svet ELEKTRONIKE

Kako v programski prostor mikrokontrolerja shraniti več programov

Avtor: mag. Vladimir Mitrović
E-pošta: vladimir.mitrovic@podravka.hr

Nekateri mikrokontrolerji imajo veliko več programskega pomnilnika, kot ga zares potrebujemo za določeno opravilo. To nam ponuja možnost, da v mikrokontroler istočasno vpišemo nekaj različic istega programa, lahko pa celo popolnoma različnih programov, ki se bodo izvajali namesto glavnega. To je lahko koristno med razhroščevanjem, za demo programe, za različne izpise in še v mnogih drugih primerih. Postopek bomo ilustrirali za mikrokontrolerje iz družin AVR in 8051, ki jih programiramo v Bascom-u, kot idejo pa ga lahko uporabimo tudi za programiranje v drugih programskih jezikih in celo z drugimi družinami mikrokontrolerjev

Takoj na začetku naj opozorimo na morebitno napačno razumevanje: postopek, ki ga bomo opisali ne omogoča shranjevanja večjega števila samostojnih programov v programski pomnilnik mikrokontrolerja! Namesto tega bomo pokazali več možnosti, kako znotraj enega programa »zapakirati« različne programe ali njihove različice, kako moramo takšen program organizirati in želeni program ali njegovo različico izbrati, da ga bo mikrokontroler izvajal. Izbiro programa lahko naredimo na več bolj ali manj zapletenih načinov, izbrali pa bomo najbolj preprostega, pri katerem program izberemo s konfiguracijskimi stikali ali z mikrostikali.

Izbira programa –

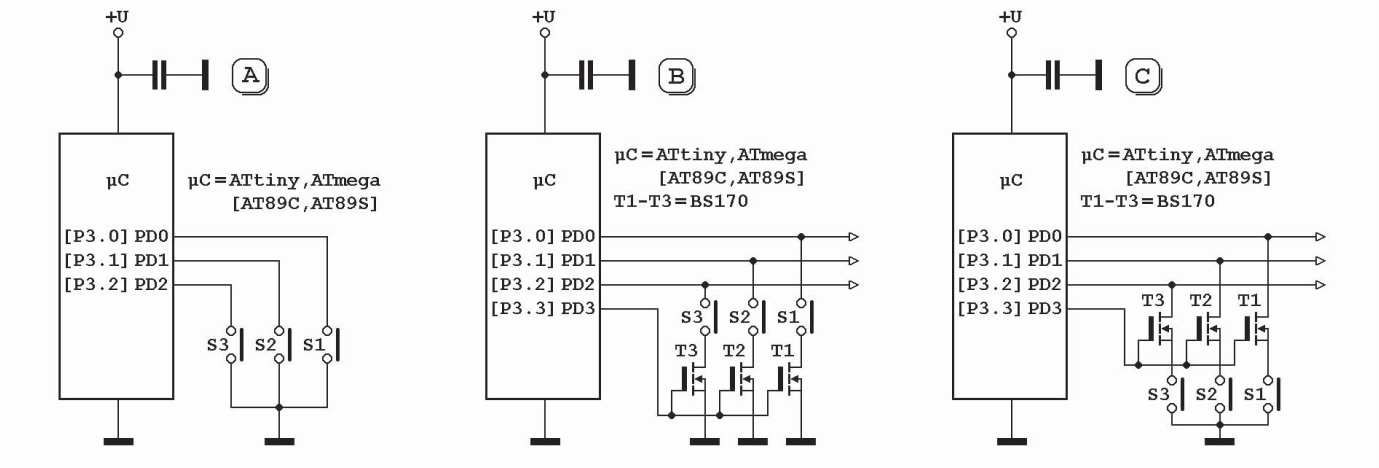
rešitev z vezjem

Slika 1 prikazuje »konfiguracijsko vezje«, sestavljeno iz treh stikal ali mikrostikal (S1, S2 in S3), ki omogočajo izbiro enega izmed osmih zapisanih programov v programskem pomnilniku mikrokontrolerja. V odvisnosti od naših potreb lahko to število še povečamo ali zmanjšamo: če dodamo le še eno stikalo, bomo lahko izbirali med 16-timi programi, kar pa bomo le redko potrebovali.

