TEKNIIKKA JA LIIKENNE

Sähkötekniikka

Elektroniikka

INSINÖÖRITYÖ

Mikrokontrolleriohjattavan lähetinvastaanottimen suunnittelu, toteutus ja testaus

Työn tekijä: Aki Virtanen
Työn ohjaaja
ja valvoja: Esa Häkkinen

Työ hyväksytty: 31.3. 2010

Esa Häkkinen
Yliopettaja

ALKULAUSE

Tämä insinöörityö tehtiin opiskelijan omasta toivomuksesta omiin ja koulun tarpeisiin. Haluan Kiittää työssäni Agilent Technologies -yritystä työssä käytetyistä mittalaitteista, sekä sovellusinsinööri Markku Salosta asiantuntevasta testausavusta.

Helsingissä 31.3.2010

Aki Virtanen

TIIVISTELMÄ

Työn tekijä: Aki Virtanen

Työn nimi: Mikrokontrolleriohjattavan lähetinvastaanottimen suunnittelu, toteutus ja testaus

Päivämäärä: 31.3.2010 **Sivumäärä:** 42 s. + 6 liitettä

Koulutusohjelma: Sähkötekniikka

Suuntautumisvaihtoehto: Elektroniikka

Työn ohjaaja ja valvoja: Yliopettaja Esa Häkkinen

Tässä insinöörityössä suunniteltiin, rakennettiin ja testattiin mikrokontrolleriohjattu lähetin-vastaanotinpari.

Työn suunnittelu- ja toteutusvaiheissa selvitettiin digitaalisen tiedonsiirron periaatteita ja ongelmia, jotka tulisi ottaa huomioon, kun suunnitellaan ja testataan digitaalista tiedonsiirtoa käyttäviä laitteita. Lisäksi selvitettiin työssä käytetyn RFID-piirin signaalin muodostamiseen ja vastaanottamiseen osallistuvien eri lohkojen toimintaa.

Työn testausvaiheessa testattiin lähetinvastaanottimen lähetysominaisuuksia mittausten ja testien avulla. Lisäksi selvitettiin, mitä ongelmia mittauksissa esiintyi ja mitä ongelmia pystyttiin ratkaisemaan mittaamalla.

Työ toteutettiin käyttämällä 300 - 1 000 MHz toimivaa RFID-piiriä. Tiedonsiirrossa lähetin-vastaanottimet käyttävät FSK-modulaatiota. Demoduloimista varten on lähetinvastaanottimissa kvadratuuri-ilmaisin. RFID-piirien toimintaa ohjaamaan valittiin Atmelin 8051-piiriperheen mikrokontrolleri, johon kirjoitettiin ohjelma Assembly-ohjelmointikielellä.

Lähetinvastaanotin-järjestelmä saatiin toimivaksi ja testattua. Lisäksi syntyi arvokasta tietoa ja kokemusta mikrokontrolleriohjauksesta, digitaalisista modulaatioista ja demodulaatioista, ohjelmoinnista ja sulautettujen järjestelmien suunnittelusta ja testauksesta.

Avainsanat: RFID, modulaatio, ohjelmointi, testaus

Metropolia

ABSTRACT

Name: Aki Virtanen

Title: Design and build of mikrocontroller controlled transceiver

Date: 31.3.2010 **Number of pages:** 42 s. + 6 appendixes

Department: Electrical Engineering

Study Programme: Electronics

Instructor and Supervisor: Esa Häkkinen, Head of Electronics Engineering

In this graduate project two microcontroller controlled transceivers was designed, builded and tested

During design and build phases principals and common problems were presented. Those problems should be noticed while designing and testing embedded systems which use digital communication. In addition, features that take part in the signal creation and reception were presented from the transceiver.

During test phase transceivers transmit parameters were tested, and test related problems were presented, in addition, with some problems that could be solved with proper measurements.

The transceiver was based on RFID-circuit operates in frequency bands between 300 - 1 000 MHz. Transceivers use FSK-modulation and have quadrature detector circuit for demodulation. Transceivers were controlled by Atmel AT89S8252 microprocessors. Program was written with Assembly.

Goals that were set to this graduate project was never fully meet. Although, valuable information and experience were gained of microprosessors, digital modulation and demodulation and design and testing of embedded systems.

Keywords: RFID, modulation, programming, testing

SISÄLLYS

LYHENTEET

FSK	*Frequency Shift Keying;* digitaalisen tietovirran lähetyksessä käytettävä taajuusmodulaatio; kantoaallon taajuutta poikkeutetaan lähetettävän bittivirran tahdissa
RFID	*Radio Frequency Identification;* Radiotaajuinen etätunnistus
SPI	*Serial Peripheral Interface; s*ynkronoitu, sarjamuotoinen tiedonsiirtoväylä
UART	*Universal Asynchronous Receiver Transmitter;* mikrokontrollereiden ominaisuus, jolla muutetaan rinnakkaismuotoista tietoa sarjamuotoiseksi; käytetään yleisesti yhdessä muiden tiedonsiirtostandardien, kuten RS-232:n, kanssa

1 JOHDANTO

Tämä insinöörityö käsittelee RFID-piiristä ja mikrokontrollerista rakennettavan lähetinvastaanotin parin suunnittelua, rakentamista ja testausta. Työn eri vaiheita selvitetään siten, että lukijalle selviää lähetinvastaanottimien ja RFID-laitteiden rakenne ja toiminta.

Insinöörityössä käsitellään ensimmäiseksi työhön valittavia komponentteja, sekä niiden ominaisuuksia. Komponenteista esitellään tarkemmin antennipiirin komponentit, AT89S8252 mikrokontrolleri ja TRC-101 RFID-piiri.

Mikrokontrolleri tarvitsee toimintaansa ohjelman, jotta lähetinvastaanotin voisi toimia. Ohjelmointia ja ohjelmointikielen valintaa käsitellään työssä laajemmin, koska ohjelmointi oli suurin yksittäinen työvaihe. Valitun ohjelmointikielen käyttö tarkoitti myös sitä, että mikropiirin ominaisuudet on selvitettävä tarkkaan.

RFID-piiriksi valitun TRC-101:n toimintaa ja ominaisuuksia käsitellään laajasti. Piiriä ohjataan sarjamuotoisen väylän kautta. Ohjaaminen suoritetaan kirjoittamalla arvoja piirin tilarekistereihin. TRC-101 piiristä käsitellään tärkeimmät tilarekisterit, niiden osoittamismuodot ja vaikutus piirin toimintaan. TRC-101 piirin tärkeimmät toimintalohkot käsitellään siten, että lukijalle selviää miten piirissä muodostetaan lähetettävä signaali ja sen modulointi, sekä miten signaali vastaanotetaan ja demoduloidaan.

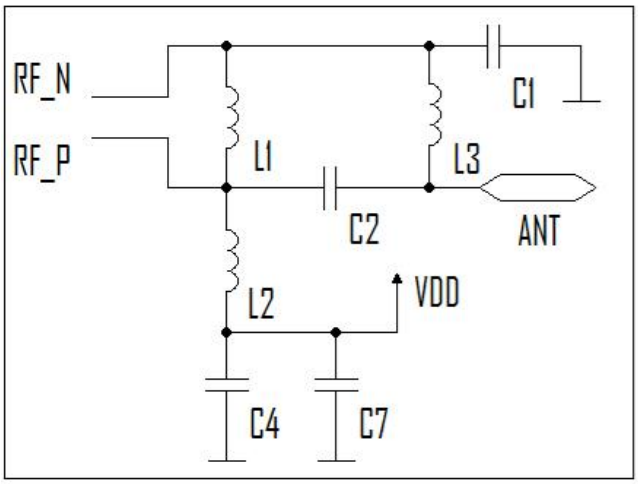
Työn lopuksi käsitellään mahdollisia parannus- ja kehitysideoita. sekä pohditaan työn eri vaiheissa esiintyneitä ongelmia ja ratkaisuja.

Kun lähetinvastaanottimet oli rakennettu valmiiksi, voitiin aloittaa ominaisuuksien testaus. Lähetinvastaanottimista testattiin muutamia lähetysominaisuuksia, jotta selviäisi vastaavatko valmistajan lupaamat arvot työssä saavutettuja arvoja.

2 LÄHETINVASTAANOTTIMEN SUUNNITTELU

2.1 Komponenttien valinta

Suunnittelu aloitettiin sopivien komponenttien valinnasta. *RFID*-piirin valinnassa päädyttiin RF Monolithicsin valmistamaan TRC-101-piiriin. Etuina on sen laaja 700 MHz:n taajuusalue sekä yhteensopivuus jo hankittuihn mikropiireihin. Komponenttien valinnassa oli suurena apuna RF Monolithicsin valmiiksi julkaisema antennisovituspiiri komponenttiarvoineen. /1, s. 4./



Kuva 1 Antennisovituspiiri. Kelat L1, L2 ja kondensaattorit C4 ja C7 muodostavat piirin, jolla synytetään biasointijännite.Loput komponentit mukaan lukien L1 muodostavat antennisovituspiirin. /1. mukaillen/

Taulukko 1.Antennisovituspiirin komponenttiarvot /1, s. 5/

Ref des	315	433	868	916
C1	6.8 pF	5.1 pF	2.7 pF	2.7 pF
C2	3.9 pF	2.7 pF	1.2 pF	1.2 pF
C4	0.1 цF	0.1 цF	0.1 цF	0.1 цF
C7	100 pF	100 pF	100 pF	100 pF
L1	56 nH	33 nH	8.2 nH	8.2 nH
L2	390 nH	390 nH	100 nH	100 nH
L3	68 nH	47 nH	22 nH	22 nH

Elektroniikan kehittyessä pienemmäksi valmistajat integroivat piireihin valmiiksi mahdollisimman paljon piirin tarvitsemia ulkoisia komponentteja.

Tämä vähentää huomattavasti tarvittavien ulkoisten komponenttien määrää. Tästä johtuen välttämättömiksi ulkoisiksi komponenteiksi jäi ainoastaan 10 MHz:n referenssioskillaattori ja antennisovituspiirin komponentit, antenniliitin mukaanlukien. Korkeimman käytettävän taajuuden jäädessä alle 1 Ghz, valittiin alustavasti antenniliittimeksi 50 ohmin BNC-liitin. Jos vastaanotossa olisi käytetty analogista kantataajuuden suodatusta, olisi tarvittu vielä yksi ylimääräinen ulkoinen kondensaattori.

Lähetinvastaanotinpiiri tarvitsee toimiakseen ulkoisen mikropiirin. Piirivalinnassa päädyttiin Atmelin 8051 perheeseen kuuluvaan AT89S8252 mikropiiriin. Mikropiiristä oli jo rakennettu ohjausyksikkö monipuolisine liitäntöineen aikaisempien vuosikurssilaisten toimesta (liite 4). /2./

2.2 Ohjelmointikielen valinta

Suunnittelun edetessä paljastui ohjelmoinnin merkitys kokonaisuudessaan. Ohjelmointi tulisi kuluttamaan suuren osan suunnittelu- ja rakennusvaiheen ajasta. Ohjelmointikieleksi valittiin alustavsti Assembly korkeamman luokan ohjelmointikielten sijaan. Suunnitteluvaiheessa tehtiin oletus, että mikrokontrollerin toiminta olisi liian hidasta sujuvaan tiedonsiirtoon. Tämän takia mikrokontrollerin tulisi pystyä suorittamaan ohjelmaa mahdollisimman nopeasti. Valintaan päädyttiin, koska mikropiirin ei tarvitse suorittaa matemaattisia operaatioita eikä signaalin käsittelyä vaan toimia ainoastaan lähetinvastaanotinpiiriä ohjaavana kontrollerina. Lisäksi mikropiirin pieni kahden kilotavun ohjelmamuisti sekä helpompi rekistereiden bittikohtainen osoittaminen nopeuttaa toteutusta.

Assembly ohjelmointikieltä voidaan pitää toisen sukupolven ohjelmointikielenä. Assemblya alemmalla tasolla ohjelmointikielten hierarkiassa on ainoastaan konekieli. Assemblyn vahvuutena on, että sillä pystytään vaikuttamaan piirin yksittäisten rekistereiden, ja näiden yksittäisiin bitteihin tehokkaasti ja vaivattomasti. Tämän ansiosta ohjelmista saadaan nopeampia ja muistin tarpeeltaan pienempiä kuin esimerkiksi C-kielellä kirjoitettuna. ohjelmarivien kirjoitusta tulee enemmän, koska korkeamman tason ohjelmointikielet suorittavat automaattisesti operaatioita, jotka nyt tarvitsee kirjoittaa itse. /6, s. 209./

Koska Assemblyllä kirjoitettaessa on tiedettävä tarkalleen ohjelmoitavan piirin ominaisuudet, on samaa ohjelmaa erittäin vaikeaa saada toimimaan muilla piirityypeillä. /6, s. 209./

2.3 Piirilevysuunnittelu

TRC-101:tä varten oli rakennettava oma piirilevy, koska ohjauksessa käytettävä AT89S8252 oli jo valmiiksi omalla piirilevyllään. Piirilevyksi valittiin levy, jossa on kuparointi molemmilla puolilla. Lähtökohtaisesti pyrittiin piirilevyn pohja pitämään yhtenäisenä maadoitustasona. Johdinten määrän kasvaessa, ajatuksesta jouduttiin osittain tinkimään. Koska ainoa suuria taajuuksia sisältävä alue piirilevyllä olisi antennisovituspiiri ja antenni, jätettiin näiden alapuolinen kuparointi yhtenäiseksi maatasoksi. Koska signaaliteissä ei siirretä korkeataajuuksisia signaaleja, voitiin nämä sijoittaa piirilevylle pinta-ala tehokkaasti läpivientien avulla käyttäen piirilevyn molempia puolia.

Laajalla, yhtenäisellä, lähellä signaaliteitä olevalla maatasolla saavutetaan paremmat häiriönsieto-ominaisuudet maatason ja johtimien välisien johdin-silmukka-alojen pienentyessä olemattomiin.

Piirilevyn suunnittelussa käytettiin CadSoftin Eagle-ilmaisohjelmaa, joka on yksinkertainen käyttää. Ohjelma sisältää myös suuret kirjastot valmiita komponenttimallinnuksia. Suunniteltu piirilevy jyrsittiin koulun jyrsimellä.

2.4 Lainsäädäntö

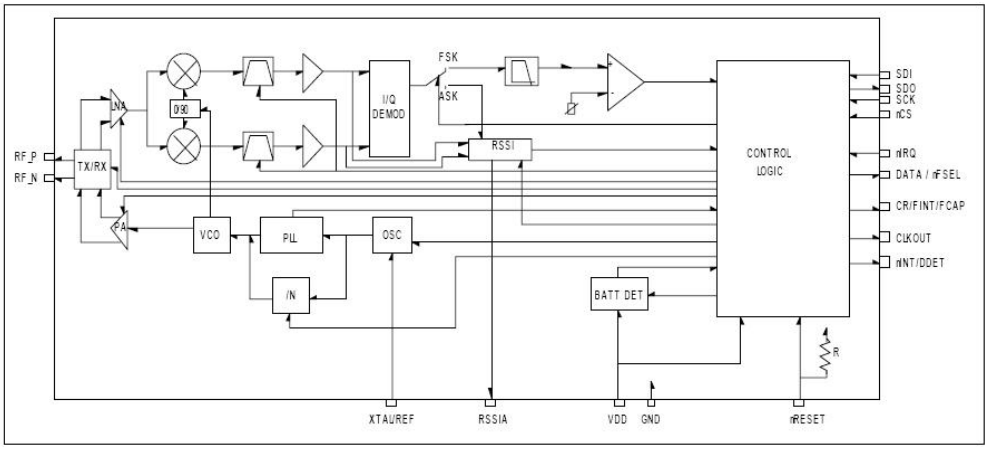
Ensimmäisestä radiolähetyksestä lähtien on langattoman median käyttö tiedonsiirrossa jatkuvasti kasvanut. Useiden eri viranomaistahojen, teollisuuden ja siviilien tiedonsiirtojen tapahtuessa langattomasti on selvää, että taajuuksien käyttöä tulee valvoa. Suomessa radiotaajuuksien valvonnasta vastaa viestintävirasto. Koska nykyään lähes kaikki viestintään soveltuvat taajuudet ovat jo käytössä, ei ole lainsäädännön kannalta mahdollista rakentaa lähetintä toimimaan millä tahansa taajuudella. Seuraukset eivät välttämättä olisi suotuisat, jos oma suuritehoinen lähetin häiritsisi lennonvalvonnan ILS-järjestelmän signaalia. TRC-101-piiri on valmistajan mukaan suunniteltu ainakin käytettäväksi etätunnistus-sovelluksissa, ja etätunnistussovelluksille on viestintävirastolta asetus, joka mahdollistaa suhteellisen vapaan käytön

asetuksen asettamissa rajoissa. / 3, s 13./ Lyhyesti ilmaistuna, asetus mahdollistaa lähetinvastaanottimen käytön taajuuskaistalla 865 - 868 MHz.

3 LÄHETINVASTAANOTTIMEEN VALITUT KOMPONENTIT

3.1 Lähetinvastaanotinpiiri TRC-101

Lähetinvastaanotinpiiriksi valittu TRC-101 on RF Monolithicsin valmistama, ja erityisesti paristokäyttöisiin langatonta tiedonsiirtoa vaativiin sovelluksiin suunniteltu. Tiedonsiirrossa se käyttää neljää kaistaa 300 - 1 000 MHz välillä. Jokaisella neljällä kaistalla on käytettävänään vähintään 95 kanavaa 100 kHz välein. Piiriin on integroitu kaikki kantataajuuden muodostamiseen, moduloimiseen ja demoduloimiseen tarvittavat ominaisuudet suunnittelun helpottamiseksi (kuva 2).

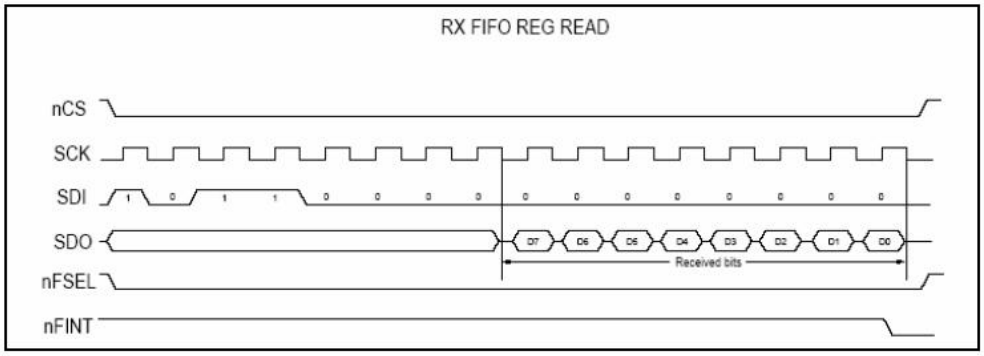


Kuva 2. TRC-101:n lohkokaavio. Kuvassa on esitettynä TRC-101:n toimintojen lohkot. Kuvan vasemmassa laidassa on differentiaalisen ulostulon portit. TX/RX-kytkimen oikealla puolella on signaalin muodostamisessa käytettävä syntetisaattori, ja sen ylä puolella vastaanottimen kvadratuuri-ilmaisin ja demodulaattori. TRC-101-piirin logiikkatoiminnot on kuvattu yhtenä Control Logic -lohkona.

Piirissä olevan SPI-väylän ansiosta piiri on yhteensopiva erittäin monen piiriperheen kanssa. Ulkoisina komponentteina piiri tarvitsee vähimmillään 10 MHz referenssioskillaattorin, mikä on suuri etu mobiileissa sovelluksissa. Tiedonsiirrossa käytetään *frequency shift keying* -modulaatiota.

TRC-101-piirissä on viisitoista 16-bittistä rekisteriä, joista 14:ään voidaan kirjoittaa ja kahta lukea. Tärkeimmät rekisterien asetukset voidaan lukea statusrekisteristä. Statusrekisteriin ei myöskään voida kirjoittaa. Koska

TRC-101-piirin rekisterit ovat 16-bittisiä, tarvitsee mikrokontrolleriin suorittaa kaksi kirjoitus- ja lukutoimenpidettä jokaista rekisteriä kohden (kuva 3).



