09D8    POLLFR A 082E    POLLIN A 098A    POLLRD A 08BA    POLLWR A 0880
POLMBM A 0B21    POLRBL A 0AEE    POLWBL A 0AC1    PRTA00 A 00FE    PRTA01 A 00FF    RDBLRS A 0AD3    RDBOOT A 0A2C
RDBUBL A 0936    RDFIFO A 0A8A    READ   A 08A4    RETA   A 0906    RETBT  A 0A23    RETFR  A 083B    RETIN  A 0997
RETMBM A 0B2E    RETRBL A 0B01    RETRD  A 095C    RETRDB A 0A6A    RETWBL A 0ACE    RETWF  A 0A85    RETWR  A 0931
RFIFO  A 08D0    WFIFO  A 0896    WRBLRS A 0A9B    WRBUBL A 090B    WRFIFO A 0A70    WRITE  A 086A

ASSEMBLY COMPLETE,  NO ERRORS
```

APPENDIX B
SERVICE INFORMATION

SERVICE INFORMATION

Typically, a Bubble Memory System will never require any special service throughout its useful life. The sequence of program flow presented in Appendix B is not required for normal read/write operation. However, power supply failure, socket contact problems, or component failures may inadvertently produce a BPK 72 system failure.

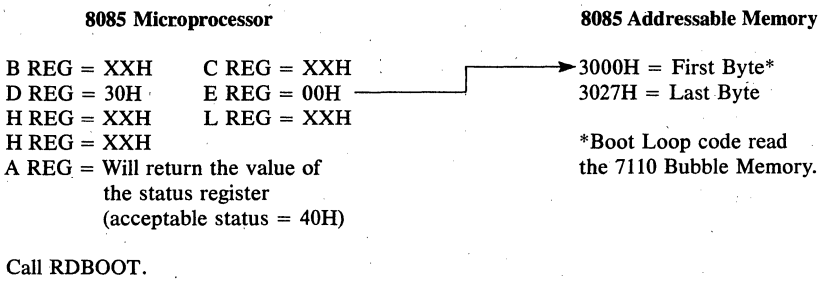
Note: Power supply failure is defined as any violation of the power supply specifications listed in the section titled, "Power Supply Requirements."

A figure titled, "BPK 72 Failure Recovery" is included in Appendix C to illustrate the sequence of events necessary to remedy a Bubble Memory System failure. The flowchart is intended as a guide for handling a Bubble Memory System failure. A system failure is defined as continued attempts that fail to read and write data correctly. Upon detection of a BPK 72 system failure, the first course of action is to verify the existence of the seeds within the 7110 Bubble Memory module. Four replicating Bubble Memory generators reside in the 7110. Each generator requires one seed from which all other bubbles are created. Under extreme circumstances such as power supply failure, one or all of the seeds can be destroyed making it impossible to write data into the 7110's storage loops. The "BPK 72 Failure Recovery" flowchart requests a call to the "seed verification procedure." The "seed verification procedure" should be followed closely to determine if any of the seeds are missing.

In the unlikely event that some or all of the seeds are lost, the "BPK 72 Failure Recovery" figure instructs the reader to perform the "procedure to reseed a 7110 Bubble Memory." The seed replacement procedure will create a seed in each of the four generators. After completing the seed replacement procedure, the "seed verification procedure" should be performed again to confirm that all four seeds are present in the 7110.

The next step in diagnosing a BPK 72 system failure is to verify the accuracy of the boot loop code within the 7110. The boot loop is a map containing information about the active and inactive storage loops. The 7110 is designed with a 15% storage loop redundancy to improve the product yield during manufacture. A diagnostic subroutine named RDBOOT can be called to read the boot loop from the 7110. It is the responsibility of the calling routine to verify that the boot loop code read from the 7110 matches byte for byte with the code found on the label attached to the case of the Bubble Memory module.

The following is an example of how to use the read Bubble Memory boot loop subroutine, RDBOOT:

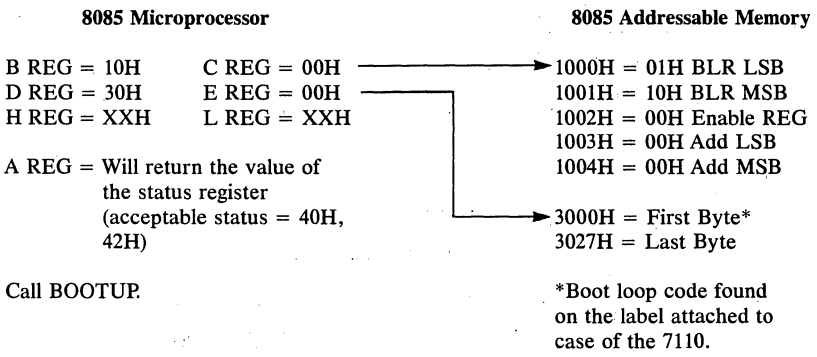


Call RDBOOT.

Additional detail regarding the use of the read Bubble Memory boot loop subroutine, RDBOOT, may be found in the software listing presented in Appendix A.

If the boot loop is incorrect, a subroutine called BOOTUP is provided for writing the boot loop into the 7110.

The following is an example of how to use BOOTUP to write the boot loop code into the 7110:



Additional detail regarding the use of the write Bubble Memory boot loop subroutine, BOOTUP, may also be found in the software listing presented in Appendix A.

After the seeds and boot loop have been examined and replaced as necessary, the remaining step is to call the initialization subroutine, INBUBL. See the section titled, "Initalizing the Bubble" for a description of how to call the initialization subroutine. If the initialization subroutine returns a status of 40H, the BPK 72 is ready to be put back into service.

Contact the local Intel field sales office in the unlikely event that the BPK 72 system failure guidelines do not eliminate the problem.

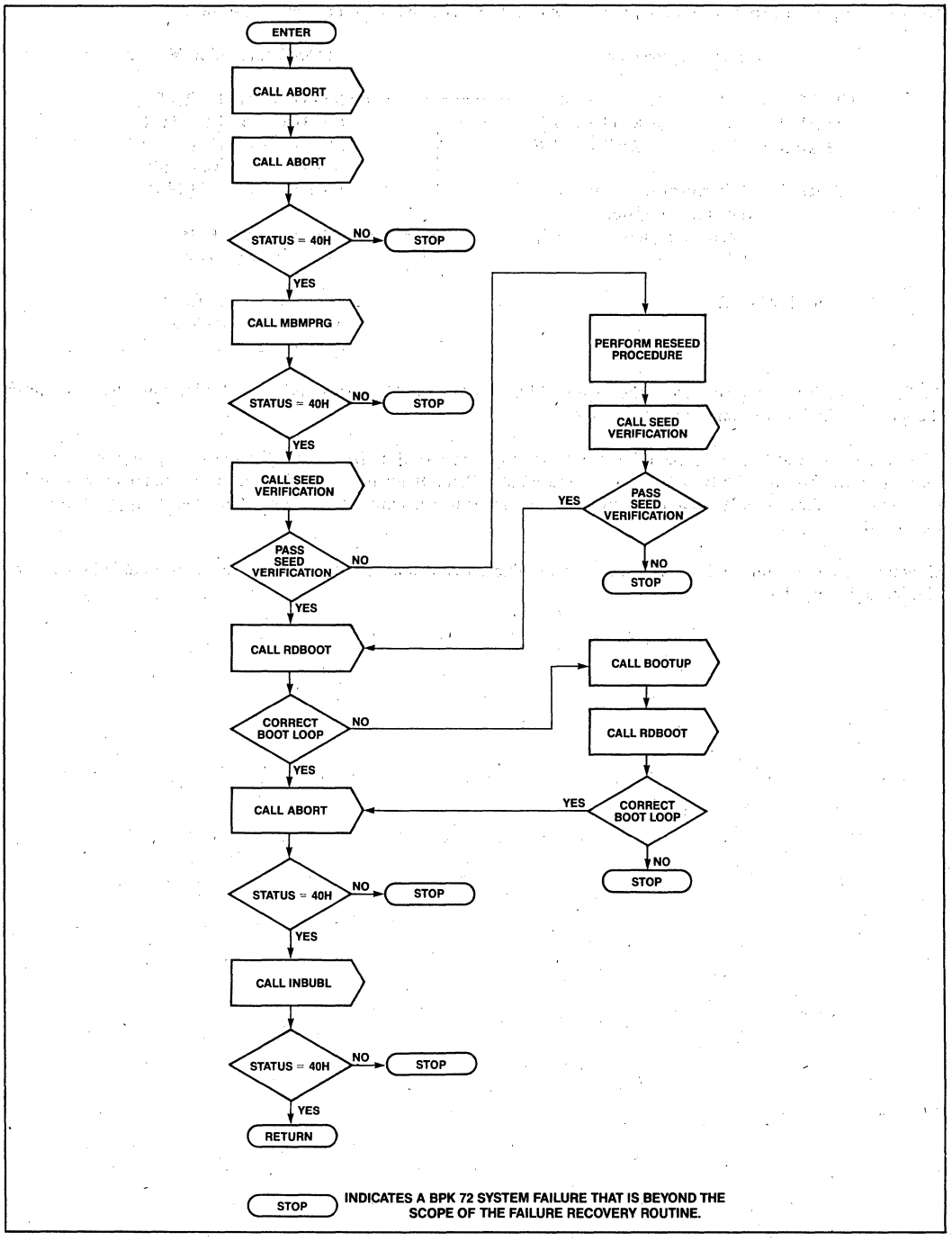
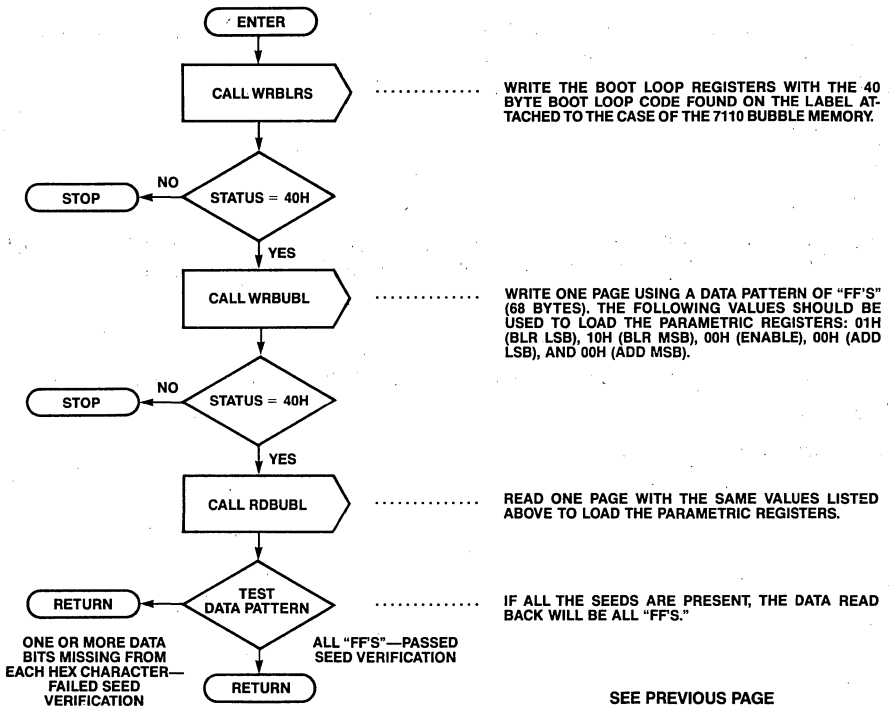


Figure 28. BPK 72 Failure Recovery



If one or more seeds are missing, the data read back will be a pattern with one or more bits missing from each hex character. One example of several possible patterns is shown below. Each pattern will typically contain a dominant pair of hex characters (i.e., "88's" or "AA's"). In any case, if seeds are missing no "FF's" will be read using the subroutine, RDBUBL.

88	88	88	88	88	88	88	88	88	00	08	88	88	88	88	88
88	88	88	88	88	08	80	88	88	88	88	88	88	88	88	88
88	A8	88	80	08	88	88	88	88	80	88	88	A8	88	8A	88
A8	88	8A	88	88	88	88	A8	88	AA	88	88	88	8A	88	88

Do not attempt to use the seed verification procedure without first performing the program sequence described in Figure 28, "BPK 72 Failure Recovery."

Figure 29. Seed Verification Procedure

PROCEDURE TO RESEED A 7110 BUBBLE MEMORY

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. Apply power to the circuit.
5. Call ABORT.
6. Call MBMPRG.
7. Call WRBUBL (1 page transfer, any location, data pattern is not important). Parametric register values; 01H (BLR LSB), 10H (BLR MSB), 00H (ENABLE), 00H (add LSB), and 00H (add MSB).
8. Remove power from circuit.
9. Remove the seed module from the 7230 socket.
10. Remove the 7230 from the seed module and reinstall the 7230 in its socket on the IMB-72 board.
11. Apply power to the circuit.
12. Reseed procedure is now complete.

intel

APPLICATION NOTE

AP-157

October 1983

Software Design and Implementation Details for Bubble Memory Systems

Richard Pierce
Applications Engineer
Intel Corporation

ORDER NUMBER: 230707-002

INTRODUCTION

The 7220-1 is a single-chip LSI Bubble Memory Controller (BMC) that implements a bubble memory storage subsystem (with up to eight bubble storage units (BSUs) per BMC). Each bubble storage unit consists of five support circuits in addition to the bubble memory chip and provides one megabit (128 kbytes) of non-volatile read/write memory. This application note examines the programmatic interface to the 7220-1 BMC and how communications between a host processor and the bubble subsystem (i.e., 7220-1) govern all bubble system operations.

The BMC provides all the control and timing signals for the bubble and support circuits. All data synchronization and error checking is automatically performed by the bubble subsystem to ensure reliable data storage. The BMC easily interfaces to microprocessor systems and communicates via a set of high-level commands. This application note explains the operations and functions performed by these instructions and provides the basic programmatic interface descriptions and program guidelines to allow you to design and implement a program module or "driver" to control all bubble system operations. In addition, several possible software interface levels are defined. While the design guidelines presented are not targeted for integration with any particular operating system, they do, however, provide conceptual information on design requirements and serve as a foundation on which to develop a modular and flexible software driver aligned with your specific application requirements.

Product Line Overview

Intel offers a complete line of bubble memory components, development kits and assembled boards.

The BPK 72 (Bubble Memory Prototype Kit) serves primarily as a means to evaluate the potential of bubble storage. The BPK 72 comes complete with all the hardware and documentation necessary to prototype a one-megabit bubble memory system. After the kit is assembled, the designer is left with the simple task of interfacing to a host processor.

The BPK 70 one-megabit bubble storage subsystem is a fully interchangable component bubble memory system. Each kit contains all the components in a Bubble Storage Unit (BSU). A single 7220-1 Bubble Memory Controller (purchased separately) can operate up to eight BPK 70 subsystems at one time. The BPK 70 is available for the production of custom systems where high volume is required.

The iSBX™ 251 Magnetic Bubble MULTIMODULE™ board is a one-megabit bubble memory mounted on a standard dual width Intel MULTIMODULE memory expansion card. Completely assembled and tested, the iSBX-251 board is fully plug compatible with all Intel iSBC Single Board Computers that have iSBX connectors.

The iSBC® 254 Bubble Memory Board is a completely assembled Intel MULTIBUS® memory board. The board can be configured with one bubble memory (128 kbytes), two bubble memories (256 kbytes), or four bubble memories (512 kbytes).

The 7220-1 BMC acts as the interface to the host processor in each of the aforementioned products, thus simplifying system programming. The basic programming techniques discussed in this application note will provide the system programmer with a complete understanding of programming requirements and shorten the software development time.

SOFTWARE INTERFACE

Basic Driver Operation

As will become evident, a basic compromise or "tradeoff" exists between the design of your application software and the capabilities of the bubble memory controller. While you will be responsible for the development of a driver to integrate the bubble into your system, the level of driver interaction and degree of flexibility will vary according to the needs of your application. If an application program is small and simple, a basic bubble driver simply may be called from the main program. At the next level of driver sophistication, the bubble system is viewed by the application program as a logical device. At this level, the key to driver design is the mapping of the "logical bubble interface" (as viewed by the application program) into the "physical bubble interface" (as implemented by the bubble drivers). This logical-to-physical mapping serves to isolate application programs and system software from the idiosyncracies of the bubble memory controller. At the

highest level of driver sophistication, the application program treats the bubble system as a collection of named data areas or files similar to the way in which data is stored and retrieved in disk operating systems. At the file system level, an application program can ignore the mechanics of bubble storage and access and merely present a "file name" to the driver to open, read or write, and then close the desired "bubble file."

At the subsystem level (i.e., "physical bubble interface"), the bubble driver is responsible for all system interaction with the bubble and is intrinsic to the efficient and reliable operation of the bubble system. The driver accepts bubble memory commands and command execution parameters from the application program, controls and monitors command execution, and returns operational status information to the application program at command completion. To perform all of these operations, the bubble driver must support the bit/byte level of the bubble memory controller's command and status registers and the "parametric" registers that define the operating mode, system configuration, and extent of the transfer. Depending on the interface level of the application software, the driver itself may be made up of a set of subroutines that are called individually by the host to perform specific bubble system operations.

SOFTWARE-SELECTIVE CONFIGURATIONS

Before you can begin to design a software driver, you must consider all of the selective configurations available within the bubble system and which of these configurations you want to support. As will be explained, the type of data transfer (i.e., direct memory access, interrupt driven, or polled), the data transfer rate, and the selective implementation of the bubble system's error correction feature all are software controlled. The complexity of the driver design depends on the system flexibility desired. For example, a driver may be designed to support only one transfer mode and may not make use of error correction. Conversely, a more generalized driver can be designed to support various transfer modes, data rates, and levels of error correction. The ensuing paragraphs define the driver responsibilities associated with the available software configurations and will aid you in designing a driver that satisfies your specific application.

Data Organization

Probably the most important aspect of the bubble memory system interface is the organization of data. From a software viewpoint, data logically is organized into blocks of bytes called "pages." During data transfer operations, one or more of these pages are transferred between the bubble(s) and the host microprocessor. A page is the smallest increment of data that can be transferred; single bytes cannot be transferred. Conceptually, the data organization within a bubble memory is analogous to a disk system. Just as disk sector sizes are fixed when a disk is formatted, bubble page sizes are established, under software control, when the bubble system is initialized. For a single bubble system, the page size is fixed at either 64 bytes when error correction is implemented or 68 bytes without error correction, and the total number of pages available is 2048. In systems with multiple bubbles, page size can vary from 64 bytes (68 bytes without error correction) to 512 bytes (544 bytes without error correction) depending on the number of bubble devices in the system. Page size is directly proportional to the system data rate and also determines the total number of available pages (address field size). As an example, consider a system consisting of two bubbles (using error correction). With two bubbles, there are two possible ways to configure the system; paralleling the two bubbles for a page size of 128 bytes and a total number of 2048 pages or treating the bubbles serially for a page size of 64 bytes and a total number of 4096 pages. The average data rate for the 128-byte page is 17.0 kbytes per second, and the average data rate for the 64-byte page is 18.5 kbytes per second.

The selection of the appropriate page size depends primarily on the data rate supported by the system. For file system implementation, an additional consideration in page size selection is bubble transfer efficiency versus bubble storage efficiency. Essentially, the following system factors must be weighed:

- Data Rate. The higher the data rate, the faster the microprocessor must respond to the demands of the bubble memory controller. Depending on the data transfer mode selected, some data rates may exceed the data transfer rate of the host microprocessor.
- Bubble Transfer Efficiency. A file consisting of a few large pages can be transferred more efficiently (faster) than a file consisting of a number of small pages.
- Bubble Storage Efficiency. For a typical file system, space must be allocated within each page to link the pages of each file together (individual pages of a file may not necessarily be contiguous). Too large or too small a page size can waste

bubble storage space. If most files are comprised of many small pages (e.g., 64 bytes), a large percentage of bubble storage would be required for establishing the fore/back pointer linkage. Conversely, if most files are smaller than a single page, a large amount of space would remain unused at the end of each page.

Buffering

Buffer operation is an extremely important factor in the reliable transfer of data between the bubble and system memory and is a major consideration in software driver design. A buffer, ideally speaking, is a memory storage area that contains the same amount of data storage as the data block to be transferred. The bubble system's bubble memory controller includes a first-in, first-out (FIFO) 40 byte data buffer that reconciles timing differences between the parallel data transfer to or from the host microprocessor and the serial data transfer to or from the bubble memory. Accordingly, when an application program requests data from a bubble, the software driver is responsible for keeping up with the FIFO for the duration of the data transfer in order to prevent the FIFO from overflowing or underflowing. The specific software driver requirements are dependent on the method of data transfer selected.

Data Transfer Interface Modes

Three distinct software interface techniques can be used to interface host system memory with the bubble system for page data transfer: DMA: interrupt driven, and polled.

In the DMA transfer mode, the BMC operates in conjunction with a DMA controller (e.g., Intel's 8257 or 8237) and uses the DRQ (data request) and DACK/ (data acknowledge) signal lines for establishing the handshake protocol. Assisted by the DMA controller, the BMC transfers data to or from the host system memory. Once the transfer begins, further program intervention is not required until the entire transfer has been completed.

In the interrupt-driven data transfer mode, the DRQ line is connected to an interrupt controller (e.g., Intel's 8259A). During non-DMA data transfers the DRQ line indicates when the BMC's FIFO is half-full (bubble read operations) or half-empty (bubble write operations). This method of interrupting results in a data block transfer arrangement in which the software is responsible for performing the appropriate transfer of data (typically 22 bytes) to or from the FIFO when the interrupt occurs. Using this technique, the software driver only processes the FIFO buffer as needed, and program waits during I/O transfers (polled I/O) are eliminated.

The polled I/O mode is the most simple to implement since no special or external hardware is required to perform data transfers. In the polled I/O mode, the software must determine when to transfer data to or from the FIFO by continually polling a status bit in the BMC's Status Register. This status bit indicates the presence or absence of data in the FIFO on a byte-by-byte basis. The polled I/O mode places significant demand on the host system's processing time since the software continuously must monitor the Status Register to ensure that FIFO overflow or underflow does not occur.

ECC Highlights

The last software controlled option to be considered in the design of your bubble driver is if the built-in error correction circuitry (ECC) within the 7242 Formatter/Sense Amplifier is to be implemented and, if so, what level of error correction is best suited to your application.

Although the inherent data integrity of your bubble memory is extremely high, the incorporation of error correction improves the overall integrity of your system by several orders of magnitude. While boosting the data integrity, the implementation of error correction adds increased overhead to both driver design and host interaction. Since the associated error handling routines can range from simple to complex, you must carefully weigh the software requirements before considering the level of error correction to be supported. In balancing driver and host responsibilities, you must understand the following factors pertaining to ECC:

- The type and nature of errors associated with bubble memories.
- The way in which ECC operates.
- The various levels of error correction available.

All of these factors are explained in detail in a later section.

COMMUNICATING WITH THE BMC

All communications between the host and the bubble memory actually are peformed through the 7220-1 BMC. The BMC has two input/output (I/O) ports; an 8-bit bidirectional data port and an 8-bit command/status port (Figure 1). The port addressed is determined by the least-significant bit of the port address byte. Conceptually, the BMC can be thought of as a disk system controller in that data in the bubble memory is organized into blocks called "pages" that are similar to disk sectors. Information such as starting page location, direction of transfer, and the number of pages to be transferred is passed to the BMC before the desired read or write operation is initiated.

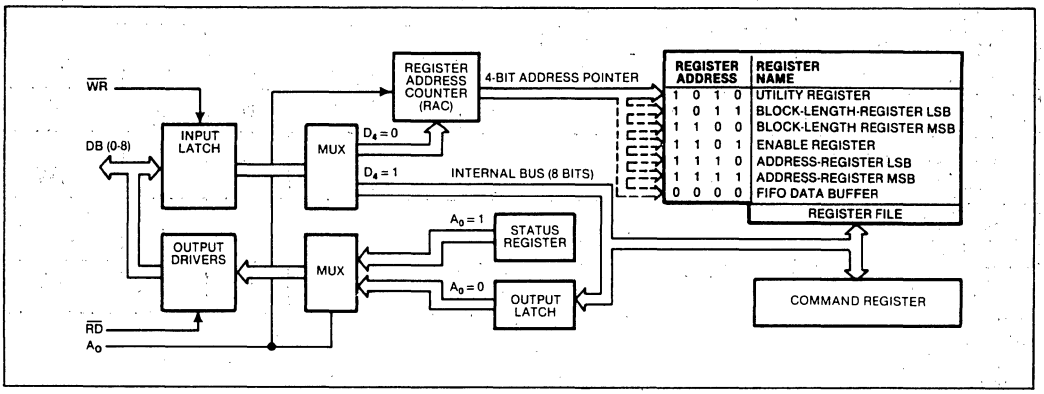


Figure 1. BMC Block Diagram

For simplicity, you can think of the BMC as a 40-byte FIFO (first-in first-out) buffer and a series of six user-accessible 8-bit registers. The FIFO passes data between the outside world and the bubble system's 7242 Formatter/Sense Amplifier and compensates for speed variations. The six registers are loaded prior to most operations and contain information regarding the upcoming transfer and the operating mode of the BMC. Since these registers always are loaded before a command is sent similar to passing parameters to a subroutine before it is invoked, this set of registers is referred to as the "parametric registers." The transfer of data between the host and the BMC's FIFO and parametric registers takes place over the 8-bit data port. The destination (FIFO or parametric register) of the data port transfer is determined by the value in another register called the RAC. As shown in Table 1, bit 4 of the command/status port byte is used to distinguish between a BMC command and either the address of one of the parametric registers or the FIFO.

Table 1. Command Port Function

Function	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Command	0	0	0	1	C	C	C	C
RAC	0	0	0	0	R	R	R	R

When bit 4 is a "one," the low-order four bits are decoded as a BMC command, and when bit 4 is a "zero," the low-order four bits are interpreted as a pointer to the parametric registers/FIFO. This 4-bit register pointer is referred to as the "Register Address Counter" or simply the "RAC."

RAC values that may be written to the command/status port and the registers selected are outlined in Table 2. The RAC points to, or selects, one of the six registers or the FIFO. Once a RAC value is written to the command status port, the next data port read or write operation transfers data between the host interface and the register addressed.

Table 2. Register Address Counter Assignments*

Register Name	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Read/ Write
Utility Register	0	0	0	0	1	0	1	0	R/W
Block Length Register (LSB)	0	0	0	0	1	0	1	1	W
Block Length Register (MSB)	0	0	0	0	1	1	0	0	W
Enable Register	0	0	0	0	1	1	0	1	W
Address Register (LSB)	0	0	0	0	1	1	1	0	R/W
Address Register (MSB)	0**	0	0	0	1	1	1	1	R/W
7220 FIFO	0	0	0	0	0	0	0	0	R/W

NOTE(S):
* With A0 = 1
** Write-only bit

Referring now to the table, notice that the register addresses are in hexadecimal order from "A" to "F" and that the FIFO has an address of zero. This arrangement of addresses is due to the RAC's auto-incrementing feature. Once a register is selected, each subsequent data port I/O read or write causes the RAC to advance and to point to the next register in the sequence. After the most significant byte of the Address Register is addressed, the RAC advances from F to 0 to point to the FIFO. When it reaches this point it no longer increments. The system now is ready to transfer data into or out of the FIFO without further instructions from the host.

After the FIFO is selected, the RAC stops incrementing and continues to point to the FIFO until the RAC again is accessed through the command/status port. The auto-incrementing feature minimizes the number of instructions required for a given command sequence and ensures that all of the required parametric information is sent to the BMC.

As a user, you are not required to utilize the auto-incrementing feature; each parametric register can be selected and loaded in any order, and specific registers may be updated as required. When individual registers are not accessed in order, each register must be specifically addressed and loaded. Until you become more familiar with the bubble system, the auto-incrementing feature is recommended.

A point to remember is that once a command has been issued to the BMC, the parametric registers must not be updated until the operation is complete. The parametric registers essentially are working registers for the BMC during command execution. When a bubble read or write operation is in progress, the Block Length Register, as explained later in this chapter, contains the terminal page count and is decremented with each page transferred. Attempting to modify this register during command execution would cause the final page count to be incorrect.

The Parametric Registers

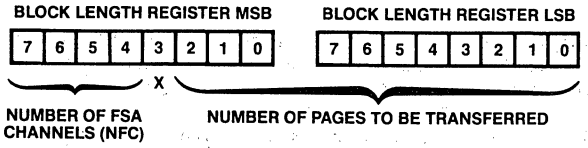
Now that you have been introduced to the Register Address Counter and its operation, let's look at the individual parametric registers addressed by the RAC in more detail.

Utility Register

The Utility Register is a user-defined register that can be both written and read. This register is not incremented or decremented by the BMC. Since the Utility Register is the first register of the RAC sequence, if the register is not used, the least-significant byte of the Block Length Register initially can be addressed to eliminate the Utility Register from the sequence. Note that the 4 Mbit bubble memory controller (7224) does not contain a utility register.

Block Length Register

The Block Length Register is made up of two 8-bit registers, a low-order byte register and a high-order byte register. 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 Block Length Register or "BLR" is a write-only register that is divided into a terminal count field and a channel field as follows:



The terminal count field is eleven bits in length and is loaded with the total number of pages to be transferred in the ensuing bubble read or write operation. With a field length of eleven bits, from 1 to 2048 pages can be transferred (all zeroes in the field indicates a 2048-page transfer).

The page width (size) is defined by the 4-bit channel field. This field actually specifies the number of formatter/sense amplifier channels available. Note that each 7242 formatter/sense amplifier has two channels to communicate with each bubble memory, therefore the acceptable values in this field select one channel (one half of a bubble memory), two, four, eight, or 16 channels. These field values correspond to page sizes of 32, 64, 128, 256, and 512 bytes (assuming error correction), respectively, when the bubble memories are operated in parallel. (The one-channel mode usually is reserved for diagnostic operations.) Table 3 shows the relationship among page size, channel field value, and formatter/sense amplifier channel selection for parallel bubble operation.

As shown in the table, the channel field bits are encoded and only one bit ever is set in the field.

For example, a channel field value of "0001" selects one bubble memory through channels 0 and 1.

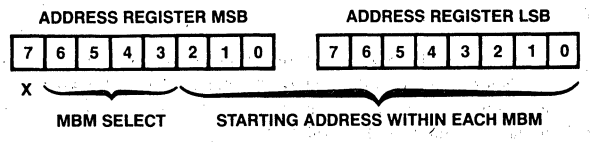
Table 3. FSA Channel Select/MBM Select

MBM Select AP, MSB Bits (6, 5, 4, 3)	"Channel Field" (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				

NOTE(S):
*Normally reserved for diagnostic operations.

Address Register

The Address Register, like the Block Length Register, is made up of two 8-bit registers; a low-order byte register and a high-order byte register. The Address Register is divided into a starting address field and an MBM (Magnetic Bubble Memory) select field show as follows:



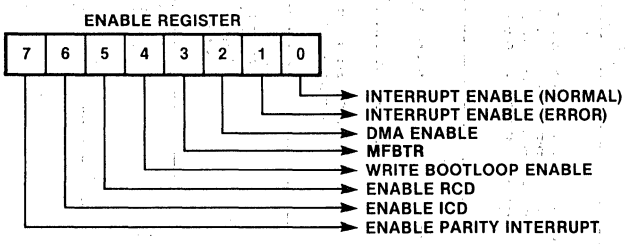
The Address Register's starting address field is eleven bits in length and is used to define on which page of a bubble's 2048 pages that the transfer is to start. The starting address field is incremented with each page transferred during multipage transfers, automatically selecting the next sequential page.

The Address Register's MBM select field is used in conjunction with the Block Length Register's channel field to control the serial selection of bubble memories or groups of bubble memories operated in parallel. To better understand the function of the MBM select field, consider a system consisting of four bubble memories operated as two banks of two bubble memories each.

Referring back to table 3, the channel field in the Block Length Register would be set to "0010" to select two bubbles in parallel and a corresponding page size of 128 bytes. To select between the two banks, the Address Register's MBM select field would be set to "0000" to select the first bank (FSA channels 0 through 3). As page 2048 is transferred to or from the first bank, the Address Register's starting address field rolls over to "0000" and increments the MBM select field to "0001" to select the second bank (FSA channels 4 through 7).

Enable Register

While the Address and Block Length Registers define the system configuration and data transfer, the Enable Register defines the various modes of operation under which the data transfer is performed and defines the conditions under which interrupts can be generated. Several of the Enable Register bits are used individually while other bits are used in combination. Figure 3 shows the individual Enable Register bit definitions.



Interrupt Enable (Normal), when set ("1"), enables the BMC to interrupt the host processor on the successful completion of a command. Conversely, if this bit is not set, an interrupt is not generated on command completion and the host processor must poll the BMC's Status Register to determine when command execution is complete.

Interrupt Enable (Error) is used in conjunction with the Enable RCD and Enable ICD bits to select various error conditions under which the BMC will terminate command execution and interrupt the host processor. The following table outlines the bits combination and corresponding error conditions recognized.

Table 4. Error Correction Options

Enable ICD (bit 6)	Enable RCD (bit 5)	Interrupt Enable (Error) (bit 1)	Interrupt Condition
0	0	0	No interrupt on error
0	0	1	Interrupt only on timing error
0	1	0	Interrupt on uncorrectable or timing error
0	1	1	*Interrupt on uncorrectable, correctable or timing error
1	0	0	Interrupt on uncorrectable or timing error
1	0	1	Interrupt on uncorrectable, correctable or timing error
1	1	0	Illegal
1	1	1	Illegal

NOTE(S):
*Normally not used.

DMA Enable, when set, enables the BMC to operate in the DMA data transfer mode. In the DMA mode, a DMA controller is interfaced to the BMC and DRQ-DACK/protocol is used to perform byte transfers. Note that in the DMA mode, the BMC activates its DRQ output *each time* it places a byte in the FIFO (bubble read operation) or each time there is room for *at least one byte* in the FIFO (bubble write operation). When the DMA Enable bit is not set, the BMC operates in the interrupt driven or polled mode. In either of these modes, the BMC's DRQ output goes active when 22 or more bytes are present in the FIFO during a bubble read operation or when there is space for 22 bytes during a bubble write operation. Note that if the DRQ is not used (i.e., polled mode), the host processor must examine the Status Register's FIFO Ready bit to determine when data should be taken from or written to the FIFO.

MFBTR, (Maximum FSA to BMC Transfer Rate), determines the maximum burst transfer rate from the FSA(s) to the BMC FIFO. This bit only applies to bubble read operations and is effective only during single-page transfers or during the transfer of the last page of a multipage transfer. Table 3 shows the effect of MFTBR bit on the transfer rate for parallel bubble operation.

Write Bootloop Enable, when set, enables the bootloop to be rewritten. Conversely, if this bit is not set and a Write Bootloop command is received, the command is aborted immediately and the Timing Error is set in the Status Register. Since writing the bootloop only is performed as a diagnostic test or to recover from a system failure, this bit provides protection against accidental rewrite of the bootloop data.

Enable RCD (Read Corrected Data), when set, causes the BMC to issue a Read Corrected Data instruction to all the FSAs in the system in response to one or more FSAs reporting an error. On receipt of the instruction, each FSA cycles the erroneous page through its error correction logic and then transfers the page to the BMC. For any FSA not reporting an error, the data harmlessly cycles through the error correction logic. When the page transfer is complete, the BMC interrogates the FSA to determine if the error was correctable (if the error was uncorrectable, erroneous data would have been transferred to the BMC).

Enable ICD (Internally Correct Data), when set, causes the BMC to issue an Internally Correct Data instruction. Any FSA reporting an error causes the instruction to be issued to all FSAs in the system.

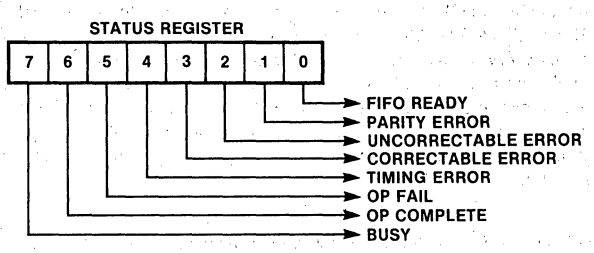
On receipt of the instruction, each FSA cycles the data through its error correction logic (regardless of whether it contained an error), but does not transfer the page to the BMC. After cycling the page, each FSA reports its error Status (correctable or uncorrectable) to the BMC. The FSA not reporting an error harmlessly cycles through the error correction logic and reports a correctable error to the BMC. Note that both the Enable RCD and Enable ICD bits cannot be set at the same time.

Enable Parity Interrupt, when set, enables the BMC to interrupt the host processor when it detects a parity error on a data byte sent from the host processor (the BMC automatically checks for odd parity on each data byte received from the host processor and implements odd parity on each byte sent to the host processor). When the Enable Parity Interrupt bit is not set and parity error is detected, no interrupt is generated (following the transfer, the Parity Error bit in the Status Register will be set).

Status Register

As stated at the beginning of this chapter, the host processor executes an I/O read instruction with the BMC's A0 address input equal to "1" to access the Status Register. As will become evident in the individual Status Register bit descriptions, the Status Register is read in response to interrupts from the BMC and, during polled data transfers, is read continually to determine when data is to be written to or read from the BMC's FIFO.

The Status Register also provides information regarding error conditions, the completion or termination of commands, and the BMC's readiness to accept new commands. The individual Status Register bits are shown as follows.



Note that bits 1 through 6 are valid only when the Busy bit (bit 7) is not set and that these bits are cleared whenever a new command is received. The Status Register also can be cleared by writing to the register address counter (RAC) with bit 5 set to "1" and any valid parametric register address (0AH through 00H).

Busy. Bit 7, the Busy bit, is set ("1") when the BMC is in the process of executing a command. The BMC sets its Busy bit shortly after a command is received (the Busy bit must be clear in order for the BMC to accept any command other than an Abort command) and keeps Busy set until the operation is complete (or until an operation is halted because an error has occurred.)

It is important to note that the Busy bit remains set until all other status bits have been updated and that it therefore is possible to see illogical bit combinations such as Busy and Op Complete at the same time. With the exception of the FIFO Available bit, all other Status Register bits should be considered valid only after the Busy bit returns to an inactive ("0") level.

OP Complete, Bit 6, Op Complete, is set when the BMC successfully completes the execution of a command.

OP Fail, Bit 5, Op Fail, is set when the BMC is unable to complete execution of a command. In general, this bit is valid only after the Busy bit returns to an inactive level; other error bits in the Status Register will be set to indicate why the operation failed.

Timing Error, Bit 4, Timing Error, is set for several error conditions. The most frequent cause of timing errors is when the host processor cannot keep up with the rate at which the BMC fills or empties its FIFO (referred to as an overflow or underflow condition). Timing errors also occur if the correct number of bits is not set in an FSA's bootloop register (to function properly, an FSA must have either 272 loops active when error correction is not implemented or 270 loops active when error correction is implemented). This condition occurs during a Read command if a mistake is made either when the bubble's bootloop is written or if the bootloop register is loaded incorrectly from the user's system. Another source of timing error is if no bootloop sync code is found during an Initialize or Read Bootloop command. The last source of a timing error is when a Write Bootloop command is received by the BMC and the Write Bootloop Enable bit is not set in the Enable Register. Of the preceding sources, regular or periodic occurrences of timing errors usually indicate an inherent inability of the host processor to meet the bubble's data transfer requirements and will be most apparent during bubble read operations, especially if the MFBTR bit is set (i.e., MFBTR=0 in the Enable Register).

Correctable Error, Bit 3, Correctable Error, is set when an FSA reports that a correctable error was detected during the last bubble read operation. Depending on the level of error correction selected, the setting of this bit indicates a correctable error in the current page held by the FSA, in the last page read, or in one of the pages read during a multipage transfer.

Uncorrectable Error, Bit 2, Uncorrectable Error, is set when an FSA reports an uncorrectable error during the last bubble read operation. Like the Correctable Error bit, the setting of this bit depends on the level of error correction selected and indicates an uncorrectable error in either the last page transferred or in the page currently held by the FSA.

Parity Error, Bit 1, Parity Error, is set when the BMC detects a parity error on the data byte sent from the host during a bubble write operation. If the Enable Parity Interrupt bit is set in the Enable Register, the BMC will terminate the write operation and interrupt the host processor when the parity error is detected.

FIFO Available, Bit 0, FIFO Available, is unique in that it is the only bit the Status Register that is valid when the Busy bit is set. During data transfer operations (i.e., when the Busy bit is set), the FIFO Available bit acts as a gate for the host microprocessor's data handling software. During bubble write operations, if the FIFO Available bit is set there is room for more data in the FIFO and the host microprocessor can proceed with the transfer. Conversely, if the FIFO Available bit is not set, the FIFO is full and the host must wait for the BMC to "catch up" before sending more data. During bubble read operations, if the FIFO Available bit is set, data has been placed in the FIFO by the BMC and can be taken by the host microprocessor; if the FIFO Available bit is not set, the data is not yet available.

FIFO

The BMC's first-in first-out (FIFO) buffer passes data between the user interface and the FSA. The primary purpose for the FIFO is to reconcile timing differences between both the user interface and the BMC and between the BMC and the FSAs.

The FIFO itself is 40 bytes wide and, including the FIFO's input and output latches and the BMC's input latch, can store up to 43 bytes. The FIFO is "dual ported" in that data can be written into one port while simultaneously being read from the other port.

When the BMC is busy executing a command, the FIFO functions as a data buffer, and when the BMC is not busy, the FIFO is available for use 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. During execution of commands that involve the transfer of data between the user interface and the FSAs, the data passes through the FIFO, and the status of the FIFO is indicated by the FIFO Ready bit in the BMC's Status Register. When the FIFO Ready bit is set ("one") during a write operation, there is space in the FIFO for more data, and when this bit is set during a read operation, data is present in the FIFO.

The BMC's DRQ output also indicates FIFO status. In the DMA data transfer mode, DRQ is used in conjunction with the DACK/ input from a DMA controller (e.g., in Intel 8257 or 8237) on the user interface to provide a standard DMA data transfer protocol. In the non-DMA transfer mode (polled or interrupt-driven data transfers), the DRQ output indicates that the FIFO is either half full or half empty according to the direction of the data transfer; during a write operation, DRQ is set when there is space for 22 more bytes, and during read operation, DRQ is set when there are 22 bytes present in the FIFO. Accordingly, when performing non-DMA data transfers, blocks of 22 bytes should be transferred to or from the FIFO so that the host processor does not spend a disproportionate amount of time servicing the DRQ-initiated interrupt or polling the BMC's Status Register.

The FIFO automatically is addressed after the last parametric register is written due to the self-incrementing nature of the Register Address Counter. Alternatively, the FIFO can be addressed explicitly by writing to address 0 of the Register Address Counter. Note that since byte 0 of the FIFO is cleared whenever the Register Address Counter is addressed, writing the parametric registers (or to the FIFO itself) must be avoided while the FIFO contains valid data. Also, when using the FIFO in the general-purpose mode, the host system is responsible for resetting the FIFO prior to any data transfer.

During read and write operations, the host system is responsible for keeping up with the data transfer in order to prevent a FIFO overflow or underflow condition. If the FIFO overflows or underflows during the data transfer, the operation is aborted and the Op Fail and Timing Error bits are set in the BMC's Status Register.

The FIFO AVAILABLE bit is also set whenever the RAC is not pointing to the BMC FIFO (RAC address = 00H). When writing the parametric registers, for example, if the user reads the status between say the Enable Register and the Address Register, the status byte would indicate FIFO AVAILABLE. This status value is forced by internal BMC logic.

COMMANDS

The BMC's command set consists of 16 commands that are selected by a 4-bit command code. As previously described, commands are passed to the BMC by writing to the BMC's command port with bit 4 of the command byte set to "1" (the Register Address Counter is addressed when bit 4 is "0"). Table 5 lists the 4-bit command codes and the corresponding commands.

For most commands sent to the BMC, the parametric registers must be loaded with operating information before the command is sent. Also, with the exception of the Abort command, commands cannot be issued to the BMC while another command is being executed (i.e., when the Busy bit in the Status Register is set). The remainder of this section offers brief descriptions of the BMC's command set; descriptions of command usage are presented in succeeding sections. Table 6 of this section presents a summary of command execution times.

The 16 commands can be grouped into categories according to their frequency of use. The most commonly used commands are:

- Abort
- Initialize
- Read Bubble Data
- Write Bubble Data

Other commands used during normal bubble system operation include:

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

The remaining commands are related to bootloop operation and are used only for diagnostic purposes:

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

Abort

The Abort command (unlike any other command), when issued while the BMC is busy executing another command, causes the termination of the command being executed. On receipt of this command, the MBMs are stopped in an orderly manner to prevent the loss of bubble data. Since the Abort command is the only command recognized by the BMC while it is busy, this command is issued following power-up or whenever the BMC is in an unknown state. The Abort command requires no prior loading of the parametric registers.

When an Abort command is issued while the BMC is not busy, the command functions as an FIFO Reset to clear any data present in the FIFO. An Abort command issued when the MBM drive coils are active (i.e., data transfer command is executing indicated by the Busy bit in the Status Register) must be followed by an Initialize command.

Table 5. BMC Command Set

D3	D2	D2	D1	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

Table 6. 7220-1 Command Execution Times

Four-Bit Command Code (Hex)	Command	Description	Performance
0	LRBLRMSK	1 FSA channel selected 2 FSA channel selected	900 μs 900 μs
1	Initialize	Best case (N=#MBM) Worst case	350+(85,200) N μs 350+(164,740) N μs
2	Read	Single page (MFBTR=0) Single page (MFBTR=1) N page transfer (MFBTR=0) N page transfer (MFBTR=1)	t_{SEEK}+8690 μs t_{SEEK}+12,770 μs t_{SEEK}+8690+(7500) (N−1) μs t_{SEEK}+12,770+(7500) (N−1) μs
3	Write	1 page transfer N page transfer	t_{SEEK}+7450 μs t_{SEEK}+7450+(7500) (N−1) μs
4	Read Seek	Best case Worst case	7350 μs 89,250 μs
5	Read BLR	Any number of FSA channels	900 μs
6	Write BLR	Any number of FSA channels	900 μs
7	Write BL	Single bubble selected	82,850 μs
8	Read FSA Status	1 bubble in system N bubbles in system	75 μs 75+40 (N−1) μs
9	Abort	1 bubble in system N bubbles in system	100 μs 100+40 (N−1) μs
A	Write Seek	Best case Worst case	7350 μs 89,250 μs
B	Read BL	Best case Worst case	86,000 μs 165,000 μs
C	Read Corrected Data	Any number of FSAs selected	1400 μs
D	FIFO Reset	N/A	50 μs
E	MBM Purge	N/A	150 μs
F	Software Reset	N/A	50 μs

Initialize

The Initialize command prepares the bubble system for subsequent operations and is used when the bubble system is powered up (following the Abort command) or whenever the system's error correction implementation is changed. The Initialize command effectively performs the following BMC commands:

- Abort
- MBM Purge
- FIFO Reset
- Read Bootloop
- Write Bootloop Register Masked

When an Initialize command is received, all internal registers within the BMC are cleared and the FIFO is reset. The BMC then reads the bootloop from each bubble and writes the corresponding bootloop information into the bootloop registers of each FSA. The bubble is left positioned at page zero (logically) corresponding to the values set in the BMC page address counters. Before an Initialize command can be issued, the following information must be loaded into the parametric registers:

- The channel field in the Block Length Register must be set to "0001" to arrange all bubbles in the system in a serial configuration to allow the individual bootloops to be read from each bubble and subsequently written to the bootloop registers of the corresponding FSA channels.
- The MBM select field in the Address Register must select the last (highest numbered) bubble in the system to inform the BMC of the number of bubbles present within the system (for a one bubble system the value would be "0000").
- The bits selecting error correction in the Enable Register must be set according to how subsequent read and write operations are to be performed (with or without error correction) since the number of 1's written to the FSA bootloop registers is not the same with error correction implemented as when error correction is not implemented. Note that simply switching between ECC modes (level 1, 2, or 3) does not require re-initialization.

Read Bubble Data

The Read Bubble Data command causes data to be read from the MBMs and into the BMC's FIFO. Immediately before the Read Bubble Data command is issued, the host computer must load, with the exception of the Utility Register, all of the parametric registers. Specifically, the following parametric information must be loaded prior to command execution:

- The channel and "number of pages to be transferred" in the Block Length Register must be set to define the page size (number of FSA channels) and number of pages to be transferred.
- The appropriate bits must be set in the Enable Register to select the transfer mode (DMA or non-DMA) and interrupt sources; if error correction is to be used (i.e., if the FSA bootloop registers have been initialized for error correction), the level of error correction must be selected.
- The MBM select and starting address fields in the Address Register must be set to define the (first) MBM and page within the MBM where the transfer is to occur.

Write Bubble Data

The Write Bubble Data command causes data read from the BMC's FIFO to be written into the MBMs. Immediately prior to issuing the Write Bubble Data command, the host computer must load the Block Length, Enable, and Address Registers as described for the Read Bubble Data command. Data should not be loaded into the BMC FIFO until after the command has been issued. Note that a write data transfer does not begin until at least two bytes of data have been loaded into the BMC FIFO.

Read Seek

The Read Seek command is used to reduce system access time of a subsequent Read Bubble Data command by positioning the page to be read at the selected bubble's output track. Note that although the Read Seek command does not cause any data to be transferred, the following information must be loaded into the parametric registers before the command is issued:

- The channel field in the Block Length Register must specify the page size (number of FSA channels).
- The error correction bits in the Enable Register must be set identical to the values used when the system was initialized.
- The MBM select field in the Address Register must be set to select the bubble containing the page to be read and the address field must specify a page number that is one less than the (first) page to be read by the subsequent Read Bubble Data command.

Write Seek

The Write Seek command is used to reduce the system access time for a subsequent Write Bubble command by positioning the page to be written at the selected bubble's input track. Note that like the Read Seek command, the Write Seek command does not cause any data to be transferred. Similarly, the following information is issued:

- The channel field in the Block Length Register must specify the page size (number of FSA channels).
- The error correction bits in the Enable Register must be set identical to the values when the system was initialized.
- The MBM select field in the Address Register must be set to select the bubble containing the page to be written and the address field must specify a page number that is one less than the (first) page to be written by the subsequent Write Bubble Data command.

MBM Purge

The MBM Purge command is used in place of the Initialize command when the bootloop register is to be loaded from an external source (once a bootloop has been identified, the bootloop register pattern can be maintained by the host and loaded directly from external memory with a Write Bootloop Register or Write Bootloop Register Masked command to conserve power and increase speed during an initialization sequence). The MBM Purge command clears the BMC's internal registers and the address field of the Address Register. The MBM select field in the Address Register and the other parametric registers are not cleared. Like the Abort command, the MBM Purge command does not require the loading of the parametric registers prior to command execution.

Read Corrected Data

The Read Corrected Data command is used only when error correction is implemented and only is applicable when error correction level 2 or 3 is selected (the Read Corrected Data command is issued automatically by the BMC when error correction level 1 is selected). When an FSA reports a correctable error, the Read Corrected Data command is issued to cause the corrected page in the FSA(s) to be read into the BMC's FIFO. Note that since the parametric registers previously were loaded for the read operation in which the error was detected and since the address field is not incremented (i.e., the address of the page in error is in the address field), it is not necessary to load the parametric registers prior to issuing the Read Corrected Data command.

Reset FIFO

The Reset FIFO command causes the BMC's FIFO (and its input and output latches) to be cleared. Normally, this command is issued prior to issuing the Write Bootloop, Write Bootloop Register, or Write Bootloop Register Masked commands to ensure that the FIFO will accept 40 bytes; the Reset FIFO command does not require loading of the parametric registers.

Software Reset

The Software Reset command clears the BMC's internal registers, and the terminal count field in the Block Length Register and starting address field in the Address Register. Additionally, the Software Reset command causes the BMC to clear the status register of every FSA channel in the system. Since the Software Reset command does not clear the FSA channel and MBM select fields in the Block Length and Address Registers or any of the Enable Register bits, it is not necessary to reinitialize the parametric registers following a Software Reset command, and no loading of the parametric registers is required prior to command execution.

Read FSA Status

The Read FSA Status command causes the BMC to read the 8-bit status register of each FSA channel in the system (the number of FSA channels specified in the channel field of the Block Length Register). This command is issued to determine the number of FSAs in a system. Following command execution, the individual status register bytes will be in the BMC's FIFO in ascending order; one byte per FSA channel. All values returned to the host will be 20H or 28H if error correction is enabled. This occurs because the FSA status is cleared when read. The BMC always reads the FSA status prior to interrupting (or informing) the host of an error. Therefore, the Read FSA status command can only read the "cleared" FSA status values (20H or 28H). No loading of the parametric registers is necessary prior to command execution.

Read Bootloop Register

The Read Bootloop Register command causes the BMC to read the bootloop register of the selected FSA channel into its FIFO. This command is used initially to ensure that the bubble system is communicating properly (bubble-to-FSA and FSA-to-BMC communication established) and is used to transfer bootloop information to the host system for subsequent bootloop initialization from an external source (e.g., user EPROM). Prior to command execution, the channel field in the Block Length Register and the MBM select field in the Address Register must be loaded with the number of FSA channels (normally two) and the corresponding bubble (FSA channel pair) to be selected. Note that since each individual FSA channel's bootloop register contains 20 bytes, reading a pair of FSA channels fills the BMC's 40-byte FIFO; reading the bootloop registers of more than two channels is possible but not recommended since data must be taken from the FIFO to avoid an overflow condition. Also, since the bootloop register data from each FSA channel pair actually is interleaved on a bit-by-bit basis before it is assembled into bytes, reading the bootloop register for a single channel likewise is not recommended. Remember that a unique BMC status value (C1H or C3H) is expected with this command (see "Considerations for Polled Command Execution").

Write Bootloop Register

The Write Bootloop Register command causes the BMC to write the contents of its FIFO into the bootloop register of the selected FSA channel(s). This command (and the Write Bootloop Register Masked command) is used during bubble system initialization when the bootloop is written from an external source rather than from the bubble itself. In order to use the Write Bootloop Register command, the bootloop register data must be loaded into the FIFO prior to command execution and the channel and MBM select fields of the Block Length and Address Registers must be loaded with the number of FSA channels (normally two) and the corresponding bubble (FSA channel pair) to be selected. Recalling that each individual FSA channel's bootloop register contains 20 bytes, 40 bytes normally are written into the FIFO to initialize the two FSA bootloop registers associated with each bubble. However, a 41st byte of zeroes *must* be written to ensure a successful command execution status. Note that the parametric registers should be loaded prior to loading the FIFO.

Proper operation of the bubble system requires that each FSA bootloop register contains either 135 (error correction selected) or 136 (error correction disabled) logic "1" bits that correspond to the 135 or 136 valid data storage loops.

Write Bootloop Register Masked

The Write Bootloop Register Masked command is identical to the Write Bootloop Register command previously described with the exception that the number of "1" bits written automatically is stopped when the required 135 or 136 logic "1" bits have been written for each channel. The parametric registers are loaded as described for the Write Bootloop Register command; the number of logic "1" bits written (135 or 136) is determined by the setting of the error correction bits in the Enable Register.

Write Bootloop

The Write Bootloop command causes the existing contents of the selected bubble's bootloop to be replaced by the 40 bytes of information currently contained in the BMC's FIFO. This command is used only after it has been determined that the existing bootloop is invalid (i.e., a storage loop previously identified as "good" has become defective) and typically is not required for the life of the bubble system. Remember that the parametric registers must be loaded prior to pre-loading the FIFO with the 40 bytes of information to be written followed by a 41st byte of zeroes.

If it should become necessary to use the Write Bootloop command, the channel field in the Block Length Register and the MBM select field in the Address Register must be loaded with the number of FSA channels and the corresponding bubble to be selected. As an additional safeguard, the Write Bootloop Enable bit in the Enable Register additionally must be set (if this bit is not set, command execution immediately will be aborted and the Timing Error bit will be set in the Status Register).

Read Bootloop

The Read Bootloop command causes the BMC to read the bootloop of the selected bubble into its FIFO. This command is used to determine if the existing bootloop is valid by comparing the bubble bootloop information to that on the label of the device. Remember that the parametric registers must be loaded prior to command execution. Also note that a BMC status value of C1H or C3H is expected (see "Considerations for Polled Command Execution").

