SERIES 60 (LEVEL 6)
MINICOMPUTER HANDBOOK
ADDENDUM A

SUBJECT

Additions and Changes to the Handbook

SPECIAL INSTRUCTIONS

This is the first addendum to AS22, Revision 2, dated September 1977. Insert the attached pages into the handbook according to the collating instructions on the back of this cover. Change bars in the margins indicate technical changes and additions; asterisks denote deletions. These changes will be incorporated into the next revision of the handbook.

Note:

Insert this cover after the handbook cover to indicate updating of the document with Addendum A.

ORDER NUMBER

AS22A, Rev. 2

January 1978

20914
3.5578
Printed in U.S.A.

Honeywell

COLLATING INSTRUCTIONS

To update the manual, remove old pages and insert new pages as follows:

Remove	Insert
3-7, 3-8	3-7, 3-8
-	3-8.1, Blank

File No.: 1S03

1/78
AS22A

INTERRUPTS

There are 64 levels of interrupt, numbered from 0 to 63; level 0 has the highest priority. clearing the computer puts it into level 0, which in effect makes it uninterruptable.

Associated with each interrupt level is a dedicated memory location which contains the interrupt vector. These locations, extending from address 0080 to address 00BF (00FF for 6/40), contain a pointer to an interrupt save area. The interrupt save area needs only to be set up by those levels that are active in a particular program. Each interrupt save area has five fixed locations and up to 16 more variable locations. These locations are as follows:

- DEV – this location contains the channel number (bits 0-9) and level (bits 10-15) of an interrupting device.
  NOTE: DEV is not updated when a level is activated by an internal condition such as a power failure (level 0), watchdog timer (level 1), trap save area overflow (level 2), real-time clock (assigned level), or execution of a LEV instruction. Interrupt levels 1, 2, and the assigned real-time clock level should not be used for devices. If these levels are assigned to devices, or if multiple devices have been assigned to the same interrupt level, software should clear DEV before each I/O operation that will interrupt. This ensures that the interrupted level can determine the interrupt source.
- ISM 1/2 – these two locations contain the 32-bit interrupt save mask. This mask controls which of the registers will be saved in the variable portion of the interrupt save area.
- P – in this word the program counter of the interrupt level is stored. It also acts as a pointer to the interrupt handling procedure for a new level, or upon restoration of an interrupted level, to the location of the next instruction to be executed.
- S – this is where the status register is automatically stored. Note that when a new interrupt level is set up, the S register is loaded from this location only as far as the privileged mode field is concerned; the level is generated automatically, and the processor ID is hard-wired.

Words 6–21 are locations for saving the machine registers under control of the interrupt save mask. If the interrupt save mask is all zeros, none of these words will be reserved.

