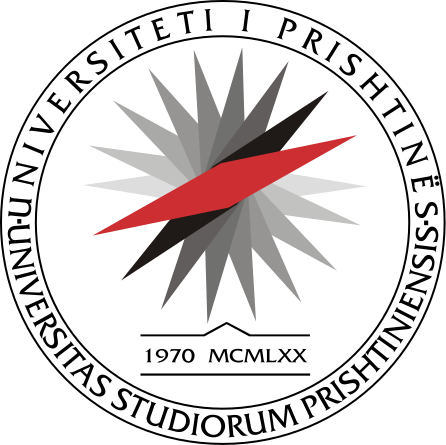
**UNIVERSITETI I PRISHTINËS “HASAN PRISHTINA”**

**FAKULTETI I INXHINIERISË ELEKTRIKE DHE KOMPJUTERIKE**



**Lënda: Automatizimi i Objekteve të Banimit**

**Tema: Kontrollimi i dritës përmes protokollit MODBUS**

**Profesor: Studenti:**

**Prof. Dr. Lavdim Kurtaj Erona Islami**

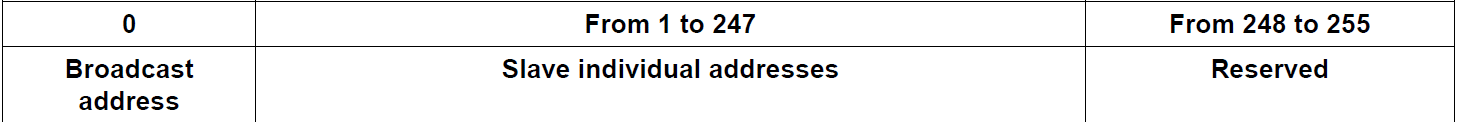
**Prishtinë, 2023**

**1.Hyrje**

Protokolli MODBUS është një protokoll komunikimi master-slave. Në bus është i lidhur vetëm një master dhe mund të lidhen prej një deri në 247 slave të ndryshëm. Komunikimi në MODBUS gjithmonë iniciohet nga masteri. Slave-at asnjëherë nuk transmetojnë të dhëna pa ardhur së pari një kërkesë nga masteri, gjithashtu slave-at nuk mund të komunikojnë ndërmjet veti. Masteri inicion vetëm një transaksion MODBUS në kohë.

MODBUS mund të implementohet në shumë mënyra, por më të zakonshmet janë implementimi përmes interfejsit fizik RS-485 ose RS-422 apo edhe TCP/IP over Ethernet.

Hapësira e adresimit për MODBUS është 256 adresa të ndryshme ku adresa 0 dhe adresat prej 248-255 janë të veçanta.

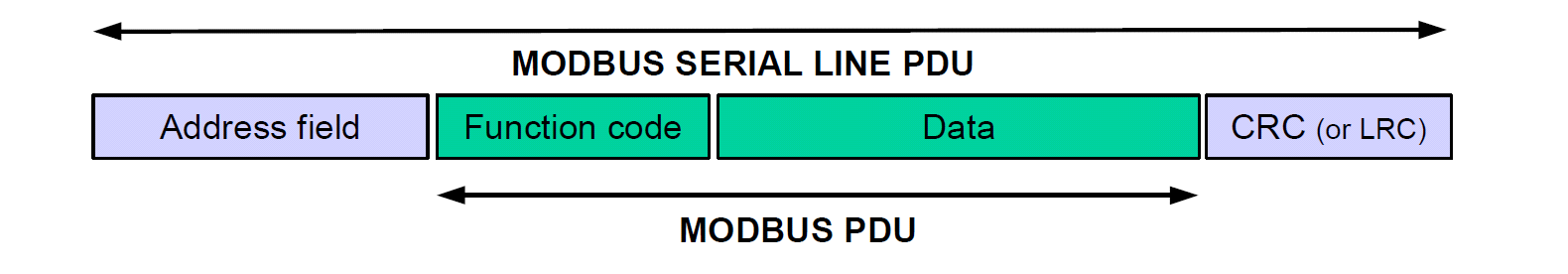


***Fig. 1. Adresimet në MODBUS.***

Adresa 0 njihet si broadcast adresë, ku në këtë rast të gjithë slave-at duhet të njohin broadcast adresën. Kjo adresë përdoret kur masteri dërgon një kërkesë për të gjithë slave-at të cilët gjenden në bus, ku vetëm e marrin kërkesën nga masteri dhe nuk kthejnë ndonjë përgjigje, prandaj këto janë karakteristikat që e dallojnë këtë adresë nga adresat tjera.

Në protokollin MODBUS masteri nuk ka ndonjë adresë dhe gjithashtu të gjithë slave-at duhet të kenë një adresë unike në bus.

Frame i MODBUS i cili komunikon përgjatë një linje serike është paraqitur në fig.2 dhe përbëhet nga: Address Field, Function Code,data dhe CRC (ose LRC).

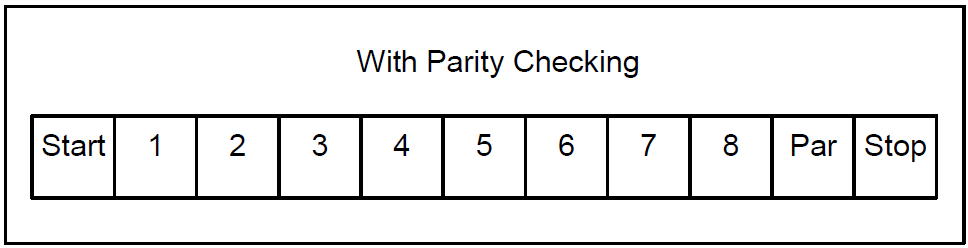


***Fig. 2. Frame i MODBUS në komunikimin serik.***

Address field është pjesa ku masteri i adreson slave-at në, ku kjo fushë përbëhet nga një bajt ose 8 bita, pastaj function code tregon se çfarë veprimi duhet të kryejë ai slave siç është leximi i ndonjë coil, leximi i regjistrave, shkrimi ne coila ose shkrimi në një ose më shumë regjistra etj. Function code përcjellët nga pjesa e data që përmban parametrat e kërkesës dhe përgjigjes.

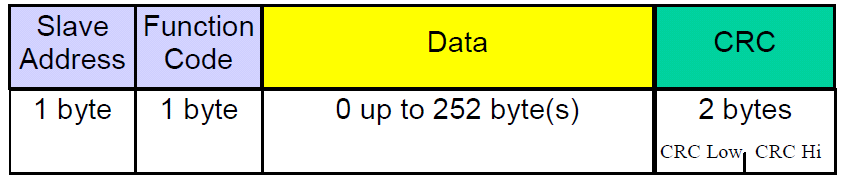
Në fund është fusha për kontrollimin e frame për ndonjë gabim e cila mundësohet me llogaritjen e CRC-së ose LRC-së varësisht se cili mod i transmetimit përdoret (RTU OSE ASCII).

Ekzistojnë dy mode të transmetimit serik në protokollin MODBUS dhe ato definohen si: modi RTU dhe modi ASCII. Kur pajisjet komunikojnë duke përdorur modin RTU (Remote Terminal Unit) secili bajt 8 bitësh në një mesazh do të përmbajë dy karaktere heksadecimale 4 bitësh. Avantazhi kryesor i këtij modi është se ka densitet më të madh të karaktereve që lejon të komunikojnë më shumë të dhëna për kohë të njëjtë krahasuar me modin e transmetimit ASCII.



***Fig. 3. Sekuenca e bitëve në modin RTU.***

Formati i secilit bajt në modin RTU është paraqitur në fig.3 ku përveç pjesës 8 bitësh përbëhet dhe nga një start bit, parity bit dhe një stop bit. Pjesa e parity bit mund të relaksohet ku në atë rast mund të luajë rolin e stop bitit dhe nganjëherë mund të themi se përbëhet nga një start bit dhe dy stop bita. Secili bit ose bajt dërgohet sipas rregullit të paracaktuar ku bitat ose bajtat transmetohen prej të majtës kah e djathta ose LSB to MSB.

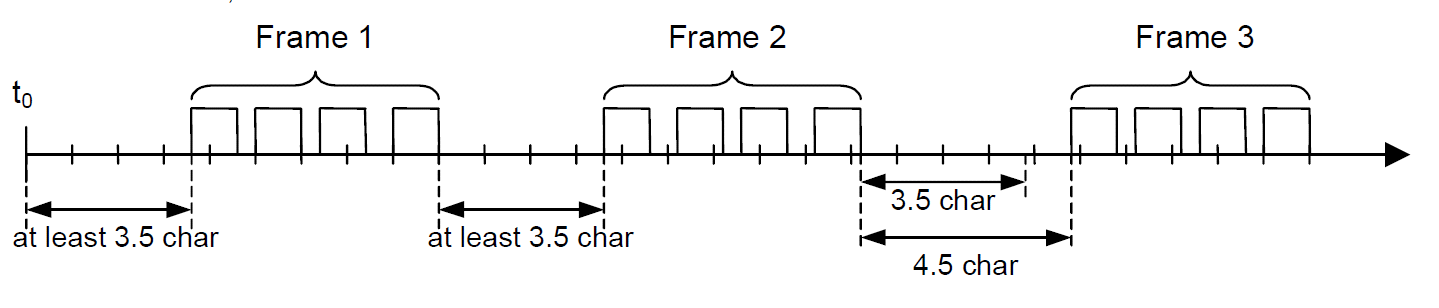


***Fig. 4. Frame i komunikimit.***

Siç shihet edhe në fig.4 pjesa e slave ka në posedim vetëm një bajt, gjithashtu edhe pjesa e function code kurse pjesa për të dhëna mund të përmbajë prej 0 deri ne 252 bajta dhe në fund pjesa e CRC ose LRC përbëhet nga dy bajta.