ERROR CORRECTION

As mentioned earlier in this application note, several factors pertaining to error correction must be understood and weighed according to your specific application before implementing error correction. From a software perspective, the selection of the appropriate level of error correction is based on the host system's requirements relative to error logging and data recovery. Before describing the individual levels of error correction and the associated software requirements, the types of errors that can occur within bubble memories are described as a preface to error correction.

Bubble Errors

To understand the function and implementation of error correction, the differences between bubble errors and semiconductor device errors must be distinguished. In conventional semiconductor memory, errors are classified as either hard (non-recoverable) or soft (recoverable). Usually, hard errors are the result of physical or irreversible damage within the device itself (e.g., oxide breakdown or junction burnout), and soft errors are the result of transient conditions and generally do not reappear when the data is rewritten.

Unlike semiconductor devices, bubble memory devices have no active elements and, as such, rarely experience hard (non-recoverable) errors. Accordingly, all bubble memory errors can be considered as soft errors (for information on irreversible failure mechanisms in bubbles, refer to the Intel 7110 Bubble Memory Reliability Report, RR-36 Order Number 210632). In order to further define the nature of soft errors in bubbles, errors are classified either as "data" errors or "read" errors. A data error occurs when a data inversion occurs within a storage loop; the data is lost, but since no physical damage has occurred, the data can be rewritten and the bubble can remain operational. Data errors typically occur when the bubble is operated beyond its safe operating region and, as such, are seldom encountered during normal operation. The most common type of soft error is a "read" error that occurs during a bubble read operation, usually as a result of noise in the detection/sense circuitry. Since the data in the storage loop is unaltered during a bubble read, the data can be recovered by simply repeating the read operation.

Error Detection/Correction Capability

In respect to the Formatter/Sense Amplifier (FSA), errors either are correctable (the FSA is able to reconstruct the data using an error correction algorithm before the data is transferred to the BMC) or uncorrectable, irrespective of the type of error (data error or read error). The error correction code used by the FSA is a 14-bit Fire code that is 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 and has proven to be well suited to the error model for the 7110 MBM. Table 7 outlines the FSA's error correction capability and probability of page errors; the bit error rate can be obtained by dividing the "probability of page in error" by the number of bits per page.

ECC Options

The FSA's error correction circuitry (ECC), in conjunction with the BMC, provides three levels of error correction. Each level places unique demands on the host system that range from simple data recovery to the logging of specific pages in which the error occurs. The desired level of error correction is selected by the setting of the appropriate bit or bits within the BMC's Enable Register. Table 8 defines the relevant Enable Register bits for the three levels of error correction available.

Table 7. FSA Error Detection/Correction Capability

Type of Error	Approximate Probability of Page in Error	% Correction	% Detection
Single Read Error	10^{-6}	100	—
Single Data Error	10^{-7}	100	—
Random Double Read Error	10^{-12}	—	100
Read Error Burst Length 2	—	100	—
Data Error Burst Length 2	—	100	—
Read Error Burst Length 3/4	10^{-9}	100	—
Random Double Data Error	10^{-14}	—	100
Read Error Burst Length 5	—	100	—
Random Triple Read Error	10^{-17}	—	100
Single Soft + Read Burst 2	—	—	100
Undetected/Uncorrected Error Escape Rate	10^{-13}	—	—

NOTES:

1. Read errors are recoverable by retry or error correction.
2. Data errors are recoverable by error correction methods only.

Table 8. Error Correction Level Selection

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

In typical bubble memory systems, the prevention of the erroneous transfer of data to the host is primary concern, and the actual location of the error (error logging by page) is secondary. This fact is especially true if the error is correctable. Both the prevention of transferring erroneous data and error logging of the page in error, however, can be satisfied by error correction, and you must select the error correction level that best fits your specific application.

ECC Operation

When error correction is implemented, each page of data written to the MBM contains additional bits for the ECC code. For example, in a single-bubble system like the BPK 72, the page size decreases from 68 bytes to 64 bytes per page when error correction is implemented. As each page of data is read from the MBM and assembled into the FSA, the FSA's error correction circuitry checks the integrity of the data. When an error is detected (in a completely assembled page), the FSA notifies the BMC (by activating its ERR.FLG/ line) and sets the appropriate bits within its internal status register according to whether or not the error is correctable. The BMC, depending on the level of error correction selected and the nature of the error (correctable or uncorrectable), either issues a command to the FSA to continue the transfer (transparent to the host) or interrupts the host and waits for additional instructions before proceeding.

The point at which the data is transferred between the FSA and the BMC's FIFO is the key to understanding the difference among the three levels of error correction. Once this timing is understood, it will become easier to see how the levels differ in terms of the interrupts generated and the intervention required by the host (software driver) during the transfer. Note that while the BMC generates an interrupt when an error is detected, it is not mandatory for the host to support interrupts; the host can read the BMC's Status Register at the completion of command execution to determine the nature of the error condition.

During a bubble read operation with error correction enabled, the FSA senses and formats the data and places the data in its two 270-bit serial FIFOs (one FIFO for each channel). The data is read in the fully buffered mode where a full page (512 bits) plus the additional 28 ECC bits are read into the two FIFOs before any data is transferred to the BMC's FIFO. By fully buffering the data, the FSA can detect an error and notify the BMC before any data is transferred. If no error is detected by the FSA, the contents of the FSA's FIFOs are written to the BMC's FIFO while the next page from the MBM is being read into the FSA (note that the 28-bits of ECC code are never transferred to the BMC's FIFO). In order to correct an error once it has been detected, the FSA cycles the data through its error correction network. Once the data is cycled, the "corrected" data is either automatically transferred to the BMC (transparent to the host) or the data is held in the FSA FIFOs awaiting further instruction from the host. As an alternative to using the "corrected" data (error correction level dependent), the page in error could simply be reread when an error was reported since a majority of the errors encountered are "read" errors. The following paragraphs describe ECC operation during each of the three levels of error correction.

Level 1

Level 1 is the minimum level of error correction and is used only when the host system is concerned with maintaining bubble data integrity. As an example of this type of application, consider a bubble-based operating system that is downloaded into user RAM for execution. With this application, the primary concern is that the entire operating system is transferred correctly rather than where the error occurred. As will be seen, Level 1 is well suited to applications where go/no-go types of data transfers are required.

If an error is detected during Level 1 operation, the FSA activates its ERR.FLG/ line to the BMC. The BMC, in response, automatically issues a Read Corrected Data (RCD) command to the FSA which causes the FSA to cycle the data through its ECC network and update its status register, and then to transfer the data to the BMC. If the soft error was correctable, valid data would have been transferred to the BMC; the BMC would increment its 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 soft error was beyond the capability of the FSA's error correction circuitry (i.e., an uncorrectable error), 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, and does not increment its Address Register. Since the host is aware that an uncorrectable error has occurred, the proper response is to repeat the entire read operation (by reloading the parametric register and reissuing the read command) since the erroneous page already has been transferred to the host.

Note that since the BMC does not increment its Address Register after reading the "uncorrectable error" status from the FSA, it is possible to perform page-specific logging of uncorrectable errors.

Level 2

Level 2 differs from Level 1 in that the transfer of erroneous (uncorrectable) data to the BMC is prevented and successive retries of the page in error are possible since the Address Register is not incremented. As with Level 1, correctable errors are transparent to the host system.

When a soft error is detected during Level 2 operation, the BMC automatically issues an Internally Corrected Data (ICD) command to the FSA.

In response to this command, the FSA cycles the data through its error correction network and updates its status register to indicate if the error is correctable or uncorrectable, but does not transfer the data to the BMC. The BMC, in turn, 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 its Address Register to the next page address to allow the read operation to continue. Like Level 1 operation, the transfer of the corrected page is transparent to the host; the subtle difference between Level 1 and Level 2 is the time required to execute since the data is cycled through the error correction network twice (first by the ICD command and again by the RCD command). While the second correction cycle does not change the data, it does, however add approximately 350 milliseconds to the transfer operation.

If the soft error is uncorrectable, command execution is halted on the page in error and the BMC interrupts the host system. When interrupted, the host system can read the BMC's Address Register to determine the page in error and can issue a subsequent Read command (without reloading the parametric register) to retry the page. If, when the page is reread, the error does not recur (i.e., if the uncorrectable error is a "read" error), command execution continues with the next page. If successive retrys are unsuccessful (i.e., if the uncorrectable error is a "data" error), the page most likely will have to be rewritten. The host system, however, can examine the erroneous page by issuing an RCD command to the BMC to transfer the uncorrectable data from the FSA. Following the transfer of the erroneous page, the BMC again will interrupt the host and will not increment its Address Register. Note that the uncorrectable data transferred to the BMC will not necessarily match the erroneous data originally read from the MBM since the FSA's error correction circuitry attempts to correct the data.

Level 3

Level 3 offers the most complete means of error handling and, at the same time, is the most software intensive level since the host system is interrupted when either a correctable or uncorrectable error is encountered. Accordingly, error logging may be performed on pages containing correctable as well as uncorrectable errors, and an unlimited number of retrys can be attempted on the erroneous page. As with Level 2, the transfer of erroneous data to the BMC can be prevented.

When a soft error is detected during Level 3 operation, the BMC automatically issues an ICD command to the FSA. Like Level 2 operation, the FSA cycles the data through its error correction network and updates its status register, but does not transfer the data to the BMC. The BMC, in turn, reads the FSA status register, updates its own Status Register, and interrupts the host system even if the error is correctable. When interrupted, the host system must examine the BMC's Status Register to determine if the error is correctable or uncorrectable and can log the address of the page in error by reading the BMC's address register (the Address Register is not incremented). Note that it is not necessary for the host to support interrupts as the host can continuously poll the BMC's Status Register for an error interrupt.

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. Note that by retrying pages with correctable errors, the host system can distinguish between "read" and "data" errors since consistently correctable errors on a page indicate a "data" error (a "read" error would not recur when the page was reread).

If the error is uncorrectable, the host system only would retry the erroneous page by issuing additional Read commands and, if successive retrys were unsuccessful, would have to rewrite the page. As with Level 2 operation, the host system can issue an RCD command to transfer the uncorrectable data from the FSA.

Status Register

When using error correction, the host system can read the BMC's Status Register 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.

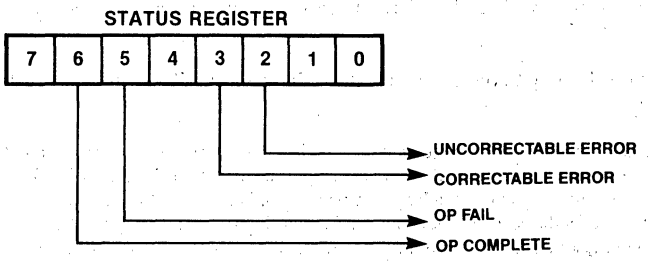


Table 9 lists the possible status indications that could occur when using error correction.

Table 9. 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.

DRIVER DESIGN

This section contains specific software interface examples that outline important considerations associated with the various ways in which a bubble system can be operated. As will be explained, command execution can be performed either in an interrupt driven mode or in a polled mode irrespective of the data transfer mode (polled, interrupt-driven, or DMA) selected to effectively provide the six unique operating modes shown in Figure 2. Since both polled and interrupt driven command execution are common to all three data transfer modes, details concerning command execution are discussed prior to examining specific examples of each data transfer mode. Before you begin to design your software driver, it is recommended that you thoroughly understand the information presented in this section.

Command Execution

To better understand the software driver's responsibilities, it is helpful to separate command execution into two phases; a command phase and a result phase. During the command phase, the driver generally (command dependent) loads the parametric registers, issues the desired command, then verifies that the command has been accepted; during the result phase, the driver determines the success or failure of the command issued either by polling the BMC's Status Register (polled mode) or through an interrupt service routine (interrupt-driven).

- Polled Mode. After loading the parametric registers (if required), the software driver issues the desired command, checks to see if the command was accepted, and then continuously polls the BMC's Status Register for an Op Complete indication.
- Interrupt-Driven Mode. After loading the parametric registers (if required), the software issues the desired command, checks to see if the command was accepted, and then waits for the interrupt to occur at successful command completion. Note that the interrupt Enable (Normal) bit must be set in the Enable Register to allow the BMC to generate the interrupt at successful command completion. Also, since an interrupt is not generated if the intended operation fails, additional provisions must be included to avoid "hanging" the system while waiting for the command completion interrupt. Error interrupts are reported according to the settings of the three "error correction" bits in the Enable Register.

Another important fact to note is that the BMC does not support the typical two-way handshaking common to most interrupt-driven peripheral devices and that interrupts from the BMC are cleared (or acknowledged) either when a subsequent command is issued or when the Register Address Counter (RAC) is written with the modifier bit (bit D5) set to "one" and with bits D3 through D0 set to "zero"; interrupts are not cleared by reading the BMC's Status Register.

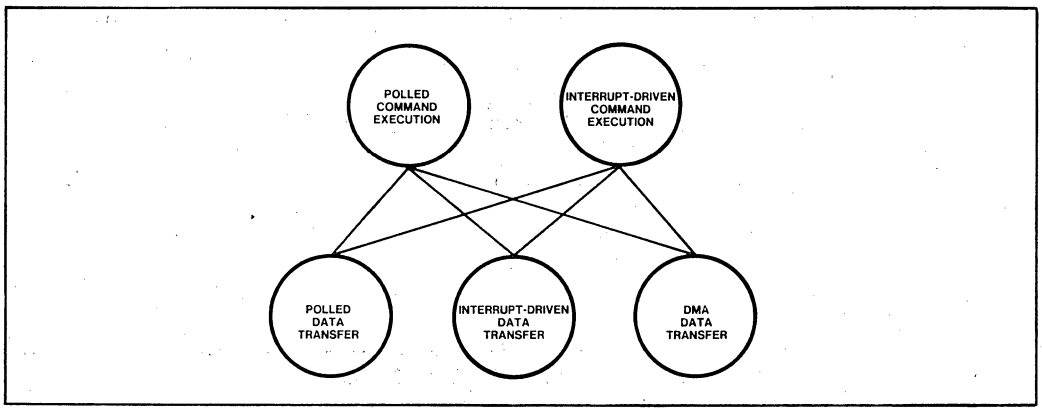


Figure 2. Bubble System Operating Modes

Considerations For Polled Command Execution

The operation of the BMC (and in particular, the BMC's Status Register) during polled command execution is the key to software driver development. Figure 3 shows the activity of the BMC at various stages during a typical command execution sequence. The disussion of the BMC's Status Register is centered around the status bits associated with command execution (i.e., the BUSY, OP COMPLETE, and OP FAIL status bits) although as will be explained, additional Status Register bits can have an effect on the state of the BUSY bit.

Before the command is sent at time T0, the BUSY bit is clear and, in fact, must be clear in order for the BMC to accept a new command (other than an Abort command). Sometime between T0 and T1, the BUSY bit is set to indicate that the command has been accepted and that command execution has begun. Note that all software examples in the remainder of this chapter include a check to ensure that the BUSY bit has been set after a command has been issued.

If the Status Register is examined between T1 and T2, the status byte value will be 80H (BUSY bit set). Depending on the command issued, the FIFO AVAILABLE bit may set at T2 if it is necessary for the host to initiate the transfer of data (see "polled data transfers" later in this section). Whether or not the FIFO AVAILABLE bit is set, the only normal status byte values between T2 and Tn-1 are 80H or 81H (BUSY and FIFO AVAILABLE).

At time Tn (completion of the command), the Status Register indicates the success or failure of the command. Generally, the BUSY bit is cleared at this time (indicating that command execution has terminated); an exception occurs if the FIFO still contains 22 or more bytes of data at command completion (i.e., when the BMC completes its internal microcode routine). During execution of read commands, it is possible for the host to leave data within the FIFO, and if 22 or more bytes are remaining, the BUSY bit will remain set even though command execution has been completed.

If the command is executed successfully and if the FIFO is empty, the status byte value will be either 40H (OP COMPLETE) or 42H (OP COMPLETE and PARITY ERROR). If data was loaded into the FIFO during the interval between T2 and Tn-2, the possible status byte values will be:

41H or 43H if the FIFO contains less than 22 bytes
C1H or C3H if the FIFO contains 22 or more bytes

In the case of unsuccessful command execution, several additional status byte values again are possible depending on the command being executed. Generally, the OP FAIL bit is set in the Status Register along with additional bits that indicate the nature of the failure (e.g., TIMING ERROR or PARITY ERROR). As with successful command execution, the BUSY bit is cleared when command execution is completed unless the FIFO still contains 22 or more bytes.

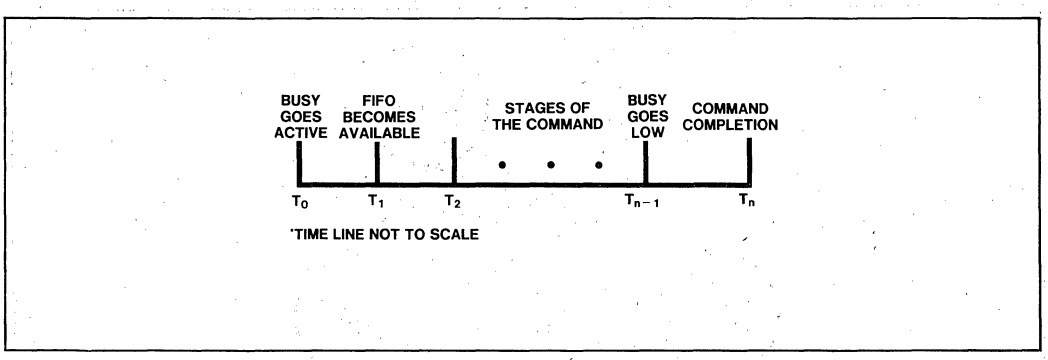


Figure 3. Command Execution Stages

Considerations For Interrupt-Driven Command Execution

As previously mentioned, when the Interrupt Enable (Normal) bit is set, an interrupt occurs only when a command is executed successfully as indicated by the setting of the Op Complete bit in the Status Register. If the Interrupt Enable (Error) bit is not set in the Enable Register and an error occurs that causes the operation to fail, an interrupt will not be generated. The user system must take this fact into account and provide either a fail-safe timer in hardware or a timeout counter in system software to guard against the possibility of "hanging" the system while waiting for an interrupt.

The following possible conditions are applicable to interrupt-driven operation:

1. After issuing a command, the host processor halts and waits for the interrupt to occur. Under this condition, the system will hang if the expected interrupt never occurs. A fail-safe timer connected to a low-level input of an interrupt controller (e.g., an Intel 8259A) would eliminate this problem.
2. In a "true" interrupt driven system where the host processor performs other tasks while the bubble command is being executed, the fail-safe timer in hardware again would be used although it is possible to use a software timeout counter. When operating in the non-DMA data transfer mode (polled or interrupt-driven data transfers), an interrupt service routine would be used to complete the data transfer and then either would wait for the command interrupt or would continuously poll the Status Register until command execution was successfully completed or until a software counter timed out (indicating unsuccessful command execution). However, devoting an excessive amount of time to polling by the host defeats the intention of an interrupt-driven system and reaffirms the need to provide a "watchdog" timer in hardware. In the DMA data transfer mode, the logical approach to detecting when execution of a command has failed is to provide a fail-safe timer in hardware that generates a timeout interrupt to the host.

An additional consideration common to all three data transfer modes is the fact that when an interrupt occurs in response to the successful execution of a command, data still may be present in the BMC's FIFO. If the interrupt service routine immediately acknowledges receipt of the command completion interrupt at the source (i.e., at the BMC), any data remaining in the BMC's FIFO when the interrupt is cleared would be destroyed. To eliminate this potential problem, an intermediate level of interrupt control would be required (again an Intel 8259A) to allow the interrupt routine to complete the transfer. The details described in the "Interrupt-Driven Data Transfer Mode" section will help to clarify the interrupt service routine requirements necessary to avoid leaving any data in the FIFO at command completion.

Data Transfer Modes

As mentioned at the beginning of this section, both polled and interrupt driven command execution can support any of the three data transfer modes. Regardless of the data transfer mode used, the basic operation of the BMC is the same for each data transfer mode. During all data transfers (i.e., during execution of a Read Bubble Data or Write Bubble Data command), the BMC continues to transfer pages of data until the page count in the Block Length Register is satisfied. Since the BMC's FIFO cannot hold a complete page of data, the host processor is required to "keep up" with the BMC as it continues to read data from, or to write data to, the MBM until all data has been transferred. At the beginning of read/write command execution, a "seek" operation is performed to locate the designated page. Once the addressed page is located within the MBM, the read data transfer begins (a write data transfer does not begin until at least two bytes of data have been loaded into the FIFO).

Polled Data Transfers

The polled data transfer mode is the most time-consuming in terms of processor overhead while at the same time is the easiest mode to implement. To support the polled data transfer mode, it is essential that the Status Register bit definitions be clearly understood and, in particular, the function of the FIFO AVAILABLE bit, since the interpretation of the bit changes according to the direction of the transfer.

From an operational viewpoint, the FIFO AVAILABLE bit acts as a gate for the FIFO-handling software. During a bubble write operation, if the FIFO AVAILABLE bit is set, there is room for additional data, and if the FIFO AVAILABLE bit is clear, the FIFO is full. During a bubble read operation, if the FIFO AVAILABLE bit is set, data has been placed in the FIFO by the BMC, and if the bit is clear, the FIFO is empty.

The BUSY bit indicates when the controller is in the process of executing a command. As previously described, the BUSY bit is set within a few microseconds following receipt of the command and remains set until the operation is completed (successfully or unsuccessfully). Remember that since the BUSY bit is internally gated with the BMC's DRQ output signal, it is possible during a read operation to obtain such logically exclusive status indications as BUSY and OP COMPLETE if the host fails to empty the FIFO following comand execution (during non-DMA data transfers, the DRQ signal acts as a "half full/half empty" flag for the BMC's FIFO).

Later in this section, a basic polled data transfer mode read/write routine is flowcharted. Note that to allow the status byte to be considered valid only after the BUSY bit is cleared and to avoid concurrent setting of the BUSY and OP COMPLETE bits at command completion, a running byte counter is implemented in software to ensure that all bytes have been removed from the FIFO. While the FIFO AVAILABLE bit alone can be used to gate the data to or from the FIFO, a byte counter is required to determine when the specified number of bytes has been transferred. Also note that a FIFO Reset command is sent prior to issuing the Write Bubble Data command as a "safety measure." Although issuing the FIFO Reset command is not mandatory, resetting the FIFO is recommended if there is any doubt regarding the state (emptiness) of the FIFO prior to initiating command execution.

Interrupt-Driven Data Transfers

The interrupt-driven data transfer mode requires less processor overhead than the polled data transfer mode (the polling wait time is eliminated) and also allows the data to be transferred in blocks rather than in individual bytes. The actual data transfer rate is fixed; the benefits of this mode only can be realized if the interrupt service routine is efficient (fast) enough to allow additional time for processing other tasks and is based on the host processor's execution speed. Accordingly, if the interrupt-driven data transfer mode is selected, make sure that the additional hardware and software required provide the desired performance increase.

The interrupt-driven data transfer mode is based on the fact that the BMC's DRQ line doubles as a "FIFO half full/FIFO half empty" indicator when the BMC is not in the DMA Mode. Physically, the DRQ line is connected either directly to the host's interrupt input or to an interrupt controller as the interrupt source for the data transfer. When an interrupt occurs, the host processor transfers a block of data to or from the FIFO in a single burst.

The recommended size of the data block transferred is 22 bytes. This block size was selected because transferring 22 bytes, and not 20 bytes (half of the FIFO), accounts for the two additional bytes held in the BMC's internal FIFO input and output latches while also optimizing the buffering capability of the BMC's FIFO. Since 22-byte block transfers rarely are exact multiples of the total number of bytes transferred, an efficient method must be devised to transfer the last remaining bytes of the transfer. While several methods are possible, the following method ensures that an interrupt is issued to transfer all of the "remainder" bytes (i.e., less than 22 bytes never will be "left" in the FIFO to prevent the DRQ line from going active and initiating the last block transfer).

As an example, assume that a two-page (128 byte) read or write data transfer is to be performed. The initial DRQ interrupt indicates that at least 22 bytes should be transferred. However, the first block transferred would be 18 bytes that correspond to the "remainder" bytes (128 mod 22 is 18). By first transferring 18 bytes, five subsequent interrupts occur (each initiating a 22-byte block transfer) to guarantee that no additional polling of the Status Register is required to transfer any "leftover" bytes. The calculation of the remainder count used for the first execution of the interrupt service routine is the responsibility of user software.

DMA Data Transfer Mode

In terms of a bubble system, direct memory access or DMA is the transfer of data between system memory and the BMC without host processor assistance or intervention. Since host processor involvement with the actual data transfer is not required, the overhead associated with software controlled (non-DMA) data transfers is eliminated.

Overall system performance is the primary factor for considering the DMA data transfer mode. That is, if the overhead associated with the available software-controlled data transfer modes degrades system performance to an unacceptable level, the implementation of DMA may prove to be the ideal solution. However, to support the DMA data transfer mode, additional hardware is required (e.g., a programmable DMA controller).

From a system perspective, the DMA hardware definition will dictate the software requirements. In a fixed system, discrete devices may be configured to perform a majority of the data transfer operations, including memory addressing, transfer direction, and terminal count (i.e., the number of bytes to be transferred).

The implementation of a programmable DMA controller such as an Intel 8257 or 8237 provides additional flexibility by allowing DMA to be performed under program control. Since it is not possible to describe all possible configurations, the BMC's operation during DMA transfers is described followed by a specific software example using an Intel 8257 DMA Controller.

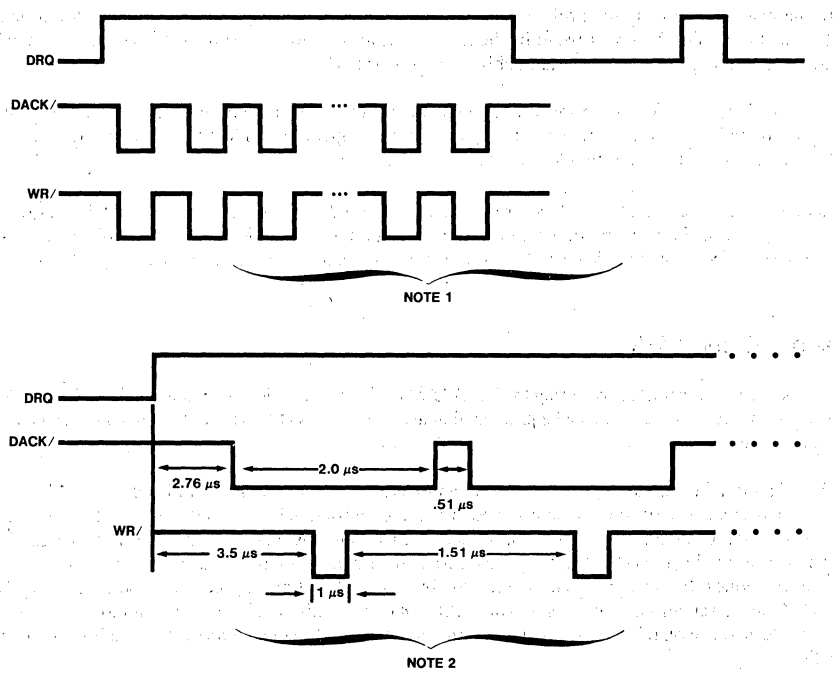
BMC Operation During DMA

The DMA data transfer mode is selected by setting the DMA ENABLE bit in the BMC's Enable Register. In the DMA mode, the BMC supports a standard two-way handshake protocol that uses the BMC's DRQ (Data Request) output signal to interact with the DMA hardware.

To initiate a DMA data transfer (assuming that the DMA hardware has been properly initialized), the BMC's parametric registers are loaded and a read or write command is issued to start the transfer. When the BMC is ready to transfer a data byte to or from system memory, it activates its DRQ output signal. The DMA hardware responds to DRQ by first gaining control of the system bus, then activating the DACK/ signal to the BMC, and finally making a read or write request to the BMC. The BMC responds to the request by placing a data byte on the bus (read operation) or accepting the byte on the bus (write operation). The DRQ signal remains active as long as the BMC either has additional bytes to transfer to system memory (and the FIFO contains at least one byte of data) or expects additional bytes from system memory (and the FIFO contains at least one empty location).

Referring to figure 4, when a Write Bubble Data command is issued, the BMC continuously requests (holds DRQ active) bytes from system memory until the FIFO is filled (i.e., data initially is transferred in a "burst" mode). Once the FIFO is filled, the BMC issues subsequent DRQ requests on a byte-by-byte basis until the last bytes have been transferred. When command execution is complete, the BUSY bit is cleared and the OP COMPLETE bit is set in the Status Register. If the INTERRUPT ENABLE (NORMAL) bit is set in the Enable Register, an interrupt is generated.

When a Read Bubble Data command is issued, the BMC activates DRQ only after the first data byte has been assembled and placed in the FIFO. While the DMA hardware is responding to the DRQ, additional bytes enter the FIFO at a minimum rate of 80 microseconds per byte (the single-MBM system transfer rate; the transfer rate increases according to the number of MBMs operated in parallel). Typical DMA response times usually are fast enough to take each byte before the next byte arrives, and data is transferred on a byte-to-byte basis.



NOTES:
1. THE DATA TRANSFER SEQUENCE INITIALLY OPERATES IN A "BURST" MODE (TYPICALLY 44 TO 46 BYTES ARE TRANSFERRED).
2. THE TIMING SHOWN IS ONE EXAMPLE OF VALUES OBTAINED DURING NORMAL OPERATION. THE VALUES MAY VARY IN YOUR SYSTEM PROVIDED THEY ARE WITHIN SPECIFICATIONS.

Figure 4. DMA Handshake Timing During Read/Write Command Execution

START-UP PROCEDURES

Whenever power is applied to the bubble memory system, a powerfail reset circuit is activated to satisfy specific power/timing sequences required by the BMC. All bubble system designs must include a powerfail reset circuit to ensure proper initialization of the bubble system.

The primary function of the powerfail reset circuit is to ensure that the bubble memory system powers up correctly. A built-in hardware time delay allows the BMC to complete its internal power-up sequence prior to enabling the additional support components; a software time delay is needed before any commands can be issued to the BMC. Following the delay, the first user communication with the BMC must be an Abort command whether the power-up is a cold start (application of power) or a warm restart routine (a RESET/ pulse applied to the 7220-1 BMC). Note that the status register should not be considered valid until the Abort command has been issued. A complete power-up flowchart, including initialization, is shown in figure 5.

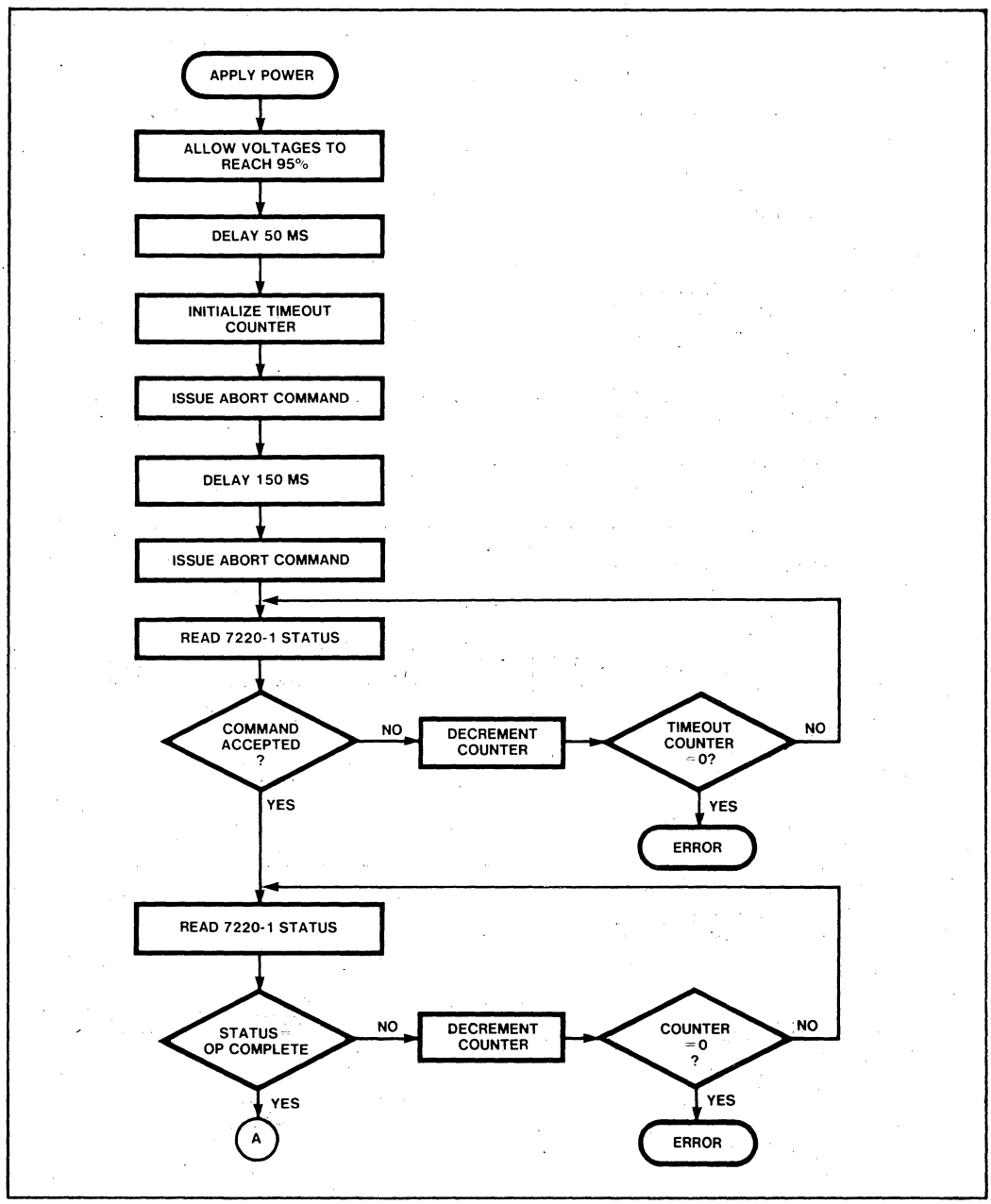


Figure 5. Power-Up Sequence (Polled Command Execution)

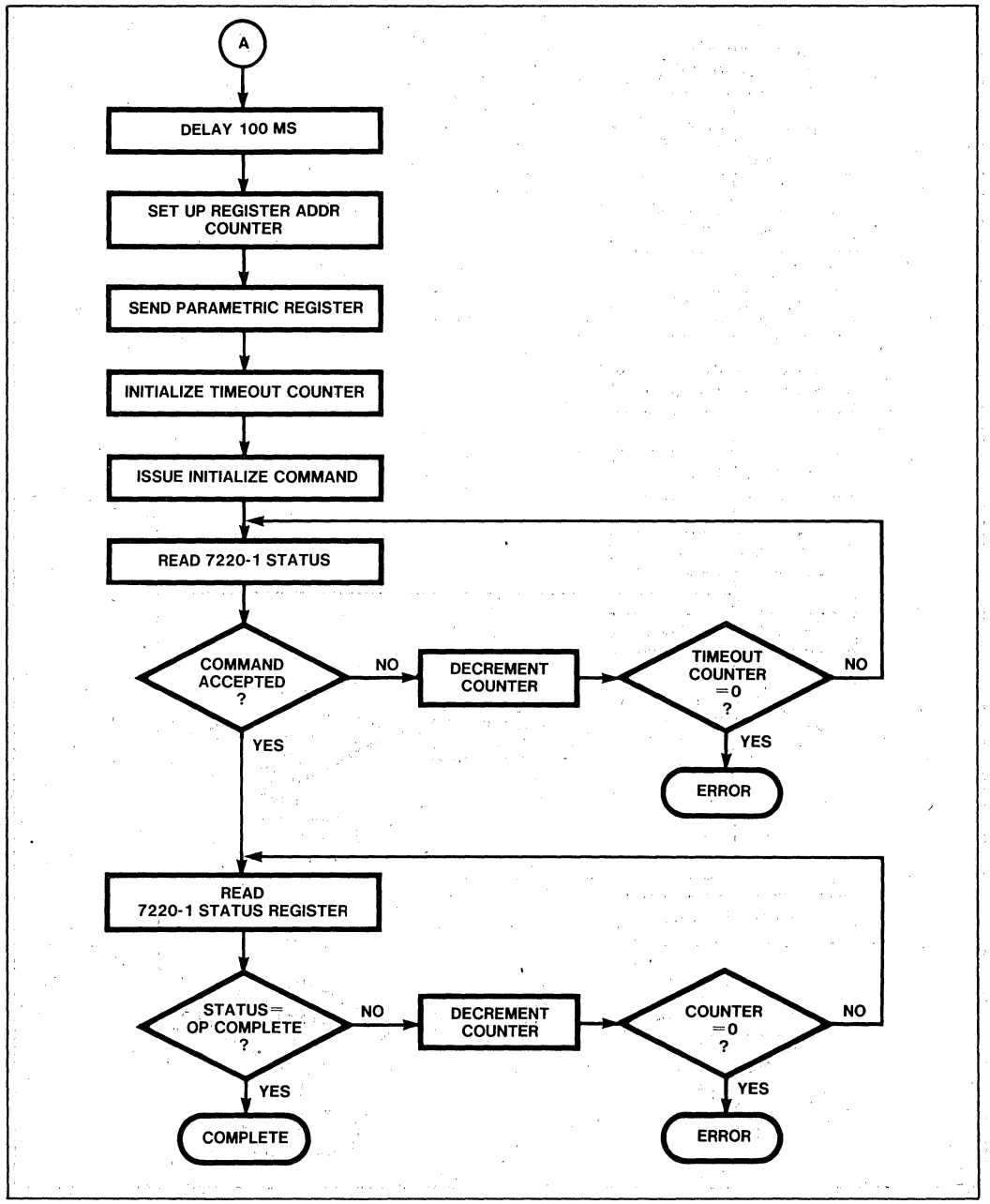


Figure 5. Power-Up Sequence (Polled Command Execution) (Cont.)

Initialization

Following successful execution of an Abort command, the user must initialize the bubble memory system using the Initialize command. The Initialize command requires that the parametric registers first are loaded with specific values. The software flowchart (figure 5) shows one example for polled command execution without error correction for a one-bubble system (i.e., the BPK 72 or iSBX 251). The Block Length Register always must be loaded with the values shown while the MBM Group Select bits in the Address Register always must select the last MBM in the system. For your particular application, set the appropriate bits in the Enable Register prior to issuing the Initialize command. Note that error execution mode changes require re-initialization of the bubble system.

The flowchart example polls the Status Register to see whether or not the initialization was successful. The OP COMPLETE bit indicates success. If an initialization problem occurs, the TIMING ERROR and OP FAIL bits will be set in the Status Register.

In an interrupt-driven system, the interpretation of the INT (interrupt) line depends on the setting of specific bits in the Enable Register. If INTERRUPT ENABLE (NORMAL) is set in the register, on the *successful* completion of command execution, the INT line will activate on the user interface. If INTERRUPT ENABLE (ERROR) is set in the register, the INT line will activate when an error condition occurs (in this case a TIMING ERROR). The user should devise interrupt service routines that can differentiate between the two interrupt sources by polling the Status Register. It is apparent from this discussion that a system that relies solely on interrupts may find it necessary to incorporate a watchdog timer (timeout counter interrupt) depending on the selection of the interrupt enable bits. A timeout counter avoids the possibility of never receiving an interrupt if a command is *unsuccessful* (this condition is possible if only the INTERRUPT ENABLE (NORMAL) bit is set).

If a system uses the BMC's DRQ signal for data transfers (DMA or interrupt mode), special precautions are necessary during Intialization. First, it is recommended that the DMA Enable bit in the Enable Register be disabled (i.e., set to 0) for the Initialize command. Once the Initialize command is successfully completed, the DMA Enable bit is enabled for the subsequent DMA data transfer.

When the BMC is in the non-DMA mode, the DRQ pin acts as a half full/half empty flag for the BMC's FIFO. Since the FIFO is used to temporarily store bootloop data during execution of the Initialize command, the DRQ pin is toggled. User-written software should therefore take appropriate precautions with respect to the DRQ pin during initialization (i.e., mask all interrupts, disable DMA controllers, etc.).

The Initialize command establishes a "working subset" of the bootloop information from the bubble and places this information into the bootloop registers of the FSA. A masking algorithm within the BMC reduces the total of 320 possible good storage loops to a working subset of 270 or 272 good loops (error correction dependent).

If the bootloop information in the MBM is incorrect, the system can inadvertently appear to be initialized correctly. As a one-time check to ensure that the system has been initialized correctly, simply read the bootloop registers from the FSA and count the number of "1's."

The count should yield:

- 270 "1's" out of 320 with error correction implemented.
- 272 "1's" out of 320 without error correction.

NORMAL OPERATION EXAMPLES

The following software flowcharts outline the important considerations and provide examples of each mode of operation previously discussed. First, figures 6 and 7 show the two possible command execution techniques. Note that in each case, the appropriate additional software must be inserted according to the data transfer mode selected. Figures 8 through 10 detail each data transfer mode. For any commands that do not involve the transfer of data (i.e., Abort, FIFO Reset, etc.), no additional software is required to be inserted in the command execution examples.

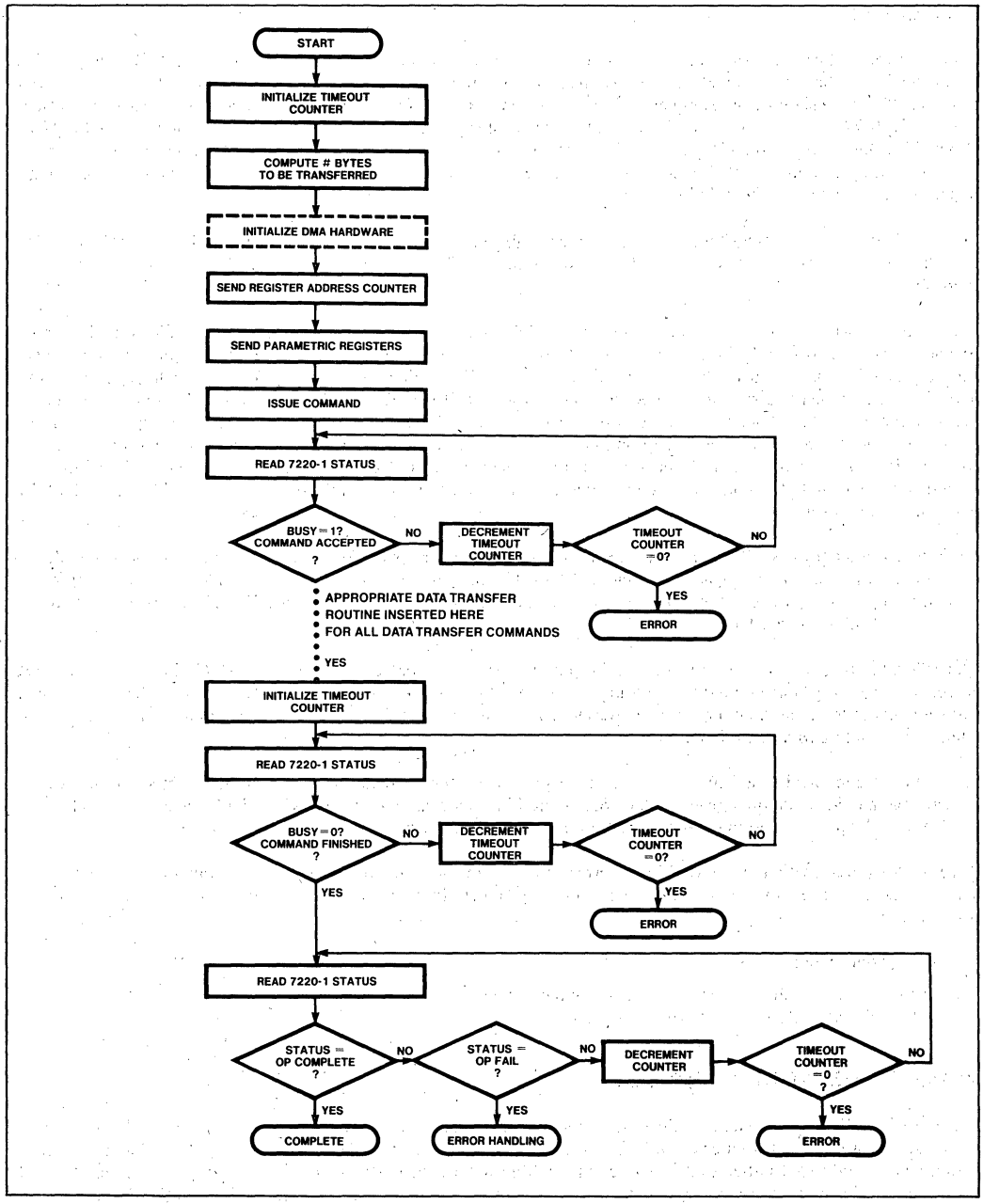


Figure 6. Polled Command Execution Routine

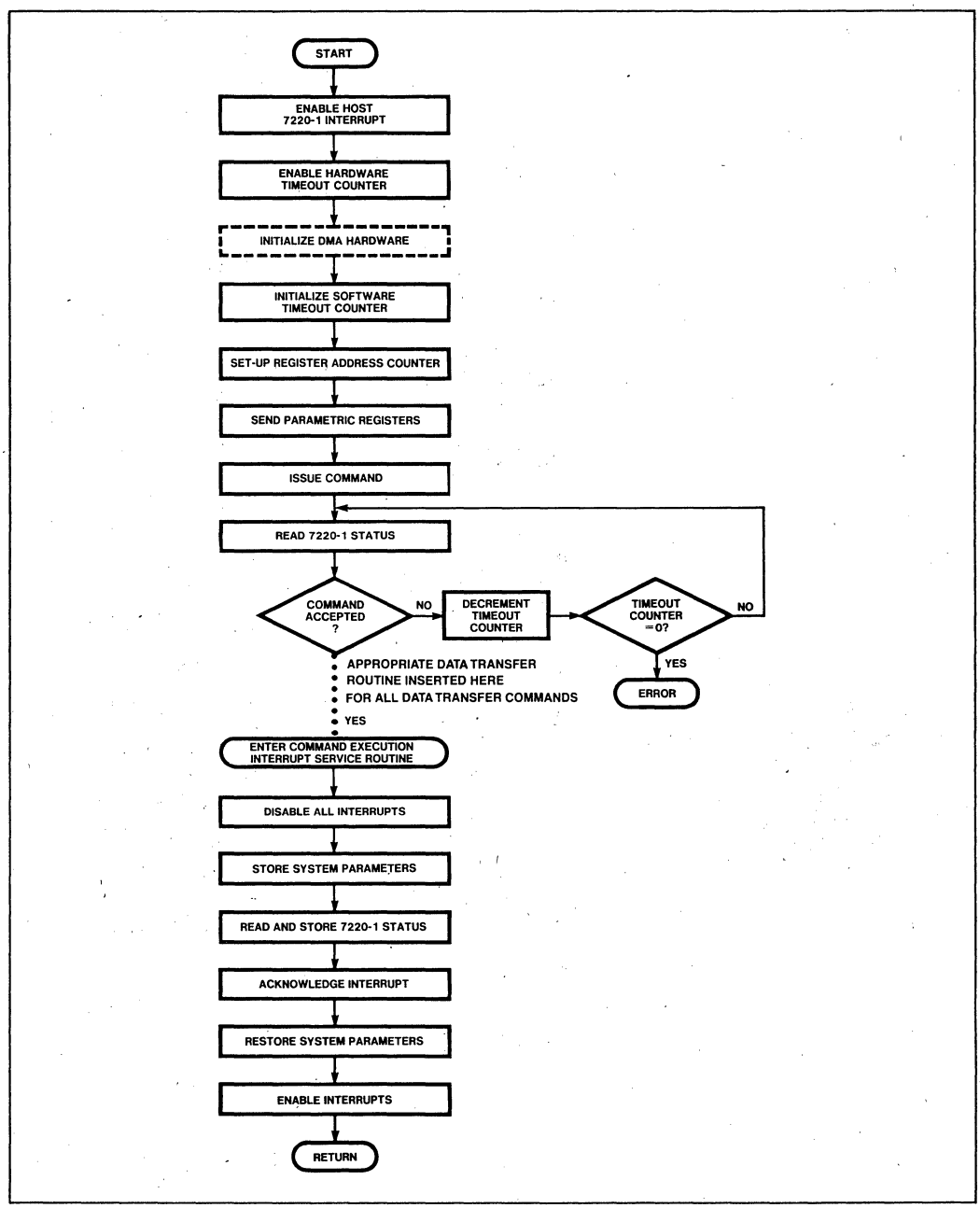


Figure 7. Interrupt-Driven Command Execution Routine

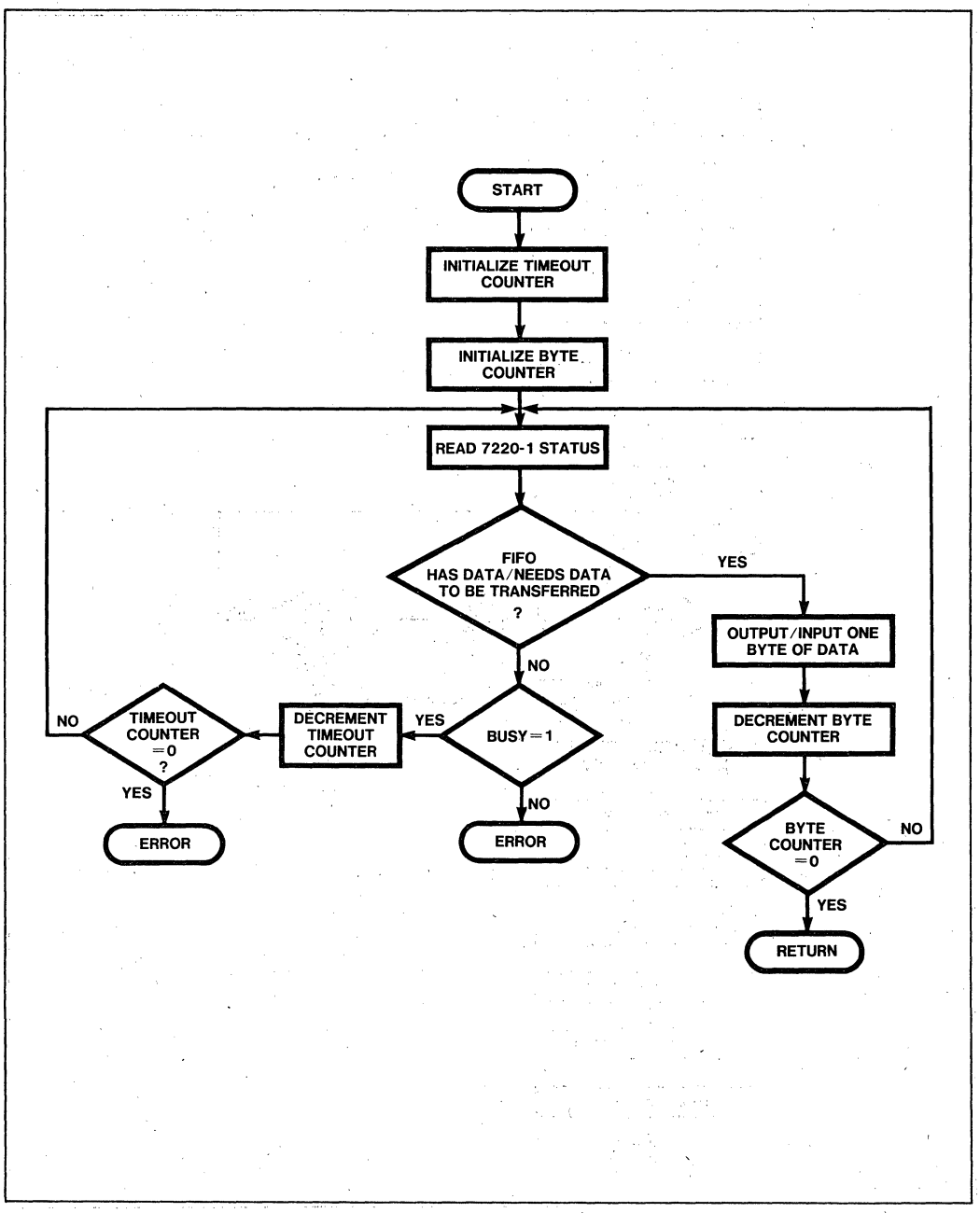


Figure 8. Polled Data Transfer Routine

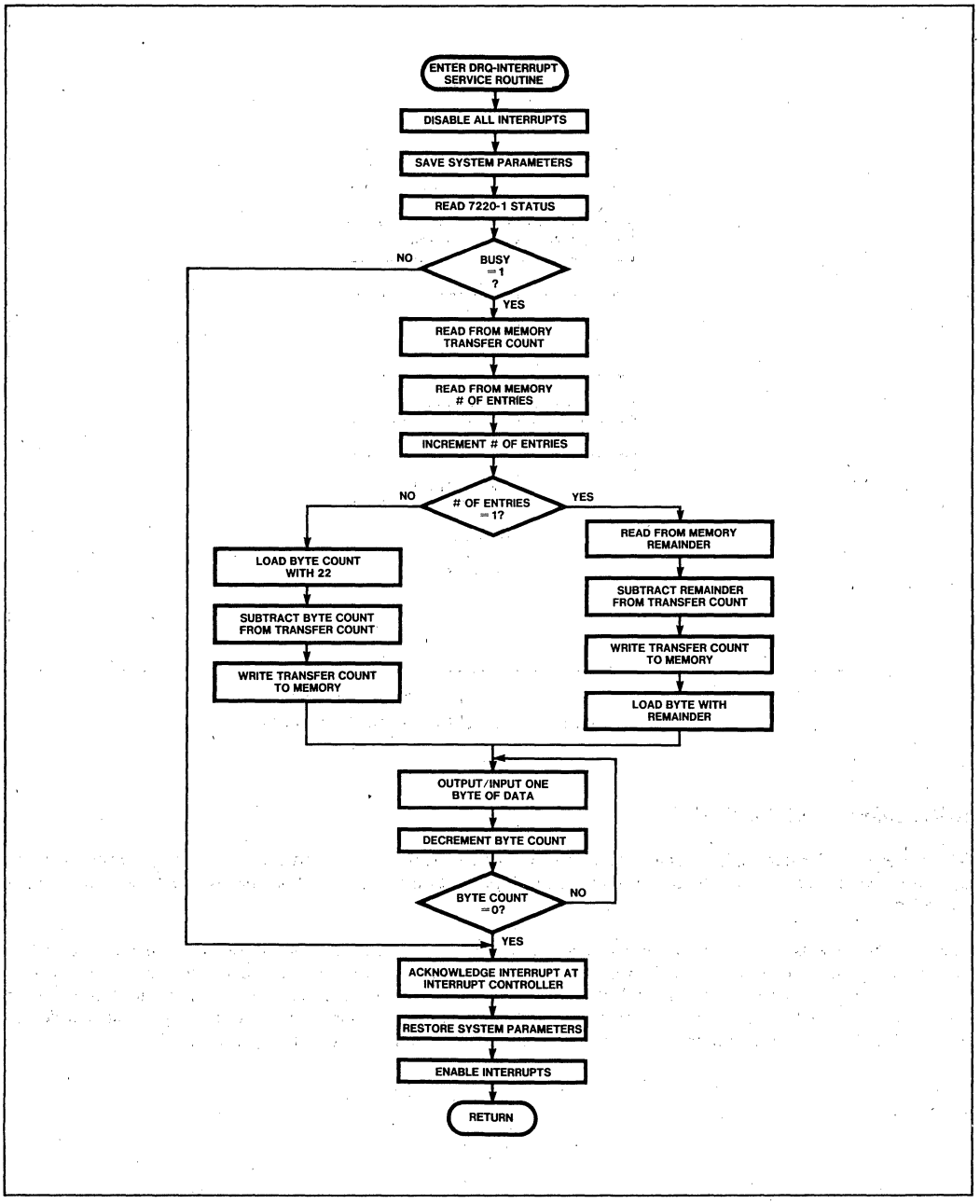


Figure 9. Interrupt-Driven Data Transfer Routine

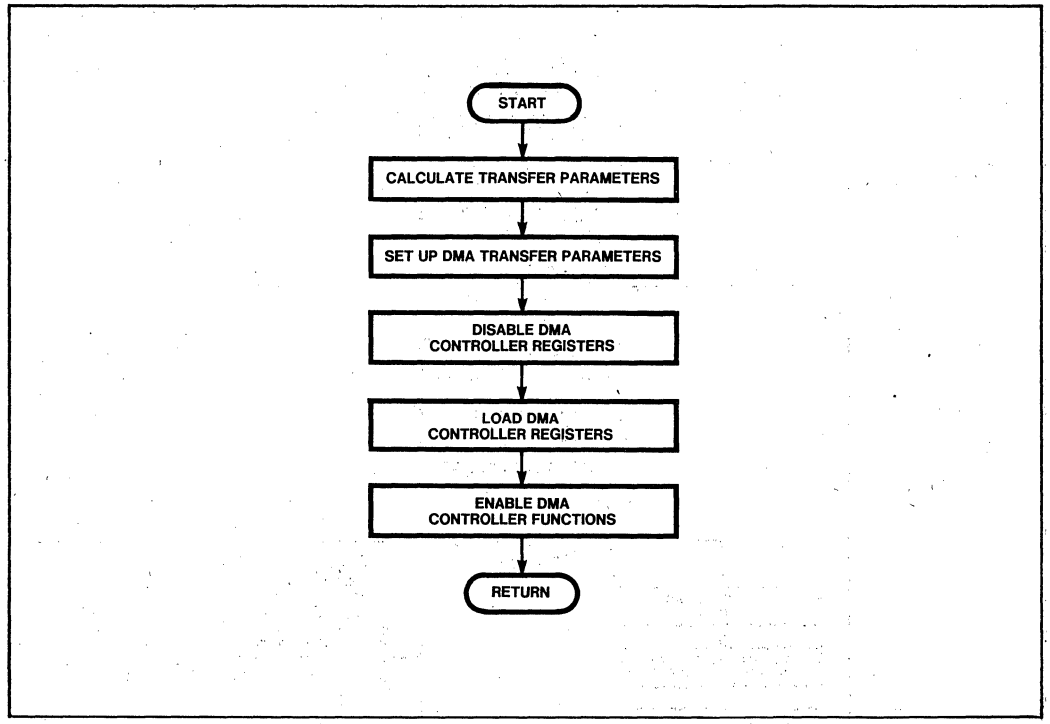


Figure 10. DMA Hardware Initialization Routine (Assumes an Intel 8257 DMA Controller)

SHUT-DOWN PROCEDURES

The power down procedure is the same regardless of whether the user voluntarily powers down the bubble memory system or power is inadvertently lost. The only special precaution is to ensure that the voltage decay rates are not exceeded.

