Accessing the CMOS Chip

Reading Values

```
int registerno, value;
outportb(0x70,registerno);          Tell the chip what to read
readvalue=inportb(0x71);            Read it
```

Writing Values

```
int registerno, value;
outportb(0x70,registerno);          Tell the chip what to write
outportb(0x71,value);               Write it
```

The Registers

Register No.	Read/Write	Function	Storage Format
0	Both	Current Second	According to Bit 2 of Status Register B
1	Both	Alarm Second	
2	Both	Current Minute	
3	Both	Alarm Minute	
4	Both	Current Hour	
5	Both	Alarm Hour	
6	Both	Current Day of Week(1 = Sunday)	
7	Both	Current Day of Month	
8	Both	Current Month	
9	Both	Last two digits of Year	
A	Both	**Status Register A** Bit 7 - Update in Progress (Read) Bits 0 to 3 - Periodic Interupt Rate 3210 Hz 3210 Hz ----------- ----------- 0001 256 0010 128 0011 8192 0100 4096 0101 2048 0110 1024 0111 512 1000 256 1001 128 1010 64 1011 32 1100 16 1101 8 1110 4 1111 2	Binary
B	Both	**Status Register B** Bit 7 - Abort Update(allow access to clock data) Bit 6 - Enable Periodic Interupt Bit 5 - Enable Alarm Interupt Bit 4 - Enable Update Ended Interupt Bit 2 - Clock Data Type 0=BCD, 1=Binary BIOS default=0 Bit 1 - Hour Data Type 0=12 hour, 1=24 hour BIOS default=1 Bit 0 - Daylight Savings Enable	Binary

Register No.	Read/Write	Function	Storage Format
C	Read	**Status Register C – Type of Interupt** Bit 7 – Any Bit 6 – Periodic Bit 5 – Alarm Bit 4 – Update Ended	Binary
E	Read	**POST Diagnostics Status** Bit 7 – Clock Lost Power Bit 6 – CMOS Bad Checksum Bit 5 – Invalid Configuration @ POST Bit 4 – Memory Size Compare Error Bit 3 – Disk or Controller Error Bit 2 – Invalid Time or Data (32nd)	Binary
F	Read	**Shutdown Status** 00 – power on reset 01 – memory size pass 02 – memory test pass 03 – memory test fail 04 – POST end, boot system 05 – JMP DWORD PTR 0:[0467h] with EOI 06 – protected tests pass 07 – protected tests fail 08 – memory size fail 09 – INT 15h block move 0A – JMP DWORD PTR 0:[0467h] without EOI	Hex
10	Read	**Floppy Drive Types** Bits 7 to 4 – Disk 0 (A:) Bits 3 to 0 – Disk 1 (B:) 0000 – no drive 0001 – 360k 0010 – 1.2M 0011 – 720k 0100 – 1.44M	Binary
12	Read	**Hard Drive Types** Bits 7 to 4 – Disk 0 (C:) Bits 3 to 0 – Disk 1 (D:) 0000 – no drive 0001 to 1110 – drive types 1111 – specified at 0=19h, 1=1Ah	Binary
14	Read	**Equipment** Bits 6 & 7 – No. Of Floppy Drives 00=1, 01=2, 10=3, 11=4 Bits 4 & 5 – Display Type 00 – none, EGA, VGA, etc. 01 – 40x25 colour 10 – 80x25 colour 11 = 80x25 monochrome Bit 1 – Math Co-processor Available Bit 0 – Floppy Drive(s) Available	Binary
15	Read	Base Memory (in kilobytes)	Low Byte
16	Read		High Byte

Register No.	Read/Write	Function	Storage Format
17	Read	Extended Memory above 1 megabyte (in kilobytes)	Low Byte
18	Read		High Byte
19	Read	Hard Drive 0 Type (only when specified at 12h	Unknown
1A	Read	Same as above for Drive 1	Unknown
2E	Read	Checksum of CMOS addresses 10H through 20H	High Byte
2F	Read		Low Byte
30	Read	Actual Extended Memory Size	Low Byte
31	Read		High Byte
32	Read	Century (First Two Digits)	BCD
33	Read	Bit 7 - IBM 128K memory installed	Binary

Converting from BCD to Decimal

```
int bcdval,decval,temp;
temp=(int)(bcdval/16);
decval=bcdval-16*temp;
```

Checking Bits from a byte

To check if bit 5 of a byte is high you can use the following code (bit 5 in decimal = 32):-

```
int binaryval;
if((binaryval & 32)==32){
   Code which happens when Bit 5 is one.
}
```

This works because 32 has binary form 100000 and the AND rule will only carry ones into the result which are in both numbers. Therefore the only bit which will be taken from the input binary value will be bit 5.

Interupts

After an interupt of any type has occured from the CMOS chip status register C must be read before another interupt may occur.