JANUARY 1975
Milt Schwartz
National Semiconductor

IMP-16L DMA – WHAT IT IS AND HOW TO USE IT

INTRODUCTION

This application note discusses the general theory of the IMP-16L DMA Bus; including timing, bus request, priority and expansion, bus controller operation, and peripheral interfacing techniques. Two examples of priority expansion and three examples of data transfers are given: CPU read from peripheral device, CPU write to peripheral device, and direct memory access by a peripheral device. Table I contains all signal nomenclature and definitions.

With direct memory access (DMA), memory and peripheral devices may communicate directly with one another using a common bus that is time multiplexed. This direct communication does not require the use of the central processor; therefore the processor may proceed with independent processing tasks. As a result, the computation speed of the system is effectively increased. Direct communication may occur between peripheral devices and memory, the central processor and memory, the central processor and a peripheral device, or any two peripheral devices. To allow this communication, all devices must be connected to a common bus by the user.

A three-bus structure is used to implement the IMP-16L DMA capability. This three-bus structure is referred to as the "System Data Bus," and consists of the following:

Data Bus: 16-bit bidirectional, utilizing TRI-STATE®. Time multiplexed for outputting address and inputting or outputting data.

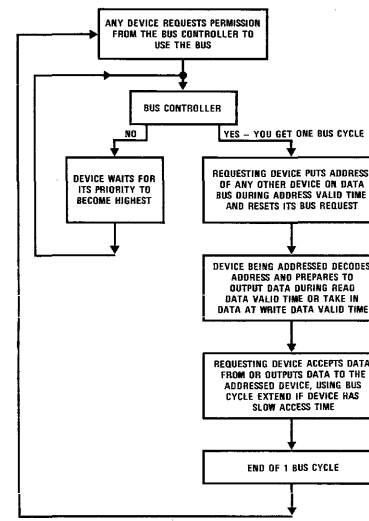
Timing Signal Bus: Clocks and data strobes.

Control Signal Bus: Priority operations, bus cycle extend, peripheral bus request and grant.

BUS OPERATION – GENERAL CONCEPT

As can be seen from Flow Chart 1, the basic idea of the bus structure is that any device tied to the bus whether it is CPU, memory or peripheral must ask permission from the bus controller (logic located on the CPU card) to use the bus for communicating with any other device.

The bus controller consists of an ordered priority circuit (which is expandable) and a timing and control circuit. The priority circuit logic decides what device in order of importance may use the bus, while the timing and control



Flow Chart 1

circuitry generates signals necessary for synchronous operation. (See Table I for signal nomenclature.) It is up to the user to choose which device should have highest priority. The user should exercise care when choosing priorities for devices to insure that all will be serviced within their response timing constraints.

BUS CONTROL

If peripheral devices are to communicate with one another, a bus-request signal must be generated (IBR*). This is an asynchronous command by the requestor over the control signal bus. A separate bus request line is provided for each device. In response to the request, permission to use the data bus for one bus cycle is granted by an ordered prioritizer and must be recognized synchronously by the requestor. (See *Figure 2.*) The user determines the order-of-priority assigned to his peripherals by hardwiring appropriate connections to the priority circuit *(Figure 3).* As soon as the requestor receives permission to use the bus, address and data must be available for transmission during the same bus cycle.

B15M15/Printed in U.S.A.

