C programiranje za Arduino

Warwick Smith

Agencija Eho
www.infoelektronika.net

Uvod

U ovoj knjizi ćete naučiti programski jezik C tako što ćete kao platformu za učenje koristiti AVR mikrokontroler zasnovan na Arduino Uno i MEGA pločicama. Atmel Studio za operativne sisteme Windows je integrisano razvojno okruženje (IDE) i softverska alatka koja se u ovoj knjizi koristi za upisivanje, kompajliranje i učitavanje C programa na Arduino pločicama.

Atmel Studio i standardni C jezik se koriste kao alternativa Arduino IDE-u koji je dizajniran da pomogne onima sa manje tehničkog znanja iz oblasti elektronike i mikrokontrolera da bi započeli projekte za mikrokontrolere. Standardni jezik C se uči na Univerzitetima kao deo smerova iz oblasti elektronskog inženjerstva i koristi se u industriji za programske mikrokontrolere. C programiranje može da se koristi na svakom mikrokontroleru za koji postoje alatke C programiranja i koji nije ograničen na Arduinu. Arduino se koristi u ovoj knjizi za učenje jezika C jer je veoma pogodan za upotrebu i lako dostupan.a strani

Učenje jezika C

Učenje novog programskog jezika i učenje kako da programirate ugrađene sisteme pokriva veliki broj informacija, jer treba naučiti mnogo stvari: sam programski jezik C, neka saznanja o arhitekturi mikrokontrolera, numeričke sisteme i logiku. Takođe treba da naučite softverski alatni lanac za programiranje i IDE, a u ovoj knjizi to je Atmel Studio IDE koji koristi GNU-ov alatni lanac za razvoj softvera za AVR mikrokontrolere. Možda je težak zadatak naučiti sve ove aspekte ugrađenih sistema i programskog jezika, ali u ovoj knjizi ćete ih naučiti korak po korak, pa će vam biti lako, ugodno i mnogo zabavno.

Tehnički termini

IDE – integrisano razvojno okruženje. IDE je softver koji obezbeđuje kompletno razvojno okruženje za pisanje i uređivanje programa, kompajliranje, učitavanje programa na ciljanu pločicu i debagiranje.

Alatni lanac (toolchain) – C alatni lanac je skup softverskih programa koji se koriste za pretvaranje programa pisanih u jeziku C u datoteku koja se može učitati u mikrokontroler.

Atmel Studio – IDE koji radi na operativnim sistemima Windows i koristi se za programiranje Atmel AVR i Atmel mikrokontrolera. Atmel Studio možete besplatno preuzeti sa Atmelovog veb sajta.

Ciljna publika

Ova knjiga je napisana za hobiste, studente i inženjere koji žele da pomoću mikrokontrolera nauče programski jezik C u ugrađenom okruženju.

AVR mikrokontroleri

U ovoj knjizi se koriste AVR mikrokontroleri koji se nalaze na Arduino Uno i Mega pločicama. Jezik C se jednako primenjuje na druge mikrokontrolere, ali kada učite

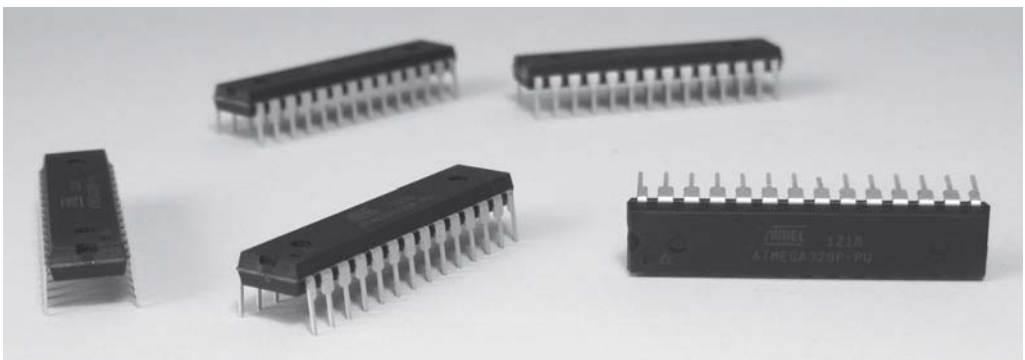
programiranje, morate da izaberete određenu arhitekturu mikrokontrolera, jer će se hardverske razlike između mikrokontrolera menjati kako oni budu programirani.

