(19) LIETUVOS RESPUBLIKA
VALSTYBINIS PATENTŲ BIURAS

(10) **LT 3323 B**

(12) **PATENTO APRAŠYMAS**

(11) Patento numeris: **3323**

(51) Int.Cl.[5]: **H01Q 7/04, H04Q 7/02**

(21) Paraiškos numeris: **IP1901**

(22) Paraiškos padavimo data: **1994 03 24**

(41) Paraiškos paskelbimo data: **1995 01 31**

(45) Patento paskelbimo data: **1995 06 26**

(31,32,33) Prioritetas: **037627, 1993 03 25, US**

(72) Išradėjas:
Alexis Torres, US
Luis R. Ortiz, US

(73) Patento savininkas:
TELULAR INTERNATIONAL, INC., No. 1 Celpage Building, Metro Office Park, Euynabo, Puerto Rico 00968, US

(74) Patentinis patikėtinis:
Rita Laurinavičiūtė, 5, UAB "Metida", Pilies g. 8/1-2, 2600 MTP Vilnius, LT

(54) Pavadinimas:
Celiuliarinis perdavimo aparatas, aparatas ir būdas siųstuvo sistemų testavimui

(57) Referatas:

Išradimas priklauso perdavimo sistemoms.

Celiuliarinė perdavimo sistemos diagnostinė sistema, įjungianti celiuliarinį interfeisinį įrenginį, kuris prijungia telefonus prie celiuliarinio siųstuvo, kurio interfeisas konvertuoja DTMF ar pulsinius skambinimo signalus į skaitmeninę formą perdavimui per celiuliarinį siųstuvą, kai skambinimas iš telefono naudojamas sujungimui su celiuliarinės sistemos abonentais. Šis išradimas ne tik tikrina siųstuvo ir jam paduodamo maitinimo funkcionalumą, bet ir atlieka interfeisinio įrenginio monotoringą.

Išradimas priklauso perdavimo sistemoms ir yra skirtas celiuliarinės perdavimo sistemos funkcionalumo patikrinimui. Šis išradimas artimas celiuliarinei interfeiso sistemai, kuri aprašyta JAV patentuose Nr. 4658096 ir Nr. 4737975. Sistemos, aprašytos šiuose patentuose, interfeisinis įrenginys susideda iš standartinio telefoninio rinkinio, faksimilinio aparato, modemo ar kito komunikacinio įtaiso, kurie prijungti prie celiuliarinio siųstuvo, kurio interfeisinis įrenginys leidžia normaliai komunikuoti minėtiems komukacijos aparatams per radijo siųstuvą. Interfeisinis įrenginys taip pat konvertuoja DTMF ar pulsinius surinkimo signalus į skaitmeninį formatą perdavimui per radijo siųstuvą, kai surinktas numeris naudojamas radijo sistemos šaukimui suformuoti.

Išradime pateikta sistema patikrina visų interfeiso įrenginio dalių funkcionalumą, bei kitas įrenginio charakteristikas: siųstuvo išduodamą galią, kuri gali kisti, ar siųstuvo dažnį, kuris gali būti keičiamas arba siųstuve, arba celiuliariniame tinkle.

Diagnostinė ir testavimo sistema, aprašyta šiame išradime, gali būti panaudota celiuliarinio tipo sistemoje, kaip ISDN ar pan., kur celiuliarinis adapteris ar interfeisinis įrenginys yra naudojamas DTMF ar pulsiniams signalams konvertuoti į skaitmeninę formą, po to perduodant juos celiuliarinio tipo siųstuvu, dirbančiu celiuliarinėje sistemoje.

Yra žinomi testiniai ir diagnostiniai įtaisai visai celiuliarinei sistemai. Taip pat yra žinomi patys save testuojantys įrenginiai. Tai aprašyta JAV patente Nr. 5016269, kur pateiktas celiuliarinis telefonas su pavojaus sistema. Ši sistema atlieka savęs testavimo

funkcijas ir pastoviai perduoda informaciją apie savo būseną į centrinę stotį per celiuliarinį tinklą. Vienok minėtame patente neatsispindi DTMF konvertatoriaus monitoringas ir diagnostinės funkcijos, kurios naudojamos anksčiau minėtuose JAV patentuose Nr. 4658096 ir Nr. 4737975, bei neįjungia skambinimo tono inicijacijos iš centrinės stoties sistemos darbingumo patikrinimui.

Celiuliarinio tipo sistemoms, kuriose gali būti naudojamas šiame išradime pateiktas įrenginys, priklauso skaitmeninės komunikacijos sistemos, turinčios intelektualias bazines stotis ir intelektualius nešiojamus terminalus, kurių kiekvienas turi tam tikrą ryšio zoną, ir konkrečiai skirtas skaitmeninei, radijo celiuliarinei, personalinei komunikacinei sistemai, palaikančiai pilną ISDN interfeisą, ir todėl galinčiai susijungti su visuomeniniu telefono tinklu ar bet kuriuo komutuojamu tinklu, kai personali komunikacinė sistema palaiko garso/duomenų/vaizdo perdavimą ir yra abipusė, funkcionali su bet kokia moduliacijos sistema, su celiuliarinio bendravimo protokolu, ir kur terminalai ir stotys valdomi programine įranga, bei visuomeninis komutuojamas telefono tinklas yra aprūpintas atitinkama tokio ryšio palaikymo programa ir duomenų baze.

Pastaruoju metu vis didėja mobilių ir portabilių komunikacijų priemonių funkcionalumas, tai leidžia atsisakyti laidinio ryšio palaikymo. Celiuliarinės sistemos kartu su paieškos ir kitomis sistemomis, leidžia realizuoti tikrą komunikacinių priemonių mobilumą. Žymūs technologiniai pasiekimai stipriai išplėtė belaidžių komunikacijų palaikymą. Šis palaikymas įjungia, bet neapsiriboja, skaitmeninį belaidį ryšį, celiuliarinį telefoninį ryšį, faksimilines ryšio

priemones ir lokalinius belaidžius tinklus, kai galimas ir ryšys per egzistuojančius visuomeninius tinklus ar kitus lokalinius tinklus (kaip kabelinės televizijos sistemos). Tokiu būdu, skaitmeninės personalinės komunikacijos sistemos (PCS) gali dirbti nepriklausomai ar kartu su lokaliais laidiniais tinklais, užpildydamos spragas, egzistuojančias dabartinėse komunikacinėse sistemose, taip atskleisdamos dar neįsisavintas sritis. PCS yra perspektyvios ir labai padidina tinklų lankstumą ir funkcionalumą. Atitinkamai PCS realizatoriai turi įsisavinti naujas nacionalines rinkas, atsižvelgiant į jų pobūdį.

Personalinės komunikacijos reikalavimai JAV keičiami priklausomai nuo jų funkcionalinio mobilumo. Vienas iš PCS privalumų yra galimybė realizuoti ryšį bet kam, bet kada ir bet kur, PCS labai padidina naudotojo mobilumą, tai išsprendžia ryšio su naudotoju problemą. PCS leidžia naudotojams nepraleisti svarbių skambučių, taip pat sutaupyti laiką ir pinigus atsakomiesiems skambučiams. PCS suderina radijo funkcionalumą ir visuomeninio telefoninio tinklo technologijas ir infrastruktūrą, ir įjungia dvipusio ir nenutraukiamo ryšio palaikymą. Svarbu pažymėti, kad PCS srityje dar yra daug vietos naujoms technologijoms ir pritaikymams, kaip privačiam ryšiui, mažesnių terminalų išdirbimui, portabiliems faksams, daugiakanaliams telefonams ir kt. Dabartinis aparatūrinis palaikymas negali realizuoti šias naujas PCS savybes. Pavyzdžiui, belaidžiai telefonai, naudojami įstaigose ar namuose, palaiko mažai kanalų ir turi mažą ryšio atstumą. Paieškos sistemos palaiko tik vienpusį ryšį. Celiuliariniai telefonai nerealizuoja visų PCS savybių. Po tam tikro laiko PCS įjungs standartinį aprūpinimą ir palaikys registracijos, automatinio šaukimo, duomenų perdavimo

ir kitas savybes. Tam, kad realizuotų daugelį šių naujų savybių yra reikalingos skaitmeninės PCS. Belaidės PCS padarys nereikalingas laidines sistemas, kitaip sakant, PCS papildys komunikacines sistemas naujomis savybėmis, pakeis šių sistemų išdirbimą, aprūpinimą ir savybes.

Šio išradimo objektas gali būti panaudotas tokiose sistemose ar kitose PCS sistemose, kur naudojamas celiuliarinio tipo adapteris ar interfeisas, kuris leidžia panaudoti standartines telefonines priemones šioje sistemoje, konvertuojant DTMF ar pulsinius signalus į skaitmeninį formatą, kuris gali būti perduotas į siųstuvą, bei palaiko kitas specifines sistemos savybes. Pvz. aliarmo sistemose taip pat gali būti panaudota paminėta sistema. PCS celiuliarinis adapteris taip pat palaiko kitas būtinas funkcijas, kaip surinkimo kodo generavimas, skambinimas ir pan., kaip aprašyta JAV patentuose 4658096 ir 4737975.

Išradimo tikslas yra savęs diagnozuojančios sistemos išdirbimas visoms celiuliarinio siųstuvo, turinčio interfeisinį įrenginį, kuris jungia standartinius telefonus su celiuliarine sistema ir verčia DTMF ar pulsinius surinkimo signalus į skaitmeninę formą, kai skambinimas vyksta iš standartinio telefono, ir siunčia juos į celiuliarinę sistemą, funkcijoms patikrinti. Šio išradimo objektas ne tik atlieka celiuliarinio siųstuvo ir jo maitinimo monitoringą, bet ir tikrina celiuliarinį interfeisą.

Testavimo aparatas gali būti prijungtas prie interfeiso, kuris prijungia komunikacinį įtaisą prie radijo siųstuvo šaukinių perdavimui ir gavimui per šį siųstuvą, ir testavimo aparatas turi savo diagnostinę sistemą interfeisinio įrenginio monitoringui.

Testavimo aparatas taip pat prijungia diagnostinius įtaisus prie interfeisinio įrenginio. Minėti diagnostiniai įrenginiai emuliuoja komunikacinio įrenginio, kaip telefono, fakso ir kt., funkcijas, generuodami atitinkamus signalus ir tokiu būdu nustato interfeisinio įrenginio darbingumą.

Testavimo aparatas taip generuoja pakėlimo signalą, tikrindamas skambinimo tono signalo generavimą interfeisiniame įrenginyje.

Testavimo aparatas taip pat generuoja DTMF signalą ir siunčia jį į interfeisinį aparatą, tuo patikrindamas interfeisinio įrenginio darbą.

Testavimo aparatas taip generuoja pakėlimo signalą ir padėjimo signalą, tikrindamas, kaip interfeisinis įtaisas atsako į ryšio nutraukimą. Testavimo aparatas taip pat atlieka skambinimo procedūrą per tinklą, kreipdamasis į tą patį interfeisinį aparatą, tuo patikrinamas užimtumo signalo generavimas.

Išradimas geriau suprantamas pagal pridedamus piešinius, kur:

Fig. 1A ir 1B parodo pagrindinius testavimo etapus;

Fig. 2 parodo pakėlimo testavimo programos blokinę schemą;

Fig. 3 parodo skambinimo testavimo programos blokinę schemą;

Fig. 4 parodo DTMF tono generatoriaus testavimo programos blokinę schemą;

Fig. 5A - 5C parodo skambučio generavimo testavimo programos blokinę schemą;

Fig. 6 parodo atsakymo į skambutį testavimo programos blokinę schemą;

Fig. 7 parodo atsakymo į skambutį testavimo programos blokinę schemą;

Fig. 8A ir 8B parodo klaidų generavimo programos blokinę schemą;

Fig. 9 parodo numetimo programos blokinę schemą;

Fig. 10 parodo sistemos blokinę schemą;

Fig. 11 - 14 parodo įvairias grandines, naudojamas celiuliarinio interfeiso ir siųstuvo testavimui, simuliuojant testuojamus įvykius.

Autodiagnostinė sistema yra skirta celiuliarinio tipo interfeisams ir sistemoms, aprašytoms JAV patentuose 4658096 ir 4737975. Papildomai, autodiagnostinė sistema gali būti naudojama ir kitose radijo ryšio sistemose, kaip IMTS, kur palaikomas ryšys tarp bazinės stoties ir terminalų, ir kur perdavėjas sujungtas su interfeisiniu adapteriu, kaip aprašyta JAV patentuose 4658096 ir 4737975, pvz. sujungiant komunikacines priemones su radijo perdavėju. Interfeisas palaiko naudojimui reikalingas funkcijas, pvz. aliarmo sistemoms nereikalinga skambučio generavimas. Sistemose, kur reikalinga tik gauti skambučius, nereikia DTMF ar

pulsinių signalų konvertuoti į skaitmeninę formą. Reikalinga programinė įranga yra pateikta JAV patentuose 4658096 ir 4737975, ir ji gali būti aktyvuojama tiek rankiniu būdu, tiek automatiškai diagnozavimo tikrinimui, priklausomai nuo testo gautų rezultatų. Gali būti patikrintas kiekvienas komponentas, pvz. DTMF konverteris, linijų perjungimo interfeisas ir kt. Papildomai gali būti patikrintas perdavėjas, baterijos ir kt. Išradime pateiktas įrenginys gali realizuoti skambutį per celiuliarinį tinklą, siekiant patikrinti linijos signalus. Taip pat išradimo įrenginys gali realizuoti skambutį, siekdamas gauti 1000 Hz toną iš nustatytos vietos, siekiant patikrinti ar celiuliarinė sistema dirba gerai. Tai labai patogu, siekiant pratestuoti ryšį, tai leidžia nustatyti, ar gedimas yra centrinėje stotyje ar terminale.

Pagal išradimą, naudojami du moduliai. Pirmasis atlieka televaldymo funkcijas, o antrasis vykdo apskaitą (teleapskaitą). Televaldymo modulis generuoja testus, tarp jų ir skambinimo nustatytu numeriu tęstą, po to praneša apie gautus rezultatus. Be to, modulis leidžia paskambinti savo paties numeriu užimtumo signalui patikrinti. Šie testai gali būti paleisti paspaudus specialų klavišą. Kai jis paspaudžiamas, šalia jo esantis šviesos diodas (LED) pradeda mirksėti. Jeigu po testo jis neužgesta, tai rodo, kad yra gedimai. Testas trunka daugiausiai 40 sek. Jeigu LED užgesta, viskas funkcionuoja gerai. Programinė ir aparatūrinė įranga leidžia paleisti testą iš šalies.

Kitas modulis (grandinė) atlieka šias funkcijas: apdoroja apskaitos signalą iš celiuliarinės sistemos ir generuoja apskaitos signalus telefonui. Įrenginys

atpažįsta signalus ir nereikalauja lokalinės apskaitos. Jis gali būti naudojamas su bet kokia celiuliarinio tipo sistema, kuri sugeba generuoti apskaitos signalus.

Televaldymo modulio blokinės schemos aprašymas

Naudotojas paleidžia testus, paspausdamas autodiagnostikos klavišą, dėl to paleidžiama autotestavimo programa.

Fig. 1A ir 1B rodo pagrindinę DO_TEST programą, iš kurios šaukiamos įvairios procedūros. Po kiekvienos procedūros yra patikrinami klaidų kodai. Esant gedimui, testavimas nutraukiamas, ir gauti rezultatai atsispindi keturiuose LED.

Pagal Fig. 1A, pirmame žingsnyje inicializuojami kintamieji ir vėliavėlės. Žingsnyje 2 programa kviečia Hook_Test funkciją, kurioje naudotojo telefonas yra atjungiamas nuo TIP ir RING linijų, ir prie šių linijų prijungiamas televaldymo modulis, kaip aprašyta JAV patentuose 4658096 ir 4737975. 3 žingsnyje patikrinamos klaidos Hook_Test. Jei aptinkama klaida, testavimas nutraukiamas, ir programa pereina į žingsnį 14 (Fig. 1B). Jeigu klaidų nėra, programa kviečia Dial_test funkciją. Toliau tikrinamas skambinimo tonas, ir jeigu surandamos klaidos, nueinama į žingsnį 14. Jeigu klaidų nėra, šaukiama DTMF_Test funkcija (žingsnis 6), ir, esant klaidoms, testavimas nutraukiamas ir einama į žingsnį 14, o jeigu klaidų nėra, kviečiama funkcija Ring_Test. Esant klaidoms, einama į žingsnį 14, nesant - kviečiama funkcija Ring_Answer_Test (žingsnis 10), kuri klaidų atveju nukreipia į žingsnį 14. Sekantis kviečiamas testas yra funkcija Make_Call_Test, tikrinanti skambinimo testą. Klaidos atveju einama į žingsnį 14,

kitaip kviečiama funkcija Fin_Self_Test 15 žingsnyje. Jeigu nesurasta klaida, programa pereina į žingsnį 14. 14 žingsnyje kviečiama Error_Acc funkcija. Gale išvalomi kintamieji, numetamos vėliavėlės ir grįžtama į normalų funkcionavimo režimą.

Fig. 2 - 9 smulkiau pateiktos aukščiau minėtos testavimo funkcijos.

Fig. 2 Hook_Test funkcija iš pradžių inicializuoja kintamuosius ir numeta vėliavėles. Žingsnyje 17 nugesinami LED. Toliau atjungiamas naudotojo telefonas nuo TIP ir RING linijų, ir prijungiamas televaldymo modulis. Po užlaikymo (21) simuliuojamas sujungimas. Po užlaikymo patikrinama sujungimo būsena (23). Jeigu nėra sujungimo (25) užstatomas 1 ERROR_FLAG ir MAIN_FLAG_ERROR. Jeigu klaidų nėra, einama į 26, kur leidžiamas skambinimo tonas, po to funkcija grąžina valdymą pagrindinei programai.

Fig. 3 pateikia Dial_Test funkcijos blokinė schema. Ši funkcija iš pradžių inicializuoja kintamuosius ir numeta vėliavėles (27). Po užlaikymo (28) testuojamas skambinimo tonas (29). Jeigu surandama klaida užstatomas 3 ERROR_FLAG ir MAIN_FLAG_ERROR (30). Nesant klaidos, 711 ms trukmės cikle patikrinamas skambinimo tonas, po to grąžinamas valdymas pagrindinei programai.

Fig. 4 pateikta DTMF_Test funkcijos, kuri patikrina DTMF konversiją į skaitmeninį kodą, blokinė schema. Uždraudžiamas skambinimo tonas (32), pirmasis DTMF tonas numetamas į 0 (33), po to siunčiamas šią reikšmę atitinkantis tonas. Po užlaikymo (35) uždraudžiamas DTMF tonas ir po sekančio užlaikymo šis tonas nuskaitomas (38) ir patikrinamas (39). Jeigu siunčiamas

ir gaunamas DTMF tonai nesutampa užstatomas 4 ERROR_FLAG ir MAIN_FLAG_ERROR (40). Jeigu jie sutampa, išrenkamas sekantis skaičius, ir taip patikrinami visi DTMF tonai. Po patikrinimo grąžinamas valdymas pagrindinei programai.

Fig. 5A ir 5B Ring_Test funkcija tikrina skambučio generavimo impulsus. Ši funkcija iš pradžių inicializuoja kintamuosius ir numeta vėliavėles (43). Modulis sugeneruoja šaukimą, ir po užlaikymo (45) generuojama skambinimo seka. Patikrinama, ar per dvi sekundes gaunamas skambutis (47-53). Jeigu viskas gerai, einama į žingsnį 54. Jeigu per 2 sekundes nesugeneruojamas skambutis, indikuojama klaida (49). 55-59 žingsniuose patikrinama, kad skambutis skambėtų mažiausiai 1.2 sek. ir nebūtų pakėlimo signalo. Esant klaidai, pereinama į žingsnį 49. Po to patikrinama ar skambutis skamba daugiau kaip 2.5 sek. (60-62). Esant klaidai, pereinama į žingsnį 49. Po to patikrinama ar yra pakėlimo signalas skambinimo (2 sek.) metu. Jam esant, einama į žingsnį 49 (klaida). Tai patikrinama du kartus. Jeigu viskas gerai, grįžtama į pagrindinę programą, jei ne - į žingsnį 49. Ten numetamas skambinimo režimas ir užstatomas 5 ERROR_FLAG ir MAIN_FLAG_ERROR (50). Po to grįžtama į pagrindinę programą Do_Test.

Fig. 6 yra parodyta Ring_Answer_Test funkcija, kuri tikrina pakėlimą, esant skambinimui iš išorės. Iš pradžių patikrinamas skambinimas ir pakėlimo signalo nebuvimas (68-69). Esant klaidai, pereinama į 73. Jeigu indikuojamas skambinimas, generuojamas pakėlimo signalas (74). Jeigu yra pakėlimo būsena (78), inicijuojama būsena ir grįžtama į pagrindinę programą. Esant

klaidai, einama į 73, kur užstatomas 6 ERROR_FLAG ir MAIN_FLAG_ERROR, po to einama į 78.

Fig. 7A ir 7B parodyta Call_Test funkcija, kuri tikrina perdavėjo maitinimą ir generuoja sujungimą su jo pačio numeriu. Iš pradžių patikrinamas maitinimas (79), ir jam esant gaunamas perdavėjo numeris, po to skambinama šiuo numeriu. Esant bet kokiai klaidai, einama į 93. Po to užstatoma IN USE būsena perdavėjuje. Esant klaidai, einama į 93. Patikrinamas užimtumo tonas (89). Jeigu jis surandamas, einama į 93 (viskas gerai). 93 žingsnyje patikrinami klaidų kodai, po to grįžtama į pagrindinę programą.

Fig. 8A ir 8B pateikta ERROR_CODE generavimas aptiktoms klaidoms, tai vėliau panaudojama išvedimui į LED. Pagal Fig. 8A, funkcija iš pradžių patikrina vieną iš dviejų klaidų požymių MAIN_FLAG_ERROR. Jeigu jis lygus 1, ERROR_CODE užsiunčiamas 1 (96), ir grįžtama į pagrindinę programą. Jeigu ne, tikrinama ar MAIN_FLAG_ERROR lygus 2, ir, jeigu taip, į ERROR_CODE užsiunčiamas 2 (98), ir grįžtama į pagrindinę programą. Jeigu ne, tikrinama ar MAIN_FLAG_ERROR lygus 3, ir, jeigu taip, į ERROR_CODE užsiunčiamas 3 (100), ir grįžtama į pagrindinę programą. Jeigu ne, tikrinama ar MAIN_FLAG_EROR lygus 4, ir, jeigu taip, į ERROR_CODE užsiunčiamas 4 (102), ir grįžtama į pagrindinę programą. Jeigu ne, tikrinma ar MAIN_FLAG_ERROR lygus 5, ir, jeigu taip, į ERROR_CODE užsiunčiamas 5 (104), ir grįžtama į pagrindinę programą. Jeigu ne, tikrinama ar MAIN_FLAG_ERROR lygus 6, ir, jeigu taip, į ERROR_CODE užsiunčiamas 6 (106), ir grįžtama į pagrindinę programą. Jeigu ne, neinama į žingsnį 107. Fig. 8B funkcija pradeda darbą, patikrindama kitą požymį MAIN1_FLAG_ERROR. Tikrinama ar MAIN_FLAG_ERROR

lygus 1, ir, jeigu taip, į ERROR_CODE užsiunčiamas 7 (108), ir grįžtama į pagrindinę programą. Jeigu ne, tikrinama ar MAIN_FLAG_ERROR lygus 2, ir, jeigu taip, į ERROR_CODE užsiunčiamas 8 (110), ir grįžtama į pagrindinę programą. Jeigu ne, tikrinama ar MAIN_FLAG_ERROR lygus 3, ir, jeigu taip, į ERROR_CODE užsiunčiamas 9 (112), ir grįžtama į pagrindinę programą. Jeigu ne, tikrinama ar MAIN_FLAG_ERROR lygus 4, ir, jeigu taip, į ERROR_CODE užsiunčiamas 10 (114), ir grįžtama į pagrindinę programą. Jeigu ne, tikrinama ar MAIN_FLAG_ERROR lygus 5, ir, jeigu taip, į ERROR_CODE užsiunčiamas 11 (116), ir grįžtama į pagrindinę programą. Po to funkcija grąžina valdymą pagrindinei programai Do_Test.