The 7230 Current Pulse Generator contains a special powerfail 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 rotating magnetic field and control signals for the 7110 function generators, in such a manner that no MBM data will be lost or invalidated, provided the voltage decay rates are met. When power is restored, the software implementation details described in the Start-Up procedures should be observed to ensure correct powerfail circuit operation.

intel

APPLICATION NOTE

AP-164

November 1983

Using CMOS Minimizes Bubble Memory Power Consumption

Peggy M. Lammer
Ulmont Smith, Jr.
Applications Engineers
Intel Corporation

ORDER NUMBER: 230826-001

INTRODUCTION

More and more microprocessor systems are becoming portable. Quite frequently, portable systems run on a limited power budget with a battery power supply. Others need to limit how much power they dissipate because their small enclosures dissipate only a limited amount of heat. When designing portables or any piece of processor controlled equipment, consider using bubble memories for mass storage. Bubble memories are solid state, rugged, reliable, and very small; a complete 128 kbyte bubble memory system will occupy less than a 10 cm x 10 cm area of board space. In addition, bubble memories are non-volatile; the memories still maintain their data integrity even if they are powered down to save energy when the processor is not accessing them.

To minimize a low power project's design time, part of the application note is a completely designed CMOS controlled switch that will power-down a bubble system when it is not being accessed. The switch is inexpensive and can reduce your standby power dissipation up to 99%.

OVERVIEW

This application note provides information on low power techniques for bubble memories. All the techniques can be incorporated into your existing bubble memory system with very little effort or expense. A large part of the note focuses on power switching because it is easy to add the extra hardware, and power switching offers the greatest amount of average power savings. For the moment you should know that power switching is supplying the bubble with power only when it is interacting with the host processor and removing the bubble system's power supply when the bubble is not in use.

There are two main parts to the application note. The first will explain why average power dissipation will vary with the frequency of bubble activity. Reading this section will give you an idea of how much power can be saved compared to your present operation.

The second part starts by covering some hardware considerations for power switching bubble memories. It then describes a power switching circuit designed to these considerations. In addition, this section discusses some software techniques and a secondary hardware technique called detector switching that will further reduce average power dissipation.

MEMORY ACTIVITY LEVELS AND POWER DISSIPATION

The bubble memory already reduces its power dissipation by only creating magnetic fields to move the bubbles when data is being accessed. When data is not being transferred, the magnetic fields need not be present, i.e., the coil drive currents which create the fields are removed and less power is dissipated. Table 1 is a break down of power dissipation by component for the Intel one bubble memory system in Figure 1. A system consists of a 7220 bubble memory controller and up to eight BPK-70 memory cells; a cell is a bubble memory and its support ICs.

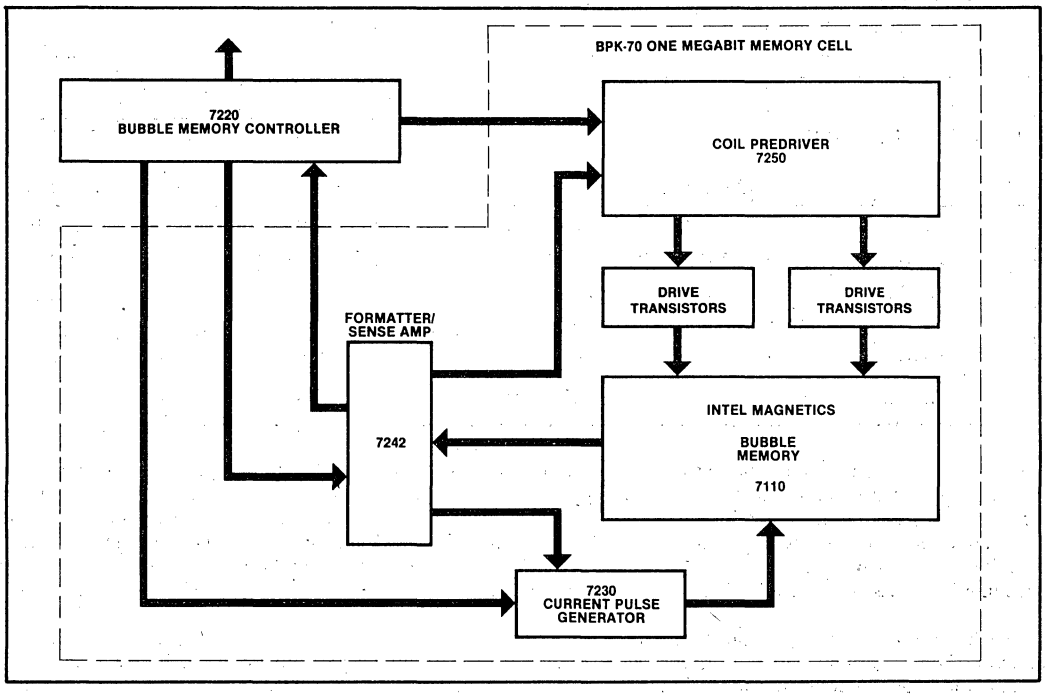


Figure 1. Bubble Memory System Block Diagram

Table 1. Bubble Memory Power Requirements

Configuration BPK70	Capacity (Bytes)	Power (Watts) +5V (Maximum)	+12V (Maximum)	Total Active (Maximum)	Total Active (Typical)	Total Standby (Maximum)	Total Standby (Typical)
1	128K	1.92	4.80	6.72	3.90	3.03	1.55
2	256K	2.79	9.60	12.39	7.30	4.57	2.60
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

Without a power switch, a bubble system not being accessed (in standby mode) will dissipate an average of 1.55 watts, 3.0 W worst case. When the host processor does access the memory, (i.e. the coils have a current running through them) the power dissipation increases to 3.9 W typically, 6.7 W worst case. Obviously, since the bubble system does nothing but dissipate 1.55 W in standby mode, it would be ideal to shut it off and dissipate zero watts. That is the purpose of a power switch. For example, the CMOS controlled switch in this note reduces standby power consumption to less than 25 mW. That is a 99% decrease in standby power consumption compared to worst case.

Figure 2 is a graph showing memory activity levels versus average power dissipation. On the graph, the average amount of time the bubble is active is represented as a percentage of duty cycle. Duty cycle is a ratio of the amount of time the bubble is active to the total time spent in standby and active modes. There are three types of systems plotted on the graph. Case 1 is worst case bubble memory systems. Case 2 represents typical systems and case 3 is typical power switched systems.

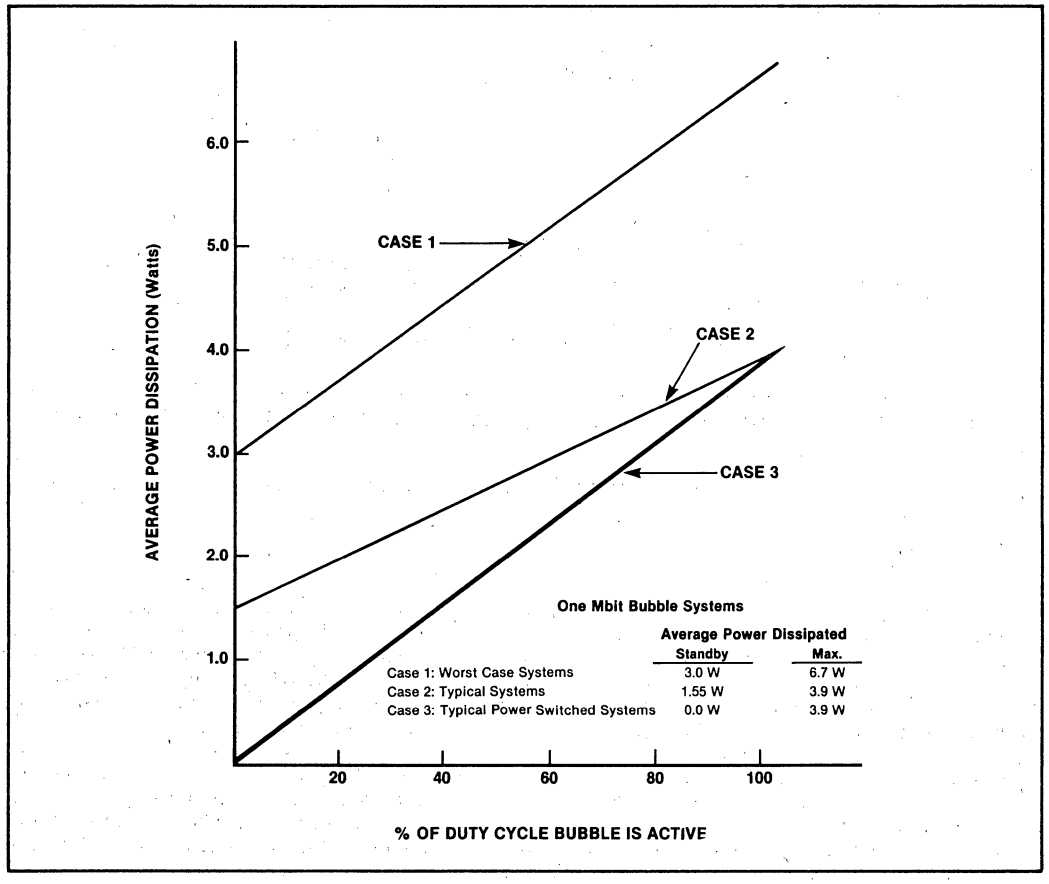


Figure 2. A Comparison of Bubble Memory Activity to the Amount of Average Power Dissipated by the Bubble Memory System

The lowest value on each plot is the power that system dissipates in standby mode. The maximum point is the power that would be dissipated if the processor was dedicated to accessing the bubble memory continuously. To give an example of an application that falls somewhere between the extremes, consider a processor monitoring vintage wine cellars. It runs a program which collects data about the cellars' environmental conditions and stores the information in the bubble memory. Figure 3 shows how power is dissipated by a typical bubble system in two minutes assuming that the program accesses the memory for an average of six seconds when it stores data.

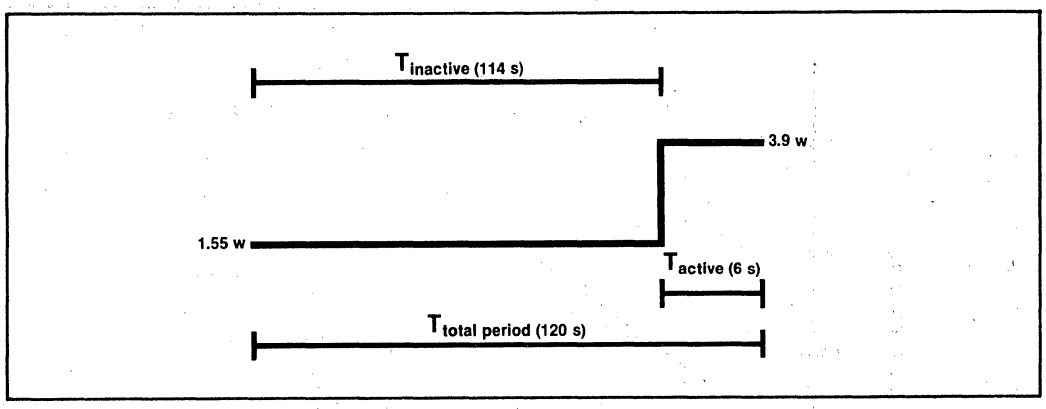


Figure 3. A Duty Cycle Period for a Bubble System with Typical Power Dissipation (Not drawn to scale)

There are two ways to figure how much average power is dissipated in the monitor example. By calculating the average,

(6s (3.90W) + 114s (1.55W)) / 120s = 1.66W

or by using the graph of Figure 2. To use the graph, the duty cycle must be calculated; six seconds is 5% of 120 seconds. After finding 5% on the % duty cycle axis, the amount of power dissipated can be read off the typical system plot. It is also interesting to note how much power would have been dissipated had the bubble system been worst case or power switched. Those values are 3.15 W and 0.35 W respectively; switching time overhead was neglected. If the monitor was to run on a battery for long periods of time, the graph indicates that it would be very worthwhile to add a power switch, about 1.40 W saved per minute over worst case design.

Generally, your duty cycle will be well below 20%. Data acquisition, portable terminal and portable PC applications have long term duty cycles of less than 10%. Even in a high transaction rate application the duty cycle is frequently small. For instance, the processor in a grocery store's point of sale terminal accesses a bubble system for information on items as often as every second. If the information can be read out in 256 byte blocks, then it will take the bubble approximately 63.5 ms to access the data, yet the duty cycle is still only about 6%, (63.5 ms/ 1 s) x 100% ~ 6%.

If your data access and standby times vary, you will want to estimate your average duty cycle so you can use the graph to determine how much power could be saved by switching the bubble. A bubble will take about 41 ms to locate specific memory locations and then it can transfer consecutive pages of 64 bytes approximately every 7.5 ms. Estimate your active unswitched time as follows,

.041s + .0075s (NUMBER OF PAGES TO TRANSFER) = ACTIVE TIME (seconds)

A switched system will have a slightly greater active time due to switching overhead,

Power-up Time + ACTIVE TIME + Power-down Time = ACTIVE TIME (switched)

This incremental difference is not neglible if ACTIVE TIME is very small. For example, the switch documented in the note takes 160 ms to power-up and 48 ms to power-down. If the grocery store's point of sale terminal used that power switch then its active time would change from 6% to 27%. From the graph, the difference in average power dissipation is .76 W, (1.8 W — 1.04 W). Switching is not recommended for systems with average duty cycle periods of less than one second.

LOW POWER DESIGN TECHNIQUES

The most efficient way to reduce your bubble system's average power consumption is to turn off the bubble when it is not being used by the processor. Designing in the simple switch documented in this section will do this quickly. All the hardware considerations necessary to successfully power switch a bubble system are included.

Two other topics will be discussed besides power switching, a secondary hardware switch for the bubble memory's detector and some software considerations including the fastest way to initialize a power switched bubble system and some software considerations such as the fastest way to initialize a power switched bubble system and some energy efficient software techniques for bubble memories in general.

HARDWARE CONSIDERATIONS FOR POWER SWITCHES

This is a basic outline of what is involved in power switching a bubble system. It is not a very complicated process. The CMOS controlled switch or one of your own design can be added in with very little system modification. Figure 4 is a block diagram of how the interface between your bubble memory and host processor will look with the sample switch in place. A table of typical values measured when the switch was validated appears in the appendix.

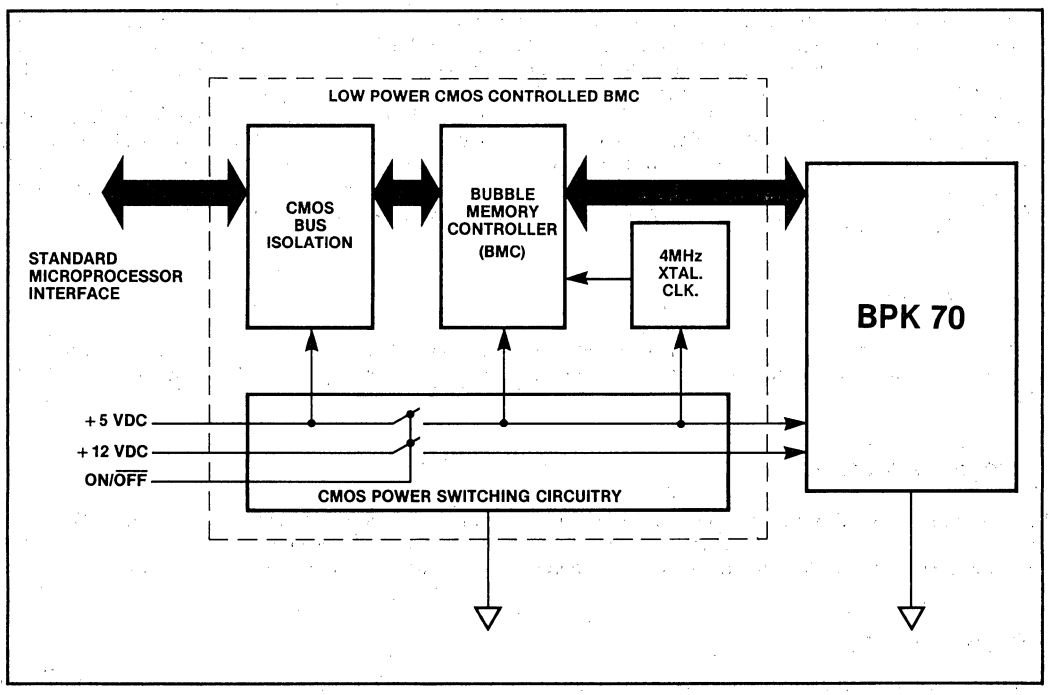


Figure 4. Block Diagram of a Low Power Bubble Memory System

Integrating A Power Switch To A Bubble Memory

There are two power supplies to the bubble system that will need to be disconnected when the bubble is in standby mode. They are +5 Vdc and +12 Vdc respectively. In addition to supplying these voltages within a ±5% voltage tolerance, the switch must also comply with the power-down specifications for Intel bubble systems. The power-down decay rates are;

Power on the 5 Vdc line must not decay at a rate exceeding 0.45 V/ms.

Power on the 12 Vdc line must not decay at a rate exceeding 1.10 V/ms.

These rates must be maintained for 120 μs after the 7220 bubble memory controller (BMC) recognizes that a power-down has occurred; for more information on power-downs, see Application Note 127, Powerfail Considerations for Magnetic Bubble Memories.

In any case, the power switch circuit described in this note maintains a 150 μs delay for itself on either a controlled power-down or on complete system power loss. The incorporated delay allows the switch to be swiftly integrated into any existing memory system whose supplies already comply with the decay rates.

Aside from the switch interface, the BMC will do its communications directly with the host processor. During power-ups and downs, these two technically sophisticated devices need to be isolated from each other so that neither one sends destructive noise to the other (the BMC in particular should not see any inputs greater than V_{CC} + 0.5 Vdc at anytime). This isolation can be accomplished with bus transcievers which maintain a large impedance between the devices. The transcievers will also add a round trip delay to the bus signals, and that delay should be added to the read and write strobes, RD/ and WR/. For example, the transcievers used in the sample switch have a round trip delay of 100 ns worst case so the two strobes should be increased from 200 ns to 300 ns. In low power systems, 300 ns strobes are not uncommon and system performance should not be compromised. A typical case, the Intel 8088 Microprocessor running at 5 MHz will still have a read strobe in excess of 300 ns.

Bubble System Clock

The BMC needs a 4 MHz, TTL level clock to do self-contained timing. The clock generator will have to be switched with the bubble memory system. Again, this is to prevent signals larger than V_{CC} +0.5 Vdc from being sent to the BMC.

Selecting Components

Selected parts should dissipate a minimum amount of power or the switch will defeat its own purpose. When the sample switch was prototyped, several CMOS components were picked because of their low power CMOS characteristics. In particular, the transcievers are compatible with either a 5 Vdc CMOS or TTL processor bus.

Switch Selection

A bubble system power switch should be as fast and reliable as the bubble system, have a small 'on' resistance, and be able to operate using the existing power supplies. There are two choices for switches, relays and FETs. FETs were chosen over relays for the prototype switch because they are faster and more reliable. Frequently, mechanical relays do not have lifespans comparable to bubble memory systems.

An N-channel FET was used on the 5 Vdc supply line. NFETS have low 'on' resistances and are for the most part inexpensive. By connecting a 5 Vdc supply to the NFET drain, a nominal 5 Vdc can be supplied to the bubble by the

source terminal when Vgs is large enough to turn on the NFET. The switch is turned on and off by changing the gate voltage, Vg, from 12 Vdc to 0.0 Vdc. When Vg is 12 Vdc, Vgs is 7 Vdc and the NFET is on; zero volts at Vg does not create a large enough Vgs to turn the NFET on.

In a similar way, a PFET on the 12 Vdc line supplies the bubble system with a nominal 12 Vdc supply. An NFET was not used because it would require 19 Vdc at the gate to set Vgs to 7 Vdc and have the FET turn on. Instead Vg switches from 0.0 Vdc to 12 Vdc.

Although the voltage drops across the FETs will be very small, they will add incrementally to the amount the bubble supplies vary from their specified voltages. For example, if the switch has a −1.5% voltage drop across it and your power supplies vary by ±3.5%, then the total operating range for the bubble's supplies is −5% to +2% nominal which is still within specifications. However, if your supplies varied by ±4%, then the bubble supplies would be out of specification by 0.5% on the low margin. To always stay within specification, operate your power supplies slightly off-set into their high margins. This procedure is not necessary, but it does add some operating margin to the system.

For example, in the sample switch the NFET for the 5 Vdc line has a worst case 'on' resistance of 0.16 ohms (@ 75 °C). The 12 Vdc line's PFET has 0.40 ohms (@ 75 °C) of 'on' resistance.

Using Table 1 the maximum power supply currents can be calculated,

Vd = +5 Vdc; 384 mA
Vs = +12 Vdc; 400 mA

and the worst case voltage drops across the FETs are then,

(+5 Vdc) Vds = (384 mA) 4) (0.16 ohms) = 62 mV
(+12 Vdc) Vds = (400 mA) (0.40 ohms) = −160 mV

Finally, the operating margins will change by,

(+5 Vdc) (62mV/5V) 100% = 1.24%;
+6.24% to −3.76% = operating margin
(+12 Vdc) (160 mV/12V) 100% = 1.33%;
+6.33% to −3.67% = operating margin

Doing a similar analysis for two bubble system with only one bubble active at anytime yields these operating margins,

(+5 Vdc) +6.79% to −3.21%
(+12 Vdc) +6.75% to −3.25%

A POWER SWITCH

Some characteristics of the CMOS controlled power switch shown in Figure 5 have already been described. To reiterate, the switch is easy to assemble and should require little if any system modifications. A user will have only one bus signal to indicate power on or off to the switches, (there is also an optional interrupt line to signal the processor that the supplies are operational, i.e., above powerfail threshold levels). The circuit itself is low power (25 mW standby max.) and has reliability compatible with the bubble memory system. Finally, unlike disks whose mechanical latency make them difficult to switch, the switch and the bubble system are all solid state so frequent switching can be accomplished rapidly.

A TTL or CMOS processor using DMA or polled data transfer modes with a bubble system (two bubble memories maximum) can handle this switch.

Driving The FET Switches

The one bus signal from the processor to the switch in Figure 5 is called POWER-ON. To supply power to the bubble system, POWER-ON is set active high; a logic low on POWER-ON signals a power-down.

POWER-ON is fed to the positive inputs of two comparators with open collector outputs; both negative inputs are set to 1.5 Vdc, i.e., POWER-ON signals greater than 1.5 Vdc are considered logic highs. The output of one comparator is sent to the BMC's RESET/ input. RESET/ is the BMC's hard reset signal. When it goes active low, it is an indication to the BMC that a power-down is occurring.

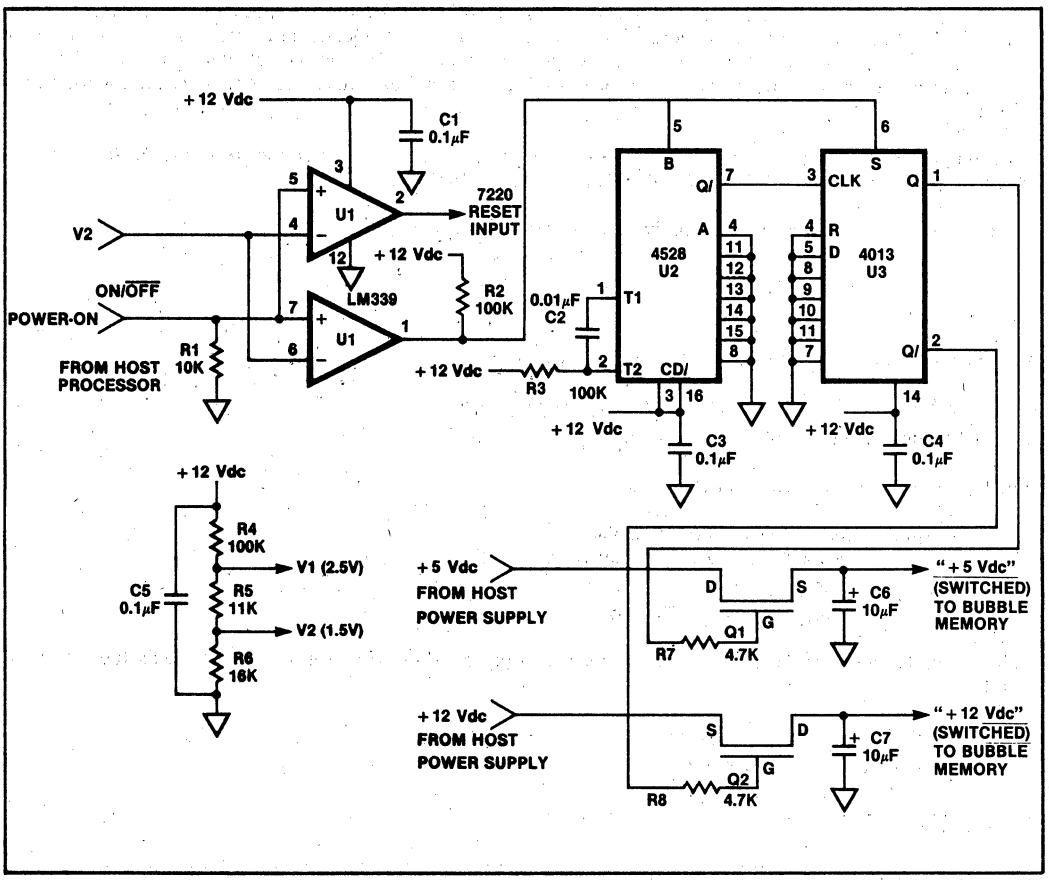


Figure 5. Bubble Memory Power Supply Switching Circuitry

Table 2. Parts List

Part	Purpose
	ALL RESISTORS ARE 0.25 W. 5% TOLERANCE.
R1 = 10 kohms	Sets gate input to low in absence of a drive signal.
R2 = 100 kohms	CMOS pull-up resistors.
R3 = 100 kohms	Sets 150 μs or greater time cosntant delay for one-shot.
R4 = 100 kohms	Voltage divider (Establishes V1 and V2 references).
R5 = 11 kohms	Voltage divider (Establishes V1 and V2 references).
R6 = 16 kohms	Voltage divider (Establishes V1 and V2 references).
R7, R8 = 4.7 kohms	Controls transition rate of Vs and Vd.
R9, R10 = 5.1 kohms	CMOS pull-up resistors.
C1, C3, C4, C5, C8, 0.1μF, 50 VDC	Power supply decoupling capacitors.
C2 = .01μF (typical) 50 VDC	Sets 150 μs or greater time constant delay for one-shot.
C6, C7 5 10 μF, 25 VDC	Power supply decoupling capacitors.
U1	LM339 low power quad comparator.
U2	4528 CMOS dual one-shot.
U3	4013 CMOS dual D flip flop.
U4, U5	74SC245 CMOS octal transcievers.
U6	74LS08 AND gate.
Q1	Siemens N-channel Econofet, BUZ71A
Q2	International Rectifier P-channel FET, IRF9531
Q3	N-channel FET, 2N6659 (Optional 7110 Bubble Memory Detector Switching)

The other comparator's output, herein referred to as Vo, is tied high to 12 Vdc through 100 kohms. It controls two CMOS components, a D flip flop through its SET input and a one-shot through its trigger input. While Vo remains high, the flip flop's outputs will remain set (Q equals + 12 Vdc and Q/ equals 0.0 Vdc) since SET, equal to Vo, will be high, 12 Vdc. The outputs, Q and Q/, are connected to the FET gates through 4.7 kohm resistors; Q drives the NFET's gate on the 5 Vdc line and Q/ drives the PFET on the 12 Vdc line. The resistors are in series with the FETs' internal capacitances and are used to control the rate at which the supplies power-up and down.

When POWER-ON goes to logic low, the output connected to RESET/ will cause the BMC to start its internal power-down routine. It will now take the BMC 120 μs (worst case) to execute an orderly shut down. During this time, the power-down decay rate specified in the hardware considerations section MUST BE SUPPORTED or data integrity may be compromised.

The power switch keeps the supply lines open by keeping the FETs on until 150 μs have elapsed after POWER-ON goes low. Vo will follow POWER-ON as it goes from logic high to logic low. The one-shot will trigger off Vo's fall-

ing edge. After waiting 150 μs, the output from the one-shot will clock the D flip flop and Q and Q/ will be reset. This resets the NFET's gate voltage to 0.0 V and the PFET's to 12 Vdc. Now both FETs are off, i.e., so are the bubble system's power supplies.

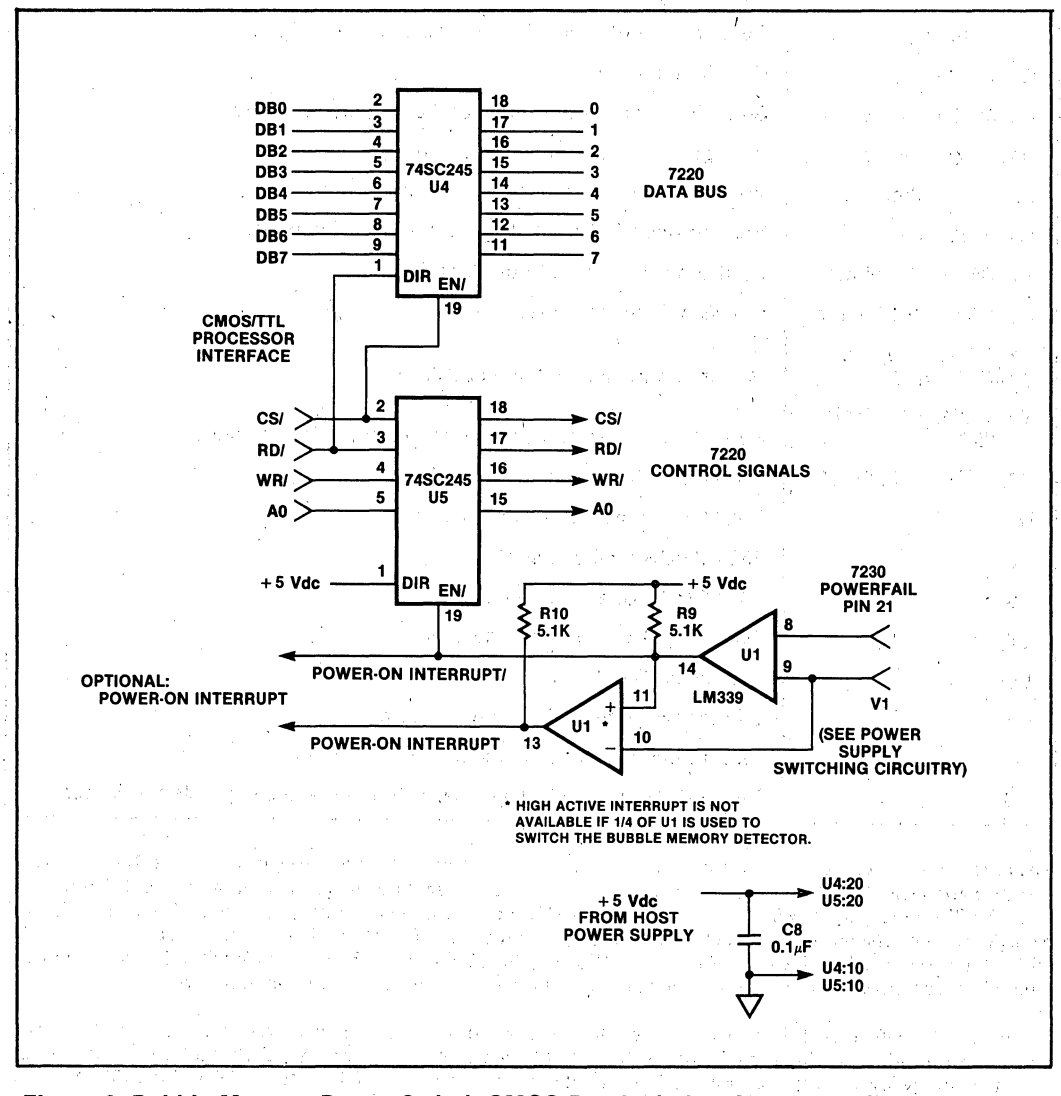


Figure 6. Bubble Memory Power Switch CMOS Bus Isolation Circuitry Polled Mode Only

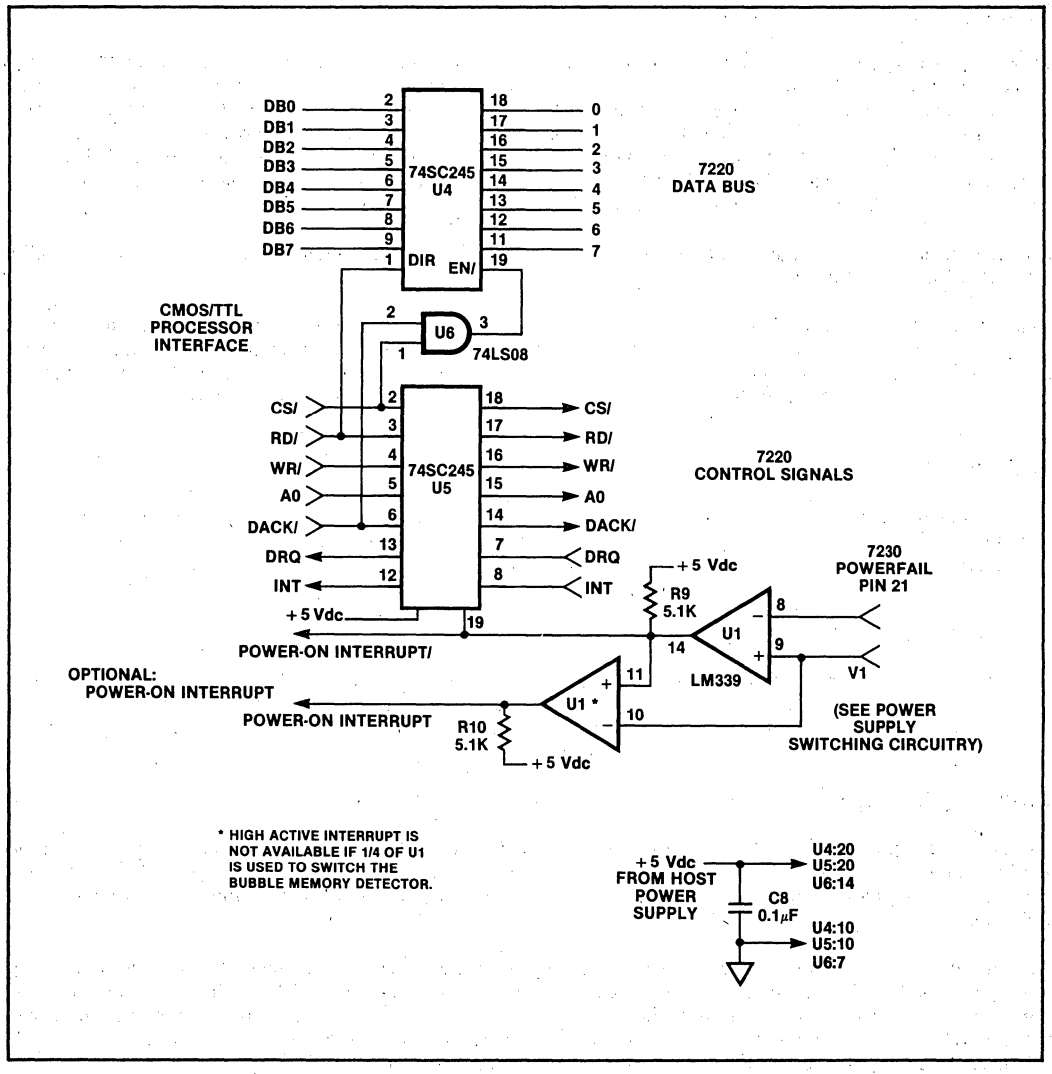


Figure 7. Bubble Memory Power Switch CMOS Bus Isolation Circuitry Polled/DMA Mode

Enabling The Interface Bus

Once the power supplies are operational, the switching circuit will enable the interface bus. Figures 6 and 7 are two possible designs for the bus transcievers; Figure 6's circuit supports polled data transfers only, Figure 7 is a modified version of Figure 6 which supports either polled or DMA. Both designs use the open collector output of a comparator to enable the BMC control signals; the output is tied high to 5 Vdc via 5.1 kohm resistors. The inputs to the comparator are 2.5 Vdc on the positive input and the 7230 current pulse generator's powerfail output, PWR.FAIL/ (pin 21), is sent to the negative input. When PWR.FAIL/ is high, greater than 2.5V, the bus is enabled, and an active low PWR.FAIL/ signal would disable the bus.

The data bus lines are not enabled until the processor selects the bubble system to do a data transfer by setting CS/ (or DACK/ for DMA) active low.

As an option, bus the output signal of the PWR.FAIL/ comparator to the host processor for an interrupt to detect when the power supplies are operational. Otherwise the processor will have to delay interacting with the bubble system until the supplies can be guaranteed operational. Invert the interrupt line with the unused fourth comparator if your processor supports active high interrupts; PWR.FAIL/ is active low on a power-down.

SOFTWARE CONSIDERATIONS

All the software information needed to successfully power switch a bubble system is presented here. Application Note 157, Software Design and Implementation Details for Bubble Memory Systems, is a useful reference if you are unfamiliar with the fundamentals of bubble memory software.

Data Transfer Modes

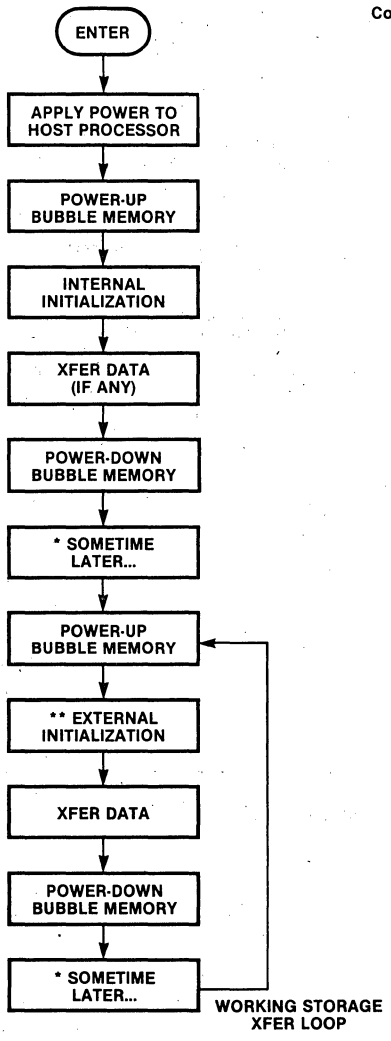
Two data transfer modes compatible with the switching circuit's bus interface are polled and DMA. Polled mode is easy and consumes the least amount of power and board space. DMA requires a DMA controller and the BMC's DRQ, DACK/ and INT signals are added to the bus interface.

Initializing The Bubble Memory

An initialization procedure must be followed after every power-up to place the BMC in a known state, to load the bootloop code into the bootloop registers and to synchronize the bubble memory to its first logical page of 64 bytes. In power switched systems, power-ups will occur before each memory access and a fast initialization routine is very desirable.

There are two ways to initialize a bubble memory. One is an internally generated command sequence executed by sending the INITIALIZE command to the BMC. The other method emulates INITIALIZE by sending the command sequence and bootloop code from the host processor to the BMC, but does not synchronize the bubble memory. This external initialization does have the advantage of being faster; worst case execution of an INITIALIZE command is 170 ms versus 5 ms for an external initialization.

Both types of initialization should be used in a power switched configuration. Send an INITIALIZE command after every cold start to synchronize the bubble memory. At the completion of the data transfer, have the BMC execute a WRITE SEEK to location (page) 395H or a READ SEEK to location 9BH. This will synchronize the bubble as the INITIALIZE command did. The bubble is non-volatile so this synchronization will not be lost even when power is removed. On the next and subsequent power-ups, use the external initialization to quickly put the BMC into a known state and place the bootloop code into the bootloop registers. Then continue to do a seek operation on each power-down to keep synchronization. Figure 8 flowcharts the operating sequence just described for low power bubble memory systems. Figures 9 and 10 are the power-up and down sequences. Flowcharts for the internal and external initialization routines are in the appendix.



Comments: The sequence of events outlined in the flowchart must be executed each time the host processor is powered-up.

* The typical time between removing the power from a Bubble Memory to powering back up should be approximately one second or greater to derive the maximum benefits from power switching.

** An external initialization is used to decrease the power-up overhead time from 235 ms with an internal initialization to less than 160 ms. However, in all cases, an internal intitialization can be substituted for an external intitialization to accommodate existing software.

Additional detail can be found in the following flowcharts:

A. Powering-Up A Bubble Memory, Fig. 9
B. Powering-Down A Bubble Memory, Fig. 10
C. Internal Initialization, Fig. 13
D. External Initialization, Fig. 14

Figure 8. Low Power Bubble Memory System Flowchart

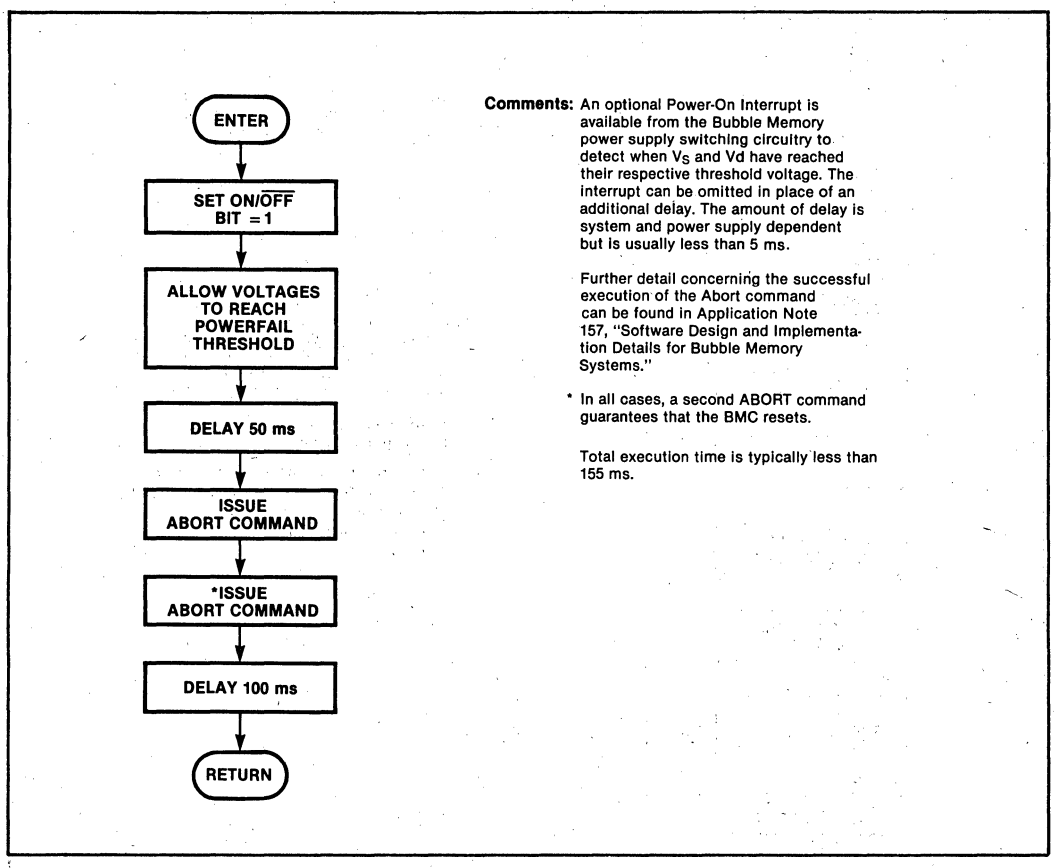


Figure 9. Power-up Flowchart for a Low Power Bubble Memory System

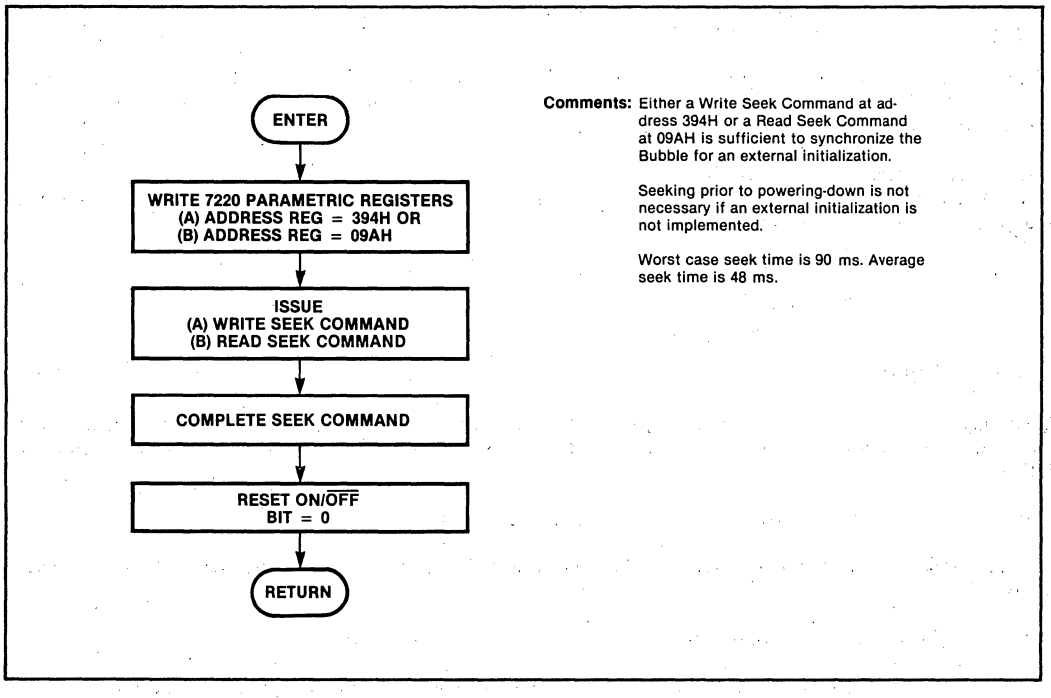


Figure 10. Power-down Flowchart for a Low Power Bubble Memory System

An easy way of keeping the bootloop code outside the bubble is to read it out of the bootloop registers, after each cold start, into the host system's RAM. Then retrieve the code from RAM for each external initialization.

The seek commands expect their operands to be the number of the page one previous to the page you wish to seek. For example, the page that should appear as the operand in the WRITE SEEK command for the initialization routine above is 394H and the READ SEEK operand should be 9AH.

If speed is not a factor, you can use the INITIALIZE command after every power-up, but the total amount of time before a data transfer can begin will still be the time it takes for the power supplies to become operational, typically 155 ms plus the time to initialize the bubble system,

Power-up + Internal Initialization = 325 ms (worst case)
Power-up + External Initialization = 160 ms (worst case)

Efficient Software

Every operation run on your system sets its own bubble memory needs. How efficiently your system responds to these needs determines how much power is dissipated. Some suggestions for energy efficient drivers and programs follow.

If information is called in a fixed sequence, store it in that fixed sequence.

Instead of repeatedly accessing the bubble for the same information, transfer the data into system RAM and retrieve it from there.

Transferring many pages of data is more efficient than doing many small transfers.

Intrinsically, running multibubble systems in serial will use less power than running parallel memories since in the former case only the coil drives of one bubble memory will be active at any one time. For example, a system running two bubbles in serial will have one active bubble memory and one bubble in standby mode any time the system is accessed. This means 5.45 W, 3.9 W + 1.55 W, will typically be dissipated during the access. Run in parallel, these same two bubbles will typically dissipate 3.9 W each or 7.8 W total during a data transfer.

DETECTOR SWITCHING

Although power switching is the primary hardware technique, the BMC has an output signal that indicates when the detector stacks are active. One advantage detector switching has over power switching is that the bubble system does not need to be reinitialized when power is reapplied.

Since the stacks are only used to sense the bubbles during read operations, the DETECTOR ON/ signal can be used to switch power to the detectors. Standby power consumption can be reduced by 20% per bubble memory if the detectors are switched off and the incremental amount you gain by leaving them off during write operations depends on the frequency and duration of your read and write operations. For example, a jet airplane's bubble memory flight recorder (using eight one mega-bit bubbles) has data written out to it on every flight. It will only be read if there is a problem during a flight. By switching off the detectors, four watts are saved on each flight (0.5 W/bubble).

Figure 11 is the circuit diagram for a detector switch. DETECTOR ON/ is inverted with a comparator; DETECTOR ON/ is the negative input, 1.5 Vdc is the positive input and the output is tied high to 5 Vdc through 100 kohms. The inverted signal is used to change an NFET's gate voltage which turns the NFET on and off. That in turn switches the detector supply on and off. Placement of the NFET is critical; lay it out as close to the bubble memory as possible.

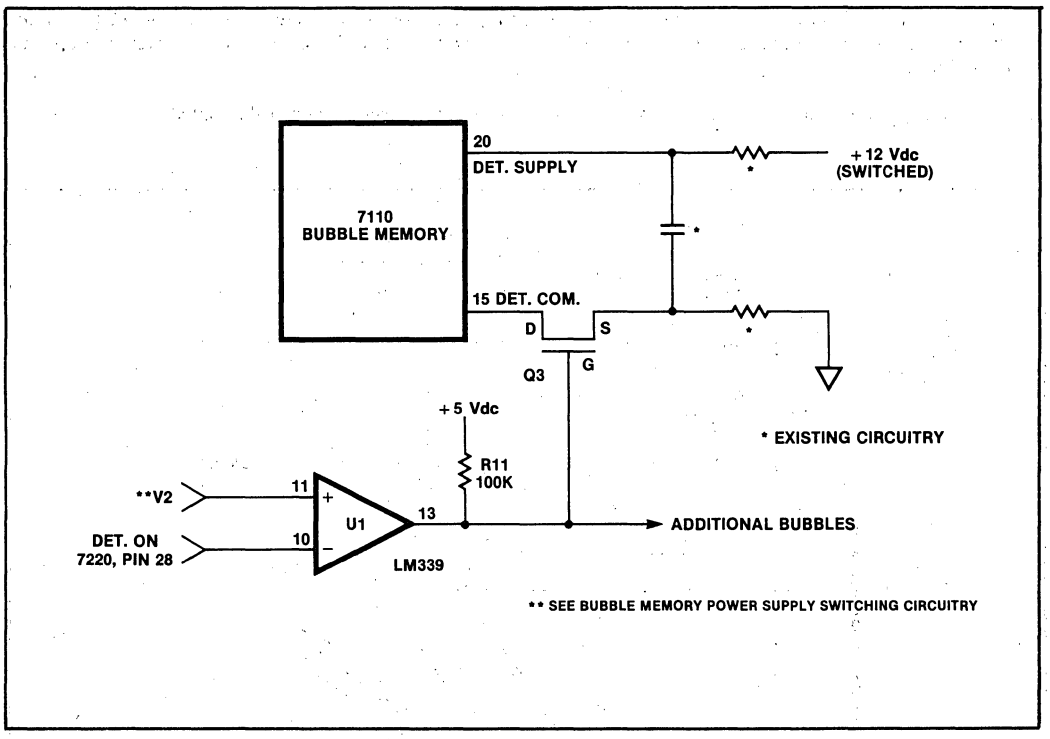


Figure 11. Bubble Memory Detector Power Supply Switching Circuitry

With a very restricted power budget, consider implementing both power and detector switching. In the sample switch, the unused fourth comparator is available for the detector switch if it is not used to invert the optional power-on interrupt.

Figure 12 is a graph comparing complete switching to the amount of average power dissipated.

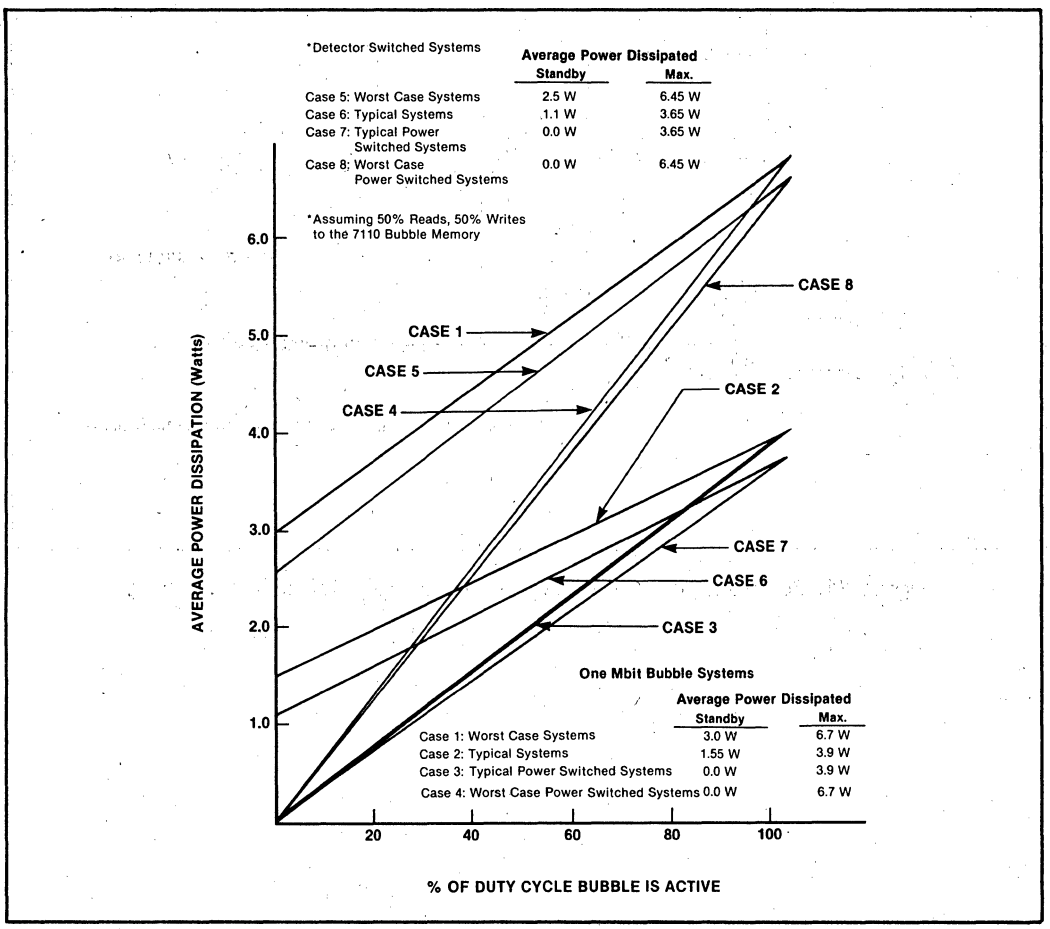


Figure 12. A Complete Comparison of Bubble Memory Activity to the Amount of Average Power Dissipated by the Bubble Memory System

SUMMARY

The main goal of this application note is to assist designers developing portable or other low power equipment. By utilizing the CMOS controlled power switch, a designer can build a simple, reliable, low power bubble memory system with a minimum of time and effort.

