

# ZAPISOVANJE DOGODKOV NA PARKIRIŠČU

## 1. Dogodki se zapisujejo pri:

1. Odpiranju rampe
  - vzrok odpiranja
  - stanje magnetnih senzorjev
2. Zapiranju rampe
  - vzrok zapiranja
  - stanje magnetnih senzorjev
3. Napaka pri odpiranju rampe
  - rampa se ni odprla ali se ni pravočasno odprla
  - stanje magnetnih senzorjev
  - stanje končnih stikal na rampi
4. Napaka pri zapiranju rampe
  - rampa se ni zaprla ali se ni pravočasno zaprla
  - stanje magnetnih senzorjev
  - stanje končnih stikal na rampi
5. Napaka na blagajni
  - napaka na tiskalniku
  - napaka na citalcu denarja
  - napaka na mehanizmu za kartice
  - zmanjkalo je kovancev za vračanje
6. Neveljavnost kartice - abonentske
  - neznan abonentska kartica (kartica ni prijavljena)
  - kartica je pretekla
  - prekoračitev časa
7. Napaka dvojni prehod pri abonentski kartici
8. Plačilo parkiranja pri kartici - blagajna
9. Plačilo, podaljšanje abonentske ali prehodne kartice - blagajna
10. Zataknjena kartica na vhodnem terminalu
11. Zmanjkalo je kartic ali papirja na vhodnem terminalu
12. Parkirisce je polno
13. Poročilo o stanju prometa
14. Vpis nastavitvev in baze kartic

## 2. Protokol serijskega prenosa

### 2.1 Dogodek [**dogodek**,predmet]

- 0=Servis
- 1=Odpiranje rampe
- 2=Zapiranje rampe
- 3=Neznana kartica (abonentska kartica ni prijavljena)
- 4=Kartica je pretekla
- 5=Prekoračitev časa
- 6=Ni kartic/Ni papirja
- 7=Parkirišče je polno
- 8=Dvojni prehod pri abonentski kartici
- 9=Plačilo ali podaljsanje kartice
- 10=nalet na rampo pri zapiranju
- 11=Stanje parkirišča
- 12=Promet blagajne – preverjanje števcov prometa vplačil
- 13=Napake
- 14=Stanje hoperja
- 15=Odpiranje vrat

### 2.2 Predmet dogodka [**dogodek**,**predmet**]

- 1=Master kartica
- 2=Abonentska kartica
- 3=Navadna kartica
- 4=Čas / Hoper 2 stetje – za stanje tik pred štetjem kovancev hoperja
- 5=Intervencija / hoper 2 stanje
- 6=Prehodna kartica
- 7= GSM\_intervencija / Hoper 2 vnos – stanje takoj po vnosu kovancev
- 8=Barlistek
- 9=Skupni promet
- 10=Dnevni promet
- 11=Hoper1 stanje
- 12=Hoper 1 vnos – stanje takoj po vnosu kovancev
- 13=Hoper 1 stetje – za stanje tik pred štetjem kovancev hoperja
- 14=Alarm – pri odpiranju vrat

### 2.3 Smer dogodka [**smer**,napake]

- 16=GSM modem
- 32 = blagajna
- 64 = vhod
- 128=izhod

## 2.4 Napake [smer,**napake**] - postavljanje posameznih bitov

### VHOD

- b0=Napaka pri zapiranjah rampe
- b1=Napaka pri odpiranju rampe
- b2=Rampa je odprta več kot 40 sekund
- b3=Napaka podajalca kartic
- b4= pri RFID : napaka\_mehanizem  
pri barlistku : napaka na tiskalniku ali malo papirja, ni papirja
- b5= napaka\_rfid\_citalca
- b6= napaka\_m\_zanke
- b7=napaka\_stikala\_rampe

### IZHOD

- b0=Napaka pri zapiranjah rampe
- b1=Napaka pri odpiranju rampe
- b2=Rampa odprta več kot 40 sekund
- b3=
- b4= napaka\_mehanizem
- b5= napaka\_rfid\_citalca
- b6= napaka\_m\_zanke
- b7=napaka\_stikala\_rampe

### BLAGAJNA

- b0=napaka na tiskalniku
- b1=napaka na citalcu
- b2= napaka papir
- b3= ni kovancev 1 za vračanje
- b4= pri RFID : napaka\_mehanizem  
pri barlistku : napaka na tiskalniku ali malo papirja, ni papirja
- b5= napaka\_rfid\_citalca
- b6= ni kovancev 2 za vračanje
- b7= alarm – odpiranje vrat

## 2.5 Čas in datum dogodka [**sek**][**min**][**ura**][**dan**][**mes,let**]

- sekunde,minute,ure,dan,mesec,leto

## 2.6 Stanja senzorjev in stikal [**senzorji**] – postavljanje posameznih bitov

- b0=prva magnetna zanka
- b1=druga magnetna zanka
- b2=zgornje končno stikalo za rampo
- b3=spodnje končno stikalo za rampo
- b4=neznano stanje zank – napaka prenosa
- b5=stanje prve magnetne zanke pred komando zapiranja ali odpiranja
- b6=stanje druge magnetne zanke pred komando zapiranja ali odpiranja

## 2.7 PROTOKOL:

### Prenos podatkov poteka po naslednjem principu:

Nadzorna naprava (dlančnik) kliče posamezne terminale z 12 baytnim nizom, ti pa odgovarjajo z 20 baytnim nizom podatkov, ce so ti zapisani v terminalu.

Nadzorna naprava kliče terminale 4 krat v sekundi. Maksimalno število terminalov je 15. (5x vhod, 5x izhod in 5x blagajna).