AVR arhitektura

AVR mikrokontroleri kompanije Atmel *(www.atmel.com)* su dostupni u 8-bitnim i 32-bitnim serijama. Ovi mikrokontroleri sadrže "AVR jezgro" koje je povezano sa memorijom i perifernim uređajima i spakovano u jedan čip. U ovom tekstu su predstavljeni samo 8-bitni AVR mikrokontroleri.

Tehnički termini

RISC – RISC označava Reduced Instruction Set Computer (računar sa smanjenim skupom naredaba), a to je procesor koji je dizajniran sa pojednostavljenim skupom naredaba kako bi se proizvele veće performanse. AVR arhitektura je dizajnirana na taj način.



Atmel AVR mikrokontroleri

Ugrađeni sistemi

Ugrađeni sistem je elektronska štampana ploča koja sadrži mikroprocesor ili mikrokontroler i druge elektronske delove koji pokreću softver i koristi se za posebne namene, a na za opšte namene kao što se koristi lični računar (PC).

Tehnički termini

Mikroprocesor – Mikropocesor je silicijumski čip, poznat i kao integrisano kolo ili I.C., koji sadrži centralnu procesorsku jedinicu (CPU) koja će primati naredbe programa iz memorije, a zatim ih izvršiti ili pokrenuti. Memorija koju zahteva mikroprocesor je eksterna u odnosu na čip mikroprocesora i obavezno je da je na štampanu ploču postavi hardverski inženjer koji je projektovao sistem, bez obzira da li će se ona koristiti kao ugrađeni sistem ili kao lični računar opštih namena. Postoje brojne arhitekture mikroprocesora, kao što su Intelova x86 arhitektura, ARM arhitektura, Atmelov AVR i mnoge druge.

Mikrokontroler – Mikrokontroler sadrži mikroprocesor i veliki broj perifernih uređaja, kao što su tajmeri, serijski portovi, ulazno/izlazni (I/O) pinovi, brojači, analogni ulazi itd, a svi oni se nalaze unutar jednog silicijumskog čipa. Mikrokontroler će na čipu obično sadržati i memoriju. Neki mikrokontroleri će biti u stanju da podrže čipove sa eksternom memorijom, a drugi će imati dovoljno memorije na čipu i neće moći da podrže eksternu memoriju.

Mikrokontroleri su dizajnirani za korišćenje u ugrađenim sistemima kao njima pripadajuće periferije na čipu i memorije koji će dizajneru ugrađenog sistema omogućiti da uštedi prostor na štampanoj ploči tako što neće morati da dodaje ove stavke kao eksterne uređaje.

Primer ugrađenog sistema je kalkulator. On je posebno napravljen da bi korisnik obavljao matematičke proračune na tastaturi – to je namenski uređaj. PC je, s druge strane, računar opšte namene koji može da se koristi za veliki broj namena, kao što su obrada teksta, igranje igara, server datoteka, veb server itd. (tj. nije ugrađeni sistem).

Drugi primeri ugrađenih sistema su mobilni telefoni i računarske tastature, zato što oni sadrže ugrađene mikrokontrolere koji ovim uređajima obezbeđuju funkcionalnost. Međutim, pametni telefoni su postali više lični računari nego namenski ugrađeni sistemi, čak i ako imaju svoje operativne sisteme. Mašina za pranje veša može da sadrži ugrađeni sistem koji reaguje na pritisak dugmeta na prednjoj tabli i kontroliše cikluse pranja.

Arduino pločice su pločice opšte namene koje se mogu koristiti za mnoge različite zadatke, ali one su, ipak, ugrađeni sistemi jer se na kraju koriste za posebne zadatke, kao što je sistem za praćenje temperature, animirani robot, automat za čokoladice ili bilo koji drugi projekat kojeg se možete setiti.

Skraćenice

Mikroprocesor – za mikroprocesor se često koristi skraćenica μP.

Mikrokontroler – za mikrokontroler se često koristi skraćenica μC ili MCU.

μ - je oznaka za "mikro" i takođe se koristi u elektronici ili u drugim mernim jedinicama iz oblasti inženjerstva, kao što je mikrofarad μF za vrednosti kondenzatora.

Odabir ugrađenog sistema na kojem ćete učiti jezik C

Za ovu knjigu je kao ugrađeni sistem izabran Arduino, a posebno Arduino pločice sa ugrađenim 8-bitnim AVR mikrokontrolerima koji mogu da se montiraju na standardne pločice koje se nazivaju moduli.