Fig. 9 pateikta Fin_Self funkcijos blokinė schema. Ši funkcija numeta interfeisą į normalaus funkcionavimo būseną ir indikuoja testo pabaigą, uždegdama LED keturis kartus. Ši funkcija prasideda nuo 117, kai televaldymo modulis generuoja pakėlimo signalą. Po to išvalomi kintamieji ir požymiai (118), užgesinami LED (119). Po užlaikymo LED įjungiami (121). Visa tai kartojama 4 kartus. Po ciklo visi LED užgesinami (124) ir valdymas perduodamas pagrindinei programai.

Toliau seka išradimo objekto aprašymas su nuorodomis į Fig. 15, prie kurio pateikta atskirų grandinių aprašymai su programine įranga.

Fig. 10 yra parodytas televaldymo modulis, pažymėtas numeriu 10. Šis modulis instaliuojamas į celiuliarinio tipo sistemos adapterį, kaip aprašyta JAV patentuose 4658096 ir 4737975, ir yra skirtas autotestavimo procesui inicijuoti arba paspaudžiant specialų klavišą, arba automatiškai kas 12 val., tai aprašyta aukščiau.

Autotesto rezultatai išvedami į LED. Televaldymo modulis taip pat turi galimybę perduoti testavimo rezultatus į aptarnavimo centrą, tai realizuojama automatiškai surenkant tam tikrą numerį. Reikalinga, kad aptarnavimo centras turėtų tam tikslui pritaikytą aparatūrinę ir programinę įrangą. Autodiagnostinis centras patikrina celiuliarinį adapterį ir jo komponentus: pagrindines interfeiso grandines, duomenų kabelius, perdavimo liniją, anteną, perdavėją, pakrovimo grandinę ir maitinimo įtampą. Televaldymo modulis testavimo metu kviečia adatoriaus celiuliarinį numerį, t.y. skambina pats sau. Indikuotas užimtumo signalas rodo, kad perdavimo grandinė veikia normaliai.

Televaldymo modulis 10 yra pritaikytas dirbti su celiuliarinio tipo adapteriu tam, kad patikrintų jo darbingumą be bereikalingo techninio personalo iškvietimo. Televaldymo modulis yra maitinamas iš interfeisinės plokštės. Televaldymo grandinė nėra autonominė ir jos operacijos priklauso nuo gaunamų valdymo signalų. Autodiagnostinė grandinė vykdo sekančius testus:

atsakymo klaidos (neteisingas numeris) - pirma, tikrinama skambučio įtampa, antra - tikrinama skambučio pabaiga, trečia - skambinimo tonas, ketvirta - MFDT tonų tikrinimas, perdavėjo testas, priėmimo grandinės testas, perdavimo grandinės testas, pakrovimo grandinės testas - tikrinamos dažnuminės charakteristikos, esant 12 kHz, perdavėjo maitinimo tikrinimas, maitinimo sistemos efektyvumo tikrinimas. Šie testai buvo aprašyti smulkiai aukščiau.

Televaldymo modulis yra prijungtas prie celiuliarinės sistemos interfeiso per 20 kontaktų jungtį. Testavimas

prasideda, įjungus jungiklį. Aptikęs gedimą, vartotojas paspaudžia testavimo klavišą, dėl to televaldymo modulis 10 atjungia telefoną nuo celiuliarinio interfeiso ir pradeda vykdyti testų seką. Televaldymo modulis taip automatiškai testuoja interfeisinę plokštę kas 12 val. Vėliau bus įjungta galimybė paleisti testus iš išorinio centro. Tam reikės pakeisti programinę įrangą ir instaliuoti papildomą programinę ir aparatūrinę įrangą aptarnavimo centre. Po testų inicijavimo, LED mirksi kas sekundę, žymėdami, kad vyksta testavimas. Pilnas testas trunka maždaug apie 40 sek. Po to testo rezultatai indikuojami LED pagalba. Po to, kai uždegami atitinkami LED, interfeisinis įtaisas grįžta į normalią būseną. Pagal degančius LED galima nustatyti gedimo tipą.

4 LED indikatorius yra skirtas nustatyti gedimui. Tai leidžia remontuojančiam personalui greitai pašalinti gedimą. Degančių LED kombinacija šviečia 60 sek. po testo pabaigos. Kombinacijų reikšmės yra pateiktos žemiau.

Po testo, jeigu buvo aptiktas gedimas, bus perduodama į televaldymo centrą apie surastą gedimą. Dėl to informacija apie kiekvieną įrenginį bus saugoma duomenų bazėje. Jeigu testas buvo pradėtas, paspaudus klavišą, nepriklausomai nuo to ar buvo surasti gedimai ar ne, informacija apie rezultatus bus perduodama į televaldymo centrą.

Žemiau pateikti klaidų kodai, atspindimi LED eilute:

4.10 KLAIDŲ KODAI

Klaidos tipas	LED5	LED4	LED3	LED2
Nėra gedimo	OFF	OFF	OFF	OFF
Ciklo klaida	OFF	OFF	OFF	ON
Ciklo klaida	OFF	OFF	ON	OFF
Skambinimo tonas	OFF	OFF	ON	ON
MFTD detektorius	OFF	ON	OFF	OFF
Skambučio detektorius	OFF	ON	ON	ON
Šaukimo atsakymas	OFF	ON	ON	OFF
Perdavėjo maitinimas	OFF	ON	ON	ON
Siųstuvas	ON	OFF	OFF	ON
Imtuvas (užimtas)	ON	OFF	OFF	OFF
12 kHz detekcija	ON	OFF	ON	ON
Įtampos šaltinio maitinimas	ON	OFF	ON	OFF
"NO SERVICE"	ON	ON	ON	ON
Perdavimo grandinė	ON	ON	ON	OFF
Nenustatyta	ON	ON	OFF	ON
Nenustatyta	ON	ON	OFF	OFF

Autotesto seka yra sekanti. Visų pirma televaldymo grandinė yra atjungiama nuo instaliacijos. Po to simuliuojama vartotojo telefono atsakymas. Jeigu viskas tvarkoje, generuojamas skambinimo tonas. Po to patikrinama skambinimo tono generacija, esant minimaliam dažniui ir lygio diapazonui. Po to pradedamas MFTD tono detektoriaus testas, kurio metu

generuojama 16 MFTD tonų, ir patikrinant ar tie tonai gerai atpažįstami. Toliau patikrinamas skambučio generatorius, po to jis tikrinamas, esant minimaliam dažniui ir lygio diapazonui. Po to yra simuliuojamas telefono atsakymas, skambinimo grandinei patikrinti. Sekantis testas patikrina 12 kHz dažnio sistemą, naudojamą celiuliarinio tipo tinkle.

Sekantys testai patikrina perdavėją. Visų pirma yra patikrinama maitinimo įtampa, kuri yra +12 V pastovi srovė, tiekiama per duomenų jungtį. Po to patikrinamas "NO SERVICE" signalo nebuvimas. Po to skambinama per celiuliarinę sistemą į testuojamą aparatą ir tokiu būdu patikrinamas užimtumo signalas. Jeigu jis randamas, perdavimo ir priėmimo grandinės funkcionuoja normaliai. Šis testas vykdomas, tik inicijavus testus rankiniu būdu. Maitinimo sistemos įtampa taip pat yra patikrinama. Jeigu įtampa yra mažesnė kaip 13,4 V, tai laikoma gedimu.

Fig. 10 pagrindinis televaldymo modulis yra prijungtas prie celiuliarinio tipo adapterio ar interfeisinės plokštės 14. Televaldymo modulyje yra grandinės, atliekančios diagnostinius testus, kaip aprašyta aukščiau. Šios grandinės yra tokios: užimtumo signalo detektorius 16 naudojamas vykdant skambinimo sau pačiam procedūrą, 12000-ciklų detektoriaus grandinė 18 naudojama pulsinio pakrovimo testavimui, kai patikrinama esant 12 kHz dažniui; MFTD generatoriaus grandinė 20 generuoja DTMF signalus, perduodamus į celiuliarinį adapterį, skambučio detektoriaus grandinė 22 naudojama skambučiams identifikuoti, skambinimo tono grandinė 24 naudojama skambinimo tono patikrinimui, įtampos detektoriaus grandinė 28 naudojama maitinimo įtampai patikrinti ir televaldymo modulio valdymo

įrenginys 30 taip pat naudojamas pranešti testo rezultatus centrinei aptarnaujančiai stočiai. Kiekviena paminėta grandinė yra parodyta Fig. 11 - 14 ir smulkiau aprašyta žemiau.

Fig. 11 - 14 yra parodytos elektrinės grandinės, aukščiau aprašytiems testams vykdyti. Fig. 11 yra parodyta sujungimo grandinė. Dalis, pažymėta "A" parodo vietą, kur televaldymo grandinė yra prijungta prie TIP ir RING linijų. Relė, pažymėta "RELAY1" naudojama naudotojo telefonui atjungti nuo celiuliarinio interfeiso testavimo metu. Tuo metu televaldymo grandinė yra prijungiama prie TIP ir RING linijų. Kai "ENABLE-TEST" signalas yra aukšto lygio, tranzistorius Q3 yra prisotinamas ir įjungiama relė RELAY1, tokiu būdu prijungiant valdymo grandinę prie TIP ir RING linijų. Fig. 11 dalis, pažymėta "B" rodo naudotojo telefono prijungimą, kur telefonas gali būti prijungtas dviem būdais. J1 yra normalaus RJ11 tipo telefoninė jungtis.

Sekcija Fig. 11 pažymėta "C" yra audio interfeiso grandinė, kuri naudojama kaip interfeisas tarp TIP ir RING linijų ir audio įtaiso. T1 yra telefoninis suderinimo transformatorius. Jo funkcija yra suderinti pastovios ir kintamos srovės charakteristikas, kai gaunami ir siunčiami audio signalai. Diodai D1 ir D2 naudojami nupjauti signalams didesniems kaip ±4,5 V. Taip pat naudojamas optronas SSR1, kuris su tranzistoriumi Q4 palaiko TIP ir RING uždarymo mechanizmą. Kai "Conn-Loop" signalas yra aukštas tranzistorius Q4 yra prisotintas, užmaitinamas SSR1, uždaroma grandinė ir atliekamas atjungimas.

Sekcija Fig. 11, pažymėta "D" yra skambučio detektorius, kuris susideda iš dviejų dalių: U2, kuri yra skambučio detektoriaus grandinė ir U9 su optronu, kuris turi CMOS signalo lygio išėjimą. Srovė, tekanti tarp 4 ir 7 U2 suformuoja išėjimo signalą, kai generuojama skambučio įtampa TIP ir RING linijose. Ši srovė per optroną U9 perduodama į išėjimą/įėjimą.

Sekcija Fig. 11 pažymėta "E" yra srovės kilpos detektorius, turintis optroną pertekančiai srovei detektuoti. Kai testas yra leidžiamas ("Enable-Test" signalas yra aukštas) ir įvykdytas atjungimas ("Conn-Loop" signalas yra aukštas) TIP ir RING linijomis teka srovė. Ši srovė yra transformuojama U17 į žemo lygio išėjimo/įėjimo signalą.

Sekcija Fig. 11 pažymėta "F" yra išėjimo/įėjimo audio grandinė sudaryta iš 1/4 U1, operacinio stiprintuvo IS. Pagrindinė šios grandinės funkcija yra izoliuoti ir sustiprinti audio signalus. Šios grandinės išėjimas vadinamas "Audio-in". Taip pat per šią sekciją DTMF išėjimas yra siunčiamas į TIP ir RING linijaas. Šis DTMF signalas yra pažymėtas "Tone-Out".

Fig. 12 "G" dalis yra DTMF tono generatorius, sudarytas iš U4, DTMF tono generatoriaus ir 1/4 U1, naudojamo kaip išėjimo stiprintuvas. Siekiant sugeneruoti DTMF toną pirmieji keturi kodo bitai paduodami į kontaktus, pažymėtus "DTMF-OUT", "DTMF1-OUT", "DTMF2-OUT" ir "DTMF3-OUT" prie U4. Užstačius reikiamus signalus, signalo "DTMF-OUT" yra pakeičiamas iš žemo į aukštą lygį, dėl to sugeneruojamas reikalingas tonas. Audio išėjimas po sustiprinimo U1:C yra pažymėtas "TONE-OUT". Tono generavimo stabdymui "DTMF-OUT" signalas numetamas į žemą lygį.

Fig. 12 sekcija "H" yra išėjimo/įėjimo portas, aptarnaujamas IS U8. Ši grandinė naudojama celiuliarinio interfeiso mikrokontrolerio duomenų nuskaitymui ir visų televaldymo modulio išėjimų kontrolei. Portai A ir C yra įėjimo portai, o portas B - išėjimo. Jungiklis J4 perjungia automatinį periodinį testą tarp 12 ir 24 val. J5 yra naudojamas leidimui sujungti su "automatiniu atsakovu".

Fig. 12 sekcijos "I.1" ir "I.2" yra maitinimo, duomenų ir kontrolės jungtys, ir įjungia J2 jungtį maitinimui prijungti, taip pat duomenų siuntimui ir gavimui. Ji taip pat įjungia 4 kontaktų jungtį į LED panelę, kurioje atspindimi testo rezultatai, ir prijungia išorinį klavišą rankiniam testo paleidimui. Sekcija "J" yra maitinimo reguliatorius, ir įjungia 5 V palaikymo grandinę, reikalingą skaitmeninėms grandinėms maitinti.

Televaldymo modulis 10 įjungia šešis paprasto tono detektorius audio traktui, dažniui ir laikui tikrinti su ROH, skambinimo, užimtumo ir perkrovimo tonais. Kiekvienas iš šių tono detektorių turi operacinį stiprintuvą ir paprasto tono detektoriaus schemą. Opamp įėjimas yra prijungtas prie linijos "AUDIO-IN". Detektavimo dažnis nustatomas varžos tarp taškų 5 ir 6 ir kondensatoriaus tarp 6 ir žemės dėka. Kai tonas atpažįstamas, detektoriaus išėjimas nuvirsta į žemą lygį.

Sekcija "K" yra skambinimo tono detektoriaus grandinė. Esant skambinimo tonui U3 įėjime, "DIAL-DETECT" linija numetama į žemą lygį. Sekcija "L" rodo užimtumo tono detektoriaus grandinę. Kai yra užimtumo tonas U6 įėjime, "BUSY-DETECT" linija numetama į žemą lygį.

Sekcija "M" Fig. 13 rodo 400 Hz tono detektoriaus grandinę. Kai yra paduotas 400 Hz tonas į U14 įėjimą, "400-DETECT" linija numetama į žemą lygį.

Sekcija "N" Fig. 13 rodo 800 Hz tono detektoriaus grandinę. Kai yra paduotas 800 Hz tonas į U13 įėjimą, "800-DETECT" linija numetama į žemą lygį. Sekcija "O" Fig. 13 rodo 1020 Hz tono detektoriaus grandinę. Kai yra paduotas 1020 Hz tonas į U16 įėjimą, "1020-DETECT" linija numetama į žemą lygį. Sekcija "P" Fig. 13 rodo 12 kHz tono detektoriaus grandinę. Kai yra paduotas 12 kHz tonas į U12 įėjimą, "12kHz-DETECT" linija numetama į žemą lygį.

Sekcija "Q" Fig. 13 rodo DTMF tono dekoderį, kuris sudarytas iš U11, DTMF tono detektoriaus, ir 1/4 U1, naudojamo kaip pradinis stiprintuvas. Kai DTMF tonas yra įėjime (IN-), valdymo linija "DTMF-STROBE" numetama į žemą lygį, kol IS dekoduoja signalą. Po tono dekodavimo, keturių bitų kodas yra paduodamas į "DTMFO-IN", "DTMF1-IN", "DTMF2-IN" ir "DTMF3-IN" U11. Kontrolinė linija "DTMF-STROBE" grįžta į aukštą lygį. Sekcija "R" stebi 4,5 V kintamą srovę ir įjungia srovės detektorių U10. Varžos R17 ir R33 suformuoja įtampos daliklį, kuris leidžia registruoti U10 įtampą, žemesnę kaip 13 V. Kai maitinimo įtampa krenta, ji įgauna baterijos įtampos lygį ne didesnį kaip 13 V. Grandinė registruoja kritimą, ir U10 liniją "AC-DETECT" numeta į žemą lygį.