Kuva 3 . 16-bittisen vastaanotto rekisterin lukeminen. Ensin lähetetään komentotavu, josta TRC-101 tietää, että käyttäjä haluaa lukea vastaanotto rekisterin sisällön. Koska vastaanotto-rekisteriä voidaan ainoastaan lukea, tulee komentotavun jälkeisen tavun arvo olla 0x00 heksadesimaalilukuna ilmoitettuna. Rekistereihin kirjoitettava arvo lähetetään komentotavun jälkeen. nFSEL- ja nFINT-porttien tilat on huomioitava kun käytetään lähetys- tai vastaanottorekisteriä.

Käyttötarkoituksesta riippuen lähetinvastaaanottimella on aina välejä, milloin sen ei tarvitse lähettää eikä vastaanottaa. TRC-101:ssä onkin torkkutoiminto jolla voidaan vähentää virran kulutusta. /1./

3.1.1 TRC-101 Rekisterien esittely

TRC-101 on monipuolinen mikropiiri, jossa on paljon erilaisia ominaisuuksia. Näitä ominaisuuksia voidaan käyttää kirjoittamalla haluttuja arvoja piiriin asetuskohtaisiin rekistereihin. Tässä kappaleessa esitetään muutamien tärkeimpien rekisterien ominaisuuksia. Tarkemmat esittelyt voi lukea datalehdestä. /1, s. 13 – 30./

Taulukko 2. Status Register. Status Register:stä voidaan likea TRC-101-piirin tilatiedot. Rekisteriä voidaan ainoastaan lukea. /1, s. 14./

Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIFTXRX	POR	FI-FOV/UR	WKINT	INTRST	LB	FIFEMP	RSSI/AT	GDQD	CRLCK	AFATGL	OFFSGN	OFF3	OFF2	OFF1	OFF0

On rekisteri, johon ei voida kirjoittaa. Rekisteristä voidaan nopeasti lukea kaikkien muiden rekisterien tärkeimmät asetukset.

Rekisteri on järjestetty siten, että ensimmäisenä luettavat kuusi bittiä sisältävät tiedon tapahtumista, jotka mahdollisesti vaativat nopeaa reaktiota. Näitä

tapahtumia ovat lähetys- ja vastaanottorekisterien tilatiedot, keskeytyspyyntöportin tilatieto, herätysajastimen tilatieto ja käyttöjännitteen tilatieto. Viimeiset kymmenen bittiä sisältävät tiedon vastaanotetun signaalin voimakkuudesta ja lähetys/vastaanottotiedon laadusta. Myös vaihelukitun silmukan tarvitsemat korjausarvot voidaan lukea täältä. /1, s. 15./

Taulukko 3.Configuration Register. Rekisterissä asetetaan yleisiä käyttöasetuksia. Komentotavun arvo on 0x80h. /1, s. 15./

Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	DATEN	FIFEN	BAND1	BAND0	CAP3	CAP2	CAP1	CAP0

Rekisterissä on asetukset lähetysrekisterin aktivointiin, vastaanottorekisterin aktivointiin, käytettävän taajuuskaistan aktivointiin ja referenssioskillaattorin kuormituskapasitanssin valintaan.

Taulukko 4.Automatic Frequency Adjust Register. Rekisterissä asetetaan ehdot ja arvot siitä, miten TRC-101-piirin tulisi säätää toimintataajuuttaan pysyäkseen samalla taajuudella lähetinvastaanotin parinsa kanssa. Komentotavun arvo on 0xC4h. /1, s. 16 - 17./

Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	1	0	0	AUTO1	AUTO2	RNG1	RNG0	STRB	ACCF	OFFEN	AFEN

Rekisterissä asetetaan toimintatapa jolla TRC-101-piirin vaihelukittusilmukka hienosäätää viritystaajuutta. Näin vastaanotin voi helposti lukittua lähetykseen. Vaihtoehtoina ovat manuaalisesti, automaattisesti tai kerran suoritettava viritys.

Manuaalisessa toimintatavassa ohjelman kirjoittaja päättää aikavälin, jolla mikrokontrolleri ilmoittaa näytteenottotaajuuden. Automaattisessa toimintatavassa TRC-101 suorittaa näytteenoton aina kun *Valid Data Detector* –portti aktivoituu. Taajuuden säätö voidaan asettaa myös tarkkuustilaan, jolloin mittausepävarmuus vähenee, mutta mittausaika kasvaa. Rekisterissä on myös liput, joilla aktivoidaan vaihelukitunsilmukan ohjaus sekä ohjauksessa käytettävän korjausarvon laskeminen.

Taulukko 5.Transmit Confiiguration Register. Rekisterissä asetetaan kaikki muut lähetysparametrit paitsi lähetystaajuus. Komentotavun arvo ilmaistaan vain seitsemällä bitillä. /1, s. 18./

Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	1	0	0	MODP	DEV3	DEV2	DEV1	DEV0	0	PWR2	PWR1	PWR0

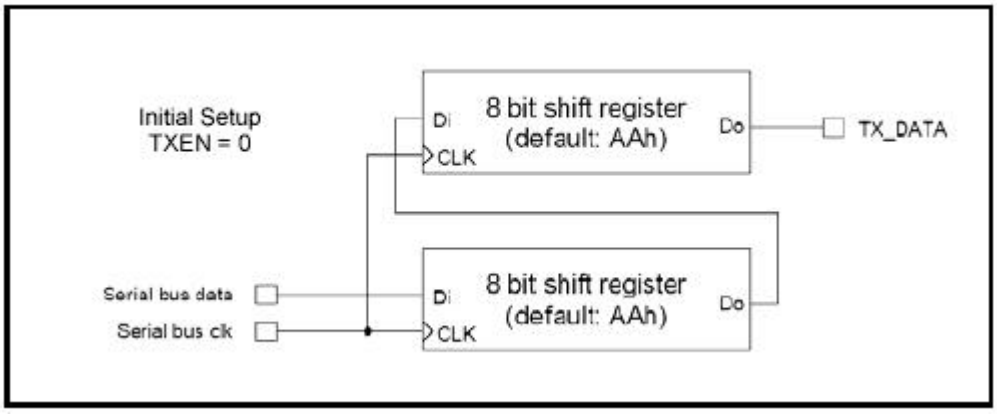
Rekisterissä on asetukset modulaation kaistanleveyttä ja polaarisuutta varten. Kaistanleveys voidaan asettaa 15 kHz:stä 240 kHz:iin 15 kHz:n välein. Polaarisuudella tarkoitetaan, kumpaa sivukaistaa käytetään lähettämään nolla- ja ykkösbitti.

Lähetystehoa voidaan säätää suhteessa suurimpaan mahdolliseen lähetystehoon. Lähetystehoa voidaan vaimentaa kolmen desibelin välein -21 desibeliin asti.

Taulukko 6.Transmit Register. Lähetyksessä käytettävä puskurirekisteri. Komentotavun arvo on 0xB8h. /1, s. 19./

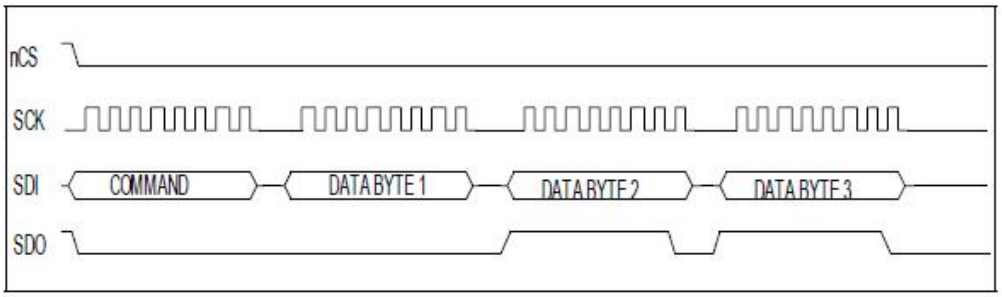
Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	0	0	0	TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0

Lähetyksessä käytettävä puskurirekisteri, mihin lähetettävä tieto kirjoitetaan. Kirjoitus suoritetaan *SPI*-väylän kautta (kuva 4). Vaikka *Transmit Register*:ssä on vain kahdeksan bittiä lähetettävälle datalle, koostuu puskuri kahdesta kahdeksan bittisestä siirtorekisteristä. Käytännössä tämä tarkoittaa, että lähetettäviä tavuja tulisi aina kirjoittaa *Transmit Register*iin parillinen määrä. Jos tavuja syötetään pariton määrä, jää TRC-101 odottamaan puuttuvaa tavua lähetyksen loppuessa. Tämä johtaa siihen, että viimeinen tavu jää lähettämättä.



Kuva 4. Transmit Registerin *muodostavat kaksi sarjaan kytkettyä siirtorekisteriä.*

Kuvassa neljä on esitettynä valmistajan suosittelema, ja työssä käytetty tapa, jolla lähetysrekisteriin tulisi kirjoittaa.



Kuva 5 Valmistajan suosittelema tapa lähetysrekisteriin kirjoittamiselle. Lähetysrekisteriin kirjoitetaan normaalisti komentotavu ja lähetettävä tavu. Erona tavalliseen rekisteriin kirjoitukseen on, että nCS-portin tila pidetään alhaalla koko lähetettävän tietovirran ajan. TRC-101-piiri pitää SDO-portin tilan alhaalla, kun lähetysrekisteriin voidaan kirjoittaa uusi tavu. /1, s. 19./

Lähetysrekisteriin kirjoitetaan ensimmäisenä komentotavu, josta TRC-101 tietää, että seuraavat tavut kirjoitetaan lähetysrekisteriin. TRC-101 muuttaa *SDO*-portin tilan nollaksi aina, kun edellinen tavu on lähetetty ja lähetysrekisteri on valmis vastaanottamaan seuraavan tavun.

Taulukko 7.Frequency Setting Register. Rekisterin arvoilla asetetaan tarkka toiminta-taajuus. Komentotavun pituus on neljä bittiä. /1, s. 20./

Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	Freq11	Freq10	Freq9	Freq8	Freq7	Freq6	Freq5	Freq4	Freq3	Freq2	Freq1	Freq0

Rekisterissä asetetaan tarkka toimintataajuus valitun taajuuskaistan sisällä. Taajuuden valinta suoritetaan kaavalla yksi

$$f_c = B_1 + \frac{B_0 + f_{val}}{4000} MHz\,, \qquad (1)$$

missä f_{val} on bittien 0 - 11 arvo desimaalilukuna ja 96 < f_{val} < *3 903.*

B_0 *ja* B_1 ovat vakioita, joiden arvot saadaan taulukosta 2.

Taulukko 8.Taajuuden asettamisessa käytettävät vakiot viitettä 1. mukaillen

Taajuuskaista (MHz)	**B1**	**B0**
315	1	31
433	1	43
868	2	43
916	3	30

Kaikkien arvojen tulee olla ilmoitettujen rajoitusten mukaisia. Jos näin ei toimita, säilyy rekisterissä jo ollut arvo voimassa.

Taulukko 9.Receiver Control Register. Rekisterissä asetetaan yleisiä vastaanotto-asetuksia. Komentotavun pituus on viisi bittiä. /1, s. 21 - 22./

Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	INT/VDI	VDIR1	VDIR0	BB2	BB1	BB0	GAIN1	GAIN0	RSSI2	RSSI1	RSSI0

Aivan kuten *Transmit Configuration* -rekisterissä, *Receiver Control* -rekisterissä asetetaan lähetyksen vastaanottoon vaikuttavia asetuksia. Rekisterissä voidaan asettaa portin 16 toimintatapa, vastaanottimen kaistanleveys, *Valid Data Detector* –piirin vasteaika, etuvahvistimen vahvistus ja digitaalisen *RSSI* –piirin raja-arvo.

Kantataajuuden modulaatiokaistaleveys on portaittain säädettävissä 67 kHz 400 kHz. Etuvahvistimen vahvistusta voidaan säätää portaittain nollasta desibelistä -21 dB:iin asti.

RSSI (Receiver Signal Strength Indicator) ilmoittaa, onko vastaanotettavan signaalin teho suurempi kuin kuin rekisteriin asetettu raja-arvo. Raja-arvo voidaan säätää välillä -103 dB:stä -73 dB:iin kuuden desibelin välein. Rekisterissä voidaan asetettaa myös portin 16 toimintamuoto sekä *Valid Data Detector* -piirin vasteaika.

Taulukko 10.Baseband Filter Register. /1, s. 23./

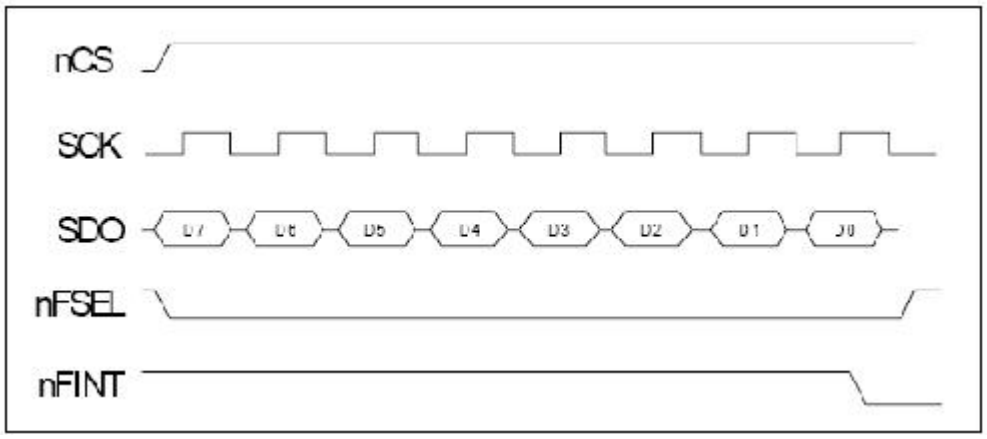
Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	1	0	CRLK	CRLC	1	FILT	1	DQLVL2	DQLVL1	DQLVL0

Rekisterissä on lisää vastaanottoon vaikuttavia asetuksia. Rekisterissä asetetaan *Clock Recovery* –piirin toimintatapa, *Valid Data Detector* –piirin raja-arvo ja kantataajuussuodatin analogiseksi tai digitaaliseksi.

Taulukko 11.FIFO Read Register. Rekisteriin luetaan vastaanotettu tavu. Rekisteriä voidaan ainoastaan lukea, ja sen komentotavun arvo on 0xB0h. /1, s. 24./

Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	0	0	0	0	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0

Vastaanotossa käytettävä *First In First Out* -tyyppinen vastaanottorekisteri. Lukeminen voidaan suorittaa kahdella eri tavalla. Joko odottamalla keskeytyspyyntöä ja lukemalla Status-rekisteristä keskeytyspyynnön syy. Tämän jälkeen lukemalla *FIFO Read Registerin* arvo (KUVA 5). Vastaanottorekisteri valinta lukemista varten toteutetaan lähettämällä SPI-väylällä ensin arvo 0xB0h.



Kuva 6 Vastaanottorekisterin lukeminen seuraamalla nFINT-portin tilaa. Tämä on vaihtoehtoinen, ja nopeampi tapa lukea vastaanottorekisteri kuin kuvassa 3 esitetty tapa, jossa odotetaan keskeytyspyyntöä ja luetaan Status-rekisteri ennen vastaanottorekisterin lukemista.

Toinen tapa on seurata *nFINT*-portin tilan muutosta, ja valita vastaanottorekisteri lukemista varten *nFSEL*-portin avulla. Vastaanottorekisteriin luetaan suoraan lukematta ensin Status Rekisteriä.

Taulukko 12.FIFO and Reset Mode Configuration Register. Receiver Control Register:n lisäksi, tässä rekiserissä asetetaan myös vastaanottoon liittyviä asetuksia. Asetukset kohdistuvat vastaanottorekisteriin lukemiseen ja keskeytyspyynnön esittämiseen. Komentotavun arvo on 0xCAh. /1, s. 25./

Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	0	FINT3	FINT2	FINT1	FINT0	0	FIFST	FILLEN	RSTEN

Rekisterissä asetetaan ehdot:

- Keskeytyspyynnön esittämisestä luettaessa vastaanotettavaa lähetystä vastaanottorekisteriin
- Tunnistetavujen käytöstä vastaanottorekisteriin lukemisen aloittamiseksi
- TRC-101:n resetoinnista havaittaessa vähintään 0,2 V:n nopea muutos käyttöjännitteessä.

TRC-101 voidaan ohjelmoida esittämään keskeytyspyyntö mikrokontrollerille, kun ennalta asetettu määrä vastaanotetusta lähetyksestä on luettu vastaanottorekisteriin. Näin varmistutaan, että mikrokontrollerilla on riittävästi aikaa valmistautua vastaanottorekisterin lukemiseen, eikä ylivuotoja tapahdu. Vastaanottorekisteri voidaan ohjelmoida myös täytettäväksi vasta, kun lähetyksestä on havaittu tunnistetavut.

Taulukko 13.Data Rate Setup Register. Rekisterissä asetetaan vastaanotettavan lähetyksen bittinopeuden arvo, ja valitaan käytetäänkö esivalitsinta. Komentotavun arvo on 0xC6h. /1, s. 26./

Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	1	1	0	PRE	BITR6	BITR5	BITR4	BITR3	BITR2	BITR1	BITR0

Tässä rekisterissä asetetaan vastaanottimelle oletusarvo vastaanotettavan signaalin bittivirran nopeudesta. Vastaanotin käyttää tätä arvoa laskiessaan aikavakiota digitaaliselle suodattimelle. Rekisterissä valitaan, käytetäänkö vastaanottimen esivalitsinta havaitsemaan mahdollisesti oletusarvoa hitaampaa bittivirtaa.

Bittivirralle lasketaan oletusarvo seuraavalla kaavalla:

$$D\mathrm{R}_{\mathrm{e}\,xp}(kbps) = \frac{10000}{29\times(BITR_{6\ldots0}+1)\times(1+PRE\times 7)}, \qquad (2)$$

jossa $BITR_{6\ldots0}$ rekisterin bittien 6 - 0 arvo desimaalilukuna, ja PRE on yksi, jos esivalitsin on aktivoitu, ja nolla jos esivalitsinta ei ole aktivoitu.

Taulukko 14.Power Management Register. Rekisterin asetuksilla hallitaan TRC-101-piirin lohkojen toimintaa kytkemällä niitä päälle tai pois päältä. Komentotavun arvo on 0x82h. /1, s. 27./

Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	0	RXEN	BBEN	TXEN	SYNEN	OSCEN	LBDEN	WKUPEN	CLKDIS

Tässä rekisterissä voidaan kytkeä päälle tai pois kantataajuuslohko, syntetisaattori, oskillaattoripiiri, oskillaattorin ulostulo, herätysajastin *Low Battery Detect* -piiri, vastaanotin- ja lähetinketju.

Taulukko 15.Wakeup Timer Period Set Register. Tähän rekisteriin kirjoitetaan halutut arvot herätysajastimen toimintaa varten. Komentotavu on vain kolme bittiä pitkä. /1, s. 28./

Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	R4	R3	R2	R1	R0	M7	M6	M5	M4	M3	M2	M1	M0

Rekisterissä asetetaan herätysajastimen herätysintervalli. Ominaisuutta ei työssä käytetty, joten sen yksityiskohtaisempi esittely sivuutetaan.

Taulukko 16.Duty Cycle Set Register. Rekisterien arvoja käytetään herätysajastimen kanssa. Näillä arvoilla asetetaan haluttu hereillä oloaika. Komentotavun arvo on 0xC8h. /1, s. 29./

Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	0	0	0	DC6	DC5	DC4	DC3	DC2	DC1	DC0	DCEN

Duty Cycle Set rekisteriä voidaan käyttää yhdessä herätysajastimen kanssa. Tässä rekisterissä asetetaan aika, jonka TRC-101 on hereillä herättyään torkusta. Tänä aikana on vastaanotinketju aktivoituna.

Taulukko 17.Battery Detect Threshold and Clock Output Register. Rekisterin arvoilla asetetaan referenssioskillaattorin taajuus, ja raja-arvo käyttöjännitteelle. /1, s. 30/

Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0	CLK2	CLK1	CLK0	LBD4	LBD3	LBD2	LBD1	LBD0

Rekisterissä asetetaan raja-arvo käyttöjännitteelle. Käyttöjännitteen alittaessa tämän arvon esitetään mikrokontrollerille keskeytyspyyntö. Lisäksi asetetaan oskillaattoriulostulon taajuus. Taajuus on ohjelmoitavissa 1 MHz:stä 10 MHz:iin (taulukko 18).