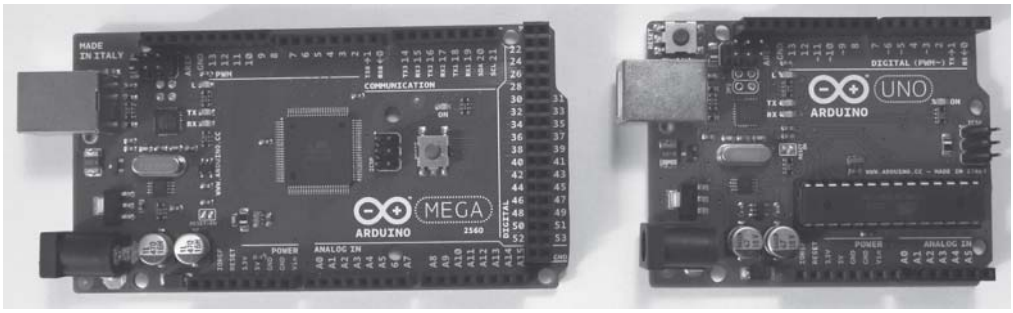
Arduino je izabran zato što je veoma dostupan, projekat je otvorenog koda i sadrži veliki broj pločica koje su kompatibilne sa komponentama drugih proizvođača. Pre nego što je Arduino postao popularan, bilo je nephodno da se za svaku knjigu o ugrađenom programiranju koristi posebna vlasnička pločica. Sa pojavom Arduina imamo standardni komercijalni hardver otvorenog koda, koji omogućava da svako pregleda i izmeni štampanu šemu i ploču.

Ovo su neki od razloga zbog kojih je izabran Arduino:

- Napajanje preko USB-a – nije potrebno eksterno napajanje

- Hardver koji se lako povezuje pomoću elektronske prototipske ploče i kratkospojnika
- USB priključak na ploči je konfigurisan kao virtuelni COM priključak na PC-iju za serijsku komunikaciju
- Dostupan sa alternativnim izvorima i kopiranim/izvedenim pločicama
- Poseduje ICSP zaglavlje za programiranje i debagiranje

Na stranicama koje slede naučićete da pišete programe koji rade na Arduino pločici ili na ugrađenim sistemima i da uključujete i isključujete LED lampice, čitate prekidače da biste proverili da li su pritisnuti ili otpušteni, šaljete poruke izvan serijskog priključka koje treba prikazati na PC-iju i mnogo štošta.



Arduino Mega (levo) i Arduino Uno (desno)

Zašto treba da koristite programski jezik C?

Kada pomislim na sve tehnološke promene koje su se desile u svetu mikrolektronike još od početka moje karijere kao programera za ugrađene programe, ono što je ostalo isto je jezik C. Arhitekture procesora i sistemi su došli i prošli, ali svima njima je zajedničko to što su programirani u jeziku C.

Razlog za popularnost i upotrebu jezika C je taj što je on standardni jezik i idealno je prilagođen programiranju ugrađenih programa u kojima je neophodan pristup hardveru sistema. Alatke za C programiranje proizvode male i brze programe, koji su obavezni kada je na ugrađenim sistemima dostupna ograničena količina memorije. Svaki put kada se na tržištu pojavi nova procesorska arhitektura, možete biti prilično sigurni da će za nju biti dostupne nove alatke za programiranje.

C programiranje je neophodno da bismo programirali druge ugrađene sisteme izvan Arduina koji ne podržavaju Arduino IDE, kao i da bismo imali više kontrole nad Arduino pločicama.

Možda ste čuli da je jezik C šifrovan i težak za čitanje, ali to je besmisleno. Jezik C je jednostavan jezik koji sadrži nekoliko ključnih reči i lako se može naučiti. Kao i kada je u pitanju svaki novi jezik, treba da upoznate sintaksu jezika. U ovoj knjizi će vam sve biti objašnjeno tako što ćemo početi od jednostavnih programa. Kada sam počeo da programiram u jeziku C, veoma sam uživao, pa se nadam se da ćete i vi.

Preduslovi

Da biste koristili ovu knjigu, treba da budete u stanju da radite na računaru pod operativnim sistemom Windows i da znate kako da preuzmete i instalirate softver na Windows sistemu.