Maximalna število klicev je 11. Ko je več kot 11 terminalov deli 11. klic na terminale od 11 do 15 (to so blagajne, ker zapisujejo manj dogodkov). Posamezni terminali morajo biti označeni od 1 do 5.

### **Klicni niz**

Dolžina klicnega protokola je 11 baytov + 1bayt CRC.

**{{vrsta terminala}[enota+1][enota+2][enota+3] .... [enota+8][ukaz][0][CRC]}**

CRC je vsota vseh baytov razen CRC.

### Vrsta terminala

- 30 vhodni terminal
- 62 izhodni terminal
- 126 blagajna

### Ukazi

- 1 = citanje dogodka
- 3 = registracija posameznih terminalov
- 5 = GSM intervencija odpiranje rampe
- 7 = GSM intervencija zapiranje rampe
- 9 = preverjanje stanjea prometa ali stanja hoperja
- 11 = preverjanje RS485 komunikacije z terminalom
- 18 = vpis časa in datuma v parkirni terminal
- 128 = citanje prijav abonentskih kartic iz terminala
- 138 = citanje nastavitv iz vhodnega terminala
- 228 = vpis prijav abonentskih kartic v terminal
- 148 = citanje nastavitv iz blagajne
- 248 = vpis nastavitv v blagajno
- 238 = vpis nastavitv v vhodni terminal

## Podatkovni niz

Dolžina podatkovnega niza je 18 baytov podatkov + 2bayta CRC. CRC se uporablja samo pri prenosu, shranjuje pa se le 18 baytni niz podatkov.

### -Vhod in izhod:

{{dogodek,predmet}[[smer,napakal][min][ura][dan][mes,let][sek][SNK0][SNK1][SNK2][SNK3] ][senzorji][K.P.L][K.P.H][Z.P.L][ZPH][0][napakeh,st. enote]}

CHK-se prenaša samo ob komunikaciji

```
DATA[0]=(dogodek << 4) + predmet;
DATA[1]=smer | (napaka & 0x0F);
DATA[2]=MIN;
DATA[3]=URA;
DATA[4]=DAN;
DATA[5]=(MES << 4) + LET;
DATA[6]=SEK;
DATA[7]=SNK[0];
DATA[8]=SNK[1];
DATA[9]=SNK[2];
DATA[10]=SNK[3];
DATA[11]=senzor;
DATA[12]=KAPACITETA_PARKIRISCA_L;
DATA[13]=KAPACITETA_PARKIRISCA_H;
DATA[14]=ZASEDENOST_PARKIRISCA_L;
DATA[15]=ZASEDENOST_PARKIRISCA_H;
DATA[16]=0;
DATA[17]=PARKPLAC | (napaka & 0xF0);
CRC_L
CRC_H
CRC=DATA[0]+.....DATA[17] // CRC je vsota vseh podatkov
```

## -KASA

PAKET ZA TRANSAKCIJE in NAPAKE:

{[dogodek,predmet][smer,napakal][min][ura][dan][mes,let][vplacilo][SNK0][SNK1][SNK2]  
[SNK3] [vracilo][zetoni][GSM vplacilo][st. racuna L][st. racuna H][0][napakeh,st. blagajne]}

**CHK-se prenaša samo ob komunikaciji**

```
DATA[0]=(dogodek << 4) + predmet;  
DATA[1]=smer | (napake & 0x0F);  
DATA[2]=MIN;  
DATA[3]=URA;  
DATA[4]=DAN;  
DATA[5]=(MES << 4) + LET;  
DATA[6]=KREDIT_L;  
DATA[7]=SNK[0];  
DATA[8]=SNK[1];  
DATA[9]=SNK[2];  
DATA[10]=SNK[3];  
DATA[11]=VRACILO;  
DATA[12]=KREDIT_H;          // ZETONI/ENOTA_ZETON;  
DATA[13]=GSM_KREDIT;  
DATA[14]=st_rac & 0xFF;  
DATA[15]=st_rac>>8;  
DATA[16]=0;// BAN_KARD;  
DATA[17]=PARKPLAC | (napake & 0xF0);  
CRC_L  
CRC_H+128  
  
CRC=DATA[0]+.....DATA[17]          // CRC je vsota vseh podatkov
```

vplacilo je podano v enotah (po 50 SIT)  
vracilo je stevilo enot zaradi preplacila

[ b e s e d a ] ..... 8 bit  
[beseda1,beseda2] .... H\_4bit,L\_4 bit  
[beseda1 H] .....visjih 8 bitov besede  
[beseda2 L] .....nizjih 8 bitov besede

SNK je serijska številka abonentske kartice  
CHC= vsota vseh baytov, prenos se zgubi

#### PAKET ZA STEVCE SKUPNEGA IN DNEVNEGA PROMETA:

**dogodek=promet blagajne, predmet=skupni promet ali dnevni promet**

```
DATA[0]=(dogodek << 4) + predmet;
DATA[1]=smer | (napake & 0x0F);
DATA[2]=MIN;
DATA[3]=URA;
DATA[4]=DAN;
DATA[5]=(MES << 4) + LET;
DATA[6]=promet_zetoni_L;
DATA[7]=promet_zetoni_H;
DATA[8]=promet_GSM_L;
DATA[9]=promet_GSM_H;
DATA[10]=promet_GSM_HH;
DATA[11]=promet_gotovina_L;
DATA[12]=promet_gotovina_H;
DATA[13]=promet_gotovina_HH;
DATA[14]=promet_vracilo_L;
DATA[15]=promet_vracilo_H;
DATA[16]=promet_vracilo_HH;
DATA[17]=PARKPLAC | (napake & 0xF0);
CRC_L
CRC_H+128

CRC=DATA[0]+.....DATA[17] // CRC je vsota vseh podatkov
```