Toliau seka programos, palaikančios išradimo objekto funkcijas, listingas:

```
nopi nosb db noge nimo nopr ep
$nolist
$include (reg252.pdf)
$list

zero                    equ         0ah
star                    equ         0bh
pound                   equ         0ch
disbuf_start            equ         80h
disbuf_end              equ         90h
msgbuf_start            equ         90h
msgbuf_end              equ         0h

main_flag               equ         0b0h
;hook_slic              1
;dial                   2
;dtmf_                  3
;ring_test              4
;ans_test               5
main_flag_error         equ         0b1h
;hook_slic_error        1
;hook_ring_error        2
;dial_error             3
;dtmf_error             4
;ring_test_error        5
;ans_test_error         6

main1_flag              equ         0b2h
;power_radio            1
;call_test              2
;call_tone              3
main1_flag_error        equ         0b3h
```

```
;power_radio_error    1
;call_test_error      2
;call_tone_error      3
;inuse                4
;1000hz               5
error_code            equ         0b4h
;hook_slic_error      1
;hook_ring_error      2
;dial_error           3
;dmft_error           4
;ring_test_error      5
;ans_test_error       6
;power_radio_error    7
;call_test_error      8
;call_tone_error      9
;insune               10
;1000hz               11

cero                  equ         00h
uno                   equ         08h
dos                   equ         04h
tres                  equ         0ch
cuatro                equ         02h
cinco                 equ         0ah
seis                  equ         06h
siete                 equ         0eh
ocho                  equ         01h
nueve                 equ         09h
estrella              equ         0dh
libra                 equ         03h

; registrai užima 00h-007h (bankas 0)
;r0          saugojimo buferis
;r1          nuoroda į duomenų buferį
;r2          skaičių skaitiklis
```

```
;r3          užlaikymo ciklas magistraliniam taimeriui
;r4          užlaikymo ciklas magistraliniam taimeriui
;r5          naudojamas taimerio
;r6          naudojamas send_time
;r7          naudojamas taimerio

; kintamųjų saugojimo vietos

time_off1          data        10h
time_off2          data        11h
time_off3          data        12h
time_on1           data        13h
time_on2           data        14h
pulse_digit        data        15h
in_use_off_timer   data        16h
ring_timer         data        17h
bell_timer         data        18h
bell_timer2        data        19h
gndstart_timer1    data        1ah
gndstart_timer2    data        1bh
;highpoint         data        1ch
;lowpoint          data        1dh
highpoint          data        35h
lowpoint           data        36h
test_min           data        1eh
;free              data        1fh

;
; atminties zona nuo 20h iki 29h naudojama pagrindiniame modulyje
;
flags              DATA        20h
fhook              bit         00h
fsend              bit         01h
fflash             bit         02h
fhang              bit         03h
```

```
ftime               bit         04h
fdigit              bit         05h
fonhook             bit         06h
fdtmfin             bit         07h

inds                DATA        21h
roam                bit         08h
noserv              bit         09h
lock                bit         0ah
rdis                bit         0bh
horn                bit         0ch
f_init              bit         0dh
f_start             bit         0eh
test_enable         bit         0fh

indasc              DATA        22h
inuse               bit         10h
ap1                 bit         11h
ap2                 bit         12h
test_end            bit         13h
;not_error          bit         14h
return_ring         bit         15h
error_flag          bit         16h
ring_second         bit         17h

io_status           DATA        23h
strobe              bit         1ch

flags2              DATA        24h
fdig_ready          bit         20h
fanswer             bit         21h
fsendtimer          bit         22h
f_inuseofftiming    bit         23h
funlocktimer        bit         24h
fdtone              bit         25h
```

```
fdecoder_busy       bit         26h
fbellsound          bit         27h

px1_temp            DATA        25h
mitelq1             bit         28h
mitelq2             bit         29h
mitelq3             bit         2ah
mitelq4             bit         2bh
power_hold          bit         2ch
spare1              bit         2dh
spare2              bit         2eh
no_ring             bit         2fh

px2_temp            DATA        26h
loopswitch_         bit         30h
ring_               bit         31h
mute                bit         32h
roh_bost            bit         33h
sw_sendtimer        bit         34h
sw_dialtone         bit         35h
sw_gndstart         bit         36h
sw_data             bit         37h

lamps_temp          DATA        27h
;free               bit         38h
;free               bit         39h
;free               bit         3ah
;free               bit         3bh
roamlamp            bit         3ch
noservlamp          bit         3dh
locklamp            bit         3eh
inuselamp           bit         3fh

flags3              DATA        28h
frohtimeout         bit         40h
```

```
fcall           bit         41h
f_one_sec       bit         42h
roh_on          bit         43h
fspec1          bit         44h
fspec2          bit         45h
fremote_hup     bit         46h
flocktimer      bit         47h

;
;  TRU uždaviniai
;

f_cmds          DATA        29h

;  TRU uždaviniai kviečiami iš pagrindinio uždavinio

;  Uždavinys 1:
;  a) Įtampos tikrinimas NEC 3700
;  b) Pakeisti rakto sovi OKI ir inicijuoti Motorola
;  c) Paleisti rdbus programą Audiovox indikatoriams nuskaityti
;  d) Stebėti Audiovox CTX - 3100A signalus ir juos nustatyti
;
;  Uždavinys 2:
;  a) NAM perjungimas NEC 3700
;
;  Uždavinys 3:
;  a) Įtampos tikrinimas NEC 3700
;  b) Pakeisti rakto sovi OKI ir inicijuoti Motorola
;  c) Paleisti rdbus programą Audiovox indikatoriams nuskaityti
;  d) Stebėti Audiovox CTX - 3100A signalus ir juos nustatyti
;
;  Uždavinys 4:
;   a) Paleisti rdbus programą Audiovox indikatoriams nuskaityti
;   b) Stebėti Audiovox CTX - 3100A signalus ir juos nustatyti
;
```

```
;
;  Atmintis 2ah - 2dh rezervuota TRU moduliams
;

;TRU_bits1        data        2ah
;TRU_bits2        data        2bh
;TRU_bits3        data        2ch
;TRU_bits4        data        2dh

io_m_b           data        2eh
td0              bit         70h
td1              bit         71h
td2              bit         72h
td3              bit         73h
dtmf_out         bit         74h
enable_test      bit         75h
enable_tone      bit         76h
enable_tone_in   bit         77h

io_m_c           data        2fh
ext_led          bit         78h
enable_special   bit         79h
enable_busy      bit         7ah
enable_16h       bit         7bh
conn_loop        bit         7ch

;
;  kiti vidiniai kintamieji
;

looptest_timer   data        30h
looptest_timer2  data        31h
display_delay    data        32h
dsp_ptr          data        33h
```

```
msg_ptr          data        34h

;
;  sekantys baitai rezervuoti TRU moduliams kaip DATA
;
;TRU_byte0       data        38h
;TRU_byte1       data        39h
;TRU_byte2       data        3ah
;TRU_byte3       data        3bh
;TRU_byte4       data        3ch
;TRU_byte5       data        3dh
;TRU_byte6       data        3eh
;TRU_byte7       data        3fh

;  stekas užima baitus 40h- 5fh

;
; skambinimo skaičiai laikomi 60h - 7fh kartu su SND kodu
;
prefix           data        60h
first_digit      data        61h
second_digit     data        62h
third_digit      data        63h

;
;  8051 portu 1 ir 3 nustatymai
;

io_select        bit         p1.2
dt_pwm           bit         p1.4
twenty_hz_pwm    bit         p1.5
;power_fail_     bit         p1.6
lc_              bit         p3.2
ring_ground_     bit         p3.4
;strobe          bit         p3.5
```

```
a0              bit         p1.6
a1              bit         p3.5

;
;  įtampos padavimo įėjimas bei pertraukimo vektorių apdorojimas
;

                org         0000h
                jmp         init

                org         0003h
                jmp         offhook_edge
                db          0ffh, 0ffh, 0ffh, 0ffh, 0ffh

                org         000bh
                jmp         onhook_timer
                db          0ffh, 0ffh, 0ffh, 0ffh, 0ffh

                org         0013h
                jmp         int1_service
                db          0ffh, 0ffh, 0ffh, 0ffh, 0ffh

                org         001bh
                jmp         offhook_timer
                db          0ffh, 0ffh, 0ffh, 0ffh, 0ffh

                org         0023h
                jmp         ser_port_service
                db          0ffh, 0ffh, 0ffh, 0ffh, 0ffh

                org         002bh
                jmp         timer2_service
                db          0ffh, 0ffh, 0ffh, 0ffh, 0ffh

                org         0033h
```

```
                jmp         pca_service

                db          '(c) Copyright Codecom'
                db          '1992'
$include(main.msg)
$include(true.msg)

init:
                %set(initmask,0)

                mov         sp,#3fh
                clr         rs0
                clr         rs1
                mov         a,#0
                mov         r0,#0
fill_ram:
                mov         @r0,a
                inc         r0
                cjne        r0,#0,fill_ram
                clr         p1.2
                mov         r1,#main_flag
                mov         @r1,#0
                inc         r1
                mov         @r1,#0
                inc         r1
                mov         @r1,#0
                inc         r1
                mov         @r1,#0
                inc         r1

                call        reset_io_m
                mov         io_m_b,#0
                call        write_io_m_b
                mov         io_m_c,#0
```

```
                call    write_io_m_c

                call    reset_io
                mov     p2,#0f8h
                mov     px1_temp,#11111111b
                call    write_px1
                mov     px2_temp,#00000000b
                call    write_px2

                call    read_px2
                mov     tmod,#00000011b

                mov     ip,#01001010b
                mov     ie,#11001011b

                setb    it0
                setb    it1
                mov     cmod,#80h
                mov     ch,#0
                mov     c,#0
                mov     ccap01,#0ffh
                mov     ccap0h,#0
                mov     ccapm0,#049h
                mov     ccapm1,#0
                mov     ccapm2,#0
                clr     tr1
                clr     tr0

;   inicializuojami baitiniai kintamieji

                mov     time_on1,#0
                mov     time_on2,#0
                mov     time_off1,#0
                mov     time_off2,#0
                mov     time_off3,#0
```

```
                mov     flags,#0
                mov     flags2,#0
                mov     flags3,#0
                mov     inds,#0ffh
                mov     indasc,#0ffh
                mov     pulse_digit,#0
                mov     f_cmds,#0

;  inicializuojami bitiniai kintamieji

                clr     test_enable
                clr     test_end
                clr     error_flag
                clr     return_flag
                setb    fonhook
                clr     f_init
                clr     fremote_hup
                clr     fcall
                call    clr_dsp
                call    clr_msg
                call    specific_inits
                mov     ccap41,#0ffh
                mov     ccap4h,#0ffh
                mov     ccamp2,#48h
                orl     cmod,#40h
                setb    cr
                setb    tr0

; pralaukti 2.22 sek., išvalyti skambinimo skaičius, nustatyti R0
; 1 skaičių buferio pradžia, ir išvalyti R2 (skambinimo skaičių)
; skaitiklis

time_wakeup:
                call    watchdog
                mov     a,time_on2
```

```
                cjne    a,#255,time_wakeup
                mov     c,sw_gndstart
                mov     loopswitch_,c
                clr     no_ring
                call    write_px2
                jmp     start
$include(tru.asm)

;
; startas ir įėjimo taškas
;

start:
                mov     r0,#60h
                mov     a,#0ffh
clr_digbuf:
                mov     @r0,a
                inc     r0
                cjne    r0,#80h,clr_digbuf
                mov     r0,#60h
                mov     r2,#0
                setb    f_start
                clr     ie0
                setb    ex0

;  begalinis ciklas kol sulaukiamas skambutis

hookchk:
                call    watchdog
                jb      test_enable,do_test
                mov     f_cmds,#1
;               call    trucmd
                jb      fhook,first1
                jnb     fanswer,hhkchk
                jmp     ring_answer
```

```
first1:
                jmp         first
do_test:
                mov         r1,#main_flag_error
                mov         @1r,#0
                mov         r1,#main1_flag_error
                mov         @1r,#0
                clr         test_end
                clr         error_flag
                clr         error_flag
                mov         r1,#main_flag
                mov         a,@r1
                anl         a,#00000001b
                call        start_hook_test
test_hook:
                jb          error_flag,rou_error
                call        dial_test
test_dial:
                jb          error_flag,rou_error
                call        start_dtmf
test_dtmf:
                jb          error_flag,rou_error
                call        start_ring_test
test_ring:
                jb          error_flag,rou_error
                call        start_ring_ans_test
test_ring_a:
                jb          error_flag,rou_error
                call        make_call
                jb          error_flag,rou_error
                call        until_fin_tim
rou_error:
                call        error_acc
;               jmp         wait_end
until_fin_tim:
```

```
                setb    test_end
                clr     tr0
                mov     time_on2,#0
                mov     time_on1,#0
                mov     test_min,#0
                setb    tr0
wait_end:
                call    watchdog
                mov     a,test_min
                cjne    a,#23,wait_end
                call    fin_self
                jmp     hookchk

; startinis testas

start_hook_test:
                mov     r1,#main_flag
                mov     a,@r1
                orl     a,#00000001b
                mov     @r1,a
                setb    enable_test
                clr     conn_loop
                call    write_io_m_b
                call    reset_io_m
                call    write_io_m_b
                call    write_io_m_c
                clr     loopswitch_
                call    write_px2
test_hup:
                setb    fremote_hup
                call    hup
                mov     lamps_temp,#0
                call    write_io_c
                mov     r6,#10
nose:
```

```
                push        6
                mov         r6,#0ffh
loop_m:
                mov         r7,#0ffh
                djnz        r7,$
                call        watchdog
                djnz        r6,loop_m
                mov         r6,#0ffh
loop_ma:
                mov         r7,#0ffh
                djnz        r7,$
                call        watchdog
                djnz        r6,loop_ma
                mov         r6,#0ffh
loop_mb
                mov         r7,#0ffh
                djnz        r7,$
                call        watchdog
                djnz        r6,loop_mb
                mov         r6,#0ffh
loop_mc:
                mov         r7,#0ffh
                djnz        r7,$
                call        watchdog
                djnz        r6,loop_mc
                pop         6
                djnz        r6,nose
test_hup2:
                clr         fanswer
                clr         fhook
                jnb         fhook,not_hook_det
                jmp         hook_error
not_hook_det:
                setb        fonhook
                setb        conn_loop
```

```
                call    write_io_m_c
                mov     r6,#0ffh
loop_m_off:
                mov     r7,#0ffh
                djnz    r7,$
                call    watchdog
                djnz    r6,loop_m_off
                mov     r6,#0ffh
loop_m_off2:
                mov     r7,#0ffh
                djnz    r7,$
                call    watchdog
                djnz    r6,loop_m_off2
                mov     r6,#0ffh
loop_m_off3:
                mov     r7,#0ffh
                djnz    r7,$
                call    watchdog
                djnz    r6,loop_m_off3
                mov     r6,#0ffh
loop_m_off4:
                mov     r7,#0ffh
                djnz    r7,$
                call    watchdog
                djnz    r6,loop_m_off4
                mov     r6,#0ffh
loop_m_off5:
                mov     r7,#0ffh
                djnz    r7,$
                call    watchdog
                djnz    r6,loop_m_off5
test_fhook:
                jb      fhook,detected_hook
                jmp     hook_error
detected_hook:
```

```
                clr         ex0
                clr         hookswout
                setb        hookswout_
                clr         fhook
dt_delay1:
                call        watchdog
                jnb         lc_,still_off_hook
                jmp         hook_error
still_off_hook:
                jb          f_start,dt_delay1
                clr         ie0
                serb        ex0
                setb        fdtone
                call        dtone
                ret
hook_error:
                mov         r1,#main_flag_error
                mov         @r1,#r1
                setb        error_flag
                ret


;  skambinimo patikrinimas

dial_test:
                mov         r1,#main_flag
                mov         a,@r1
                orl         a,#00000010b
                mov         @r1,a
                mov         r6,#0ffh
delay_dial:
                mov         r7,#0ffh
                djnz        r7,$
                call        watchdog
                djnz        r6,delay_dial
```

```
                mov     r6,#0ffh
delay_dial2:
                mov     r7,#0ffh
                djnz    r7,$
                call    watchdog
                djnz    r6,delay_dial2
                clr     tr1
                mov     time_off1,#0
                mov     time_off2,#0
                setb    tr1
verify_dial:
                call    read_io_m_a
                anl     a,#00000001b
                cjne    a,#00000001b,dial_ok
                call    watchdog
                move    a,time_off2
                cjne    a,#10,verify_dial
                mov     r1,#main_flag_error
                mov     @r1,#3
                setb    error_flag
                ret
dial_ok:
                ret
;  dtmf startas

start_dtmf:
                clr     fdtone
                mov     ccapm1,#0
                anl     io_m_b,#0f0h
                orl     io_m_b,#cero
                call    send_dtmf
                call    delay_dtmf
                call    disable_dtmf
                call    delay_dtmf
                call    get_test
```

```
test_cero:
                cjne        a,#0h,dtmf_error_flag
                anl         io_m_b,#0f0h
                orl         io_m_b,#uno
                call        send_dtmf
                call        delay_dtmf
                call        disable_dtmf
                call        delay_dtmf
                call        get_test
test_1:
                cjne        a,#1h,dtmf_error_flag
                anl         io_m_b,#0f0h
                orl         io_m_b,#dos
                call        send_dtmf
                call        delay_dtmf
                call        disable_dtmf
                call        delay_dtmf
                call        get_test
test_2:
                cjne        a,#2h,dtmf_error_flag
                anl         io_m_b,#0f0h
                orl         io_m_b,#tres
                call        send_dtmf
                call        delay_dtmf
                call        disable_dtmf
                call        delay_dtmf
                call        get_test
test_3:
                cjne        a,#3h,dtmf_error_flag
                anl         io_m_b,#0f0h
                orl         io_m_b,#cuatro
                call        send_dtmf
                call        delay_dtmf
                call        disable_dtmf
```

```
                call        delay_dtmf
                call        get_test
test_4:
                cjne        a,#4h,dtmf_error_flag
                jmp         go_to_5
dtmf_error_flag:
                jmp         error-dtmf
go_to_5
                anl         io_m_b,#0f0h
                orl         io_m_b,#cinco
                call        send_dtmf
                call        delay_dtmf
                call        disable_dtmf
                call        delay_dtmf
                call        get_test
test_5:
                cjne        a,#5h,dtmf_error_flag
                anl         io_m_b,#0f0h
                orl         io_m_b,#seis
                call        send_dtmf
                call        delay_dtmf
                call        disable_dtmf
                call        delay_dtmf
                call        get_test
test_6:
                cjne        a,#6h,dtmf_error_flag
                anl         io_m_b,#0f0h
                orl         io_m_b,#siete
                call        send_dtmf
                call        delay_dtmf
                call        disable_dtmf
                call        delay_dtmf
                call        get_test
test_7:
                cjne        a,#7h,dtmf_error_flag
```

```
                anl         io_m_b,#0f0h
                orl         io_m_b,#ocho
                call        send_dtmf
                call        delay_dtmf
                call        disable_dtmf
                call        delay_dtmf
                call        get_test
test_8:
                cjne        a,#8h,dtmf_error_flag1
                anl         io_m_b,#0f0h
                orl         io_m_b,#nueve
                call        send_dtmf
                call        delay_dtmf
                call        disable_dtmf
                call        delay_dtmf
                call        get_test
test_9:
                cjne        a,#9h,dtmf_error_flag1
                anl         io_m_b,#0f0h
                orl         io_m_b,#05h
                call        send_dtmf
                call        delay_dtmf
                call        disable_dtmf
                call        delay_dtmf
                call        get_test

test_a:
                cjne        a,#0ah,dtmf_error_flag1
                anl         io_m_b,#0f0h
                orl         io_m_b,#0dh
                call        send_dtmf
                call        delay_dtmf
                call        disable_dtmf
                call        delay_dtmf
                call        get_test
```

```
test_b:
                cjne        a,#0h,dtmf_error_flag1
                jmb         go_to_c
dtmf_error_flag1:
                jmp         error_dtmf
go_to_c
                anl         io_m_b,#0f0h
                orl         io_m_b,#03h
                call        send_dtmf
                call        delay_dtmf
                call        disable_dtmf
                call        delay_dtmf
                call        get_test
test_c:
                cjne        a,#0h,dtmf_error_flag1
                anl         io_m_b,#0f0h
                orl         io_m_b,#0bh
                call        send_dtmf
                call        delay_dtmf
                call        disable_dtmf
                call        delay_dtmf
                call        get_test
test_d:
                cjne        a,#0h,dtmf_error_flag1
                anl         io_m_b,#0f0h
                orl         io_m_b,#07h
                call        send_dtmf
                call        delay_dtmf
                call        disable_dtmf
                call        delay_dtmf
                call        get_test
test_e:
                cjne        a,#0h,dtmf_error_flag1
                anl         io_m_b,#0f0h
                orl         io_m_b,#0fh
```

```
                call        send_dtmf
                call        delay_dtmf
                call        disable_dtmf
                call        delay_dtmf
                call        get_test
test_f:
                cjne        a,#0h,dtmf_error_flag1
                ret
error_dtmf:
                mov         r1,#main_flag_error
                mov         @r1,#4
                setb        error_flag
                ret

;  skambučio testo startas

start_ring_test:
                mov         r1,#main_flag
                mov         a,@r1
                orl         a,#00001000b
                mov         @r1,a
                clr         ring_second
                clr         conn_loop
                call        write_io_m_c
                setb        fremote_hup
                call        hup
                mov         r6,#6
 loop_ring_wait:
                push        6
                mov         r6,#0ffh
 loop_mbb:
                mov         r7,#0ffh
                djnz        r7,$
                call        watchdog
                djnz        r6,loop_mbb
```

```
                mov     r6,#0ffh
loop_mcc:
                mov     r7,#0ffh
                djnz    r7,$
                call    watchdog
                djnz    r6,loop_mcc
                mov     r6,#0ffh
                pop     6
                djnz    r6,loop_ring_wait
                clr     ap1
                jb      lc_,loop_ring_off
                jmp     ring_flag
loop_ring_off:
                clr     tr0
                mov     time_on2,#0
                mov     time_on1,#0
                setb    tr0
wait_2_sec:
                call    watchdog
                jnb     fanswer,not_answer
                jmp     ring_flag
not_answer:
                call    watchdog
                anl     a,#00000100b
                cjne    a,#00000100b,ring_starts
                mov     a,time_on2
                cjne    a,#200,wait_2_sec
                jmp     ring_flag
ring_starts:
                mov     r6,#0ffh
loop_mbbc:
                mov     r7,#offh
                djnz    r7,$
                call    watchdog
                djnz    r6,loop_mbbc
```

```
                mov         r6,#0ffh
loop_mccd:
                mov         r7,#0ffh
                djnz        r7,$
                call        watchdog
                djnz        r6,loop_mccd
ring_start:
                clr         tr0
                mov         time_on2,#0
                mov         time_on1,#0
                setb        tr0
wait_130msec:
                call        watchdog
                jnb         fanswer,not_answer1
                jmp         ring-flag
not_answer1:
                call        read_io_m_a
test_130:
                anl         a,#00000100b
                cjne        a,#00000100b,ring_expire
                jmp         ring_flag
ring_expire:
                mov         a,time_on2
                cjne        a,#90,wait_130msec
wait_520msec:
                call        watchdog
                call        read_io_m_a
test_250:
                anl         a,#00000100b
                cjne        a,#00000100b,ring_not_exp
                            ire
ring_not_expire:
                mov         a,time_on2
                cjne        a,#250,wait_250msec
test_flag:
```

```
                jmp         ring_flag
ring_stop_:
                clr         tr0
                mov         time_on2,#0
                mov         time_on1,#0
                setb        tr0
wait_2_seca:
                call        watchdog
                jnb         fanswer,not_answer2
                jmp         ring_flag
not_answer2:
                mov         a,time_on2
                cjne        a,#255,wait_2_seca
                jb          ring_second,ring_stop1
                setb        ring_second
                jmp         loop_ring_off
ring_flag:
                setb        ap1
                mov         r1,#main_flag_error
                mov         @r1,#5
                setb        error_flag
ring_stop1:
test_timbre:
                ret
;  atsakyti į skambutį

start_ring_ans_test:
                mov         r1,#main_flag
                mov         a,@r1
                orl         a,00010000b
                mov         @r1,a
loop_ring_off2:
                clr         tr0
                mov         time_on2,#0
                mov         time_on1,#0
```

```
                setb        tr0
wait_2_sec2:
                call        watchdog
                jnb         fanswer,not_answera
                jmp         ans_flag
not_answera:
                call        read_io_m_a
                anl         a,#00000100b
                anl         a,#00000100b,ring_present
                mov         a,time_on2
                cjne        a,#200,wait_2_sec2
                jmp         ans_flag
ring_present:
                setb        conn_loop
                call        write_io_m_c
                mov         r7,#0ffh
wait_ans:
                mov         r6,#0ffh
                djnz        r6,$
                call        watchdog
                djnz        r7,wait_ans
                mov         r7,#0ffh
wait_ansa:
                mov         r6,#0ffh
                djnz        r6,$
                call        watchdog
                djnz        r7,wait_ansa
test_ans:
                jb          fanswer,ok_ring
                jmp         ans_flag
ans_flag:
                mov         r1,#main_flag_error
                mov         @r1,#6
                setb        error_flag
                setb        ap1
```

```
                clr         fanswer
                ret
ok_ring:
timbre:
                clr         fanswer
                setb        ap1
                ret

;  šaukinys radijui patikrinti

make_call:
                mov         r1,#main1_flag
                mov         a,@r1
                orl         a,#000000001b
                mov         @r1,a
test_mute:
                jnb         power_,power_is_ok
                mov         r1,#main_flag_error
                mov         @r1,#1
                setb        error_flag
                jmp         pre-onhook
power_is_ok:
                mov         a,#func
                call        wrbus
                mov         a,#7
                call        wrbus
                call        clr_dsp
                mov         a,#star
                call        wrbus
                mov         r7,#0ffh
wait_min:
                mov         r6,#0ffh
                djnz        r6,$
                call        watchdog
                djnz        r7,wait_min
```

```
                mov     r7,#0ffh
wait_mina:
                mov     r6,#0ffh
                djnz    r6,$
                call    watchdog
                djnz    r7,wait_mina
                mov     a,#9h
                call    wrbus
                mov     a,#0ah
                call    wrbus

                mov     a,#dspbuf_start
                ass     a,#3
                mov     r1,a
                mov     a,@r1
                anl     a,#0fh
                call    wrbus

                inc     r1
                mov     a,@r1
                anl     a,#0fh
                call    wrbus

                inc     r1
                mov     a,@r1
                anl     a,#0fh
                call    wrbus

                inc     r1
                mov     a,@r1
                anl     a,#0fh
                call    wrbus

                inc     r1
                mov     a,@r1
```

```
                anl         a,#0fh
                call        wrbus

                inc         r1
                mov         a,@r1
                anl         a,#0fh
                call        wrbus

                inc         r1
                mov         a,@r1
                anl         a,#0fh
                call        wrbus

                setb        fsend
                mov         a,#send
                call        wrbus
                setb        enable_special
                call        write_io_m_c
test_mute1:
                clr         mute
                call        write_px2
                clr         tr1
                mov         time_off1,#0
                mov         time_off2,#0
                setb        tr1
delay_call:
                mov         a,time_off2
                call        watchdog
                cjne        a,#80,delay_call
test_mute2:
                clr         tr1
                mov         time_off1,#0
                mov         time_off2,#0
                mov         time_off3,#0
                setb        tr1
```

```
cont_waiting:
                jnb         inuse,still_inuse
                mov         r1,#main1_flag_error
                mov         @r1,#4
                jmp         call_error
still_inuse:
                call        read_io_m_a
                anl         a,#00000001b
                cnje        a,#00000001b,busy_ok
                call        read_io_m_a
                anl         a,#00001000b
                cnje        a,#00001000b,t1000hz
                mov         a,time_off2
                call        watchdog
                cjne        a,#1h,cont_waiting
                mov         r1,#main1_flag_error
                mov         @r1,#3
call_error
                mov         a,#end1
                call        wrbus
                setb        error_flag
                jmp         pre_onhook
t1000hz:
busy_ok:
                mov         a,#end1
                call        wrbus
pre_onhook:
                clr         conn_loop
                call        write_io_m_b
                call        hup
                ret

;  klaidos kodo nustatymas

error_acc:
```

```
                    mov     r1,main_flag_error
                    mov     a,@r1
                    cjne    a,#1,check_hook_ring
                    mov     r1,#error_code
                    mov     @r1,#1
                    jmp     exit_error
check_hook_ring:
                    cjne    a,#2,check_dial_flag
                    mov     r1,#error_code
                    mov     @r1,#2
                    jmp     exit_error
check_dial_flag:
                    cjne    a,#3,check_dtmf_flag
                    mov     r1,#error_code
                    mov     @r1,#3
                    jmp     exit_error
check_dtmf_ring:
                    cjne    a,#4,check_ring_flag
                    mov     r1,#error_code
                    mov     @r1,#4
                    jmp     exit_error
check_ring_flag:
                    cjne    a,#5,check_ring_ans_flag
                    mov     r1,#error_code
                    mov     @r1,#5
                    jmp     exit_error
check_ring_ans_flag:
                    cjne    a,#6,check_power_flag
                    mov     r1,#error_code
                    mov     @r1,#6
                    jmp     exit_error
check_power_flag:
                    mov     r1,#main1_flag_error
                    mov     a,@r1
```

```
                    cjne        a,#1,check_call_test
                    mov         r1,#error_code
                    mov         @r1,#7
                    jmp         exit_error
check_call_test:
                    cjne        a,#2,check_busy_test
                    mov         r1,#error_code
                    mov         @r1,#8
                    jmp         exit_error
check_busy_test:
                    cjne        a,#3,check_inuse_test
                    mov         r1,#error_code
                    mov         @r1,#9
                    jmp         exit_error
check_inuse_test:
                    cjne        a,#4,check_1000_test
                    mov         r1,#error_code
                    mov         @r1,#10
                    jmp         exit_error
check_1000_test:
                    cjne        a,#5,exit_error
                    mov         r1,#error_code
                    mov         @r1,#11
                    jmp         exit_error
exit_error:
                    ret

;  grįžti į normalią būseną

fin_self:
                    clr         conn_loop
                    call        write_io_m_c
                    call        hup
                    clr         enable_test
                    call        write_io_m_b
```

```
                clr     test_enable
                clr     test_end
                clr     error_flag
                jnb     error_flag,turn_on_led
                clr     ext_led
                call    write_io_m_c
                call    loop_bad
turn_on_led:
                setb    ext_led
                call    write_io_m_c
                call    loop_bad

                setb    ext_led
                call    write_io_m_c
                call    loop_bad

                setb    ext_led
                call    write_io_m_c
                call    loop_bad

                setb    ext_led
                call    write_io_m_c
                call    loop_bad

                setb    ext_led
                call    write_io_m_c
                call    loop_bad
test_4on:
                clr     ext_led
                call    write_io_m_c
                call    loop_bad
                ret
loop_bad:
                mov     r7,#0ffh
wait_emb:
```

```
                mov     r6,#0ffh
                djnz    r6,$
                call    watchdog
                djnz    r7,wait_emb
                mov     r7,#0ffh
wait_emba:
                mov     r6,#0ffh
                djnz    r6,$
                call    watchdog
                djnz    r7,wait_emba
                ret

;  ateiti čia po skambinimo paruošimo

first:
                call    clr_msg
                clr     ex0
                clr     hookswout
                setb    hookswout_
                clr     fhook
dt_delay:
                call    watchdog
                mov     f_cmds,#4
                call    trucmd
                jb      f_start,dt_delay
                clr     ie0
test_lc:
                jnb     lc_,alert_check
test_lc1:
                call    hup
                jmp     start
alert_check:
                setb    ex0
                jb      inuse,make_dt
                jmp     wait
```

```
; 500 ms generuojamas skambinimo tonas

make_dt:
                    setb        fdtone
                    call        dtone
                    call        get
                    jb          inuse,check_prefix
                    clr         fdtone
                    mov         ccapm1,#0
                    jmp         wait
check_prefix:
                    jb          lock,prefix_0
                    cjne        a,#pound,nosend
prefix_0:
                    cjne        a,#zero,prefix_1
                    jmp         store_it
prefix_1:
                    cjne        a,#1,no_prefix
                    jmp         store_it
no_prefix:
                    inc         r0
                    inc         r1
;  išsaugoti skaičių ir gauti kitą

store_it:
                    cjne        r0,#7fh,do_store_it
                    jmp         check_number
do_store_it:
                    mov         @r0,a
                    inc         r0
;  patikrinti įvestus skaičius ir juos išanalizuoti

check_number:
                    clr         fsendtimer
                    mov         a,first_digit
```

```
                cjne        a,#0ffh,check_one
                jmp         check_send
check_one:
                cjne        a,#pound,check_send
                mov         a,second_digit
                cjne        a,#0ffh,check_two
                jmp         check_send
check_two:
                cjne        a,#pound,check_send
                mov         a,third_digit
                cjne        a,#pound,check_unlock
                jnb         lock,nosend
                jmp         lock_it
check_nam:
                cjne        a,#start,check_send
                mov         a,third_digit
                cjne        a,#0ffh,check_nam1
                jmp         check_send
check_nam1:
                mov         f_cmds,#2
                call        trucmd

;  užbaigti esant blogoms skaičių sekoms

nosend:
                clr         funlocktimer
                setb        fonhook
                setb        ie0
                jmp         start
check_unlock:
                cjne        a,#0ffh,unlocking_it
                jmp         check_send
unlocking_it:
                setb        funlocktimer
                jmp         get_next
```

```
;  jeigu skaičių seka neatitinka specialių funkcijų

check_send:
                    jb          sw_sendtimer,check_int1
                    setb        fsendtimer
                    jmp         get_next
check_int1:
                    mov         a,prefix
                    cjne        a,#zero,check_op
                    mov         a,first_digit
                    cjne        a,#1,check_op
                    setb        fsendtimer
                    jmp         get_next
check_op:
                    mov         a,prefix
                    cjne        a,#10,check_service
                    cjne        r2,#0,check_service
                    setb        fsendtimer
check_service:
                    cjne        r2,#3,check_distance
                    mov         a,second_digit
                    cjne        a,#1,check_distance
                    mov         a,third_digit
                    cjne        a,#1,check_distance
                    jmp         send_it
check_distance:
                    cjne        r2,#10,check_local
                    jmp         send_it
check_local:
                    cjne        r2,#7,get_next
                    setb        fsendtimer
                    mov         a,second_digit
                    subb        a,#1
                    jz          get_next
```

```
                    mov         a,second_digit
                    subb        a,#10
                    jz          get_next
                    clr         fsendtimer
                    jmp         send_it
get_next:
                    inc         r2
                    call        get
                    jmp         store_it

;
;  skambinimo tonas, pasiųsti numerį
;
;patikrinti TRU ir baigti darbą, jei atblokavimo kodas neįvestas

send_it:
                    jb          lock,send1
locked:
                    mov         a,third_digit
                    cjne        a,#ffh,send1

;  pasiųsti skambinimo numerį

send1:
                    setb        mute
                    call        write_px2
                    mov         @r0,#send
                    mov         a,#clear
                    call        wrbus
                    mov         r0,#60h
                    mov         a,@r0
                    cjne        a,#0ffh,out
                    inc         r0
out:
                    mov         a,@r0
```

```
                    call        wrbus
                    jnb         fhang,send_ok
test_lc2:
                    call        hup
                    jmp         start
send_ok:
                    inc         r0
                    cjne        a,#send,out
                    setb        fsend
                    clr         mute
                    call        write_px2

;  skambinimas padarytas, laukti kitų skaičių

wait:
                    call        get
                    jnb         sw_data,wait
                    call        wrbus
                    jmp         wait

;
;  skambinimo tono generatorius
;

;generuojamas reikalingas tonas

dtone:
                    push        psw
                    jb          lock,serv_check
                    mov         dptr,#table_350
                    jmp         tone_out
serv_check:
                    jb          noserv,roam_check
                    mov         dptr,#table_620
                    jmp         tone_out
```

```
roam_check:
                    jb          sw_dialtone,normal
                    jb          roam,normal
                    mov         dptr,#table_roam
                    jmp         tone_out
normal:
                    mov         dptr,#table_dt
tone_out:
                    mov         lowpoint,dpl
                    mov         highpoint,dph
                    mov         ccapm1,#01000010b
                    pop         psw
                    ret
;
;  gauti skaičius iš klaviatūros
;

;  įvertinama gaunami skaičiai ir nustatymai

get:
                    push        psw
get1:
                    mov         f_cmds,#3
                    call        trucmd
                    call        watchdog

;  laukti pulsinio režimo

get_pulse:
                    jnb         fdigit,get_dtmf
                    clr         fdtone
                    mov         ccapm1,#0
                    mov         a,#2
                    clr         c
                    subb        a,time_off2
```

```
                jnc     get_dtmf
                mov     a,#131
                clr     c
                subb    a,time_off1
                jnc     get_dtmf
                mov     a,pulse_digit
                mov     pulse_digit,#0
                setb    fdig_ready
                clr     fdigit

;  patikrinti specialias sąlygas

get_dtmf:
                jnb     fdtmfin,hang
                clr     fdtone
                mov     ccapm1,#0
                clr     fdtmfin
                mov     a,px1_temp
                anl     a,#0fh
                setb    fdig_ready

;  patikrinti nutraukimą

hang:
                jnb     fhang,gflash
                pop     psw
test_lc3:
                call    hup
                dec     sp
                dec     sp
                jnb     fremote_hup,no_remote
                clr     fremote_hup
no_remote:
                jmp     start
```

```
; patikrinti mirksėjimą

gflash:
                    jnb         fflash,time
                    clr         fflash
                    mov         ccapm1,#0
                    jb          fdtone,just_flash
                    jb          fsend,just_flash
                    clr         fdtone
                    pop         psw
                    dec         sp
                    dec         sp
                    jamp        send_it

; generuoti SEND toną

just_flash:
                    clr         fdtone
                    mov         f_cmds,#5
                    call        trucmd
                    mov         a,#send
                    call        wrbus
                    setb        fsend
                    pop         psw
                    dec         sp
                    dec         sp
                    jmp         wait

;  patikrinti surinkimo užlaikymą

time:
                    jnb         ftime,send_time
                    pop         psw
                    dec         sp
                    dec         sp
```

```
                    jmp         rohtone

    ; patikrinti ar siuntimo laikas

    send_time:
                    jnb         fsendtimer,unlock_time
                    mov         r6,time_off2
                    cjne        r6,#56,umlock_time
                    clr         fsendtimer
                    pop         psw
                    dec         sp
                    dec         sp
                    jmp         send_it

    ;  patikrinti atblokavimą

    unlock_time:
                    jnb         funlocktimer,lock_time
                    mov         r6,time_off2
                    cjne        r6,#28,look
                    clr         funlocktimer
                    pop         psw
                    dec         sp
                    dec         sp
                    jb          lock,no_lock
                    jmp         unlock_it
    no_unlock:
                    setb        fonhook
                    setb        ie0
                    jmp         start

    ;  mažiau kaip 4 skaičių blokavimas

    lock_time:
```

```
                    jnb         flocktimer,look
                    mov         r6,time_off2
                    cjne        r6,#28,look
                    clr         flocktimer
                    pop         psw
                    dec         sp
                    dec         sp
                    jmp         go_lock

;  patikrinti ar pilnas skaičius

look:
                    jb          fdig_ready,got_a_digit
                    jmp         get1

;  gautas skaičius

got_a_digit:
                    clr         fdig_ready
got_inuse:
                    move        time_off1,#0
                    move        time_off2,#0
                    move        time_off3,#0
                    pop         psw
                    ret

;
;  gauti skaičių iš klaviatūros
;

;  įvertinama gaunami skaičiai ir nustatymai

get_test:
                    push        psw
get_dtmf_test:
```

```
                    jnb         fdtmfin,no_dtmf_test
                    clr         fdtone
                    mov         ccapm1,#0
                    clr         fdtmfin
                    mov         a,px1_temp
                    anl         a,#0fh
                    setb        fdig_ready
no_dtmf_test:
                    clr         fdig_ready
                    mov         time_off1,#0
                    mov         time_off2,#0
                    mov         time_off3,#0
                    pop         psw
                    ret

;  pasiųsti skambinimo toną

send_dtmf:
                    call        watchdog
                    clr         dtmf_out
                    call        write_io_m_b
                    setb        dtmf_out
                    call        write_io_m_b
                    ret

;  pasiųsti skambinimo toną

disable_dtmf:
                    clr         dtmf_out
                    call        write_io_m_b
                    ret
delay_dtmf:
                    clr         tr1
                    mov         time_off1,#0
                    mov         time_off2,#0
```

```
                    setb        tr1
wait_until_dtmf:
                    mov         a,time_off2
                    call        watchdog
                    cjne        a,#4,wait_until_dtmf
                    ret

;
;  ROH tono generatorius
;
rohtone:
                    clr         fdtime
                    clr         fdtone
                    mov         dptr,#table_400
                    mov         highpoint,dph
                    mov         lowpoint,dpl
                    setb        roh_boost
                    call        write_px2
rohtone_on:
                    mov         ccapm1,#01000010b
check_hangup:
                    call        watchdog
                    jnb         fhang,check_timeout
test_lc4
                    call        hup
                    jmp         start
check_timeout:
                    jb          frohtimeout,roh_timeout
                    jb          roh_on,rohtone_on
                    mov         ccapm1,#0
                    jmp         check_hangup
roh_timeout:
                    mov         ccapm1,#0
                    clr         roh_boost
                    setb        loopswitch_
```

```
                call        write_px2
test_lc5:
                call        hup
                jnb         sw_gndstart,looptest
                jmp         looptest_end
looptest:
                mov         b,#40
loopt1:
                mov         a,time_on2
                add         a,#5
loopt2:
                call        watchdog
                cjne        a,time_on2,loopt2
                djnz        b,loopt1
                clr         loopswitch_
                call        write_px2
                mov         looptest_timer2,#215
loopt3:
                mov         looptest_timer2,#0ffh
                djnz        looptest_timer,$
                call        watchdog
                djnz        looptest_timer2,loopt3
                jb          lc_,looptest_end
                setb        loopswitch_
                call        write_px2
                jmp         looptest
looptest_end:
                clr         frohtimeout
                mov         time_on1,#0
                mov         time_on2,#0
                jmp         start
;
;  nutraukimo paprogramė
;
hup:
```

```
                call    watchdog
                clr     ex0
                clr     fdigit
                mov     pulse_digit,#0
                clr     fdig_ready
                clr     fdtmfin
                clr     flocktimer
                clr     funlocktimer
                jnb     sw_gndstart,hup1
                setb    loopswitch_
                call    write_px2
                mov     a,time_on2
                add     a,#75
                push    acc
                jmp     hup2
hup1:
                jb      frohtimeout,hup2
                clr     loopswitch_
                call    write_px2
hup2:
                clr     ftime
                clr     fsendtimer
                mov     ccapm1,#0
                clr     fdtone
                clr     roh_boost
                call    write_px2
                jb      fremote_hup,no_end_sent
                mov     a,#endcall
                call    wrbus
no_end_sent:
                clr     fsend
                setb    hookswout
                clr     hookswout_
                jnb     sw_gndstart,hup4
                pop     acc
```

```
                    clr         c
hup3:
                    call        watchdog
                    cjne        a,time_on2,$+3
                    jnc         hup3
hup4:
                    setb        fonhook
                    clr         fhang
                    clr         f_one_sec
                    clr         f_inuseofftiming
                    setb        ex0
                    ret

;
;  TRU būsenos pakeitimas
;

update_displays:
                    push        acc
                    push        psw
upspec1:
                    jnb         fspec1,continue_update
                    call        spec1
continue_update:
                    jb          test_enable,go_to_self
                    mov         a,lamps_temp
                    mov         c,roam
                    cpl         c
                    mov         roamlamp,c
                    mov         c,noserv
                    cpl         c
                    mov         noservlamp,c
                    mov         c,lock
                    cpl         c
                    mov         locklamp,c
```

```
                    mov         c,inuse
                    cpl         c
                    mov         inuselamp,c
                    call        writw_io_c
                    jb          sw_dialtone,not_maint
test_push:
                    call        read_io_m_a
                    anl         a,#00010000b
                    cjne        a,#0h,not_push
                    setb        test_enable
                    setb        fremote_hup
                    setb        loopswitch_
                    call        write_px2
not_push:
                    jnb         test_enable,not_maint
go_to_self:
                    jnb         test_end,not_error_yet
                    jnb         error_flag,pass_ok
test_led:
                    mov         r0,#error_code
                    mov         a,@r0
                    rl          a
                    rl          a
                    rl          a
                    rl          a
                    mov         lamps_temp,a
                    call        write_io_c
                    setb        ext_led
                    jmp         write_to_c
pass_ok:
                    clr         ext_led
                    jmp         write_to_c
not_error_yet:
                    jb          ext_led,clr_led
                    setb        ext_led
```

```
                jmp         write_to_c
clr_led:
                clr         ext_led
write_to_c:
                call        write_io_m_c
not_maint:
                mov         a, lamps_temp
                jb          frohtimeout, compare
                mov         c, ap2
                anl         c, ap1
                jnc         compare
                jnb         fonhook, compare
                clr         loopswitch_
                call        write_px2
                jb          fcall, reset_ring_timer
                setb        fcall
                mov         bell_timer, #0
                mov         bell_timer2, #0
                setb        fbellsound
                mov         dptr, #table_20
                mov         lowpoint, dpl
                mow         highpoint, dph
                setb        twenty_hz_pwm
                mov         ccapm2, #01000010b
test1:
                clr         ring_
                setb        no_ring
                call        write_px2
reset_ring_timer:
                mov         ring_timer, #0
compare:
                xrl         a, lamps_temp
                jnb         f_one_sec, dtbits
                mov         c, acc.7
                anl         c, inuselamp
```

```
                    jnc         timing_in_use_off_
                    mov         in_use_off_timer, #28
                    setb        f_inuseofftiming
timing_in_use_off:
                    jnb         inuselamp,
                                cont_in_use_timing
                    clr         f_inuseofftiming
cont_in_use_timing:
                    jnb         f_inuseofftiming, dtbits
                    djnz        in_use_off_timer, dtbits
                    clr         f_inuseofftiming
                    jb          fonhook, dtbits
disconnect:
                    setb        fremote_hup
                    setb        loopswitch_
                    call        write_px2
dtbits:
                    anl         a, #01110000b
                    jz          ring_timeout
                    jnb         fdtone, ring_timeout
                    mov         ccamp1, #0
                    call        dtone
ring_timeout:
                    jnb         fcall, update_done
                    inc         ring_timer
                    mov         a, #53
                    clr         c
                    subb        a, ring_timer
                    jnc         update_done
                    jb          fbellsound,
                                update_done
check_stop_ring_ps:
                    mov         a, #3
                    clr         c
                    subb        a, bell_timer2
```

```
                    jnc         update_done
                    mov         ccapm2, #0
                    clr         twenty_hz_pwm
                    clr         fcall
                    jnb         sw_gndstart, update_done
                    setb        loopswitch_
                    call        write_px2
update_done:
                    jb          fcall, no_reset_io
                    jnb         lc_, no_reset_io
                    jb          frohtimeout, no_reset_io
no_reset_io
                    pop         psw
                    pop         acc
                    ret

;  perkrovimo patikrinimas

watchdog:
                    push        ie
                    clr         ea
                    mov         ccap41, #0
                    mov         ccap4h, ch
                    pop         ie
                    ret

;
;  taimerio pertraukimų apdorojimo programa
;

onhook_timer:
                    push        psw
                    push        acc
                    clr         tr1
                    clr         tf1
```

```
mute_check:
                    jb          mute, on_spec2
                    setb        mute
                    call        write_px2
on_spec2:
                    jnb         fspec2, on_spec2_done
                    call        spec2
on_spec2_done:
                    djnz        display_delay, gs_init
                    call        update_displays
gs_init:
                    jb          fcall, cont_service
                    jnb         sw_gndstart, cont_service
                    jnb         fonhook, cont_service
                    jb        ring_ground_, cont_groundstart
                    clr         loopswitch_
                    call        write_px2
                    mov         gndstart_timer1, #0
                    mov         gndstart_timer2, #0
cont_groundstart:
                    inc         gndstart_timer1
                    mov         a, gndstart_timer1
                    cjne        a, #90, cont_service
                    inc         gndstart_timer2
                    mov         gndstart_timer1, #0
                    mov         a, gndstart_timer2
                    cjne        a, #40, cont_service
                    setb        loopswitch_
                    call        write_px2
cont_service:
                    jb          fcall, ring_control
                    jmp         onhook_timers
ring_control:
                    jb          fbellsound, timing_ring
                    jb          ring_ground_, timing_ring
```

```
                    mov         a, bell_timer2
                    cjne        a, #0, start_bellsound
                    mov         a, #28
                    clr         c
                    subb        a, bell_timer
                    jnc         timing_ring
                    jmp         start_bellsound
timing_ring
                    inc         bell_timer
                    mov         a, bell_timer
                    cjne        a, #90, ringer_on
                    mov         bell_timer, #0
                    inc         bell_timer2
ringer_on
                    jnb         fbellsound, ringer_off
check_stop_bellsound:
                    mov         a, bell_timer2
                    cjne        a, #59, onhook_timers
                    mov         a, bell_timer
                    cjne        a, #45, onhook_timers
stop_bellsound:
                    mov         ccapm2, #0
                    clr         fbellsound
                    mov         dptr, #table_null
                    mov         lowpoint, dpl
                    mov         highpoint, dph
                    mov         ccamp2, #01000010b
                    clr         ea
                    clr         no_ring
                    call        write_px2
                    setb        ea
                    call        watchdog
                    jmp         reset_bell_timers
ringer_off:
                    mov         a, bell_timer2
```

```
                    cjne        a, #120, onhook_timers
start_bellsound:
                    call        reset_io
                    call        write_px1
                    call        write_px2
                    clr         ea
                    setb        no_ring
                    call        write_px2
                    setb        ea
                    mov         ccapm2, #0
                    mov         dptr,#table_20
                    mov         lowpoint, dpl
                    mov         highpoint, dph
                    mov         ccapm2, #01000010b
test_relay1:
                    setb        fbellsound
reset_bell_timers:
                    mov         bell_timer, #0
                    mov         bell_timer2, #0
onhook_timers:
                    inc         time_on1
                    mov         a,time_on1
                    cjne        a,#36,onhook_done
                    inc         time_on2
                    move        time_on1,#0
                    mov         a,time_on2
                    cjne        a,#0,not_inc255
                    inc         test_min
not_inc255:
                    jb          frohtimeout, onhook_done
test_fhang1:
                    jb          fonhook, onhook_done
test_fhang2:
                    mov         a, time_on2
                    cjne        a, #30, onhook_done
```

```
test_fhang3:
                    setb        fhang
                    clr         f_inuseofftiming
onhook_done:
                    pop         acc
                    pop         psw
                    reti
;
; 1 taimerio pertraukimo apdorojimas
;
offhook_timer
                    push        psw
                    push        acc
                    jnb         fspec2,check_strobe
                    call        spec2
check_strobe:
                    call        read_px1_strobe
                    jb          strobe,decode_dtmf
                    clr         fdecoder_busy
                    jmp         ttime
decode_dtmf:
                    jb          fdecoder_busy,ttime
                    call        read_px1
                    setb        fdtmfin
                    setb        fdecoder_busy
ttime:
                    inc         time_off1
                    mov         a,time_off1
                    cjne        a,#0,t_one_sec
                    inc         time_off2
                    call        update_displays
                    mov         a,time_off2
                    cjne        a,#0,t_one_sec
                    inc         time_off3
t_one_sec:
```

```
                mov         a,time_off2
                cjne        a,#14,trohtimer
                setb        f_one_sec
trohtimer:
                jb          fsend,tdone
                jnb         inuse,tdone
                inc         r5
                cjne        r5,#36,trohstart
                inc         r7
                mov         r5,#0
                cjne        r7,#10,trohstart
                cpl         roh_on
                mov         r7,#0
trohstart:
                mov         a,time_off3
                cjne        a,#2,trohquit
                mov         a,time_off2
                cjne        a,#51,trohquit
                setb        ftime
trohquit:
                mov         a,time_off3
                cjne        a,#5,tdone
                mov         a,time_off2
                cjne        a,#127,tdone
                setb        frohtimeout
tdone:
                jnb         f-start,tdone0
                mov         a,time_off2
                cjne        a,#4,tdone1
                clr         f_start
tdone0:
                jnb         lc_,tdone1
test_tr0:
                clr         tr1
                setb        tr0
```

```
tdone1:
                    pop         acc
                    pop         psw
                    reti
;
;  išorinio pertraukimo aptarnavimo programa
;
offhook_edge:
                    jb          frohtimeout,offhook_return1
                    jnb         fcall,not_inc_call
                    jnb         no_ring,ok_wait
                    setb        ring_
                    call        write_px2
                    clr         no_ring
                    call        write_px2
                    setb        return_ring
                    clr         ie0
                    clr         tr0
                    clr         tf0
ok_wait:
                    mov         r7,#0ffh
loop_ring:
                    mov         r6,#0ffh
                    djnz        r6,$
                    call        watchdog
                    djnz        r7,loop_ring
                    jnb         lc_,no_set
esta_set:
                    jnb         return_ring,do_not_return
                    call        reset_io
                    call        write_px1
                    call        write_px2
                    setb        no_ring
                    clr         ring_
                    call        write_px2
```

```
                clr         return_ring
                mov         r7,#07fh
loop_ring1
                mov         r6,#0ffh
                djnz        r6,$
                call        watchdog
                djnz        r7,loop_ring1
do_not_return:
                call        reset_io
                call        write_px1
                call        write_px2
                setb        tr0
                mov         t10,#0
                reti
offhook_return1:
                jmp         offhook_return
no_set:
not_inc_call:
                clr         tr0
                clr         tf0
                push        psw
                push        acc
                clr         ie0
                setb        tr1
                clr         fhang
                mov         th0,#0
                mov         time_off1,#0
                mov         time_off2,#0
                mov         time_off3,#0
                clr         mute
                call        write_px2
                jnb         fonhook,check_flash
                clr         fonhook
ok_off_hook:
                jnb         fcall,set_hook_flag
```

```
                setb        fanswer
                clr         fcall
                setb        ring_
                call        write_px2
                clr         no_ring
                call        write_px2
                mov         ccapm 2,#0
                clr         twenty_hz_pwm
                jmp         offhook_done
set_hook_flag:
                setb        fhook
                jmp         offhook_done
check_flash:
                mov         a,#19
                clr         c
                subb        a,time_on2
                jnc         check_digit
                jnb         lock,offhook_done
                setb        fflash
                jmp         offhook_done
check_digit:
                mov         a,#1
                clr         c
                subb        a,time_on2
                jnc         offhook_done
                inc         pulse_digit
                setb        fdigit
offhook_done:
                mov         time_on1,#0
                mov         time_on2,#0
                mov         t10,#0
                pop         acc
                pop         psw
offhook_return:
                reti
```

```
;
;  PCA pertraukimų apdorojimas
;
$ge
pca_service:
                    %if(%intmask ne 0) then (
                    push            ie
                    mov             ie,#%intmask
                    call            masklabel
                    push            acc
                    push            psw
                    mov             a,ccapm1
                    orl             a,ccapm2
                    anl             a,#00000010b
                    jz              pca_done
do_pca:
                    mov             a,#0
read_table:
                    movc            a,@a+dptr
                    cjne            a,#255,write_tone
                    mov             dph,highpoint
                    mov             dpl,lowpoint
                    mov             a,#0
                    jmp             read_table
write_tone:
                    mov             ccap1h,a
                    mov             ccap2h,a
                    inc             dptr
pca_done:
                    clr             ccf0
                    clr             ea
                    mov             ccap01,#0ffh
                    mov             a,ccap0h
                    inc             a
                    mov             ccap0h,a
```

```
                setb    ea
                pop     psw
                pop     acc
                pop     ie
                ret
masklabel:
                reti
                )else(
                push    acc
                push    psw
                mov     a,ccapm1
                orl     a,ccapm2
                anl     a,#00000010b
                jz      pca_done
do_pca:
                mov     a,#0
read_table:
                movc    a,@a+dptr
                cjne    a,#255,write_tone
                mov     dph,highpoint
                mov     dpl,lowpoint
                mov     a,#0
                jmp     read_table
write_tone:
                mov     ccap1h,a
                mov     ccap2h,a
                inc     dptr
pca_done:
                clr     ccf0
                clr     ea
                mov     ccap01,#0ffh
                mov     a,ccap0h
                inc     a
                mov     ccap0h,a
                setb    ea
```

```
                    pop         psw
                    pop         acc
                    reti
                    ) fi
$noge
;
;  išvedimo/įvedimo palaikymas
;
write_px1:
                    push        ie
                    clr         ea
                    clr         io_select
                    push        acc
                    push        0
                    setb        a0
                    clr         a1
                    mov         a,px1_temp
                    anl         a,#11110000b
                    mov         b,a
                    mov         a,px2_temp
                    anl         a,#00001111b
                    orl         a,b
                    mov         r0,#00h
                    movx        @r0,a
                    setb        a0
                    setb        a1
                    pop         0
                    pop         acc
                    pop         ie
                    ret
read_px1:
                    push        ie
                    clr         ea
                    clr         io_select
                    push        acc
```

```
                push        0
                clr         a0
                clr         a1
                mov         r0,#0h
                movx        a,@r0
                anl         a,#00001111b
                anl         px1_temp,#11110000b
                orl         px1_temp,a
                setb        a0
                setb        a1
                pop         0
                pop         acc
                pop         ie
                ret
read_px1_strobe:
                push        ie
                clr         ea
                clr         io_select
                push        acc
                push        0
                clr         a0
                clr         a1
                mov         r0,#0h
                movx        a,@r0
                mov         io_status,a
                setb        a0
                setb        a1
                pop         0
                pop         acc
                pop         ie
                ret

write_px2:
                push        ie
                clr         ea
```

```
                clr         io_select
                push        acc
                push        0
                setb        a0
                clr         a1
                mov         a,px1_temp
                anl         a,#11110000b
                mov         b,a
                mov         a,px2_temp
                anl         a,#00001111b
                orl         a,b
                mov         r0,#00h
                movx        @r0,a
                setb        a0
                setb        a1
                pop         0
                pop         acc
                pop         ie
                ret
write_io_c:
                push        ie
                clr         ea
                clr         io_select
                push        acc
                push        0
                clr         a0
                setb        a1
                mov         a,lamps_temp
                anl         a,#11110000b
                rl          a
                rl          a
                rl          a
                rl          a
                mov         r0,#00h
                movx        @r0,a
```

```
                setb    a0
                setb    a1
                pop     0
                pop     acc
                pop     ie
                ret
read_px2:
                push    ie
                clr     ea
                clr     io_select
                push    acc
                push    0
                clr     a0
                setb    a1
                mov     r0,#0h
                movx    a,@r0
                anl     a,#11110000b
                anl     px2_temp,#00001111b
                orl     px2_temp,a
                setb    a0
                setb    a1
                pop     0
                pop     acc
                pop     ie
                ret
read_io_m_a:
                push    ie
                clr     ea
                setb    io_select
                push    acc
                push    0
                clr     a0
                clr     a1
                mov     r0,#0h
                movx    a,@r0
```

```
                setb    a0
                setb    a1
                clr     io_select
                pop     0
                pop     acc
                pop     ie
                ret
write_io_m_b:
                push    ie
                setb    io_select
                clr     ea
                push    acc
                push    0
                setb    a0
                clr     a1
                mov     a,io_m_b
                mov     r0,#00h
                movx    @r0,a
                setb    a0
                setb    a1
                clr     io_select
                pop     0
                pop     acc
                pop     ie
                ret
write_io_m_c:
                push    ie
                setb    io_select
                clr     ea
                push    acc
                push    0
                clr     a0
                setb    a1
                mov     a,io_m_c
                mov     r0,#00h
```

```
                movx    @r0,a
                setb    a0
                setb    a1
                clr     io_select
                pop     0
                pop     acc
                pop     ie
                ret
reset_io_m:
                push    ie
                clr     ea
                setb    io_select
                push    acc
                push    0
                setb    a0
                setb    a1
                mov     r0,#0h
                mov     a,#10010000b
                movx    @r0,a
                setb    a0
                setb    a1
                clr     io_select
                pop     0
                pop     acc
                pop     ie
                ret
reset_io:
                push    ie
                clr     ea
                setb    io_select
                push    acc
                push    0
                setb    a0
                setb    a1
                mov     r0,#0h
```

```
                mov         a,#10011000b
                movx        @r0,a
                setb        a0
                setb        a1
                clr         io_select
                pop         0
                pop         acc
                pop         ie
                ret

;  papildomos programos

clr_msg:
                push        1
                mov         r1,#msgbuf_start
                mov         msg_ptr,r1
clr_msg1:
                mov         @r1,#55h
                inc         r1
                cjne        r1,#msgbuf_end,clr_msg1
                pop         1
                ret
savemg:
                push        1
                mov         r1,msg_ptr
                cjne        r1,#msgbuf_end,saveml
                mov         r1,#msgbuf_start
saveml:         mov         @r1,a
                inc         r1
                mov         msg_ptr,r1
                pop         1
exit:
                ret
clr_dsp:
                push        1
```

```
                    mov         r1,#dspbuf_start
                    mov         dsp_ptr,r1
clr_dsp1:
                    mov         @r1,#0ffh
                    inc         r1
                    cjne        r1,#dspbuf_end,clr_dsp1
                    pop         1
                    ret
save_dsp:
                    push        1
                    mov         r1,dsp_ptr
                    cjne        r1,#dspbuf_end,save_ds
                                p1
                    mov         r1,#dspbuf_start
save_dsp1:
                    mov         2r1,a
                    inc         r1
                    mov         dsp_ptr,r1
                    pop         1
save_dsp2:
;include(SPAIN.asm)
                    ret
$nolist
$include(SPAIN4.asm)
$list
                    nop
                    nop
                    end;
```

