

42

“We humans are only a breath; none of us are truly great.”

Programming CMOS RAM

CMOS RAM is a random access memory made up of Complementary Metal Oxide Semiconductor (CMOS). CMOS is used for storing setup information in PC. It is used in hardware components that are powered by battery. It is widely used because of its low power consumption. CMOS RAM's size is usually referred as 64 or 128 byte. In fact, CMOS RAM is actually built into the Real-Time Clock (RTC) which has address space of 64 or 128 bytes. The clock registers of RTC use the first 16 bytes. So this CMOS RAM is actually 48 or 112 bytes.

42.1 Viewing contents of CMOS RAM

42.1.1 Logic

CMOS data are accessible via I/O ports 70h and 71h. First send the respective address of CMOS to I/O port 70h and then read the data from I/O port 71h.

Caution

Any write to port 70h should be followed by an action to port 71h, otherwise RTC will be left in an unknown state.

42.1.2 Code

Following is the code to view contents of CMOS RAM. As I said earlier, CMOS RAM is available in two sizes: 64 & 128 bytes. Here I assume that the size of my CMOS RAM is 128 bytes. You need not know the exact size of CMOS RAM for basic operations like viewing contents. However you must know the exact size of CMOS RAM for hazardous operations like clearing CMOS RAM.

```
#include <dos.h>
#define CMOS_ADDR (0x70) /* address port of CMOS */
#define CMOS_DATA (0x71) /* data port for CMOS */

int main( void )
{
    int offset, data;
    const int size = 128; /* or 64 depending upon your system */
    for ( offset=0; offset<size ; ++offset )
    {
        disable( );
```

```


    outportb( CMOS_ADDR, offset );
    data = inportb( CMOS_DATA );
    enable( );
    printf( "%0xX ", data );
}
return(0);
} /*--main( )-----*/

```

42.2 Diagnose CMOS RAM

42.2.1 Logic

The above program outputs just the hexadecimal contents of CMOS RAM. But to diagnose CMOS RAM we must know the structural design of CMOS RAM.

Each CMOS Register is 1 byte (8bits) in size. Following tables show description of each bits in CMOS registers. Ralf Brown's Interrupt List found on CD  also provides a clean note on CMOS Registers. For a better understanding the reader is advised to have a look on CMOS.LST file of Ralf Brown's Interrupt List.

| AT REAL TIME CLOCK STATUS REGISTER A | | | | | |
|--------------------------------------|-----|------|--------------------------|-----------------------------------|--|
| 7 | 654 | 3210 | FUNCTION | ALLOWABLE VALUES | |
| X | | | UPDATE IN PROGRESS | 1=DATE/TIME BEING UPDATED, 0=NOT | |
| | XXX | | 22 STAGE DIVIDER | DEFAULT=010, 32.768 KHZ TIME BASE | |
| | | XXXX | RATE SELECTION FREQUENCY | DEFAULT=0110, 1.024 KHZ | |

| AT REAL TIME CLOCK STATUS REGISTERS B | | | | | | | | | |
|---------------------------------------|---|---|---|---|---|---|---|---------------------|---------------------------------------|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | NAME | ALLOWABLE VALUES |
| X | | | | | | | | SET, 1 PER SECOND | 0=UPDATE NORMALLY, 1=ABORT UPDATE |
| | X | | | | | | | PERIODIC INT ENABLE | 0=DISABLE INT (DEFAULT), 1=ENABLED |
| | | X | | | | | | ALARM INT ENABLE | 0=DISABLED (DEFAULT), 1=ENABLED |
| | | | X | | | | | UPDATE END INT ENA. | 0=DISABLED (DEFAULT), 1=ENABLED |
| | | | | X | | | | SQUARE WAVE ENABLE | 0=DIS (DEF), 1=ENA, PER REG A 0-3 |
| | | | | | X | | | DATE MODE | 0=BCD (DEFAULT), 1=BINARY |
| | | | | | | X | | 24/12 MODE | 0=12 HOUR, 1=24 HOUR FORMAT (DEFAULT) |
| | | | | | | | X | DAYLIGHT SAVING ENA | 0=DISABLED (DEFAULT), 1=ENABLED |

| AT REAL TIME CLOCK STATUS REGISTER C | | | | | | |
|--------------------------------------|---|---|---|------|-----------|-----------------------|
| 7 | 6 | 5 | 4 | 3210 | NAME | ALLOWABLE VALUES |
| X | | | | | IRQF FLAG | READ ONLY |
| | X | | | | PF FLAG | READ ONLY |
| | | X | | | AF FLAG | READ ONLY |
| | | | X | | UF FLAG | READ ONLY |
| | | | | XXXX | RESERVED | SHOULD ALWAYS BE ZERO |