APPENDIX A

Typical Measured Values @ 25 °C

Configuration — One megabit Bubble Memory System incorporating a polled mode interface.

		Vs (+5VDC)	Vd (+12VDC)
Case 1:	Power-On = Off		
	Bubble Memory System Power Consumption	0.19 mW	13.66 mW
Case 2:	Power-On = On		
	Bubble Memory in Standby		
*	Total Bubble Memory System Current	260 mA	39.4 mA
*	Bubble Memory System Power Consumption	1.3 W	0.5 W
	Voltage Drop Across the FET Switch	32.1 mV	9.5 mV
Case 3:	Power-On = On		
	Bubble Memory Actively Transferring Data		
*	Total Bubble Memory System Current	262 mA	252 mA
*	Bubble Memory System Power Consumption	1.3 W	3.0 W
	Voltage Drop Across the FET Switch	32.4 mV	61.8 mV
Power-Up Rise Time (Power-On = 1)		150 μs	600 μs
Power-Down Fall Time (Power-On = 0)		15 ms	250 ms

*Includes an Intel 8284A clock generator to produce the required 4 MHz clock for the 7220 controller and BPK-70 (7242 Formatter Sense Amplifier).

Clock Specifications:	**Parameter**	**Min.**	**Max.**
	Clock Period	249.75 ns	250.25 ns
	Clock Phase Width (High)	45%	55%
	Input Signal Rise Time		30 ns

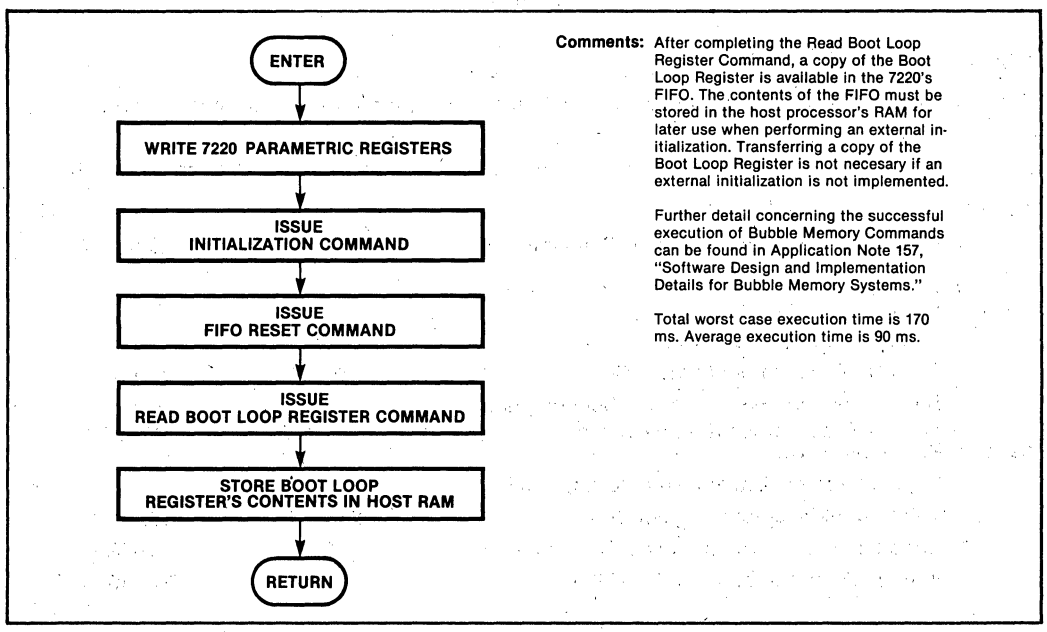


Figure 13. Internal Initialization Flowchart

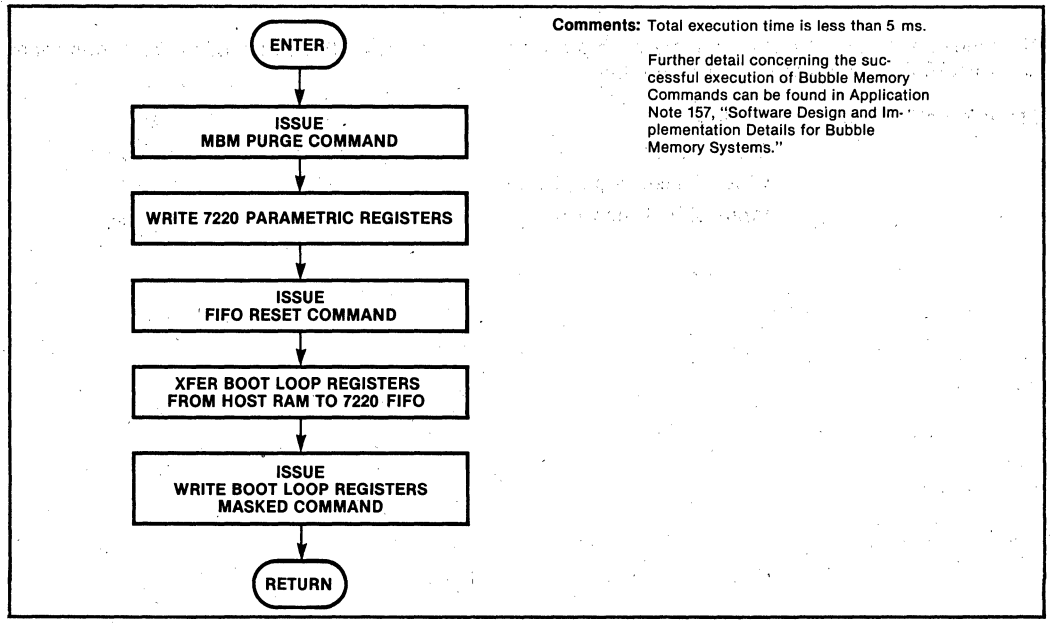


Figure 14. External Initialization Flowchart

intel

ARTICLE REPRINT

AR-243

Thin-film detectors, X-ray lithography deliver 4-Mbit bubble chip

Next-generation bubble memory chip is even smaller than the compatible, 1-Mbit device; set of support circuits takes care of memory system requirements.

Propelled by X-ray lithography and thin-film permalloy detectors, bubble memory chips have climbed to the 4-Mbit level.

Using X-ray lithography, Intel Corp. (Santa Clara, Calif.) has managed to reduce the periodicity between bubbles from 11.2 (for its 1-Mbit chip) to 5.6 μm and feature sizes from 1.25 to 0.75 μm. At the same time, thin-film permalloy detectors, replacing thick-film versions, nearly double the signal strength of the detected bubbles (Fig. 1).

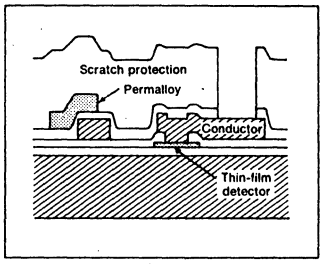
Moreover, a novel multiplexing technique handles the outputs from the eight on-chip detectors, which is double the number used on the 1-Mbit chip. This technique, which Intel is keeping under wraps, permits the higher-density chip to fit into a 22-pin package.

The outcome of all that is the 7114, plus a complement of six support circuits. The 7114 retains the basic architecture of the 1-Mbit 7110, and all the support circuits are pin-compatible with the chips that support the 7110. Aside from a few software changes to handle the larger memory space, the upgrade is totally transparent to the system user, claims Mike Eisele, bubble memory product manager. Thus in many cases the older bubble chips can be removed from a system and new ones plugged in.

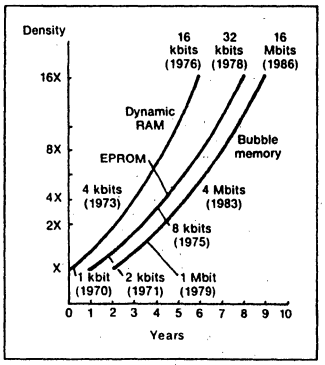
Dave Bursky

Electronic Design

However, the support chips cannot control the 1-Mbit device, and some minor hardware changes must be made to accommodate the smaller package used for the 4-Mbit chip. The package's dimensions—1.46 by 1.35 in.—represent a savings of nearly 0.9 in.^{2} over the 1-Mbit package's 1.7 by 1.68 in. In addition, the smaller package, which has DIP-like pins, eliminates the need for a socket in many cases and also has a lower profile to permit board spacings as close as 0.6 in. The same package will be used by Motorola Inc. (Phoenix, Ariz.) when it builds the second-generation 1-Mbit chip as called for in the alternative-source agreement signed earlier this year with Intel (ELECTRONIC DESIGN, July 8, p. 23).



1. A key element of Intel's 4-Mbit bubble memory is this thin-film permalloy detector structure, which delivers twice the output signal of the previously used thick-film detector.



2. Following the same growth curve as UV EPROMs and dynamic RAMs, bubble memory technology still has a good way to go to reach the 16-Mbit level projected for 1986.

However, to bring the price of the bubble memories down to what Eisele feels would be attractive for system users—about $150 for a 4-Mbit chip by 1986—Intel has turned to a Perkin-Elmer X-ray lithography system in what it believes to be the first commercial use of X-ray systems. (Other companies, though, are not very far behind—many semiconductor manufacturers have very active research and development programs to make X-ray systems practical on the production line.)

The production process for the 4-Mbit chip includes 90% of the process steps used for the 1-Mbit device, thus sharing much of the learning-curve experience, in the short run.

Functionally, the 4-Mbit device will appear to operate just like the 1-Mbit memory. However, when the 7114 operates at the 50-kHz field rate of the 1-Mbit device, the access time is double that of the smaller chip, since the loops are longer. But the data rate is double that of the 1-Mbit chip because more detector outputs are multiplexed and then fed out from the chip. Also, a version of the 4-Mbit chip will operate at twice the field rate (100 kHz), for an access time of 41 ms—almost the 40-ms access

time of the 1-Mbit chip.

There will be a full kit of parts available from Intel when samples of the memory will be available next year. The largest chip will be the 7224 controller, which duplicates the functions of the 7220 controller but has the internal changes needed to handle the larger memory space. Similarly, the other circuits are the 7234 current-pulse generator, the 7244 formatter-sense amplifier, the 7250 coil predriver, and the 7254 coil drivers.

Bubble memory capacity has been quadrupling about every four to five years. This follows very closely what happened to UV EPROMs (Fig. 2), even though EPROMs went through doubling cycles every two years.

intel

ARTICLE REPRINT

AR-250

November 1982

Bubble Chip Packs 4 Mbits Into 1-Mbit Space

Hudson Washburn
Design Engineer

Sam Nicolino
Design Engineer
Intel Corporation
Santa Clara, California

Electronic Design
ORDER NUMBER: 210842-001

BehindTheCover

Right after putting their 1-Mbit bubble memory chip into production several years ago, designers at Intel decided to try various sections of what would be needed to build a 4-Mbit device. Although several were fabricated and proved functional, priorities in ironing out the production problems for the 1-Mbit chip forced them to put the 4-Mbit design on the back burner, working on it as a secondary project. Finally, though, the years of patience are paying off, and as our cover story in this issue (p. 1) highlights, the 4-Mbit magnetic bubble memory—the i7114—is functional.

Fortunately, the designers have been able to time the developments so that both the bubble chip and its associated support chips will be ready at the same time. As Mike Eisele, product manager for the Magnetic Bubble Memory Division, notes, that wasn't the case for the 1-Mbit device—it took Intel a lot longer than it expected to make the controller fully functional.

In developing the 4-Mbit memory, Hudson Washburn, design engineer, expected that the control elements on the chip—the bubble generator, transfer gates, replicator, and detector—would be the most difficult sections to get to work, whereas he thought that the propagation paths would be relatively simple to implement. But when actually trying to create the memory chip, he and the other researchers found that the control sections performed fine after only a few iterations while the propagation paths turned out to be the tricky development problem.

Additionally, mastering the technology needed to build the 4-Mbit bubble chip was a long, hard process with many half steps back, Washburn says. However, work on the 1-Mbit device also helped the bigger memory: Every time something happened that caused yield problems on the 1-Mbit chip, work was stopped on the new circuit. When the problem or problems on the 1-Mbit process were solved, the designers applied what they learned to the 4-Mbit technology.

Also, the designers decided to use a thin-film detector structure to boost the signal-to-noise ratio of the output signal. Although building this detector adds a second critical masking level to the production process, the decrease in yield due to the additional step is expected to be more than offset by faster testing. As it turns out, testing tends to be a major part of the chip cost as the capacity reaches 4 Mbits, according to Dave Dossetter, bubble memory product marketing engineer.

Perhaps appropriately for a 4-Mbit memory, Intel worked with a manufacturer of lithography equipment and a mask maker to use X-ray lithography. Although contact printing was employed during development, Intel plans to put X-ray lithography to work for volume production, which would make it the first such commercial use.

A 4-Mbit bubble memory chip, supported by a full complement of six dedicated circuits, stands poised for applications ranging from industrial control to telecommunications to personal computers.

Bubble chip packs 4 Mbits into 1-Mbit space

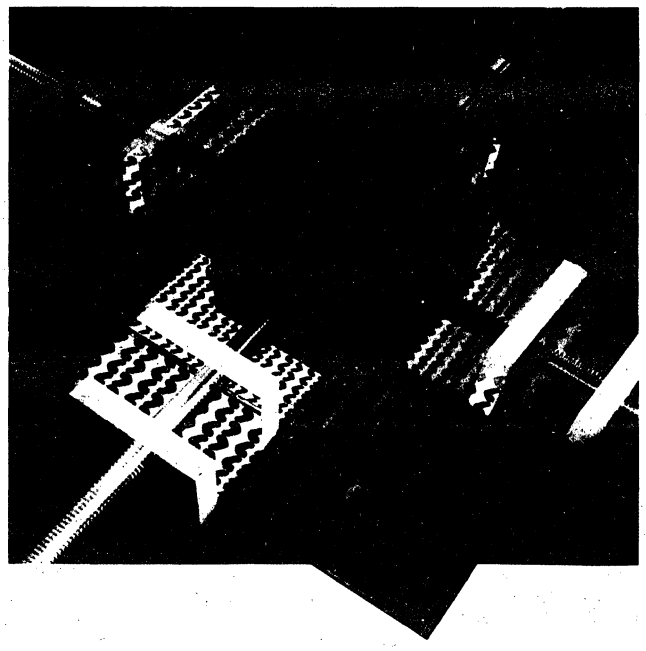
Bubble memories sport a hefty list of advantages for mass storage applications. Yet because of the complexity of interfacing them, most designers have shied away from these devices, leaving them outcasts. But the sheer appeal of 4 Mbits tucked into a 20-pin package, coupled with a set of components that takes care of the complexities of linking a bubble chip to conventional host computers, makes an extremely attractive option for those designers who have previously resigned themselves to simpler but less attractive mass storage.

As for those who have already taken the plunge into bubbles with the chip's 1-Mbit predecessor, the 7110, upgrading to the 4-Mbit 7114 requires only minimal changes.

Some of those ready to benefit from a simplified bubble memory system are portable equipment makers, who will take advantage of the compactness and nonvolatility of bubble chips. Industrial control and robotics manufacturers will appreciate bubble devices' resistance to hostile environments, since they have no moving mechanical parts to succumb to shock, corrosion, or high humidity. These last three qualities also are important to telecommunications suppliers, who need low-cost, reliable buffers for PABX and other message-carrying systems.

Still, to reap the rewards inherent in bubble memories, a full complement of support circuits must accompany the bubble chip itself. Those companions are ready, in the form of the 7224 bubble memory controller, the 7244 formatter and sense amplifier, the 7250 coil predriver, the 7254 VMOS driver transistor, and the 7234 current pulse driver.

Despite these components, a 4-Mbyte bubble memory system takes less space than the previous 1-Mbyte design, since the new bubble chip's package is both narrower, allowing more chips per board, and shorter, giving more room to stack boards next to one another (see "More Memory in Less Space"). Furthermore, the support components are interchangeable and, like the bubble chips, do not have to be matched sets, as was often true of other bubble devices. In fact, any bubble chip is guaranteed to

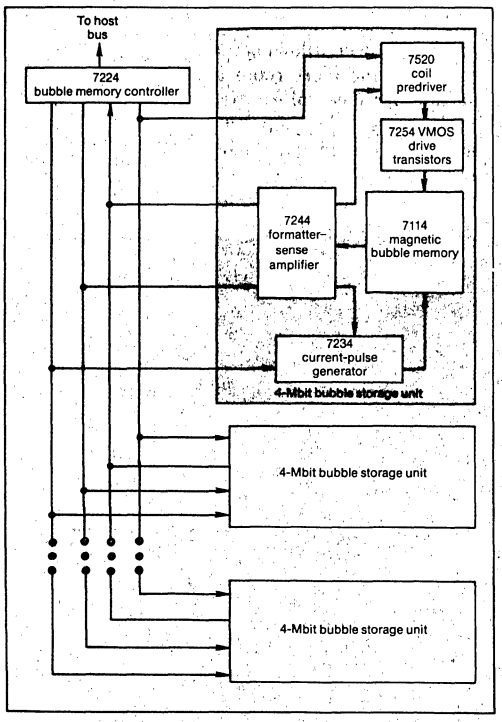


Hudson Washburn, Design Engineer
Sam Nicolino, Design Engineer
Intel Corp.
3065 Bowers Ave., Santa Clara, Calif. 95051

work with any support component, so that components can be replaced in the field without fine tuning.

Also, because the 4-Mbit bubble chip was designed to be compatible with the same hardware and software developed for the 1-Mbit version, the support circuits for both have the same pinouts. Most of the register bits are the same, too. The only differences are those in which the larger memory capacity affects how the bits are defined. Consequently, from a software perspective, any revisions to upgrade to the 4-Mbit chip are minor.

As with the 1-Mbit system, the user's interface with the 4-Mbit system remains simple. The software is written so that, first, parameters are passed to the controller by loading its registers, followed by commands. In addition, data is written or read in any of three transfer modes—DMA, polled, or interrupt—and the controller's 40-byte FIFO acts as a buffer between the host and formatter–sense amplifier chips. The formatter–sense amplifier is responsible for sending and receiving serial data between the bubble and the controller. The host system therefore need only monitor the controller's status register to determine when it is busy and to see if a transfer operation was successful.



1. The key to building a 4-Mbyte bubble memory system is the ability of the bubble chip's support ICs to simplify the interface with the host. Five such ICs plus a single 4-Mbit chip (shaded) form the basic memory block. Up to seven additional blocks in parallel, all governed by one memory controller chip, complete the system.

The bubble memory controller is the bubble chip's link to the host. It communicates with the host over an 8-bit bidirectional data bus; a single address line (A_0); and a chip-selection, a read and a write control, and an interrupt line. In addition, a ninth data bit line (D_8) can be used to detect parity errors.

The remaining input and output lines of the controller connect the formatter–sense amplifier, the coil predriver, and the current-pulse generator. These components, plus a pair of VMOS drive transistor chips, make up a 4-Mbit bubble storage unit (Fig. 1). Up to eight such units may be connected to a single controller, allowing users to trade off the number of pages against the individual page size to fit their data transfer requirements.

The controller close up

To understand the software and hardware interface with the bubble subsystem requires and understanding of the controller. An HMOS chip, it is housed in a 40-pin DIP and divided into 10 functional blocks (Fig. 2).

The host processor operates the bubble memory system by reading from, or writing to, specific registers within the bubble memory controller. The host selects each register by placing an address on lines A_0 and D_0 through D_4. Specifically, the status register and command register are directly addressed using these six bits; a third register, the register address counter, is also directly addressed and in turn indirectly addresses the remaining registers, including the block-length register, the FIFO data buffer, and the enable register. These remaining registers are called parametric registers because they contain the flags and parameters that determine exactly how the controller will respond to commands written in the command register. The parametric registers are located in a register file and are selected with addresses 1011 through 1111. In general, the parametric registers must be loaded before commands are issued to the controller.

Parametric registers are loaded when they are addressed by the register address counter. The controller automatically increments the counter by one after each data transfer between the host and a parametric register. Thus there is no need to reload the address register in the case of multiple register reads and writes.

The address register increments, starting with the address first loaded, until it reaches binary address 1111. It then wraps around to 0000 and halts until it is reloaded with another address. However, when

line A_0 is zero, all data transfers are with the FIFO. In addition, any other commands or a controlled stop sequence will reset the address counter to 0000, which is the FIFO address.

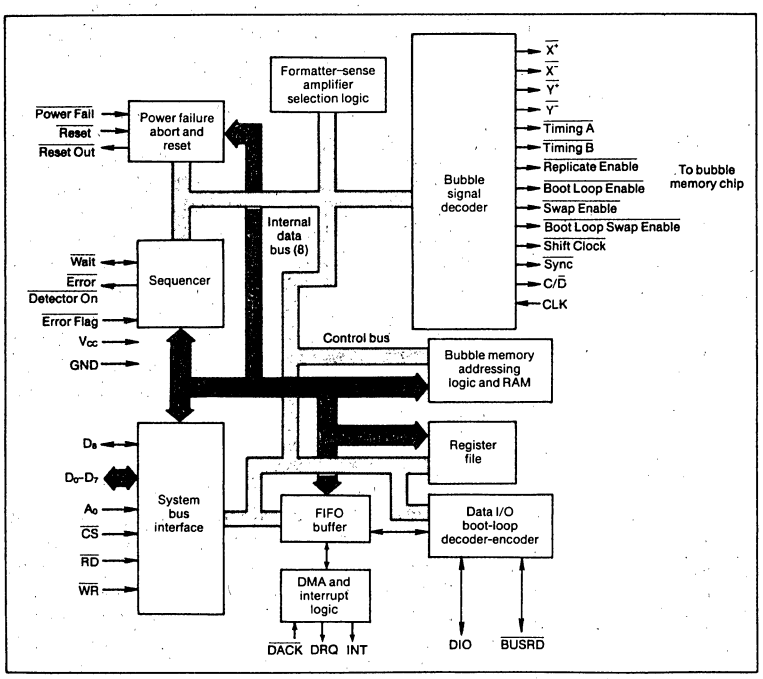
The most commonly used commands (see the table) are Initialize, Read Bubble Data, and Write Bubble Data. Others used in a typical operation are Read Seek, Write Seek, Read Formatter–Sense Amp Status, and Reset FIFO. In addition, two commands —Zero Access Read Seek and Zero Access Read Bubble Data—slash the data access time by a factor of more than 150. Zero Access Read Bubble Data returns the first byte of data in the FIFO within 50 μs after the command is sent, provided the address is known in advance of the access command.

Parameters first

Commands are written by the host into an 8-bit write-once command register. Depending on the command, certain parameters must already be written into their respective registers. For example, the Initialize command must be preceded by the number of formatter—sense amplifiers in the block-length register's first four MSB locations (Fig. 3a). Similarly, before issuing a Read Bubble Data command, the starting address information must already be set in the address register (Fig. 3b), as must be the number of system pages in the block-length register. Thus each command has its specific set of parametric requirements that must be established before it is issued.

If the parametric conditions have been set, the command is issued using a 5-bit command code. For example, Initialize is 00001, Read Bubble Data is 00010, and so on.

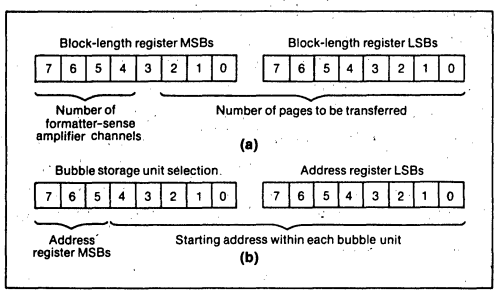
Information about any error condition, the completion or termination of a command, or the controller's readiness is stored in the status register. The host can directly address this register by setting the A_0 line and examining the eight status flags. The status register is updated every microsecond. Bits 1 through 6 (Fig. 4a) are set during command



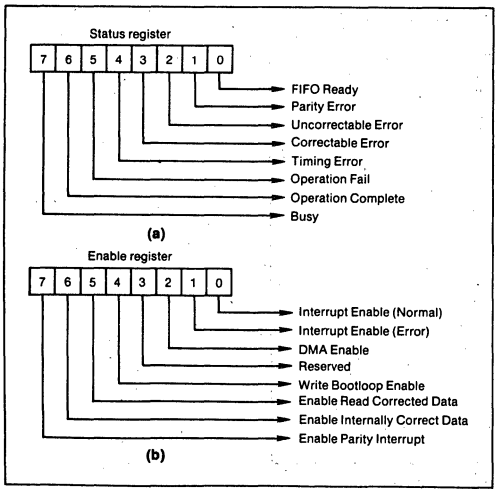
2. The 7224 bubble memory controller interfaces the bubble storage units with the host processor. It performs 10 functions, each represented by a block. The host is connected to an 8-bit data bus with an optional parity bit, a single address line, a chip-selection line, and a read and a write control line. Interrupt and DMA handshaking also are available.

execution and are reset when a new command is issued. The flags in the status register indicate whether the controller is executing a command or has completed one. In addition, they show whether an uncorrectable error or a timing error has occurred. Also, using a parity bit, the controller checks the data the host sends it and generates an odd parity for the data it sends to the host. Any parity errors are flagged.

The system page size and the number of pages to be transferred in response to a single bubble memory data read or write command are set by the block-length register, a 16-bit write-once register. The system page size is proportional to the number of bubble storage units operating in parallel during a data read or write operation. Each bubble chip requires two formatter–sense amplifier channels, with bits 4 through 7 specifying the number of such channels to be accessed. For example, in a 4-Mbyte system, if bits 7 to 4 are 0001, two channels will be accessed, each page will contain 512 bits, and there will be 65,172 pages. Setting the bits to 0100 specifies eight channels, 2048 bits per page, and 16,384 pages.



3. The parametric registers set the basic conditions for transfers between the host and the bubble memory system. The block-length register gives the number of formatter–sense amplifier channels and the number of system pages in a block (a). The address register gives the starting address for a read or write command (b).



4. The status register bits (a) tell the host about any data errors, the state of the controller's readiness, or whether a command was completed properly or not. The register is updated every microsecond and indicates whether a data error was correctable or not, in addition to pointing out parity and timing errors. The enable register bits (b) specify several conditions, including interruption on an error, DMA enabling, and parity error interruption.

The right address

Which bubble memory group is accessed and what the starting address location is within that group are determined by the contents of the address register. Each bubble chip has 8192 address locations for reading or writing data. Consequently, 13 bits are needed to specify an individual bubble storage unit's starting address. Which of the units to be read from or written to is indicated by address register bits 5 through 7. How the controller interprets these bits depends on the number of bubble storage units in a group as specified by the block-length register. For example, if the formatter–sense amplifier channels are numbered 0 through F_{16} and the number of formatter channel bits of the block-length register are set at 0000, the address register bits will specify channels 0 through 7. If, on the other hand, the block-length register bits are in the sequence 0001, the address register bits select the formatter—sense amplifier channel pairs and address register bits 0110 select channels C and D.

The address range for a 4-Mbyte subsystem is 0000–FFFF, or 65,172 pages. Selecting address register bits 0111 puts the data in the last 8192 pages of bubble storage.

Enable register controls

Certain functions in the formatter–sense amplifier and the controller are governed by setting bits in the enable register (Fig. 4b). For example, setting the Enable Parity Interrupt stops the host when the controller detects a parity error on the data bus lines (D_0–D_7). Also, the controller operates in a DMA data transfer mode when the DMA Enable bit is set. In this mode the Data Request and Data Acknowledge interface signals become operational; otherwise, the controller supports interrupt-driven or polled data transfer modes. As a result, users have a choice of three data transfer methods.

The Interrupt Enable (Normal) bit, when set to a 1, allows the controller to interrupt the host system when a command is successfully executed. The Interrupt Enable (Error) bit works in conjunction with

Bubbles by the block

The basic technology of the 7114 4-Mbit bubble chip—known as field access, conductor-first permalloy—is the same as used to build the earlier 7110, a 1-Mbit part, except for several important refinements. These refinements quadruple the bit density and the data transfer rate.

The increased density is produced by halving the period of the basic memory cell (called an asymmetric propagator) to 5.5 μm. The resultant chip size is 501 by 580 mils (compared with the 1-Mbit's 512 by 614 mils). A 0.75-μm minimum feature size, smaller than that of any silicon chip, is being printed now in development volumes using optical contact lithography. However, X-ray lithography techniques will be used for production volumes to achieve repeatible results despite the small minimum-feature size.

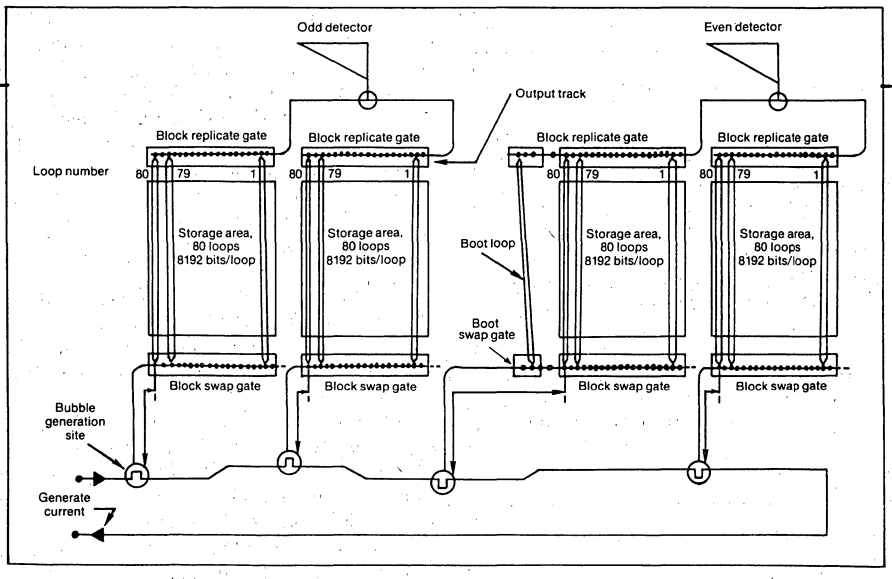
In addition, a thin-film detector was developed that doubles the detected bubble signal compared with the previous thick-film detectors. This makes doubling the data rate feasible. Further, doubling the field rotation rate from 50 to 100 kHz also doubled the data rate, producing the overall 400% increase, which also means an average random access time of 40 ms. (A 50-kHz version will be introduced first that has twice the data rate of the 1-Mbit chip and an 80-ms access time.)

Like the technology, the architecture of the 4-Mbit chip is an enhanced version of the 1-Mbit design. Both use block-swapping and replicating schemes to write and read bubbles in parallel, to ensure nonvolatile storage, and to permit the use of multiplexed replication generators to reduce the number of external pins.

The page length is fixed at 512 bits (64 bytes), but the number of pages has been quadrupled for the 4-Mbit part. Both chips are organized into identical halves. Thus, from an architectural perspective, the higher-density chip looks like a 1-Mbit part with four times the number of pages and either twice (50 kHz) or four times (100 kHz) the data rate.

Actually, the 7114 is divided into eight octants, each comprising 80 minor loops, and each loop containing 8192 bits (see the figure). The 7110, in comparison, is split into four quadrants, each with 80 minor loops, but each loop contains only 4096 bits. Also, whereas the 7110 was designed to sense one bit per side per field rotation, the 7114 senses two bits. In the 50-kHz 4-Mbit part, the longer loops are compensated for by the two-bit-per-rotation sensing.

Like the 1-Mbit device, the 4-Mbit chip has redundant loops to ensure a high yield of devices with the full 4,194,304 bits of storage capacity. Redundancy increases yields and so lowers device cost. During manufacture, each device is individually tested and a record of faulty loop locations is written and stored in the device's bootstrap loop, known as the "boot loop." The boot loop's contents are used by the 7224 bubble memory controller during initialization, reading, and writing to provide a full 4-Mbit memory space to the user while keeping redundant loops invisible. The major-track, minor-loop architecture used by both the 7114 and the 7110 to accomplish the writing, reading, and nonvolatile storage of data also maintains the reliability inherent in bubble technology.



the other enable register bits to support three levels of error correction.

At the first level, setting Enable Internally Correct Data causes the controller to send a command to a formatter–sense amplifier when an error has been detected. The formatter–sense amplifier responds by internally cycling the data through its error-correction network. On completion, it sends its status to the controller, indicating whether or not the error was corrected.

For the second level, the Enable Read Corrected Data bit prompts the controller to issue a command to the appropriate formatter-sense amplifier when an error has been detected. The formatter—sense amplifier then corrects the error if possible and transfers the corrected data to the controller. When the data transfer is complete, the controller reads the formatter–sense amplifier's status to determine whether the error was corrected. Otherwise, faulty data could be transferred to the controller and possibly to the host.

Lastly, setting the Write Bootloop Enable bit permits writing into the bootstrap loop, called here just the "boot loop." Normally, the loop should only be read, but under special circumstances a user may wish to write into it.

Bubble controller command codes

D_4	D_3	D_2	D_1	D_0	Command name
0	0	0	0	0	Write—Boot Loop Register Masked
0	0	0	0	1	Initialize
0	0	0	1	0	Read Bubble Data
0	0	0	1	1	Write Bubble Data
0	0	1	0	0	Read Seek
0	0	1	0	1	Read Boot Loop Register
0	0	1	1	0	Write Boot Loop Register
0	0	1	1	1	Write Boot Loop
0	1	0	0	0	Read Formatter–Sense Amp Status
0	1	0	0	1	Abort
0	1	0	1	0	Write Seek
0	1	0	1	1	Read Boot Loop
0	1	1	0	0	Read Corrected data
0	1	1	0	1	Reset FIFO
0	1	1	1	0	Memory Unit Purge
0	1	1	1	1	Software Reset
1	0	0	1	0	Zero Access Read Bubble Data
1	0	1	0	0	Zero Access Read Seek

More memory in less space

Instead of a leadless package requiring a second, leaded socket, the 7114 4-Mbit bubble chip is housed in a leaded package that can be placed in a socket or soldered directly to a PC board. Like the 1-Mbit package, it has 20 pins. However, the distance between pin rows is smaller, making the footprint smaller and allowing designers to incorporate more components onto the board. Also because the package's height is smaller, boards can be spaced as close as 0.6 in. to one another. Thus consequently, either more boards can be accommodated or the overall system size can be made smaller. As a result, a 4-Mbyte bubble memory system can be built in less space than a 1-Mbyte bubble system.

The FIFO as a data buffer

All data moving between the host and the bubble units passes through the 40-byte FIFO buffer. As a result, the data transfer is asynchronous, with timing constraints relaxed somewhat for both the formatter–sense amplifier and the host system. When the controller is busy executing a command, the FIFO functions as a data buffer; however, when the controller is not busy, the FIFO is available to the host as a general-purpose FIFO register bank.

Actually, a total of 43 bytes of data may be stored in the controller: 40 bytes in the FIFO, 1 byte each in its input and output latch, and 1 byte in the controller's input latch. During execution of a command involving a data transfer between the host and the formatter–sense amplifiers, the data passes through the FIFO and its status is indicated by the FIFO Ready bit in the storage register.

The FIFO is addressed automatically after the last parametric register has been written into; alternatively, the host can explicitly address the FIFO by writing the address 0000 into the register address counter. Also, after a Write Bubble Data, a Write Boot-Loop Register, or a Write Boot-Loop Register Masked command is issued, the controller delays the data transfer until there are at least two bytes of data in the FIFO. Furthermore, it is the host system's responsibility to keep up with the data transfer during execution of a command; otherwise the FIFO could underflow or overflow. If either case occurs, a Timing Error bit is set in the status register.

A look at data transfer

The boot-loop register plays a key role in data transfer both for writing and reading. This 160-bit register contains information detailing the configuration of good and bad loops in the corresponding channel of each bubble chip.

Each bit of the register corresponds to a minor loop in the bubble chip. As data passes through the latter's I/O latches, the contents of the boot-loop register are used during reading to remove the bits corresponding to bad loops and during writing the contents are used to insert 0s in those bit positions that correspond to bad loops.

Meanwhile, the error-correction block implements a 14-bit Fire code error-detection and -correction process. If it 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 that passes through the FIFO during a data write operation. When data is being read, this circuitry checks the data block and notifies the controller with an error flag when an error has been detected.

As stated earlier, a Write Bubble Data command from the controller to the formatter–sense amplifier permits data from the controller to be written into the good loops of the memory unit. If the error correction is activated, the amplifier automatically adds the 14 error-correction bits to the end of each 256-bit data block.

Similarly, a Read Bubble Data command enables the formatter–sense amplifier to read data from the bubble chip, as was also mentioned previously. This 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 FIFOs. If the error correction is selected, data to be read is first buffered. That is, a full block (270 bits) of data is collected in the FIFO before any bits are read out. As a result, the error-correction circuitry detects any errors and interrupts the controller before any data is sent. If there are no errors, the 270-bit block is read from the FIFO and sent to the controller while the next block is loaded into the FIFO.

In contrast, an Internally Correct Data sequence forces the formatter–sense amplifier to cycle the data internally through the error-correction network without sending any of it to the controller. At the end of the operation, the amplifier sets a Correctable or Uncorrectable Error bit in its status register. If the error is correctable, the controller has the option of issuing a Read Corrected Data command. This command cycles the data through the error-correction circuitry as it is being read by the controller. After all 256 bits have been transferred to the controller, the formatter–sense amplifier status register indicates whether the error was found to be correctable or not. The Read Corrected Data command is used even when the data has been previously corrected by the Internally Correct Data command. □

The authors wish to thank Dave Dossetter, Product Marketing Engineer, and Dick Pierce, Marketing Applications Engineer, for their invaluable assistance in preparing this article.

intel

ARTICLE REPRINT

AR-271

April 1983

New Bubble-Memory Packaging Cuts Board Space and Manufacturing Costs

Art Thorp
Intel Corp.
Santa Clara, Calif.

ORDER NUMBER: 230642-001

New bubble-memory packaging cuts board space and manufacturing costs

Low-profile 4-Mb bubble-memory package is interchangeable with 1-Mb types and also lets printed-circuit boards be spaced on 0.6-in. centers

by Art Thorp, *Intel Corp., Santa Clara, Calif.*

☐ Designing a second-generation product gives an engineering team the chance to put in all the improvements they realized were needed after the first design was formalized. The new 7114 4-megabit magnetic-bubble memory from Intel makes the most of this opportunity in terms of its ease of both use and manufacturing.

Despite the quadrupled bit density, the 7114's leaded package is smaller in all three dimensions than the leadless package of its 1-Mb predecessor, the 7110. It occupies less space on a printed-circuit board and has a lower profile—low enough for the boards carrying it to fit into adjacent rather than alternate slots in standard card cages. Moreover, chip and package are far easier and cheaper to assemble.

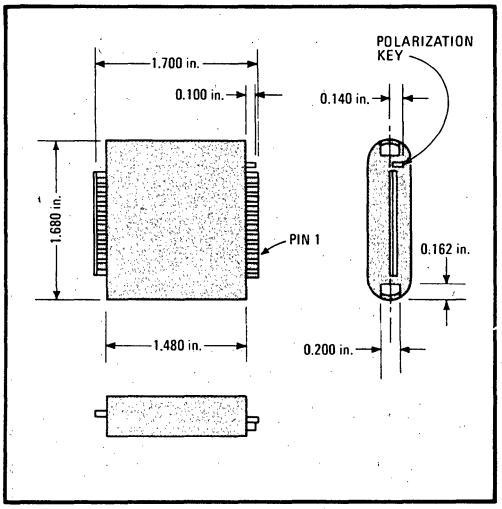
Nor is that convenience compromised by a lack of compatibility with the 7110. The pinouts are the same, and the pin spacings sufficiently similar to make it simple to upgrade from the 7110 to 7114. Also, the support circuits essential to the control of each bubble memory are either identical or so alike as to be interchangeable.

More specifically, the first-generation 1-Mb bubble device is a 520-by-620-mil chip in a leadless package that needs a socket; the assemblage has a footprint of 2.20 by 1.825 in. (Fig. 1) and an overall height of 0.430 in.

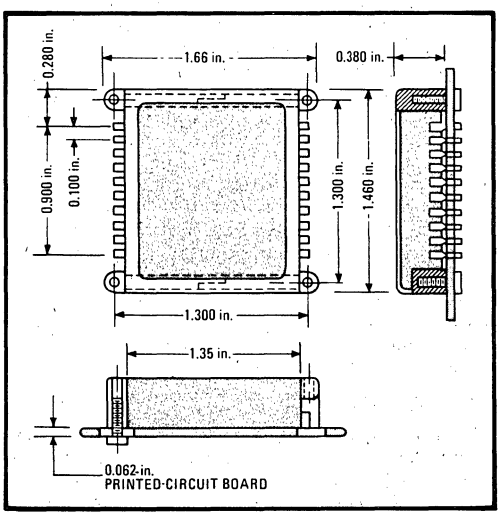
In contrast, the 4-Mb chip and a forthcoming 1-Mb device are smaller—580 by 500 mils—and their leaded "thin-C" dual in-line package has a footprint of only 1.66 by 1.46 in. (Fig. 2). Also, its height is now only 0.375 in., so that when inserted either directly into a pc board or in a zero-profile socket, it leaves ample clearance for the 0.6-in. card spacing normal in commercial card cages.

Design goals

Without scaling down device geometries, it would have been impossible to fit a garnet chip containing four times as many bubble domains into a package of the same size, let alone a smaller one. Thus the first order of business was at least to halve device geometries in both dimen-



1. Leadless. The leadless package of this first-generation 1-megabit magnetic-bubble memory has a footprint of 2.20 by 1.825 inches and an overall height of 0.430 in. when socketed. Thus boards cannot be spaced the standard 0.6 in. apart.



2. Thin and leaded. Despite its 4-Mb capacity, this bubble chip fits in a leaded package with only a 1.66-by-1.46 in. footprint and 0.375-in. profile. Cards carrying these packages or using zero-profile sockets for them can be set into a card cage with the standard 0.6-in. spacing.

sions. The production application of X-ray lithography, in fact, yields 4-Mb bubble chips that are smaller than the 1-Mb one in the current 7110 leadless package.

In addition to different die dimensions, the smaller package required smaller magnets and coils with different dimensions to help control the flow of magnetic bubbles on the garnet chip. A program for designing models of coil size and shape was therefore developed and run on an IBM Personal Computer.

Either the 1- or 4-Mb scaled-down bubble device could have been placed in the same package as the original 1-Mb memory if that had been desired. Instead, it was decided not to settle for the existing package but to produce a new one that, while compatible with the 7110, would be more useful to the engineer—namely, by being smaller and allowing standard pc board spacing. Equally important, if not more so, was the decision to make the production process more efficient and cost-effective.

Nevertheless, there were to be no compromises in the stiff specifications for durability, magnetic shielding, and temperature range. In essence, the design goal was for the new package to be at least as good as the first in some aspects and better in others.

A thinsy

One major concern in moving to a new package is its effect on users who are already producing systems containing the first-generation version and who plan to continue manufacturing while introducing the later one. Unless pinout and spacings are absolutely identical, the transition to a new package cannot be totally painless.

However, in this case, maintaining the identical spacings both within and between the two rows of pins would eliminate any benefit gained by a smaller package. Thus, the decision was to keep the 7114's pinout and adjacent pin spacings the same as on the leadless 7110 package. Only the separation between the two rows of pins has been made smaller on the new memories.

As a result those engineers now manufacturing equipment using the previous 7110 model can lay out their pc boards in such a manner as to accommodate either the first-generation package or, with a minimal amount of revision, the new one. The trick is to elongate and drill the pin trace pads on the board for two holes per pin, as shown in Fig 3. For new layouts, this scheme should be used from the very start. Existing system boards can be modified this way with little effort. If board-level diagnostic and maintenance operations require the use of sockets for the packages, the pc-board holes should be dimensioned for the zero-profile–socket contacts, such as Augat Holtite types. Otherwise, the advantage of the 0.6-in. board spacing will be lost.

Attacking manufacturing costs

In many ways, making magnetic-bubble chips is similar to making integrated circuits, but there are significant differences—for instance, semiconductors do not need wire coils. Therefore, it is reasonable to assume that the major manufacturing cost factors in the one process will not be the same for the other.

In the original 7110, the garnet bubble chip is first die-bonded to a ceramic substrate and then wire-bonded to conductors metalized onto the ceramic. Next, the field and drive coils are put in place around the garnet chip, and all the components are potted using a liquid epoxy compound. Both the use of ceramic and the potting process contribute heavily to memory cost.

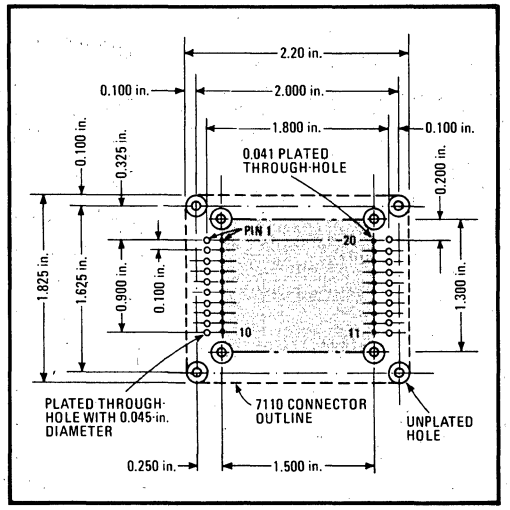
The 7114 package design is more economical on both counts. For the ceramic substrate, the design team substituted a special pc-board material that is thinner, lighter in weight, and lower-cost.

Next, the designers tackled the problem of potting the chip, substrate, and coil combination. Packages containing ICs often employ transfer molding of a thermoset epoxy compound. But when applied to the bubble assembly, the high pressures involved in this process—about 500 to 1,000 pounds per square inch—routinely deformed the bubble memory's wire coils and degraded their electrical performance unacceptably.

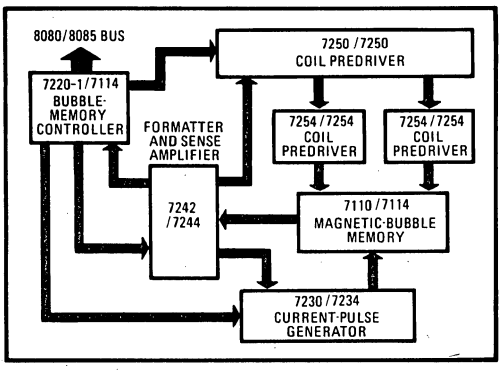
Indeed, for the 7110, manual potting had seemed unavoidable, even though production personnel spent an average of 1 hour on each bubble assemblage, first placing it in a mold, then pouring in liquid epoxy, placing it in a high-temperature oven to cure, removing the mold from the oven, and finally extracting the assembly.

Nonetheless, for the 7114 a fast alternative was found in the liquid injection-molding process used by some manufacturers for high-voltage insulators. Unlike transfer molding, it employs only low pressure, in the region of 15 psi, and it speeds up the potting process to more than 50 devices per hour.

Thus by replacing the ceramic substrate with pc-board material and by substituting liquid injection molding for the manual pouring of a potting compound, the package engineering team had a very favorable impact on the cost and throughput of manufacturing. An added advantage is that, with the ceramic removed, the possibility of chip-



3. Four for one. If this pc board layout is followed, it is possible to plug either a socketed 7110, a leaded 4-Mb, or a leaded 1-Mb bubble-memory package into this hole pattern. The socketed 7110 goes into the outer set of holes, while the newer packages fit the inner set.



4. Upgrade. Along with the bubble chip, all the other ICs of the 1- and 4-Mb memory systems are mechanically and electrically interchangeable so far as the pc board layout is concerned. The diagram shows the chip set for a 4-Mb system in color and the 1-Mb system in black.

ping the exposed edges has been reduced. The table on this page summarizes the packaging and manufacturing aspects of the new and old bubble package methods.

There is always the danger that improving one aspect of a product may inadvertently degrade another. In this case, however, that has not been the result. For example, the original 7110 package was designed to offer protection from external magnetic fields to a level of 20 oersteds. Furthermore, the 7110 is specified to operate over a standard temperature range of 0° to 75°C.

In each case, the new package offers the same or better specifications than the leadless package. In terms of mechanical reliability, both the leadless and leaded package meet and exceed all vibration and shock test limits specified in MIL-STD-883. The leaded package precludes many mechanical problems since it does not depend on a leadless package socket for interfacing with the board.

One for one

The original 1-Mb bubble memory was developed along with a set of support ICs that handled all of its complex timing and drive functions and made its interface with a microprocessor bus indistinguishable from that of any *bona fide* peripheral semiconductor. Considerable engineering effort was applied to make these support circuits interchangeable.

Consequently, any support chip works with any bubble memory. This is in marked contrast to otherwise similar devices that need matched sets of support components.

Intel's 7114 4-Mb device uses the same architecture as the 7110, but now has eight identical sections (called octants) instead of four, and each section is enlarged to store double the number of bubbles it does in the current 7110. The result is a fourfold increase in capacity. However, all of the same pins are brought out in the same order on both bubble memories. Thus the pinout is identical and allows for the use of the same new package for both the new 1-Mb and 4-Mb devices.

Those support ICs that are not affected by the increased capacity, such as the coil predriver (7250) and drivers (7254), are used with either memory.

FIRST VERSUS SECOND GENERATION MAGNETIC BUBBLE MEMORIES

Type	7110 Leadless	7110 Leaded Thin C	7114 Leaded Thin C
Memory (Mb)	1	1	4
Die size (in.)	0.620 by 0.520	0.580 by 0.500	0.580 by 0.500
Substrate material	ceramic	printed-circuit board	printed-circuit board
Potting technique	liquid epoxy by hand	liquid injection molding	liquid injection molding

Those support ICs that have been designed in conjunction with the new memory are the same size and have the same pinouts as their counterparts in the 7110 1-Mb subsystems. What has changed is some of the programmable parameters and the descriptions of their associated registers. These are all involved with the new bubble-memory controller chip—the IC that interfaces the microprocessor bus with the 4-Mb memory.

Examining the effect of these changes in upgrading from a 7110 1-Mb to a 7114 4-Mb package reveals that the modifications are really minimal—the new support ICs can be designed into the same board layouts as their 1-Mb cousins. Users will have only to make some modifications in their software to handle minor differences in addressing and configuration initialization.

Intel's bubble-memory system therefore can have the same configuration whether working with 1-Mb or 4-Mb devices. A single bubble-memory controller acts as the interface between the microprocssor bus and one or more bubble storage subsystems.

The 7224 controller for the 4-Mb bubble-memory device is housed in a standard 40-pin, dual in-line package and takes up about 2 by 0.5 in. on a board. A single controller operates up to eight storage subsystems for a maximum capacity of 4 megabytes.

Each bubble-memory subsystem contains a monolithic formatter and sense amplifier in a standard 20-pin DIP; a current-pulse–generator chip in a 22-pin DIP; a coil-predriver chip in a 16-pin DIP; a pair of quad V-groove MOS driver chips, each in a 14-pin DIP; and of course the bubble device itself. One subsystem takes up less than 3 by 4 in., and a 4-megabyte board of eight of them plus a a controller could be constrained to 6.75 by 12 in.

Obviously, boards laid out for the 7110 1-Mb memory and its support family would be approximately the same size for one fourth the amount of memory. However, in a card cage with a standard 0.6-in. spacing, boards built using the original leadless packages could not be stacked in adjacent slots.

Converting from the earlier 7110 1-Mb to a 4-Mb system essentially requires modifications to the pin pads of the 7110's leadless package. The 7220-1 bubble-memory controller is the same size and has the same pinout as the 7224. The same is true for the 7242 formatter and sense amplifier and its 7244 replacement, as well as for the 7230 current-pulse generator and its 7234 substitute. Figure 4 shows how an identical board can support either a 7114 4-Mb or a leaded or leadless 7110 1-Mb system.

Space limitations, interface details, and other such criteria will typically dictate the actual board layout. □

intel

ARTICLE REPRINT

AR-272

April 1983

Bubble-Memory Support Chips Allow Tailored-System Design

Richard Pierce
Applications Engineer
Intel Corporation

APRIL 1983
ORDER NUMBER: 230645-001

Bubble-memory support chips allow tailored-system design

Using special support chips gives you flexibility in designing a bubble-memory system. And understanding design tradeoffs helps configure a system that best suits your needs.

Richard Pierce, Intel Corp

Designing a bubble-memory system—with its advantages of small size, high reliability and nonvolatility—is easy when you use system support devices. Such a family of LSI chips allows you to tailor a system to meet requirements on specs such as access time and power consumption and to modularly expand the system for increased storage capacity. How you configure and use a bubble-memory system involves tradeoffs, though, and balancing those tradeoffs requires knowledge of the intended system application. This article discusses bubble-system tradeoffs and other design considerations and presents a specific design using Intel's 1M-bit bubble-memory device and family of support chips.

Interface appears as a peripheral controller

The devices available for building a 1M-bit bubble-memory system are the 7110 magnetic-bubble module, the 7220-1 bubble-memory controller, the 7242 formatter/sense amplifier, the 7230 current-pulse generator and the 7250 coil predriver (see **box,** "Bubble-memory devices"). Communication with the 7220-1 controller (**Fig 1**) is the key function, because this device provides the bubble memory's only interface with the outside world. It allows you to interface with a bubble-memory system through a standard μP bus just as with any peripheral. Software directs the controller to choose one of three memory-access methods: direct memory access (DMA), interrupt or poll.

The 7220-1 bubble-memory controller (BMC) has several functional sections. The system-bus interface provides an asynchronous interface to a host processor, transferring 12.5k bytes/sec with a 4-MHz system clock. The controller also has a 40-byte first-in first-out (FIFO) memory buffer through which data passes on its way to the 7242 formatter/sense amplifier (FSA). This FIFO's primary purpose is to reconcile timing differences between the user and the FSA, because the system bus typically can transfer data much more rapidly than the bubble memory can.

The BMC's DMA and interrupt logic handle data transfers. In addition, an internal register file contains six user-accessible 8-bit registers: a command register, a status register and four registers that store information pertaining to the system's operational mode and configuration. The command register accepts user-issued codes that initiate data transfers, and the status register indicates the BMC's current state.

The remaining functional sections of the BMC

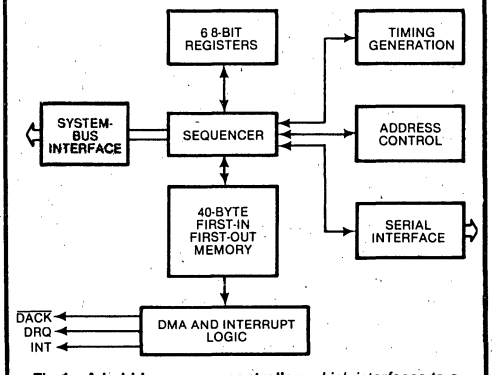


Fig 1—A bubble-memory controller, *which interfaces to a host μP through a system bus, generates all support-circuit timing and control signals, maintains memory-address information and performs data conversions for transfers to a bubble-memory module.*

Communicate with a bubble memory as with a peripheral controller

generate all the support-circuit timing and control signals, maintain memory-address information and perform parallel-to-serial and serial-to-parallel conversions for transfers to the bubble-memory module.

