Tampereen ammattiopisto
Pyynikin ammattioppilaitos

KÄVIJÄLASKURI

Opinnäytetyö
Sähköalan
perustutkinto/lukio
Elektroniikka-asentaja
Marko Viitanen
31.10.2003

SISÄLLYS

1 Johdanto

Tämän työn tarkoituksena on osoittaa kykyni omaan koulutukseeni liittyvässä itsenäisesti tehdyssä projektissa.

Sain idean tähän työhön kuultuani luokkatoverini ja opettajan välisen keskustelun, jossa luokkatoverini mietti, minkä työn tekisi. Idea ei ole kuitenkaan suora kopio, mutta siitä lähti ajatus tähän työhön.

Alun perin ideoin tekeväni laitteeseen tunnistuksen, jolla selviäisi kumpaan suuntaan kävijälaskurin ohi kuljetaan, jotta voitaisiin laskea, kuinka monta ihmistä olisi vielä huoneen sisällä. Luovuin kuitenkin ideasta, koska tästä tunnistuksesta ei olisi kovinkaan suurta hyötyä, ja kytkentään olisi tarvittu enemmän 7-segmenttinäyttöjä kaikkien lukujen näyttämiseen. Kytkennästä olisi myös tullut paljon monimutkaisempi sekä kalliimpi tarvittavien lisäkomponenttien takia.

PIC-Mikrokontrolleria käytän työssäni, koska huomasin sen soveltuvan tällaiseen toimintaan juuri olleella Sulautetut Järjestelmät –kurssilla (Kyrölä 2003). Kurssin aikana teimme pienen kytkennän PIC:ä käyttäen, jolloin pääsin kokeilemaan, mihin se pystyy. Lisäksi internetistä löytyy runsaasti materiaalia koskien PIC ohjelmointia, joten ongelmia sen käytön suhteen tuskin tulisi.

2 Miten kävijälaskuri toimii

2.1 Kävijälaskurin yleinen toiminta

Kävijälaskurin idea on yksinkertainen, sen on tarkoitus laskea ohi kulkeneiden ihmisten määrän ja näyttää lukumäärän neljällä 7-segmenttinäytöllä, jolloin maksimikävijämäärä, jonka se voi näyttää on 9999. Kävijälaskuri voidaan asentaa esimerkiksi ulko-oveen laskemaan siitä kulkeneiden ihmisten määrää.

Kävijälaskurin toiminta perustuu PIC-mikrokontrollerin kykyyn laskea pulsseja. Laskurissa on vastaanottopuoli sekä lähetinpuoli, lähetin laitetaan oven toiseen laitaan ja tähdätään tarkasti toisessa laidassa olevaan vastaanottimeen. Kun ovesta kuljetaan, lähettimen ja vastaanottimen välillä oleva säde katkeaa, jolloin komparaattorikytkentäinen operaatiovahvistin vaihtaa lähdön tilan nollata voltista viiteen volttiin. PIC-mikrokontrolleri huomaa tämän muutoksen, lisää yhden kävijän muistiinsa ja piirtää uuden kävijämäärän 7-segmenttinäytöille.

Komparaattorikytkennän herkkyyttä voi säätää kytkennässä olevalla trimmerillä, jolloin pienempikin valomäärä riittää lähetyspuolelta, jotta laskuri toimisi. Kytkentä ei kuitenkaan saa olla liian herkkä, jolloin normaali sisävalaistus riittäisi saamaan kytkennän aktiiviseksi.

2.2 Komparaattori

Koska IR-transistoria, joka toimii kuten transistori, paitsi että kantaa ohjataan infrapunan tai muun lähes saman taajuisen valon avulla, ei anna ulos kunnon jännitevaihtelua, oli minun tehtävä LM324-operaatiovahvistinpiirillä toteutettu komparaattori. OP-vahvistimessa on viisi kytkettävää jalkaa: kääntävä sisäänmeno, ei-kääntävä sisäänmeno, ulostulo, positiivinen jännite ja negatiivinen jännite. Sen toiminta perustuu siihen, että kahteen sisäänmenoon annetaan jännitettä, kun toinen jännite kasvaa toista isommaksi, piiri antaa ulostuloon joko 0 V tai 5 V, riippuen kummassa näistä jaloista isompi jännite on. (National Semiconductor 2000)

Molemmat sisäänmenot kytketään vastuksen kautta jännitteeseen. Toinen sisäänmeno kytketään lisäksi trimmerin kautta maihin, ja toinen IR -transistorin kautta maihin. Tällä tavalla kytkettäessä vaikuttaa IR-transistoriin tuleva infrapunan tai muun lähes saman taajuisen valon määrä siihen, antaako operaatiovahvistin 0 V vai 5 V ulostuloon. Ja koska toinen jalka on kytketty trimmerin kautta maihin, voidaan sillä säätää laitteen herkkyyttä.