338 A to Z of C

| AT CMOS STATUS REGISTER D | | | |
|---------------------------|---------|---------------|--------------------------------------|
| 7 | 6543210 | NAME | ALLOWABLE VALUES |
| X | | VALID RAM BIT | 0=BATT DEAD,RAM INVALID, 1=BATT GOOD |
| | XXXXXXX | RESERVED | SHOULD ALWAYS BE ZERO |

| AT CMOS DIAGNOSTICS BYTE | | | | | | | | |
|--------------------------|---|---|---|---|---|----|---------------------|---------------------------------|
| 7 | 6 | 5 | 4 | 3 | 2 | 10 | NAME | ALLOWABLE VALUES |
| X | | | | | | | POWER STAT OF RTC | 1=CHIP HAS LOST POWER, 0=NOT |
| | X | | | | | | CHECKSUM STATUS | 0=CHECKSUM OK, 1=NOT OK |
| | | X | | | | | CONFIGURATION INFO | 0=VALID INFO, 1=NOT VALID |
| | | | X | | | | MEMORY SIZE COMPARE | 0=SAME SIZE, 1=NOT SAME SIZE |
| | | | | X | | | FIXED DISK STATUS | 0=OK, 1=DRIVE OR ADAPTER FAILED |
| | | | | | X | | TIME STATUS | 0=TIME IS OK, 1=TIME NOT OK |
| | | | | | | XX | RESERVED | |

| AT CMOS DRIVE TYPE BYTE | | | |
|-------------------------|------|----------------------|---|
| 7654 | 3210 | FUNCTION | ALLOWABLE VALUES |
| XXXX | | TYPE OF FIRST DRIVE | 0000=NO DRIVE, 0001=360K 5.25" 0010=1.2M 5.25" 0011=720K 3.5" 0100=1.44M 3.5" |
| | XXXX | TYPE OF SECOND DRIVE | |

| AT CMOS FIXED DRIVE TYPES | | | |
|---------------------------|------|-------------------|---|
| 7654 | 3210 | NAME | ALLOWABLE VALUES |
| XXXX | | FIXED DISK C TYPE | 0000=NO DRIVE 1H TO 0EH SEE CHART |
| | XXXX | FIXED DISK D TYPE | 0000=NO DRIVE 1H TO 0EH SEE CHART IF BYTE= 0FH THEN SEE EXTENDED BYTE FOR DRIVE TYPE |

| AT CMOS EQUIPMENT BYTE | | | | | | |
|------------------------|----|----|---|---|-----------------------|---|
| 76 | 54 | 32 | 1 | 0 | NAME | ALLOWABLE VALUES |
| XX | | | | | NUMBER OF DISK DRIVES | 00=1,01=2,10=3,11=4 |
| | XX | | | | PRIMARY DISPLAY TYPE | 00=DISPLAY HAS BIOS or EGA, 01=40 COL CGA, 10=80 COL CGA, 11=MDA, 101=EGA |
| | | XX | | | NOT USED | |
| | | | X | | MATH COPROCESSOR | 0=NOT INSTALLED, 1=INSTALLED |
| | | | | X | DISK DRIVES AVAILABLE | 0=NO DRIVES, 1=DISK DRIVES AVAILABLE |

| AT CMOS DRIVE C AND D EXTENDED DRIVE TYPE BYTES | | |
|--|-------------------|--------------------------|
| 76543210 | NAME | ALLOWABLE VALUES |
| XXXXXXXX | DRIVE C TYPE BYTE | SEE NEXT CHART FOR TYPES |
| XXXXXXXX | DRIVE D TYPE BYTE | SEE NEXT CHART FOR TYPES |
| IF FIXED DRIVE 4 BITS FOR C IS 0-0EH IGNOR EXTENDED C IF FIXED DRIVE 4 BITS FOR D IS 0-0EH IGNOR EXTENDED D | | |