IŠRADIMO APIBRĖŽTIS

1.Celiuliarinis ar celiuliarinio tipo perdavimo aparatas, turintis celiuliarinį ar celiuliarinio tipo siųstuvą-imtuvą, celiuliarinį ar celiuliarinio tipo interfeisinį įrenginį ir standartinį telefoninį rinkinį, prijungtą prie minėto celiuliarinio, ar celiuliarinio tipo siųstuvo-imtuvo, kai minėtas interfeisinis įrenginys konvertuoja DTMF ar pulsinio tipo skambinimo signalus į skaitmeninę formą siuntimui į minėtą celiuliarinį ar celiuliarinio tipo siųstuvą-imtuvą, kai skambina iš telefono, formuojant šaukimą celiuliarinėje ar celiuliarinio tipo sistemoje, b e - s i s k i r i a n t i s tuo, kad turi:

autodiagnostines priemones celiuliarinio, ar celiuliarinio tipo siųstuvo-imtuvo ir celiuliarinio ar celiuliarinio tipo interfeisinio įrenginio funkcionavimui stebėti ir pranešti,

priemones papildomam minėtų autodiagnostinių priemonių prijungimui prie celiuliarinio ar celiuliarinio tipo interfeisinio įrenginio,

priemones papildomam prijungimui minėtų autodiagnostinių priemonių, turinčias priemones standartinio telefoninio rinkinio atjungimui nuo celiuliarinio ar celiuliarinio tipo interfeisinio irenginio, kai autodiagnostinės priemonės yra prijungtos prie celiuliarinio ar celiuliarinio tipo interfeisinio įrenginio,

autodiagnostines priemones, turinčias priemones, galinčias simuliuoti standartinio telefoninio rinkinio funkcijas, nustatant ar celiuliarinis ar celiuliarinio tipo interfeisinis įrenginys yra funkcionalus.

2.Celiuliarinis perdavimo aparatas pagal 1 punktą, b e s i s k i r i a n t i s tuo, kad standartinio telefoninio aparato funkcijas simuliuojančios priemonės turi priemones pakėlimo signalo generavimui celiuliariniam ar celiuliarinio tipo interfeisiniam įrenginiui.

3.Celiuliarinis perdavimo aparatas pagal 2 punktą, b e s i s k i r i a n t i s tuo, kad autodiagnostinės priemonės turi priemones skambinimo tono detektavimui, kurį sugeneruoja celiuliarinis ar celiuliarinio tipo interfeisinis įrenginys, atsakydamas į pakėlimo signalo sugeneravimą.

4.Celiuliarinis perdavimo aparatas pagal 1 punktą, b e s i s k i r i a n t i s tuo, kad standartinio telefoninio aparato funkcijas simuliuojančios priemonės turi priemones DTMF signalo generavimui ir jo siuntimui į celiuliarinį ar celiuliarinio tipo interfeisinį įrenginį.

5.Celiuliarinis perdavimo aparatas pagal 4 punktą, b e s i s k i r i a n t i s tuo, kad autodiagnostinės priemonės turi priemones DTMF signalo detektavimui, kurį sugeneruoja celiuliarinis ar celiuliarinio tipo interfeisinis įrenginys, atsakydamas į priemonių sugeneruotą DTMF signalą.

6.Celiuliarinis perdavimo aparatas pagal 1 punktą, b e s i s k i r i a n t i s tuo, kad standartinio telefoninio aparato funkcijas simuliuojančios priemonės turi priemones sujungimo signalo generavimui ir jo siuntimui į celiuliarinį ar celiuliarinio tipo interfeisinį įrenginį.

7.Celiuliarinis perdavimo aparatas pagal 6 punktą, b e s i s k i r i a n t i s tuo, kad autodiagnostinės

priemonės turi priemones išorinio skambinimo generavimui, siekiant paleisti celiuliarinio ar celiuliarinio tipo interfeisinio įrenginio skambučio generatorių, bei priemones šio sugeneruoto skambučio detektavimui.

8.Celiuliarinis perdavimo aparatas pagal 7 punktą, b e s i s k i r i a n t i s tuo, kad standartinio telefoninio aparato funkcijas simuliuojančios priemonės turi priemones padėjimo signalo generavimui ir jo siuntimui į minėtą celiuliarinį ar celiuliarinio tipo interfeisinį įrenginį, kai celiuliarinio tipo interfeisinis įrenginys generuoja skambučius, siekiant nustatyti ar minėtas celiuliarinis ar celiuliarinio tipo interfeisinis įrenginys teisingai atjungs skambučio signalą, atėjus telefoniniam kvietimui.

9.Celiuliarinis perdavimo aparatas pagal 1 punktą, b e s i s k i r i a n t i s tuo, kad autodiagnostinės priemonės turi priemones išeinančiam skambinimui generuoti per celiuliarinį ar celiuliarinio tipo interfeisinį įrenginį ir siųstuvą-imtuvą, per celiuliarinę ar celiuliarinio tipo telefoninę sistemą, skambinimui grįžtant atgal į celiuliarinį ar celiuliarinio tipo siųstuvą-imtuvą, bei minėtos autodiagnostinės priemonės turi priemones užimtumo signalo, generuojamo minėto celiuliarinio ar celiuliarinio tipo siųstuvo-imtuvo, atsakant į atėjusį skambutį, detektavimu, ir priemones išeinančiam skambinimui generuoti numeriu, kuris priskirtas minėtam siųstuvui-imtuvui, prie kurio jos yra prijungtos.

10.Testavimo aparatas celiuliarinio ar celiuliarinio tipo interfeisinio įrenginio darbingumo tikrinimui, turintis celiuliarinį ar celiuliarinio tipo interfeisinį įrenginį, prijungtą prie standartinio telefoninio rinkinio ar telefoninio tipo prietaiso, ir

interfeisinis įrenginys gali konvertuoti prijungto telefoninio tipo prietaiso DTMF ar pulsinio tipo skambinimo signalus į skaitmeninę formą jų siuntimui į celiuliarinį ar celiuliarinio tipo siųstuvą-imtuvą, b e s i s k i r i a n t i s tuo, kad turi diagnostines priemones celiuliarinio ar celiuliarinio tipo interfeisinio įrenginio funkcionavimui stebėti ir pranešti.

11.Testavimo aparatas pagal 10 punktą, b e s i s k i - r i a n t i s tuo, kad turi priemones papildomam prijungimui diagnostinių priemonių prie minėto celiuliarinio ar celiuliarinio tipo interfeisinio tipo įrenginio, ir diagnostinės priemonės turi priemones, galinčias simuliuoti standartinio telefoninio tipo įrenginio funkcijas, nustatant ar celiuliarinis ar celiuliarinio tipo interfeisinis įrenginys yra funkcionalus.

12.Testavimo aparatas pagal 11 punktą, b e s i s k i - r i a n t i s tuo, kad minėtos priemonės papildomam prijungimui, turi priemones telefoninio tipo įrenginio atjungimui nuo celiuliarinio ar celiuliarinio tipo interfeisinio įrenginio, kai diagnostinės priemonės yra prijungtos prie celiuliarinio ar celiuliarinio tipo interfeisinio įrenginio.

13.Testavimo aparatas pagal 11 punktą, b e s i s k i - r i a n t i s tuo, kad priemonės galinčios simuliuoti standartinio telefoninio tipo įrenginio funkcijas, turi priemones pakėlimo signalo generavimui minėtam celiuliariniam ar celiuliarinio tipo įrenginiui.

14.Testavimo aparatas pagal 13 punktą, b e s i s k i - r i a n t i s tuo, kad diagnostinės priemonės turi priemones skambinimo tono detektavimui, kurį generuoja

celiuliarinis ar celiuliarinio tipo interfeisinis įrenginys, atsakydamas į pakėlimo signalo generavimą.

15.Testavimo aparatas pagal 11 punktą, b e s i s k i r i a n t i s tuo, kad telefoninio tipo įtaiso funkcijas stimuliuojančios priemonės turi priemones DTMF signalo generavimui ir jo siuntimui į celiuliarinio ar celiuliarinio tipo interfeisinį įrenginį.

16.Testavimo aparatas pagal 15 punktą, b e s i s k i r i a n t i s tuo, kad diagnostinės priemonės turi priemones DTMF signalo detektavimui, kurį generuoja celiuliarinis ar celiuliarinio tipo interfeisinis įrenginys, atsakydamas į minėtų priemonių generuotą DTMF signalą.

17.Testavimo aparatas pagal 15 punktą, b e s i s k i r i a n t i s tuo, kad telefoninio tipo įtaiso funkcijas simuliuojančios priemonės turi priemones sujungimo signalo generavimui ir jo siuntimui į celiuliarinį ar celiuliarinio tipo interfeisinį įrenginį.

18.Testavimo aparatas pagal 10 punktą, b e s i s k i r i a n t i s tuo, kad telefoninio tipo įtaiso funkcijas simuliuojančios priemonės turi priemones ateinančio šaukimo signalo generavimui ir jo siuntimui į minėtą siųstuvą-imtuvą, siekiant nustatyti ar celiuliarinis ar celiuliarinio tipo interfeisinis įrenginys įjungs savo skambučio generatorių, bei diagnostines priemones, įjungiančias skambučio signalo detektorių.

19.Testavimo aparatas pagal 18 punktą, b e s i s k i r i a n t i s tuo, kad diagnostinės priemonės turi priemones padėjimo signalo generavimui ir jo siuntimui

į celiuliarinį ar celiuliarinio tipo interfeisinį įrenginį, kai celiuliarinis ar celiuliarinio tipo interfeisinis įrenginys generuoja skambučius, siekiant nustatyti ar celiuliarinis ar celiuliarinio tipo interfeisinis įrenginys teisingai atjungs skambučio signalą atėjus telefoniniam kvietimui.

20.Testavimo aparatas pafal 10 punktą, b e s i s k i r i a n t i s tuo, kad diagnostinės priemonės turi priemones išeinančiam skambinimui per celiuliarinį ar celiuliarinio tipo telefoninį tinklą, kai celiuliarinis ar celiuliarinio tipo interfeisinis įrenginys yra prijungtas prie siųstuvo-imtuvo, ko pasekoje vyksta skambinimas sau pačiam, ir minėtos diagnostinės priemonės turi priemones užimtumo signalo, generuojamo celiuliarinio ar celiuliarinio tipo siųstuvo-imtuvo, atsakant į atėjusį skambutį, detektavimui ir priemones išeinančiam skambinimui generuoti numeriu, kuris priskirtas minėtam siųstuvui-imtuvui, prie kurio jos yra prijungtos.

21.Testavimo būdas tikrinti diagnostinio įrenginio priemonėmis celiuliarinio tipo interfeisinio įrenginio darbingumą, kai celiuliarinis ar celiuliarinio tipo interfeisinis įrenginys yra prijungiamas prie standartinio telefoninio rinkinio ar telefoninio tipo prietaiso, o interfeisinis įrenginys gali konvertuoti DTMF ar pulsinio tipo skambinimo signalus iš prijungto telefono tipo prietaiso į skaitmeninę formą jų siuntimui į celiuliarinio ar celiuliarinio tipo siųstuvą-imtuvą, b e s i s k i r i a n t i s tuo, kad įjungia minėto celiuliarinio ar celiuliarinio tipo interfeisinio įrenginio funkcionavimo stebėjimą ir ataskaitų siuntimą.

22.Testavimo būdas pagal 21 punktą, b e s i s k i r i a n t i s tuo, kad prijungia minėtas diagnostines

priemones prie celiuliarinio ar celiuliarinio tipo interfeisinio įrenginio, bei prijungia diagnostines priemones, galinčias simuliuoti standartinio telefoninio tipo įrenginio funkcijas, nustatant ar celiuliarinis ar celiuliarinio tipo interfeisinis įrenginys yra funkcionalus.

23.Testavimo būdas pagal 22 punktą, b e s i s k i r i a n t i s tuo, kad atjungia telefoninio tipo įrenginį nuo minėto celiuliarinio ar celiuliarinio tipo interfeisinio įrenginio, kai minėtos diagnostinės priemonės yra prijungiamos prie celiuliarinio ar celiuliarinio tipo interfeisinio įrenginio.

24.Testavimo būdas pagal 22 punktą, b e s i s k i r i a n t i s tuo, kad generuoja pakėlimo signalą celiuliariniam ar celiuliarinio tipo interfeisinio tipo interfeisiniam įrenginiui.

25.Testavimo būdas pagal 24 punktą, b e s i s k i r i a n t i s tuo, kad detektuoja skambinimo toną, kurį sugeneruoja minėtas celiuliarinis ar celiuliarinio tipo interfeisinis įrenginys, atsakydamas į minėto pakėlimo signalo sugeneravimą.

26.Testavimo būdas pagal 22 punktą, b e s i s k i r i a n t i s tuo, kad generuoja DTMF signalą ir siunčia į celiuliarinio ar celiuliarinio tipo interfeisinį įrenginį.

27.Testavimo būdas pagal 26 punktą, b e s i s k i r i a n t i s tuo, kad detektuoja DTMF signalą, kurį sugeneruoja minėtas celiuliarinis ar celiuliarinio tipo interfeisinis įrenginys, atsakydamas į minėtų priemonių sugeneruotą DTMF signalą.

28.Testavimo būdas pagal 26 punktą, b e s i s k i - r i a n t i s tuo, kad generuoja sujungimo signalą ir ją siunčia į minėtą celiuliarinį ar celiuliarinio tipo interfeisinį įrenginį.

29.Testavimo būdas pagal 21 punktą, b e s i s k i - r i a n t i s tuo, kad generuoja ateinantį šaukimo signalą ir ją siunčia į minėtą siųstuvą-imtuvą, siekiant nustatyti ar celiuliarinis ar celiuliarinio tipo interfeisinis įrenginys įjungs savo skambučio generatorių, bei detektuoja skambučio signalą.

30.Testavimo būdas pagal 29 punktą, b e s i s k i - r i a n t i s tuo, kad generuoja padėjimo signalą ir ją siunčia į minėtą celiuliarinį ar celiuliarinio tipo interfeisinį įrenginį, kai minėtas celiuliarinis ar celiuliarinio tipo interfeisinis įrenginys generuoja skambučius, siekiant nustatyti ar celiuliarinis ar celiuliarinio tipo interfeisinis įrenginys teisingai atjungs skambučio signalą atėjus telefoniniam kvietimui.

31.Testavimo būdas pagal 21 punktą, b e s i s k i - r i a n t i s tuo, kad generuoja išeinantį skambinimą per celiuliarinį ar celiuliarinio tipo telefoninį tinklą, kai celiuliarinis ar celiuliarinio tipo interfeisinis įrenginys yra prijungtas prie siųstuvo-imtuvo, ko pasekoje vyksta skambinimas sau pačiam, ir detektuojamas užimtumo signalas, generuojamas minėtam celiuliarinio ar celiuliarinio tipo perdavėjui atsakant į skambutį.

32.Celiuliarinio perdavimo aparatas, turintis radijo siųstuvą-imtuvą, interfeisinį įrenginį, prijungtą prie minėto siųstuvo-imtuvo, kai minėtas interfeisinis įrenginys prijungia minėtą komunikacinį įrenginį prie minėto radijo siųstuvo-imtuvo kviečiant ar atsakant per

minėtą radijo siųstuvą-imtuvą, b e s i s k i r i a n t i s tuo, kad turi autodiagnostines priemones minėto siųstuvo-imtuvo ir interfeisinio įrenginio funkcionavimui stebėti ir pranešti, priemones papildomam prijungimui minėtų autodiagnostinių priemonių prie minėto interfeisinio įrenginio, priemones papildomam prijungimui minėtų autodiagnostinių priemonių, turinčias priemones minėto komunikacinio įrenginio atjungimui nuo minėto interfeisinio įrenginio, kai autodiagnostinės priemonės yra prijungiamos prie interfeisinio įrenginio, minėtas autodiagnostines priemones turinčias priemones, galinčias simuliuoti komunikacinio įrenginio funkcijas, nustatant ar interfeisinis įrenginys yra funkcionalus.

33.Celiuliarinis perdavimo aparatas pagal 32 punktą, b e s i s k i r i a n i s tuo, kad komunikacinio įrenginio funkcijas simuliuojančios priemonės turi priemones pakėlimo signalo generavimui minėtam interfeisiniam įrenginiui.