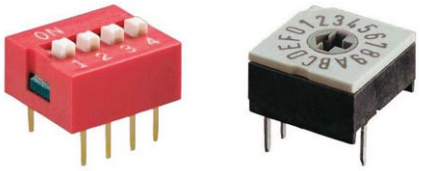
Slika 2 prikazuje DIL-stikala (levo) in kodno stikalo (desno), ki sta primerna za uporabo v vezju na sliki 1. V prikazanem stikalu so vgrajena po 4 mikrostikala. DIL-stikala proizvajajo v izvedbah z različnim številom vgrajenih mikrostikal; za uporabo v vezju na sliki 1 bo najprimernejše trojno stikalo. Razen v heksadecimalni izvedbi s slike 2, so kodna stikala lahko še BCD (4 mikrostikala, 10 položajev) ali v oktalni izvedbi (3 mikrostikala, 8 položajev). Za uporabo v vezju na sliki 1 bi bilo najprimernejša oktalna izvedba, uporabimo pa lahko tudi vse ostale, če priključek mikrostikala, ki predstavlja najpomembnejši bit, pustimo neprikljućen.

V primerjavi z DIL-stikali, je kodno stikalo primernejša rešitev, saj je s položajem vrtljivega dela nazorno prikazan izbrani program. Slaba stran kodnih stikal pa je, da imajo po en priključek vsakega od vgrajenih mikrostikal interno povezan na skupni izhodni priključek stikala, zato ga v vseh primerih ne moremo uporabiti. DIL-stikalo je v tem smislu bolj prilagodljivo, saj imajo vsi priključki mikrostikal lastne priključke tudi na ohišju.

Slika 1 prikazuje tri različice, od katerih je leva (slika 1a) najenostavnejša. V tej vezavi lahko uporabimo DIL stikalo ali kodno stikalo. Slabost te vezave je, da se priključki mikrokontrolerja,



Slika 1: S priključitvijo stikal omogočimo izbiro programa



Slika 2: DIL-stikala (levo) in kodna stikala (desno) so primerna za uporabo v vezju na sliki 1

ki so uporabljeni za branje stikala, ne morejo uporabljati v druge namene. Ta vezava je dobesedno potratna z viri mikrokontrolerja: položaj teh stikal namreč beremo na začetku programa in morda občasno še med izvajanjem programa, sicer pa so ti priključki popolnoma neizkoriščeni.

To lahko popravimo z dodajanjem treh MOSFET tranzistorjev (slika 1b). Potrebujemo še četrti priključek, s katerim vklapljamo in izklapljamo tranzistorje. Položaje konfiguracijskih stikal lahko preberemo samo takrat, kadar so tranzistorji vključeni. Ko tranzistorje izklopimo, ostanejo priključki za branje stikal prosti i njih lahko uporabimo kot običajne vhodne ali izhodne priključke (za druge namene ostane neuporaben le priključek, s katerim vklapljamo in izklapljamo tranzistorje, na sliki P3.3/PD3). Predpogoj pri tem je, da elementi ali vezja, ki so priključena na te priključke, ne bodo vplivali na rezultat pri branju stikal ali da po drugi strani sklenjeno mikrostikalo ne bo vplivalo na delovanje priključenega vezja. Priporočljivo je, da te priključke uporabimo le za enostavne naloge, na primer za prižiganje signalnih LED diod. Pa vendar - če dobro poznamo protokol komunikacije, jih lahko uporabimo celo za pošiljanje podatkov na alfanumerični prikazovalnik ali za komunikacijo z drugimi napravami.

Zaradi le enega skupnega priključka kodna stikala niso primerna za uporabo v vezju na sliki 1b. Lahko jih bomo uporabili z majhno predelavo vezja, kar je prikazano na sliki 1c. V tem vezju so položaji tranzistorjev in stikal zamenjani, tako da so stikala s svojim skupnim priključkom sedaj povezana na maso.

Priključki mikrokontrolerja, ki so na sliki 1 uporabljeni za branje stanja stikal in upravljanje s tranzistorji, so skladni s primeri programov v nadaljevanju. Z ustreznimi spremembami v programih je mogoče na enak način uporabiti katerokoli drugo kombinacijo priključkov.

Izbira programa - programske rešitve

Čeprav smo na sliki 1 ponudili več različnih vezav, so programski primeri v nadaljevanju uporabni pri vseh. Z drugimi besedami - struktura programa se ne spremeni, čeprav se odločimo za drugo vezavo. Edina podrobnost, ki je odvisna od vezave, je branje stanja stikal: v vezjih na slikah 1b in 1c je potrebno pred branjem vklopiti tranzistorje, če pa uporabimo vezje s slike 1a, to seveda ni potrebno.