Komparaattorikytkentään voisi tehdä hystereesiä kytkemällä ulostulon tietynkokoisen vastuksen kautta sisäänmenoon, jolloin jännite ei pääsisi heilumaan ulostulossa puolelta toiselle, vaan sisäänmenoihin tulevan jännitteen tarvitsisi muuttua hieman enemmän, jotta ulostulon tila vaihtuisi. Koska kytkentä ei ole vielä valmis, en tiedä onko tämä tarpeellista, mutta jos laskuri laskee yhden kävijän moneen kertaan, se pitää lisätä. Piirilevyn suunnittelussa on otettu tämä huomioon ja jätetty tälle komponentille valmis paikka.

2.3 Yleistä ohjelmoinnista

PIC mikrokontrollerin ohjelma on koodattu PIC:n omalla Assembly -ohjelmointikielellä. Koodin olisi voinut tehdä myös C:llä, mutta silloin ohjelman koosta olisi voinut tulla liian iso ja ehkä myös hitaampi. Puhtaalla Assemblyllä ohjelmoimalla saa koodista juuri sellaisen kuin haluaa, eikä siihen tule kääntämisestä johtuvia ylimääräisiä käskyjä. Myös tietynpituisen hidastuksen tekeminen olisi hyvin vaikeaa muilla kielillä.

Koodaus tapahtui käyttäen Microchipin toimittamaa MPLAB IDE -ohjelmaa, jota olimme koulussakin käyttäneet. Ohjelmalla konekielinen koodi käännetään HEX –muotoon konekieleksi, jolloin se on avattavissa IC-PROG -ohjelmalla, jolla tämä HEX muotoinen ohjelma siirretään PIC:n muistiin binaareina.

Ohjelma siirretään PIC:n muistiin PC:hen liitettävän erillisen ohjelmointilaitteen avulla, johon PIC-piiri asetetaan ja tämän jälkeen annetaan ohjelman hoitaa loput. Tämän prosessin yhteydessä PIC:n muisti nollataan ja ohjelmoinnin jälkeen varmistetaan, että ohjelmointi sujui onnistuneesti. Ohjelma voidaan uusia tarvittaessa tuhansia kertoja eikä uutta piiriä tarvitse ostaa. Ohjelmointilaitteen rakensin luokkatoverilta saamieni

neuvojen perusteella, jotka hän oli löytänyt internetistä ja pienellä vaivalla muokannut omia tarpeitaan vastaavaksi. Tämä ohjelmointilaite tukee lisäksi montaa eri PIC-tyyppiä. (Kyrölä 2003)

2.4 Ohjelmakoodin toiminta

Ohjelma alkaa yleisillä määrittelyillä, joissa esimerkiksi nollataan piirin ulostulot ja määritellään, mitkä portit toimivat sisäänmenoina ja mitkä ulostuloina. Tämän jälkeen tallennetaan ohjelmassa määritellyt 7-segmenttinäytön numeroiden binaarikoodit PIC:n muistiin numerojärjestyksessä myöhempää käyttöä varten. Myös keskeytykset laitetaan päälle tässä vaiheessa.