34.Celiuliarinis perdavimo aparatas pagal 33 punktą, b e s i s k i r i a n t i s tuo, kad autodiagnostinės priemonės turi priemones skambinimo tono detektavimui, kurį sugeneruoja interfeisinis įrenginys, atsakydamas į pakėlimo signalo sugeneravimą.

35.Celiuliarinis perdavimo aparatas pagal 32 punktą, b e s i s k i r i a n t i s tuo, kad komunikacinio įrenginio funkcijas simuliuojančios priemonės turi priemones DTMF signalo generavimui ir jo siuntimui į interfeisinį įrenginį.

36.Celiuliarinis perdavimo aparatas pagal 35 punktą, b e s i s k i r i a n t i s tuo, kad minėtos autodiagnostinės priemonės turi priemones DTMF signalo

detektavimui, kurį sugeneruoja interfeisinis įrenginys atsakydamas į priemonių sugeneruotą DTMF signalą.

37.Celiuliarinis perdavimo aparatas pagal 32 punktą, b e s i s k i r i a n t i s tuo, kad autodiagnostinės priemonės turi priemones išorinio skambinimo generavimui, siekiant paleisti interfeisinio įrenginio skambučio generatorių, bei priemones šio sugeneruoto skambučio detektavimui.

38.Celiuliarinis perdavimo aparatas pagal 32 punktą, b e s i s k i r i a n t i s tuo, kad autodiagnostinės priemonės turi priemones išeinančiam skambinimui generuoti per interfeisinį įrenginį ir siųstuvą-imtuvą, per radijo telefono sistemą, kuriai priklauso siųstuvas-imtuvas, skambinimui generuoti per interfeisinį įrenginį ir siųstuvą-imtuvą, per radijo telefono sistemą, kuriai priklauso siųstuvas-imtuvas, skambinimui grįžtant atgal į siųstuvą-imtuvą, bei autodiagnostinės priemonės įjungia priemones užimtumo signalo, generuojamo minėto siųstuvo-imtuvo, atsakant į skambutį, detektavimui, ir priemones išeinančiam skambinimui generuoti numeriu, kuris priskirtas minėtam siųstuvui-imtuvui, prie kurio jos yra prijungtos.

39.Testavimo aparatas tikrinti interfeisinio įrenginio darbingumą, turintis interfeisinį įrenginį , prijungtą prie komunikacinio įrenginio, prijungto prie radijo siųstuvo-imtuvo gaunant ir siunčiant pranešimus per radijo siųstuvą-imtuvą, b e s i s k i r i a n t i s tuo, kad turi diagnostines priemones minėto interfeisinio įrenginio funkcionavimui stebėti ir pranešti.

40.Testavimo aparatas pagal 39 punktą, b e s i s k i r i a n t i s tuo, kad turi priemones papildomam prijungimui diagnostinių priemonių prie interfeisinio

įrenginio, diagnostines priemones, įjungiančias priemones, galinčias simuliuoti komunikacinio įrenginio funkcijas, nustatant ar interfeisinis įrenginys yra funkcionalus.

41.Testavimo būdas tikrinti interfeisinio įrenginio darbingumą diagnostinio įtaiso pagalba, kai interfeisinį įrenginį prijungia tarp komunikacinio įrenginio ir radijo siųstuvo-imtuvo, gaunant ir siunčiant pranešimus radijo siųstuvo-imtuvo pagalba, b e s i s k i r i a n t i s tuo, kad stebi ir praneša apie interfeisinio įrenginio funkcionavimą.

42.Testavimo būdas pagal 41 punktą, b e s i s k i r i a n t i s tuo, kad prijungia diagnostines priemones prie interfeisinio įrenginio, bei prijungia diagnostines priemones, galinčias simuliuoti komunikacinio įrenginio funkcijas, nustatant ar interfeisinis įrenginys yra funkcionalus.

43.Testavimo būdas pagal 42 punktą, b e s i s k i r i a n t i s tuo, kad atjungia komunikacinį įrenginį nuo interfeisinio įrenginio, kai diagnostinės priemonės yra prijungiamos prie interfeisinio įrenginio.

D0_TEST funkcija

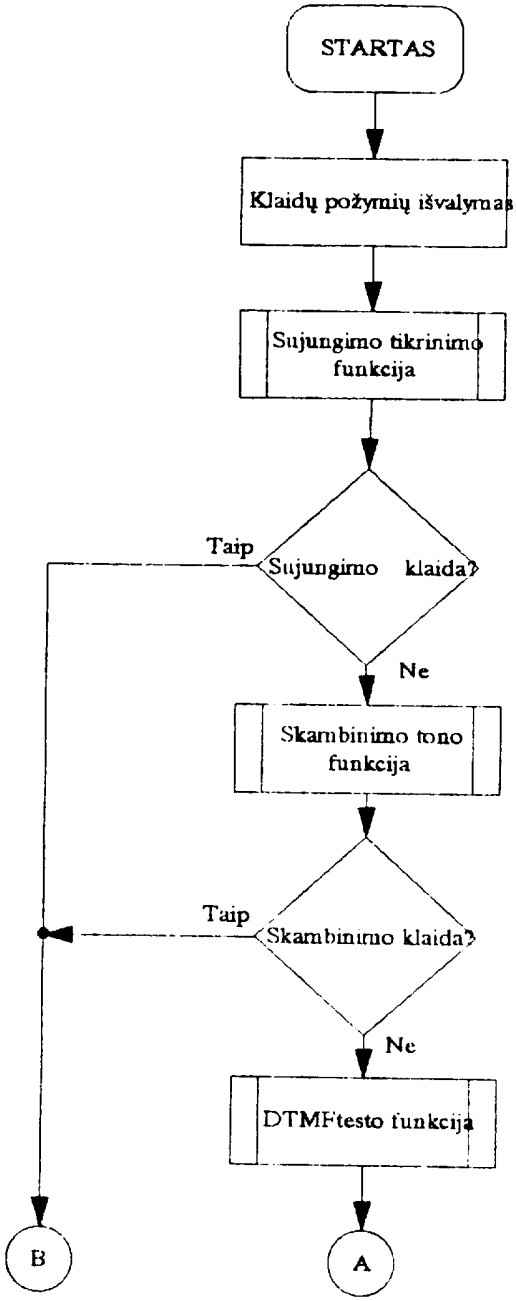

Fig.1A

D0_TEST funkcija (tęsinys)

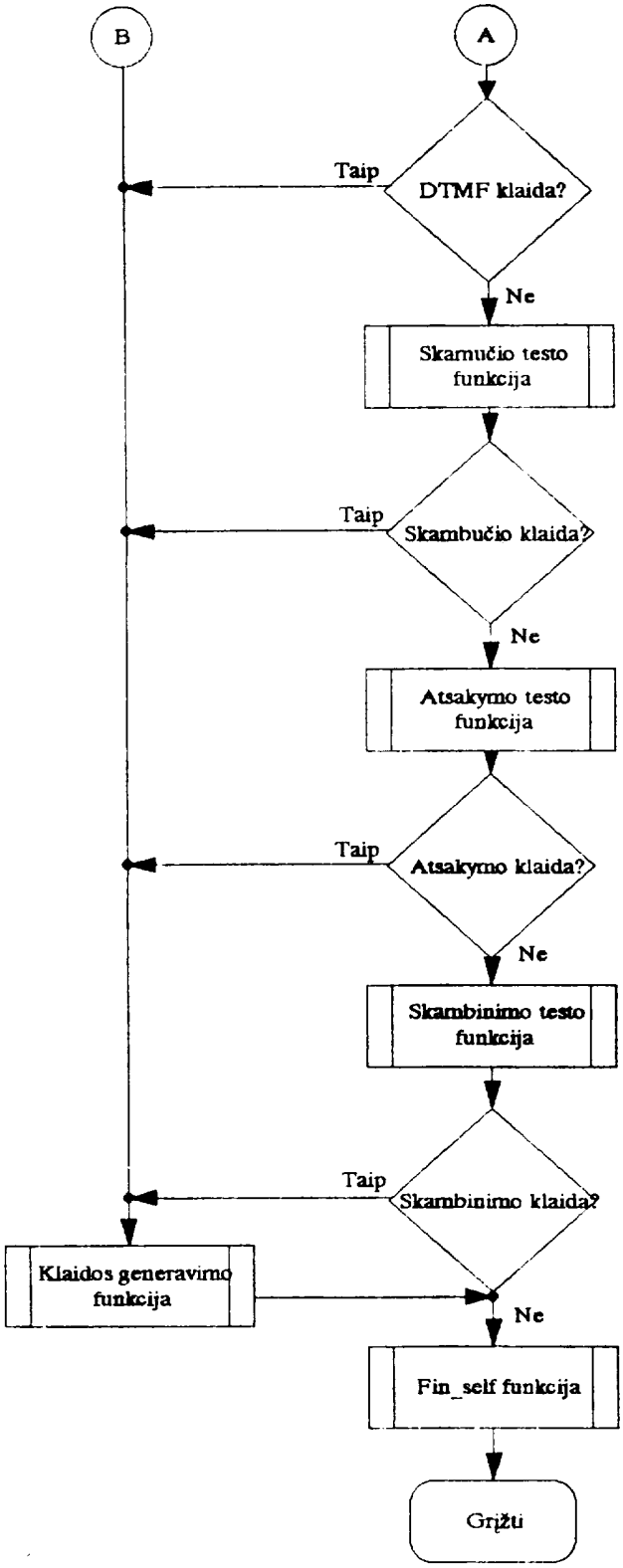

Fig.1B

HOOK_TEST funkcija

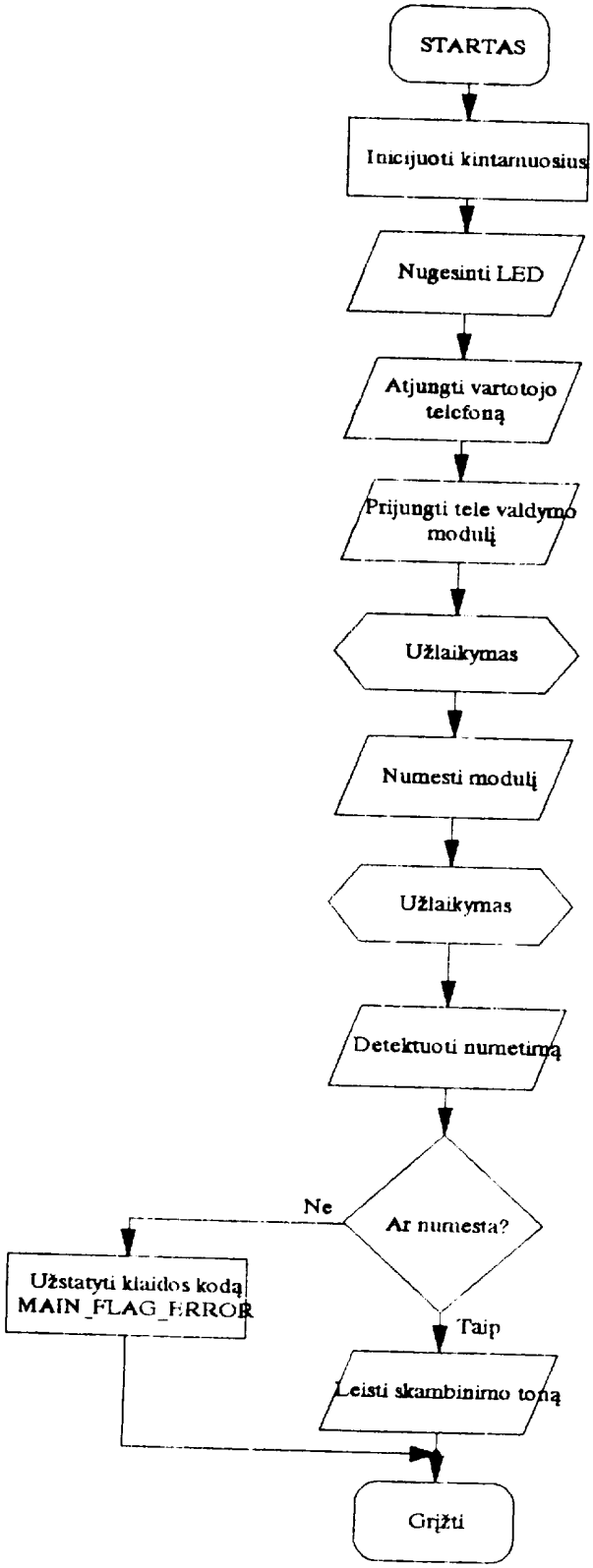

Fig.2

DIAL_TEST funkcija

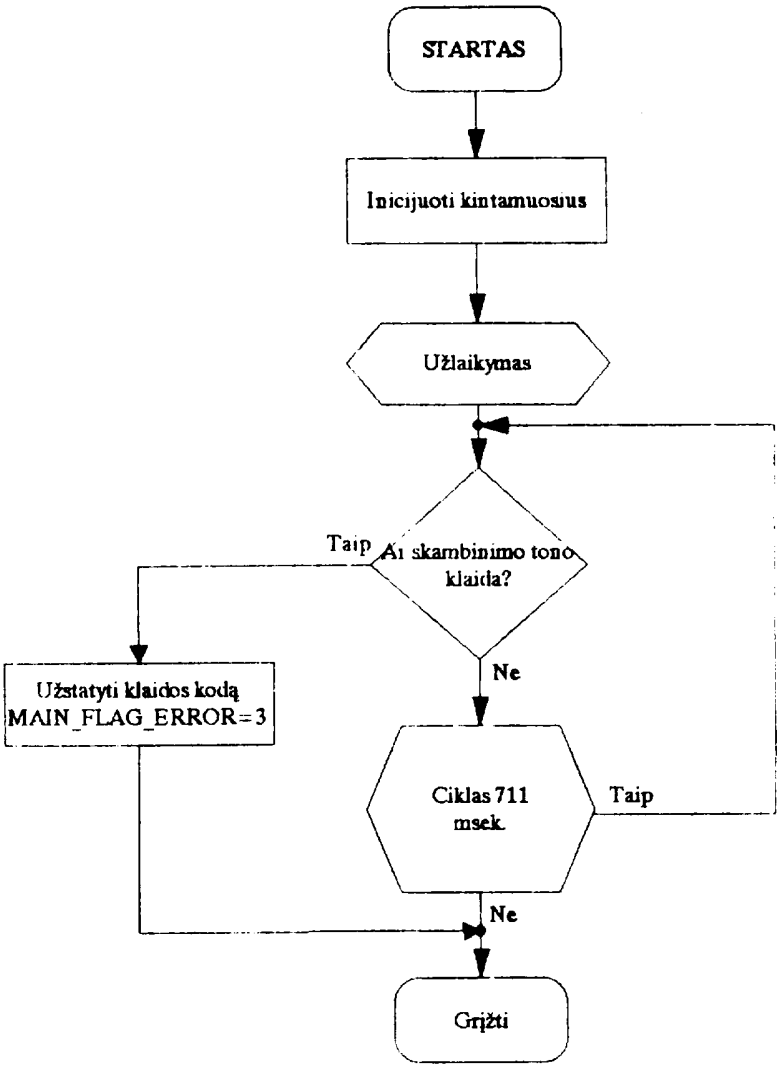

Fig.3

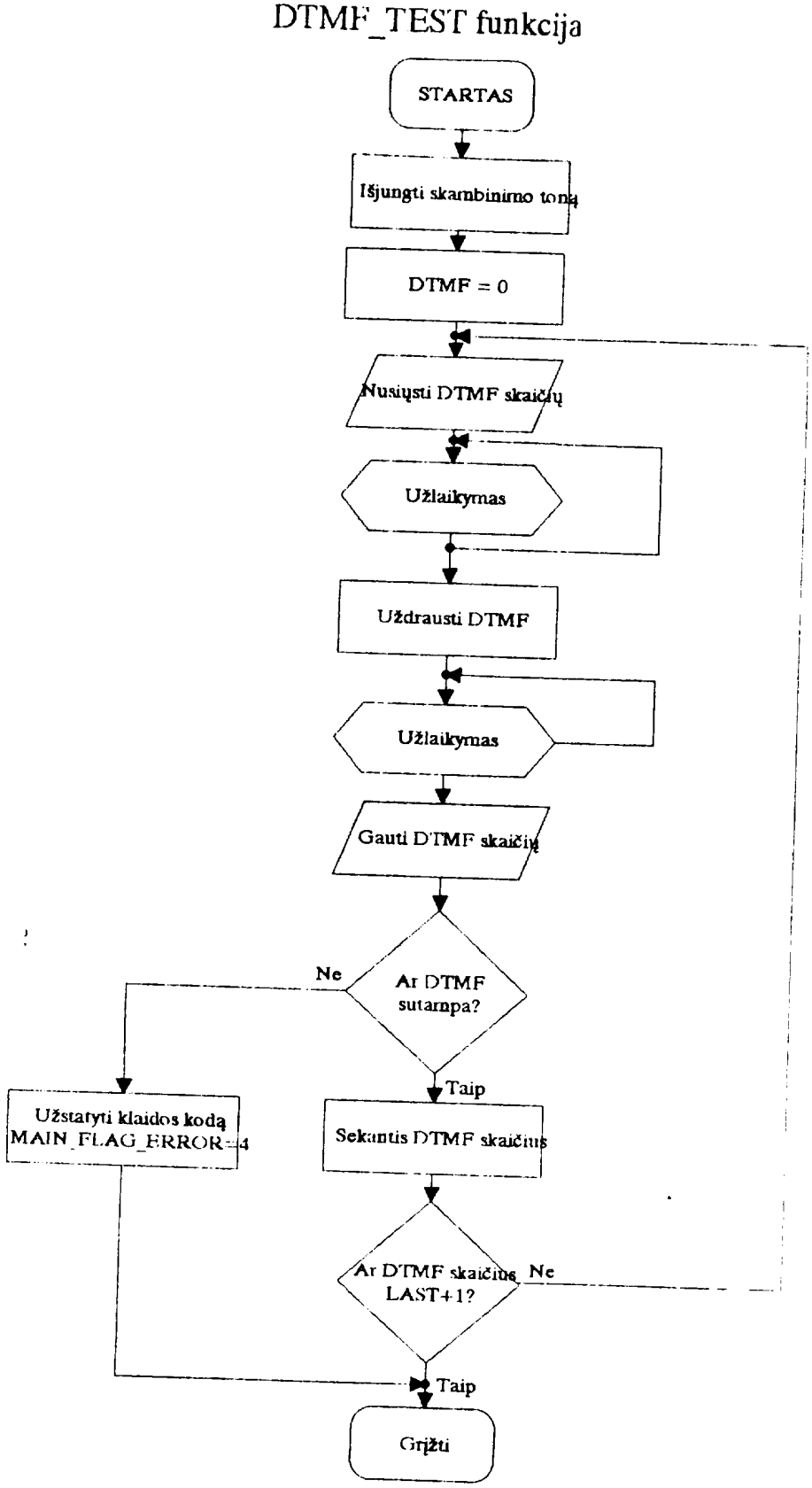
DTMF_TEST funkcija
STARTAS
Išjungti skambinimo toną
DTMF = 0
Nusiųsti DTMF skaičių
Užlaikymas
Uždrausti DTMF
Užlaikymas
Gauti DTMF skaičių
Ar DTMF sutampa?
Ne
Taip
Užstatyti klaidos kodą MAIN_FLAG_ERROR=4
Sekantis DTMF skaičius
Ar DTMF skaičius LAST+1?
Ne
Taip
Grįžti

Fig.4

RING_TEST funkcija

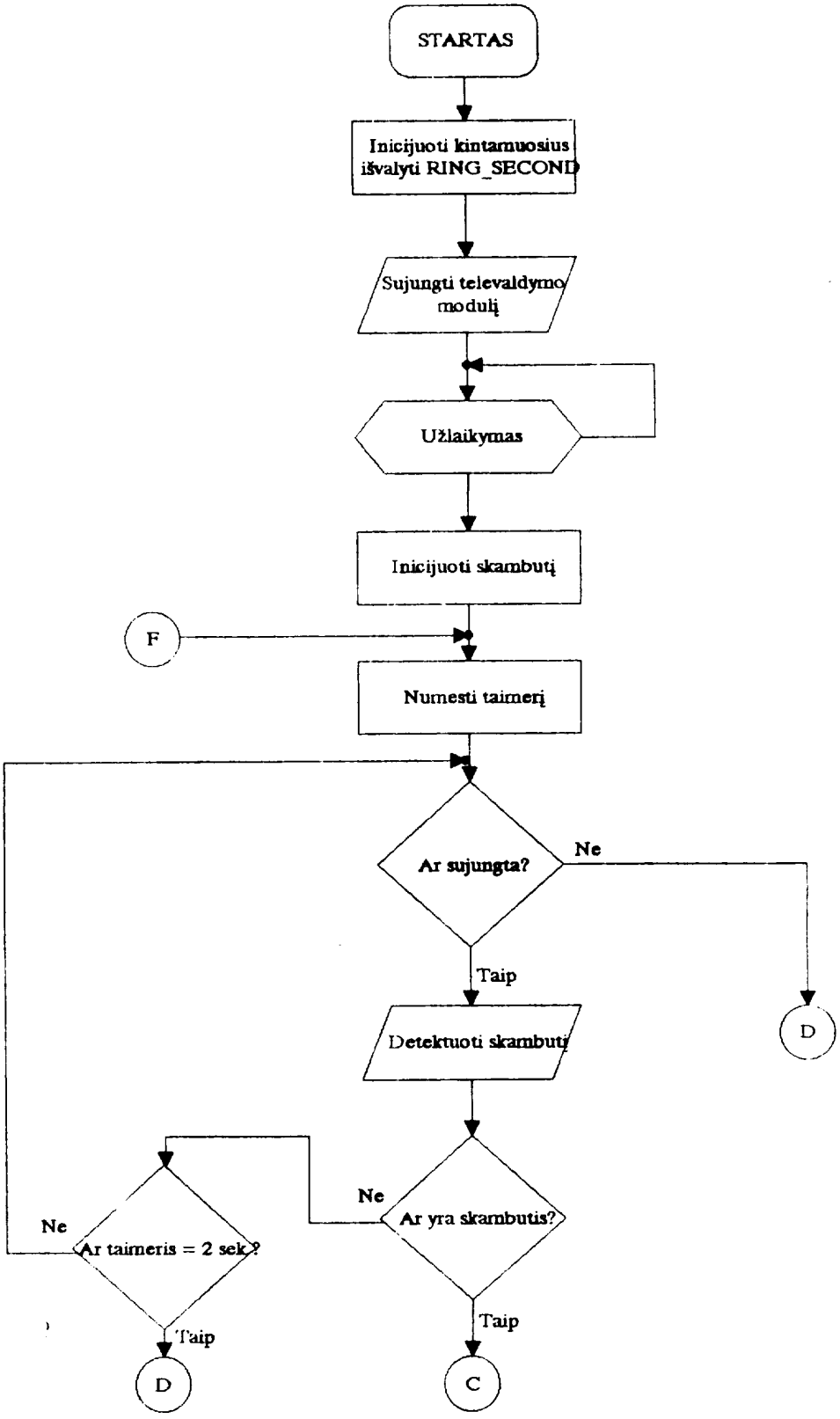

Fig.5A

RING_TEST funkcija (tęsinys)