#### PAKET ZA STANJE KOVANECV ZA VRACANJE DENARJA:

```
DATA[0]=(dogodek << 4) + predmet;
DATA[1]=smer | (napake & 0x0F);
DATA[2]=MIN;
DATA[3]=URA;
DATA[4]=DAN;
DATA[5]=(MES << 4) + LET;
DATA[6]=stanje_kovancev1_za_vracanje_L;
DATA[7]=stanje_kovancev1_za_vracanje_H;
DATA[8]=stanje_kovancev2_za_vracanje_L;
DATA[9]=stanje_kovancev2_za_vracanje_H;
DATA[10]=0;
DATA[11]=0;
DATA[12]=0;
DATA[13]=0;
DATA[14]=0;
DATA[15]= st_rac & 0xFF;
DATA[16]= st_rac>>8;
DATA[17]=PARKPLAC | (napake & 0xF0);
CRC_L
CRC_H+128

CRC=DATA[0]+.....DATA[17] // CRC je vsota vseh podatkov
```

#### VNOS KOVANECV HOPERJA SE IZRACUNA IZ RAZLIKE STANJ:

**VNOS = dogodek(predmet=vnos\_hoper) - dogodek(predmet=stanje\_hoperja)**

**Oba dogodka imata enak čas in datum in številko računa !**

#### **Izračun vrednosti števecv :**

**Vrednost = števec\_L + števec\_H x 256 + števec\_HH x 65536**

## Časovna opredelitev protokola

### Čas pošiljanja nizov:

- klicni niz 13ms
- podatkovni niz 21ms

### Časovna slika protokola:

[klicni\_niz1=13ms][pauza 8ms][klicni\_niz2=13ms][puza 8ms].....[klicni\_niz15=13ms][pauza 21ms]  
[ podatkovni niz 1 21ms ][ ..... ][ podatk. niz]

Terminal odgovori na klic v 1ms.

Čas med klicnimi nizi mora vedno biti daljši ali enaki od podatkovnega niza, ker bi v nasprotnem primeru dva terminala istočasno pošiljala podatke na skupno vodilo.

### 3. RFID KARTICE

Kartica ima 33 memorijskih celic od naslova 0 do 32. Celice so velikosti 32 bitov. Kartica vedno vpisuje in cita eno memorijsko lokacijo. Serijsko številko zapiše proizvajalec kartic. Ima možnost 32 bitnega kodiranja.

#### NAVADNA KARTICA ZA NAKLJUČNO PARKIRANJE

1.Login = 0x01010101;

2.Memorijske lokacije:

- 32 = serijska številka (SN1,SN2,SN3,SN4 – 4 bayt)
- 3 = čas vhoda ali plačila na blagajni

3.Zapis časa, lokacija 3(B1,B2,B3,B4):

O=LETO&32;

LETO=(LETO&15)\*16;

[MINUTE],[URE],[DAN+O],[LETO,MESEC]

#### ABONENTSKA IN MASTER KARTICA

k1=1;

k2=0x11;

DATA1=255-SN1+5;

DATA2=SN2+17;

DATA3=255-SN3+33;

DATA4=SN4+12;

k3=(DATA1^DATA3)+57;

k4=(DATA2^DATA4)+28;

Upoštevati je treba da so spremenljivke 8 bitne in se prenosi izgubijo

1.Login =k1,k2,k3,k4

2. Memorijske lokacije:

- 32 = serijska številka (SN1,SN2,SN3,SN4 – 4 bayt)
- 5 = datum veljavnosti kartice
- 6 = status prehoda (0x0F 00 00 00 - izhod, 0xF0000000-vhod)
- 4 = tip kartice(0xEE 00 00 00 - abonentska kartica)

3.Zapis datuma veljavnosti kartice, lokacija 5(B1,B2,B3,B4):

O=LETO&32;

LETO=(LETO&15)\*16;

[x],[x],[DAN+O],[LETO,MESEC]

#### MASTER KARTICA

Ne upošteva statusa prehoda zato je možno z njo opraviti več prehodov v isto smer.

Vse je enako kot pri abonentski kartici le da je na memorijski lokaciji 4 zapis 0x64 00 00 00.

#### PREHODNA KARTICA

Je namenjena za določeno število parkiranj (prehodov cez vhodni ali izhodni terminal).

Vse je enako kot pri abonentski kartici le da je na memorijski lokaciji 4 zapis 0xDE 00 00 00.

Na memorijski lokaciji 3 je zapisano število dovoljenih prehodov.

[prehodi][255-prehodi][x][x]

#### **4. Program STATISTIKA18 COMPAQ IPAQ dlančniku**

Je namenjen za shranjevanje, prenos dogodkov, javljanje napak, GSM intervencijo (npr. odpiranje rampe preko SMS sporočila) na parkirišču.

Program avtomatsko tvori datoteko za vsak mesec v oblike **0xmesec leto (0x32003)**.

Posamezne dogodke je možno pregledovati z programom **Obdelava18**. Za celovit pregled podatkov dlančnik preko GSM modema pošilja podatke v računalnik nadzornega centra.