Taulukko 18.Oskillaattorin ulostulolle valittavat taajuudet

Oskillaattorin taajuus (MHz)	CLK2	CLK1	CLK0
1	0	0	0
1,25	0	0	1
1,66	0	1	0
2	0	1	1
2,5	1	0	0
3,33	1	0	1
5	1	1	0
10	1	1	1

Tiedonsirrossa voidaan käyttää kahta tunnistustavua aina lähetyksen alussa ilmoittamaan vastaanottajalle vastaanotettavasta lähetyksestä. Tunnistustavujen arvot ovat 0x2Dh ja 0xD4h. Nollaamalla *FIFO Fill Start Condition* -bitti, aktiovoidaan tunnistetavun käyttö. Vastaanottaja aloittaa kirjoittamaan lähetyksestä tunnistetavua seuraavat bitit vastaanottorekisteriin. Jos *FIFO Fill Start Condition* -bitti asetetaan, kirjoitetaan vastaanottorekisteriin kaikki mitä vastaanotin havaitsee riippumatta siitä, onko se kohinaa vai varsinaista lähetystä.

FIFO Fill Bit Count -biteillä asetetaan vastaanottorekisteriin luettavien bittien lukumäärä ennen keskeytyspyynnön esittämistä mikrokontrollerille. Tämä mahdollistaa mikrokontrollerille sopivasti aikaa valmistautua vastaanottorekisterin lukemista varten.

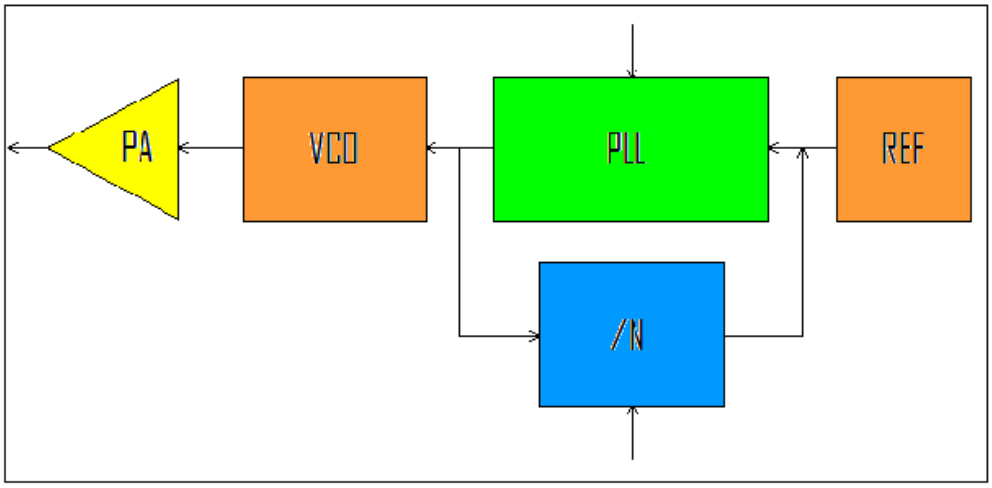
TRC-101-piiri voidaan myös ohjelmoida resetoimaan itsensä, jos se havaitsee 0,2 V:n piikin käyttöjännitteessä. Tämä toiminto aktivoidaan nollaamalla *Disable RESET Mode* -bitti.

3.1.2 Lähetinketju

Lähetinketju koostuu oskillaattorista, syntetisaattorista, vahvistimesta ja lähetysrekisteristä. Yksinkertaistaen voidaan sanoa, että lähetyksessä käytettävä signaali synnytetään jänniteohjatulla oskillaattorilla, jota vahvistetaan vielä vahvistimella ennen antennille syöttämistä. Jotta lähetettävästä signaalista saataisiin taajuusvakaampi, käytetään sen muodostamiseen lisäksi syntetisaattoria.

Syntetisaattori muodostuu referenssioskillaattorista, vaihelukitusta silmukasta ja jänniteohjatusta oskillaattorista. Syntetisaattori saa referenssisignaalin ulkoiselta 10 MHz:n oskillaattorilta, mistä muodostetaan vaihelukitulla silmukalla tarkka ohjausjännite jänniteohjatulle oskillaattorille (kuva 22). Syntetisaattorin avulla saadaankin luoduksi vakaampi taajuuksisia signaaleita kuin käyttämällä ainoastaan yhtä oskillaattoria signaalin muodostamisessa.

Vahvistin on avoin kollektori tyyppinen transistorivahvistin. Vahvistimen vahvistusta voidaan säätää maksimista kolmen desibelin välein -21 dB:iin asti.

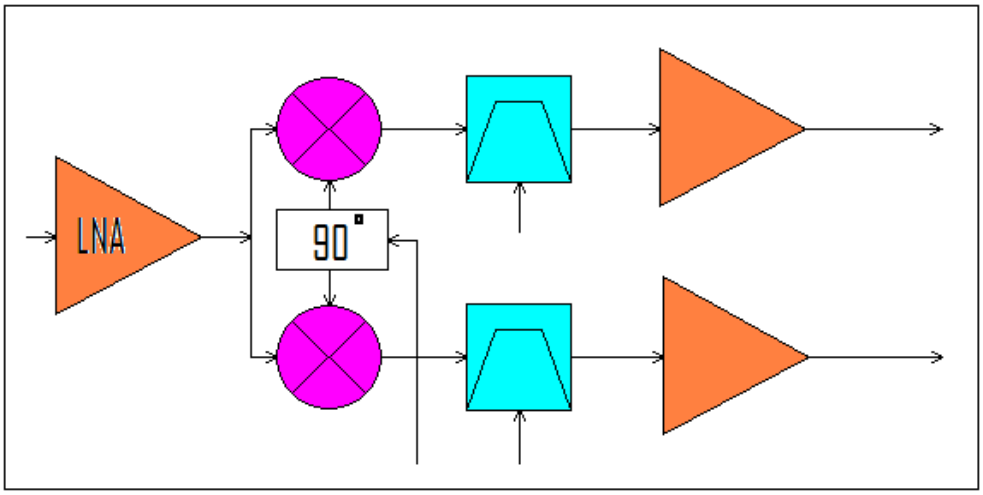


Kuva 7 Syntetisaattorin kuvan 2 lohkokaaviota mukaillen. Syntetisaattorissa johdetaan ulostulotaajuus negatiivisella takaisinkytkennällä vaihelukitulle silmukalle. Vaihelukittusilmukka synnyttää referenssioskillaattorin ja ulostulon taajuuksista erotaajuuden . Vaihelukitun silmukan synnyttämällä tasajännitteellä, joka on verrannollinen erotaajuuteen, ohjataan jänniteohjattua oskillaattoria./6/

3.1.3 Vastaanotinketju

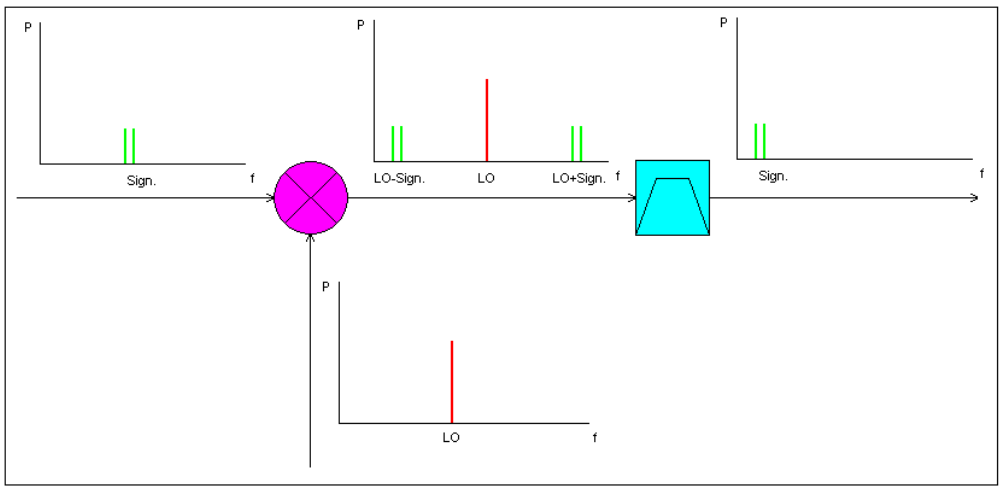
Lähetystä vastaanotettaessa, signaalin reitille osuu antennin jälkeen etuvahvistin. Etuvahvistimella voidaan tarvittaessa vahvistaa vastaanotettua signaalia. Esimerkiksi erittäin kohinallisissa olosuhteissa, joissa vastaanotettavan signaalin voimakkuus on samaa suuruus luokkaa kuin kohinan, saadaan signaali paremmin esiin käyttämällä esivahvistusta.

Koska kohinaa esiintyy sattumanvaraisesti, vahvistuu signaali suhteessa kohinaan enemmän. Esivahvistimen sisääntulo on 250 ohmia, joka tulisi huomioida antennisovitusta suunniteltaessa.



Kuva 4 Vastaanotinketjun etuvahvistin ja kvadratuuri-ilmaisin kuvan 2 lohkokaaviota mukaillen.

TRC-101:n vastaanotin on rakennettu siten, että vastaanotettu signaali sekoitetaan kiinteälle välitaajuudelle ennen demoduloimista. Vastaanotetun signaalin sekoittamisessa matalammalle taajuudelle on etuna se, että voidaan käyttää kiinteä taajuuksisia ja kapeampikaistaisia suodattimia. Tällä tavalla vastaanottimen selektiivisyys saadaan paremmaksi, kun sekoituksessa syntyvät ei-halutut signaalit saadaan paremmin suodatetuksi.

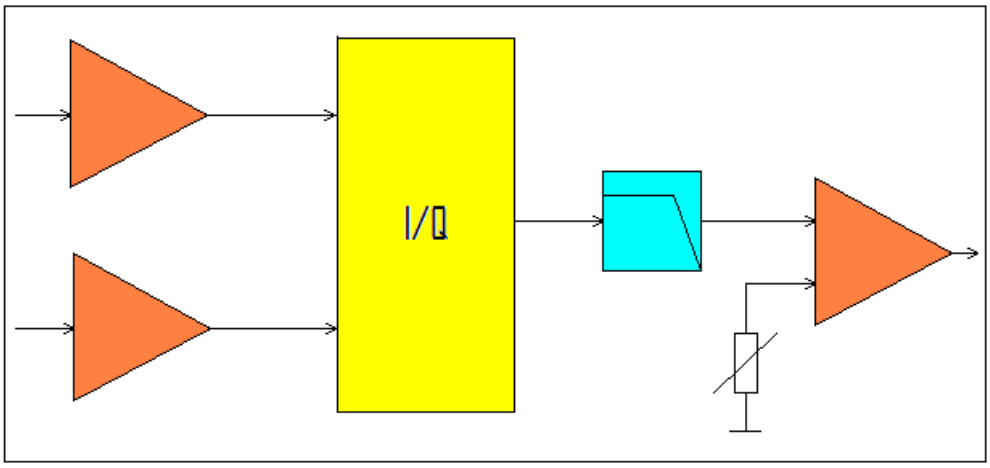


Kuva 8. Kuvassa 8 on esitettynä kuinka vastaanottimessa toteutetaan välitaajuudelle sekoitus ja ei haluttujen signaaleiden suodatus. Kun sekoittajaan johdetaan vastaanotettu ja paikallisen oskillaattorilta tuleva signaali, syntyy sekoittajan ulostulo

TRC-101:een on rakennettu kvadratuuri-ilmaisin. Kvadratuuritopologialla toteutetussa ilmaisimessa vastaanotettu signaali johdetaan kahteen eri signaalitiehen. Toisen signaalin vaihetta poikkeutetaan ensimmäiseen nähden

yhdeksänkymmentä astetta. Molemmat signaalit ohjataan tämän jälkeen omille komparaattoreille. Komparaattoreilla muodostetaan signaaleista sakara-aaltoa I/Q-demodulaattorille.

Demodulaattorilta tuleva signaali voidaan suodattaa joko analogisella tai digitaalisella alipäästösuodattimella. TRC-101 laskee digitaaliselle alipäästösuodattimelle aikavakion automaattisesti Data Rate Setup Register:iin asetetusta arvosta. Vastaanotin laskee myös FINT-portista saatavan kellosignaalin taajuuden tästä arvosta. Digitaalista alipäästösuodatinta käytettäessä, muodostetaan suodatetusta signaalista lopullinen sakara-aalto komparaattorilla.



Kuva 9 Vastaanotinketjun I/Q-demodulaattori, digitaalinen alipäästösuodatin ja komparaattori kuvan 2 lohkokaaviota mukaillen. Jos käytetään analogista alipäästösuodatusta, ohjataan signaali I/Q-demodulaattorilta suoraan sisäiselle kymmenen kilo-ohmin vastukselle ja porttiin 7.

Vastaanotinketjun jälkimmäinen osa, joka koostuu I/Q-demodulaattorista, digitaalisesta-alipäästösuodattimesta ja komparaattorista. Analoginen alipäästösuodatin muodostuu TRC-101:n sisäisestä kymmenen kilo-ohmin vastuksesta ja ulkoisesta kondensaattorista.

3.2 Ohjaava mikropiiri AT89S8252

AT89S8252 on 8051-piiriperheeseen kuuluva kahdeksanbittinen, tietoliikennesovelluksiin suunniteltu mikropiiri. Mikropiirissä on valmiina SPI- ja UART-väylät sekä neljä 8-bittistä rinnakkaisporttia, joita voidaan käyttää sisään- ja ulostuloina. Työn kannalta hyödyllisin ominaisuus on SPI-väylä, jota

käytetään kommunikoinnissa lähetinvastaanotinpiirin kanssa. Lisäksi käytetään neljän rinnakkaisportin yksittäisiä portteja viestittämään tilanmuutoksista.

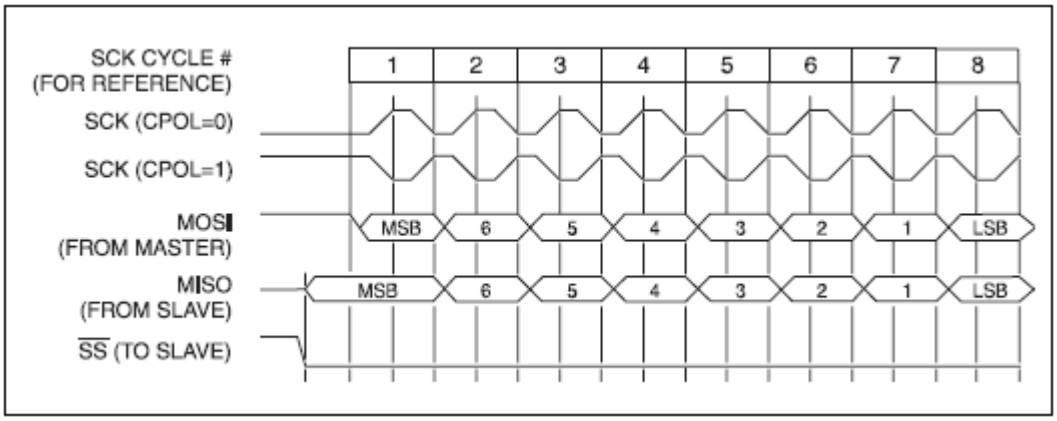
AT89S8252 on CMOS-teknologialla toteutettu kahdeksanbittinen tietoliikennesovelluksiin suunniteltu mikrokontrolleri. Siinä on kahdeksan kilotavua flash-muistia kirjoitettavaa ohjelmaa varten, 256 tavua käyttömuistia ja kahden kilotavun EEPROM-muisti. Toisin kuin käyttömuistissa, EEPROM-muistiin tallennettu tieto säilyy, vaikka virta mikropiirille katkaistaan. Mikropiirin vahvuuksiin voidaan lukea siihen rakennetut sisäiset UART- ja SPI-väylät, yhdeksän keskeytyslähdettä, kolme ajastinta tai laskuria ja kaksi eri virran säästötilaa.

Kaikille edellä mainituille ominaisuuksille on myös vähintään yksi keskeytysvektori osoitettuna kullekin. Mikropiiri tarvitsee yhden operaation suorittamiseen 12 kellopulssia, ja käskystä riippuen yksi käsky vaatii yhdestä neljään operaatiota. Kellotaajuuden ollessa 11,0592 MHz, oli lähtökohtaisesti oletettavissa, että mikropiiri ei tule suorituskyvyltään vastaamaan vaatimuksia. /2./

SPI (Serial Peripheral Interface) on tehokas työkalu mikropiirien väliseen tiedonsiirtoon. Väylään kuuluvista mikropiireistä on yksi aina isäntä ja loput renkejä. Jos renkejä on useampi kuin yksi, pitää ottaa käyttöön lisää signaaliteitä. Isäntä tarvitsee yhden niin sanotun *slave-select*, orjan valinta signaalin jokaista orjaa kohden. Orjat tarvitsevat vain yhden lisäsignaalitien. Pienin määrä signaaliteitä, kolme kappaletta, tarvitaan siis kahden mikropiirin ollessa samalla väylällä. Tietoa lähetetään tavu kerrallaan, ja jokaisen kellopulssin jälkeen keskenään keskustelevat mikropiirit vaihtavat lähetysväylän tilaa. Tietoa voidaan siis lähettää ja vastaanottaa samanaikaisesti. Väylälle valittu isäntä on kuitenkin ainoa, joka voi lähettää kellosignaalia, ja näin ollen määrää milloin väylällä liikennöidään.

Väylä on toiminnaltaan täysin staattinen, eli sen kellosignaalilla ei ole minimitaajuutta. Tämä mahdollistaa sen, että tietoa voidaan siirtää myös manuaalisesti vaihtamalla signaaliteiden tilaa. Väylää käytettäessä automaattisesti, nopeus määräytyy käytettävän kiteen mukaan. Maksiminopeus on kuitenkin kiteen taajuus jaettuna neljällä. Tässä tapauksessa 2,76 MHz.

SPI:tä ei ole standardisoitu, eikä tästä syystä ole kaikissa mikropiireissä täysin samanlainen. Asetuksissa voidaan kuitenkin määritellä, muutetaanko väylän tilaa kellosignaalin nousevalla vai laskevalla reunalla, luetaanko signaaliteiden tila negatiivisen vai positiivisen puolijakson aikana ja lähetetäänkö eniten merkitsevä vai vähiten merkitsevä bitti ensin.



Kuva 10. Kuvasta nähdään SPI-väylän ajoituskaavio. Kellosignaalin polaarisuutta ja vaiheistusta voidaan muuttaa paremman yhteensopivuuden saavuttamiseksi. Lisäksi voidaan valita lähetetäänkö eniten vai vähiten merkitsevä bitti ensin, ja toimiiko piiri orjana vai isäntänä. /2 ,s 19./

3.3 Diskreettikomponentit

Antennisovituspiirin komponentteja valittiin 868 MHz:n kaistan asettamien vaatimuksien mukaan. 868 MHz kaistan leveys on 9,51 MHz, mikä ei ollut ongelma, koska kaistalta käytettäisiin vain yhtä kiinteätä taajuutta. 868 MHz taajuudella toimivia keloja ja kondensaattoreita oli jo vaikeampi löytää. Näinkin korkealla taajuudella toimivien komponenttien etuna on, että ne ovat pääasiassa todella pieniä pintaliitoskomponentteja, jolloin tilaa piirilevyllä säästyy paljon. Antennisovituspiirin komponenteille ei valmistajan puolesta annettu muita vaatimuksia kuin induktanssi ja kapasitanssiarvot ja ulostulokapasitanssin Q-arvo.

