CCP (*Capture, Compare and PWM*) modul

1 CCP modul

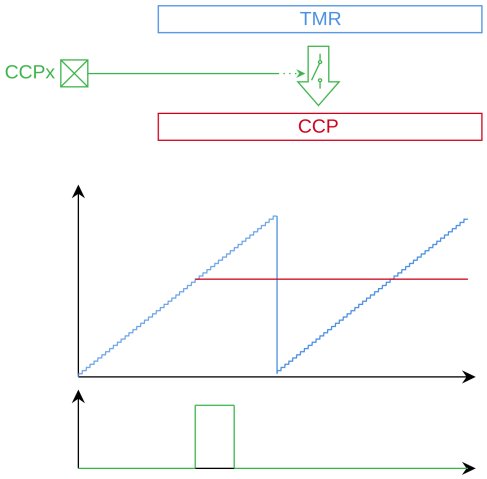
CCP (*Capture Compare PWM*) predstavlja modul mikrokontrolera koji je moguće koristiti za različite namene kao što su merenje vremena, upravljanje određenim događajima, kao i generisanje povorke četvrtki određene frekvencije i faktora ispune. Dati modul može raditi u jednom od tri moda rada:

1. Capture - detekcija spoljašnjeg događaja i merenje vremena
2. Compare - upravljanje događajima u zadatim vremenskim trenucima
3. PWM (*Pulse Width Modulation*) - generisanje signala po principu impulsno-širinske modulacije

CCP moduli mikrokontrolera su povezani sa tajmerskim modulima mikrokontrolera. Svaki CCP modul mikrokontrolera, pored registara namenjenih za konfiguraciju modula, sadrži poseban registar za podatke. Ovaj registar je u sprezi sa tajmerskim registrom i omogućava uvid u vrednost tajmerskog registra koja se neprestano menja, kao i poređenje sa datom vrednošću.

1.1 Capture

CCP modul u *Capture* modu rada omogućava detekciju određenih spoljašnjih događaja. Dodatno, kako je CCP modul povezan sa određenim tajmerskim modulima koji mere vreme na osnovu signala takta određene frekvencije, moguće je i merenje vremena do generisanja datog događaja.



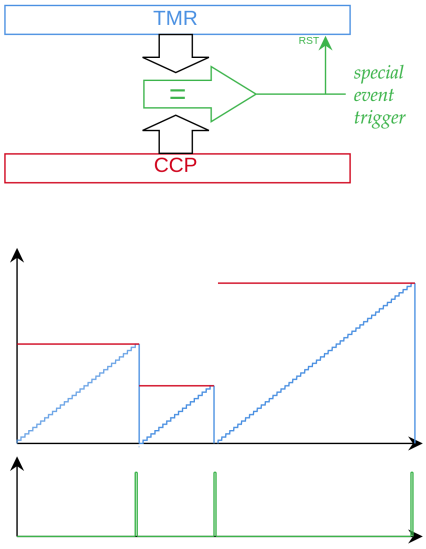
Slika 1: Princip rada CCP modula u *Capture* modu

Naime, kada se na pinu mikrokontrolera koji je namenjen određenom CCP modulu dogodi promena, na primer uzlazna ivica, CCP modul u Compare modu detektuje

ovu promenu i može da generiše odgovarajući prekid postavljanjem flag bita na jedinicu, kao i da pristupi sadržaju određenog TMR registra. Dakle, tajmerski modul meri vreme od 0 do maksimalne vrednosti i po detekciji događaja na pinu CCP modula, trenutna vrednost tajmerskog registra upisuje se u CCP registar. Na ovaj način, u trenutku kada je događaj detektovan imamo uvid u vreme koje je proteklo do generisanja događaja. Događaj koji CCP modul u Capture modu može da detektuje na pinu jeste promena stanja sa 0 na 1 (uzlazna ivica) ili promena stanja sa 1 na 0 (silazna ivica). Dodatno, upotrebom preskalera u okviru CCP modula, moguće je detektovati događaj na svaku n-tu promenu stanja na pinu. Princip rada CCP modula u Capture modu ilustrovan je na slici 1.

1.2 Compare

U *Compare* modu rada CCP modul mikrokontrolera moguće je koristiti za upravljanje događajima u zadatim vremenskim intervalima. Dakle, za razliku od Capture moda rada u kome se čeka spoljašnji događaj, u Compare modu vrši se generisanje događaja. Ovi događaji podrazumevaju promenu stanja na pinu, generisanje unutrašnjih prekida, pokretanje drugih modula i sl. U ovom modu rada u odgovarajući CCP registar neophodno je upisati vrednost koja odgovara vremenu koje je potrebno da protekne do generisanja određenog događaja. Tajmerski modul koji je u sprezi sa CCP modulom meri vreme i vrednost tajmerskog registra inkrementira se svakim otkucajm signala takta koji se dovodi na tajmerski modul. Na komparatoru u okviru CCP modula vrši se poređenje promenljivog sadržaja tajmerskog registra i vrednosti koja je upisana u CCP registar. Kada se vrednosti u ova dva registra izjednače, moguće je generisati prekid i upravljati određenom izlaznom logikom, odnosno, izvršiti određenu akciju. U ovom trenutku neophodno je i resetovati tajmer, kako bi bilo moguće započeti novu sekvencu merenja zadatog vremena i generisanja događaja. Opisani princip rada Compare moda CCP modula prikazan je na slici 2.

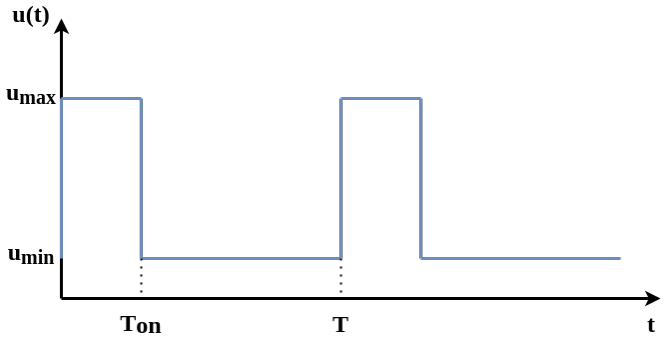


Slika 2: Princip rada CCP modula u *Compare* modu

1.3 PWM

PWM (*Pulse Width Modulation*) predstavlja metodu generisanja analognih signala na osnovu digitalnih izlaza sa pinova. Ukoliko posmatramo digitalni signal na izlaznim pinovima mikrokontrolera sa slike 3, čija je perioda T, njegovu srednju vrednost moguće je izračunati kao:

$$\overline{u} = \frac{1}{T}\int_0^T u(t)dt$$
$$\overline{u} = \frac{1}{T}(\int_0^{T_{on}} u(t)dt + \int_{T_{on}}^T u(t)dt)$$
$$\overline{u} = \frac{1}{T}(T_{on}u_{max} + Tu_{min} - T_{on}u_{min})$$
$$\overline{u} = \frac{T_{on}}{T}u_{max} + u_{min} - \frac{T_{on}}{T}u_{min}$$



