2.3.5. Programska koda

Program, ki teče v procesorju in opravlja nalogo merjenja frekvence in jakosti signala, je napisan v zbirnem programskem jeziku zbirniku. Ta je nizkonivojski jezik druge ~~generacije~~ ravni (~~prva generacija~~ najnižja raven je strojna koda). Sestavljen je iz ukaznih kod – mnemonikov, ki so v procesorju definirani po ISA (Instruction Set Architecture) arhitekturi. Vsak mnemonik predstavlja en ukaz ali operacijo, s katerimi dostopamo do posameznih registrov procesorja oziroma vplivamo na podatke v njih.

Programsko kodo tako pišemo na ~~nivoju~~ ravni operacij med registri, s čimer lahko ob dobrem znanju programiranja dosežemo visoko algoritemsko učinkovitost. Za razliko od višjih programskih jezikov, kot je na primer jezik C++, ki potrebujejo prevajalnik, zbirniški jezik prevajalnika ne potrebuje. V strojno kodo ga pretvori pomožni program Zbirnik (Assembler). Poleg pretvarjanja v strojno kodo zbirnik opravi le nekaj preprostih, a duhamornih opravil: preračuna razdalje do label, preračuna razdalje do 32-bitnih konstant v LTORG, vstavlja makroje, omogoča zapis pogosto uporabljanih konstant z labelami itd

Na začetku programske kode določimo konstante mikrokrmilnika, kot so frekvenca oscilatorja TCXO (*KVARC*), konstante za PWM (*PWMR0, PWMR1, PWMR1A*), takt za analogno-digitalni pretvornik (*ADC_DIV*) in druge. Nato dodelimo pomnilnik različnim spremeljivkam (*SKLAD, JAKOST, STANJE, ...*). Zatem inicializiramo različne enote mikrokrmilnika: Flash branje MAM, vklop enot PWRCLK, nastavitev takta vhodno/izhodnih enot APBDIV, fazno sklenjeno zanko za frekvenco jedra PLL, vhodno-izhodne enote GPIO, digitalno-analogni pretvornik DAC, vmesnik UART0, LCD krmilnik HD44780 in časovnilk TIMER0. Določimo začetno stanje inštrumenta ter pozdravni napis. Pred tem definiramo rutini ZNAK in UKAZ za izpis na LCD zaslon ter rutine za različne zakasnitve: 1 µs, 100 µs, 4 ms, 100 ms, 1 s. Diagram poteka programske kode je predstavljen na Sliki 2.10.

(tu manjka slika, na katero se sklicujete)

Glavna zanka programa mora poskrbeti za to, da se več različnih opravil izvaja istočasno: merjenje amplitude, merjenje frekvence, vnos ukazov s tipk oziroma UART0. Različna opravila se sicer izvajajo znotraj vhodno/izhodnih enot PWM, TIMER0, AD0, GPIO, UART0, a zahtevajo različno hitro ukrepanje procesorja. Hitra opravila: merjenje jakosti signala, ukazi s tipk in UART0 zahtevajo hitro ukrepanje procesorja v notranji zanki 50ms. Za počasnejšo meritev frekvence, ki traja 100ms ali 1s, zadošča zunanja zanka za meritev frekvence.

V ~~glavni~~ zunanji zanki programa najprej inicializiramo enoto PWM (*INIPWM*). V inicializaciji PWM preverimo spremenljivko *STANJE* in ugotovimo, katera časovna baza je izbrana, ali 1 Hz ali 10 Hz. V ~~glavni~~ zunanji zanki ~~s testiranjem~~ preverimo, ali se je meritev začela oziroma če so vrata odprta. Če so, preskočimo v notranjo zanko za merjenje jakosti signala in vnos ukazov. Tu nadaljujemo z meritvijo jakosti signala preko analogno-digitalnega pretvornika (ADC). Vsaka meritev jakosti signala pomeni povprečenje 16384 zaporednih A/D pretvorb, kar traja približno 50 ms. Nato preberemo morebitne ukaze iz zunanjega računalnika preko vmesnika UART0 ~~in~~ oziroma iz tipk.

Jakost signala lahko na zaslon izpišemo bodisi s številsko vrednostjo v enoti dBm bodisi grafično z rastočo vrstico, kar izbiramo s tipkami ali ukazi iz računalnika. V primeru, da je izbrani način izpisa v številski obliki z enoto, vrednost jakosti signala preračunamo s pomočjo izmerjene frekvence, da dosežemo večjo natančnost. Podrobnejša razlaga se

nahaja v poglavju *2.3.9. Popravek frekvenčnega odziva AD8309.* Za številskim rezultatom izpišemo enoto *dBm*. Meritev je natančna le v omejenem področju jakosti signala. To območje je za opisani merilnik med -60 dBm in +15 dBm. Zato v številskem načinu izpisa izpišemo *LOW*, ko je jakost merjenega signala nižja od -60 dBm, ter *HIGH*, ko jakost merjenega signala preseže zgornjo mejo +15 dBm. Kadar jakost signala izpisujemo grafično, ne izpišemo enote, niti ne preračunavamo vrednosti s pomočjo frekvence, saj nas pri grafičnem izpisu zanima le približna ocena, kje na skali se nahajamo.

