Elektrotehnički fakultet
Univerzitet u Sarajevu
Odsjek za automatiku i elektroniku
Zimski semestar, a.g. 2020/21.
Predmet: Praktikum automatike
Profesor: Red. prof. dr. Samim Konjicija

Realizacija parking rampe

Ime i prezime: Faris Hajdarpašić (18230)
Ime i prezime: Tarik Krivošija (18584)

Februar 2021. godine

Sažetak (Abstract)

Ovaj rad se bavi realizacijom parking rampe na bazi mikrokontrolera PIC16F1939. Za realizaciju ovog zadatka koristen je taster za otvaranje rampe, IR senzor za detekciju da li se automobil nalazi ispod rampe, motor za pokretanje rampe, te potenciometar za detekciju pozicije rampe. Motor se rotira u oba smijera pa se koristi H-Most za njegovo upravljanje. Kod je u potpunosti realiziran preko prekida, i to prekida za EEPROM i AD konverziju, Timer0 prekid, te prekidi na uzlaznu i silaznu ivicu na pinovima povezanim sa tasterom i IR senzorom. Okruzenja koja su koristena za izradu ovog seminarskog rada su MPLAB X IDE i Proteus.

This paperwork deals with realization of parking ramp based on microcontroller PIC16F1939. For the realization of this project we used a button for opening the ramp, IR sensor for detecting whether car is under the ramp, a motor for starting the ramp, and potentiometer for ramp-position detection. Motor can rotate in both directions, so we used H-Bridge to control it. The code is fully realized with interrupts, like interrupts for EEPROM and AD conversion, Timer0 interrupt, and interrupts on the ascending and descending edge on the pins connected to the button and IR sensor. The IDEs used for this paperwork are MPLAB X IDE and Proteus.

Sadržaj

1 Teoretski uvod

1.1 Opis problema

Naš zadatak je bio realizacija pojednostavljenje parking rampe. Potrebno je realizirati DC motor kojim ćemo pokretati rampu. Rampa će se podizati ukoliko je pritisnut taster, a spuštat će se nakon što auto prodje ispod nje što provjeravamo pomoću IC senzora. Takodjer pri ulasku na parking, vrijeme ulaska automobila je potrebno zabilježiti u memoriju kako bi se kasnije mogla izvršiti naplata.

Može se primijetiti da imamo 1 analogni ulaz a to je potenciometar pomoću kojeg očitavamo trenutni ugao rampe, 2 digitalna ulaza a to su taster i napon na izlazu IC senzora i 2 digitalna izlaza a to su zapravo pinovi koji su potrebni za okretanje DC motora u jednu i u drugu stranu čime dižemo odnosno spuštamo rampu.

Koristit ćemo A/D konverziju u svrhu očitanja napona sa potenciometra, prekid A/D konverzije, prekid na uzlaznu (pritisak tastera) i na silaznu ivicu (IC senzor), prekid modula TIMER0 (vrijeme ulaska i upis u eeprom).

1.2 Opis korištenih modula

1.2.1 A/D konverzija

A/D konverzija predstavlja proces pretvaranja analognog signala u digitalnu riječ, odnosno binarni broj koji ćemo moći pohraniti u registar. A/D konverzijom upravljamo pomoću 4 registra, a to su ADRESH, ADRESL, ADCON0, ADCON1. Kada smo konfigurisali A/D konverziju, ono što je bitno naglasiti je da smo odabrali desno poravnanje, odnosno setovali smo bit ADFM=1, što znači da ćemo koristiti i ADRESL i ADRESH odnosno 10-bitnu A/D konverziju. Pošto se kod pravilnog rada rampe ugao mijenja u opsegu (0-90) stepeni, dobiva se da je rezolucija mjerenja 0.089 stepeni, što je za upravljanje rampom više nego zadovoljavajuće.