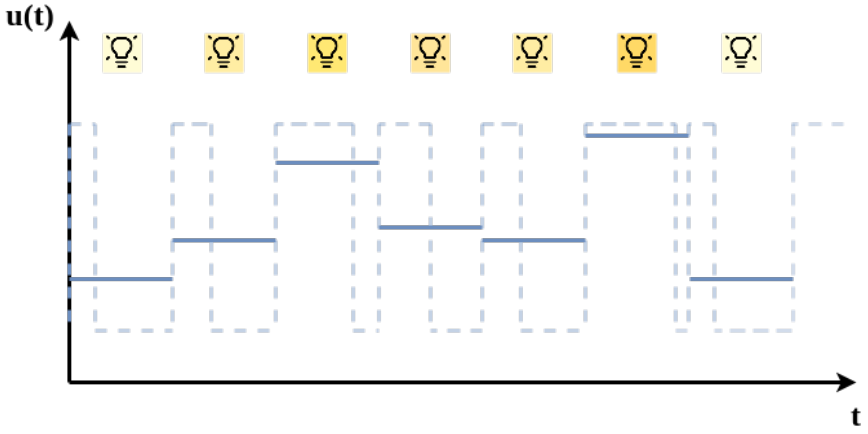
Slika 3: Digitalni signal

U prethodnom izrazu T_{on} označava vreme u kojem je vrednost signala $u(t)$ na visokom naponskom nivou u_{max} (stanje logičke 1). Odnos $D = \frac{T_{on}}{T}$ označava deo periode T koji signal provede u stanju logičke 1, a ovaj parametar naziva se **faktorom ispune** (eng. *duty cycle*). Dakle, srednju vrednost signala je moguće izraziti u odnosu na faktor ispune D kao:

$$\overline{u} = Du_{max} + u_{min} - Du_{min}$$
$$\overline{u} = Du_{max} + (1 - D)u_{min}$$

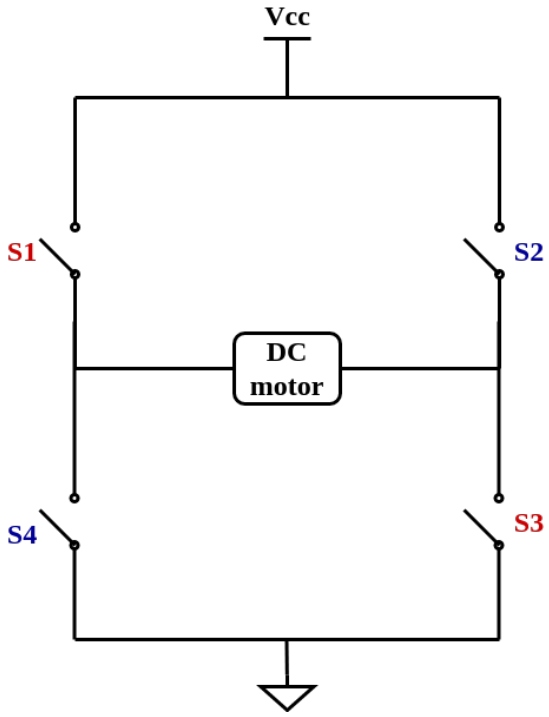
Ukoliko se na izlazne pinove mikrokontrolera poveže potrošač kojim se želi upravljati na određeni način na osnovu promene stanja na pinovima, dobija se upravo efekat usrednjavanja digitalnog signala na datom potrošaču. Kako su potrošači trome komponente, za brzo promenljive digitalne signale koji se dovode na njih, ponašaju se kao NF filtri, te se od povorke 0 (u_{min}) i 1 (u_{max}) koje se putem digitalnih pinova dovode na potrošač dobijaju srednje vrednosti datog signala. Na ovaj način, modulisanjem širine impulsa, odnosno, faktora ispune signala, utiče se na srednju vrednost signala tj. napon koji se dovodi na potrošač. Na slici 4 ilustrovan je primer promene osvetljaja LED na osnovu napona koji se dovodi na diodu. Sa porastom faktora ispune D, povećava se napon $\overline{u} = Du_{max} + (1 - D)u_{min}$, a samim tim i

osvetljaj, dok sa smanjenem vremena koje signal provodi u stanju logičke 1, opada faktor ispune i srednja vrednost napona, te je osvetljaj LED slabiji.



Slika 4: Promena osvetljaja LED modulacijom širine impulsa digitalnog signala

PWM signali se često koriste za pokretanje i upravljanje brzinom obrtaja DC motora. Za upravljanje i kontrolu DC motora tipično se koristi takozvani H-most (eng. *H-bridge*). Na slici 5 prikazana je struktura H-mosta u kojoj je povezan DC motor. Smer i brzina obrtanja DC motora kontrolisani su uz pomoć prekidača S1, S2, S3 i S4. Ovi prekidači najčešće su realizovani kao MOSFET-i koji su kontrolisani određenim naponom.



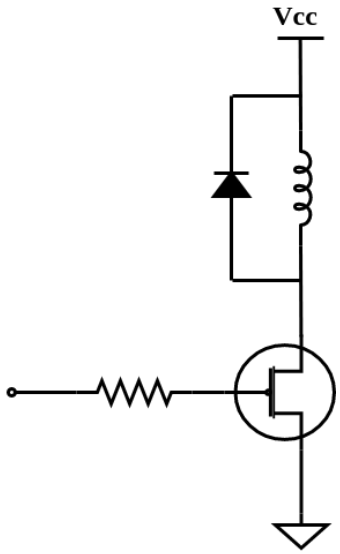
Slika 5: H-most

Za jedan smer obrtanja DC motora koriste se upareni prekidači S1 i S3. Kada su prekidači S1 i S3 zatvoreni, S2 i S4 su otvoreni i struja protiče preko S1, motora i S3, ka masi, čime se ostvaruje jedan smer obrtanja motora. U suprotnom, kada se otvore S1 i S3, a zatvore S2 i S4, ostvaruje se tok struje preko S2, motora i S4, ka masi, što dovodi do suprotonog smera obrtanja DC motora. Između aktivacije parova prekidača S1-S3 i S2-S4 neophodno je obezbediti određeno vreme zadrške, kako se ne bi desilo da oba para MOSFET-a budu aktivirana istovremeno. Dakle, dovođenjem povorke impulsa modulisane širine tj. faktora ispune na prekidače H-mosta, upravlja se DC motorom.

Treba imati u vidu da je motor induktivni potrošač i za njega važi Faradejev zakon:

$$e = -N\frac{d\Phi}{dt}$$
$$e = -NL\frac{di}{dt}$$

Dakle, ukoliko se motor naglo isključi dolazi do nagle promene struje ($\frac{di}{dt} \to \infty$) što uzrokuje indukovanje ogromne elektromotorne sile negativnog predznaka, odnosno, javlja se nagli skok napona na krajevima motora. Iz ovog razloga paralelno sa motorom vezuje se zamajna (eng. *flyback*) dioda (slika 6) čija je uloga da provede u slučaju nagle promene napona, i da tu naglu promenu ublaži.

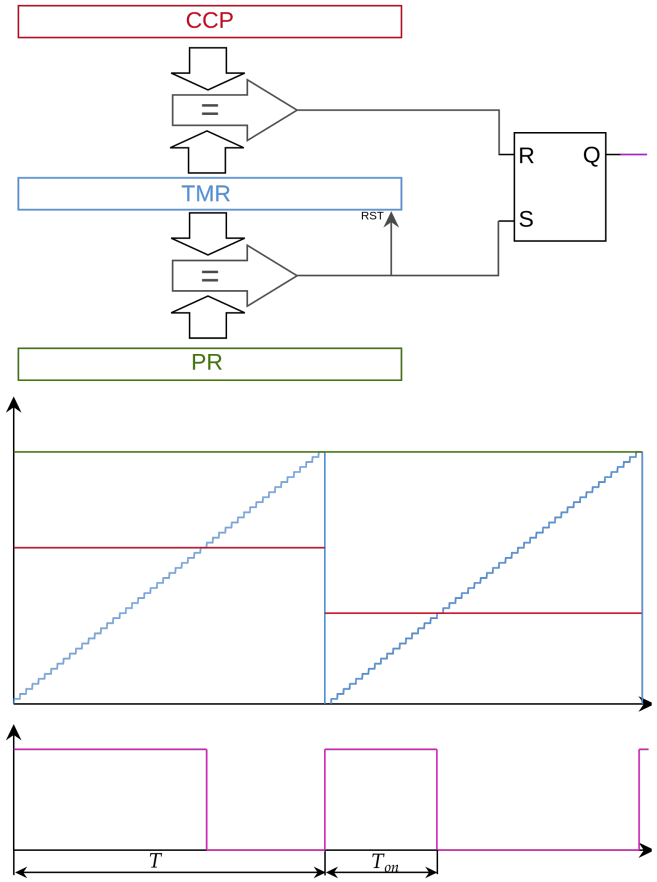


Slika 6: H-most

Pored H-mosta prikazanog na slici 5, koji predstavlja pun most, postoji i polumost, koji zahteva upotrebu dva prekidača, ali i dodatnih napajanja, te se ovakva struktura ređe koristi.