Mesazhi ne MODBUS dërgohet duke i bërë të ditur pranuesit se kur është koha e fillimit të pranimit të mesazhit dhe kur mbaron mesazhi. Nëse pranohen mesazhe të copëtuara atëherë në frame të atij mesazhi do të detektohet gabimi dhe masteri e injoron atë mesazh.

Në modin RTU secili frame është i ndarë me një interval kohor të “heshtjes” e cila llogaritet të jetë rreth 3.5tch.



***Fig. 5. Koha e kaluar që mundëson pranimin e tërë mesazhit.***

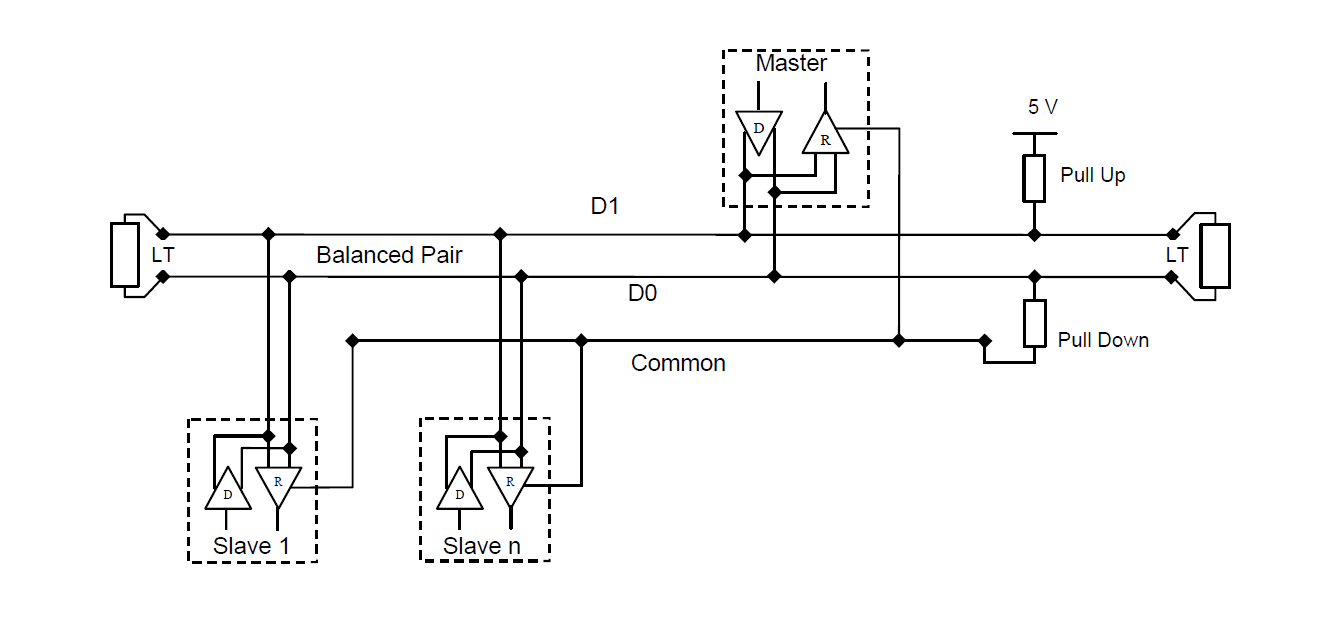
Gjithashtu ekziston edhe kufizimi tjetër i cili thotë që nëse distanca kohore për dërgimin e një bajti brenda një frame të caktuar është më e madhe se 1.5tch mesazhi i mund të konsiderohet i parëndësishëm dhe slave mund ta injorojë tërësisht atë paketë.

MODBUS mund të përdoret për shpejtësi të ndryshme të komunikimit e cila mund të jetë prej mbi 19200bps pastaj 9600bps,1200bps, 2400bps, 4800bps, 56kbps, 115kbps etj.

Siguria e pakteve të MODBUS është e bazuar në dy lloje të kontrollimit të gabimit:

* Kontrolli i paritetit (çift ose tek) duhet të aplikohet në secilin bajt.
* Kontrolli i frame (LRC ose CRC) duhet të aplikohet në tërë paketin.

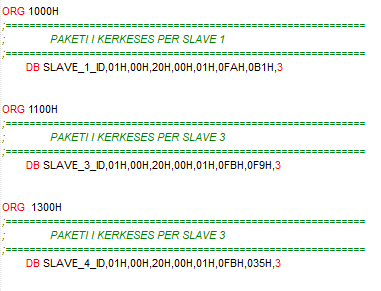
Masteri është konfiguruar nga përdoruesi të presë për një kohë të caktuar (Response time-out) përpara se të abortoj transaksionin. Ky interval është vënë aq gjatë saqë cilido slave të këtë mundësi të përgjigjet normalisht (unicast kërkesa). Nëse slave detekton ndonjë gabim atëherë e injoron atë paket dhe nuk kthen përgjigje tek masteri. Koha e caktuar do të kalojë programi i masterit do të konsideron këtë si një gabim. Gjithashtu edhe nëse ndonjë slave jo ekzistent adresohet nga masteri atëherë përsëri do të gjenerojë një gabim.

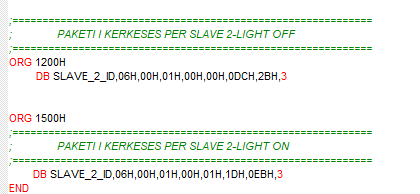
Në fig.5 është paraqitur forma e lidhjeve të slave dhe masterit në bus duke përdorur interfejsin fizik RS-485, ku siç shihet vendosen dy rezistenca terminuese në fundin e linjave për të detektuar se ku është fundi i linjës. Gjithashtu vendosen një pull-up rezistencë dhe një pull-down rezistencë të cilat përdoren për definimin e gjendjes së linjave.

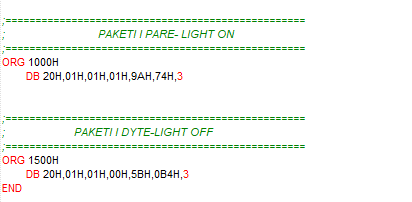
Në këtë projekt kam realizuar lidhjen e komponentëve në interfejsin fizik RS-485 dhe duke përdorur 5 mikrokontrollera 8051 ku njeri prej tyre luan rolin e masterit kurse katër njësitë tjera luajnë rolin e slave.

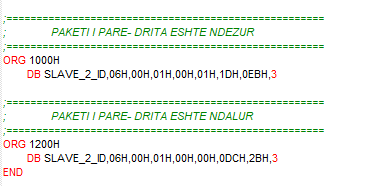
Për realizimin e kësaj detyre kam përdorur portin serik në modin 3, ku në këtë mod shpejtësia e komunikimit varet nga tajmeri 1 dhe gjithashtu lejohet edhe vendosja e bitit të 9 ose bitit të paritetit. Tek pjesa për dërgim biti i paritetit vendosen në regjistrin e veçantë SCON i cili përmban dy bita TB8 dhe RB8. Qasja në këta dy bita mund të bëhet drejtpërdrejtë pasi që ky regjistër është bit-addressable. TB8 paraqet bitin në të cilin vendoset vlera e bitit të paritetit për bajtin që jemi duke e dërguar, kurse RB8 paraqet bitin i cili tek ana e pranuesit e pranon bitin e pariteti i cili është dërguar nga njëra prej njësive. Me anë të kësaj pasi që të merret bajti nga buferi krahasohet se a është e njëjtë vlera e bitit të paritetit dhe kjo paraqet edhe një siguri shtesë.

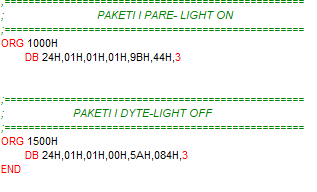
Tre slave-at lexojnë vlerën e ndërprerësit kurse slave tjetër është i lidhur me dritën e cila ndezët në varësi të kërkesës që merr slave. Për implementimin e kësaj detyre në të tre slave-at kam definuar funksionet të cilat përmbajnë gjendjen e ndërprerësave dhe në varësi nga gjendja e ndërprerësave, i dërgohet kërkesa slave tjetër duke shkruar në një regjistër të tij dhe në varësi të vlerës së regjistrit ndezët apo ndalet drita. E gjithë kjo varet nga gjendja në të cilën ndodhen ndërprerësat.Pas çdo pranimi të kërkesës llogaritet CRC-ja kurse në secilin bajt që pranohet bëhet llogaritja e bitit të paritetit e cila është një standard më i lartë i sigurisë. Frame i secilit bajt përmban 11 bita, një start bit, parity bit dhe një stop bit në fund. Pasi sistemi të aktivizohet në master fillon koha e numërimit prej 3.5tch dhe pas kësaj ai kalon në modin e dërgimit të kërkesës ku në varësi nga vlera e ndërprerësave i dërgohet kërkesa slave-it për ndezjen ose ndaljen e dritës.

