SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

SVEUČILIŠNI PREDDIPLOMSKI STUDIJ ELEKTROTEHNIKE

Edi Ivančić

PROGRAMSKA PODRŠKA ZA POSTAVLJANJE NAČINA RADA MIKROKONTROLERA

ZAVRŠNI RAD

Rijeka, 2011.

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
SVEUČILIŠNI PREDDIPLOMSKI STUDIJ ELEKTROTEHNIKE

Edi Ivančić

PROGRAMSKA PODRŠKA ZA POSTAVLJANJE NAČINA RADA MIKROKONTROLERA

ZAVRŠNI RAD

Student: Edi Ivančić
JMBAG: 0069042507
Mentor: izv. prof.dr. sc. Viktor Sučić
Kolegij: Digitalna logika
Smjer: Automatika

Rijeka, 2011.

(this page is intentionally left blank)

(ubaci original zadatka završnog rada)

Predgovor 

Zahvaljujem se izv. prof. dr. sc. Viktoru Sučiću što je prihvatio moju zamolbu za izradu završnog rada iz kolegija „Digitalna logika“ iako prethodno nisam radio izborni projekt iz istog kolegija. Zahvaljujem se i kolegi Paolu Zenzeroviću na velikoj pomoći i savjetima pri izradi završnog rada i dostupnosti za konzultacije u vezi rada. Zahvalio bih se i kolegi Stjepanu Šijanu koji je prethodno radio izborni projekt iz „Digitalne logike“, a koji me pobliže upoznao s tematikom, te nesebično podijelio materijale za učenje. Na kraju se zahvaljujem, no ne i najmanje, svojim roditeljima na velikoj potpori i razumijevanju koje mi redovito pružaju, pa tako i pri pisanju završnog rada.

Napomenuo bih samo da mi je žao što nisam izabrao i izborni projekt iz kolegija „Digitalna logika“, budući da sam se u međuvremenu poprilično zainteresirao za tu temu, a i vjerujem da bi završni rad bio napravljen nešto temeljitije i koji mjesec prije roka.

SADRŽAJ

1. Uvod

Programska podrška za postavljanje načina rada mikrokontrolera krajnjem korisniku znatno olakšava generiranje vrijednosti kontrolnih registara mikrokontolera, Izrađena programska podrška za postavljanje načina rada mikrokontrolera temelji se na grafičkom sučelju u kojem korisnik odabire način rada mikrokontrolera. Glavna prednost ove programske podrške je ta što je korisniku dovoljno minimalno znanje o mikrokontrolerima kako bi uspješno generirao kod s potrebnim vrijednostima kontrolnih registara nužnima za izvršenje željene operacije. Na ovaj način je dovoljno unijeti ili odabrati parametre koji su krajnjem korisniku potrebni u radu mikrokontrolera, a programska će podrška sama generirati vrijednosti koje će mikrokontroler „razumjeti“ u vidu koda napisanog za programski jezik mikroPaskal. Budući da je programskim jezikom Visual Basic moguće jednostavno izraditi aplikaciju s grafičkim sučeljem, programska podrška za postavljanje načina mikrokontrolera izrađena je u njemu.

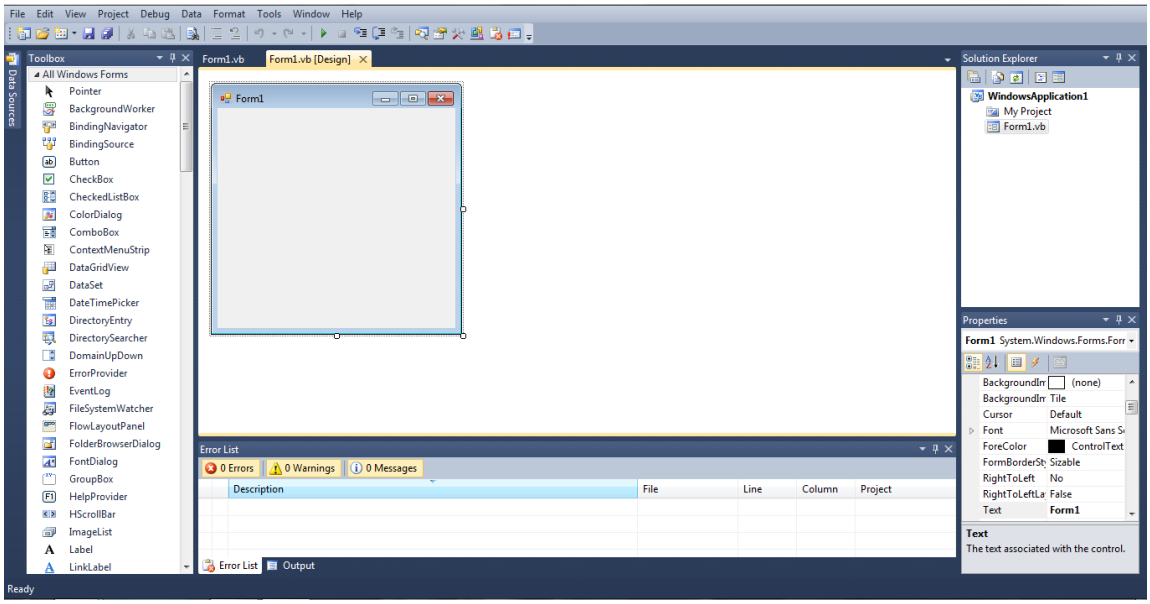
Uz općeniti opis strukture mikrokontrolera, u ovom završnom radu posebno će se opisati oni elementi mikrokontrolera čije se vrijednosti registara mogu generirati u stvorenoj programskoj podršci za postavljanje rada mikrokontrolera. Također će se opisati način rada programske podrške za postavljanje načina rada mikrokontrolera, kako bi je u budućnosti bilo moguće slobodno i s razumijevanjem mijenjati.

2. Opis korištene programske opreme – Visual Basic

2.1. Uvod

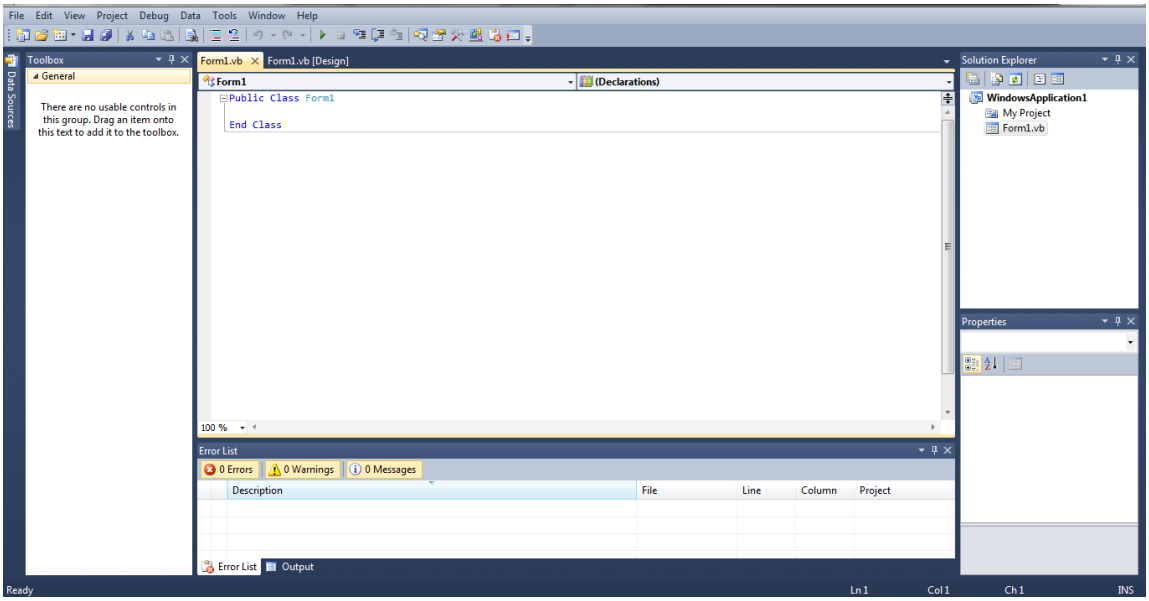
Visual Basic je programski jezik kojeg je razvila tvrtka Microsoft-a, a temelji se na programskom jeziku BASIC. Visual Basic, poznat i kao VB, služi za programiranje radnji (eng. *event drive*) i za stvaranje radnog sučelja (eng. *development environment*). VB pruža jednostavnost BASIC-a zajedno s potpunim pristupom Windows API-ju *(Application Programming Interface*). API je skup određenih pravila i specifikacija koje programeri slijede tako da se mogu služiti uslugama ili resursima operacijskog sustava (u ovom slučaju Windows C[illegible]a) ili nekog drugog složenog programa kao standardne biblioteke rutina. VB omogućava brzu izradu aplikacija (RAD – *Rapid Application Development*) s grafičkim korisničkim sučeljem (GUI – *Graphical User Interface*). VB-om možemo stvarati .exe datoteke, *ActiveX* kontrole ili DLL datoteke, ali je najčešće korišten za stvaranje Windows aplikacija.

Poput BASIC-a, VB je napravljen na način da bude jednostavan za učenje i korištenje početnicima[illegible]o uz stvaranje jednostavnih aplikacija sa grafičkim sučeljem, VB korisnicima omogućava i stvaranje kompleksnih aplikacija. Programiranje u VB-u se sastoji od vizualnog slaganja komponenti ili kontroli na formu (grafičko sučelje programa), specificiranja značajki te komponente ili kontrole te pisanja dodatnog koda koji omogućava povećanje same funkcionalnosti.



Slika 2.1 Stvaranje grafičkog sučelja

Sa slike 1.1 je vidljiva jednostavnost stvaranja GUI-a programa. Povlačenjem (eng *drag-and-drop*) kontrola (*checkbox-eva*, *label-a*, *mbobox-eva* itd.) iz *toolbox-a* na formu (obrazac, „prozor“) stvaramo sučelja. Kontrole imaju vlastite atribute i *event handler-e* (rukovatelje događajima) koji poprimaju početne vrijednosti prilikom stvaranja kontrole, ali ih je moguće i mijenjati. Neki atributi se mogu mijenjati i prilikom rada aplikacije temeljem korisnikovih naredbi što omogućuje stvaranje dinamične aplikacije.



Slika 2.2 Unos koda

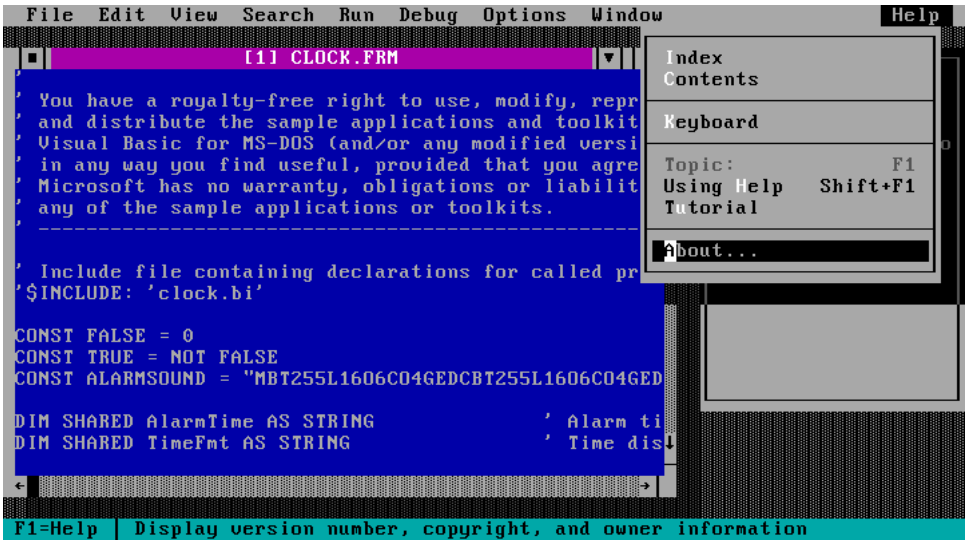
Na slici 2.2 prikazano je sučelje za unos koda. Svakoj komponenti moguće je pridodati događaj koji će se izvesti određenom akcijom. Tako je moguće pritiskom na gumb (eng. *button*) povećati veličinu forme, prikazati tekst, komunicirati preko serijskog porta itd..

2.2. Razvoj

Spajanje mogućnost *drag-and-drop-a* za stvaranje korisničkog sučelja, izvedena iz generatora formi stvorenog od strane Alana Coopera i tvrtke Tripod, i programskog jezika BASIC omogućilo je stvaranje Visual Basica 1991.

Razvoj Visual Basica:

- Visual Basic 1.0 (svibanj 1991)
- Visual Basic 1.0 za MS-DOS (rujan 1992)
- Visual Basic 2.0 (studeni 1992)
- Visual Basic 3.0 (ljeto 1993)
- Visual Basic 4.0 (kolovoz 1995)
- Visual Basic 5.0 (veljača 1997)
- Visual Basic 6.0 (ljeto 1998)
- Visual Basic .NET (2002)
- Visual Basic 2005 (2004)
- Visual Basic 2008
- Visual Basic 2010



Slika 2.3 Sučelje Visual Basic-a za MS-DOS [S1]

Za izradu programske podrške za postavljanje načina rada mikrokontrolera korišten je besplatan Microsoft Visual Basic Express.

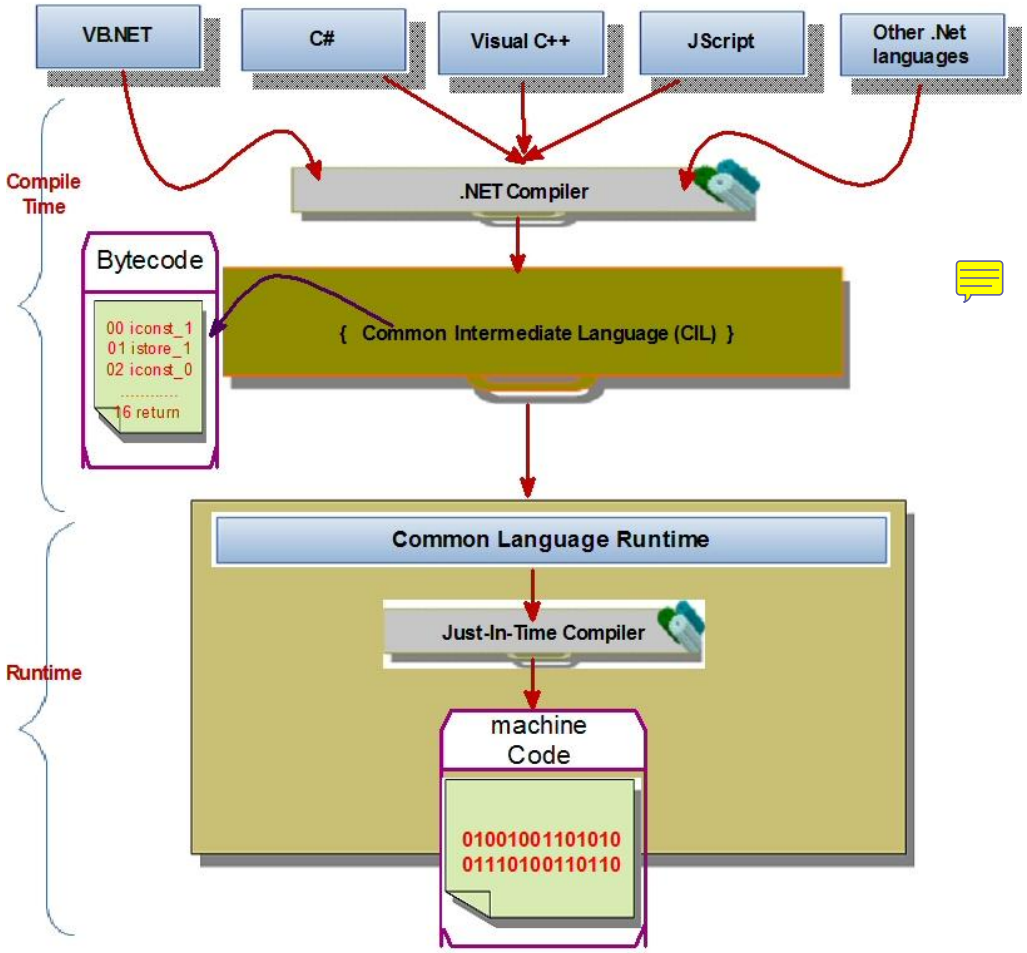
2.3. Visual Basic .NET

Visual Basic .NET, danas poznat i kao Visual Basic, je programski jezik koji se temelji na .NET Framework-u. Microsoft .NET je okružje za razvoj aplikacija koje sadrži velik broj knjižnica za nadogradnju OS i podržava nekoliko programskih jezika čime omogućuje korisnicima lakšu izradu aplikacija. Naime, osim dijela koji programer sam stvara, VB sadrži i dio koji sam vrši programiranje za sve ostale što treba programu. Taj se dio naziva *runtime*, a Visual Basic 6 koristio je *runtime* MSVBVM60. Tu istu funkciju u VB.NET-u obavlja dio .NET Frameworka koji se naziva CLR (*Common Language Runtime*).

CLR je softverski sustav u kojem se kod izvršava. Kada korisnik pokrene aplikaciju pisanu za .NET platformu, CLR ju izvršava kako bi joj osigurao stabilnost i funkcionalnost. Instrukcije u programu se u realnom vremenu prevode u izvorni strojni kod koji razumije računalo. Za taj je posao zaslužan JIT-kompajler (*Just In Time*). Upravo prevođenje u izvorni strojni kod računala, omogućilo je .NET-u prelazak na druge operativne sustave kao što su Linux ili MacOS (putem pomoćnog, *third-party* MONO sustava). Kako kompajliranje zasigurno usporava izvršavanje aplikacija, ono će se izvršavati samo jednom, a njegov će se rezultat spremiti kako bi se kasnije mogao koristiti bez ponovnog kompajliranja.

Aplikacije za .NET platformu mogu se pisati u raznim programskim jezicima, gotovo svim poznatijim. CLR, međutim, ne poznaje niti jedan taj jezik - on dobiva naredbe isključivo u jeziku nazvanom *Microsoft Intermediate Language* (MSIL), temeljen na pravilima koja se nazivaju *Common Language Specifications* (CLS). Stoga je jasno da mora postojati kompajler koji će programski jezik u kojem programer piše kod prevesti u MSIL kako bi ga CLR razumio. Ovi kompajleri nazivaju se IL-kompajleri te su dostupni za velik broj programskih jezika. Microsoft je izdao kompajlere za pet jezika: C#, J#, C++, Visual Basic i JScript.

Kako se svi ovi jezici prvo pretvaraju u MSIL, sasvim je svejedno u kojem će se od njih pisati aplikacije. Iz ovoga također proizlazi i druga velika mogućnost .NET-a - višejezično pisanje aplikacija. Tako sada više nije nužno da svi programeri koji rade na određenom projektu poznaju isti programski jezik, važno je samo da je podrška za njihov jezik dostupna u .NET-u, odnosno da postoji IL-kompajler za njihov jezik.

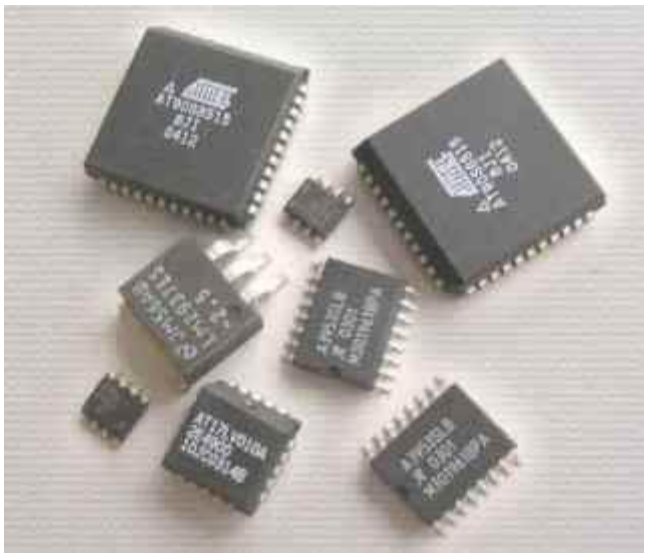


Slika 2.4 Način rada CLR-a [S2]

Iako se klasični VB i VB.NET u sintaksi gotovo i ne razlikuju, VB.NET više ne koristi mnoge knjižnice korištene u starom VB-u, već koristi nove koji su složenije za korištenje, ali daleko moćnije od onih korištenih kod klasičnog Visual Basica.