U okviru CCP modula mikrokontrolera u PWM modu rada moguće je vršiti generisanje digitalnih signala različitih faktora ispune i impulsno-širinskom modulacijom upravljati različitim potrošačima, na prethodno opisan način. Dakle, povorka logičkih 0 i 1 vodi se na određeni CCPx pin mikrokontrolera, a potrošač koji je

povezan na dati pin digitalni signal "vidi" kao njegovu srednju vrednost. Perioda i faktor ispune digitalnog signala definišu se upisom vrednosti u PR i CCP registre mikrokontrolera i njihovim poređenjem sa vrednošću u odgovarajućem tajmerskom registru (slika 7).



Slika 7: Princip rada CCP modula u *PWM* modu

Naime, u CCP registar upisuje se vrednost koja odgovara željenom vremenu visokog naponskog nivoa u signalu T_{on}, dok se vrednost koja odgovara vremenu periode signala T upisuje u PR registar. Tajmerski modul meri vreme, a kada se vrednost njegovog TMR registra izjednači sa vrednošću CCP registra, znači da je proteklo vreme T_{on}, te se na reset ulaz SR leča dovodi logička 1 pa je izlaz SR leča 0 i izlazni pin se postavlja u stanje logičke 0. Daljim inkrementiranjem vrednosti u tajmerskom registru vrednost se uvećava dok se ne izjednači sa vrednošću u PR registru, odnosno, dok ne protekne vreme od jedne periode signala. U trenutku kada se izjednače vrednosti TMR i PR registra SR leč se setuje, te je njegov izlaz logička 1, a samim tim i vrednost na izlaznom pinu postaje logička 1. Dodatno, neophodno je resetovati tajmerski moduo, kako bi počeo da broji impulse od 0, odnosno, kako bi bilo moguće generisati sledeću periodu izlaznog signala.

2 CCP modul PIC18F87K22 mikrokontrolera

U okviru PIC18F87K22 mikrokontrolera postoji sedam CCP modula i tri ECCP (*Enhanced CCP*) modula. Svaki od ovih modula može da radi sa tačno određenim tajmerskim modulima. Za rad sa svakim CCP modulom mikrokntrolera namenjena su tri SFR registra:

1. CCPxCON (CCPx Control Register) - kontrolni registar
2. CCPTMRS0, CCPTMRS1, CCPTMRS2 (CCP Timee Select Register 0,1,2) - registar za odabir tajmerskog modula
3. CCPRxL - registar za donji bajt podataka
4. CCPRxH - registar za gornji bajt podataka

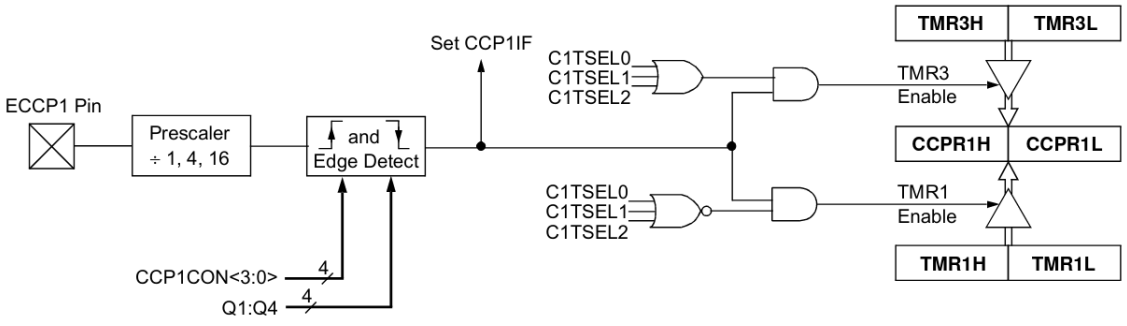
CCPxCON registar namenjen je za odabir moda rada CCP modula i konfiguraciju samog modula u odnosu na odabrani mod. Kao što je prethodno navedeno, jedan CCP modul može da radi sa tačno određenim tajmerskim modulima mikrokontrolera, stoga su neophodni registri CCPTMRS0, CCPTMRS1, CCPTMRS2, pomoću kojih je za svaki od CCP modula moguće odabrati odgovarajući tajmerski modul. Registri CCPRxL i CCPRxH predstavljaju registre namenjene za pristup vrednosti tajmerskog registra (*Capture*), zadavanje željenog vremena koje je potrebno izmeriti (*Compare*) ili zadavanje faktora ispune izlaznog signala (*PWM*).

2.1 Capture

Capture modul mikrokontrolera prikazan je na slici 8. Na pinu mikrokontrolera detektuje se promena, specificirana bitovima 3:0 u okviru CCPxCON registra. Preskalerom u okviru CCP modul omogućava se detekcija svake, svake četvrte ili svake šesnaesete promene stanja. U tabeli 1 date su vrednosti koje je potrebno upisati na mesta bitova 3:0 CCPxCON registra kako bi se na pinu CCP modula detektovala svaka silazna, svaka uzlazna, svaka četvrta uzlazna ili svaka šesnaesta uzlazna ivica. Ukoliko se CCP modulom detektuje događaj na odgovarajućem pinu, moguće je generisati prekid i postaviti odgovarajući CCPxIF fleg bit na 1. Takođe, u *Capture* modu neophodno je odabrati tajmerski registar čija će vrednost biti sačuvana u CCPR1H:CCPR1L, podešavanjem vrednosti odgovarajućih bitova CCPTMRSx registara. Za *CCP1* modul moguće je odabrati ili tajmer 1 ili tajmer 3. Važno je istaći da tajmerski registri rade u 16-bitnom modu rada.

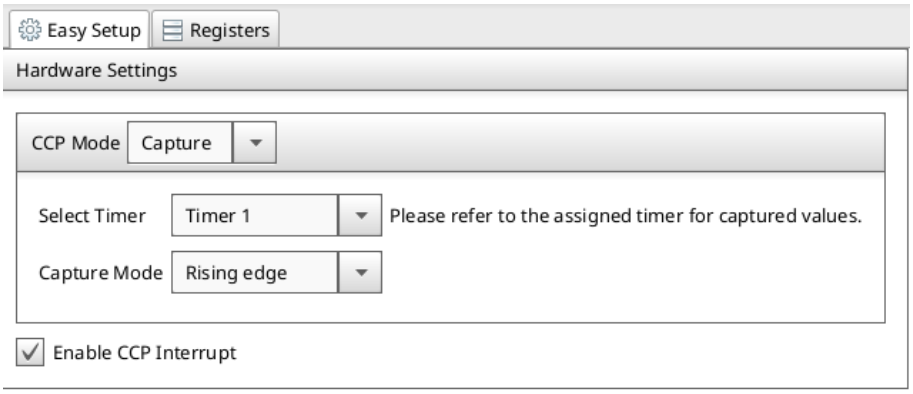
CCPxCON$< 3:0 >$	Capture
0100	svaka silazna ivica
0101	svaka uzlazna ivica
0110	svaka četvrta uzlazna ivica
0111	svaka šesnaesta uzlazna ivica