Bilo kakvo poznavanje elektronike će biti od pomoći. Na primer, trebalo bi da znate šta je LED, prekidač, I.C. i otpornik. U tekstu ću govoriti o ovim osnovnim elektronskim komponentama. Ako niste upoznati sa ovim stavkama, postoji mnogo dostupnih knjiga iz kojih možete naučiti osnove elektronike. Dobar izvor na Vebu o osnovama elektronike, prototipskim pločama i uvodu u Arduinu je stranica: *startingelectronics.org/beginners/start-electronics-now/.*

Bilo kakvo poznavanje programiranja u nekom jeziku će pomoći, ali nije nephodno.

Pretpostavlja se da posedujete i znanje osnovne matematike, tj. sabiranje, oduzimanje, množenje, deljenje, brojanje i malo algebre. Ako niste dobri u matematici, to nije problem, jer ćete koristiti samo osnovnu matematiku, a programi koje ćete pisati će vam, u stvari, pomoći da bolje razumete matematiku.

Moraćete da imate vezu sa Internetom da biste preuzeli softverske programe i primere programa koji se koriste u ovoj knjizi (svi korišćeni softveri su besplatni).

Zahtevi hardvera

Za napajanje Arduino pločice i komunikaciju između Arduina i PC-ija potreban je standardni USB kabl. Eksterno napajanje se može koristiti za napajanje Arduino pločice, ali USB kabl će i dalje biti potreban za serijsku komunikaciju sa PC-ijem.

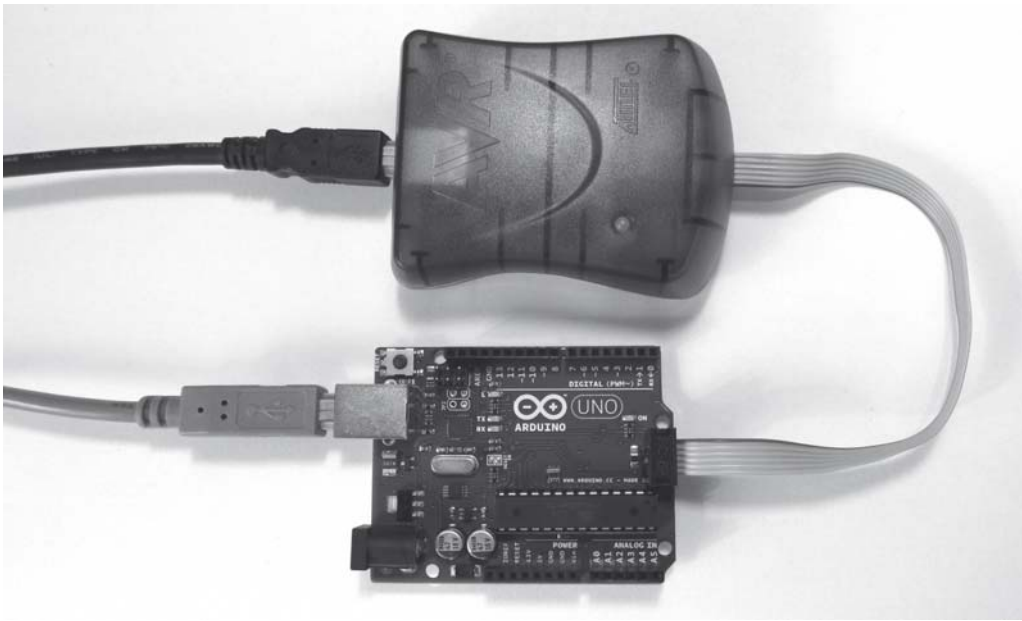
Elektronske komponente, kao što su LED, otpornici i prekidači, će biti potrebne i navode se u svakom odeljku knjige kada budu bile neophodne.

Kada koristite Atmel Studio IDE, uređaj za programiranje je potreban za učitavanje programa u mikrokontroleru na Arduino pločici. Uređaj za programiranje se povezuje sa PC-ijem pomoću USB kabla i 6-pinskog zaglavlja koji je na Arduino pločici označen kao ICSP. Više informacija o potrebnoj Arduino pločici i uređaju za programiranje možete pronaći u nastavku.

Arduino pločica

Arduino Uno sa Atmega328P mikrokontrolerom ili Arduino MEGA 2560 će biti potreban da biste pratili primere programa u ovoj knjizi. Prilikom programiranja primera koda korišćene su pločice Arduino Uno R3 i Arduino MEGA 2560 R3.

Ako još niste kupili Arduino pločicu, ja vam preporučujem da nabavite Arduino MEGA 2560, jer ona ima više memorije i ulazno/izlaznih pinova od Uno-a.



