Section 2
System Architecture

SYSTEM BUS

Each Level 6 Model has a system bus, of which there are two different kinds. The Model 23 utilizes a dual, synchronous bidirectional bus while the Models 33, 43, 47, 53, and 57 have a single, asynchronous, split-cycle bus (Megabus). Megabus systems are field-upgradable from one model to the next. Figure 2-1 provides system architecture diagrams.

CONNECTABLE UNITS

There are three principal types of bus-connectable (plug-in) units in all Level 6 systems: central processors, input/output controllers, and memories. One or more of each type can be included in a system. All are interconnected via the Level 6 system bus.

Up to 11 connectable units are supported by the Model 23 and up to 23 connectable units are supported by the Models 33, 43, 47, 53, and 57. In this context, a connectable unit generally is synonymous with a board and can be any one of the following:

- o Central Processor
- o Memory Controller with 8K to 128K 16-bit words of memory
- o Multiple Device Controller
- o Other Peripheral Controllers
- o Multiline Communications Processor
- o Scientific Instruction Processor
- o Commercial Instruction Processor
- o General Purpose DMA Interface

Several controllers of the same type may be included in a system. Each is identified by a unique module address which is easily field-adjustable.

Figure 2-2 illustrates a typical Model 43 system with memory management, 96K words of memory, 512 megabytes of disk storage, a console, a line printer, a card reader, 12 CRTs, and a communications link to a host. All of these elements fit into eight of the ten available Megabus slots. Note that if High-Density MOS Memory were used in place of the standard type, only six Megabus slots would be used instead of the eight shown. If further expansion is required, additional chassis can be added until the maximum 23-module capacity is reached.

INPUT/OUTPUT CHANNELS

Input/output controllers are used to control one or more peripheral devices or communications lines. A multiline communications controller can, for example, handle up to 8 full-duplex lines or 16 simultaneous input/output paths. Each path is considered to be a channel and will have a unique channel address. The logical capacity of the system is 1024 channels. Every device will have at least two channels assigned to it, even though both are not always used. Odd numbered channels are used for output devices, even numbered channels for input devices. Thus, a line printer requires two channel addresses, but only the odd numbered channel will be used. Two diskette units on a single controller would have four channel numbers assigned, one for each device in each direction.