## 5. TCP PRENOS PODATKOV IN UKAZOV

### 5.1 PROTOKOL

[U] [ ] [ št. parkirišča ] [št. ukaza] [izvor] [cilj] [št.cilja] [ukaz] [CRC]

[ U ]	=	črka U	
[ ]	=	presledek	
[ št. parkirišča ]	=	oznaka parkirišča dvomestna številka (primer '0' '1')	
[št.ukaza]	=	številka 0..255 to številko pri ACK vrne modem, da veš na kateri ukaz dobiš odgovor	
[ izvor]	=	VHOD	- 0x1E (30)
		IZHOD	- 0x3E (62)
		BLAGAJNA	- 0x7E (126)
		DATOTEKA	- 0x2E (46)
		UKAZ	- 0x9E (158)
[ cilj]	=	VHOD	- 0x1E (30)
		IZHOD	- 0x3E (62)
		BLAGAJNA	- 0x7E (126)
		DATOTEKA	- 0x2E (46)
[št. izvora ali cilja]	=	TERMINALI	-1,2,3,4,5
[ukaz]	=	<u>MODEM CITA BAZO KARTIC</u>	- 0x80 (128)
		IZ TERMINALA	
		<u>MODEM PRECITA NASTAVITVE</u>	- 0x94 (148)
		IZ BLAGAJNE	
		<u>MODEM PRECITA NASTAVITVE</u>	- 0x8A (138)
		IZ VHODNEGA TERMINALA	
		<u>MODEM VPISIE BAZO KARTIC</u>	- 0xE4 (228)
		V TERMINAL	
		<u>MODEM VPISIE NASTAVITVE V BLAGAJNO</u>	- 0xF8 (248)
		<u>MODEM VPISIE NASTAVITVE V VHODNI T.</u>	- 0xEE (238)
		<u>CENTER NALOZI DATOTEKO BAZE KARTIC</u>	- 0x17 (23)
		V MODEM (XMODEM protokol)	
		<u>CENTER PRECITA DATOTEKO BAZE KARTIC</u>	- 0x37 (55)
		IZ MODEMA (XMODEM protokol)	
		<u>CENTER NALOZI DATOTEKO NASTAVITEV</u>	- 0x19 (25)
		BLAGAJNE V MODEM	
		<u>CENTER NALOZI DATOTEKO NASTAVITEV</u>	- 0x19 (27)
		VHODNEGA TERMINALA V MODEM	
		<u>CENTER PRECITA DATOTEKO NASTAVITEV</u>	- 0x39 (57)
		IZ MODEMA (XMODEM protokol)	
		<u>CENTER PRECITA DATOTEKO NASTAVITEV</u>	- 0x3B (59)
		VHODNEGA TERRMINALA IZ MODEMA	
		(XMODEM protokol)	
		<u>UKAZ-ODPRI RAMPO</u>	- 0x05 (5)

UKAZ-PREGLED STANJA ZASEDENOSTI - 0x09 (9)  
ALI STANJE HOPERJA BLAGAJNE  
UKAZ-PREVERJANJE PRISOTNOSTI - 0x0B (11)  
UKAZ-SINHRONIZACIJA CASA IN DATUMA - 0x12 (18)

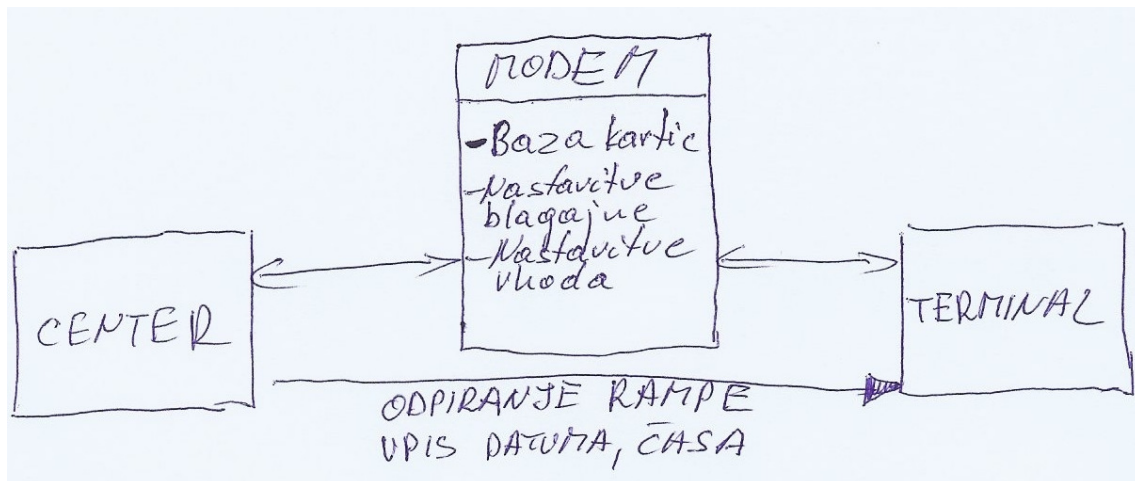
[CRC] = XOR vseh bytov niza razen CRCja

UKAZ-PREVERJANJE PRISOTNOSTI: preveri se prisotnost posameznega terminala.  
Če terminal obstaja vrne ACK, v nasprotnem primeru ni odziva !

UKAZ-STANJE ZASEDENOSTI ALI STANJE HOPERJA:

Na ta ukaz se odziva le tisti vhodni terminal, ki šteje promet na parkirišču (običajno je to VHOD1). Odgovor na ukaz je dogodek stanje prometa. Blagajna vrne dogodek stanja HOPERJA vračanja denarja.

DATOTEKA: baza kartic (dolžina 8192) in nastavitve (dolžina 2048) se vpišeta v FLASH spominu modema.



Center dostopa direktno na terminal le ob vpisu časa in datuma in ob odpiranju zapornice !  
Baze in nastavitve vpišuje v terminale preko datotek, ki jih mora predhodno vpisati v modem!  
Če hoče center prebrati datoteko iz terminala mora najprej poslati ukaz pri katerem modem prebere datoteko iz terminala nato pa center prebere datoteko iz modema !

