MOTOROLA
Semiconductor Products Inc.

AN-831
Application Note

AN IEEE-488 BUS INTERFACE USING DMA

By
Mike Newman
Manager Technical Marketing
Austin, Texas

INTRODUCTION

This application note provides information about using the MC6809 processor to form a Talker/Listener IEEE-488 System. An overview of a data transfer operation, the General Purpose Interface Bus (GPIB), and some direct memory access techniques are given for review purposes prior to the actual system implementation.

The Talker/Listener device consists of an MC6809 processor, an MC68488 general purpose interface adapter device, and an MC6844 direct memory access controller. Hardware and software considerations are discussed. The listing of an example program is also given.

DATA TRANSFER OVERVIEW

The standard method of transferring data between memory and a peripheral device is to have the transfer controlled by a processor. To perform this transfer, the processor initiates a read instruction which places the data byte in the accumulator of the processor followed by a write instruction completing the transfer. The generalized sequence needed to transfer a data byte between a peripheral device and memory is as follows:

1. The peripheral device alerts the processor when a data byte is to be transferred. The processor recognizes this through either an interrupt sequence or a polling procedure.
2. The processor executes a load instruction to read the data from the peripheral device and loads it into an accumulator, which is used as a temporary holding register.
3. The processor executes a store instruction to write the data from the accumulator into the appropriate memory location.

This sequence shows that at least two software instructions (load and store) are required for each data transfer and that additional software is required to recognize when it is time to transfer each data byte.

In an interrupt driven system, the processor also needs to recognize the interrupt request, complete the current instruction, stack the appropriate internal registers, and enter an interrupt handler routine to determine what course of action is necessary concerning the interrupt.

The MC6809 allows three different types of interrupts, interrupt request ($\overline{IRQ}$), fast interrupt request ($\overline{FIRQ}$), and non-maskable interrupt ($\overline{NMI}$). The entire machine state is saved for $\overline{IRQ}$ and $\overline{NMI}$. This can take up to 20 E clock pulses. The $\overline{FIRQ}$ is a faster responding interrupt in that only the contents of the condition code register and the program counter are saved. This can take up to 12 E clock pulses. If any other internal registers need to be saved when using $\overline{FIRQ}$, they need to be saved via software.

actions to be taken within the devices. The word controller does not refer to a processor on the instrument side of the GPIB.

The controller alters activity on the bus by sending interface messages. The active controller is the only device capable of sending interface messages. It does this in one of two ways:

1. Uniline Messages — The controller can send a message over any one of the five general interface management lines.
2. Multiline Messages — The controller can send a message over the eight data bus lines. It does this by asserting the attention ($\overline{ATN}$) general interface management line signifying to all devices on the bus that the eight bus lines contain a multiline message rather than data.

These messages are interface commands which do not interact directly with the measurement process of an instrument. They interact only with the interface logic within connected devices. The primary purpose of these messages is to carry out the proper protocol in setting up, maintaining, and terminating an orderly flow of device dependent messages. (Device dependent messages refer to the information being sent by the addressed talker device to the addressed listener devices and not the messages used to control the interface.) The multiline and uniline messages are used to address devices to be talkers or listeners, to tell a device to ignore or not ignore front panel settings, to inquire about any problems the device has, to reset the interface circuitry, to begin making a measurement, etc.

Addresses are assigned to each device so it can respond to addressed commands. Using this address, the controller can pick out a specific device and instruct it to be either a talker or listener. The controller does not assign addresses; this must come from some external means such as a set of switches attached to the device or a subroutine resident in the software controlling the device. The address is placed in the GPIB interface for the device during an initialization sequence. Once resident in the interface circuitry, the device can respond to addressed commands. The address is a 15-bit digital number that allows the controller to talk to a particular device.

A talker sends a data byte over the GPIB to a listener or listeners using an asynchronous three-wire handshake. The transfer begins when the talker asserts data available ($\overline{DAV}$) and is completed when the slowest listener accepts the data byte by asserting data accepted (DAC). The third handshake line, ready for data (RFD), is used to let the talker know that the listeners are ready for data. There are actually four states in a data transfer.

1. The talker generates a new byte.
2. The states of the data bus signal lines settle.
3. The listeners accept the data.
4. The listeners become ready for the next byte.

Since there can be many listeners (maximum of 14; 14 listeners plus one talker for 15 devices maximum), it is possible to have some that respond very quickly (e.g., a disk) and some that respond slowly (e.g., a teletype) to the same data byte. In this case, the overall speed of transmission over the bus is governed by, and cannot exceed the response rate of the slowest active listener.

The following example is given to demonstrate the command structure of the GPIB bus and how this relates to the internal processor system of a device. In this example, a device assigned a GPIB address of 3 is to send a block of data using DMA to a device assigned a GPIB address of 1. One procedure for establishing this link is as follows:

1. Once connected to the system (other devices may also be connected to this system), the power to each device is turned on. The unique GPIB address for each device is placed in its respective general purpose interface adapter(MC68488) during a power-on initialization sequence by the processor along with other appropriate initialization procedures.
2. The GPIB controller takes control of the bus by asserting $\overline{ATN}$ and, with the appropriate interface commands, clears all devices on the bus. Remember that the GPIB controller only talks to the general purpose interface adapter (MC68488) and not directly to the device processor. It is up to the MC68488 to alert the processor through either a polling or an interrupt routine when the processor needs to take action.
3. The GPIB controller makes device 3 a listener and sends it information concerning the upcoming DMA block transfer. The MC68488 interprets these bytes as data and flags the processor on a per byte basis. The processor software interprets these data bytes as device dependent messages. These messages provide information such as the precise data to be sent, the format of the data, mode of processor transfer — DMA or non-DMA, etc.
4. The GPIB controller clears device 3 and makes device 1 a listener. Step 3 is repeated to device 1; however, in this case the information pertains to device 1 as the recipient of the block of data.
5. The GPIB controller leaves device 1 in the listen mode and assigns device 3 to be a talker. The GPIB controller now releases control of the GPIB, by negating $\overline{ATN}$ allowing the data transfer to take place.
6. The talker now sends the data in a byte-per-byte sequence to the listener. Each byte is accepted by the listener according to the asynchronous handshake.
7. When the last byte is sent, the talker alerts both the listeners and the controller that the next byte is the last byte of the data block by asserting the $\overline{EOI}$ general interface management line. The end of a data string can also be indicated by a special sequence of data characters (e.g., carriage return followed by line feed) which are interpreted in software.
8. The GPIB controller can now reconfigure the bus for the next data transfer.

DIRECT MEMORY ACCESS MODES OF OPERATION

The MC6844 (DMAC) is capable of three modes of DMA transfer, they are: three-state cycle steal, halt cycle steal, and halt burst. Only the halt burst and three-state cycle steal modes were considered for this system controller since the MC6809 can handle these modes efficiently. The characteristics of these modes are:

Halt Burst Mode — In this mode, the processor is halted and removed from the bus (the appropriate output lines placed in the high-impedance state) while a block of data is transferred between memory and the GPIB. The DMAC manages the control lines (e.g., R/$\overline{W}$, address lines, etc.) and keeps track of how many bytes have been transferred, returning control to the processor when the last byte has been sent. Therefore, if the DMAC has been programmed for a 16K byte transfer, the processor is removed from the bus at the beginning of the transfer and is not brought back on the bus until all 16K bytes have been transferred. This mode of operation provides the direct memory access system with the highest data transfer rate capability; however, even though the DMAC can operate at this high data transfer rate, the actual transfer rate cannot exceed the rate at which the GPIA can issue request.

The main advantage of the halt burst mode is the high data transfer capabilities. The main disadvantage is that the processor is halted during the entire transfer.

Three-State Cycle Steal — In this mode, the processor is neither halted nor removed from the bus for any extended length of time. Rather, the operations of the processor are temporarily suspended and the processor removed from the bus (the appropriate output lines are placed in the high-impedance state) while the DMAC transfers one byte of data. At the end of this transfer, control is given back to the processor. If a block of data is being transferred, the processor is placed back on the bus between each transfer for at least one processor clock cycle. This method of direct memory access operation is slower than the halt burst mode, but does not cause the processor to relinquish control of the bus for long periods of time.

The MC68488 GPIA cannot issue direct memory access transfer requests at a high enough rate to take advantage of the high data transfer rate capabilities of the halt burst mode. This is due to the inherent functionality of the GPIA and the IEEE-488 bus. The GPIA must acknowledge each data byte on the bus before it can issue the next transfer request. This can take up to seven processor clock cycles. In addition, the data on the GPIB is transferred in an asynchronous fashion and cannot be transferred at a rate faster than it can be accepted by the slowest listening device. In many applications the data rate on the bus can be very slow; and as a result, the transfer requests being issued to the DMAC for the device in question could be occurring at a rate considerably slower than one every seven processor clock cycles. If the halt burst mode were used, the MC6809 would be inactive during the non-DMA time that the DMAC is waiting for a transfer request from the GPIA. To take advantage of the non-DMA time and allow the MC6809 to do processing during this time, the three-state cycle steal mode of operation was chosen. Now the processor can be brought back on the bus to perform tasks in between DMA transfers.

SYSTEM OVERVIEW

The DMA system given in this application is essentially divided into seven major circuits as shown in Figure 2. The following paragraphs provide a brief description of each of these circuits. A description of how these circuits are interconnected as a working system is also provided.

MC6809 MICROPROCESSOR — The MC6809 is an advanced member of the MC6800 microprocessor family. It has special DMA capabilities that allow highly efficient DMA data transfers. During non-DMA conditions, the MC6809 continues to operate the system. The MC6809 initializes the other circuits in the system (e.g., MC6844, MC68488, and the display). At other times, it can be used to execute special purpose programs.

MC6844 DIRECT MEMORY ACCESS CONTROLLER — The MC6844 requests control of the bus from the MC6809 and issues the appropriate commands (via the R/$\overline{W}$ line, grant line, and address lines) to perform data transfers. The direct memory access controller never actually receives the data, it directs the flow of the data from one place to the other at the correct time and in the required direction. After the transfer is complete, the MC6844 returns control to the MC6809.

MC68488 GENERAL PURPOSE INTERFACE ADAPTER — The MC68488 provides the interface between the IEEE-488 bus and a processor controlled system. After initialization, the GPIB system controller places the MC68488 in either a talk mode when it is to send data or in a listen mode if it is to receive data.

SYNCHRONIZATION CIRCUITRY — The synchronization circuitry performs two functions: 1) It synchronizes the DMA request signal from the DMAC with the quadrature (Q) signal from the MC6809 by ensuring that the DMA request is not presented to the MC6809 $\overline{DMA}/\overline{BREQ}$ input during the last quarter cycle of the E signal. 2) The end or identify ($\overline{EOI}$) line on the general purpose interface byte is used by a talker to indicate to the listeners that the next data byte received is the last byte of a block. In this system, this line is applied to the synchronization circuitry to disable DMA transfer requests to the MC6809. The $\overline{EOI}$ input to the synchronization circuitry is used only when DMA transfers are being made from the GPIA to memory.

