আহসান হাবীব পলাশ

# TSR প্রোগ্রামিং

(পূর্ব প্রকাশিতের পর)

## CLOCK

গত সংখ্যার আলোচনার ভিত্তিতে খুব সহজে একটা TSR CLOCK এর প্রোগ্রাম লেখা যায়। প্রথমে gettim ()এর মাধ্যমে সিস্টেম থেকে সময় পড়ে নিতে হবে। পূর্বেই বলা হয়েছে timer চিপ প্রতি সেকেণ্ডে ১৮.২ টা pulse পাঠায় এবং প্রতিটি Pulse এর সাথে ইন্টারাপ্টা ৮–এর ISR এক্সিকিউট করে। সুতরাং উপরের ইন্টারাপ্টের অ্যাড্রেসকে প্রতিস্থাপন করার জন্য যে ISR লিখতে হবে সেখানে হিসাব রাখতে হবে এটা কত বার কল হলো। count নামে একটা Global variable ডিক্লেয়ার করে প্রতিবার কলের জন্য count এর মান এক বাড়িয়ে দিতে হবে। এই নতুন ISR একটা ফাংশনকে কল করবে যে স্ক্রীনে সময় দেখাবে। এই ফাংশনে দেখতে হবে count এর মান ১৮ হলো কি না, যদি হয় তবে সেকেণ্ডের মান এক বাড়িয়ে দিয়ে count কে শূন্য করে দিতে হবে। যেহেতু সেকেণ্ডে ১৮.২ টা বিট দেয় সুতরাং প্রতি ৫ সেকেণ্ডে ঘড়িটি একটি বিট হারাবে। এটা ঠিক করার জন্য প্রতি (৫ × ১৮) = ৯০ সেকেণ্ড পরপর সেকেণ্ডের মান এক কমিয়ে দিতে হবে। সেকেণ্ডের মান ৬০ হলে মিনিটকে এবং মিনিটের জন্য ঘন্টাকে পরিবর্তন করে মানগুলো মনিটরে দেখালেই কাজ শেষ। তবে এত ঝামেলায় না যেয়ে Count এর মান ১৮ হওয়া মাত্র সিস্টেম থেকে আবার সময় পড়ে নিয়ে দেখানো যেতে পারে।

সময়কে স্ক্রীনে দেখানোর জন্য এমন কোন ফাংশন ব্যবহার করা উচিত না, যেটা সময় নেয় বেশী। এক্ষেত্রে video RAM এর সাহায্য নেওয়া শ্রেয়। খুব সংক্ষেপে video RAM হলো স্ক্রীনের হুবহু একটা অবয়ব RAM–এ থাকে। সেখানে সরাসরি কিছু লেখা মানে স্ক্রীনেই Echo করা। এখানে কাজ খুব দ্রুত হয়। Video RAM–এ কিভাবে লিখতে হয় তা CLOCK প্রোগ্রামের মধ্যে দেওয়া আছে। নীচে সম্পূর্ণ প্রোগ্রামটা দেওয়া হলো। প্রোগ্রামটিকে EXE বানিয়ে DOS প্রম্পট থেকে রান করালে স্ক্রীনে CLOCK দেখা যাবে। TURBO C এর এডিটর থেকে OSSHELL এর মাধ্যমে বেরিয়ে এসে এটাকে রান না করানো ভালো। এতে মেমোরী ম্যানেজমেণ্টে সমস্যা হতে পারে।

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

void    tsr_clock();
void    check_video_mode();
void    interrupt dos_idle();
void    interrupt new_int8();
void    interrupt (*old_int28)();
void    interrupt (*old_int8)();
void    interrupt (*old_int64)();
void    print_ram (int ,int ,char );

int             count =0;
char    far     *dos_active;
char    busy = 0;
long    int     VIDEO_BASE;
struct  time    t;
main()
{       union   REGS  r;
        struct  SREGS  s;

        check_video_mode();
        old_int64 = getvect(0x64);
        if (!old_int64)  setvect(0x64,new_int8); /* set a flag */
        else {
                puts("TSR is already installed ");
                return;
        }
        r.h.ah = 0x34;
        int86x(0x21,&r,&r,&s);  /* obtain dos active flag address */
        dos_active = MK_FP(s.es,r.x.bx);
        old_int28 = getvect(0x28);  /* get old address */
        old_int8 = getvect(8);
        setvect(0x28,dos_idle); /* replace old  by new */
        setvect(8,new_int8);
        keep(0,1000); /* make TSR */
}
void    interrupt new_int8() /* new ISR */
{       count++;
        (*old_int8)(); /* call original routine */
        if (!*dos_active && !busy )     tsr_clock();
}
void     interrupt dos_idle()
{       (*old_int28)();
        if (!busy ) tsr_clock();
}
void            tsr_clock() /* TSR routine */
{       if (count >= 18) { /* due to disk I/O count may be greater than 18 */
                count = 0;
                if (!busy)  {
                        gettime(&t);
                        busy = !busy;  /* don't allow second time activation */
                        print_ram(75,1,t.ti_sec%10+0x30);/*0x30 is used to make
ASCII code*/
                        print_ram(74,1,t.ti_sec/10+0x30);
                        print_ram(73,1,':');
                        print_ram(72,1,t.ti_min%10+0x30);
                        print_ram(71,1,t.ti_min/10+0x30);
                        print_ram(70,1,':');
                        print_ram(69,1,t.ti_hour%10+0x30);
                        print_ram(68,1,t.ti_hour/10+0x30);
                        busy = !busy;