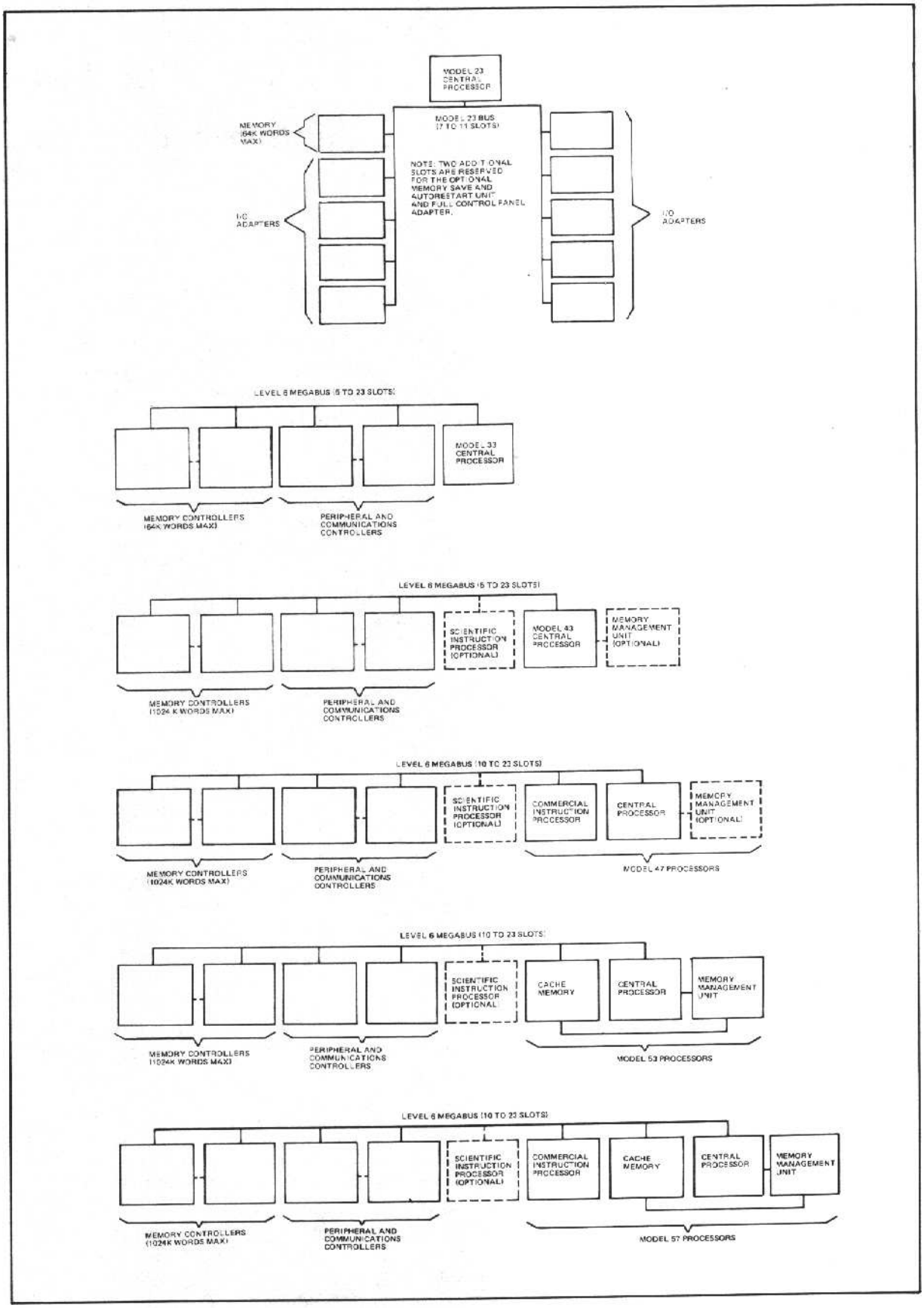


Figure 2-1. System Architecture

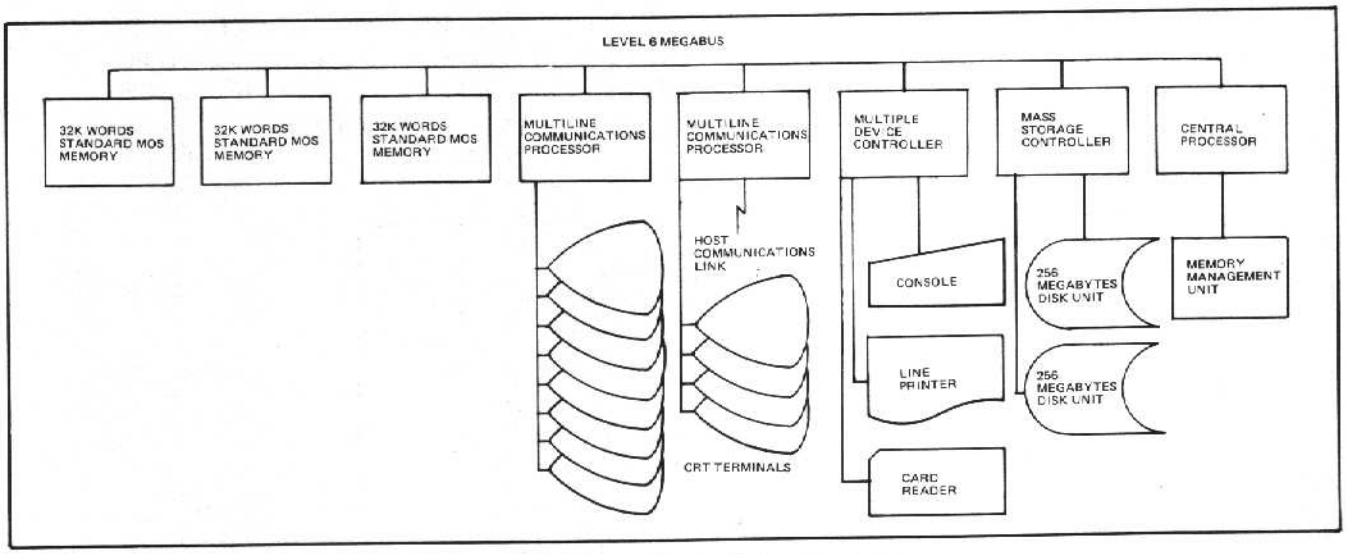
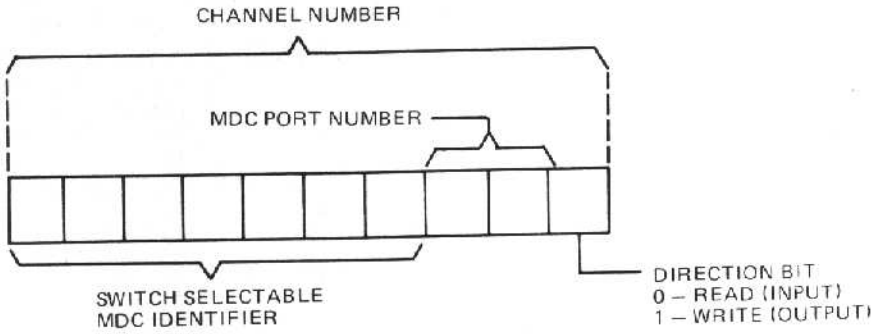