| AT HARD DISK TYPES | | | | | | |
|--------------------|----------------|-------------|----------|-----------|-----------------|---------|
| DISK TYPE | CYLINDER COUNT | TOTAL HEADS | PRE COMP | LAND ZONE | SECTORS PER/TRK | SIZE MB |
| 1 | 306 | 4 | 128 | 305 | 17 | 10.1 |
| 2 | 615 | 4 | 300 | 615 | 17 | 20.4 |
| 3 | 615 | 6 | 300 | 615 | 17 | 30.6 |
| 4 | 940 | 8 | 512 | 940 | 17 | 62.4 |
| 5 | 940 | 6 | 512 | 940 | 17 | 46.8 |
| 6 | 615 | 4 | NONE | 615 | 17 | 20.4 |
| 7 | 462 | 8 | 256 | 511 | 17 | 30.6 |
| 8 | 733 | 5 | NONE | 733 | 17 | 30.4 |
| 9 | 900 | 15 | NONE | 901 | 17 | 112.0 |
| 10 | 820 | 3 | NONE | 820 | 17 | 20.4 |
| 11 | 855 | 5 | NONE | 855 | 17 | 35.4 |
| 12 | 855 | 7 | NONE | 855 | 17 | 49.6 |
| 13 | 306 | 8 | 128 | 319 | 17 | 20.3 |
| 14 | 733 | 7 | NONE | 733 | 17 | 42.5 |
| 16 | 612 | 4 | 0 | 663 | 17 | 20.5 |
| 17 | 977 | 5 | 300 | 977 | 17 | 40.5 |
| 18 | 977 | 7 | NONE | 977 | 17 | 56.7 |
| 19 | 1024 | 7 | 512 | 1023 | 17 | 59.5 |
| 20 | 733 | 5 | 300 | 732 | 17 | 30.4 |
| 21 | 733 | 7 | 300 | 732 | 17 | 42.5 |
| 22 | 733 | 5 | 300 | 733 | 17 | 30.4 |
| 23 | 306 | 4 | 0 | 336 | 17 | 10.1 |
| 25 | 615 | 4 | 0 | 615 | 17 | 20.4 |
| 26 | 1024 | 4 | NONE | 1023 | 17 | 34.0 |
| 27 | 1024 | 5 | NONE | 1023 | 17 | 42.5 |
| 28 | 1024 | 8 | NONE | 1023 | 17 | 68.0 |
| 29 | 512 | 8 | 256 | 512 | 17 | 34.0 |
| 30 | 615 | 2 | 615 | 615 | 17 | 10.2 |
| 31 | 989 | 5 | 0 | 989 | 17 | 41.0 |
| 32 | 1020 | 15 | NONE | 1024 | 17 | 127.0 |
| 35 | 1024 | 9 | 1024 | 1024 | 17 | 76.5 |
| 36 | 1024 | 5 | 512 | 1024 | 17 | 42.5 |
| 37 | 830 | 10 | NONE | 830 | 17 | 68.8 |
| 38 | 823 | 10 | 256 | 824 | 17 | 68.3 |
| 39 | 615 | 4 | 128 | 664 | 17 | 20.4 |
| 40 | 615 | 8 | 128 | 664 | 17 | 40.8 |
| 41 | 917 | 15 | NONE | 918 | 17 | 114.1 |
| 42 | 1023 | 15 | NONE | 1024 | 17 | 127.3 |
| 43 | 823 | 10 | 512 | 823 | 17 | 68.3 |
| 44 | 820 | 6 | NONE | 820 | 17 | 40.8 |
| 45 | 1024 | 8 | NONE | 1024 | 17 | 68.0 |
| 46 | 925 | 9 | NONE | 925 | 17 | 69.1 |
| 47 | 699 | 7 | 256 | 700 | 17 | 40.6 |

