Timo Väyrynen

PSK31-ERILLISMODEEMI

Insinöörityö
Kajaanin ammattikorkeakoulu
Tekniikan ja liikenteen ala
Tietotekniikan koulutusohjelma
Kevät 2005

Kajaanin ammattikorkeakoulu

INSINÖÖRITYÖ
TIIVISTELMÄ

Osasto	Koulutusohjelma
Tekniikka	Tietotekniikka
Tekijä(t)	
Timo Väyrynen	
Työn nimi	
PSK31-Erillismodeemi	
Vaihtoehtoiset ammattiopinnot	**Ohjaaja(t)**
Langaton tiedonsiirto	Jukka Heino
Aika	**Sivumäärä**
Kevät 2005	28 + 14

Tiivistelmä

Insinöörityön aiheena oli rakentaa PC-tietokoneen sarjaportin ja radiolähetin-vastaanottimen välille liitettävä ulkoinen modeemi, joka mahdollistaa radio-amatööreille tyypillisen kahden tietokoneen välisen tekstimuotoisen keskustelun radio-aaltojen välityksellä.

Modeemin tiedonsiirtonopeus on 31,25 bittiä sekunnissa, ja se käyttää vaiheensiirtoavainnus-modulaatiota. Sillä voidaan lähettää ja vastaanottaa noin 50 sanaa minuutissa Varicode-nimisen tiedonkoodausmenetelmän ansiosta.

Modeemi on rakennettu Atmelin AT89LV51-prosessorin ympärille käyttäen pohjana Kajaanin ammattikorkeakoulun prosessorikorttia. Kortille tarvittavat lisäkytkennät tehtiin osin kiertoliitoksin, osin juottamalla. Modeemin ohjelmisto tehtiin C-ohjelmointikielellä.

Luottamuksellinen	
Kyllä	
Ei	X
Hakusanat	
Varicode, modeemi, vaiheensiirtoavainnus	
Säilytyspaikka	
Kajaanin ammattikorkeakoulu	

Kajaanin ammattikorkeakoulu
Kajaani Polytechnic

ABSTRACT
THESIS

Faculty	Degree programme
Faculty of Engineering	Information Technology

Author(s)
Timo Väyrynen

Title
An External Processor Based PSK31 Modem

Optional professional studies	Instructor(s) / Supervisor(s)
Wireless Communications	Jukka Heino
Date	**Total number of pages**
29 March 2005	28 + 14

Abstract

The purpose of this Bachelor's thesis was to design and build an external processor based PSK31 modem to be used by radio amateurs. PSK31 is a digital communications mode which is intended for live keyboard-to-keyboard conversations through radio waves. It uses Binary Phase Shift Keying modulation without error correction. Its data rate is 31.25 bps, which means about fifty words per minute.

PSK31 uses variable-length coding to improve efficiency in terms of the average character duration. This means that different characters are represented by a variable-length combination of bits where the shortest bit-lengths are used for the most common letters.

The equipment of the modem is based on commercial electronic components. The main component is the Atmel AT89LV51 microprocessor. The heart of the system is a computer program which has been written using the C programming language.

Confidential

Yes

No X

Keywords
Varicode, phase shift keying, modem, 8051 microprocessor

Deposited at
Kajaani Polytechnic

ALKUSANAT

Tämä insinöörityö on tehty Kajaanin ammattikorkeakoululle. Haluan kiittää kaikkia työn onnistumiseen vaikuttaneita henkilöitä. Erityisesti haluan kiittää työni valvojaa, lehtori Jukka Heinoa, jolta sain arvokkaita neuvoja kytkennän ja erityisesti ohjelmiston suunnittelussa. Kiitän myös lehtori Risto Airaksista ja laboratorioinsinööri Ismo Talusta monet pulmatilanteet ratkaisseista neuvoista ja vinkeistä. Luonnollisesti kiitän myös lehtori Eero Soinista ja yliopettaja Kaisu Korhosta työn kielellisestä ohjauksesta.

Kajaanissa 29.3.2005

Timo Väyrynen

SISÄLLYSLUETTELO

KÄYTETYT TERMIT

ASCII-KOODI	PC-tietokoneissa käytetty tapa, jossa jokaisen merkin esittämiseen käytetään kahdeksan bitin mittaista bittijonoa.
BITTI	Tiedon määrän perusyksikkö digitaalitekniikassa. Ykkönen tai nolla.
BPSK	Binary phase shift keying, binäärinen vaiheensiirto-avainnus, menetelmä, jossa jokainen lähetettävä nolla-bitti aikaansaa 180° vaihesiirron moduloituun kantoaaltoon.
KANTOAALTO	Suurtaajuinen radiolähete, johon siirrettävä informaatio moduloidaan.
MODULAATIO	Menetelmä, jonka avulla siirrettävä tieto voidaan sisällyttää kantoaaltoon.
VARICODE	Variable length coding, koodaustapa, jossa merkin esittämiseen käytettyjen bittien lukumäärä on kääntäen verrannollinen merkin yleisyyteen.

1 JOHDANTO

Langaton tiedonsiirto pitkillä yhteysetäisyyksillä ei olisi luonnollisesti mahdollista ilman radioaaltoja. Nykymuotoisen, ei puhelähetteillä tapahtuvan radioaaltoja hyödyntävän pitkän matkan langattoman tiedonsiirron voidaan katsoa alkaneeksi vuonna 1901, jolloin italialainen keksijä Guglielmo Marconi sähkötti kuuluisan S-kirjaimensa Euroopasta Atlantin yli Amerikkaan. Tämän jälkeen tiedonsiirtomenetelmät ovat kehittyneet hyvin paljon, mutta pääperiaatteiltaan nykypäivän langaton tiedonsiirto noudattaa jo yli sata vuotta sitten esitettyjä pääperiaatteita.

PSK31 on digitaalinen, radiokaukokirjoitusta muistuttava kapeakaistainen lähetysmoodi. Siitä on tullut viime aikoina yksi amatööritaajuuksien suosituimmista digitaalisista lähetysmoodeista. Yksi syy nopeaan suosion kasvuun löytyy kotitietokoneiden kasvaneista prosessointitehoista ja täysin ilmaiseksi internetistä ladattavissa olevista PSK31:n vastaanottoon ja lähetykseen suunnitelluista tietokoneohjelmista.

Tyypillinen PSK31:n lähetykseen ja vastaanottoon kykenevä laitteisto koostuu PC-tietokoneesta ja radiolähetin-vastaanottimeen liitetystä äänikortista, sekä PC-tietokoneella ajettavasta ohjelmasta. PSK31:n käyttämän vaiheensiirto-avainnuksen ohjelmallinen koodaaminen ja dekoodaaminen syö osansa käytettävissä olevasta prosessointitehosta, jonka vuoksi käytettävän PC-tietokoneen tulee olla suhteellisen tehokas.

Insinöörityön tarkoituksena oli suunnitella ulkoinen, PSK31:n lähetykseen ja vastaanottoon kykenevä modeemi, jonka lisäksi tarvitaan vain sarja-liikenteeseen kykenevä tietokone, tai muu laite, jossa on EIA232-standardin mukainen sarjaportti.

2 RADIOTEKNIIKKA

Radioaallot muodostuvat sähkö- ja magneettikentistä. Sähkömagneettinen säteily on poikittaista aaltoliikettä, joka etenee suoraviivaisesti valon nopeudella ($c \approx 300\,000\,km/s$) tyhjiössä. Sähkö- ja magneettikentät värähtelevät samanvaiheisina kohtisuorassa sekä toisiaan että etenemissuuntaansa vastaan. Aallonpituus • on suoraan verrannollinen värähtelytaajuuteen f kaavan 1 mukaisesti.

$$\mathrm{I} = \frac{c}{f} \qquad (1)$$

2.1 Kantoaalto

Kantoaallolla tarkoitetaan suurtaajuista sähkömagneettista säteilyä, joka muodostetaan radiolähettimellä. Itse kantoaalto ei sisällä mitään informaatiota, vaan on yleensä puhdasta siniaaltoa, johon varsinainen informaatio moduloidaan.

2.2 Modulointi

Moduloinnilla tarkoitetaan kantoaallon muuttamista niin, että vastaanottava pää tietää, mitä lähettävä pää näillä kantoaallon muutoksilla tarkoittaa. Yksinkertaisin moduloimalla lähetettävissä oleva tieto lienee lähettää vastaanottavalle päälle tieto siitä, lähetetäänkö kantoaaltoa vai ei. Tämä on periaatteessa yksinkertaisinta toteuttaa lähettämällä kantoaaltoa, kun halutaan kertoa vastaanottajalle kantoaallon olevan päällä, ja katkaista kantoaallon lähetys, kun vastaanottajan halutaan tietävän, että kantoaalto ei ole päällä. Tätä periaatetta voitaisiin soveltaa suoraan binääriseen tiedonsiirtoon esimerkiksi pitämällä kantoaaltoa päällä ykkös-bittiä lähetettäessä ja kantoaaltoa pois päältä nolla-bittiä lähetettäessä. Tämä menetelmä ei kuitenkaan sovi esimerkiksi analogisiin audiolähetyksiin, koska analogisissa lähetteissä audiosignaali voi periaatteessa olla missä tahansa tilassa ”nollan ja ykkösen välillä”. Tämän vuoksi tieto moduloidaan yleisimmin joko kantoaallon amplitudiin, taajuuteen tai vaiheeseen.

2.3 Vaiheensiirtoavainnus

Vaiheensiirtoavainnus on modulointimenetelmä, jossa siirrettävä informaatio moduloidaan kantoaallon vaiheeseen. Vaihemodulaatio sopii hyvin digitaaliseen tiedonsiirtoon, sillä yksinkertaisimmillaan vaihemodulaatio vain kääntää kantoaallon vaihetta 180° sovitulla symbolirajalla. Vaihemodulaation avulla voidaan periaatteessa toteuttaa hyvinkin monimutkaisia modulaatioratkaisuja lyhentämällä vaihesiirtoa esimerkiksi 90° tai 45°. Vaihesiirron lyhentäminen kuitenkin kasvattaa virheriskiä, jonka vuoksi luotettavuus kärsii. Käytettäessä vaiheensiirtoavainnusta hyvin häiriöisillä radiotaajuuksilla joudutaan yleensä tyytymään kaksivaiheiseen avainnukseen, jossa vaihe kääntyy 180°.