Figure 2-2. Typical Model 43 Configuration

The channel number for each data path is a ten-bit value. Of these the high-order bits are factory-assigned by means of an adjustable rotary switch which is easily field-changeable. The low-order bits identify the "port" on the controller, with the least significant bit the direction bit – odd for output, even for input. A typical MDC device channel number is shown below.



Multiple devices of the same type can thus be assigned to different channels. Channel numbers are a means of addressing input/output units and have nothing to do with either bus priority or interrupt priority levels.

INTERUNIT COMMUNICATION

Besides differing in its basic architecture from the rest of Level 6, the Model 23 also differs in its interunit communication. The Model 23 peripheral adapters (i.e., console, printer, and diskette) execute all data transfers in the Data Multiplex Control (DMC) mode while the Communications Adapter executes its data transfers in the Direct Memory Access (DMA) mode. The peripheral and communications controllers for the Model 33 and larger models, however, *all* operate in the DMA mode. In DMC operation, the CP must control the data transfer operation, whereas in DMA operation, after the channel is set up by the CP, data transfers are effected independently of the CP. The remaining portion of this section is primarily concerned with the Megabus architecture and the DMA mode of operation.

If a processor wants to store a word in memory, it sends that word together with its memory address down the Megabus to the memory. In this case, it acts as the master and the memory acts as the slave. The transfer of the information from a master unit to a slave unit takes a single *bus cycle*. There are several types of bus cycles, and various units can become masters and various units slaves. The bus cycle described above is called a write cycle and is pictured in Figure 2-3.

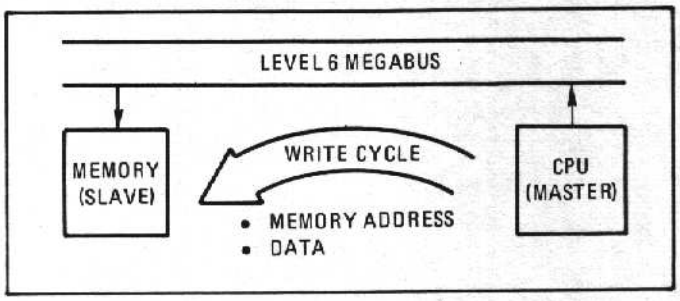


Figure 2-3. Write Cycle

The bandwidth of the bus is 6 million bytes per second, a bus cycle taking approximately 300 ns. The bus is asynchronous in design, permitting units of varying speeds to operate effectively on the same system. The extremely high speed of the bus allows a great many operations to be multiplexed, thereby giving a high degree of overlap among various system elements.

Megabus Priority

Any unit on the Megabus can become the master. To do so, it must have a higher priority than any other unit simultaneously also trying to become a master. Bus priority is dependent upon the relative position of the unit on the bus. Memories have the highest priority, intervening units have lesser priority, and the central processor has the lowest. While this is shown in Figure 2-4 as the highest priority on the left, in actuality the memories are physically at the bottom of a unit and the central processor is at the top. Therefore, the highest priority devices are those toward the bottom.