## 5.2 ODGOVOR NA PAKET

[A] [ ] [št. parkirišča] [št. ukaza] [oznaka] [potrditev] [CRC]

[ A ]	=	črka A
[ ]	=	presledek
[ št. parkirišča ]	=	oznaka parkirišča dvomestna številka (primer '0' '1')
[št.ukaza]	=	številka 1..255 to številko pri ACK vrne modem, da veš na kateri ukaz dobiš odgovor. <b>0 je rezervirana</b> za prijavo terminala ob connectu
[ oznaka]	=	0-struktura paketa 1-komande 2-komunikacije z parkirnim terminalom 3-izvrsitve komande 4-prisotnosti datoteke v modemu
[ potrditev]	=	6 -ACK potrditev 21-NACK ni potrditve (napaka)
[CRC]	=	XOR vseh bytov niza razen CRCja

### 5.3 PRIMERI KOMUNIKACIJE CENTRA – MODEMA

Številka izvora je v naslednjih primerih nepomemben podatek !

- center naloži datoteko baze kartic v GSM modem parkirišča 01  
(datoteka centra → datoteka modema):  
**poslani niz** = [85] [32] [48,49] [0][46] [46] [1] [23] [98]  
**odgovor** = timeout=10s, ACK(1), NACK(0)  
**XMODEM protokol**
- center prečita iz GSM modema datoteko baze kartic parkirišča 01  
(datoteka centra ← datoteka modema):  
**poslani niz** = [85] [32] [48,49] [1][46] [46] [5] [55] [71]  
**odgovor** = timeout=10s , ACK(1), NACK(0,4)  
**XMODEM protokol**
- center naloži datoteko nastavitv v GSM modem parkirišča 01  
(datoteka centra → datoteka modema):  
**poslani niz** = [85] [32] [48,49] [1][46] [46] [5] [25] [71]  
**odgovor** = timeout=10s , ACK(1), NACK(0)  
**XMODEM protokol**
- center naloži datoteko nastavitv vhodnega terminala v GSM modem parkirišča 01  
(datoteka centra → datoteka modema):  
**poslani niz** = [85] [32] [48,49] [1][46] [46] [5] [27] [107]  
**odgovor** = timeout=10s , ACK(1), NACK(0)  
**XMODEM protokol**
- center prečita iz GSM modema datoteko nastavitv vhodnega terminala parkirišča 01  
(datoteka centra ← datoteka modema):  
**poslani niz** = [85] [32] [48,49] [1][46] [46] [5] [59] [75]  
**odgovor** = timeout=10s , ACK(1), NACK(0,4)
- center prečita iz GSM modema datoteko nastavitv parkirišča 01  
(datoteka centra ← datoteka modema):  
**poslani niz** = [85] [32] [48,49] [1][46] [46] [5] [57] [71]  
**odgovor** = timeout=10s , ACK(1), NACK(0,4)  
**XMODEM protokol**

## 5.4 PRIMERI KOMUNIKACIJE MODEMOM – TERMINAL

- modem prečita bazo kartic iz vhodnega terminala 1  
(terminal vhod → datoteka modema):  
**poslani niz** = [85] [32] [48,49] [3][30] [46] [1] [128] [198]  
**odgovor** = timeout1=10s ,ACK(1), NACK(0)  
**odgovor** = timeout2=45s ,NACK(2)  
**odgovor** = timeout3=60s ,ACK(3), NACK(3)
  
- modem prečita bazo kartic iz izhodnega terminala 1  
(terminal izhod → datoteka modema):  
**poslani niz** = [85] [32] [48,49] [3][62] [46] [1] [128] [230]  
**odgovor** = timeout1=10s ,ACK(1), NACK(0)  
**odgovor** = timeout2=45s ,NACK(2)  
**odgovor** = timeout3=60s ,ACK(3), NACK(3)
  
- modem prečita bazo kartic iz blagajne 1  
(terminal blagajna → datoteka modema):  
**poslani niz** = [85] [32] [48,49] [3][126] [46] [1] [128] [166]  
**odgovor** = timeout1=10s ,ACK(1), NACK(0)  
**odgovor** = timeout2=45s ,NACK(2)  
**odgovor** = timeout3=60s ,ACK(3), NACK(3)
  
- modem vpiše bazo kartic v vhodni terminal 1  
(terminal vhod ← datoteka modema):  
**poslani niz** = [85] [32] [48,49] [4][46] [30] [1] [228] [114]  
**odgovor** = timeout1=10s ,ACK(1), NACK(0,4)  
**odgovor** = timeout2=45s ,NACK(2)  
**odgovor** = timeout3=60s ,ACK(3), NACK(3)
  
- modem vpiše bazo kartic v izhodni terminal 2  
(terminal izhod ← datoteka modema):  
**poslani niz** = [85] [32] [48,49] [4][46] [62] [2] [228] [134]  
**odgovor** = timeout1=10s ,ACK(1), NACK(0,4)  
**odgovor** = timeout2=45s ,NACK(2)  
**odgovor** = timeout3=60s ,ACK(3), NACK(3)
  
- modem vpiše bazo kartic v blagajno 2  
(terminal blagajna ← datoteka modema):  
**poslani niz** = [85] [32] [48,49] [4][46] [126] [2] [228] [194]  
**odgovor** = timeout1=10s ,ACK(1), NACK(0,4)  
**odgovor** = timeout2=45s ,NACK(2)  
**odgovor** = timeout3=60s ,ACK(3), NACK(3)
  
- prečitaj nastavitve iz vhoda 2 in jo vpiši v datoteko nastavitvev modema  
(terminal vhod → datoteka modema):  
**poslani niz** = [85] [32] [48,49] [5][30] [46] [2] [148] [215]  
**odgovor** = timeout1=10s ,ACK(1), NACK(0)  
**odgovor** = timeout2=45s ,NACK(2)  
**odgovor** = timeout3=60s ,ACK(3), NACK(3)

- prečita nastavitve iz blagajne 1 in jo vpiši v datoteko nastavitvev modema (terminal blagajna → datoteka modema):