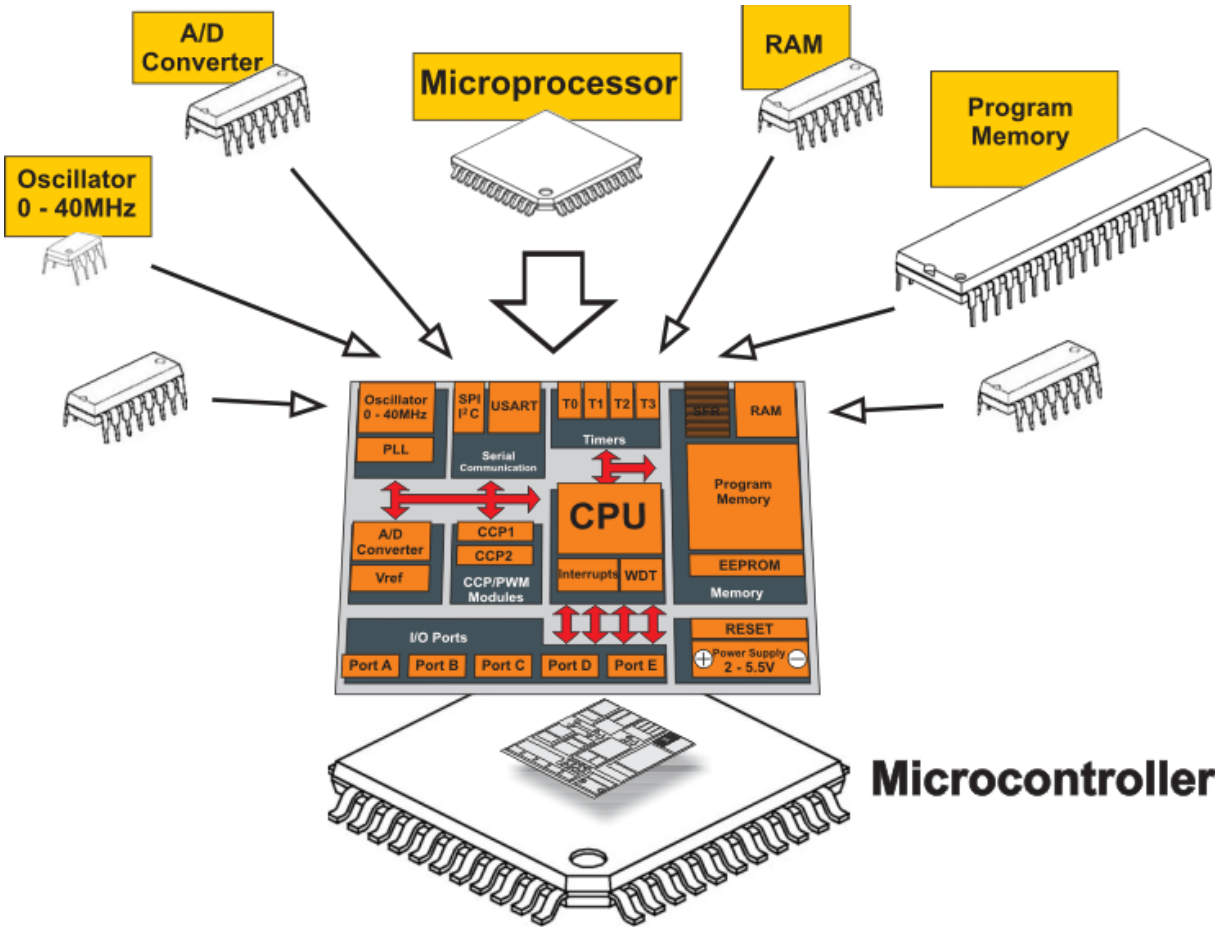
3. Mikrokontroler

Mikrokontroler je elektronički uređaj koji obuhvaća komponente mikroprocesorskog sustava (mikroprocesor, ROM, RAM) na jednom čipu. Ukratko, mikrokontroler je malo računalo, a složenost mu ovisi o složenosti zadaće koju izvršava.



Slika 3.1 Nekoliko vrsta mikrokontrolera [S3]

Mikroprocesor je, s druge strane, optimiziran da upravlja tokom podataka između odvojene memorije i vanjskih (perifernih) uređaja smještenih izvan mikroprocesora. Veze s mikroprocesorom sadrže adrese, kontrole i sabirnice što omogućuje izbor jedne vanjske jedinice te slanje ili primanje podataka s nje. Budući da se kod mikrokontrolera procesor i vanjske jedinice nalaze na istoj silicijskoj pločici, oni su samostalni i rijetko imaju sabirnice koje se produžuju izvan njihovog kućišta.



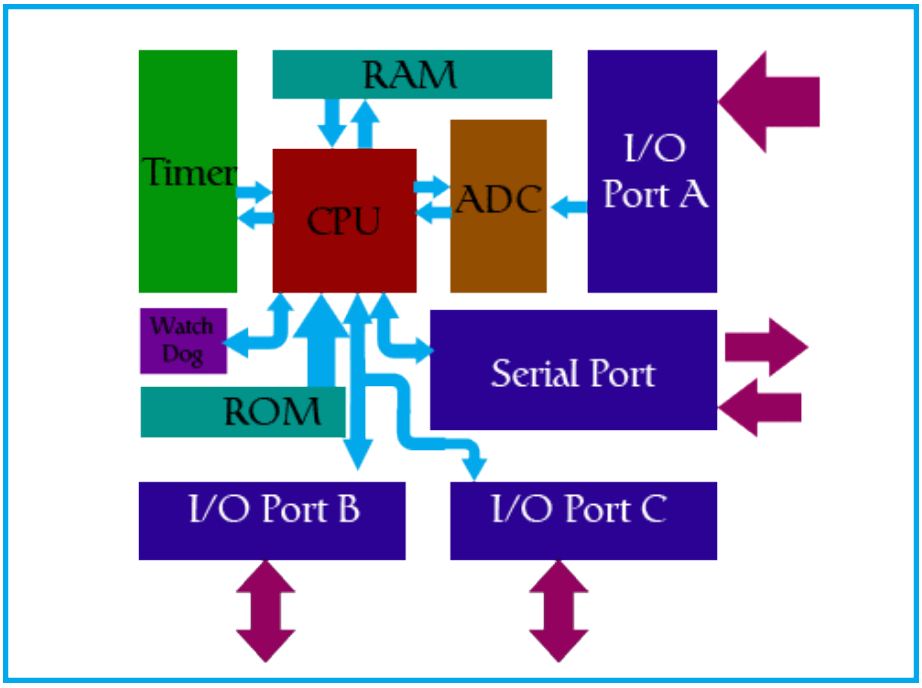
Slika 3.2 Osnove komponente mikrokontrolera [S4]

Neka od osnovnih svojstva mikrokontrolera su:

- Relativno mali radni takt reda 10 MHz
- Mali broj jednostavnih instrukcija, red veličine oko 100
- Radna memorija (RAM) reda KB
- Stalna memorija s programskim kodom u PROM ili EPROM izvedbi
- Brojači različitih namjena kao sat, brojač impulsa, BCD brojač i drugi
- Brojač za nadzor ispravnog rada - WDT (Watch Dog Timer)
- Ulazno/Izlazni kanali (*portovi*) za prihvat i slanje podataka
- A/D i D/A pretvornici razlučivosti prema namjeni, uobičajeno 8 bit-ni
- Širok raspon napona napajanja

Svi navedeni elementi ne moraju se nužno nalaziti u mikrokontroleru. Od namjene mikrokontrolera ovisit će njegov izbor, pa će neki imati više U/I portova, drugi A/D pretvornika i slično.

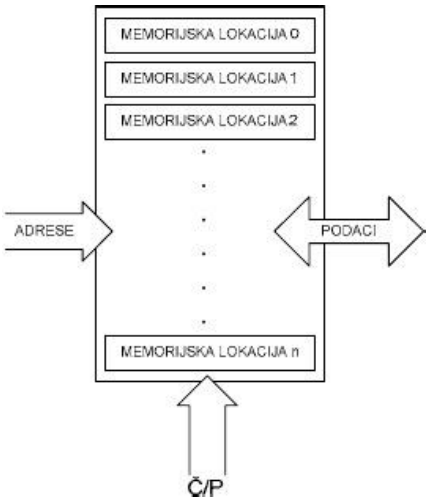
3.1. Osnovna arhitektura mikrokontrolera



Slika 3.3 Osnovna arhitektura mikrokontrolera [S5]

3.1.1. Memorijska jedinica

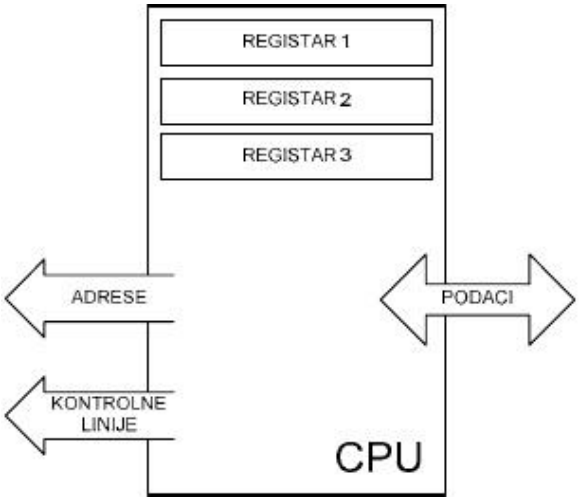
Memorija je dio mikrokontrolera čiji je zadatak pamćenje podataka. Za određeni ulaz dobivamo sadržaje određene adresirane memorijske lokacije. Postoje dva koncepta: adresiranje i memorijska lokacija. Memorija se sastoji od svih memorijskih lokacija, a adresiranje nije ništa drugo nego biranje jedne od njih. To znači da moramo izabrati željenu memorijsku lokaciju, a zatim čekati sadržaj te lokacije. Osim čitanja memorija mora podržavati i pisanje u nju. To je omogućeno dodavanjem dodatne linije koja se naziva kontrolna linija. Označavamo je s R/W (Č/P, *read/write*), a koristi se na sljedeći način: ako je R/W = 1 vrši se čitanje, a u suprotnom vrši se pisanje.



Slika 3.4 Model memorijske lokacije [S6]

3.1.2. CPU (Central Processing Unit)

Mikroprocesor (CPU) je najvažniji elektronički sklop računala, koji preuzima programske naredbe i na osnovu njih obrađuje podatke. Naziv "mikro'' koristi se zbog malih dimenzija. Sadrži do nekoliko milijuna tranzistora ukomponiranih na silicijskoj pločici upakiranoj u plastično kućište. Vrijeme izvođenja instrukcija je reda mikrosekunde, a broj instrukcija i veličina binarnog podatka osnovni su parametri koji definiraju kvalitetu mikroprocesora. Mikroprocesori koji se u računalnim sustavima opće namjene koriste dizajnirani su za rad s podacima duljine 4, 8, 16, 32 i 64 bit-a.



Slika 3.5 Primjer centralne procesorske jedinice sa tri registra [S7]

Registri su memorijske lokacije čija je uloga da pomognu pri obavljanju raznih matematičkih operacija ili bilo kojih drugih operacija sa podacima gdje god da se oni nalazili. Imamo dvije nezavisne cjeline (memoriju i CPU) koje nisu međusobno

povezane, čime je spriječena bilo kakva razmjena podataka. Ako na primjer želimo zbrojiti sadržaj dvije lokacije iz memorije i njihov rezultat ponovo vratiti u memoriju potrebna nam je veza između memorije i CPU, to jest moramo imati neki ″put″ preko kojeg podaci idu iz jednog bloka u drugi, a ti ″putevi″ se zovu sabirnice.

3.1.3. Sabirnica

Prijenos podataka između sklopova unutar mikroprocesora i unutar računala vrlo je značajan dio posla, a obavlja se preko sabirnica (eng. *bus*), višežilnim prijenosnim putevima između pojedinih sklopova. Broj vodova sabirnice ovisi o broju bitova koji opisuju podatke koje je potrebno prenijeti. U suštini prijenos podataka je paralelan.

Postoje dva osnovna sustava sabirnica:

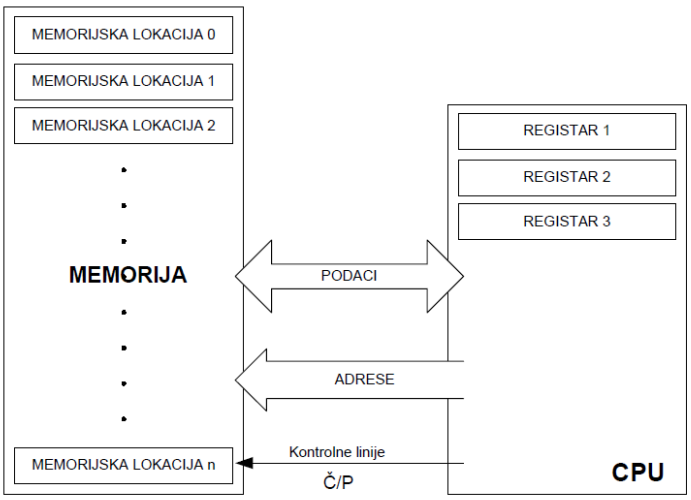
- Unutrašnje sabirnice - veze unutar mikroprocesora.
- Vanjske sabirnice - veze u računalu izvan mikroprocesora.

Međusobno su odvojene međusklopovima jer se često promet na njima ne odvija istom brzinom. Razmjena podataka u mikroprocesoru obavlja se višestruko brže nego između sklopova računala, te ih je stoga potrebno odvojiti. O prijenosu podataka između unutrašnje i vanjske sabirnice brinu se posebni upravljači (eng. *controller*).

Unutarnja i vanjska komunikacija odvija se preko tri odvojene sabirnice:

- Podatkovne sabirnice – dvosmjerna
- Adresne sabirnice – jednosmjerna
- Upravljačke sabirnice – jednosmjerna

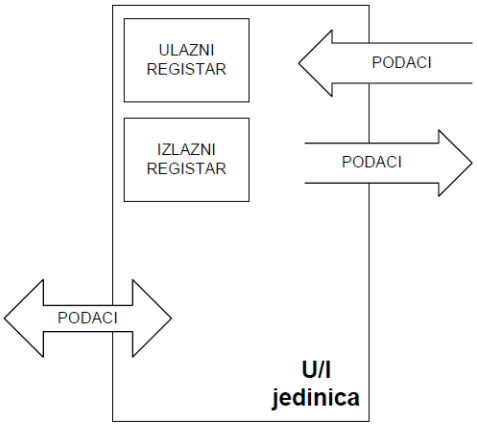
Naravno, komunikacija posredstvom triju sabirnica ubrzava sustav. Podatku, koji je na sabirnici podataka, preko adresne sabirnice određuje se mjesto na koje će se uputiti, a upravljačka sabirnica će prijenos odobriti.



Slika 3.6 Primjer komunikacije memorije i centralne procesorske jedinice pomoću sabirnica [S8]

3.1.4. U/I jedinica

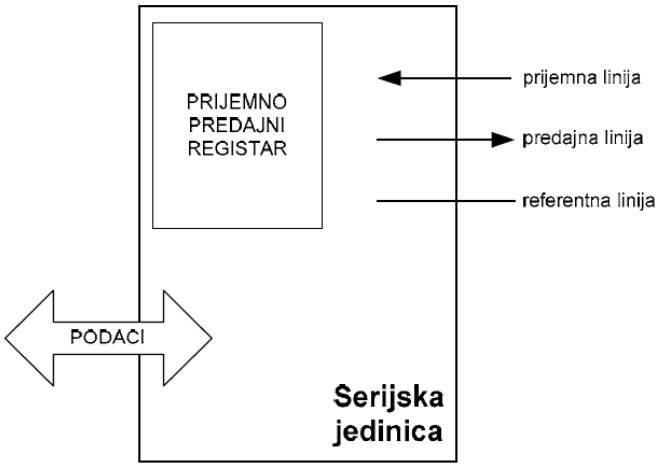
Ove memorijske lokacije se nazivaju portovi. Portovi mogu biti ulazni, izlazni ili dvosmjerni. Kod rada s portovima potrebno je prvo odabrati port s kojim se radi. Kada s njima radimo, portovi se ponašaju kao memorijske lokacije. Moguće je jednostavno pisati ili čitati s njih, te to može biti vidljivo s pinova mikrokontrolera. Također je moguće softverski odrediti koji će se portovi koristiti kao izlazi, a koji kao ulazi. Budući da je svaki U/I port kontroliran SFR-om (*Special Function Register*), unosom logičke jedinice u jedan bit SFR-a, odgovarajući pin je konfiguriran kao ulaz. Dovođenjem logičke nule pin postaje izlaz.



Slika 3.7 Primjer U/I jedinice [S9]

3.1.5. Serijska komunikacija

Serijskom komunikacijom mikrokontroleru omogućujemo povezivanje na daljinu. Paralelni prijenos nije pogodan za velike udaljenosti zbog većeg broja vodiča. Da bi prijenos funkcionirao treba odrediti skup pravila po kojima će se odvijati. Skup pravila po kojem se vrši prijenos podataka nazivamo protokol. Serijski prijenos podataka dijelimo na sinkroni i asinkroni. Sinkroni prijenos između dva uređaja radi pod zajedničkim taktom. Asinkroni prijenos koristimo kad ne znamo frekvencijski takt drugog uređaja, ovu vrstu prijenosa koristimo za slanje manjih podataka.



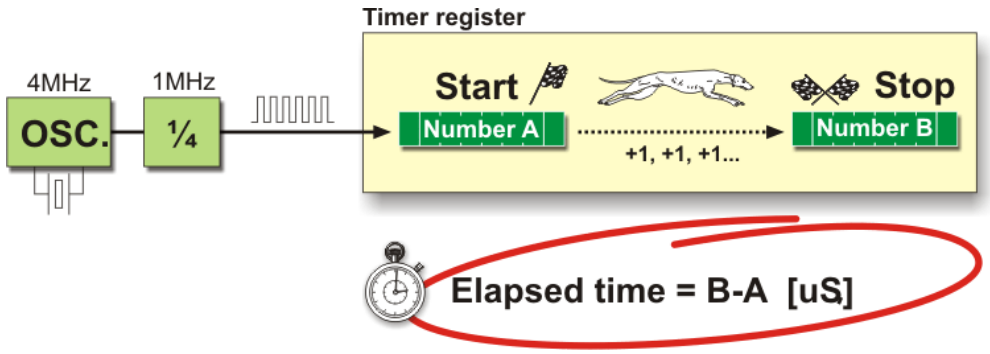
Slika 3.8 Serijska jedinica [S10]

3.1.6. Tajmer/Brojač

Oscilator mikrokontrolera koristi kristal kvarca za rad posebice zbog toga što je njegova frekvencija jasno definirana i vrlo stabilna, a generirani pulsevi su uvijek iste širine što ih čini idealnima za mjerenje vremena. Ukoliko je potrebno mjeriti vrijeme između dva događaja, dovoljno je brojiti pulseve koji dolaze iz tog oscilatora, a to je upravo ono što tajmer radi. Mnogi programi koriste te elektroničke „štoperice“, a to su najčešće 8 ili 16 bitni SFR-i čiji se sadržaj povećava svakim pulsom. Kad je registar pun generira se prekid (eng. *interrupt*).

Ako registri tajmera koriste interni kristal kvarca za rad tada je moguće mjeriti vrijeme između dva događaja (ukoliko je početno vrijeme T1, a krajnje T2, proteklo vrijeme jednako je razlici T2-T1). Ako registri koriste pulseve iz vanjskog izvora tada taj tajmer postaje brojač.

3.1.6.1. Način rada

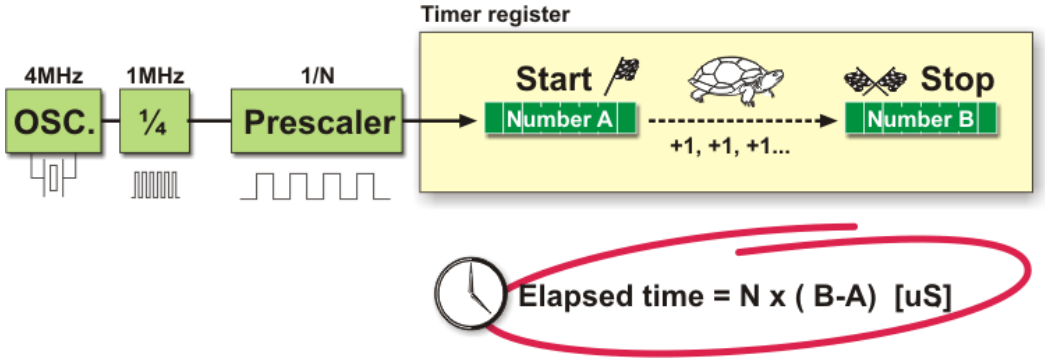


Slika 3.9 Mjerenje vremena tajmera [S11]

Na način prikazan na slici 3.9 jednostavno je izmjeriti kratke vremenske intervale (do 256 μs) zato jer je to najveći broj koji jedan 8-bitni registar može imati. Taj nedostatak može se riješiti na nekoliko načina: korištenjem sporijeg oscilatora, registara s više bitova, korištenjem preskalera (eng. *prescaler)* ili prekida. Prva dva rješenja imaju neke mane pa se najčešće koriste prekidi ili preskaler.

- **Korištenje preskalera u radu tajmera**

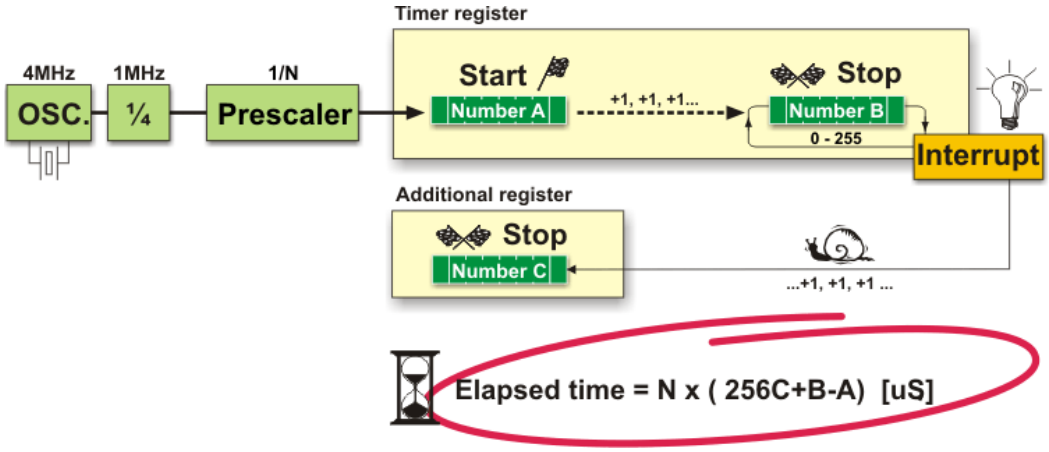
Preskaler je elektronički uređaj koji smanjuje frekvenciju za neki faktor. To znači da, ovisno o vrijednosti preskalera, moramo na ulaz dovesti 1, 2, 4 ili više pulseva na ulaz kako bi generirali jedan puls na izlazu. Jednom kad je takav krug stvoren unutar mikrokontrolera, preskalerov omjer je moguće mijenjati unutar programa. Koristi se kada je potrebno mjeriti duže vremenske periode. Preskaler koriste ili tajmer ili WDT (*Watch Dog Timer*).