TABLE I. Signals Available to the User

NAME	WHERE GENERATED	DESCRIPTION
TIMING BUS SIGNALS		
ICLKA	CPU Card	Interface Bus Clock A, Same Phase as CLKA
ICLKA*	CPU Card	Interface Bus Clock A
ICLKG	CPU Card	Interface Bus Clock B, Same Phase as CLKB
ICLKB	CPU Card	Interface Bus Clock B
ICLKB*	CPU Card	Interface Address Data Strobe
IADS	CPU Card	Interface Address Data Strobe
IADS*	CPU Card	Interface Write Data Strobe, Same Phase as WDS on CPU
IWDS	CPU Card	Interface Write Data Strobe
IWDS*	CPU Card	Interface Read Data Strobe, Same Phase as RDS on CPU
IRDS*	CPU Card	Interface Read Data Strobe
BICLKB	CPU Card	Buffered Bus Clock B
CONTROL BUS SIGNALS		
IRMC*	CPU Card and User	Interface Read Memory Cycle, Open Collector
IWMC*	CPU Card and User	Interface Write Memory Cycle, Open Collector
IRPC*	CPU Card and User	Interface Read Peripheral Cycle, Open Collector
IWPC*	CPU Card and User	Interface Write Peripheral Cycle, Open Collector
IBHLD*	CPU Card and User	Interface Bus Hold, Input to Timing and Control Circuit, Used to Extend Bus Cycle, Open Collector
IBR0*	CPU Card	Interface Bus Request 0, Highest Priority Input
IBR1*	CPU Card	Interface Bus Request 1, Second Highest Priority
IBR2*	CPU Card	Interface Bus Request 2, Third Highest Priority
IBR3*	CPU Card	Interface Bus Request 3, Fourth Highest Priority
IPS0*	CPU Card	Interface Priority Select 0, Corresponds to IBR0*
IPS1*	CPU Card	Interface Priority Select 1, Corresponds to IBR1*
IPS2*	CPU Card	Interface Priority Select 2, Corresponds to IBR2*
IPS3*	CPU Card	Interface Priority Select 3, Corresponds to IBR3*
APVO*	CPU Card	Interface Approve Out (Note 1)
APVI*	CPU Card	Interface Priority Approve In (Note 1), Normally Connected to APVO* if no Priority Expansion
IAI	CPU Card	Interface Inverter A Input (Note 2)
IBI	CPU Card	Interface Inverter B Input (Note 2)
ICI	CPU Card	Interface Inverter C Input (Note 2)
IDI	CPU Card	Interface Inverter D Input (Note 2)
IAO	CPU Card	Interface Inverter A Output (Note 2)
IBO	CPU Card	Interface Inverter B Output (Note 2)
ICO	CPU Card	Interface Inverter C Output (Note 2)
IDO	CPU Card	Interface Inverter D Output (Note 2)
DATA BUS SIGNALS		
IDB00	CPU Card	Interface Data Bit 0
IDB01	CPU Card	Interface Data Bit 1
IDB02	CPU Card	Interface Data Bit 2
IDB03	CPU Card	Interface Data Bit 3
IDB04	CPU Card	Interface Data Bit 4
IDB05	CPU Card	Interface Data Bit 5
IDB06	CPU Card	Interface Data Bit 6
IDB07	CPU Card	Interface Data Bit 7
IDB08	CPU Card	Interface Data Bit 8
IDB09	CPU Card	Interface Data Bit 9
IDB10	CPU Card	Interface Data Bit 10
IDB11	CPU Card	Interface Data Bit 11
IDB12	CPU Card	Interface Data Bit 12
IDB13	CPU Card	Interface Data Bit 13
IDB14	CPU Card	Interface Data Bit 14
IDB15	CPU Card	Interface Data Bit 15

Note 1: Provide for expansion of DMA bus to more than four DMA devices.

Note 2: Spare inverters.

Note 3: The asterisk (*) indicates that the associated signal is an active low signal (that is, ICLKB and ICLKB* are complement signals; ICLKB is the active high, and ICLKB* is the active low).

(See timing requirements of *Figure 1* for address and data valid.) A bus cycle is normally about 1μs in duration. However, provisions are made to extend this time for slow responding peripheral devices or memory. (See *Figure 2.*) To extend the bus cycle, the user may generate a signal IBHLD* (interface bus hold). This signal must be a low level and must make the transition to the low level during the time that address data strobe (IADS*) is low and ICLKB* is high. The duration of the extended bus cycle may be up to 4μs and may asynchronously end with a low-to-high transition. By definition, the duration of a bus cycle is the time interval between the negative-going edge of IADS* and the positive-going edge of write data strobe (IWDS*) or read data strobe (IRDS*). A new bus cycle may be initiated simultaneously with the ending of the previous one. This condition is true whether or not a bus hold has been performed.