**poslani niz** = [85] [32] [48,49] [5][126] [46] [2] [138] [169]  
**odgovor** = timeout1=10s ,ACK(1), NACK(0)  
**odgovor** = timeout2=45s ,NACK(2)  
**odgovor** = timeout3=60s ,ACK(3), NACK(3)

- modem vpiše datoteko nastavitvev v vhodni terminal 1 (terminal vhod ← datoteka modema):

**poslani niz** = [85] [32] [48,49] [6][46] [30] [1] [238] [173]  
**odgovor** = timeout1=10s ,ACK(1), NACK(0,4)  
**odgovor** = timeout2=45s ,NACK(2)  
**odgovor** = timeout3=60s ,ACK(3), NACK(3)

- modema vpiše datoteko nastavitvev v blagajno 1 (terminal blagajna ← datoteka modema):

**poslani niz** = [85] [32] [48,49] [6][46] [126] [1] [248] [219]  
**odgovor** = timeout1=10s ,ACK(1), NACK(0,4)  
**odgovor** = timeout2=45s ,NACK(2)  
**odgovor** = timeout3=60s ,ACK(3), NACK(3)

## 5.5 PRIMERI KOMUNIKACIJE CENTR - TERMINAL

- center odpre rampo na vhodnem terminalu 1  
(center ukaz → terminal):  
**poslani niz** = [85] [32] [48,49] [2][158] [30] [1] [5] [242]  
**odgovor** = timeout1=10s ,ACK(1), NACK(0)  
**odgovor** = timeout2=45s , ACK(3), NACK(2,3)
- center odpre rampo na izhodnem terminalu 2  
(center ukaz → terminal):  
**poslani niz** = [85] [32] [48,49] [2][158] [62] [2] [5] [209]  
**odgovor** = timeout1=10s ,ACK(1), NACK(0)  
**odgovor** = timeout2=45s , ACK(3), NACK(2,3)
- center poslje cas in datum v terminal izhod 2  
(center ukaz → terminal):  
**poslani niz** = [85] [32] [48,49] [2][158] [62] [2] [18] [198]  
**k ukaznemu nizu mora biti pripet niz časa in datuma:**  
**[SEK] [MIN] [URA] [DAN] [MES] [LET] [DAN\_TED] [CRC]**  
**odgovor** = timeout1=10s ,ACK(1)  
**odgovor** = timeout2=45s , ACK(3), NACK(2,3)  
**CRC EXOR vseh bytov časa in datuma razen CRC**
- center poslje ukaz za preverjanje stanja prometa na vhod 1  
(center ukaz → terminal):  
**poslani niz** = [85] [32] [48,49] [2][158] [30] [1] [9] [254]  
**odgovor** = timeout1=10s ,ACK(1)  
**odgovor** = timeout2=45s , ACK(3), NACK(2,3)  
**ACK(1)** = sledi zapis dogodka stanja prometa na parkirišču
- center poslje ukaz za preverjanje prisotnosti terminala vhod 2  
(center ukaz → terminal):  
**poslani niz** = [85] [32] [48,49] [2][158] [30] [2] [11] [244]  
**odgovor** = timeout1=10s ,ACK(1)  
**odgovor** = timeout2=45s , ACK(3), NACK(2,3)
- center poslje ukaz za preverjanje stanje hoperja na blagajni 2  
(center ukaz → terminal):  
**poslani niz** = [85] [32] [48,49] [2][158] [126] [2] [9] [255]  
**odgovor** = timeout1=10s ,ACK(1)  
**odgovor** = timeout2=45s , ACK(3), NACK(2,3)  
**ACK(1)** = sledi zapis dogodka stanja prometa na parkirišču

## 5.6 CONNECT MODEMA NA CENTER

**Modem se ob konektu na center predstavi z predstavitvenim nizom !**

**Predstavitveni niz :**

[65] [32] [št. parkirišča] [0] [0] [6] [CRC]

[ št. parkirišča ] = oznaka parkirišča dvomestna številka (primer '0' '1')

Primer, modem parkirisca 01 se skonekta na center:

[65] [32] [48] [49] [0] [0] [6] [103]

## 5.7 PRENOS DOGODKOV

**Po predstavitvenem nizu sledijo paketi dogodkov:**

[P] [ ] [št. parkirišča] + standarni paket dogodka 18 byte + CRC

**(glej zgoraj Podatkovni niz)**

[ št. parkirišča ] = oznaka parkirišča dvomestna številka (primer '0' '1')

## 6. XModem Protocol z CRC za prenos datotek med centrom in modemom

### Introduction

The Xmodem protocol was created years ago as a simple means of having two computers talk to each other. With its half-duplex mode of operation, 128- byte packets (1024- byte packets 1K XMODEM), ACK/NACK responses and CRC data checking, the Xmodem protocol has found its way into many applications. In fact most communication packages found on the PC today have a Xmodem protocol available to the user.

### Theory of Operation

Xmodem is a half-duplex communication protocol. The receiver, after receiving a packet, will either acknowledge (ACK) or not acknowledge (NAK) the packet. The CRC extension to the original protocol uses a more robust 16-bit CRC to validate the data block and is used here. Xmodem can be considered to be receiver driven. That is, the receiver sends an initial character 'C' to the sender indicating that it's ready to receive data in CRC mode. The sender then sends a 133-byte packet, the receiver validates it and responds with an ACK or a NAK at which time the sender will either send the next packet or re-send the last packet. This process is continued until an EOT is received at the receiver side and is properly ACKed to the sender. After the initial handshake the receiver controls the flow of data through ACKing and NAKing the sender.