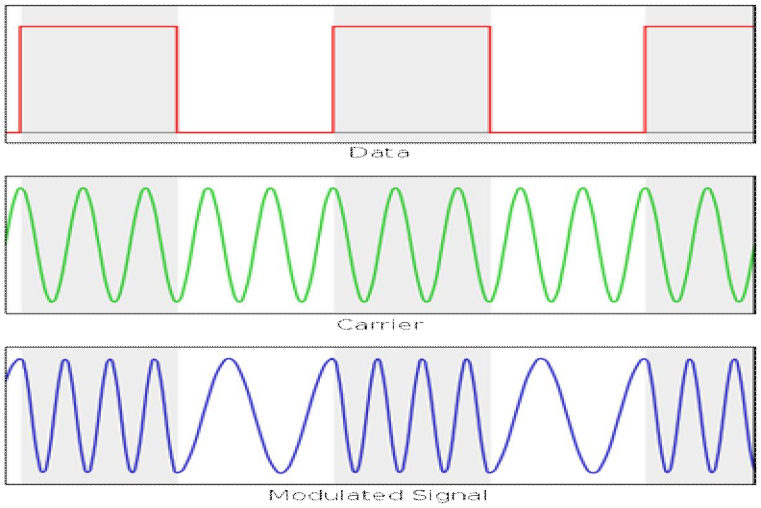
Valmistajan mukaan referenssioskillaattoriksi kelpaisi halpa tarvike- oskillaattori. Yhteensopivuutta eri valmistajien oskillaattorien kanssa oli vielä lisätty integroimalla piiriin viritettävä sisäinen kuormakapasitanssi. Kuormakapasitanssi on säädettävissä 8,5 - 16 pF 0,5 pF välein. Oskillaat- toriksi valittiinkin lopulta halpa 10 MHz:n kideoskillaattori ja antenniliittimeksi piirilevyyn juotettava BNC-liitin.

Erilliskomponenttina voitaisiin vielä mainita käytettävät neljännesaallon mittainen piiska-antenni ja puoliaaltodipoliantenni. Piiska-antenni toteutettiin liittämällä BNC-liittimeen sopivan pituinen pala johtoa. TRC-101:n pitäisi pystyä ohjaamaan puoliaaltodipolia ilman antennisovituspiiriä. Puoliaaltodipolin antennihaarat juotettiinkin kiinni antennisovituspiiriin. Koska varsinaista sovitusta ei tarvittu, jätettiin piiriin kiinni ainoastaan komponentit biasointia varten.

4 TIEDONSIIRRON PERIAATE

TRC-101-piiri käyttää tiedonsiirrosta FSK-modulaatiota. Käytännössä tämä tarkoittaa, että kantoaallon taajuutta poikkeutetaan keskitaajudesta lähetettävän bittivirran mukaan.

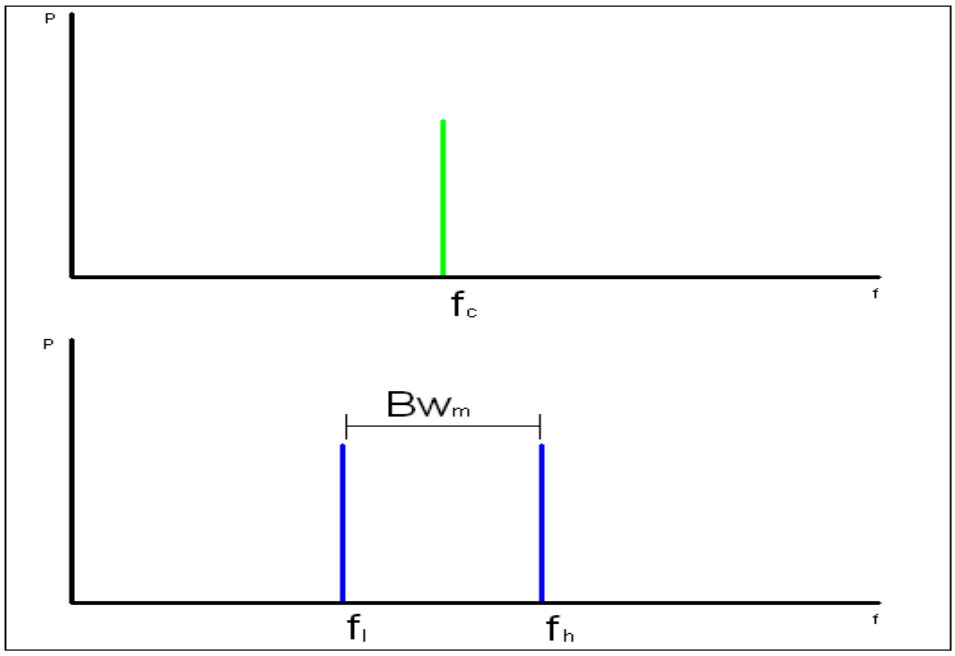
Bittivirran tilanmuutos ilmaistaan lähetyksessä kasvattamalla tai vähentämällä kantoaallon taajuutta asetusten mukaan. Voidaankin sanoa, että käytetään ylempää ja alempaa sivukaistaa.



Kuva 11 FSK-modulaatio aikatasossa. Kuvassa on punaisella värillä piirretty kantoaaltoa moduloiva sakara-aalto. Moduloitu signaali on piirretty kuvaan sinisellä värillä. Kuvan tapauksessa ykkösbitti lähetetään kantoaallon taajuutta kasvattamalla. /7./

Kuvassa 11 on esitettynä FSK-modulaatio aikatasossa. Kuvan tapauksessa on ykkös-bitin lähettämisessä käytetty ylempää sivukaistaa, ja nolla-bitin lähettämisessä alempaa sivukaistaa.

Kuvassa 12 on esitettynä kuvan 11 signaaleista kantoaallon sekä molempien sivukaistojen sijainnit lähetyksen aikana. Ideaalisessa lähetystilanteessa molempia sivukaistoja ei nähtäisi samanaikaisesti.



Kuva 1. FSK-modulaatio taajuustasossa. Ylemmässä kuvassa on vihreällä värillä piirretty moduloimaton kantoaalto. Alemmassa kuvassa on sinisellä värillä piirretty molemmat sivukaistat. Käytännössä nähtäisiin ainoastaan toinen sivukaista kerrallaan.

Jotta tiedonsiirto olisi mahdollista kahden lähetinvastaanottimen välillä, on niiden toimittava samalla taajuudella. Vastaanottimessa on kaistanpäästösuodatin, jonka leveys määrää, kuinka leveää taajuuskaistaa vastaanotin voi kuunnella. Suodattimen kaistanleveyden arvon tulisikin olla vähintään niin suuri, kuin on lähetyksessä käytettävän modulaation kaistanleveys. Jos näin ei ole, vastaanotin ei näe kuin osan lähetyksestä. Pahimmassa tapauksessa lähetystä ei nähdä ollenkaan.

Asetuksista voidaan myös valita kumpaa sivukaistaa käytetään nolla- ja ykkösbitin lähetyksessä. Tämä on myös tarkeä tieto vastaanottimen kannalta. Jos vastaanotin ei tiedä, kumpaa sivukaistaa lähetin käyttää kummankin bitin kohdalla, tulkitsee se lähetetyn tavun väärin.

Vastaanottava lähetinvastaanotin ottaa jatkuvasti näytteitä vastaanotettavasta signaalista. Jotta lähetetyt symbolit tulkittaisiin oikein, on vastaanottimen tiedettävä lähettimen symbolinopeus. Tässä tapauksessa symbolinopeus on sama asia kuin bittinopeus. Jos modulaatiopisteitä olisi käytössä esimerkiksi

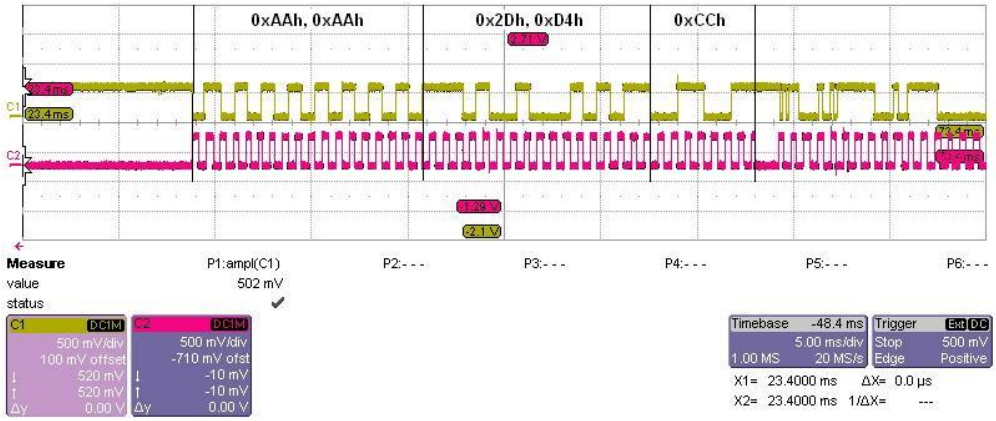
neljä, eli voitaisiin yhdellä kertaa lähettää kahden perättäisen bitin tilatiedon, olisi symbolinopeus bittinopeus jaettuna kahdella. Jos lähettimen bittinopeutta ei tunneta, ja vastaanotin ei tiedä käytettävää bittinopeutta, lähetyksestä ei saada luetuksi oikeaa sisältöä. Vastaanotin saattaa kyllä havaita ja jopa tunnistaa lähetyksen.

TRC-101:n *Data Rate Setup Register*:ssä asetetaan käytettävä bittinopeus. Rekisterissä valitaan myös, käytetäänkö esivalitsinta. Ilman esivalitsinta saavutetaan suuremmat bittinopeudet kuin esivalitsimen kanssa. Ilman esivalitsinta bittinopeus on noin kahdeksan kertaa suurempi kuin esivalitsimen kanssa.

Lähetinvastaanottimilla voidaan vastaanottaa lähetyksiä kahdella eri tavalla. Vastaanottotoiminta voidaan suorittaa kokonaan TRC-101:llä, jolloin mikrokontrollerille esitetään keskeytyspyyntö, kun vastaanotettava lähetys on tunnistettu tunnistetavuista. Jokaisen vastaanotetun lähetyksen jälkeen tulee *FIFO And Reset Mode Configuration Register*:stä nollata ja asettaa uudelleen FILLEN-bitti. Lähetys voidaan tunnistaa myös ilman tunnistetavuja mutta vastaanotin on tällöin huomattavasti herkempi häiriöille, joista seuraa huomattavasti enemmän vääriä hälytyksiä. Molemmissa edellä mainituissa vastaanottotavoissa on käytettävä digitaalista suodatinta.

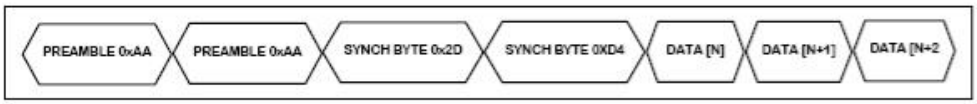
Toinen tapa vastaanottaa lähetys, on käyttää analogista suodatinta ja lukea vastaanotettu bittivirta suoraan nFSEL-portista. Mikrokontrollerilla voidaan joko lukea nFSEL-porttia jatkuvasti ja odottaa tunnistetavuja. Toinen vaihtoehto on käyttää TRC-101:n *Valid Data Detect* -toimintoa, jolloin mikrokontrollerille esitetään keskeytyspyyntö kun mahdollinen lähetys havaitaan. Lähetyksen aitouden tarkistaminen jää kuitenkin mikrokontrollerin suoritettavaksi.

Jotta mikropiiri tietäisi milloin signaalista tulisi ottaa näyte, voidaan näytteenotto tahdistaa FINT-portista saatavaan kellosignaaliin. Kun mikropiiri havaitsee postimerkkiä vastaavan bittivirran portissa, lukee se postimerkkiä seuraavat tavut muistiin. Mikropiirille ei ole asetettu varsinaista näytteenottotaajuutta. Mikropiiri odottaa FINT-portin tilanmuutoksia ja ottaa näytteen nFSEL-portin tilasta, kun FINT-portissa on nollataso. Voidaankin sanoa, että mikropiiri on asetettu liipaisemaan aina havaittuaan laskevan reunan FINT-portissa.



Kuva 2.Vastaanotettu lähetys aikatasossa oskilloskoopilla mitattuna. Keltaisella on piirretty bittivirta ja punaisella kellosignaali. Mittaus on suoritettu TRC-101-piirin nFSEL- ja FINT-porteista.

Kuvasta 13 nähdään lähetinvastaanottimen vastaanottama lähetys. Mittaus on suoritettu koulun LeCroyn valmistamalla oskilloskoopilla. Kuvasta nähdään, että vastaanottimeen asetettu bittinopeuden oletusarvo on liian suuri, koska kellosignaali ei pysy tahdissa vastaanotetun lähetyksen kanssa.



Kuva 3. Esimerkkikuva lähetettävästä paketista. /1, s. 8./

Kuvassa 14 on esitettynä miten lähetettävä datapaketti tulisi muodostaa

5 LÄHETINVASTAANOTTIMEN OHJELMOINTI

Kirjoitettavalle ohjelmalle asetettiin suunnitteluvaiheessa kolme vaatimusta. Nopea toteutettavuus, mahdollisimman pieni tilanvaraus ja toimintavarmuus. Kaksi ensiksi mainittua vaatimusta saatiin täytetyksi kiitettävästi ohjelmointikielen valinnalla.

Toimintavarmuudella tarkoitetaan ennen kaikkea, että ohjelma ei saa missään tilanteessa toimia väärin. Ohjelmaa suoritettaessa käytetään

samanaikaisesti useita mikropiirin eri toimintoja, sekä käsitellään päällekkäisiä keskeytyspyyntöjä. Tämä edellyttää ohjelman tilasta ilmoittavien lippujen hyödyntämistä ja niiden jatkuvaa tarkistusta. Tilannetta missä pysyvästi menetetään kosketus toiseen lähetinvastaanottimeen ilman laitteistovikaa, ei saa tapahtua.

Ohjelma on tehty mahdollisimman yksinkertaiseksi. Siinä ei ole muita toimintoja, kuin päätelaitteelta syötetyn tiedon lähetystä vastaanottimelle. Tai toisinpäin, vastaanotetun tiedon välitystä päätelaitteelle. Päätelaitteiden sovittaminen lähetinvastaanottimeen jätetään käyttäjän suoritettavaksi.

5.1 Lähetinvastaanottimen alustus

PC:tä tai muuta mikrokontrollerisovellusta käynnistettäessä suoritetaan aina ensimmäisenä jonkinlainen alustus. AT89S8252 mikropiirillä suoritettavan ohjelman koko saa olla korkeintaan 64 kilotavun suuruinen. Niin suuren ohjelman kirjoittaminen vaatisi ulkoisen ohjelmamuistin käyttöä, koska piirissä on vain kaksi kilotavua sisäistä ohjelmamuistia,

Ennen kuin ohjelmaa voidaan suorittaa ulkoisesta muistista, on mikropiirille kerrottava mistä ulkoisesta osoitteesta ohjelman suoritus aloitetaan, tai miten ulkoinen muistikapasiteetti on jaettu. Tätä varten mikropiirin sisäiseen ohjelmamuistiin tallennetaan alustusohjelma, niin sanottu *bootloader*. Koska lähetinvastaanottimessa on koko ohjelma kirjoitettu mikropiirin sisäiseen ohjelmamuistiin, ei varsinaista alustusohjelmaa tarvita.

Käynnistettäessä lähetinvastaanotin suoritetaan kuitenkin TRC-101:n ja mikropiirin rekisterien alustus. Mikropiiristä aktivoidaan tarvittavat keskeytyspyynnöt ja -prioriteetit, ja TRC-101:een ladataan oletusasetukset tiedonsiirtoa varten ja aktivoidaan rekisterit. Alustusten jälkeen siirrytään varsinaiseen pääohjelmaan.

5.2 Tiedonsiirto protokolla

Kaikki tieto lähetetään pakettimuodossa. Jokainen paketti koostuu viidestä tavusta, joista varsinaista dataa on yksi tavu. Lähetyksen alussa lähetetään

joka kerta tavu, jonka arvo on 0xAAh. Näin synnytetään riittävästi vaihtelua lähetyssignaaliin, jotta vastaanotin voi lukittua signaalin tahtiin.

Varsinainen paketti, josta myös vastaanotto aloitetaan, alkaa kahdella tunnistetevulla eli postimerkillä. Näiden tavujen arvo on 0x2Dh ja 0xD4h ja ne lähetetään kirjoitetussa järjestyksessä. Vastaanotin tunnistaa postimerkin ja lukee sitä seuraavat tavut muistiin. Jokainen paketti voidaan myös ohjelmoida kuitattavaksi. Kuittaus on muuten samanlainen paketti kuin muutkin, mutta datatavun arvo sovittaisiin. Paketin lähetyksen ja kuittauksen jälkeen voi kumpi vain lähetinvastaanottimista lähettää seuraavan paketin.

5.3 Ohjelmassa käytetyt funktiot

Tässä luvussa esitellään ohjelman tiedonssiirron kannalta oleellisimmat funktiot. Ohjelmointitapaan eikä funktioiden yksityiskohtaiseen tarkasteluun ei paneuduta.

Useissa funktioissa on etuliite, joka kertoo funktion tyypin. Funktion nimestä selviää funktion varsinainen toiminta. Esimerkiksi *Init*-etuliitteen omaavat funktiot ovat alustusfunktioita, ja *set*-etuliitteen omaavat funktiot ovat asetusfunktioita.

Main-funktiota suoritetaan aina kun tiedonsiirtoa ei esiinny. Funktiossa seurataan Rx_read- ja Tx_DATA-lipun tilaa. Lippujen tilan muuttuessa siirrytään suorittamaan lähetys- tai vastaanottotoimenpiteitä. Lippujen tilan muutokset suoritetaan keskeytyspyyntöfunktioissa.

Lähetinvastaanotin voidaan ohjelmoida suorittamaan muitakin toimintoja tiedonsiirron lisäksi. Main-funktioon voidaan lisätä toimenpiteitä, joita lisätoimintojen suorittamiseen vaaditaan.

Init_ -etuliitteen omaavat funktiot ovat kaikki alustusfunktioita, jotka suoritetaan ensimmäisenä virran päälle kytkemisen tai resetoinnin jälkeen. Funktioissa alustetaan mikrokontrollerin ja TRC-101:n rekisterit.

Set_ etuliitteen omaavissa funktioissa asetetaan halutut arvot TRC-101:n rekistereihin. Jokaiselle TRC-101:n rekisterille on varattu oma asetusfunktionsa.

Enable_ etuliitteen omaavat funktiot kytkevät päälle TRC-101:stä halutun ominaisuuden kuten esimerkiksi lähetysrekisterin. Jokaiselle käytettävälle ominaisuudelle on kirjoitettu oma funktionsa.

Disable_ etuliitteen omaavat funktiot kytkevät pois päältä TRC-101:stä halutun ominaisuuden. Jokaiselle käytettävälle ominaisuudelle on kirjoitettu oma funktionsa.

Tiedon lähettäminen suoritetaan Tx-funktiossa. Tx-funktioon siirrytään aina kun halutaan suorittaa lähetys. Tx-funktiossa suoritetaan ainoastaan lähetettävän tiedon kirjoittaminen TRC-101:n lähetysrekisteriin.

Rx-funktiossa suoritetaan silmukkaa, jossa seurataan FINT-portin ja Rx_Time_out-lipun tilaa. Jos Rx_Time_out-lipun tila muuttuu, lopetetaan Rx-funktion suoritus. Silmukasta siirrytään suorittamaan Read_FIFO-funktiota, jos havaitaan vastaanotettava lähetys. Read_FIFO-funktiossa luetaan TRC-101:n vastaanottorekisterin arvo mikrokontrollerin muistiin.

Rx_end funktiossa asetetaan rekistereihin R0 ja R1 oletusarvot sekä asetetaan FILLEN bitti.

5.4 Toiminta keskeytyspyyntöjen aikana

Keskeytyksen ollessa aktiivinen keskeytyspyyntöä esitettäessä, aloittaa mikropiiri ohjelman suorittamisen keskeytysvektorin osoittamasta muistipaikasta. Mikropiirin muistista on varattu jokaiselle keskeytyspyynnölle kahdeksan tavua muistia. Näiden kahdeksan tavun aikana tulee ohjelman suorittaa keskeytyspyyntörutiinit tai hypätä toiseen osoitteeseen, johon keskeytyspyyntörutiinin kirjoitusta on jatkettu.

Ohjelmassa käytetään kolmea eri keskeytyspyyntöä. Ulkoista keskeytyspyyntöä käytetään ilmoittamaan mikropiirille, kun päätelaitteella on tavu lähetettävänä. Tässä kesksytyspyyntörutiinissa asetetaan tilalippu, joka

ilmoittaa tavun olevan valmis lähetettäväksi. Lisäksi poistetaan kyseinen keskeytyspyyntö käytöstä.

SPI-väylän keskeytyspyyntö on eniten käytetty ja kooltaan suurin. Keskeytyspyyntö esitetään joka kerta, kun tavu on joko lähetetty tai vastaanotettu SPI-väylällä. SPI:n keskeytyspyyntörutiinissa käytännössä hallitaan kaikkea väylällä tapahtuvaa tiedonsiirtoa.