BUS PRIORITY

To gain control of the bus for one cycle, the requestor synchronously (with negative-going edge of ICLKB) sends a negative-going signal IBR* (interface bus request) to the bus controller on the CPU card. If this IBR* is the highest priority at the time, then a corresponding negative-going signal IPS* (Interface Priority Select) is immediately sent from the priority network *(Figure 3)* to the requesting peripheral device. The peripheral device must synchronously sample and latch the IPS* signal on the negative-going edge of ICLKB*. IBR* may go high as soon as IADS* is generated. The peripheral then must generate the correct signal for the type of bus transaction required, namely IRPC*, IWPC*, IRMC*, IWMC* *(Figures 1 and 6)*. If back-to-back cycles are desired, then the peripheral holds IBR* low through the complete cycle. Upon termination of the present bus cycle (positive-going edge of IRDS*), the IBR* of the same peripheral device immediately is recognized if it is still the highest priority. The bus cycle then repeats itself as previously described. Again, IBR* must be held low until the next IADS* is generated. No device which makes more than two back-to-back requests should be given higher priority than the CPU, as the CPU must be guaranteed access to the bus once every 8μs.

PRIORITY EXPANSION

Refer to *Figures 4 and 5* for expansion techniques. Note that whatever expansion techniques are utilized, a signal must be sent from the priority circuit to the bus timing control section of the CPU card. This signal line must be connected to the CPU card input pin labeled APVICPU*. When a logic 1, this signal indicates that a bus request (IBR*) has been generated.

PERIPHERAL INTERFACE TECHNIQUES

First Situation

The CPU is the BOSS (bus master) and requests data (reads) from a peripheral device using the RIN or Load^2 instructions. It is assumed that 8 bits of peripheral device address are necessary for this example and that the CPU is expecting 16 bits of data to be returned.

The following sequence of events occurs (refer to *Figures 1, 2, 3, 6 and 7*). The CPU generates an IBR*, if this is the highest priority at the time, a corresponding IPS* is

Note 1: IRPC*, IWPC*, IRMC*, IWMC* are used for decoding by memory and other peripherals.

*Denotes an active low signal.