Itse pääohjelmassa piirretään näytöille muistipaikoista löytyvät numerot. Piirto tapahtuu siten, että ensin siirretään W-rekisteriin 7-segmenttinäytön 0-kuviota merkitsevän muistinpaikan osoite, johon lisätään toisen muistipaikan sisältö joka määrää mones numero on kyseessä. Eli jos kyseessä on numero 5, W-rekisteriin laitetaan 0:n muistipaikka +5, joka on samalla 7-segmenteille tulevan kuvion 5 osoite. Kun tämä muistipaikan osoite tallennetaan FSR-rekisteriin, voidaan INDF-rekisteristä lukea muistipaikan arvo, joka kirjoitetaan portti B:hen. Kun tämä on tehty, aktivoidaan vielä portti A:sta sen näytön bitti, jota halutaan ohjata. Sen jälkeen ohjelmassa on pieni viive, jotta luku ehtisi näkyä näytöllä. (INOUE, SEIICHI 1998)

Viive on tehty käyttäen kahta muistipaikkaa. Ensin toisen arvo muutetaan joksikin arvoksi, joka määrää viiveen pituuden. Tämän jälkeen tuleva koodi merkitään otsikolla Viive_loop1, jotta siihen voidaan palata myöhemmin. Myös toisen muistipaikan arvo muutetaan halutuksi arvoksi. Merkitsemme seuraavaa koodia otsikolla Viive_loop2. Kun nyt ruvetaan vähentämään toisesta muistipaikasta, joka viimeksi määriteltiin, yksi kerrallaan lukuja, tarkastetaan jokaisen vähennyksen jälkeen onko luku 0. Jos luku ei ole 0, palaa ohjelma koodin kohtaan Viive_loop2. Jos luku on 0, vähennetään nyt ensimmäisestä muistipaikasta yksi, ja tarkastetaan onko se 0. Jos ensimmäinen muistipaikka ei ole 0, palataan koodissa taaksepäin kohtaan Viive_loop1, jolloin toiseen muistipaikkaan lisätään taas haluttu luku. Jos taas ensimmäinen muistipaikka

on 0, palaa koodi kohtaan josta kutsuttiin viivettä, ja jatkaa toimintaansa. (Microchip 2003)

Kun PIC saa ulkopuolisen pulssin RB0-porttiin, eli keskeytyksen, hyppää ohjelma paikkaan 0x04, joka vastaa hexadesimaalilukua 4, riippumatta siitä missä se on menossa. Tästä paikasta hypätään aliohjelmaan, jossa lisätään kävijöiden määrää. Tämän ajaksi laitetaan keskeytykset pois käytöstä jottei PIC hyppäisi uudestaan samaan paikkaan. Kun kävijä on lisätty, laitetaan keskeytykset uudestaan päälle ja palataan takaisin paikkaan, jossa viimeksi oltiin.(Mikroelektronika 2003)

Ohjelmakoodi tulee liitteeksi numero 3, kun se tulee valmiiksi.

2.5 Yleiset ongelmat

Laskurissa eräänä ongelmana on se, miten saisi laskettua eri kävijät mahdollisimman tarkasti. Tätä voi koittaa saada korjattua säätämällä trimmeristä kytkennän herkkyyttä, ja mahdollisesti lisäämällä IR-LED:n tehoa. Tämän lisäksi voi joku tahallaan heiluttaa kättään kävijälaskurin valokeilassa, jolloin kävijöiden määrä nousee. Tehokkain tapa lisätä tahallaan kävijöitä olisi osoittaa IR-transistoria kaukosäätimellä, josta tulee suuritaajuista signaalia. Lisäksi jos kytkentä on säädetty tarpeeksi herkälle ja se ottaa vastaan loisteputkista tulevan valon, saa se niistä 50 Hz taajuudella valoa, jolloin kävijöitä ilmestyy lisää 50 kappaletta sekunnissa. Tämän vuoksi kytkennän IR-transistori onkin suojattu ulkopuolista valoa vastaan laittamalla sen päälle pala kutistesukkaa, joka on lämmön vaikutuksesta kutistuvaa muovimateriaalia. Tämä taas vaikeuttaa hieman IR-transistorin kohdistusta IR-LED:iin, mutta tämä ei ole kovin suuri ongelma.

Myös virrankulutus on suuri ongelma: sekä lähettimelle että itse kytkennän puolelle tarvitsee oman pariston. Lähetin käyttää n. 1 V ja 100 mA, kytkentä n. 9 V 150 mA. Paristoja tarvitsisi täten vaihtaa lähes joka päivä. Parempi ratkaisu olisi saada kytkentään verkkovirtaa muuntajan läpi, jolloin tällainen virrankulutus ei haittaisi. Testikäyttö on kuitenkin pakko tehdä paristoja käyttäen, koska minulla ei ole muuntajia käytettävissäni juuri tällä hetkellä.