Če boste uporabili kodno stikalo, morate najprej preveriti, kako se obnaša vaš primerek: nekatera stikala imajo v položaju »0« vsa mikrostikala sklenjena in jih mikrokontroler bere kot »000«, pri drugih so stikala v tem položaju razklenjena, zato jih mikrokontroler vidi kot »111«. To seveda velja za vsa mikrostikala v kodnem stikalu, zato je pojav prisoten v vseh položajih kodnega stikala: položaj »1« je lahko kodiran kot »001« ali kot »110« in tako naprej. Razlika v kodiranju ne vpliva na strukturo programa, vpliva pa na ukaze razvejanja v programu.

Na začetku programa najprej definiramo spremenljivko »S321«, v katero bomo po branju zapisali stanje stikal in konfiguriramo priključke vrat za njihovo branje:

```
Dim S321 As Byte

Config Pind.0 = Input
Portd.0 = 1
Config Pind.1 = Input
Portd.1 = 1
Config Pind.2 = Input
Portd.2 = 1
Config Pind.3 = Output
Portd.3 = 0
```

Če bomo uporabili katero izmed vezav s tranzistorji, bomo krmilni priključek nastavili kot izhod in ga postavili v stanje »0« (tranzistorji izklopljeni):

Napisati moramo tudi podprogram za branje stikal. V različici brez tranzistorjev stanje stikal preberemo z enim samim ukazom:

```
Beri_s123:
   S321=Pind And &B00000111
Return
```

V različicah vezij s tranzistorji moramo pred branjem stikal vklopiti tranzistorje:

```
Beri_s123:
   Portd.3 = 1
   Waitms 25
   S321=Pind And &B00000111
   Portd.3 = 0
Return
```

Ko stikala preberemo, tranzistorje spet izklopimo (Portd.3 = 0). Kadar zaradi drugih zahtev stikala beremo na V/I priključkih, ki niso »drug poleg drugega« ali celo pripadajo različnim vratom, bo moral biti podprogram za branje stikal približno takšen:

```
Beri_s123:
   S321 = 0
   S321.0 = Pinx.y
   S321.1 = Pinx.y
   S321.2 = Pinx.y
Return
```

Namesto »x.y« moramo vpisati resnična imena V/I priključkov, ki si priključeni na stikala. Seveda moramo pred tem te priključke definirati kot vhodne in jim vklopiti notranje »pull-up« upore, kot smo to uvodoma naredili pri PIND.0, PIND.1 in PIND.2. Kadar imamo praktično vezavo s tranzistorji, jih je treba pred branjem seveda še vklopiti in po branju izklopiti.

Primer 1

V najpreprostejšem primeru se stanje stikal prebere le na začetku programa, potem pa glede na vrednost spremenljivke »S321« skočimo na enega od programov:

```
Do
   Gosub Beri_s321
   If S321 = &B00000000
```

```
      Then Goto Prog0
   If S321 = &B00000001
      Then Goto Prog1
   If S321 = &B00000010
      Then Goto Prog2
   If S321 = &B00000011
      Then Goto Prog3
'  ...
Loop
```

Število »IF« ukazov bo odvisno od števila programov, ki smo jih vključili. V našem primeru imamo štiri programe. Če je prebrana takšna kombinacija stikal, ki še nima svojega programa, se bomo vrteli v »izločitveni zanki« in čakali toliko časa, da se stanje stikal spremeni. Takoj, ko bo prebrana kombinacija, ki ji pripada kateri od programov, skočimo iz izločitvene zanke na začetek tega programa. Čeprav takšen izhod iz DO-LOOP zanke ni dokumentiran, je v Bascomu dovoljen in za sabo ne pušča nobenih nerešenih sklicev.

Program, na katerega smo skočili, ima lahko običajno strukturo; začne se z inicijalizacijo spremenljivk, V/I priključkov in drugih mikrokontrolerjevih virov in se nadaljuje z neskončno zanko, v kateri se izvajajo predvidene naloge:

```
Prog0:
'...
'inicijalizacija
'...

Do
'...
'glavna zanka Prog0
'...
Loop
```

Na enak način definiramo tudi ostale programe.