Interface circuitry depends on transfer mode

The type of circuitry you design to interface with the BMC depends on the mode of data transfer chosen. For a polled implementation, the requirements reduce to interfacing to a standard μP bus. One such interface design, for the 8088, appears in **Fig 2**; it consists of address-decode logic, data-bus-decode and buffering logic, a clock circuit and miscellaneous control logic. The clock circuit must provide a 4-MHz (±0.1%) system clock with a 50% (±5%) duty cycle.

Fig 2's system operates from 12 and 5V only, and one important part of this design is its automatic power-fail circuitry (**Fig 3**), which monitors these voltages. An important aspect of bubble-memory devices is that the coil drive current (which moves bubbles around within the device) must always have the proper phase and amplitude, without transients, to ensure data integrity. **Fig 3**'s power-fail circuit prevents transients and—when voltages drop 6% below normal level—stops the coil currents while maintaining the proper phase. You can also expand the power-fail circuit to include recommended features such as ac power-fail and ac or dc overvoltage protection.

Bubble-memory devices

Several different devices **(figure)** make up a bubble-memory system. At the heart of the system is the 7110 magnetic-bubble memory (MBM), with a user data capacity of 1M bits (128k bytes).

This chip embodies a major-track/minor-loop architecture, in which bubbles serially propagate into and out of the memory tracks and get stored in minor (storage) loops (EDN, September 1, 1982, pg 198). This organization permits creation of redundant storage loops, increasing device yield by tolerating defects in as many as 15% of the loops. Testing by the manufacturer detects the unusable loops and writes a map—showing the usable loops—into an additional storage loop called a boot loop. The map also appears on the 7110's label.

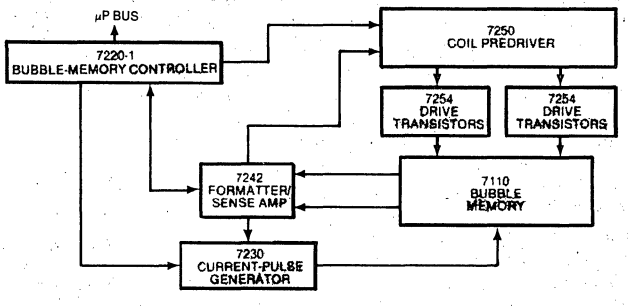
Internally, the 7110 consists of two identical 512k-bit sections. Although these sections are essentially independent, they operate simultaneously to give a factor-of-two data-rate improvement.

User interface to the system occurs through a 7220-1 bubble-memory controller (BMC). This device provides the system-bus interface, performs serial-to-parallel and parallel-to-serial data conversions, generates all timing and control signals necessary for proper operation of support circuitry and interprets and executes user requests for data transfers. The BMC's interface makes the bubble-memory system look like a peripheral to a μP-system bus.

The 7242 formatter/sense amplifier (FSA) interfaces independently to each half of the bubble memory. Its integrated sense amplifier accepts low-level voltage signals from the MBM's bubble detectors during read operations, and the device also performs data-formatting tasks that include the transparent handling of the MBM's redundant loops. In addition, the FSA sends TTL-level control signals to the 7230 current-pulse generator (CPG) during write operations.

The 7230 supplies the current pulses that generate bubbles in the MBM and transfer them into and out of the storage loops. The CPG's integrated power-fail-detection circuitry initiates an orderly shutdown of the current sources when power fails.

The 7520 coil predriver (CPD) interfaces the 7220-1 BMC to the two 7254 drive transistors, which supply the relatively high peak currents required by the 7110 MBM's X and Y coils. These currents induce the in-plane rotating magnetic field that moves the magnetic bubbles.



Support chips *allow the design of a complete bubble-memory system around Intel's 7110 magnetic-bubble memory device.*

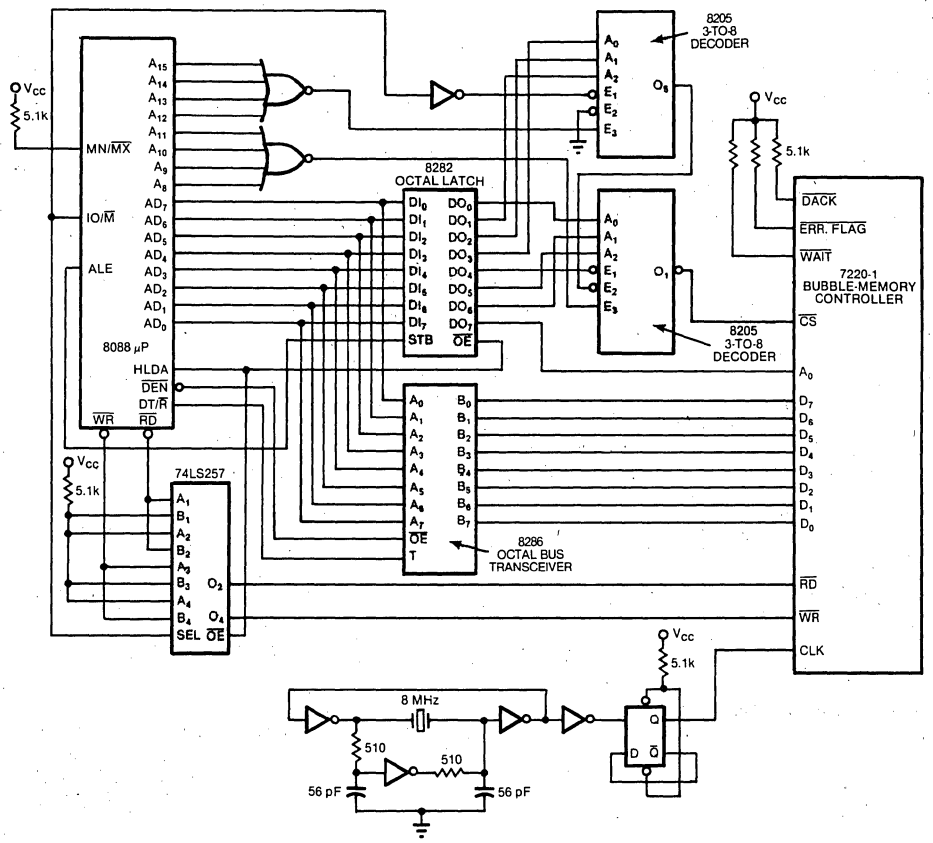


Fig 2—Interfacing a bubble memory *can be almost as simple as interfacing a standard μP bus. This interface connects a 7220-1 bubble-memory controller with an 8088 μP.*

Guaranteeing power-supply requirements is an important part of the design process. The supply voltages must be within 5% of their specified values, and the power-off/power-fail decay rates (**Table 1**) must allow 150 μsec max for an orderly shutdown and reset of all support circuits. No restrictions apply to voltage rise times or sequencing.

A simple calculation determines your system's required storage capacitance to achieve **Table 1**'s voltage decay rates. The worst-case power-supply capacitance requirement is

$$\frac{Q_{MAX}}{V_{MIN}} = \frac{\Delta I_{MAX} \Delta T_{MAX}}{\Delta V_{MIN}}$$

Typical capacitance values for a system with one bubble-memory module, excluding any additional current drain from unrelated circuitry, are 805 and 350 μF for the 5 and 12V supplies, respectively.

Another important design factor in custom bubble-memory boards is careful circuit layout to minimize interference from the 7110's large drive signals with

Choose a memory-access method: direct, interrupt or poll

nearby small sense signals. The pin assignments of the 7110 and its support devices optimize board layout and maximize circuit density, and the package layout used in Intel's Bubble Prototype Kit (BPK-72) helps ensure error-free operation. As shown in **Fig 4**, this layout places all support circuits near the 7110; note particularly that the 7110 and the 7242 FSA must not be physically separated.

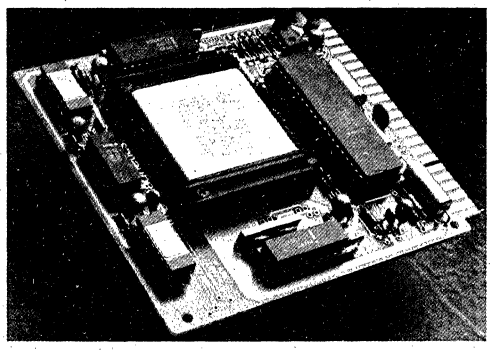


Fig 4—Proper component layout *helps ensure error-free bubble-memory operation. In this Intel Bubble Prototype Kit (BPK-72), all support circuits are close to the 7110 memory module.*

Software controls the system

Software is a vital part of bubble-memory design, too; it's the major contributor to a system's efficient and reliable operation. Software interface modules (often called bubble drivers) accept processor-issued commands and control and monitor command execution; they also return status information to the processor.

Software drivers depend on the mode of data transfer: interrupt-driven I/O, DMA or polled. In a DMA implementation, the software need only set up the DMA controller's memory-address and transfer counts and initiate the data transfer; Intel's 8257 DMA-controller hardware automatically handshakes with the μP and BMC to perform each transfer.

In the interrupt mode, on the other hand, the software is responsible for performing memory-read and -write operations to transfer 22-byte data blocks to and from the BMC's FIFO on receipt of an interrupt. A BMC output signal—typically wired as a second-level interrupt—indicates when the BMC's FIFO becomes half full (during read operations) or half empty (during write operations).

The third I/O method—polling—is similar to the interrupt-driven mode except that in it, data moves one byte at a time. The software determines when to transfer data by continually polling a bit in the BMC's status register. This status bit indicates the presence or

TABLE 1 — BUBBLE-MEMORY POWER-SUPPLY REQUIREMENTS

VOLTAGE	MARGIN	POWER-OFF/POWER-FAIL DECAY RATE
12V	±5%	< 1.10 V/mSEC
5V	±5%	< 0.45 V/mSEC

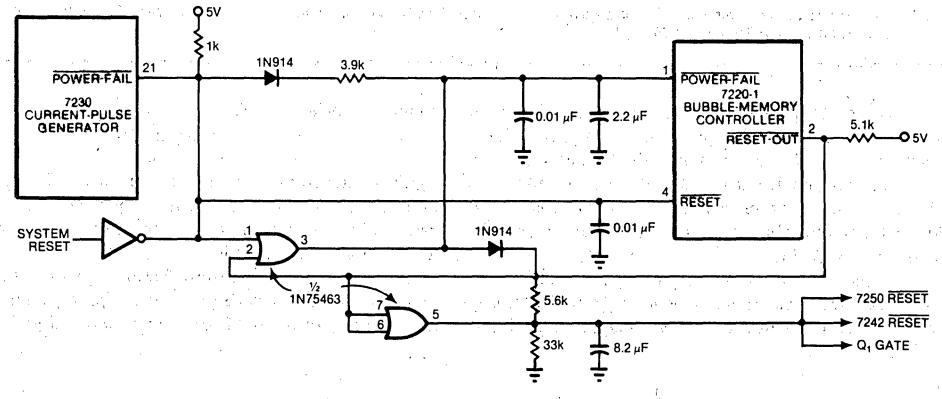


Fig 3—Power-fail circuitry *monitors the 12 and 5V supplies for* **Fig 2***'s circuit and maintains data integrity in case of power failure by stopping coil currents in the proper phase.*

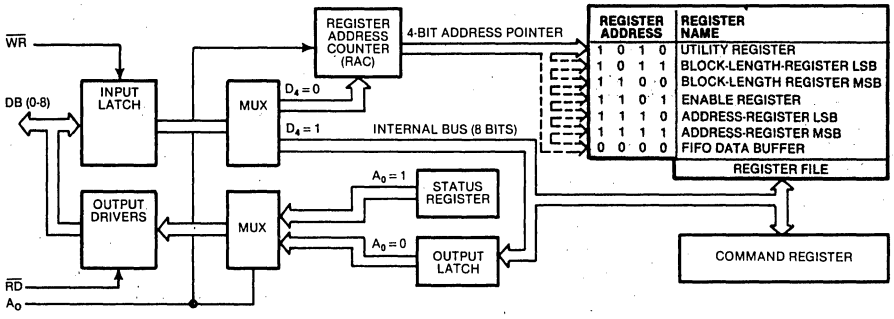


Fig 5—Software accesses the bubble-memory controller *through a system-bus interface consisting of two I/O ports. A ONE on address line A_0 selects the command/status port, and a ZERO selects the bidirectional data port.*

absence of data—to be read or written by the host processor—in the BMC's FIFO.

Although the polling mode is simple to implement, its software requirements are the most demanding. Because data transfers one byte at a time, the software must continually monitor the status register to ensure that the FIFO doesn't underflow or overflow with data.

Each of these data-transfer modes has unique advantages that must be weighed with each particular application. The DMA mode, for example, permits the processor to continue executing instructions while a transfer is in progress. The interrupt and polled modes, on the other hand, offer lower cost but are often too slow in systems incorporating multiple bubble devices connected in parallel. Eight 7110 bubble modules in parallel can transfer data at rates as high as 100k bytes/sec, and these high-performance systems normally use the DMA mode.

Understand protocols and definitions

Developing the software drivers for any bubble-memory system requires a clear understanding of command protocols and register definitions. The software communicates with the bubble-memory controller through the system-bus interface, consisting of two I/O ports selected by the state of the least significant address line (**Fig 5**). When A_0 is ONE (active), the command/status port gets selected; when it's ZERO (inactive), the bidirectional data port gets selected.

The command/status port serves a dual function: Reading the port accesses the status register, and writing to the port accesses a register determined in part by data bit D_4. This register is the command register if D_4 is ONE; it's one of the six parametric registers or the BMC's FIFO if D_4 is ZERO. In the latter case, the contents of a register-address counter (RAC) specify one of the seven possibilities.

The command register accepts one of 16 interface commands. The initiation of nondata transfers occurs merely by writing the proper command code to the command register, while a data transfer (initialize or read/write) requires prior loading of the parametric registers. Those registers specify the operating mode and system configuration, plus the data transfer's starting address and length (in 64-byte pages). After loading of the command register, the BMC automatically executes the command. An indication of success or failure returns in the status register.

An autoincrementing feature in the RAC facilitates loading the parametric registers before data-transfer commands. After the processor loads a value—specifying a particular parametric register—into the RAC via the command port, the next write operation to the data port accesses that register. Each write operation also increments the RAC, so each subsequent operation accesses the next register in sequence. After incrementing to address 0 (the BMC's FIFO), however, the RAC stops incrementing and continues to point to the FIFO until modified by software.

Software drivers perform data transfers

An example set of BMC software drivers (**Fig 6**) shows how data-transfer operations occur in the polled mode. The three drivers perform a power-up procedure, write registers, and read and write data.

The power-up procedure (**Fig 6a**) enables bubble-memory support devices in an orderly fashion and permits subsequent reading and writing of memory

Meet power-supply requirements to ensure reliable operation

pages. The first command it issues is Abort; the host then loads the parametric registers with appropriate values for a 1M-bit system configuration and issues an Initialize command. During command execution, the host processor constantly polls the BMC's status register to determine when the command finishes. An additional status-register check determines whether the command and/or data transfer is successful.

Successful completion of the Initialize command

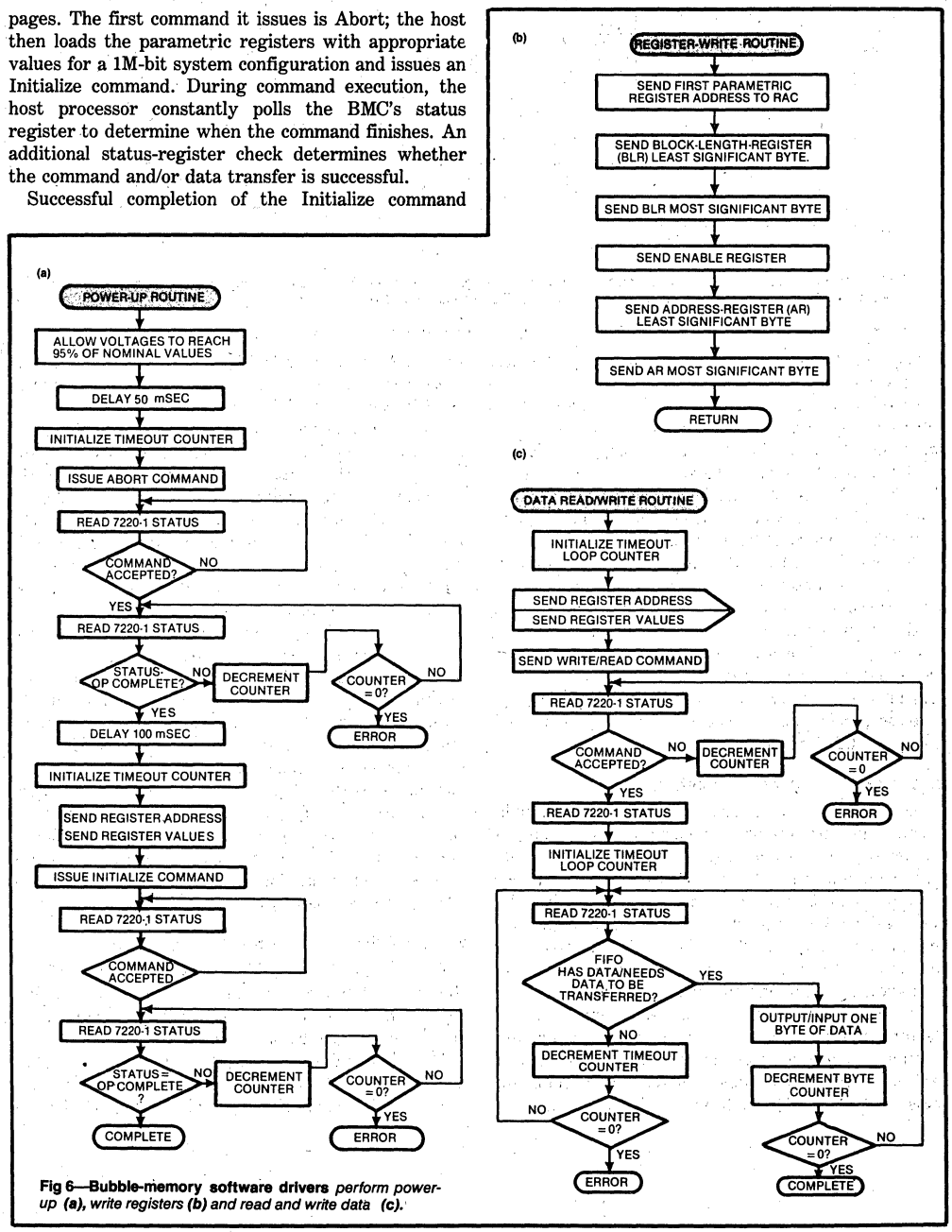


Fig 6—Bubble-memory software drivers *perform power-up* ***(a)****, write registers* ***(b)*** *and read and write data* ***(c)****.*

Proper physical layout helps reduce errors

indicates that the system is ready to transfer data. To initiate a data transfer, the host loads the parametric registers (**Fig 6b**) with the memory-page address and number of pages to transfer and then issues a read or write command (**Fig 6c**). This command transfers data between the system bus and the BMC's FIFO at a rate of 80 μsec/byte, excluding access time. The software polls the status register to determine when to transfer each byte and also maintains a count of the bytes transferred. In systems operating in the DMA or interrupt mode, however, maintaining this count is unnecessary.

Of course, all software driver routines should contain error-handling capabilities, but the examples shown here ignore these capabilities for the sake of simplicity. Numerous error-correction options exist in a bubble-memory system, though, and all are selectable under software control. As an example, the 7242 formatter/sense amplifier uses a 14-bit error-correction code with each 256-bit block of data. This code can correct all single-error bursts of five bits or less, improving the data error rate by several orders of magnitude while remaining user transparent.

Expand memory in modules

Having considered a bubble-memory-system design, turn to memory-expansion considerations. One BMC can control as many as eight bubble-memory modules, and with multiple BMCs, you can configure even larger systems (**Fig 7**). A memory module—providing expansion in increments of 128k bytes—consists of a 7110 bubble device, one 7230 current-pulse generator, one 7250 coil predriver, a 7242 formatter/sense amplifier and two 7254 drive-transistor packages.

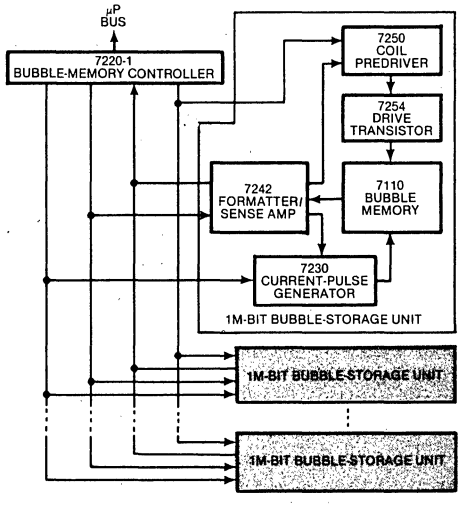


Fig 7—Bubble-memory expansion *occurs in increments. Each 7220-1 controller can handle as many as eight 128k-byte modules.*

Expansion can occur in three ways. The first approach uses the BMC's built-in ability to handle multiple bubble devices. This scheme relies on the BMC to time-slice the serial bus between the BMC and the appropriate number of 7242 FSAs in the system and to output appropriate control signals to the 7242, the 7230 and the 7250. Data flow in this expanded system is similar to that in a single-module system.

A second expansion approach takes advantage of provisions in the BMC for paralleling controllers. This approach provides a greater word width at the system bus and still allows each BMC to accommodate as many as eight bubble devices.

A third option switches banks of bubble devices into or out of a circuit under external control. (Switching is possible because each support device has a chip-select pin.) The maximum number of devices in each bank remains eight, but the number of banks is unlimited. You can even multiplex entire subsystems of this type, because the BMC chip itself has a chip-select input.

Nonvolatility permits power savings

Your design of a bubble-memory system can also take into account the system's nonvolatility in order to minimize power consumption. Consider, for example, a bubble system for a portable terminal that stores inventory and sales figures and periodically transmits them to a central computer. The bubble memory in such a terminal requires power only during memory access, so you can remove power from the bubble system during much of the time that the terminal is in use.

In some applications—for example, systems that only occasionally transmit a small amount of data—you can also reduce power consumption by using a faster initialization scheme. The usual initialization procedure, which reads boot-loop information from a bubble device to identify redundant storage loops, requires as much as 160 msec per device—a significant percentage of power-on time for such systems. If you store the boot-loop information in EPROM, though, you can download it from there much more rapidly and thus with less power.

Design tradeoffs occur at system level

Still other design considerations exist at the system level; design tradeoffs concern such factors as memory capacity, access time, data rate and power consump-

Software interface modules contribute to system efficiency

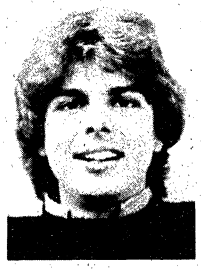
TABLE 2 — BUBBLE-MEMORY PERFORMANCE PARAMETERS

	ONE MBM	FOUR MBMs	EIGHT MBMs OPERATED IN PARALLEL	EIGHT MBMs MULTIPLEXED ONE AT A TIME
CAPACITY	128k BYTES	512k BYTES	1M BYTES	1M BYTES
NOMINAL DATA RATE	68 kHz	272 kHz	544 kHz	68 kHz
AVERAGE ACCESS TIME	48 mSEC	48 mSEC	48 mSEC	48 mSEC
POWER DISSIPATION (100% DUTY FACTOR)	6W	20W	40W	11W
STANDBY POWER	1.55W	3.7W	7.0W	7.0W
BOARD AREA	16 IN.^{2}	45 IN.^{2}	90 IN.^{2}	90 IN.^{2}

tion. As **Table 2** shows, the 7110 bubble module's access time averages 48 msec (7.4 msec best case, 80 msec worst case), independent of the number of modules in a system. You can increase the data rate, however, by operating the devices in parallel. The approach requires more power, but using eight 7110s instead of one increases the nominal bit rate from 68 to 544 kHz.

Finally, you can improve the overall data rate by reducing the average time required to access a memory page. The access time for any single page is random, but no access delay occurs for succeeding pages. Thus, in time-critical applications where successive page accesses aren't random, least recently used (LRU) and lookahead algorithms can help reduce page-access time. **EDN**

Author's biography



Richard Pierce works in Intel's Nonvolatile Memory Div (Santa Clara, CA), supplying customer support and developing new applications. He holds a BSEE degree from Purdue University and previously worked for Cincinnati Electronics. Dick is a member of the IEEE, and his hobbies include biking, photography and skiing.

ADVANCE INFORMATION

BPK 5V74
4MBIT BUBBLE MEMORY SUBSYSTEM

BPK 5V74-4	10°C To 55°C

- **4 Mbit (512K Bytes) Non-Volatile, Solid-state, Read/Write Bubble Memory Subsystem**
- **Interfaces to Host Microprocessor Via Additional Bubble Memory Controller**
- **Contains Bubble Memory and ICs for Production with 4Mbit Bubble Memory**
- **Modularity Provides Expansion Up to Eight Subsystems Per Controller**
- **Maximum Data Rate of 200K bit/sec with One Subsystem**
- **Maximum Data Rate of 1.6M bit/sec with Eight Subsystems in Parallel and Time Multiplexed**
- **Average Random Access Time of 88 ms**
- **Bubble Memory in Leaded Package**

The BPK 5V74 Bubble Memory Subsystem is a modular building block used to design bubble memory systems. In a complete bubble memory system, an 7224 Bubble Memory Controller (BMC) interfaces the BPK 5V74 subsystem to the host processor.

The modular Intel subsystem provides a path for density expansion. One BMC can interface up to eight 4 Mbit subsystems. Thus, a 4 MBit (512 KByte) system can be expanded up to a 32 MBit system by adding subsystems. BMC's can be combined in parallel to further expand the system memory capacity.

Together, the BPK 5V74 Bubble Storage Subsystem and a 7224 controller provide a reliable mass storage system for any application. This bubble memory system can be customized to the particular layout and form factor of many different systems.

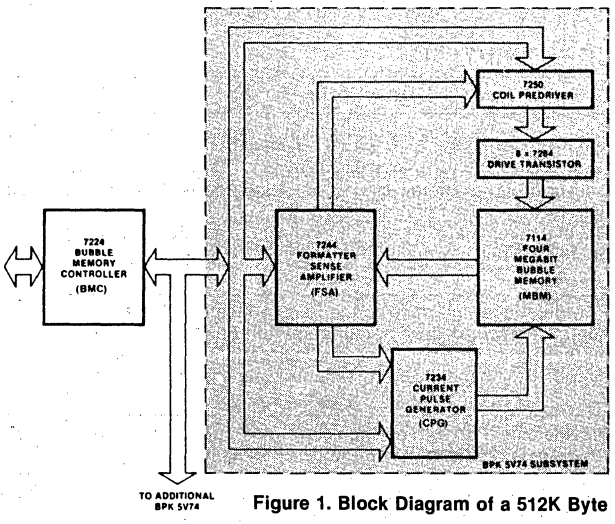


Figure 1. Block Diagram of a 512K Byte Bubble Storage System

OCTOBER 1983
ORDER NO. 210804-002

FUNCTIONAL DESCRIPTION

An BPK 5V74 subsystem and a 7224 controller comprise a complete bubble memory system. The 4 MBit BMC, the 7224, provides the interface between the host microprocessor and the bubble memory subsystem and provides all the timing and control signals to the subsystem. The user interface of the BMC is compatible with microprocessor bus systems for 8080, 8085, 8086, 8088, 80186, 80286 and other standard microprocessors. The BMC is a software driven device utilizing 18 convenient commands. The design engineer's primary responsibility is interfacing to the BMC. This is comparable to interfacing a disk drive controller.

The BPK 5V74 consists of one 4 MBit Magnetic Bubble Memory (MBM) and additional support IC's (see Figure 1). These are the basic components to build a non-volatile, solid-state, read/write military memory system utilizing 4 MBit bubble memory. The bubble memory is in a leaded package. The complete family of LSI support circuits has been designed to handle the complex analog interface associated with bubble devices. The immediate support circuitry for the MBM consists of — an 7250 Coil Predriver (CPD), eight 7264 MOS FETs Transistor Packs, an 7234 Current Pulse Generator (CPG), and an 7244 Formatter/Sense Amplifier (FSA).

Data integrity is insured by the automatic error correction designed into the BPK 5V74.

The average random access time of a 4 MBit subsystem is 88 ms with a 200Kbit/sec maximum data transfer rate. Operating several subsystems in parallel, the BMC uses time division multiplexing. Therefore, the maximum data rate increases correspondingly for the whole system.

Operating subsystems serially, one MBM being accessed at a time, the maximum data transfer rate is still 200Kbit/sec. If low power consumption is a critical design goal, the bubble memory subsystem can be powered down when it is not being accessed, thus reducing the average power consumption.

The data in the 4 Mbit subsystem is organized in 8192 pages, each with 64 bytes. Conceptually, the data organization with pages is analogous to a disk system's sectors. In system's with multiple bubble memories, the page size can vary from 64 bytes to 512 bytes depending on the number of subsystems and if the subsystems are operating in parallel or serially, being accessed one at a time.

The BPK 5V74 subsystem has matched components. Each of the components in the subsystem is described in more detail in the rest of this data sheet.

BPK 5V74 FUNCTIONAL DESCRIPTION

Item	Description	Part Number
4 MBit Bubble Memory	20-pin leaded package which provides 4 megabit of non-volatile storage.	7114
Current Pulse Generator	Converts digital timing signals to analog current pulses suited to the drive requirements of the MBM. The CPG provides the replicate, swap, generate, boot replicate, and bootswap pulses required by the MBM. 22 Pin DIP Package.	7234
Dual Formatter/Sense Amp	Provides direct interface to the Bubble Memory. The FSA contains on-chip sense amplifiers, a full FIFO data block buffer, burst error detection and correction circuits, and circuitry for handling of the bubble memory redundant loops. 20 Pin DIP package.	7244
Coil Predriver	Provides the high voltage, high current outputs to drive the Quad VMOS transistors. 16 Pin DIP package.	7250
VMOS Coil Drive Transistors (8)	Switches the required current to drive the X and Y coils of the Bubble Memory. 3 Pin Discrete.	7264

For additional packaging information see the packaging information section.

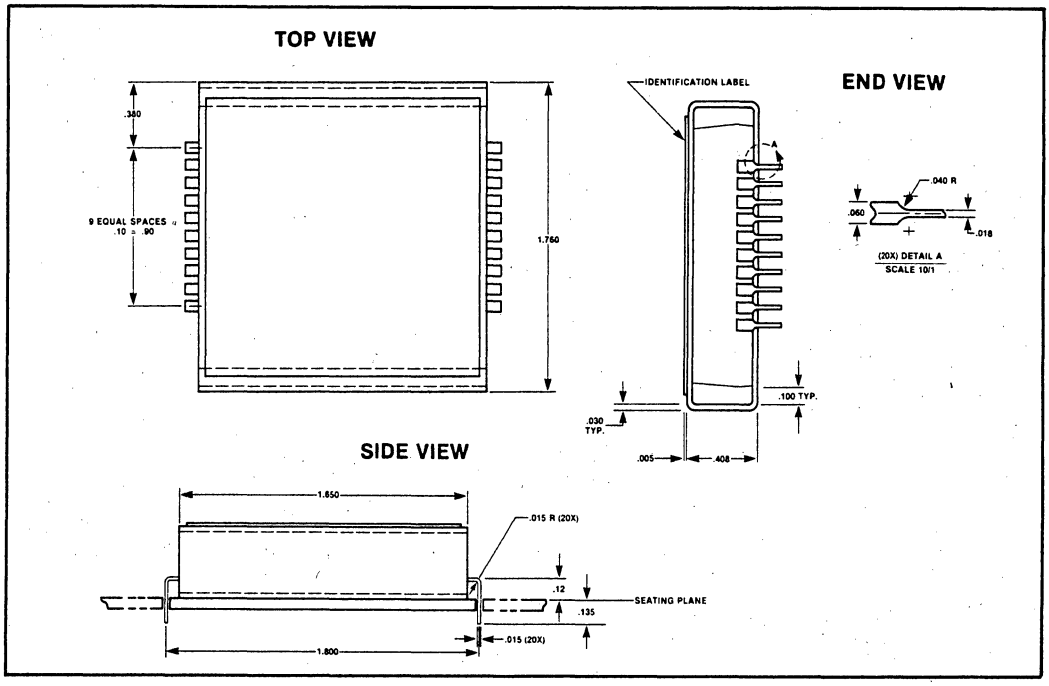


Figure 2. 4MBit Leaded Bubble Memory Package

BPK 5V74 TEMPERATURE RANGE

Bubble Memory Temperature Ranges		Support Circuits Min. Operating Temperature	Description
Operating	Non-Volatile Storage		
10° to 55°C Case	−20 to +75°C	10° to +55°C Ambient	4 Mbit Bubble Storage Sub-System

SPECIFICATIONS

Capacity

512K Byte per BPK 5V74
Maximum of 8 BPK 5V74 per 7224 Controller

Performance

Avg. Access Time..... 88 msec

Data Organization

64 bytes per page
8192 pages per BPK 5V74

Addressing Scheme

Logical page number

Environmental

Temperature: See Ordering Information
Operating Humidity: 0—95% Non-Condensing

DATA TRANSFER RATES (Examples of System Configurations)

Parameter	One BPK 5V74 Unit	Four BPK 5V74 Operated in Parallel[1]	Eight BPK 5V74 Operated in Parallel[1]	Eight BPK 5V74 Multiplexed One at a Time[1]
Capacity	512 kilobytes	2048 kilobytes	8 megabyte	8 megabyte
Average Data Rate (kilobits/sec)	136	544	1088	136
Maximum Date Rate (kilobits/sec) (Burst)	200	800	1600	200

NOTE:
1. Multiple Bubble subsystems can be operated in parallel for maximum performance or multiplexed to conserve power.

BPK 5V74 POWER SUPPLY REQUIREMENTS

Voltage	Margin	Power Off/Power Fail Decay Rate
+12 Volt	±1%	less than 1.10 volts/msec
+5 Volt	±5%	less than 0.45 volts/msec

- Voltage sequencing — no restrictions
- Power on voltage rate of rise — no restrictions
- The power supply requirements based on recommended power fail circuitry as shown in Figure 3.
- The 12V ±1% may be supplied by:
  1. Using such power supply.
  2. Use voltage regulator with 5V ± 5% input and 12V ±1% output as used in BPK 5V75 prototype kit. Circuitry — See Figure 4.

BPK 5V74 POWER CONSUMPTION

(Includes 7114, 7234, 7244, 7250, 7264)

Standby	Typical:	0.7 W
	Maximum:	1.8 W
Active	Typical:	3.7 W
	Maximum:	6.2 W

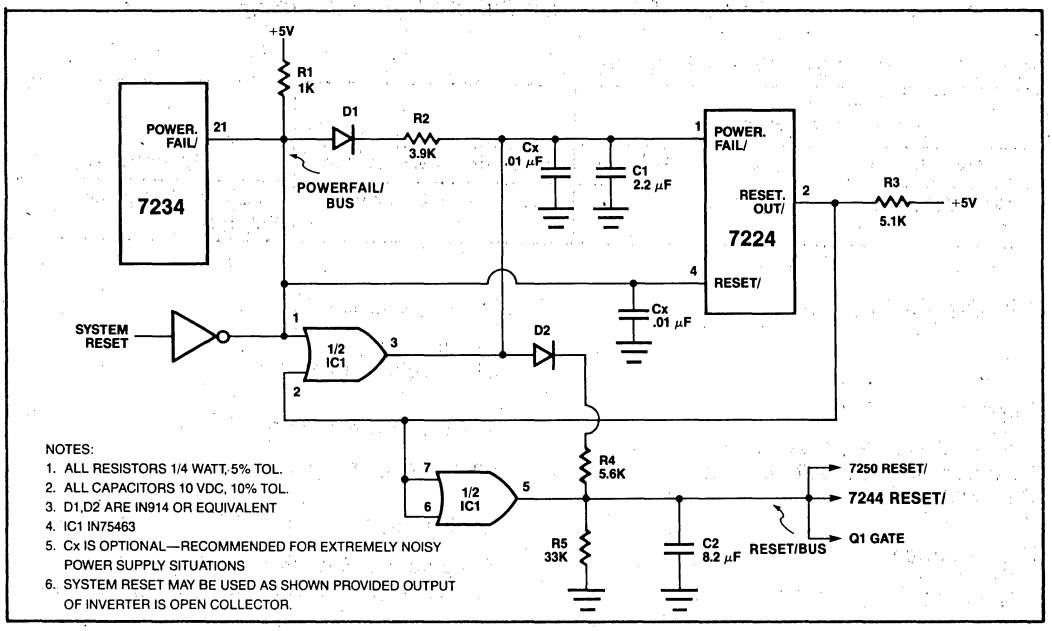


Figure 3. Power Fail Circuit

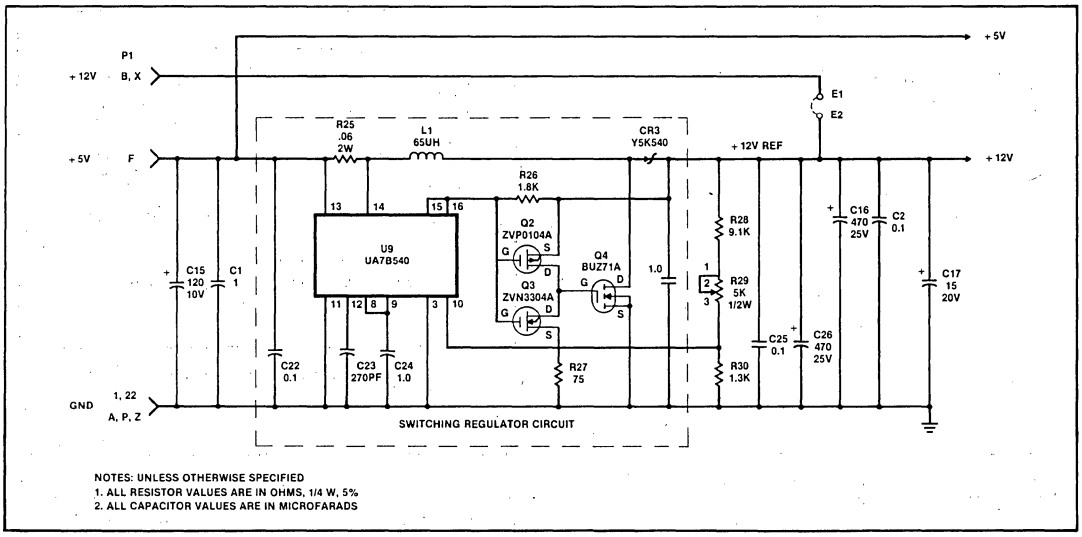


Figure 4. BPK 5V75 Voltage Regulator Circuit

PIN DESCRIPTIONS

7114

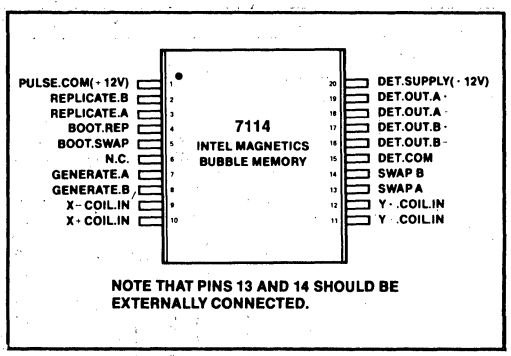


Figure 5. 7114 Pin Configuration

Table 1. 7114 Pin Description

Symbol	Pin No.	I/O	Source/Destination	Description
BOOT.REP	4	I	7234 CPG	Two-level current pulse input for reading the boot loop.
BOOT.SWAP	5	I	7234 CPG	Single-level current pulse for writing data into the boot loop. This pin is normally used only in the manufacture of the MBM.
DET.COM	15	I		Ground return for the detector bridge.
DET.OUT	16 — 19	O	7244 FSA	Differential pair (A+, A− and B+, B−) outputs which have signals of several millivolts peak amplitude.
DET.SUPPLY	20	I		+12 volt supply pin.
GEN.A and GEN.B	7, 8	I	7234 CPG	Two-level current pulses for writing data onto the input track.
PULSE.COM	1	I		+12 volt supply pin.
REP.A and REP.B	3, 2	I	7234 CPG	Two-level current pulses for replicating data from storage loops to output track.
SWAP.A and SWAP.B	13, 14	I	7234 CPG	Single-level current pulse for swapping data from input track to storage loops.
X−.COIL.IN. X+ COIL.IN.	9, 10	I	7264	Terminals for the X or inner coil.
Y−.COIL.IN. Y+.COIL.IN.	11, 12	I	7264	Terminals for the Y or outer coil.

7234

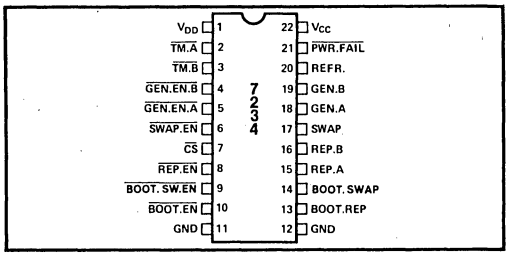


Figure 6. 7234 Pin Configuration

Table 2. 7234 Pin Description

Symbol	Pin No.	I/O	Source/Destination	Description
$\overline{\text{BOOT.EN}}$	10	I	7224 BMC	An active low input enabling the BOOT.REP output current pulse.
BOOT.REP	13	O	7114 MBM	An output providing the current pulse for bootstrap loop replication in the bubble memory.
BOOT.SWAP	14	O	7114 MBM	An output providing a current pulse which may be used for writing data into the bootstrap loop.
$\overline{\text{BOOT.SW.EN}}$	9	I	7224 BMC	An active low input enabling the BOOT.SWAP output current pulse.
CS	7	I	7244 FSA	An active low input for selecting the chip. The chip powers down during deselect.
GEN.A	18	O	7114 MBM	An output providing the current pulse for writing data into the "A" quads of the bubble memory.
GEN.B	19	O	7114 MBM	An output providing the current pulse for writing data into the "B" quads of the bubble memory.
$\overline{\text{GEN.EN.A}}$	5	I	7244 FSA	An active low input enabling the GEN.A output current pulse.
$\overline{\text{GEN.EN.B}}$	4	I	7244 FSA	An active low input enabling the GEN.B output current pulse.
$\overline{\text{PWR.FAIL}}$	21	O	7224 BMC	An active low, open collector output indicating that either V_{CC} or V_{DD} is below its threshold value.
REFR.	20	I	External Resistor	The pin for the reference current generator to which an external resistance must be connected.
REP.A	15	O	7114 MBM	An output providing the current pulse for replication of data in the "A" quads of the bubble memory.
REP.B	16	O	7114 MBM	An output providing the current pulse for replication of data in the "B" quads of the bubble memory.
$\overline{\text{REP.EN}}$	8	I	7224 BMC	An active low input enabling the REP.A and REP.B outputs.
SWAP	17	O	7114 MBM	An output providing the current pulse for exchanging the data between the input track and the storage loops in the bubble memory.
$\overline{\text{SWAP.EN}}$	6	I	7224 BMC	An active low input enabling the SWAP output.
$\overline{\text{TM.A}}$	2	I	7224 BMC	An active low timing signal determining the cut pulse widths of the BOOT.REP, GEN.A, GEN.B, REP.A and REP.B outputs.
$\overline{\text{TM.B}}$	3	I	7224 BMC	An active low timing signal determining the transfer pulse widths of the BOOT.REP, GEN.A, GEN.B, REP.A and REP.B outputs.

7244

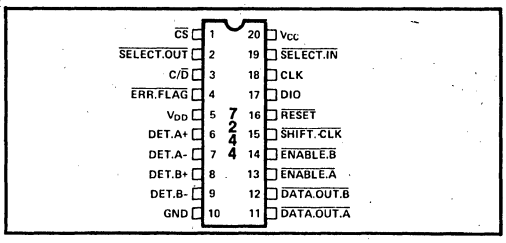


Figure 7. 7244 Pin Configuration

Table 3. 7244 Pin Description

Symbol	Pin No.	I/O	Source/Destination	Description
C/$\overline{D}$	3	I	7224 BMC	Command/$\overline{\text{Data}}$ signal. This signal shall cause the FSA to enter a receive command mode when high and to interpret the serial data line as data when low. Any previously active command will be immediately terminated by C/$\overline{D}$.
CLK	18	I	Clock	Same TTL-level clock used to generate internal timing as used for 7220-1.
$\overline{CS}$	1	I	External	An active low signal used for multiplexing of FSAs. The FSA is disabled whenever $\overline{CS}$ is high (i.e., it presents a high impedance to the bus and ignores all bus activity).
$\overline{DATA.OUT.A}$, $\overline{DATA.OUT.B}$	11, 12	O	7234 CPG	Output data from the FIFO to the MBM generate circuitry. Used to write data into the bubble device (active low).
DET.A+, DET.A−, DET.B+, DET.B−	6, 7, 8, 9	I	7114 MBM	Differential signal lines from the MBM detector.
DIO	17	I/O	7224 BMC	The Serial Bus data line (a bidirectional active high signal).
$\overline{ENABLE.A}$, $\overline{ENABLE.B}$	13, 14	O	7234 CPG/7250	TTL-level outputs utilized as chip selects for other interface circuits. They shall be set and reset by the Command Decoder under instruction of the Controller (active low).
$\overline{ERR.FLG}$	4	O	7224 BMC	An error flag used to interrupt the Controller to indicate that an error condition exists. It shall be an open drain, active low signal.
$\overline{RESET}$	16	I	Power Fail Circuit	An active low signal that shall reset all flags and pointers in the FSA as well as disabling the chip as the CS signal does. The $\overline{RESET}$ pulse width must be 5 clock periods to assure the FSA is properly reset.
$\overline{SELECT.IN}$	19	I	7224 BMC	An input utilized for time-division multiplexing. An active low signal whose presence indicates that the FSA is to send or receive data from the Serial Bus during the next two clock periods.
$\overline{SELECT.OUT}$	2	O	7244 FSA	The $\overline{SELECT.IN}$ pulse delayed by two clocks. It shall be connected to the $\overline{SELECT.IN}$ pin of the next FSA. It is delayed by two clocks because the FSA is a dual-channel device. Channel A shall internally pass $\overline{SELECT.IN}$ to Channel B (delayed by one clock).
$\overline{SHIFT.CLK}$	15	I	7224 BMC	A Controller-generated clock signal that shall be used to clock data out of the bubble I/O Output Latch to the bubble module during a write operation and to cause bubble signals to be converted by the Sense Amp and clocked into the Bubble I/O Input Latch on a read.

7250

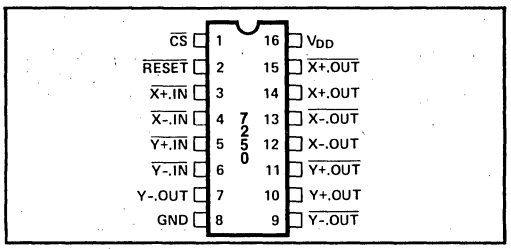


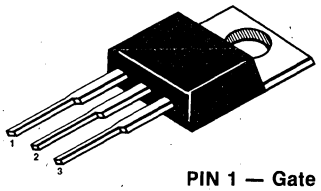
Figure 8. 7250 Pin Configuration

Table 4. 7250 Pin Description

Symbol	Pin No.	I/O	Source/Destination	Description
$\overline{CS}$	1	I	7244 FSA	Chip select. It is active low. When high chip is deselected and I_{DD} is significantly reduced.
$\overline{RESET}$	2	I	Power Fail Circuit	Active low input from RESET.OUT of MD7220-5 Controller forces 7250 outputs inactive so that bubble memory is protected in the event of power supply failure.
$\overline{X+IN}$, $\overline{X-.IN}$	3, 4	I	7224 BMC	Active low inputs from Controller which turn on the high-current X outputs.
X−.OUT $\overline{X-.OUT}$ $\overline{X+.OUT}$ X+.OUT	12, 13, 14, 15	O	7264	High-current outputs and their complements for driving the gates of the 7264 transistors which in turn drive the X coils of the bubble memory.
$\overline{Y+.IN}$, $\overline{Y-.IN}$	5, 6	I	7224 BMC	Active low inputs from Controller which turn on the high-current Y outputs.
Y−OUT $\overline{Y-.OUT}$ $\overline{Y+.OUT}$ Y+.OUT	7,9,10,11	O	7264	High-current outputs and their complements for driving the gates of the 7264 quad transistors which in turn drive the Y coils of the bubble memory.

7264

Four matched pair of N- and P-channel transistors. In industry standard TO-220 Discrete packaging.



PIN 1 — Gate
PIN 2 & TAB — Drain
PIN 3 — Source

Symbol	Pin No.	I/O	Source/Destination	Description
N-Channel				
G	1	I	7250	Gate Drive Signal
D	2	O	7114	Coil Drive Current
S	3	I	Ground	—
P-Channel				
G	1	I	7250	Gate Drive Signal
D	2	O	7114	Coil Drive Current
S	3	I	Ground	—

ABSOLUTE MAXIMUM RATINGS*

7114

Operating Temperature ..... 10°C to 55°C Case
Relative Humidity 95%
Shelf Storage Temperature (Data Integrity Not Guaranteed) ..... −55°C to +125°C
Voltage Applied to DET.SUPPLY ..... 14 Volts
Voltage Applied to PULSE. COM ..... 14 Volts
Continuous Current between DET.COM and Detector Outputs ..... 20 mA
Coil Current ..... 0.5A D.C.
External Magnetic Field for Non-Volatile Storage ..... 20 Oersteds
Non-Operating Handling Shock (without socket) 200G
Operating Vibration (2 Hz to 2 kHz with socket) 20G

COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

SUPPORT I.C.'S

	7234	7244	7250	7264
Temperature Under Bias	−40 to 100°C	−10 to +85°C	−40 to 100°C	−40 to 100°C
Storage Temperature	−65 to +150°C	−65 to +150°C	−65 to +150°C	−55 to +150°C
Voltage Input	−0.5 to +7V		−0.5 to V_{DD} +0.5	
V_{CC}	−0.5 to +7V	−0.5 to +7V		
V_{DD}	−0.5 to +12.6V	−0.5 to +14V		
Gate Voltage				15V
Output Current			250mA	
Power Dissipation 80°C	1W			1.05W
Power Dissipation 25°C		1W		2W
Continuous Drain Current				2A
Peak Drain Current				3A

D.C. CHARACTERISTICS

The BPK 5V74 is designed as a true subsystem. All D.C. characteristics that describes the interfacing to the subsystem is included in this section.

7234 (T_A 0°C to +70°C; V_{CC} = 5.0V ±5%, ±5% V_{DD} = 12V ±5%; unless otherwise specified.)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
I_{IL}	Input Low Current			−0.4	mA	V_{IL} = 0.4V, V_{CC} = 5.25V
I_{IH}	Input High Current			20	μA	V_{IH} = V_{CC} = 5.25V
V_{IL}	Input Low Voltage			0.8	V	
V_{IH}	Input High Voltage	2.0			V	
V_C	Input Clamp Voltage			−1.5	V	I = −18 mA, V_{CC} = 4.75V
I_{CEX1}	Output Leakage Current (All Outputs except PWR.FAIL)			1.0	mA	V_{CC} = 5.25V, V_{DD} = 12.6V
I_{CEX2}	$\overline{\text{PWR.FAIL}}$ Output Leakage Current			40	μA	V_{OH} = V_{CC} = 5.25V
V_{OL}	$\overline{\text{PWR.FAIL}}$ Output Low Voltage			0.4	V	I_{OL} = 4 mA, V_{CC} = 4.75V
I_{CC1}	Current from V_{CC}—Selected		30	45	mA	CS = V_{IL}, V_{CC} = 5.25V
I_{DD1}	Current from V_{DD}—Selected		20	35	mA	CS = V_{IL}, V_{CC} = 5.25V
I_{DD2}	Current from V_{DD}—Power Down		12	19	mA	CS = V_{IH}, V_{DD} = 12.6V

7244 (T_A = 0°C to 70°C; V_{CC} = 5.0V +5%, −10%: V_{DD} = 12V ±5%)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
V_{IL}	Input Low Voltage	−0.5		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC} +0.5	V	
V_{OL}	Output Low Voltage (All Outputs Except $\overline{\text{SELECT.OUT}}$)			0.45	V	I_{OL} = 3.2mA
V_{OLSO}	Output Low Voltage ($\overline{\text{SELECT.OUT}}$)			0.45	V	I_{OL} = 1.6mA
V_{OH}	Output High Voltage (All Outputs Except $\overline{\text{SELECT.OUT}}$)	2.4			V	I_{OH} = 400 μA
V_{OHSO}	Output High Voltage ($\overline{\text{SELECT.OUT}}$)	2.4			V	I_{OH} = 200 μA
V_{THR}	Detector Threshold		6.8		mV	V_{DD} = 12.0V
$\lvert I_{IN} \rvert$	Input Leakage Current			10	μA	$0 \leq V_{IN} \leq V_{CC}$
$\lvert I_{OFL} \rvert$	Output Float Leakage			10	μA	$0.45 \leq V_{OUT} \leq V_{CC}$
I_{CC}	Power Supply Current from V_{CC}			120	mA	
I_{DD}	Power Supply Current from V_{DD}			30	mA	

7250 (T_A = 0° to 70°C; V_{DD} = 12V +5%, −10%; unless otherwise specified)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
$\|I_{IN}\|$	Input Current			5	μA	V_I = 0.8V
V_{IL}	Low-Level Input Voltage			0.8	V	
V_{IH}	High-Level Input Voltage	2.2			V	
I_{DD0}	Supply Current			4.5	mA	Chip Deselected: $\overline{CS}$ = V_{IH}, V_{DD} = 12.6V
I_{DD1}	Supply Current			75	mA	f = 100 kHz, V_{DD} = 12.6V, Outputs Unloaded

ADVANCE INFORMATION

BPK 5V75
4MBIT BUBBLE MEMORY PROTOTYPE KIT

BPK 5V75-4	10°C To 55°C

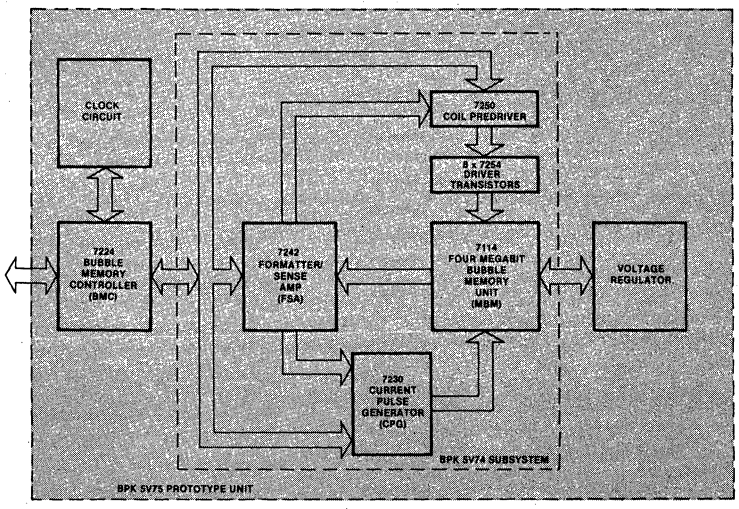
- **Assembled and Tested 4MBit Bubble Memory Prototype Kit on 6" x 4" PC Board**
- **Complete with Powerfail Data Protection and Clock Circuitry**
- **Built-in Error Detection/Correction**
- **Interfaces with Intel 8080/85/86/88 186/286 and Other Standard Microprocessors**
- **Software Driver for Bubble Memory Kit on Diskette for Intel Microcomputer Development System**
- **4Mbit, Non-Volatile, Read-Write, Solid-State Memory in Leaded Dense Package**
- **Average Random Access Time of 88ms**
- **Maximum Data Rate of 200K bit/sec**
- **Operates From +5V Only**
- **Complete Documentation and Interfacing Information Included**

The BPK 5V75 prototype kit is a completely assembled and tested 4Mbit bubble memory evaluation tool. It is ideal for the design engineer that wants the opportunity to quickly evaluate how a bubble memory solution improves and adds value to an end-product by providing a compact solid-state memory that also reliably keeps the data at any power down.