void print_ram(int x,int y,char ch)
{       char far *p;
        x+=(y-1)*80;
        x*=2;
        p=MK_FP(VIDEO_BASE,x);
        *p=ch;
}
void check_video_mode()
{       union           REGS    r;
        int                     video;

        r.h.ah = 15;
        video = int86(0x10,&r,&r) & 255;
        if (video == 7) VIDEO_BASE = 0xb000;
                else    VIDEO_BASE = 0xb800;
```

## HOTKEY

CLOCK এর প্রোগ্রামে ইনপুটের ভিত্তিতে কাজ করার কোন উদাহরণ নেই। অনেক TSR প্রোগ্রামের অ্যাকটিভেশন কিবোর্ড ইনপুটের উপর নির্ভর করতে পারে। সুতরাং এ সম্পর্কে না বললে অসম্পূর্ণ থেকে যাবে।

যখনই কিবোর্ডের কোন কী প্রেস করা হয় তখন ইন্টারাপ্ট ৯–এর ISR এক্সিকিউট করে। এই ISR পোর্ট থেকে একটা ক্যারেক্টার পড়ে বাফারে রেখে দেয়, এটাকে ক্যারেক্টার বাফার বলে। যখন DOS বা BIOS এর কিবোর্ড ইনপুট ফাংশনকে কল করা হয়, তখন ফাংশনটি পোর্ট থেকে না পড়ে শুধুমাত্র ক্যারেক্টার বাফার চেক করে এবং বাফার থেকে ক্যারেক্টার পড়ে নেয়। সুতরাং প্রোগ্রামের মধ্যে বাফার চেক করে সহজেই বোঝা যায় কোন কী প্রেস করা হয়েছে কিনা। যদি আমাদের প্রত্যাশিত কী প্রেস করা হয় তাহলে তার জন্য প্রয়োজনীয় কাজ করা যেতে পারে।

এখন দেখা যাক বাফার থেকে কিভাবে পড়তে হবে। এই বাফারটা একটা সারির মতো। এর একটা Start এবং একটা End pointer থাকে। Start pointer যে Key প্রেস করা হয়েছে তাকে পয়েন্ট করে থাকে। এই Start pointer এর লোকেশন 0000:041A (ডেসিমালে ১০৫০)। সুতরাং এই লোকেশনে একটা পয়েন্টার সেট করে সহজেই বাফার থেকে ক্যারেক্টার পড়া যায়। যদি Start ও End এর মান একই হয় তাহলে বোঝা যায় বাফার শূন্য।

ধরা যাক, কোন TSR প্রোগ্রাম F1 প্রেস করলে সক্রিয় এবং F2 প্রেস করলে নিষ্ক্রিয় হয়। TSR কে কিভাবে নিষ্ক্রিয় করা যায় তা পরে আলোচনা করা হবে। এখন দেখা যাক F1 এবং F2 চেক করার জন্য যে ISR লিখতে হবে সেটা কেমন।

```
void interrupt Key_press()
{
int far *t2=(int far*) 1050;
char far *t=(char far*) 1050;
(*old_key) ();/*old ISR*/
if (*t!=*(t+2)) {/*not Empty*/
t+=*t-30+5;
switch(*t){
case 59:Key=1;break;
case 60:key=2;break;
}
*(t2+1)=*t2/*make buffer zero*/
}
```

এখানে শুধু কী–এর একটা মান দেওয়া হয়েছে। পরে এই মান অনুসারে TSR রুটিনের এক্সিকিউশন নিয়ন্ত্রণ করা হবে।

একটা উদাহরণ দেওয়া যাক। এমন একটা প্রোগ্রাম লিখতে হবে যা পাঁচ মিনিট ধরে কোন কী প্রেস না করলে স্ক্রীনকে অফ (blank) করে দেবে। ঠিক ক্যালকুলেটরের auto power off–এর মতো। কাজটা খুবই সোজা। দেখতে হবে বাফারে কোন ক্যারেক্টার আছে কি না। বাফার শূন্য থাকলে অর্থাৎ কোন কী প্রেস করা না হলে সময় হিসাব করতে হবে। কোন কী প্রেস করা মাত্র এই হিসাবকৃত সময়কে শূন্য করে দিতে হবে। CLOCK প্রোগ্রামের মতো এখানেও count ব্যবহার করতে হবে। এর মান ৫×৬০×১৮.২ হওয়ার পরও যদি কোন কী প্রেস করা না হয় তাহলে স্ক্রীনে যা আছে সেভ করে রেখে স্ক্রীনটা পুরোপুরি মুছে দিতে হবে। এরপর কোন কী প্রেস করা মাত্র স্ক্রীনকে Restore করতে হবে এবং count কে শূন্য করে দিতে হবে।

(চলবে)

## ভুল সংশোধন

জুলাই '৯৩ সংখ্যার "আপনার পিসিটিকে আরো কাছ থেকে দেখুন-১" প্রবন্ধের ২য় পৃষ্ঠার ২য় কলামের শেষের দিকের লাইন "যেসব সিস্টেম-এর ....... অনুপাতে সেট করা যায়।"-এর পরিবর্তে নীচের লেখাটি পড়তে হবে।

"যেসব সিস্টেম-এর সিপিইউ ৮ মেগাহার্জজ-এর চেয়ে বেশী দ্রুত গতিসম্পন্ন এবং হার্ড-ডিস্কের অ্যাকসিস টাইম ৪০ মিলিসেকেণ্ডের চেয়ে কম সেই হার্ড-ডিস্কের ইণ্টারলিভ ১:২ অনুপাতে সেট করা যায়।"