2.4 Modulaatio PSK31:ssä

Nimensä mukaisesti PSK31:ssä tieto moduloidaan kantoaallon vaiheeseen nopeudella 31,25 bittiä sekunnissa. Käytössä on kaksivaiheinen binaarinen differentiaalinen vaiheensiirtoavainnus. Kaksivaiheisuus tarkoittaa tässä sitä, että kantoaallon vaihe joko kääntyy 180° (eli polariteetti vaihtuu) symbolirajalla tai pysyy ennallaan. Differentiaalisuus tarkoittaa sitä, ettei polariteetilla itsellään ole merkitystä, vaan nolla tarkoittaa polariteetin kääntymistä ja ykkönen vakiona pysymistä.

2.5 Varicode

Variable length coding, lyhyemmin Varicode on Peter Martinez -nimisen radioamatöörin kehittämä tiedonkoodausmenetelmä. Varicode perustuu löyhästi jo aikanaan lennättimessä käytettyyn tapaan esittää yleisimmin käytetyt merkit mahdollisimman lyhyessä ajassa, eli käyttämällä merkin lähetykseen mahdollisimman vähän ”pisteitä ja viivoja”. Tätä voidaan soveltaa suoraan nykypäivän digitaalitekniikkaan korvaamalla pisteet ja viivat ykkösillä ja nollilla.

Varicode määrittelee kaikki ASCII-koodiston mukaiset 128 merkkiä uusiksi esittäen eniten käytetyt pienet kirjaimet huomattavasti vähemmällä bittimäärällä kuin harvemmin käytetyt isot kirjaimet. Myös pienet ja isot kirjaimet on vielä jaoteltu tiettyyn järjestykseen merkin yleisyyden mukaan.

ASCII-siirtoa käytettäessä on olemassa vaara, että vastaanotin kadottaa synkronointinsa, eikä se tiedä, onko se vastaanottamassa edellisen merkin loppua, vai ollaanko kenties menossa jo seuraavan merkin puolivälissä. Tämä on otettu huomioon Varicoden suunnittelussa siten, ettei Varicodena lähetettävä merkki sisällä normaalitilanteessa kahta peräkkäistä nolla-bittiä, eikä ala tai pääty nollalla. Kaksi peräkkäistä nolla-bittiä voi esiintyä ainoastaan merkkien välissä tai silloin, kun lähetetään tyhjää.

Varicoden käyttö tiedonkoodaukseen kasvattaa keskimääräistä tiedonsiirtonopeutta nostaen sen noin 50 sanaan minuutissa käytettäessä siirtonopeutta 31,25 bittiä sekunnissa[1]. Varicode ASCII-koodiston mukaisessa järjestyksessä on esitetty liitteessä A.

3 TAVOITTEET

Työn tavoitteena oli rakentaa PC-tietokoneen sarjaporttiin ja radiolähetin-vastaanottimeen liitettävä ulkoinen modeemi, joka mahdollistaa radio-amatööreille tyypillisen kahden tietokoneen välisen tekstimuotoisen keskustelun radioaaltojen välityksellä.

Tiedonsiirto PC-tietokoneen ja modeemin välillä tapahtuu sarjaportin kautta ASCII-muotoisena tiedonsiirtona nopeudella 1200 bittiä sekunnissa. Tiedonsiirto modeemin ja radiolähettimen välillä tapahtuu PSK31-spesifikaation mukaisesti siniaallolla 180° vaihesiirroin nopeudella 31,25 bittiä sekunnissa. Varsinainen kantoaallon modulointi ja demodulointi tapahtuu radiolähetin-vastaanottimessa.

4 TYÖN SUORITUS

Työn suoritus jakaantui laitteiston suunnitteluun ja rakentamiseen sekä ohjelmiston suunnitteluun ja testaukseen.

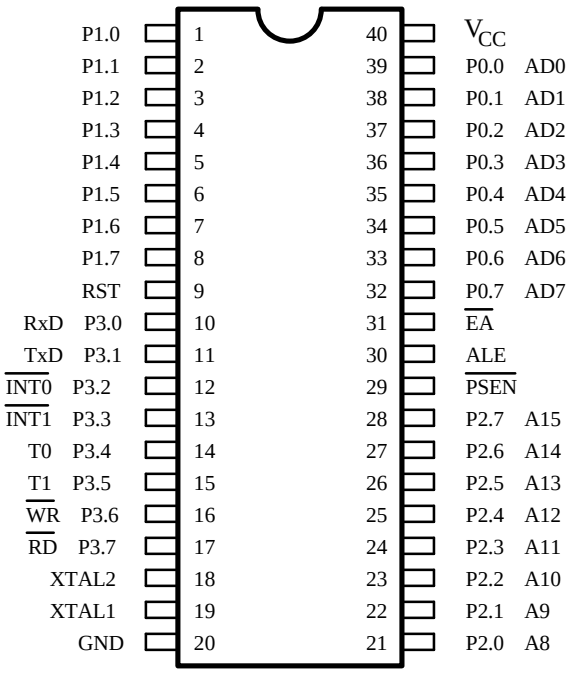
4.1 Laitteisto

Laitteisto pohjautuu Kajaanin ammattikorkeakoulun prosessorikorttiin (AMKTEL1 1.12.1997), joka rakentuu Intelin MCS-51-piiriperheen ympärille. Kortti sisältää valmiit johdotukset prosessorille, käyttöjännitteille ja usein tarvittaville oheispiireille.

Kortille tehdyt muutokset ja lisäykset on tehty osin juottamalla, osin kiertoliitoksin. Työn tekemiseen käytetyt komponentit ja piirit löytyivät Kajaanin ammattikorkeakoulun laboratoriosta. Kytkentä ja siihen käytetyt komponentit on esitetty kokonaisuudessaan liitteissä B ja C.

MCS (MicroController System)-51-piiriperhe on alun perin Intelin suunnittelema mikrokontrolleriperhe, johon kuuluu kaikkiaan useita kymmeniä eri piirejä. Piirit poikkeavat toisistaan piirille integroidun muistin määrän ja tyypin sekä piirille integroitujen I/O-ominaisuuksien osalta.

Perheen peruspiirin 8051:n perusominaisuuksia ovat Rantalan [2] mukaan mm. 8-bittinen keskusyksikkö, 64 kilotavua ohjelmamuistia ja datamuistia, 4 kilotavua sisäistä ROM-muistia ja 128 tavua sisäistä RAM-muistia. 32 kaksisuuntaista ja yksittäin osoitettavaa I/O-linjaa, jotka on ryhmitelty neljäksi 8 bitin portiksi P0, P1, P2 ja P3. Kaksi 16-bittistä ajastinta/laskuria ja täysin kaksisuuntainen full duplex -sarjaportti. Kuvassa 1 on esitetty 40-nastaisen DIL-kotelon kantakytkentä.



Kuva 1. 40-nastaisen DIL-kotelon kantakytkentä [2]

Kuten kuvasta 1 voidaan huomata, on monella linjalla kaksi vaihtoehtoista toimintaa. Esimerkiksi Portti P0 toimii ulkoisia väyliä käytettäessä multipleksoituna data/osoite-väylänä, jolla kulkee osoiteväylän 8 alinta bittiä sekä 8-bittinen data, portin P2 antaessa osoiteväylän 8 ylintä bittiä.

Kajaanin ammattikorkeakoulun prosessorikortti on alun perin suunniteltu käytettäväksi ulkoista ohjelmamuistia käyttävien prosessorien kanssa, joten kortti sisältää valmiit johdotukset myös lukko- ja EPROM-piireille.

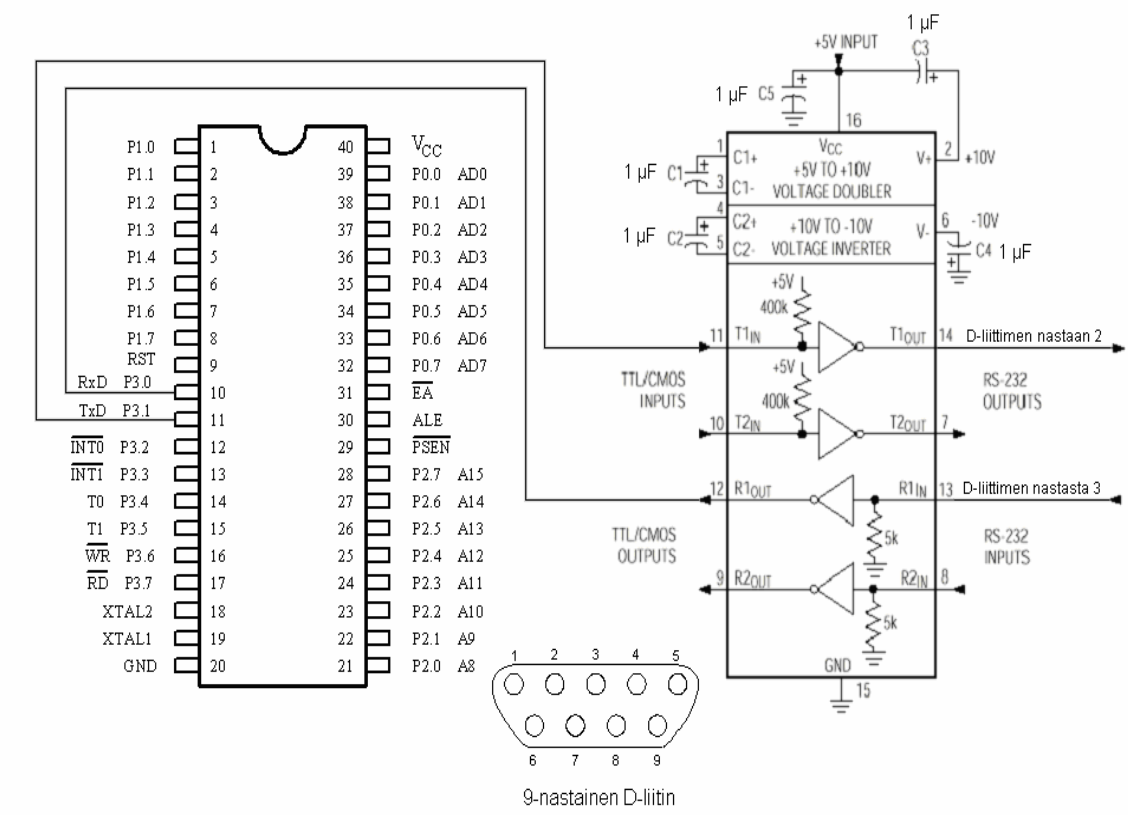
Työssä käytettiin Atmelin valmistamaa MCS-51-yhteensopivaa AT89LV51-prosessoria, joka vastaa ominaisuuksiltaan perheen peruspiiriä 8051:tä, mutta on lisäksi varustettu sisäisellä, uudelleenohjelmoitavalla flash-muistilla.