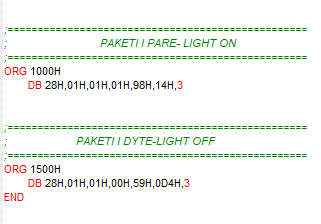






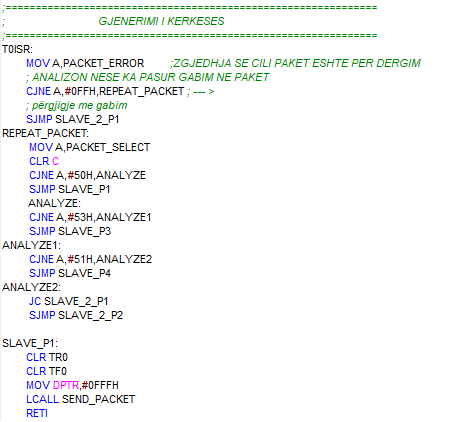


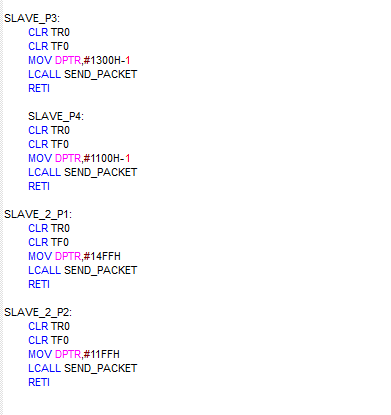




***Fig. 9. Paketet të cilët gjenerohen nga masteri.***

Në bazë të kërkesave dhe përgjigjeve masteri dhe secili slave dërgojnë njërin nga këto dy pakete.





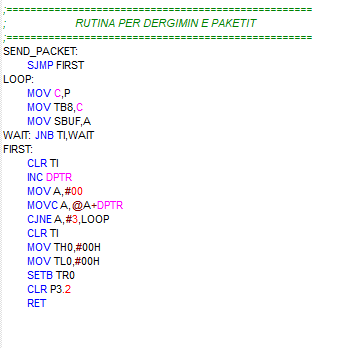
***Fig. 10. Gjenerimi i kërkesës.***

Siç e cekëm më lartë në momentin kur kalon koha prej 3.5tch aktivizohet interrapti i tajmerit dhe kalojmë në modin e dërgimit të kërkesës nga masteri kurse katër slave-at janë duke pritur për ndonjë kërkesë.

Siç shihet edhe nga fig.10 shikohet nëse ka ndonjë gabim në paketin i cili është pranuar dhe nëse ka atëherë masteri gjeneron përsëri kërkesën të cilën e ka gjeneruar herën e fundit për slave të njëjtë. Bajti me emrin PACKET\_SELECT analizon nëse ka ndodhur ndonjë ndërrim i gjendjes në ndërprerës dhe nëse ka ndodhur atëherë ai gjeneron kërkesën për slave 2 të cilin e urdhëron të ndalë ose të ndezë dritën varësisht se cila është gjendja e ndërprerësve.

Pasi zgjidhet slave i dëshiruar caktohet vendi ku gjendet paketi për dërgim dhe pastaj thirret rutina tjetër e paraqitur në fig.11 për dërgimin e bajtave prej masterit tek ndonjëri slave.

Pasi që dërgohet kërkesa atëherë masteri kalon në një gjendje pritje prej rreth 70ms dhe nëse gjatë kësaj kohe nuk pranohet ndonjë paket atëherë masteri konsideron që ka ndodhur ndonjë gabim dhe gjeneron përsëri kërkesën për slave-in e njëjtë deri sa të pranojë ndonjë përgjigje .

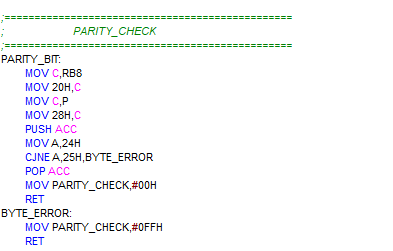


***Fig. 11. Rutina për dërgimin e paketit përmes portit serik.***

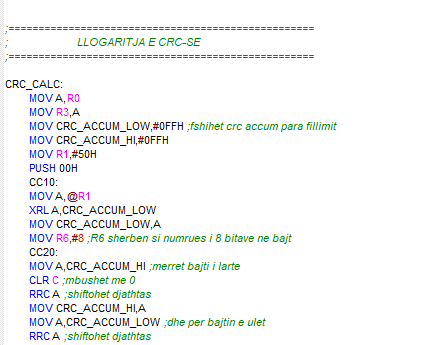
Gjatë kësaj kohe të katër slave-at pranojnë këtë paket por në varësi se për cilin paket është dërguar kërkesa, ai slave bën analizimin e këtij paketi kurse slave-at tjerë e injorojnë tërësisht këtë paket.

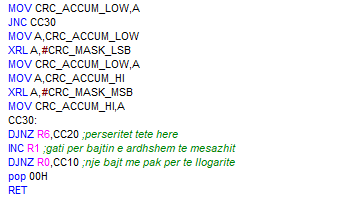
Për të pesta njësitë pasi që gjenerohet interrapti serik në varësi se cila pranon kërkesë dhe në këtë pjesë bëhet edhe llogaritja e CRC-së dhe analizimi i paketit. Pjesa e llogaritjes së CRC-së është realizuar në formë të një rutine e cila përdori polinomin A001 për llogaritjen e CRC-së dhe është paraqitur në fig.13.

Gjithashtu edhe për llogaritjen e bitit të paritetit përdoret një rutinë e veçantë e cila është e njëjtë për të gjitha njësitë dhe aplikohet në momentin kur pranohet një bajt nga porti serik dhe është paraqitur ne fig.12.



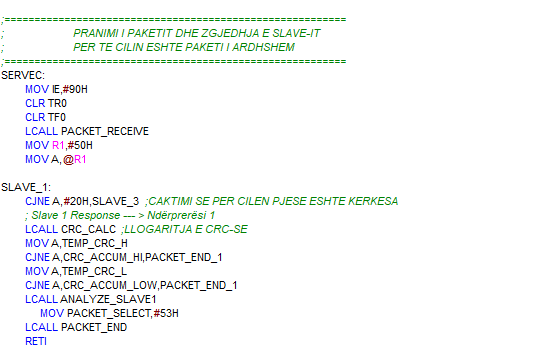
***Fig. 12. Llogaritja e bitit të paritetit.***

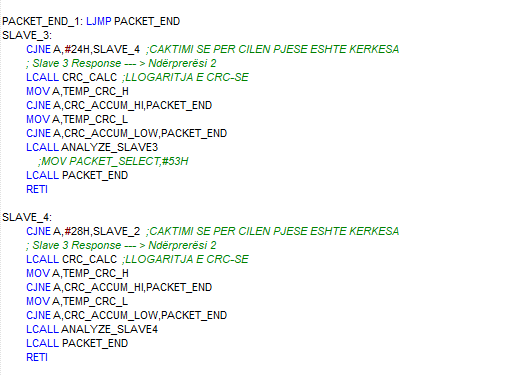


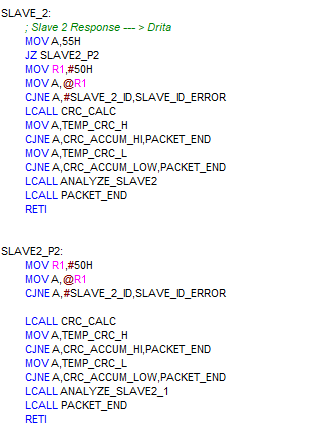


***Fig. 13. Llogaritja e CRC-së.***

Në fig.14 është paraqitur rutina për pranimin e paketit nga masteri dhe zgjedhja se cili slave dhe paket do të gjenerohen në kërkesën e ardhshme. Siç shihet në këtë pjesë bëhet edhe llogaritja e CRC-së dhe analizimi i paketit i cili ka ardhur. Po ashtu vlen të ceket se edhe 4 slave-at tjerë kanë logjikën e ngjashme të pranimit sikurse masteri vetëm se analizimi i paketit mund të dallojë nga masteri. Gjithashtu në fig.15 është paraqitur edhe pjesa për pranimin e paketit ku bëhet pranimi i të gjithë bajtave dhe pjesa e rëndësishme që mund të shohim në këtë rast janë edhe kohërat të cilat i kemi cekur më herët, me kushtet që kishin ku fundi dhe fillimi i paketit kuptoheshin pasi që kalonte koha prej 3.5tch dhe nëse hapësira kohore për pranimin e bajtit është më e madhe se 1.5tch atëherë merret që paketi është pranuar gabim dhe gjenerohet edhe një vonesë kohore prej 2tch për të arritur kohën e plotë prej 3.5tch dhe pastaj del nga kjo rutinë dhe me këtë nënkuptohet që është edhe fundi i paketit dhe nëse ky paket ka ardhur i mangët atëherë gjatë llogaritjes se gabimeve do të vërehet që ka ndodhur një gabim dhe paketi i pranuar do të injorohet tërësisht.