Taulukko 19. AT89S8252-mikrokontrollerin keskeytyspyyntövektorit. /4, s.114./ Keskeytyspyynnön jälkeen ohjelman suoritus aloitetaan kesketytyspyyntövektorin osoittamasta muistipaikasta.

Source	Vector Address
IE0 (External Interrupt 0)	0003h
TF0 (Timer 0 Overflow Flag)	000Bh
IE1 (External Interrupt 1)	0013h
TF1 (Timer 1 Overflow Flag)	001Bh
RI + TI (Serial Port Interrupt)	0023h
TF2 + EXF2 (Timer 2 Overflow Flag)	002Bh

Kolmas käytetty keskeytyspyyntö kuuluu mikropiirin toiselle kahdesta ajastimesta. Ajastinta käytetään laskemaan aikaa, jonka TRC-101 tarvitsee vaihdettaessa lähetystilasta kuunteluun ja takaisin. Ajastimella lasketaan myös aika, jonka paketin lähettänyt odottaa kuittausta vastaanottimelta. Kun kuittausaika on kulunut loppuun, esittää ajastin keskeytyspyynnön. Keskeytyspyyntörutiinissa asetetaan lippu, jota tarkistetaan Rx-funktiossa. Lipun tilan muutoksesta ohjelma tietää lopettaa kuuntelun.

6 MITTAUKSIA JA MITTAUSLAITTEITA

Suoritettujen mittauksien tarkoituksena oli tarkistaa toimisiko valmistajan datalehdessä lupaamien arvojen mukaisesti. Lähetinvastaanottimista mitattiin lähetystehon ja modulaatiokaistanleveyden tarkkuus sekä bittinopeus. Työn alussa sunniteltiin myös antennisovituspiirin impedanssin mittaamista piirianalysaattorilla. Mittauksesta kuitenkin luovuttiin, kun huomattiin lähetystehojen vastaavan riittävällä tarkkuudella valmistajan lupaamia arvoja.

Lähetystehot mitattiin Agilentin N9020A signaalianalysaattorilla. Bittinopeuden selvittäminen mittaamalla onnistui helposti 89600 Vector Signal Analyzer -ohjelmalla. Ohjelma on tehokas työkalu kun suunnitellaan ja testataan nykyisiä ja tulevaisuuden tiedonsiirtojärjestelmiä. Ohjelmalla voidaan esimerkiksi demoduloida 3G, WLAN, LTE ja WiMAX formaatteja. /5, s. 3./

Ohjelmalla voidaan nauhoittaa signaalianalysaattorilla tehtyjä mittauksia tiedostoiksi. Tiedostoon nauhoitettuja mittauksia voidaan jälkeenpäin toistaa, käsitellä ja analysoida tietokoneella ilman signaalianalysaattoria. Ohjelmalla pystytään siis suorittamaan nauhoitetulle mittaukselle kaikkia samoja toimenpiteitä, kuin jos käytössä olisi vektorisignaalianalysaattori.

> Suomennos alkuperäistekstistä: Agilentin 89600 Vector Signal Analysis -ohjelma mahdollistaa kompleksisten, aikatasossa vaihtelevien signaalien karakterisoinnin samanaikaisesti yksityiskohtaisen spektri-, modulaatio- ja aaltomuotoanalyysin kanssa. /5, s. 3./

Vektorisignaalianalysaattorin ero tavalliseen signaalianalysaattoriin on, että se nimensä mukaisesti esittää mitattavasta signaalista myös vaihetiedon. Tämän ominaisuuden ansiosta vektorisignaalianalysaattori soveltuu digitaalisesti moduloitujen signaalien mittaamiseen

Lähetyksessä käytetyn modulaation kaistanleveys mitattiin koulun Rohde & Schwarz:n valmistamalla spektrianalysaattorilla. Kaistanleveys mitattiin seitsemällä eri asetuksella. Mittaukset suoritettiin kytkemällä lähetinvastaanottimen antenniliitin 22 nF:n kytkentäkondensaattorin kautta spektrianalysaattoriin. Lähetinvastaanottimen ja spektrianalysaattorin maatasoja ei yhdistetty.

Työssä ylivoimaisesti eniten käytetty mittauslaite oli oskilloskooppi. Käytössä oli Agilentin DSO-sarjan oskilloskooppi. Oskilloskoopissa oli neljä kanavaa ja sillä voitiin mitata signaaleja, joiden taajuus on alle 1,6 GHz:ä. Työn myöhemmässä vaiheessa käytettiin myös koulun kaksikanavaisia oskilloskooppeja. Tässä vaiheessa kaikki ohjelman funktiot toimivat, eikä SPI-väylän liikennettä tarvinnut enää seurata.

Kun lähetinvastaanottimen toiminnan testaus aloitettiin ohjelmoinnin yhteydessä, oli vähintään neljän eri signaalitien tilaa pystyttävä mittaamaan samanaikaisesti. SPI-väylän liikenteen seuraamiseen tarvittiin kahta väylää. Resetointi- ja keskeytyspyyntöportin tilan seuraamiseen tarvittiin myös kaksi kanavaa.

7 MITTAUS- JA TESTAUSTULOKSET

Lähetinvastaanottimen lähetystehot saatiin mitatuksi kätevästi bittinope- usmittauksen yhteydessä.

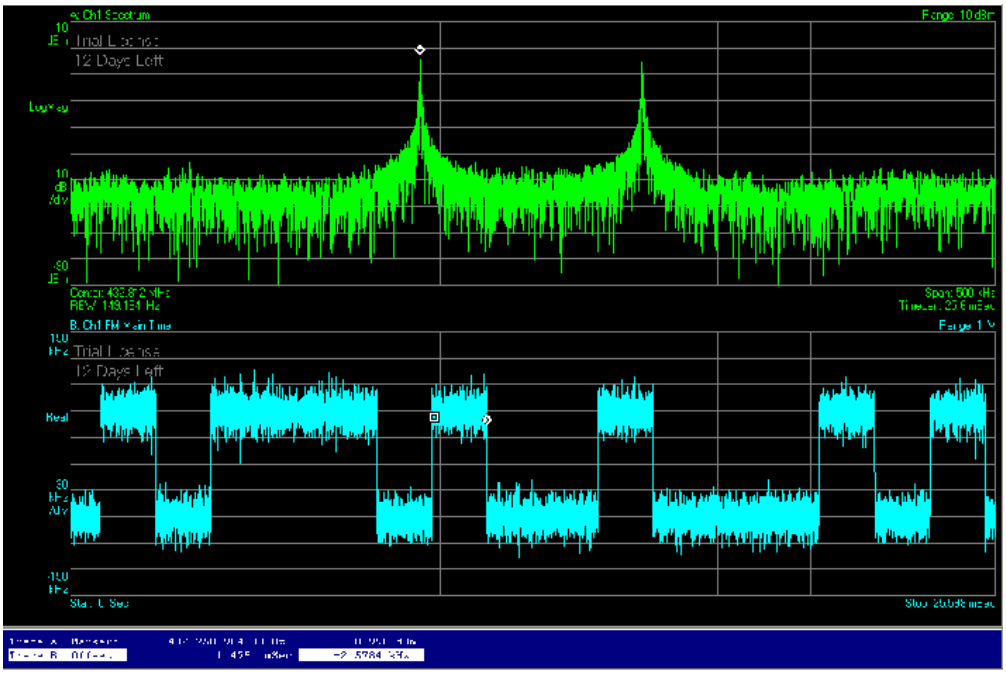
Taulukko 20. Spektrianalysaattorilla mitatut lähetystehot.

Nimellinen lähetysteho (dBm)	Mitattu lähetysteho (dBm)
0	-0,9
-3	-3,8
-6	-6,9
-9	-9,9
-12	-12,8
-15	-15,7
-18	-18,7
-21	-21,8

Vaikka signaalianalysaattoria ei olekaan varsinaisesti tarkoitettu tehon mittaamiseen, voidaan mitattujen tulosten perusteella sanoa, että lähetystehot vastaavat riittävällä tarkkuudella valmistajan lupaamia arvoja. 400 MHz:n taajuudella hyvän signaalianalysaattorin mittaukseen aiheuttama virhe on noin ±0,5 desibelin suuruinen. Lähetysteho mitattiin ainoastaan lähetinvastaanottimesta, jossa on antenniliitin.

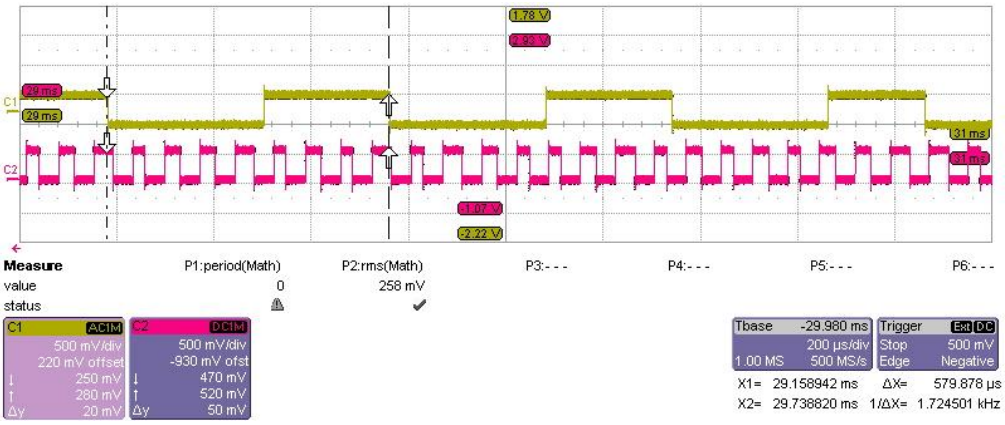
Bittinopeus saatiin selville tutkimalla 89600 Vector Signal Analyzer -ohjelmalla modulaation sivukaistoja aikatasossa. Kuvassa 30 on pysäytetty

kuva nauhoitetusta mittauksesta. Kuvassa sinisellä värillä piirretty signaali esittää modulaation taajuusvaihtelua aikatasossa. Nauhoitus on pysäytetty lähetyksen postimerkin kohdalle. Postimerkki voidaan lukea kuvasta, kun muistetaan, että ylemmällä sivukaistalla lähetettiin nolla-bitti.



Kuva 4. Pysäytetty kuva 89600 Vector Signal Analyzer -ohjelmalla nauhoitetusta mittauksesta. Kuvassa vihreällä lähetyksen spektri ja sinisellä modulaatio ajan funktiona.

Vastaanottimen näytteenottotaajuus mitattiin myös oskilloskoopilla. Oskilloskooppimittauksilla selvitettiin asetetun bittinopeuden, ja esivalitsimen käytön suhdetta todelliseen näytteenottotaajuuteen. Mittaukset tehtiin Le Croyn oskilloskoopilla.



Kuva 5. Kuvassa nähdään oskilloskooppimittaus lähetinvastaanottimien FINT-portin tahdistussignaaleista. Lähetinvastaanottimessa, jonka kuvaaja on piirretty keltaisella värillä, on esivalitsin käytössä. Toisen lähetinvastaanottimen, jossa esivalitsin ei ole käytössä, kuvaaja on piirretty punaisella.

Kuvasta 16 nähdään, että ilman esivalitsinta saavutetaan noin kahdeksan kertainen tiedonsiirtonopeus esivalitsimen kanssa tehtyyn tiedonsiirtoon verrattuna. Kuvaajat on mitattu lähetinvastaanottimien FINT-porteista.

Taulukko 21. Mittaustulokset lähetyksessä käytettävistä modulaatiokaistanleveyksistä. Mittaukset on suoritettu lähetinvastaanottimelle, jossa oli antenniliitin ja piiskaantenni. Mittaus suoritettiin 22 nF kytkentäkondensaattorin kautta. Taulukon arvoista on havaittavissa, että sivukaistat eivät ole täysin symmetrisiä keskitaajuuden suhteen.

Asetettu BW (kHz)	**Keskitaajuus (MHz)**	**Mitattu BW (KHz)**	**Alempisivukaista (MHz)**	**Poikkeama keskitaajuudesta (kHz)**
15	432,8	30,00	432,80	15,00
60	432,8	120,0	432,75	50,00
120	432,8	240,0	432,69	10,00
180	432,8	360,0	432,63	10,00
225	432,8	453,0	432,59	16,50

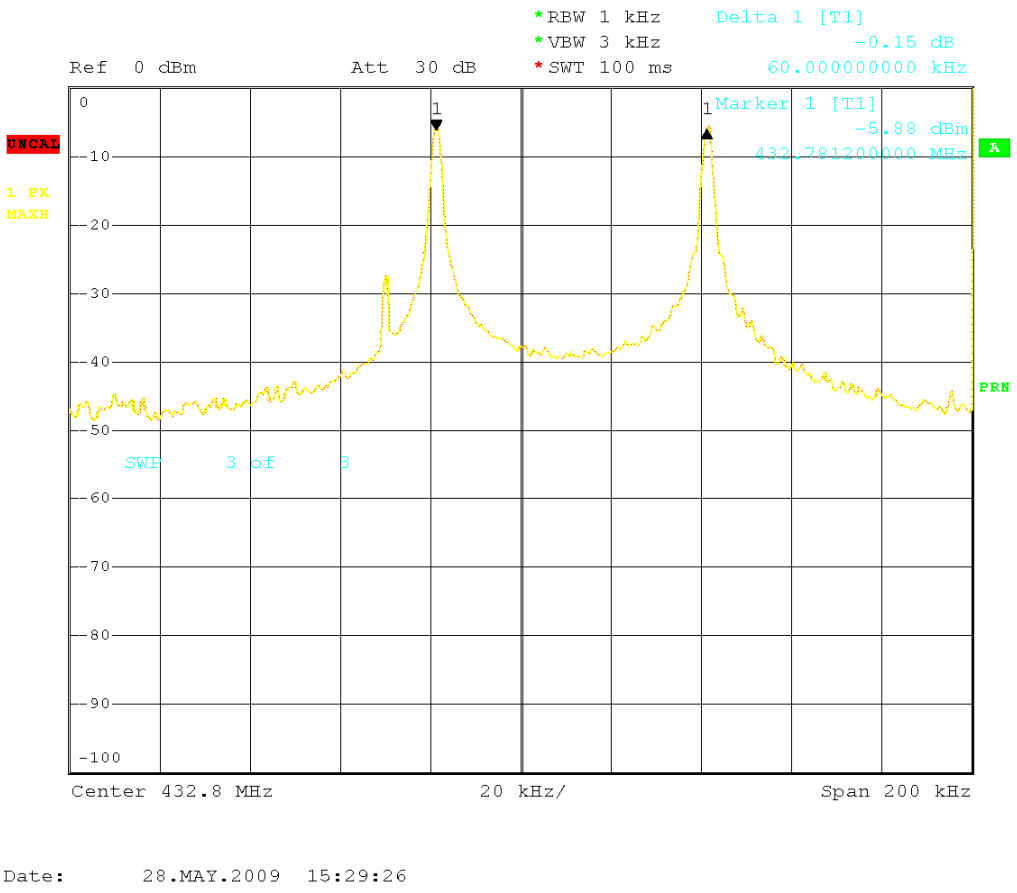
Taulukossa kaksikymmentäyksi on mittaustulokset lähetyksessä käytettävistä modulaatiokaistanleveyksistä. Mittaukset suoritettiin 22 nF kytkentäkondensaattorin kautta. Kuten tuloksista voidaan havaita, on mitattu kaistanleveys yhtä suuri kuin lähetinvastaanottimeen asetettu. Ainoana poikkeuksena 225 kHz:n kaistanleveys joka on kolme kilohertsiä suurempi kuin asetettu kaistanleveys. Tuloksista havaitaan myös, että modulaatio ei ole täysin symmetristä keskitaajuuden suhteen. Taulukon kuusi oikeanpuoleisimmasta sarakkeesta voidaan lukea modulaatiossa syntyvän poikkeaman suuruus ja suunta. Lähetinvastaanottimien kaikkia modulaatiokaistanleveyksiä ei testattu.

Taulukko 22. Mittaustulokset lähetyksessä käytetyistä modulaatiokaistanleveyksistä. Mittaukset on suoritettu lähetinvastaanottimelle, jossa käytettiin dipoliantennia. Tuloksista havaitaan, että sivukaistat eivät ole täysin symmetrisiä keskitaajuuden suhteen.

Asetettu BW (kHz)	**Keskitaajuus (MHz)**	**Mitattu BW (KHz)**	**Alempisivukaista (MHz)**	**Poikkeama keskitaajuudesta (kHz)**
15	432,8	30,00	432,80	15,00
60	432,8	120,0	432,75	50,00
120	432,8	241,0	432,69	10,00
180	432,8	362,0	432,63	10,00
225	432,8	452,0	432,58	10,00

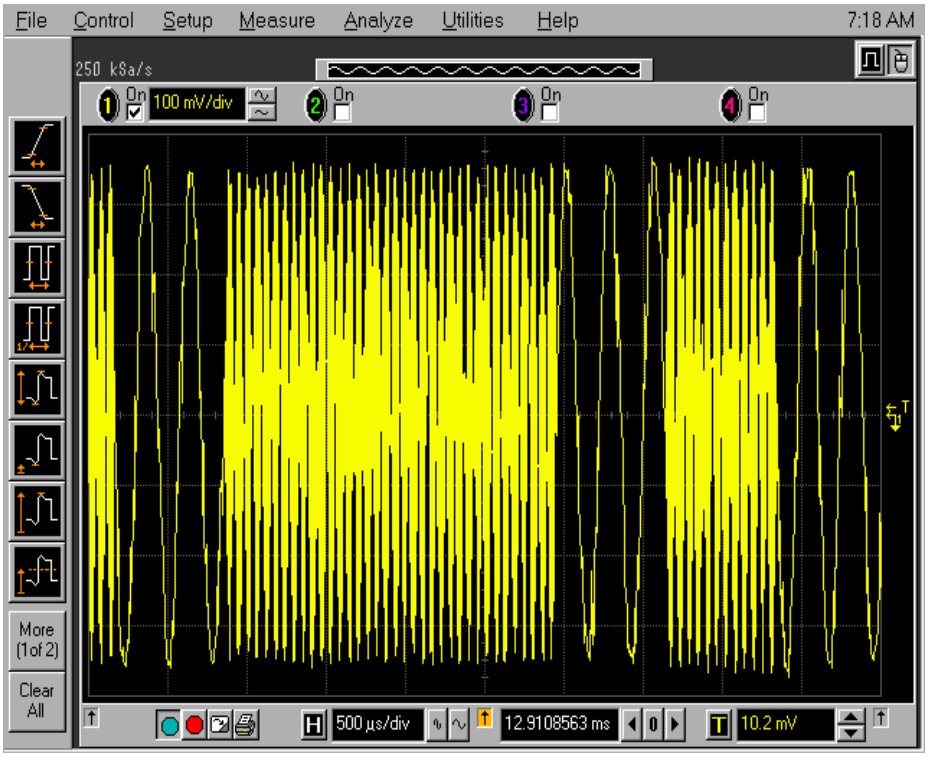
Taulukossa 22 on mittaustulokset dipoliantennia käyttävän lähetinvastaanottimen kaistanleveyksistä. Mittaus suoritettiin asettamalla spektri- analysaattorin sisääntuloon piiska-antenni. Mittaustuloksista havaitaan, että molempien lähetinvastaanottimien modulaatioissa on poikkeamaa.

Kuvasta 17 nähdään selvästi kaistanleveyden poikkeama keskitaajuudesta. Kuvan osoittimet on asetettuna sivukaistojen kohdille osoittamaan kaistanleveyttä ja alemman sivukaistan taajuutta.



Kuva 6.Spektrianalysaattorin kuva kaistanleveysmittauksesta. Kuvasta havaitaan moduloinnin epäsymmetrisyys. Mittaus on suoritettu käyttämällä spektrianalysaattorin Max hold -ominaisuutta, joka piirtää ainoastaan jokaisen mittauspisteen suurimman arvon. Alemman sivukaistan vasemmalla puolella näkyvä piikki on satunnainen ilmiö, joka on Max hold -ominaisuuden käytön takia piirretty kuvaan.