Slika 3.10 Korištenje preskalera u radu tajmera [S12]

- **Korištenje prekida u radu tajmera**

Ako registar tajmer 8-bitni, najveći broj koji možemo zapisati je 255 (65 535 za 16-bitni registar). Ukoliko je taj broj premašen, tajmer će se resetirati i početi brojati od nule. To se stanje naziva preljev (eng. *overflow*). Ukoliko je to omogućeno unutar programa, takav preljev može uzrokovati prekid. Na primjer, registri stanja korišteni za brojanje sekundi, minuta ili dana mogu biti promijenjeni u rutini prekida. Cijeli proces (osim same rutine) se automatski izvodi u pozadini, što omogućuje glavnom krugu mikrokontrolera da izvodi druge operacije.



Slika 3.11 Korištenje prekida u radu tajmera [S13]

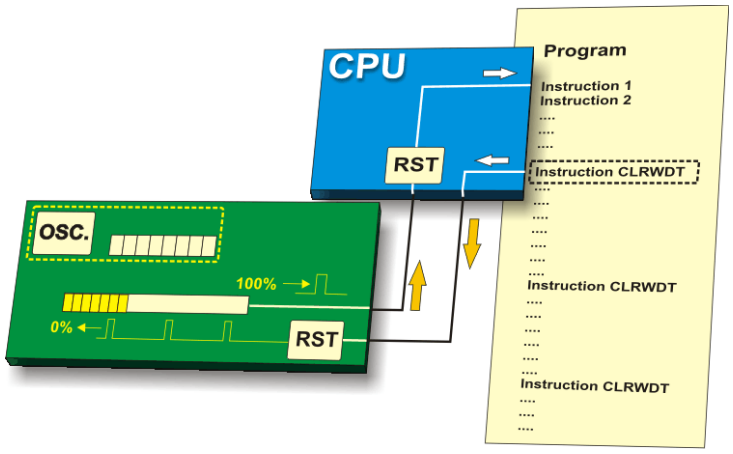
3.1.7. Brojač

Ako tajmer daje impulse u ulazni pin mikrokontrolera on postaje brojač. jedina razlika je ta što se pulsevi koji se broje dovode kroz portove, a njihova širina najčešće nije definirana.

3.1.8. Watch Dog Timer

Program u njega upisuje nulu svaki put kad kada se pravilno izvrši. Ako signal izostane, neće doći do upisivanja nule (znači da se mikrokontroler vrti u beskonačnoj petlji), vrijednost brojača će se povećavati do svoje maksimalne vrijednosti kada će sam resetirati mikrokontroler.

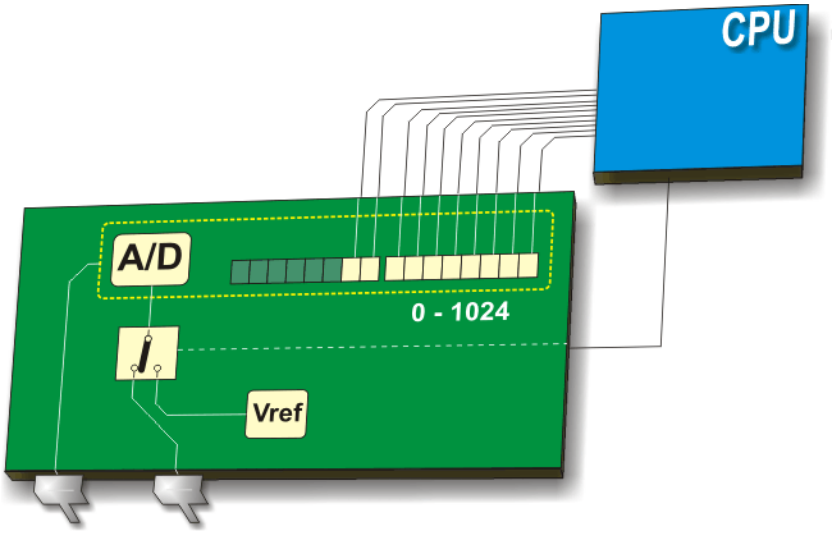
Na taj način onemogućava se duži nepravilan rad mikrokontrolera bez obzira na uzrok neispravnosti. Ova metoda višestruko povećava sigurnost sustava kojeg mikrokontroler nadzire/upravlja.



Slika 3.12 Način rada WDT-a [S14]

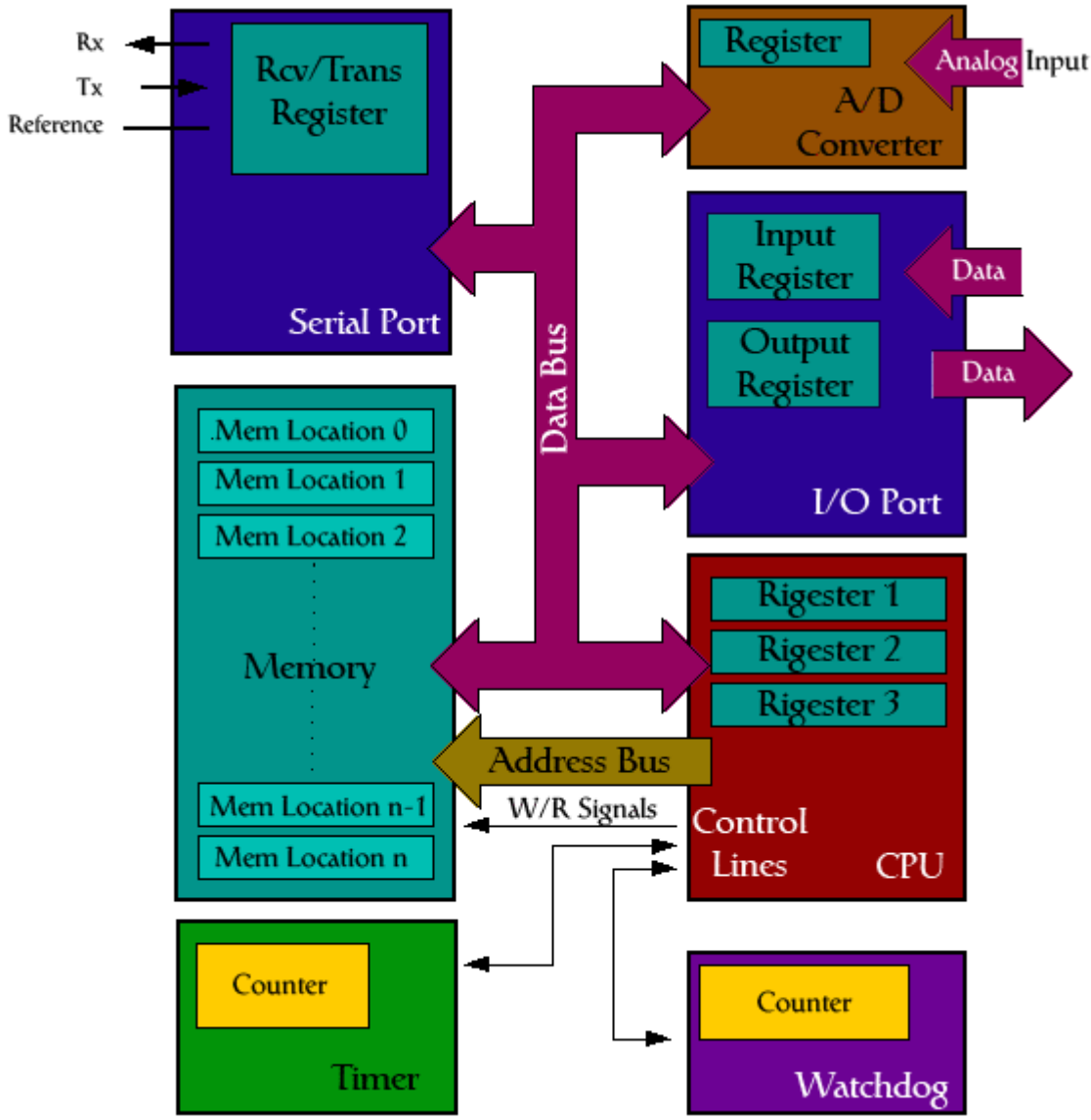
3.1.9. A/D pretvornik

Budući da su vanjski signali drukčiji od onih koje mikrokontroler razumije (logičke jedinice i nule), oni moraju biti pretvoreni kako bi ih mikrokontroler razumio. Analogno-digitalni pretvornik je elektronički uređaj koji kontinuirane signale pretvara u diskretne digitalne brojeve. Ovaj se modul dakle koristi kako bi se ulazni signal pretvorio iz analogne u binarnu vrijednost i proslijedio prema CPU-u na daljnju obradu.



Slika 3.13 A/D pretvornik [S15]

Svi dosad navedeni dijelovi mikrokontrolera sačinjavaju njegovu unutrašnjost prema sljedećoj slici (Slika 3.14) :



Slika 3.14 Unutrašnjost mikrokontrolera [S16]

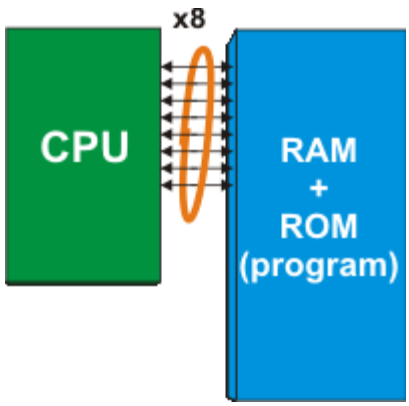
3.1.10. Unutarnja arhitektura

3.1.10.1. von Neumannova arhitektura

Mikrokontroleri koji koriste ovu arhitekturu imaju jedan memorijski blok i jednu 8-bitnu sabirnicu. Budući da se svi podatci izmjenjuju u jednoj sabirnici, komunikacija je spora i neefikasna. CPU može čitati instrukcije ili čitati/zapisivati podatke iz/u memoriju. Oboje nije moguće izvoditi istovremeno budući da instrukcije i podatci koriste istu sabirnicu. Na primjer, ukoliko program traži se memorijski registar RAM-a nazvan „SUM“ povećava za jedan (instrukcija: incf SUM), mikrokontroler će učiniti sljedeće:

1. Pročitati dio programa koji govori ŠTO treba napraviti (u ovom slučaju to je incf za povećanje)
2. Pročitati dalje istu instrukciju koja određuju na KOJEM podatku treba biti izvedena (u ovom slučaju to je SUM registar)
3. Nakon povećanja, sadržaj registra treba biti zapisan u registar iz kojeg je pročitan (adresa SUM registra)

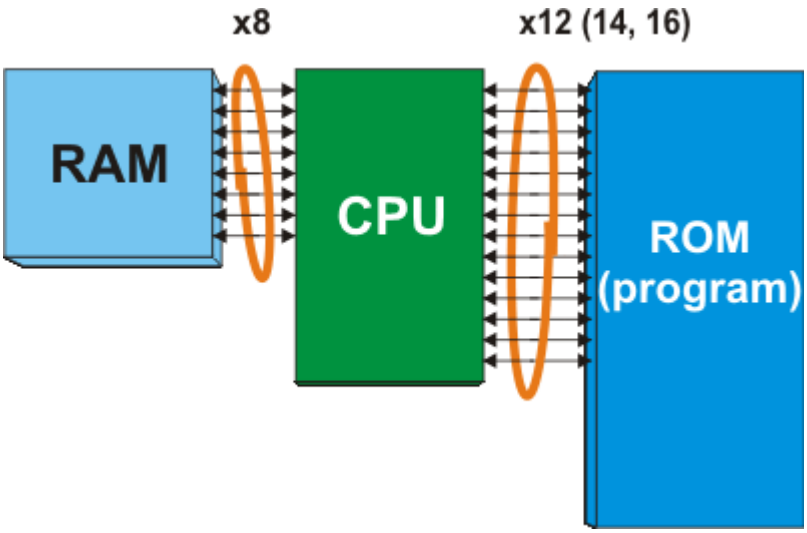
Ista sabirnica se koristi za sve tri operacije.



Slika 3.15 von Neumannova arhitektura [S17]

3.1.10.2. Harvard arhitektura

Mikrokontroleri koji koriste ovu arhitekturu imaju dvije sabirnice podataka. Jedna je 8-bitna i spaja CPU i RAM. Druga je 12, 14 ili 16 bitna i spaja CPU i ROM. Sukladno tome CPU može čitati instrukcije i pristupati memoriji u isto vrijeme. Budući da su svi registri RAM-a 8-bitni, svi se podatci unutar CPU-a razmjenjuju u tom formatu. Program pisan za neke od tih mikrokontrolera biti će spremljen u unutarnjem ROM-u mikrokontrolera prije nego se kompajlira u strojni jezik. Međutim, te memorijske lokacije nemaju 8, već 12, 14 ili 16 bitova. Ostatak bitova – 4, 6 ili 8 predstavlja instrukciju koja CPU-u govori što treba napraviti s 8-bitnim podatkom.



Slika 3.16 Harvard arhitektura [S18]

Prednosti ovakvog dizajna su sljedeće:

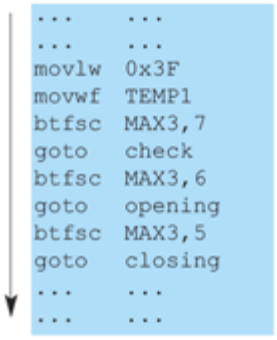
1. Svi podatci u programu su 8-bitni. Budući da sabirnica korištena za čitanje programa ima više linija (12, 14 ili 16) i instrukcije i podatci mogu biti čitani istovremeno korištenjem tih slobodnih bitova. Prema tome, sve instrukcije su izvršene u jednom ciklusu. Jedina iznimka je skok (eng. *jump*) instrukcija koja se izvodi u dva ciklusa.

2. Zahvaljujući činjenici da su ROM i RAM odvojeni, CPU može izvršiti dvije instrukcije istovremeno. Jednostavno, dok se izvodi čitanje i pisanje u RAM (kraj jedne instrukcije), sljedeća instrukcija se čita u drugoj sabirnici.

3. Korištenjem von Neumannove arhitekture nije nam poznato koliko će memorije zauzeti neki program. U osnovi, svaka programska instrukcija zauzima dvije memorijske lokacije (jedna sadrži informaciju o tome što treba biti izvedeno, a druga na čemu treba biti izvedeno). S Harvard arhitekturom, programska sabirnica je šira od jednog bajta, što omogućuje da se svaka programska riječ sastoji od instrukcije i podataka.

3.1.11. Set instrukcija

Instrukcije koje mikrokontroler može razumjeti su poznate kao set instrukcija. Pri pisanju programa u asemblerskom jeziku, mi zapravo zadajemo zadatke mikrokontroleru određivanjem instrukcija u redoslijedu u kojem moraju biti izvršene. Glavno ograničenje je broj slobodnih instrukcija.



Slika 3.17 Program u asembleru [S19]

3.1.11.1. RISC (Reduced Instruction Set Computer)

U ovom slučaju, mikrokontroler razumije i izvršava samo osnovne operacije (zbrajanje, oduzimanje, kopiranje itd.). Sve ostale kompliciranije operacije se izvode kombinirajući osnovne (na primjer, množenje se izvodi sukcesivnim zbrajanjem). Ograničenja su očita, no postoje i velike prednosti. Prvo, jezik je jednostavan za učenje. Uz to mikrokontroler je vrlo brz pa korisnik vidi samo rezultat svih aritmetičkih operacija.

3.1.11.2. CISC (Complex Instruction Set Computer)

CISC je suprotnost RISC-u. Mikrokontroleri su dizajnirani tako da razumiju preko 200 različitih instrukcija, vrlo su brzi, a mogu izvoditi mnogo operacija. Glavna mana ovog jezika je ta što je težak za učenje.

Instrukcija	Opis	Trajanje u strojnim ciklusima	Utječe na STATUS bit-ove
addwf f, d	Zbrajanje akumulatora i registra f	1	C, DC, Z
andwf f, d	Logički I akumulatora i registra f	1	Z
clrf f	Brisanje registra f	1	Z
clrw	Brisanje akumulatora	1	Z
comf f, d	Komplement registra f	1	Z
decf f, d	Smanjenje registra f za 1	1	Z
decfsz f,d	Smanjenje registra f za 1, preskače sljedeću instrukciju ako je rezultat 1	1(2)	nijedan
incf f, d	Povećanje registra f za 1	1	Z
incfsz f, d	Povećanje registra f za 1, preskače sljedeću instrukciju ako je rezultat 0	1(2)	nijedan
iorwf f, d	Inkluzivni ILI akumulatora i f registra	1	Z
movf f, d	Premještanje sadržaja rgistra f	1	Z
movwf f	Premještanje sadržaja akumulatora u registar f	1	nijedan
nop	Nul-operacija	1	nijedan
rlf f, d	Rotiranje registra f u lijevo kroz Carry bit	1	C
rrf f, d	Rotiranje registra f u desno kroz Carry bit	1	C
subwf f, d	Oduzimanje akumulatora od registra f	1	C, DC, Z
swapf f, d	Zamjena riječi registra f	1	nijedan
xorwf f, d	Ekskluzivni ILI akumulatora i registra f	1	Z
5.4.1.1.1 Operacije nad bit-ovima			
bcf f, b	Brisanje bit-a b registra f	1	nijedan
bsf f, b	Postavljanje bit-a b registra f	1	nijedan
btfsc f, b	Preskače sljedeću instrukciju ako je rezultat 0	1(2)	nijedan
btfss f, b	Preskače sljedeću instrukciju ako je rezultat 0	1(2)	nijedan
Instrukcije nad konstantama i kontrolne instrukcije			
addlw k	Zbrajanje akumulatora i konstante k	1	C, DC, Z
andlw k	Logički I akumulatora i konstante k	1	Z
call k	Poziv potprograma	2	nijedan
clwrdt	Brisanje sigurnosnog brojača	1	$\overline{TO}$, $\overline{PD}$
goto k	Grananje programa	2	nijedan
iorlw k	Inkluzivni ILI akumulatora i konstante k	1	Z
movlw k	Upisivanje konstante k u akumulator	1	nijedan
retfie	Povratak iz prekidne rutine	2	nijedan
retlw k	Povratak iz potprograma s vrijednošću k u akumulatoru	2	nijedan
return	Povratak iz potprograma	2	nijedan
sleep	Postavljanje mikrokontrolera u standby mode	1	$\overline{TO}$, $\overline{PD}$
sublw k	Oduzimanje akumulatora od konstante k	1	C, DC, Z
xorlw	Ekskluzivni ILI akumulatora i konstante k	1	Z

Tablica 3.1 Set instrukcija za mikrokontroler PIC16F84 [T1]

3.2. PIC mikrokontroleri

PIC mikrokontroleri razvijeni su od tvrtke Microchip Technology. Pravo ime ovih mikrokontrolera je PICmicro (*Peripheral Interface Controller*), ali je poznatiji kao PIC. Prvi njegov predak je 1975. stvorila tvrtka General Instruments. Čip nazvan PIC1650 bio je namijenjen za potpuno drugačije namjene. Deset godina kasnije, dodavanjem EEPROM memorije, pretvoren je u pravi PIC mikrokontroler.

Family	ROM [Kbytes]	RAM [bytes]	Pins	Clock Freq. [MHz]	A/D Inputs	Resolution of A/D Converter	Compar-ators	8/16 – bit Timers	Serial Comm.	PWM Outputs	Others
Base-Line 8 - bit architecture, 12-bit Instruction Word Length											
PIC10FXXX	0.375 - 0.75	16 - 24	6 - 8	4 - 8	0 - 2	8	0 - 1	1 x 8	-	-	-
PIC12FXXX	0.75 - 1.5	25 - 38	8	4 - 8	0 - 3	8	0 - 1	1 x 8	-	-	EEPROM
PIC16FXXX	0.75 - 3	25 - 134	14 - 44	20	0 - 3	8	0 - 2	1 x 8	-	-	EEPROM
PIC16HVXXX	1.5	25	18 - 20	20	-	-	-	1 x 8	-	-	Vdd = 15V
Mid-Range 8 - bit architecture, 14-bit Instruction World Length											
PIC12FXXX	1.75 - 3.5	64 - 128	8	20	0 - 4	10	1	1 - 2 x 8 1 x 16	-	0 - 1	EEPROM
PIC12HVXXX	1.75	64	8	20	0 - 4	10	1	1 - 2 x 8 1 x 16	-	0 - 1	-
PIC16FXXX	1.75 - 14	64 - 368	14 - 64	20	0 - 13	8 or 10	0 - 2	1 - 2 x 8 1 x 16	USART I2C SPI	0 - 3	-
PIC16HVXXX	1.75 - 3.5	64 - 128	14 - 20	20	0 - 12	10	2	2 x 8 1 x 16	USART I2C SPI	-	-
High-End 8 - bit architecture, 16-bit Instruction Word Length											
PIC18FXXX	4 - 128	256 - 3936	18 - 80	32 - 48	4 - 16	10 or 12	0 - 3	0 - 2 x 8 2 - 3 x 16	USB2.0 CAN2.0 USART I2C SPI	0 - 5	-
PIC18FXXJXX	8 - 128	1024 - 3936	28 - 100	40 - 48	10 - 16	10	2	0 - 2 x 8 2 - 3 x 16	USB2.0 USART Ethernet I2C SPI	2 - 5	-
PIC18FXXKXX	8 - 64	768 - 3936	28 - 44	64	10 - 13	10	2	1 x 8 3 x 16	USART I2C SPI	2	-

Tablica 3.2 Vrste PIC mikrokontrolera i njihove specifikacije [T2]

3.2.1. Programiranje

Programiranje mikrokontrolera izvodi se u jezicima poput Asemblera, C-a, Basic-a i Pascal-a. Asembler spada u niže jezike čije programiranje traje vrlo dugo, ali zauzimaju najmanje mjesta u memoriji i daju najbolje rezultate u vezi brzine izvršenja programa. Programi u C-u su jednostavniji za pisanje, lakši za razumijevanje, ali sporiji za izvršenje od onih pisanih u asembleru. Basic je najjednostavniji za učenje, a njegove instrukcije su najbliže čovjekovom načinu razmišljanja. Međutim, poput C-a, i on je sporiji od asemblera.