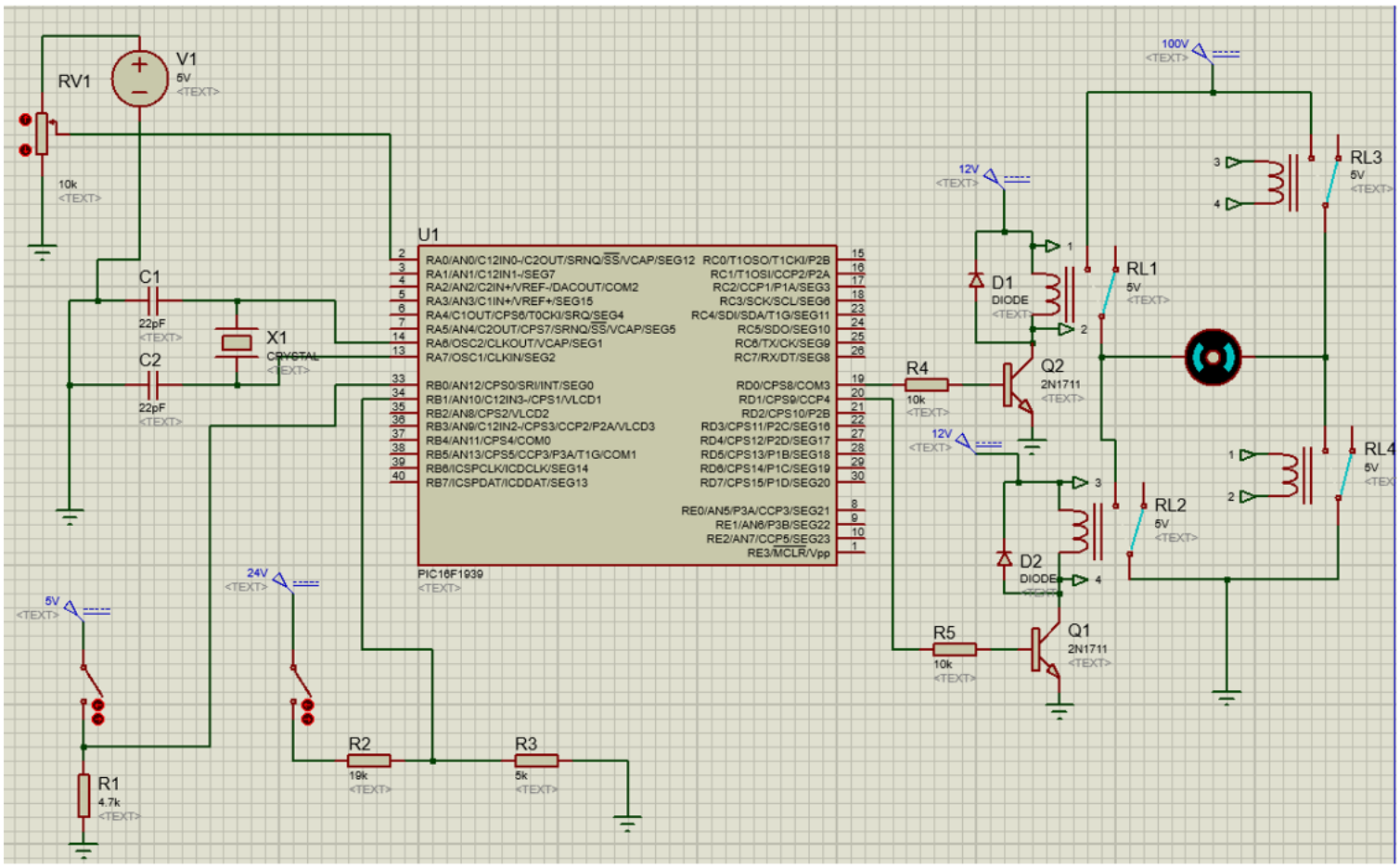
1.2.2 TIMER0

Prekid modula TIMER0 se dešava kad god brojač TMR0 predje sa vrijednosti 255 na 0. Minimalan period pozivanja ovog prekida je 1.2µs a maksimalan 0.026s bez korištenja nekog vanjskog brojača. Mi smo odlučili da koristimo prescaler 1:128, te da na početku postavimo TMR0 na vrijednost 130. Na taj način preko formule T = (4/8Mhz)*(255-130)*128 dobivamo T=8ms. Ovo je period pozivanja prekidne rutine za ovu vrstu prekida. Takodjer unutar ove prekidne rutine ćemo setovati bit ADGO na vrijednost 1 te svakih 8ms vršiti A/D konverziju. Za mjerenje vremena ulaska automobila na parking, biće nam potreban i vanjski brojač za dobivanje prije svega sekundi, a zatim minuta i sati. Biće pretpostavljeno da je brojanje počelo u ponoć, što znači da će se vrijeme računati od 00:00:00. Svakim pritiskom tastera vrijeme će biti zabilježeno u EEPROM memoriji.