Kuvassa 18 nähdään oskilloskoopilla tehty mittaus lähetyksestä. Mittaus oli ensimmäisiä lähetinvastaanottimelle suoritettuja mittauksia. Mittauksen tarkoituksena oli havaita lähetys Tx-funktion toiminnan varmaistamiseksi. Mainittakoon, että mittauslaitetta käytettiin ainoastaan indikaattorina, jotta voitiin todeta lähetinvastaanottimen toimivan. Mittaustuloksia ei tässä vaiheessa vielä kirjattu.



Kuva 7.DSO8600-oskilloskoopilla mitattu paketin lähetys aikatasossa. Mittaus suoritettu 22 nF:n kytkentäkondensaattorin läpi.

8 POHDINTAA

Insinöörityötä aloitettaessa asetettiin kaikki tavoitteet erittäin korkealle. Näin voitiin toimia, koska työtä ei tehty minkään yrityksen toimeksiannosta vaan tekijän toivomuksesta. Tämä mahdollisti sen, että lähetinvastaanottimia pystyttäisiin kehittämään mahdollisimman paljon. Tällä tavalla myös insinöörityön tekijä saisi mahdollisimman paljon kokemusta ja oppia.

8.1 Lähtökohdat ja suunnittelu

Lähtökohtana oli, että lähetinvastaanottimet kykenisivät taajuushyppivään tiedonsiirtoon ja lähetystehon säätelyyn olosuhteiden mukaan. Ohjelman tulisi olla toimintavarma, eikä sen tulisi toimia väärin missään tilanteessa. Työn edetessä tuli eteen kuitenkin ongelmia ongelmien jälkeen. Osa ongelmista olisi todennäköisesti voitu välttää, jos olisi ollut aikaisempaa kokemusta sulautetuista järjestelmistä ja ohjelmoinnista. Työn aikana olikin siis kartutettava tietoa reilusti lisää.

Työn suunnittelu- ja toteutusvaiheiden aikana kohdattiin sekä onnistumisia että vastoinkäymisiä. Sanonta ”hyvin suunniteltu, on puoliksi tehty”, pitää hyvin paikkansa. Kun ensimmäisiä piirilevyjä suunniteltiin ja piirrettiin, tehtiin pieniä oikaisuja maadoituksissa. Osa oikaisuista tehtiin, koska suunnitteluohjelmasta ei osattu käyttää kaikkia ominaisuuksia. Osa ongelmista olisi voitu välttää paremmilla muistiinpanoilla, joita olisi voitu käyttää piirilevyn kuparointien tarkastamisessa apuna.

8.2 AT89S8252:n ja TRC-101:n välinen toiminta

Koska ohjelma kirjoitettiin Assembly-ohjelmointikielellä, piti mikropiiristä opetella, miten sen muisti on jaettu, miten sitä osoitettaisiin ja miten sitä voidaan ohjelmoijan toimesta jakaa. Jos ohjelma olisi voitu kirjoittaa muulla ohjelmointikielellä kuten C:llä, ei mikropiiristä olisi tarvinnut tietää muuta kuin muistin määrä. Keskeytyspyyntöjen, väylien ja ajastimien käyttö lisäsi jokainen osaltaan opeteltavien asioiden lukumäärää.

TRC-101-piirin ohjaamisesta oli datalehdessä hyvin selkeät ohjeet ja kuvat. Rekisterien osoitus ja niihin kirjoittaminen ja niistä lukeminen osoittautui juuri niin helpoksi kuin oli datalehdessä esitetty. Alustustoimenpiteet saatiinkin varsin vaivattomasti toteutetuksi, mutta varsinainen ohjaaminen osoittautuikin varsin hankalaksi.

TRC-101:tä voidaan pitää mikrokontrollerina, vaikka se vaatiikin mikropiirin toimiakseen. Se suorittaa jatkuvasti itsenäisesti useita toimintoja ja ilmoittaa niistä mikrokontrollerille keskeytyspyynnöin tai tilalipuin. Se voi myös suorittaa kaikki lähetys- ja vastaanottotoimenpiteet täysin itsenäisesti. Näiden ominaisuuksien ansiosta TRC-101:n ja mikropiirin yhteistoimintaa voidaan kuvata kahden mikrokontrollerin yhteistoimintana sarjaväylän välityksellä. Ennalta oli oletettavissa, että mikrokontrollereiden välistä yhteistoimintaa ei olisi helppoa toteuttaa. Tämä osoittautui todeksi insinöörityön edetessä.

TRC-101:ssä on yhdeksän porttia, joiden tilan muutoksilla se viestii mikrokontrollerin kanssa. Kolme näistä porteista kuuluu SPI-väylään. Ohjelmoinnin edistyessä aloitettiin TRC-101:n ominaisuuksien käynnistäminen ja testaaminen. Silloin huomattiin, että samanaikaisesti oli pystyttävä seuraamaan vähintään neljän portin tilaa. Samaan ongelmaan törmättiin, kun

lähetinvastaanottimista testattiin lähetyksen ja vastaanoton toimivuutta. Nopeasti selvisi, että tavallisella kaksikanavaisella oskilloskoopilla ei työtä saataisi valmiiksi seuraavankaan lukuvuoden aikana.

AT89S8252-mikrokontrollerin ominaisuudet olivat riittävät insinöörityön loppuun saattamiseksi. TRC-101:llä saavutettavat 256 kilobittiä sekunnissa olevat tiedonsiirtonopeudet ovat kuitenkin liian suuria AT89S8252:lla toteutettavaksi. Niin nopea tiedonsiirto tarkoittaa, että yksi bitti pitäisi pystyä lukemaan alle kolmessa mikrosekunnissa. AT89S8252:lla yhden käskyn suorittaminen kestää vähintään 12 mikrosekuntia. Mikrokontrolleria pitäisi siis vaihtaa.

8.3 Tiedonsiirron ongelmia ja ratkaisuja

Yleistäen voidaan sanoa, että tiedon lähettäminen on helpompaa kuin vastaanottaminen. Väite pitää paikkansa myös tässä tapauksessa. RF Monolithics julkaisi datalehden mukana hyvin toimivan antennisovituspiirin piirikaavion piiska-antennin ohjaamiseen. TRC-101 oli myös suunniteltu ohjaamaan dipoli- tai silmukka-antennia ilman sovituspiiriä, joten antennisovituksesta ei aiheutunut ongelmia. Datalehdessä oli selkeästi esitetty, missä järjestyksessä TRC-101:n lohkot tulisi aktivoida, miten lähetysrekisteriin kirjoittaminen tulisi toteuttaa ja mistä lähetettävän paketin tulisi koostua.

Lähetyksen vastaanottaminen osoittautui erittäin hankalaksi toimenpiteeksi. Työn alussa suunniteltiin, että kaikki vastaanottotoiminta jätettäisiin TRC-101:n suoritettavaksi. Suunnitelmasta kuitenkin luovuttiin, koska TRC-101:ä ei saatu tunnistamaan tunnistetavuja. Ongelman aiheuttajaksi epäiltiin aluksi virheellistä bittinopeutta lähettävässä lähetinvastaanottimessa, koska datalehden lukemisen jälkeen jäi mielikuvaksi, että asetettava bittinopeus vaikuttaa ainoastaan vastaanotossa. Tästä syystä lähetyksen bittinopeus tarkistettiin myös mittauksella. Ongelman todelliseksi aiheuttajaksi paljastui virhe datalehdessä. Datalehden aikaisemmassa painoksessa oli virheellisesti ilmoitettu, että FILT-bitti tulisi asettaa, jos digitaalinen suodatin halutaan kytkeä päälle. Uudessa painoksessa virhe oli korjattu. Tästä virheestä johtuen suoritettiin vastaanottaminen analogista suodatinta käyttäen. Vastaanotettu lähetys luettiin FINT- ja nFSEL-portista, ja tunnistetavut tunnistettiin mikrokontrollerilla. Toimintatapa oli kuitenkin erittäin häiriöaltis ja kasvatti ohjelman tilanvarausta huomattavasti.

Kun havaitut virheet oli huomioitu, voitiin ohjelma kirjoittaa uudelleen siten, että TRC-101:llä suoritetaan kaikki vastaanotossa tarvittavat toimenpiteet alkuperäisen suunnitelman mukaisesti. Ohjelmasta tuli tilanvaraukseltaan pienempi ja yksinkertaisempi. Tämä mahdollistaa myös muiden tehtävien suorittamisen tiedon lähetyksen ja vastaanoton ohessa.

8.4 Kehitys- ja parannusideoita

Mikrokontrolleriohjattulähetinvastaanotin on aiheena niin laaja, että sen eri osa-alueiden kehittämisistä voisi tehdä useita insinööritöitä. Tässä insinöörityössä kehitettiin toimiva laite, selvitettiin toimintateoriaa, ja testaamalla varmistettiin lähetinvastaanottimen toimivuus. Koska lähetinvastaanottimet rakennettiin käyttäen RFID-piiriä, voisi yhtenä kehityssuuntana pitää RFID-tarran kehittämistä.

Insinöörityössä käytettiin antenneina puoliaaltodipolia ja piiska-antennia. Molemmat toteutettiin piirilevyn ulkopuolelle. Molemmat antenniratkaisut toimivat erittäin hyvin. Olisikin mielenkiintoista verrata mittaustuloksia vastaaviin, piirilevylle rakennettuihin antenneihin. Piirilevyn materiaalin, antennijohtimen poikkipinta-alan ja antennimuodon vaikutuksista saataisiin arvokasta tietoa tuleville insinöörityön tekijöille. Mobiililaitteiden ongelmana on, ja tulee aina olemaan, että tiedonsiirtonopeus on suoraan verrannollinen antennin säteilevään pinta-alaan.

TRC-101-piirissä on ominaisuuksia, joiden ansiosta lähetinvastaanottimista voidaan kehittää erittäin älykäs tiedonsiirtojärjestelmä. Suurimpana rajoittimena on ohjauksessa käytettävä mikrokontrolleri. Mikropiirivalmistajat tuovat markkinoille jatkuvasti nopeampia, pienempiä, monipuolisempia ja pienemmän virrankulutuksen omaavia mikropiirejä. TRC-101:n maksimi tiedonsiirtonopeudet saavutettaisiin helposti mikrokontrolleria vaihtamalla. Käytettävän muistin koko todennäköisesti myös moninkertaistuisi. Tämä mahdollistaisi monipuolisemman ohjelman kehityksen, ja mahdollisesti ohjelmointikielen vaihtamisen korkeamman luokan ohjelmointikieleen. Esimerkiksi C-kielellä kirjoitettua ohjelmaa voitaisiin käyttää myös muiden mikrokontrollerien kanssa. Tarpeeksi nopealle mikrokontrollerille jäisi myös paljon hetkiä, jolloin TRC-101:tä ei tarvitsisi ohjata. Tämä mahdollistaisi mikrokontrollerin käytön myös päätelaitteen ohjaamisessa.

Koska elektroniikkaa käytetään paljon paristokäyttöisissä sovelluksissa, on käytettävien komponenttien virrankulutuksen oltava pieni. Yksi kehittelysuunta voisi olla mahdollisimman taloudellisen lähetinvastaanottimen kehittäminen. Virrankulutukseen voidaan vaikuttaa paljon käyttöjännitteen ja kellotaajuuden pienentämisellä sekä valitsemalla mikrokontrolleri, jolla on pieni virrankulutus. Edellä mainitut ovat helposti toteutettavissa. Ohjelmalta pieni virrankulutus puolestaan vaatii vähän käskyjä ja paljon torkkuja. Ohjelmointikielen ja funktioissa käytettävien käskyjen valintaa tulisi harkita.

Elektroniikkalaitteiden sähkömagneettisten häiriöiden vähentämiseksi on Euroopan unionissa säädetty normit sähkömagneettisille häiriöille. Jotta laite pääsisi Euroopan unionin jäsenmaissa markkinoille, on sen täytettävä normeissa asetetut vaatimukset. Lähetinvastaanottimien testaaminen EU:n normien mukaisesti on välttämätöntä, jos lähetinvastaanottimia kehitellään myytäväksi tuotteeksi asti. Lähetinvastaanottimien häiriönsietokyky olisi myös tarpeelllista selvittää, jos niitä suunnitellaan käytettäväksi osana suurempaa sovellusta.

9 JOHTOPÄÄTÖKSET

Lähetinvastaanottimet saatiin rakennetuksi ja toimiviksi. Toisessa lähetinvastaanottimessa käytetään sovituspiiriä ja piiska-antennia. Toiseen rakennettiin puoliaaltodipoli ilman antennisovituspiiriä.

Lähetinvastaanottimiin kirjoitetulla ohjelmalla pystytään vastaanottamaan ja lähettämään paketteja kahden lähetinvastaanottimen välillä. Lähetettävän bittivirran pituus voidaan määritellä halutun pituiseksi yksinkertaisella ohjelmamuutoksella. Kaikista lähetyksistä voidaan myös tarvittaessa lähettää sovitun merkkijonon sisältämä kuittausviesti.

Insinöörityöstä saatiin paljon kokemusta ja tietoa sulautettujen järjestelmien suunnittelusta ja toteutuksesta. Koska ohjelma kirjoitettiin Assembly-ohjelmointikielellä, piti mikropiiristä opetella keskeytysvektorien osoittamat muistiosoitteet, käyttömuistin sijainti ja muistin eri osa-alueiden osoitusmuodot ohjelmoinnin onnistumiseksi. Näin opittiin yhden piiriperheen mikropiirien rakenne, ominaisuudet ja toiminta hyvin tarkkaan.

Toteutus- ja testausvaiheiden aikana korostui erityisesti huolellisuuden ja työnteon jaksottamisen tärkeys. Kun ohjelmaa kirjoitettiin niin päädyttiin usein tilanteisiin, joissa ohjelmarivejä oli kirjoitettu useita kymmeniä ilman testausta. Tehtiin siis liian suuria kokonaisuuksia. Tästä seurasi, että testaamisen ja korjaamisen määrä kasvoi ja samalla huolellisuus kärsi. Jos työn tekovaiheita olisi pidetty hieman lyhyempinä, niin huolellisuus olisi parantunut. Myös huolellisten toimintakaavioiden tekeminen ennen työn aloittamista olisi ollut suotavaa. Ohjelman kirjoitus ja tarkistaminen olisi helpottunut, jos olisi ollut parempi suunnitelma, vuokaavio ja enemmän etukäteen laskettuja toiminta-aikoja.

Toteutus- ja testausvaiheiden aikana huomattiin myös kuinka tärkeitä kunnon työkalut ovat. Kun työ aloitettiin ei mittauslaitteisiin kiinnitetty huomiota. Koska ei ollut aikaisempaa kokemusta näin suuresta sulautetusta järjestelmästä niin oletettiin, että kaksikanavainen oskilloskooppi riittäisi. Myöskään ei ajateltu kuinka lähetys- ja vastaanotto-ominaisuuksia testattaisiin. Koska paketin lähetys kestää alle 100 ms, on sitä vaikea saada mitattua pyyhkäisevällä signaalianalysaattorilla. Lisäksi korkea toimintataajuus rajoitti mahdollisuuksia mitata lähetystä oskilloskoopilla.

Datalehden vaikean tulkittavuuden ja siinä esiintyneen pienen, mutta ongelmallisen, virheen vuoksi tehtiin paljon ylimääräistä työtä. Koska TRC-101:n asetuksista ja niiden kytkentäjärjestyksistä oli epäselvyyttä, käytettiin testaamisessa paljon aikaa väärien vaihtoehtojen eliminoimiseen. Lopulta toiminta johti suunnitelmien muuttamiseen ja epäselvien ominaisuuksien selvittämiseen mittausten avulla. Alkuperäisen suunnitelman sijasta lähetyksen vastaanotto toteutettiinkin myös improvisoiduin keinoin, joista valmistajalla ei ollut mainintaa dokumenteissa. Voisi sanoa, että työn aikana keksittiin uusi tapa vastaanottaa lähetyksiä käyttäen analogista suodatinta.

Koska datalehden tulkitsemisessa ilmeni ongelmia, olisi bittinopeuteen liittyvistä ongelmista selvitty nopeasti mittauksien avulla. Näin olisi säästetty huomattavasti aikaa, ja saatu mittauslaitealan kokemuksella paikattua kokemuksen puutetta suunnittelussa.

Insinöörityöstä saatu kokemus ja opit olivat suurenmoisia. Työn aikana esiintyi paljon ongelmia, joista opittiin ja joita osataan tulevaisuudessa välttää. Lisäksi syntyi dokumentti, josta on varmasti hyötyä tuleville vuosikurssilaisille.

LÄHTEET

/1/ RF Monolithics. TRC-101:n datalehti /verkkodokumentti/ viitattu 10.3.2009/. Saatavissa : http://www.rfm.com/products/data/trc101.pdf

/2/ Atmel. AT89S8252:n datalehti /verkkodokumentti/ viitattu 10.3.2009/. Saatavissa: http://www.atmel.com/dyn/resources/prod_documents/doc0401.pdf

/3/ Viestintävirasto. Määräys luvasta vapaiden radiolähettimien yhteistaajuuksista ja käytöstä /verkkodokumentti/ viitattu 20.4.2009/. Saatavissa: http://www.ficora.fi/attachments/suomiry/5DfHIU2rt/viestintavirasto15Y2008M.pdf

/4/ Atmel. AT89S8252:n Hardware Manual /verkkodokumentti/ viitattu 22.5.2009/. Saatavissa: http://atmel.com/dyn/resources/prod_documents/doc4316.pdf

/5/ Agilent Technologies. 89600 Vector Signal Analyzer -ohjelman datalehti/ viitattu 26.5.2009/. Saatavissa: http://cp.literature.agilent.com/litweb/pdf/5989-1679EN.pdf

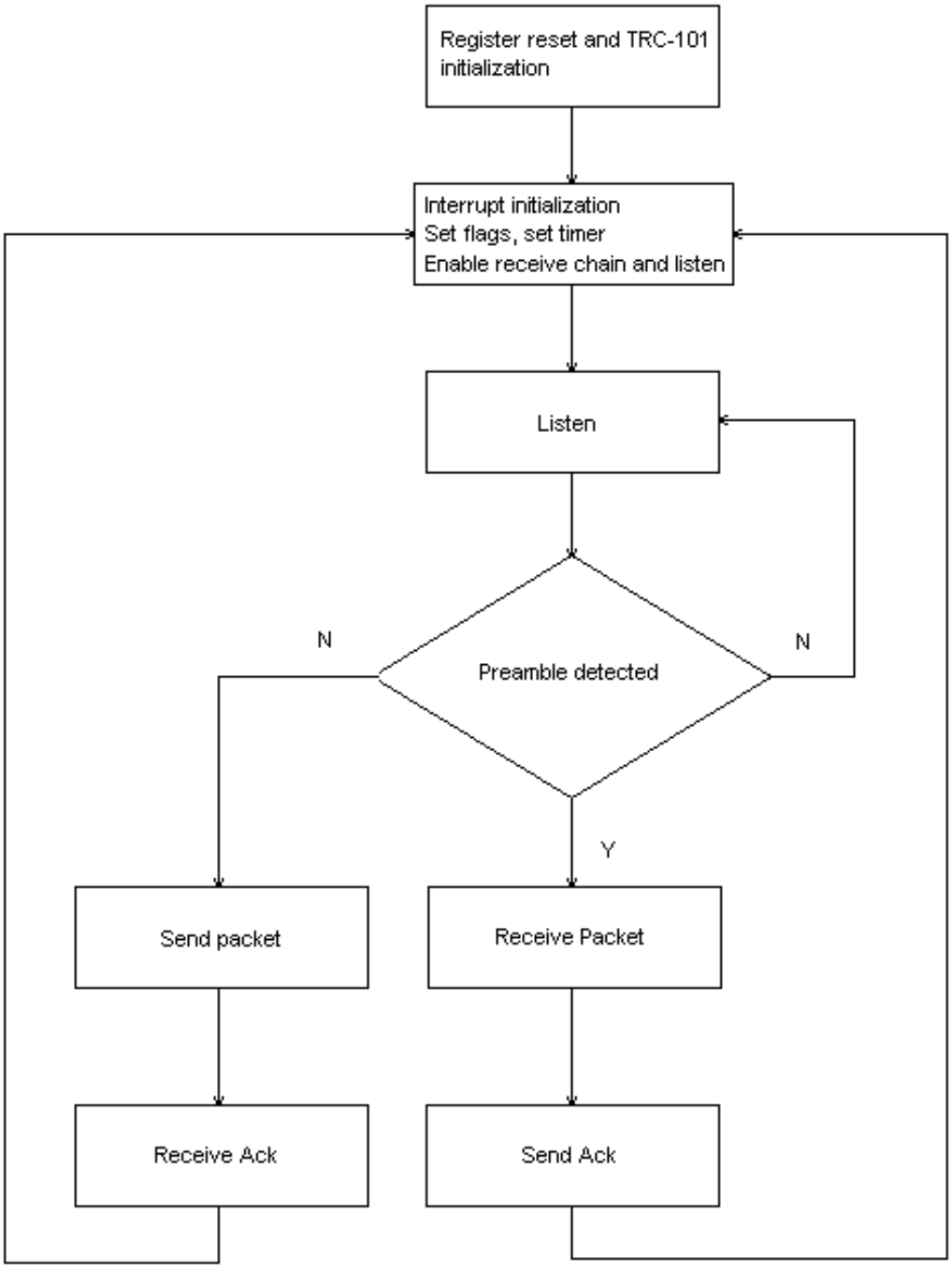
/6/ Wikipedia *The Free Encyclopedia.* /Verkkodokumentti/ viitattu 20.3.2009/. Saatavissa: http://en.wikipedia.org/wiki/Phase-locked_loop