Tabela 1: Podešavanje Capture moda rada u CCPxCON registru



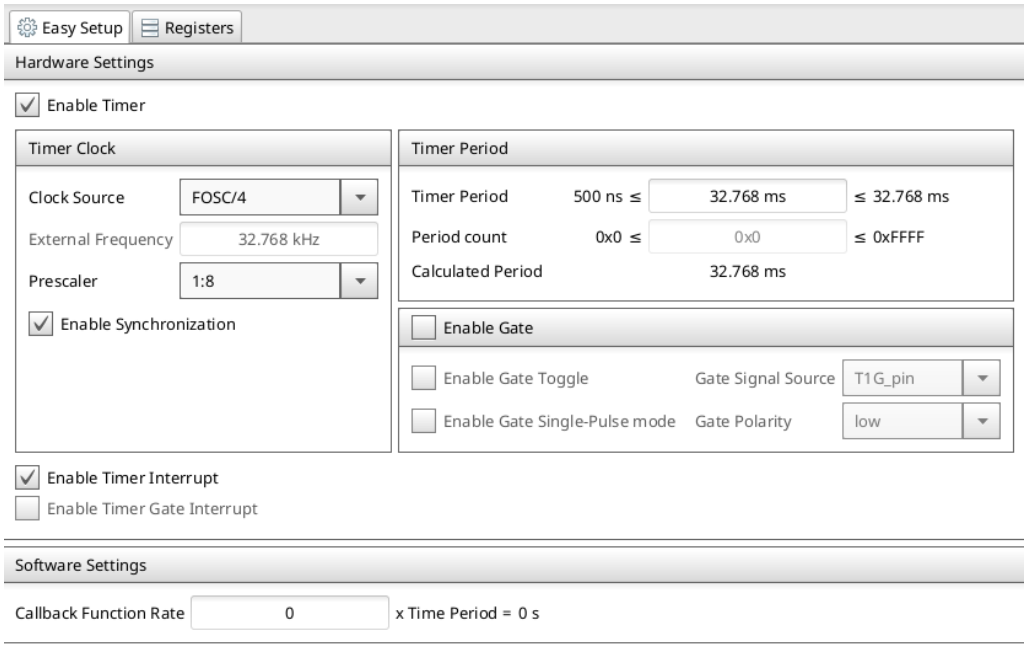
Slika 8: *Capture* modul mikrokontrolera

Kako bi se podesio *CCP* u *MCC* alatu, neohhodno ga je dodati iz padajućeg menija *Device Resources* → *Peripherals*. Podešavanja nakon odabira prikazana su na slici 9. Iz padajućeg menija *CCP mode* odabran je mod *Capture*, gde se pin *CCPx* podešava kao ulazni. Za *Capture* mode neophodno je odabrati koji će tajmer biti korišćen (padajući meni *Select Timer*), gde je u ovom slučaju odabran tajmer 1. Na kraju je neophodno odabrati mod rada (*Capture mode*) gde je kao izvor prekida odabrana uzlazna ivica na pinu mikrokontrolera. Odabirom *Enable CCP Interrupt* uključuje se generisanje prekidne rutine kada se generiše *Compare* prekid.



Slika 9: Podešavanje *CCP* modula za rad u *Capture* modu

Pošto CCP modul u ovom slučaju koristi tajmer 1 neophodno ga je i konfigurisati, što je i prikazano na slici 10. Ova konfiguracija odgovara slučaju koji će biti opisan u primeru 1. Frekvencija inkrementacije tajmerskog registra data je kao $\frac{F_{osc}}{4}$, a zatim se vrednost dovodi na preskaler koji frekvenciju deli u odnosu 1:8. Frekvencija takta mikrokontrolera je 64 MHz, onda je frekvencija inkrementacije tajmerskog registra 2 MHz. Važno je podesiti da period tajmera bude maksimalan tj. da meri vreme od 32,768 ms (od 0 do 65535 inkremenata) upisom u *Timer Period* labelu.



Slika 10: Podešavanje *TMR1* modula za rad sa *CCP* modulom

Nakon generisanja koda u *MCC* alatu u folderu ccpx.c dobijaja se niz funkcija za rukovanje *CCP* modulom u *Capture* modu.

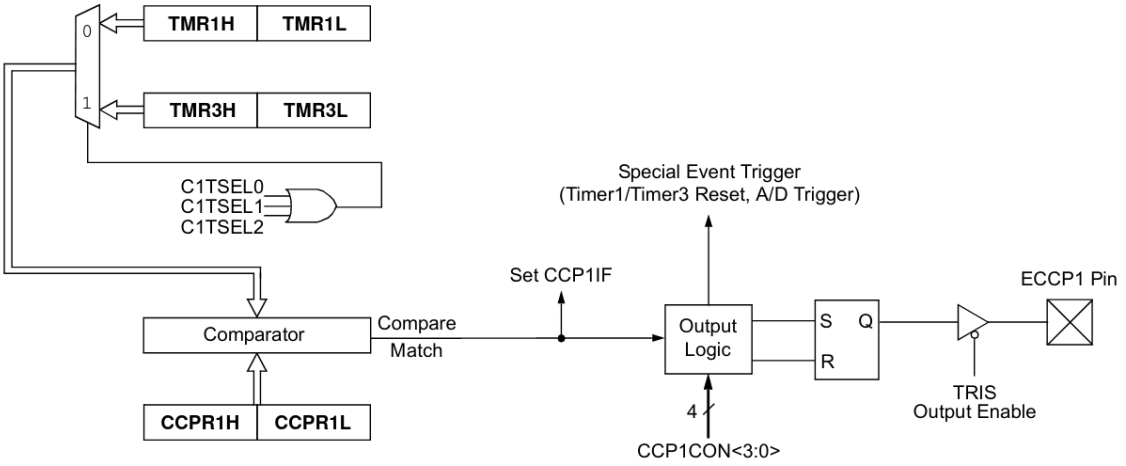
Funkcija *CCPx_Initialize* podešava registar *CCPxCON* na 0x05 što uključuje *Capture* mod sa detekcijom svake uzlazne ivice, nakon čega se resetuje sadržaj u *CCPxH* i *CCPxL* registrima koji čuvaju vrednost tajmera nakon detekcije pritiska tastera. Konfiguracijom *CCPRTMRS1* registra bira se željeni tajmer koji će se koristiti u *Capture* modu, a zatim se pozivom *CCP4_SetCallBack* bira prekidna rutina koja će se izvršiti prilikom generisanja prekida. Na kraju se briše *flag* koji dojavljuje prekid i uključuje se prekid (*PIRxbits.CCPxIF* i *PIExbits.CCPxIE* bitovi).

Prilikom generisanja prekida prvo se poziva funkcija*CCP4_CaptureISR* koja briše *CCPxIF* bit, zatim očitava vrednosti upisane u *CCPxH* i *CCPxL* registrima i na kraju poziva podešenu *callback* funkciju (ako korisnik ne specificira drugačije podrazumevano se poziva *CCP4_DefaultCallBack* funkcija).

2.2 Compare

U Compare modu rada vrši se poređenje sadržaja registara CCPRxH:CCPRxL i TMRxH:TMRxL. Na slici 11 prikazan je CCP modul PIC18F87K22 mikrokontrolera u Compare modu rada. Podešavanjem vrednosti određenih bitova CCPTMRSx registara vrši se selekcija tajmerskog modula, tako da ukoliko je izlaz ILI kola logička 1 za rad sa ECCP1 modulom bira se tajmer 3, dok je za nulu na izlazu ILI kola odabran tajmer 1. Kada se vrednost u tajmerskom registru TMRxH:TMRxL izjednači sa vrednošću koja je upisana u CCPRxH:CCPRxL registar moguće je generisati prekid (setovanje CCPxIF fleg bita) i vršiti određenu akciju (upravljanje izlaznom logikom). U tabeli 2 date su vrednosti koje je neophodno upisati u CCPxCON registar tako da CCP modul u Compare modu rada vrši promenu stanja

CCPx pina, generisanje prekida ili pokretanje događaja na drugim modulima (*Special Event Trigger*).