Käyttämällä prosessorin sisäistä ohjelmamuistia selvitään vähemmillä oheiskomponenteilla, koska lukko- ja EPROM-piirejä ei tarvita ollenkaan. Prosessorin sisäisen ohjelmamuistin käyttö sallittiin kytkemällä prosessorin EA (External Acces) käyttöjännitteeseen. Kortin kytkentä muutoksineen on esitetty liitteessä C.

4.2 Sarjaportin puskurointi

AT89LV51-prosessorissa on sisäänrakennettu full duplex -sarjaliikenne-ominaisuus porttien P3.0 ja P3.1 kautta. Full duplex tarkoittaa sitä, että prosessori voi lähettää ja vastaanottaa dataa yhtä aikaa. Vastaanotin on puskuroitu niin, että se voi vastaanottaa jo seuraavaa merkkiä, vaikka edellistä ei ole vielä luettu vastaanottorekisteristä.

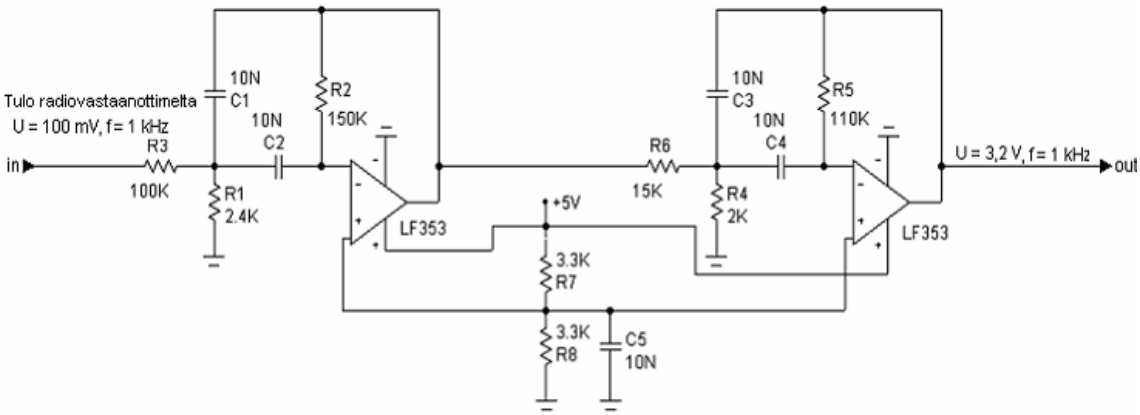
AT89LV51:n sarjaportti on EIA-232-standardin mukainen lukuun ottamatta EIA-232:n vaatimia sarjaliikennöintijännitteitä. Tämän vuoksi AT89LV51:n ja PC:n sarjaportin väliin joudutaan kytkemään puskuripiiri muuntamaan signaalit TTL-tasoisesta EIA-232-standardin mukaiseksi ja päinvastoin. Puskuripiirinä käytettiin MAX232-piiriä, jonka toiminta ja kytkentä on esitetty lähemmin kuvassa 2.



Kuva 2. AT89LV51:n liitäntä puskuripiiriin MAX232 [3]

4.3 Tulevan signaalin käsittely

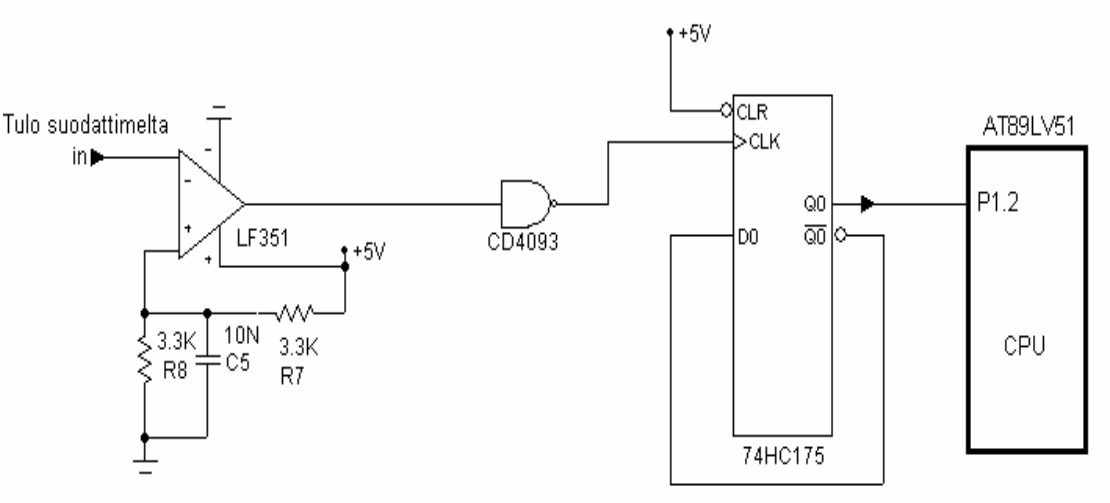
AT89LV51 on digitaalinen laite, jonka porteista lähtevä ja portteihin tuleva signaali tulee olla TTL-tasoista kanttiaaltoa. Jotta radiolähettimeltä tuleva PSK31-signaali saadaan prosessorin ymmärtämään muotoon, pitää se muuttaa TTL-tasoiseksi. PSK31-signaali on perusmuodossaan 1 kHz taajuista sinimuotoista signaalia, jonka seassa on usein radiotieltä mukaan kertynyttä häiriötä. Häiriöiden poistamiseksi ja siniaallon vakavoittamiseksi rakennettiin kuvan 3 mukainen kaistanpäästösuodatin.



Kuva 3. Kaistanpäästösuodatin

Kaistanpäästösuodattimen vahvistukseksi valittiin pienen sisäänmenojännitteen vuoksi 15 dB. Suodattimen keskitaajuudeksi valittiin luonnollisesti 1 kHz, jolloin rajataajuuksiksi kuvan 4 komponenttiarvoilla muodostui 850 ja 1150 Hz.

Suodatettu siniaalto vahvistettiin kanttipulssiksi takaisinkytkemättömällä operaatiovahvistimella. Vahvistettu, jännitteeltään ja muodoltaan lähes TTL-tasoinen signaali stabiloitiin Schmitt Trigger -liipaisukomponentilla, ja taajuus puolitettiin kahdella jakajaksi kytketyllä D-kiikulla. Käytetty kytkentä on esitetty kuvassa 4.



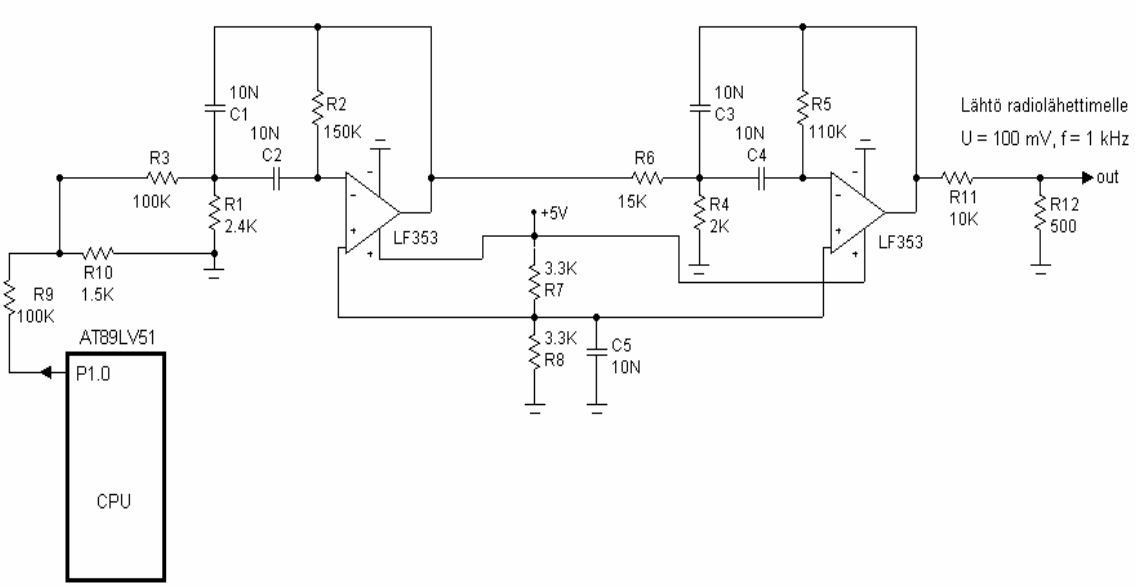
Kuva 4. Signaalin kulku suodattimelta prosessorille

Kytkennässä käytetty keinomaapiste ja +5 V käyttöjännite mahdollistivat signaalin vahvistamisen ja leikkaamisen lähes TTL-tasoiseksi pelkällä operaatiovahvistimella. Taajuuden puolittamiseen käytetty D-kiikku ei kuitenkaan tätä pelkällä operaatiovahvistimella tuotettua TTL-signaalia hyväksynyt, joten väliin laitettiin vielä Schmitt Trigger -liipaisukomponentti, joka ”pakottaa” signaalin aidoksi TTL-signaaliksi.

Taajuuden puolittaminen ei olisi ollut välttämätöntä, mikäli käytössä olisi ollut tehokkaampi prosessori. Tässä tapauksessa taajuuden puolittaminen oli kuitenkin välttämätöntä takaamaan prosessorille riittävästi laskenta-aikaa.

4.4 Lähtevän signaalin käsittely

Kuten radiovastaanottimen lähtö, radiolähettimen tulokin toimii 100 mV_{p-p} jännitteellä. Tämän vuoksi AT89LV51:n portista P1.0 lähtevä TTL-tasoinen kanttipulssi pitää muuntaa 100 mV_{p-p} siniaalloksi. Muunnokseen käytettiin kuvassa 3 esitettyä kaistanpäästösuodatinta lisäämällä jännitejaot ennen ja jälkeen suodattimen. Ajamalla signaali kaistanpäästösuodattimen läpi signaali muuttuu kanttiaallosta siniaalloksi. Lisäksi kaistanpäästösuodatus poistaa kanttiaallosta aiheutuvat turhat harmoniset kerrannaiset. Kuvan 3 suodatin lisäyksineen on esitetty kuvassa 5.