Aplikacija izrađena za završni rad generira kod pisan u programskom jeziku mikroPascal koji je jednostavan za učenje i korištenje, sadrži mnoge ugrađene funkcije, no nije fleksibilan i popularan poput C-a.

```
program LED_blinking_PASCAL;
begin
  PORTC := 0;
  TRISC := 0;
  ANSEL := 0;
  ANSELH := 0;
  while TRUE do
  begin
      PORTC := not PORTC;
      Delay_ms(1000);
  end;
end.
```

```
program LED_blinking_BASIC;
main
  PORTC = 0;
  TRISC = 0;
  ANSEL = 0;
  ANSELH = 0;
  while TRUE
      PORTC = not PORTC
      Delay_ms(1000)
  wend;
end.
```

```
void main() {
  PORTC = 0;
  TRISC = 0;
  ANSEL = 0;
  ANSELH = 0;
  while (1) {
      PORTC =  ~PORTC
      Delay_ms(1000);
  }
}
```

U gornjim primjerima prikazana je razlika u sintaksi programa za „blinkanje“ LED diodi spojenih na PORTC u intervalu od jedne sekunde.

```
bsf                                                    03h,5
movlw       00h
movwf       85h
bcf         03h,5
```

Za usporedbu, u gornjem kodu je napisana naredba za postavljanje svih pinova PORTA kao izlaznih u Asembleru kod mikrokontrolera PIC16F84. Kod Pascala, a slično i kod Basica i C-a, ta se naredba piše u jednoj liniji – PORTA := 0; Vidljivo je da je za programiranje u asembleru nužno znati set instrukcija mikrokontrolera te adrese registara što znatno produžuje vrijeme pisanje programa.

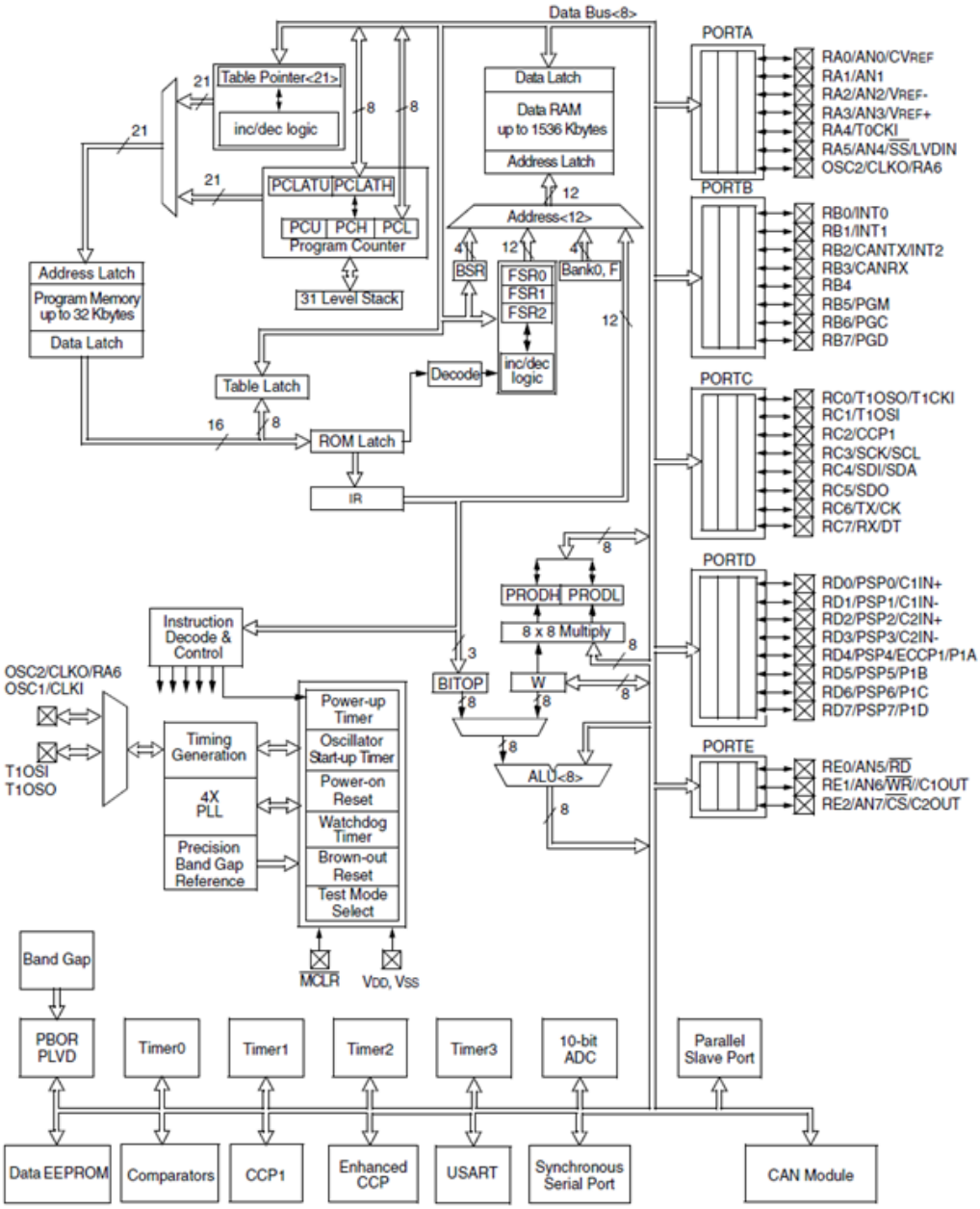
4. Opis razvijenog računalnog programa i registara čije se vrijednosti generiraju

4.1. PIC 18F448

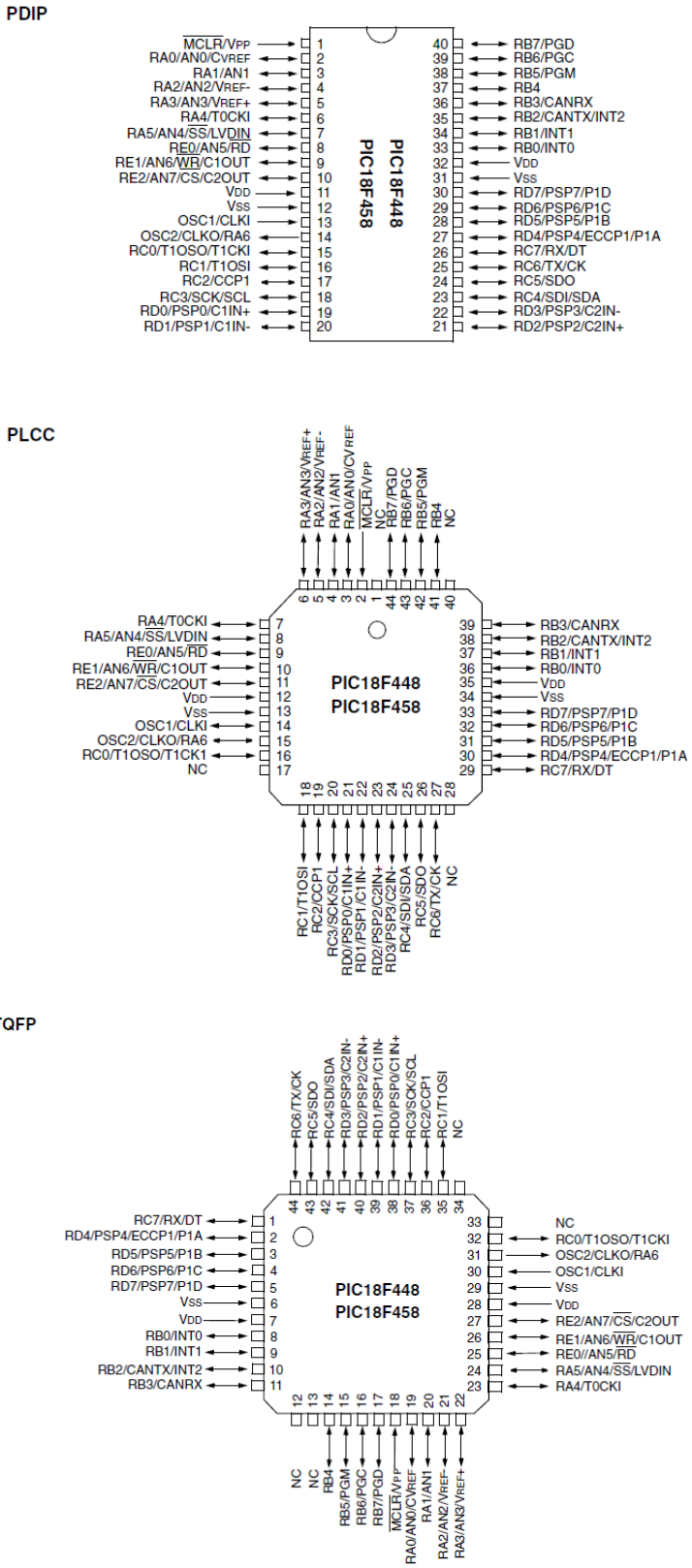
U sljedećim poglavljima biti će pojedinačno opisana svaka kartica Visual Basic programa. Kako svakim dijelom programa generiramo kod za upravljanje određenim dijelovima PIC mikrokontrolera, ti dijelovi će biti detaljnije opisani na primjeru mikrokontrolera PIC18F448.

Features		PIC18F448
Operating Frequency		DC – 40 MHz
Internal Program Memory	Bytes	16K
	# of Single-Word Instructions	8192
Data Memory (Bytes)		768
Data EEPROM Memory (Bytes)		256
Interrupt Sources		21
I/O Ports		Ports A, B, C, D, E
Timers		4
Capture/Compare/PWM Modules		1
Enhanced Capture/Compare/ PWM Modules		1
Serial Communications		MSSP, CAN, Addressable USART
Parallel Communications (PSP)		Yes
10-bit Analog-to-Digital Converter		8 input channels
Analog Comparators		2
Analog Comparators VREF Output		Yes
Resets (and Delays)		POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)
Programmable Low-Voltage Detect		Yes
Programmable Brown-out Reset		Yes
CAN Module		Yes
In-Circuit Serial Programming™ (ICSP™)		Yes
Instruction Set		75 Instructions
Packages		40-pin PDIP 44-pin PLCC 44-pin TQFP

Tablica 4.1 Značajke mikrokontrolera PIC18F448 [T3]



Slika 4.1 Blok shema mikrokontrolera PIC18F448 [S20]



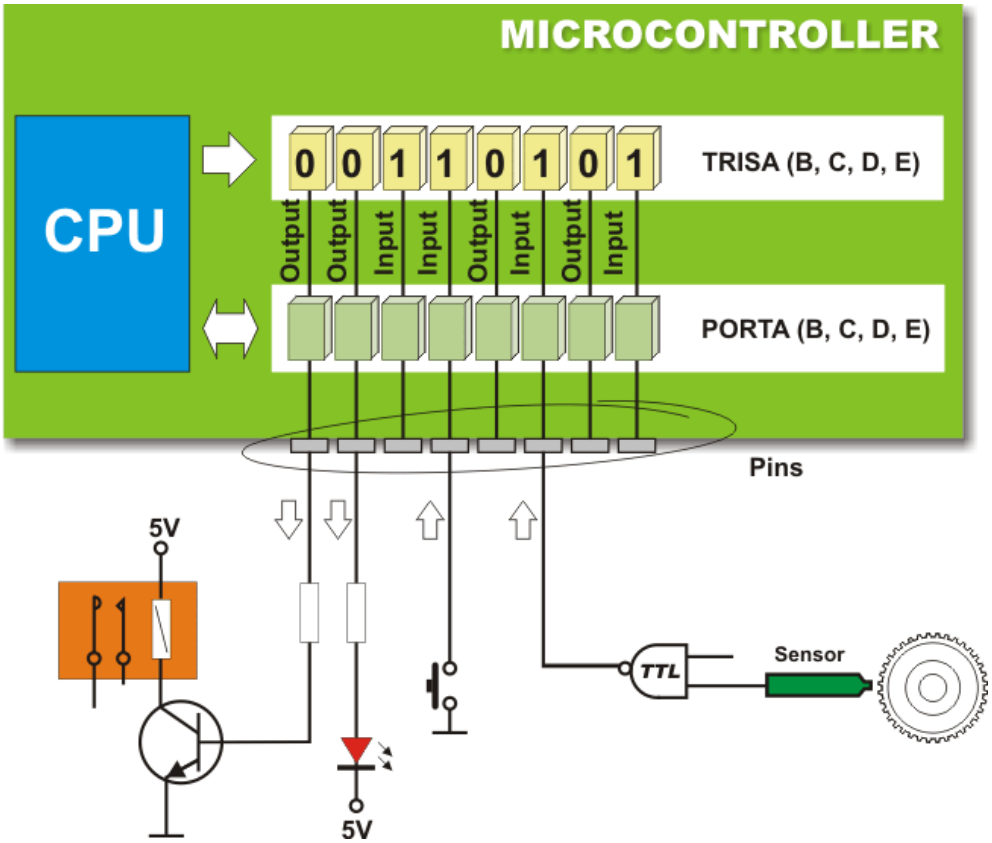
Slika 4.2 Dijagram pinova mikrokontrolera PIC 18F448 [S21]

4.2. Ulazi/Izlazi

Jedna od najvažnijih značajki mikrokontrolera je broj ulaznih i izlaznih pinova korištenih za spajanje s vanjskim uređajima. PIC18F448 sadrži 34 pina podijeljena u pet portova nazvanih A, B, C, D. Svi pinovi imaju nekoliko zajedničkih osobina:

- iz praktičnih razloga, skoro svi I/O pinovi imaju dvije ili tri funkcije, a ukoliko se pin koristi za neku drugu funkciju, on ne može biti korišten kao ulazno-izlazni pin.
- svaki port ima svoj „satelit“ tj. odgovarajući TRIS registar koji određuje svojstva, ali ne i sadržaj portova.

Postavljanjem nekog bita TRIS registra na logičku nulu (dovođenjem napona 0V), odgovarajući pin porta je definiran kao izlaz (napon 5V). Sukladno tome, postavljanjem jedinice na neki od bitova TRIS registra, odgovarajući pin je postavljen kao ulaz.



Slika 4.3 Ulazno/izlazni portovi [S22]

4.2.1. Portovi

4.2.1.1. PORTA i TRISA registar

PORTA je 7-bitni, dvosmjerni port. Njime upravljamo pomoću registra TRISA. Svi pinovi PORTA (RA) su digitalni, a pet ih je i analogno. Registrom ADCON1 određujemo hoće li pojedini ulazi biti digitalni ili analogni na portu A i E za ovaj PIC.

Name	Bit#	Buffer	Function
RA0/AN0/CVREF	bit 0	TTL	Input/output, analog input or analog comparator voltage reference output.
RA1/AN1	bit 1	TTL	Input/output or analog input.
RA2/AN2/VREF-	bit 2	TTL	Input/output, analog input or VREF-.
RA3/AN3/VREF+	bit 3	TTL	Input/output, analog input or VREF+.
RA4/T0CKI	bit 4	ST/OD	Input/output, external clock input for Timer0, output is open-drain type.
RA5/AN4/$\overline{SS}$/LVDIN	bit 5	TTL	Input/output, analog input, slave select input for synchronous serial port or Low-Voltage Detect input.
OSC2/CLKO/RA6	bit 6	TTL	Oscillator clock output or input/output.

Tablica 4.2 Funkcije PORTA [T4]

Iz gornje tablice je vidljivo da je za ovaj PIC na portu A moguće je postaviti pinove RA0, RA1, RA2, RA3 i RA5 kao analogne ulaze. Pin RA4 je multipleksiran s ulazom sata Timer0 modula, on je također i ulaz s Schmittovim okidnim sklopom. Svi ostali pinovi imaju TTL (eng. *Transistor-transistor logic*) ulazne nivoe. TRISA registar upravlja svim pinovima porta A čak i kada su postavljeni kao analogni ulazi, a korisnik mora osigurati da su bitovi u registru TRISA postavljeni na jedinicu kada se ti pinovi koriste kao takvi.

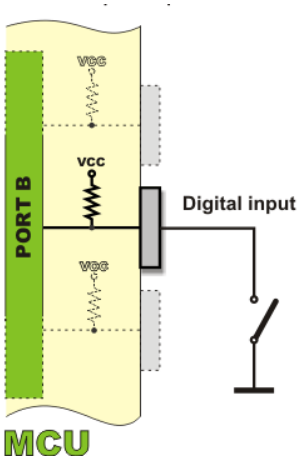
4.2.1.2. PORT B i TRISB registar

Port B je 8-bitni, dvosmjerni port. Hoće li pin služiti kao ulaz ili izlaz određuje TRISB. Svi pinovi Porta B imaju unutarnji *pull-up* otpornik što ih čini idealnima za spajanje tipkala, optokaplera i prekidača. *Pull-up* otpornik je povezan između napona napajanja mikrokontrolera i ulaznog pina, pri čemu je sklopka uvijek spojena između pina i mase. Kada je sklopka otvorena, *pull-up* otpornik dovodi napon napajanja direktno na pin, postavljajući ga u stanje logičke jedinice sve dok se ne pritisne sklopka. Bez njega bi napon na pinu najvjerojatnije otišao u nedefinirano stanje. Svi *pull-up* otpornici se uključuju postavljanjem bita $\overline{RPBU}$ INTCON2 registra na nulu

(komplementirani bit). Četiri pina (RB4-RB7) mogu izazvati prekid, ali samo ako su postavljeni kao ulazi. Njima se upravlja pomoću INTCON registra.

Name	Bit#	Buffer	Function
RB0/INT0	bit 0	TTL/ST(1)	Input/output pin or external interrupt 0 input. Internal software programmable weak pull-up.
RB1/INT1	bit 1	TTL/ST(1)	Input/output pin or external interrupt 1 input. Internal software programmable weak pull-up.
RB2/CANTX/ INT2	bit 2	TTL/ST(1)	Input/output pin, CAN bus transmit pin or external interrupt 2 input. Internal software programmable weak pull-up.
RB3/CANRX	bit 3	TTL	Input/output pin or CAN bus receive pin. Internal software programmable weak pull-up.
RB4	bit 4	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB5/PGM	bit 5	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. Low-voltage serial programming enable.
RB6/PGC	bit 6	TTL/ST(2)	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. Serial programming clock.
RB7/PGD	bit 7	TTL/ST(2)	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. Serial programming data.

Tablica 4.3 Funkcije PORTB [T5]



Slika 4.4 Pin sa pull-up otpornikom [S23]

4.2.1.3. PORTC i TRISC registar

Port C je također 8-bitni,dvosmjerni port, no za razliku od porta A i B, svi njegovi pinovi imaju ulaze s Schmittovim okidnim sklopom. Za pravilan rad mikrokontrolera preporučuje se da ulazni naponi napajanja (za napajanje od 5 V) budu u rasponu od 0 do 0.8 V za logičku nulu, i od 2.4 do 5 V za logičku jedinicu. No ukoliko na pin dovedemo napon od, na primjer, 1.5 V tada njega mikrokontroler može pročitati ili kao nulu ili kao jedinicu tj. javlja se nedefinirano stanje. Za sprečavanje pojavljivanja ovakvog slučaja koriste se ulazi s Schmittovim okidnim sklopom. Stanja

na takvom pinu promijeniti će se tek onda kada se signal približi njegovoj definiranoj razini, a ukoliko se nakon toga vrati na neodređeno stanje, ulaz će i dalje pokazivati njegovo posljednje stabilno stanje. Pri uključivanju vanjskih funkcija, posebnu pažnju treba obratiti na definiranje TRIS bitova za svaki pin porta C budući da neki vanjski uređaji poništavaju stanje TRIS bita postavljajući pin kao izlaz, dok neki uređaji postavljaju pin kao ulazni.

Name	Bit#	Buffer Type	Function
RC0/T1OSO/T1CKI	bit 0	ST	Input/output port pin, Timer1 oscillator output or Timer1/Timer3 clock input.
RC1/T1OSI	bit 1	ST	Input/output port pin or Timer1 oscillator input.
RC2/CCP1	bit 2	ST	Input/output port pin or Capture 1 input/Compare 1 output/ PWM1 output.
RC3/SCK/SCL	bit 3	ST	Input/output port pin or synchronous serial clock for SPI™/I²C™.
RC4/SDI/SDA	bit 4	ST	Input/output port pin or SPI data in (SPI mode) or data I/O (I²C mode).
RC5/SDO	bit 5	ST	Input/output port pin or synchronous serial port data output.
RC6/TX/CK	bit 6	ST	Input/output port pin, addressable USART asynchronous transmit or addressable USART synchronous clock.
RC7/RX/DT	bit 7	ST	Input/output port pin, addressable USART asynchronous receive or addressable USART synchronous data.

Tablica 4.4 Funkcije PORTC [T6]

Drugi pin porta C je spojen na CCP (*Capture/Compare/PWM*) modul, što nam je korisno za generiranje PWM signala koji je opisan u jednom od sljedećih poglavlja

.