Po izpisu jakosti signala preverimo, ~~če~~ ali se je meritev frekvence zaključila. V tem primeru skočimo iz hitre notranje zanke v počasnejšo zunanjo zanko. Meritev frekvence traja 100 ms ali 1 s, odvisno od izbranega časa vrat. Ko je meritev frekvence zaključena, preverimo, kateri čas vrat je izbran. Če je to 100 ms (10 Hz), potem rezultat pomnožimo z deset, da dobimo pravilno vrednost. Rezultat nato shranimo v spremenljivko *FREKVNC.* To spremenljivko uporabimo pri preračunavanju vrednosti jakosti signala, kakor je opisano v prejšnjem odstavku.

Rezultat meritve frekvence pretvorimo iz dvojiške v desetiško obliko in izpišemo na zaslon. Število pretvorimo v desetiško tako, da ga delimo z deset, ostanek shranimo na sklad in količnik delimo naprej z deset. To ~~počnemo~~ storimo tolikokrat, ~~časa, dokler se zanka ne konča~~, na kolikor desetiških mest preračunavamo rezultat. Rezultat izpisujemo na deset mest, zato se zanka desetkrat ponovi. ~~Dobljeni rezultat je zapisan v obratnem vrstnem redu, zato ga izpišemo iz desne proti levi, da dobimo pravilno številko~~. V opisanem postopku dobimo številke rezultata v obratnem vrstnem redu, najprej najnižjo številko. Zato jih zapišemo na sklad, da jih v običajnem vrstnem redu, torej najprej najvišjo številko, preberemo in izpišemo na LCD ter pošljemo na UART0.

Zaradi preglednosti brišemo vodilne ničle. To storimo tako, da ~~cifro~~ številko, enako nič, nadomestimo s presledkom. Nato preverimo, če smo na mestu enic, pred decimalno piko, kjer ničle nikoli ne brišemo. V primeru, da je tam v resnici ničla, te ne nadomestimo s presledkom, ampak na tisto mesto vpišemo nič. V decimalnem delu ničel ne brišemo. Za rezultatom zapišemo na LCD še mersko enoto *MHz*.

Na oddajnik UART0 ne pošiljamo merske enote, pač pa dva znaka za skok v novo vrstico: <CR> in <LF>. Oddajnik UART0 sicer vsebuje medpomnilnik FIFO za 16 znakov, kar pri hitrosti 9600bps, start bit, 8 podatkovnih bitov in stop bit oddamo v 16.7ms. Ker meritev frekvence oddamo z manj kot 16 znaki in novega nezultata ne pričakujemo prej kot v 100ms, se nam ni treba bati, da bi prekoračili zmogljivost medpomnilnika FIFO.

~~Program~~ Zunanjo zanko zaključimo tako, da resetiramo PWM (*RESPWM*) in se vrnemo na začetek ~~programske kode~~ glavne zanke programa.

2.3.6. Upravljanje števca

Merilnik upravljamo s štirimi tipkami. Tipke so bile prvotno mišljene za upravljanje z menuji: gor, dol, levo in desno. Naš program je zaenkrat tako preprost, da lahko vsaki tipki dodelimo kar svojo nalogo. Z dvema izbiramo čas vrat, drugi dve pa služita za izbiro načina izpisa jakosti vhodnega signala.

Poleg tipk lahko merilnik upravljamo tudi preko zunanjega računalnika. Za povezavo uporabimo USB kabel do USB/COM pretvornika FT231 na ploščici mikrokrmilnika, kot vmesnik med računalnikom in procesorjem služi enota mikrokrmilnika: UART0. Preko nje lahko merilnik upravljamo ter iz njega zajemamo meritve. Za to na računalniku

potrebujemo terminalski program, na primer Termite. V nastavitvah programa izberemo ~~ustrezen~~ pripadajoča COM ~~vhod~~ vrata (odvisno od računalnika) in ustrezen baud rate (9600 bps).

Meritve, ki jih zajemamo, so rezultati meritve frekvence, ki se izpisujejo vsaka v novi vrstici. Preko računalnika števec upravljamo tako, da v terminalski program vpišemo določeno črko, ki igra enako vlogo, kot če bi pritisnili tipko z enako funkcijo. Za izbiro časa vrat tako vpišemo „p“ ali „P“ za počasno meritev, ko je čas vrat 1 s, za hitro meritev s časom vrat 100 ms pa vpišemo „h“ ali „H“. Med načinoma izpisa jakosti signala izbiramo s črko „s“ ali „S“ za stolpec oziroma rastočo vrstico ter s črko „d“ ali „D“ za številski izpis v enoti dBm.