Kuva 5. Signaalin kulku prosessorilta radiolähettimelle

4.5 Ohjelman suunnittelu

Ohjelma rakentuu pääohjelmasta ja useasta aliohjelmasta. Itse pääohjelma käsittää vain sarjaportin ja laskurien alustukset sekä vastaanotin- ja lähetinaliohjelman valintasilmukan. Ohjelman kulku vuokaaviotasolla on esitetty liitteessä D. Varsinainen ohjelma on esitetty liitteessä E.

4.5.1 Sarjaportin toimintamuodon ja nopeuden valinta

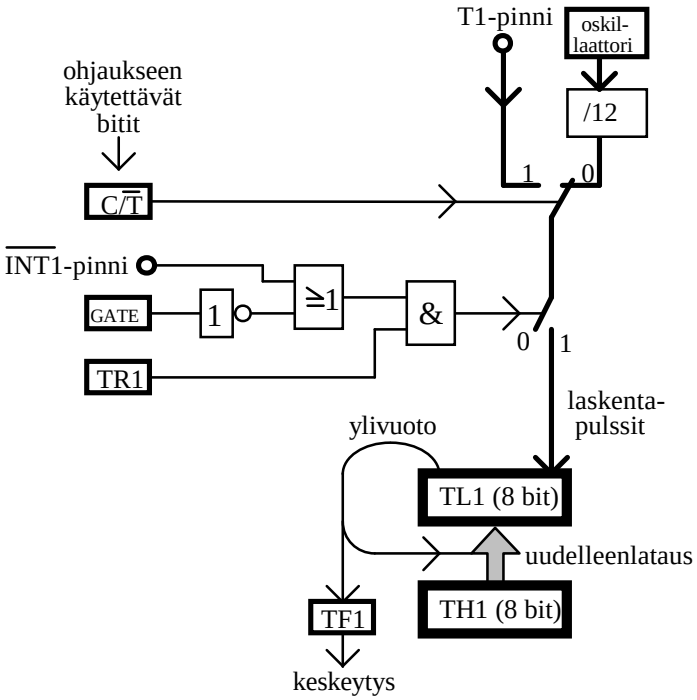
Sarjaportin toimintamuodoksi valittiin toimintamuoto 1, koska tässä toiminta-muodossa sarjaportin liikennöintinopeus voidaan asettaa AT89LV51:n sisäisen laskurin avulla. Toimintamuodossa 1 sarjaportti toimii 10-bittisessä tilassa. Tässä toimintamuodossa sarjamuotoinen tieto lähetetään TxD-pinnin kautta ja vastaanotetaan RxD-pinnin kautta. Tiedon muoto on:

- yksi START-bitti (0)
- 8 databittiä (LSB-bitti ensin)
- yksi STOP-bitti (1).

Vastaanotossa STOP-bitti menee SCON-rekisterin RB8-bitin arvoksi. Sarja-muotoisen tiedon bittikehys toimintamuodossa 1 on esitetty kuvassa 6.

Kuva 6. Sarjamuotoisen tiedon bittikehys toimintamuodossa 1 [2]

Sarjaportin liikennöintinopeus määritettiin halutuksi käyttämällä baudi-generaattorina laskuri 1:tä. Laskuri 1 asetettiin toimimaan automaatti-latauksessa toimintamuodossa 2. Toimintamuoto 2 on 8-bittinen laskentamuoto, jossa alempi laskurirekisteri TL1 toimii 8-bittisenä laskurina, johon uusi arvo latautuu automaattisesti laskurirekisteristä TH1. TL1:n ylivuoto aiheuttaa sekä keskeytyspyynnön että uudelleenlatautumisen. Uudelleenlataus jättää TH1:n arvon muuttumattomaksi. Laskurin 1 toiminta toimintamuodossa 2 on esitetty kuvassa 7.



Kuva 7. Laskurin 1 toiminta automaattilatauksessa toimintamuodossa 2 [2]

Kun tiedettiin, että modeemin tiedonsiirtonopeus on vain 31,25 bittiä sekunnissa ja käytössä on 12 MHz kide, oli yksinkertaisinta asettaa sarjaportti toimimaan nopeudella 1200 bit/s. Nopeuden asetus tehdään ratkaisemalla laskurirekisteriin TH1 asetettava arvo kaavalla 2.

$$1200 = \frac{1}{32} \cdot \frac{12\,\text{MHz}}{12 \cdot (256 - \text{TH1})} \qquad (2)$$

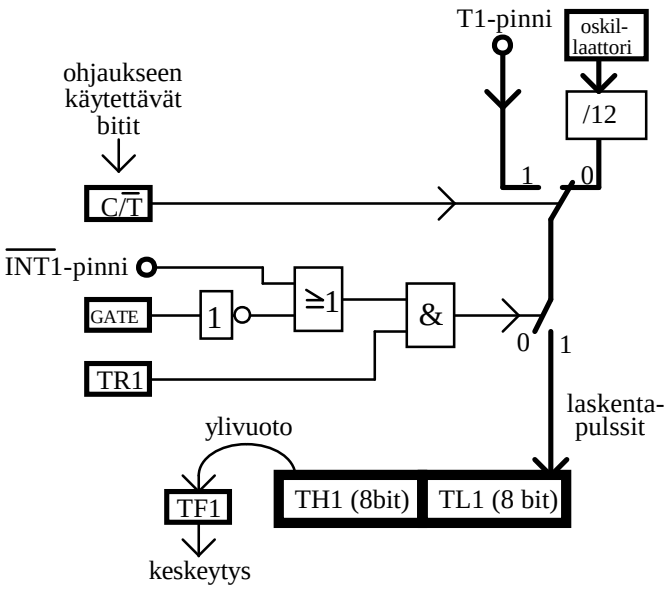
$$\Rightarrow \text{TH1} = \frac{\left(\frac{12\,\text{MHz}}{1200} - 32 \cdot 12 \cdot 256\right)}{-(32 \cdot 12)} = \text{E6H}$$

Laskurirekisteriin ladattiin kaavan 2 mukaisesti E6 Hex.

4.5.2 Viiveen muodostus

Lähetykseen käytettävä 1 kHz pulssi muodostettiin porttiin P1.0 asettamalla portti 500 µs välein loogiseen ”1”- ja ”0”- tilaan. Pulssin muodostamiseen tarvittava viive muodostettiin ajastin/laskuri 0:lla ja sitä kontrolloivalla viive-aliohjelmalla. Ajastin 0 asetettiin toimintamuotoon 1 asettamalla TMOD-rekisterin arvoksi TMOD=0x01.

Toimintamuoto 1 on 16-bittinen, ohjelmallisesti ohjattavissa oleva laskentatila, jossa ajastin laskee koko ajan ylöspäin päivittäen rekisterien TLO ja TH0 sisältöä. Ajastimen 0 toiminta toimintamuodossa 1 on esitetty kuvassa 8.



Kuva 8. Ajastimen 0 toiminta toimintamuodossa 1 [2]

Toimintamuodossa 1 ajastimen toimintaa voidaan ohjata ohjelmallisesti lataamalla ajastimen TL0 ja TH0 rekistereihin sopivat alkuarvot. Koska AT89LV51:n konejakso koostuu 12 kellojaksosta, on laskentataajuus 12 MHz kiteellä 1 MHz (T = 1 µs). Suurin 16-bittisellä ajastimella saavutettava viive on näin ollen $2^{16} * 1\mu s$.

Sopivat alkuarvot 500 µs viiveelle saatiin laskettua kaavalla 3.

$$2^{16} * 1\mu s - x * 1\mu s = 500\mu s \rightarrow x = 65036 = FE23\,Hex \qquad (3)$$

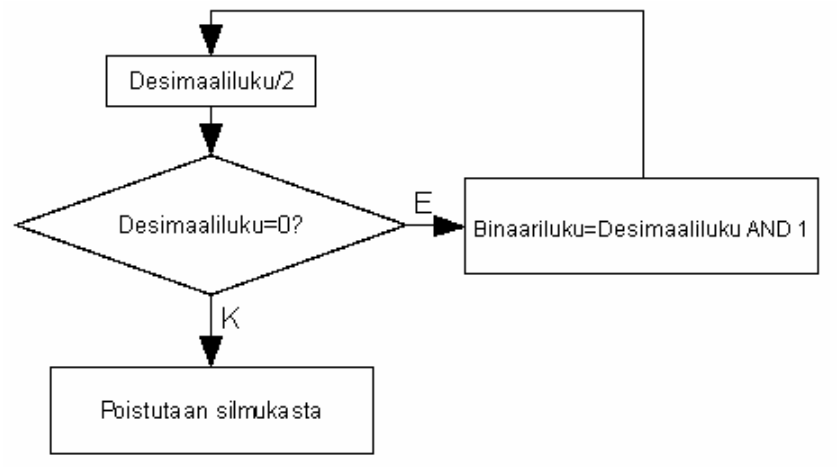
Rekistereihin ladattiin kaavan 3 mukaisesti TL0 = 23 ja TH0 = FE.

4.5.3 Lähetin

Koska PSK31 on jatkuva-aaltoinen lähetysmoodi, joutuu prosessori lähettämään kanttipulssia ”ikuisesti” porttiin P1.0, vaikka varsinaisia merkkejä ei lähetettäisikään. Tämä asettaa omat vaatimuksensa sarjaportin RI-lipun tilaa tutkivalle lähetysaliohjelmalle. Lähetysaliohjelmaa ei voida toteuttaa yksinkertaisella pollausmenetelmällä, vaan sarjaportti on asetettava toimimaan keskeytysmenetelmällä. Keskeytysmenetelmä mahdollistaa merkin lukemisen paikalliseen merkkivälimuistiin keskeytyksen tullessa, vaikka muun ohjelman suoritus on vielä kesken.

PSK31:ssä käytetyt varicode-merkit on talletettu ohjelman suorituksen aikana ROM-muistista haettavaan varicode-taulukkoon desimaalilukuna. Merkit sijaitsevat taulukossa ASCII-koodiston mukaisessa järjestyksessä indeksoinnin yksinkertaistamiseksi. Tämä tarkoittaa käytännössä sitä, että kun sarjaportin kautta vastaanotetaan esimerkiksi ASCII-merkki 20, löytyy vastaava merkki varicode-taulukon muistipaikasta 20.

Desimaalilukuna tallennetut merkit joudutaan muuttamaan binaariseen muotoon ennen lähetystä. Desimaali-binaari-muunnos on toteutettu while-silmukalla AND-operaation avulla. Silmukan toiminta on esitetty kuvassa 9.