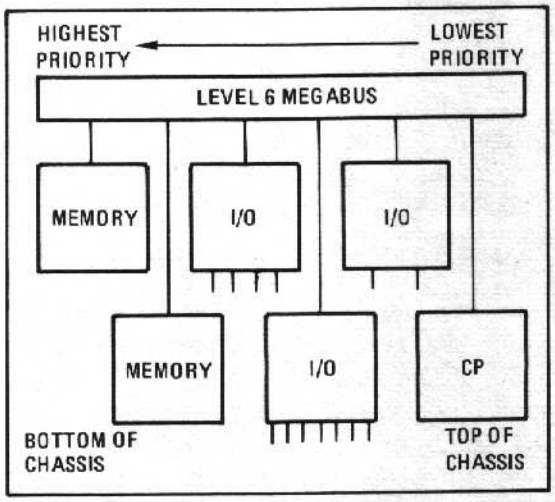


Figure 2-4. Bus Priority

Types of CP and Memory Transfers

The write cycle shown in Figure 2-3 is one type of transfer between a central processor and memory. A write requires a single bus cycle. Another common operation occurs when the central processor wants to read a word of data from memory during either an instruction fetch or an operand fetch. A read operation requires two bus cycles: a read request cycle and a read response cycle. During a read request cycle, the central processor is the master and the memory the slave. The central processor sends a memory address plus its own processor channel number to the memory. Accompanying this transfer is a signal that says a response is requested. Upon receipt of this request, the memory accesses data and initiates a response cycle. For this cycle the memory becomes the master and the CP becomes the slave. The memory puts the data, together with the central processor channel number, on the bus and the CP accepts it.

Figure 2-5 illustrates a typical read operation for a processor with: 1) a single-fetch memory and 2) a double-fetch memory. As can be seen, the single-fetch memory read operation is a two-bus cycle operation, while the double-fetch memory read operation is a three-bus cycle operation and uses two memory cycles. In either case, both units will appear busy to any other unit that tries to address them during this time – with the exception of the central processor, which can accept interrupts from other units, request other cycles, or receive responses between the read request and the read response bus cycles.

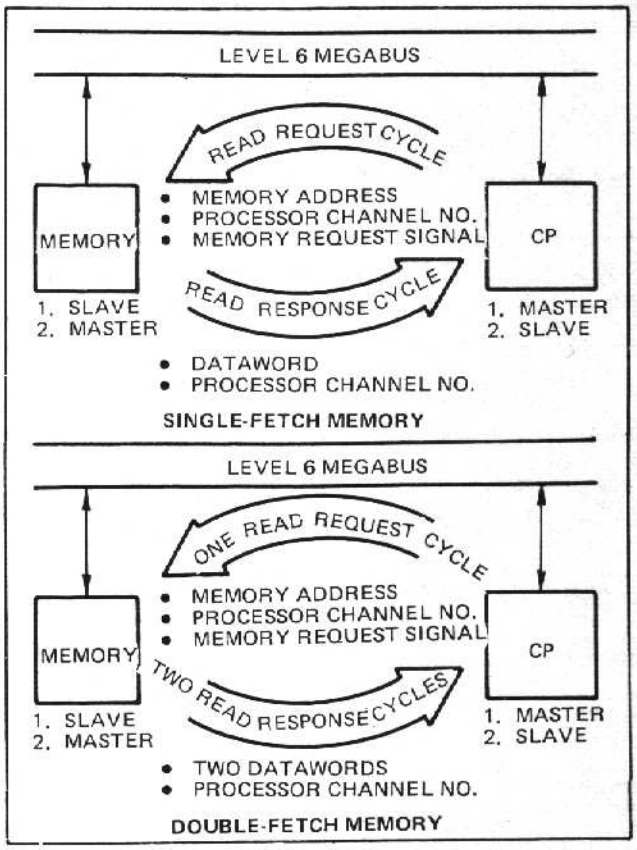


Figure 2-5. Read Operation

The memory does not initiate the read response cycle until after it has accessed the data. This is done during a 650-ns memory cycle (for single-fetch memory) which partially overlaps the two bus cycles (see Figure 2-6); or two overlapping 550-ns memory cycles (for double-fetch memory) that partially overlap three bus cycles. During this time the bus is free to accept requests from other units, interleaving bus cycles and effectively overlapping operations.

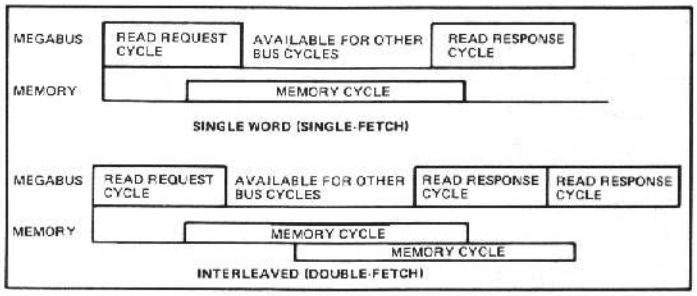


Figure 2-6. Timing of Memory Read Operation

Certain central processor to memory operations operate in read-modify-write mode and require three cycles: a read request, a read response, and a write cycle. Among these are a group of central processor instructions specifically designed to support multiprocessor operation in that initiation of these instructions locks memory for the full three cycles.[1] This feature allows one processor to set flags in memory to be tested by another processor, without the chance of the second processor testing the flags while the change is taking place.