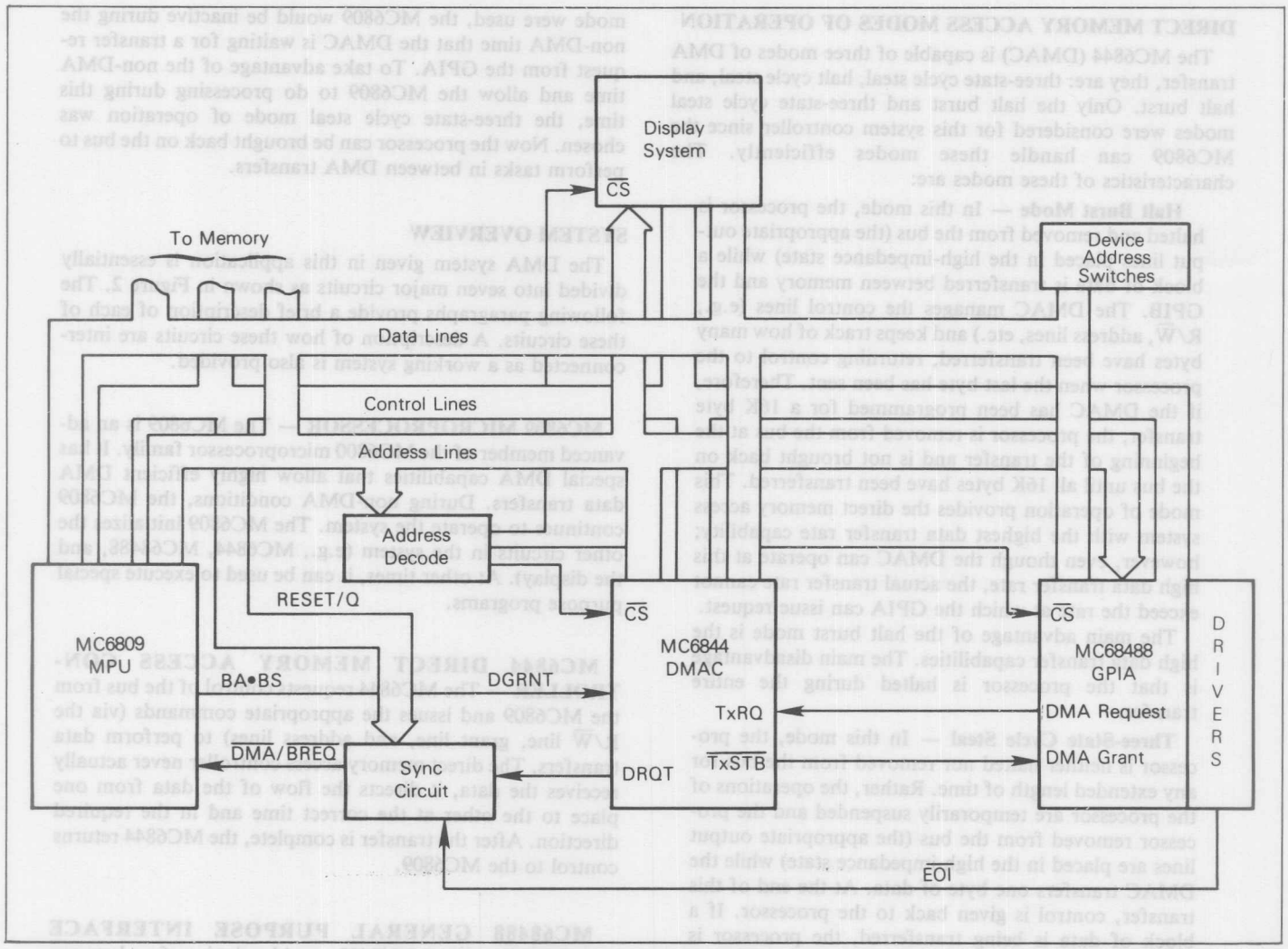

Figure 2. DMA System Block Diagram

DISPLAY SYSTEM — The display system provides a visual indication of: how many blocks of data have been transferred, whether the device is a talker or a listener, and whether the device is in a local or remote state.

DEVICE ADDRESS SWITCHES — This set of toggle switches is isolated from the data bus by buffers. They are used to select the device address for the GPIB, i.e., the address that the GPIB controller uses when sending addressed commands. These switches are manually set to the desired address. The MC6809 initialization program reads the address by enabling the buffers and places it in the MC68488.

OPERATION

This system allows bidirectional data transfers in either a non-DMA mode or a three-state cycle steal DMA mode.

The software is a simplified test program which demonstrates the DMA capability of the system and is not intended as a general purpose application program. The test program only allows data transfers in the DMA mode. After the initialization sequence, the MC6809 simply monitors the GPIA for the direction of data transfer. The DMAC is not initialized during the system initialization sequence. The software initializes the display and GPIA and then enters a monitor loop leaving the DMAC disabled. When the direction of transfer is established, the MC6809 branches to a routine that initializes the DMAC accordingly. For system simplicity, the characteristics of the transfer (e.g., number of bytes to be transferred and beginning memory address) are constants in the DMAC initialization routine. The only variable is direction and this is determined by monitoring the address status register of the GPIA.

The DMAC is not initialized until the direction of transfer has been established by the GPIB controller. The controller does this by sending either my talk address (MTA) or my listen address (MLA). When the GPIA receives either an MTA or MLA, it sets the appropriate talker active state (TACS) or the listener active state (LACS) status bit in the address status register. The MC6809 polls the address status register for status information and initializes the DMAC to transfer data from memory to GPIA if the TACS bit is set and from GPIA to memory if the LACS bit is set.

INITIALIZATION SEQUENCE — A power-on reset places the display system, DMAC, and GPIA in a reset state. During the initialization routine shown in Figure 3, the display system and GPIA are initialized.

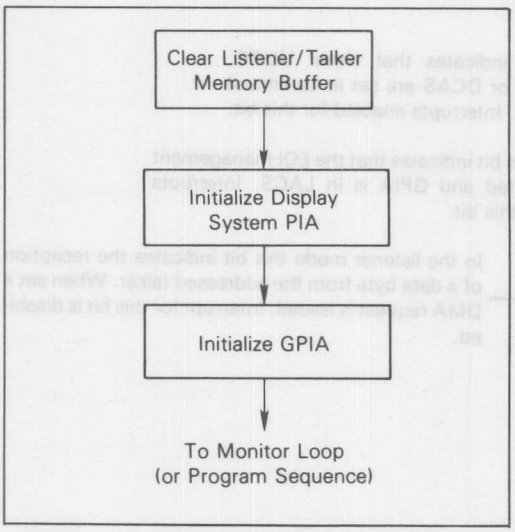

Figure 3. Initialization Routine Flow Chart

The display system has an MC6821 peripheral interface adapter (PIA) which drives two seven-segment displays and three indicator lights. During initialization, the PIA lines that control the seven-segment displays are programmed as outputs and set to zero causing the displays to read a $00. In addition, the lines that control the indicator lights are programmed as outputs and set to zero keeping the indicator light off.

The GPIA is initialized next. The first step is for the MC6809 to read the address selected by the address switches and place this value in the GPIA address register (R4W). This is the value that the GPIB system controller will use to send addressed commands to this device. The next step is to remove the GPIA software reset by writing a $00 to the auxiliary command register (R3W). Until the software reset is removed (bit 7 of R3W written to zero), the only register in the GPIA that can be accessed is the address register. After R3W is written with $00, the MC6809 programs the address mode register (R2W) with a $80. This deselects certain status bits in the interrupt and command status registers from being set. The GPIA ignores any conditions on the GPIB that cause the GET status bit in the interrupt status register to be set and also any conditions that prevent the UACG, UUCG, and DCAS status bits in the command status register from being set. The interrupt mask register is then set up to enable interrupt capability on certain conditions. The interrupt mask register is programmed with $86. This allows interrupts to occur if the END status bit is set or the CMD status bit is set. A summary of interrupt and command status registers is given in Figure 4.

Since bit 7, R2W was set during initialization, the only bits in the command status reigster that can cause the CMD status bit to be set are remote local change (RLC) or serial poll active state (SPAS). The RLC status bit is used to determine the state of the remove local indicator light. The serial poll active state feature is not used in this system, and if this bit gets set and causes an interrupt, the system software goes to a trap routine and displays $E4 on the display.

MONITORING SEQUENCE — After the initialization sequence, the MC6809 software enters the monitor loop shown in Figure 5. The primary purpose of this routine is to set the indicator lights to indicate how the GPIA has been addressed (talk or listen) and initialize DMAC. The first procedure that is executed in the monitor loop is a reset and set of the GPIA interrupt mask register. Since the GPIA interrupt structure is edge sensitive to the setting of its status bits, the reset/set sequence of the interrupt mask register ensures that if a second interrupt bit gets set while a prior one is still set, this second interrupt is not missed. Now the address status register (R2R) of the GPIA is monitored. If the LACS bit is set, the listen status indicator is turned on and the DMAC initialized to transfer data from the GPIA to memory. If the TACS bit is set, the talker indicator light is turned on and the talker memory buffer is loaded with "dummy" values for the example test transfer. The DMAC is now initialized to transfer data from memory to GPIA.

After the direction of DMA transfer is established and the DMA controller initialized, the program enters the wait loop shown in Figure 6. The system enters this loop and waits for a DMA transfer request to be issued by the GPIA. The wait loop is not a necessary part of the system and in many applications can be replaced by the MC6809 performing some task. While in the wait loop, the software checks the address status register for any change in the addressed state. The following conditions result:

1. If there is not a change in address status of the GPIA, no action is taken and the program continually cycles through the wait loop.
2. If the GPIA is unaddressed (e.g., receiving an unlisten or untalk command), the program turns off the DMAC and goes to the monitor loop. This unaddressed condition is detected by monitoring the my address (ma) status bit in the GPIA.
3. If the addressed state changes from talker to listener or from listener to talker during a DMA block transfer, the wait loop branches to a trap routine and $E1 is