Kuva 9. Desimaali-binaari-muunnos

Silmukka palauttaa käsiteltävän desimaaliluvun eniten merkitsevän bitin binaarilukuna. Bitillä ohjataan pulssi-aliohjelman toimintaa. Pulssi-aliohjelma tuottaa 1 kHz kanttipulssia 32 ms pätkissä ikuisessa silmukassa. Mikäli muunnossilmukan palauttama eniten merkitsevä bitti on ”1”, jatkuu pulssi-aliohjelman suoritus ennallaan. Mikäli muunnos silmukan palauttama bitti on ”0”, kääntyy pulssi-aliohjelman tuottaman 1 kHz kanttipulssin vaihe 180•. Kun merkki on lähetetty kokonaan ja poistutaan silmukasta, lähetetään pulssi-aliohjelmalla kaksi peräkkäistä vaiheenkääntöä, joka ilmoittaa PSK31-vastaanottimelle merkin lähetyksen päättyneen.

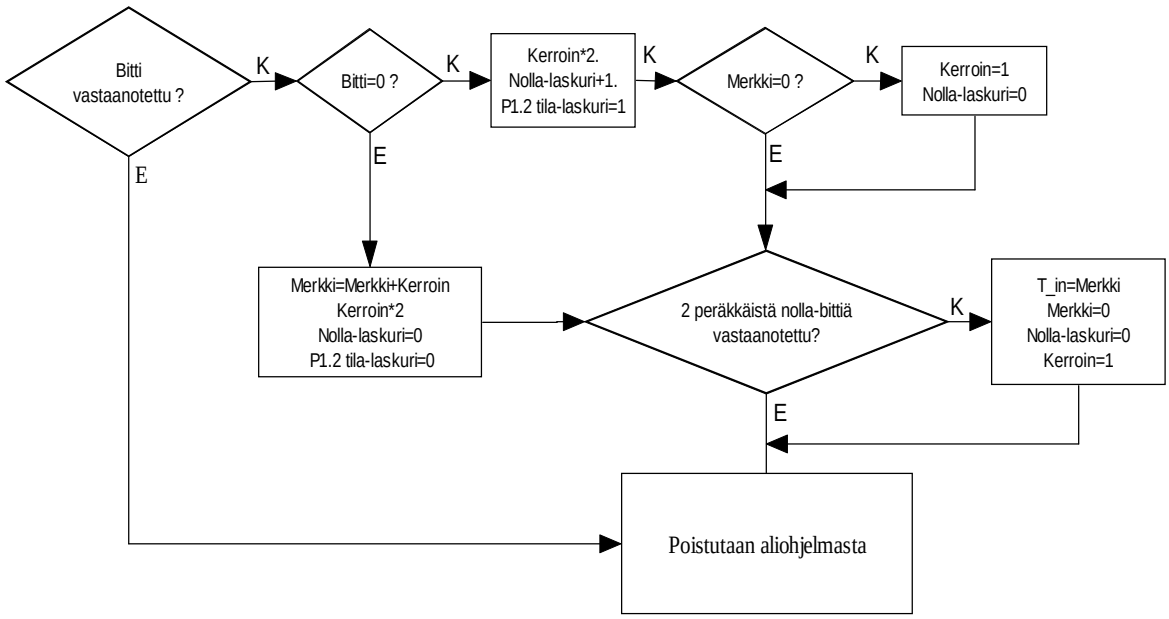
Silmukkaa pyöritetään niin kauan, kuin paikallisessa merkkivälimuistissa on lähetettäviä merkkejä. Mikäli paikallinen merkkivälimuisti on tyhjä, lähetetään peräkkäisiä 180• vaiheenkääntöpulsseja ”ikuisesti”, kunnes merkkivälimuistiin luetaan uusi lähetettävä merkki.

4.5.4 Vastaanotin

Myös vastaanotin käyttää hyväkseen varicode-taulukkoa. Siinä missä lähettimessä varicode-taulukon indeksi vastaanotetaan suoraan sarjaportin kautta, joudutaan vastaanottimessa indeksi etsimään vertaamalla radiovastaanottimelta vastaanotettua merkkiä varicode-taulukon sisältöön.

Merkin vastaanotto ja tunnistus on toteutettu laskemalla ohjelmallisesti porttiin P1.2 tulevien kanttipulssien ykkös- ja nollatasojen lukumäärää ja kestoa. Mikäli pulssien kestolla ei olisi merkitystä laskennan kannalta, voitaisiin käyttää rautapohjaista, laskurilla toteutettua laskentaa. Tässä tapauksessa se ei kuitenkaan ole mahdollista, koska merkin tunnistamiseksi tarvitaan tieto sekä pulssien lukumäärästä että kestosta.

Bitin tunnistus on toteutettu mittaamalla, kuinka kauan portti P1.2 on tilassa ”1” tai ”0”. Jokainen portin tilan muutos kasvattaa ohjelmallisen laskurin arvoa yhdellä. Mikäli portti on jommassakummassa tilassa yli 1 ms ajan, asetetaan apubitti nollaksi. Kun laskuri saavuttaa arvon 32, siirrytään bitin vastaanottoaliohjelmaan. Vastaanottoaliohjelman toiminta on esitetty kuvassa 10.



Kuva 10. Vastaanottoaliohjelman toiminta

Vastaanottoaliohjelma ajetaan aina, kun ohjelmallinen laskuri saavuttaa arvon 32. Mikäli vastaanotettava bitti on nolla vastaanottoaliohjelman kahtena peräkkäisenä ajokertana, tiedetään merkin vastaanoton päättyneen.

Vastaanotettu merkki siirretään välimuistiin, josta merkkiä verrataan varicode-taulukon sisältämiin merkkeihin. Kun vastaavuus löytyy, lähetetään vastaavan merkin indeksi sarjaportin kautta PC:lle, tyhjennetään välimuisti ja odotellaan uuden merkin siirtymistä välimuistiin.

5 TESTAUS

Laitteistoa ja ohjelmistoa testattiin Kajaanin ammattikorkeakoulun laboratoriossa kahdella PC-tietokoneella. Laitetta käytettiin kahden tietokoneen välillä muuntamaan toiselta tietokoneelta sarjaportin kautta tuleva ASCII-muotoinen sarjaliikenne toisella tietokoneella pyörineen PSK31-vastaanotinohjelman ymmärtämään muotoon ja päinvastoin. Laite kytkettiin PSK31-lähetin-vastaanotin -ohjelmaa pyörittäneen koneen äänikortin line-in- ja line-out-liitäntöihin. Laitteen toimintaa testattiin molempiin suuntiin käymällä normaalia ”näppäimistöltä toiselle” -keskustelua koneiden välillä. Laitteen toiminnassa ei havaittu puutteita tulosignaalin ollessa suunnitellun 100 $mV_{p\text{-}p}$ tuntumassa. Jännitteen laskiessa tai noustessa vastaanottimen virheherkkyys kasvoi.

6 YHTEENVETO

Työssä rakennettu laitteisto oli komponenteiltaan ja kytkennöiltään varsin yksinkertainen. Haastavinta työssä oli suunnitella ja optimoida ohjelmakoodi niin, että prosessori suoriutuu kaikista tarvittavista laskutoimituksista käytettävissä olevan laskenta-ajan puitteissa. Suurimmat ongelmat työn kannalta aiheutuivat juuri työssä käytetystä varicode-taulukosta ja siihen talletetuista 16-bittisistä luvuista.

Itse prosessori kykenee rautatason laskentaan vain 8-bittisillä luvuilla, joten laskenta jouduttiin suorittamaan huomattavasti enemmän laskenta-aikaa vievänä ohjelmallisena laskentana. Tämä aiheutti ongelmia sekä vastaanotto- että lähetysaliohjelman suunnitteluun, koska molemmissa tapauksissa joudutaan lähetys ja vastaanottoporttien ohjaamisen lisäksi huolehtimaan vielä sarjaliikenteestä ja laskenta-aikaa syövästä 16-bittisestä ohjelmallisesta laskennasta.

Laskenta saatiin kuitenkin suorittua käytettävissä olevan ajan puitteissa jakamalla eniten laskenta-aikaa vievät toimet pienempiin kokonaisuuksiin ja ripottelemalla ne ohjelmakoodin sekaan suoritettavaksi prosessorin ”jouto-aikana”.

LÄHTEET

1 PSK31 ”virallinen” kotisivu. Luettu 12.2.2005. [WWW-dokumentti] http://www.aintel.bi.ehu.es/psk31.html

2 Rantala, P. Mikrotietokonetekniikka. Tietokotka. Liite 2: Intel MCS-51 -perhe.

3 MAX232-piirin datalehti ja kytkentä AT89LV51:een. Luettu 20.3.2004. [WWW-dokumentti] http://www.rentron.com/project02.htm

LIITTEET

Merkki	Desimaalilukuna	ASCII Binaarilukuna	Varicode
NUL	0	00000000	1010101011
SOH	1	00000001	1011011011
STX	2	00000010	1011101101
ETX	3	00000011	1101110111
EOT	4	00000100	1011101011
ENQ	5	00000101	1101011111
ACK	6	00000110	1011101111
BEL	7	00000111	1011111101
BS	8	00001000	1011111111
TAB	9	00001001	11101111
LF	10	00001010	11101
VT	11	00001011	1101101111
FF	12	00001100	1011011101
CR	13	00001101	11111
SO	14	00001110	1101110101
SI	15	00001111	1110101011
DLE	16	00010000	1011110111
DC1	17	00010001	1011110101
DC2	18	00010010	1110101101
DC3	19	00010011	1110101111
DC4	20	00010100	1101011011
NAK	21	00010101	1101101011
SYN	22	00010110	1101101101
ETB	23	00010111	1101010111
CAN	24	00011000	1101111011
EM	25	00011001	1101111101
SUB	26	00011010	1110110111
ESC	27	00011011	1101010101
FS	28	00011100	1101011101
GS	29	00011101	1110111011
RS	30	00011110	1011111011
US	31	00011111	1101111111
SP	32	00100000	1
!	33	00100001	111111111
"	34	00100010	101011111
#	35	00100011	111110101
$	36	00100100	111011011
%	37	00100101	1011010101
&	38	00100110	1010111011

