```
+555555555555555555555555555555555555555555555555555555555555555555555555555555+

                               $$$$$   $$$$   $$$
                               $      $      $   $
                               $      $      $   $
                               $$$$   $      $   $
                               $      $  $$  $   $
                               $      $   $  $   $
                               $$$$$   $$$$   $$$

                        USER=EGO  QUEUE=LPT  DEVICE=@LPA
              SEQ=11  QPRI=127  LPP=63  CPL=80  COPIES=1  LIMIT=13

                          CREATED:  28-MAR-78  10:42:18
                         ENQUEUED:  28-MAR-78  16:22:30
                         PRINTING:  28-MAR-78  16:22:33

                              PATH=:UDD:EGO:IOSW.LS

                 $$$$$   $$$    $$$   $   $         $       $$$
                   $    $   $  $   $  $   $         $      $   $
                   $    $   $  $      $   $         $      $
                   $    $   $   $$$   $ $ $         $       $$$
                   $    $   $      $  $ $ $         $          $
                   $    $   $  $   $  $$ $$    $$   $      $   $
                 $$$$$   $$$    $$$   $   $    $$   $$$$$   $$$

+555555555555555555555555555555555555555555555555555555555555555555555555555555+


                               AOS XLPT REV 01.00
```

1 Introduction

The Input/Output system of the MTA is considered in two parts. The logical part, that is the structure as seened by the programmer and as defined by the instruction set. The other part is the physical part, that is the structure of the busses. This physical structure includes bus protocols, electrical and timing characteristics, and other physical attributes. Both types are discussed independently in this chapter.

2 Logical I/O Structure

The foundation of the logical I/O structure is to provide a mapping from the logical address space to the I/O device controllers. This structure is extended by mapping device controller registers into the logical address space.

This mapping functions in the following manner. The MTA logical address is translated into a physical address. The resulting physical address references 1 of 2 subsystems. One of these subsystems is main memory. Main memory is indicated when the most significant bit of the physical address is a 0. The subsystem is I/O. I/O is indicated when the most significantt bit of the physical address is a 1. The physical address produced references a device controller, registers contained in the device controller, and control flags.

3 Nova/Eclipse Compatibility

Presently, the MTA physical I/O structure is defined to be the structure of the Nova/Eclipse. Controllers which adhere to this structure are referenced when the following physical address is produced.

```
0              11 12             17 18           28
     --------------------------------------------------
     I 1 0 0 ... 0 0  I DEVICE  CODE   I SPECIFIER    I
```

---

where bits 0-11 indicate the Nova/Eclipse Bus

where bits 12-17 indicate one of 64 devices

where bits 18-28 (the page offset field of the logical address) indicates the device controller's register and function field. In particular bits 26-27 specify:

00-status
01- A register
10-B register
11 - C register.

Bit 28 is a don't care.

and bits 18 - 19 specify:
00 - idle
01 - start
10 - clear
11 - pulse

The type of reference, that is read or write is a function of the specification of the presently executing instruction. When the logical address whose translated addresss is a physical 1/0 address is used a source reference (data is to be read from the device), a DATA IN request is initaited. When the logical address is used as a destination reference (i.e., data is to be written to the device), a DATA OUT request is initiated to the specified device.

4 Other I/O considerations

The following sections discuss some of the secondary issues and presents specifications as to the reaction of the I/O system to ill-defined or ill-conceived operations.

Physical Address bits 0-11

Physical address bits 0-11 being 1 0 0 ... 0 0 denote the Npva/Eclipse bus. Thjese bits are controlled by the system software. They are determined as part of the logical to physical address translation. These bits are NOT 1 0 0 ... 0 0 due to either of two reasons, One reason is that tsystem software incorrectly specified the field. The other reason is that an additional I/O bus that is not the original Nova/Eclipse bus exists. In the latter case, if such an additionoal bus exists, the interface to the original Nova/Eclipse bus must recognize 1 0 0 . . . 0 0 in bits 0 -11 as its address. In implementations with only one I/O bus, this recognition is optional. That is, a field with a value other than 1 0 0 ... 0 0 has no meaning and is an error.

Device Code bits 12-17

There is no architectureal specification as to the response to a command transmitted to a non-existing device. Simply stated addressing a non-existing device is a programming error.

Specification Field bits 18 - 28

The interpretation of the specification field follows the same rules as would be the case on devices attached to the Nova/Eclipse bus. Bits 18-19, the F field ,and bits 26-27, the register field are interpreted as appropriate for the specified device. Bits 20-25, & 28 are don't care. That is their value has no impact on the performance of the specified command on the specified device.

Operand Length Consideration

The most intriquing or confusing issue is the interpretation for a request for an operand that does not match the contraints of the I/O system. For example, consider the instruction <move- word> <A> <B> (move 32 bits) where <A> referecnes device 5, the A register. Is register A and B fetched, is just register A fetched, is an error conditon signalled, or is the response implementation dependent.

In deciding which of these alternatives is appropriate, it is necessary to step back and analyse the meaning of an operand reference not equal to 16 bits. Mapping the I/O system into the logical address space while offering substantial flexibility and possible future augmentations also allows accesses in a manner that has no meaning, or never had meaning in the Nova/Eclipse world. This is true since non 16 bit references could not be performed by the Nova/Eclipse. Thus, I believe to impose meaning on an otherwise meaningless (or certainly for the vast majority of time) is a fool's errand. I recommend the last alternative, that is the operand length is a don't care with respect to the interpretation of the I/O command.

Thus if a <move - word><A><B> is specified and the source operand is a I/O register, the 32 bit operand happens to be whatever 32 bits are developed when the 16 bits of the specified register is placed on the CPU memory bus. Conversely, when an I/O register is a destination the least significant 16 bits of the operand placed on the memory bus is used for the DATA OUT. In other words the interface between the CPU memory and the Nova/Eclipse I/O system is 16 bits wide. In particular the least significant 16 bits of the bus. Adopting this protocal works equally well for move or move double float instructions.