3 Kävijälaskurin valmistus

3.1 Kytkentä

3.1.1 Suunnittelu

Kytkennän suunnittelu on aloitettu aivan alusta. Internetistä ei löytynyt mitään vastaavia kytkentöjä, joten minun piti toteuttaa kytkentä keräämällä itse materiaali sitä varten. Tärkeimpänä komponenttina kytkennässä toimii PIC Mikrokontrolleri. Suunnittelu alkoi ensin luonnoksilla, joilla koitin selvittää mitä komponentteja kytkentään tarvitsisi ja mitä kytkennän tulisi tehdä.

Valmiin kytkennän (Liite 1) suunnittelin koulussa PowerLogic-ohjelmalla, jota olemme käyttäneet aiemminkin. Suunnittelu tapahtuu lisäämällä ohjelmaan tarvittavat komponentit ja sen jälkeen liittämällä ne toisiinsa halutulla tavalla. Kaikkia komponentteja ohjelmasta ei löydy, mutta niitä voi simuloida eri komponenteilla joissa vain on samanlainen jalkasijoittelu. Koska 7-segmenttinäyttöjä ei ohjelmasta löytynyt, jouduin tekemään jokaista näyttöä varten kahdeksan liitäntänastaa.

3.1.2 Komponenttien mitoitus

Koska komponenttien arvot riippuvat hyvin paljon siitä, millaisen kytkennän tekee, oli kaikkien komponenttien arvot mietittävä erikseen. Vastukset oli mitoitettava siten, että 7-segmenttinäyttöjen kirkkaus olisi riittävä, mutta virtaa ei pääsisi näyttöihin liikaa jolloin ne voisivat hajota. Myös transistorien ohjaukseen käytetyt vastukset oli katsottava tarkkaan jotta ohjaus onnistuisi ilman liian suurta tehonkäyttöä. PIC Mikrokontrollerin kiteen arvo vaikuttaa myös suuresti ohjelman toimintaan, valitsin sen mahdollisimman suureksi jotta ohjelma toimisi nopeasti.

Komparaattorikytkentään oli myös valittava huolella vastukset. Trimmerissä oli oltava tarpeeksi suuri säätövara, jotta kytkennästä saisi riittävän herkän. Selvittääkseni tarvittava trimmerin arvo, oli minun mitattava IR -transistori eri valoisuustasoilla.

3.1.3 Komponenttien hankinta

Yrittäessäni minimoida kytkennän tekokustannukset, on osa osista otettu suoraan vanhoista videonauhureista. Esimerkiksi PIC:lle kellopulssia antava 10 MHz kide on juotettu irti vanhasta videonauhurista, joka löytyi koululla olevasta romukopasta. Myöskin kytkennässä oleva IR-transistori on napattu samanlaisesti, tämä onkin hyvä asia sillä tätä komponenttia ei löytynyt tarkastamistani elektroniikkamyymälöistä. Tässä oleva ainoa haittapuoli on, että kyseisille komponenteille ei välttämättä löydy mistään tarkempia tietoja, joten arvot oli selvitettävä tekemällä sarja testejä. 7-segmenttinäytöt löysin kotoani, mutta niistäkään ei ollut mitään dokumentointia olemassa, joten selvitin näyttöjen tyypin ja jalkajärjestyksen kytkemällä kaksi AA-paristoa yhteen ja koittamalla kytkeä niitä eri jalkoihin, lopulta yksi segmentti syttyi ja totesin näyttöjen olevan yhteiskatodikytkettyjä. Loput segmentit sai helposti selvitettyä laittamalla positiivinen jännite eri nastoihin.

3.1.4 Koekytkentä

Koekytkennän toteutin koekytkentälevylle, jossa on kuparipuolella kytketty kaikki pisteet pitkittäissuunnassa yhteen. Levy on täynnä reikiä, joihin komponentit asetetaan. Ylimääräiset pitkittäissuuntaiset kuparit voi poistaa esimerkiksi puukkoa apuna käyttäen.

Koska minulla ei ollut tietoa, miten komparaattori toimisi, minun piti testata tätä kytkentää erillisenä. Komparaattorin ulostuloon laitoin LEDin jotta näkisin varmasti, koska ulostulossa on jännite. Kytkentä tuntui toimivan täydellisesti ja pienen testaamisen jälkeen totesin, että 2 m päässä 1 V ja 100 mA käyttävä IR-LED riittää aivan hyvin.