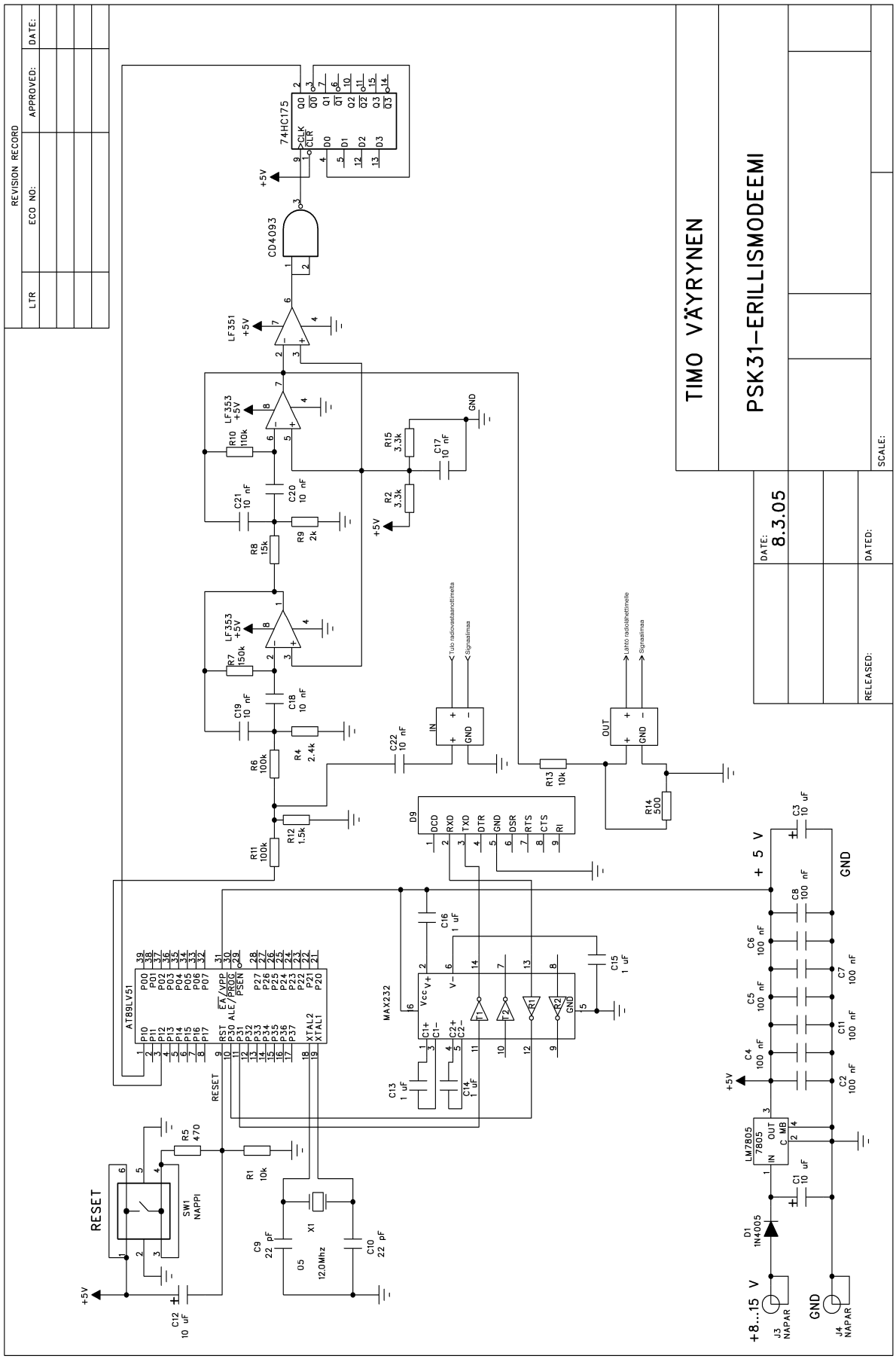
‘	39	00100111	101111111
(	40	00101000	11111011
)	41	00101001	11110111
*	42	00101010	101101111
+	43	00101011	111011111
,	44	00101100	1110101
-	45	00101101	110101
.	46	00101110	1010111
/	47	00101111	110101111
0	48	00110000	10110111
1	49	00110001	10111101
2	50	00110010	11101101
3	51	00110011	11111111
4	52	00110100	101110111
5	53	00110101	101011011
6	54	00110110	101101011
7	55	00110111	110101101
8	56	00111000	110101011
9	57	00111001	110110111
:	58	00111010	11110101
;	59	00111011	110111101
<	60	00111100	111101101
=	61	00111101	1010101
>	62	00111110	111010111
?	63	00111111	1010101111
@	64	01000000	1010111101
A	65	01000001	1111101
B	66	01000010	11101011
C	67	01000011	10101101
D	68	01000100	10110101
E	69	01000101	1110111
F	70	01000110	11011011
G	71	01000111	11111101
H	72	01001000	101010101
I	73	01001001	1111111
J	74	01001010	111111101
K	75	01001011	101111101
L	76	01001100	11010111
M	77	01001101	10111011
N	78	01001110	11011101
O	79	01001111	10101011

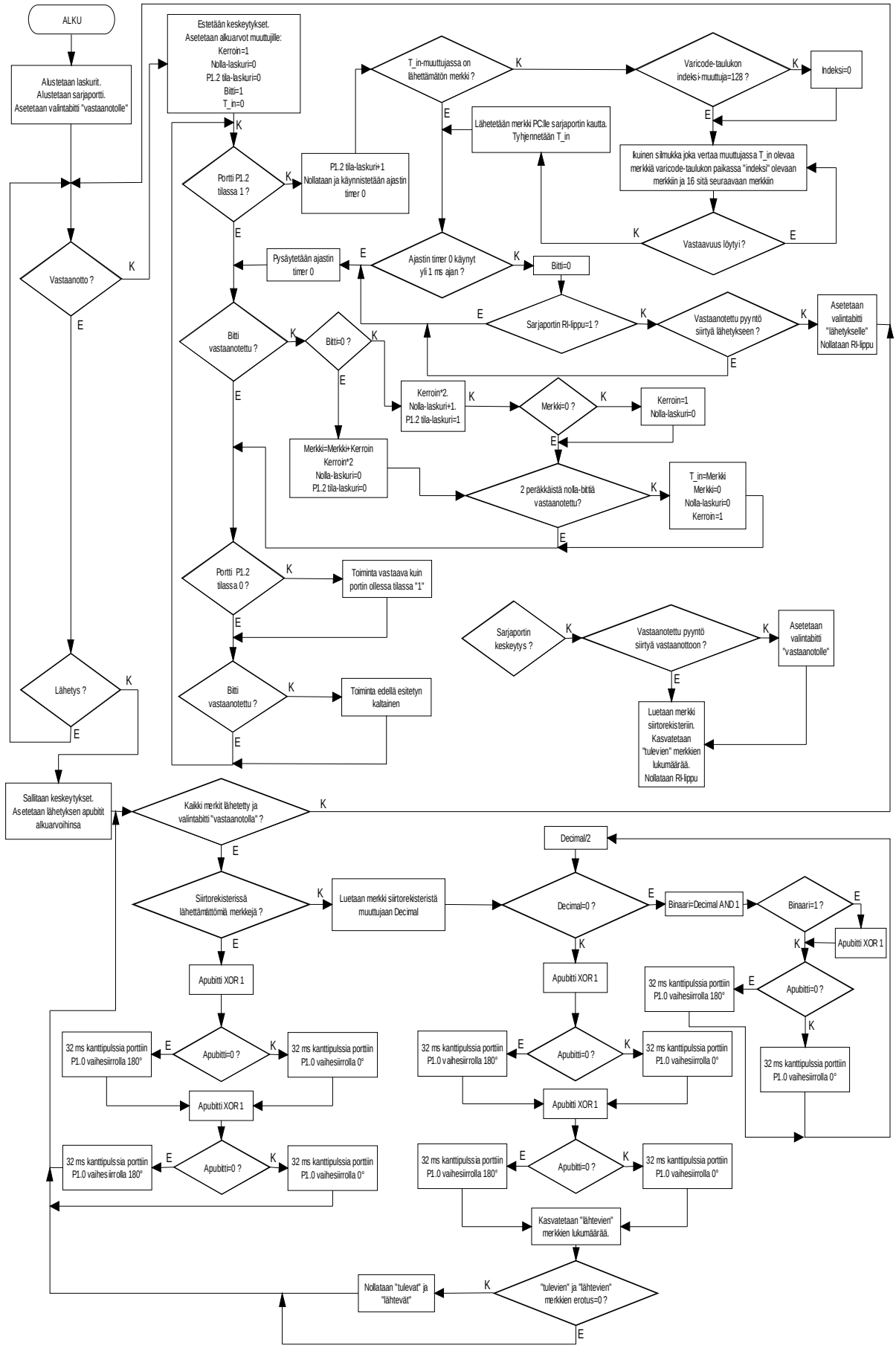
P	80	01010000	11010101
Q	81	01010001	111011101
R	82	01010010	10101111
S	83	01010011	1101111
T	84	01010100	1101101
U	85	01010101	101010111
V	86	01010110	110110101
W	87	01010111	101011101
X	88	01011000	101011101
Y	89	01011001	101110101
Z	90	01011010	101111011
[	91	01011011	1010101101
\	92	01011100	111110111
]	93	01011101	111101111
^	94	01011110	111111011
_	95	01011111	1010111111
`	96	01100000	101101101
a	97	01100001	1011
b	98	01100010	1011111
c	99	01100011	101111
d	100	01100100	101101
e	101	01100101	11
f	102	01100110	111101
g	103	01100111	1011011
h	104	01101000	101011
i	105	01101001	1101
j	106	01101010	111101011
k	107	01101011	10111111
l	108	01101100	11011
m	109	01101101	111011
n	110	01101110	1111
o	111	01101111	111
p	112	01110000	111111
q	113	01110001	110111111
r	114	01110010	10101
s	115	01110011	10111
t	116	01110100	101
u	117	01110101	110111
v	118	01110110	1111011
w	119	01110111	1101011
x	120	01111000	11011111

y	121	01111001	1011101
z	122	01111010	111010101
{	123	01111011	1010110111
|	124	01111100	110111011
}	125	01111101	1010110101
~	126	01111110	1011010111
DEL	127	01111111	1110110101