/7/ Wikipedia *The Free Encyclopedia.* /Verkkodokumentti/ viitattu 21.3.2009/. Saatavissa: http://en.wikipedia.org/wiki/Frequency-shift_keying

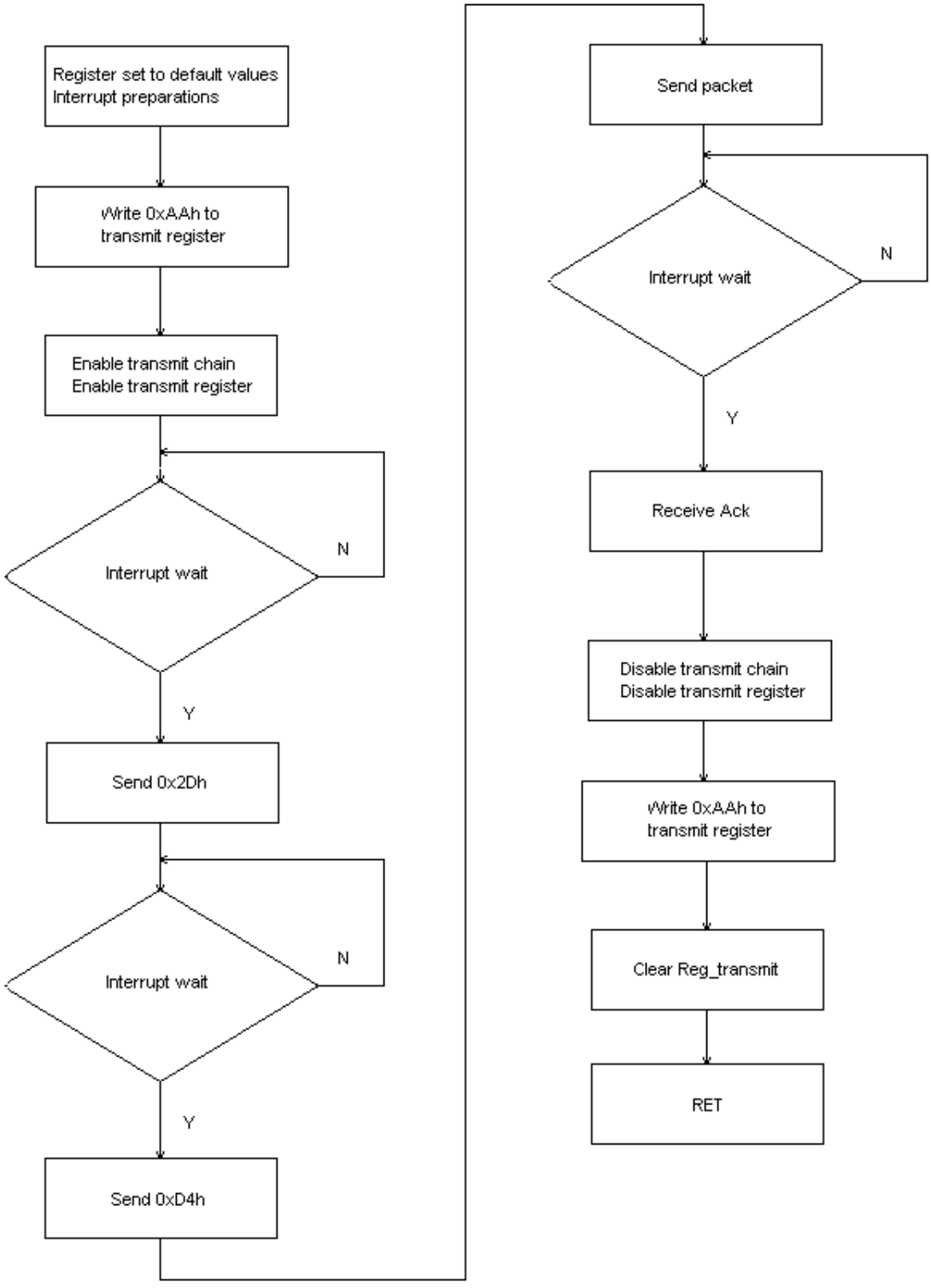
/8/ J. Glenn Brookshear, *Computer Science an overview.* Seventh edition. Boston, San Franscisco, New York, Toronto, Sydney, Tokyo, Singapore, Madrid, Mexico City, Munich, Paris, Cape Town, Hong Kong, Montreal. Addison Wesley. 2003.

/9/ Matthew M. Radmanesh, *Radio Frequency and Microwave Electronics.* London, Sydney, Toronto, Mexico, New Delhi, Tokyo, Rio De Janeiro. Prentice-Hall. 2001.