Ongelmia koekytkennässä tuli heti kun lisäsin siihen PIC-piirin sekä yhden 7-segmenttinäytön. Pulsseja ei tuntunut tulevan ollenkaan. Tarkemmissa selvittelyissä huomasin, ettei käyttämäni operaatiovahvistinpiiri uA741 antanut kunnon nollatasoa, vaan kun sen piti olla nolla, tuli jännitettä 1 V. uA741 on ilmeisesti suunniteltu kaksipuoleiselle virtalähteelle, eikä se täten toimi oikein yksipuoleisella jännitteellä.

Tämän ongelman sain ratkaistua lisäämällä OP-vahvistimen lähtöön LEDin, jonka kynnysjännitteen ansiosta 1 V:n jännite ei päässyt ollenkaan läpi, ja OP -vahvistimen ohjaama transistori sai tarvitsemansa nollatason. Oikeassa työssä käytän LM324 OP-vahvistinpiiriä, jonka pitäisi toimia yksipuoleisellakin virtalähteellä (National Semiconductor 2000). Jos ongelmia tulee, tiedän kuitenkin missä vika voi olla.

Pienten ongelmien jälkeen koekytkentä rupesi toimimaan, kuten sen oli tarkoitus. Testaus oli tärkeätä, koska muuten olisin saattanut tehdä suunnitelmiini virheen, ja minun olisi pitänyt jälkeenpäin ruveta muokkaamaan valmista kytkentää, ja tämä olisi ollut hieman työlästä.

3.2 Piirilevy

3.2.1 Suunnittelu

Piirilevyn kuparointi ja komponenttisijoittelu (Liite 2) ovat suunniteltu käyttäen PowerPCB-ohjelmaa, jonka käytöstä olemme saaneet koulussa opetusta. Koska PowerLogic ja PowerPCB ovat saman valmistajan tekemiä, voi PowerLogic suunnitelman avata PowerPCB:hen, jolloin itselle jää tehtäväksi vain komponenttien sijoittelu ja kuparien vetäminen. Tämä on kuitenkin hyvin työlästä, ja aikaa minulta kului tähän lähes neljä tuntia.

Ongelmia minulle tuottivat eniten 7-segmenttinäytöt, joista jokaiseen tarvitsi kahdeksasta eri paikasta johdot. Lopulta sain suunniteltua piirilevyn siten, että hyppylankoja tarvitsi vain noin kymmenen.

Ensimmäisenä työnä oli levittää komponentit, ja katsoa mitkä komponentit kuuluivat lähekkäin ja asetella ne paremmin. Seuraavaksi vedetään kupareita tiettyjen pisteiden väliin, jotta nähdään voisiko osat sijoitella vielä paremmin. Kun kaikki kuparit ovat vedetty, nähdään paljon hyppylankoja työhön tullaan tarvitsemaan. Viimeiseksi sijoittelua voi vielä optimoida, jotta ylimääräisistä hyppylangoista päästäisiin eroon ja piirilevyn koosta saataisiin pienempi.

3.2.2 Piirilevyn valmistus

Piirilevy valmistetaan käyttäen koululta löytyviä laitteita. Ensimmäisenä valoherkälle piirilevylle valotetaan kytkennän piirikaavio, jossa on piirilevylle tulevat kuparit. Tämän jälkeen piirilevy uitetaan kehitysnesteessä ja laitetaan syövytykseen. Tämä vaihe kestää noin 15 minuuttia. Kun piirilevy on syöpynyt, hiotaan kupareista pois valoherkkä kalvo, jotta juottaminen olisi mahdollista, ja suojataan kuparit juotosaktiivisella fluxilla.

4 Arvioiva päätäntä

Kokonaisuudessaan työn tekeminen oli hyvin opettavainen kokemus jota tehdessäni sain hyvää kokemusta sekä PIC Mikrokontrollerin että OP -vahvistimen käytöstä monimutkaisissa kytkennöissä.

Mielestäni työn tekeminen onnistui paremmin kuin odotin. Jouduin luopumaan joistakin tavoitteistani, mutta muuten työn toteutus vastaa alkuperäistä suunnitelmaa. Jos tekisin työn uudestaan, koittaisin ehkä suunnitella kytkennän hieman erilailla komponenttien minimoimiseksi.