OSALUETTELO

Vastus R1, R3, R13	10 k•
Vastus R2, R15	3,3 k•
Vastus R4	2,4 k•
Vastus R5	470 •
Vastus R6, R11	100 k•
Vastus R7	150 k•
Vastus R8	15 k•
Vastus R9	2 k•
Vastus R10	110 k•
Vastus R12	1,5 k•
Vastus R14	500 •
Kondensaattori C1, C3	10 • F
Kondensaattori C2, C4, C5, C6, C7, C8, C11	100 nF
Kondensaattori C9, C10, C12	22 pF
Kondensaattori C13, C14, C15, C16	1 • F
Kondensaattori C17, C18, C19, C20, C21, C22	10 nF
Diodi D1	1N4005
Prosessori	AT89LV51
Kide	12,0 MHz
Jänniteregulaattori	LM7805
Sarjaliikenneohjain	MAX232
Painonappi	SW1
Operaatiovahvistin	LF351
Operaatiovahvistin	LF353
NAND Schmitt Trigger	CD4093B
D-kiikku	74HC175
D-liitin, uros	D9
Prosessorikortti	AMKTEL1 1.12.1997
Ruuviliitin	IN
Ruuviliitin	OUT


REVISION RECORD
LTR
ECO NO:
APPROVED:
DATE:
TIMO VÄYRYNEN
PSK31-ERILLISMODEEMI
DATE: 8.3.05
DATED:
RELEASED:
SCALE:
74HC175
CD4093
LF351
LF353
AT89LV51
MAX232
LM7805
7805
1N4005
RESET
SW1 NAPPI
12.0Mhz
Tulo radiovastaanottimelta
Signaalimaa
Lähtö radiolähettimelle
Signaalimaa
D9
DCD
RXD
TXD
DTR
GND
DSR
RTS
CTS
RI
+ 5 V
GND
+8...15 V
J3 NAPAR
J4 NAPAR