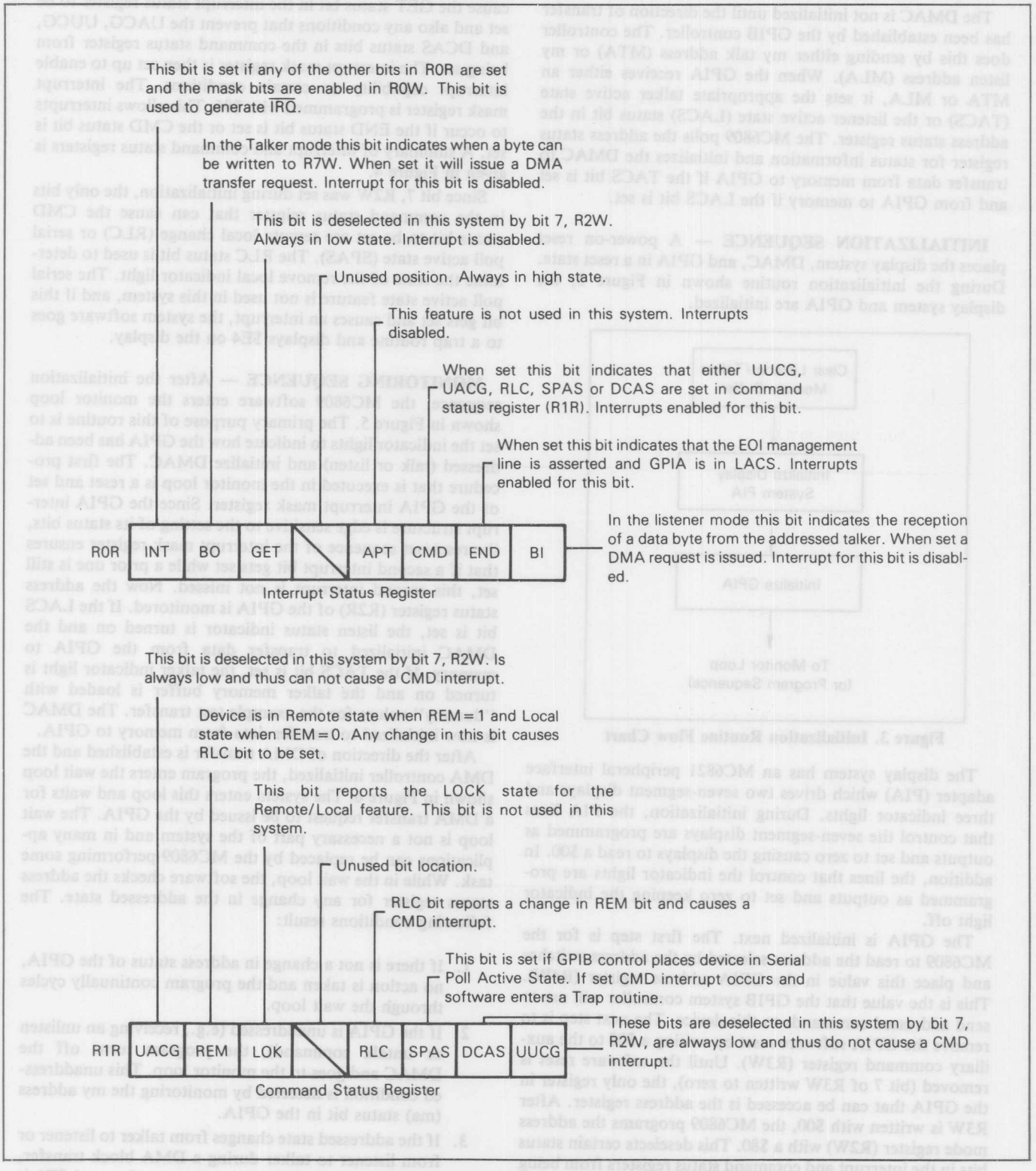

Figure 4. GPIA Interrupt and Command Status Register

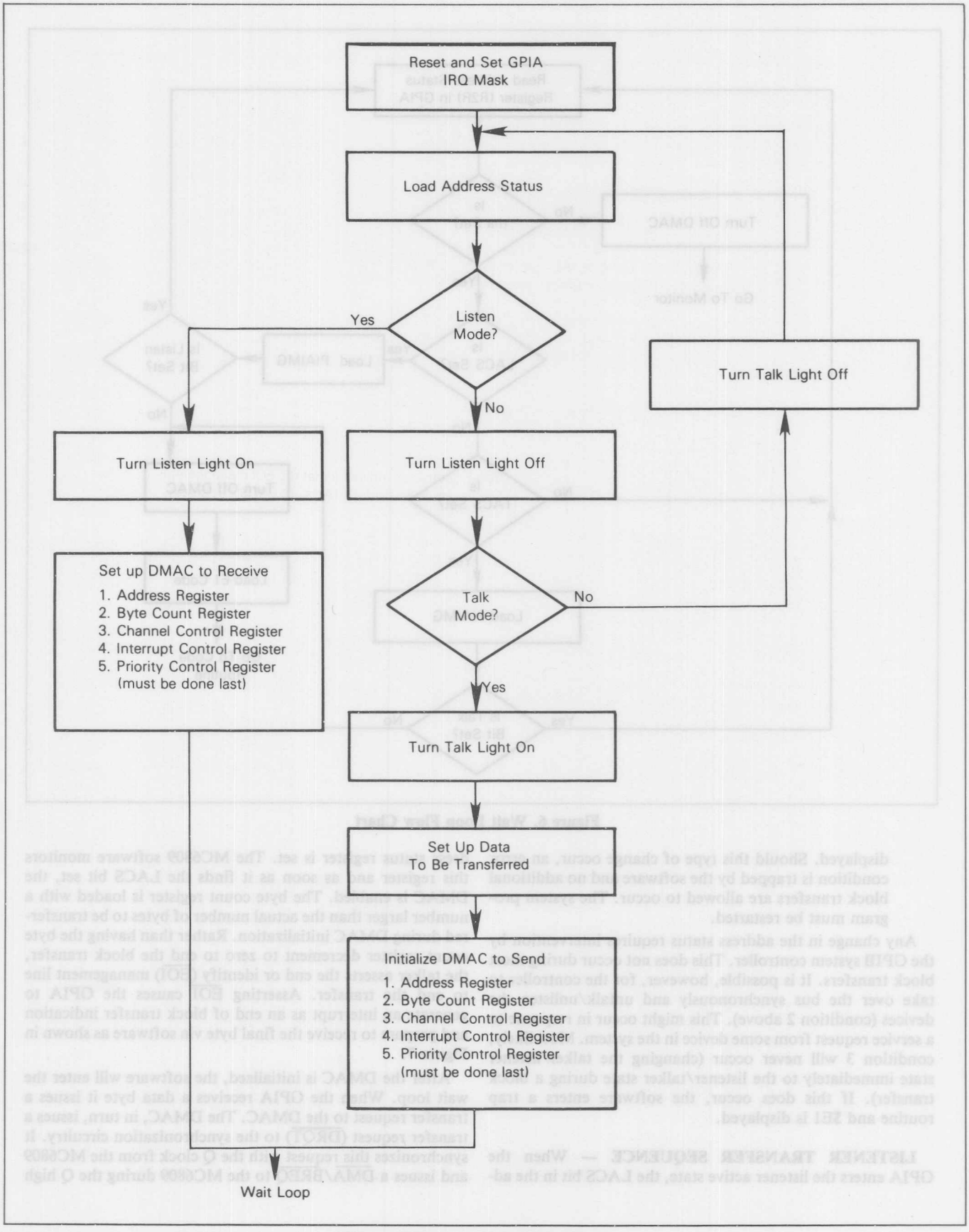

Figure 5. Monitor Loop Flow Chart

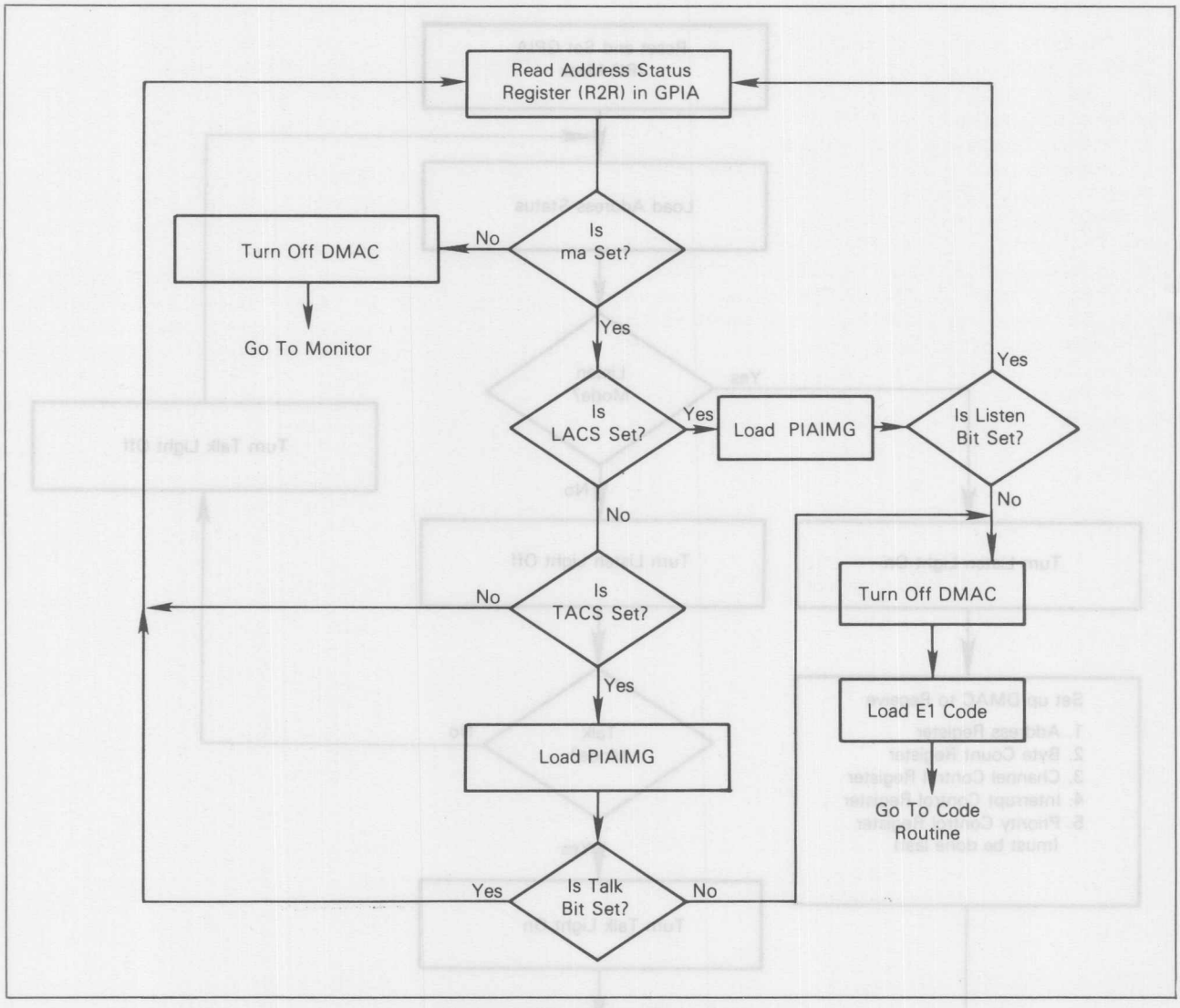

Figure 6. Wait Loop Flow Chart

displayed. Should this type of change occur, an error condition is trapped by the software and no additional block transfers are allowed to occur. The system program must be restarted.

Any change in the address status requires intervention by the GPIB system controller. This does not occur during most block transfers. It is possible, however, for the controller to take over the bus synchronously and untalk/unlisten the devices (condition 2 above). This might occur in response to a service request from some device in the system. Most likely, condition 3 will never occur (changing the talker/listener state immediately to the listener/talker state during a block transfer). If this does occur, the software enters a trap routine and $E1 is displayed.

LISTENER TRANSFER SEQUENCE — When the GPIA enters the listener active state, the LACS bit in the address status register is set. The MC6809 software monitors this register and as soon as it finds the LACS bit set, the DMAC is enabled. The byte count register is loaded with a number larger than the actual number of bytes to be transferred during DMAC initialization. Rather than having the byte count register decrement to zero to end the block transfer, the talker asserts the end or identify ($\overline{EOI}$) management line to end the transfer. Asserting $\overline{EOI}$ causes the GPIA to generate an interrupt as an end of block transfer indication and prepare to receive the final byte via software as shown in Figure 7.

After the DMAC is initialized, the software will enter the wait loop. When the GPIA receives a data byte it issues a transfer request to the DMAC. The DMAC, in turn, issues a transfer request ($\overline{DRQT}$) to the synchronization circuitry. It synchronizes this request with the Q clock from the MC6809 and issues a $\overline{DMA}/\overline{BREQ}$ to the MC6809 during the Q high

time. The low input on the MC6809 $\overline{DMA}/\overline{BREQ}$ pin stops instruction execution at the end of the current cycle (E pulse). The processor address and data lines go to a high-impedance state and the BA and BS output lines go to a 1 to indicate that the present cycle is the dead cycle used to transfer control to the DMAC. The BA and BS outputs are ANDed to become a DMA grant input to the DMAC. Once the DMAC has bus control, it issues a DMA grant to the GPIA. During the E pulse, while DMA grant to the GPIA is high, the data is actually transferred. The GPIA releases the transfer request line to the DMAC. The DMAC releases the $\overline{DMA}/\overline{BREQ}$ input to the MC6809 and, after one dead cycle, the MC6809 removes the high-impedance state from the address and data lines and takes control of the bus. The processor is free to perform other tasks. The transfer uses three E pulses (one pulse for the transfer and one dead cycle before and after the transfer). Each data byte is transferred using this same procedure.

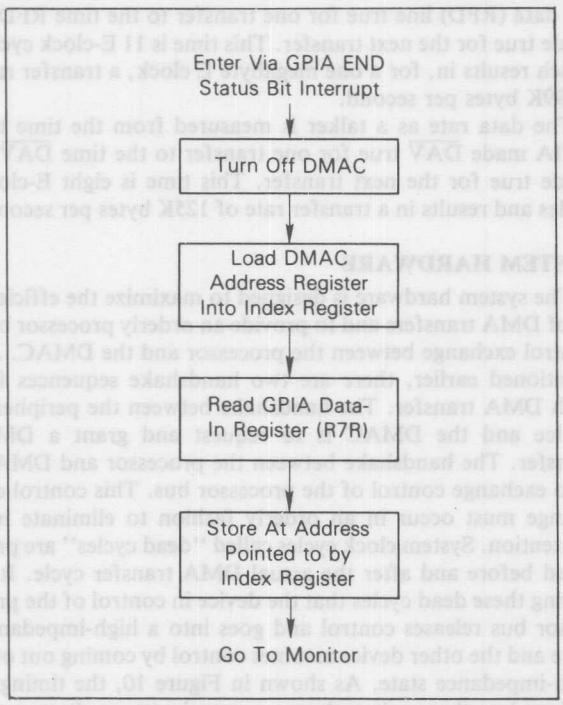

Figure 7. Receive Last Byte Routine Flow Chart

Prior to receiving the last byte of data, the GPIB talker drives the $\overline{EOI}$ line low. The $\overline{EOI}$ line is an input to the synchronization circuitry and, when asserted, prevents a DMA request from the DMAC to the MC6809 from being issued. This ensures that the MC6809 does not release control of the bus to the DMAC for the last byte transfer. In addition, the $\overline{EOI}$ line causes the END status bit in the GPIA to be set which in turn sends an interrupt to the MC6809. When the MC6809 software detects the END status bit set, it branches to a special routine, and the last byte is transferred to memory via processor software. The last byte is transferred by software since the processor must be used to read the status of the MC68488 for the occurrence of an $\overline{EOI}$. The software also disables the DMAC. The software returns to the monitor loop when the last byte is in memory. Reception of this last byte causes the GPIB talker to release the $\overline{EOI}$ line.