Application information on microprocessor interfacing is included in the kit. A Bubble Memory Kit software driver is also included on a diskette for the Intel Microcomputer Development System.

The bubble memory (7114) and the support circuits (7234, 7244, 7250, 7264) in the kit are described in more detail on the BPK 5V74 Bubble Memory Subsystem data sheet. The Bubble Memory Controller (7224) in the kit is described in more detail on the 7224 data sheet.

For production purposes, the bubble memory and the support circuit components can be ordered as kits in the BPK 5V74 Subsystem. The 7224, controlling up to eight BPK 5V74's, is ordered separately.



OCTOBER 1983
ORDER NUMBER: 230868-001

COMPONENT TEMPERATURE SPECIFICATIONS*

Part Number	7110A Magnetic Bubble Memory Temperature		Support Circuits Min. Operating Temperature	Description
	Operating	Non-Volatile Storage		
BPK 5V75-4	10° to 55°C Case	−20° to 75°C	10° to 55°C Ambient	4 Mbit Bubble Memory Prototype Kit

* The bubble memory prototype kit is assembled and functionally tested to facilitate the prototyping process. The board is tested at 10°C and 40°C ambient temperature.

BPK 5V75 BUBBLE MEMORY PROTOTYPE BOARD

Item	Description	Part Number
4 Mbit Bubble Memory	20-pin leaded package which provides 4 megabit of non-volatile storage.	7114-4
Bubble Memory Controller	User interface, performs serial-to-parallel and parallel-to-serial data conversions. Generates timing signals.	7224
Current Pulse Generator	Converts digital timing signals to analog current pulses suited to the drive requirements of the 7114 MBM. The CPG provides the replicate, swap, generate, boot replicate, and bootswap pulses required by the MBM.	7234
Dual Formatter/Sense Amp	Provides direct interface to the 7114 Bubble Memory. The FSA contains on-chip sense amplifiers, a full FIFO data block buffer, burst error detection and correction circuits, and circuitry for handling of the bubble memory redundant loops.	7244
Coil Predriver	Provides the high voltage, high current outputs to drive the 7264 transistors.	7250
8 Coil Drive Transistors	Switches the required current to drive the X and Y coils of the 7114 Bubble Memory.	7264
Prefabricated Printed Circuit Board		IMB 75
Clock Circuit		
Voltage Regulator		

ADDITIONAL ITEMS

Diskette	Software driver for bubble memory kit, configured for Intel Microcomputer Development System.	
BPK 75 Bubble Memory Prototype Kit User's Manual	Literature	

SPECIFICATIONS

Capacity

512K Byte per BPK 5V75

Performance

Avg. Access Time . . . . . 88 msec
Maximum Data Transfer Rate . . . . 200 Kbits/sec
Average Data Transfer Rate . . . . . 136 Kbits/sec

Data Organization

64 bytes per page
8192 pages per BPK 5V75

Addressing Scheme

Logical page number

Environmental

Temperature: Temperature specifications.
Operating Humidity: 0—95% Non-Condensing

BPK 5V75 POWER SUPPLY REQUIREMENTS

Voltage	Margin	Power Off/Power Fail Decay Rate
+5 Volt	±5%	less than 0.45 volts/msec

- Power on voltage rate of rise — no restrictions

BPK 5V75 POWER CONSUMPTION

Board including bubble memory, support ICs, controller and voltage regulator.

Total Standby
- Typical 1.4 W
- Maximum 3.2 W

Total Active
- Typical 5.5 W
- Maximum 9.6 W

PRELIMINARY

BPK 70
1MBIT BUBBLE MEMORY SUBSYSTEM

BPK 70-1	0°C to 75°C
BPK 70-4	10°C to 55°C
BPK 70-5	– 20°C to 85°C

- **1 Mbit (128K Bytes) Non-volatile, Solid-state, Read/Write Bubble Memory Subsystem**
- **Interfaces to Host Microprocessor Via 7220 Bubble Memory Controller**
- **Maximum Data Rate of 100K bit/sec with One Subsystem**
- **Maximum Data Rate of 800K bit/sec with Eight Subsystems in Parallel and Time Multiplexed**
- **Contains Bubble Memory and all ICs for Production with 1Mbit Bubble Memory**
- **Modularity Provides Expansion Up to Eight Subsystems per Controller**
- **Average Random Access Time of 48 ms**
- **Bubble Memory in 20-pin Leadless Package Using Socket**

The BPK 70 Bubble Memory Subsystem is a modular building block used to design bubble memory systems. In a complete bubble memory system, a 7220 (7220-4 or 7220-5) Bubble Memory Controller (BMC) interfaces the BPK 70 subsystem to the host processor.

The modular Intel subsystem provides a path for density expansion. One 7220/7220-4 BMC can interface up to eight 1 Mbit BPK 70-1/BPK 70-4 subsystems. Thus, a 1 MBit (128 KByte) system can be expanded up to a 8 MBit system by adding subsystems. A 7220-5 can interface up to four BPK 70-5 subsystems. BMC's can be combined in parallel to further expand the system memory capacity.

Together, the BPK 70 Bubble Storage Subsystem and a 7220 controller provide a reliable mass storage system for any application. This bubble memory system can be customized to the particular layout and form factor of many different systems.

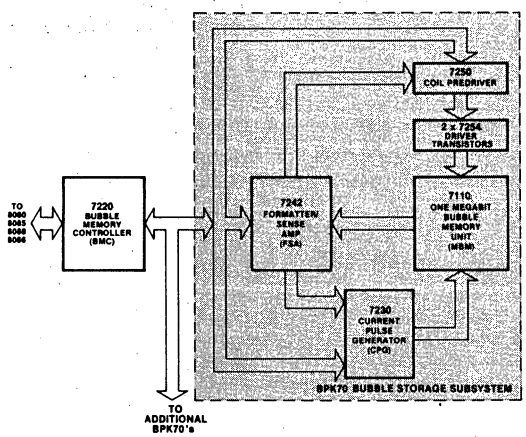


Figure 1. Block Diagram of Single Bubble Memory System—128K Bytes

OCTOBER 1983
ORDER NO. 230870-001

FUNCTIONAL DESCRIPTION

A BPK 70 subsystem and a controller comprise a complete bubble memory system. The 1 MBit BMC, the 7220, provides the interface between the host microprocessor and the bubble memory subsystem and provides all the timing and control signals to the subsystem. The user interface of the BMC is compatible with microprocessor bus systems for 8080, 8085, 8086, 8088, 80186, 80286 and other standard microprocessors. The BMC is a software driven device utilizing 16 convenient commands. The design engineer's primary responsibility is interfacing to the BMC. This is comparable to interfacing a disk drive controller.

The BPK 70 consists of one 1 MBit Magnetic Bubble Memory (MBM) and five support IC's (see Figure 1). These are the basic components to build a non-volatile, solid-state, read/write military memory system utilizing 1 MBit bubble memory. The bubble memory is in a leadless package and can directly be soldered to tbe board. The complete family of LSI support circuits has been designed to handle the complex analog interface associated with bubble devices. The immediate support circuitry for the MBM consists of five integrated circuit components — an 7250 Coil Predriver (CPD), two 7254 Quad VMOS Drive Transistor Packs, an 7230 Current Pulse Generator (CPG), and an 7242 Formatter/Sense Amplifier (FSA).

Data integrity is insured by the automatic error correction designed into the BPK 70-5.

The average random access time of a 1 MBit subsystem is 48 ms with a 100Kbit/sec maximum data transfer rate. Operating several subsystems in parallel, the BMC uses time division multiplexing. Therefore, the maximum data rate increases correspondingly for the whole system.

Operating subsystems serially, one MBM being accessed at a time, the maximum data transfer rate is still 100Kbit/sec. If low power consumption is a critical design goal, the bubble memory subsystem can be powered down when it is not being accessed, thus reducing the average power consumption.

The data in the 1 Mbit subsystem is organized in 2048 pages, each with 64 bytes. Conceptually, the data organization with pages is analogous to a disk system's sectors. In systems with multiple bubble memories, the page size can vary from 64 bytes to 512 bytes (256 for BPK 70-5) depending on the number of subsystems and if the subsystems are operating in parallel or serially, being accessed one at a time. In the 10° to 55°C and 0 to 75°C temperature range, respectively, the components in the subsystem are fully interchangeable.

Each of the components in the subsystem is described in more detail in the rest of this data sheet. For additional information, consult the Intel **BPK-72 Bubble Memory Prototype Kit User's Manual,** Order Number: 121685.

BPK 70 FUNCTIONAL DESCRIPTION

Item	Description	Part Number
1 MBit Bubble Memory	20-pin leadless package which provides 1 megabit of non-volatile storage.	7110
Socket	Provides reliable mounting of 1Mbit bubble memory.	7905
Current Pulse Generator	Converts digital timing signals to analog current pulses suited to the drive requirements of the MBM. The CPG provides the replicate, swap, generate, boot replicate, and bootswap pulses required by the MBM. 22 Pin DIP Package.	7230
Dual Formatter/Sense Amp	Provides direct interface to the Bubble Memory. The FSA contains on-chip sense amplifiers, a full FIFO data block buffer, burst error detection and correction circuits, and circuitry for handling of the bubble memory redundant loops. 20 Pin DIP package.	7242
Coil Predriver	Provides the high voltage, high current outputs to drive the Quad VMOS transistors. 16 Pin DIP package.	7250
Quad VMOS Coil Drive Transistors	Switches the required current to drive the X and Y coils of the Bubble Memory. 14 Pin DIP Package.	7254

For additional packaging information, see the last section of Packaging Information in the Memory Components Handbook.

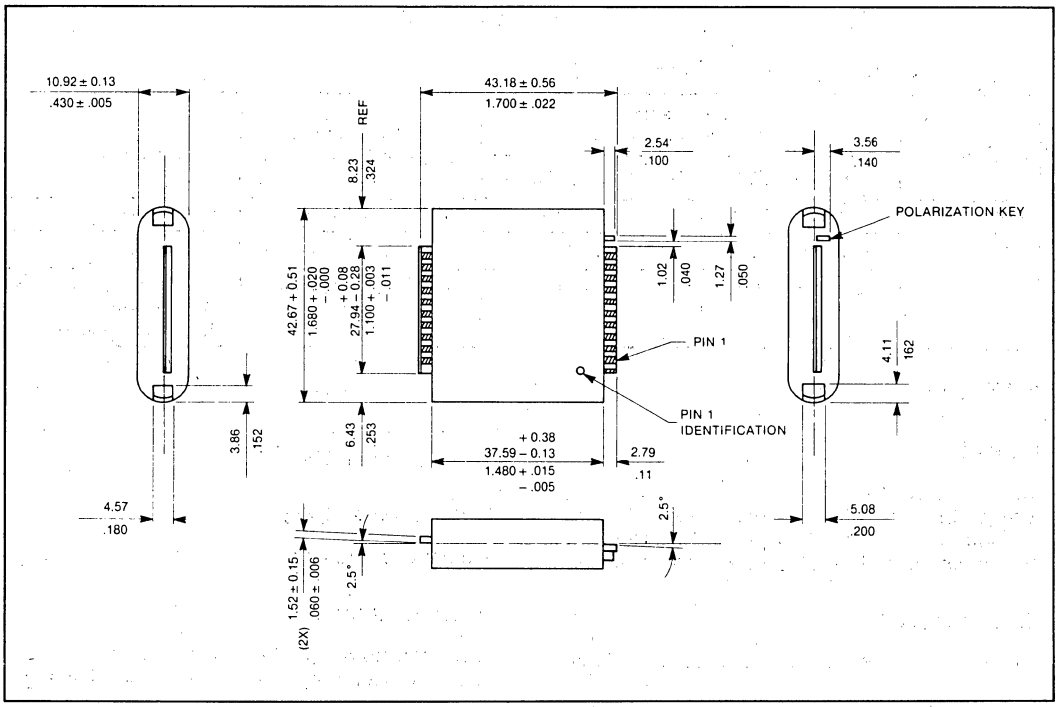


Figure 2. Bubble Leadless Package

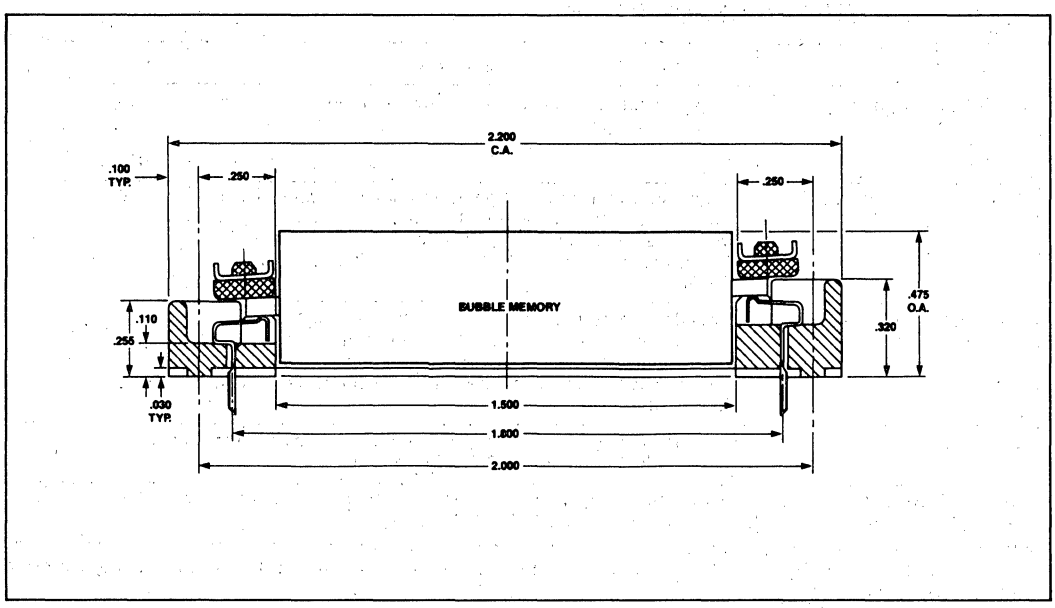


Figure 3. Socket 7905

BPK 70 TEMPERATURE RANGE

Part Number	Bubble Memory Temperature Ranges		Support Circuits Operating Temperature (T_A)	Description
	Operating (T_C)	Non-Volatile Storage		
BPK 70-1	0 to 75°C Case	−40 to 90°C	0 to 70°C Ambient	See Below
BPK 70-4	10 to 55°C Case	−20 to 75°C	10 to 55°C Ambient	See Below
BPK 70-5	−20 to 85°C Case	−55 to +100°C	−20 to 85°C Ambient	1 Mbit Bubble Storage Sub-system, Leadless Package

SPECIFICATIONS

Capacity

128K Byte per BPK 70
Maximum of 8 BPK 70-1 per 7220 Controller
Maximum of 8 BPK 70-4 per 7220-4 Controller
Maximum of 4 BPK 70-5 per 7220-5 Controller

Performance

Avg. Access Time..... 48 msec

Data Organization

64 bytes per page
2048 pages per BPK 70

Addressing Scheme

Logical page number

Environmental

Temperature: See Ordering Information
Operating Humidity: 0—95% Non-Condensing

BPK 70 POWER CONSUMPTION

BPK 70 Components	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
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	0	1.300	1.300	0.550	0	0
7220	1.050	0	1.050	0.500	1.050	0.500
System	1.92	4.80	6.72	3.90	3.03	1.55

For larger systems in which a 7220 is controlling several BPK 70 subsystems, the power consumption can be estimated by using the above data. By using power switching techniques the bubble memory system's standby power can be reduced to almost zero. For details, see AP-164, **Using CMOS to Minimize Bubble Memory Power Consumption.**

BPK 70 POWER SUPPLY REQUIREMENTS

Voltage	Margin	Power Off/Power Fail Decay Rate
+12 Volt	±5%	less than 1.10 volts/msec
+5 Volt	±5%	less than 0.45 volts/msec

- Voltage sequencing — no restrictions
- Power on voltage rate of rise — no restrictions
- The power supply requirements shown are based on the recommended power fail circuitry as shown in Figure 4.

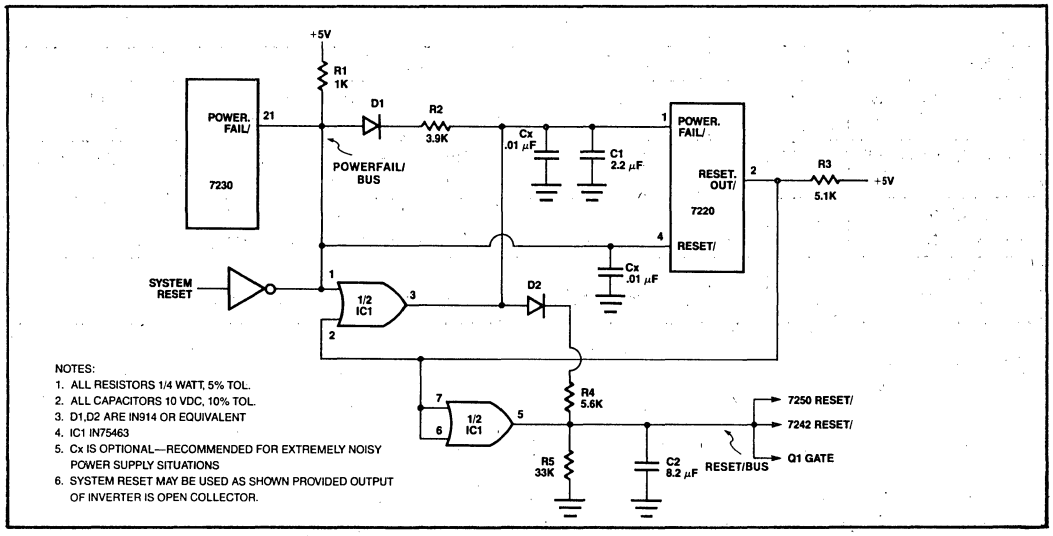


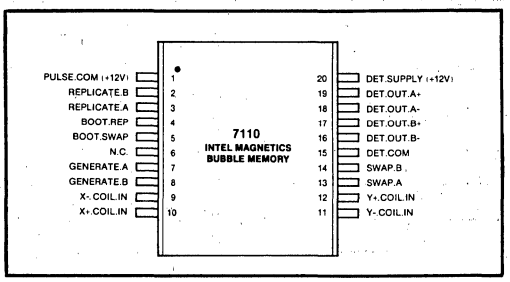
Figure 4. Recommended Power Fail Circuitry

Below the 7110 MBM is described in more detail. For other support ICs are described in the BPK70A data sheet.

Consult the **BPK72 User's Manual** for Additional Information.

PIN DESCRIPTION

7110



NOTE THAT PINS 13 AND 14 SHOULD BE EXTERNALLY CONNECTED.

Figure 5. 7110 Pin Configuration

Table 1. 7110 Pin Description

Symbol	Pin No.	I/O	Source/Destination	Description
BOOT.REP	4	I	7230 CPG	Two-level current pulse input for reading the boot loop.
BOOT.SWAP	5	I	7230 CPG	Single-level current pulse for writing data into the boot loop. This pin is normally used only in the manufacture of the MBM.
DET.COM	15	I		Ground return for the detector bridge.
DET.OUT	16 — 19	O	7242 FSA	Differential pair (A+, A− and B+, B−) outputs which have signals of several millivolts peak amplitude.
DET.SUPPLY	20	I		+ 12 volt supply pin.
GEN.A and GEN.B	7, 8	I	7230 CPG	Two-level current pulses for writing data onto the input track.
PULSE.COM	1	I		+ 12 volt supply pin.
REP.A and REP.B	3, 2	I	7230 CPG	Two-level current pulses for replicating data from storage loops to output track.
SWAP.A and SWAP.B	13, 14	I	7230 CPG	Single-level current pulse for swapping data from input track to storage loops.
X− .COIL.IN. X+ COIL.IN.	9, 10	I	7254/VMOS	Terminals for the X or inner coil.
Y− .COIL.IN. Y+ .COIL.IN.	11, 12	I	7254/VMOS	Terminals for the Y or outer coil.

ABSOLUTE MAXIMUM RATINGS*

Operating Temperature ..... −20°C to +85°C Case
Relative Humidity 95%
Shelf Storage Temperature (Data Integrity Not Guaranteed) ..... −65°C to +150°C
Voltage Applied to DET.SUPPLY ..... 14 Volts
Voltage Applied to PULSE.COM ..... 12.6 Volts
Continuous Current between DET.COM and Detector Outputs ..... 10 mA
Coil Current ..... 0.5A D.C.
External Magnetic Field for Non-Volatile Storage ..... 20 Oersteds
Non-Operating Handling Shock (without socket) 200G
Operating Vibration (2 Hz to 2 kHz with socket) 20G

COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS (T_C = Specified in earlier temperature range table, V_{DD} = 12V ± 5%)

7110

Parameter	7110-1, -4 Limits			7110-5 Limits		Unit
	Min.	Nom.[1]	Max.	Min.	Max.	
RESISTANCE: PULSE.COM to GEN.A or GEN.B	9	30	59	8	61.5	ohms
RESISTANCE: PULSE.COM to REP.A or REP.B	9	20	26	8	27	ohms
RESISTANCE: PULSE.COM to SWAP.A or SWAP.B	44	100	149	40	155.5	ohms
RESISTANCE: PULSE.COM to BOOT.REP	3.5	8	24	3	25	ohms
RESISTANCE: PULSE.COM to BOOT.SWAP	5	15	36	4.5	37.5	ohms
RESISTANCE: DET.OUT A+ to DET.OUT.A−	670	1030	1903	620	1984	ohms
RESISTANCE: DET.OUT B+ to DET.OUT B−	670	1030	1903	620	1984	ohms
RESISTANCE: DET.COM to DET.SUPPLY	355	600	1050	338	1095	ohms
X.COIL RESISTANCE		4.6				ohms
Y.COIL RESISTANCE		2.0				ohms
X.COIL INDUCTANCE		97				μH
Y.COIL INDUCTANCE		80				μH
OPERATING POWER		1.20	1.75			watts
STANDBY POWER		0.25	.45			watts

DRIVE REQUIREMENTS CHARACTERISTICS[2] (T_C = Specified in temperature range table)

7110

Symbol	Parameter	Min.	Nom.[1]	Max.	Units
f_R	Field Rotation Frequency	49.95	50.000	50.05	kHz
I_{px}	X.Coil Peak Current		600		ma
I_{py}	Y.Coil Peak Current		750		ma
Θ_{1x}	X.Coil Positive Turn-On Phase	268	270	272	degrees
Θ_{2x}	X.Coil Positive Turn-Off Phase	16	18	20	degrees
Θ_{3x}	X.Coil Negative Turn-On Phase	88	90	92	degrees
Θ_{4x}	X.Coil Negative Turn-Off Phase	196	198	200	
Θ_{1y}	Y.Coil Positive Turn-On Phase	0	0	0	degrees
Θ_{2y}	Y.Coil Positive Turn-Off Phase	106	108	110	degrees
Θ_{3y}	Y.Coil Negative Turn-On Phase	178	180	182	degrees
Θ_{4y}	Y.Coil Negative Turn-Off Phase	286	288	290	degrees

CONTROL PULSE REQUIREMENTS (T_C = Specified in temperature range table)[2]

7110

Pulse	Amplitude			Pulse of Leading Edge (Degrees)			Width (Degrees)		
	Min.	Nom.[1]	Max.	Min.	Nom.[1]	Max.	Min.	Nom.[1]	Max.
GEN.A, GEN.B CUT	62	75	81	266 86	270 (Odd) 90 (Even)	274 94	3	6.75	8
GEN.A, GEN.B TRANSFER	34	40	49	266 86	270 (Odd) 90 (Even)	274 94	86	90	94
REP.A, REP.B CUT	170	200	240	268	270	277	3	6.75	8
REP.A, REP.B TRANSFER	126	145	160	268	270	277	86	90	94
SWAP	111	125	134	176	180	184	513	517	521
BOOT.REP CUT	85	100	110	268	270	277	3	6.75	8
BOOT.REP TRANSFER	63	75	80	268	270	277	86	90	94
BOOT.SWAP	63	75	80	176	180	184		360	

NOTES:

1. Nominal values are measured at T_C = 25°C.
2. 7110-5 is sold only as a matched part with the 7230-5. Matched parts are tested over temperature range for V_{DD} = 12V ±5%.

PRELIMINARY

BPK 70A
1MBIT BUBBLE MEMORY SUBSYSTEM

BPK 70A-1	0°C to 75°C
BPK 70A-4	10°C to 55°C
BPK 70A-5	−20°C to 85°C

- **1 Mbit (128K Bytes) Non-volatile, Solid-state, Read/Write Bubble Memory Subsystem**
- **Interfaces to Host Microprocessor Via 7220 Bubble Memory Controller**
- **Maximum Data Rate of 100K bit/sec with One Subsystem**
- **Maximum Data Rate of 800K bit/sec with Eight Subsystems in Parallel and Time Multiplexed**
- **Contains Bubble Memory and all ICs for Production with 1Mbit Bubble Memory**
- **Modularity Provides Expansion Up to Eight Subsystems per Controller**
- **Average Random Access Time of 48 ms**
- **Bubble Memory in 20-pin Leaded Package**

The BPK 70A Bubble Memory Subsystem is a modular building block used to design bubble memory systems. In a complete bubble memory system, a 7220 (7220-4 or 7220-5) Bubble Memory Controller (BMC) interfaces the BPK 70A subsystem to the host processor.

The modular Intel subsystem provides a path for density expansion. One 7220/7220-4 BMC can interface up to eight 1 Mbit BPK 70A-1/BPK 70A-4 subsystems. Thus, a 1 MBit (128 KByte) system can be expanded up to a 8 MBit system by adding subsystems. A 7220-5 can interface up to four BPK 70A-5 subsystems. BMC's can be combined in parallel to further expand the system memory capacity.

Together, the BPK 70A Bubble Storage Subsystem and a 7220 controller provide a reliable mass storage system for any application. This bubble memory system can be customized to the particular layout and form factor of many different systems.

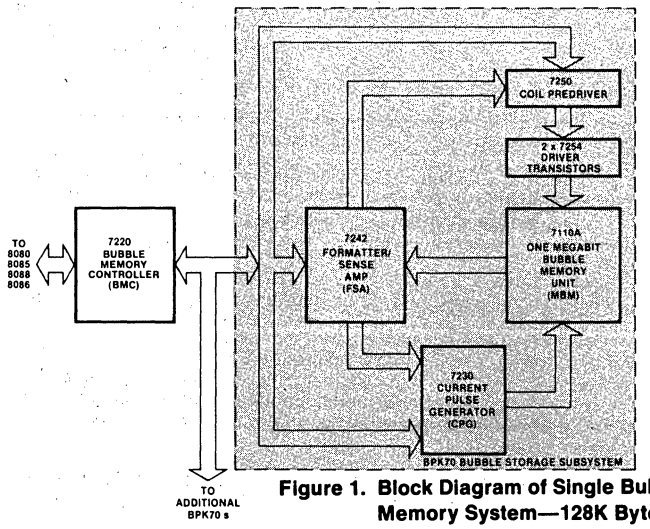


Figure 1. Block Diagram of Single Bubble Memory System—128K Bytes

OCTOBER 1983
ORDER NO. 210804-002

FUNCTIONAL DESCRIPTION

An BPK 70A-5 subsystem and a controller comprise a complete bubble memory system. The 1 MBit BMC, the 7220, provides the interface between the host microprocessor and the bubble memory subsystem and provides all the timing and control signals to the subsystem. The user interface of the BMC is compatible with microprocessor bus systems for 8080, 8085, 8086, 8088, 80186, 80286 and other standard microprocessors. The BMC is a software driven device utilizing 16 convenient commands. The design engineer's primary responsibility is interfacing to the BMC. This is comparable to interfacing a disk drive controller.

The BPK 70A consists of one 1 MBit Magnetic Bubble Memory (MBM) and five support IC's (see Figure 1). These are the basic components to build a non-volatile, solid-state, read/write military memory system utilizing 1 MBit bubble memory. The bubble memory is in a leaded package. The complete family of LSI support circuits has been designed to handle the complex analog interface associated with bubble devices. The immediate support circuitry for the MBM consists of five integrated circuit components — an 7250 Coil Predriver (CPD), two 7254 Quad VMOS Drive Transistor Packs, an 7230 Current Pulse Generator (CPG), and an 7242 Formatter/Sense Amplifier (FSA).

Data integrity is insured by the automatic error correction designed into the BPK 70A.

The average random access time of a 1 MBit subsystem is 48 ms with a 100Kbit/sec maximum data transfer rate. Operating several subsystems in parallel, the BMC uses time division multiplexing. Therefore, the maximum data rate increases correspondingly for the whole system.

Operating subsystems serially, one MBM being accessed at a time, the maximum data transfer rate is still 100Kbit/sec. If low power consumption is a critical design goal, the bubble memory subsystem can be powered down when it is not being accessed, thus reducing the average power consumption.

The data in the 1 Mbit subsystem is organized in 2048 pages, each with 64 bytes. Conceptually, the data organization with pages is analogous to a disk system's sectors. In systems with multiple bubble memories, the page size can vary from 64 bytes to 512 bytes (256 for BPK 70-5) depending on the number of subsystems and if the subsystems are operating in parallel or serially, being accessed one at a time. In the 10° to 55°C and 0 to 75°C temperature range, respectively, the components in the subsystem are fully interchangeable.

Each of the components in the subsystem is described in more detail in the rest of this data sheet. For additional information, consult the Intel **BPK-72 Bubble Memory Prototype Kit User's Manual,** Order Number: 121685.

BPK 70A FUNCTIONAL DESCRIPTION

Item	Description	Part Number
1 MBit Bubble Memory	20-pin leaded package which provides 1 megabit of non-volatile storage.	7110A
Current Pulse Generator	Converts digital timing signals to analog current pulses suited to the drive requirements of the MBM. The CPG provides the replicate, swap, generate, boot replicate, and bootswap pulses required by the MBM. 22 Pin DIP Package.	7230
Dual Formatter/Sense Amp	Provides direct interface to the Bubble Memory. The FSA contains on-chip sense amplifiers, a full FIFO data block buffer, burst error detection and correction circuits, and circuitry for handling of the bubble memory redundant loops. 20 Pin DIP package.	7242
Coil Predriver	Provides the high voltage, high current outputs to drive the Quad VMOS transistors. 16 Pin DIP package.	7250
Quad VMOS Coil Drive Transistors	Switches the required current to drive the X and Y coils of the Bubble Memory. 14 Pin DIP package.	7254

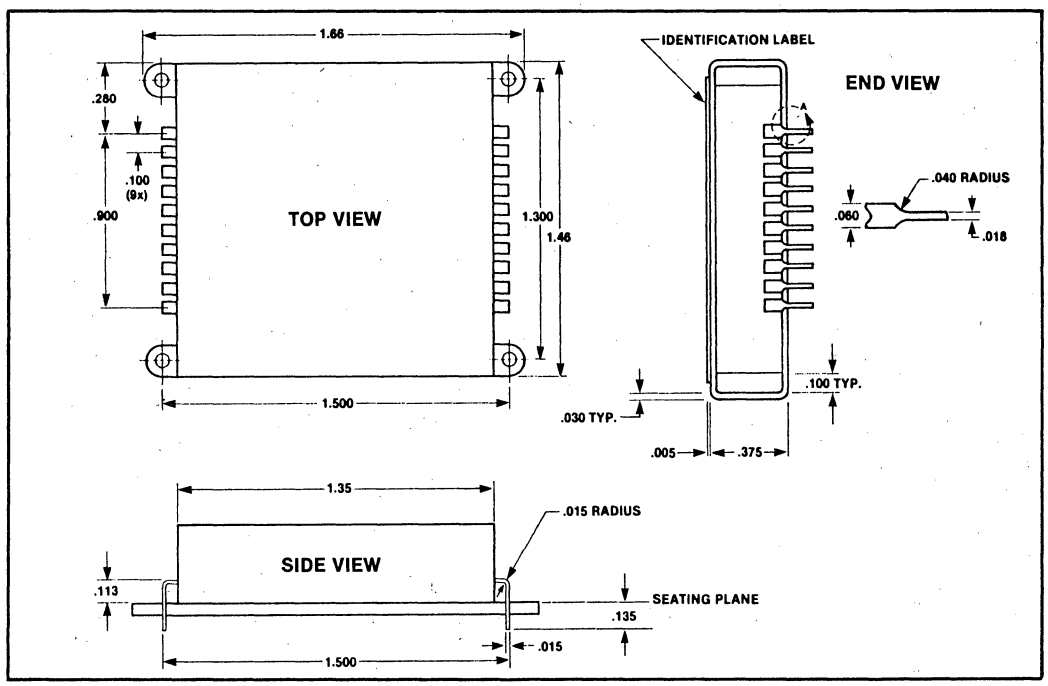


Figure 2. 1MBit Leaded Bubble Memory Package

BPK 70A TEMPERATURE RANGE

Part Number	Bubble Memory Temperature Ranges		Support Circuits Operating Temperature (T_A)	Description
	Operating (T_C)	Non-Volatile Storage		
BPK 70A-1	0 to 75°C Case	−40 to 90°C	0 to 70°C Ambient	See Below
BPK 70A-4	10 to 55°C Case	−20 to 75°C	10 to 55°C Ambient	See Below
BPK 70A-5	−20 to 85°C Case	−55 to +100°C	−20 to 85°C Ambient	1 Mbit Bubble Storage Sub-system, Leaded Package

SPECIFICATIONS

Capacity

128K Byte per BPK 70A
Maximum of 8 BPK 70A-1 per 7220 Controller
Maximum of 8 BPK 70A-4 per 7220-4 Controller
Maximum of 4 BPK 70A-5 per 7220-5 Controller

Performance

Avg. Access Time..... 48 msec

Data Organization

64 bytes per page
2048 pages per BPK 70A

Addressing Scheme

Logical page number

Environmental

Temperature: See Ordering Information
Operating Humidity: 0—95% Non-Condensing

BPK 70A POWER CONSUMPTION

BPK70A Components	Power (Watts)					
	+5V (Maximum)	+12V (Maximum)	Total Active (Maximum)	Total Active (Typical)	Total Standby (Maximum)	Total Standby (Typical)
7110A	0	1.740	1.740	1.480	0.440	0.290
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	0	1.300	1.300	0.550	0	0
7220-5	1.050	0	1.050	0.500	1.050	0.500
System	1.92	4.80	6.72	3.90	3.03	1.55

For larger systems in which a 7220 is controlling several BPK 70A subsystems, the power consumption can be estimated by using the above data. By using power switching techniques the bubble memory system's standby power can be reduced to almost zero. For details, see AP-164, **Using CMOS to Minimize Bubble Memory Power Consumption.**

BPK 70A POWER SUPPLY REQUIREMENTS

Voltage	Margin	Power Off/Power Fail Decay Rate
+12 Volt	±5%	less than 1.10 volts/msec
+5 Volt	±5%	less than 0.45 volts/msec

- Voltage sequencing — no restrictions
- Power on voltage rate of rise — no restrictions
- The power supply requirements shown are based on the recommended power fail circuitry as shown in Figure 3.

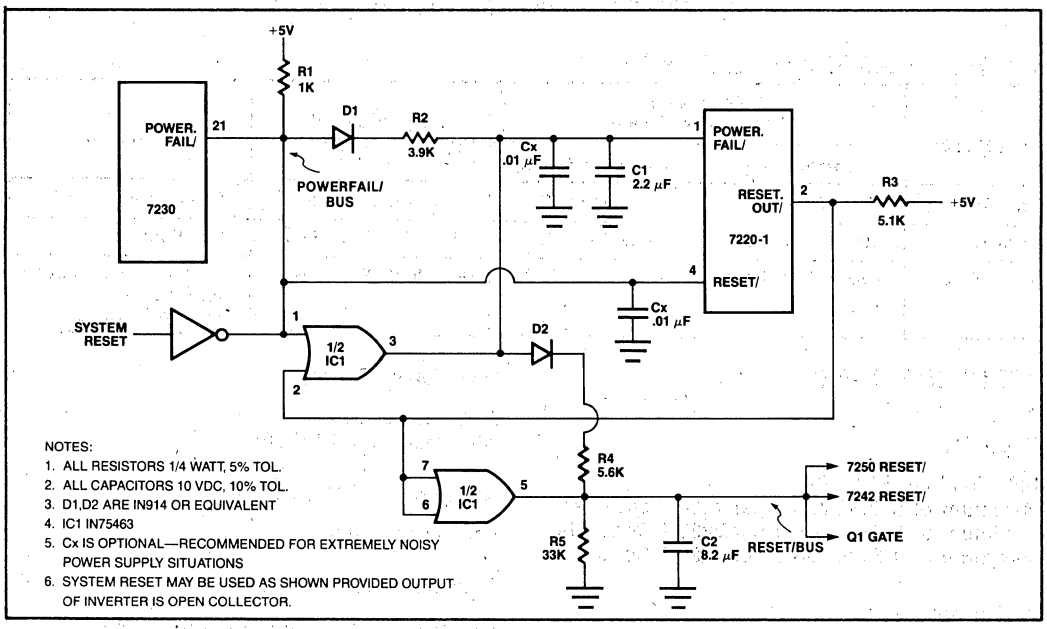


Figure 3. External Powerfail Circuit Solution

PIN DESCRIPTIONS

7110A

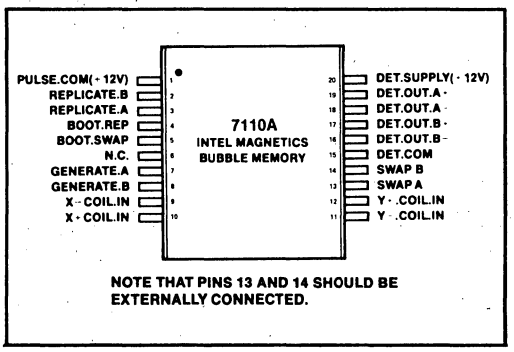


Figure 4. 7110A Pin Configuration

Table 1. 7110A Pin Description

Symbol	Pin No.	I/O	Source/Destination	Description
BOOT.REP	4	I	7230 CPG	Two-level current pulse input for reading the boot loop.
BOOT.SWAP	5	I	7230 CPG	Single-level current pulse for writing data into the boot loop. This pin is normally used only in the manufacture of the MBM.
DET.COM	15	I		Ground return for the detector bridge.
DET.OUT	16 — 19	O	7242 FSA	Differential pair (A+, A− and B+, B−) outputs which have signals of several millivolts peak amplitude.
DET.SUPPLY	20	I		+12 volt supply pin.
GEN.A and GEN.B	7, 8	I	7230 CPG	Two-level current pulses for writing data onto the input track.
PULSE.COM	1	I		+12 volt supply pin.
REP.A and REP.B	3, 2	I	7230 CPG	Two-level current pulses for replicating data from storage loops to output track.
SWAP.A and SWAP.B	13, 14	I	7230 CPG	Single-level current pulse for swapping data from input track to storage loops.
X−.COIL.IN. X+ COIL.IN.	9, 10	I	7254/VMOS	Terminals for the X or inner coil.
Y−.COIL.IN. Y+.COIL.IN.	11, 12	I	7254/VMOS	Terminals for the Y or outer coil.

7230

EXTERNAL RESISTOR REQUIREMENTS

Connect a 1% 3.48K ohm resistor based between pin 20 and ground or rreferenced current switch as outlined in BPK72 User's Manual.

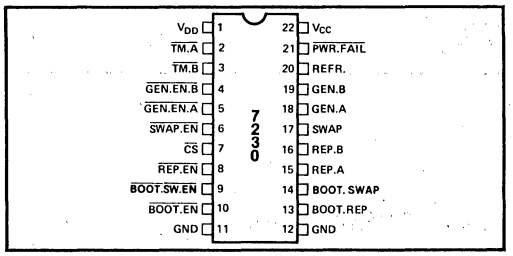


Figure 5. 7230 Pin Configuration

Table 2. 7230 Pin Description

Symbol	Pin No.	I/O	Source/Destination	Description
$\overline{\text{BOOT.EN}}$	10	I	7220 BMC	An active low input enabling the BOOT.REP output current pulse.
BOOT.REP	13	O	7110A MBM	An output providing the current pulse for bootstrap loop replication in the bubble memory.
BOOT.SWAP	14	O	7110A MBM	An output providing a current pulse which may be used for writing data into the bootstrap loop.
$\overline{\text{BOOT.SW.EN}}$	9	I	7220 BMC	An active low input enabling the BOOT.SWAP output current pulse.
CS	7	I	7242 FSA	An active low input for selecting the chip. The chip powers down during deselect.
GEN.A	18	O	7110A MBM	An output providing the current pulse for writing data into the "A" quads of the bubble memory.
GEN.B	19	O	7110A MBM	An output providing the current pulse for writing data into the "B" quads of the bubble memory.
$\overline{\text{GEN.EN.A}}$	5	I	7242 FSA	An active low input enabling the GEN.A output current pulse.
$\overline{\text{GEN.EN.B}}$	4	I	7242 FSA	An active low input enabling the GEN.B output current pulse.
$\overline{\text{PWR.FAIL}}$	21	O	7220 BMC	An active low, open collector output indicating that either V_{CC} or V_{DD} is below its threshold value.
REFR.	20	I	External Resistor	The pin for the reference current generator to which an external resistance must be connected.
REP.A	15	O	7110A MBM	An output providing the current pulse for replication of data in the "A" quads of the bubble memory.
REP.B	16	O	7110A MBM	An output providing the current pulse for replication of data in the "B" quads of the bubble memory.
$\overline{\text{REP.EN}}$	8	I	7220 BMC	An active low input enabling the REP.A and REP.B outputs.
SWAP	17	O	7110A MBM	An output providing the current pulse for exchanging the data between the input track and the storage loops in the bubble memory.
$\overline{\text{SWAP.EN}}$	6	I	7220 BMC	An active low input enabling the SWAP output.
$\overline{\text{TM.A}}$	2	I	7220 BMC	An active low timing signal determining the cut pulse widths of the BOOT.REP, GEN.A, GEN.B, REP.A and REP.B outputs.
$\overline{\text{TM.B}}$	3	I	7220 BMC	An active low timing signal determining the transfer pulse widths of the BOOT.REP, GEN.A, GEN.B, REP.A and REP.B outputs.

7242

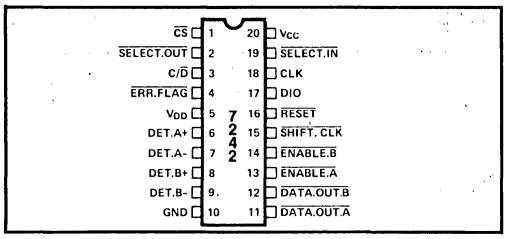


Figure 6. 7242 Pin Configuration

Table 3. 7242 Pin Description

Symbol	Pin No.	I/O	Source/Destination	Description
C/$\overline{D}$	3	I	7220 BMC	Command/$\overline{\text{Data}}$ signal. This signal shall cause the FSA to enter a receive command mode when high and to interpret the serial data line as data when low. Any previously active command will be immediately terminated by C/$\overline{D}$.
CLK	18	I	Clock	Same TTL-level clock used to generate internal timing as used for 7220.
$\overline{CS}$	1	I	External	An active low signal used for multiplexing of FSAs. The FSA is disabled whenever $\overline{CS}$ is high (i.e., it presents a high impedance to the bus and ignores all bus activity).
$\overline{DATA.OUT.A}$, $\overline{DATA.OUT.B}$	11, 12	O	7230 CPG	Output data from the FIFO to the MBM generate circuitry. Used to write data into the bubble device (active low).
DET.A+, DET.A−, DET.B+, DET.B−	6, 7, 8, 9	I	7110A MBM	Differential signal lines from the MBM detector.
DIO	17	I/O	7220 BMC	The Serial Bus data line (a bidirectional active high signal).
$\overline{ENABLE.A}$, $\overline{ENABLE.B}$	13, 14	O	7230 CPG/7250	TTL-level outputs utilized as chip selects for other interface circuits. They shall be set and reset by the Command Decoder under instruction of the Controller (active low).
$\overline{ERR.FLG}$	4	O	7220 BMC	An error flag used to interrupt the Controller to indicate that an error condition exists. It shall be an open drain, active low signal.
$\overline{RESET}$	16	I	Power Fail Circuit	An active low signal that shall reset all flags and pointers in the FSA as well as disabling the chip as the CS signal does. The $\overline{RESET}$ pulse width must be 5 clock periods to assure the FSA is properly reset.
$\overline{SELECT.IN}$	19	I	7220 BMC	An input utilized for time-division multiplexing. An active low signal whose presence indicates that the FSA is to send or receive data from the Serial Bus during the next two clock periods.
$\overline{SELECT.OUT}$	2	O	7242 FSA	The $\overline{SELECT.IN}$ pulse delayed by two clocks. It shall be connected to the $\overline{SELECT.IN}$ pin of the next FSA. It is delayed by two clocks because the FSA is a dual-channel device. Channel A shall internally pass $\overline{SELECT.IN}$ to Channel B (delayed by one clock).
$\overline{SHIFT.CLK}$	15	I	7220 BMC	A Controller-generated clock signal that shall be used to clock data out of the bubble I/O Output Latch to the bubble module during a write operation and to cause bubble signals to be converted by the Sense Amp and clocked into the Bubble I/O Input Latch on a read.

7250

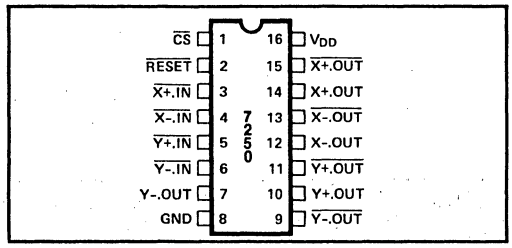


Figure 7. 7250 Pin Configuration

Table 4. 7250 Pin Description

Symbol	Pin No.	I/O	Source/Destination	Description
$\overline{CS}$	1	I	7242 FSA	Chip select. It is active low. When high chip is deselected and I_{DD} is significantly reduced.
$\overline{RESET}$	2	I	Power Fail Circuit	Active low input from RESET.OUT of 7220 Controller forces 7250 outputs inactive so that bubble memory is protected in the event of power supply failure.
$\overline{X+IN}$, $\overline{X-.IN}$	3, 4	I	7220 BMC	Active low inputs from Controller which turn on the high-current X outputs.
X−.OUT $\overline{X-.OUT}$ $\overline{X+.OUT}$ X+.OUT	12, 13, 14, 15	O	7254/VMOS	High-current outputs and their complements for driving the gates of the 7254 VMOS quad transistors which in turn drive the X coils of the bubble memory.
$\overline{Y+.IN}$, $\overline{Y-.IN}$	5, 6	I	7220 BMC	Active low inputs from Controller which turn on the high-current Y outputs.
Y−OUT $\overline{Y-.OUT}$ $\overline{Y+.OUT}$ Y+.OUT	7,9,10,11	O	7254/VMOS	High-current outputs and their complements for driving the gates of the 7254 VMOS quad transistors which in turn drive the Y coils of the bubble memory.

7254

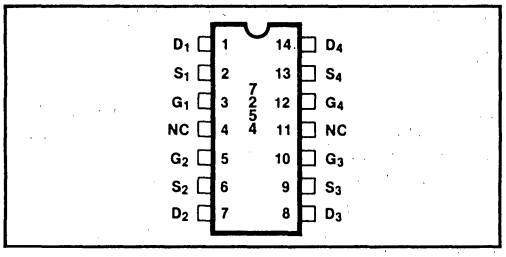


Figure 8. 7254 Pin Configuration

Table 5. 7254 Pin Configuration

Symbol	Pin No.	I/O	Source/Destination	Description
D_1	1	O	7110A MBM	Coil Drive Current
S_1	2	I	Ground	
G_1	3	I	7250 CPD	Gate Drive Signal
G_2	5	I	7250 CPD	Gate Drive Signal
S_2	6	I	+ 12V	
D_2	7	O	Coil	Coil Drive Current
D_3	8	O	Coil	Coil Drive Current
S_3	9	I	Ground	
G_3	10	I	7250 CPD	Gate Drive Signal
G_4	12	I	7250 CPD	Gate Drive Signal
S_4	13	I	+ 12V	
D_4	14	O	Coil	Coil Drive Current

ABSOLUTE MAXIMUM RATINGS*

7110A

Operating Temperature ..... −20°C to +85°C Case
Relative Humidity 95%
Shelf Storage Temperature (Data Integrity Not Guaranteed) ..... −65°C to +150°C
Voltage Applied to DET.SUPPLY ..... 14 Volts
Voltage Applied to PULSE.COM ..... 12.6 Volts
Continuous Current between DET.COM and Detector Outputs ..... 10 mA
Coil Current ..... 0.5A D.C.
External Magnetic Field for Non-Volatile Storage ..... 20 Oersteds
Non-Operating Handling Shock 200G
Operating Vibration (2 Hz to 2 kHz) 20G

**COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

SUPPORT I.C.'S*

	7230	7242	7250	7254
Temperature Under Bias	−40 to 100°C	−40 to 100°C	−40 to 100°C	−40 to 100°C
Storage Temperature	−65 to +150°C	−65 to +150°C	−65 to +150°C	−65 to +150°C
Voltage Input	−0.5 to +7V		−0.5 to V_{DD} +0.5	
V_{CC}	−0.5 to +7V	−0.5 to +7V		
V_{DD}	−0.5 to +12.6V	−0.5 to +14V		
Gate Voltage				15V
Output Current			250mA	
Power Dissipation 80°C	1W			1.05W
Power Dissipation 25°C				1.75W
Continuous Drain Current				2A
Peak Drain Current				3A

D.C. CHARACTERISTICS

7110A (T_C = Range Specified in temperature range table, V_{DD} = 12V ± 5%)

Parameter	7110A-1, 7110A-4 Limits			7110A-5 Limits[5]		Unit
	Min.	Nom.[1]	Max.	Min.	Max.	
RESISTANCE: PULSE.COM to GEN.A or GEN.B	7.5	30	59	8	61.5	ohms
RESISTANCE: PULSE.COM to REP.A or REP.B	7.5	20	26	8	27	ohms
RESISTANCE: PULSE.COM to SWAP.A or SWAP.B	42	100	149	40	155.5	ohms
RESISTANCE: PULSE.COM to BOOT.REP	2	8	24	3	25	ohms
RESISTANCE: PULSE.COM to BOOT.SWAP	3.5	15	36	4.5	37.5	ohms
RESISTANCE: DET.OUT A+ to DET.OUT.A−	570	1030	1903	530	1984	ohms
RESISTANCE: DET.OUT B+ to DET.OUT B−	570	1030	1903	530	1984	ohms
RESISTANCE: DET.COM to DET.SUPPLY	320	600	1050	290	1095	ohms
X.COIL RESISTANCE		4.2		329		ohms
Y.COIL RESISTANCE		2.7				ohms
X.COIL INDUCTANCE		135				μH
Y.COIL INDUCTANCE		93				μH
OPERATING POWER		1.20	1.75			watts
STANDBY POWER		0.25	.45			watts

7230 (T_A = Specified in temperature range table; V_{CC} = 5.0V ±5%, ±5% V_{DD} = 12V ±5%; unless otherwise specified)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
I_{IL}	Input Low Current			−0.4	mA	V_{IL} = 0.4V, V_{CC} = 5.25V
I_{IH}	Input High Current			20	μA	V_{IH} = V_{CC} = 5.25V
V_{IL}	Input Low Voltage			0.8	V	
V_{IH}	Input High Voltage	2.0			V	
V_C	Input Clamp Voltage			−1.5	V	I = −18 mA, V_{CC} = 4.75V
I_{CEX1}	Output Leakage Current (All Outputs except PWR.FAIL)			1.0	mA	V_{CC} = 5.25V, V_{DD} = 12.6V
I_{CEX2}	$\overline{\text{PWR.FAIL}}$ Output Leakage Current			40	μA	V_{OH} = V_{CC} = 5.25V
V_{OL}	$\overline{\text{PWR.FAIL}}$ Output Low Voltage			0.4	V	I_{OL} = 4 mA, V_{CC} = 4.75V
I_{CC1}	Current from V_{CC}—Selected		30	45	mA	CS = V_{IL}, V_{CC} = 5.25V
I_{DD1}	Current from V_{DD}—Selected		20	35	mA	CS = V_{IL}, V_{CC} = 5.25V
I_{DD2}	Current from V_{DD}—Power Down		12	19	mA	CS = V_{IH}, V_{DD} = 12.6V

7242 (T_A = Specified in temperature range table; V_{CC} = 5.0V +5%, −10% V_{DD} = 12V ±5%)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
V_{IL}	Input Low Voltage	−0.5		0.8	V	
V_{IH}	Input High Voltage	2.2		V_{CC} +0.5	V	
V_{OL}	Output Low Voltage (All Outputs Except $\overline{\text{SELECT.OUT}}$)		.2	0.45	V	I_{OL} = 3.2mA
V_{OLSO}	Output Low Voltage ($\overline{\text{SELECT.OUT}}$)		.2	0.45	V	I_{OL} = 1.6mA
V_{OH}	Output High Voltage (All Outputs Except $\overline{\text{SELECT.OUT}}$)	2.4	3.0		V	I_{OH} = 400 μA
V_{OHSO}	Output High Voltage ($\overline{\text{SELECT.OUT}}$)	2.4			V	I_{OH} = 200 μA
V_{THR}	Detector Threshold	2.3	2.5	2.7	mV	V_{DD} = 12.0V
$\lvert I_{IL} \rvert$	Input Leakage Current		0	5	μA	$0 \leq V_{IN} \leq V_{CC}$
$\lvert I_{OFL} \rvert$	Output Float Leakage		0	10	μA	$0.45 \leq V_{OUT} \leq V_{CC}$
I_{CC}	Power Supply Current from V_{CC}		35	120	mA	
I_{DD}	Power Supply Current from V_{DD}		5	30	mA	

*Minimum V_{IH} is 2.2V for the 7242-5 device.

7250 (T_A = Specified in temperature range table; V_{DD} = 12V ±5%; −10%; unless otherwise specified)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
$\lvert I_{IN} \rvert$	Input Current			5	μA	V_I = 0.8V
V_{IL}	Low-Level Input Voltage			0.8	V	
V_{IH}	High-Level Input Voltage	2.2			V	
V_{OL1}	Output Low Voltage			2.0	V	I_{OL} = 100 mA
V_{OL2}	Output Low Voltage			0.2	V	I_{OL} = 10 mA
V_{OH1}	Output High Voltage	V_{DD}−2			V	I_{OH} = −100 mA
V_{OH2}	Output High Voltage	V_{DD}−0.2			V	I_{OH} = −10 mA
I_{OL}	Output Sink Current	100			mA	V_{OL} = 2.0V
$\lvert I_{OH} \rvert$	Output Source Current	100			mA	V_{OH} = V_{DD}−2.0V
I_{DD0}	Supply Current			4.5	mA	Chip Deselected: $\overline{CS}$ = V_{IH}, V_{DD} = 12.6V
I_{DD1}	Supply Current			75	mA	f = 100 kHz, V_{DD} = 12.6V, Outputs Unloaded

7254 All limits apply for N- and P-Channel transistors, T_A = specified in temperature range table; unless otherwise specified.