Table 1. XmodemCRC Packet Format

Byte 1	Byte 2	Byte 3	Bytes 4-131	Bytes 132-133
Start of Header	Packet Number	(Packet Number)	Packet Data	16-bit CRC

**Definitions:**The following defines are used for protocol flow control.

Symbol	Description	Value
SOH	Start of Header (128 byte packets)	0x01
STK (SOH)	Start of 1K Header (1024 byte packets 1K XMODEM)	0x02
EOT	End of Transmission	0x04
ACK	Acknowledge	0x06
NAK	Not Acknowledge	0x15
ETB	End of Transmission Block (Return to Amulet OS mode)	0x17
CAN	Cancel (Force receiver to start sending C's)	0x18
C	ASCII 'C'	0x43

Byte 1 of the XmodemCRC packet can only have a value of SOH, EOT, CAN or ETB anything else is an error. Bytes 2 and 3 form a packet number with checksum, add the two bytes together and they should always equal 0xff. Please note that the packet number starts out at 1 and rolls over to 0 if there are more than 255 packets to be received. Bytes 4 - 131 form the data packet and can be anything. Bytes 132 and 133 form the 16-bit CRC. The high byte of the CRC is located in byte 132. The CRC is calculated only on the data packet bytes (4 - 131).

### Synchronization

The receiver starts by sending an ASCII 'C' (0x43) character to the sender indicating it wishes to use the CRC method of block validating. After sending the initial 'C' the receiver waits for either a 3 second time out or until a buffer full flag is set. If the receiver is timed out then another 'C' is sent to the sender and the 3 second time out starts again. This process continues until the receiver receives a complete 133-byte packet.

### Receiver Considerations

This protocol NAKs the following conditions: 1. Framing error on any byte 2. Overrun error on any byte 3. Duplicate packet 4. CRC error 5. Receiver timed out (didn't receive packet within 1 second) On any NAK, the sender will re-transmit the last packet. Items 1 and 2 should be considered serious hardware failures. Verify that sender and receiver are using the samebaud rate, start bits and stop bits. Item 3 is usually the sender getting an ACK garbled and re-transmitting the packet. Item 4 is found in noisy environments. And the last issue should be self-correcting after the receiver NAKs the sender.

Sender						Receiver
					<---	'C'
						Times Out after 3 Seconds
					<---	'C'
SOH	0x01	0xFE	Data	CRC	--->	Packet OK
					<---	ACK
SOH	0x02	0xFD	Data	CRC	--->	(Line Hit during Data Transmission)
					<---	NACK
SOH	0x02	0xFD	Data	CRC	--->	Packet OK
					<---	ACK
SOH	0x03	0xFC	Data	CRC	--->	Packet OK
(ACK Gets Garbled)					<---	ACK
					<---	ACK
SOH	0x04	0xFB	Data	CRC	--->	(UART Framing Error on Any Byte)
					<---	NACK
SOH	0x04	0xFB	Data	CRC	--->	Packet OK
					<---	ACK
SOH	0x05	0xFA	Data	CRC	--->	(UART Overrun Error on Any Byte)
					<---	NACK
SOH	0x05	0xFA	Data	CRC	--->	Packet OK
					<---	ACK
EOT					--->	Packet OK
					<---	ACK
ETB					--->	Finished
Finished					<---	ACK

### Sample crc calculation code

```
int calcrc(char *ptr, int count) // calcrc(char *polje, int dolzina)
{
    // XMODEM dolzina=128
    // 1K XMODEM dolzina=1024
    int crc;
    char i;
    crc = 0;
    while (--count >= 0)
    {
        crc = crc ^ (int) *ptr++ << 8;
        i = 8;
        do
        {
            if (crc & 0x8000)
                crc = crc << 1 ^ 0x1021;
            else
                crc = crc << 1;
        } while(--i);
    }
    return (crc);
}
```

**Če sender pošlje namesto SOH STK, to pomeni da bo poslal podatkovni paket dolžine 1024 bytov !**  
**Za prenos v XMODEM protokolu uporabiš TCP paket.**  
**Končna potrditev uspešnega prenosa je ACK na ETB !**

```

TCPSendXMODEM() /* Posiljanje datoteke preko XMODEMa */
{
    char SOH=1, STK=2, EOT=4, ACK=6, NAK=21, ETB=23;
    char MPFC=1, MPFSOH=2, MPFEOT=3, MPFETB=4, MPFACK=5;
    char string1[1100], state;
    char c, i, blok=1;
    char stevilo_blokov;
    char *kazalec=&c, napaka=0;
    int naslov=0, len, t_out, crc, rclen;

    state=MPFC;
    t_out=tm()+13000; /* Za timeout 60 sekund */
    while((t_out>tm()) && (napaka<10)) /* Timeout */
    {
        if(state==MPFC)
        {
            len=1;
            if (!tcpr(sck, kazalec, &len))
            {
                printf("%d\n", c);
                if (c=='C') state=MPFSOH;
            }
        }
        if(state==MPFSOH)
        {
            t_out=tm()+2200; /* Za timeout 10 sekund */
            len=3;
            string1[0]=SOH;
            string1[1]=blok;
            string1[2]=255-blok;
            mcpy(string1+3, (datamemo+naslov), BLOKLEN);
            /* CRC */
            crc=ccrc(16, 0x1021, 0, 0, string1+3, BLOKLEN);
            string1[BLOKLEN+3]=crc>>8;
            string1[BLOKLEN+4]=crc;
            len=BLOKLEN+5;
            tcps(sck, string1, &len);
            state=MPFACK;
            printf(" %d", blok);
        }
    }
}