TALKER TRANSFER SEQUENCE — The GPIB system controller instructs a device to send data by sending its talk address (MTA). When the MC68488 is made a talker, it moves into the talker active state and the TACS bit in the address status register is set. If set, the MC6809 initializes the DMAC to transfer data from memory to GPIA. The DMAC byte count register is loaded with the number N-1, where N is the number of bytes to be transferred. A DMA transfer is used for N-1 bytes. The last byte (N) is sent to the GPIA via MC6809 software. The last byte is sent this way because just prior to sending the last byte the MC6809 must set the forced end or identify (feoi) bit in the auxiliary command register of the GPIA. This causes the $\overline{EOI}$ management line to go low and alert the listener(s) that the next byte is the last byte of the block. Figure 8 is a flowchart of the send last byte routine.

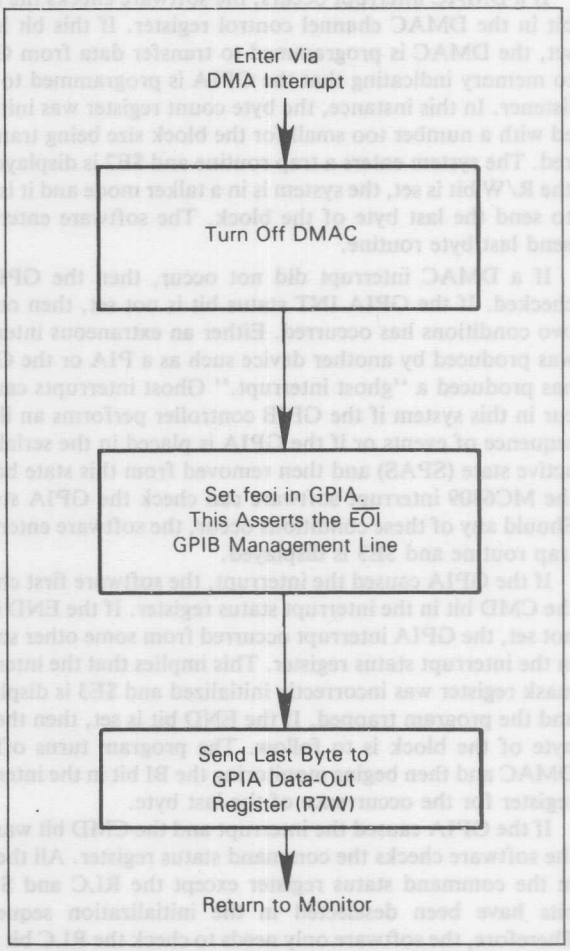

Figure 8. Send Last Byte Routine Flow Chart

As soon as the MC68488 enters the talker active state, a transfer request is issued indicating that the MC68488 is an active talker and the output buffer is empty. Each time the byte written to the GPIA output buffer is accepted by the listener(s) on the bus, another transfer request is issued. The transfer request handshake sequence between the MC68488, MC6844, and MC6809 is the same in the talker mode as it is for the listener mode.

INTERRUPT HANDLING — There are two sources of interrupts, the DMAC and the GPIA. When an interrupt occurs, the software checks to see if the DMAC caused the interrupt, as shown in Figure 9. The DMAC only generates an interrupt when the byte count register decrements to 0. Recall that, in the listener mode, the byte count register is programmed with a hex number larger than the number of bytes to be transferred. In the talker mode, the byte count register is programmed with N-1, where N is the number of bytes to be transferred. Therefore, the only time the DMAC can generate an interrupt in this system is when the GPIA is in the talker mode and is ready to transfer the last data byte from memory to GPIA.

If a DMAC interrupt occurs, the software checks the R/W bit in the DMAC channel control register. If this bit is not set, the DMAC is programmed to transfer data from GPIA to memory indicating that the GPIA is programmed to be a listener. In this instance, the byte count register was initialized with a number too small for the block size being transferred. The system enters a trap routine and $E2 is displayed. If the R/W bit is set, the system is in a talker mode and it is time to send the last byte of the block. The software enters the send last byte routine.

If a DMAC interrupt did not occur, then the GPIA is checked. If the GPIA INT status bit is not set, then one of two conditions has occurred. Either an extraneous interrupt was produced by another device such as a PIA or the GPIA has produced a "ghost interrupt." Ghost interrupts can occur in this system if the GPIB controller performs an illegal sequence of events or if the GPIA is placed in the serial poll active state (SPAS) and then removed from this state before the MC6809 interrupt software can check the GPIA status. Should any of these conditions occur, the software enters the trap routine and $E3 is displayed.

If the GPIA caused the interrupt, the software first checks the CMD bit in the interrupt status register. If the END bit is not set, the GPIA interrupt occurred from some other source in the interrupt status register. This implies that the interrupt mask register was incorrectly initialized and $E3 is displayed and the program trapped. If the END bit is set, then the last byte of the block is to follow. The program turns off the DMAC and then begins monitoring the BI bit in the interrupt register for the occurrence of the last byte.

If the GPIA caused the interrupt and the CMD bit was set, the software checks the command status register. All the bits in the command status register except the RLC and SPAS bits have been deselected in the initialization sequence. Therefore, the software only needs to check the RLC bit and, if it is not set, can assume that the interrupt was caused by SPAS. Since the SPAS feature of the GPIB is not used in this system, this occurrence causes the software to enter a trap routine. If the RLC bit was set, then the software checks the REM bit to see if the device is in local or remote and operates the remote/local indicator light accordingly.

DATA RATE — The data rate in this type of system is a function of the response of the device being communicated with. During the testing of this operation, a Hewlett Packard GPIB Emulator which has a TTL response rate was used (negligible when compared with the 6809/6844/68488 system). Because of this, the data rates for the system in this application are primarily a function of the 6809/6844/68488 system and any increase from combining the response rates for devices on both sides of the communications link can be considered negligible. The data rate differs slightly depending on whether the GPIA is a talker or a listener. This time difference is a result of the GPIA itself. The data rate as a listener is measured from the time the GPIA made the ready for data (RFD) line true for one transfer to the time RFD is made true for the next transfer. This time is 11 E-clock cycles which results in, for a one megabyte E clock, a transfer rate of 99K bytes per second.

The data rate as a talker is measured from the time the GPIA made $\overline{DAV}$ true for one transfer to the time $\overline{DAV}$ is made true for the next transfer. This time is eight E-clock cycles and results in a transfer rate of 125K bytes per second.

SYSTEM HARDWARE

The system hardware is designed to maximize the efficiency of DMA transfers and to provide an orderly processor bus control exchange between the processor and the DMAC. As mentioned earlier, there are two handshake sequences for each DMA transfer. The handshake between the peripheral device and the DMAC is to request and grant a DMA transfer. The handshake between the processor and DMAC is to exchange control of the processor bus. This control exchange must occur in an orderly fashion to eliminate bus contention. System clock cycles called "dead cycles" are provided before and after the actual DMA transfer cycle. It is during these dead cycles that the device in control of the processor bus releases control and goes into a high-impedance state and the other device assumes control by coming out of a high-impedance state. As shown in Figure 10, the timing is designed so that each exchange occurs in one cycle to maximize system efficiency and yet prevent both devices from trying to be in control of the processor bus at the same time. There is a time during each dead cycle where both the processor and DMAC are off the bus and the processor bus and control lines are in the high-impedance state. To prevent a spurious write or read during this time, a signal called DMAVMA is generated which disables the chip select of all peripheral devices.

To ensure that the entire post dead cycle has a DMAVMA, a signal called first quarter (FQ) is used to provide DMAVMA for the first quarter of every MC6809 E clock period. Since the first quarter is not used by peripheral devices, this operation does not pose any system problems.