1.2.3 EEPROM

U zadatku nije striktno traženo korištenje EEPROM memorije, medjutim mi smo zaključili da je ovdje potrebno koristiti ovaj modul iz razloga što automobili koji udju na parking moraju negdje ostati zabilježeni kao i njihovo vrijeme ulaska kako bi se kasnije mogla izvršiti naplata. EEPROM memorija ima jednu veliku prednost, a to je da se njen sadržaj neće brisati kada dodje do prekida napajanja, što je isto tako u ovom slučaju poželjno. Adrese lokacija u EEPROMU su od $0x00$ do $0xFF$. Upisivanje ćemo vršiti pomoću instrukcije $eeprom_write()$, a čitanje nećemo jer to izlazi iz opsega zadatka (čitanje bi se vršilo na izlazu automobila s parkinga).

1.3 Shema zadatka



Slika 1. Shema zadatka

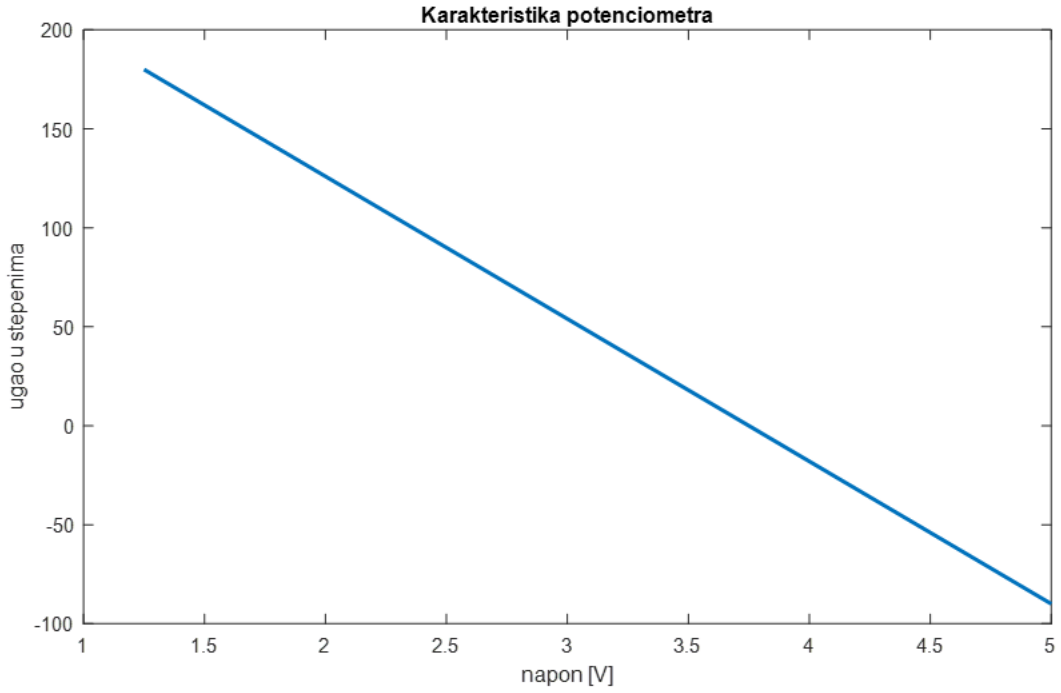
2 Simulacijski i eksperimentalni rezultati

2.1 Karakteristika potenciometra

U ovom poglavlju će biti detaljno opisana shema sa slike 1.

Za analogni ulaz je uzet napon koji sa potenciometra dovodimo na pin RA0. Potrebno je odrediti ugao kao funkciju dovedenog napona.