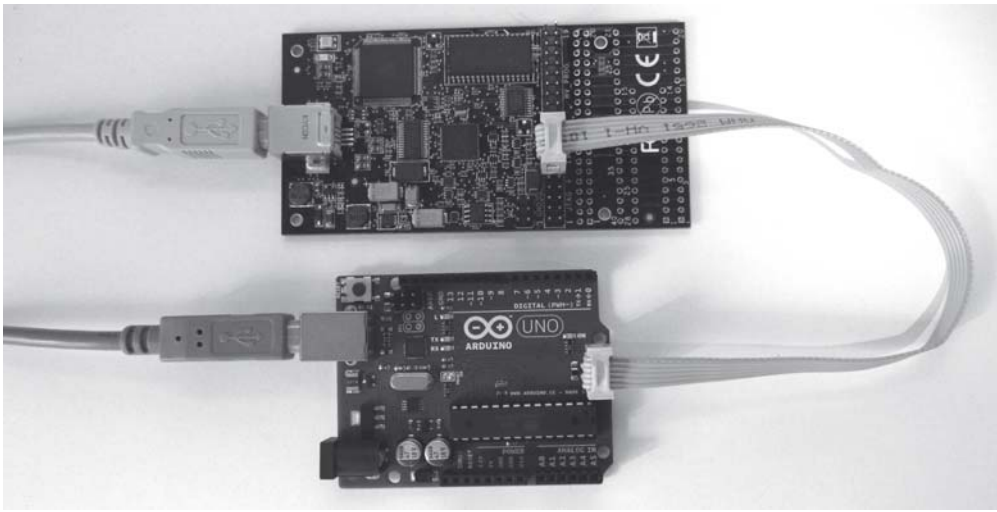
Uređaj za programiranje Atmel AVRISP mkII i Arduino Uno

Uređaj za programiranje

Uređaj za programiranje AVRISP mkll (broj dela ATAVRISP2) ili AVR Dragon uređaj za programiranje i debager (broj dela ATAVRDRAGON) je potreban za učitavanje programa na Arduino pločici. Oba ova uređaja su uređaji kompanije Atmel i potpuno su kompatibilni sa okruženjem Atmel Studio.

Ja bih preporučio da nabavite uređaj AVR Dragon, jer on poseduje mogućnosti debagiranja i u stanju je da obavlja i HV programiranje koje može da se koristi za "unbrickovanje" DIP AVR mikrokontrolera. AVRISP mkll može da programira samo AVR mikrokontrolere i ne poseduje mogućnosti debagiranja. AVR Dragon se ne isporučuje sa kablovima, pa će vam biti potreban standardni USB kabl, kao i 6-smerni trakasti kabl sa ženskim 6-smernim (2 x 3) zaglavljima na svakom kraju.

Na tržištu postoje drugi AVR uređaji za programiranje različitih proizvođača koji su možda jeftiniji od uređaja kompanije Atmel, ali da biste ih koristili biće vam potreban eksterni softver. Oni, takođe, zahtevaju dodatna podešavanja da biste ih koristili u okruženju Atmel Studio i najverovatnije neće posedovati nikakve mogućnosti debagiranja.



Atmel AVR Dragon uređaj za programiranje/debagiranje i Arduino Uno

Tehnički termini

Debagiranje – Debagiranje znači pronalaženje i otklanjanje programskih grešaka. Programer ugrađenih programa ili softverski inženjer će napisati program i testirati ga da bi video da li radi na očekivani način. Ako postoji problem sa programom, tj. "Zašto se LED ne uključuje kada se to očekuje?", programer ugrađenih programa će debagirati program.

AVR mikrokontroleri na Arduino pločicama u ovoj knjizi mogu da budu debagirani pomoću AVR Dragona povezanog sa ICSP zaglavljem na pločici. Kada softverski program u okviru okruženja Atmel Studio pozove debager koji radi na PC-iju, on pristupa ugrađenom sistemu preko ICSP zaglavlja pomoću AVR Dragona i omogućava programeru da se kreće kroz program i pregleda sadržaj memorije u mikrokontroleru. To pomaže programeru da pronađe (programsku) grešku i da je popravi.

Pristup koji je korišćen u ovoj knjizi

U ovoj knjizi smo koristili isključivo besplatne softvere ili softvere otvorenog koda. Programiranje ćete početi da učite već od poglavlja 1 nakon što instalirate i podesite alatke za programiranje.