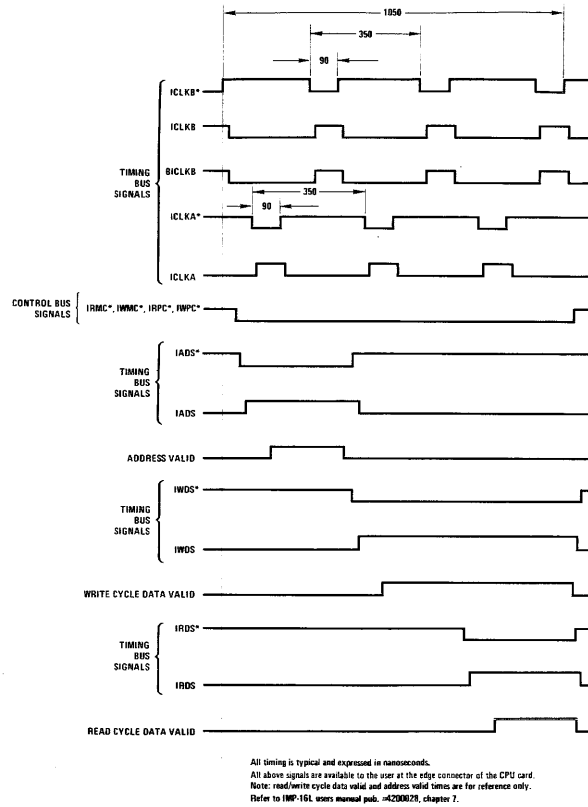


FIGURE 1. System Timing

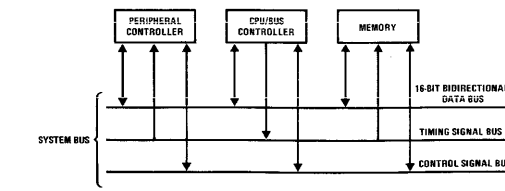
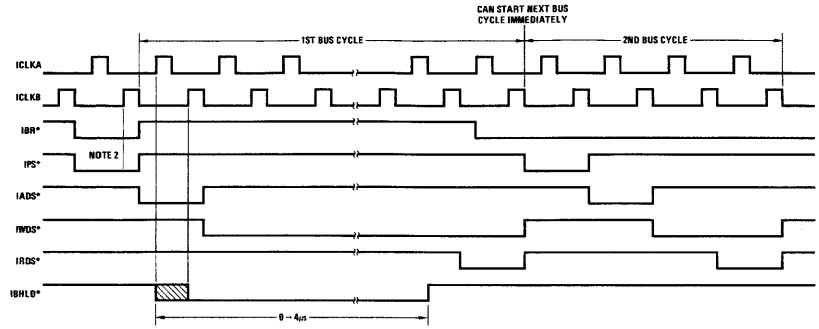


FIGURE 1A. System Data Bus Structure

generated by the priority network on the CPU card. The first CLKB (negative-going edge) after IPS* is generated causes signals IADS* and IRPC*, and simultaneously all IPS* signals are forced "high" (disabled). The bus cycle is initiated by the negative-going edge of IADS*. IADS* is used as a gating signal by the CPU to put the 8-bit address on the bus. The peripheral device addressed must decode the address bits and latch the decoded signal using IRPC*, IADS and ICLKB (positive-going edge). When the address is decoded, the peripheral device gates its data to the bus during IRDS time. The CPU gates the data in using IRDS and CLKA (positive-going edge). The bus cycle is terminated by the positive-going edge of IRDS*.

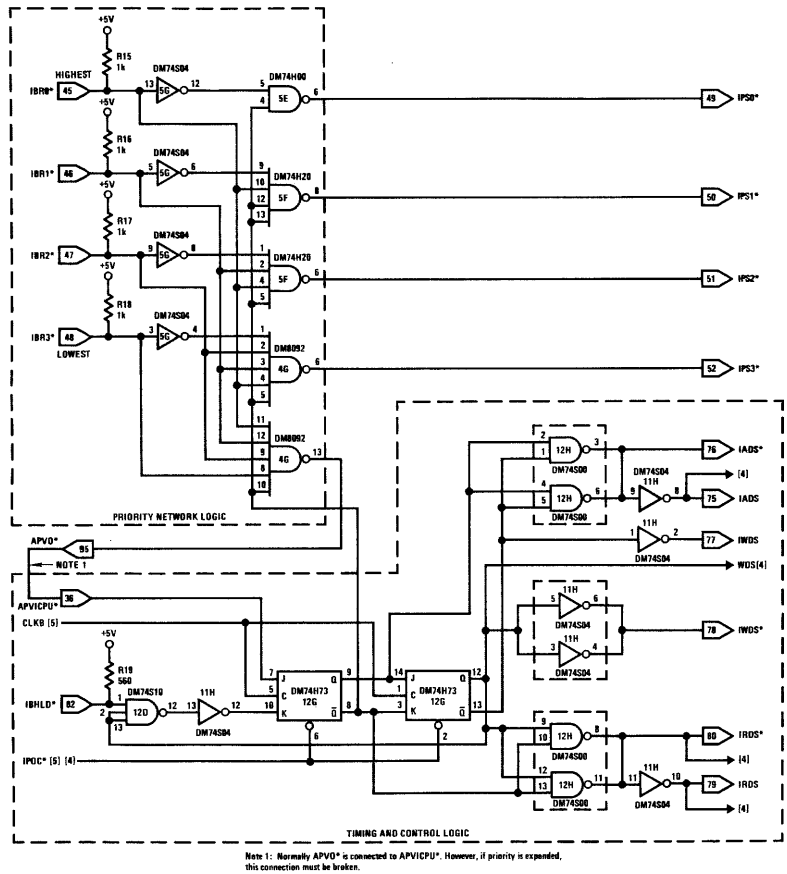
Second Situation

The CPU is the bus master and writes data to the peripheral using ROUT or Store[2] instructions. Eight bits of address are used and the CPU transmits 16 bits of data to the peripheral device. The following sequence of events occur (refer to *Figures 1, 2, 3, 6 and 8*). The CPU generates an IBR*. If this is the highest priority at the time, then a corresponding IPS* is generated by the priority network on the CPU card. The first CLKB negative-going edge after IPS* is generated causes signals IADS* and IWPC*, and all IPS* signals are simultaneously forced high (disabled). The bus cycle is initiated with the negative-going edge of IADS*. IADS* is used as a gating



Note 1: IBHLD* must not make a "1" to "0" transition while ICLKB is a "1."
Note 2: Requesting peripheral must recognize IPS* and use ICLKB to "Latch."