Karakteristika dobivena u **MATLAB**® je dat na slici 2.



Slika 2. Zavisnost ugla od napona na potenciometru

Analitički zapisano: $y = -72x + 270$.

Pretpostavljeno je da je referentni ugao kada je rampa u položaju koji odgovara 9h na satu. Pošto se ugao može mijenjati od 0 do 90 stepeni to znači da će se napon na potenciometru mijenjati izmedju 2.5V što odgovara rampi u potpuno podignutom položaju i 3.75V što odgovara referentnom uglu. Sa sheme vidimo da 2.5V dobivamo kada je na potenciometru 5k, a 3.75V dobivamo kada je na potenciometru 2.5k.

Pin RB0 koristimo kao digitalni ulaz koji će detektovati pritisak tastera.

Pin RB1 isto tako koristimo kao digitalni ulaz koji će dobivati vrijednost napona sa izlaza IC senzora. Bitno je naglasiti da nismo mogli direktno dovesti 24V na ulaz našeg mikrokontrolera već smo morali izvršiti naponsko razdvajanje jer digitalni ulaz ovog mikrokontrolera prima vrijednosti 0 i 5 volti.

2.2 Analiza koda

U ovom poglavlju ćemo pojasniti korišteni kod našeg projekta. Prikazat ćemo i rezultate simulacije.

Na slici 3, prikazana je inicijalizacija koristenih portova. Na PORTA smo samo spojili potenciometar na AN0/RA0 pin, pa smo citav port prograsili kao ulazni i analogni, zbog velike ulazne otpornosti koju pinovi imaju kada se proglase kao ulazni, i tako ih je sigurnije deklarisati, radi nezeljenih efekata. Na pinove RB0 i RB1 smo spojili taster i senzor, te smo citav port proglasili za digitalni i ulazni, jer senzor daje ili log1 ili log0, u zavisnosti od toga da li se vozilo nalazi ispod rample ili ne. Sto se tice PORTD, pomocu pinova RD0 i RD1 kontrolisemo smjer vrtnje motora, sto smo realizirali preko H-Mosta. Zato smo te pinove proglasili kao izlazne i digitalne.

```
/* Inicijalizacija portova */
void port_init(void) {
    ANSELA = 0xFF;
    TRISA = 0xFF;

    ANSELB = 0x00;
    TRISB = 0xFF;


    ANSELD = 0x00;
    TRISD = 0xFC;
    LATD = 0x00;
    Mgore = 0;
    Mdole = 0;
```

Slika 3. Inicijalizacija portova

Na slici 4, prikazana je inicijalizacija AD konverzije. Potrebni registri su ADCON0 i ADCON1. Za ADCON0 registar, podesili smo pin koji se koristi za AD konverziju a to je RA0 - tako da je bite CHS¡4:0¿ trebalo postaviti na vrijednost 0. Takodjer se treba ukljuciti modul za AD konverziju. U ADCON1 registru smo podesili da rezultat konverzije bude u desnom poravnanju, tako da najznacajnijih 8 bita se smjesta u ADRESH registar, a zadnja 2 bita u najzancajnija mjesta ADRESL registra. Za clock konverzije odabrali smo vanjski RC oscilator. Za negativni referentni naponski nivo uzeli smo V_{SS} a za pozitivni referentni nivo uzeli smo V_{DD}.

```
/* Inicijalizacija ADC-a */
void adc_init(void) {
    /* ADCON0 */
    ADCON0bits.CHS4 = 0;
    ADCON0bits.CHS3 = 0;
    ADCON0bits.CHS2 = 0;
    ADCON0bits.CHS1 = 0;
    ADCON0bits.CHS0 = 0;
    ADCON0bits.ADON = 1;

    /* ADCON1 */
    ADCON1bits.ADFM = 1;
    ADCON1bits.ADCS2 = 1;
    ADCON1bits.ADCS1 = 1;
    ADCON1bits.ADCS0 = 1;
    ADCON1bits.ADNREF = 0;
    ADCON1bits.ADPREF1 = 0;
    ADCON1bits.ADPREF0 = 0;
}
```