Input/Output Transfers

All I/O transfers occur in direct memory access (DMA) mode. Once a channel is set up by the central processor, data transfers are effected via the bus independent of the central processor. When a channel wants to input a word to memory, it becomes the master and initiates a write cycle transmitting the data and the memory address to the memory (Figure 2-7). If the channel is connected to an output device and it wants to receive a word from memory, it initiates a read request cycle and then, after the memory has accessed the data, a second bus cycle, the read response, is initiated with the memory as master and the channel controller as slave (Figure 2-8).

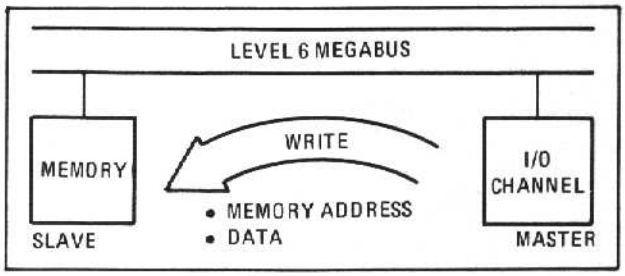


Figure 2-7. I/O DMA Input Operation

[1] The six CP instructions that "lock" memory are INC, DEC, LBF, LBT, LBC, and LBS. (See Section 4 for details.)

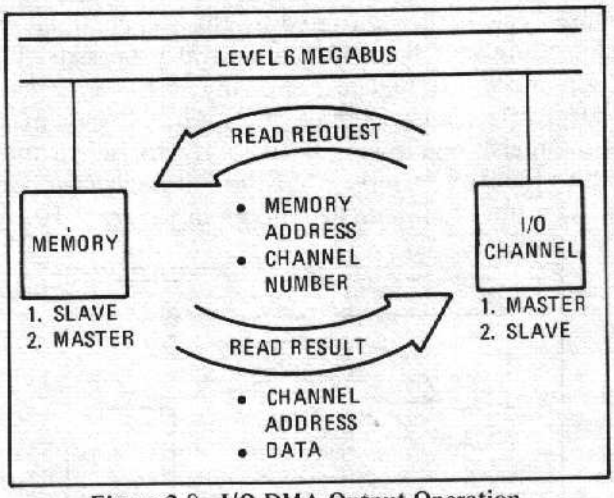


Figure 2-8. I/O DMA Output Operation

Multiple I/O units (including those on the same controller) can be operating simultaneously and can multiplex on a word basis to the same memory as long as the combined data rates do not exceed the maximum memory transfer rate of approximately 1.1 million words per second. As this rate is approached the central processor is locked out since I/O channels have a higher bus priority than the CP. If a second (or third) memory controller is added, true overlap can be attained, with the central processor communicating to one memory unit at the same time as the input/output channel is communicating to the second memory unit.

Input/Output Commands

In order for an input/output channel to initiate a DMA block transfer, it must first be set up by a central processor. This is done via an I/O output command. The central processor sends several words of information to the I/O controller, taking a single I/O write cycle to send each word (see Figure 2-9). The processor sends a word of information together with a channel number and a function code defining what that word of data is. Typical words that must be sent from a central processor to a controller are *task* words identifying the operation to be performed, *configuration* words showing such things as the track and sector of a disk file, *address* words showing the address in memory where the transfer is to start, and *range* words showing how many words or bytes of data are to be transferred.

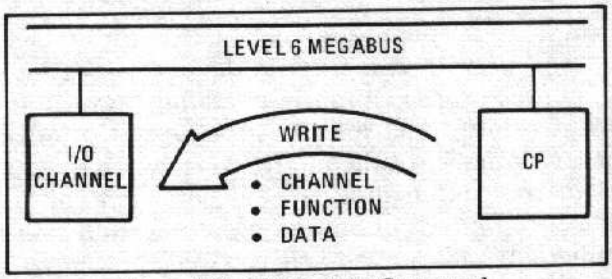


Figure 2-9. I/O Output Command

In addition to I/O output commands, the processor can initiate I/O input commands. These are two-cycle operations in which the processor issues a read request to the I/O channel, sending it both a channel number and a function code, plus its own processor channel number. The read response cycle is then initiated by the I/O channel, with the contents of a particular channel register being sent back to the central processor. Typical registers which are read by the processor are *status* registers showing status and error information, *range* registers typically read after a DMA transfer to determine how many characters have been read in, and *identification* registers showing a fixed ID for a particular device type. The latter allows the software to identify exactly what type of device is on a particular channel.