ALKU
Alustetaan laskurit.
Alustetaan sarjaportti.
Asetetaan valintabitti "vastaanotolle"
Estetään keskeytykset.
Asetetaan alkuarvot muuttujille:
Kerroin=1
Nolla-laskuri=0
P1.2 tila-laskuri=0
Bitti=1
T_in=0
Vastaanotto ?
Portti P1.2 tilassa 1 ?
P1.2 tila-laskuri+1
Nollataan ja käynnistetään ajastin timer 0
T_in-muuttujassa on lähettämätön merkki ?
Varicode-taulukon indeksi-muuttuja=128 ?
Indeksi=0
Lähetetään merkki PC:lle sarjaportin kautta.
Tyhjennetään T_in
Ikuinen silmukka jolla vertaa muuttujassa T_in olevaa merkkiä varicode-taulukon paikassa "indeksi" olevaan merkkiin ja 16 sitä seuraavaan merkkiin
Vastaavuus löytyi ?
Ajastin timer 0 käynyt yli 1 ms ajan ?
Pysäytetään ajastin timer 0
Bitti=0
Sarjaportin RI-lippu=1 ?
Vastaanotettu pyyntö siirtyä lähetykseen ?
Asetetaan valintabitti "lähetykselle"
Nollataan RI-lippu
Bitti vastaanotettu ?
Bitti=0 ?
Kerroin*2.
Nolla-laskuri+1.
P1.2 tila-laskuri=1
Merkki=0 ?
Kerroin=1
Nolla-laskuri=0
Merkki=Merkki+Kerroin
Kerroin*2
Nolla-laskuri=0
P1.2 tila-laskuri=0
2 peräkkäistä nolla-bittiä vastaanotettu?
T_in=Merkki
Merkki=0
Nolla-laskuri=0
Kerroin=1
Portti P1.2 tilassa 0 ?
Toiminta vastaava kuin portin ollessa tilassa "1"
Bitti vastaanotettu ?
Toiminta edellä esitetyn kaltainen
Sarjaportin keskeytys ?
Vastaanotettu pyyntö siirtyä vastaanottoon ?
Asetetaan valintabitti "vastaanotolle"
Luetaan merkki siirtorekisteriin.
Kasvatetaan "tulevien" merkkien lukumäärää.
Nollataan RI-lippu
Lähetys ?
Sallitaan keskeytykset.
Asetetaan lähetyksen apubitit alkuarvoihinsa
Kaikki merkit lähetetty ja valintabitti "vastaanotolla" ?
Siirtorekisterissä lähettämättömiä merkkejä ?
Luetaan merkki siirtorekisteristä muuttujaan Decimal
Decimal/2
Decimal=0 ?
Binaari=Decimal AND 1
Binaari=1 ?
Apubitti XOR 1
Apubitti=0 ?
32 ms kanttipulssia porttiin P1.0 vaihesiirrolla 180°
32 ms kanttipulssia porttiin P1.0 vaihesiirrolla 0°
Kasvatetaan "lähtevien" merkkien lukumäärää.
"tulevien" ja "lähtevien" merkkien erotus=0 ?
Nollataan "tulevat" ja "lähtevät"

```
/* PSK31 Erillismodeemin Softa 22.2.2005
   Timo Väyrynen
   Kajaanin ammattikorkeakoulu */

#include <io51.h>
#include <stdio.h>
#include <string.h>

#define PULSSI P1.0   /* Määritetään lähetysportti */
#define IN P1.2       /* Määritetään vastaanottoportti */

/* ROM-muistiin jäävä taulukko, joka sisältää varicode-merkit ASCII-
koodiston mukaisessa järjestyksessä */

code static int VARICODE[]={0x355,0x36D,0x2DD,0x3BB,0x35D,0x3EB,
0x3DD,0x2FD,0x3FD,0xF7,0x17,0x3DB,0x2ED,0x1F,0x2BB,0x357,0x3BD,
0x2BD,0x2D7,0x3D7,0x36B,0x35B,0x2DB,0x3AB,0x37B,0x2FB,0x3B7,0x2AB,
0x2EB,0x377,0x37D,0x3FB,0x1,0x1FF,0x1F5,0x15F,0x1B7,0x2AD,0x375,
0x1FD,0xDF,0xEF,0x1ED,0x1F7,0x57,0x2B,0x75,0x1EB,0xED,0xBD,0xB7,
0xFF,0x1DD,0x1B5,0x1AD,0x16B,0x1AB,0x1DB,0xAF,0x17B,0x16F,0x55,
0x1D7,0x3D5,0x2F5,0x5F,0xD7,0xB5,0xAD,0x77,0xDB,0xBF,0x155,0x7F,
0x17F,0x17D,0xEB,0xDD,0xBB,0xD5,0xAB,0x177,0xF5,0x7B,0x5B,0x1D5,
0x15B,0x175,0x15D,0x1BD,0x2D5,0x1DF,0x1EF,0x1BF,0x3F5,0x16D,0x3ED,
0xD,0x7D,0x3D,0x2D,0x3,0x2F,0x6D,0x35,0xB,0x1AF,0xFD,0x1B,0x37,0xF,
0x7,0x3F,0x1FB,0x15,0x1D,0x5,0x3B,0x6F,0x6B,0xFB,0x5D,0x157,0x3B5,
0x1BB,0x2B5,0x3AD,0x2B7};

void laskurien_alustus();
void sarjap_alustus();
void viive();         /* 0,5 ms:n viive 1 kHz pulssia varten */
void vastaanotin();   /* Vastaanottimen ”pääohjelma” */
void vastaanotto();   /* Muunnos varicodesta ASCII:ksi */
void lahetin();       /* Lähettimen ”pääohjelma” */
void lahetys();       /* Muunnos ASCII:sta varicodeksi */
void pulssi();        /* 1 kHz PSK-signaalin muodostus */
int vast_bitti=1;     /* Vastaanoton apubitti */
int laskuri=0;        /* Laskee tulevien kanttipulssien määrää */
int muisti=0;         /* Laskee vaihesiirtojen määrää */
int vast_merkki=0;    /* Muuttuja johon merkki ”kasataan” */
int kerroin=1;        /* Merkin “kasaukseen” liittyvä kerroin */
int T_in=0;           /* Vastaanoton ”merkkivälimuisti” */
int bitti=0;          /* Lähetyksen apubitti */
int R_in=0;           /* “Tulevien” ja “lähtevien” merkkien laskurit */
int R_out=0;
int taulukko[64];     /* Lähetyksen merkkivälimuisti */

int i;
int j=0;              /* Indeksit */
int lisays=0;         /* Indeksiä kasvattava muuttuja */
int valinta=0;        /* Lähettimen/vastaanottimen valintabitti */

int binaari;          /* ASCII --> varicode muunnoksessa */
int decimal;          /* tarvittavat muuttujat */

main()
{
  EA=0;               /* Estetään kaikki keskeytykset */
  laskurien_alustus();
  sarjap_alustus();
```

```
  EX1=0;              /* Ulkoinen keskeytys EX1 estetty */
  EX0=0;              /* Ulkoinen keskeytys EX0 estetty */
  EA=1;               /* Sallitaan määritetyt keskeytykset */

  for(;;)
  {
    if (valinta==0)
    {
      vastaanotin();  /* Control+R käynnistää vastaanottimen */
    }
    if (valinta==1)
    {
      lahetin();      /* Control+S käynnistää lähettimen */
    }
  }
}
/*********************************************************************/
interrupt void SCON_int()       /* Sarjaportin keskeytysaliohjelma */
{
  if (RI==1)
  {
    if (SBUF==18)               /* Jos painettu Control+R */
    {
      valinta=0;                /* Asetetaan apubitti nollaksi */
    }
    taulukko[R_in]=VARICODE[SBUF];     /* Luetaan SBUF:n sisältö */
    R_in++;                     /* Luettu --> kasvatetaan */
    RI=0;                       /* "tulevien" merkkien lukumäärää */
  }
  if (TI==1)          /* Lähetyksen aiheuttama keskeytys. */
  {                   /* Tarvitaan ainoastaan "herättelemään */
    TI=0;             /* 8051 keskeyttämään myös vastaanotettaessa */
  }
}
/*********************************************************************/
void laskurien_alustus()
{
ET0=0;      /* Timer 0 keskeytykset estetty */
ET1=0;      /* Timer 1 keskeytykset estetty */
PCON=0x00;  /* Timer1 sarjaportin käytössä */
TMOD=0x21;  /* Timer 1 8-bit tilassa 2, Timer 0 16-bit tilassa 1 */ */
TH1=0xE6;   /* Asetetaan sarjaportin toimintanopeus 1200 bit/s */
TR1=1;      /* Käynnistetään laskuri timer1 */
}
/*********************************************************************/
void sarjap_alustus()
{
  SM0=0;     /* Sarjaportin toimintamuoto 1 */
  SM1=1;     /* Eli start + 8 databittiä + stop */
  ES=1;      /* Sallitaan sarjaportin keskeytys */
  PS=0;      /* Sarjaporttikeskeytyksillä matala prioriteetti */
  REN=1;     /* Käynnistetään sarjaportti */
  RI=0;      /* Nollataan vastaanoton keskeytyslippu */
  TI=0;      /* Nollataan lähetyksen keskeytyslippu */
  SBUF=0x00;          /* Herätellään 8051 keskeytysjärjestelmä */
}                     /* lähettämällä merkki "NUL" */
/*********************************************************************/
```

```
void vastaanotin()
{
  IN=1;               /* Asetetaan portti P1.2 tilaan 1 */
  EA=0;               /* Estetään keskeytykset */
  vast_bitti=1;
  laskuri=0;          /* Asetetaan vastaanoton */
  muisti=0;           /* apumuuttujat alkuarvoihinsa */
  vast_merkki=0;
  kerroin=1;

  for(;;)
  {
    if(IN==1)         /* Jos portti P1.2 on tilassa 1 */
    {
      laskuri++;      /* Kasvatetaan P1.2 tilan muutoksia laskevan */
      TL0=0x00;       /* muuttujan arvoa yhdellä */
      TH0=0x00;
      TR0=1;          /* Nollataan ja käynnistetään ajastin timer 0 */

      if(T_in!=0)     /* Jos T_in-muuttujassa on lähettämätön merkki */
      {               /* Etsitään merkin ASCII vastine taulukosta */
        if(lisays==128)         /* Jos edellisellä kierroksella */
        {                       /* päästy taulukon loppuun, */
          lisays=0;   /* nollataan indeksin kasvatusmuuttuja */
        }
        for(j=lisays;j<lisays+16;j++) /* Käydään läpi 16 */
        {                             /* seuraavaa merkkiä */
          if (T_in==VARICODE[j]) /* Jos vastaavuus löytyi */
          {
            SBUF=j;             /* Lähetetään merkki PC:lle */
            TI=0;               /* sarjaportin kautta. */
            T_in=0;             /* Nollataan TI ja T_in */
          }
        }
      lisays=lisays+16;         /* 16 merkkiä läpikäyty */
      }
      while(IN==1)
      if(TH0>=0x04)             /* Jos pulssin pituus on yli 1 ms */
      {
        vast_bitti=0;           /* Asetetaan vastaanoton */
        if (RI==1)              /* apubitti nollaksi */
        {
          if (SBUF==19)         /* Jos painettu Control+S */
          {
            RI=0;               /* Nollataan RI-lippu
            valinta=1;          /* Lähetin päälle */
            return;             /* Palataan pääohjelmaan */
          }
          else                  /* Jos esimerkiksi vahingossa */
          {                     /* painettu jotain muuta näppäintä */
            RI=0;               /* nollataan RI-lippu */
          }
        }
      }
    TR0=0;                      /* Pysäytetään ajastin timer 0 */
    }
    if (laskuri==32)            /* Tarkistetaan onko pulssin tila */
    {                           /* muuttunut 32 kertaa */
      vastaanotto();            /* Jos on, siirrytään bitin */
    }                           /* vastaanottoaliohjelmaan */
```

```
    if(IN==0)         /* Jos portti P1.2 on tilassa 0 */
    {
      laskuri++;      /* Kasvatetaan P1.2 tilan muutoksia laskevan */
      TL0=0x00;       /* muuttujan arvoa yhdellä */
      TH0=0x00;       /* Nollataan ja käynnistetään ajastin timer 0 */
      TR0=1;
      if(T_in!=0)     /* Jos T_in-muuttujassa on lähettämätön merkki */
      {               /* Etsitään merkin ASCII vastine taulukosta */
        if(lisays==128)         /* Jos edellisellä kierroksella */
        {                       /* päästy taulukon loppuun, */
          lisays=0;   /* nollataan indeksin kasvatusmuuttuja */
        }
        for(j=lisays;j<lisays+16;j++) /* Käydään läpi 16 */
        {                             /* seuraavaa merkkiä */
          if (T_in==VARICODE[j]) /* Jos vastaavuus löytyi */
          {
            SBUF=j;             /* Lähetetään merkki PC:lle */
            TI=0;               /* sarjaportin kautta. */
            T_in=0;             /* Nollataan TI ja T_in */
          }
        }

        lisays=lisays+16;       /* 16 merkkiä läpikäyty */
      }
      while(IN==0)
      if(TH0>=0x04)             /* Jos pulssin pituus on yli 1 ms */
      {
        vast_bitti=0;           /* Asetetaan vastaanoton */
        if (RI==1)              /* apubitti nollaksi /*
        {
          if (SBUF==19)         /* Jos painettu Control+S */
          {
            RI=0;               /* Nollataan RI-lippu
            valinta=1;          /* Lähetin päälle */
            return;             /* Palataan pääohjelmaan */
          }
          else                  /* Jos esimerkiksi vahingossa */
          {                     /* painettu jotain muuta näppäintä */
            RI=0;               /* nollataan RI-lippu */
          }
        }
      }
    TR0=0;                      /* Pysäytetään ajastin timer 0 */
    }
    if (laskuri==32)            /* Tarkistetaan onko pulssin tila */
    {                           /* muuttunut 32 kertaa */
      vastaanotto();            /* Jos on, siirrytään bitin */
    }                           /* vastaanottoaliohjelmaan */

  }
}
```

```
/*****************************************************************/
void vastaanotto()
{
  if (vast_bitti==0)            /* Vastaanotettu bitti=0: */
  {
    kerroin=kerroin*2;          /* Kasvatetaan vain kerrointa */
    vast_bitti=1;               /* Asetetaan bitti takaisin 1:ksi */
    muisti++;                   /* Lisätään muistiin */
    laskuri=1;                  /* Pulssin tila-laskurissa 1 */
    if (vast_merkki==0)         /* Jos ennen nollaa */
    {                           /* ei ole vastaanotettu */
      kerroin=1;      /* ykköstä, eli vast_merkki-muuttuja on tyhjä: */
      muisti=0;       /* asetetaan muuttujat alkuarvoihinsa */
    }
  }
  else                /* Vastaanotettu bitti=1: */
  {                   /* Kasvatetaan muuttujaa kertoimen verran */
    vast_merkki=(vast_merkki+kerroin);
    kerroin=kerroin*2;          /* Jos ei ole vastaanotettu */
    muisti=0;                   /* kahta peräkkäistä nollaa, */
    laskuri=0;                  /* tyhjennetään muisti-muuttuja */
  }                             /* ja nollataan laskuri */
  if (muisti==2)      /* Jos vastaanotettu 2 peräkkäistä nollaa: */
  {                             /* siirretään merkki vastaanoton /*
    T_in=vast_merkki;           /*merkkivälimuistiin */
    vast_merkki=0;
    muisti=0;         /* asetetaan apumuuttujat alkuarvoihinsa */
    kerroin=1;
  }
}
/*****************************************************************/
void lahetin()
{
  EA=1;               /* Sallitaan keskeytykset */
  R_in=0;
  R_out=0;            /* Asetetaan lähetyksen */
  bitti=0;            /* apumuuttujat alkuarvoihinsa */
  for(;;)
  {
    if ((valinta==0) & (R_in-R_out==0)) /* Kaikki merkit lähetetty */
    {                            /* ja Control+R painettu: */
      return;                   /* palataan pääohjelmaan */
    }
    if (R_in-R_out!=0)  /* Siirtorekisterissä */
    {                  /* lähettämättömiä merkkejä? */
      lahetys();        /* Siirrytään ASCII--> varicode muunnos */
    }                   /* aliohjelmaan */

    else              /* Muussa tapauksessa lähetetään */
    {
      bitti=bitti^1;  /* ”00” eli peräkkäisiä vaihesiirtoja*/
      pulssi();
      bitti=bitti^1;
      pulssi();
    }
  }
}
/*****************************************************************/
```

```
void lahetys()
{
  decimal=taulukko[R_out];       /* Luetaan merkki muuttujaan */
  while (decimal!=0)             /* Pyöritään silmukassa */
  {                              /* kunnes decimal=0 */
    binaari=(decimal & 1);       /* decimal AND 1 -operaatio */
    if (binaari==1)              /* palauttaa eniten merkitsevän */
    {                            /* bitin binaarinä */
      pulssi();      /* Jos binaari=1: pulssin vaihetta ei käännetä */
    }
    else
    {
      bitti=bitti^1;  /* Jos binaari=0: invertoidaan apubitti */
      pulssi();       /* pulssin vaihe kääntyy 180 astetta */
    }
    decimal=(decimal >> 1);      /* decimal/2 */
  }        /* Silmukka jatkaa pyörimistään ellei decimal/2=0 */
  bitti=bitti^1;      /* Merkki kokonaisuudessaan lähetetty: */
  pulssi();           /* lähetetään ”00” eli 2 */
  bitti=bitti^1;      /* peräkkäistä vaihesiirtoa */
  pulssi();
  R_out++;            /* Kasvatetaan ”lähtevien” lukumäärää yhdellä */
  if (R_in-R_out==0)  /* Jos "tulevien" ja "lähtevien” */
  {                   /* erotus on nolla */
    R_in=0;           /* Nollataan "siirtorekisteri-laskurit" */
    R_out=0;
  }
}
/*********************************************************************/
void pulssi()         /* Varsinainen PSK-signaalin lähetys */
{
  if (bitti==0)       /* Jos bitti=0: */
  {
    for (i=0;i<32;)   /* pyöritään silmukassa 32 kierrosta */
    {
      PULSSI=1;       /* ykkönen ensin */
      viive();        /* 0,5 ms viive, f=1 kHz */
      PULSSI=0;       /* nolla */
      viive();        /* yhteensä 32 ms 1 kHz */
      i++;            /* taajuista kanttipulssia */
    }
  }
  else                /* Jos bitti=1 */
  {
    for (i=0;i<32;)   /* toiminta kuten edellä */
    {
      PULSSI=0;       /* mutta vaihe kääntyy 180 astetta */
      viive();
      PULSSI=1;
      viive();
      i++;
    }
  }
}
/*********************************************************************/
```

```
void viive()          /* 0,5 ms viive 1 kHz pulssia varten */
{
  TL0=0x23;           /* Ladataan ajastimeen FE23 Hex */
  TH0=0xFE;
  TR0=1;              /* Käynnistetään ajastin */
  while(TF0==0);      /* Odotellaan TH0 ylivuotoa */
  TR0=0;              /* Nollataan ylivuotobitti */
  TF0=0;              /* Pysäytetään ajastin */
}
```