Slika 4. Inicijalizacija AD konverzije

Na slici 5, prikazana je inicijalizacija postavki za koristenje EEPROM memorije. Potrebno je bite $EEPGD$ i $CFGS$ podesiti na 0, te bit $WREN$ podesiti na 1, cime smo omogucli pisanje i brisanje u i iz EEPROM memorije.

```
/* Inicijalizacija EEPROM-a */
void eeprom_init(void) {
    EECON1bits.EEPGD = 0;
    EECON1bits.CFGS = 0;
    EECON1bits.WREN = 1;
}
```

Slika 5. Inicijalizacija EEPROM memorije

Na slici 6, prikazana je inicijalizacija postavki za koristenje Timer0 prekida. Omogucili smo prescaler, postavljanjem bita PSA na vrijednost 0, te smo njegovu vrijednost postavili na 128. Pocetnu vrijednost TMR0 smo postavili na 130.To znaci da ce se Timer0 pozivati svakih 8 ms.

```
/* Inicijalizacija TIMER0 prekida */
void tmr0_init(void) {
    OPTION_REGbits.TMR0CS = 0;
    OPTION_REGbits.TMR0SE = 0;
    /* Dodijelio sam prescaler */
    OPTION_REGbits.PSA = 0;
    /* prescaler = 128 */
    OPTION_REGbits.PS2 = 1;
    OPTION_REGbits.PS1 = 1;
    OPTION_REGbits.PS0 = 0;
    TMR0 = 130;
}
```

Slika 6. Inicijalizacija Timer0 prekida

Na slici 7, prikazana je inicijalizacija postavki za koristenje INT prekida. INT prekid je vezan za pin RB0, tako da je to prekid koji ce se desavati pritiskom tastera. Postavljanjem ovog bita na vrijednost 1, omogucavamo desavanje prekida na uzlaznu ivicu signala. Znaci taj se prekid desava kada se pritisne taster.

```
/* Inicijalizacija INT prekida - RB0 (taster) */
void int_init(void) {
    OPTION_REGbits.INTEDG = 1;
}
```

Slika 7. Inicijalizacija INT prekida

Na slici 8, prikazana je inicijalizacija postavki za koristenje prekida na promjenu na PORTB. Ovaj prekid je vezan za pinove PORTB-a, a mi smo ga iskoristili za pin RB1, tako da je to prekid koji ce se desavati nakon sto vozilo prodje rampu. Koristen je registar $IOCBN$ tako da dobijemo prekid na silaznu ivicu signala, jer nam tad taj prekid i treba, jer to silazna ivica signalizira da je automobil prosao rampu.

```
/* Inicijalizacija prekida za PORTB - RB1 (IR senzor) */
void portb_prekid_init(void) {
    IOCBNbits.IOCBN1 = 1;
}
```

Slika 8. Inicijalizacija prekida za promjenu signala na PORTB

Na slici 9, prikazana je inicijalizacija svih potrebnih prekida. Ovako je laksi pregled svih prekida koji se koriste u programu, zato smo napravili posebnu funkciju, da prekidi budu na jednom mjestu. Podesili smo 2 glavna prekida ($PEIE$ i GIE), te prekide za Timer0 prekid($TMR0IE$), INT prekid ($INTE$), prekid na silaznu ivicu signala na RB1 ($IOCIE$), prekid za AD konverziju ($ADIE$), te prekid za EEPROM ($EEIE$). Takodjer, resetovali smo njihove flagove na 0, iako je to podrazumijevana vrijednost, nije lose pisati, radi opce prakse. Svaki put kada nastupi prekid ovaj flag se automatski setuje na 1, pa ga je uvijek nakon desavanja prekida potrebno resetovati na 0, da ne bi doslo do ponovnog pozivanja prekida kada se on nije desio.

```
/* Inicijalizacija postavki za sve navedene prekide na jednom mjestu */
void prekidi_init(void) {
    INTCONbits.GIE = 1;
    INTCONbits.PEIE = 1;

    INTCONbits.TMR0IF = 0;
    INTCONbits.TMR0IE = 1;

    INTCONbits.INTF = 0;
    INTCONbits.INTE = 1;

    INTCONbits.IOCIF = 0;
    INTCONbits.IOCIE = 1;

    PIR1bits.ADIF = 0;
    PIE1bits.ADIE = 1;

    PIR2bits.EEIF = 0;
    PIE2bits.EEIE = 1;
}
```

