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 (IRQ), fast interrupt request (FIRQ), and non-maskable interrupt (NMI). The entire machine state is saved for IRQ and NMI. This can take up to 20 E clock pulses. The 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 FIRQ, they need to be saved via software.

©MOTOROLA INC., 1981
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 inter-
face 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 (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 in-
teract directly with the measurement process of an instru-
ment. They interact only with the interface logic within con-
ected 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 prob-
lems 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 switch-
es attached to the device or a subroutine resident in the soft-
ware controlling the device. The address is placed in the
GPIB interface for the device during an initialization se-
quence. 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 par-
ticular 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 (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 possi-
ble 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 com-
mand 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 se-
quence by the processor along with other appropriate
initialization procedures.

2. The GPIB controller takes control of the bus by asser-
ting ATN and, with the appropriate interface com-
mands, 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 informa-
tion 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 con-
troller now releases control of the GPIB, by negating
ATN allowing the data transfer to take place.

6. The talker now sends the data in a byte-per-byte se-
quence 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 EOI general in-
terface 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/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 CONTROL — The MC6844 requests control of the bus from the MC6809 and issues the appropriate commands (via the R/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 DMA/BREQ input during the last quarter cycle of the E signal. 2) The end or identify (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 EOI input to the synchronization circuitry is used only when DMA transfers are being made from the GPIA to memory.
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](image)

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 UACC, 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 register 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 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
This bit is set if any of the other bits in ROR are set and the mask bits are enabled in ROW. This bit is used to generate IRQ.

In the Talker mode this bit indicates when a byte can be written to R7W. When set it will issue a DMA transfer request. Interrupt for this bit is disabled.

This bit is deselected in this system by bit 7, R2W. Always in low state. Interrupt is disabled.

Unused position. Always in high state.

This feature is not used in this system. Interrupts disabled.

When set this bit indicates that either UUCG, UACG, RLC, SPAS or DCAS are set in command status register (R1R). Interrupts enabled for this bit.

When set this bit indicates that the EOI management line is asserted and GPIA is in LACS. Interrupts enabled for this bit.

In the listener mode this bit indicates the reception of a data byte from the addressed talker. When set a DMA request is issued. Interrupt for this bit is disabled.

This bit is deselected in this system by bit 7, R2W. Is always low and thus can not cause a CMD interrupt.

Device is in Remote state when REM = 1 and Local state when REM = 0. Any change in this bit causes RLC bit to be set.

This bit reports the LOCK state for the Remote/Local feature. This bit is not used in this system.

Unused bit location.

RLC bit reports a change in REM bit and causes a CMD interrupt.

This bit is set if GPIB control places device in Serial Poll Active State. If set CMD interrupt occurs and software enters a Trap routine.

These bits are deselected in this system by bit 7, R2W, are always low and thus do not cause a CMD interrupt.

Figure 4. GPIA Interrupt and Command Status Register
Set up DMAC to Receive
1. Address Register
2. Byte Count Register
3. Channel Control Register
4. Interrupt Control Register
5. Priority Control Register
   (must be done last)

Reset and Set GPIA IRQ Mask

Load Address Status

Listen Mode?

Yes

Turn Listen Light On

No

Turn Listen Light Off

Set up DMAC to Receive
1. Address Register
2. Byte Count Register
3. Channel Control Register
4. Interrupt Control Register
5. Priority Control Register
   (must be done last)

Talk Mode?

Yes

Turn Talk Light On

No

Turn Talk Light Off

Set Up Data To Be Transferred

Initialize DMAC to Send
1. Address Register
2. Byte Count Register
3. Channel Control Register
4. Interrupt Control Register
5. Priority Control Register
   (must be done last)

Wait Loop

Figure 5. Monitor 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 SEI 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 (EOI) management line to end the transfer. Asserting 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 (DRQT) to the synchronization circuitry. It synchronizes this request with the Q clock from the MC6809 and issues a DMA/BREQ to the MC6809 during the Q high
time. The low input on the MC6809 DMA/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 DMA/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 EOI line low. The 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 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 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 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 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.

**DMAC** — The DMAC is to exchange control of the processor bus with the GPIA. In the initial state, the DMAC is programmed 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 DMAC is programmed to transfer data from GPIA to memory indicating that the GPIA is programmed to be a talker. In this instance, the byte count register was initialized with a number small enough 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 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 DAV true for one transfer to the time 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
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/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/W, and proceed with the data transfer when a DMA Grant occurs. Therefore, no R/W inverters or data bus drivers are needed.

SYNCHRONIZATION CIRCUITRY — The synchronization circuitry is shown in Figure 12. During a transfer the gating of EOI and DRQT prevents the data transfer request (from the DMAC) from being applied to the processor when EOI is asserted. With no transfer request applied to the MC6809 it resumes a normal operation. In parallel with the assertion of EOI, the MC68488 has issued an interrupt request (IRQ) to the MC6809 to service a last byte condition signified by the presence of 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 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 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 DMA/REQ 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).
NOTE: The bracketed terms represent the same signal as the term just above.

Figure 11. System Timing Diagram
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 DRQT. The synchronization circuitry passes the request instantly since Q is high, and the MC6809 receives a DMA/BREQ input. At the beginning of the dead cycle (if the 125 nanosecond lead time on 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/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, 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/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 receive 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 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 data to the GPIB. One and one-half cycles later (4), the GPIA issues 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.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>The LACS/TACS bit in the GPIA is set, but the listener/talker software flag bit (PIAIMG) is not set. This condition could occur using a DMA block transfer if the GPIA system controller readdresses...</td>
</tr>
</tbody>
</table>
Figure 13. Listener Mode Timing Diagram
6809 System Delay

GPIA Delay From Last Acceptance to New "Not Valid"

GPIA Request Delay From RFD (Emulator Box) to DMA Request

Box Response Time

Figure 14. Talker Mode Timing Diagram
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.