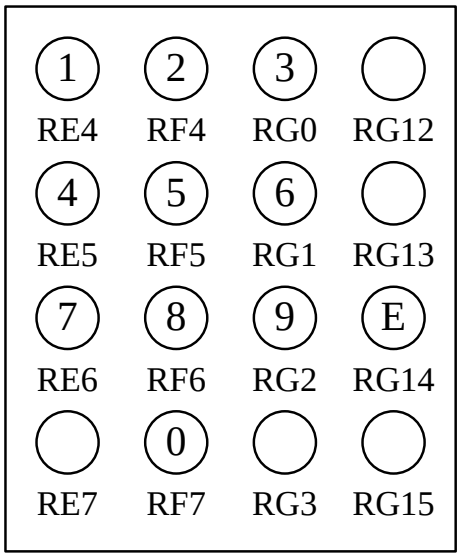
*Zadatak*

Realizovati digitalni časovnik sa datumom uz pomoć mikrokontrolerske razvojne ploče LV 24-33. Instrument treba da sadrži tastaturu na poziciji kako je označeno na slici 1. Osim cifara od 0 do 9 tastatura treba da sadrži i taster ENTER (na slici 1 označen sa E).

U fazi inicijalizacije časovnika na displeju se ispisuju redom poruke: Godina:, Mjesec:, Dan:, Sat:, Minut:, Sekund:. Nakon svake od poruka potrebno je unijeti odgovarajući podatak i aktivirati taster ENTER. Po završenoj inicijalizaciji na displeju se ispisuje vrijeme i datum u formatu kako je prikazano na slici 2, pri čemu je dd-dan, mm-mjesec, yyyy-godina, HH-sat, MM-minut, SS-sekund. Pri realizaciji instrumenta koristiti *Timer23*.




**Slika 1**


**Slika 2**

Inicijalizacija časovnika podrazumijeva zadavanje trenutka od kojeg se vrijeme počinje mjeriti, odnosno, definisanje trenutnog datuma, časa, minuta i sekunde. Te vrijednosti se zadaju pomoću odabrane tastature na razvojnoj mikrokontrolerskoj ploči (slika 1), a njihovo očitavanje se vrši u okviru glavnog programa, prije startovanja brojača:

```
  godina = Lcd_Read_Int("Godina:");
  mjesec = Lcd_Read_Int("Mjesec:");
  dan = Lcd_Read_Int("Dan:");
  sat = Lcd_Read_Int("Sat:");
  minut = Lcd_Read_Int("Minut:");
  sekund = Lcd_Read_Int("Sekund:");
```

Funkcija `Lcd_Read_Int(char *label)` vraća cjelobrojnu vrijednost podatka unijetog posredstvom tastature. Ovom funkcijom se takođe ispisuju odgovarajuće poruke na LCD-u, kao neka vrsta uputstva pri inicijalizaciji časovnika.

```
int Lcd_Read_Int(char *label)
{
  char buffer[17];
  Lcd_Cmd(_Lcd_Clear);
  Lcd_Out(1, 1, label);
  Lcd_Cmd(_LCD_SECOND_ROW);
  Lcd_Read(buffer);
  Lcd_Cmd(_LCD_CLEAR);
  return StrToInt(buffer);
}
```

U okviru funkcije `Lcd_Read_Int(char *label)` poziva se funkcija `Lcd_Read(char * buffer)`. Ova funkcija ″očitava″ karakter po karakter sa LCD-a do klika na taster ENTER (RG14) i povezuje ″očitane″ karaktere u string.

```
void Lcd_Read(char * buffer)
{
  int i = 0;
  do{
    if(button_released(0, &PORTE,  4)) { Lcd_Chr_Cp('1'); buffer[i++] = '1'; }
    if(button_released(1, &PORTF,  4)) { Lcd_Chr_Cp('2'); buffer[i++] = '2'; }
    if(button_released(2, &PORTG,  0)) { Lcd_Chr_Cp('3'); buffer[i++] = '3'; }
    if(button_released(3, &PORTE,  5)) { Lcd_Chr_Cp('4'); buffer[i++] = '4'; }
    if(button_released(4, &PORTF,  5)) { Lcd_Chr_Cp('5'); buffer[i++] = '5'; }
    if(button_released(5, &PORTG,  1)) { Lcd_Chr_Cp('6'); buffer[i++] = '6'; }
    if(button_released(6, &PORTE,  6)) { Lcd_Chr_Cp('7'); buffer[i++] = '7'; }
    if(button_released(7, &PORTF,  6)) { Lcd_Chr_Cp('8'); buffer[i++] = '8'; }
    if(button_released(8, &PORTG,  2)) { Lcd_Chr_Cp('9'); buffer[i++] = '9'; }
    if(button_released(9, &PORTF,  7)) { Lcd_Chr_Cp('0'); buffer[i++] = '0'; }
  }while(i < 16 && !button_released(10, &PORTG, 14));
  buffer[i] = '\0';
}
```