Slika 9. Inicijalizacija svih prekida

Prije nego sto krenemo na objasnjavanje prekidne rutine, koja je glavna funkcija u nasem programu, prilozit cemo koristene pomocne varijable, radi lakse orjentacije prilikom objasnjavanja.

```
double ugao,napon;
char podizi = 0;
char spustaj = 0;
int sekunde=0,minute=0,sati=0;
int broj_automobila = 0;
int brojac=0;
int adresa=0;
```

Slika 10. Spisak pomocnih varijabli

Sada cemo objasniti svaki prekid do kojeg dodje ponaosob. Krenimo od prvog *if* uslova, a to je uslov za INT prekid. Ovaj prekid se desava zbog pritiska tastera, i to na pozitivnu ivicu. To znaci da je vozac vozila pritisnuo taster, te pokrecemo motor u jedom smjeru i podizemo rampu. Varijabli *podizi* dajemo vrijednost 1, da signaliziramo da se rampa podize, jer je u jednom trenutnu trebamo zaustaviti, tj. kad bude u uspravnoj poziciji - sto dobivamo ocitavanjem potenciometra. Varijablu *spustaj* postavljamo na 0. Takodjer u ovom prekidu, ocitavamo da je usao n-ti automobil te redni broj automobila i njegovo vrijeme (sate, minute i sekunde) zapisujemo u EEPROM memoriju.

```
void interrupt isr(void){
    /* Desio se prekid zbog pritiska tastera */
    if(INTCONbits.INTE && INTCONbits.INTF && ugao<=0) {
        INTCONbits.INTF = 0;
        eeprom_write(adresa++,broj_automobila++);
        eeprom_write(adresa++,sati);
        eeprom_write(adresa++,minute);
        eeprom_write(adresa++,sekunde);
        /* Motor se podize */
        Mgore = 1;
        Mdole = 0;

        podizi = 1;
        spustaj = 0;

    }
```

Slika 11. Prekid uzrokovan pritiskom tastera na RB0 pinu

Sljedeci *if* uslov je za prekid na silaznu ivicu pina RB1. Ovaj prekid se desava kada automobil prodje ispod rampe - zato se koristi prekid na silaznu ivicu. Posto je automobil prosao ispod rampe, pokrecemo motor u obrnutom smjeru i tada se spusa rampa. Varijabli *spustaj* dajemo vrijednost 1, da signaliziramo da se rampa spusta, jer je u jednom trenutnu trebamo zaustaviti, tj. kad bude u uspravnoj poziciji - sto dobivamo ocitavanjem potenciometra. Varijablu *podizi* postavljamo na 0.

```
    /* Prekid na donju ivicu - IR Senzor */
    if(INTCONbits.IOCIE && IOCBFbits.IOCBF1 && ugao>=90) {
        INTCONbits.IOCIF = 0;
        IOCBFbits.IOCBF1 = 0;

        /* Motor se spusta */
        Mgore = 0;
        Mdole = 1;

        podizi = 0;
        spustaj = 1;
    }
```

Slika 12. Prekid uzrokovan silaznom ivicom signala na pinu RB1

Sljedeci *if* uslov je uzrokovan Timer0 prekidom. Ovaj prekid se desava svakih 8 ms, jer smo tako podesili vrijednost $TMR0$ i vrijednost *prescaler*-a. Ovaj prekid koristimo jer zelimo da svako malo ocitavamo vrijednost potenciometra, sto ustvari predstavlja poziciju rampe - pa posto je on povezan na $AN0$ pin, to znaci da svako malo trebamo pokretati AD konverziju postavljanjem GO bita na 1. Ovaj prekid se desava kada TMR0 promijeni vrijednost sa 255 na 0, tj. prilikom overflow-a (a sto ovisi od prescalera, tj. koliko se uzlaznih ivica racuna kao jedna - u nasem slucaju 128). U ovom *if* uslovu takodjer brojimo vrijeme, koje cemo zapisati u EEPROM memoriju.

```
/* Pozivanje prekida TMR0 za postavljanje AD konverzije*/
if(INTCONbits.TMR0IE && INTCONbits.TMR0IF) {
    INTCONbits.TMR0IF = 0;
    TMR0 = 130;


    if(brojac==125){
        sekunde++;
        brojac=0;
        if(sekunde==60){
            minute++;
            sekunde=0;
            if(minute==60){
                sati++;
                minute=0;
                if(sati==24){
                    sati=0;
                }
            }
        }
    }
    else brojac ++;
    ADCON0bits.GO = 1; // poziva se ADC prekid kad se zavrsi konverzija

}
```