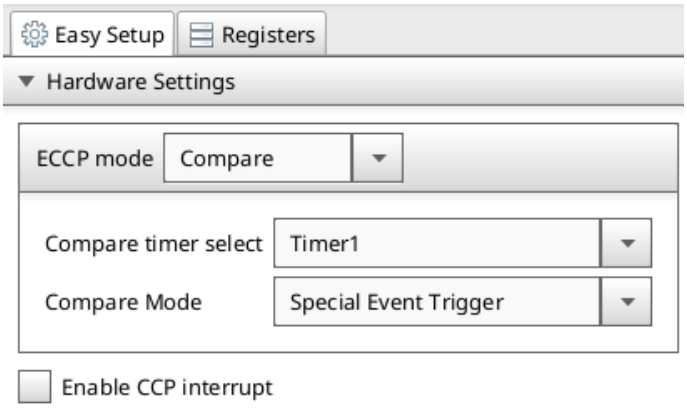


Slika 11: *Compare* modul mikrokontrolera

CCPxCON$< 3:0 >$	Compare
1000	postavljanje CCPx pina na 0
1001	postavljanje CCPx pina na 1
1010	generisanje softverskog prekida
1011	Special Event Trigger

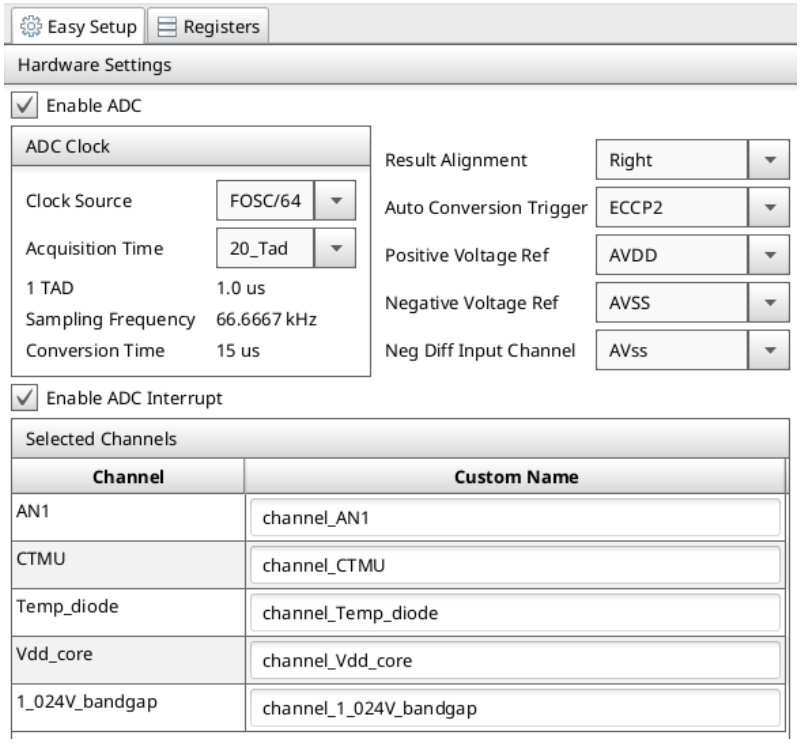
Tabela 2: Podešavanje Compare moda rada u CCPxCON registru

Kako bi se podesio *CCP* modul za rad u *Compare* modu neophodno je u *MCC* alatu dodati *CCP* modul iz *Device Resources* $\rightarrow$ *Peripherals*. U ovom slučaju biće opisana podešavanja *ECCP* (*Enhanced Capture Compare PWM*), koji je sličan *CCP* modul sa nekim dodaznim mogućnostima, što je prikazano na slici 12. Kako bi se koristio *Compare* mode neophodno je odabrati iz padajućeg menija *ECCP mode* funkcionalnost *Compare*. Takođe neophodno je specificirati tajmer koji će se korsiti za rad sa *ECCP* modulom i u ovom slučaju je odabran tajmer 1 (Odabirom iz padajućeg menija *Compare timer select* opcije *Timer1*). Pošto će se u primeru 2, *ECCP* modul koristiti kao izvor takta za početak konvetzije AD konvertora pod *Compare Mode* opcijom odabran je *Special Event Trigger*. Pošto će se koristiti prekid po završetku AD konverzije, *Compare* prekid neće biti generisan. Podešavanja tajmerskog registra su identična kao i na slici 10 izuzev što je u ovom slučaju isključeno generisanje prekida tajmerskog registra jer nikada neće doći do prekoračenja (*CCP* modul u *Compare* modu resetuje tajmer kada se vrednosti *TMRxH:TMRxL* i *CCPRxH:CCPRxL* izjednače).



Slika 12: Podešavanje *CCP* modula za rad u *Capture* modu

Pored tajmera i *ECCP* modula neophodno je podesiti i AD konvertor. Podešavanja AD konvertora prikazana su na slici 13. Važno je istaći da je*Auto Conversion Trigger* padajući meni postavljen na *ECCP2*, što znači kada dođe do izjednačenja vrednosti *TMRxH:TMRxL* i *CCPRxH:CCPRxL* AD konvertor počinje konverziju. Kako bi se korisnik obavestio o završetku konverzije uključen je prekid odabirom *Enable ADC Interrupt* opcije.



Slika 13: Podešavanje *AD* konvertorskog modula za taktovanje od *Compare* modula

Nakon generisanja koda u *eccp2.c* fajlu date su funkcije za konfiguraciju i rad sa *ECCP* modulom. Funkcije koje su generisane za tajmer i adc neophodne za rad sa *ECCP* modulom biće detaljnije opisane u primeru 2.

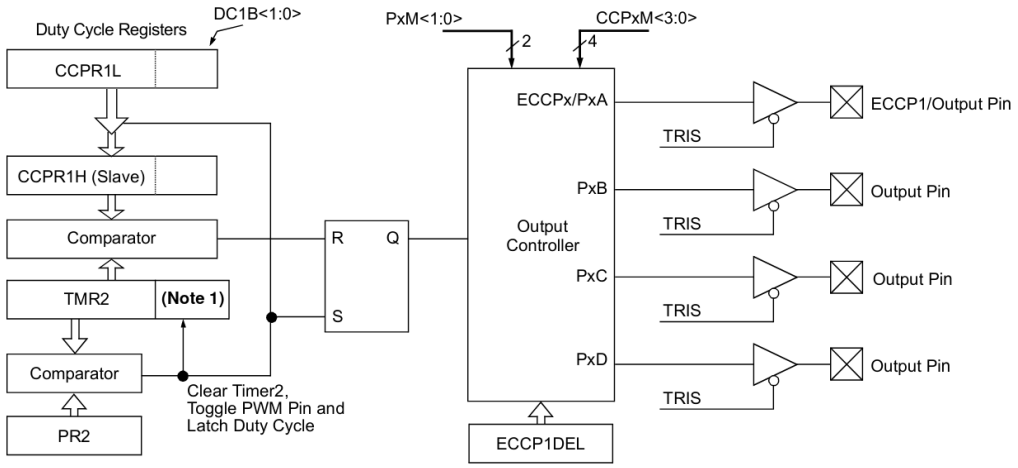
Funkcija *ECCPx_Initialize* inicijalizuje *ECCPx* modul za rad u *Compare* modu podešavanjem *CCPxCON* registra na 0x0B (x je u ovom slučaju 2). Registri za poređenje *CCPRxH* i *CCPRxL* se inicijalno postavljaju na 0 i podešavanjem *CCPTMRSx* registra bira se tajmer sa kojim će se porediti zadati pragovi.

ECCP2_SetCompareCount funkcija podešava pragove tj. u registre *CCPRxH* i *CCPRxL* upisuje vrednosti prosleđene kao parametar.

Funkcija *ECCP2_IsCompareComplete* proverava da li je vrednost tajmera dosegla postavljeni prag i u slučaju da jeste vraća *true* u suprotnom vraća *false*.