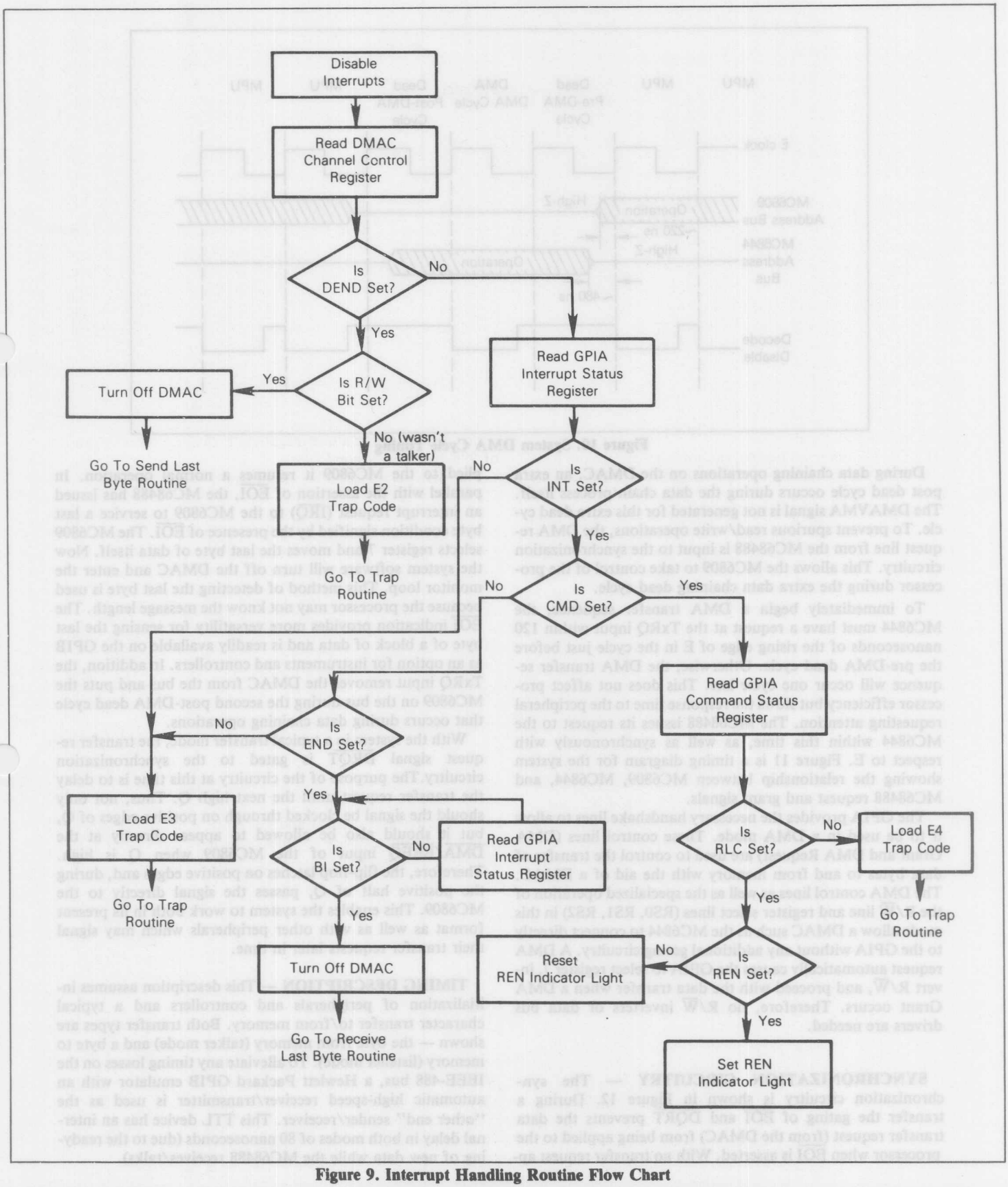

Figure 9. Interrupt Handling Routine Flow Chart

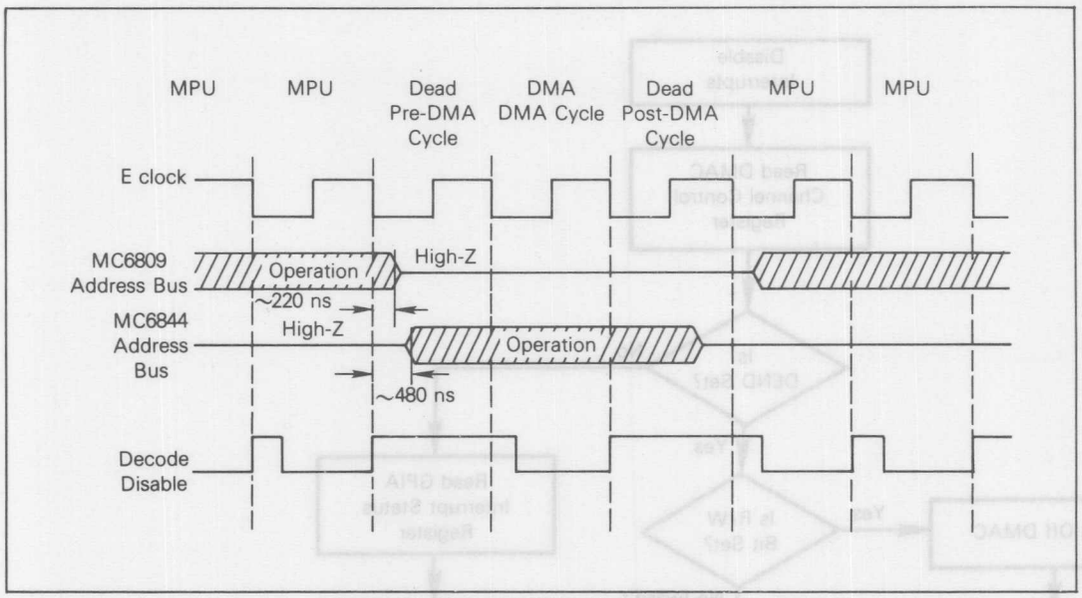

Figure 10. System DMA Cycle Timing

During data chaining operations on the DMAC, an extra post dead cycle occurs during the data chain process itself. The DMAVMA signal is not generated for this extra dead cycle. To prevent spurious read/write operations, the DMA request line from the MC68488 is input to the synchronization circuitry. This allows the MC6809 to take control of the processor during the extra data chaining dead cycle.

To immediately begin a DMA transfer sequence, the MC6844 must have a request at the TxRQ input within 120 nanoseconds of the rising edge of E in the cycle just before the pre-DMA dead cycle. Otherwise, the DMA transfer sequence will occur one cycle late. This does not affect processor efficiency but slows the response time to the peripheral requesting attention. The MC68488 issues its request to the MC6844 within this time, as well as synchronously with respect to E. Figure 11 is a timing diagram for the system showing the relationship between MC6809, MC6844, and MC68488 request and grant signals.

The GPIA provides the necessary handshake lines to allow it to be used in a DMA mode. These control lines (DMA Grant and DMA Request) are used to control the transfer of data bytes to and from memory with the aid of a DMAC. The DMA control lines as well as the specialized operation of the R/$\overline{W}$ line and register select lines (RS0, RS1, RS2) in this mode allow a DMAC such as the MC6844 to connect directly to the GPIA without any additional gating circuitry. A DMA request automatically causes the GPIA to select register 7, invert R/$\overline{W}$, and proceed with the data transfer when a DMA Grant occurs. Therefore, no R/$\overline{W}$ inverters or data bus drivers are needed.

SYNCHRONIZATION CIRCUITRY — The synchronization circuitry is shown in Figure 12. During a transfer the gating of $\overline{EOI}$ and $\overline{DQRT}$ prevents the data transfer request (from the DMAC) from being applied to the processor when $\overline{EOI}$ is asserted. With no transfer request applied to the MC6809 it resumes a normal operation. In parallel with the assertion of $\overline{EOI}$, the MC68488 has issued an interrupt request ($\overline{IRQ}$) to the MC6809 to service a last byte condition signified by the presence of $\overline{EOI}$. The MC6809 selects register 7 and moves the last byte of data itself. Now the system software will turn off the DMAC and enter the monitor loop. This method of detecting the last byte is used because the processor may not know the message length. The $\overline{EOI}$ indication provides more versatility for sensing the last byte of a block of data and is readily available on the GPIB as an option for instruments and controllers. In addition, the TxRQ input removes the DMAC from the bus and puts the MC6809 on the bus during the second post-DMA dead cycle that occurs during data chaining operations.

With the system in a typical transfer mode, the transfer request signal $\overline{DRQT}$ is gated to the synchronization circuitry.The purpose of the circuitry at this time is to delay the transfer request until the next high Q. Thus, not only should the signal be clocked through on positive edges of Q, but it should also be allowed to appear directly at the $\overline{DMA}$/$\overline{BREQ}$ input of the MC6809 when Q is high. Therefore, the flip-flop latches on positive edges and, during the positive half of Q, passes the signal directly to the MC6809. This enables the system to work both in its present format as well as with other peripherals which may signal their transfer requests later in time.

TIMING DESCRIPTION — This description assumes initialization of peripherals and controllers and a typical character transfer to/from memory. Both transfer types are shown — the byte from memory (talker mode) and a byte to memory (listener mode). To alleviate any timing losses on the IEEE-488 bus, a Hewlett Packard GPIB emulator with an automatic high-speed receiver/transmitter is used as the "other end" sender/receiver. This TTL device has an internal delay in both modes of 80 nanoseconds (due to the readying of new data while the MC68488 receives/talks).

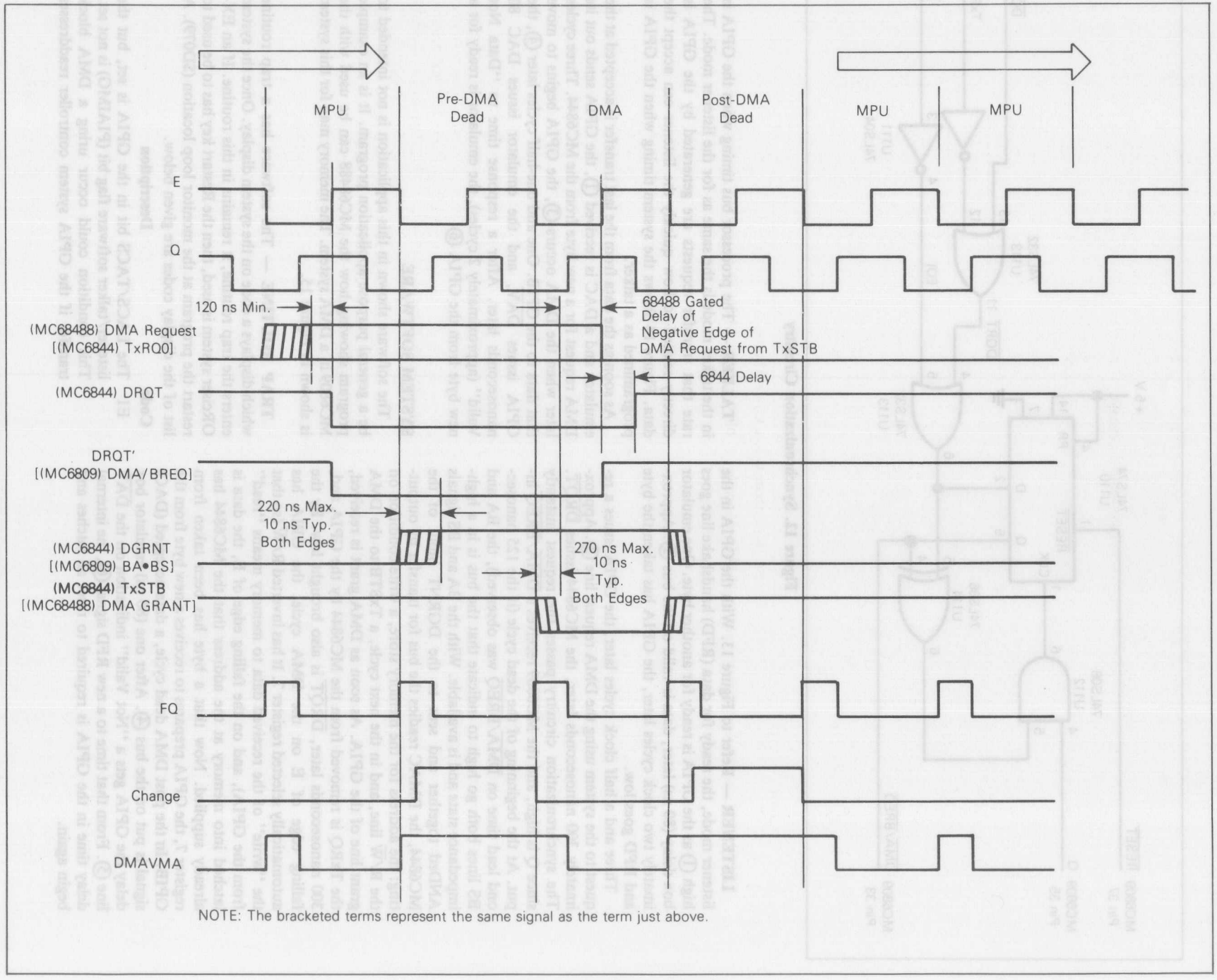

Figure 11. System Timing Diagram

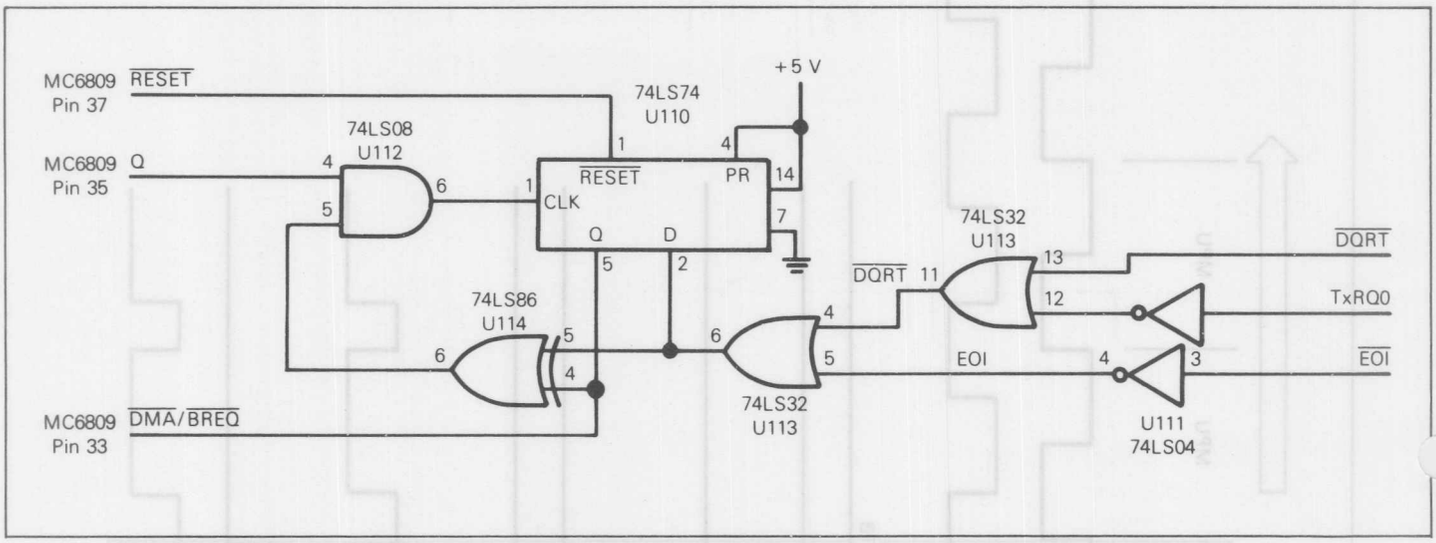

Figure 12. Synchronization Circuitry

LISTENER — Refer to Figure 13. With the GPIA in the listener mode, the ready for data (RFD) handshake line goes high ① as the GPIA is ready for another byte. One emulator box delay (80 ns) later, data is valid on the bus ②. Approximately two clock cycles later, the GPIA has taken the byte and RFD goes low.