Funkcija `button_released(int i, unsigned int *port, unsigned int pin)` ispituje da li je došlo do klika na određeni taster. Ukoliko jeste, funkcija kao rezultat vraća jedinicu, dok u suprotnom vraća nulu.

```
int button_released(int i, unsigned int *port, unsigned int pin)
{
    if (Button(port, pin, 1, 1))
    {
      states[i] = 1;
    }
    if (states[i] && Button(port, pin, 1, 0))
    {
      states[i] = 0;
```

```
      return 1;
    }
    return 0;
}
```

Funkcija `StrToInt(char *s)` služi za konvertovanje stringa u cijeli broj.

```
int StrToInt(char *s)
{
  int num_digits;
  char digit;
  char c;
  int result = 0;
  int exp;
  int i=0;
  int st;
  int j=0;

  num_digits = strlen(s);     //dužina stringa
  while(s[i] != '\0')         //petlja se izvršava dok se ne dođe do
                              //terminacionog karaktera
  {
    c = s[i];
    digit = c - '0';
    st=1;
    exp = num_digits-i-1;
      for (j=0 ; j < exp; j++)
      {
            st*=10;
      }
    result+=digit*st;
    i++;
  }
  return result;
}
```

Slijedi primjer konverzije stringa ″128″ u broj 128:

- Prva iteracija: `i=0, c='1', digit=1, exp=2, st=100, result=0+1*100=100,`
- Druga iteracija: `i=1, c='2', digit=2, exp=1, st=10, result=100+2*10=120,`
- Treća iteracija: `i=2, c='8', digit=8, exp=0, st=1, result=120+8*1=128.`

Nakon što je časovnik inicijalizovan, potrebno je otpočeti sa mjerenjem vremena, odnosno, startovati brojač. Da bi se vrijeme mjerilo sa korakom od 1 s, pogodno je konfigurisati brojač tako da se prekid na prekoračenje brojača generiše svake sekunde. Ukoliko brojač koristi interni takt frekvencije $f_{osc}$ i preskaler 1:64, vrijeme trajanja jednog takta po kome brojač radi je $2*64/f_{osc}$, što za $f_{osc}$=10 MHz, iznosi 12.8 μs. Prekid na prekoračenje brojača generisaće se svake sekunde ukoliko se za periodu brojača odabere vrijednost 1 s/12.8 μs=$78125_{(10)}$= $1312D_{(16)}$. Može se primijetiti da se izračunata vrijednost ne može upisati u 16-bitni registar, stoga se kao brojač koristi *timer23* koji predstavlja ″kombinaciju″ *timer2*-a i *timer3*-a. Povezivanje *timer2*-a i *timer3*-a u 32-bitni brojač *timer23* ostvaruje se upisom logičke jedinice na poziciju T32 T2CON registra:

```
T2CONbits.T32 = 1;
```

Zadavanje izračunate periode brojača vrši se na sljedeći način:

```
PR3 = 0x0001;
PR2 = 0x312D;
```

(Opis T2CON registra može se naći u dokumentu PIC24FJ96GA010.pdf na strani 106.)

Po svakoj odbrojanoj sekundi generiše se prekid na prekoračenje brojača, što treba iskoristiti za osvježavanje stanja časovnika i trenutno vrijeme prikazati na LCD-u - rutina `Timer23Int()`.

```
void Timer23Int() org 0x0024
{
  char line1[16];
  char line2[16];

  IFS0bits.T3IF=0;

  sekund++;
  brojac_sekundi++;
  if(sekund>59) { sekund=0; minut++; }
  if(minut>59) { minut=0; sat++; }
  if(sat>23) { sat=0; dan++; }
  if(dan>28 && mjesec==2 && (godina%4)) { dan=1; mjesec++; }
  else if(dan>29 && mjesec==2 && !(godina%4)) { dan=1; mjesec++; }
  else if(dan>30 && (mjesec==4 || mjesec==6 || mjesec==9 || mjesec==11)) {dan=1;
mjesec++; }
  else if(dan>31) { dan=1; mjesec++; }
  if(mjesec>12) { mjesec=1; godina++; }

  sprintf(line1, "%.2d %.2d %.4d", dan, mjesec, godina);
  sprintf(line2, "%.2d:%.2d:%.2d", sat, minut, sekund);

  Lcd_Cmd(_LCD_CLEAR);
  Lcd_Out(1, 1, line1);
  Lcd_Out(2, 1, line2);
}
```