U prvih sedam poglavlja ove knjige ćete naučiti osnove jezika C, tako što ćete rezultate iz C programa koji rade na Arduinu poslati na PC da biste ih prikazali u prozoru Terminal i primili podatke iz prozora Terminal koje je uneo korisnik. U poglavlju 8 i drugim ćete nadograditi znanje na osnove jezika C koje ste već naučili u prvim poglavljima i videćete programe koji koriste periferije mikrokontrolera za praktičnije ugrađene aplikacije.

Nekoliko dobrih saveta

Iskoristite vreme da biste naučili i razumeli kako programi funkcionišu pre nego što pređete na sledeće poglavlje. Eksperimentišite sa kodom tako što ćete ga menjati da biste videli šta će se desiti i probajte da napišete sopstvene programe, ili da modifikujete verzije primera programa da biste doprineli učenju kako budete napredovali kroz knjigu.

Ako ne možete da kucate, naučićete da kucate dok budete učili jezik C. To sam ja uradio dok sam učio program i nikada nisam zažalio. Uvek bih primere programa unosio tako što bi ih kucao, umesto da ih pokrenem sa diska.

Učenje jezika C i ugrađenih sistema je odlična zabava i velika avantura, pa želim da i vi u potpunosti uživate dok budete koristili ovu knjigu i učili nove stvari.

Prateće datoteke i veb sajt za podršku

Prateća zipovana datoteka je dostupna za preuzimanje od kompanije Elektor i sadrži ceo izvorni kod jezika C za primere programa u ovoj knjizi. Prateće datoteke preuzmite sa Elektorovog veb sajta na adresi *www.elektor.com* tako što ćete iz menija sa knjigama potražiti ovu knjigu ili upotrebiti polje za pretragu da biste pronašli stranicu knjige.

Prateći veb sajt možete pronaći na adresi *wspublishing.net/avr-c* koji sadrži dodatne tekstove i informacije povezane sa ovom knjigom i biće ažuriran sa svim greškama koje su pronađene.

Poglavlje 1 • Vaš prvi program u jeziku C

U ovom poglavlju ćete raditi sledeće:

- Preuzeti i instalirati Atmel Studio
- Preuzeti i instalirati program za emulaciju terminala
- Instalirati datoteke šablona da biste olakšali kreiranje novih projekata
- Povezati Arduino pločicu i uređaj za programiranje sa PC-ijem
- Napisati prvi program u jeziku C
- Kompajlirati, učitati i pokreniti svoj prvi program

1.1 • Preuzimanje i instaliranje Atmel Studia

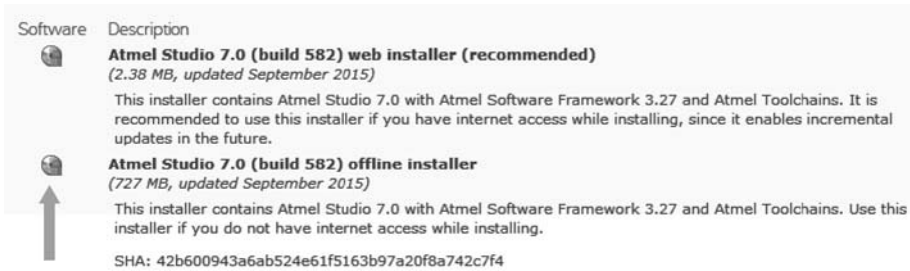
Preuzimanje i instaliranje Atmel Studia je jednostavno i ne bi trebalo da predstavlja bilo kakav problem. Kratke instrukcije za preuzimanje i instalaciju su predstavljene u nastavku. Detaljnije instrukcije sa snimcima ekrana možete pronaći na veb sajtu za podršku na adresi: *wspublishing.net/avr-c*.

1.1.1 • Preuzimanje Atmel Studia

Atmel Studio možete preuzeti tako što ćete posetiti Atmel Studio veb sajt i na stranici potražiti link sa instalacionim programom za Atmel Studio.

Atmel Studio veb stranica: *www.atmel.com/tools/atmelstudio.aspx*.

Dostupna su dva preuzimanja Atmel Studia – veb i offline verzija instalacionog programa. Preporučujem da preuzmete offline verziju koja će vam omogućiti da Atmel Studio instalirate na bilo kom PC-iju, bez obzira da li je povezan sa Internetom ili ne.