Tällainen suurimmaksi osaksi itse suunniteltu ja toteutettu työ auttaisi varmasti jokaista ymmärtämään hieman paremmin komponenttien toimintaa, suoraan kopioimalla en olisi oppinut juuri mitään.

Lähteet

KYRÖLÄ, ANTERO 2003: Sulautetut Järjestelmät –kurssi. Lokakuu 2003.

Microchip 1998: PIC16F84 Data Sheet. http://www.microchip.com. Elokuu 2003.

National Semiconductor 2000: LM324 Data Sheet. http://www.national.com. Syyskuu 2003.

Mikroelektronika 2003: PIC Microcontrollers – Online. http://www.mikroelektronika.co.yu/english/. Syyskuu 2003.

INOUE, SEIICHI 1998: Count-down timer. http://www.interq.or.jp/japan/se-inoue/e_pic6_3.htm. Syyskuu 2003

LIITE 1, ohjelmakoodi

```
;"Kävijälaskuri" - Marko Viitanen

;1: bc 2: abged 3: abcdg 4: bcfg 5: afgcd
;6: acdefg 7: abc 8: abcdefg 9: abcdfg 0: abcdef
;Määrittelyitä
patr_yks     equ b'00001100'
patr_kaks     equ b'10110110'
patr_kolme     equ b'10011110'
patr_nelja    equ b'11001100'
patr_viis     equ b'11011010'
patr_kuus     equ b'11111010'
patr_seitteman   equ b'00001110'
patr_kaheksan   equ b'11111110'
patr_yheksan   equ b'11011110'
patr_nolla     equ b'01111110'

;Muuttujille varataan muistialueet
looptemp       equ 0x10
looptemp2      equ 0x11
counter        equ 0x12

ykkoset        equ 0x13
kympit         equ 0x14
sadat          equ 0x15
tuhannet       equ 0x16
naytto         equ 0x17

nolla          equ 0x18
yks            equ 0x19
kaks           equ 0x1a
kolme          equ 0x1b
nelja          equ 0x1c
viis           equ 0x1d
kuus           equ 0x1e
seitteman      equ 0x1f
kaheksan       equ 0x20
yheksan        equ 0x21

temp_w         equ 0x22
temp_s         equ 0x23

  LIST P=16F84, F=INHX8M  ;Mikrokontrollerin tyypin määritys

  __CONFIG 0x3ff2         ;Mikrokontrollerin asetukset hexoina
  include "P16F84.inc"    ;Sisällytetään mikrokontrollerin kirjasto

  ORG     0x00            ;ohjelman alku
    goto alku        ;hypätään keskeytyksen yli
  ORG 0x04     ;keskeytys osoite, tänne ohjelma hyppää kun RB0 saa pulssin
```

```
    goto keskeytys   ;Hyppää keskeytykseen
  ORG 0x05     ;Ohjelma alkaa tästä
alku
  ;hieman viivettä käynnistykseen
  call viive
  call viive
  call viive
  call viive
  call viive
  call viive
  call viive
  call viive
  call viive
  call viive
  call viive
  call viive

  ;Muistipankkia pitää vaihtaa koska TRISA ja TRISB
  ;sijaitsevat muistipankissa yksi.
  bsf    STATUS, RP0 ;Vaihdetaan muistipankkia ykköseksi
  clrf   TRISA       ;Nollataan A-portit ulostuloiksi
  clrf   TRISB       ;Nollataan B-portit ulostuloiksi
  bsf     TRISB,0     ;RB0 portti sisäänmenoksi
  bcf    STATUS, RP0 ;Muistipankki takaisin nollaksi

  clrf   PORTB  ;Nollataan B-porttien tilat
  clrf   PORTA  ;Nollataan A-porttien tilat
  MOVLW   0x01  ;luku 01H W-rekisteriin
  MOVWF   counter ;W-rekisterin sisältö counter-muuttujan osoittamaan muistipaikkaan
  MOVWF   naytto ;W-rekisterin sisältö naytto-muuttujan muistipaikkaan
  ;Nollataan kaikkien näyttöjen lukemat
  clrf  ykkoset
  clrf  kympit
  clrf  sadat
  clrf  tuhannet

  ;Tallennetaan eri 7-segmenttinäytöille laitettavien lukujen bitit muistipaikkoihin