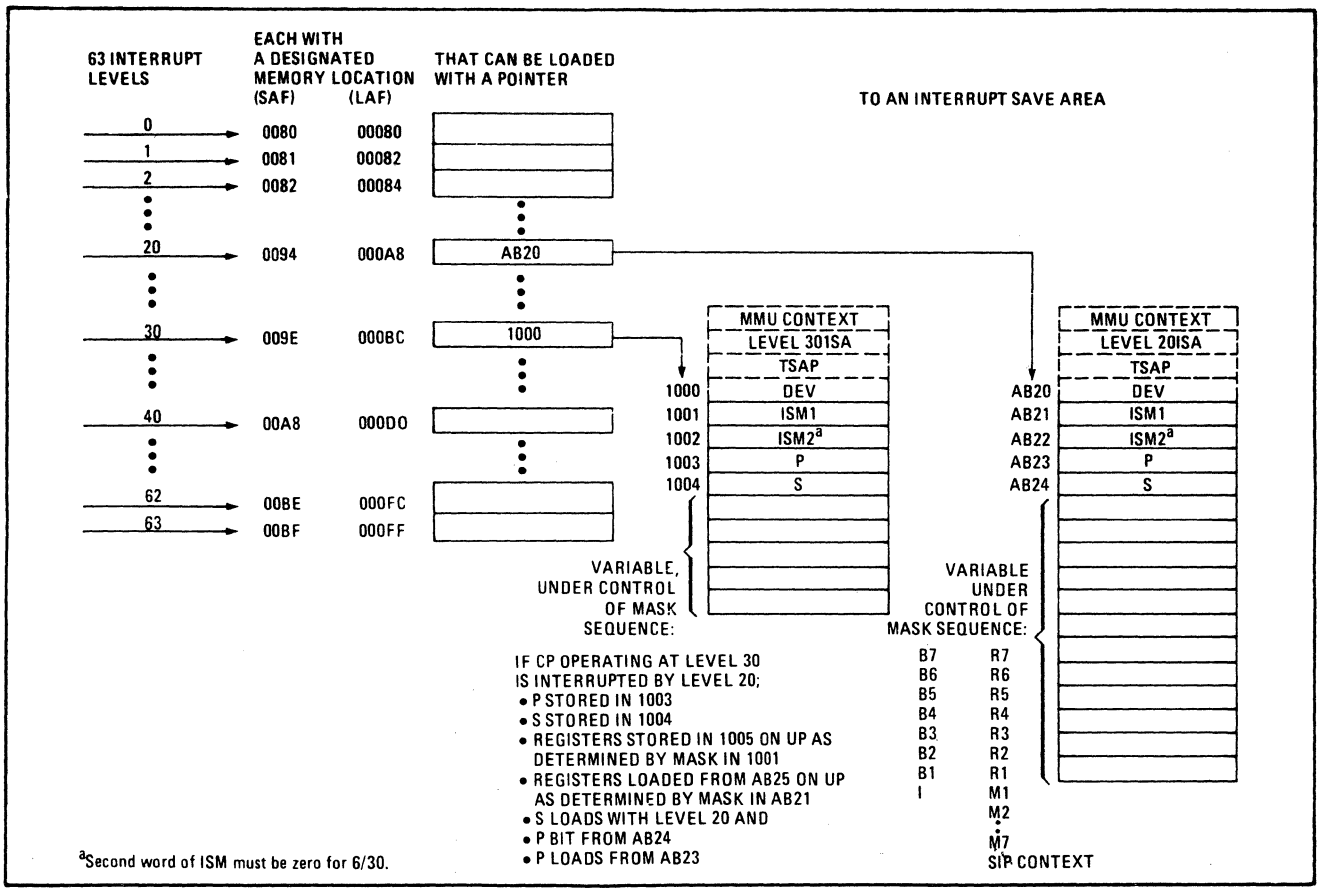

Figure 3-5. Interrupt Action

The mask bits are scanned from right (bit 15) to left (bit 0), ISM1 first and then ISM2.

It should also be noted that the word prior to the one pointed to by the interrupt vector (that is, the word prior to the device word) is also a dedicated word. This is called the TSAP and points to a trap save area. (See discussion of Traps.)

Figure 3-5 shows this action of an interrupt. Both levels 20 and 30 have had their interrupt save areas set up – level 20s at AB20, as pointed to by its vector at 0094; and level 30s at 1000, as pointed to by its vector at 009E.

If level 30 is interrupted by level 20, firmware automatically saves the register contents in the level 30 interrupt save area and loads the registers with the values in the level 20 interrupt save area. One of these values is the starting address of the level 20 interrupt subroutine which is automatically loaded (in this example) from location AB23 into P.

Associated with each interrupt level is a dedicated flag bit which is set when the interrupt is initiated. These bits are stored in four dedicated memory locations, 0020-0023. In the example above, both bits 20 and 30 would be set. At the end of the interrupt routine for level 20, a level change instruction (LEV) would be executed. This would clear the bit for level 20 and then scan the table to determine which is the next highest scheduled level. If no intermediate interrupts (such as level 25) were pending, it would scan the table, find bit 30 set, and therefore return to level 30.

TABLE 3-1. EVENT INTERRUPT LEVEL ASSIGNMENT

Event Causing Interrupt	Level Assignment	Comment
Incipient Power Failure	0	Highest priority
Watchdog Timer Runout	1	–
Overflow of Trap Context Save Area	2	–
Real-Time Clock	–	Level is contained in main memory location 0016 (HEX)
Device Requiring Service	–	Level is either fix wired or dynamically controlled by software
LEV Instruction	–	Level specified in instruction

The LEV instruction can set or reset activity flags, change the current level, inhibit interrupts, or do a "quick change" without saving and restoring context. By changing the current level to level 3 all device interrupts can be inhibited; level 2 (overflow of trap context scan area), level 1 (watchdog timer runout), and level 0 (incipient power failure) will still be enabled. Table 3-1 shows the assignment of interrupt levels.

TRAPS

Traps are caused by events such as overflows, parity errors, addressing nonexistent resources, executing a scientific instruction on a 6/30 model or a scientific error on a 6/40 model. A trap can occur at any priority level, and several can be nested at the same level. A trap could be entered at one level, that level interrupted during the execution of the trap routine, and then the same trap routine reentered in the new level. See Table 3-2.

Each type of trap has its own trap vector containing a pointer to the trap-handler procedure. Also utilized is a pointer in location 10 to the next available trap save area. The latter are pooled, and pointers to the next available area are automatically adjusted by firmware. When a trap occurs, some (but not all) register contents are automatically stored in the trap save area. The pointer in the first word of the interrupt save area for the current level is adjusted so that it points to the trap save area; the pointer in the trap save area points to any other traps that occurred at the same interrupt level. Thus, several traps may be nested at the same interrupt level. At the end of the execution of the trap-handler procedure, a return from trap (RTT) must be executed; this does a restoration of the partial context that was stored, unlike the trap save area, and returns all pointers to their original state.

The relationship of traps and interrupts, and their vector linkage are shown in Figure 3-6. The trap vector (TV) points to the trap-handler procedure. The trap save areas contain the following information:

TSAL – Trap save area link

I – Contents of indicator register

R3 – Contents of register R3

Instr – The instruction causing the trap

Z – Miscellaneous information

A – An address related to the trap condition (details depend upon trap type)

P – The program counter; e.g., on a branch, the location to which the branch would go if it were not trapped

B3 – The contents of register B3.

Note that the address of this area is noted in the Trap Save Area Pointer (TSAP) in the interrupt save area for the current level.