Three and a half clock cycles later, the GPIA issues a request to the system using the DMA request line ③. Approximately 300 nanoseconds later, the MC6844 issues $\overline{\text{DRQT}}$. The synchronization circuitry passes the request instantly since Q is high, and the MC6809 receives a $\overline{\text{DMA/BREQ}}$ input. At the beginning of the dead cycle (if the 125 nanosecond lead time on $\overline{\text{DMA/BREQ}}$ was observed), the BA and BS lines both go high to indicate that the bus is in a high-impedance state and is available. With the BA and BS signals ANDed together and sent to the DGRNT input of the MC6844, the DMAC readies the bus for transfer by outputting: the address for the memory store, a write condition on the R/$\overline{\text{W}}$ line, and in the next cycle, a TxSTB to the DMA grant line of the GPIA. As soon as DMA grant is received, the TxRQ is removed from the MC6844 by the GPIA and, 300 nanoseconds later, $\overline{\text{DRQT}}$ is also brought low. By the falling edge of E on the DMA cycle, the GPIA has automatically selected register 7. It has inverted R/$\overline{\text{W}}$ (so that the "write" of the received data to memory means "read" from the GPIA), and on the falling edge of E, the data is latched into memory at the address that the MC6844 has already supplied. Now that a byte has been taken from register 7, the GPIA prepares to receives a new byte from the GPIB. In the post DMA dead cycle, a data accepted (DAC) signal is put on the bus ④. After one (80 ns) emulator box delay the GPIA gets a "Not Valid" indication on the $\overline{\text{DAV}}$ line ⑤. From that time to a new RFD signal ⑥, the internal delay time in the GPIA is required to reset all latches and begin again.

TALKER — The processor bus timing when the GPIA is in the talker mode is the same as for the listener mode. The rate that transfer requests are generated by the GPIA is directly related to how quickly the listener can accept the data. Figure 14 shows the system timing when the GPIA is programmed as a talker.

As soon as the data from the last transfer is accepted at the emulator and a DAC is received ①, the GPIA sends out its DMA request for a new byte from the MC6844. Three cycles later when the DMA occurs ③, the GPIA begins to move that data to the GPIB. One and one-half cycles later ④, the GPIA issues $\overline{\text{DAV}}$, and the emulator issues DAC 80 nanoseconds later. After a response time to "Data Not Valid" (approximately 2 cycles), the emulator is ready for a new byte from the GPIA ⑥.

SYSTEM SOFTWARE

The software shown in this application is not intended to be a general purpose application program. It is an example program showing how the MC68488 can be used with the MC6809 in a DMA system. The memory map for this system is shown in Figure 15.

TRAP ROUTINE — The software has a trap routine which displays a code on the system display. Once the system enters the trap routine, it remains in this routine. If an EX-ORciser system is used, then the Restart key has to be used to restart the program at the monitor loop location ($D079). A list of the display codes are given below.

Code	Description
E1	The LACS/TACS bit in the GPIA is set, but the listener/talker software flag bit (PIAIMG) is not set. This condition could occur uring a DMA block transfer if the GPIA system controller readdresses

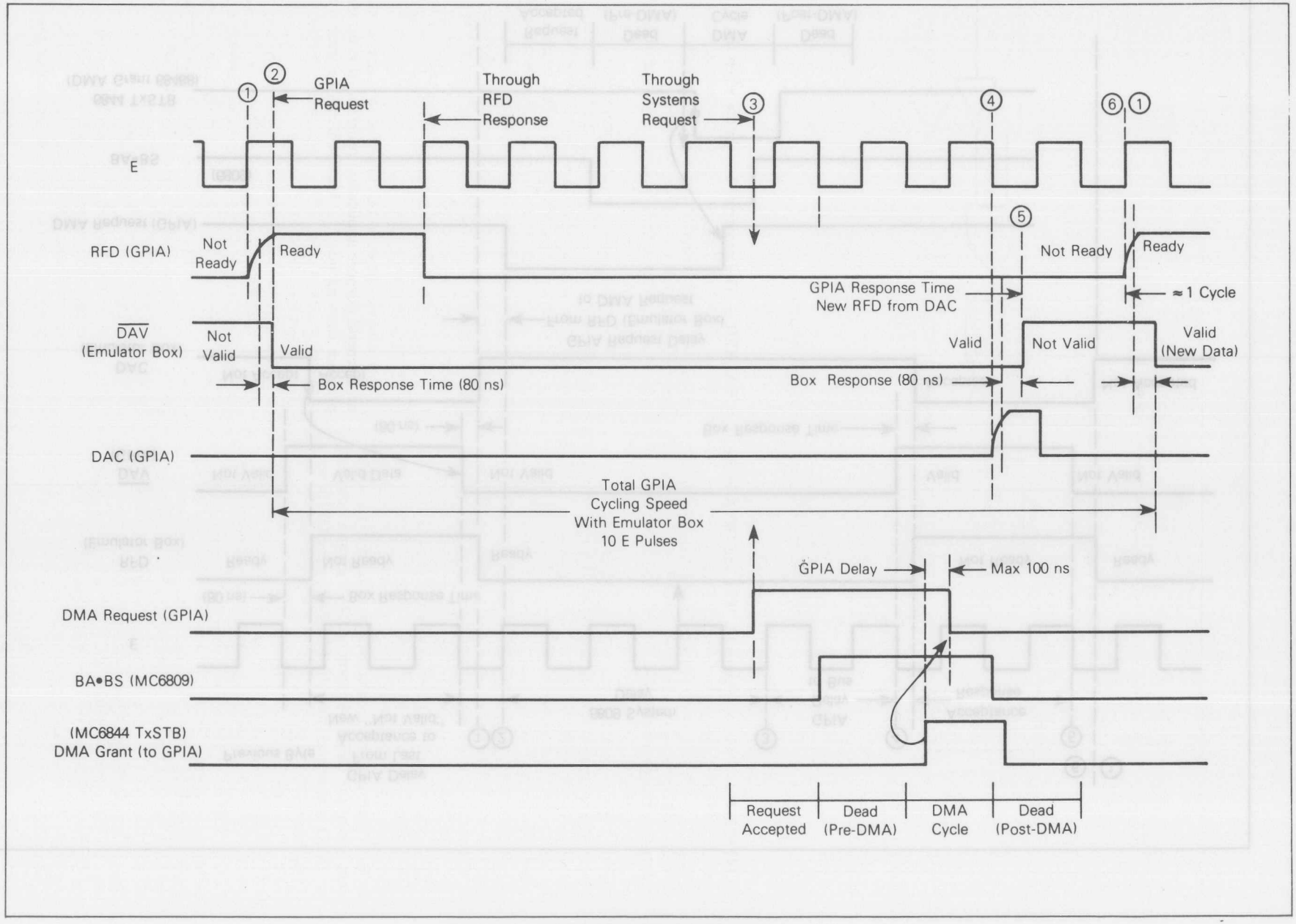

Figure 13. Listener Mode Timing Diagram

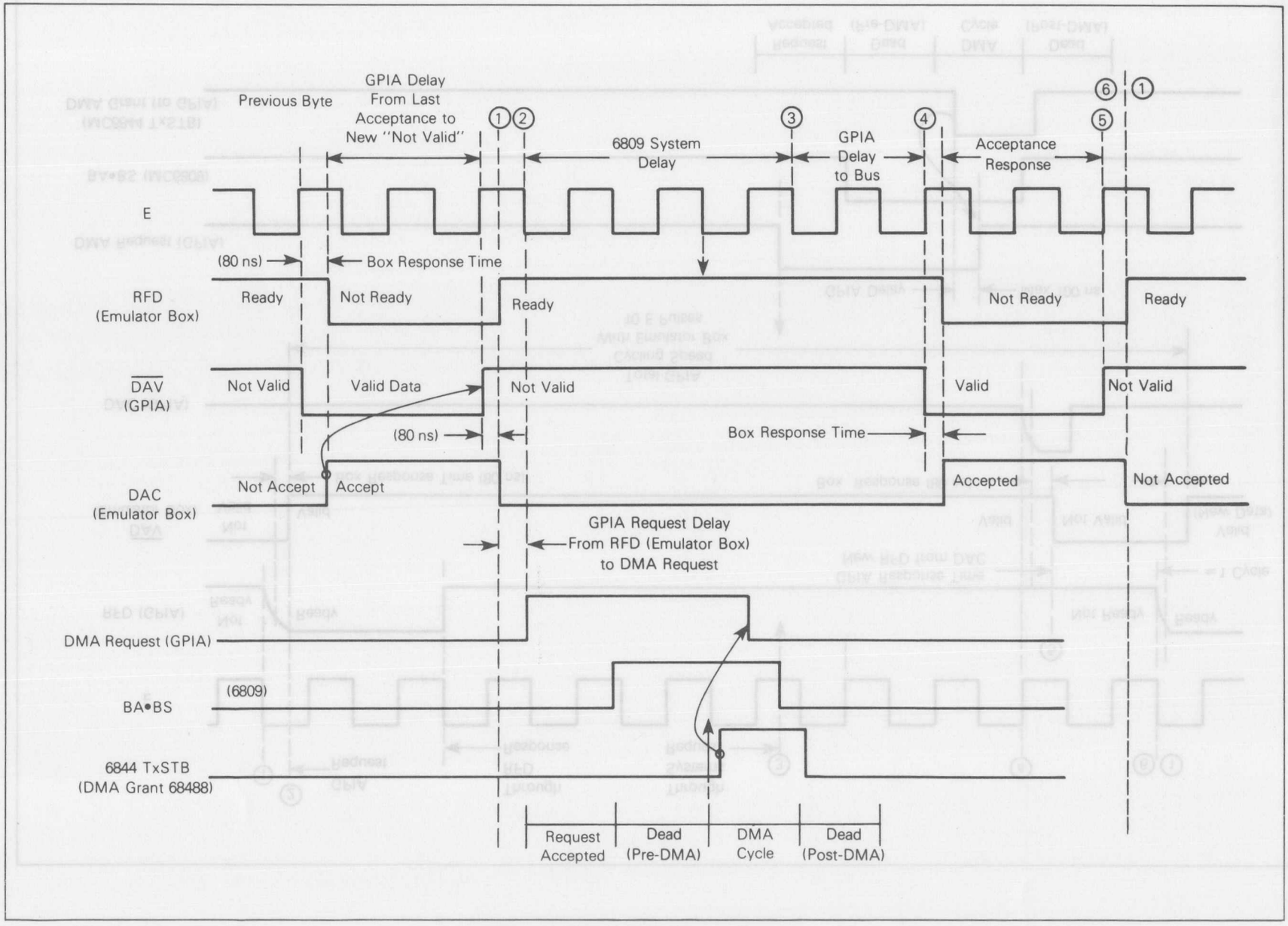

Figure 14. Talker Mode Timing Diagram

Figure 15. Memory Map

Memory Function	Memory Location
MC68488 Registers	$E060-$E067
MC6844 Registers	$E040-$E056
Display System (PIA Registers)	$E070-$E073
Main Program	ORG at $D000
Receive Memory Buffer	$D800-$D8FF
Talker Memory Buffer	$D800-$DBFF

the GPIA to be a talker when it was a listener or vice versa.

E2 The DMAC caused the interrupt, but the system was not programmed to be a talker. Under normal operations, the DMAC should only interrupt the MC6809 when the system is in the listener mode. If it interrupts when the system is in the listener mode, then the count in the DMAC byte count register was exceeded by the actual number of bytes in the block received. The byte count register must be initialized with a larger number or the block of data to be transferred must be broken up into smaller blocks.