Symbol	Parameter	Limits				Test Conditions
		Min.	Typ.	Max.	Unit	
BV_{DSS}	Drain-Source Breakdown Voltage	20			V	V_{GS} = 0, I_D = 10 μA
$V_{GS}(th)$	Gate-Source Threshold Voltage	0.8			V	V_{GS} = V_{DS}, I_D = 1 mA, T_A = 25°C
		0.65			V	V_{GS} = V_{DS}, I_D = 1 mA, T_A = 85°C
I_{GSS}	Gate Leakage Current			100	nA	V_{GS} = 12V, V_{DS} = 0, T_A = 25°C
I_{DSS}	Drain Leakage Current			500	nA	V_{GS} = 0, V_{DS} = 20V, T_A = 25°C
R_{DS}	On-Resistance for sum of Q1+Q2, Q3+Q4 (Note 1)	2.0	2.5	3.0	Ω	V_{GS} = 11.4V, 1_D = 1A, T_A = 25°C
V_{F1}	Parasitic Diode Forward Voltage (Note 1)			.75	V	V_{GS} = 0V, I_D = 50 mA, T_A = 25°C
V_{F2}	Parasitic Diode Forward Voltage (Note 1)			1.20	V	V_{GS} = 0V, I_D = 1000 mA, T_A = 25°C

NOTE:
1. Pulse test—80 μs pulse, 1% duty cycle, r_{DS} increase 0.8%/°C.

DRIVE REQUIREMENTS CHARACTERISTICS[2] (T_C = Specified in temperature range table)

7110A

Symbol	Parameter	Min.	Nom.[1]	Max.	Units
f_R	Field Rotation Frequency	49.95	50.000	50.05	kHz
I_{px}	X.Coil Peak Current		600		ma
I_{py}	Y.Coil Peak Current		750		ma
Θ_{1x}	X.Coil Positive Turn-On Phase	268	270	272	degrees
Θ_{2x}	X.Coil Positive Turn-Off Phase	16	18	20	degrees
Θ_{3x}	X.Coil Negative Turn-On Phase	88	90	92	degrees
Θ_{4x}	X.Coil Negative Turn-Off Phase	196	198	200	
Θ_{1y}	Y.Coil Positive Turn-On Phase	0	0	0	degrees
Θ_{2y}	Y.Coil Positive Turn-Off Phase	106	108	110	degrees
Θ_{3y}	Y.Coil Negative Turn-On Phase	178	180	182	degrees
Θ_{4y}	Y.Coil Negative Turn-Off Phase	286	288	290	degrees

CONTROL PULSE REQUIREMENTS (T_C = Specified in temperature range table)[2]

7110A

Pulse	Amplitude			Pulse of Leading Edge (Degrees)			Width (Degrees)		
	Min.	Nom[1]	Max.	Min.	Nom.[1]	Max.	Min.	Nom.[1]	Max.
GEN.A, GEN.B CUT	62	70	81	266 86	270 (Odd) 90 (Even)	274 94	3	6.75	8
GEN.A, GEN.B TRANSFER	34	40	49	266 86	270 (Odd) 90 (Even)	274 94	86	90	94
REP.A, REP.B CUT	170	190	240	268	270	277	3	6.75	8
REP.A, REP.B TRANSFER	126	140	160	268	270	277	86	90	94
SWAP	111	120	134	176	180	184	513	517	521
BOOT.REP CUT	85	95	110	268	270	277	3	6.75	8
BOOT.REP TRANSFER	63	70	80	268	270	277	86	90	94
BOOT.SWAP [2]	63	70	80	176	180	184		360	

NOTES:

1. Nominal values are measured at T_C = 25°C.
2. Boot.Swap is not normally accessed during operating. It is utilized at the factory to write the index address and redundant loop information onto the bootstrap loops before shipment.

A.C. CHARACTERISTICS*

7230 V_{CC} = 5V ±5%, V_{DD} = 12V ±5%

Symbol	Parameter	Min.	Max.	Unit
t_{ENON}	Delay On		260	ns
t_{DISOFF}	Delay Off		70	ns
t_{CSON}	CS Enable		500	ns
t_{CSOFF}	CS Disable		70	ns

*These parameters are sample tested, not 100% tested.

POWER FAIL CHARACTERISTICS**

7230 T_A = See temperature range table.

	Min.	Typ.	Max.
V_{CCTH}	4.43V	4.60V	4.70V
V_{DDTH}	10.75V	11.10V	11.28V

**Power fail characteristics apply to 7110A Bubble Memory Data Integrity only and not to full memory operation.

A.C. CHARACTERISTICS (T_A = Specified in earlier temperature range table; V_{CC} = 5.0V +5%, −10%; V_{DD} = 12V ±5%; C_L = 120pF; unless otherwise noted)

7242

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_p	Clock Period	240	500	ns	
t_f	Clock Phase Width	.45 t_p	.55 t_p		
t_n t_f	Clock Rise and Fail Time		30	ns	
t_{SIC}	$\overline{\text{SELECT.IN}}$ Setup Time to CLK	50		ns	
t_{CDC}	C/$\overline{\text{D}}$ Setup Time to CLK	50		ns	
t_{CYC}	$\overline{\text{SELECT.IN}}$ or $\overline{\text{SHIFT.CLK}}$ Cycle Time	20 t_p			
t_{DC}	DIO Setup Time to Clock (Read Mode)	50		ns	
t_{CSC}	$\overline{\text{CS}}$ Setup Time to CLK	100		ns	
t_{RIC}	$\overline{\text{RESET.IN}}$ Setup Time to CLK	100		ns	
t_{IH}	Control Input Hold Time for C/D, $\overline{\text{SELECT.IN}}$ and DIO	10		ns	
t_{CSOL}	CLK to $\overline{\text{SELECT.OUT}}$ Leading Edge Delay		100	ns	C_L = 50 pF
t_{CSOT}	CLK to $\overline{\text{SELECT.OUT}}$ Trailing Edge Delay		80	ns	C_L = 50 pF
t_{CDV}	CLK to DIO Valid Delay*		100	ns	
t_{CDH}	CLK to DIO Hold Time*	0		ns	
t_{CDE}	CLK to DIO Enabled from Float*		100	ns	
t_{SIDE}	$\overline{\text{SELECT.IN}}$ Trailing Edge to DIO Enabled from Float*		70	ns	
t_{CDF}	CLK to DIO Entering Float*		100	ns	
t_{SCDO}	$\overline{\text{SHIFT.CLK}}$ to DATAOUT Delay*		200	ns	
t_{SCWR}	$\overline{\text{SHIFT.CLK}}$ Width (Read)	4 t_p	t_{CYC} − 11 t_p		
t_{SCWW}	$\overline{\text{SHIFT.CLK}}$ Width (Write)	t_p	t_{CYC} − 2 t_p		

7250 (T_A = Specified in earlier temperature range table; V_{DD} = 12V ±5%, unless otherwise specified)

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
t_{p1}	Propagation Delay from $\overline{X+.IN}$, $\overline{X-.IN}$, $\overline{Y+.IN}$, $\overline{Y-.IN}$			100	ns	500 pF Load
t_{p2}	Propagation Delay from $\overline{CS}$ or $\overline{RESET}$			150	ns	500 pF Load
t_r	Rise Time (10% to 90%)			45	ns	500 pF Load
t_F	Fall Time (90% to 10%)			45	ns	500 pF Load
t_s	Skew Between an Output and Its Complements			20	ns	

7254 (T_A = 25°C)

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
$T_{ON}(N)$	N-Channel Turn-On Time			20	ns	
$t_{ON}(P)$	P-Channel Turn-On Time			30	ns	
$t_{OFF}(N)$	N-Channel Turn-Off Time			20	ns	
$t_{OFF}(P)$	P-Channel Turn-Off Time			30	ns	

CAPACITANCE* (T_A = 25°C)

7230

Symbol	Parameter	Typ.	Max.	Unit	Test Conditions*
C_{IN}	Input Capacitance		10	pF	

*This parameter is periodically sampled and not 100% tested. Condition of measurement is f = 1 MHz.

7242 (T_A = 25°C, V_{CC} = 0V, f = 1 MHz)*

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
C_{IN}	Input Capacitance		10	pF	
C_{OUT}	Output Capacitance		10	pF	
C_{DIO}	DIO Capacitance		10	pF	

*DIO Write Mode.

7250 (T_A = 25°C, V_{DD} = 0V, V_{BIAS} = 2V, f = 1 MHz)*

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
C_{IN}	Input Capacitance			10	pF	

*This parameter is periodically sampled and is not 100% tested.

7254 T_A = 25°C

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
$C_{iss}(N)$	N-Channel Input Capacitance			175	pF	V_{GS} = 0, V_{DS} = 12V, f = 1 MHz
$C_{iss}(P)$	P-Channel Input Capacitance			190	pF	V_{GS} = 0, V_{DS} = 12V, f = 1 MHz

OUTPUT CURRENTS

7230 (T_A = Specified in earlier temperature range table; V_{CC} = 5.0V ±5%, V_{DD} = 12V ±5%)

Parameter	Current (mA)			Test Conditions Voltage Out	
	Min.	Nom.	Max.	Min.	Max.
GEN.A, GEN.B CUT	62	75	81	5.5	11.6
GEN.A, GEN.B TRANSFER	34	40	49	5.5	12.2
REP.A, REP.B CUT	170	200	240	3.4	9.3
REP.A, REP.B TRANSFER	126	145	160	3.4	11.4
SWAP	111	125	134	2.7	9.9
BOOT.REP CUT	85	100	110	7.7	12.1
BOOT.REP TRANSFER	63	75	80	7.7	12.4
BOOT.SWAP	63	75	80	9.0	12.3

PRELIMINARY

BPK 72A
1MBIT BUBBLE MEMORY PROTOTYPE KIT

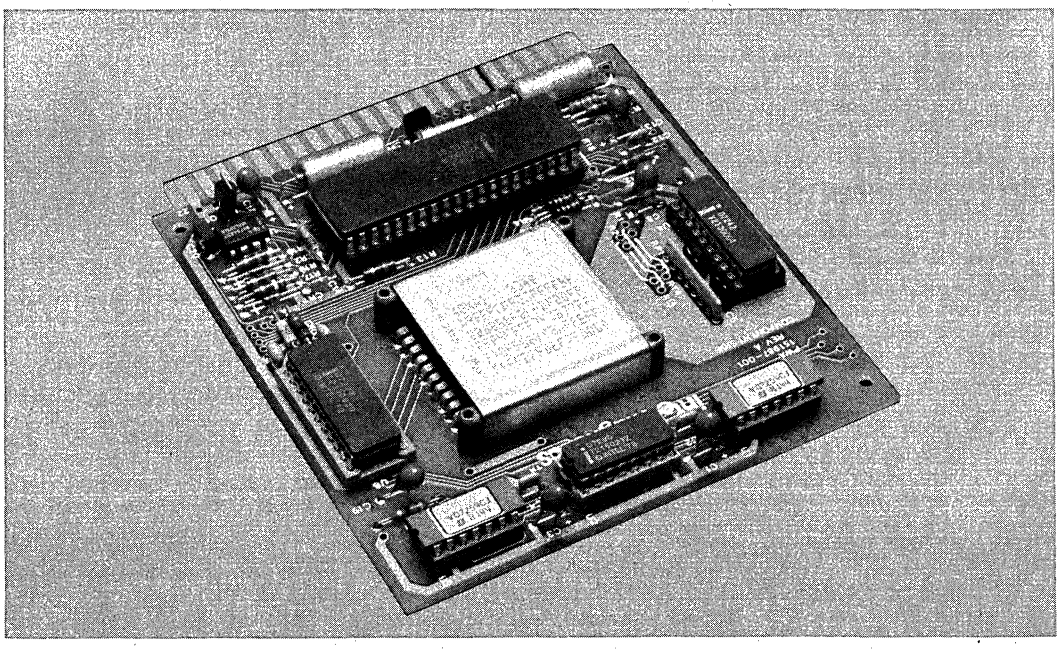
BPK 72A-1	0 To 75°C
BPK 72A-4	10°C To 55°C
BPK 72A-5*	– 20°C To 85°C

- **Assembled and Tested 1MBit Bubble Memory Prototype Kit on 4" x 4" PC Board**
- **Complete with Powerfail Data Protection and Clock Circuitry**
- **Built-in Error Detection/Correction**
- **Interfaces with Intel 8080/85/86/88 186/286 and Other Microprocessors**
- **Software Driver for Bubble Memory Kit Diskette for Intel MDS* System**
- **1Mbit, Non-Volatile, Read-Write, Solid-State Memory in Leaded Dense Package**
- **Average Random Access Time of 48ms**
- **Maximum Data Rate of 100K bit/sec**
- **Operates from + 5V and + 12V Power Supplies**
- **Complete Documentation and Interfacing Information Included**

The BPK 72A prototype kit is a completely assembled and tested 1Mbit bubble memory evaluation tool. It is ideal for the design engineer that wants the opportunity to fast evaluate how a bubble memory solution improves and adds value to an end-product by providing a compact solid-state memory that also reliably keeps the data at any powerdown.

Application information on microprocessor interfacing is included in the kit. A Bubble Memory Kit software driver is also included on a diskette for the Intel MDS* System.

*MDS is a registered trademark of Mohawk Data Sciences Corp.



OCTOBER 1983
ORDER NO. 230871-001

COMPONENT TEMPERATURE SPECIFICATIONS

Part Number	7110A Magnetic Bubble Memory Temperature		Support Circuits Min. Operating Temperature	Description
	Operating	Non-Volatile Storage		
BPK 72A-1*	0° to 75°C Case	−40° to 90°C	0° to 70°C Ambient	1 Mbit Bubble Memory Prototype Kit Assembled
BPK 72A-4*	10° to 55°C Case	−20° to 75°C	10° to 55°C Ambient	1 Mbit Bubble Memory Prototype Kit Assembled
BPK 72A-5	−20° to 85°C Case	−40° to 100°C	−20° to 85°C Ambient	1 Mbit Bubble Memory Prototype Kit

(* The bubble memory prototype kit is assembled and functionally tested to facilitate the prototyping process. The board is tested at the following ambient temperatures:

BPK 72A-1 0°C and 55°C BPK 72A-4 10°C and 40°C)

BPK 72A BUBBLE MEMORY PROTOTYPE KIT

The BPK 72A has a compact leaded bubble memory package and does not require a socket.

The bubble memory (7110A) and the support circuits (7230, 7242, 7250, 7254) in the BPK 72A kit are described in more detail on the BPK 70A Bubble Memory Subsystem data sheet. The Bubble Memory Controller (7220) in the BPK 72A kit is described in more detail on the 7220 data sheet.

For production purposes, the bubble memory and the support circuit components can be ordered as the BPK 70A Subsystem. The 7220, controlling up to eight BPK 70A's, is ordered separately.

Item	Description	Part Number
1 Mbit Bubble Memory	20-pin leaded package which provides 1 megabit of non-volatile storage.	7110A-1/7110A-4/ 7110A-5
Bubble Memory Controller	User interface, performs serial-to-parallel and parallel-to-serial data conversions. Generates timing signals.	7220-1/7220-4/7220-5
Current Pulse Generator	Converts digital timing signals to analog current pulses suited to the drive requirements of the 7110 MBM. The CPG provides the replicate, swap, generate, boot replicate, and bootswap pulses required by the MBM.	7230/7230-4/7230-5
Dual Formatter/Sense Amp	Provides direct interface to the 7110 Bubble Memory. The FSA contains on-chip sense amplifiers, a full FIFO data block buffer, burst error detection and correction circuits, and circuitry for handling of the bubble memory redundant loops.	7242/7242-5
Coil Predriver	Provides the high voltage, high current outputs to drive the 7254 Quad VMOS transistors.	7250/7250-5
2 Quad VMOS Coil Drive Transistors	Switches the required current to drive the X and Y coils of the 7110 Bubble Memory.	7254
Prefabricated Printed Circuit Board		IMB 72
Additional Items		
BPK 72 Bubble Memory Prototype Kit User's Manual	Literature, User's Manual, Software Guide	121685-002
Diskette	Software driver for bubble memory kit, configured for Intel MDS.	
Memory Components Handbook	Application Notes and Data Sheets.	210830
Seed Module	Recreates a lost seed bubble.	7901

SPECIFICATIONS

Capacity

128K Byte per BPK 72A

Performance

Avg. Access Time . . . . . 48 msec
Maximum Data Transfer Rate . . . . 100 Kbits/sec
Average Data Transfer Rate . . . . . 68 Kbits/sec

Data Organization

64 bytes per page
2048 pages per BPK 72A

Addressing Scheme

Logical page number

Environmental

Temperature: Temperature specifications
Operating Humidity: 0—95% Non-Condensing

BPK 72A POWER SUPPLY REQUIREMENTS

Voltage	Margin	Power Off/Power Fail Decay Rate
+ 12 Volt	± 5%	less than 1.10 volts/msec
+ 5 Volt	± 5%	less than 0.45 volts/msec

- Voltage sequencing — no restrictions
- Power on voltage rate of rise — no restrictions

BPK 72A POWER CONSUMPTION

Power (Watts)					
+ 5V (Maximum)	+ 12V (Maximum)	Total Active (Maximum)	Total Active (Typical)	Total Standby (Maximum)	Total Standby (Typical)
1.92	4.80	6.72	3.90	3.03	1.55

By using power switching techniques the bubble memory system's standby power consumption can be reduced to virtually zero. Application Note 164: **Using CMOS to Minimize Bubble Memory Power Consumption.** (Order Number: 230826-001)

PRELIMINARY

7220
CONTROLLER FOR 1MBIT BPK 70A BUBBLE MEMORY SUBSYSTEM

7220-1	0° To 75°C
7220-4	10°C To 55°C
7220-5	– 20°C To 85°C

- **Provides Interface between Host Microprocessor and 1 Mbit Bubble Subsystems**
- **Interfaces to 8080/85/86/88/186/286 and other Standard Microprocessors**
- **Controls Up to Eight BPK 70A-1, -4, Subsystems (or BPK70-1, -4)**
- **Controls up to Four BPK 70A-5 Bubble Memory Subsystems or BPK70-5**
- **16 Easy-to-Use Commands**
- **Three Modes of Data Transfer**
  - **— DMA**
  - **— Polled**
  - **— Interrupt**
- **Transfer of Single (64 bytes) or Multiple Pages of Data**

The 7220 is a complete 1 Mbit Bubble Memory Controller (BMC) that provides the interface between the microprocessor host and the 1 Mbit Bubble Memory Subsystem. All communication between the host processor and the bubble memory is performed through the controller.

The BMC interfaces easily to any Intel microprocessor or other standard microprocessor. The user has 16 easy-to-use commands available. Information such as the starting page location, the number of pages to be transferred and a read or write command is passed to the BMC before the read or write operation is initiated.

The design engineer writes a bubble memory software driver to integrate the bubble memory into his system. This interfacing with the BMC is similar to interfacing a disk drive controller. Application notes and manuals describe the details of interfacing to the BMC.

The BMC can transfer data in DMA, interrupt or polled mode. Data is transferred in and out of the bubble memory subsystem via the controller in single or multiple pages. A page size may vary from 64 bytes in a single bubble system and up to 512 bytes in an eight bubble system. (Maximum page size of 256 bytes for 7220-5.)

The BMC has an eight bit data bus plus parity bit. Word length expansion to 16 bit is possible by operating two controllers in parallel.

The BMC generates all the timing and control signals to the subsystem.

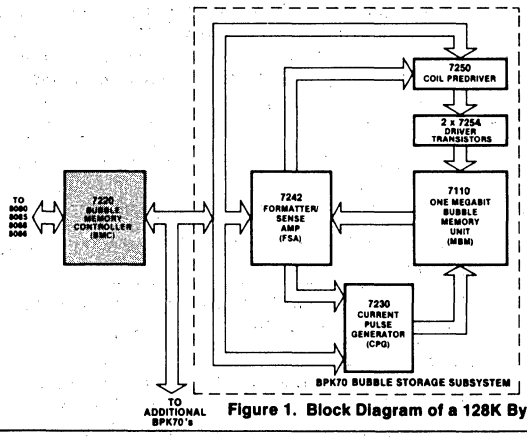


Figure 1. Block Diagram of a 128K Byte Bubble Storage System

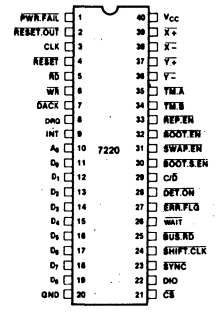


Figure 2. 7220 Pin Configuration

OCTOBER 1983
ORDER NO. 210286-002

One 7220-1 or 7220-4 BMC can control up to eight BPK 70A-1, -4 subsystems. This provides an easy expansion path to expand any 1 Mbit (128K byte) system by adding on additional subsystems. One 7220-5 can control up to four BPK 70A-5 subsystems.

The BMC is manufactured using Intel's high performance HMOS process and is packaged in a standard 40-pin dual-in-line package. All inputs are directly TTL compatible and the device uses a single +5 Volt supply.

HARDWARE DESCRIPTION

The 7220 Bubble Memory Controller is packaged in a 40-pin Dual In-Line Package (DIP). The following lists the individual pins and describes their function.

Table 1. Pin Description

Signal Name	Pin No.	I/O	Source/Destination	Description
V_{CC}	40	I		+5 VDC Supply
GND	20	I		Ground
$\overline{PWR.FAIL}$	1	I	7230 CPG	A low forces a controlled stop sequence and holds BMC in an IDLE state (similar to $\overline{RESET}$).
$\overline{RESET.OUT}$	2	O	7250 CPD/7242 FSA 7230 Reference Current Switch	An active low signal to disable external logic initiated by $\overline{PWR.FAIL}$ or $\overline{RESET}$ signals, but not active until a stopping point in a field rotation is reached (if the BMC is causing the bubble memory drive field to be rotated).
CLK	3	I	Host Bus	4 MHz, TTL-level clock.
$\overline{RESET}$	4	I	Host Bus	A low on this pin forces the interruption of any BMC sequencer activity, performs a controlled shut-down, and initiates a reset sequence. After the reset sequence is concluded, a low on this pin causes a low on the $\overline{RESET.OUT}$ pin, furthermore, the next BMC sequencer command must be either the Initialize or Abort command; all other commands are ignored.
$\overline{RD}$	5	I	Host Bus	A low on this pin enables the BMC output data to be transferred to the host data bus (D_0-D_8).
$\overline{WR}$	6	I	Host Bus	A low on this pin enables the contents of the host data bus (D_0-D_8) to be transferred to the BMC.
$\overline{DACK}$	7	I	Host Bus	A low signal is a DMA acknowledge. This notifies the BMC that the next memory cycle is available to transfer data. This line should be active only when DMA transfer is desired and the DMA ENABLE bit has been set. $\overline{CS}$ should not be active during DMA transfers except to read status. If DMA is not used, $\overline{DACK}$ requires an external pullup to V_{CC} (5.1K ohm).
DRQ	8	O	Host Bus	A high on this pin indicates that a data transfer between the BMC and the host memory is being requested.
INT	9	O	Host Bus	A high on this pin indicates that the BMC has a new status and requires servicing when enabled by the host CPU.
A_0	10	I	Host Bus	A high on this pin selects the command/status registers. A low on this pin selects the data register.
D_0-D_7	11-18	I/O	Host Bus	Host CPU data bus. An eight-bit bidirectional port which can be read or written by using the $\overline{RD}$ and $\overline{WR}$ strobes. D_0 shall be the LSB.
D_8	19	I/O	Host Bus	Parity bit.

Table 1. Pin Description (Continued)

Signal Name	Pin No.	I/O	Source/Destination	Description
$\overline{CS}$	21	I	Host Bus	Chip Select Input. A high on this pin shall disable the device to all but DMA transfers (i.e., it ignores bus activity and goes into a high impedance state).
DIO	22	I/O	7242 FSA	A bidirectional active high data line that shall be used for serial communications with 7242 FSA devices.
$\overline{SYNC}$	23	O	7242 FSA	An active low output utilized to create time division multiplexing slots in a 7242 FSA chain. It shall also indicate the beginning of a data or command transfer between BMC and 7242 FSA.
$\overline{SHIFT.CLK}$	24	O	7242 FSA	A controller generated clock that initiates data transfer between selected FSAs and their corresponding bubble memory devices. The timing of $\overline{SHIFT.CLK}$ shall vary depending upon whether data is being read or written to the bubble memory.
$\overline{BUS.RD}$	25	O	*	An active low signal that indicates that the DIO line is in the output mode, i.e., BMC is sending data to FSA. It shall be used to allow off-board expansion of 7242 FSA devices.
$\overline{WAIT}$	26	I/O	*	A bidirectional pin that shall be tied to the $\overline{WAIT}$ pin on other BMCs when operated in parallel. It shall indicate that an interrupt has been generated and that the other BMCs should halt in synchronization with the interrupting BMC. $\overline{WAIT}$ is an open collector active low signal. Requires an external pullup resistor to V_{CC} (5.1K ohm).
$\overline{ERR.FLG}$	27	I	7242 FSA	An active low input generated externally by 7242 FSA indicating that an error condition exists. It is an open collector input which requires an external pullup resistor (5.1K ohm).
$\overline{DET.ON}$	28	O	*	An active low signal that indicates the system is in the read mode and may be detecting. It is useful for power saving in the MBM.
C/$\overline{D}$	29	O	7242 FSA	A high on this line indicates that the BMC is beginning an FSA command sequence. A low on this line indicates that the BMC is beginning a data transmit or receive sequence.
$\overline{BOOT.SW.EN}$	30	O	7230 CPG	An active low signal which may be used for enabling the BOOT.SWAP of the 7230 CPG.
$\overline{SWAP.EN}$	31	O	7230 CPG	An active low signal used to create the swap function in external circuits.
$\overline{BOOT.EN}$	32	O	7230 CPG	An active low signal enabling the bootstrap loop replicate function in external circuitry.
$\overline{REP.EN}$	33	O	7230 CPG	An active low signal used to enable the replicate function in external circuitry.
$\overline{TM.B}$	34	O	7230 CPG	An active low timing signal generated by the decoder logic for determining TRANSFER pulse width.
$\overline{TM.A}$	35	O	7230 CPG	An active low timing signal generated by the decoder logic for determining CUT pulse width.
$\overline{Y-}$, $\overline{Y+}$, $\overline{X-}$, $\overline{X+}$	36-39	O	7250 CPD	Four active low timing signals generated by the decoding logic and used to create coil drive currents in the bubble memory device.

*Not used in minimum (128K byte) system.

FUNCTIONAL DESCRIPTION

The 7220 Bubble Memory Controller provides the user interface to the bubble memory system. The BMC generates all memory system timing and control, maintains memory address information, interprets and executes user request for data transfers, and provides a Microprocessor-Bus compatible interface for the magnetic bubble memory system.

Figure 3 is a block diagram of the 7220 Bubble Memory Controller (BMC). The following paragraphs describe the functions of the individual functional sections of the BMC.

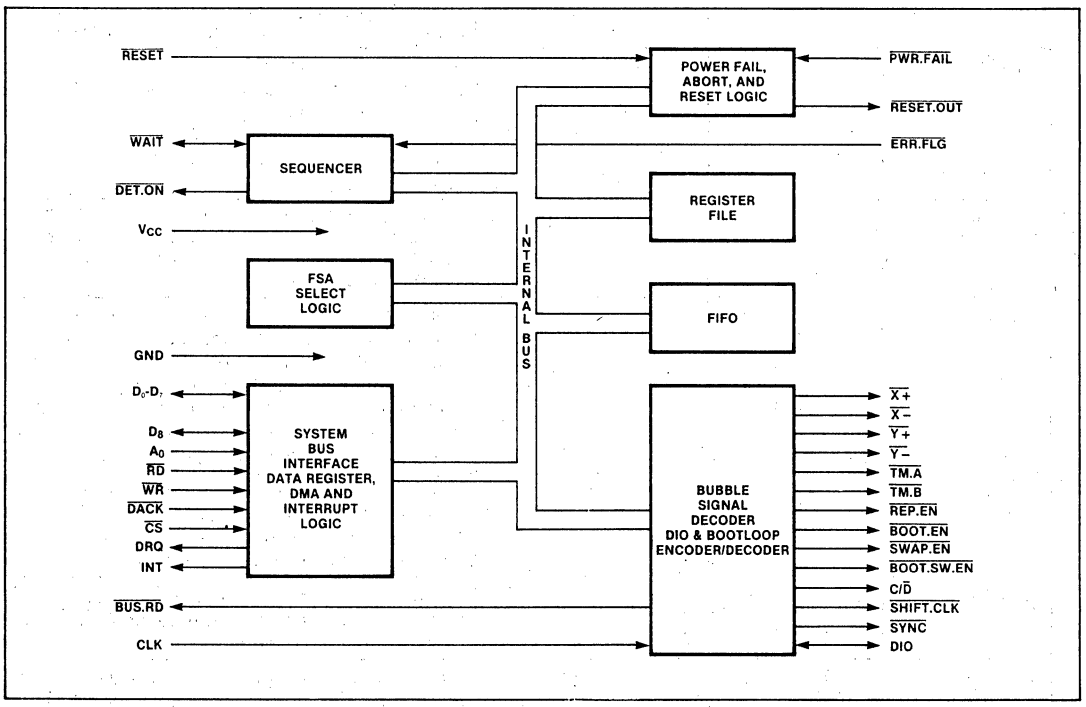


Figure 3. 7220 Bubble Memory Controller (BMC), Block Diagram

System Bus Interface—The System Bus Interface (SBI) logic contains the timing and control logic required to interface the BMC to a non-multiplexed bus. The logic also contains the circuitry to check and generate odd parity on transfers across the bus. The interface has input data, output data, and status data latches. The BMC can interface asynchronously to the host CPU. With a 4-MHz clock, it is capable of sustaining a 1.14 Mbyte per second transfer rate, while data is available in the BMC FIFO.

FIFO—The FIFO consists of a 40 x 8 bit FIFO RAM for data storage. The FIFO block also contains input and output data latches, providing double data buffering, to improve the R/W cycle times seen at the system bus interface. The FIFO may be used as a general purpose FIFO when a command is not being executed by the BMC Sequencer. In this mode, the FIFO READY status bit becomes a FIFO not-empty indicator indicating that the RAM and input/output latches have at least one byte of data.

DMA and Interrupt Logic—The DRQ pin has two functions:

(1) If the DMA enable bit in the enable register is set, the DRQ pin, in conjunction with the $\overline{DACK}$ pin, provides a standard DMA transfer capability; i.e., it has the ability to handshake with an 8257 or 9517/8237 DMA controller chip.

(2) If the DMA enable bit is reset, the DRQ pin acts as a "ready for data transfer interrupt" pin. It becomes active when 22 bytes may be read from or written into the BMC; it is reset when this condition no longer exists.

Register File—The register file contains 7 eight-bit registers that are accessible by the host CPU. Refer to the Register Section for details.

MBM Address Logic and RAM—The MBM address logic consists of the block length counter, starting address counter, adder, and MBM Address RAM. The MBM Address RAM is used to store the next available page address for each of up to 8 dual FSAs. The address maintained is the read address; the write address is generated, when needed, by adding a constant to the stored read address.

The block length counter enables multiple page transfers of up to 2048 pages in length.

The starting address counter is used as a register to hold the desired start address. Once the start address is reached, the counter is incremented on each subsequent page transfer so that its value is equal to the present read address.

DIO Bootloop Decoder/Encoder—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 $\overline{\text{BUS.RD}}$ 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 bootloop data during a Read Bootloop or Initialize operation, and encodes the bootloop data during a Write Bootloop operation.

Sequencer—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.

Power Fail and Reset—Provides a means of resetting the bubble systems in an orderly manner, when activated by the $\overline{\text{PWR.FAIL}}$ signal, the $\overline{\text{RESET}}$ signal, or the ABORT command. The additive noise on the $\overline{\text{PWR.FAIL}}$ pin should be less than 150 mV for proper powerfail operation.

FSA Select Logic block contains the logic which controls the timing of the interaction between the BMC and the FSAs. 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.

Bubble Signal Decoder block contains the logic for creating all the MBM timing signals. The BMC to bubble memory interface consists of active low timing signals. The starting and stopping point of each signal is determined by the decoder logic. Each signal may occur every field rotation or only once in a number of field rotations. The field rotation in which a timing pulse occurs is controlled by the sequencer logic.

Figure 4 and Table 2 illustrate the typical timing signals for the BMC. These signals are described in the following paragraphs.

$\overline{X+}$, $\overline{X-}$, $\overline{Y+}$, and $\overline{Y-}$ go to the 7250 CPDs, and are used to enable the coil drive currents in the MBMs.

$\overline{TM.A}$ and $\overline{TM.B}$ go to the 7230 CPGs, and are used to determine, respectively, the pulse widths for the CUT and TRANSFER functions used in replicating and generating the bubbles.

Table 2. 7220 BMC Timing (Degrees)**

Signal	Start	Width	End
$\overline{X+}$	270°	108°	378°
$\overline{Y+}$	0°	108°	108°
$\overline{X-}$	90°	108°	198°
$\overline{Y-}$	180°	108°	288°
$\overline{TM.A}$ (ODD)	270°	4°	274.5°
$\overline{TM.A}$ (EVEN)	90°	4°	94.5°
$\overline{TM.B}$ (ODD)	270°	90°	360°
$\overline{TM.B}$ (EVEN)	90°	90°	180°
$\overline{BOOT.EN}$	252°	108°	360°
$\overline{REP.EN}$	252°	108°	360°
$\overline{SWAP.EN}$	180°	5.7°	697°
$\overline{BOOT.SW.EN}$	180°	DC*	180°
$\overline{SHIFTCLK (RD)}$	186.75°	99°	285.75°
$\overline{SHIFTCLK (WR)}$	72°	288°	360°

*Stays low for 4118 field rotation periods when writing the MBM Bootloop.

**All phases relative to $\overline{Y-}$ start phase. All entries ±1.26° except $\overline{TM.A}$ width which is ±0.5°.

$\overline{SWAP.EN}$, $\overline{REP.EN}$, $\overline{BOOT.SW.EN}$, and $\overline{BOOT.EN}$ 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.

$\overline{SHIFT.CLK}$ goes to the FSAs. It is used to control the timing of events at the interface between each FSA and its corresponding MBM. (Refer to 7242 FSA Specification for a description of the BMC/FSA interface.)

$\overline{SYNC}$ and C/$\overline{D}$ control the serial communications between the BMC and the FSAs (on the DIO line).

USER-ACCESSIBLE REGISTERS

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

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 A_0) and data bus lines D_0 through D_4.

Both Command Register (CMDR) and Register Address Counter (RAC) are 4-bit registers which are loaded from D_0-D_3. The status register is selected and read by a single read request. The command register is selected and loaded by a single write request. The remaining registers are accessed indirectly, and 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.

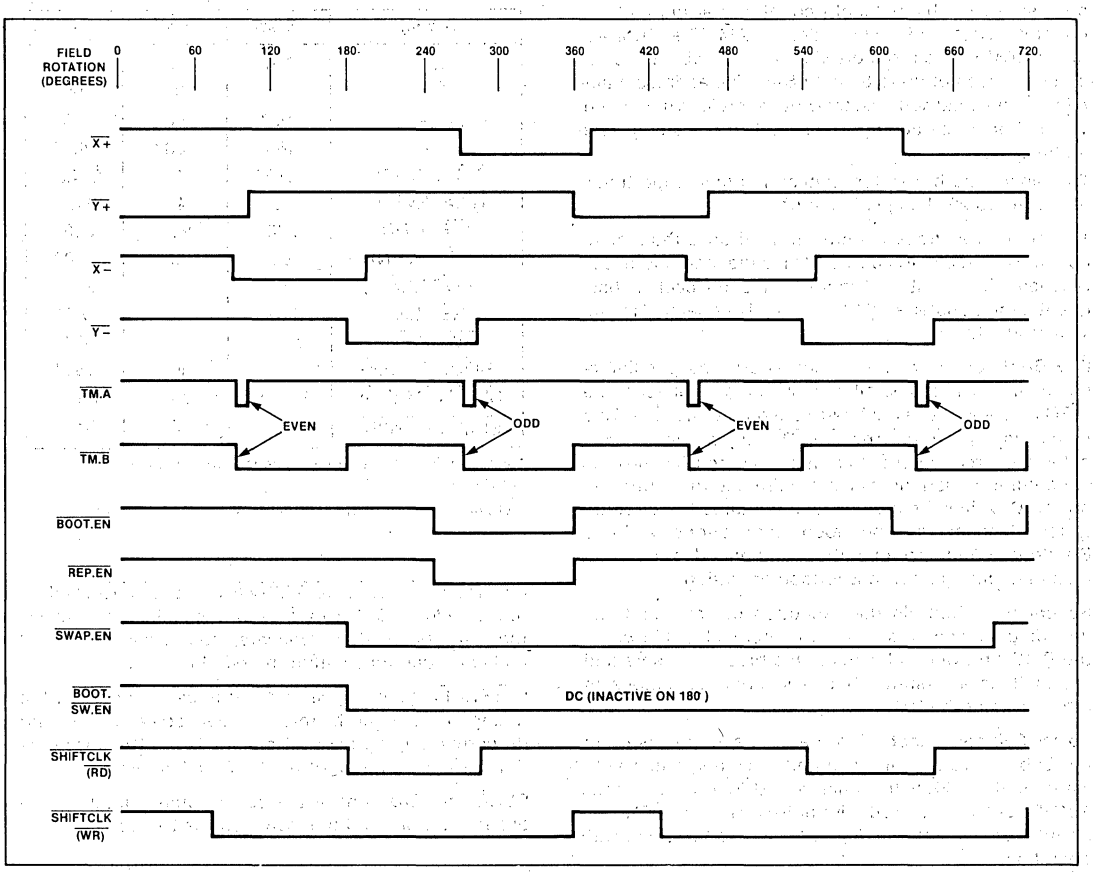


Figure 4. 7220 BMC Timing Diagram

Table 3 gives a complete listing of the address asignments 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 addressed indirectly by the contents of RAC.

Table 3. 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

Table 3. Address Assignments for the User-Accessible Registers (Continued)

RAC A0	B3	B2	B1	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

The register file contains the registers with address 1010 through 1111. These registers are also called parametric registers because they contain flags and parameters that determine exactly how the BMC will respond to commands written to the CMDR.

To facilitate such operations, the BMC automatically increments the RAC by one count after each transfer of data to or from a parametric register.

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.

REGISTER 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.

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

Table 7 is a listing of the commands and their functions.

Table 4. Command Code Definitions

D3	D2	D1	Do	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 most commonly used commands in normal operation are:

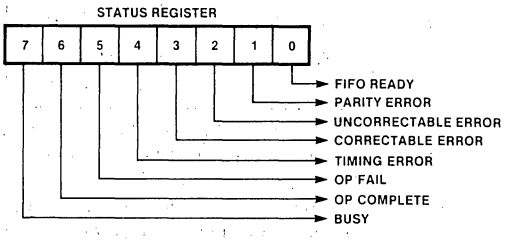
Initialize
Read Bubble Data
Write Bubble Data
Reset FIFO
Read Seek
Write Seek
Abort
Read Corrected Data
Software Reset
Read FSA Status
MBM Purge

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

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 = 1) indicates that the BMC is in the process of executing a command. When equal to 0, BUSY indicates that the BMC is ready to receive a new command. In the case of Read Bubble Data, Read Bootloop, read Bootloop Register, or Read Corrected Data commands, BUSY may also indicate that the data has not been completely removed from the FIFO, and that DRQ is still active. BUSY will then drop as soon as DRQ does (after the user has finished reading the data remaining in the FIFO).

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

OP FAIL (when = 1) indicates that the BUSY bit has gone inactive with either the TIMING ERROR or UNCORRECTABLE ERROR bits active.

TIMING ERROR (when = 1) indicates that a 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 underflow. TIMING ERROR is also set if no bootloop sync word is found during initialization, or if a Write Bootloop command is issued when the WRITE BOOTLOOP ENABLE bit is equal to zero in the enable register.

CORRECTABLE ERROR (when = 1) indicates that a 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 at least one 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 D_0-D_8.

FIFO READY has two functions. The FIFO READY functions are as follows:

NOTE: IF RAC ≠ FIFO, FIFO READY = 1

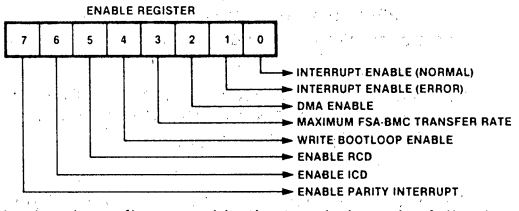
STATUS BITS		READ	WRITE
FIFO READY	BUSY		
1	1	data in FIFO	space in FIFO
0	1	no data	no space
1	0	— data in FIFO —	
0	0	— FIFO empty —	

Although the status word can be read at any time, the status information, bit 1 through 6, is not valid until the BUSY bit is low.

STR Bits 1 through 6 are reset when a new command is issued. They may also be reset by making a write request (WR = 0) to the BMC with $A_0 = 1$, $D_4 = 0$, and $D_5 = 1$ (Modifier Bit) (that is, writing the RAC with $D_5 = 1$). This operation also resets the "INT" pin to "0". NOTE: A byte of FIFO data can be lost when using this procedure if the RAC is written to other than the FIFO address when data is still present in FIFO.

Enable Register (ER) 8 Bits, Write Only

The user sets various bits of the enable register to 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 D_0-D_7.

ENABLE ICD enables the BMC to give the Internally Correct Data command to the FSAs when an error has been detected by the FSA's error detection and correction circuitry. Each 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).

ENABLE RCD enables the BMC to give the Read Corrected Data command to the FSAs when an error has been detected. This causes each FSA to correct the error (if possible) and also to transfer the corrected data to the BMC. The Read Corrected Data command is also used to read into the BMC data previously corrected by the FSA in response to an Internally Correct Data command. In either case, when the data transfer has been completed, the BMC reads each FSA's status to determine whether or not the error was correctable. In the case of an uncorrectable error, bad data may have been sent to the user. The value of ENABLE RCD affects the action of INTERRUPT ENABLE (ERROR).

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 controls the maximum burst transfer rate from FSA(s) to BMC FIFO. This rate is variable on the "last page" of a multiple page transfer. (In one page transfers the last page is the only page.) See Table 5 for effects of this bit on the various 7220 commands.

Table 5. MFBTR Bit Definitions

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 Bubble Data."

DMA ENABLE (when = 1) enables the BMC to operate in DMA data transfer mode, using the DRQ and $\overline{DACK}$ 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.

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

TE = Timing Error, CE = Correctable Error,
UCE = Uncorrectable Error.

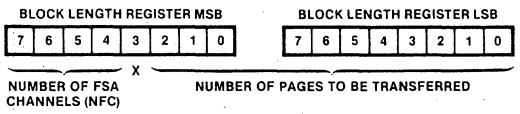
INTERRUPT ENABLE (NORMAL) (when = 1) enables the BMC to interrupt the host system (via the INT line), when a command execution has been successfully completed (OP COMPLETE = 1 in the STR).

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:

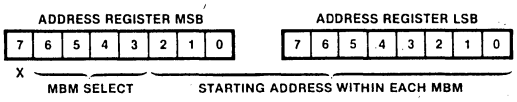


The system page size is proportional to the number of magnetic bubble memory modules (MBMs) operating in parallel during the data read or write operation. Each MBM requires two FSA channels. Bits 4 through 7 of BLR MSB actually specify the number of FSA channels to be accessed.

The BLR LSB, together with the 3 least significant bits of the BLR MSB, specify 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.

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.

Table 6 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 6. Selection of FSA Channels

AR MSB Bits (6,5,4,3)	BLR MSB Bits (7,6,5,4)				
	0000	0001	0010	0100	1000
0000	0	0,1	0,1,2,3	0 to 7	0 to F
0001	1	2,3	4,5,6,7	8 to F	
0010	2	4,5	8,9,A,B		
0011	3	6,7	C,D,E,F		
0100	4	8,9			
0101	5	A,B			
0110	6	C,D			
0111	7	E,F			
1000	8				
1001	9				
1010	A				
1011	B				
1100	C				
1101	D				
1110	E				
1111	F				

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 FIFO allows the data transfer to proceed in an 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. 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.

FUNCTIONAL OPERATION

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.

Each one-Megabit Bubble Memory (MBM) operates in its own domain, and is unaffected by the number of bubble memories in the system. The roles played by the MBM's immediate support circuitry can be described as if the system contained only one MBM module.

Data Flow Within the Magnetic Bubble Memory (MBM) System (Single MBM Systems)

During a read operation, data flows as follows: The data from the MBM is input to 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 data is paired up with the corresponding bit in the FSA's bootloop 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.

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 $\overline{\text{SYNC}}$ pulse to the $\overline{\text{SELECT.IN}}$ 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 $\overline{\text{SYNC}}$ pulses, once every 20 or 80 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. From this FIFO, the data bytes are written onto the user interface.

During a write operation, the data flow consists of the corresponding operations in the reverse order.

INTERFACING REQUIREMENTS

All communications between the host microprocessor, and the bubble memory is performed through the 7220 BMC. The **BPK 72 Bubble Memory Prototype Kit User's Manual,** Order Number: 121685, contains detailed information on how to use and interface the BMC. Below the general principles are described, for detailed guidelines please refer to the BPK 72 Manual. For software considerations, please also see Application Note AP-157. (Order Number: 230707)

First the hardware interfacing requirements and second the software interfacing requirements are described.

HARDWARE INTERFACE REQUIREMENTS

User Interface Signals

The source, destination and function of the user interface signals are described in Table 1 in the data sheet.

Table 7. Detailed Command Descriptions

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. All the parametric registers must be properly set up before issuing the Initialize command.
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 and the starting address for the read operation is 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 and the starting address for the write operation is 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 before issuing the Write Bubble Data command.
Read Seek	The Read Seek command rotates the selected MBMs to a designated page address location. 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.

Table 7. Detailed Command Descriptions (Continued)

Write Seek	The Write Seek command rotates the selected MBMs to a designated page address location. 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.
Abort	The Abort command causes a controlled termination of the command currently being executed by the BMC. The Abort command will be accepted by the BMC (and is typically issued) when the BMC is busy.
MBM Purge	The MBM Purge command clears all BMC registers, counters, and the MBM address RAM Furthermore, it determines how many FSA channels are present in the system and stores this value in the 7220. The "INITIALIZE" command uses this command as a subroutine.
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 an error has been detected. The data cycles through the error correction network of the FSA. After the data has been read, 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).
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 selected FSAs in the system. No reinitialization 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 FSAs, and to store this information in the BMC FIFO. The Read FSA Status command is independent all parametric registers.
Read Bootloop Register	The Read Bootloop Register command causes the BMC to read the bootloop register of the selected FSA channels and to store this information in the BMC FIFO. Twenty bytes are transferred for each FSA channel selected.
Write Bootloop Register Masked	Proper operation of the FSAs during data transfer to or from the MBMs requires that the bootloop register contain (if error correction is used) exactly 270 logic 1s for each FSA bootloop register. The user may select any subset of 270 "good" loops from the total number of available loops (if error correction is not used, 270 replaced by 272). As an alternative, the Write Bootloop Register Masked command may be used. This command counts the number of logic 1s and masks out the remaining 1s after the proper count has been reached. The Initialize command uses this command as a subroutine.
Read Bootloop	The Read Bootloop command causes the BMC to read the bootloop from the selected MBM, and to store the decoded bootloop information in the BMC FIFO. The Initialize command uses this command as a subroutine.
Write Bootloop	The Write Bootloop command causes the existing contents of the selected MBM's bootloop to be replaced by new bootloop data based on 40 bytes of information stored in the FIFO (the user must actually write 41 bytes, where the 41st byte is all 0s). Encoding of the bootloop data is done by the BMC hardware.

System Timing

As shown on the timing diagrams in the WAVEFORM section the typical read/write cycle timing provides sufficient tolerance to allow most currently available microprocessors to be easily adapted to the BMC timing requirements.

User Data Transfer Rate Requirements

The maximum data rate for the user interface is a function of the number of MBMs operated in parallel as outlined in table 8. The rates listed must be considered in relation to the data transfer mode (polled, interrupt-driven, or DMA) to be implemented in order to be sure that the host system software and hardware are capable of keeping up with the data transfer. In other words, the BMC requires the host CPU to be able to sustain the maximum data rate transfer rate for the minimum data transfer (e.g., for a one bubble system keep up the transfer rate for at least 64 bytes = one page).

Table 8. User Data User 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 kbytes/second	50 kbytes/second	12.5 kbytes/second
2	25 kbytes/second	100 kbytes/second	25 kbytes/second
4	50 kbytes/second	200 kbytes/second	50 kbytes/second
8	100 kbytes/second	400 kbytes/second	100 kbytes/second

Hardware Interfacing for Data Transfer

The BMC supports three data transfer modes, i.e. DMA, interrupt-driven and polled.

To support DMA, a hardware mechanism is required for servicing the BMC's data transfer requests. While several hardware implementations are possible, one common configuration is the Intel 8257 DMA controller.

To support an interrupt-driven system an Intel 8259 Programmable Interrupt Controller is often used.

The polled data transfer mode relies almost exclusively on the software interaction between the host processor and the BMC to control the transfer of data.

Multiple MBM-System

A BMC is capable of processing data and of supplying the required timing and control signals for operating up to four Bubble Storage Units (BSUs), each of which is capable of storing 128 kbytes of user data. A BSU consists of a 128 kbyte MBM and its five immediate IC support chips (i.e. a MBPK 70A-5 Kit).

SOFTWARE INTERFACE REQUIREMENTS

To use the BMC, the user has to write a "bubble memory software driver".

The bubble driver is responsible for all the system interaction with the bubble memory controller and is intrinsic to the efficient and reliable operation of the bubble system. The driver accepts bubble memory commands and command execution parameters from the application program, controls and monitors command execution, and returns operational status information to the application program at command completion. To perform all of these operations, the bubble driver must support the bit/byte level of the bubble memory controller's command and status registers and the parametric registers that define the operating mode, system configuration, and extent of the transfer.

The level of the software driver complexity is a function of the specific application needs. Regardless, a set of basic drivers must be developed that in turn are integrated into a system at the appropriate level. If an application program is small and simple, a basic bubble driver may simply be called from the main program.

At the highest level of driver sophistication, the application program treats the bubble system as a collection of named data areas of files similar to the way in which data is stored and retrieved in disk operating systems. At the file system level, an application program can ignore the mechanics of bubble storage and access and merely present a file name to the driver to open, read, or write, then close the desired bubble file.

Data Organization

From a software viewpoint, data logically is organized into blocks of bytes called pages. During data transfer operations, one or more of these pages are transferred between the bubble(s) and the host microprocessor. A page is the smallest increment of data that can be transferred; single bytes cannot be transferred. Conceptually, the data organization within a bubble memory is analogous to a disk system. Just as disk sector sizes are fixed when a disk is formatted, bubble page sizes are established, under software control when the bubble system is initialized.

For a single bubble system, the page size is fixed at either 64 bytes when error correction is implemented or 68 bytes without error correction, and the total number of pages available is 2048. In systems with multiple bubbles, page size can vary from 64 bytes (68 bytes without error correction) to 256 bytes (272 bytes without error correction) depending on the number of bubble devices in the system. Page size is directly proportional to the system data rate and also determines the total number of available pages (address field size). The selection of the appropriate page size depends primarily on the data rate supported by the system. The higher the data rate, the faster the microprocessor must respond to the demands of the bubble memory controller.

Buffering

The bubble memory controller includes a FIFO data buffer that, although only 40 bytes long, reconciles timing

differences between the parallel data transfer to or from the host microprocessor and the serial data transfer to or from the Bubble Memory Subsystem. Accordingly, when an application program requests data from a bubble, the software driver is responsible for keeping up with the FIFO for the duration of the data transfer in order to prevent the FIFO from overflowing or underflowing.

Command Execution

Command execution can be performed either in an interrupt driven mode or in a polled mode irrespective of the data transfer mode (polled, interrupt-driven, or DMA).

Data Transfer Mode

As described earlier in the hardware section, three data transfer modes are available (polled, interrupt-driven or DMA).

System performance, additional hardware and software overhead are all important considerations when choosing the appropriate mode for your application.

Error Correction

The bubble memory system has a built-in error correction. To insure highest data integrity, the error correction should always be used. Three levels of error correction are available.

Communication with the BMC

All communications between the host and the bubble memory actually are performed through the BMC. The BMC has two input/output (I/O) ports, an eight-bit bidirectional data port, and an eight-bit command/status port. Conceptually, a bubble memory system can be thought of as a disk system in that data in the bubble memory is organized into blocks called pages in bubble technology that are similar to disk sectors. Information such as starting page location, direction of transfer, and the number of pages to be transferred is passed to the BMC before the desired read or write operation is initiated.

The general procedure for communicating with the BMC is:

Set-up the BMC for data transfer communication by loading specific parameters in user-accessible registers.

Send the desired command.

Read the status register to determine if command is accepted.

If applicable, transfer (i.e., read or write) data.

Read the status register until BMC is not busy (or under some conditions "INT" pin).

Examine the status register to determine whether the operation was successful.

For all details and exceptions to this general description, see AP-157 or the BPK 72 User's Manual.

ABSOLUTE MAXIMUM RATINGS

Temperature under bias ..... −40 to +100°C
Storage Temperature ..... −65°C to +150°C
All Input or Output Voltages and
V_{CC} Supply Voltage ..... −0.5V to 7V

**NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS (T_A = see front page; V_{CC} = 5.0V +5%, −10%; * = 7220-5)

Symbol	Parameter	Min.	Max.	Unit	Test Condition
V_{IL}	Input Low Voltage		0.8	V	
$V_{IH(1)}$	Input High Voltage (all but $\overline{PWR.FAIL}$)	2.0(2.2*)	V_{CC}+0.5V	V	
$V_{IH(2)}$	Input High Voltage ($\overline{PWR.FAIL}$)	2.5	V_{CC}+0.5V	V	
$V_{OL(1)}$	Output Low Voltage (All outputs except $\overline{DET.ON}$, $\overline{BUS.RD}$, $\overline{SHIFT.CLK}$, and $\overline{SYNC}$		.45	V	I_{OL} = 3.2 mA
$V_{OL(2)}$	Output Low Voltage $\overline{DET.ON}$, $\overline{BUS.RD}$, $\overline{SHIFT.CLK}$, $\overline{SYNC}$		.45	V	I_{OL} = 1.6 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = 400μA
$\|I_{IL}\|$	Input Leakage Current		10	μA	$0 \leq V_{IN} \leq V_{CC}$
$\|I_{OFL}\|$	Output Float Leakage		10	μA	$0.45 \leq V_{OUT} \leq V_{CC}$
I_{CC}	Power Supply Current from V_{CC}		200	mA	

A.C. CHARACTERISTICS

(T_A = see front page; V_{CC} = 5.0V + 5%, −10%, C_L = 150pF; unless otherwise noted.)

Symbol	Parameter	Min.	Max.	Unit	Test Condition
t_P	Clock Period	249.75	250.25	ns	
t_ϕ	Clock Phase Width (High Time)	.45 t_P	.55 t_P		
t_R-t_F	Input Signal Rise and Fall Time		30	ns	

FSA INTERFACE TIMINGS (under pin loading)

Symbol	Parameter	Min.	Max.	Unit	Test Condition
t_{CDV}	CLK to DIO Valid Delay		150	ns	Under Pin Loads*
t_{CDF}	CLK to DIO Entering Float	10	250	ns	Under Pin Loads*
t_{CDE}	CLK to DIO Enabled from Float		150	ns	Under Pin Loads*
t_{CDH}	CLK to DIO Hold Time	0		ns	Under Pin Loads*
t_{CSOL}	CLK to $\overline{SYNC}$ Leading Edge Delay		120	ns	Under Pin Loads*
t_{CSOT}	CLK to $\overline{SYNC}$ Trailing Edge Delay	10	100	ns	Under Pin Loads*
t_{DC}	DIO Setup Time to Clock	80		ns	Under Pin Loads*
t_{DHC}	DIO Hold Time from Clock	0		ns	Under Pin Loads*
t_{COL}	CLK to Output Leading Edge		150	ns	Under Pin Loads*
t_{COT}	CLK to Output Trailing Edge	0	190	ns	Under Pin Loads*
t_{EW}	$\overline{ERR.FLG}$ Pulse Width	200		ns	Under Pin Loads*
t_{SCFT}	$\overline{SHIFTCLK}$ to $\overline{Y-}$ Trailing Edge	80	200	ns	Under Pin Loads*

A.C. CHARACTERISTICS (Continued) (T_A = see front page; V_{CC} = 5.0 ± 5%, −10%; C_L = 150pF; unless otherwise noted; ** = 7220-5.)