4.2.1.4. PORTD i TRISD registar

Također 8-bitni, dvosmjerni port čiji pinovi koriste međuspremnik (eng. *buffer*) s Schmittovim okidnim sklopom kada se koriste kao ulazi/izlazi, a kada je port konfiguriran kao paralelni *slave* port (eng. *Parallel Slave Port*) koriste *buffer* s TTL logikom.

Name	Bit#	Buffer Type	Function
RD0/PSP0/C1IN+	bit 0	ST/TTL(1)	Input/output port pin, Parallel Slave Port bit 0 or C1IN+ comparator input.
RD1/PSP1/C1IN-	bit 1	ST/TTL(1)	Input/output port pin, Parallel Slave Port bit 1 or C1IN- comparator input.
RD2/PSP2/C2IN+	bit 2	ST/TTL(1)	Input/output port pin, Parallel Slave Port bit 2 or C2IN+ comparator input.
RD3/PSP3/C2IN-	bit 3	ST/TTL(1)	Input/output port pin, Parallel Slave Port bit 3 or C2IN- comparator input.
RD4/PSP4/ECCP1/P1A	bit 4	ST/TTL(1)	Input/output port pin, Parallel Slave Port bit 4 or ECCP1/P1A pin.
RD5/PSP5/P1B	bit 5	ST/TTL(1)	Input/output port pin, Parallel Slave Port bit 5 or P1B pin.
RD6/PSP6/P1C	bit 6	ST/TTL(1)	Input/output port pin, Parallel Slave Port bit 6 or P1C pin.
RD7/PSP7/P1D	bit 7	ST/TTL(1)	Input/output port pin, Parallel Slave Port bit 7 or P1D pin.

Tablica 4.5 Funkcije PORTD [T7]

Postavljanje četvrtog bita registra TRISE – PSPMODE na jedinicu, port D je konfiguriran kao paralelni *slave* port (eng. *Parallel Slave Port* – PSP) ili mikroprocesorski port. U slave načinu rada port se asinkrono čita, a izvana je moguće pisati u njega. Postavljanjem kontrolnog bita PSPMODE pinovi porta E postaju kontrolni ulazi za mikroprocesorski port. Tada pin RE0 postaje $\overline{RD}$ (*read*), RE1 $\overline{WR}$ (*write*), a RE2 $\overline{CS}$ (*chip select*). Ukoliko su na $\overline{RD}$ i $\overline{CS}$ logičke nule, tada je moguće čitati s PSP-a, a kada su na $\overline{WR}$ i $\overline{CS}$ moguće je pisati u PSP.

4.2.1.5. PORTE i TRISE registar

PORTE je 3- bitni, dvosmjerni port. Njegov TRIS registar je drukčiji od TRIS registara drugih portova. Dok je kod drugih portova TRIS registar služio za postavljanje pina kao ulaznog ili izlaznog, kod TRISE registar za to služe samo prva tri bita. Četvrti bit je već spomenuti PSPMODE, a ostala tri bita kontroliraju ulazni i izlazni međuspremnik u mikroprocesorskom načinu rada.

R-0	R-0	R/W-0	R/W-0	U-0	R/W-1	R/W-1	R/W-1
IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0
bit 7							bit 0

Slika 4.5 TRISE registar [S24]

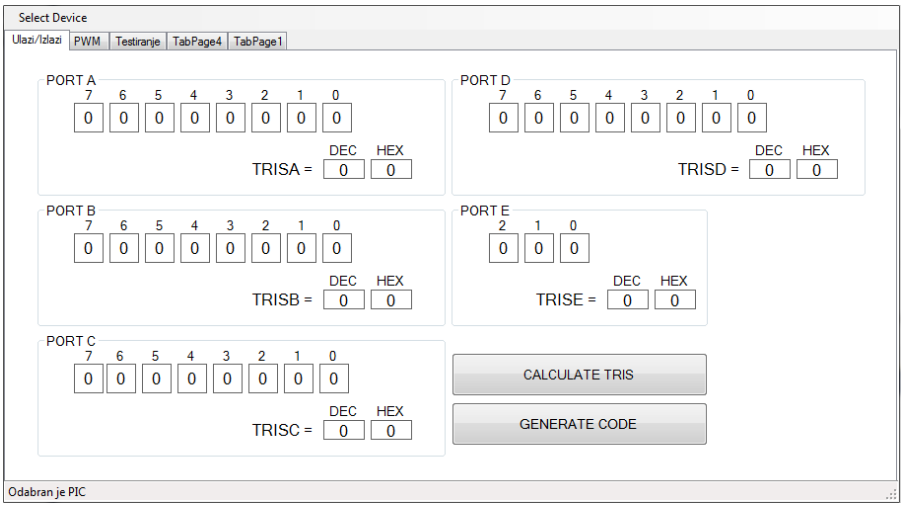
Name	Bit#	Buffer Type	Function
RE0/AN5/$\overline{RD}$	bit 0	ST/TTL(1)	Input/output port pin, analog input or read control input in Parallel Slave Port mode.
RE1/AN6/$\overline{WR}$/C1OUT	bit 1	ST/TTL(1)	Input/output port pin, analog input, write control input in Parallel Slave Port mode or Comparator 1 output.
RE2/AN7/$\overline{CS}$/C2OUT	bit 2	ST/TTL(1)	Input/output port pin, analog input, chip select control input in Parallel Slave Port mode or Comparator 2 output.

Tablica 4.6 Funkcije PORTD [T8]

4.2.1.6. Sažetak

- Pri izradi uređaja potrebno je odabrati port kroz koji će mikrokontroler komunicirati s okolinom. Ukoliko korisnik namjerava koristiti digitalne ulaze/izlaze moguće je koristiti sve portove, dok je za kao analogne moguće koristiti samo određene portove
- Svaki port se može konfigurirati kao ulazni ili izlazni pomoću svog TRIS registra
- Analogne portove potrebno je definirati na početku programa
- Sklopke se spajaju na port B budući da on ima implementirane pull-up otpornik. *Pull-up* otpornike je potrebno aktivirati
- Ponekad je potrebno reagirati što prije na promjenu stanja ulaznih pinova. Kako ne bi trebali pisati program za to dovoljno je spojiti takve ulaze na port B i aktivirati prekid na promjenu napona.

4.2.2. Razvijeni računalni program – Ulazi/Izlazi



Slika 4.6 Korisničko sučelje

Na gornjoj slici (Slika 4.6) prikazan je izgled sučelja za postavljanje pinova mikrokontrolera kao ulaznih ili izlaznih. Jednostavno se pritiskom na kvadrat ispod broja bita porta mijenja vrijednost tog bita. Kod za jedan takav kvadrat, a i za sve ostale je sljedeći:

```
Private Sub LabelPortE1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles LabelPortE1.Click
        If LabelPortE1.Text = 0 Then
            LabelPortE1.Text = 1
        Else
            LabelPortE1.Text = 0
        End If
    End Sub
```

Nakon što se svim pinovima zadaju željene vrijednost, pritiskom na tipku „Izračunaj TRIS“ moguće je izračunati dekadsku vrijednost TRIS registra porta koja nam koristi u mikroPaskalu. Kod za računanje TRISA je:

```
Private Sub ButtonIzracunaj_Click_1(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ButtonIzracunaj.Click

        If ToolStripStatusLabelPIC.Text = "" Then
            MsgBox("Odaberite PIC!", 64, "Error!")
            Exit Sub
        End If

        LabelTrisARez.Text = Convert.ToDouble(LabelPortA7.Text) * 2 ^ 7 +
Convert.ToDouble(LabelPortA6.Text) * 2 ^ 6 + Convert.ToDouble(LabelPortA5.Text) * 2 ^
5 + Convert.ToDouble(LabelPortA4.Text) * 2 ^ 4 + Convert.ToDouble(LabelPortA3.Text) *
2 ^ 3 + Convert.ToDouble(LabelPortA2.Text) * 2 ^ 2 +
Convert.ToDouble(LabelPortA1.Text) * 2 ^ 1 + Convert.ToDouble(LabelPortA0.Text) * 2 ^
0
        LabelTrisARezHex.Text = "0x" & Hex(LabelTrisARez.Text)
```

```
    End Sub
```

Iz koda je vidljivo da je nužno tekst iz *labela* pretvoriti u odgovarajuću brojčanu vrijednost korištenjem funkcije *Convert.ToDouble* (ili CDbl). Nakon toga slijedi računanje dekadske vrijednost binarnog broja. U kod je unesen IF uvjet koji onemogućuje računanje TRIS registra ukoliko PIC nije odabran budući da svi PIC-evi ne koriste sve pinove. Tako, na primjer PIC18F448 ne koristi sedmi bit porta A pa se on pri odabiru PIC-a deaktivira:

```
Private Sub F448ToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles F448ToolStripMenuItem.Click

        datasheet.Filter = "Text (.txt)|dsPIC18F448.txt"

        If datasheet.ShowDialog = DialogResult.OK Then
            FormDS.Text = "PIC18F448"
            FormDS.LabelPIC.Text = "PIC18F448"
            fileReader = My.Computer.FileSystem.ReadAllText(datasheet.FileName)
            FormDS.LabelDSPIC.Text = fileReader
            FormDS.Show()
            ToolStripStatusLabelPIC.Text = "18F448"
            LabelPortA7.Enabled = False
            Label1.Enabled = False
            LabelPortA7.Text = 0
        End If

End Sub
```

Prilikom odabira željenog PIC-a pojavljuje se prozor koji traži odabir tekstualne datoteke u kojoj se nalaze svi podatci relevantni za generiranje kodova u programu. Izgled takve datoteke za PIC18F448 je:

```
trisA_as_input = 255
trisB_as_input = 255
trisC_as_input = 255
trisD_as_input = 255
trisE_as_input = 7
adcon1 = 0110
Fosc = 40 MHz
Max_PWM_frequency = 10 MHz
Max_timer_period_8bit = 6,5536 ms
Max_timer_period_16bit = 1677,7216 ms
```

U datoteci se nalaze vrijednosti TRIS registara koje odgovarajući port postavljaju kao ulazni, vrijednost ADCON1 registra za postavljanje digitalnih ulaza ili izlaza na PORTA i PORTE (za PIC18F448), vrijednost frekvencije oscilatora, maksimalne frekvencije PWM, te najvećeg perioda tajmera u 8-bitnom i 16-bitnom načinu rada. U prozoru za odabir tekstualne datoteke, koji se poziva funkcijom *ShowDialog*, moguće je odabrati samo tekstne datoteke specifičnog naziva. To se postiže funkcijom *Filter*. Na kraju nam još preostaje generirati kod što činimo pritiskom na tipku „Generiraj kod“:

```
    Private Sub ButtonGenKod_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ButtonGenKod.Click

        browser.RootFolder = Environment.SpecialFolder.Desktop
        browser.Description = "Odaberite lokaciju za spremanje koda!"

        If browser.ShowDialog = DialogResult.OK Then
            kod = browser.SelectedPath & "\CodeUI.txt"
            trisA = LabelTrisARez.Text
            trisB = LabelTrisBRez.Text
            trisC = LabelTrisCRez.Text
            trisD = LabelTrisDRez.Text
            trisE = LabelTrisERez.Text

            Dim pisac As New System.IO.StreamWriter(kod)

            pisac.Write("trisA := " & trisA & ";" & vbNewLine)
            pisac.Write("trisB := " & trisB & ";" & vbNewLine)
            pisac.Write("trisC := " & trisC & ";" & vbNewLine)
            pisac.Write("trisD := " & trisD & ";" & vbNewLine)
            pisac.Write("trisE := " & trisE & ";")
            pisac.Close()

            MsgBox("Kod je spremljen u " & kod & "!", 64, "Operation Completed!")
        End If

    End Sub
```

Pritiskom tipke „Generiraj kod“ pojavljuje se prozor koji traži zadavanje lokacije na disku za spremanje koda. Nakon što je ona odabrana (pritiskom na OK) u varijablu *kod* (*string* tip) upisuje se odabrana lokacija, a na njen kraj dodaje se „\CodeUI.txt“ što predstavlja naziv datoteke. Da bi mogli upisivati u datoteku, deklarira se varijabla *pisac* kao System.IO.StreamWriter(). Da bi koristili *StreamWriter* nužno je u VB uvesti *System.IO namespace* (imenski prostor) na početku programa:

```
Imports System.IO
```

System.IO u .NET Framework-u pruža nekoliko klasa za rad s tekst dokumentima, binarnim dokumentima, tokovima podataka i direktorijima. Klasa koja nam je potrebna za unos teksta u dokument je *StreamWriter*. Ta klasa omogućuje kopiranje, brisanje, upravljanje atributima i postojanjem datoteka. Nakon deklariranja varijable kao *StreamWriter* slijedi unos teksta za koji nam služi funkcija *Write*. Kao što je vidljivo iz koda, pod navodnike postavljamo tekst, a varijable koje želimo u tekstu moramo spojiti s ostatkom teksta pomoću operatora & (AND). Dobiven kod spreman za unos u mikroPascal je:

```
trisA := 80;
trisB := 199;
trisC := 217;
trisD := 237;
trisE := 3;
```

4.3. Testiranje

4.3.1. ADCON1 registar

ADCON1 registrom mikrokontrolera PIC18F448 moguće je portove A i E postaviti kao analogne ulaze ili digitalne ulaze/izlaze.

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

Slika 4.7 AD Control registar 1 [S25]

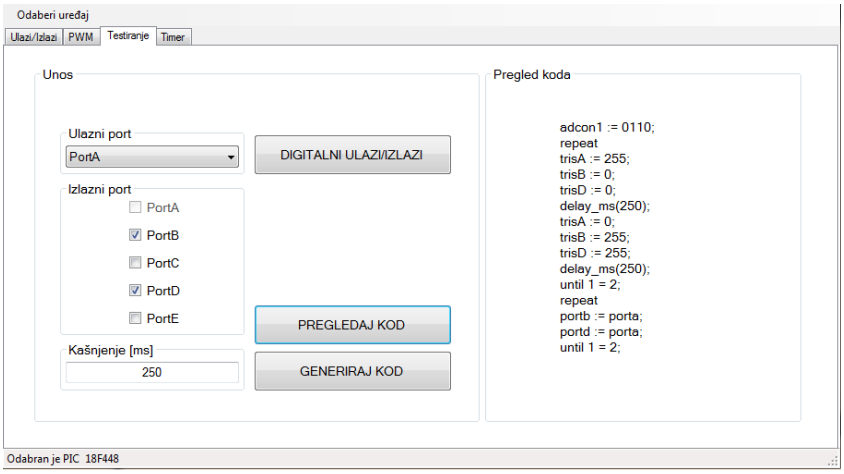
Za postavljanje digitalnih ulaza/izlaza ili analognih izlaza dovoljna su nam prva četiri bita – PCFG0, PCGF1, PCGF2 i PCGF3. Njihovom kombinacijom moguće je postaviti portove kao analogne ili digitalne prema sljedećoj tablici:

PCFG	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-
0000	A	A	A	A	A	A	A	A	VDD	VSS
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS
0010	D	D	D	A	A	A	A	A	VDD	VSS
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS
0100	D	D	D	D	A	D	A	A	VDD	VSS
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS
011x	D	D	D	D	D	D	D	D	—	—
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2
1001	D	D	A	A	A	A	A	A	VDD	VSS
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2
1110	D	D	D	D	D	D	D	A	VDD	VSS
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2

Tablica 4.7 Kombinacije ADCON1 bitova [T9]

Analogni referentni napon je softverski podesiv, a moguće je odabrati pozitivni ili negativni napon napajanja uređaja (V_{DD} i V_{SS}) ili naponsku razinu na RA3/AN3/V_{REF+} i RA2/AN2/V_{REF-} pinu.

4.3.2. Razvijeni računalni program – Testiranje



Slika 4.8 Korisničko sučelje

U ovom dijelu programa moguće je generirati kod za testiranje mikrokontrolera. Odabirom tipke „Digitalni ulazi/izlazi“ PORTA i PORTE se postavljaju kao digitalni ulazi/izlazi pomoću ADCON1 registra. Nakon što se odabere ulazni port i izlazni portovi, potrebno je odabrati kašnjenje (*delay*) u milisekundama. Na taj način će se svakih toliko milisekundi mijenjati vrijednost svakog odabranog TRIS registra (s 255 na 0 i obrnuto). Tipkom „Generiraj kod“ generira se kod za unos u mikroPaskal. Kao i u prethodnom slučaju i ovdje se prilikom odabira uređaja učitavaju vrijednosti iz *datasheet-a*. Ovdje se koriste vrijednosti registra TRIS za svaki od portova. Kod za učitavanje jednog TRIS registra je sljedeći:

```
For i = 0 To UBound(lines) - LBound(lines)
                If lines(i).StartsWith("trisA_as_input") Then
                    For j = 1 To Len(lines(i))
                        If (Mid(lines(i), j, 1) = "=") Then
                            For k = j + 1 To Len(lines(i))
                                If (IsNumeric(Mid(lines(i), k, 1))) = True Then
                                    temp_trisA_in = temp_trisA_in & Mid(lines(i), k,
1)
                                End If
                            Next
                            Exit For
                        End If
                    Next
                    Exit For
                End If
            Next

trisA_in = CDbl(temp_trisA_in)
```

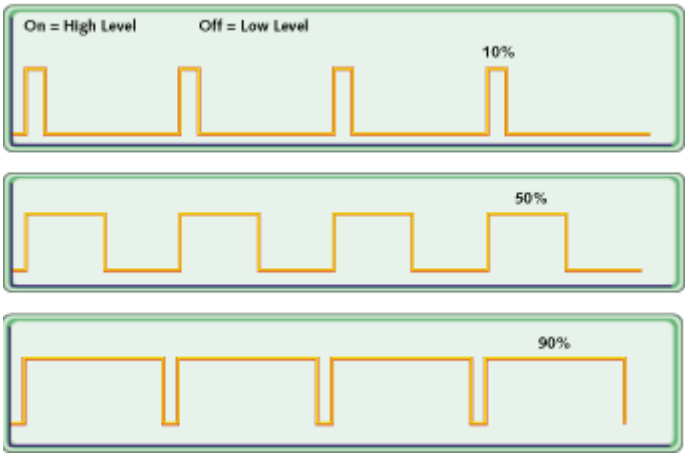
Kako se portovi koji su definirani kao ulazni ne mogu koristiti kao izlazi nužno je onemogućiti izbor onih izlaznih portova koji su već odabrani kao ulazni. Za to nam služi sljedeći dio koda:

```
    Private Sub ComboBoxUlazniPort_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBoxUlazniPort.SelectedIndexChanged

        If ToolStripStatusLabelPIC.Text = "" Then
            MsgBox("Odaberite PIC!", 64, "Error!")
            ComboBoxSekunde.SelectedItem = ""
            Exit Sub
        End If

            trisA = 0
            trisB = 0
            trisC = 0
            trisD = 0
            trisE = 0

            Select Case ComboBoxUlazniPort.SelectedIndex
                Case 0
                trisA = trisA_in
                Case 1
                trisB = trisB_in
                Case 2
                trisC = trisC_in
                Case 3
                trisD = trisD_in
                Case 4
                trisE = trisE_in
        End Select

            MsgBox("Svi ostali portovi su postavljeni kao izlazni!", 64, "Operation
Completed")


        CheckBoxA.Enabled = True
        CheckBoxB.Enabled = True
        CheckBoxC.Enabled = True
        CheckBoxD.Enabled = True
        CheckBoxE.Enabled = True

        Select Case ComboBoxUlazniPort.SelectedIndex
            Case 0
                CheckBoxA.Enabled = False
            Case 1
                CheckBoxB.Enabled = False
            Case 2
                CheckBoxC.Enabled = False
            Case 3
                CheckBoxD.Enabled = False
            Case 4
                CheckBoxE.Enabled = False
        End Select
    End Sub
```

4.4. PWM

4.4.1. Uvod

Pulsno širinska modulacija (eng. *Pulse Width Modulation* – PWM) je tehnika upravljanja analognim krugovima digitalnim izlazima mikrokontrolera. PWM ima širok spektar primjene od mjerenja i komunikacije pa sve do upravljanja i pretvorbe snage.
U suštini PWM-om upravljamo digitalnim signalom iz kojeg dobivamo odgovarajuću analogni signal. PWM signal je digitalan budući da se u određenom vremensko periodu napajanje potpuno isključuje ili uključuje. Naponski ili strujni izvor se dovodi na analogni potrošač u serija uključenih i isključenih impulsa. Korištenjem dovoljno uske periode, moguće je postići bilo koju analognu vrijednost.



Slika 4.9 Signal različitih radnih ciklusa [S26]

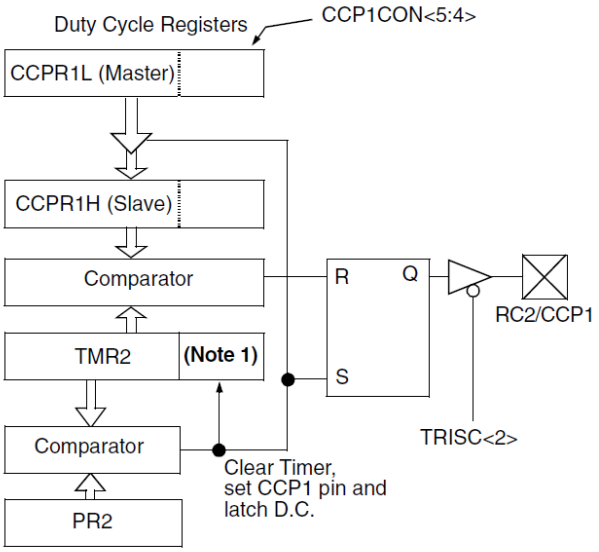
Na slici 4.9 prikazan je signal izlaza PWM-a na 10 % radnog ciklusa (eng. *duty cycle*). To jest, signal je uključen na 10 % periode, a isključen ostalih 90 %. Pri tome mu faktor popunjenosti (omjer uključenog vremena i periode) iznosi 0.1. Na drugoj i trećoj slici (Slika 4.9b i 4.9c) faktor popunjenosti iznosi 0.5 i 0.9. Ova tri signala predstavljaju tri vrijednost analognog signala, na 10 %, 50 % i 90 % ukupne snage. Ukoliko bi napajanje iznosilo 9 V, a radni ciklus PWM bio 10 %, to bi značilo da rezultirajući analogni signal iznosi 0.9 V.
Jedan od načina primjene PWM signala je upravljanje prekidačem za uključivanje i isključivanje žarulje. U periodu u kojem je prekidač uključen žarulja svijetli, a u periodu kada je prekidač isključen žarulja ne svijetli. Kako žarulja ne bi treperila potrebna je velika modulacijska frekvencija. Ta frekvencija obično iznosi od 1 kHZ do 200 kHz.