The two cycles of an I/O input command are shown in Figure 2-10.

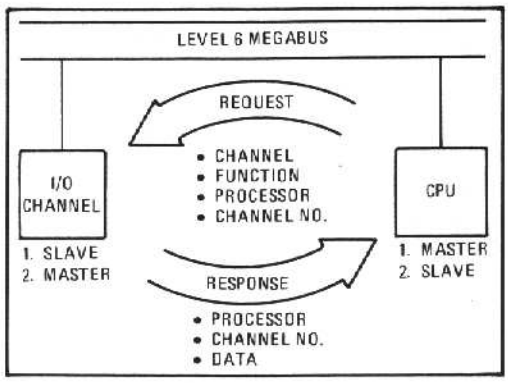


Figure 2-10. I/O Input Command

Interrupt Levels

One I/O output command in particular is important at this point. This is the output interrupt control function. Utilizing this command, a processor can assign an interrupt level to a particular channel. There are 64 interrupt levels. Any channel can be assigned any interrupt level by the central processor. Several can have the same interrupt level if necessary. To assign an interrupt level the processor executes an I/O output command which transmits a data word containing the interrupt level and the CP channel number to the I/O channel. This then becomes the interrupt level at which that channel may in the future interrupt. It should be noted that priority levels are thus software-assigned and are completely independent of physical position.

Interrupt Commands

When a channel wants to interrupt because of a certain condition in the controller (typically the end of a block transfer), the channel initiates an interrupt cycle (see Figure 2-11). In this cycle, it places its channel number and its interrupt level on the Megabus and attempts to transmit these to a processor (not necessarily the same one which had previously set up the interrupt level). A register in the processor (the S register) defines the priority level at which the processor is currently operating. Priority levels are assigned such that level 63 is the lowest priority and level 0 is the highest priority (reserved for power failure interrupt). If the level number of the interrupting device is numerically lower (higher priority) than the level of the central processor, an interrupt will occur. If not, then the central processor will reject the interrupt and no more interrupt cycles will be placed upon the bus by the I/O channel until it receives a signal from the central processor stating that the priority level of the central processor is changing and that it should thus reinitiate the interrupt request.

An interrupt request can be received by the central processor at any time, including the time between the two cycles of a read operation, although it will not be processed until the instruction is complete.

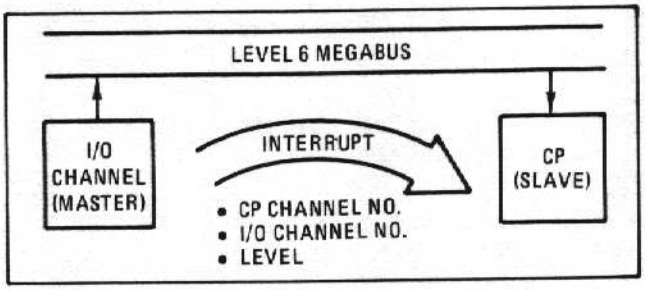


Figure 2-11. Interrupt Cycle

Interrupt Action

When an interrupt is accepted, the central processor typically assumes the level of the interrupting source. It remains at this higher priority level until interrupted by some still higher priority level, or until the CP finishes the interrupt routine and executes an instruction (the LEV instruction) that returns control to either the previously interrupted level or to an intervening level.

Consider the following (see Figure 2-12):

1. Processor running at level 40 sets up devices A, B, and C at interrupt levels 20, 30, and 25, respectively.
2. Device B requests interrupt and is accepted. Processor now runs at level 30.
3. Device A requests interrupt and is accepted. Processor now runs at level 20.
4. Device C requests interrupt but is not accepted. It will not try again until processor changes levels.
5. Processor finishes processing level 20 and issues LEV instruction. This allows C to request interrupt again; it is now accepted and processor runs at level 25.
6. Processor finishes processing level 25 and issues LEV instruction; no higher interrupts are pending, so it resumes level 30 processing.
7. Processor finishes level 30, issues LEV, and goes back to original level 40.