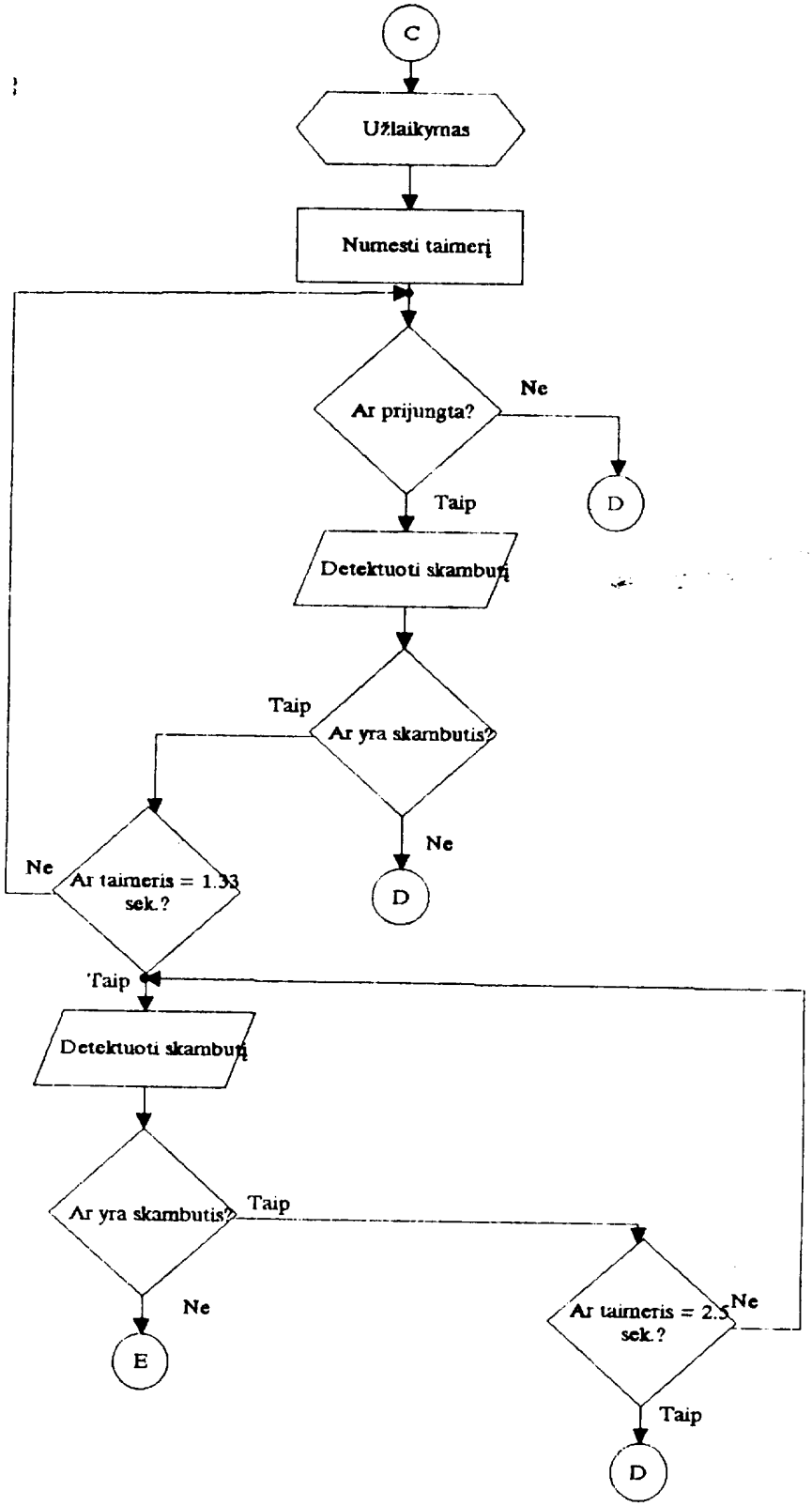

Fig.5B

RING_TEST funkcija (tęsinys)

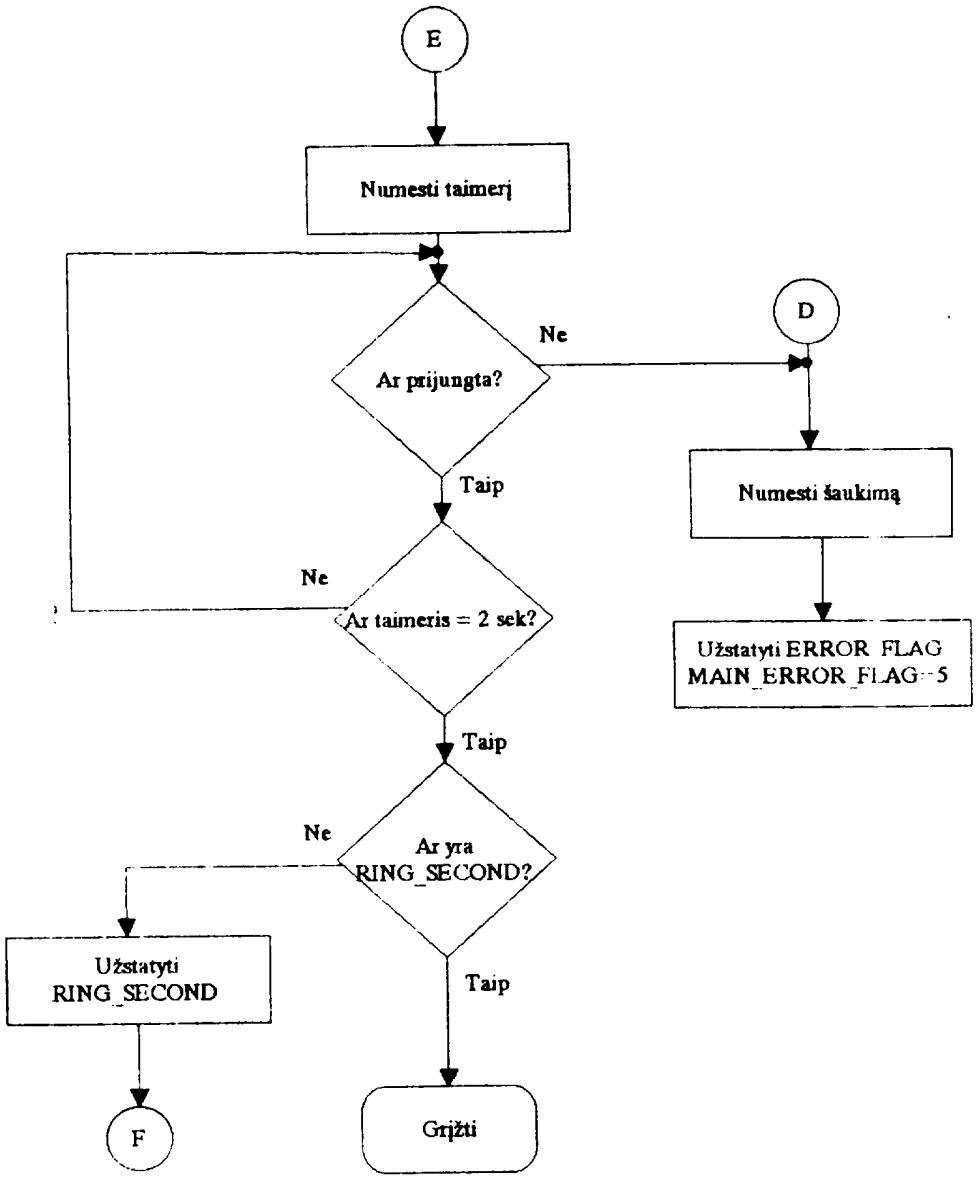

Fig.5C

RING_ANS funkcija

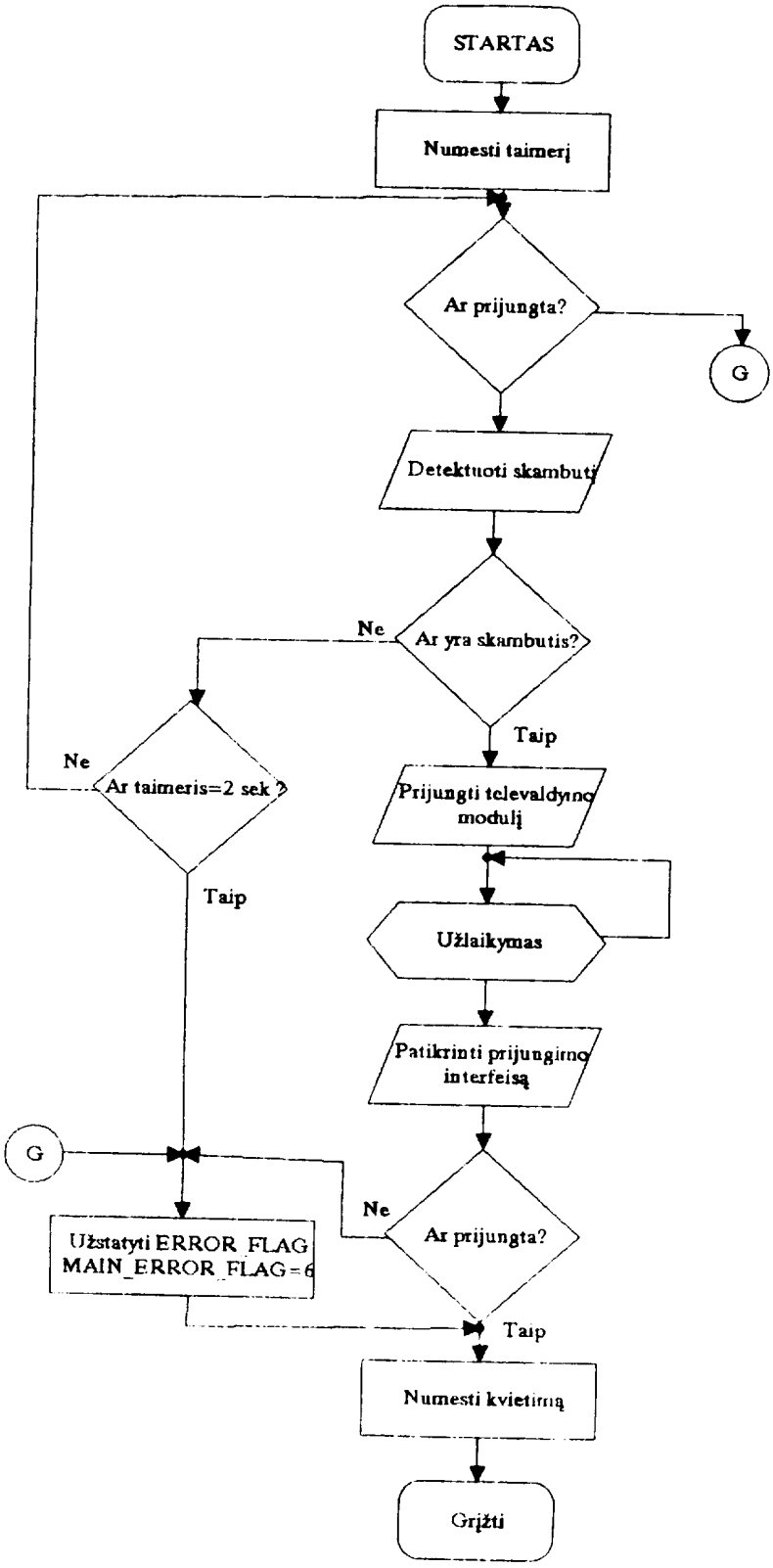

Fig.6

CALL_TEST funkcija

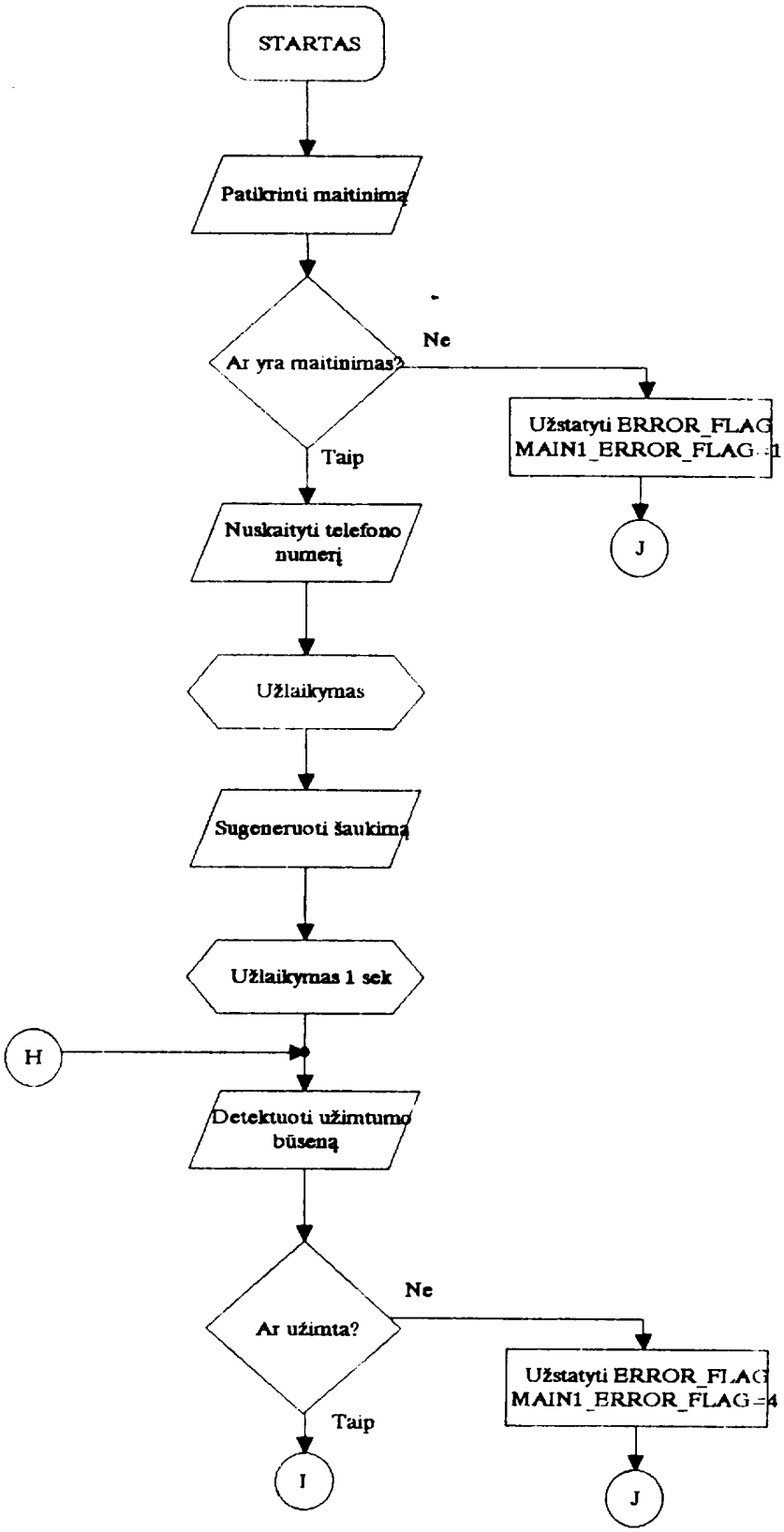

Fig.7A

CALL_TEST funkcija (tęsinys)

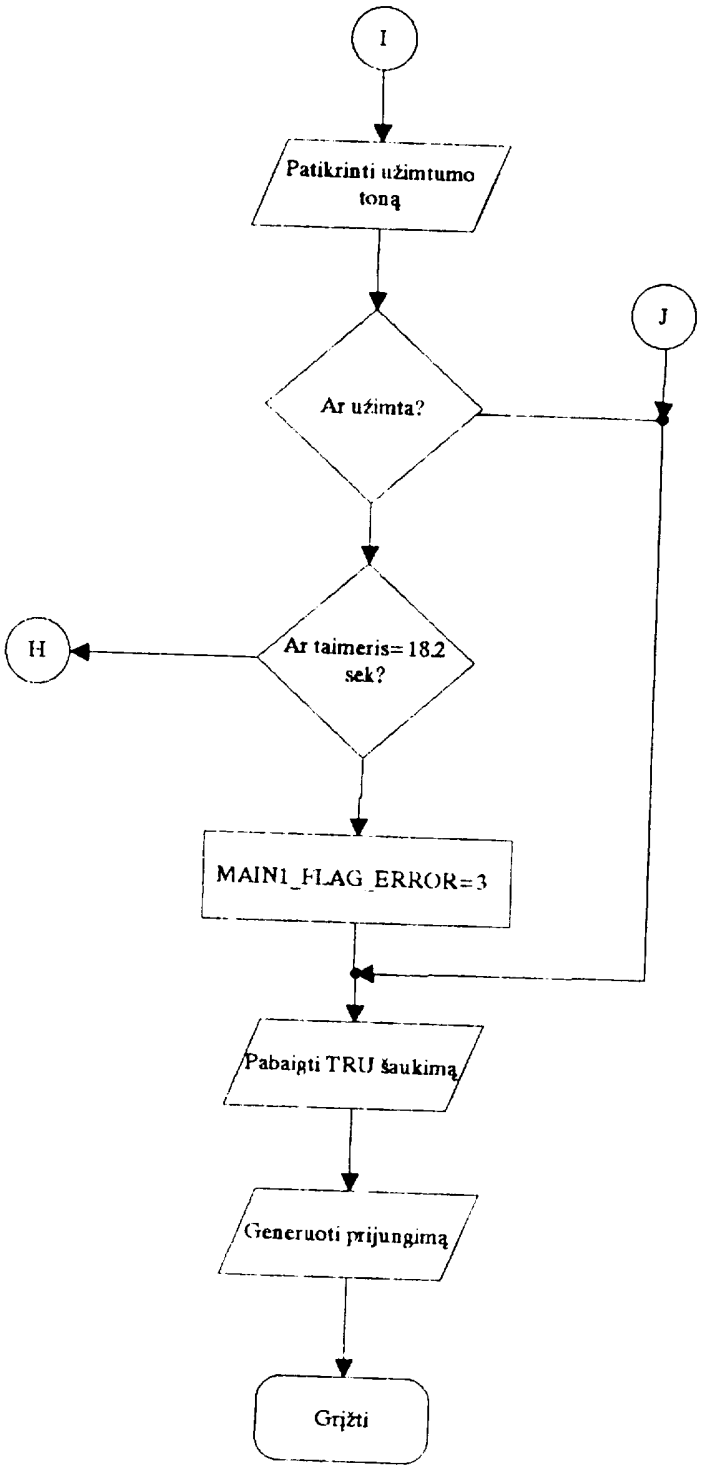

Fig.7B

ERROR_ACC funkcija

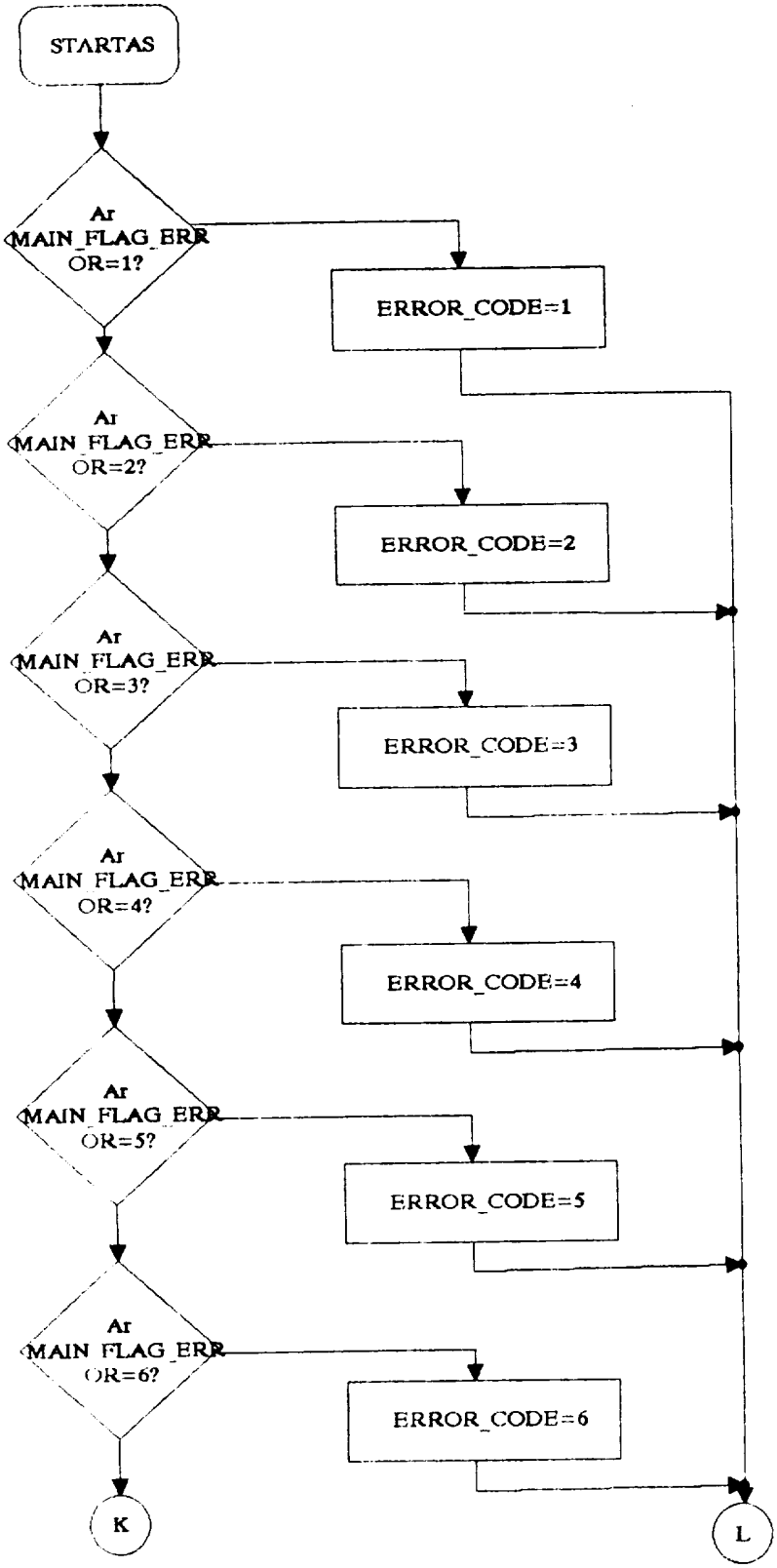

Fig.8A

ERROR_ACC funkcija (tęsinys)