        ;joista ne voidaan myöhemmin ottaa kätevästi esiin
  movlw  patr_yks
  movwf  yks
  movlw  patr_kaks
  movwf  kaks
  movlw  patr_kolme
  movwf  kolme
  movlw  patr_nelja
  movwf  nelja
  movlw  patr_viis
  movwf  viis
  movlw  patr_kuus
  movwf  kuus
  movlw  patr_seitteman
  movwf  seitteman
```

```
  movlw  patr_kaheksan
  movwf  kaheksan
  movlw  patr_yheksan
  movwf  yheksan
  movlw  patr_nolla
  movwf  nolla
  bsf 81h,INTEDG    ;Keskeytys pulssin nousevalla reunalla
  bsf INTCON, INTE ;Ulkoinen (Portti RB0) keskeytys päälle
  bsf INTCON, GIE  ;Kaikki keskeytykset päälle

  bcf     STATUS    , DC
;Ohjelmalooppi
ohjelmanalku
  CALL    Piirto       ;kutsuu Piirto-aliohjelmaa
  GOTO    ohjelmanalku ;Palaa alkuun

Piirto              ;Piirto-aliohjelma

  movlw b'00001000' ;Ensimmäisen näytön valitseva bittikuvio
  movwf PORTA       ;Ensimmäisen näytön valitseva bitti päälle PORTA:ssa
  movlw   nolla     ;Nolla-kuvion muistipaikan osoite W-rekisteriin
  addwf   ykkoset,W ;Lisätään W-rekisteriin ensimmäisen luvun numeroarvo
  movwf   FSR       ;Siirretään W-rekisterin sisältö FSR-rekisteriin
  movf    INDF,W    ;Luetaan W-rekisteriin INDF-rekisterin arvo
                    ;joka on FSR-rekisteriin siirretyn muistipaikan osoitteen arvo
  MOVWF   PORTB     ;Siirretään arvo PORTB:hen
  call    viive     ;Viivettä jotta luku ehtii näkyä näytöllä

  movlw b'00000100' ;Toisen näytön valitseva bittikuvio
  movwf  PORTA      ;Toisen näytön valitseva bitti päälle PORTA:ssa
  movlw   nolla     ;Nolla-kuvion muistipaikan osoite W-rekisteriin
  addwf   kympit,W  ;Lisätään W-rekisteriin toisen luvun numeroarvo
  movwf   FSR       ;Siirretään W-rekisterin sisältö FSR-rekisteriin
  movf    INDF,W    ;Luetaan W-rekisteriin INDF-rekisterin arvo
                    ;joka on FSR-rekisteriin siirretyn muistipaikan osoitteen arvo
  MOVWF   PORTB     ;Siirretään arvo PORTB:hen
  call    viive     ;Viivettä jotta luku ehtii näkyä näytöllä

  movlw b'00000010' ;Kolmannen näytön valitseva bittikuvio
  movwf  PORTA      ;Kolmannen näytön valitseva bitti päälle PORTA:ssa
  movlw   nolla     ;Nolla-kuvion muistipaikan osoite W-rekisteriin
  addwf   sadat,W   ;Lisätään W-rekisteriin toisen luvun numeroarvo
  movwf   FSR       ;Siirretään W-rekisterin sisältö FSR-rekisteriin
  movf    INDF,W    ;Luetaan W-rekisteriin INDF-rekisterin arvo
                    ;joka on FSR-rekisteriin siirretyn muistipaikan osoitteen arvo
  MOVWF   PORTB     ;Siirretään arvo PORTB:hen
  call    viive     ;Viivettä jotta luku ehtii näkyä näytöllä

  movlw b'00000001' ;Neljännen näytön valitseva bittikuvio
  movwf  PORTA      ;Neljännen näytön valitseva bitti päälle PORTA:ssa
  movlw   nolla     ;Nolla-kuvion muistipaikan osoite W-rekisteriin
  addwf   tuhannet,W;Lisätään W-rekisteriin toisen luvun numeroarvo
```

```
  movwf   FSR       ;Siirretään W-rekisterin sisältö FSR-rekisteriin
  movf    INDF,W    ;Luetaan W-rekisteriin INDF-rekisterin arvo
                    ;joka on FSR-rekisteriin siirretyn muistipaikan osoitteen arvo
  MOVWF   PORTB     ;Siirretään arvo PORTB:hen
  call    viive     ;Viivettä jotta luku ehtii näkyä näytöllä