Slika 1-1 Preuzmite Atmel Studio

Kliknite sliku diska sa leve strane izabrane verzije Atmel Studia (slika 1-1) i otvoriće se nova veb stranica. Sada možete da kreirate Atmel nalog, prijaviti se pomoću postojećeg Atmel naloga ili preuzeti datoteku kao gost. Da biste preuzeli Atmel Studio kao gost, unesite svoje podatke i važeću adresu elektronske pošte. U elektronskoj poruci ćete dobiti link ka datoteci na koji možete da kliknete biste otvorili stranicu koja sadrži link sa instalacionom datotekom za Atmel Studio. Kliknite link da biste započeli preuzimanje.

Prednost kreiranja Atmel naloga je u tome što nakon instaliranja Atmel Studia možete da instalirate ispravke i dodatke za Atmel Studio koji zahtevaju prijavljivanje

na Atmel nalog. Ako obavite preuzimanje kao gost, uvek možete kasnije da kreirate Atmel nalog.

Nakon što završite preuzimanje, datoteka pod nazivom kao što je **as-installer-7.0.582-full.exe** će se nalaziti u fascikli preuzimanja. Tačan naziv datoteke može da se razlikuje u zavisnosti od toga da li je datoteka preuzimanja novija ili ne.

1.1.2 ● Instaliranje Atmel Studia

Da biste instalirali Atmel Studio, jednostavno dva puta kliknite preuzetu datoteku, što će započeti instalaciju. Moraćete da se složite sa licencom da biste mogli da nastavite instalaciju. Uradite standardnu punu instalaciju i ostavite podrazumevane nazive instalacione fascikle.

1.2 ● Preuzimanje i instaliranje emulatora terminala

Emulator terminala pod nazivom Tera Term se koristi za programiranje ulaza i izlaza između PC-ija i Arduino pločice pomoću USB kabla. Tera Term možete preuzeti sa adrese: *en.osdn.jp/projects/ttssh2/releases/*

Preuzmite zipovanu Tera Term datoteku, koja se u vreme pisanja ove knjige nalazila u verziji 4.88, i dodelite joj naziv **teraterm-4.88.zip** koji sadrži fasciklu pod nazivom **teraterm.4.88**.

Tera Term nije neophodno instalirati, samo kopirajte fasciklu **teraterm-4.88** iz zipovane datoteke u bilo koju pogodnu lokaciju na PC-iju, tj. desktopu. U fascikli teraterm-4.88 dva puta kliknite datoteku pod nazivom ttermpro.exe da biste pokrenuli Tera Term.

Najbolje bi bilo da podesite Windows tako da prikazuje nastavke datoteke u File Manageru da biste lakši prepoznali tipove datoteke. Ako omogućite da nastavci datoteke budu vidljivi, to će prikazati nastavak .exe datoteke ttermpro.exe, kao i nastavak c datoteka programa napisanih u jeziku C.

Prikazivanje nastavaka naziva datoteke u Windows 7 File Manageru

Da biste prikazali nastavke naziva datoteke u File Manageru, prvo otvorite File Manager, a zatim kliknite **Organize** → **Folder** and search options da bi iskočio okvir za dijalog Folder Options. U okviru za dijalog kliknite karticu View. U polje Advanced settings poništite izbor polja Hide extensions for known file types. U okviru za dijalog kliknite OK da biste snimili promene.

1.3 ● Instaliranje datoteka šablona

Prateće Elektorovo preuzimanje sadrži dve datoteke šablona koje se koriste da biste započeli nove projekte u Atmel Studiu pod nazivom **template_mega_dragon.zip i template_uno_dragon.zip** koje se nalaze u fascikli **Templates** datoteke preuzimanja. Da biste instalirali ove datoteke šablona, otvorite fasciklu Documents, a potom otvorite **Atmel Studio → 7.0 → Templates → ProjectTemplates** i kopirajte datoteke u podfasciklu Project Templates.