FIGURE 2. Bus Cycle Extend Timing



Note 1: Normally APVO* is connected to APVICPU*. However, if priority is expanded, this connection must be broken.

FIGURE 3. CPU Logic

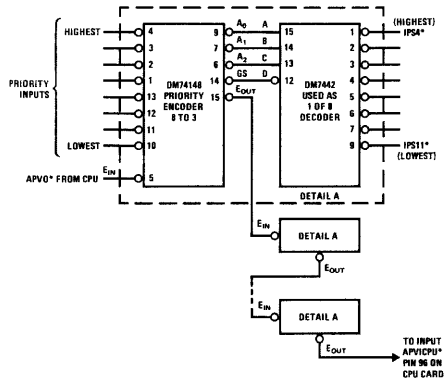


FIGURE 4. Priority Expansion Method No. 1

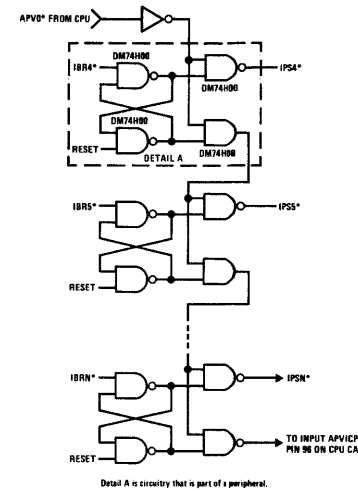
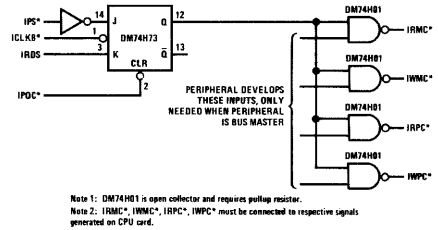


FIGURE 5. Priority Expansion Method No. 2



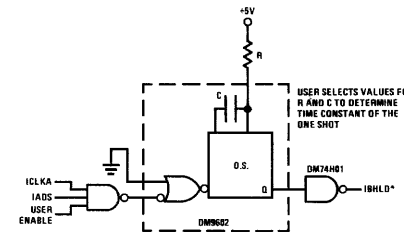


FIGURE 6. Possible Methods for Implementation of Interface Signals Generated by the Peripheral

signal by the CPU to put the 8-bit address on the bus. The peripheral device decodes the address and latches the decoded signals using IWPC*, IADS and ICLKB (positive-going edge). Next, the CPU gates the 16 data bits on the bus during IWDS* time using WDS and WRP as gating signals. (WDS occurs at the same time as IWDS*.) The peripheral device must take data from the bus using IWDS* and ICLKB (positive going edge). The bus cycle is terminated with the positive going edge of IWDS*.

It may be noted that the benefits of using memory reference instructions such as store, load, add, etc. rather than RIN, ROUT are speed and flexibility of data manipulation because all four accumulators may be used. To achieve these benefits memory and peripheral area locations should be allocated prior to system design.

Third Situation

The peripheral device is the bus master and communicates directly with memory. For this example, the peripheral device presents 16 address bits on the bus and expects 16 data bits in return. The following sequence of events occur (refer to *Figures 1, 2, 3, 6, and 9*). The peripheral

Note 2: If a load or store or any standard instruction other than RIN. ROUT is used then the peripheral address should be in an unused memory area.

device generates an IBR*, if this peripheral is the highest priority at the time; then a corresponding IPS* is generated by the priority network on the CPU card. The first CLKB after IPS* causes IADS* to be generated from the CPU. Simultaneously, the peripheral must generate an IRMC*. Again, all IPS* signals are forced high on the negative-going edge of the same CLKB signal that generated IADS*. Now, the peripheral device is the BUS master. The peripheral has the address bits ready in a TRI-STATE register awaiting IADS* as a gating signal to output the address over the bus to memory. Memory latches the address and outputs 16 bits of data on the bus during IRDS. The peripheral device must be ready to clock in the data using IRDS and ICLKB (positive-going edge). The bus cycle ends simultaneously with the positive-going edge of IRDS*.