Slika 13. Prekid koji se desi nakon overflow-a vrijednosti Timer0

I na kraju zadnji *if* uslov desava se zbog zavrsene AD konverzije. Kada smo postavili *GO* bit na vrijednost 1, nakon zavrsavanja AD konverzije, poziva se prekid AD konverzije. U ovom *if*-uslovu provjeravamo da li se rampa podize i spusta i dokle se podigla ili spustila. Kada se rampa dize i dosegne uspravan polozaj - tada zaustavljamo motor. Isto tako radimo i kada se rampa spusta, te se spusti skroz - tada takodjer zaustavljamo motor.

```
/* ADC prekid */
if(PIE1bits.ADIE && PIR1bits.ADIF) {
    PIR1bits.ADIF = 0;
    napon=5/1023.0*(ADRES);
    ugao=-72*napon+270;

    /* Rampa spustena */
    if(spustaj==1 && ugao<=0) {
        /* Iskljucio se motor - Da li ce pasti rampa??? */
        Mgore = 0;
        Mdole = 0;
        spustaj = 0;

    }

    /* Rampa podignuta */
    if(podizi==1 && ugao>=90) {
        /* Iskljucio se motor - Da li ce pasti rampa??? */
        Mgore = 0;
        Mdole = 0;
        podizi = 0;

    }
}
```

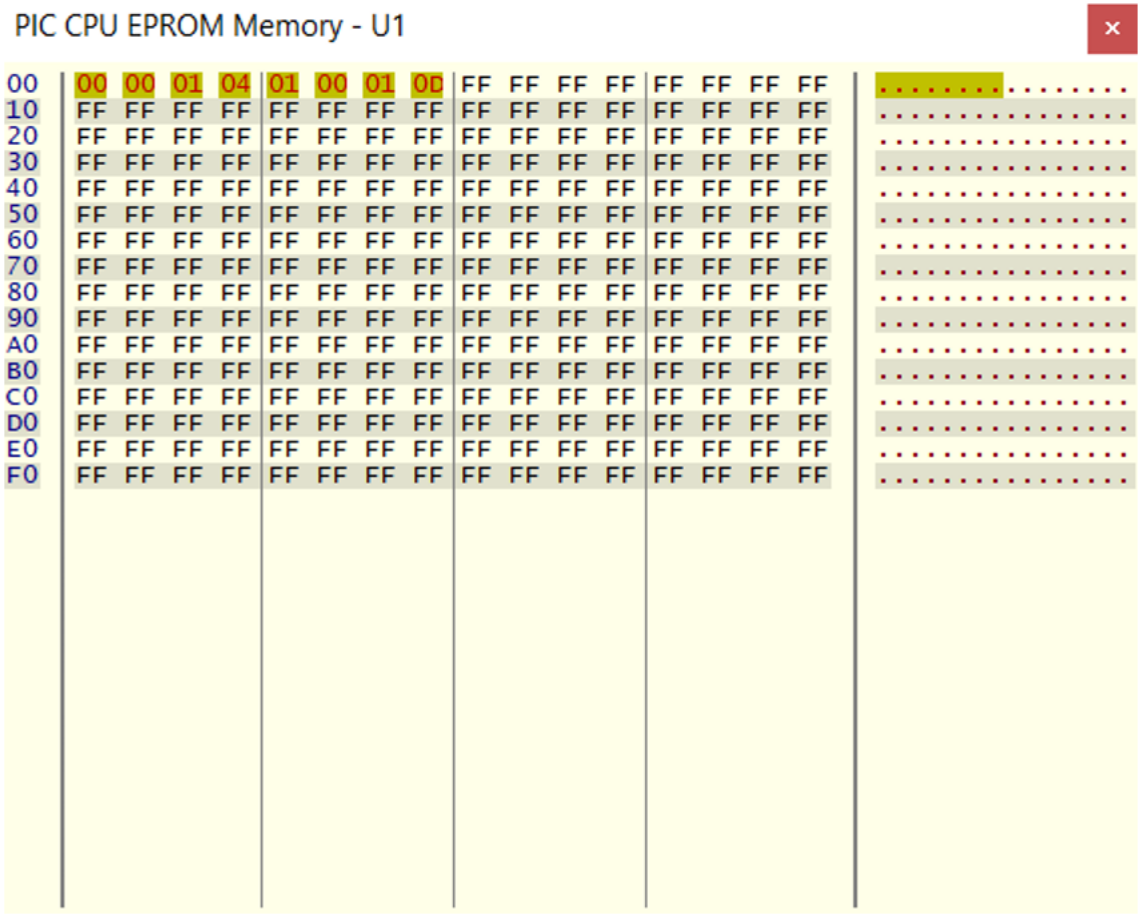
Slika 14. Prekid koji se desi nakon zavrsetka AD konverzije