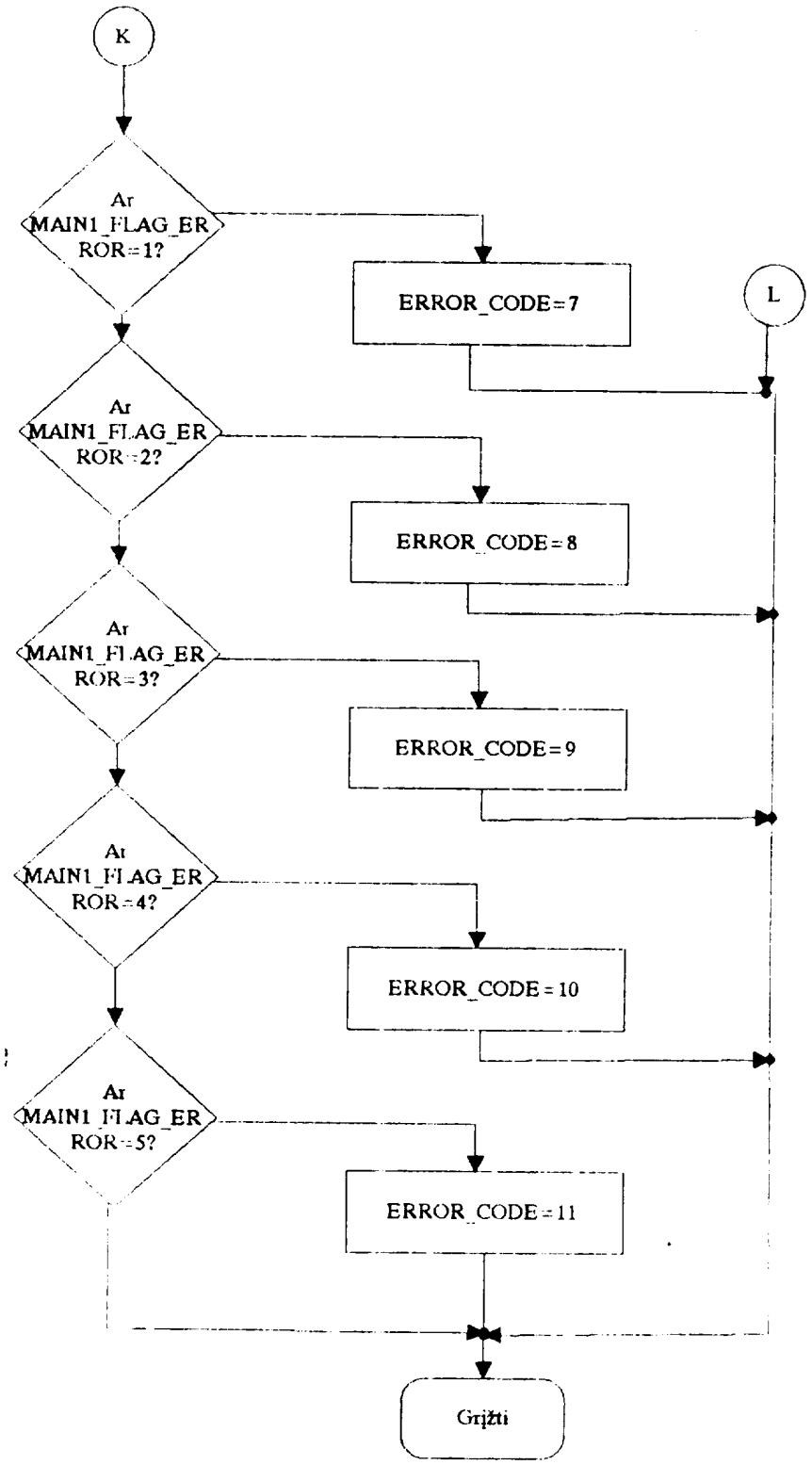

Fig.8B

FIN_SELF funkcija

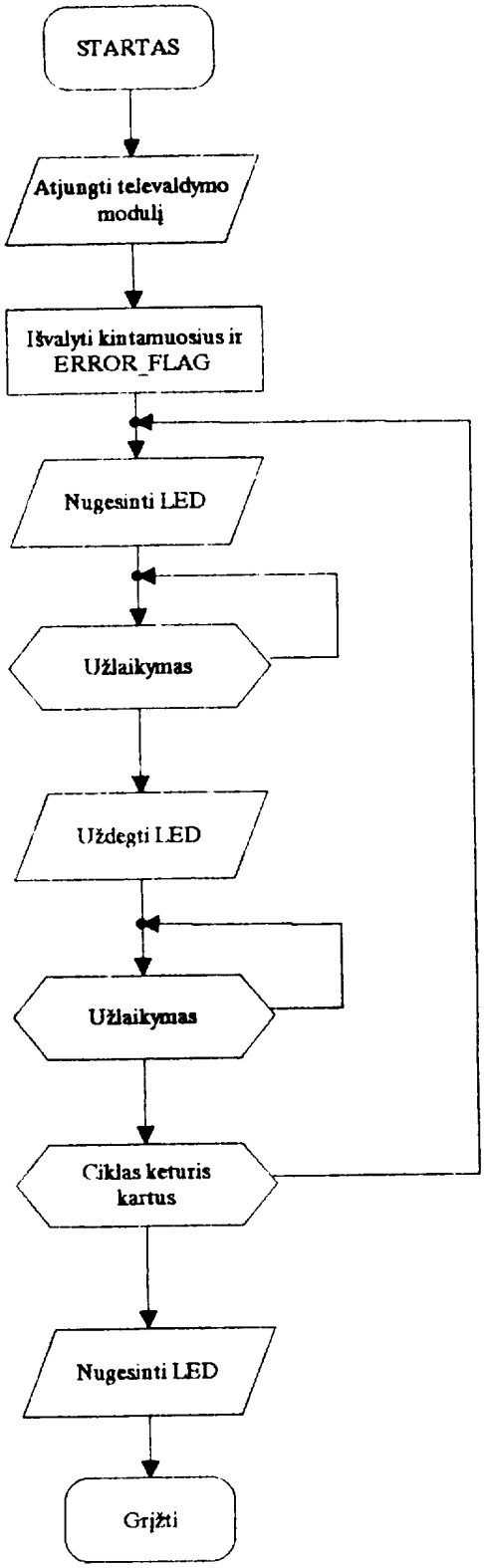

Fig.9

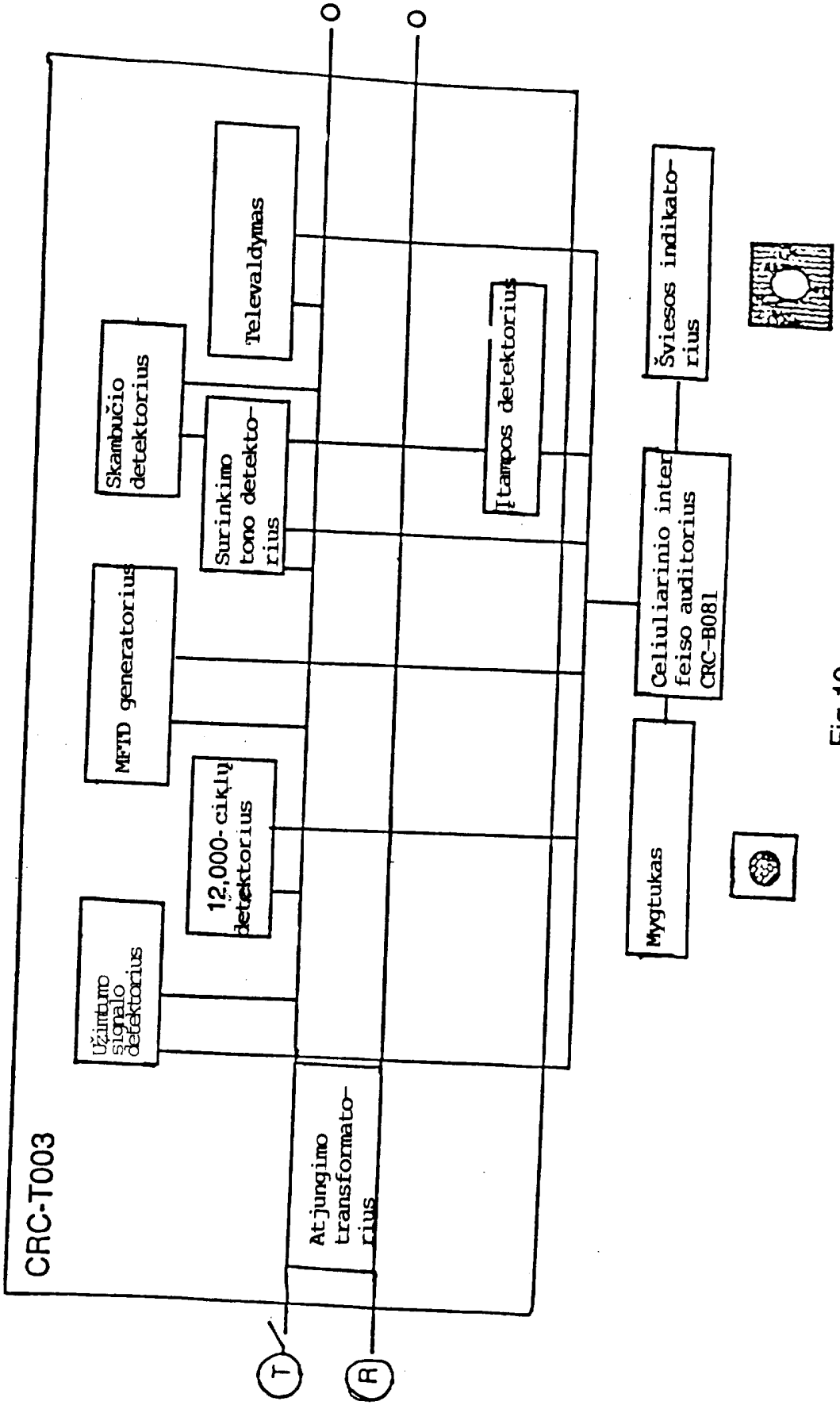
CRC-T003
Užimtumo signalo detektorius
12,000-ciklų detektorius
MFTD generatorius
Skambučio detektorius
Surinkimo tono detektorius
Televaldymas
Įtampos detektorius
Atjungimo transformatorius
T
R
Mygtukas
Celiuliarinio interfeiso auditorius CRC-B081
Šviesos indikatorius

Fig.10

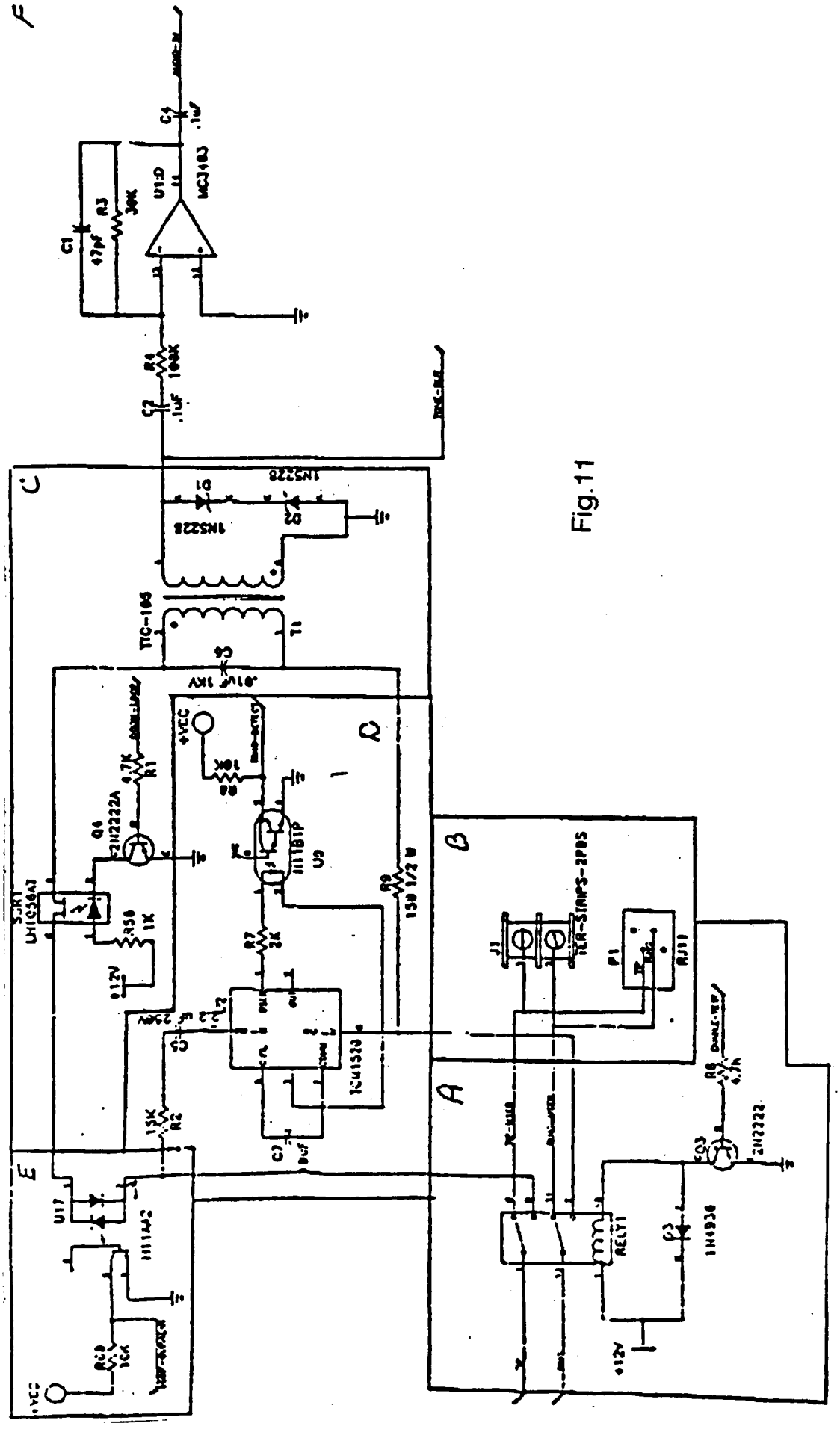

Fig.11

LT 3323 B

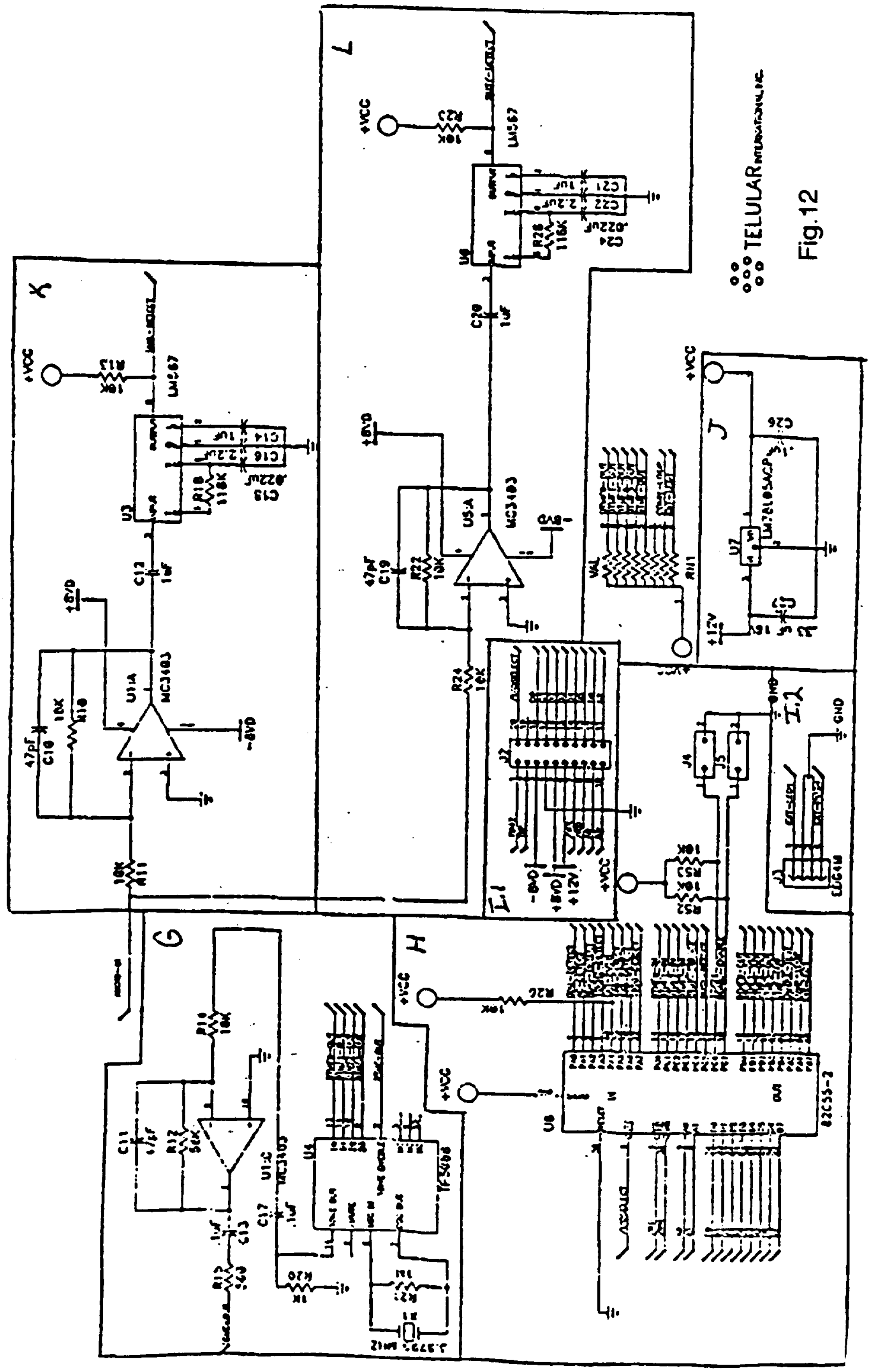

Fig.12

LT 3323 B

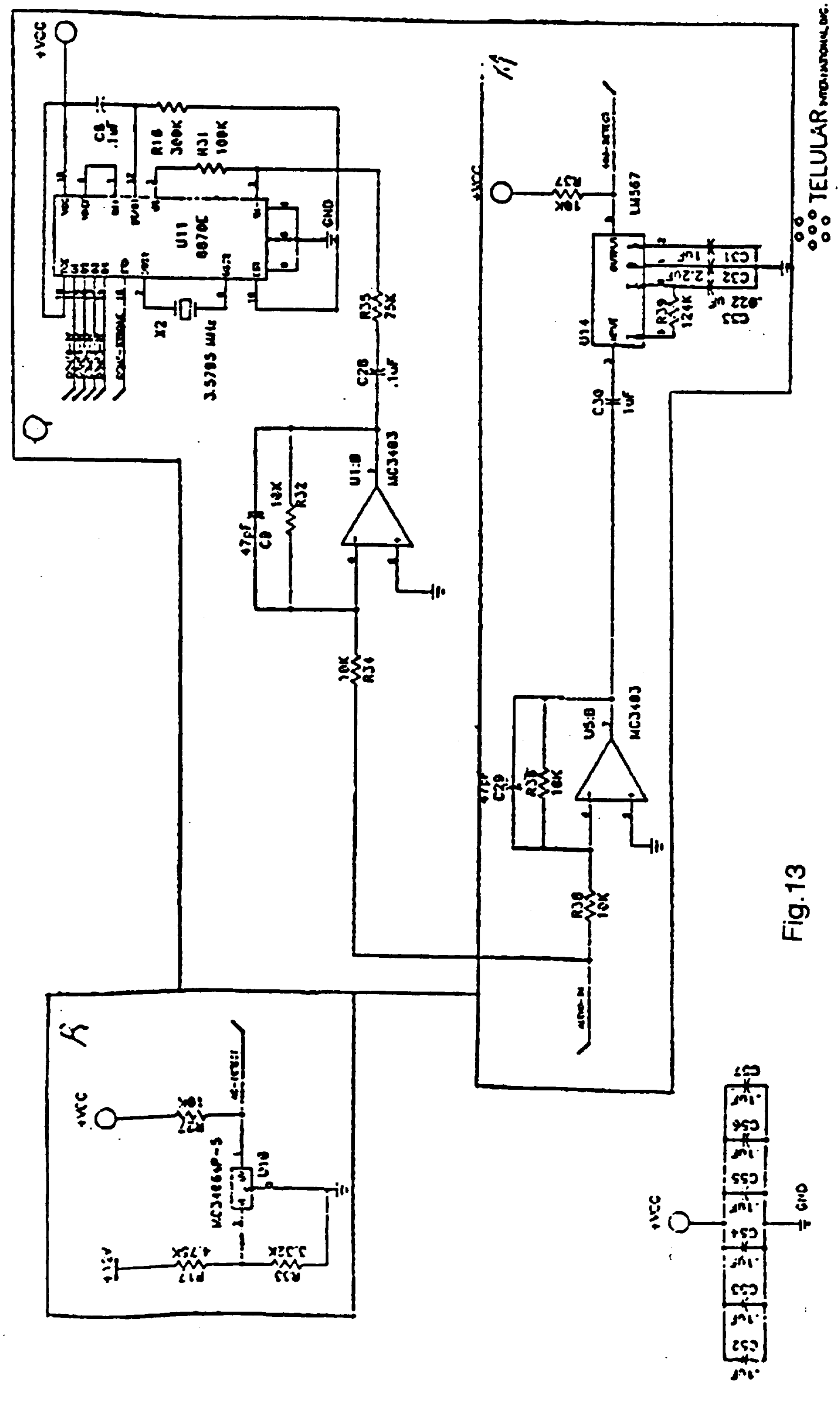

Fig.13

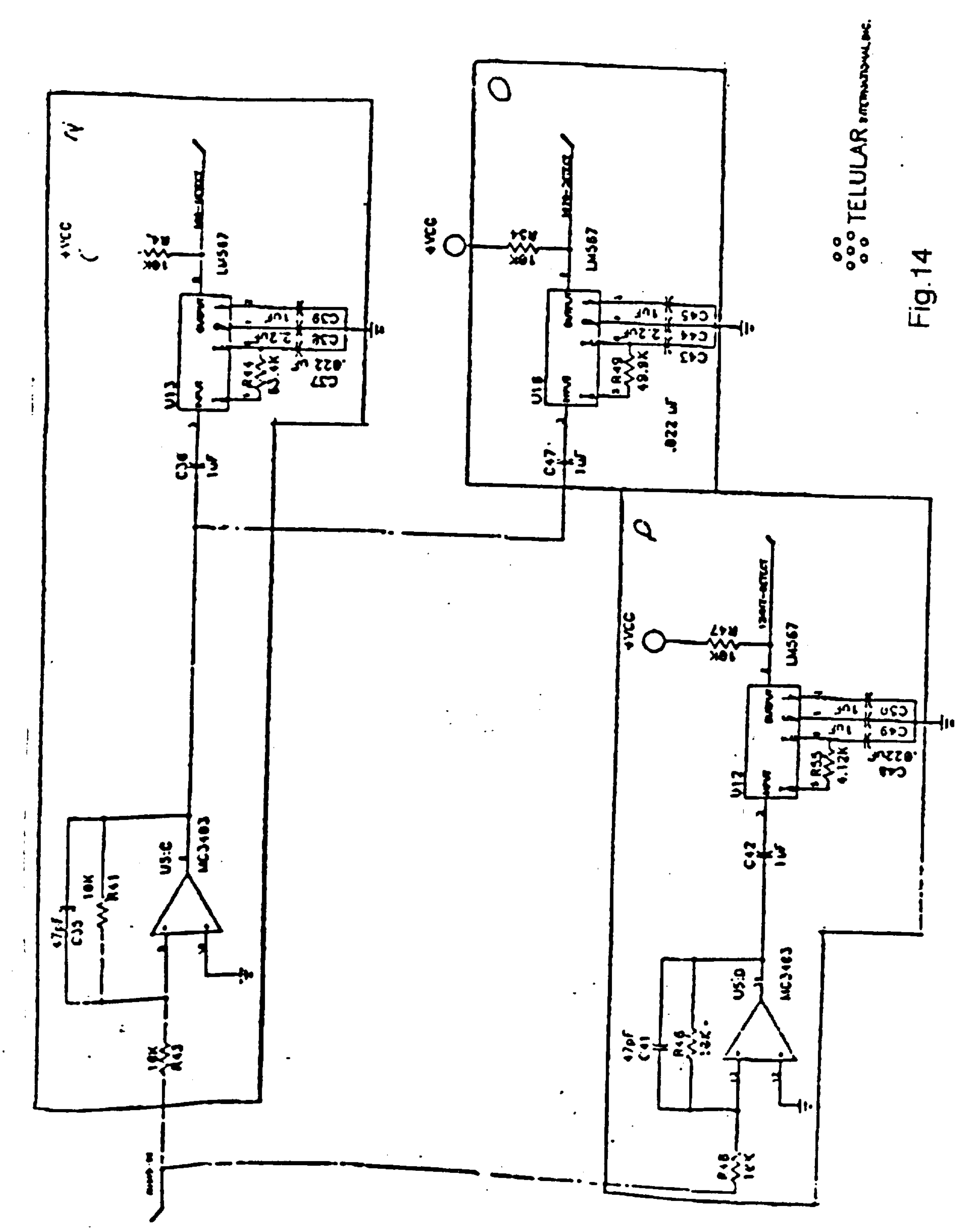

Fig.14