## Prilog

| Interrupt Source | Vector Number | IVT Address | AIVT Address | Interrupt Bit Locations | | |
|---|---|---|---|---|---|---|
| | | | | Flag | Enable | Priority |
| ADC1 Conversion Done | 13 | 00002Eh | 00012Eh | IFS0<13> | IEC0<13> | IPC3<6:4> |
| Comparator Event | 18 | 000038h | 000138h | IFS1<2> | IEC1<2> | IPC4<10:8> |
| CRC Generator | 67 | 00009Ah | 00019Ah | IFS4<3> | IEC4<3> | IPC16<14:12> |
| External Interrupt 0 | 0 | 000014h | 000114h | IFS0<0> | IEC0<0> | IPC0<2:0> |
| External Interrupt 1 | 20 | 00003Ch | 00013Ch | IFS1<4> | IEC1<4> | IPC5<2:0> |
| External Interrupt 2 | 29 | 00004Eh | 00014Eh | IFS1<13> | IEC1<13> | IPC7<6:4> |
| External Interrupt 3 | 53 | 00007Eh | 00017Eh | IFS3<5> | IEC3<5> | IPC13<6:4> |
| External Interrupt 4 | 54 | 000080h | 000180h | IFS3<6> | IEC3<6> | IPC13<10:8> |
| I2C1 Master Event | 17 | 000036h | 000136h | IFS1<1> | IEC1<1> | IPC4<6:4> |
| I2C1 Slave Event | 16 | 000034h | 000034h | IFS1<0> | IEC1<0> | IPC4<2:0> |
| I2C2 Master Event | 50 | 000078h | 000178h | IFS3<2> | IEC3<2> | IPC12<10:8> |
| I2C2 Slave Event | 49 | 000076h | 000176h | IFS3<1> | IEC3<1> | IPC12<6:4> |
| Input Capture 1 | 1 | 000016h | 000116h | IFS0<1> | IEC0<1> | IPC0<6:4> |
| Input Capture 2 | 5 | 00001Eh | 00011Eh | IFS0<5> | IEC0<5> | IPC1<6:4> |
| Input Capture 3 | 37 | 00005Eh | 00015Eh | IFS2<5> | IEC2<5> | IPC9<6:4> |
| Input Capture 4 | 38 | 000060h | 000160h | IFS2<6> | IEC2<6> | IPC9<10:8> |
| Input Capture 5 | 39 | 000062h | 000162h | IFS2<7> | IEC2<7> | IPC9<14:12> |
| Input Change Notification | 19 | 00003Ah | 00013Ah | IFS1<3> | IEC1<3> | IPC4<14:12> |
| Output Compare 1 | 2 | 000018h | 000118h | IFS0<2> | IEC0<2> | IPC0<10:8> |
| Output Compare 2 | 6 | 000020h | 000120h | IFS0<6> | IEC0<6> | IPC1<10:8> |
| Output Compare 3 | 25 | 000046h | 000146h | IFS1<9> | IEC1<9> | IPC6<6:4> |
| Output Compare 4 | 26 | 000048h | 000148h | IFS1<10> | IEC1<10> | IPC6<10:8> |
| Output Compare 5 | 41 | 000066h | 000166h | IFS2<9> | IEC2<9> | IPC10<6:4> |
| Parallel Master Port | 45 | 00006Eh | 00016Eh | IFS2<13> | IEC2<13> | IPC11<6:4> |
| Real-Time Clock/Calendar | 62 | 000090h | 000190h | IFS3<14> | IEC3<13> | IPC15<10:8> |
| SPI1 Error | 9 | 000026h | 000126h | IFS0<9> | IEC0<9> | IPC2<6:4> |
| SPI1 Event | 10 | 000028h | 000128h | IFS0<10> | IEC0<10> | IPC2<10:8> |
| SPI2 Error | 32 | 000054h | 000154h | IFS2<0> | IEC0<0> | IPC8<2:0> |
| SPI2 Event | 33 | 000056h | 000156h | IFS2<1> | IEC2<1> | IPC8<6:4> |
| Timer1 | 3 | 00001Ah | 00011Ah | IFS0<3> | IEC0<3> | IPC0<14:12> |
| Timer2 | 7 | 000022h | 000122h | IFS0<7> | IEC0<7> | IPC1<14:12> |
| Timer3 | 8 | 000024h | 000124h | IFS0<8> | IEC0<8> | IPC2<2:0> |
| Timer4 | 27 | 00004Ah | 00014Ah | IFS1<11> | IEC1<11> | IPC6<14:12> |
| Timer5 | 28 | 00004Ch | 00014Ch | IFS1<12> | IEC1<12> | IPC7<2:0> |
| UART1 Error | 65 | 000096h | 000196h | IFS4<1> | IEC4<1> | IPC16<6:4> |
| UART1 Receiver | 11 | 00002Ah | 00012Ah | IFS0<11> | IEC0<11> | IPC2<14:12> |
| UART1 Transmitter | 12 | 00002Ch | 00012Ch | IFS0<12> | IEC0<12> | IPC3<2:0> |
| UART2 Error | 66 | 000098h | 000198h | IFS4<2> | IEC4<2> | IPC16<10:8> |
| UART2 Receiver | 30 | 000050h | 000150h | IFS1<14> | IEC1<14> | IPC7<10:8> |
| UART2 Transmitter | 31 | 000052h | 000152h | IFS1<15> | IEC1<15> | IPC7<14:12> |

Slika 3 IVT (*Interrupt Vector Table*)