2.3 PWM

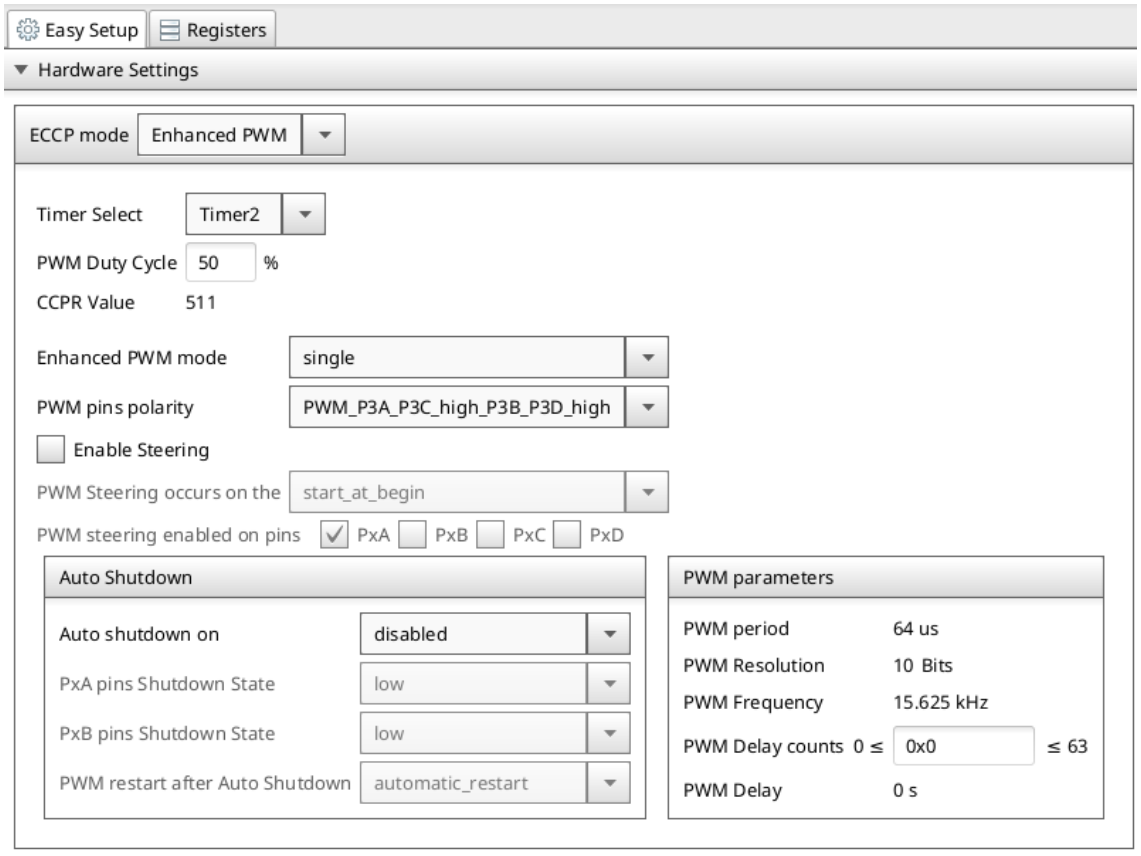
Na slici 14 prikazana je blok šema *Enhanced PWM* modula mikrokontrolera. Blok desno od *RS Latch*-a predstavlja izlazni kontrolni stepen koji podešava koji od pinova će se koristiti i u kom modu. Ovaj modul može da generiše PWM po jednom pinu, ali podržava i mogućnost povezivanja sa H mostom i H polumostom. PWM signal se generiše poređenjem tajmerskog registra sa dva komparatora jednim koji određuje faktor ispune (*CCPR1H*) i drugim koji određuje period (*PR2*). Kada se vrednost tajmerskog registra izjednači sa vrednosti u *PR2* registru tajmer se resetuje i izlaz *RS Latch*-a setuje, tj. počinje generisanje *PWM* periode. U slučaju kada se izjednače *CCPR1H* i tajmerski registar izlatni *RS Latch* se resetuje, pa se na ovaj način određuje faktor ispune. Važno je istaći da sadržaj *CCPR1H* mora uvek biti manji od *PR2*. Pošto se ovakvim poređenjima registara dobiaj 8-bitna rezolucija, a pošto je frekvencija inkrementacije tajmera $\frac{F_{osc}}{4}$ moguće je posmatrati dva bita definisana F_{osc} i $\frac{F_{osc}}{2}$ kako bi se dobila 10-bitna rezolucija. Takođe CCPR1H registru se pridodaju dva bita *DC1B¡1:0¿* koji se sada zajedno porede sa tajmerskim registrom sa pridodatim bitovima.



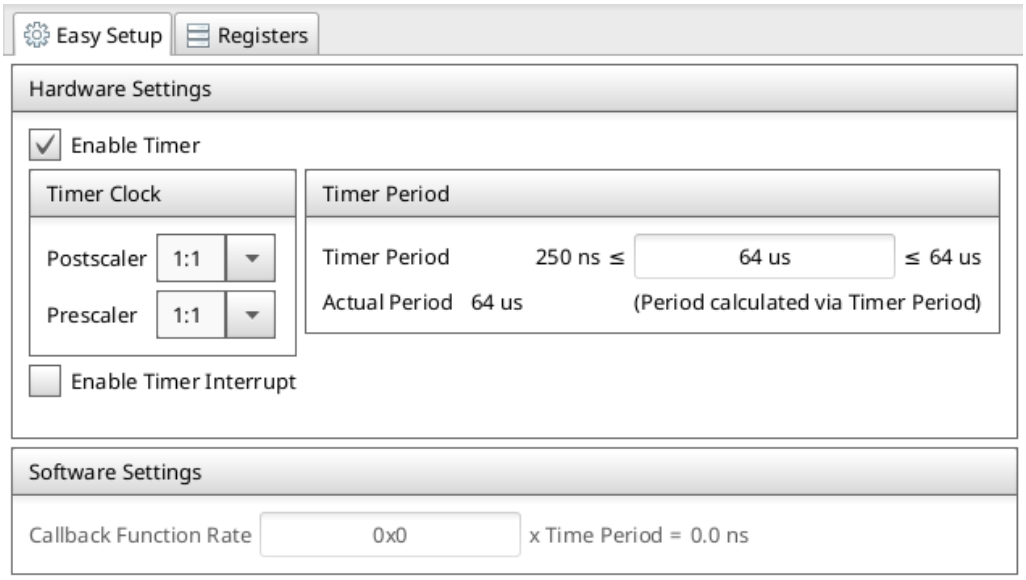
Slika 14: *PWM* modul mikrokontrolera

PWM mod je podešen dodavanjem *ECCPx* modula iz *Device Resources* → *Peripherals*. Na slici 15 data su podešavanja *ECCPx* modula za rad u *Enhanced PWM* modu (u ovom slučaju x je 3). U padajućem meniju *Timer Select* bira se tajmer koji će biti izvor za generisanje *PWM* signala, u ovom slučaju je odabran tajmer 2. Labela *PWM Duty Cycle* predstavlja inicijalnu vrednost faktora ispune i

u ovom slučaju ona iznosi 50 %. Na slici 16 data su podešavanja tajmera 2. Izvor takta je takt instrukcija (u ovom slučaju frekvencija oscilatora je 16 MHz pa je takt inkrementacije tajmerskog registra 4 MHz). Pošto se u ovom slučaju koristi 8-bitni tajmerski registar, maksimalno vreme koje može tajmer meriti je 64 μs što ujedno predstavlja i frekvenciju *PWM* signala ($\frac{1}{64}$ MHz=15,625 kHz). Na osnovu podešavanja tajmerskog registra *PWM parameters* na slici 15 određene su vrednosti periode i rezolucije *PWM* signala. U PWM slignal je moguće i uneti kašnjenje podešavanjem *PWM Delay Counts* polja, u ovom slučaju je ono podešeno na 0. Pošto je faktor ispune podešen na 50 % i rezolucija *PWM* signala je 10-bita startna vrednost u CCPR registru će biti 511. U padajućem meniju *Enhanced PWM mode* odabran je *single* mod gde se koristi samo jedan pin kao izvor *PWM* signala, a moguće je odabrati modove za rad sa H mostom ili H polumostom. *PWM pins polarity* padajući meni podešava polaritet *PWM* signala. *Enable Steering* opcija omogućuje da se u *single* modu *PWM* signal dovede na bilo koji od 4 pina predviđena za upravljanje punim mostom, što daje dodatnu fleksibilnost u projektovanju sistema. Ova opcija nije uključena, pa će se za generisanje *PWM* signala koristiti podrazumevani pin. Takođe, *EPWM* modul omogućuje da se u slučaju nekog događaja izvrši automatsko gašenje *PWM* signala. Ovo je korisno u slučaju praćenja struje potrošača/aktuator kako bi se pravovremeno isključio *PWM* signal, a potrošač/aktuator zaštitio od oštećivanja.