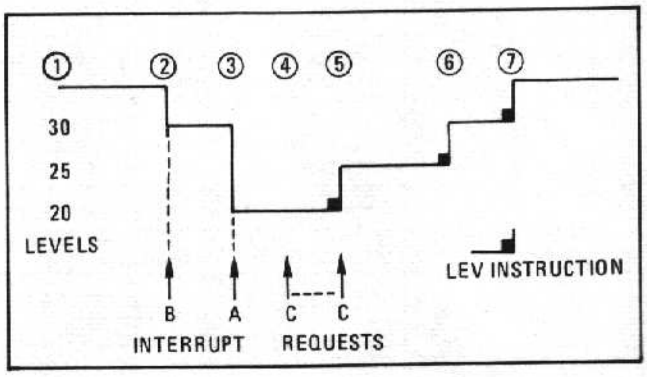


Figure 2-12. Interrupt Action

Context Switching

Each interrupt level has its own individual context save area pointed to by a dedicated vector. Upon interrupt, the CP firmware (after clearing all pending traps at the running level) automatically stores the contents of all active registers in the context save area of the interrupted level and loads the registers as specified for the interrupting level. The number of registers whose context is saved and restored is under the control of a mask for each interrupt level, with the mask set up by the software. Thus, anywhere from 2 to 18 (for Model 33) or 2 to 31 (for larger models) registers may be automatically switched upon initiation of a new level. (For details on registers and the action of interrupts, see Section 3.)

SUMMARY OF BUS OPERATIONS

Figure 2-13 sumarizes the various Megabus operations described. The Megabus itself contains the following 51 information signals:

- o 24 address bits
- o 16 data bits
- o 6 control bits
- o 5 integrity bits

There are also 17 other lines used for timing and other functions. Data lines typically hold a 16-bit data word or, on data requests, the "return address" of the requesting channel. The address lines contain either the channel number of the destination or a memory address as designated by one of the control lines. It should be noted that memory addresses are not

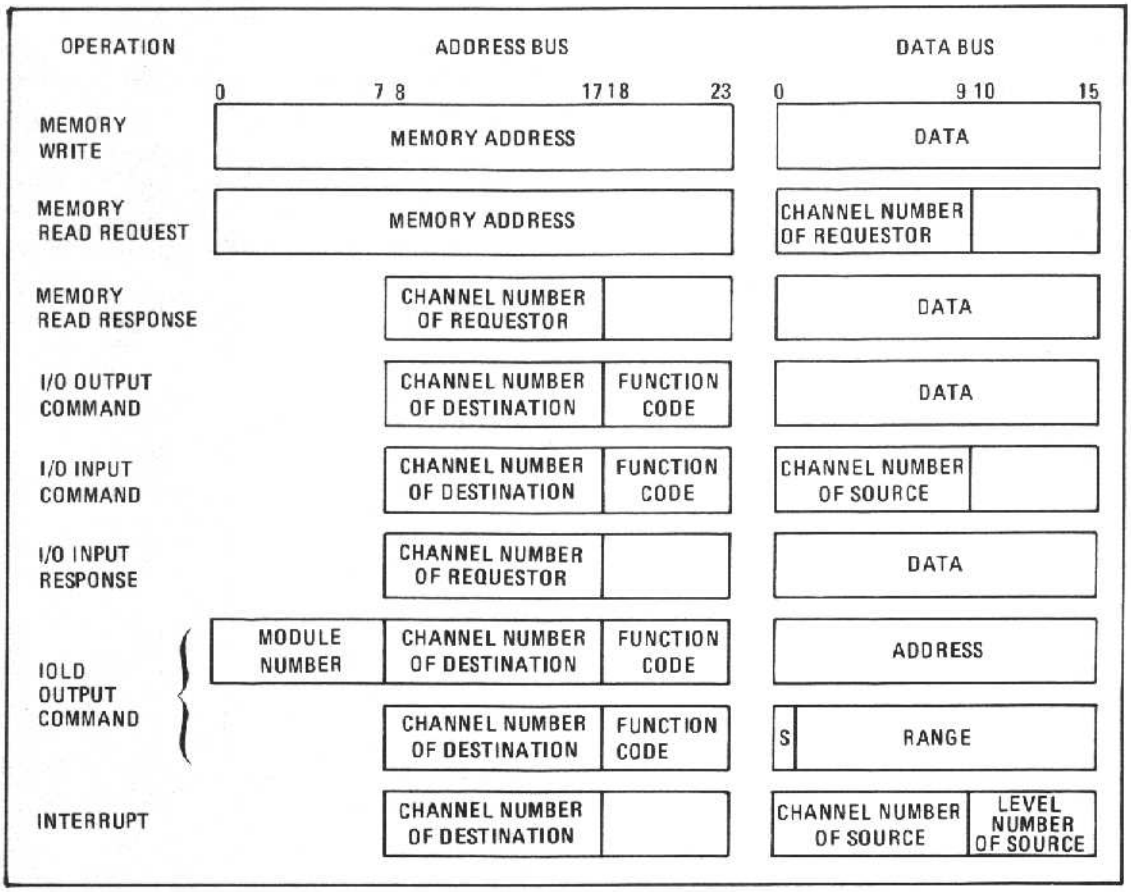


Figure 2-13. Data and Address Bus Formats

used to specify registers in various I/O channels. The latter are specified by a 10-bit channel number and a 6-bit function code, leaving the full range of possible memory addresses for addressing memory. A 32K system will have all 32K words of memory addressable and a 128K system will have 128K words addressable, etc.