4.4.2. CCP MODUL mikrokontrolera

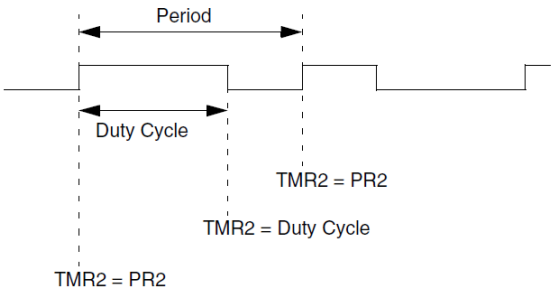
Za rad s PWM-om koristimo CCP (*Capture/Compare/PWM*) modul mikrokontrolera u PWM načinu rada. PIC18F448 ima dva takva modula – CCP1 i ECCP1 (*Enhanced Capture /Compare/PWM*). Razlika između njih je ta što ECCP ima *Enhanced* PWM funkcionalnost i sposobnost automatskog isključenja.

4.4.2.1. CCP1 u PWM načinu rada

Budući da se pin CCP1 modula nalazi na portu C, bit 2 TRISC registra treba biti postavljen na nulu kako bi koristili CCP1 pin kao izlaz.



Slika 4.10 Blok dijagram [S27]



Slika 4.11 Izlazni signal [S28]

- **period PWM-a**

Izlazni period T je određen pomoću PR2 registra tajmer TMR2, koji služi kao vremenska baza za PWM. Period PWM-a računa se po sljedećoj formuli:

$$PWM\ Period = (PR2 + 1) \cdot 4 \cdot \text{Tosc} \cdot (\text{Vrijednost preskalera TMR2}) \qquad (4.1.)$$

Napomena: u poglavlju „Tajmeri“ tajmer TMR2 je detaljno obrađen.
U jednadžbi 4.1, PR2 predstavlja vrijednost PR2 registra, to jest izlazni period, *Tosc* je period internog oscilatora tajmera TMR2 (množi se s četiri zato jer se frekvencija oscilatora dijeli s četiri), a vrijednost preskalera se postavlja bitovima T2CKPS2:T2CKPS0 kontrolnog registra T2CON tajmera TMR2. Postskaler tajmera se ne koristi.

- **radni ciklus PWM-a**

Radni ciklus PWM-a se određuje s osam bitova iz CCPRL1 registra (MSB – *Most Significant Bit*) i četvrtim i petim bitom CCP1CON registra (LSB – *Least Significant Bit*). Ta 10 bitna vrijednost je predstavljena s CCPRL1:CCPCON<4:5>, a formula za računanje radnog ciklusa je:

$$Duty\ Cycle = (\text{CCPRL1:CCPCON} < 5:4 >) \cdot \text{Tosc} \cdot (\text{Vrijednost preskalera TMR2}) \qquad (4.2)$$

- **rezolucija PWM-a (bitovi)**

PWM signal nije ništa drugo nego sekvenca pulseva s različitim radnim ciklusima. Za jednu određenu frekvenciju (broj pulseva u sekundi) postoji određen broj kombinacija radnih ciklusa. Taj broj se naziva rezolucija i mjeri se u bitovima. Na primjer, 10 bitna rezolucija rezultira s 2^{10} (1024) diskretnih radnih ciklusa, dok 8 bitna s 256. Rezolucija je određena s PR2 registrom, a računa se po formuli:

$$PWM\ Resolution = \frac{\log\left(\frac{Fosc}{Fpwm}\right)}{log(2)} \qquad (4.3)$$

4.4.2.2. Postavke za rad s PWM-om

- postavljanje PWM perioda u PR2 registar
- postavljanje radnog ciklusa PWM-a zapisujući u CCPR1L registar i bitove 4 i 5 CCP1CON registra
- postavljanje CCP1 pina kao izlaznog upisivanjem nule na drugi bit TRISC registra
- postavljanje vrijednosti preskalera TMR2 i uključivanjem Timer2
- konfiguriranje CCP1 modula za PWM način rada

PWM Frequency	2.44 kHz	9.76 kHz	39.06 kHz	156.3 kHz	312.5 kHz	416.6 kHz
Timer Prescaler (1, 4, 16)	16	4	1	1	1	1
PR2 Value	0FFh	0FFh	0FFh	3Fh	1Fh	17h
Maximum Resolution (bits)	10	10	10	8	7	5.5

Tablica 4.8 PWM frekvencije i rezolucije za *Fosc* = 40Mhz [T10]

4.4.3. PWM u mikroPaskalu

mikroPaskal sadrži knjižnicu olakšava rad s PWM modulom. Umjesto prethodno opisanog postupka, za rad s PWM u mikroPaskalu dovoljne su četiri funkcije:

- **PWM_Init()**

Inicijalizira PWM modul s radnim ciklusom 0. U zagrade se unosi željena frekvencija koja mora biti konstanta, a ne varijabla. Ta frekvencija ne smije biti veća od *Fosc*/4. Proračun frekvencije PWM vrši kompajler.

- **PWM_Set_Duty()**

Postavlja radni ciklus PWM-a. Parametar koji unosimo u zagrade je broj od 0 do 255, pri čemu je 0 0 %, 127 50 %, a 255 100 % po formuli:

$$Posto * 255/100 \qquad (5.4)$$

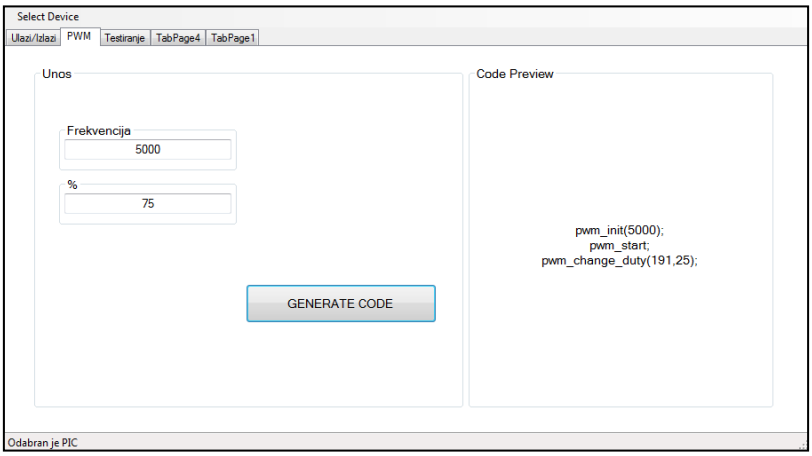
- **PWM_Start**

Pokreće PWM

- **PWM_Stop**

Zaustavlja PWM

4.4.4. Razvijeni računalni program – PWM



Slika 4.12 Korisničko sučelje

Nakon što odaberemo karticu PWM program nam omogućuje unos frekvencije PWM-a i radnog ciklusa. Nakon unosa pritiskom na tipku „Generiraj kod“ generira se kod, a s desne strane se prikazuje pregled koda. Kod je:

```
Private Sub ButtonGenKod2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ButtonGenKod2.Click

        If ToolStripStatusLabelPIC.Text = "" Then
            MsgBox("Odaberite PIC!", 64, "Error!")
            Exit Sub
        End If

        If TextBoxFrekv.Text = "" Or TextBoxPosto.Text = "" Or Not
IsNumeric(TextBoxFrekv.Text) And IsNumeric(TextBoxPosto.Text) Then
            MsgBox("Unesite numeričke vrijednosti!", 64, "Error!")
            Exit Sub
        End If

        Dim frek, posto, pwm As Double
        Dim temp As String = ""
        Dim maxfrek As Double
        Dim br_dot_frek As Integer = 0
```

```
        frek = Replace(TextBoxFrekv.Text, ".", ",")
        posto = Replace(TextBoxPosto.Text, ".", ",")

        Dim lines() As String = IO.File.ReadAllLines(datasheet.FileName)

        For i = 0 To UBound(lines) - LBound(lines)
            If lines(i).StartsWith("Max_PWM_frequency") Then
                For j = 1 To Len(lines(i))
                    If (Mid(lines(i), j, 1) = "=") Then
                        For k = j + 1 To Len(lines(i))
                            If br_dot_frek = 0 Then
                                If (Mid(lines(i), k, 1) = ",") Then
                                    temp = temp & Mid(lines(i), k, 1)
                                    br_dot_frek = br_dot_frek + 1
                                End If
                            End If
                            If (IsNumeric(Mid(lines(i), k, 1))) = True Then
                                temp = temp & Mid(lines(i), k, 1)
                            End If
                            If (Mid(lines(i), k, 1)) = "k" Then
                                temp = temp & "e3"
                            End If
                            If (Mid(lines(i), k, 1)) = "M" Then
                                temp = temp & "e6"
                            End If
                        Next
                        Exit For
                    End If
                Next
                Exit For
            End If
        Next

        If temp = "" Then
            MsgBox("Greška pri čitanju datoteke! Provjerite datasheet!", 64, "Error")
            Exit Sub
        End If

        maxfrek = CDbl(temp)

        If posto > 100 Or posto < 0 Then
            MsgBox("Unesite ispravnu postotnu vrijednost!", 64, "Error!")
        End If


        If frek > maxfrek Then
            MsgBox("Unesite frekvenciju prema datasheet-u!", 64, "Error!")
        End If

        If posto <= 100 And posto >= 0 And frek <= maxfrek Then
            browser.RootFolder = Environment.SpecialFolder.Desktop
            browser.Description = "Odaberite lokaciju za spremanje koda!"


            If browser.ShowDialog = DialogResult.OK Then
                frek = Replace(TextBoxFrekv.Text, ".", ",")
                posto = Replace(TextBoxPosto.Text, ".", ",")

                pwm = posto * 255 / 100
                pwm_string = Replace(pwm, ",", ".")

                kod = browser.SelectedPath & "\CodePWM.txt"
```

```
                Dim pisac As New System.IO.StreamWriter(kod)
                pisac.Write("pwm_init(" & frek & ");" & vbNewLine)
                pisac.Write("pwm_start;" & vbNewLine)
                pisac.Write("pwm_change_duty(" & pwm_string & ");")
                pisac.Close()
                MsgBox("Kod je spremljen u " & kod & "!", 64, "Operation Completed!")

                fileReader = My.Computer.FileSystem.ReadAllText(kod)
                LabelCodePrevPWM.Text = fileReader

            End If
        End If

    End Sub
```

Kako bi mogli uspoređivati unesene vrijednosti sa maksimalni vrijednostima upisanima u tekstualnu datoteku PIC mikrokontrolera, potrebno je te vrijednosti „izvući“ iz nje. Zbog toga u polje lines spremamo sve linije datoteke, svaku u svoje polje. Nakon toga svaku liniju koda provjeravamo počinje „Max_PWM_frequency“ što označava maksimalnu PWM frekvenciju. Kada je to polje pronađeno, program provjerava sve znakove linije funkcijom *Mid* (string, redni broj znaka, željena duljina), a u varijablu *temp* sprema brojeve, decimalan zarez (samo prvi) i slova k (10^3) i M (10^6) koja dodaje u odgovarajućem obliku na kraj *stringa*. Nakon uspješnog očitanja vrijednosti u varijabla *temp* se pretvara u tip *Double* i sprema se u novu varijablu *maxfrek*. Ukoliko je unesena frekvencija manja od maksimalne (*maxfrek*), a željeni postotci ispravni, računa se radni ciklus PWM po formuli 5.4. Budući da mikroPascal koristi decimalne točke umjesto zareza (za razliku od VB-a koji točke zanemaruje), nužno je funkcijom *Replace* zamijeniti zarez točkom.

Za razliku od taba Ulazi/Izlazi, ovaj *tab* sadrži i *code preview* (pregled koda) koji prikazuje kod koji će se generirati, a koji se mijenja svakom promjenom *textbox-eva*. Kod za promjenu vrijednosti frekvencije je:

```
    Private Sub TextBoxFrekv_TextChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TextBoxFrekv.TextChanged

        Dim temp As String = ""
        Dim frek As String
        Dim temp_frek As Double
        Dim br_dot_frek As Integer = 0


        For i = 1 To Len(TextBoxFrekv.Text)
            If br_dot_frek = 0 Then
                If (Mid(TextBoxFrekv.Text, i, 1) = ".") Then
                    temp = temp & Mid(TextBoxFrekv.Text, i, 1)
                    br_dot_frek = br_dot_frek + 1
```

```
                End If
                If (Mid(TextBoxFrekv.Text, i, 1) = ",") Then
                    temp = temp & Mid(TextBoxFrekv.Text, i, 1)
                    br_dot_frek = br_dot_frek + 1
                End If
            End If
            If (IsNumeric(Mid(TextBoxFrekv.Text, i, 1))) = True Then
                temp = temp & Mid(TextBoxFrekv.Text, i, 1)
            End If
        Next

        temp.Replace(".", ",")

        If temp <> "" Then
            temp_frek = CDbl(temp)
        End If

        If temp_frek > 100000000 Then
            TextBoxFrekv.Text = ""
            br_dot_frek = 0
            Exit Sub
        End If

        frek = Replace(temp_frek, ",", ".")

        If LabelCodePrevPWM.Text.Contains("pwm_init(") Then
            LabelCodePrevPWM.Text = "pwm_init(" & frek & ");" & vbNewLine &
"pwm_start;" & vbNewLine & "pwm_change_duty(" & pwm_string & ");" & vbNewLine
        Else : LabelCodePrevPWM.Text = "pwm_init(" & frek & ");" & vbNewLine &
"pwm_start;" & vbNewLine
        End If

    End Sub
```

Kao i kod čitanja iz datoteke, funkcijom *Mid* provjeravamo znakove unesene u *textbox*. Dobivenu varijablu pretvaramo u tip *Double*, a ukoliko je veća od maksimalne postavlja se na nulu (*textbox* se prazni). Ako je ispravna stavlja se u varijablu *frek* i mijenja joj se decimalan zarez točkom. Nakon toga, u ovisnosti od toga je li unesen željeni radni ciklus, ispisuje se pregled koda.

Konačni kod za unos u programski alat mikroPascal izgleda:

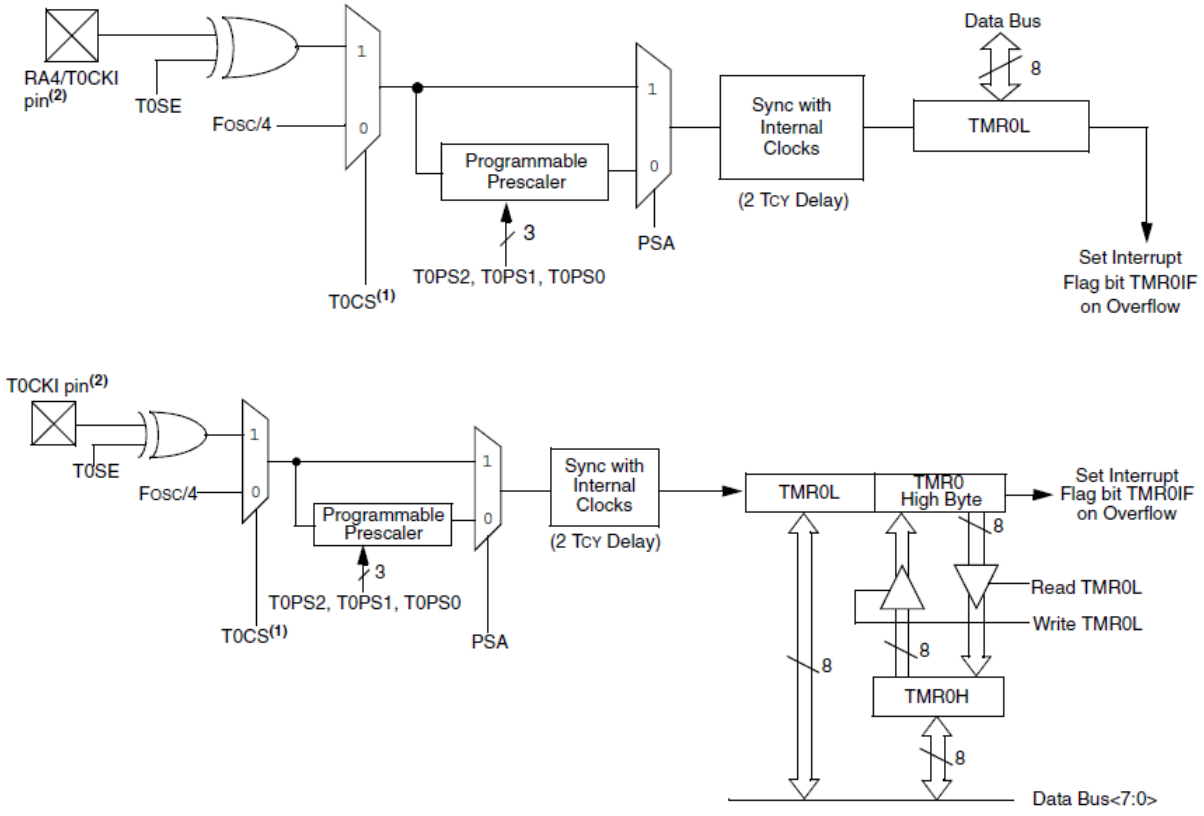
```
pwm_init(5000);
pwm_start;
pwm_change_duty(127.5);
```

4.5. Tajmer

PIC 18F448 sadrži četiri tajmera – Timer0, Timer1, Timer3 i Timer4.

4.5.1. TIMER0

Timer0 sadrži softverski podesiv 8-bitni ili 16-bitni registar TMR0, moguće mu je pridruživanje 8-bitnog preskalera, može koristiti interni ili eksterni generator takta, a za eksterni moguć je odabir okida na padajući ili rastući brid. Moguć je prekid na preljev (eng. *overflow*).