Prednosti te rešitve:

- rešitev je zelo »pregledna«, ker so programi popolnoma ločeni eden od drugega,
- vsak program ima strukturo, ki je enaka tisti, ki bi jo imel tudi v primeru, da je v programski pomnilnik vpisan sam,
- vsak program lahko izkorišča vse vire mikrokontrolerja neodvisno od ostalih programov v pomnilniku (izjema so V/I priključki za branje stanja stikal, kot smo pojasnili prej),
- programi si lahko delijo isti RAM prostor za definiranje spremenljivk (Dim ... Overlay),
- programi imajo lahko skupne podprograme.

Pomanjkljivost te rešitve:

- ker se branje stanja stikal izvaja le enkrat, čisto na začetku glavnega programa, bomo spremembo stanja stikal zaznali le, če resetiramo mikrokontroler.

Primer 2

V tem primeru se vsi programi nahajajo znotraj »izločitvene zanke«:

```
Do
   Gosub Beri_s321
   If S321=&B00000000 Then
      '* Prog0 *
      '...
      'inicijalizacija
      'Prog0
      '...
      While S321=&B00000000
         '...
         'glavna zanka Prog0
         '...
      Gosub Beri_s321
      Wend
      '
      ' počisti za sabo
      ' End If

   If S321=&B00000001 Then
      '* Prog1 *
      '...
      'inicializacija Prog1
      '...
      While S321=&B00000001
         '...
         'glavna zanka Prog1
         '...
         Gosub Beri_s321
      Wend
      '
      'počisti za sabo
      '
   End If

'  ...
Loop
```

Po uvodnem branju stikal sledi skupina IF ukazov, s katerimi preverimo

katerega od programov moramo pognati. V našem primeru sta zaradi preglednosti vključena le dva programa, seveda pa jih po enakem pravilu dodamo še več. Ko je izpolnjen pogoj v IF stavku, se začne izvajati izbrani program. Program začne z inicializacijo spremenljivk, V/I priključkov in drugih virov mikrokontrolerja, potem pa nadaljuje z WHILE-WEND zanko, v kateri program izvaja določene aktivnosti vse do trenutka, ko se spremeni stanje konfiguracijskih stikal. Da bi program zaznal spremembo stanja stikal, je potrebno njihovo stanje prebrati pri vsakem prehodu skozi WHILE-WEND zanko. V našem primeru je klic podprograma »Beri_S321« postavljen čisto na konec zanke. Če se je stanje stikal spremenilo, bo program zapustil trenutno WHILE-WEND zanko in se »ujel« v naslednjo, v tisto, ki ustreza novemu stanju stikal.

Da bi prehod iz zanke v zanko oziroma zaključek enega programa in začetek izvajanja drugega potekal brez težav, je nujno, da vsak zaključen program po zapuščanju While-Wend zanke »počisti za sabo«. To „čiščenje“ bo odvisno od tega, katere vire mikrokontrolerja je pravkar zaključeni program uporabljal: ustaviti je treba časovnike, onemogočiti aktivne prekinitve in podobno. Vsebine spremenljivk ni potrebno čistiti, če vsak od »ugnezdenih« programov na začetku inicializira spremenljivke, ki jih uporablja.

Prednosti te rešitve:

- če je prišlo do spremembe stanja stikal, se bo novi program začel izvajati takoj, ko se izvede zadnji ukaz glavne zanke programa, ki se trenutno izvaja,
- vsak program ima strukturo, ki je enaka tisti, ki bi jo imel tudi v primeru, da je v programski pomnilnik vpisan sam,
- vsak program lahko izkorišča vse vire mikrokontrolerja neodvisno od ostalih programov v pomnilniku (izjema so V/I priključki za branje stanja stikal, kot smo pojasnili prej),
- programi si lahko delijo isti RAM prostor za definiranje spremenljivk (Dim ... Overlay),
- programi imajo lahko skupne podprograme.

Pomanjkljivosti te rešitve:

- bolj zapletena struktura glavnega programa,
- potrebno je „čiščenje“ po zaključku programa.