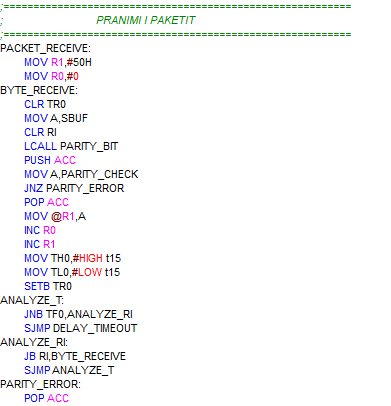


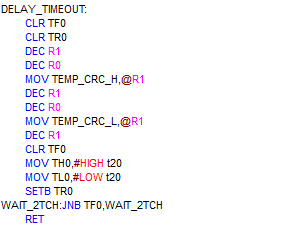






***Fig. 14. Pranimi i paketit nga masteri dhe caktimi i paketit të ardhshëm për dërgim.***





***Fig. 15. Rutina për pranimin e paketit.***

**2. Kodi në assembler për master:**

CRC\_ACCUM\_LOW EQU 0DH ; bank 1 regjistri R5

CRC\_ACCUM\_HI EQU 0EH ; bank 1 regjistri R6

CRC\_MASK\_LSB EQU 001H ;CRC-16 polinomi

CRC\_MASK\_MSB EQU 0A0H

TEMP\_CRC\_H EQU 40H

TEMP\_CRC\_L EQU 41H

VLERA\_BAJTIT EQU 60H

t15 EQU 63952 ;VONESA PREJ 1.5tch

t35 EQU 61848 ;VONESA PREJ 3.5tch

t20 EQU 63416 ;VONESA PREJ 2tch

SLAVE\_1\_ID EQU 20H

SLAVE\_4\_ID EQU 24H

SLAVE\_3\_ID EQU 28H

SLAVE\_2\_ID EQU 30H

VENDOSJA\_PAKETIT EQU 50H

PACKET\_ERROR EQU 64H

BYTE\_COUNT EQU 63H

COIL\_START\_ADDRESS EQU 1FH

PACKET\_ERROR1 EQU 67H

PARITY\_CHECK EQU 42H

PACKET\_SELECT EQU 70H

ORG 0000H

LJMP MAIN

ORG 000BH

LJMP T0ISR

ORG 0023H

LJMP SERVEC

ORG 0030H

MAIN:

MOV PACKET\_ERROR,#00H

MOV PACKET\_ERROR1,#00H

MOV PACKET\_SELECT,#50H

MOV SP,#35H

MOV SCON,#0D0H ;TI=1, RI=0, REN=1

MOV TMOD,#21H ;TIMER1=MOD2-AUTORELOAD,TIMER0=MOD1-16BIT

MOV PCON,#00H ;SINGLE BAUD RATE

MOV TH1,#0FDH ;9600 BAUD, #0xF3=>1200 BAUD (12 MHz)

SETB TR1 ;startohet timeri per portin serik

MOV TH0,#HIGH t35

MOV TL0,#LOW t35

SETB TR0

MOV A,#00

MOV IE,#92H ;lejohet interapti i tajmerit0 dhe iportit serik

MOV 20H,#00H ;Slave 1

MOV 21H,#00H

MOV 22H,#00H ;Slave 3

CLR TI

SETB P3.2

SJMP $

;==============================================================

; GJENERIMI I KERKESES

;==============================================================

T0ISR:

MOV A,PACKET\_ERROR ;ZGJEDHJA SE CILI PAKET ESHTE PER DERGIM

; ANALIZON NESE KA PASUR GABIM NE PAKET

CJNE A,#0FFH,REPEAT\_PACKET ; --- >

; përgjigje me gabim

SJMP SLAVE\_2\_P1

REPEAT\_PACKET:

MOV A,PACKET\_SELECT

CLR C

CJNE A,#50H,ANALYZE

SJMP SLAVE\_P1

ANALYZE:

CJNE A,#53H,ANALYZE1

SJMP SLAVE\_P3

ANALYZE1:

CJNE A,#51H,ANALYZE2

SJMP SLAVE\_P4

ANALYZE2:

JC SLAVE\_2\_P1

SJMP SLAVE\_2\_P2

SLAVE\_P1:

CLR TR0

CLR TF0

MOV DPTR,#0FFFH

LCALL SEND\_PACKET

RETI

SLAVE\_P3:

CLR TR0

CLR TF0

MOV DPTR,#1300H-1

LCALL SEND\_PACKET

RETI

SLAVE\_P4:

CLR TR0

CLR TF0

MOV DPTR,#1100H-1

LCALL SEND\_PACKET

RETI

SLAVE\_2\_P1:

CLR TR0

CLR TF0

MOV DPTR,#14FFH

LCALL SEND\_PACKET

RETI

SLAVE\_2\_P2:

CLR TR0

CLR TF0

MOV DPTR,#11FFH

LCALL SEND\_PACKET

RETI

;=========================================================

; PRANIMI I PAKETIT DHE ZGJEDHJA E SLAVE-IT

; PER TE CILIN ESHTE PAKETI I ARDHSHEM

;=========================================================

SERVEC:

MOV IE,#90H

CLR TR0

CLR TF0

LCALL PACKET\_RECEIVE

MOV R1,#50H

MOV A,@R1

SLAVE\_1:

CJNE A,#20H,SLAVE\_3 ;CAKTIMI SE PER CILEN PJESE ESHTE KERKESA

; Slave 1 Response --- > Ndërprerësi 1

LCALL CRC\_CALC ;LLOGARITJA E CRC-SE

MOV A,TEMP\_CRC\_H

CJNE A,CRC\_ACCUM\_HI,PACKET\_END\_1

MOV A,TEMP\_CRC\_L

CJNE A,CRC\_ACCUM\_LOW,PACKET\_END\_1

LCALL ANALYZE\_SLAVE1

MOV PACKET\_SELECT,#53H

LCALL PACKET\_END

RETI

PACKET\_END\_1: LJMP PACKET\_END

SLAVE\_3:

CJNE A,#24H,SLAVE\_4 ;CAKTIMI SE PER CILEN PJESE ESHTE KERKESA

; Slave 3 Response --- > Ndërprerësi 2

LCALL CRC\_CALC ;LLOGARITJA E CRC-SE

MOV A,TEMP\_CRC\_H

CJNE A,CRC\_ACCUM\_HI,PACKET\_END

MOV A,TEMP\_CRC\_L

CJNE A,CRC\_ACCUM\_LOW,PACKET\_END

LCALL ANALYZE\_SLAVE3

;MOV PACKET\_SELECT,#53H

LCALL PACKET\_END

RETI

SLAVE\_4:

CJNE A,#28H,SLAVE\_2 ;CAKTIMI SE PER CILEN PJESE ESHTE KERKESA

; Slave 3 Response --- > Ndërprerësi 2

LCALL CRC\_CALC ;LLOGARITJA E CRC-SE

MOV A,TEMP\_CRC\_H

CJNE A,CRC\_ACCUM\_HI,PACKET\_END

MOV A,TEMP\_CRC\_L

CJNE A,CRC\_ACCUM\_LOW,PACKET\_END

LCALL ANALYZE\_SLAVE4

LCALL PACKET\_END

RETI

SLAVE\_2:

; Slave 2 Response --- > Drita

MOV A,55H

JZ SLAVE2\_P2

MOV R1,#50H

MOV A,@R1

CJNE A,#SLAVE\_2\_ID,SLAVE\_ID\_ERROR

LCALL CRC\_CALC

MOV A,TEMP\_CRC\_H

CJNE A,CRC\_ACCUM\_HI,PACKET\_END

MOV A,TEMP\_CRC\_L

CJNE A,CRC\_ACCUM\_LOW,PACKET\_END

LCALL ANALYZE\_SLAVE2

LCALL PACKET\_END

RETI

SLAVE2\_P2:

MOV R1,#50H

MOV A,@R1

CJNE A,#SLAVE\_2\_ID,SLAVE\_ID\_ERROR

LCALL CRC\_CALC

MOV A,TEMP\_CRC\_H

CJNE A,CRC\_ACCUM\_HI,PACKET\_END

MOV A,TEMP\_CRC\_L

CJNE A,CRC\_ACCUM\_LOW,PACKET\_END

LCALL ANALYZE\_SLAVE2\_1

LCALL PACKET\_END

RETI

SLAVE\_ID\_ERROR:

MOV TH0,#0FFH

MOV TL0,#0F0H

MOV IE,#92H

CLR TF0

SETB TR0

SETB P3.2

RETI

;==================================================================

; PERFUNDIMI I PAKETIT

;==================================================================

PACKET\_END:

MOV R1,#50H

MOV R0,#0

MOV IE,#92H

CLR TF0

MOV TH0,#HIGH t35

MOV TL0,#LOW t35

SETB P3.2

SETB TR0

RET

;==========================================================

; PRANIMI I PAKETIT

;==========================================================

PACKET\_RECEIVE:

MOV R1,#50H

MOV R0,#0

BYTE\_RECEIVE:

CLR TR0

MOV A,SBUF

CLR RI

LCALL PARITY\_BIT

PUSH ACC

MOV A,PARITY\_CHECK

JNZ PARITY\_ERROR

POP ACC

MOV @R1,A

INC R0

INC R1

MOV TH0,#HIGH t15

MOV TL0,#LOW t15

SETB TR0

ANALYZE\_T:

JNB TF0,ANALYZE\_RI

SJMP DELAY\_TIMEOUT

ANALYZE\_RI:

JB RI,BYTE\_RECEIVE

SJMP ANALYZE\_T

PARITY\_ERROR:

POP ACC

DELAY\_TIMEOUT:

CLR TF0

CLR TR0

DEC R1

DEC R0

MOV TEMP\_CRC\_H,@R1

DEC R1

DEC R0

MOV TEMP\_CRC\_L,@R1

DEC R1

CLR TF0

MOV TH0,#HIGH t20

MOV TL0,#LOW t20

SETB TR0

WAIT\_2TCH:JNB TF0,WAIT\_2TCH

RET

;======================================================

; ANALIZA E SLAVE DHE PAKETAVE

;======================================================

ANALYZE\_SLAVE1:

MOV R1,#51H ; Function Code position

MOV A,@R1

CJNE A,#01H,FUNCTION\_ERROR

INC R1 ; Byte Count position

MOV A,@R1

MOV BYTE\_COUNT,A

INC R1 ; Data Byte position

MOV A,@R1

CLR C

;

; A ka ndryshim të gjendjes?