340 A to Z of C

42.2.2 Code

This is the C code to read the contents of CMOS setup registers and diagnose it. It analyzes the power of battery, checksum etc through the contents of CMOS registers. Once I received this code from someone else. I am not aware of the real author. The author assumes the size of the CMOS to be 64 bytes.

```
#include <stdio.h>
#include <dos.h>

typedef struct
{
    char  seconds;      /* AT Real Time Clock (RTC): Seconds */
    char  secalarm;    /* AT RTC: Seconds Alarm */
    char  minutes;     /* AT RTC: Minutes */
    char  minalarm;    /* AT RTC: Minutes Alarm */
    char  hours;       /* AT RTC: Hours */
    char  hrsalarm;    /* AT RTC: Hours Alarm */
    char  dayofweek;   /* AT RTC: day of week */
    char  dayofmon;    /* AT RTC: day of month */
    char  month;       /* AT RTC: month */
    char  year;        /* AT RTC: year */
    char  aregister;   /* STATUS REGISTER A */
    char  bregister;   /* STATUS REGISTER B */
    char  cregister;   /* STATUS REGISTER C */
    char  dregister;   /* STATUS REGISTER D */
    char  diagnostic; /* Diagnostics status byte */
    char  shutdown;   /* Shutdown status byte */
    char  diskettes;  /* A & B diskette types */
    char  reserved1;  /* undefined */
    char  harddrive;  /* C & D hard drive types */
    char  reserved2;  /* undefined */
    char  equipment;  /* equipment byte */
    char  lowbyte;    /* low byte of base memory */
    char  highbyte;   /* high byte of base memory */
                    /* 100h = 256k, 200h = 512k, 280h = 640k */
    char  extlow;     /* low byte of extended memory */
    char  exthigh;    /* high byte of extended memory */
                    /* 200h=512k;400h=1024k;etc to 3c00h=15360k */
    char  drivec;     /* more data on drive c */
    char  drived;     /* more data on drive d */
    char  reserved[19]; /* reserved */
    unsigned checksum;
    char  extlow1;    /* same as extlow */
    char  exthigh1;  /* same as exthigh */
    char  century;    /* binary coded decimal value for century */
                    /* 19h = 1900 for example */
}
```

```

        char  infoflag;    /* bit 7 set = top 128k installed */
        char  info[12];
    } CMOS, *CMOSPTR;

#define      CMOS_ADDR    0x70        /* address port of CMOS */
#define      CMOS_DATA    0x71        /* data port for CMOS */

void GetCMOS( char *cmosdata )        /* read CMOS data (64 bytes) */
{
    unsigned char j, byte;

    for ( j=0; j<64; j++ )
    {
        disable( );                /* disable interrupts */
        outportb( CMOS_ADDR, j );    /* specify byte to get */
        byte= inportb( CMOS_DATA );  /* get data */
        enable( );                  /* enable interrupts */
        *cmosdata++ = byte;          /* save CMOS data */
    }
} /*--GetCMOS( )-----*/

void ReadCMOS( void )
{
    static char *floppy[] = {
        "None",
        "360K 5.25-inch",
        "1.2M 5.25-inch",
        "720K 3.5-inch",
        "1.44M 3.5-inch"
    };
    static char *display[] = {
        "EGA",                        /* 00 */
        "40 column CGA",             /* 01 */
        "80 column CGA",             /* 10 */
        "MDA",                        /* 11 */
    };
    static char *math[] = {
        "Not Installed",
        "Installed"
    };
    static char *diag[] = {
        "Time",
        "Hard Dr",
        "Memory",
        "CnfInfo",
        "Chksum",
    };
}

```

342 A to Z of C

```

        "PwrOK"
    };
static char *status[] = {
    "OK",
    "Not OK"
};
static char *hardtbl[] = {
    "
    " | Drive | Cylinder | Heads/ | Pre- | Land | Sectors | Size |
    " | Type | (Tracks) | Sides | Comp | Zone | Per Trk | (MB) |
    " |-----|-----|-----|-----|-----|-----|-----|
};
static char *harddisk[] = {
    " | None | --- | -- | --- | --- | -- | ---- |
    " | 1 | 306 | 4 | 128 | 305 | 17 | 10.1 |
    " | 2 | 615 | 4 | 300 | 615 | 17 | 20.4 |
    " | 3 | 615 | 6 | 300 | 615 | 17 | 30.6 |
    " | 4 | 940 | 8 | 512 | 940 | 17 | 62.4 |
    " | 5 | 940 | 6 | 512 | 940 | 17 | 46.8 |
    " | 6 | 615 | 4 | NONE | 615 | 17 | 20.4 |
    " | 7 | 462 | 8 | 256 | 511 | 17 | 30.6 |
    " | 8 | 733 | 5 | NONE | 733 | 17 | 30.4 |
    " | 9 | 900 | 15 | NONE | 901 | 17 | 112.0 |
    " | 10 | 820 | 3 | NONE | 820 | 17 | 20.4 |
    " | 11 | 855 | 5 | NONE | 855 | 17 | 35.4 |
    " | 12 | 855 | 7 | NONE | 855 | 17 | 49.6 |
    " | 13 | 306 | 8 | 128 | 319 | 17 | 20.3 |
    " | 14 | 733 | 7 | NONE | 733 | 17 | 42.5 |
    " | 16 | 612 | 4 | 0 | 663 | 17 | 20.5 |
    " | 17 | 977 | 5 | 300 | 977 | 17 | 40.5 |
    " | 18 | 977 | 7 | NONE | 977 | 17 | 56.7 |
    " | 19 | 1024 | 7 | 512 | 1023 | 17 | 59.5 |
    " | 20 | 733 | 5 | 300 | 732 | 17 | 30.4 |
    " | 21 | 733 | 7 | 300 | 732 | 17 | 42.5 |
    " | 22 | 733 | 5 | 300 | 733 | 17 | 30.4 |
    " | 23 | 306 | 4 | 0 | 336 | 17 | 10.1 |
    " | 25 | 615 | 4 | 0 | 615 | 17 | 20.4 |
    " | 26 | 1024 | 4 | NONE | 1023 | 17 | 34.0 |
    " | 27 | 1024 | 5 | NONE | 1023 | 17 | 42.5 |
    " | 28 | 1024 | 8 | NONE | 1023 | 17 | 68.0 |
    " | 29 | 512 | 8 | 256 | 512 | 17 | 34.0 |
    " | 30 | 615 | 2 | 615 | 615 | 17 | 10.2 |
    " | 31 | 989 | 5 | 0 | 989 | 17 | 41.0 |
    " | 32 | 1020 | 15 | NONE | 1024 | 17 | 127.0 |
    " | 35 | 1024 | 9 | 1024 | 1024 | 17 | 76.5 |
    " | 36 | 1024 | 5 | 512 | 1024 | 17 | 42.5 |
    " | 37 | 830 | 10 | NONE | 830 | 17 | 68.8 |
    "