Primer 3

Podobno kot smo videli v drugem, tudi tretji primer omogoča »tekoči« prehod iz enega u drug program, ne da bi bilo potrebno resetirati mikrokontroler. WHILE-WEND zank, znotraj katerih se je v drugem primeru vrtel izbrani program (dokler je bilo tako nastavljeno s konfiguracijskimi stikali) tu ni več, zato je program postal enostavnejši:

```
'...
    'skupna inicializacija
    'vseh programov
'...

Do
   Gosub Beri_s321
   If S321=&B00000000 Then
        '* Prog0 *
        '...
        'glavna zanka Prog0
        '...
   End If

   If S321=&B00000001 Then
        '* Prog1 *
        '...
        'glavna zanka Prog1
        '...
   End If

'  ...
Loop
```

Vlogo glavne DO-LOOP zanke vseh programov (v tem primeru sta sicer

samo dva, vendar jih lahko na enak način dodamo še več) je prevzela DO-LOOP zanka glavnega (izbirnega) programa zato je programski tok sedaj takšen:

- na začetku izbirne DO-LOOP zanke glavnega programa preberemo stanje stikal in izvedemo tisti IF-END IF ukaz, ki ustreza trenutnemu stanju stikal,
- znotraj vsakega IF-END IF para se nahaja celoten program ali ena od različic programa,
- po izhodu iz trenutno aktivne IF-END IF strukture bo program preskočil vse ostale IF stavke, dokler ga LOOP ukaz ne vrne na začetek zanke in ponovno branje stanja stikal,
- dokler se stanje stikal ne spremeni, se bo ciklično izvajala ista IF-END IF struktura - izvajal se bo isti izbrani program,
- takoj, ko se bo stanje stikal spremenilo, se bo začela izvajati druga IF-END IF struktura – tekel bo na novo izbran program.

Ker v tako poenostavljeni rešitvi programi ne morejo imeti lastne rutine za inicializacijo, ampak si vsi programi delijo isto, bo ta rešitev ustrezna, če želimo skupaj »zapakirati« nekaj podobnih različic istega programa. Če so programi različni, si smejo iste vire mikrokontrolerja (na primer časovnike) deliti samo takrat, če jih uporabljajo na enak način. Delijo si lahko tudi iste spremenljivke, če se njihova vsebina ne prenese v naslednje izvrševanje glavne programske zanke.

Prednosti te rešitve:

- bolj enostavna in ob manjši porabi pomnilnika, kot v drugem primeru,
- če se je spremenilo stanje na stikalih, se bo novi program začel izvajati takoj, ko se bo izvedel zadnji ukaz IF-END IF strukture programa, ki se trenutno izvaja,
- programi imajo lahko skupne podprograme.

Pomanjkljivosti te rešitve:

- vsi programi uporabljajo skupno rutino za inicializacijo,
- programi si lahko delijo iste RAM spremenljivke samo v primeru, da pri vsakem novem vstopu v IF-END IF strukturo izvedemo njihovo ponovno inicializacijo,
- posamezni program lahko uporablja nek vir mikrokontrolerja samo takrat, če je to v skladu z načinom, kot ga uporabljajo drugi programi iz »paketa« ali če se ob vsakem novem vstopu v IF-END IF strukturo znova izvede njegova inicializacija.

Primeri programov za AVR in 8051 mikrokontrolerje

V uredništvu revije lahko dobite celotne primere programov »Bkutak14_1.bas«, »Bkutak14_2.bas« in »Bkutak14_3.bas«. V teh primerih je celotna struktura primerov 1, 2 in 3 za po štiri programe, z možnostjo razširitve na še večje število vgrajenih programov. Če uporabljate drug mikrokontroler namesto omenjenega ATtiny2313, morate le spremeniti ukaz, ki določa ciljni mikrokontroler ($regfile = »attiny2313.dat«), vse ostalo v programih ostaja enako. Če uporabljate druge V/I priključke za branje stanja stikal, naredite tako, kot je pojasnjeno v uvodnem delu članka.

Za uporabnike mikrokontrolerjev iz družine 8051 so pripravljeni primeri »Bkutak14_8051_1.bas«, »Bkutak14_8051_2.bas« in »Bkutak14_8051_3.bas«. Programske strukture so skoraj enake, razlike so samo pri imenovanju V/I priključkov in njihovi konfiguraciji (tu je potrebno le postavljanje vhodnega priključka v stanje »1«). Veljajo iste pripombe kot za AVR programe. Tudi te primere programov lahko dobite v uredništvu revije Svet elektronike.

www.svet-el.si

MEGAPIN
RAZVOJNO ORODJE

Združenih več funkcij.

Razvojno Orodje Vsebuje:
- 8 tipk
-programator AVR
-razhroščevalnik
-USB napajanje
-SD podnožje
-JTAG programiranje (ni v verziji B)

MegaPin postrežemo z vsem

programator AVR
JTAG programiranje
razhroščevalnik
SD kartica

WWW.SVET-EL.SI