; --- > komanda te Slave i dritës bartet vetëm kur ka ndryshim të gjendjes së dritës

;

; CJNE A,20H,TRANSITION

; MOV PACKET\_SELECT,#50H

; SJMP SELECT

;TRANSITION:

; JC LIGHT\_OFF

; MOV PACKET\_SELECT,#00H

; SJMP SELECT

;LIGHT\_OFF:

; MOV PACKET\_SELECT,#0FFH

;SELECT:

MOV 20H,A

;

;

; Gjendja e ndërprerësit bartet te drita

; --- > komanda te Slave i dritës bartet pas çdo leximi të gjemdjes së ndërprerësit

;

; JNB ACC.0,LIGHT\_OFF ; --- > Light OFF

; ; --- > Light ON

;LIGHT\_ON:

; MOV PACKET\_SELECT,#00H

; SJMP SELECT

;LIGHT\_OFF:

; MOV PACKET\_SELECT,#0FFH

;SELECT:

; MOV 20H,A

;

RET

FUNCTION\_ERROR:

LCALL PACKET\_END

RET

ANALYZE\_SLAVE3:

MOV R1,#51H

MOV A,@R1

CJNE A,#01H,FUNCTION\_ERROR

INC R1

MOV A,@R1

MOV BYTE\_COUNT,A

INC R1

MOV A,@R1

CLR C

;

; Funksionaliteti i ndërprerësit alternativ

;

mov c,acc.0

anl c,20h.0

cpl c ; toggle the value of C (used as flag)

jnc TURN\_ON ; if C=0 (flag off), turn on the light

; if C=1 (flag on), turn off the light

CLR P2.0 ; Set bit 0 of register P2 to logic low to turn off the light

MOV PACKET\_SELECT,#0FFH ; set PACKET\_SELECT to 0xFF (light off packet)

SJMP SELECT3

TURN\_ON:

CJNE A,22H,TRANSITION3

MOV PACKET\_SELECT,#51H

SJMP SELECT3

TRANSITION3:

JC LIGHT\_OFF3

MOV PACKET\_SELECT,#00H

SJMP SELECT3

LIGHT\_OFF3:

CLR P2.0 ; Set bit 0 of register P2 to logic low to turn off the light

MOV PACKET\_SELECT,#0FFH

SELECT3:

MOV 22H,A

RET

ANALYZE\_SLAVE4:

MOV R1,#51H

MOV A,@R1

CJNE A,#01H,FUNCTION\_ERROR

INC R1

MOV A,@R1

MOV BYTE\_COUNT,A

INC R1

MOV A,@R1

CLR C

;

; Funksionaliteti i ndërprerësit alternativ

;

mov c,acc.0

;cpl c

anl c,20h.0

;mov 20h.1,c

;mov c,20h.0

;cpl c

;or;l c,acc.0

;anl c,20h.1

;cpl c

;mov c, 20h.0

cpl c

;clr c

;

;CJNE A,21H,TRANSITION3

;MOV PACKET\_SELECT,#51H

;SJMP SELECT3

;TRANSITION3:

JC LIGHT\_OFF4

MOV PACKET\_SELECT,#00H

SJMP SELECT4

LIGHT\_OFF4:

MOV PACKET\_SELECT,#0FFH

SELECT4:

MOV 21H,A

RET

ANALYZE\_SLAVE2:

MOV R1,#51H

MOV A,@R1

CJNE A,#06,FUNKSION\_GABIM1

INC R1

MOV A,@R1

CJNE A,#00H,FUNKSION\_GABIM1

INC R1

MOV A,@R1

CJNE A,#01H,FUNKSION\_GABIM1

INC R1

MOV A,@R1

CJNE A,#00H,FUNKSION\_GABIM1

INC R1

MOV A,@R1

CJNE A,#01H,FUNKSION\_GABIM1

MOV PACKET\_ERROR,#00H

MOV PACKET\_SELECT,#50H

RET

FUNKSION\_GABIM1:

MOV PACKET\_ERROR,#0FFH

RET

ANALYZE\_SLAVE2\_1:

MOV R1,#51H

MOV A,@R1

CJNE A,#06,FUNKSION\_GABIM2

INC R1

MOV A,@R1

CJNE A,#00H,FUNKSION\_GABIM2

INC R1

MOV A,@R1

CJNE A,#01H,FUNKSION\_GABIM2

INC R1

MOV A,@R1

CJNE A,#00H,FUNKSION\_GABIM2

INC R1

MOV A,@R1

CJNE A,#00H,FUNKSION\_GABIM2

MOV PACKET\_ERROR1,#00H

MOV PACKET\_SELECT,#50H

RET

FUNKSION\_GABIM2:

MOV PACKET\_ERROR1,#0FFH

RET

;===================================================

; LLOGARITJA E CRC-SE

;===================================================

CRC\_CALC:

MOV A,R0

MOV R3,A

MOV CRC\_ACCUM\_LOW,#0FFH ;fshihet crc accum para fillimit

MOV CRC\_ACCUM\_HI,#0FFH

MOV R1,#50H

PUSH 00H

CC10:

MOV A,@R1

XRL A,CRC\_ACCUM\_LOW

MOV CRC\_ACCUM\_LOW,A

MOV R6,#8 ;R6 sherben si numrues i 8 bitave ne bajt

CC20:

MOV A,CRC\_ACCUM\_HI ;merret bajti i larte

CLR C ;mbushet me 0

RRC A ;shiftohet djathtas

MOV CRC\_ACCUM\_HI,A

MOV A,CRC\_ACCUM\_LOW ;dhe per bajtin e ulet

RRC A ;shiftohet djathtas

MOV CRC\_ACCUM\_LOW,A

JNC CC30

MOV A,CRC\_ACCUM\_LOW

XRL A,#CRC\_MASK\_LSB

MOV CRC\_ACCUM\_LOW,A

MOV A,CRC\_ACCUM\_HI

XRL A,#CRC\_MASK\_MSB

MOV CRC\_ACCUM\_HI,A

CC30:

DJNZ R6,CC20 ;perseritet tete here

INC R1 ;gati per bajtin e ardhshem te mesazhit

DJNZ R0,CC10 ;nje bajt me pak per te llogarite

pop 00H

RET

;================================================

; PARITY\_CHECK

;================================================

PARITY\_BIT:

MOV C,RB8

MOV 20H,C

MOV C,P

MOV 28H,C

PUSH ACC

MOV A,24H

CJNE A,25H,BYTE\_ERROR

POP ACC

MOV PARITY\_CHECK,#00H

RET

BYTE\_ERROR:

MOV PARITY\_CHECK,#0FFH

RET

;===================================================

; RUTINA PER DERGIMIN E PAKETIT

;===================================================

SEND\_PACKET:

SJMP FIRST

LOOP:

MOV C,P

MOV TB8,C

MOV SBUF,A

WAIT: JNB TI,WAIT

FIRST:

CLR TI

INC DPTR

MOV A,#00

MOVC A,@A+DPTR

CJNE A,#3,LOOP

CLR TI

MOV TH0,#00H

MOV TL0,#00H

SETB TR0

CLR P3.2

RET

ORG 1000H

;============================================================

; PAKETI I KERKESES PER SLAVE 1

;============================================================

DB SLAVE\_1\_ID,01H,00H,20H,00H,01H,0FAH,0B1H,3

ORG 1100H

;============================================================

; PAKETI I KERKESES PER SLAVE 3

;============================================================

DB SLAVE\_3\_ID,01H,00H,20H,00H,01H,0FBH,0F9H,3

ORG 1300H

;============================================================

; PAKETI I KERKESES PER SLAVE 3

;============================================================

DB SLAVE\_4\_ID,01H,00H,20H,00H,01H,0FBH,035H,3

;============================================================

; PAKETI I KERKESES PER SLAVE 2-LIGHT OFF

;============================================================

ORG 1200H

DB SLAVE\_2\_ID,06H,00H,01H,00H,00H,0DCH,2BH,3

ORG 1500H

;============================================================

; PAKETI I KERKESES PER SLAVE 2-LIGHT ON

;============================================================

DB SLAVE\_2\_ID,06H,00H,01H,00H,01H,1DH,0EBH,3

END

**3. Kodi në assembler për slave 1:**

CRC\_ACCUM\_LOW EQU 0DH ; bank 1 regjistri R5

CRC\_ACCUM\_HI EQU 0EH ; bank 1 regjistri R6

CRC\_MASK\_LSB EQU 001H ;CRC-16 polinomi

CRC\_MASK\_MSB EQU 0A0H

TEMP\_CRC\_H EQU 40H

TEMP\_CRC\_L EQU 41H

t15 EQU 63952 ;VONESA PREJ 1.5tch

t35 EQU 61848 ;VONESA PREJ 3.5tch

t20 EQU 63416 ;VONESA PREJ 2tch

PAKETI\_GABIM1 EQU 67H

PARITY\_CHECK EQU 42H

ORG 0000H

LJMP MAIN

ORG 000BH

LJMP T0ISR

ORG 0023H

LJMP SERVEC

ORG 0030H

MAIN:

MOV SP,#35H

MOV SCON,#0D0H ;TI=1, RI=0, REN=1

MOV TMOD,#21H ;TIMER1=MOD2-AUTORELOAD,TIMER0=MOD1-16BIT

MOV PCON,#00H ;SINGLE BAUD RATE

MOV TH1,#0FDH ;9600 BAUD, #0xF3=>1200 BAUD (12 MHz)

SETB TR1 ;startohet timeri per portin serik

MOV P0,#00H

MOV IE,#90H ;lejohet interapti i tajmerit0 dhe iportit serik

CLR TI

CLR P3.2

POLLING:

MOV C,P0.0

MOV 00H,C

SJMP POLLING

;==============================================================

; KTHIMI I PERGJIGJES PAS KALIMIT TE KOHES PREJ 3.5tch

;==============================================================

T0ISR:

CLR TR0

CLR TF0

CLR C

MOV C,00H

JC PACKET\_2

MOV DPTR,#14FFH

LCALL SEND\_PACKET

RETI

PACKET\_2:

MOV DPTR,#0FFFH

LCALL SEND\_PACKET

RETI

;=========================================================

; PRANIMI I PAKETIT DHE ANALIZIMI

;=========================================================

SERVEC:

CLR TF0

LCALL PACKET\_RECEIVE

MOV R1,#50H

MOV A,@R1

CJNE A,#20H,ID\_GABIM

LCALL CRC\_CALC

MOV A,TEMP\_CRC\_H

CJNE A,CRC\_ACCUM\_HI,ID\_GABIM

MOV A,TEMP\_CRC\_L

CJNE A,CRC\_ACCUM\_LOW,ID\_GABIM

LCALL ANALYZE\_DATA

MOV TH0,#0FFH

MOV TL0,#00H

MOV IE,#92H

CLR TF0

SETB TR0

SETB P3.2

RETI

ID\_GABIM:

CLR P3.2

MOV R1,#50H

MOV R0,#0

RETI

;=========================================================

; PRANIMI I PAKETIT

;=========================================================

PACKET\_RECEIVE:

MOV R1,#50H

MOV R0,#0

BYTE\_RECEIVE:

MOV A,SBUF

CLR RI

LCALL PARITY\_BIT

PUSH ACC

MOV A,PARITY\_CHECK

JNZ PARITY\_ERROR

POP ACC

MOV @R1,A

INC R0

INC R1

MOV TH0,#HIGH t15

MOV TL0,#LOW t15

SETB TR0

ANALYZE\_T:

JNB TF0,ANALYZE\_RI

SJMP DELAY\_TIMEOUT

ANALYZE\_RI:

JB RI,BYTE\_RECEIVE

SJMP ANALYZE\_T

PARITY\_ERROR:

POP ACC

DELAY\_TIMEOUT:

CLR TF0

DEC R1

DEC R0

MOV TEMP\_CRC\_H,@R1

DEC R1

DEC R0

MOV TEMP\_CRC\_L,@R1

DEC R1

MOV TH0,#HIGH t20

MOV TL0,#LOW t20

SETB TR0

WAIT\_2TCH:JNB TF0,WAIT\_2TCH

RET

;===================================================

; LLOGARITJA E CRC-SE

;===================================================

CRC\_CALC:

MOV A,R0

MOV R3,A

MOV CRC\_ACCUM\_LOW,#0FFH ;fshihet crc accum para fillimit

MOV CRC\_ACCUM\_HI,#0FFH

MOV R1,#50H

PUSH 00H

CC10:

MOV A,@R1

XRL A,CRC\_ACCUM\_LOW

MOV CRC\_ACCUM\_LOW,A

MOV R6,#8 ;R6 sherben si numrues i 8 bitave ne bajt

CC20:

MOV A,CRC\_ACCUM\_HI ;merret bajti i larte

CLR C ;mbushet me 0

RRC A ;shiftohet djathtas

MOV CRC\_ACCUM\_HI,A

MOV A,CRC\_ACCUM\_LOW ;dhe per bajtin e ulet

RRC A ;shiftohet djathtas

MOV CRC\_ACCUM\_LOW,A

JNC CC30

MOV A,CRC\_ACCUM\_LOW

XRL A,#CRC\_MASK\_LSB

MOV CRC\_ACCUM\_LOW,A

MOV A,CRC\_ACCUM\_HI

XRL A,#CRC\_MASK\_MSB

MOV CRC\_ACCUM\_HI,A

CC30:

DJNZ R6,CC20 ;perseritet tete here

INC R1 ;gati per bajtin e ardhshem te mesazhit

DJNZ R3,CC10 ;nje bajt me pak per te llogarite

pop 00H

RET

;=============================================

; ANALIZIMI I TE DHENAVE

;============================================

ANALYZE\_DATA:

MOV R1,#55H

MOV A,@R1

MOV B,#8

DIV AB

MOV R4,A

MOV A,B

CJNE A,#00,INCREASE

RET

INCREASE:

INC R4

RET

;================================================

; PARITY\_CHECK

;================================================

PARITY\_BIT:

MOV C,RB8

MOV 20H,C

MOV C,P

MOV 28H,C

PUSH ACC

MOV A,24H

CJNE A,25H,BYTE\_ERROR

POP ACC

MOV PARITY\_CHECK,#00H

RET

BYTE\_ERROR:

MOV PARITY\_CHECK,#0FFH

RET

;===================================================

; RUTINA PER DERGIMIN E PAKETIT

;===================================================

SEND\_PACKET:

SJMP FIRST

LOOP:

MOV C,P

MOV TB8,C

MOV SBUF,A

WAIT: JNB TI,WAIT

FIRST:

CLR TI

INC DPTR

MOV A,#00

MOVC A,@A+DPTR

CJNE A,#3,LOOP

MOV IE,#90H

CLR P3.2

RET

;==================================================

; PAKETI I PARE- LIGHT ON

;==================================================

ORG 1000H

DB 20H,01H,01H,01H,9AH,74H,3

;==================================================

; PAKETI I DYTE-LIGHT OFF

;==================================================

ORG 1500H

DB 20H,01H,01H,00H,5BH,0B4H,3

END

**4. Kodi në assembler për slave 2:**

CRC\_ACCUM\_LOW EQU 0DH ; bank 1 regjistri R5

CRC\_ACCUM\_HI EQU 0EH ; bank 1 regjistri R6

CRC\_MASK\_LSB EQU 001H ;CRC-16 polinomi

CRC\_MASK\_MSB EQU 0A0H

TEMP\_CRC\_H EQU 40H

TEMP\_CRC\_L EQU 41H

t15 EQU 63952 ;VONESA PREJ 1.5tch

t35 EQU 61848 ;VONESA PREJ 3.5tch

t20 EQU 63416 ;VONESA PREJ 2tch

HI\_ADDRESS EQU 60H

L\_ADDRESS EQU 61H

HI\_VALUE EQU 62H

L\_VALUE EQU 63H

SLAVE\_2\_ID EQU 30H

PAKETI\_GABIM1 EQU 67H

PARITY\_CHECK EQU 42H

ORG 0000H

LJMP MAIN

ORG 000BH

LJMP T0ISR

ORG 0023H

LJMP SERVEC

ORG 0030H

MAIN:

CLR P3.2

MOV SP,#65H

MOV SCON,#0D0H ;TI=1, RI=0, REN=1

MOV TMOD,#21H ;TIMER1=MOD2-AUTORELOAD,TIMER0=MOD1-16BIT

MOV PCON,#00H ;SINGLE BAUD RATE

MOV TH1,#0FDH ;9600 BAUD, #0xF3=>1200 BAUD (12 MHz)

SETB TR1 ;startohet timeri per portin serik

MOV IE,#90H ;lejohet interapti i tajmerit0 dhe iportit serik

CLR TI

LIGHT\_CONTROL:

MOV P2,L\_VALUE

SJMP LIGHT\_CONTROL

;==============================================================

; KTHIMI I PERGJIGJES PAS KALIMIT TE KOHES PREJ 3.5tch

;==============================================================

T0ISR:

CLR TR0

CLR TF0

MOV A,L\_VALUE

JZ PACKET\_SELECT

MOV DPTR,#0FFFH

LCALL SEND\_PACKET

RETI

PACKET\_SELECT:

MOV DPTR,#11FFH

LCALL SEND\_PACKET

RETI

;=========================================================

; PRANIMI I PAKETIT DHE ANALIZIMI

;=========================================================

SERVEC:

CLR TF0

LCALL PACKET\_RECEIVE

MOV R1,#50H

MOV A,@R1

CJNE A,#30H,ID\_GABIM

LCALL CRC\_CALC

MOV A,TEMP\_CRC\_H

CJNE A,CRC\_ACCUM\_HI,ID\_GABIM

MOV A,TEMP\_CRC\_L

CJNE A,CRC\_ACCUM\_LOW,ID\_GABIM

LCALL ANALYZE\_DATA

JZ ID\_GABIM

MOV TH0,#0FFH

MOV TL0,#00H

MOV IE,#92H

CLR TF0

SETB TR0

SETB P3.2

RETI

ID\_GABIM:

CLR P3.2

MOV R1,#50H

MOV R0,#0

RETI

;=========================================================

; PRANIMI I PAKETIT

;=========================================================

PACKET\_RECEIVE:

MOV R1,#50H

MOV R0,#0

BYTE\_RECEIVE:

MOV A,SBUF

CLR RI

LCALL PARITY\_BIT

PUSH ACC

MOV A,PARITY\_CHECK

JNZ PARITY\_ERROR

POP ACC

MOV @R1,A

INC R0

INC R1

MOV TH0,#HIGH t15

MOV TL0,#LOW t15

SETB TR0

ANALYZE\_T:

JNB TF0,ANALYZE\_RI

SJMP DELAY\_TIMEOUT

ANALYZE\_RI:

JB RI,BYTE\_RECEIVE

SJMP ANALYZE\_T

PARITY\_ERROR:

POP ACC

DELAY\_TIMEOUT:

CLR TF0

DEC R1

DEC R0

MOV TEMP\_CRC\_H,@R1

DEC R1

DEC R0

MOV TEMP\_CRC\_L,@R1

DEC R1

MOV TH0,#HIGH t20

MOV TL0,#LOW t20

SETB TR0

WAIT\_2TCH:JNB TF0,WAIT\_2TCH

RET

;===================================================

; LLOGARITJA E CRC-SE

;===================================================

CRC\_CALC:

MOV A,R0

MOV R3,A

MOV CRC\_ACCUM\_LOW,#0FFH ;fshihet crc accum para fillimit

MOV CRC\_ACCUM\_HI,#0FFH

MOV R1,#50H

PUSH 00H

CC10:

MOV A,@R1

XRL A,CRC\_ACCUM\_LOW

MOV CRC\_ACCUM\_LOW,A

MOV R6,#8 ;R6 sherben si numrues i 8 bitave ne bajt

CC20:

MOV A,CRC\_ACCUM\_HI ;merret bajti i larte

CLR C ;mbushet me 0

RRC A ;shiftohet djathtas

MOV CRC\_ACCUM\_HI,A

MOV A,CRC\_ACCUM\_LOW ;dhe per bajtin e ulet

RRC A ;shiftohet djathtas

MOV CRC\_ACCUM\_LOW,A

JNC CC30

MOV A,CRC\_ACCUM\_LOW

XRL A,#CRC\_MASK\_LSB

MOV CRC\_ACCUM\_LOW,A

MOV A,CRC\_ACCUM\_HI

XRL A,#CRC\_MASK\_MSB

MOV CRC\_ACCUM\_HI,A

CC30:

DJNZ R6,CC20 ;perseritet tete here

INC R1 ;gati per bajtin e ardhshem te mesazhit

DJNZ R3,CC10 ;nje bajt me pak per te llogarite

pop 00H

RET

;=============================================

; ANALIZIMI I TE DHENAVE

;============================================

ANALYZE\_DATA:

MOV R1,#51H

MOV A,@R1

CJNE A,#06,FUNKSION\_GABIM

INC R1

MOV A,@R1

MOV HI\_ADDRESS,A

INC R1

MOV A,@R1

MOV L\_ADDRESS,A

INC R1

MOV A,@R1

MOV HI\_VALUE,A

INC R1

MOV A,@R1

MOV L\_VALUE,A

MOV A,#0FFH

RET

FUNKSION\_GABIM:

MOV A,#00H

RET

;================================================

; PARITY\_CHECK

;================================================

PARITY\_BIT:

MOV C,RB8

MOV 20H,C

MOV C,P

MOV 28H,C

PUSH ACC

MOV A,24H

CJNE A,25H,BYTE\_ERROR

POP ACC

MOV PARITY\_CHECK,#00H

RET

BYTE\_ERROR:

MOV PARITY\_CHECK,#0FFH

RET

;===================================================

; RUTINA PER DERGIMIN E PAKETIT

;===================================================

SEND\_PACKET:

SJMP FIRST

LOOP:

MOV C,P

MOV TB8,C

MOV SBUF,A

WAIT: JNB TI,WAIT

FIRST:

CLR TI

INC DPTR

MOV A,#00

MOVC A,@A+DPTR

CJNE A,#3,LOOP

MOV IE,#90H

CLR P3.2

RET

;=====================================================

; PAKETI I PARE- DRITA ESHTE NDEZUR

;=====================================================

ORG 1000H

DB SLAVE\_2\_ID,06H,00H,01H,00H,01H,1DH,0EBH,3

;=====================================================

; PAKETI I PARE- DRITA ESHTE NDALUR

;=====================================================

ORG 1200H

DB SLAVE\_2\_ID,06H,00H,01H,00H,00H,0DCH,2BH,3

END

**5. Kodi në assembler për slave 3:**

CRC\_ACCUM\_LOW EQU 0DH ; bank 1 regjistri R5

CRC\_ACCUM\_HI EQU 0EH ; bank 1 regjistri R6

CRC\_MASK\_LSB EQU 001H ;CRC-16 polinomi

CRC\_MASK\_MSB EQU 0A0H

TEMP\_CRC\_H EQU 40H

TEMP\_CRC\_L EQU 41H

t15 EQU 63952 ;VONESA PREJ 1.5tch

t35 EQU 61848 ;VONESA PREJ 3.5tch

t20 EQU 63416 ;VONESA PREJ 2tch

PAKETI\_GABIM1 EQU 67H

PARITY\_CHECK EQU 42H

ORG 0000H

LJMP MAIN

ORG 000BH

LJMP T0ISR

ORG 0023H

LJMP SERVEC

ORG 0030H

MAIN:

MOV SP,#35H

MOV SCON,#0D0H ;TI=1, RI=0, REN=1

MOV TMOD,#21H ;TIMER1=MOD2-AUTORELOAD,TIMER0=MOD1-16BIT

MOV PCON,#00H ;SINGLE BAUD RATE

MOV TH1,#0FDH ;9600 BAUD, #0xF3=>1200 BAUD (12 MHz)

SETB TR1 ;startohet timeri per portin serik

MOV P0,#00H

MOV IE,#90H ;lejohet interapti i tajmerit0 dhe iportit serik

CLR TI

CLR P3.2

POLLING:

MOV C,P0.0

MOV 00H,C

SJMP POLLING

;==============================================================

; KTHIMI I PERGJIGJES PAS KALIMIT TE KOHES PREJ 3.5tch

;==============================================================

T0ISR:

CLR TR0

CLR TF0

CLR C

MOV C,00H

JC PACKET\_2

MOV DPTR,#14FFH

LCALL SEND\_PACKET

RETI

PACKET\_2:

MOV DPTR,#0FFFH

LCALL SEND\_PACKET

RETI

;=========================================================

; PRANIMI I PAKETIT DHE ANALIZIMI

;=========================================================

SERVEC:

CLR TF0

LCALL PACKET\_RECEIVE

MOV R1,#50H

MOV A,@R1

CJNE A,#24H,ID\_GABIM

LCALL CRC\_CALC

MOV A,TEMP\_CRC\_H

CJNE A,CRC\_ACCUM\_HI,ID\_GABIM

MOV A,TEMP\_CRC\_L

CJNE A,CRC\_ACCUM\_LOW,ID\_GABIM

LCALL ANALYZE\_DATA

MOV TH0,#0FFH

MOV TL0,#00H

MOV IE,#92H

CLR TF0

SETB TR0

SETB P3.2

RETI

ID\_GABIM:

CLR P3.2

MOV R1,#50H

MOV R0,#0

RETI

;=========================================================

; PRANIMI I PAKETIT

;=========================================================

PACKET\_RECEIVE:

MOV R1,#50H

MOV R0,#0

BYTE\_RECEIVE:

MOV A,SBUF

CLR RI

LCALL PARITY\_BIT

PUSH ACC

MOV A,PARITY\_CHECK

JNZ PARITY\_ERROR

POP ACC

MOV @R1,A

INC R0

INC R1

MOV TH0,#HIGH t15

MOV TL0,#LOW t15

SETB TR0

ANALYZE\_T:

JNB TF0,ANALYZE\_RI

SJMP DELAY\_TIMEOUT

ANALYZE\_RI:

JB RI,BYTE\_RECEIVE

SJMP ANALYZE\_T

PARITY\_ERROR:

POP ACC

DELAY\_TIMEOUT:

CLR TF0

DEC R1

DEC R0

MOV TEMP\_CRC\_H,@R1

DEC R1

DEC R0

MOV TEMP\_CRC\_L,@R1

DEC R1

MOV TH0,#HIGH t20

MOV TL0,#LOW t20

SETB TR0

WAIT\_2TCH:JNB TF0,WAIT\_2TCH

RET

;===================================================

; LLOGARITJA E CRC-SE

;===================================================

CRC\_CALC:

MOV A,R0

MOV R3,A

MOV CRC\_ACCUM\_LOW,#0FFH ;fshihet crc accum para fillimit

MOV CRC\_ACCUM\_HI,#0FFH

MOV R1,#50H

PUSH 00H

CC10:

MOV A,@R1

XRL A,CRC\_ACCUM\_LOW

MOV CRC\_ACCUM\_LOW,A

MOV R6,#8 ;R6 sherben si numrues i 8 bitave ne bajt

CC20:

MOV A,CRC\_ACCUM\_HI ;merret bajti i larte

CLR C ;mbushet me 0

RRC A ;shiftohet djathtas

MOV CRC\_ACCUM\_HI,A

MOV A,CRC\_ACCUM\_LOW ;dhe per bajtin e ulet

RRC A ;shiftohet djathtas

MOV CRC\_ACCUM\_LOW,A

JNC CC30

MOV A,CRC\_ACCUM\_LOW

XRL A,#CRC\_MASK\_LSB

MOV CRC\_ACCUM\_LOW,A

MOV A,CRC\_ACCUM\_HI

XRL A,#CRC\_MASK\_MSB

MOV CRC\_ACCUM\_HI,A

CC30:

DJNZ R6,CC20 ;perseritet tete here

INC R1 ;gati per bajtin e ardhshem te mesazhit

DJNZ R3,CC10 ;nje bajt me pak per te llogarite

pop 00H

RET

;=============================================

; ANALIZIMI I TE DHENAVE

;============================================

ANALYZE\_DATA:

MOV R1,#55H

MOV A,@R1

MOV B,#8

DIV AB

MOV R4,A

MOV A,B

CJNE A,#00,INCREASE

RET

INCREASE:

INC R4

RET

;================================================

; PARITY\_CHECK

;================================================

PARITY\_BIT:

MOV C,RB8

MOV 20H,C

MOV C,P

MOV 28H,C

PUSH ACC

MOV A,24H

CJNE A,25H,BYTE\_ERROR

POP ACC

MOV PARITY\_CHECK,#00H

RET

BYTE\_ERROR:

MOV PARITY\_CHECK,#0FFH

RET

;===================================================

; RUTINA PER DERGIMIN E PAKETIT

;===================================================

SEND\_PACKET:

SJMP FIRST

LOOP:

MOV C,P

MOV TB8,C

MOV SBUF,A

WAIT: JNB TI,WAIT

FIRST:

CLR TI

INC DPTR

MOV A,#00

MOVC A,@A+DPTR

CJNE A,#3,LOOP

MOV IE,#90H

CLR P3.2

RET

;==================================================

; PAKETI I PARE- LIGHT ON

;==================================================

ORG 1000H

DB 24H,01H,01H,01H,9BH,44H,3

;==================================================

; PAKETI I DYTE-LIGHT OFF

;==================================================

ORG 1500H

DB 24H,01H,01H,00H,5AH,084H,3

END

**6. Kodi në assembler për slave 4:**

CRC\_ACCUM\_LOW EQU 0DH ; bank 1 regjistri R5

CRC\_ACCUM\_HI EQU 0EH ; bank 1 regjistri R6

CRC\_MASK\_LSB EQU 001H ;CRC-16 polinomi

CRC\_MASK\_MSB EQU 0A0H

TEMP\_CRC\_H EQU 40H

TEMP\_CRC\_L EQU 41H

t15 EQU 63952 ;VONESA PREJ 1.5tch

t35 EQU 61848 ;VONESA PREJ 3.5tch

t20 EQU 63416 ;VONESA PREJ 2tch

PAKETI\_GABIM1 EQU 67H

PARITY\_CHECK EQU 42H

ORG 0000H

LJMP MAIN

ORG 000BH

LJMP T0ISR

ORG 0023H

LJMP SERVEC

ORG 0030H

MAIN:

MOV SP,#35H

MOV SCON,#0D0H ;TI=1, RI=0, REN=1

MOV TMOD,#21H ;TIMER1=MOD2-AUTORELOAD,TIMER0=MOD1-16BIT

MOV PCON,#00H ;SINGLE BAUD RATE

MOV TH1,#0FDH ;9600 BAUD, #0xF3=>1200 BAUD (12 MHz)

SETB TR1 ;startohet timeri per portin serik

MOV P0,#00H

MOV IE,#90H ;lejohet interapti i tajmerit0 dhe iportit serik

CLR TI

CLR P3.2

POLLING:

MOV C,P0.0

MOV 00H,C

SJMP POLLING

;==============================================================

; KTHIMI I PERGJIGJES PAS KALIMIT TE KOHES PREJ 3.5tch

;==============================================================

T0ISR:

CLR TR0

CLR TF0

CLR C

MOV C,00H

JC PACKET\_2

MOV DPTR,#14FFH

LCALL SEND\_PACKET

RETI

PACKET\_2:

MOV DPTR,#0FFFH

LCALL SEND\_PACKET

RETI

;=========================================================

; PRANIMI I PAKETIT DHE ANALIZIMI

;=========================================================

SERVEC:

CLR TF0

LCALL PACKET\_RECEIVE

MOV R1,#50H

MOV A,@R1

CJNE A,#28H,ID\_GABIM

LCALL CRC\_CALC

MOV A,TEMP\_CRC\_H

CJNE A,CRC\_ACCUM\_HI,ID\_GABIM

MOV A,TEMP\_CRC\_L

CJNE A,CRC\_ACCUM\_LOW,ID\_GABIM

LCALL ANALYZE\_DATA

MOV TH0,#0FFH

MOV TL0,#00H

MOV IE,#92H

CLR TF0

SETB TR0

SETB P3.2

RETI

ID\_GABIM:

CLR P3.2

MOV R1,#50H

MOV R0,#0

RETI

;=========================================================

; PRANIMI I PAKETIT

;=========================================================

PACKET\_RECEIVE:

MOV R1,#50H

MOV R0,#0

BYTE\_RECEIVE:

MOV A,SBUF

CLR RI

LCALL PARITY\_BIT

PUSH ACC

MOV A,PARITY\_CHECK

JNZ PARITY\_ERROR

POP ACC

MOV @R1,A

INC R0

INC R1

MOV TH0,#HIGH t15

MOV TL0,#LOW t15

SETB TR0

ANALYZE\_T:

JNB TF0,ANALYZE\_RI

SJMP DELAY\_TIMEOUT

ANALYZE\_RI:

JB RI,BYTE\_RECEIVE

SJMP ANALYZE\_T

PARITY\_ERROR:

POP ACC

DELAY\_TIMEOUT:

CLR TF0

DEC R1

DEC R0

MOV TEMP\_CRC\_H,@R1

DEC R1

DEC R0

MOV TEMP\_CRC\_L,@R1

DEC R1

MOV TH0,#HIGH t20

MOV TL0,#LOW t20

SETB TR0

WAIT\_2TCH:JNB TF0,WAIT\_2TCH

RET

;===================================================

; LLOGARITJA E CRC-SE

;===================================================

CRC\_CALC:

MOV A,R0

MOV R3,A

MOV CRC\_ACCUM\_LOW,#0FFH ;fshihet crc accum para fillimit

MOV CRC\_ACCUM\_HI,#0FFH

MOV R1,#50H

PUSH 00H

CC10:

MOV A,@R1

XRL A,CRC\_ACCUM\_LOW

MOV CRC\_ACCUM\_LOW,A

MOV R6,#8 ;R6 sherben si numrues i 8 bitave ne bajt

CC20:

MOV A,CRC\_ACCUM\_HI ;merret bajti i larte

CLR C ;mbushet me 0

RRC A ;shiftohet djathtas

MOV CRC\_ACCUM\_HI,A

MOV A,CRC\_ACCUM\_LOW ;dhe per bajtin e ulet

RRC A ;shiftohet djathtas

MOV CRC\_ACCUM\_LOW,A

JNC CC30

MOV A,CRC\_ACCUM\_LOW

XRL A,#CRC\_MASK\_LSB

MOV CRC\_ACCUM\_LOW,A

MOV A,CRC\_ACCUM\_HI

XRL A,#CRC\_MASK\_MSB

MOV CRC\_ACCUM\_HI,A

CC30:

DJNZ R6,CC20 ;perseritet tete here

INC R1 ;gati per bajtin e ardhshem te mesazhit

DJNZ R3,CC10 ;nje bajt me pak per te llogarite

pop 00H

RET

;=============================================

; ANALIZIMI I TE DHENAVE

;============================================

ANALYZE\_DATA:

MOV R1,#55H

MOV A,@R1

MOV B,#8

DIV AB

MOV R4,A

MOV A,B

CJNE A,#00,INCREASE

RET

INCREASE:

INC R4

RET

;================================================

; PARITY\_CHECK

;================================================

PARITY\_BIT:

MOV C,RB8

MOV 20H,C

MOV C,P

MOV 28H,C

PUSH ACC

MOV A,24H

CJNE A,25H,BYTE\_ERROR

POP ACC

MOV PARITY\_CHECK,#00H

RET

BYTE\_ERROR:

MOV PARITY\_CHECK,#0FFH

RET

;===================================================

; RUTINA PER DERGIMIN E PAKETIT

;===================================================

SEND\_PACKET:

SJMP FIRST

LOOP:

MOV C,P

MOV TB8,C

MOV SBUF,A

WAIT: JNB TI,WAIT

FIRST:

CLR TI

INC DPTR

MOV A,#00

MOVC A,@A+DPTR

CJNE A,#3,LOOP

MOV IE,#90H

CLR P3.2

RET

;==================================================

; PAKETI I PARE- LIGHT ON

;==================================================

ORG 1000H

DB 28H,01H,01H,01H,98H,14H,3

;==================================================

; PAKETI I DYTE-LIGHT OFF

;==================================================

ORG 1500H

DB 28H,01H,01H,00H,59H,0D4H,3

END