Slika 4.13 Blok dijagram tajmera TIMER0 u 8-bitnom i 16-bitnom načinu rada [S29]

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

Slika 4.14 T0CON registar [S30]

Na slici 4.14 prikazan je kontrolni registar tajmera Timer0 – T0CON (*Timer0 Control*). Sedmim bitom – TMR0ON uključujemo (1) ili isključujemo (0) Timer0. Ukoliko bitu T8BIT pridružimo jedinicu, Timer0 će biti konfiguriran kao 8-bitni

brojač/tajmer. U suprotnom je konfiguriran kao 16-bitni. U 8-bitnom načinu rada tajmer broji od 0 do 255 (FF), a u 16-bitnom od 0 do 65536 (FFFF). Kada vrijednost modula Timer0 prijeđe iz 255 u 0 za 8-bitni način rada ili iz 65536 u 0 za 16-bitni, uključuje se prekid na preljev. Taj preljev postavlja bit TMR0IF INTCON registra na jedinicu, a da bi se prekid ponovno uključio potrebno je taj bit postaviti na nulu. Postavljanjem bita TMR0IE na nulu, prekid se ne događa. Da bi Timer0 koristili kao tajmer potrebno je postaviti peti, T0CS, bit na logičku nulu. Tada koristimo unutarnji generator takta čiju frekvenciju dijelimo sa četiri (jedan ciklus instrukcija jednak je četiri ciklusa generatora takta). Ukoliko je frekvencija generatora takta 40 MHz, tajmeru Timer0 u 8-bitnom načinu rada trebati će 25,6 μs za brojenje od 0 do 0 (jedan ciklus) po formuli

$$t = \frac{256 - TMR0}{\frac{Fosc}{4}} \qquad (7.1)$$

gdje je preskaler vrijednost preskalera (1, 2, 4, 8, 16, 32, 64, 128, 256), TMR0 vrijednost koja se nalazi u registru TMR0, a *Fosc* frekvencija oscilatora. Vidljivo je da se povećanjem vrijednosti TMR0 smanjuje vrijeme za brojenje od 0 do 0. U 16-bitnom načinu rada broj 256 mijenjamo s 65536. Kada je bit T0CS postavljen na logičku jedinicu, Timer0 radi kao brojač. U tom načinu rada vrijednost Timer0 će se povećavati na svaki rastući ili padajući brid signala s pina RA4/T0CK1 ovisno o vrijednosti bita T0SE. Povećanje na rastući brid postižemo postavljanjem bita T0SE na nulu, a postavljanjem na nulu postižemo povećanje na padajući brid. Da bi mogli koristiti vanjski generator takta važno je zadovoljiti određene zahtjeve. Ti zahtjevi osiguravaju sinkronizaciju vanjskog i unutarnjeg generatora takta. Također postoji i kašnjenje u povećanju vrijednosti modula Timer0 nakon sinkronizacije. Četvrtim bitom, PSA, određujemo hoće li tajmer koristiti preskaler ili ne. Kao što je vidljivo na blok dijagramu modula Timer0, ukoliko je PSA bit 1 tada se ne koristi preskaler, već se koristi WDT čiji je omjer 1:1. Ako je PSA bit 0 tada frekvenciju signal dijelimo preskalerom. Prva tri bita – 0, 1, 2 služe za odabir vrijednosti preskalera, a vrijedi:

PS2, PS1, PS0	*TMR0 RATE
000	1:2
001	1:4
010	1:8
011	1:16
100	1:32
101	1:64
110	1:128
111	1:256

Tablica 4.9 Vrijednosti preskalera [T10]

Korištenjem preskalera povećavamo vrijeme koje tajmer može izbrojati prije preljeva, po formula 7.1 postaje:

$$t = \frac{prescaler*(256-TMR0)}{\frac{Fosc}{4}} \qquad (7.2)$$

ili za 16-bitni način rada:

$$t = \frac{prescaler*(65536-TMR0)}{\frac{Fosc}{4}} \qquad (7.3)$$

4.5.2. INTCON registar

7	6	5	4	3	2	1	0
GIE	PEIE	T0IE	INTE	RABIE	T0IF	INTF	RABIF

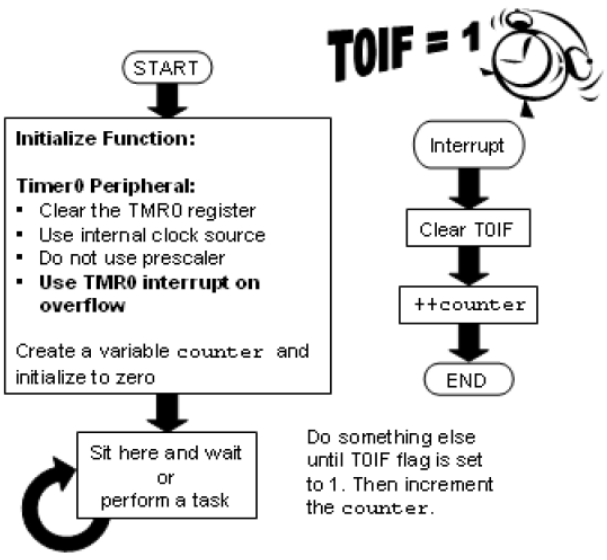
Slika 4.15 INTCON (Interrupt Control) registar [S31]

GIE : *Global Interrupt Enable*
T0IE : *Timer0 Overflow Interrupt bit*
TOIF: *Timer0 Overfow Interrupt flag bit*

Bitovi 2, 5 i 7 bitni su za rad s tajmerima. Sedmim, GIE, bitom omogućujemo (1) ili onemogućujemo (0) prekide. kada dođe do prekida njegova se vrijednost postavlja

na nulu kako bi bili sigurni da se neće neki dugi prekid javiti. Postavljanjem bita TMR0IE na logičku jedinicu omogućujemo prekid na preljev, a logičkom jedinicom onemogućujemo. Drugi bit signalizira je li došlo do preljeva. Ukoliko dođe do preljeva njegova vrijednost iznosi 1 te ga je potrebno postaviti na 0 kako bi ponovno mogli koristiti prekid. Iz formule 7.2 za vrijeme 8-bitnog tajmera moguće je izračunati maksimalno moguće vrijeme koje tajmer može izmjeriti prije preljeva. Za *Fosc* = 40 MHz ono iznosi 6,5536 ms. Ukoliko želimo postići nekoliko puta veće vrijeme od maksimalnog nužno je koristiti prekide. Kada dođe do preljeva bit T0IF postaje 1 pokreće se rutina mikrokontrolera koja se naziva ISR (*Interrupt Service Routine*). Kada se ISR pokrene, mikrokontroler prestaje s trenutnim zadatkom te skače na ISR kako bi odradio prekid. ISR rutina je u mikroPaskalu prikazana kao procedura *interrupt*. Ako u nju brojač (eng. *counter*) čija će se vrijednost povećavati svakim prekidom, moguće je povećati maksimalno vrijeme tajmera za faktor brojača po formuli:

$$t = \frac{prescaler*(256-TMR0)*counter}{\frac{Fosc}{4}} \qquad (7.4)$$



Slika 4.16 Algoritam za povećanje varijable *counter* pomoću prekida [S32]

Programsko rješenje za PIC16F887 u programskom alatu mikroPascalu je

```
var counter : word;

procedure Interrupt();
  begin
    Inc(counter);                // Povećanje vrijednosti varijable counter za jedan svakim
prekidom
    TMR0   := 96;
    INTCON := 0x20;           // Postavljanje T0IE na jedan, a T0IF na nula
  end;

begin
  OPTION_REG := 0x84;       // Pridruživanje preskalera
  TMR0  := 96;              // Postavljanje početne vrijednosti TMR0
  INTCON := 0xA0;           // Omogućavanje prekida
  counter := 0;             // Inicijalizacija brojača

  while TRUE do
    begin
      if (counter = 500) then
        begin
                                  // Željena radnja
          counter := 0;           // Resetiranje varijable counter
        end;
    end;
end.
```

U gornjem primjeru vidimo da ukoliko dođe do prekida, varijabla *counter* se povećava za jedan u posebnoj proceduri *interrupt* (ISR). U glavnom programu definirana je petlja koja omogućuje povećanje varijable *counter* do 500, a tada se izvršava željena radnja. Na taj način vrijeme dobiveno po formuli množimo s 500, a time je moguće dobiti bilo koje željeno vrijeme tajmera.

Kako bi pravilno koristili Timer0 potrebno je:

- Odabrati način rada tajmera T0CS bitom;
- Ukoliko je potrebno koristiti preskaler, nužno je postaviti PSA bit na nulu te odabrati vrijednost preskaler nultim, prvim i trećim bitom kontrolnog registra tajmera;
- Ako se koriste prekidi potrebno je postaviti bitove GIE i TMR0IE INTCON registra na jedinicu.

4.5.3. TIMER1

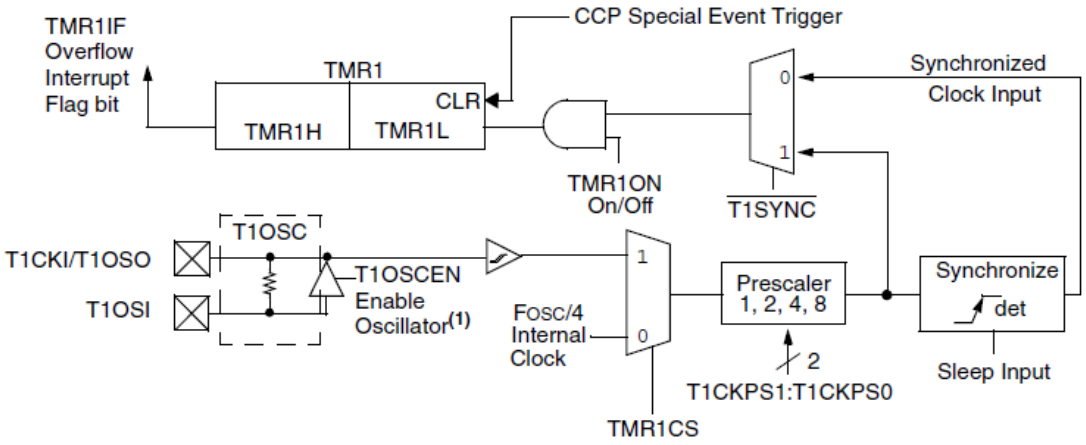
Timer1 ima 16-bitni TMR1 registar koji se sastoji od dva registra (TMR1L i TMR2H). Broji do 65536 u jednom ciklusu prije nego dođe do nule. Kao i Timer0, moguć je prekid na preljev te odabir internog ili vanjskog generatora takta. Za razliku od tajmera Timer0, tajmeru Timer1 moguće je pridružiti 8-bitni preskaler s vrijednostima 1:1, 1:2, 1:4 i 1:8.

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

Slika 4.17 T1CON registar [S33]

Bitom RD16 određujemo hoće li se čitanje i pisanje u Timer1 vršiti u jednoj 16-bitnoj operaciji (1) ili dvije 8-bitne (0). Bitovi T1CKPS1 i T1CKPS0 služe za postavljanje preskaler*a*, a bitom T1OSCEN uključujemo ili isključujemo oscilator. Prvi bit, TMR1CS služi za odabir eksternog (1) ili internog generatora takta (0). Ukoliko je odabran eksterni, postavljanjem logičke nule na treći, komplementirani, bit sinkroniziramo eksterni generator takta. Bitom TMR1ON uključujemo ili isključujemo Timer1. Timer1 može raditi u tri načina rada – kao tajmer, asinkroni ili sinkroni brojač. Ukoliko je nula na bitu TMR1CS, vrijednost tajmera se povećava svaki instrukcijski ciklus. Dovođenjem logičke jedinice na TMR1CS, vrijednost mu se povećava na svaki rastući brid signala vanjskog generatora takta

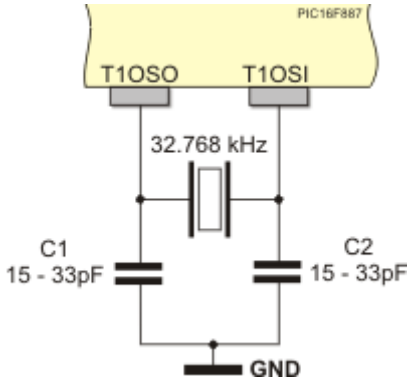
Ukoliko je odabran oscilator (T1OSCEN = 1) pinovi RC1/T1OSI i RC0/T1OSO/T1CKI postaju ulazi, a vrijednost nultog i prvog bita registra TRISC se zanemaruje.



Slika 4.18 Blok dijagram tajmera Timer1 u 8-bitnom načinu rada [S34]

4.5.3.1. Oscilator tajmera Timer1

Kristalni oscilator spaja se između pinova T1OSI (*Timer1 Oscillator Input*) i TIOSO (*Timer1 Oscillator Input*). Uključuje se postavljanjem bita TOSCEN na logičku jedinicu. Dodatni krug je namijenjen za frekvencije do 50 kHz, odnosno za kristal frekvencije 32 kHz. Oscilator može raditi i za vrijeme s*leep* moda (najmanja potrošnja energije).

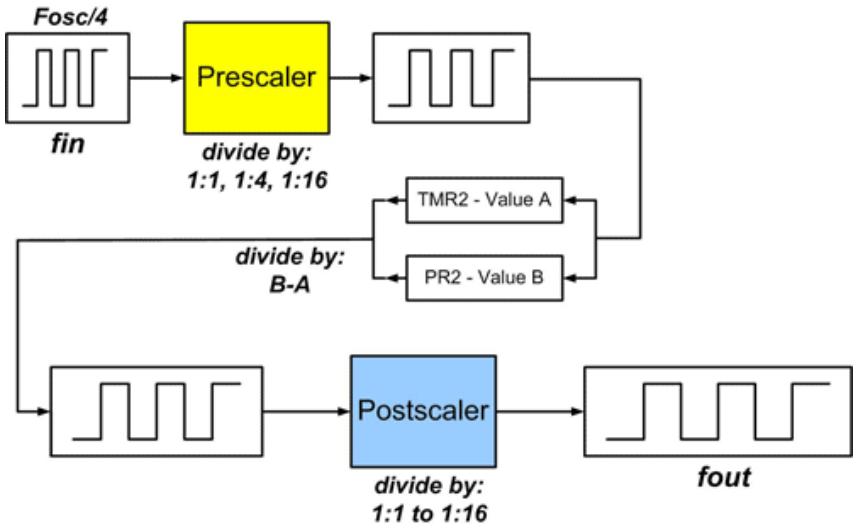


Slika 4.19 Oscilator tajmera Timer1 [S35]

4.5.4. TIMER2

Ovaj tajmer se sastoji od dva 8-bitna registra – TMR2 i PR2. TMR se prilikom resetiranja postavlja na 0 i služi za unos početne vrijednosti tajmera. Registar PR2 se resetiranjem postavlja na 255 i služi za unos konačne vrijednosti do koje tajmer broji. TMR2 se povećava dok ne dostigne vrijednost registra PR2, a nakon toga se resetira na nulu i postavlja vrijednost TMR2IF na jedinicu (prekid). Za razliku od tajmera Timer0 i Timer1, Timer2 se sastoji od preskalera i preskalera. Preskaler dijeli frekvenciju signala generatora takta prije brojanja, postskaler frekvenciju signala

nakon što on izađe iz komparatora (uspoređuje vrijednost registra TMR2 s vrijednošću registra PR2).



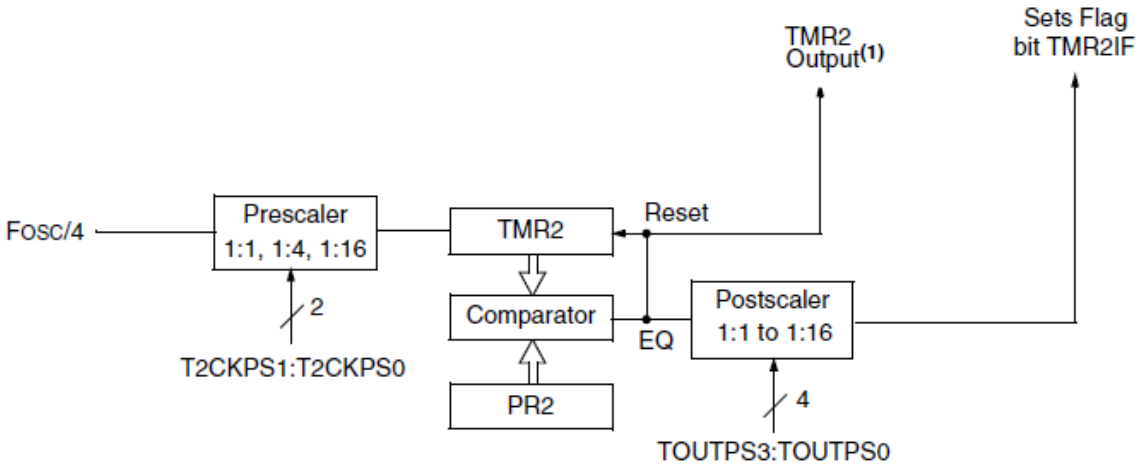
Slika 4.20 Preskaler i postskaler [S36]

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

Slika 4.21 T2CON registar [S37]

Nultim i prvim bitom određujemo vrijednost preskalera (1:1, 1:4 ili 1:16), a trećim, četvrtim, petim i šestim vrijednost postskalera (1:1, 1:2, 1:3, 1:4, …, 1:16).

Drugi bit nam služi za uključivanje ili isključivanje tajmera kako bi smanjili potrošnju energije).



Slika 4.22 Blok dijagram tajmera Timer2 [S38]

Formula za računanje vremena tajmera je:

$$t = \frac{prescaler*(PR2-TMR2)*postscaler}{\frac{Fosc}{4}} \qquad (7.5)$$

4.5.5. TIMER3

Ovaj tajmer je jednak tajmeru Timer1 osim što se koristi kao generator takta za CCP1 i ECCP1 modul.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T3ECCP1	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON
bit 7							bit 0

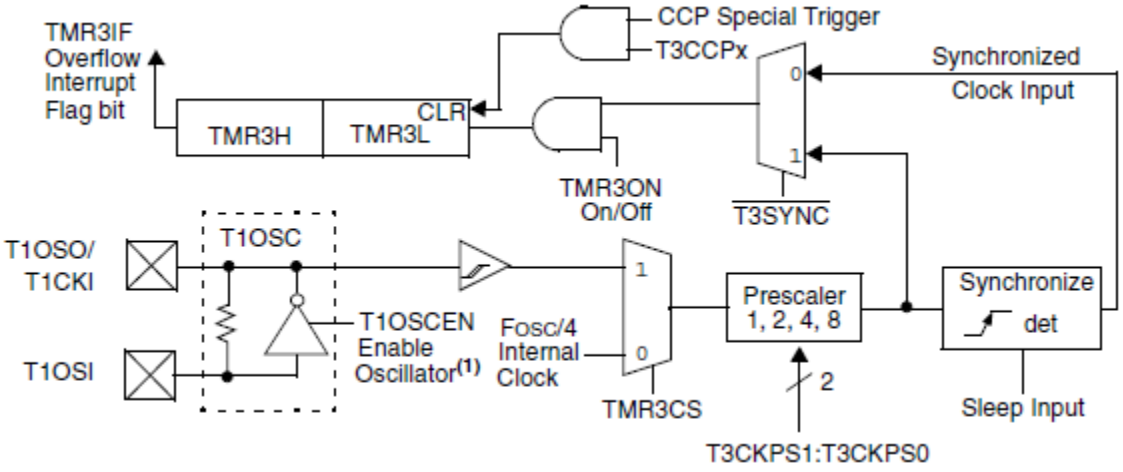
Slika 4.23 T2CON registar [S39]

Kontrolni registar je jednak onome tajmer Timer1 s razlikom što ovaj registar koristi šesti bit, a treći bit mu ne služi za uključivanje i isključivanje. Kombinacijom trećeg (T3CCP1) i šestog bita (R3ECCP1) odabiremo hoće li Timer1 ili Timer3 biti izvor za CCP1 ili ECCP1 modul.

T3ECCP1	T3CCP1	
1	x	Timer3 je izvor generatora takta za CCP1 i ECCP1 modul
0	1	Timer1 je izvor generatora takta za CCP1, a Timer3 za ECCP1
0	0	Timer1 je izvor generatora takta za CCP1 i ECCP1

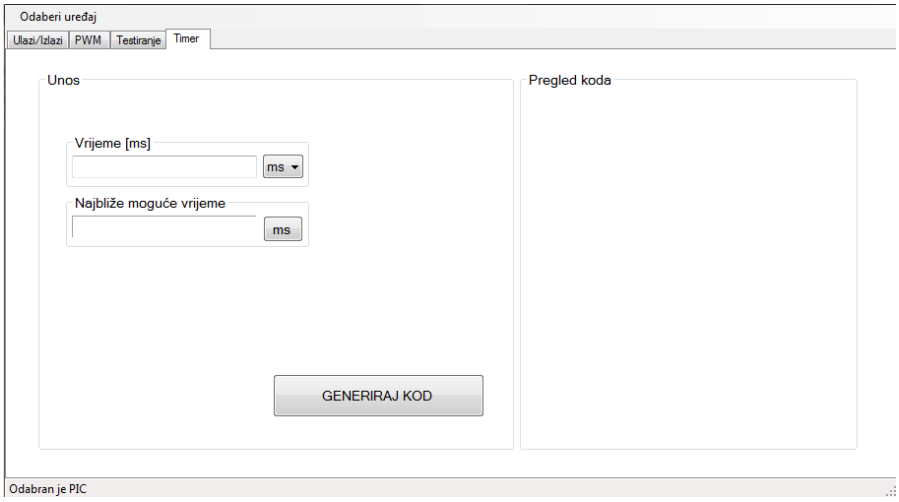
Tablica 4.10 Konfiguracija T3CON registra

Timer3 može raditi u tri načina rada – kao tajmer, brojač ili kao asinkroni brojač. Način rada se odabire bitom TMR3CS.



Slika 4.24 Blok dijagram tajmera Timer3 [S40]

4.5.6. Razvijeni računalni program – Timer



Slika 4.25 Korisničko sučelje

Odabirom taba „Timer“, pojavljuje se korisničko sučelje koje omogućuje unos željenog vremena koje će tajmer izbrojiti prije nego dođe do prekida. Unos vremena moguć je u milisekundama, mikrosekundama i nanosekundama. U donjem tekstnom okviru pojavljuje se najbliža vrijednost željenom vremenu koju je moguće ostvariti. Kao što je bio slučaj s unosom u kartici PWM, program i ovdje provjerava je li unesena vrijednost numerička vrijednost, prihvaća samo jedan znak za decimalna mjesta (točku ili zarez) te omogućuje upis do određenih brojeva. Nakon što se unese željena vrijednost i odabere vremenska jedinica, u dijelu „Pregled koda“ prikazuje se kod koji se može generirati ukoliko su sve vrijednosti u skladu s *datasheet-om*. Budući da je kod za dobivanje koda za pregled i koda koji će se generirati praktički

jednak, biti će prikazan samo kod za generiranje koda. Pritiskom na tipku „Generiraj kod“ program kao i u prethodnim slučajevima provjerava je li odabran mikrokontroler kako bi mogli očitavati vrijednosti iz *datasheet-a*. Ukoliko je on odabran program očitava relevantne podatke iz *datasheet-a*, a to su za ovaj slučaj frekvencija *Fosc* i maksimalan period tajmera Timer0 koji se koristi. Kod za očitavanje je sljedeći:

```
            Dim lines() As String = IO.File.ReadAllLines(datasheet.FileName)

            temp1 = ""
            temp2 = ""
            temp3 = ""
            temp_trisA_in = ""
            temp_trisB_in = ""
            temp_trisC_in = ""
            temp_trisD_in = ""
            temp_trisE_in = ""

            For i = 0 To UBound(lines) - LBound(lines)
                If lines(i).StartsWith("Max_timer_period_8bit") Then
                    For j = 1 To Len(lines(i))
                        If (Mid(lines(i), j, 1) = "=") Then
                            For k = j + 1 To Len(lines(i))
                                If (Mid(lines(i), k, 1) = ",") Then
                                    temp1 = temp1 & Mid(lines(i), k, 1)
                                End If
                                If (IsNumeric(Mid(lines(i), k, 1))) = True Then
                                    temp1 = temp1 & Mid(lines(i), k, 1)
                                End If
                                If (Mid(lines(i), k, 1)) = "m" Then
                                    temp1 = temp1 & "e-3"
                                End If
                                If (Mid(lines(i), k, 1)) = "u" Then
                                    temp1 = temp1 & "e-6"
                                End If
                                If (Mid(lines(i), k, 1)) = "n" Then
                                    temp1 = temp1 & "e-9"
                                End If
                            Next
                            Exit For
                        End If
                    Next
                    Exit For
                End If
            Next

            For i = 0 To UBound(lines) - LBound(lines)
                If lines(i).StartsWith("Max_timer_period_16bit") Then
                    For j = 1 To Len(lines(i))
                        If (Mid(lines(i), j, 1) = "=") Then
                            For k = j + 1 To Len(lines(i))
                                If (Mid(lines(i), k, 1) = ",") Then
                                    temp2 = temp2 & Mid(lines(i), k, 1)
                                End If
                                If (IsNumeric(Mid(lines(i), k, 1))) = True Then
                                    temp2 = temp2 & Mid(lines(i), k, 1)
                                End If
                                If (Mid(lines(i), k, 1)) = "m" Then
```

```
                                temp2 = temp2 & "e-3"
                            End If
                            If (Mid(lines(i), k, 1)) = "u" Then
                                temp2 = temp2 & "e-6"
                            End If
                            If (Mid(lines(i), k, 1)) = "n" Then
                                temp2 = temp2 & "e-9"
                            End If
                        Next
                        Exit For
                    End If
                Next
                Exit For
            End If
        Next
```

Prvo se deklarira varijabla *lines* kao polje, a u koju se spremaju sve linije *datasheet-a*, svaka u svoje polje. Budući da nama treba samo ona linija koja počinje s „Max_timer_period“ jer se u njoj nalazi maksimalan period tajmera, postavlja se IF naredba koja provjera počinje li linija s „Max_timer_period“. Ako postoji, program ulazi u novi FOR petlju koja provjerava svaki element linije, odnosno polja. nakon što naiđemo na znak jednakosti, program ulazi u treću petlju. sada se provjerava je li svaki od znakova u polju broj, decimalna točka ili zarez, slovo m (milisekunda), u (mikrosekunda) ili n (nanosekunda) na jednak način na koji se to provjeravalo za u karticama za generiranje PWM i testiranje mikrokontrolera funkcijom *Mid*. Ukoliko je čitanje bilo uspješno dobivene vrijednosti se spremaju u varijable *mintime* (maksimalno vrijeme tajmera) i *Fosc* (frekvencija generatora takta).

```
        Select Case ComboBoxSekunde.SelectedItem
            Case "ns" : exp = 0.000000001
            Case "us" : exp = 0.000001
            Case "ms" : exp = 0.001
        End Select
```

Varijabli *t* se pridružuje vrijeme uneseno u program, funkcijom *Replace* se decimalna točka zamjenjuje decimalnim zarezom, a cijela se vrijednost množi s eksponentom odabrane vremenske jedinice. Razlog zbog kojeg se decimalna točka zamjenjuje zarezom je taj što Visual basic zanemaruje decimalnu točku pa će tako broj 1.27 biti u nekoj varijabli tipa *Double* biti prikazan kao 127. S decimalnim zarezom to nije slučaj. Naredbom *Case* svakom elementu *combobox-a ComboBoxSekunde*, s kojim se bira vremenska jedinica, se pridružuje odgovarajući eksponent.