  return            ;Palaa takaisin




keskeytys                 ;Keskeytys-aliohjelma

  BCF    INTCON, INTF   ;Nollataan keskeytyslippu
  BCF    INTCON, GIE    ;Poistetaan keskeytykset käytöstä
  movwf   temp_w         ;Tallennetaan W-rekisterin arvo temp_w:n näyttämään muistipaikkaan
  movf    STATUS, W      ;Siirretään STATUS-rekisterin arvo W-rekisteriin
  movwf   temp_s         ;Tallennetaan W-rekisterin arvo temp_w:n näyttämään muistipaikkaan


  movlw b'00000000'      ;Tyhjennetään näyttö
  movwf PORTB


  incf    ykkoset,1      ;Lisätään ensimmäiseen lukemaan yksi
  movf    ykkoset,w      ;Siirretään lukema W-rekisteriin
  sublw   .10            ;Vähennetään kymmenestä W-rekisterin arvo
  btfss   STATUS,Z       ;Tuliko tulokseksi nolla?
  goto    kesk_lop       ;jos ei niin keskeytyksen loppuun
  clrf  ykkoset        ;Jos tulos nolla niin nollataan lukema
  incf    kympit,1       ;Lisätään seuraavaan lukemaan yksi
  movf    kympit,W       ;Siirretään lukema W-rekisteriin
  sublw   .10            ;Vähennetään kymmenestä W-rekisterin arvo
  btfss   STATUS,Z       ;Tuliko tulokseksi nolla?
  goto    kesk_lop     ;jos ei niin keskeytyksen loppuun
  clrf    kympit         ;Jos tulos nolla niin nollataan lukema
  incf    sadat,1        ;Lisätään seuraavaan lukemaan yksi
  movf    sadat,W        ;Siirretään lukema W-rekisteriin
  sublw   .10            ;Vähennetään kymmenestä W-rekisterin arvo
  btfss   STATUS,Z       ;Tuliko tulokseksi nolla?
  goto    kesk_lop       ;jos ei niin keskeytyksen loppuun
  clrf    sadat          ;Jos tulos nolla niin nollataan lukema
  incf    tuhannet,1     ;Lisätään seuraavaan lukemaan yksi
  movf    tuhannet,W     ;Siirretään lukema W-rekisteriin
  sublw   .10            ;Vähennetään kymmenestä W-rekisterin arvo
  btfss   STATUS,Z       ;Tuliko tulokseksi nolla?
  goto    kesk_lop     ;jos ei niin keskeytyksen loppuun
  clrf    tuhannet       ;Jos tulos nolla niin nollataan lukema

kesk_lop
```

```
  movf    temp_s, W      ;Siirretään temp_s:n osoittaman muistipaikan arvo W-rekisteriin
  movwf   STATUS         ;Siirretään W-rekisterin arvo STATUS-rekisteriin
  swapf   temp_w, F      ;Käännä temp_w:n osoittaman muistipaikan puolitavut
  swapf   temp_w, W      ;Käännä puolitavut takaisin ja tallenna W-rekisteriin

  bsf     INTCON, GIE    ;Keskeytykset päälle

  retfie                 ;Palataan takaisin ohjelmaan

;Viive-aliohjelma
viive       movlw  .2            ;Desimaaliluku 2 W-rekisteriin
            movwf   looptemp     ;W-rekisteri looptempin osoittamaan muistipaikkaan
loophyp1    movlw  .80           ;Desimaaliluku 80 W-rekisteriin
            movwf   looptemp2    ;W-rekisteri looptemp2:n osoittamaan muistipaikkaan
loophyp2    decfsz  looptemp2, f ;Vähennetään looptemp2:sta yksi ja tarkastetaan tuliko nolla
            goto    loophyp2     ;Jos ei nolla niin hyppää takaisin kohtaan loophyp2
            decfsz  looptemp , f ;Jos nolla, vähennä looptemppiä ja tarkasta tuliko nolla
            goto    loophyp1     ;Jos ei nolla, hyppää kohtaan loophyp1
            return               ;Jos nolla, palaa ohjelmaan

END
```