CONCLUSION

As can be seen from the preceeding examples, the direct memory access feature (DMA) of the IMP-16L affords easy interface to peripherals using standard TTL logic elements. The advantages of the DMA feature are high throughput per bus cycle, straight forward decoding at the peripheral, capability of the central processor to proceed with independent processing tasks, ease of interface to floppy discs, conventional discs, and nine channel magnetic tapes.

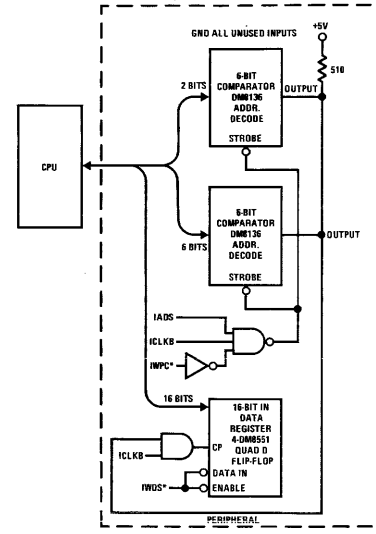


FIGURE 8. Typical Interface for the Case where CPU Writes to Peripheral

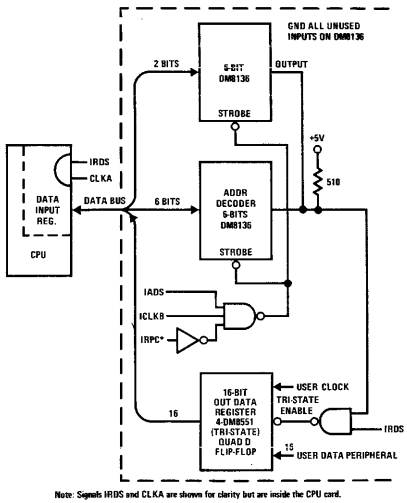


FIGURE 7. Typical Interface for the Case where CPU Reads Data from the Peripheral

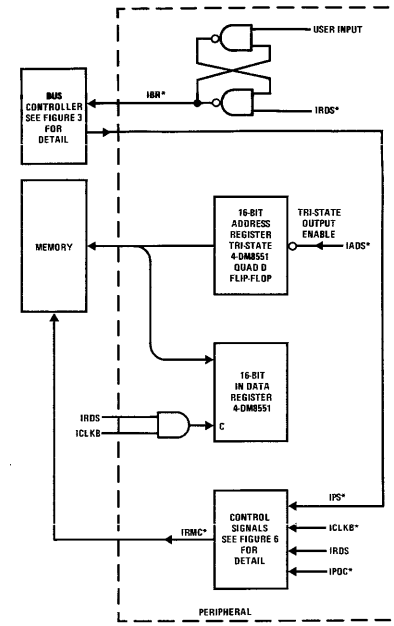


FIGURE 9. Typical Interface Circuit for a Peripheral Communicating Directly with Memory

Manufactured under one or more of the following U.S. patents: 3083262, 3189758, 3231797, 3303356, 3317671, 3323071, 3381071, 3408542, 3421025, 3426423, 3440498, 3518750, 3519897, 3557431, 3560765, 3566218, 3571630, 3575609, 3579059, 3593069, 3597640, 3607469, 3617859, 3631312, 3633052, 3638131, 3648071, 3651565, 3693248.

National Semiconductor Corporation
2900 Semiconductor Drive, Santa Clara, California 95051, (408) 732-5000/TWX (910) 339-9240
National Semiconductor GmbH
808 Fuerstenfeldbruck, Industriestrasse 10, West Germany, Tele. (08141) 1371/Telex 05-27649
National Semiconductor (UK) Ltd.
Larkfield Industrial Estate, Greenock, Scotland, Tele. (0475) 33251/Telex 778-632