Nakon što program utvrdi da je unesena vrijednost numerička i u dopuštenim granicama za odabrani mikrokontroler, program računa varijable koje su nužne za generiranje koda za mikroPaskal.

```
For i = 0 To 8

            If t > maxtime_8bit Then
                TMR0(i) = 65536 - Fosc * t / (4 * prescaler(i))

                If TMR0(i) > 0 Then
                    t1(i) = prescaler(i) * (65536 - Math.Round(TMR0(i))) * 4 / Fosc

                    If Math.Abs(t1(i) - t) < min Then
                        min = Math.Abs(t1(i) - t)
                        br = i
                    End If

                End If


            Else : TMR0(i) = 256 - Fosc * t / (4 * prescaler(i))

                If TMR0(i) > 0 Then
                    t1(i) = prescaler(i) * (256 - Math.Round(TMR0(i))) * 4 / Fosc

                    If Math.Abs(t1(i) - t) < min Then
                        min = Math.Abs(t1(i) - t)
                        br = i
                    End If

                End If

            End If
            If i < 8 Then
                prescaler(i + 1) = prescaler(i) * 2
            End If
        Next

        t2 = t1(br) / exp
```

Gornji kod služi za računanje vrijednosti varijable *TMR0* i izbor odgovarajuće vrijednosti preskalera. *TMR0* se u FOR petlji devet puta računa, budući da je tajmeru timer0 mikrokontrolera PIC18F448 moguće pridružiti devet različitih kombinacija preskalera. Ukoliko je uneseno vrijeme veće maksimalnog vremena kojeg je moguće postići u 8-bitnom načinu rada tajmera, vrijednost varijable *TMR0* računa se iz formule (7.2). Nakon što izračunamo *TMR0*, u novu varijablu *t1* računamo vrijeme s zaokruženom varijablom *TMR0* (funkcija *Math.Round*). Računanje te varijable moguće je samo ako je dobivena vrijednost TRM0 pozitivna. Sljedeća IF naredba provjerava je li odstupanje vremena *t1* od traženog vremena manje od prijašnjeg minimalnog odstupanja. U početku programa varijabli *min* pridružujemo neku veliku

vrijednost (na primjer 1000) kako bi bili sigurni da će svako buduće odstupanje biti manje od nje. Kada se taj uvjet ispuni u varijablu *min* postavljamo vrijednost novog najmanjeg odstupanja koje će služiti kao osnova za uspoređivanje s novim vrijednostima odstupanja varijable *t1* od željenog vremena *t*. Uz to varijabla br dobiva vrijednost varijable *i*, a koja će nam označavati mjesto u polju na kojemu se nalazi vrijeme *t1* koje ima najmanje odstupanje od željenog i vrijednost preskalera i *TMR0* potrebnih za dobivanje te vrijednosti. Na kraju u novu varijablu *t2* postavljamo onu vrijednost *t1* koja najmanje odstupa od tražene vrijednosti te ju dijelimo s odabranim eksponentom vremenske jedinice.

Omjer	8-bitni kontrolni registar	Hex	Dec	16-bitni kontrolni registar	Hex	Dec
1:1	10001000	0x88	136	11001000	0xC8	200
1:2	10000000	0x80	128	11000000	0xC0	192
1:4	10000001	0x81	129	11000001	0xC1	193
1:8	10000010	0x82	130	11000010	0xC2	194
1:16	10000011	0x83	131	11000011	0xC3	195
1:32	10000100	0x84	132	11000100	0xC4	196
1:64	10000101	0x85	133	11000101	0xC5	197
1:128	10000110	0x86	134	11000110	0xC6	198
1:256	10000111	0x87	135	11000111	0xC7	199

Tablica 4.11 Konfiguracija T0CON registra

U skladu s tablicom 4.9 dvije FOR petlje u programu pridodaju vrijednosti varijablama T0PS_8 (kontrolni registar za 8-bitni tajmer) i T0PS_16 (za 16-bitni tajmer):

```
For i = 1 To 8
            If i > 1 Then
                T0PS_16(i) = T0PS_16(i - 1) + 1
            End If
        Next

        T0PS_8(0) = 200
        T0PS_8(1) = 192

        For i = 1 To 8
            If i > 1 Then
                T0PS_8(i) = T0PS_8(i - 1) + 1
            End If
        Next
```

Kako se tajmer u 16-bitnom načinu rada sastoji od dva 8-bitna registra – TMR0L i TMR0H, vrijednost *TMR0* potrebno je postaviti u istoimene varijable u VB-u kako bi ih bilo moguće ispisati.

```
Dim TMR0H As String = ""
            Dim TMR0L As String = ""
            Dim TMR0_temp As String = ""

            TMR0_temp = Hex(Math.Round(TMR0(br)))

            If Len(TMR0_temp) < 3 Then
                For i = 1 To Len(TMR0_temp)
                    TMR0L = TMR0L & Mid(TMR0_temp, i, 1)
                Next
            End If

            If Len(TMR0_temp) = 3 Then
                For i = 1 To 1
                    TMR0H = TMR0H & Mid(TMR0_temp, i, 1)
                Next
                For i = 2 To Len(TMR0_temp)
                    TMR0L = TMR0L & Mid(TMR0_temp, i, 1)
                Next
            End If

            If Len(TMR0_temp) = 4 Then
                For i = 1 To 2
                    TMR0H = TMR0H & Mid(TMR0_temp, i, 1)
                Next
                For i = 3 To Len(TMR0_temp)
                    TMR0L = TMR0L & Mid(TMR0_temp, i, 1)
                Next
            End If
```

Trima IF naredbama provjeravamo broj znakova u heksadecimalnoj tekstnoj varijabli *TMR0_temp*. Ukoliko sadrži dva ili manje znaka, cijela njezina vrijednost se sprema u varijablu *TMR0L*. Ako ima tri znaka, prvi znak se sprema u varijablu *TMR0H*, a preostala dva u *TMR0L*. U slučaju s četiri znaka, prva dva se zapisuju u *TMR0H*, a zadnja dva u *TMR0L*.

Sada su izračunate sve varijable potrebne za rad s tajmerima u mikroPascalu, te jedino što preostaje je ispisivanje u tekstnu datoteku:

```
kod = browser.SelectedPath & "\CodeTimer.txt"
            If t > maxtime_8bit Then
                objWriter.Write("procedure Interrupt();" & vbNewLine & "    begin" &
vbNewLine & "      // Ovdje unesite svoj kod" & vbNewLine & _
                                "      TMR0H := 0x" & TMR0H & ";" & vbNewLine & _
                                "      TMR0L := 0x" & TMR0L & ";" & vbNewLine & _
                                "        INTCON := 0x20;" & vbNewLine & "    end;" &
vbNewLine)
                objWriter.Write(vbNewLine & "begin" & vbNewLine)
```

```
                objWriter.Write("   T0CON := 0x" & Hex(T0PS_16(br)) & ";" & vbNewLine)
                objWriter.Write("   TMR0H := 0x" & TMR0H & ";" & vbNewLine)
                objWriter.Write("   TMR0L := 0x" & TMR0L & ";" & vbNewLine)
                objWriter.Write("   INTCON := 0xA0;" & vbNewLine)
                objWriter.Write("end.")
            Else
                objWriter.Write("procedure Interrupt();" & vbNewLine & "    begin" &
vbNewLine & "      // Ovdje unesite svoj kod" & vbNewLine & _
                                "         TMR0 := " & Math.Round((TMR0(br))) & ";" &
vbNewLine & _
                                "        INTCON := 0x20;" & vbNewLine & "    end;" &
vbNewLine)
                objWriter.Write(vbNewLine & "begin" & vbNewLine)
                objWriter.Write("   T0CON := 0x" & Hex(T0PS_8(br)) & ";" & vbNewLine)
                objWriter.Write("      TMR0 := " & Math.Round((TMR0(br))) & ";" &
vbNewLine)
                objWriter.Write("   INTCON := 0xA0;" & vbNewLine)
                objWriter.Write("end.")
            End If

            objWriter.Close()
            MsgBox("Kod je spremljen u " & kod & "!", 64, "Operation Completed!")

            fileReader = My.Computer.FileSystem.ReadAllText(kod)
            LabelCodePrevTimer.Text = fileReader
```

Ukoliko uređaj nije odabran, program će kao *Fosc* koristiti 40 MHz te će u skladu s tim računati vrijednosti najvećeg mogućeg vremena za 8-bitni i 16-bitni način rada. Naravno, da bi kod generirali nužno je odabrati uređaj pa će se te vrijednosti pojavljivati samo prilikom unosa željenog vremena i prikaza koda u okviru „Pregled koda“. Ako je uređaj odabran, očitane vrijednosti varijabli iz *datasheet-a* stavljaju se u odgovarajuće varijable. Pa će se tako prilikom promjene teksta u *textbox-u* za unos traženog vremena dogoditi:

```
Fosc = 40000000
            maxtime_16bit = 1.6777216
            maxtime_8bit = 0.0065536

            If dsON > 0 Then
                maxtime_8bit = CDbl(temp1)

                maxtime_16bit = CDbl(temp2)

                Fosc = CDbl(temp3)
            End If
```

to jest, početne vrijednosti varijabli će se zamijeniti sljedećima ukoliko je *datasheet* očitan. Varijabla *dsOn*, koja je nužna za pridruživanje tih vrijednost – IF uvjet, se sa nule povećava za jedan svakim otvaranjem *datasheet-a*:

```
Private Sub F448ToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles F448ToolStripMenuItem.Click

        datasheet.Filter = "Text (.txt)|dsPIC18F448.txt"

        If datasheet.ShowDialog = DialogResult.OK Then
            dsON = dsON + 1
        End If

    End Sub
```

5. Zaključak

Kao što je u ovom završnom radu prikazano, razumijevanje načina rada i postavljanje vrijednosti pojedinih registara mikrokontrolera je vrlo složeno bez njegovih uputa za rad (eng. *datasheet)* ili iskustva u radu s mikrokontrolerima. Kako bi se olakšao rad s mikrokontrolerima izrađena je programska podrška za postavljanje načina rada mikrokontrolera na osnovi grafičkog sučelja. Iz opisa registara u ovom završnom radu, moguće je dobiti pojednostavljen uvid u *datasheet* mikrokontrolera što je korisno budući da vrlo teško shvatiti *datasheet* mikrokontrolera bez određene razine predznanja. Kako bi se postigla bolja funkcionalnost rada mikrokontrolera, moguće je generirani kod mijenjati u programskom alatu mikroPascal. Važno je napomenuti da ova programska podrška predstavlja bazu, budući da je trenutno moguće raditi samo s mikrokontrolerom PIC18F448. Za veće mogućnosti potrebno je izraditi tekstualne datoteke koje će sadržavati vrijednosti koje programska podrška zahtjeva, a čija je primjer izrađen na temelju datoteke „dsPIC18F448“. Naravno, potrebno je te mikrokontrolere dodati i u samu programsku podršku sitnijim dodatcima u kodu pomoću programskog jezika Visual Basic. Smatram da je cilj ovog završnog rada – pojednostavljivanje generiranja vrijednosti određenih registara postignut, te vjerujem da će ova programska podrška uz buduće nadogradnje naći svoju primjenu.

6. Literatura

[S1] Slika 2.3 Visual Basic za MS-DOS
http://en.wikipedia.org/wiki/File:Microsoft_Visual_Basic_for_MS-DOS_(Professional_Edition_Version1.00).png

[S2] Slika 2.4 Način rada CLR-a
http://prabirchoudhury.files.wordpress.com/2008/12/clr_image21.jpg

[S3] Slika 3.1 Nekoliko vrsta mikrokontrolera
http://i42.tinypic.com/a2910w.jpg

[S4] Slika 3.2 Osnove komponente mikrokontrolera
http://www.mikroe.com/img/publication/pic-books/pic-microcontrollers/chapter/00/fig0-1.gif

[S5] Slika 3.3 Osnovna arhitektura mikrokontrolera
http://www.pictutorials.com/microcontroller_architectur.gif

[S6] Slika 3.4 Model memorijske lokacije
Tehničko veleučilište u Zagrebu, Goran Malčić i Velimir Rajković, „Mikrokontrolerski sustavi“, str. 30

[S7] Slika 3.5 Primjer centralne procesorske jedinice sa tri registra
Tehničko veleučilište u Zagrebu, Goran Malčić i Velimir Rajković, „Mikrokontrolerski sustavi“, str. 31

[S8] Slika 3.6 Primjer komunikacije memorije i centralne procesorske jedinice pomoću sabirnica
Tehničko veleučilište u Zagrebu, Goran Malčić i Velimir Rajković, „Mikrokontrolerski sustavi“, str. 32

[S9] Slika 3.7 Primjer U/I jedinice
Tehničko veleučilište u Zagrebu, Goran Malčić i Velimir Rajković, „Mikrokontrolerski sustavi“, str. 33

[S10] Slika 3.8 Serijska jedinica
Tehničko veleučilište u Zagrebu, Goran Malčić i Velimir Rajković, „Mikrokontrolerski sustavi“, str. 33

[S11] Slika 3.9 Mjerenje vremena tajmera
http://www.mikroe.com/img/publication/pic-books/pic-microcontrollers/chapter/00/fig0-28.gif

[S12] Slika 3.10 Korištenje prekida u radu tajmera
http://www.mikroe.com/img/publication/pic-books/pic-microcontrollers/chapter/00/fig0-29.gif

[S13] Slika 3.11 Korištenje preskalera u radu tajmera
http://www.mikroe.com/img/publication/pic-books/pic-microcontrollers/chapter/00/fig0-30.gif

[S14] Slika 3.12 Način rada WDT-a
http://www.mikroe.com/img/publication/pic-books/pic-microcontrollers/chapter/00/fig0-31.gif

[S15] Slika 3.13 A/D pretvornik
http://www.mikroe.com/img/publication/pic-books/pic-microcontrollers/chapter/00/fig0-32.gif

[S16] Slika 3.14 Unutrašnjost mikrokontrolera
http://www.pictutorials.com/microcontroller_basic_block.gif

[S17] Slika 3.15 von Neumannova arhitektura
http://www.mikroe.com/img/publication/pic-books/pic-microcontrollers/chapter/00/fig0-33.gif

[S18] Slika 3.16 Harvard arhitektura
http://www.mikroe.com/img/publication/pic-books/pic-microcontrollers/chapter/00/fig0-34.gif

[S19] Slika 3.17 Program u asembleru
http://www.mikroe.com/img/publication/pic-books/pic-microcontrollers/chapter/00/fig0-35.gif

[S20] Slika 4.1 Blok shema mikrokontrolera PIC18F448
Microchip Technology, PIC18FXX8 Data Sheet, str. 9

[S21] Slika 4.2 Dijagram pinova mikrokontrolera PIC18F448
Microchip Technology, PIC18FXX8 Data Sheet, str. 2, 3

[S22] Slika 4.3 Ulazno/izlazni portovi
http://www.mikroe.com/img/publication/pic-books/pic-microcontrollers/chapter/03/fig3-1.gif

[S23] Slika 4.4 Pin sa pull-up otpornikom
http://www.mikroe.com/img/publication/pic-books/pic-microcontrollers/chapter/03/fig3-7.gif

[S24] Slika 4.5 TRISE registar
Microchip Technology, PIC18FXX8 Data Sheet, str. 105

[S25] Slika 4.7 AD Control registar 1
Microchip Technology, PIC18FXX8 Data Sheet, str. 242

[S26] Slika 4.9 Signal različitih radnih ciklusa
http://www.netrino.com/images/glossary/PWMFigure1.gif

[S27] Slika 4.10 Blok dijagram
Microchip Technology, PIC18FXX8 Data Sheet, str. 128

[S28] Slika 4.11 Izlazni signal
Microchip Technology, PIC18FXX8 Data Sheet, str. 128

[S29] Slika 4.13 Blok dijagram tajmera TIMER0 u 8-bitnom i 16-bitnom načinu rada
Microchip Technology, PIC18FXX8 Data Sheet, str. 110

[S30] Slika 4.14 T0CON registar
Microchip Technology, PIC18FXX8 Data Sheet, str. 109

[S31] Slika 4.15 INTCON (Interrupt Control) registar
Microchip Technology, Timers: Timer0 Tutorial (Part 1), str. 5

[S32] Slika 4.16 Algoritam za povećanje varijable *counter* pomoću prekida
Microchip Technology, Timers: Timer0 Tutorial (Part 2), str. 4

[S33] Slika 4.17 T1CON registar
Microchip Technology, PIC18FXX8 Data Sheet, str. 113

[S34] Slika 4.18 Blok dijagram tajmera Timer1 u 8-bitnom načinu rada
Microchip Technology, PIC18FXX8 Data Sheet, str. 114

[S35] Slika 4.19 Oscilator tajmera Timer1
http://www.mikroe.com/img/publication/
pic-books/pic-microcontrollers/chapter/04/fig4-7.gif

[S36] Slika 4.20 Preskaler i postskaler
http://www.microcontrollerboard.com/images/
TIMER2_prescaler_postscaler.jpg

[S37] Slika 4.21 T2CON registar
Microchip Technology, PIC18FXX8 Data Sheet, str. 117

[S38] Slika 4.22 Blok dijagram tajmera Timer2
Microchip Technology, PIC18FXX8 Data Sheet, str. 118

[S39] Slika 4.23 T2CON registar
Microchip Technology, PIC18FXX8 Data Sheet, str. 119

[S40] Slika 4.24 Blok dijagram tajmera Timer3
Microchip Technology, PIC18FXX8 Data Sheet, str. 120

[T1] Tablica 3.1 Set instrukcija za mikrokontroler PIC16F84
Tehničko veleučilište u Zagrebu, Goran Malčić i Velimir Rajković, „Mikrokontrolerski sustavi“, str. 53

[T2] Tablica 3.2 Vrste PIC mikrokontrolera i njihove specifikacije
http://www.mikroe.com/eng/chapters/view/1/
introduction-world-of-microcontrollers/

[T3] Tablica 4.1 Značajke mikrokontrolera PIC18F448
Microchip Technology, PIC18FXX8 Data Sheet, str. 7

[T4] Tablica 4.2 Funkcije PORTA
Microchip Technology, PIC18FXX8 Data Sheet, str. 95

[T5] Tablica 4.3 Funkcije PORTB
Microchip Technology, PIC18FXX8 Data Sheet, str. 99

[T6] Tablica 4.4 Funkcije PORTC
Microchip Technology, PIC18FXX8 Data Sheet, str. 101

[T7] Tablica 4.5 Funkcije PORTD
Microchip Technology, PIC18FXX8 Data Sheet, str. 103

[T8] Tablica 4.6 Funkcije PORTD
Microchip Technology, PIC18FXX8 Data Sheet, str. 106

[T9] Tablica 4.7 Kombinacije ADCON1 bitova
Microchip Technology, PIC18FXX8 Data Sheet, str. 129

[T10] Tablica 4.8 PWM frekvencije i rezolucije za *Fosc* = 40Mhz
Microchip Technology, PIC18FXX8 Data Sheet, str. 242

[T10] Tablica 4.9 Vrijednosti preskalera
Microchip Technology, Timers: Timer0 Tutorial (Part 1), str. 8