Slika 15: Podešavanje *CCP* modula za rad u *PWM* modu



Slika 16: Podešavanje *TMR2* modula za rad sa *CCP* modulom

Prilikom generisanja koda generiše se fajl *epwmx.c*.

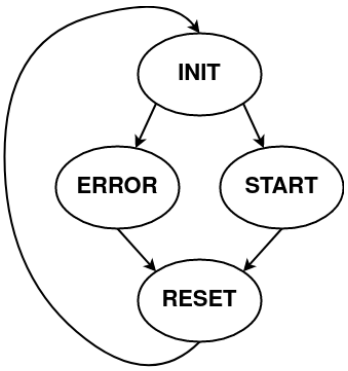
Funckija *EPWM3_Initialize* inicijalizuje *EPWM* modul u *PWM single* modu podešavanjem registra *CCPxCON* na 0x3C, nakon čega se isključuje *Auto shutdown* funkcionalnost podešavanjem *ECCP3AS* na 0x00. Kako bi se koristio *PWM* samo na izlaznom pinu *PxA*, registar *PSTRxCON* podešen je na 0x01. Kako bi se isključila pauza između generisanja dve uzastopne periode *PWM* signala podešen je *ECCPxDEL* registar. Zatim je u registre *CCPxH* i *CCPxL* upisana je vrednost kako bi se definisao faktor ispune. Važno je istaći pošto se za poređenje koristi samo 8-bitni *CCPxL* registar, a rezolucije *PWM* signala je 10 bita, na registar se pridodaju još dva bita smeštena u *CCPxONbits.DCxB*. U ovom slučaju je *CCPxONbits.DCxB* postavljen na 3, a *CCPxL* registar je postavljen na 127, pa je njihovom konkatenacijom poređena vrednost *CCPxL:CCPxONbits.DCxB* 511 odn. faktor ispune je 50 %. Na kraju se odabirom u registru *CCPTMRSxbits.CxTSEL* bira tajmer koji je izvor takta za *PWM* signal u ovom slučaju je to *Timer2*.

Funckija *EPWM3_LoadDutyValue* učitava novu vrednost faktora ispune upisom u *CCPRxL* i *CCPxCON* registre kako bi se dobila 10-bitna rezolucija.

3 Primeri

Primer 1: *Merenje brzine reakcije*

U listingu koda 1 dat je kod za merenje brzine reakcije korisnika na uključenje dioda. Prepoznata su četiri stanja, prikazana na slici 17, i to: INIT, ERROR, START i RESET.



Slika 17: Primer 1 - mašina stanja

U INIT stanju inicijalizuje se modul za merenje vremena (funkcija *TMR1_Reload* postavlja *TMR1* registar da broji od 0) kao i promenjive koje se koriste za merenje vremena (*time* meri globalno proteklo vreme od pokretanja tajmera, a *flag_ccp* dojavljuje kada se desio *capture* prekid). Kako bi se izbeglo da se korisnik prilikom merenja brzine reakcije ne navikne na pauzu između generisanja svetlosnog signala pobude, generiše se pauza na slučajan način u opsegu od 1000 ms do 3000 ms. Ovo je realizovano korišćenjem funkcije *rand*. Prilikom generisanja pauze, svake milisekunde, proverava se da li je prevremeno pritisnut taster tj. da li se desio *capture* prekid *ccp* modula proverom *flag_ccp* promenljive. Promenljiva *flag_ccp* se postavlja na *true* iz prekidne rutine *CCP4_DefaultCallBack* date u fajlu *ccp.c* kada se desi *capture* prekid. U slučaju da je korisnik pritisnuo taster prelazi se u stanje ERROR, a ako korisnik nije pritisnuo taster i pauza je istekla prelazi se u stanje START.

U ERROR stanju ispisuje se poruka korisniku da je prevremeno pritisnuo taster, posredstvom *UART* protokola, nakon čega se prelazi u stanje RESET.

Stanje START pokreće tajmer pozivom funkcije *TMR1_StartTimer* čime se počinje merenje vremena, a odmah zatim uključuju se diode na portu B kako bi se dao signal korisniku da pritisne taster. Tajmerski registar meri brzinu instrukcija ($\frac{F_{osc}}{4}$) i ima podešen preskaler 1:8, pa je brzina uvećavanja tajmerskog registra 0,5 μs. Pošto je tajmerski registar podešen da radi u 16-bitnom modu, maksimalno vreme koje može izmeriti (uvećavanje od 0 do 65535) iznosi 32768 μs. Zbog toga se promenljiva *time* koja skladišti proteklo vreme sa svakim prekidom *TMR1* modula uvećava za 32768. Paralelno sa tim se u *main* funkciji proverava da li je korisnik pritisnuo taster ili da li je promenljiva *time* veća ili jednaka od 10^7 i dok jedan od tih uslova nije ispunjen izvršavanje je blokirano u *while* petlji. Ako korisnik ne odreaguje za 10 s na pritisak tastera (promenljiva *time* bude veća ili jednaka 10^7) smatra se da eksperiment nije pravilno izveden, pa se ispisuje poruka da je vreme isteklo. U slučaju da je korisnik

pritisnuo taster ispisuje se poruka o proteklom vremenu. Neophodno je na promenvljivu *time* dodati i vreme koje je proteklo od poslednjeg prekida tajmerskog modula do pritiska tastera i ono će biti sačuvano u promenljivoj *captured_timer*. Promenljiva *captured_timer* preuzima vrednost u tajmerskom registru kada je korisnik pritisnuo taster tj. prilikom *capture* prekida u funkciji *CCP4_DefaultCallBack*. Ovu vrednost je neophodno pomnožiti sa 0,5 kako bi se dobila vrednost u μs, pa je zbog toga izvršen jedno pomeranje u desno odnosno deljenje sa 2. Na kraju se prelazi u stanje RESET.

U RESET stanju se isključuju diode i ispisuje se poruka korisniku da pritisne taster *RA0*, nakon čega se prelazi u stanje INIT i eksperiment počinje sa ponovnim izvršavanjem.