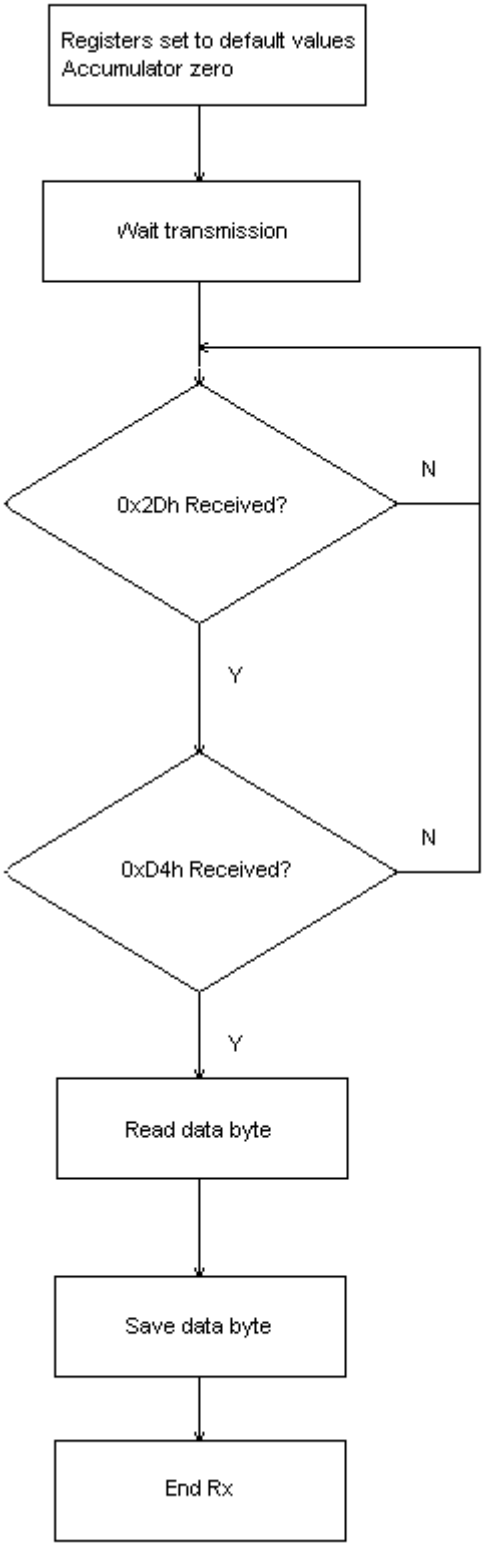
OHJELMAN VUOKAAVIO



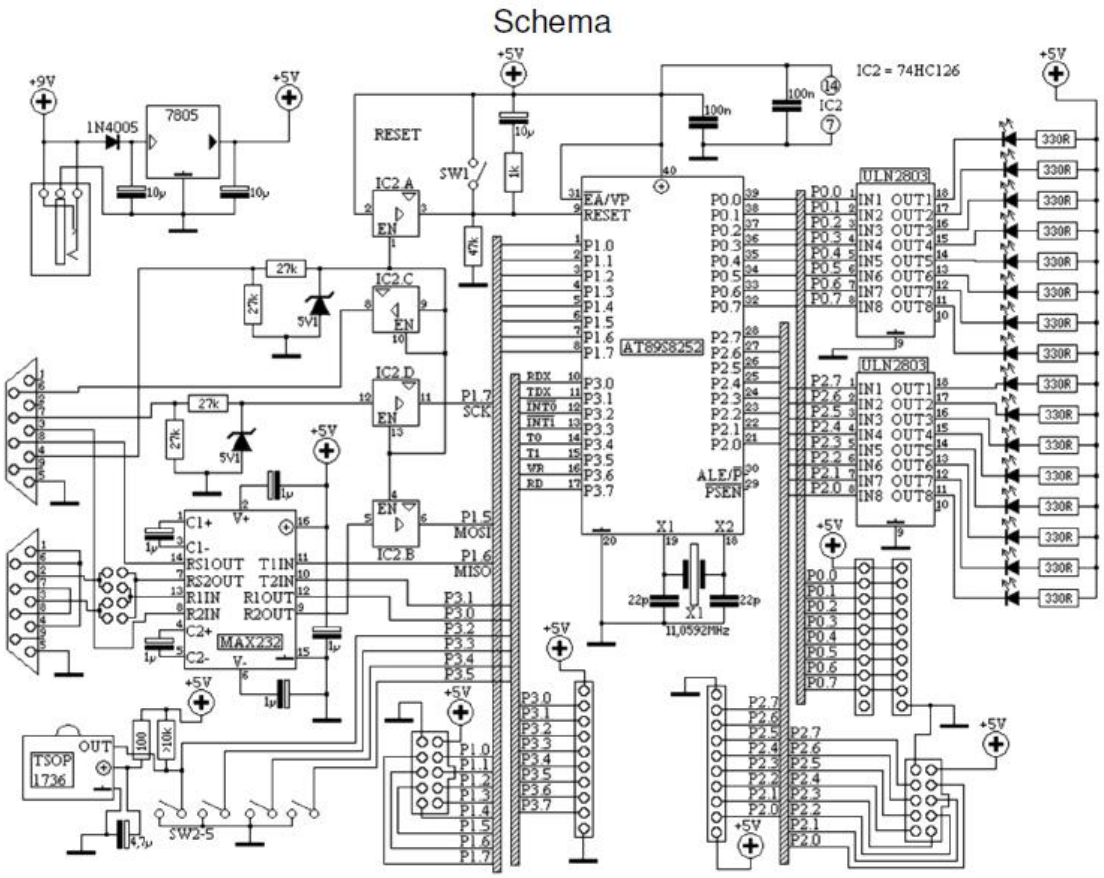
TX-FUNKTION VUOKAAVIO



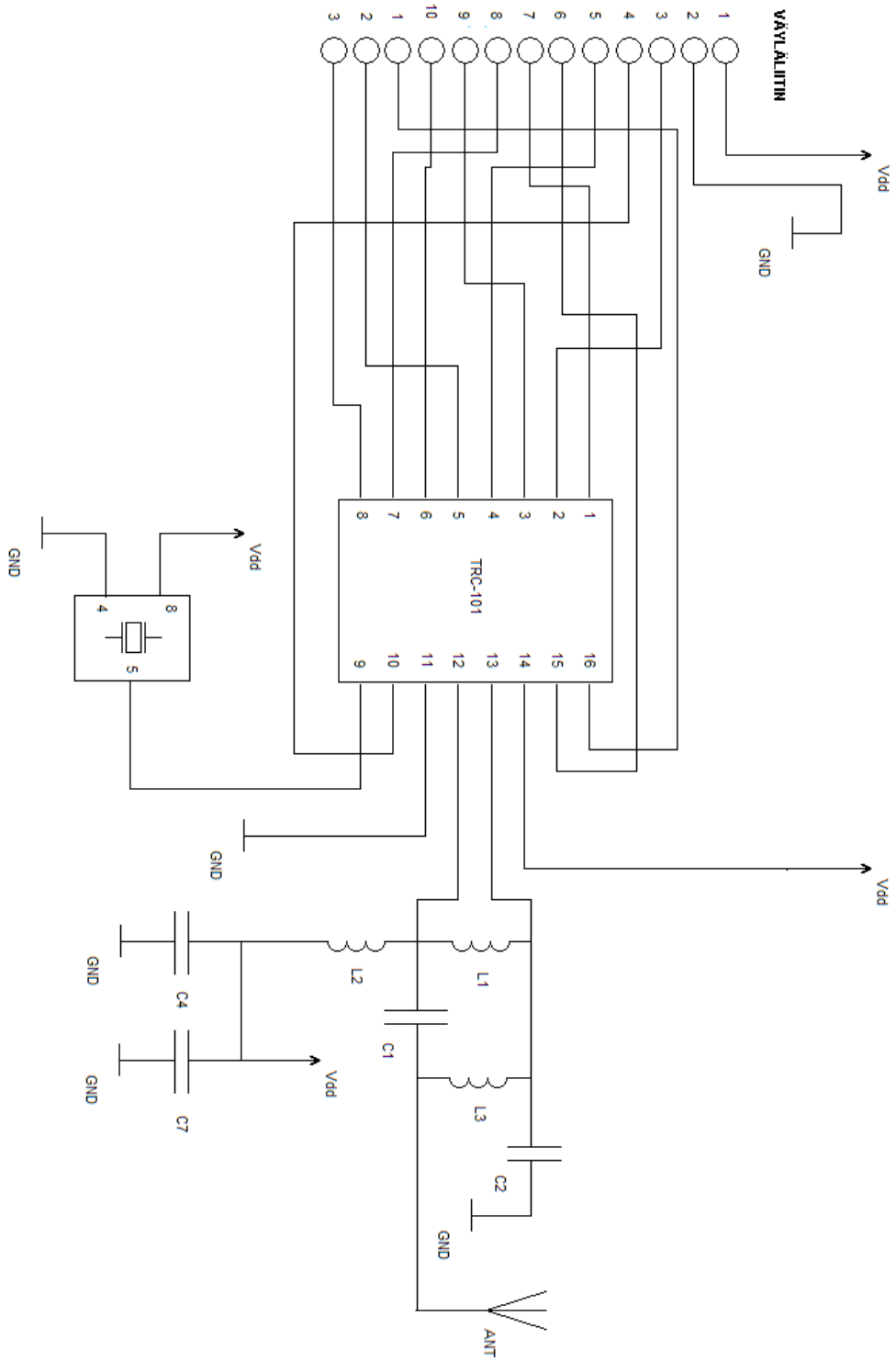
RX-FUNKTION VUOKAAVIO



AT89S8252-MIKROKONTROLLERIOHJAIMEN KAAVIOKUVA



TRC-101:N PIIRIKAAVIO



MIKROKONTROLLERIN OHJELMA

```
$NOMOD51
$INCLUDE   (AT898252.h)

;-----------------------------------------------
; Definitions
;-----------------------------------------------

; General status mask register definitions
transmit_completed  BIT Reg_status.0
stop                                    BIT Reg_status.1
read                                    BIT Reg_status.2
preamb                                  BIT Reg_status.3
transmit                                BIT Reg_status.4
command                                 BIT Reg_status.5
Exit_Rx                                 BIT Reg_status.6
TRC_Int                                 BIT Reg_status.7

; Receive option register definitions
FIFO_read                               BIT Reg_receive.0
RX_options_Tx                           BIT Reg_receive.1
RX_read                                 BIT Reg_receive.2
RX_Ack                                  BIT Reg_receive.3
RX_no_Ack                               BIT Reg_receive.4
;RX_no_comm                             BIT Reg_receive.5
RX_init                                 BIT Reg_receive.6
timer0_flag                             BIT Reg_receive.7

; Transmit option register
Tx_continue                             BIT Reg_transmit.0
Tx_DATA_ready                           BIT Reg_transmit.1
Tx_Ack                                  BIT Reg_transmit.2
Tx_skip                                 BIT Reg_transmit.3
Tx_preamb                               BIT Reg_transmit.4
Tx_Data                                 BIT Reg_transmit.5
Tx_f                                    BIT Reg_transmit.6
Tx_options_Rx                           BIT Reg_transmit.7

; User defined registers
Reg_status                      DATA 20H   ; General status register
Reg_transmit                    DATA 21H   ; Transmit status register
Reg_receive                     DATA 22H   ; Receive status register
Status_reg_low                  DATA 23H   ; Status register low byte
Status_reg_hi                   DATA 24H   ; Status register high byte
transmit_byte                   DATA 25H   ; This byte is transmitted
receive_byte                    DATA 26H   ; Received data byte
Cont_opt                        DATA 27H   ; Data transfer cont. options
;Indirect_Add                   DATA 28H   ; Help byte used during rx
;Preamb1                        DATA 29H   ;
serial_data                     DATA 30H   ; Byte transferred via SPI

; Port definitions
nRESET                EQU       P1.0       ; No use
;nIRQ                 EQU       P1.1       ; No use
FINT                  EQU       P1.2       ; Recovered clock
nFSEL                 EQU       P1.3       ; Data out
nCS                   EQU       P1.4       ; Chip select
DDET                  EQU       P3.2       ; Valid data detect port
```

```
DATA_wait  EQU        P3.4       ;

/* ---------------------------------------- */
/* Program                                  */
/* ---------------------------------------- */


org 000h
JMP Rst

EXT_Int_0:
; External interrupt 0 vector address

org 003h
           SETB RX_read                    ; Data ready flag set
           CLR EX0                         ; Disable ext. int. 0
RETI

Timer_0_int:
; Timer 0 interrupt vector address

org 00Bh
           JMP 046h
;          ORL TCON, #040h
;          XRL TCON, #010h
           RETI

EXT_Int_1:
; External interrupt 1 vector address

org 013h
           ;          CLR TR1
           JMP 02Ch

Timer_1_int:
; Timer 1 interrupt vector address

org 01Bh
           SETB Rx_no_Ack                  ; Set flag for timer 1
                                           ; overflow
           CLR TR1                         ; Stop timer 1
           CLR ET1                         ; Disable timer 1 interrupt
           RETI

SPI_Int:
; Serial interrupt vector address

org 023H
JMP 26Eh
RETI

; External interrupt 1 routines

org 02Ch
           SETB Tx_DATA_ready
           CLR EX1
           CLR Tx_Ack
;          CALL delay
;          CALL delay
;          CALL delay
```

```
           RETI

DATA_ready:
;          MOV R0, P2            ; Requires bus reading operations
           SETB Tx_DATA
           SETB Tx_DATA_ready
           RETI

; Timer 0 interrupt routines

org 046h
;          Timer 0 function
           CLR TR0
           CLR ET0
           SETB Exit_Rx
           RETI

Timer_set:
;          Timer register presets
;          Sleep time 0,1s
;          wakeup time 0,01s

           ; Timers 16-bit timers
           MOV TMOD, #011h
           ; Timer0 set
           MOV TH0,  #000h
           MOV TL0,  #000h
           ; Timer1 set
           MOV TH1,  #000h
           MOV TL1,  #000h
           RET

; ------------------------------------------------------'
; ------------------------------------------------------'

Rst:

; Reset-function.
; All register are set to default values

           MOV TMOD, #011h
           MOV Receive_byte, #000h
           MOV Reg_status, #000h
           MOV Reg_transmit, #000h
           MOV Reg_receive, #000h
           MOV transmit_byte, #0AAh
           MOV P2, #000h

           CALL Init

Start:
; Start Function

           CLR Tx_DATA_ready
           SETB Tx_Ack

           CALL Init_interrupts  ; Write default values to interrupt
                                 ; register
           CALL Timer_set        ; Set timers registers to zero
;          ORL TCON, #010h       ; Starts sleep time timer
;CALL Tx
```

```
            SETB P2.0               ; Indicator

            MOV serial_data, #081h ; Receiver chain enable command
            CALL PMR_set           ; Receiver chain enable

            JMP Rx

            JMP Start
RET

delay:
; General delay function used during testing.

            SETB TR1
            JNB TF1, $
            CLR TR1
            CLR TF1
            SETB TR1
            JNB TF1, $
            CLR TR1
            CLR TF1
            MOV TH0,  #000h
            MOV TH1,  #000h
RET

Rx:

; Funktion for reading data from
; comparator

            MOV R0, #007h          ; Local state counter for
                                   ; reading packet data
            MOV R1, #026h          ; Register holds the address of
                                   ; receive_byte-register
            MOV R2, #00Eh          ; Local state counter for
                                   ; reading preamble
;           MOV IE, #000h          ; All requests disable
            MOV A,     #000h       ; Accumulator zero

;           SETB TR0               ; Timer 0 start

            first_zero_check:
            ; Funktions for checking the preceeding
            ; zeroes in 0x2D

                        JB Tx_DATA_ready, End_Rx ; Check Timer 0 status
                                                 ; flag
                        JB Rx_no_Ack, End_Rx     ; Exit Rx if no Ack
                        JB FINT, $               ; Wait for falling edge
                        JNB FINT, $              ; Wait for rising edge
                        JNB nFSEL, first_zero_check      ;

            second_zero_check:
            ; Funktions for checking the preceeding
            ; zeroes in 0x2D

                        JB Tx_DATA_ready, End_Rx ; Check Timer 0
                        JB Rx_no_Ack, End_Rx     ; Exit Rx if no Ack
                        JB FINT, $               ; Wait for falling edge
                        JNB FINT, $              ; Wait for rising edge
                        JNB nFSEL, first_zero_check
```

```
third_zero_check:
; Funktions for checking the preceeding
; zeroes in 0x2D

          JB Tx_DATA_ready, End_Rx ; Check Timer 0
JB Rx_no_Ack, End_Rx               ; Exit Rx if no Ack
          JB FINT, $               ; Wait for falling edge
          JNB FINT, $              ; Wait for rising edge
          JNB nFSEL, first_zero_check

Preamble_0x2D_read:
; Preamble check

          DEC R2
          RL A                     ; Rotate A left for next
bit
          JB FINT, $               ; Wait for falling edge
          JNB FINT, $              ; Wait for rising edge

          JNB nFSEL, Increment_A
          CJNE R2, #008h, Preamble_0x2D_read
          CJNE A, #02Dh, Rx
          CLR EX1                  ; Timer 0 stop
          MOV A, #000h
          JMP Preamble_0xD4_read

Increment_A:
; Preamble check
; Increment Accumulator

          INC A                    ; Increment Accumulator
          CJNE R2, #008h, Preamble_0x2D_read
          CJNE A, #02Dh, Rx
          CLR EX1                  ; Timer 0 stop
          MOV A, #000h             ; Set A to zero value

Preamble_0xD4_read:
; Preamble check

          DEC R2
          RL A                ; Rotate A left for next bit
          JB FINT, $          ; Wait for falling edge
          JNB FINT, $         ; Wait for rising edge

          JNB nFSEL, Increment_A2
          CJNE R2, #000h, Preamble_0xD4_read
          CJNE A, #0D4h, Rx
          MOV A, #000h        ; Set A to zero value
          JMP Begin_read

Increment_A2:
; Preamble check
; Increment Accumulator

          INC A               ; Increment Accumulator
          CJNE R2, #000h, Preamble_0xD4_read
          CJNE A, #0D4h, Rx
          MOV A, #000h
          JMP Begin_read
```

```
End_Rx:
; Funktion for post receive operations

;          CLR P2.4                        ; Indicator to help testing
           MOV IE, #090h
           MOV IP, #010h
;          CLR TF0
           CLR P2.0
           CALL Disable_Comm_chains        ; Comm. chains disable
           CALL Status_read                ; Check status check

           JB Tx_Ack, Send_Ack             ; Send acknowledgement
                                           ; if Tx_Ack-flag is set
           JB Tx_DATA_ready, Send_DATA     ; Send packet if
Tx_DATA_ready-
                                           ; flag is set
           JMP Start
RET

           Begin_read:
           ; Packet data read

                     JB FINT, $            ; Wait for falling edge
                     JNB FINT, $           ; Wait for rising edge

                     JNB nFSEL, Increment_A3
                     RL A                  ; Rotate A left for next bit
                     DJNZ R0, Begin_read

                     JB FINT, $            ; Wait for falling edge
                     JNB FINT, $           ; Wait for rising edge
                     JNB nFSEL, Increment_A4

                     JMP Save_data

           Increment_A3:
           ; Packet data read
           ; Increment Accumulator

                     INC A                 ; Increment Accumulator
                     RL A                  ; Rotate A left for next bit
                     DJNZ R0, Begin_read

                     JB FINT, $            ; Wait for falling edge
                     JNB FINT, $           ; Wait for rising edge
                     JNB nFSEL, Increment_A4

                     JMP Save_data

           Increment_A4:
           ; Packet data read
           ; Increment Accumulator

                     INC A                 ; Increment Accumulator

           Save_data:
           ; Funktion for saving the packet data byte
           ; read from TRC-101
/* ------------------------------------------------------ */
/*          Needs operations for saving data
                                          */
```

```
/*          more dynamically
                                                                  */
/* ------------------------------------------------------------ */

;                     MOV R0, #007h         ; Default value for read loop
                      MOV @R1, A            ; Write Accumulator to ad-
dress found in R1
;                     INC  R1               ; Increment R1
                      MOV A, #000h          ; Set A to zero value
;                     CJNE R1, #028h, Begin_read

                      MOV P0, receive_byte
                      JMP End_Rx

Send_Ack:
; Funktion to prepare program to
; send acknowledgement.

           SETB P2.2                        ; Indicator to help testing
           MOV TH0,  #0FEh                  ; Load timer 0 registers
           MOV TL0,  #060h                  ; for >450us delay
           CLR Tx_Ack                       ; Clear flag

           SETB TR0                         ; Timer 0 stert

           MOV transmit_byte, #0CCh         ; Acknowledge byte

           JNB TF0, $                       ; Wait timer 0 overflow
           CLR TF0                          ; Clear flags
           CLR TR0                          ; Clear flags

           CALL Tx                          ; Send acknowledgement
           CLR P2.2                         ; Indicator to help testing
           JMP Start
RET

Send_DATA:
; Function to prepare program to
; send packet.

           MOV TH0,  #0FEh                  ; Load timer 0 registers
           MOV TL0,  #060h                  ; for >450us delay

           SETB TR0                         ; Timer 0 start

           MOV transmit_byte, #0A5h         ; Data packet
/* ------------------------------------------------------------ */
/*          Needs operations for reading data
                                                                  */
/*          dynamically from memory
                                                                  */
/* ------------------------------------------------------------ */

;          MOV serial_data, transmit_byte
;          INC transmit_byte

           JNB TF0, $                       ; Wait for timer 0 overflow
           CLR TF0                          ; Clear flags
           CLR TR0                          ; Clear flags

           CLR Tx_Data_ready                ; Clear flags
```

```
          CALL Tx                          ; Send data
          CALL Receive_Ack                 ; Receive Ack
          JMP Start
RET

Receive_Ack:
; Funktion to prepare program to
; receive acknowledgement

          SETB P2.1                        ; Indicator to help testing
          SETB ET1                         ; Timer 1 interrupt enable
          MOV TH0,  #0FEh                  ; Load timer 0 registers
          MOV TL0,  #060h                  ; for >450us delay

          SETB TR0                         ; Timer 0 start

          JNB TF0, $                       ; Wait for timer 0 overflow
          CLR TF0                                    ; Clear flags
          CLR TR0                                    ; Clear flags

          MOV serial_data, #081h           ; Receiver chain enable
          CALL PMR_set                     ; Receiver chain enable

          CALL Timer_set                   ; Set timers registers to
zero
          SETB TR1                         ; Start timer 1

          CALL Rx                          ; Receive acknowledgement
          CLR P2.1                         ; Indicator to help testing
          JMP Start
RET

Disable_Comm_chains:
;         Function disables all active comm. chains.
          MOV serial_data, #001
          CALL PMR_set
          RET

Status_read:
; Function reads TRC-101 Status Register

          CLR transmit_completed
          CLR nCS
          MOV R3, Reg_status
          MOV Reg_status, #004h
          SETB read
          MOV SPDR, #000h
          JNB transmit_completed, $
          SETB nCS
          MOV Reg_status, R3
          RET

TX:
          ; General transmit funktion
          MOV Reg_status, #000h
          CLR transmit_completed
          CPL P2.4
          MOV IP, #000h         ; Disable int priorities priority
          MOV IE, #090h         ; SPI int enabled, Ext int 1 enabled

; Writing 0AAh to transmit register. Preamble for next transmission.
```

```
            CLR nCS
            MOV SPDR, #0B8h
            SETB command
            MOV serial_data, #0AAh
            JNB transmit_completed, $
            CLR transmit_completed
            SETB nCS

            MOV serial_data, #021h          ; Transmit chain enable
            CALL PMR_set                    ; Transmit chain enable

            CLR nCS
            MOV serial_data, #09Fh
            CALL CR_set
            SETB nCS
            JB P3.3, $                      ; Interrupt wait
            CLR nCS
            SETB Tx_preamb

            MOV SPDR, #0B8h                 ; Command code

            MOV serial_data, #02Dh          ; Preamble
            JNB transmit_completed, $
            CLR transmit_completed

            MOV serial_data, #0D4h          ; Preamble
            JNB transmit_completed, $
            CLR transmit_completed

    MOV serial_data, transmit_byte
            JNB transmit_completed, $
            CLR transmit_completed

            MOV serial_data, Cont_opt       ; Continue options sent
            JNB transmit_completed, $
            CLR transmit_completed

            SETB stop

            JNB transmit_completed, $
            CLR transmit_completed

            JB P3.3, $                      ; Interrupt wait
            SETB nFSEL
            SETB nCS
            MOV Reg_status, #000h
            CALL Disable_Comm_chains        ; Transmit chain disable

            MOV serial_data, #01Fh
            CALL CR_set
            SETB nCS

; Writing 0AAh to transmit register. Preamble for next transmission.

            CLR nCS
            MOV SPDR, #0B8h
            SETB command
            MOV serial_data, #0AAh
            JNB transmit_completed, $
            CLR transmit_completed
```

```
            SETB nCS

            MOV Reg_status, #000h
;           CLR P2.4                            ; Indicator to help testing
;           CLR Tx_Ack

;           INC transmit_byte
RET

SPI_interrupt_routines:
org 26Eh
            MOV R7,SPSR
            MOV ACC,R7
            JNB ACC.7,exit
            JB transmit, break3
            JB TRC_int, exit
            JB command, break1
            JB Tx_preamb, skip2
            JB FIFO_read, break5
            JB read, break4
skip2:
            SETB transmit_completed
            CLR Tx_preamb
            SETB transmit
;           JB P3.3, $
            MOV SPDR, serial_data
            RETI
out:
            JB FIFO_read, break5
            JB preamb, break3
            JB transmit, break6
            JB command, break1
            JB read, break4

            break1:
            JB stop, exit
            SETB stop
            MOV SPDR, serial_data
;           INC P2

            RETI

            break3:                             ; Transmit
            SETB transmit_completed
            JB P3.3, $
            JB stop, exit
            MOV SPDR, serial_data
            RETI

            break4:
            JB stop, return
            MOV Status_reg_hi, SPDR
            SETB stop
            MOV SPDR, #000h
            RETI

            break5:
            SETB nFSEL
            SETB transmit_completed
            MOV serial_data, SPDR
            RETI
```

```
            break6:
            JB stop, exit
;           JNB P3.3, $
            CPL transmit_completed
            MOV SPDR, serial_data
            RETI

            return:
            SETB transmit_completed
            CLR stop
            CLR read
            MOV Status_reg_low, SPDR
            RETI

            exit:
            SETB transmit_completed
            MOV R4, SPDR
            SETB nCS
;           SETB P2.4
            CLR stop
;           CLR TR0
            RETI

org 2F0h
Init:
; TRC-101 registers initialization function

            CALL Init_SPI                                          ;
            Interrupt and SPI initialization

            MOV serial_data, #02Fh
            CALL CR_set
            ;          Control Register
            CALL AFAR_set ;Automatic Frequency Adjust Register
            CALL TCR_set  ;       Transmit Configuration Register
            CALL FSR_set  ;       Frequency Setting Register
            CALL RCR_set  ;       Receiver Control Register
            CALL BFR_set  ;       Baseband filter register

            MOV serial_data, #085H
            CALL FRMCR_set ; FIFO and RESET mode configuration register
;           CALL WTPR_set  ; Wake-up Timer Period Register
            CALL DRSR_set  ; Data Rate Setup Register
            MOV serial_data, #001h
            CALL PMR_set   ; Power Management Register
;           CALL DCSR_set  ; Duty Cycle Set Register
            JNB nRESET, $  ; Wait nRESET port to rise
            CALL Status_read ; Read cause of interrupt And to clear nIRQ

            JB P2.0, next
            SETB P2.0
;           JMP Init

next:
            NOP
            CLR P2.0
            RET

RET
```

```
;org 332h
Init_SPI:
;          Initializes SPI-bus for operation with TRC-101

;          f_SCK = f/16
;          CPHA = 0 -> Clock phase
;          CPOL = 0 -> CLock polarity
;          MSTR = 1 -> Mastermode
;          DORD = 0 -> MSB-first
;          SPIE = 1 -> enable SPI-interrupts
;          SPE  = 1 -> SPI-enable


           MOV SPCR,#010h        ; /* Master mode */
           ORL SPCR,#002h        ; /* Fclk Periph/128 */
           ANL SPCR,#0F7h        ; /* CPOL=0; transmit mode example */
           ORL SPCR,#080h
           ORL SPCR,#040h        ; /* run spi */
           SETB ES
           ; /* enable spi interrupt */
           SETB EA
RET

Init_Interrupts:
;          SETB EX0              ; /* External interrupt 0 enabled
           SETB EX1              ; /* External interrupt 1 enabled
           SETB PS               ; /* SPI interrupt priority flag
;          SETB IE.1             ; /* Timer 0 interrupt enable
;          SETB IE.3             ; /* Timer 1 interrupt enable
RET


CR_set:
;          Configuration Register
;                     - (Bit 7) Internal TX register enable
;                     - (Bit 6) FIFO enable
;                     - (Bit 5-4) Band select
;                     - (Bit 3-0) Load capacitance select
;                     - Program default value = 808Fh

           SETB command
           CLR nCS
           MOV SPDR, #080h
;          MOV serial_data, #01Fh
           JNB transmit_completed, $
           CLR transmit_completed
           MOV Reg_status, #000h
           SETB nCS
RET

AFAR_set:
;          Automatic Frequency Adjust Register
;                     - (Bit 7-6) Mode selection
;                               - Automatic frequency adjustment(low ac-
curacy)
;                     - (Bit 5-4) Allowable frquency offset
;                               - Frequency offset range, -> No limit
;                     - (Bit 3) Manual frequency adjustment strobe
;                     - (Bit 2) High accuracy mode
;                     - (Bit 1) Frequency refister enable
;                     - (Bit 0) Offset frequency enable
```

```
;                                   - Frequency offset calculation enabled
;                         - Program default value = C4F1h

            SETB command
            CLR nCS
            MOV SPDR, #0C4h
            MOV serial_data, #0F7h
            JNB transmit_completed, $
            CLR transmit_completed
            MOV Reg_status, #000h
            SETB nCS
RET

TCR_set:
;           Transmit Configuration Register
;                         - (Bit 8) Modulation polarity
;                                   - Logic 0 = Low deviation
;                                   - Logic 1 = High deviation
;                         - (Bit 7-4) Modulation bandwith
;                                   - Modulation deviation = 240kHz
;                         - (Bit 2-0) Transmit power
;                                   - TXD Power = max.
;                         - Program default value = 98F0h

            SETB command
            CLR nCS
            MOV SPDR, #099h
            MOV serial_data, #030h
            JNB transmit_completed, $
            CLR transmit_completed
            MOV Reg_status, #000h
            SETB nCS
RET

FSR_set:
;           Frequency Setting within band -Register
;                         - f_c = 310,24 MHz
;                         - (Bit 11-0) Frequency set
;                         - Program default value = A060h

            SETB command
            CLR nCS
            MOV SPDR, #0A4H
            MOV serial_data, #064H
            JNB transmit_completed, $
            CLR transmit_completed
            MOV Reg_status, #000h
            SETB nCS
RET

RCR_set:
;           Receiver Control Register
;                         - (Bit 10) Pin 16 Function
;                                   - Pin 16(DDeT) = valid-data-out flag
;                         - (Bit 9-8) Valid data detector response time
;                                   - Valid data detector response time =
fast
;                         - (Bit 7-5) Receiver baseband bandwith
;                                   - Receiver baseband bandwidth = 400 kHz
;                                   - Recommendation, at least twice the TXD
FSK deviation!
```

```
;                    - (Bit 4-3) Receiver LNA gain
;                              - Receiver LNA gain = 0dBm
;                    - (Bit 2-0) Digital RSSIA threshold
;                              - Receive Signal Strength indica-
tor(RSSI) threshold = -103 dBm
;          - Program default value = 9420h

           SETB command
           CLR nCS
           MOV SPDR, #092H
           MOV serial_data, #0A5H
           JNB transmit_completed, $
           CLR transmit_completed
           MOV Reg_status, #000h
           SETB nCS
RET

BFR_set:
;          Baseband filter register
;                    - (Bit 7) Automatic clock recovery
;                    - (Bit 6) Manual clock recovery
;                    - (Bit 4) Filter type 1 = Analog
;                              - Digital filter
;                    - (Bit 2-0) Data quality thershold
;                    - Program default value = C2B0h
           SETB command
           CLR nCS
           MOV SPDR, #0C2H
           MOV serial_data, #06FH
           JNB transmit_completed, $
           CLR transmit_completed
           MOV Reg_status, #000h
           SETB nCS
RET

FRMCR_set:
;          FIFO and RESET mode configuration register
;                    - (Bit 7-4) FIFO fill bit count
;                    - (Bit 2) FIFO fill start condition
;                    - (Bit 1) Synchronous pattern fill
;                    - FIFO fill start when preamble 2DD4h is received
;                    - Program default value = CA45h

           SETB command
           CLR nCS
           MOV SPDR, #0CAH
;          MOV serial_data, #043H
           JNB transmit_completed, $
           CLR transmit_completed
           MOV Reg_status, #000h
           SETB nCS
RET


DRSR_set:

           SETB command
           CLR nCS
           MOV SPDR, #0C6h
           MOV serial_data, #018h                    ;#098h
           JNB transmit_completed, $
```

```
            CLR transmit_completed
            MOV Reg_status, #000h
            SETB nCS
RET

PMR_set:
;           Power Management Register
;                       - (Bit 7) Receiver chain enable
;                       - (Bit 6) Baseband circuit enable
;                       - (Bit 5) Transmit chain enable
;                       - (Bit 4) Synthesizer enable
;                       - (Bit 3) Crystal scillator enable
;                       - (Bit 2) Low battery detector
;                       - (Bit 1) Wake up timer enable
;                       - (Bit 0) Clock output disable
            CLR transmit_completed
            SETB command
            CLR nCS
            MOV SPDR, #082h
;           MOV serial_data, #001h
            JNB transmit_completed, $
            CLR transmit_completed
            MOV Reg_status, #000h
            SETB nCS
RET

WTPR_set:
;           Wake-up Timer Period Register
;                       - (Bit 12-8) Exponent
;                       - (Bit 7-0) Multiplier
;                       - Timer set to 0.8s

SETB command
            CLR nCS
            MOV SPDR, #0E2h
            MOV serial_data, #019h
            JNB transmit_completed, $
            CLR transmit_completed
            MOV Reg_status, #000h
            SETB nCS
RET

DCSR_set:
;           Duty Cycle Set Register
;                       - (Bit 7-1) Duty Cycle Multiplier
;                       - (Bit 0) Duty Cycle Mode Enable
;                       - DutyC Cycle set to 20% ( ~160ms )

            SETB command
            CLR nCS
            MOV SPDR, #0C8h
            MOV serial_data, #0F9h
            JNB transmit_completed, $
            CLR transmit_completed
            MOV Reg_status, #000h
            SETB nCS
RET
end
```