E3 Neither the DMAC nor the GPIA interrupt bits are set. The interrupt was caused by another device or the GPIA produced a "ghost interrupt." In this system the only way the GPIA produces a "ghost interrupt" is if the GPIB system controller places the GPIA in the serial poll active state (SPAS) and then removes it from this state before the MC6809 can respond to the interrupt.

E4 The SPAS bit is set. This occurs if the GPIB system controller sends the serial poll enable command and then sends the device talk address placing the GPIA in the serial poll active state.

EXAMPLE PROGRAM LISTING — The following program listing is an example program to show how the MC68488 can be used with the MC6809 in a DMA mode.

```
PAGE  001  GPIA2   .SA:0  GPIA1

00001                          *              MC68488(GPIA)
00002                          *            6809 - DMA SYSTEM
00003                          *                 8/26/79
00004                          *
00005                          *
00006                                NAM    GPIA1
00007                                OPT    LLF=80,ABS
00008A D000                          ORG    $D000
00009                          *
00010                          *THE FOLLOWING ARE GPIA REGISTER ADDRESS
00011                          *LOCATIONS
00012                          *
00013          E060     A R0R    EQU    $E060    INTERRUPT REG
00014          E060     A R0W    EQU    $E060    INTERRUPT MASK REG
00015          E061     A R1R    EQU    $E061    COMMAND STATUS REG
00016          E062     A R2R    EQU    $E062    ADDRESS STATUS REG
00017          E062     A R2W    EQU    $E062    ADDRESS MODE REG
00018          E063     A R3R    EQU    $E063    AUXILLARY COMMAND REG
00019          E063     A R3W    EQU    $E063    AUXILLARY COMMAND REG
00020          E064     A R4R    EQU    $E064    ADDRESS SWITCH REG
00021          E064     A R4W    EQU    $E064    ADDRESS REGISTER
00022          E065     A R5R    EQU    $E065    SERIAL POLL REG
00023          E065     A R5W    EQU    $E065    SERIAL POLL REG(WRITE)
00024          E066     A R6R    EQU    $E066    COMMAND PASS-THRU REG
00025          E066     A R6W    EQU    $E066    PARALLEL POLL REG
00026          E067     A R7R    EQU    $E067    DATA IN REG
00027          E067     A R7W    EQU    $E067    DATA OUT REG
00028                          *
00029                          *THEFOLLOWING ARE DMAC REGISTER ADDRESS LOCATIONS
00030                          *
00031          E050     A CHCON  EQU    $E050    CHANNEL CONTROL REG
00032          E054     A PRICON EQU    $E054    PRIORITY CONTROL REG
00033          E055     A INTCON EQU    $E055    INTERRUPT CONTROL REG
00034          E040     A ADDH0  EQU    $E040    HIGH ORDER ADD BYTE
00035          E041     A ADDL0  EQU    $E041    LOW ORDER ADDRESS BYTE
00036          E042     A BYTEH0 EQU    $E042    HIGH ORDER BYTE COUNT
00037          E043     A BYTEL0 EQU    $E043    LOW ORDER BYTE COUNT
00038          E04C     A ADDH3  EQU    $E04C    HIGH ORDER ADD BYTE
00039          E04D     A ADDL3  EQU    $E04D    LOW ORDER ADDRESS BYTE
00040          E04E     A BYTEH3 EQU    $E04E    HIGH ORDER BYTE COUNT
00041          E04F     A BYTEL3 EQU    $E04F    LOW ORDER BYTE COUNT
00042          E056     A DCHAIN EQU    $E056    DATA CHAIN REG
00043                          *
00044                          *THEFOLLOWING ARE PIA REGISTER ADDRESS
00045                          *LOCATIONS
00046                          *
00047          E071     A CRA    EQU    $E071    CONTROL REG A
00048          E073     A CRB    EQU    $E073    CONTROL REG B
00049          E070     A PRA    EQU    $E070    PERIPHERAL REG A
00050          E072     A PRB    EQU    $E072    PERIPHERAL REG B
00051          E070     A DDA    EQU    $E070    DATA DIRECTION REG A
00052          E072     A DDB    EQU    $E072    DATA DIRECTION REG B
00053                          *
00054A D000      00FF     A LBYCNT FDB    $00FF    MAX NO. LISTEN BYTES
00055A D002      00FE     A TBYCNT FDB    $00FE    N-1 OF N TALK BYTES
00056A D004      D800     A LMEMPT FDB    $D800    LISTEN MEM BUF POINTER
00057A D006      DB00     A TMEMPT FDB    $DB00    TALK MEM BUF POINTER
00058A D008      04       A RENON  FCB    $04      REN LIGHT ON MASK
```

```
PAGE  002  GPIA2   .SA:0  GPIA1

00059A D009      FB       A RENOFF FCB    $FB      REN LIGHT OFF MASK
00060A D00A      01       A TALKON FCB    $01      TALK LIGHT ON MASK
00061A D00B      FE       A TALKOF FCB    $FE      TALK LIGHT OFF MASK
00062A D00C      02       A LISTON FCB    $02      LISTEN LIGHT ON MASK
00063A D00D      FD       A LISTOF FCB    $FD      LISTEN LIGHT OFF MASK
00064A D00E      63       A SCALER FCB    $63      BLOCK DISPLAY PRESCALER
00065A D00F      86       A MASK   FCB    $86      GPIA INTERRUPT MASK
00066A D010      80       A DSEL   FCB    $80      DESELECTS STATUS BITS
00067
00068A D011      0001     A BLOCK  RMB    1        NO. OF PRESCALED BLOCKS TRANSF
00069A D012      0001     A COMP   RMB    1        PRESCALE COUNT COMPARE
00070A D013      0001     A PIAIMG RMB    1        IMAGE OF LED LIGHTS
00071                          *
00072                          *
00073                          ***INITIALIZATION ROUTINE**
00074                          *
00075                          *
00076A D014 1A   10       A        ORCC   #$10     DISABLE INTERRUPTS
00077                          *
00078                          *CLEAR LISTENER & TALKER MEM BUFFERS
00079                          *
00080A D016 4F                   CLRA
00081A D017 B7   D012     A        STA    COMP     SET COMP TO ZERO
00082A D01A BE   D004     A        LDX    LMEMPT   GET LIST. MEM POINTER
00083A D01D 10BE D000     A        LDY    LBYCNT   GET LIST. BYTE COUNT
00084A D021 A7   84       A LOOP1  STAA   0,X      CLEAR MEM LOCATION
00085A D023 30   1F       A        LEAX   -1,
00086A D025 31   3F       A        LEAY   -1,Y      DEC TO GET NEXT LOCATION
00087A D027 26   F8   D021 A        BNE    LOOP1    IF NOT, CLEAR ANOTHER
00088A D029 30   1F       A        LEAX   -1,X
00089A D02B 31   3F       A        LEAY   -1,Y
00090A D02D A7   84       A LOOP2  STA    0,X      CLEAR MEM LOCATION
00091A D032 30   1F       A        LEAX   -1,X      MOVE TO NEXT MEM LOCATION
00092A D034 31   3F       A        LEAY   -1,Y      IS THIS LAST LOCATION OF BUFFE
00093A D036 26   F8   D030 A        BNE    LOOP2    IF NOT, CLEAR ANOTHER
00094                          *
00095                          *INITIALIZE GPIA
00096                          *
00097A D038 B6   E064     A        LDA    R4R      READ ADDRESS SWITCHES
00098A D03B B7   E064     A        STA    R4W      PLACE ADDRESS IN GPIA
00099A D03E 86   00       A        LDA    #$00
00100A D040 B7   E063     A        STA    R3W      REMOVE GPIA RESET
00101A D043 B6   D010     A        LDA    DSEL     DESEL CERTAIN STATUS
00102A D046 B7   E062     A        STA    R2W      BITS FROM CAUSING IRQ
00103
00104A D049 B6   D00F     A        LDA    MASK     GET IRQ MASK BYTE
00105A D04C B7   E060     A        STA    R0W      SET UP GPIA IRQ MASK
00106                          *
00107                          *INITIALIZE PIA
00108                          *
00109A D04F 86   00       A        LDA    #$00     SELECT DATA DIRECTION
00110A D051 B7   E071     A        STA    CRA      REG FOR A PORT
00111A D054 B7   E073     A        STA    CRB      SELECT PORT B DATA DIR.
00112A D057 86   FF       A        LDA    #$FF
00113A D059 B7   E070     A        STA    DDA      PORT A AS OUTPUT
00114A D05C B7   E072     A        STA    DDB      PORT B AS OUTPUT
00115A D05F 86   04       A        LDA    #$04
00116A D061 B7   E071     A        STA    CRA      SELECT PERIPH. REG A
```

```
00117A D064 F7   E073    A        STB    CRB      SELECT PERIPH. REG B
00118A D067 86   00      A        LDA    #$00
00119A D069 B7   E070    A        STA    PRA      INIT. PORT A TO ZERO
00120A D06C B7   E072    A        STA    PRB      INIT  PORT B TO ZERO
00121                             *
00122                             *ASSIST09 INTERRUPT PROCEDURE
00123                             *
00124A D06F 1C   FF      A        ANDCC  #$EF     ENABLE SYS. INTERRUPTS
00125A D071 86   0C      A        LDA    #12
00126A D073 30   8D 00E3          LEAX   IRQ,PCR
00127A D077 3F                    SWI
00128A D078      09      A        FCB    9
00129                             *
00130                             *FALL THROUGH TO MONITOR
00131                             *
00132                             ***MONITOR**
00133                             *
00134A D079 86   00      A MONIT  LDA    #$00
00135A D07B B7   E060    A        STA    R0R      RESET GPIA IRG MASK
00136A D07E B6   D00F    A        LDA    MASK
00137A D081 B7   E060    A        STA    R0R      SET GPIA IRQ MASK
00138                             *
00139                             *CHECK GPIA TO SEE IF ACTIVE TALKER/LISTENER
00140                             *
00141A D084 F6   E062    A LOOP7  LDB    R2R      LOAD GPIA ADD. STATUS
00142A D087 C5   04      A        BITB   #$04     IS LACS SET
00143A D089 26   1E   D0A9        BNE    LACS     IF YES, SET UP DMAC
00144A D08B B6   D013    A        LDA    PIAIMG   IF NO,
00145A D08E B4   D00D    A        ANDA   LISTOF   TURN OFF LISTEN LIGHT
00146A D091 B7   E072    A        STA    PRB      SEND TO FRONT PANEL
00147A D094 B7   D013    A        STA    PIAIMG   UPDATE PIAIMG
00148A D097 C5   08      A        BITB   #$08     IS TACS SET
00149A D099 26   45   D0E0        BNE    TACS     IF YES, SET UP DMAC
00150A D09B B6   D013    A        LDA    PIAIMG   IF NO,
00151A D09E B4   D00B    A        ANDA   TALKOF   TURN OFF TALK LIGHT
00152A D0A1 B7   E072    A        STA    PRB      SEND TO FRONT PANEL
00153A D0A4 B7   D013    A        STA    PIAIMG   UPDATE PIAIMG
00154A D0A7 20   DB   D084        BRA    LOOP7    TEST GPIA ADD STATUS
00155                             *
00156                             *SET UP DMAC FOR LISTEN ROUTINE
00157                             *
00158A D0A9 B6   D013    A LACS   LDA    PIAIMG
00159A D0AC B4   D00B    A        ANDA   TALKOF   TURN OFF TALK LIGHT
00160A D0AF 8A   D00C    A        ORA    LISTON   TURN ON LISTEN LIGHT
00161A D0B2 B7   E072    A        STA    PRB      SEND TO FRONT PANEL
00162A D0B5 B7   D013    A        STA    PIAIMG   UP DATE PIAIMG