MEMORY ADDRESSES

The basic unit of addressing is a word. A Model 23 or 33 has 16-bit address registers; therefore, it can address 64K words of memory. For byte addresses, a 17th bit is appended and, therefore, it can directly address 128K bytes of memory. This is also the way a Model 43 or larger model works in the short address format (SAF) mode. The Model 43 or larger model, however, has 20-bit address registers. When in the SAF mode, only 16 bits of these registers are utilized. When in long address format (LAF) mode, all 20 bits are used for word addressing. Thus, the Model 43 can directly address 1 million words of main memory. Again, for byte addresses an extra bit is appended. In this case, a 21-bit address results, allowing the Model 43 to directly address 2 million bytes of memory.

The Level 6 Megabus has 24 address lines utilized to address memory to the *byte* level. Thus, the Megabus has an architecture that is "open-ended" in that it can support processors with a direct addressing capability of over 16 million bytes. Of these 24 lines, 17 are actually used by Model 33 systems and 21 by Model 43 and larger systems.

On word addresses, a 17-bit address is always generated in a Model 23 and 33 or in a larger model operating in SAF mode. A Model 43 operating in LAF mode will generate a 21-bit address. If a byte-oriented instruction is being executed, the CP informs the memory via a control line on the Megabus (otherwise the memory assumes a word address). The memory then uses the low-order Megabus address bit to determine which byte of a word should be written into (the remaining byte will not be altered). The low-order bit is always ignored for memory reads, the entire word is returned, and the receiver determines which byte to use.

Through the use of indexing, 17-bit byte addresses are generated in SAF mode, 21-bit in LAF mode. A byte count in an index register is shifted one place to the right before being added to a 16-bit word address of a Model 23 and 33 or a 20-bit word address of a Model 43. Thus, if an index register contained the quantity 5 and a word address of 1000 was specified, the fifth byte (starting at zero) from the left-half of word 1000 would be selected or the right byte in word 1002 (see Figure 2-14). Notice that byte addresses are sequential (left to right) in memory and that data is thus in its correct sequence.

A similar scheme is used with *bit instructions* to address bits; in this case a bit count is shifted four places to the right prior to being added to a word address.

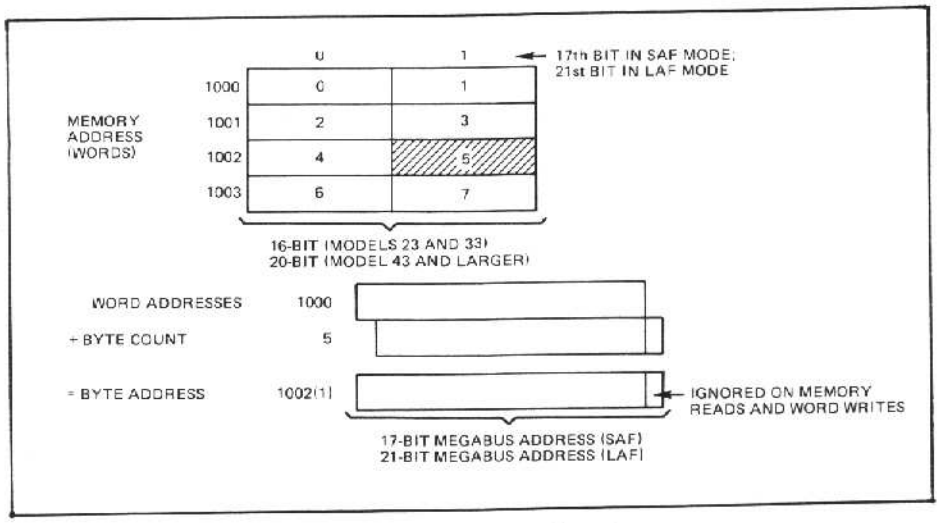


Figure 2-14. Byte Addressing