```

```

if(state==MPFACK)
{
    len=1;
    if (!tcpr(sck, kazalec, &len))
    {
        if(c==ACK)
        {
            naslov=naslov+BLOKLEN;
            napaka=0;
            ++blok;
            if(naslov<filedolz) state=MPFSOH;
            else state=MPFEOT; /*Konec datoteke */
        }
        else
        {
            ++napaka;
            state=MPFSOH;
        }
    }
}
if(state==MPFEOT)
{
    len=1;
    c=EOT;
    tcps(sck, kazalec, &len);
    state=MPFETB;
}
if(state==MPFETB)
{
    len=1;
    if (!tcpr(sck, kazalec, &len))
    {
        printf("%d\n", c);
        if(c==ACK)
        {
            len=1;
            c=ETB;
            tcps(sck, kazalec, &len);
            return 1;
        }
    }
}
}
return 0;
}

```

```

TCPLoadXMODEM(char *datamemo)
{/*Sprejem datoteke datamemo preko 1K XMODEMa */
    char SOH=1, STK=2, EOT=4, ACK=6, NAK=21, ETB=23;
    char MPFC=1, MPFSOH=2, MPFEOT=3, MPFETB=4, MPFACK=5;
    char MPFBLOCK=6;
    char MPFDATA=7;
    char string1[1100];
    char state,blok,c,i;
    int len,t_out,t_c;
    char blockLen,lResult;
    char tempData[1100];
    int CRC,crc,naslov=0;
    char prvic=1,napaka=0;

    state=MPFSOH;
    t_out=tm()+13000; /* Za timeout 60 sekund */
    t_c=tm()+700; /* Timeout 4 sekunde za znak 'C' */
    mset(datamemo,0,8192); /* resetira vsebino datoteke */
    while(1)
    {
        while(1)
        {
            if(t_out<tm()) return 0; /* Timeout 60s */
            len=1;
            if(!tcpr(sck,&c,&len))
            {
                prvic=0;
                break;
            }
            else
            {
                if(prvic)
                {
                    if (t_c<tm())/* Timeout 3 sekunde za*/
                    {
                        /* znak 'C' */
                        t_c=tm()+700;
                        len=1;
                        c='C';
                        prtfl("C\n");
                        tcps(sck,&c,&len);
                    }
                }
            }
        }
    }
}

```

```

while(1)
{
    if(state==MPFSOH)
    {
        if(c==SOH)
        {
            state=MPFBLOCK;
            BLOKLEN=128;
            break;
        }
        if(c==STK) /* 1K XMODEM */
        {
            state=MPFBLOCK;
            BLOKLEN=1024;
            break;
        }
        if (c==EOT)
        {
            len=1;
            c=ACK;
            tcps(sck, &c, &len);
            return 1;
        }
        else
        {
            len=1;
            c=NAK;
            if(++napaka==10) return 0;
            tcps(sck, &c, &len);
        }
        break;
    }
    if(state==MPFBLOCK)
    {
        blockLen=0;
        state=MPFDATA;
        blok=c;
        printf("%d ", blok);
        break;
    }
}

```

```

if(state==MPFDATA) /*Sprejem podatkov. paketa*/
{
    while((blockLen<(BLOKLEN+2))&&(t_out>tm()))
    {
        len=(BLOKLEN+2)-blockLen;
        if(!tcpr(sck,string1,&len))
        {
            mcpy(tempData+blockLen,string1,len);
            blockLen=blockLen+len;
        }
    }

    CRC=tempData[BLOKLEN]<<8;
    CRC=CRC | tempData[BLOKLEN+1];
    crc=ccrc(16,0x1021,0,0,tempData,BLOKLEN);
    if((CRC==crc)&&((blok+c)==255)&&blok)
    {
        napaka=0;
        lResult=ACK;
    }
    else
    {
        if(++napaka==10) return 0;
        printf("CRC ERR\n");
        lResult=NAK;
    }

    len=1;
    c=lResult;
    tcps(sck,&c,&len);
    state=MPFSOH;
    if(!napaka)
    { /* Vpis v datoteko 'datamemo' */
        mcpy((datamemo+naslov),tempData,BLOKLEN);
        naslov=naslov+BLOKLEN;
        if(naslov>filedolz) return 0;
    }
    t_out=tm()+13000;
}
break;
}
}
return 0;
}

```

## 7. VSEBINA SIM KARTICE MODEMA

1. SMS CENTER	<b>+38641001333</b>	številka centra za SMS sporočila
2.		
3. <b>85.10.22.148</b>	<b>8090</b>	IP in PORT serverja
4. <b>mobitel</b>	123456789	GPRS USER
5. <b>internet</b>	12345678	GPRS PASS
6. <b>internetpro</b>	112345566	GPRS APN
7.		
8. PARKIRISCE	<b>01</b>	številka parkirišča
9.		
10. CILJ1	<b>+38641640229</b>	1. številka prejemnika SMS sporočila
11. CILJ2	0	2. številka prejemnika SMS sporočila
12. REZERVA	0	rezervna lokacija
13. DOMAČ OPERATER 1	<b>29341</b>	številka domačega operaterja

Zapis poudarjeno so podatki, ki jih je potrebno obvezno vpisati !

# 7. IZPIS NA COM IZHODU OB ZAGONU MODEMA

(BRATE=115200,8,1,N)

>SEKOM ONLINE PARKIR 1.8 <

število shranjenih dogodkov

=====

M=0

čitanje SIM kartice

=====

IP:PORT = 85.10.22.148 : 8090  
USER:PASS = mobitel : internet  
APN = internetpro  
PARK = P 01  
SMS CENTER=+38641001333  
CILJ1=+38641640229  
SMS CENTER=OK

CONNECT na IP:PORT

=====

CGDCONT-ENAD-PDP-OPEN SOC 0 – WAIT - CONNECT

Skeniranje terminalov

=====

SCAN  
SCAN  
SCAN

•  
•  
•