READ CYCLE (HOST INTERFACE)

Symbol	Parameter	Min.	Max.	Unit	Test Condition
t_{AC}	Select Setup to $\overline{RD}\downarrow$	0		ns	
t_{CA}	Select Hold from $\overline{RD}\uparrow$	0		ns	
t_{RR}	$\overline{RD}$ Pulse Width	200		ns	
t_{AD}	Data Delay from Address		150(200**)	ns	
t_{RD}	Data Delay from $\overline{RD}\downarrow$		150(200**)	ns	
t_{DF}	Output Float Delay	10	100	ns	
t_{DC}	DACK Setup to $\overline{RD}\downarrow$	0		ns	
t_{CD}	DACK Hold from $\overline{RD}\uparrow$	0		ns	
t_{KD}	Data Delay from $\overline{DACK}\downarrow$		150(200**)	ns	
t_{CYCR}	"Read" Cycle Time	(DMA Mode) $4t_p$-t_ϕ		ns	In non DMA mode. t_{CYCR} Min. = $6t_p$-t_ϕ

WRITE CYCLE (HOST INTERFACE)

Symbol	Parameter	Min.	Max.	Unit	Test Condition
t_{AC}	Select Setup to $\overline{WR}\downarrow$	0		ns	
t_{CA}	Select Hold from $\overline{WR}\uparrow$	0		ns	
t_{WW}	$\overline{WR}$ Pulse Width	200		ns	
t_{DW}	Data Setup to $\overline{WR}\uparrow$	200		ns	
t_{WD}	Data Hold from $\overline{WR}\uparrow$	0		ns	
t_{DC}	$\overline{DACK}$ Setup to $\overline{WR}\downarrow$	0		ns	
t_{CD}	$\overline{DACK}$ Hold from $\overline{WR}\uparrow$	0		ns	
t_{CYCW}	"Write" Cycle Time	$4t_P + t_{ww}$			
t_{CQ}	Request Hold from $\overline{RD}$ or $\overline{WR}$ (Non-Burst Mode)		200	ns	
t_{DEADW}	Inactive Time between $\overline{WR}\uparrow$ and $\overline{WR}\downarrow$	$4t_P$		ns	
t_{DEADR}	Inactive Time between $\overline{RD}\uparrow$ and $\overline{RD}\downarrow$	150			

MD7250-MD7230 INTERFACE TIMINGS

Symbol	Parameter	Min.	Max.	Unit	Test Condition
t_{CBL}	CLK to Bubble Signal Leading Edge		250(275**)	ns	Under Pin Loads*
t_{CBT}	CLK to Bubble Signal Trailing Edge		250(275**)	ns	Under Pin Loads*

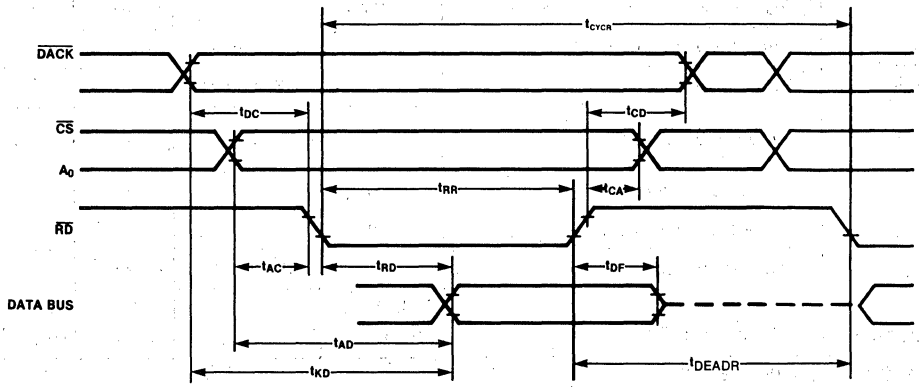
*Bubble Pin Loads Shown Below

PIN LOADINGS

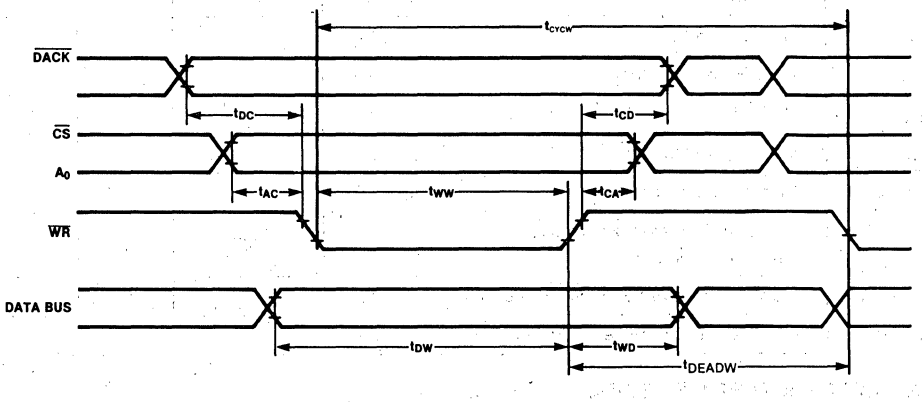
Pin Names	Value	Unit
$\overline{X+}$, $\overline{X-}$, $\overline{Y+}$, $\overline{Y-}$	150	pF
$\overline{TM.A}$, $\overline{TM.B}$, $\overline{REP.EN}$, $\overline{BOOT.EN}$, $\overline{SWAP.EN}$, $\overline{BOOT.SW.EN}$, $\overline{C/D}$, $\overline{ERR.FLG}$, $\overline{WAIT}$, $\overline{SYNC}$, DIO	100	pF
$\overline{DET.ON}$ & $\overline{SHIFT.CLK}$	100	pF
$\overline{BUS.READ}$	10	pF

WAVEFORMS

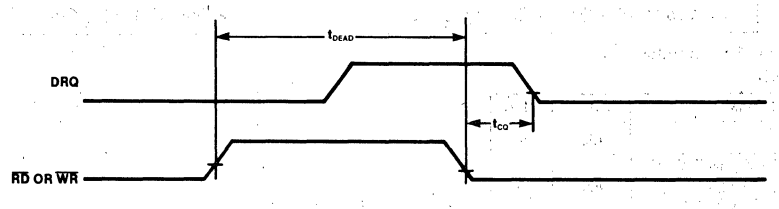
READ (HOST INTERFACE)



WRITE (HOST INTERFACE)



DMA (HOST INTERFACE)



ADVANCE INFORMATION

7224
CONTROLLER FOR 4MBIT BPK 5V74 BUBBLE MEMORY SUBSYSTEM

- **Provides Interface between Host Microprocessor and 4 Mbit Bubble Memory Subsystems**
- **Interfaces to 8080/85/86/88/186/286 and Other Standard Microprocessors**
- **Controls up to Eight Bubble Memory Subsystems**
- **18 Easy-to-Use Commands**
- **Three Modes of Data Transfer**
  - **— DMA**
  - **— Polled**
  - **— Interrupt**
- **Transfer of Single (64 bytes) or Multiple Pages of Data**

The 7224 is a complete 4 Mbit Bubble Memory Controller (BMC) that provides the interface between the microprocessor host and the 4 Mbit Bubble Memory Subsystem. All communication between the host processor and the bubble memory is performed through the controller.

The BMC interfaces easily to any Intel microprocessor or other standard microprocessor. The user has 18 easy-to-use commands available. Information such as the starting page location, the number of pages to be transferred and a read or write command is passed to the BMC before the read or write operation is initiated.

The 18 commands of the 4Mbit BMC is a superset of the 1Mbit BMC's 16 commands providing an easy up-grade of software from the 1Mbit to the 4Mbit system.

The design engineer writes a bubble memory software driver to integrate the bubble memory into his system. This interfacing with the BMC is similar to interfacing a disk drive controller. Application notes and manuals describe the details of interfacing to the BMC.

The BMC can transfer data in DMA, interrupt or polled mode. Data is transferred in and out of the bubble memory subsystem via the controller in single or multiple pages. A page size may vary from 64 bytes in a single bubble system and up to 512 bytes in an eight bubble system.

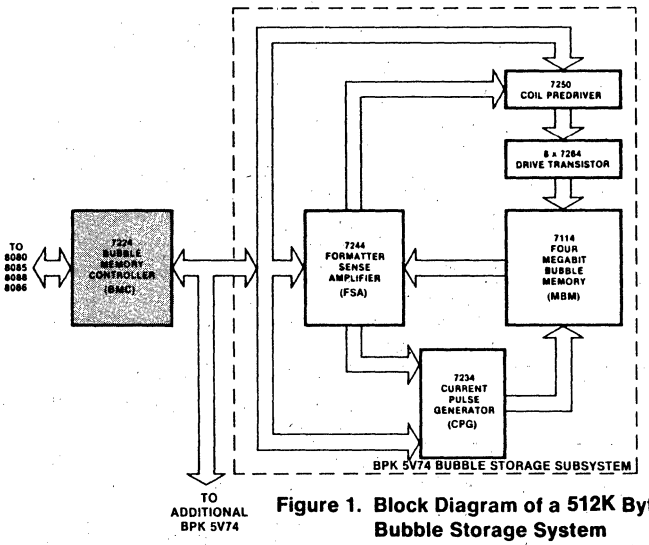


Figure 1. Block Diagram of a 512K Byte Bubble Storage System

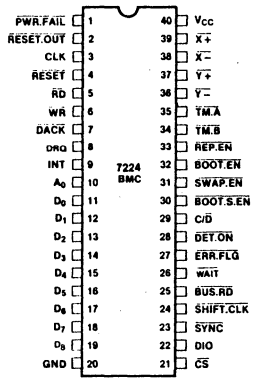


Figure 2. Pin Configuration

OCTOBER 1983
ORDER NO. 230865-001

The BMC has an eight bit data bus plus parity bit. Word length expansion to 16 bit is possible by operating two controllers in parallel.

The BMC generates all the timing and control signals to the BPK 5V74 subsystem.

One BMC can control up to eight BPK 5V74 subsystems. This provides an easy expansion path to expand any 4 Mbit (512K byte) system up to 32Mbit (4Mbyte) by adding on additional BPK 5V74 subsystems.

The BMC is manufactured using Intel's high performance HMOS process and is packaged in a standard 40-pin dual-in-line package. All inputs are directly TTL compatible and the device uses a single +5 Volt supply.

HARDWARE DESCRIPTION

The 7224 Bubble Memory Controller is packaged in a 40-pin Dual In-Line Package (DIP). The following lists the individual pins and describes their function.

Table 1. Pin Description

Signal Name	Pin No.	I/O	Source/Destination	Description
V_{CC}	40	I		+5 VDC Supply
GND	20	I		Ground
$\overline{PWR.FAIL}$	1	I	7234 CPG	A low forces a controlled stop sequence and holds BMC in an IDLE state (similar to $\overline{RESET}$).
$\overline{RESET.OUT}$	2	O	7250 CPD/7244 FSA 7234 Reference Current Switch	An active low signal to disable external logic initiated by $\overline{PWR.FAIL}$ or $\overline{RESET}$ signals, but not active until a stopping point in a field rotation is reached (if the BMC is causing the bubble memory drive field to be rotated).
CLK	3	I	Host Bus	2 MHz, TTL-level clock.
$\overline{RESET}$	4	I	Host Bus	A low on this pin forces the interruption of any BMC sequencer activity, performs a controlled shut-down, and initiates a reset sequence. After the reset sequence is concluded, a low on this pin causes a low on the $\overline{RESET.OUT}$ pin, furthermore, the next BMC sequencer command must be either the Initialize or Abort command; all other commands are ignored.
$\overline{RD}$	5	I	Host Bus	A low on this pin enables the BMC output data to be transferred to the host data bus (D_0-D_8).
$\overline{WR}$	6	I	Host Bus	A low on this pin enables the contents of the host data bus (D_0-D_8) to be transferred to the BMC.
$\overline{DACK}$	7	I	Host Bus	A low signal is a DMA acknowledge. This notifies the BMC that the next memory cycle is available to transfer data. This line should be active only when DMA transfer is desired and the DMA ENABLE bit has been set. $\overline{CS}$ should not be active during DMA transfers except to read status. If DMA is not used, $\overline{DACK}$ requires an external pullup to V_{CC} (5.1K ohm).
DRQ	8	O	Host Bus	A high on this pin indicates that a data transfer between the BMC and the host memory is being requested.
INT	9	O	Host Bus	A high on this pin indicates that the BMC has a new status and requires servicing when enabled by the host CPU.
A_0	10	I	Host Bus	A high on this pin selects the command/status registers. A low on this pin selects the data register.
D_0-D_7	11-18	I/O	Host Bus	Host CPU data bus. An eight-bit bidirectional port which can be read or written by using the $\overline{RD}$ and $\overline{WR}$ strobes. D_0 shall be the LSB.
D_8	19	I/O	Host Bus	Parity bit.

Table 1. Pin Description (Continued)

Signal Name	Pin No.	I/O	Source/Destination	Description
$\overline{CS}$	21	I	Host Bus	Chip Select Input. A high on this pin shall disable the device to all but DMA transfers (i.e., it ignores bus activity and goes into a high impedance state).
DIO	22	I/O	7244 FSA	A bidirectional active high data line that shall be used for serial communications with 7244 FSA devices.
$\overline{SYNC}$	23	O	7244 FSA	An active low output utilized to create time division multiplexing slots in a 7244 FSA chain. It shall also indicate the beginning of a data or command transfer between BMC and 7244 FSA.
$\overline{SHIFT.CLK}$	24	O	7244 FSA	A controller generated clock that initiates data transfer between selected FSAs and their corresponding bubble memory devices. The timing of $\overline{SHIFT.CLK}$ shall vary depending upon whether data is being read or written to the bubble memory.
$\overline{BUS.RD}$	25	O	To User External Circuit	An active low signal that indicates that the DIO line is in the output mode, i.e., BMC is sending data to FSA. It shall be used to allow off-board expansion of 7244 FSA devices.
$\overline{WAIT}$	26	I/O	To Alternate Controller(s) When User System Uses More Than One Controller.	A bidirectional pin that shall be tied to the $\overline{WAIT}$ pin on other BMCs when operated in parallel. It shall indicate that an interrupt has been generated and that the other BMCs should halt in synchronization with the interrupting BMC. $\overline{WAIT}$ is an open collector active low signal. Requires an external pullup resistor to V_{CC} (5.1K ohm).
$\overline{ERR.FLG}$	27	I	7244 FSA	An active low input generated externally by 7244 FSA indicating that an error condition exists. It is an open collector input which requires an external pullup resistor (5.1K ohm).
$\overline{DET.ON}$	28	O	To User External Circuit	An active low signal that indicates the system is in the read mode and may be detecting. It is useful for power saving in the MBM.
C/$\overline{D}$	29	O	7244 FSA	A high on this line indicates that the BMC is beginning an FSA command sequence. A low on this line indicates that the BMC is beginning a data transmit or receive sequence.
$\overline{BOOT.SW.EN}$	30	O	7234 CPG	An active low signal which may be used for enabling the BOOT.SWAP of the 7234 CPG.
$\overline{SWAP.EN}$	31	O	7234 CPG	An active low signal used to create the swap function in external circuits.
$\overline{BOOT.EN}$	32	O	7234 CPG	An active low signal enabling the bootstrap loop replicate function in external circuitry.
$\overline{REP.EN}$	33	O	7234 CPG	An active low signal used to enable the replicate function in external circuitry.
$\overline{TM.B}$	34	O	7234 CPG	An active low timing signal generated by the decoder logic for determining TRANSFER pulse width.
$\overline{TM.A}$	35	O	7234 CPG	An active low timing signal generated by the decoder logic for determining CUT pulse width.
$\overline{Y-}$, $\overline{Y+}$, $\overline{X-}$, $\overline{X+}$	36-39	O	7250 CPD	Four active low timing signals generated by the decoding logic and used to create coil drive currents in the bubble memory device.

FUNCTIONAL DESCRIPTION

The 7224 Bubble Memory Controller provides the user interface to the bubble memory system. The BMC generates all memory system timing and control, maintains memory address information, interprets and executes user request for data transfers, and provides a Microprocessor-Bus compatible interface for the magnetic bubble memory system.

Figure 3 is a block diagram of the 7224 Bubble Memory Controller (BMC). The following paragraphs describe the functions of the individual functional sections of the BMC.

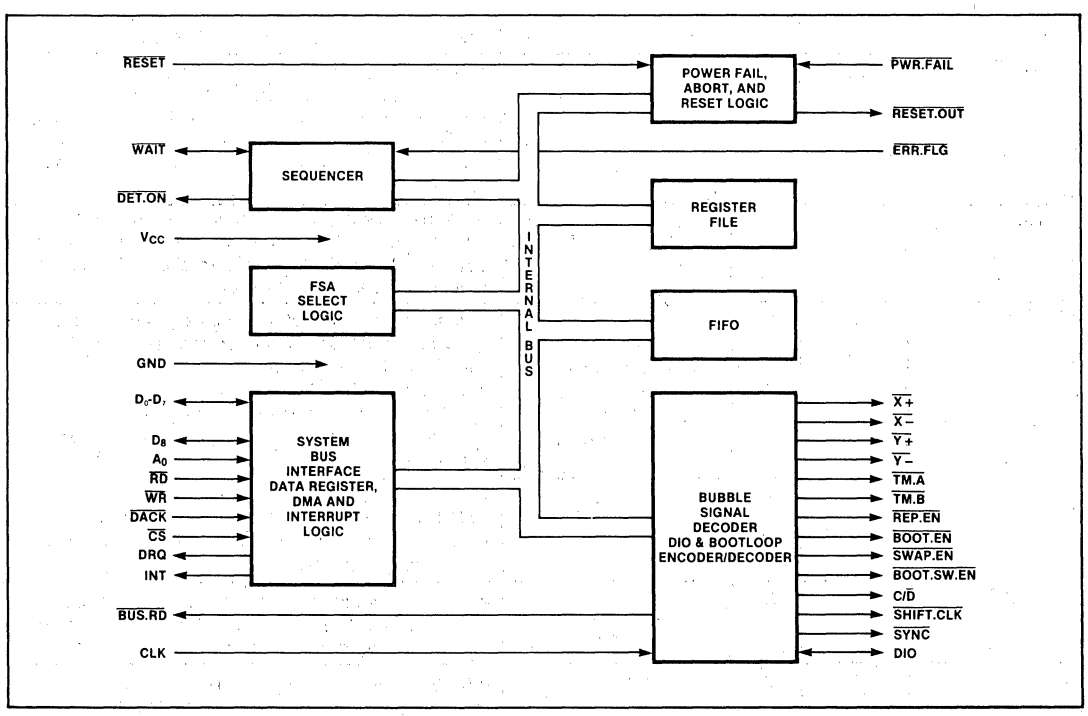


Figure 3. 7224 Bubble Memory Controller (BMC), Block Diagram

System Bus Interface—The System Bus Interface (SBI) logic contains the timing and control logic required to interface the BMC to a non-multiplexed bus. The logic also contains the circuitry to check and generate odd parity on transfers across the bus. The interface has input data, output data, and status data latches. The BMC can interface asynchronously to the host CPU. With a 2-MHz clock, it is capable of sustaining a 1 Mbyte per second transfer rate while data is available in the BMC FIFO.

FIFO—The FIFO consists of a 40 x 8 bit FIFO RAM for data storage. The FIFO block also contains input and output data latches, providing double data buffering, to improve the R/W cycle times seen at the system bus interface. The FIFO may be used as a general purpose FIFO when a command is not being executed by the BMC Sequencer. In this mode, the FIFO READY status bit becomes a FIFO not-empty indicator indicating that the RAM and input/output latches have at least one byte of data.

DMA and Interrupt Logic—The DRQ pin has two functions:

(1) If the DMA enable bit in the enable register is set, the DRQ pin, in conjunction with the $\overline{DACK}$ pin, provides a standard DMA transfer capability; i.e., it has the ability to handshake with an 8257 or 9517/8237 DMA controller chip.

(2) If the DMA enable bit is reset, the DRQ pin acts as a "ready for data transfer interrupt" pin. It becomes active when 22 bytes may be read from or written into the BMC; it is reset when this condition no longer exists.

Register File— The register file contains 6 eight-bit registers that are accessible by the host CPU. Refer to the Register Section for details.

MBM Address Logic and RAM—The MBM address logic consists of the block length counter, starting address counter, adder, and MBM Address RAM. The MBM Address RAM is used to store the next available page address for each of up to 8 dual FSAs. The address maintained is the read address; the write address is generated, when needed, by adding a constant to the stored read address.

The block length counter enables multiple page transfers of up to 2048 pages in length.

The starting address counter is used as a register to hold the desired start address. Once the start address is reached, the counter is incremented on each subsequent page transfer so that its value equal to the present read address. There are 8192 possible starting addresses.

DIO Bootloop Decoder/Encoder—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 $\overline{\text{BUS.RD}}$ 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 bootloop data during a Read Bootloop or Initialize operation, and encodes the bootloop data during a Write Bootloop operation.

Sequencer—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.

Power Fail and Reset—Provides a means of resetting the bubble systems in an orderly manner, when activated by the $\overline{\text{PWR.FAIL}}$ signal, the $\overline{\text{RESET}}$ signal, or the ABORT command. The additive noise on the $\overline{\text{PWR.FAIL}}$ pin should be less than 150 mV for proper powerfail operation.

FSA Select Logic block contains the logic which controls the timing of the interaction between the BMC and the FSAs. The FSA selection is determined by the four high-order bits in the BLR and the three high-order bits in the AR, both set by the user.

Bubble Signal Decoder block contains the logic for creating all the MBM timing signals. The BMC to bubble memory interface consists of active low timing signals. The starting and stopping point of each signal is determined by the decoder logic. Each signal may occur every field rotation or only once in a number of field rotations. The field rotation in which a timing pulse occurs is controlled by the sequencer logic.

Figure 4 and Table 2 illustrate the typical timing signals for the BMC. These signals are described in the following paragraphs.

$\overline{\text{X+}}$, $\overline{\text{X-}}$, $\overline{\text{Y+}}$, and $\overline{\text{Y-}}$ go to the 7250 CPDs, and are used to enable the coil drive currents in the MBMs.

$\overline{\text{TM.A}}$ and $\overline{\text{TM.B}}$ go to the 7234 CPGs, and are used to determine, respectively, the pulse widths for the CUT and TRANSFER functions used in replicating and generating the bubbles.

Table 2. 7224 BMC Timing (Degrees)**

Signal		Start	Width
$\overline{\text{X}}\pm$		270°	108°
$\overline{\text{Y+}}$		0°	108°
$\overline{\text{X-}}$		90°	108°
$\overline{\text{Y-}}$		180°	108°
$\overline{\text{TM.A}}$ (LATE)		279°	94.5°
$\overline{\text{TM.A}}$ (EARLY)		99°	94.5°
$\overline{\text{TM.B}}$ (LATE)		279°	90°
$\overline{\text{TM.B}}$ (EARLY)		99°	90°
$\overline{\text{BOOT.EN}}$		261°	126°
$\overline{\text{REP.EN}}$		261°	126°
$\overline{\text{SWAP.EN}}$		180°	153°
$\overline{\text{BOOT.SW.EN}}$		180°	DC*
SHIFTCLK (RD)	LATE	230.5°	45°
	EARLY	40.5°	63°
SHIFTCLK (WR)	LATE	261°	126°
	EARLY	81°	126°

*Stays low for 8211 field rotation periods when writing the MBM Bootloop.

**All phases relative to Y+ start phase. All entries ± 1.26 except $\overline{\text{TM.A}}$ width which is ± 0.5.

$\overline{\text{SWAP.EN}}$, $\overline{\text{REP.EN}}$, $\overline{\text{BOOT.SW.EN}}$, and $\overline{\text{BOOT.EN}}$ all go to the 7234 CPG. They are used to enable, respectively, the data swap, data replicate, boot swap, and boot replicate functions within the MBMs.

$\overline{\text{SHIFT.CLK}}$ goes to the FSAs. It is used to control the timing of events at the interface between each FSA and its corresponding MBM. (Refer to 7244 FSA Specification for a description of the BMC/FSA interface.)

$\overline{\text{SYNC}}$ and C/$\overline{\text{D}}$ control the serial communications between the BMC and the FSAs (on the DIO line).

USER-ACCESSIBLE REGISTERS

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

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 A_0) and data bus lines D_0 through D_5.

The Command Register (CMDR) is an 8-bit register which is loaded from D_0 — D_7. Register Address Counter (RAC) is a 4-bit register which is loaded from D_0 — D_3. The status register is selected and read by a single read request. The command register is selected and loaded by a single write request. The remaining registers are accessed indirectly, and 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.

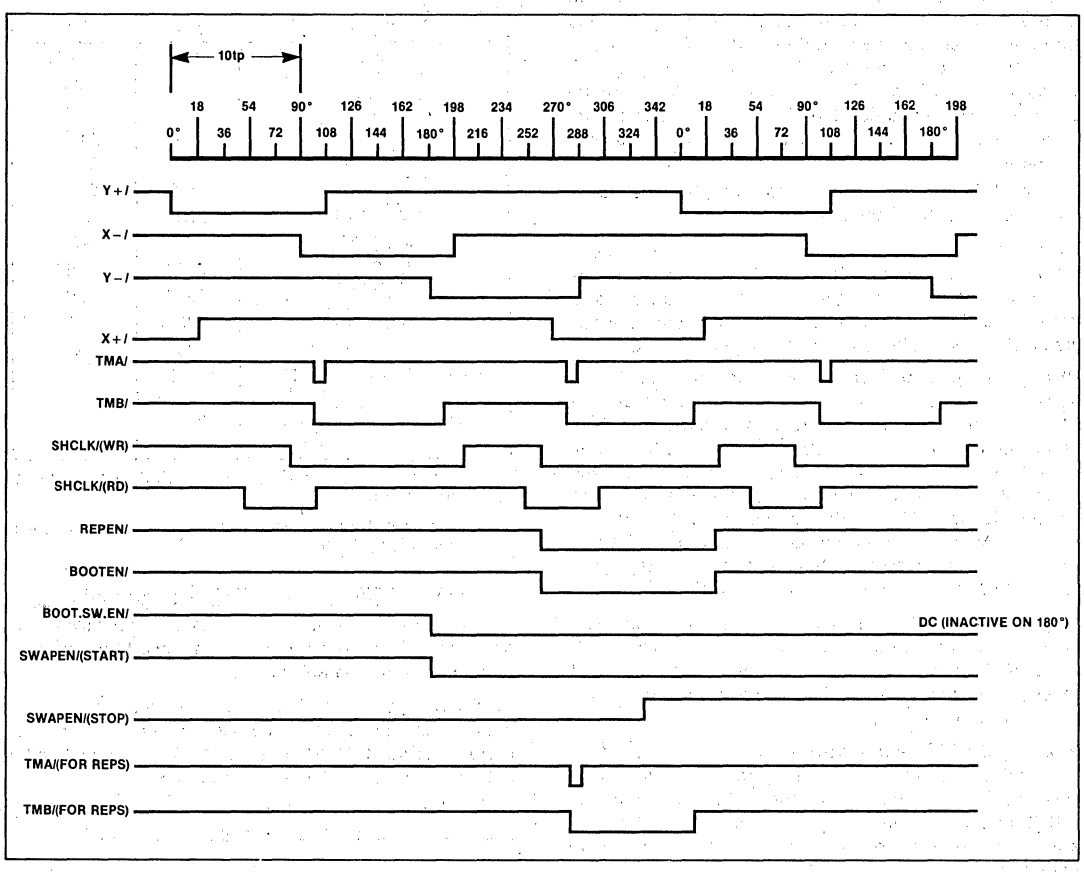


Figure 4. 7224 BMC Timing Diagram

Table 3 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 addressed indirectly by the contents of RAC.

Table 3. 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 C 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

Table 3. Address Assignments for the User-Accessible Registers (Continued)

RAC A0	B3	B2	B1	B0	Symbol	Name of Register	Read/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	Read or Write
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
CCCCC = 5-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 (When written high will clear any pending interrupt from 7224 without destroying any data present in the FIFO and its associated latches.

The register file contains the registers with address 1011 through 1111. These registers are also called parametric registers because they contain flags and parameters that determine exactly how the BMC will respond to commands written to the CMDR.

To facilitate such operations, the BMC automatically increments the RAC by one count after each transfer of data to or from a parametric register.

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.

REGISTER DESCRIPTIONS

Command Register (CMDR) 4 Bits, Write Only

The user issues a command to the BMC by writing a 5-bit command code to the CMDR.

Table 4 lists the 5-bit command codes used to issue the eighteen commands recognized by the BMC.

Table 7 is a listing of the commands and their functions.

Table 4. Command Code Definitions

D5	D3	D2	D1	Do	Command Name
1	0	0	0	0	Write Bootloop Register Masked
0	0	0	0	1	Initialize
0	0	0	1	0	Read Bubble Data
0	0	0	1	1	Write Bubble Data
0	0	1	0	0	Read Seek
0	0	1	0	1	Read Bootloop Register
0	0	1	1	0	Write Bootloop Register
0	0	1	1	1	Write Bootloop
0	1	0	0	0	Read FSA Status
0	1	0	0	1	Abort
0	1	0	1	0	Write Seek
0	1	0	1	1	Read Bootloop
0	1	1	0	0	Read Corrected Data
0	1	1	0	1	Reset FIFO
0	1	1	1	0	MBM Purge
0	1	1	1	1	Software Reset
1	0	0	0	1	Zero Access Read Seek
1	0	0	1	0	Zero Access Read Bubble Data

The most commonly used commands in normal operation are:

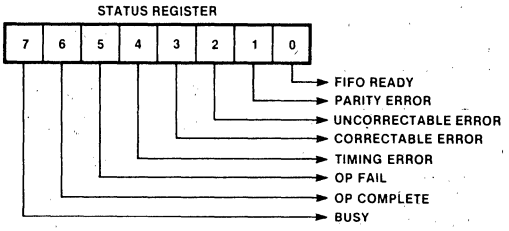
Initialize
Read Bubble Data
Write Bubble Data
Reset FIFO
Read Seek
Write Seek
Abort
Read Corrected Data
Software Reset
Read FSA Status
MBM Purge
Zero Access Read Seek
Zero Access Read Bubble Data

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

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 = 1) indicates that the BMC is in the process of executing a command. When equal to 0, BUSY indicates that 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 BUSY bit has gone inactive with either the TIMING ERROR or UNCORRECTABLE ERROR bits active.

TIMING ERROR (when = 1) indicates that a 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 underflow. TIMING ERROR is also set if no bootloop sync word is found during initialization, or if a Write Bootloop command is issued when the WRITE BOOTLOOP ENABLE bit is equal to zero in the enable register, or the Write Bootloop Register Masked command is sent without an adequate number of 1's present in data pattern.

CORRECTABLE ERROR (when = 1) indicates that a 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 at least one 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 D_0-D_8.

FIFO READY has two functions. The FIFO READY functions are as follows:

NOTE: IF RAC ≠ FIFO, FIFO READY = 1

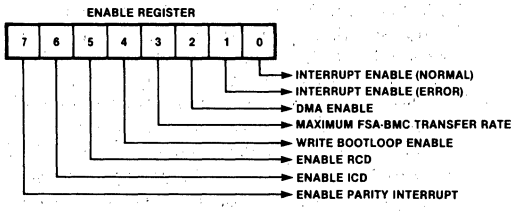
STATUS BITS		READ	WRITE
FIFO READY	BUSY		
1	1	data in FIFO	space in FIFO
0	1	no data	no space
1	0	— data in FIFO —	
0	0	— FIFO empty —	

Although the status word can be read at any time, the status information, bit 1 through 6, is not valid until the BUSY bit is low.

STR Bits 1 through 6 are reset when a new command is issued. They may also be reset by making a write request (WR=0) to the BMC with A_0=1, D_4=0, and D_5=1 (Modified bit) (that is, writing the RAC with D_5 = 1). This operation also resets the "INT" pin to "0". NOTE: A byte of FIFO data can be lost when using this procedure if the RAC is written to other than the FIFO address when data is still present in FIFO.

Enable Register (ER) 8 Bits, Write Only

The user sets various bits of the enable register to 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 D_0-D_7.

ENABLE ICD enables the BMC to give the Internally Correct Data command to the FSAs when an error has been detected by the FSA's error detection and correction circuitry. Each 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).

ENABLE RCD enables the BMC to give the Read Corrected Data command to the FSAs when an error has been detected. This causes each FSA to correct the error (if possible) and also to transfer the corrected data to the BMC. The Read Corrected Data command is also used to read into the BMC data previously corrected by the FSA in response to an Internally Correct Data command. In either case, when the data transfer has been completed, the BMC reads each FSA's status to determine whether or not the error was correctable. In the case of an uncorrectable error, bad data may have been sent to the user. The value of ENABLE RCD affects the action of INTERRUPT ENABLE (ERROR).

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.

DMA ENABLE (when = 1) enables the BMC to operate in DMA data transfer mode, using the DRQ and $\overline{\text{DACK}}$ 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.

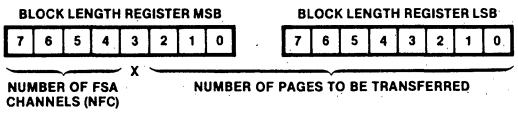
Enable ICD	Enable RCD	Interrupt Enable (ERROR)	Interrupt Action
0	0	1	Interrupt on TE only
0	1	0	Interrupt on UCE or TE
0	1	1	Interrupt on UCE, CE, or TE
1	0	0	Interrupt on UCE or TE
1	0	1	Interrupt on UCE, CE, or TE
1	1	0	Not used
1	1	1	Not used

TE = Timing Error, CE = Correctable Error,
UCE = Uncorrectable Error.

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).

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:

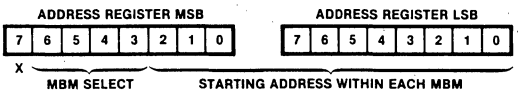


The system page size is proportional to the number of magnetic bubble memory modules (MBMs) operating in parallel during the data read or write operation. Each MBM requires two FSA channels. Bits 4 through 7 of BLR MSB actually specify the number of FSA channels to be accessed.

The BLR LSB, together with the 3 least significant bits of the BLR MSB, specify 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.

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 8192 possible starting address locations for a data read or write operation, hence the requirement for 13 bits in the starting address.

The selection of the MBMs to be read or written is specified by AR MSB Bits 57. The BMCs interpretation of these bits depends on the number of MBMs in a group, which is specified by BLR MSB Bits 4-7.

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

Table 6. Selection of FSA Channels

AR MSB Bits (7,6,5)	BLR MSB Bits (7,6,5,4)				
	0000	0001	0010	0100	1000
000	0	0,1	0,1,2,3	0 to 7	0 to 7
001	1	2,3	4,5,6,7		
010	2	4,5			
011	3	6,7			
100	4				
101	5				
110	6				
111	7				

Table 5. 4Mbyte System Page Size, Page Address Range, and Data Transfer Performance Configuration

BLR MSB 7 6 5 4	NFC	System Page Size	# of Pages	Address Range	MBM Data Transfer Rate
0 0 0 1	2	64 byte	64K	0000 — FFFF	25K bytes/sec
0 0 1 0	4	128 byte	32K	0000 — 7FFF	50K bytes/sec
0 1 0 0	8	256 byte	16K	0000 — 3FFF	100K bytes/sec
1 0 0 0	16	512 byte	8K	0000 — 1FFF	200K bytes/sec

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 FIFO allows the data transfer to proceed in an 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. 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.

FUNCTIONAL OPERATION

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.

Each four Megabit Bubble Memory (MBM) operates in its own domain, and is unaffected by the number of bubble memories in the system. The roles played by the MBM's immediate support circuitry can be described as if the system contained only one MBM module.

Data Flow Within the Magnetic Bubble Memory (MBM) System (Single MBM Systems)

During a read operation, data flows as follows: The data from the MBM is input to 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 data is paired up with the corresponding bit in the FSA's bootloop 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 is applied to each block of 256 data bits.

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 $\overline{\text{SYNC}}$ pulse to the $\overline{\text{SELECT.IN}}$ 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 $\overline{\text{SYNC}}$ pulses, once every 20 or 80 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. From this FIFO, the data bytes are written onto the user interface.

During a write operation, the data flow consists of the corresponding operations in the reverse order.

INTERFACING REQUIREMENTS

All communications between the host microprocessor, and the bubble memory is performed through the 7224 BMC. Below the general principles are described, for detailed guidelines please refer to the BPK 75 Manual.

First the hardware interfacing requirements and second the software interfacing requirements are described.

Table 7. Detailed Command Descriptions

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. All the parametric registers must be properly set up before issuing the Initialize command.
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 and the starting address for the read operation is 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 and the starting address for the write operation is 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 before issuing the Write Bubble Data command.
Read Seek	The Read Seek command rotates the selected MBMs to a designated page address location. 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.

Table 7. Detailed Command Descriptions (Continued)

Write Seek	The Write Seek command rotates the selected MBMs to a designated page address location. 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.
Abort	The Abort command causes a controlled termination of the command currently being executed by the BMC. The Abort command will be accepted by the BMC (and is typically issued) when the BMC is busy.
MBM Purge	The MBM Purge command clears all BMC registers, counters, and the MBM address RAM. Furthermore, it determines how many FSA channels are present in the system and stores this value in the 7224. The "INITIALIZE" command uses this command as a subroutine.
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 an error has been detected. The data cycles through the error correction network of the FSA. After the data has been read, 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).
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 selected FSAs in the system. No reinitialization 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 FSAs, and to store this information in the BMC FIFO. The Read FSA Status command is independent all parametric registers.
Read Bootloop Register	The Read Bootloop Register command causes the BMC to read the bootloop register of the selected FSA channels and to store this information in the BMC FIFO. Twenty bytes are transferred for each FSA channel selected.
Write Bootloop Register Masked	Proper operation of the FSAs during data transfer to or from the MBMs requires that the bootloop register contain exactly 270 logic 1s for each FSA bootloop register. The user may select any subset of 270 "good" loops from the total number of available loops. As an alternative, the Write Bootloop Register Masked command may be used. This command counts the number of logic 1s and masks out the remaining 1s after the proper count has been reached. The Initialize command uses this command as a subroutine.
Read Bootloop	The Read Bootloop command causes the BMC to read the bootloop from the selected MBM, and to store the decoded bootloop information in the BMC FIFO. The Initialize command uses this command as a subroutine.
Write Bootloop	The Write Bootloop command causes the existing contents of the selected MBM's bootloop to be replaced by new bootloop data based on 40 bytes of information stored in the FIFO (the user must actually write 41 bytes, where the 41st byte is all 0s). Encoding of the bootloop data is done by the BMC hardware.
Zero Access Read Bubble Data	The Zero Access Read Bubble Data command functions exactly the same as the Read Bubble Data command except it must be preceeded by the Zero Access Read Seek command. The parametric registers are written prior to the Zero Access Read Seek command and should not be rewritten for the Zero Access Read Bubble Data command.
Zero Access Read Seek	The Zero Access Read Seek command operates similiarly to the Read Seek command, but it reads the first page of data from the MBM into the FSA(s) FIFO. This eliminates the first page overhead involved in the Read Bubble Data command. The latency for the first page data is only the time required to read the data from the 7244(s). The values written into the parametric registers prior to the issuance of the Zero Access Read Seek command are identical to those written for a Read Bubble Data command. The seeking address for the Zero Access Read Seek command is equal to the desired read address. The Zero Access Read Seek command increments the Ad and decrements the BLR. Since the Zero Access Read Bubble Data command expects this increment and decrement, it is used for this data transfer instead of the Read Bubble Data command.

HARDWARE INTERFACE REQUIREMENTS

User Interface Signals

The source, destination and function of the user interface signals are described in Table 1 in the data sheet.

System Timing

As shown on the timing diagrams in the WAVEFORM section the typical read/write cycle timing provides sufficient tolerance to allow most currently available microprocessors to be easily adapted to the BMC timing requirements.

User Data Transfer Rate Requirements

The maximum data rate for the user interface is a function of the number of MBMs operated in parallel as outlined in table 8. The rates listed must be considered in relation to the data transfer mode (polled, interrupt-driven, or DMA) to be implemented in order to be sure that the host system software and hardware are capable of keeping up with the data transfer. In other words, the BMC requires the host CPU to be able to sustain the maximum data rate transfer rate for the minimum data transfer (e.g., for a one bubble system keep up the transfer rate for at least 64 bytes = one page).

Table 8. User Data User Transfer Rate Requirements

Number of MBMs Operating in Parallel	Maximum Data Transfer Rate Between BMC FIFO and the FSAs during Write Bubble Data Commands
1	12.5 kbytes/second
2	25 kbytes/second
4	50 kbytes/second
8	100 kbytes/second

Hardware Interfacing for Data Transfer

The BMC supports three data transfer modes, i.e. DMA, interrupt-driven and polled.

To support DMA, a hardware mechanism is required for servicing the BMC's data transfer requests. While several hardware implementations are possible, one common configuration is the Intel 8257 DMA controller.

To support an interrupt-driven system an Intel 8259 Programmable Interrupt Controller is often used.

The polled data transfer mode relies almost exclusively on the software interaction between the host processor and the BMC to control the transfer of data.

Multiple MBM-System

A BMC is capable of processing data and of supplying the required timing and control signals for operating up to eight Bubble Storage Units (BSUs), each of which is capable of storing 512 kbytes of user data. A BSU consists of a 512 kbyte MBM and its five immediate IC support chips (i.e. a BPK 74 Kit).

SOFTWARE INTERFACE REQUIREMENTS

To use the BMC, the user has to write a "bubble memory software driver".

The bubble driver is responsible for all the system interaction with the bubble memory controller and is intrinsic to the efficient and reliable operation of the bubble system. The driver accepts bubble memory commands and command execution parameters from the application program, controls and monitors command execution, and returns operational status information to the application program at command completion. To perform all of these operations, the bubble driver must support the bit/byte level of the bubble memory controller's command and status registers and the parametric registers that define the operating mode, system configuration, and extent of the transfer.

The level of the software driver complexity is a function of the specific application needs. Regardless, a set of basic drivers must be developed that in turn are integrated into a system at the appropriate level. If an application program is small and simple, a basic bubble driver may simply be called from the main program.

At the highest level of driver sophistication, the application program treats the bubble system as a collection of named data areas of files similar to the way in which data is stored and retrieved in disk operating systems. At the file system level, an application program can ignore the mechanics of bubble storage and access and merely present a file name to the driver to open, read, or write, then close the desired bubble file.

Data Organization

From a software viewpoint, data logically is organized into blocks of bytes called pages. During data transfer operations, one or more of these pages are transferred between the bubble(s) and the host microprocessor. A page is the smallest increment of data that can be transferred; single bytes cannot be transferred. Conceptually, the data organization within a bubble memory is analogous to a disk system. Just as disk sector sizes are fixed when a disk is formatted, bubble page sizes are established, under software control when the bubble system is initialized.

For single bubble system, the page size is fixed at 64 bytes with error correction. The total number of pages available is 8092. In systems with multiple bubbles, page size can vary from 64 bytes to 512 bytes depending on the number of bubble devices in the system. Page size is directly proportional to the system data rate and also determines the total number of available pages (address field size). The selection of the appropriate page size depends primarily on the data rate supported by the system. The higher the data rate, the faster the microprocessor must respond to the demands of the bubble memory controller.

Buffering

The bubble memory controller includes a FIFO data buffer that, although only 40 bytes long, reconciles timing differences between the parallel data transfer to or from the host microprocessor and the serial data transfer to or from the Bubble Memory Subsystem. Accordingly, when an application program requests data from a bubble, the software driver is responsible for keeping up with the FIFO for the duration of the data transfer in order to prevent the FIFO from overflowing or underflowing.

Command Execution

Command execution can be performed either in an interrupt driven mode or in a polled mode irrespective of the data transfer mode (polled, interrupt-driven, or DMA).

Data Transfer Mode

As described earlier in the hardware section, three data transfer modes are available (polled, interrupt-driven or DMA).

System performance, additional hardware and software overhead are all important considerations when choosing the appropriate mode for your application.

Error Correction

The bubble memory has a built-in error detection. Three levels of error correction are available.

Communication with the BMC

All communications between the host and the bubble memory actually are performed through the BMC. The BMC has two input/output (I/O) ports, an eight-bit bidirectional data port, and an eight-bit command/status port. Conceptually, a bubble memory system can be thought of as a disk system in that data in the bubble memory is organized into blocks called pages in bubble technology that are similar to disk sectors. Information such as starting page location, direction of transfer, and the number of pages to be transferred is passed to the BMC before the desired read or write operation is initiated.

The general procedure for communicating with the BMC is:

Set-up the BMC for data transfer communication by loading specific parameters in user-accessible registers.

Send the desired command.

Read the status register to determine if command is accepted.

If applicable, transfer (i.e., read or write) data.

Read the status register until BMC is not busy (or under some conditions "INT" pin).

Examine the status register to determine whether the operation was successful.

For all details and exceptions to this general description consult the BPK 75 User's Manual. (Available 1st Quarter 1984).

ABSOLUTE MAXIMUM RATINGS

Temperature under bias ..... −40 to +100°C
Storage Temperature ..... −65°C to +150°C
All Input or Output Voltages and
V_{CC} Supply Voltage ..... −0.5V to 7V

**NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

(T_A = 0 to 70°C, V_{CC} = 5.0V +5%, −10%)

Symbol	Parameter	Min.	Max.	Unit	Test Condition
V_{IL}	Input Low Voltage		0.8	V	
$V_{IH(1)}$	Input High Voltage (all but $\overline{PWR.FAIL}$)	2.0	V_{CC}+0.5V	V	
$V_{IH(2)}$	Input High Voltage ($\overline{PWR.FAIL}$)	2.5	V_{CC}+0.5V	V	
$V_{OL(1)}$	Output Low Voltage (All outputs except $\overline{DET.ON}$, $\overline{BUS.RD}$, $\overline{SHIFT.CLK}$, and $\overline{SYNC}$		.45	V	I_{OL} = 3.2 mA
$V_{OL(2)}$	Output Low Voltage $\overline{DET.ON}$, $\overline{BUS.RD}$, $\overline{SHIFT.CLK}$, $\overline{SYNC}$		.45	V	I_{OL} = 1.6 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = 400μA
I_{IN}	Input Leakage Current		10	μA	$0 \leq V_{IN} \leq V_{CC}$
$\lvert I_{OFL} \rvert$	Output Float Leakage		10	μA	$0.45 \leq V_{OUT} \leq V_{CC}$
I_{CC}	Power Supply Current from V_{CC}		200	mA	

A.C. CHARACTERISTICS

(T_A = 0 to 70°C, V_{CC} = 5.0V + 5%, −10%; C_L = 150pF; unless otherwise noted.)

Symbol	Parameter	Min.	Max.	Unit	Test Condition
t_P	Clock Period	499.5	500.5	ns	
t_ϕ	Clock Phase Width (High Time)	.45 t_P	.55 t_P		
t_R-t_F	Input Signal Rise and Fall Time		30	ns	

FSA INTERFACE TIMINGS (under pin loading)

Symbol	Parameter	Min.	Max.	Unit	Test Condition
t_{CDV}	CLK to DIO Valid Delay		200	ns	Under Pin Loads*
t_{CDF}	CLK to DIO Entering Float		250	ns	Under Pin Loads*
t_{CDE}	CLK to DIO Enabled from Float		200	ns	Under Pin Loads*
t_{CDH}	CLK to DIO Hold Time	0		ns	Under Pin Loads*
t_{CSOL}	CLK to $\overline{SYNC}$ Leading Edge Delay		200	ns	Under Pin Loads*
t_{CSOT}	CLK to $\overline{SYNC}$ Trailing Edge Delay		150	ns	Under Pin Loads*
t_{DC}	DIO Setup Time to Clock	150		ns	Under Pin Loads*
t_{DHC}	DIO Hold Time from Clock	0		ns	Under Pin Loads*
t_{COL}	CLK to Output Leading Edge		200	ns	Under Pin Loads*
t_{COT}	CLK to Output Trailing Edge		200	ns	Under Pin Loads*
t_{EW}	$\overline{ERR.FLG}$ Pulse Width	200		ns	Under Pin Loads*
t_{SCFT}	$\overline{SHIFTCLK}$ to $\overline{Y-}$ Trailing Edge	80	200	ns	Under Pin Loads*

A.C. CHARACTERISTICS (Continued) (T_A = 0 to 70°C, V_{CC} = 5.0 + 5%, −10%; C_L = 150 pF; unless otherwise noted.)

READ CYCLE (HOST INTERFACE)

Symbol	Parameter	Min.	Max.	Unit	Test Condition
t_{AC}	Select Setup to $\overline{RD}\downarrow$	0		ns	
t_{CA}	Select Hold from $\overline{RD}\uparrow$	0		ns	
t_{RR}	$\overline{RD}$ Pulse Width	200		ns	
t_{AD}	Data Delay from Address		200	ns	
t_{RD}	Data Delay from $\overline{RD}\downarrow$		200	ns	
t_{DF}	Output Float Delay	10	100	ns	
t_{DC}	DACK Setup to $\overline{RD}\downarrow$	0		ns	
t_{CD}	DACK Hold from $\overline{RD}\uparrow$	0		ns	
t_{KD}	Data Delay from $\overline{DACK}\downarrow$		200	ns	
t_{CYCR}	"Read" Cycle Time	(DMA Mode) $4t_p$		ns	In non DMA mode. t_{CYCR} Min. = $6t_p$

WRITE CYCLE (HOST INTERFACE)

Symbol	Parameter	Min.	Max.	Unit	Test Condition
t_{AC}	Select Setup to $\overline{WR}\downarrow$	0		ns	
t_{CA}	Select Hold from $\overline{WR}\uparrow$	0		ns	
t_{WW}	$\overline{WR}$ Pulse Width	200		ns	
t_{DW}	Data Setup to $\overline{WR}\uparrow$	200		ns	
t_{WD}	Data Hold from $\overline{WR}\uparrow$	0		ns	
t_{DC}	$\overline{DACK}$ Setup to $\overline{WR}\downarrow$	0		ns	
t_{CD}	$\overline{DACK}$ Hold from $\overline{WR}\uparrow$	0		ns	
t_{CYCW}	"Write" Cycle Time	$4t_p$			
t_{CQ}	Request Hold from $\overline{RD}$ or $\overline{WR}$ (Non-Burst Mode)		200	ns	
t_{DEADW}	Inactive Time between $\overline{WR}\uparrow$ and $\overline{WR}\downarrow$	$4t_p$		ns	
t_{DEADR}	Inactive Time between $\overline{RD}\uparrow$ and $\overline{RD}\downarrow$	150			

7250-7234 INTERFACE TIMINGS

Symbol	Parameter	Min.	Max.	Unit	Test Condition
t_{CBL}	CLK to Bubble Signal Leading Edge		275	ns	Under Pin Loads*
t_{CBT}	CLK to Bubble Signal Trailing Edge		275	ns	Under Pin Loads*

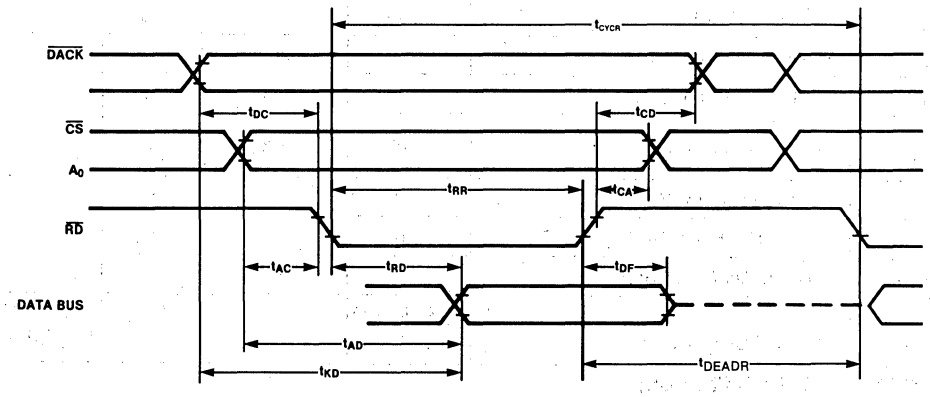
*Bubble Pin Loads Shown Below

PIN LOADINGS

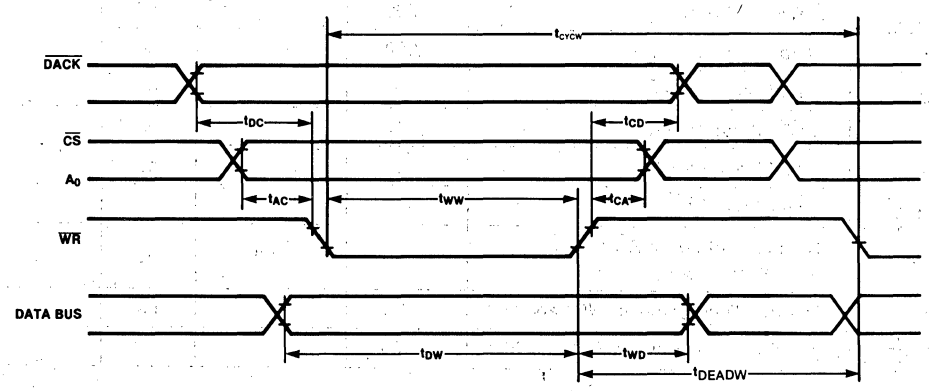
Pin Names	Value	Unit
$\overline{X+}$, $\overline{X-}$, $\overline{Y+}$, $\overline{Y-}$	150	pF
$\overline{TM.A}$, $\overline{TM.B}$, $\overline{REP.EN}$, $\overline{BOOT.EN}$, $\overline{SWAP.EN}$, $\overline{BOOT.SW.EN}$, $\overline{C/D}$, $\overline{ERR.FLG}$, $\overline{WAIT}$, $\overline{SYNC}$, DIO	100	pF
$\overline{DET.ON}$ & $\overline{SHIFT.CLK}$		
$\overline{BUS.READ}$	10	pF

WAVEFORMS

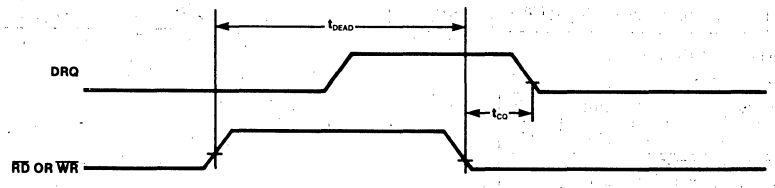
READ (HOST INTERFACE)



WRITE (HOST INTERFACE)

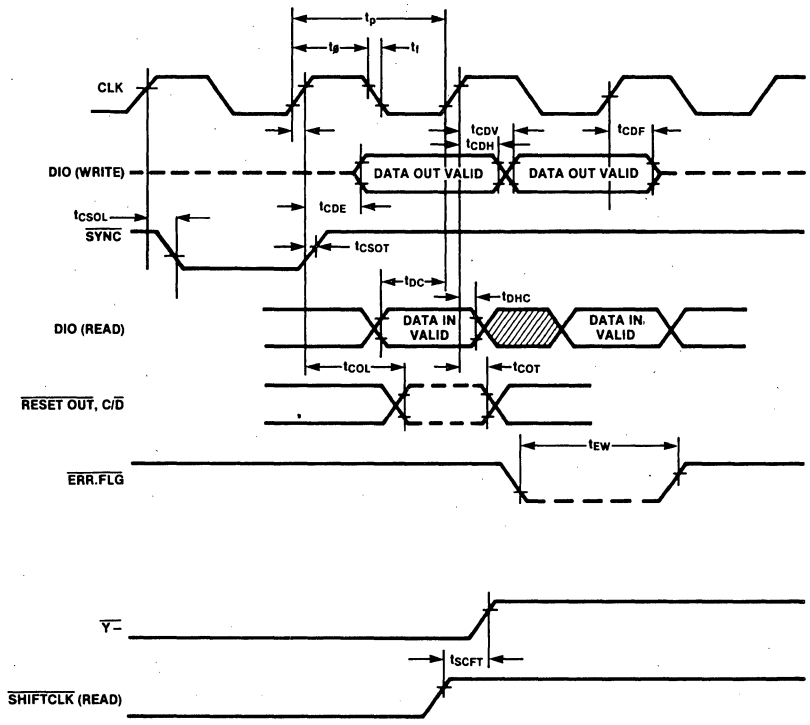


DMA (HOST INTERFACE)



WAVEFORMS (Continued)

7244 INTERFACE TIMINGS



7250 & 7234 INTERFACE TIMINGS