00163A D0B8 BE   D004    A        LDX    LMEMPT   GET LIST. START ADD
00164A D0BB BF   E040    A        STX    ADDH0    PUT IN DMAC CHAMMEL 0
00165A D0BE BF   E04C    A        STX    ADDH3    PUT IN DMAC CHANNEL 3
00166A D0C1 BE   D000    A        LDX    LBYCNT   GET NO. OF BYTES
00167A D0C4 BF   E042    A        STX    BYTEH0   PUT IN DMAC CHANNEL 0
00168A D0C7 BF   E04E    A        STX    BYTEH3   PUT IN DMAC CHANNEL 3
00169A D0CA 86   04      A        LDA    #$04     SELECT UP COUNT, TSC
00170A D0CC B7   E050    A        STA    CHCON    STEAL, & MEM WRITE
00171A D0CF 86   81      A        LDA    #$81     SELECT IRO ON DEND
00172A D0D1 B7   E055    A        STA    INTCON   PUT IN DMAC
00173A D0D4 86   00      A        LDA    #$00
00174A D0D6 B7   E056    A        STA    DCHAIN   DISABLE DATA CHAIN FEATURE
```



```
00175A D0D9 86   01      A        LDA    #$01
00176A D0DB B7   E054    A        STA    PRICON   ENABLE CH. 0 TRANSFER REQUEST
00177A D0DE 20   49   D129        BRA    WAIT
00178                             *
00179                             *SET UP DMAC FOR TALK ROUTINE
00180                             *
00181A D0E0 B6   D013    A TACS   LDA    PIAIMG
00182A D0E3 B4   D00D    A        ANDA   LISTOF   TURN OFF LISTEN LIGHT
00183A D0E6 8A   D00A    A        ORA    TALKON   TURN ON TALK LIGHT
00184A D0E9 B7   E072    A        STA    PRB      SEND TO FRONT PANEL
00185A D0EC B7   D013    A        STA    PIAIMG   UPDATE PIAIMG
00186                             *
00187                             *LOAD TALKER MEM BUFFER
00188A D0EF BE   D006    A        LDX    TMEMPT   GET MEM POINTER
00189A D0F2 10BE D002    A        LDY    TBYCNT   GET NO. OF BYTES
00190A D0F6 C6   00      A        LDB    #$00
00191A D0F8 E7   84      A LOOP3  STB    0,X      STORE DATA BYTE IN MEM
00192A D0FA 5C                    INCB             INC. NO. TO BE STORED
00193A D0FB 30   1F      A        LEAX   -1,X     DEC. ADDRESS POINTER
00194A D0FD 31   3F      A        LEAY   -1,Y     DECRIMENT BYTE COUNT
00195A D0FF 26   F7   D0F8        BNE    LOOP3    IF NOT LAST DO ANOTHER
00196A D101 BE   D006    A        LDX    TMEMPT   GET TALK BUF ADD.
00197A D104 BF   E040    A        STX    ADDH0    PUT IN DMAC CHANNEL 0
00198A D107 BF   E04C    A        STX    ADDH3    PUT IN DMAC CHANNEL 3
00199A D10A BE   D002    A        LDX    TBYCNT   GET NO. OF BYTES
00200A D10D BF   E042    A        STX    BYTEH0   PUT IN DMAC CHANNEL 0
00201A D110 BF   E04E    A        STX    BYTEH3   PUT IN DMAC CHANNEL 3
00202A D113 86   05      A        LDA    #$05     SELECT UP COUNT, TSC
00203A D115 B7   E050    A        STA    CHCON    STEAL , & MEM READ
00204A D118 86   81      A        LDA    #$81     SELECT IRO ON DEND
00205A D11A B7   E055    A        STA    INTCON   PUT IN DMAC
00206A D11D 86   00      A        LDA    #$00     DISABLE DATA CHAINING
00207A D11F B7   E056    A        STA    DCHAIN   FEATURE OF DMAC
00208A D122 86   01      A        LDA    #$01
00209A D124 B7   E054    A        STA    PRICON   ENABLE CH 0 TRANSFER
00210A D127 20   00   D129        BRA    WAIT
00211                             *
00212                             *WAIT LOOP - WAITS FOR A DMA REQUEST TO OCCUR. TALK/L
00213                             *            CONDITION RECOGNIZED AND DMAC HAS BEEN S
00214                             *            ACCORDINGLY. WAIT LOOP ALSO CHECKS FOR A
00215                             *            IN GPIA ADDRESS STATUS.
00216                             *            IF ADDRESSED DIFFERENTLY THAN IT WAS
00217                             *            WHEN WAIT LOOP ENTERRED AN E1 TRAP
00218                             *
00219                             *            WILL BE PRODUCED.
00220                             *
00221A D129 B6   E062    A WAIT   LDA    R2R      LOAD GPIA ADD. STATUS
00222A D12C 85   80      A        BITA   #$80     IS MA BIT SET
00223A D12E 27   22   D152        BEQ    OFF      IF NO, GO TURN OFF DMAC
00224
00225A D130 85   04      A        BITA   #$04     IS LACS BIT SET
00226A D132 27   11   D145        BEQ    TACHIT   IF NO, GO TEST TACS
00227A D134 B6   D013    A        LDA    PIAIMG   IF YES, SEE IF LISTEN FLAG SET
00228A D137 85   02      A        BITA   #$02     IS LISTEN FLAG SET
00229A D139 26   EE   D129        BNE    WAIT     IF YES, ALL IS `OK` - CHECK R2
00230A D13B 86   00      A TRAP1  LDA    #$00     IF NO, ALL IS NOT OK
00231A D13D B7   E054    A        STA    PRICON   TURN OFF DMAC
00232A D140 86   E1      A        LDA    #$E1     LOAD ACC A WITH TRAP CODE
```

```
PAGE  005  GPIA2   .SA:0  GPIA1

00233A D142 16   009B D1E0        LBRA   TRAP      GO TO TRAP ROUTINE
00234A D145 85   08       A TACBIT BITA   #$08      IS TACS BIT SET
00235A D147 27   E0   D129        BEQ    WAIT      IF NO, GO TEST ADDRESS STATUS
00236A D149 B6   D013     A        LDA    PIAIMG    IF YES, CHECK TALK
00237A D14C 85   01       A        BITA   #$01      IS TALK FLAG SET
00238A D14E 26   D9   D129        BNE    WAIT      IF YES, ALL `OK`
00239A D150 20   E9   D13B        BRA    TRAP1     IF NO, GO SET E1 DISPLAY
00240A D152 86   00       A OFF    LDA    #$00      TURN OFF DMAC
00241A D154 B7   E054     A        STA    PRICON
00242A D157 16   FF1F D079        LBRA   MONIT     GO TO MONITOR
00243
00244                      *
00245                      *
00246                      ***INTERRUPT ROUTINE**
00247                      *
00248                      *
00249                      *
00250                      *CHECK FOR DMAC INTERRUPT
00251                      *
00252A D15A B6   E050     A IRQ    LDA    CHCON     LOAD DMAC CONTROL REG
00253A D15D 85   80       A        BITA   #$80      IS IRQ FROM DMAC
00254A D15F 27   0F   D170        BEQ    GPIA      IF NO, GO CHECK GPIA
00255A D161 85   01       A        BITA   #$01      IS DMA IN TALK MODE
00256A D163 27   07   D16C        BEQ    TRAP2     IF NO, GO TO TRAP2
00257A D165 86   00       A        LDA    #$00      IF YES,
00258A D167 B7   E054     A        STA    PRICON    TURN OFF DMAC
00259A D16A 20   31   D19D        BRA    TALAST    GO SEND LAST BYTE
00260A D16C 86   E2       A TRAP2  LDA    #$E2      LOAD ACC A WITH TRAP2 CODE
00261A D16E 20   70   D1E0        BRA    TRAP      GO TO TRAP ROUTINE
00262                      *
00263                      *CHECK FOR GPIA INTERRUPT
00264                      *
00265A D170 B6   E060     A GPIA   LDA    R0R       GET GPIA IRQ STATUS
00266A D173 85   80       A        BITA   #$80      IS IRQ FROM GPIA
00267A D175 27   11   D188        BEQ    TRAP3     IF NO, GO TO TRAP3 ROUTINE
00268A D177 85   04       A        BITA   #$04      IS CMD BIT SET
00269A D179 26   11   D18C        BNE    RLC       IF YES, IS RLC SET
00270A D17B 85   02       A        BITA   #$02      IF NO, GO TO TRAP3
00271A D17D 27   09   D188        BEQ    TRAP3     IF NO, GO TO TRAP3
00272A D17F 85   01       A BI     BITA   #$01      IF END IS YES, IS BI
00273A D181 26   2A   D1AD        BNE    LILAST    IF YES, GET LAST BYTE
00274A D183 B6   E060     A        LDA    R0R       IF NO, LOAD ROR
00275A D186 20   F7   D17F        BRA    BI        AND TEST BI AGAIN
00276A D188 86   E3       A TRAP3  LDA    #$E3      LOAD ACC A WITH TRAP3 CODE
00277A D18A 20   54   D1E0        BRA    TRAP      GO TO TRAP ROUTINE
00278A D18C B6   E061     A RLC    LDA    R1R       GET GPIA COMMAND STATUS
00279A D18F 85   08       A        BITA   #$08      IS RLC SET
00280A D191 27   06   D199        BEQ    TRAP4     IF NO, GO TO TRAP4
00281A D193 85   40       A        BITA   #$40      IF YES, IS REM SET
00282A D195 27   3B   D1D2        BEQ    REMOFF    IF NO, TURN OFF REN LIGHT
00283A D197 20   21   D1BA        BRA    REMON     IF YES, TURN ON REN LIGHT
00284A D199 86   E4       A TRAP4  LDA    #$E4      LOAD ACC A WITH TRAP4 CODE
00285A D19B 20   43   D1E0        BRA    TRAP      GO TO TRAP ROUTINE
00286                      *
00287                      *SEND LAST BYTE AS A TALKER
00288                      *
00289A D19D 86   01       A TALAST LDA    #$01
00290A D19F B7   E063     A        STA    R3W       SET feoi
```

```
PAGE  006  GPIA2   .SA:0  GPIA1

00291A D1A2 BE   E040     A        LDX    ADDH0     GET ADD OF LAST BYTE
00292A D1A5 A7   84       A        STA    0,X       GET LAST BYTE
00293A D1A7 B7   E067     A        STA    R7W       SEND LAST BYTE
00294A D1AA 16   FECC D079        LBRA   MONIT
00295                      *
00296                      *RECEIVE LAST BYTE ROUTINE
00297                      *
00298A D1AD 86   00       A LILAST LDA    #$00
00299A D1AF B7   E054     A        STA    PRICON    TURN OFF DMAC
00300A D1B2 BE   E040     A        LDX    ADDH0     LOAD LAST BYTE ADDRESS IN X RE
00301A D1B5 B6   E067     A        LDA    R7R       GET LAST BYTE
00302A D1B8 A7   84       A        STA    0,X       STORE IT
00303
00304                      *
00305                      *SET REMOTE ENABLE LIGHT INDICATOR
00306                      *
00307A D1BA B6   D008     A REMON  LDA    RENON     GET RENON MASK
00308A D1BD BA   D013     A        ORA    PIAIMG    `OR` WITH CURRENT STATUS
00309A D1C0 B7   D013     A        STA    PIAIMG    UPDATE PIAIMG
00310A D1C3 B7   E072     A        STA    PRB
00311                      *
00312                      *RESET AND SET GPIA MASK REG
00313                      *
00314A D1C6 86   00       A RESETM LDA    #$00
00315A D1C8 B7   E060     A        STA    R0W       RESET GPIA IRQ MASK
00316A D1CB B6   D00F     A        LDA    MASK
00317A D1CE B7   E060     A        STA    R0W       SET GPIA IRQ MASK
00318A D1D1 3B                    RTI
00319                      *
00320                      *RESET REMOTE ENABLE LIGHT INDICATOR
00321                      *
00322A D1D2 B6   D009     A REMOFF LDA    RENOFF    GET RENOFF MASK
00323A D1D5 B4   D013     A        ANDA   PIAIMG    TURN OFF REN BIT
00324A D1D8 B7   D013     A        STA    PIAIMG    UPDATE PIAIMG
00325A D1DB B7   F072     A        STA    PRB       TURN REN OFF
00326A D1DE 20   E6   D1C6        BRA    RESETM
00327                      *
00328                      *TRAP ROUTINE
00329                      *
00330A D1E0 B7   E070     A TRAP   STA    PRA       SEND TRAP CODE TO DISPLAY
00331A D1E3 20   FB   D1E0        BRA    TRAP
00332                             END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000
```