Chapter 10

Programming in C

Lesson 11

C Programming Example for Real Time Clock

Real time clock (RTC)

- Based on interrupts at the regular intervals
- Facilitates scheduling of different system tasks at the regular intervals

8051 for RTC interrupts

- Mode 2 can be programmed to get interrupts at the programmed intervals

C-program for regular interrupts after 16 ms

- #include <reg51.h> /* Include header file for the registers and SFRs of 8051. */
- struct {
  unsigned int rtc_ms;
- unsigned char rtc_s;
  unsigned char rtc_m;
- unsigned char rtc_hr;
  unsigned int rtc_day;
  } *newtime*;

Main

```
void main (void)
 {
.unsigned int numOVT0;
  /* Assign initial values 0*/
  numOV = 0; /* Number of overflows = 0.8*/
```

Main

- num_ms=0; /* Number of ms = 0 */
- newtime.rtc_ms = 0; /* rtc time ms = 0 */
- newtime.rtc_s= 0; /* rtc time s = 0 */
- newtime.rtc_m =0; /* rtc time m =0 */
- newtime.rtc_hr = 0; /* rtc time hr = 0 */
- newtime_rtc_day =0; /* rtc time day = 0 */
-

Main

```
EA = 1;
 ET0 =1;  /* Code for specifying T0 in mode 2
    and overflow after every 250 μs.*/
  While (1) {
;} /* While loop endlessly for ever. */ .
} /* End of the main */
```

Interrupt Function

```
void timer0ISR (void) interrupt 1 using 3  {
  if (numOVT0 < 4)  {    /* count T0 overflow
   from 0 up to 3 */
   numOVT0 ++; } else
 {numOVT0 = 0;}; /* Count T0 overflows from
   0 on next overflow */
```

Interrupt Function

```
if (newtime.num_ms < 16)  {    /* count
   num_ms, number of ms up to 15 */
   newtime.num_ms ++;} else /* Increment
   num_ms from 0 up to 16. */
 {newtime.num_ms = 0; /* Count num_ms from
   0 from next ms */
RTISR (newtime); /* Call RTCISR */
}
```

Routine RTCISR at regular Intervals to update system time

```
void RTCISR (newtime)  {
    if (rtc_ms < 1000)  {    /* count rtc_ms from 0
   up to 999 */
   newtime.rtc_ms += 16; } else
 { newtime.rtc_ms = 0; newtime.rtc_s ++;}; /*
   reset rtc_ms and increment rtc_s */
```

Routine RTCISR at regular Intervals to update system time

```
if (rtc_s < 60)  {    /* count rtc_s from 0 up to 59
   */
   newtime.rtc_s ++; } else
 { newtime.rtc_s  = 0; newtime.rtc_m ++;}; /*
   reset rtc_s and increment rtc_m */
```

Routine RTCISR at regular Intervals to update system time

```
if (rtc_m < 60)  {    /* count rtc_m from 0 up to
   59 */
  newtime.rtc_m ++; } else
 { newtime.rtc_m+ = 0; newtime.rtc_hr ++;}; /*
   reset rtc_m and increment rtc_hr */
```

Routine RTCISR at regular Intervals to update system time

```
if (rtc_hr < 24)  {    /* count rtc_hr from 0 up to
   24 */
   newtime.rtc_hr ++; } else
 { newtime.rtc_hr = 0; newtime.rtc_day ++;}; /*
   reset rtc_hr and increment rtc_day */


/* Statements for the actions on RTC 16 ms
   interrupt */
  }  /* End of RTCISR after 16 ms*/}
```

Summary

We learnt

- Based on interrupts at the regular intervals
- Facilitates scheduling of different system tasks at the regular intervals
- Program for real time clock interrupts every 16 ms
- Routine for update system time

End of Lesson 12 on

C Programming Example for
Real Time Clock