Listing 1: Merenje brzine reakcije

```
#include "mcc_generated_files/mcc.h"

void main(void)
{

    SYSTEM_Initialize();
    INTERRUPT_GlobalInterruptEnable();
    INTERRUPT_PeripheralInterruptEnable();

    uint16_t pause;
    typedef enum {INIT, ERROR, START, RESET} state_t;
    state_t state;

    state = INIT;

    while (1)
    {
        switch(state)
        {
            case INIT:
                flag_ccp = false;
                pause = 1000+rand()%2000;
                TMR1_Reload();
                time = 0;

                while(pause--)
                {
                    __delay_ms(1);
                    if(flag_ccp) break;
                }

                state = flag_ccp ? ERROR : START;
                break;
            case ERROR:
                printf("Prevremeno pritisnut taster!\n");
                state = RESET;
                break;
            case START:
                TMR1_StartTimer();
                LATB=0xFF;
                while(!(flag_ccp || time >=1E7));

                TMR1_StopTimer();
                if(time>=1E7)
                    printf("Vreme je isteklo!\n");
                else
                {
                    time += (captured_timer>>1);
                    printf("Vreme reakcije je: %.2f ms!\n",time/1000.0);
```

```
50                  }
51                  state = RESET;
52                  break;
53              case RESET:
54                  LATB = 0x00;
55                  printf("Pokusajte ponovo pritiskom na RA0!\n");
56                  while(!PORTAbits.RA0);
57                  state = INIT;
58                  break;
59              default:
60                  printf("Odabrano nepostojece stanje!");
61          }
62      }
63  }
64  /************************tmr1.h*****************************************/
65  volatile uint32_t time = 0;
66  /************************tmr1.c*****************************************/
67  void TMR1_DefaultInterruptHandler(void)
68  {
69      time+=32768;
70  }
71  /************************ccp.h*******************************************/
72  volatile uint16_t captured_timer=0;
73  volatile bool flag_ccp;
74  /************************ccp.c*******************************************/
75  static void CCP4_DefaultCallBack(uint16_t capturedValue)
76  {
77      flag_ccp = true;
78      captured_timer = capturedValue;
79  }
```

Primer 2: *Ekvidistantno odabiranje*

U listingu koda 2 prikazan je primer u kom je konfigurisan *CCP* modul u *compare* modu. Komparator *CCP* modula poredi vrednost sa tajmerom *TMR1* i kada se vrednosti izjednače generiše reset tajmera i *special event trigger* koji daje signal AD konvertoru da započne odabiranje. Po završetku konverzije AD konvertor generiše prekid koji daje signal procesoru da treba preuzeti konvertovanu vrednost. AD konverzija se vrši po kanalu AN1, zbog toga je u inicijalizaciji pozvana funkcija *ADC_StartConversion* kako bi se odabrao kanal po kom se vrši odabiranje. Pošto je takt procesora 64 MHz, onda je takt instrukcija 16 MHz i ako je preskaler podešen na 1:8 onda je vreme uvećavanja tajmerskog registra 0,5 μs. Kako bi se ostvarilo odabiranje brzinom 100 Hz neophodno je podesiti granicu *compare* modula na 20000 (0x4E20) što odgovara vremenu od 10 ms. Ovo je podešeno korišćenjem funkcije *ECCP2_SetCompareCount*. Promenljiva *flag_adc* se postavlja na true kada AD konvertor završi sa konverzijom odbirka u digitalnu vrednost tj. generiše se prekid u prekidnoj rutini *ADC_DefaultInterruptHandler*. Nakon generisanja prekida u glavnom programu se preuzima vrednost konverzije pozivom funkcije (*ADC_GetConversionResult*). Vrednosti se šalju u formatu pogodnom za prikaz u *Data Visualizer* alatu.

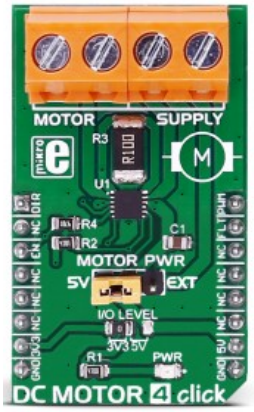
Listing 2: Ekvidistantno odabiranje signala

```
1 #include "mcc_generated_files/mcc.h"
2
3 void main(void)
4 {
5     SYSTEM_Initialize();
```

```
6       INTERRUPT_GlobalInterruptEnable();
7       INTERRUPT_PeripheralInterruptEnable();
8
9       uint16_t adc;
10      ADC_StartConversion(channel_AN1);
11      ECCP2_SetCompareCount(0x4E20);
12      while (1)
13      {
14          if(flag_adc){
15              flag_adc = false;
16              adc = ADC_GetConversionResult();
17              EUSART1_Write(0x03);
18              EUSART1_Write((uint8_t)adc);
19              EUSART1_Write((uint8_t)(adc>>8));
20              EUSART1_Write(0xFC);
21          }
22      }
23  }
24  /*******************adc.h**********************/
25  volatile bool flag_adc;
26  /*******************adc.c**********************/
27  void ADC_DefaultInterruptHandler(void)
28  {
29      flag_adc = true;
30  }
```

Primer 3: *Upravljanje pomoću PWM signala*

U ovom primeru korišćena je *DC Motor Click* pločica, povezana na treće podnožje *mikrobus* konektora na razvojnoj ploči. *DC Motor Click* pločica se kontroliše posredstvom četiri pina: EN, DIR, PWM i FLT. Uloga EN pina je da spuštanjem na nizak nivo aktivira kolo za upravljanje motorom, ako je ovaj pin na visokom stanju bez obzira što je doveden *PWM* signal motor se neće obrtati. *DIR* pin predstavlja pin kojim se upravlja smerom motora u zavisnosti od logičkog stanja na pinu motor će se okretati u jednu ili u drugu stranu. U slučaju da dođe do pregreveranja čipa ili do proticanja struje veće od preporučene *FLT* pin će biti postavljen na stanje logičku nulu i time obavestiti kontroler u suprotnom pin će biti na logičkoj jedinici. Takođe, kratkim spajanjem dva od tri pina na *MOTOR PWR* konektoru moguće je odabrati napajanje motora, da li će motor biti napajan sa 5 V sa razvojnog sistema ili sa terminala *SUPPLY* (kratkospojnik u položaju *EXT*).



Slika 18: *DC Motor Click* pločica

U listingu koda 3 dat je kod za generisanje *PWM* signala, gde korisnik pritiskom na tastere povećava ili samnjuje faktor ispune. *DIR* i *EN* pin su postavljeni na nizak nivo, kako bi se definisao smer obrtanja motora i kako bi se uključio motor. Pritiscima na tastere *INT0*(*RB0*) i *INT1*(*RB1*), smanjuje se odnosno povećava se faktor ispune *PWM* signala. Važno je istaći da je u ovom slučaju rezolucija *PWM* signala 10 bita pa je i vrednosti neophodno ograničiti na ospeg od 0 do 1023. Pošto se uvećavanje vrednosti faktora ispune menja sa korakom 50 u prekidnim rutinama *INT0_DefaultInterruptHandler* i *INT1_DefaultInterruptHandler* vrednost promenljive *duty_cycle* je ograničene na 0 i 1000, redom. Takođe, iz prekidnih rutina vrednost promenljive *flag* se postavlja na *true* kako bi se "dojavilo" glavnom programu kako je došlo do promene vrednosti faktora ispune. U glavnom programu se čeka na promenu i kada je vrednost faktora ispune promenjena postavlja se nova vrednost pozivom funkcije *EPWM3_LoadDutyValue* sa parametrom faktora ispune *duty_cycle*.

Listing 3: PWM upravljanje motorom

```
#include "mcc_generated_files/mcc.h"

void main(void)
{
    SYSTEM_Initialize();
    INTERRUPT_GlobalInterruptEnable();
    INTERRUPT_PeripheralInterruptEnable();

    DIR_SetLow();
    EN_SetLow();
    while (1)
    {
        if(flag)
        {
            flag = false;
            EPWM3_LoadDutyValue(duty_cycle);
        }
    }
}
/****************ext_int.h*********************/
volatile int16_t duty_cycle = 500;
volatile bool flag;
/****************ext_int.c*********************/
void INT0_DefaultInterruptHandler(void)
{
    duty_cycle = duty_cycle<0 ? 0 : duty_cycle-50;
    flag = true;
}

void INT1_DefaultInterruptHandler(void)
{
    duty_cycle = duty_cycle>1000 ? 1000 : duty_cycle+50;
    flag = true;
}
```