2.3 Rezultat simulacije

Pinove RD0 i RD1 koristimo za upravljanje DC motorom. Za realizaciju je korišten H most. Kada je na pinu RD0 logička jedinica rampa će se podizati, a kada je na pinu RD1 logička jedinica rampa se spušta.

Ostalo je još da prikažemo upis u eeprom memoriju. Upis se vrši pritiskom tastera. Prvi bajt je redni broj automobila, drugi bajt su sati, treći bajt su minute i četvrti sekunde (dalje se ponavlja). Dakle za pohranu jednog automobila potrebno je 4 bajta. Ograničenje koje je prisutno za ovu realizaciju je to da se pretpostavlja da je program pokrenut tačno u ponoć odnosno TMR0 je na početku izvršavanja jednak nuli i vrijeme se računa od 0. Moglo bi se odabrati i neko drugo početno vrijeme, ili da recimo pitamo korisnika da unese tačno vrijeme prije pokretanja kako bi program radio korektno, odnosno kako bi vozač dobio karticu sa tačno upisanim vremenom.

Na sljedećoj slici vidimo upis u EEPROM prva 2 automobila :



Slika 15. EEPROM memorija

3 Zakljucak

Cilj ovog projekta je bila realizacija parking rampe. Realizacija je bila uradjena kroz simulacione programe *MATLAB®*, *Proteus* i koristenjem *MPLAB X IDE* za pisanje koda.

Na pocetku smo spomenuli kako se koriste EEPROM, Timer0, AD konverzija, te je ovo sve realizirano u kodu, koristeci dodatno jos neke prekide. Citav kod se oslanja na koristenje prekida, i kao takav laksa je realizacija u odnosu na proceduralni stil programiranja. Data je i shema spajanja.

U drugom dijelu smo dali karakteristiku potenciometra koji je vezan za motor i koji nam daje informaciju o poziciji rampe, te je objasnjen kod i dat rezultat simulacije - tj. upis u EEPROM memoriju.
Ukratno receno, u ovom projektu smo realizirali mogucnost pritiska tastera za ulazak u parking - shodno tome se podize rampa, te vozilo prolazi rampu, te nakon sto je prodje, rampa se spusta. Zabiljezen je redni broj automobila te vrijeme njegovog ulaska. Naravno ovaj projekat je lako nadogradiv, te su mogucnosti nadogradnje velike. Tu se misli na prosirivanje mogucnosti koje parking ima, kao npr: realizacija druge rampe za izlazak, racunanje ukupno provedenog vremena na parkingu, signalizacija popunjenosti parkinga itd.

References

[1] Prof.dr. Samim Konjicija, Predavanja iz predmeta Praktikum automatike, Elektrotehnički fakultet, Univerzitet u Sarajevu

[2] Microchip dokumentacija za PIC16F1939