```

| | | | | | | | | |
|---|----|------|----|------|------|----|-------|-----|
| " | 38 | 823 | 10 | 256 | 824 | 17 | 68.3 | " |
| " | 39 | 615 | 4 | 128 | 664 | 17 | 20.4 | " |
| " | 40 | 615 | 8 | 128 | 664 | 17 | 40.8 | " |
| " | 41 | 917 | 15 | NONE | 918 | 17 | 114.1 | " |
| " | 42 | 1023 | 15 | NONE | 1024 | 17 | 127.3 | " |
| " | 43 | 823 | 10 | 512 | 823 | 17 | 68.3 | " |
| " | 44 | 820 | 6 | NONE | 820 | 17 | 40.8 | " |
| " | 45 | 1024 | 8 | NONE | 1024 | 17 | 68.0 | " |
| " | 46 | 925 | 9 | NONE | 925 | 17 | 69.1 | " |
| " | 47 | 699 | 7 | 256 | 700 | 17 | 40.6 | "}; |

```

CMOS      cmosdata;
char      *iptr = (char *)&cmosdata;
int       j, k, drive;

GetCMOS( iptr );      /* read 64 bytes of CMOS data */
printf( "CMOS Diagnostics Status:\n" );
j = (cmosdata.diagnostic >> 2);
for ( k=0; k<6; k++)
  {
    printf( "%-7s: %s\n", diag[k], status[(j & 1)] );
    j >>= 1;
  }
printf( "\nCMOS Equipment Information:\n" );
printf( "Display: %s\n", display[(cmosdata.equipment >> 4) & 3] );
printf( " Coproc: %s\n", math[(cmosdata.equipment & 2)] );
drive = 'A';
j = (cmosdata.equipment & 1) * (1 + (cmosdata.equipment >> 6));
printf( " Floppy: %d\n",j );
if ( j )
  {
    printf( "Drive %c: %s\n", drive++,
            floppy[(cmosdata.diskettes >> 4)] );
    printf( "Drive %c: %s\n", drive++,
            floppy[(cmosdata.diskettes & 0x0f)] );
  }
printf( "Hard Dr: " );
if ( cmosdata.harddrive ) /* at least 1 hard drive */
  {
    printf( "\n" );
    for ( j=0; j<4; j++ )
      printf( "          %s\n",hardtbl[j] );
    j = (cmosdata.harddrive >> 4);
    k = (cmosdata.harddrive & 0x0f);
    if (j == 15)
      j = (cmosdata.drivec);
    if (k == 15)

```


344 A to Z of C

```
        k = (cmosdata.drived);
printf( "Drive %c: %s\n", drive++, harddisk[j] );
printf( "Drive %c: %s\n", drive, harddisk[k] );
printf( "          |-----|
          |-----|-----|-----|\n" );
    }
else
    printf( "None\n" );
iptr = (char *)&cmosdata;
printf( "\nHex Dump of CMOS RAM:\n" );
for ( j=0,k=0 ; j<64; j++ )
    {
        printf( "%02x ", *iptr++ );
        k++;
        if ( k == 16 )
            {
                k = 0;
                printf( "\n" );
            }
    }
} /*--ReadCMOS( )-----*/

int main( void )
{
    ReadCMOS( );
    return(0);
} /*--main( )-----*/
```

42.3 Illegal Operation

By programming CMOS RAM, we can even remove the setup password through programs. It is explained in “Illegal Codes” unit.