Poglavlje 2 • Osnove programskog jezika C

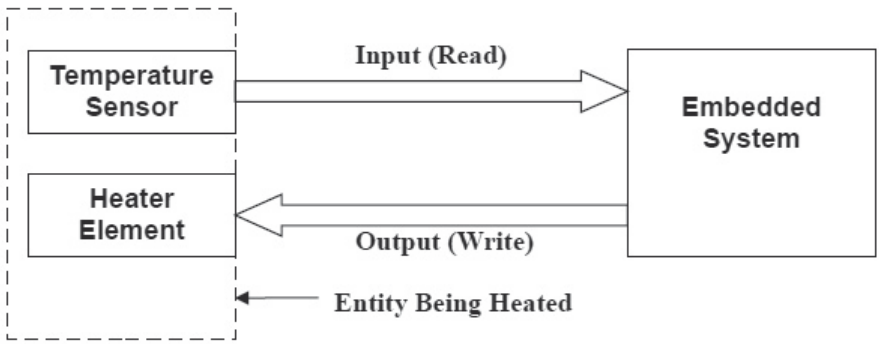
U ovom poglavlju ćete raditi sledeće:

- Pisati neke jednostavnije C programe
- Naučiti ulaz, izlaz i promenljive
- Raditi neke osnovne matematičke proračune
- Naučiti kako da pronađete i popravite greške programa

U ovom poglavlju i poglavljima koji slede postoje brojni primeri programa. Savetujem vam da unesete i pokrenete sve primere programa iz ove knjige. Slobodno izmenite programe i eksperimentišete sa njima kada budete razumeli kako oni funkcionišu. To će vam pomoći da naučite jezik C.

2.1 • Ulaz, izlaz i promenljive

Svi programi rade pomoću ulaza i izlaza. Program hello koristi izlaz isključivo tako što prikazuje poruku u prozoru emulatora terminala. Poruka koja ja poslata iz Arduina čini izlaz iz Arduina. Ako se terminalski program koristi za slanje znakova u Arduinu, to je poznato kao ulaz, zato što se oni šalju na Arduino. Većina programa će primiti neki ulaz, obraditi ga i zatim proizvesti izlaz koji je koristan. Ugrađeni sistem može očitati temperaturu (ulaz), utvrditi da li je temperatura previše visoka i zatim isključiti ili uključiti grejni element (izlaz), kao što je prikazano na slici 2-1.



Slika 2-1 Primer ulaza i izlaza u ugrađenom sistemu

Ako u Arduinu hoćemo da obavimo spoljno merenje ili da dobijemo vrednost koju ćemo dalje obraditi, treba da tu vrednost skladištimo u memoriju AVR mikrokontrolera tako da bismo mogli da je koristimo u programu. Vrednost koja je očitana se skladišti u promenljivoj.

Za razliku od nepromenljive ili konstantne vrednosti, promenljiva je deo memorije koja se koristi za skladištenje vrednosti promenljive. Kada Arduino pročita vrednost, on je prepoznaje, a to može biti bilo koja vrednost unutar opsega. Ako kažem da Arduino očitava spoljnu temperaturu, to znači da Arduino pokreće program koji ukazuje Arduinu da treba da pročita vrednost. Čitanje vrednosti je dobijanje i skladištenje te vrednosti u memoriju kao promenljive.

Da bismo na koristan način prikazali ulaz, izlaz i promenljive na Arduinu, koristićemo broj od korisnika programa (koji je korisnik uneo pomoću tastature), obaviti sabiranje broja i ispisati rezultat u terminalskom programu. Da bismo to uradili, treba da:

1. dobijemo rezultat koji je korisnik uneo pomoću tastature
2. uskladištimo vrednost na Arduinu da bismo mogli da joj dodamo broj
3. prikažemo rezultat sabiranja korisniku u terminalskom programu

Počnimo tako što ćemo pogledati kako da ispišete broj u terminalskom programu. Ponovo ćemo koristiti funkciju `printf()`. Kreirajte novi C projekat iz datoteke šablona pod nazivom number i dodajte sledeće linije koda u C datoteku.

Naziv projekta: number

Lokacija projekta: CH2/number

main.c

```
#include <stdio.h>
#include "stdio_setup.h"

int main(void)
{
    UartInit();

    printf("Printing a number: 19\r\n");
    printf("Printing a number: %d", 19);

    while(1) {
    }
}
```

Rezultat programa

```
Printing a number: 19
Printing a number: 19
```

Sačuvajte datoteku, napravite projekat i učitajte program.

Program pokazuje da, kao i pomoću prve funkcije `printf()`, broj možete ispisati kao deo teksta, ili kao spoljni broj koji će biti zamenjen tekstom. U drugoj funkciji `printf()`, broj 19 se zamenjuje specifikatorom formata %d. Specifikatore formata ćemo pogledati detaljnije u narednim programima. Nama je potreban drugi način ispisivanja broja koji je promenljiva (u ovom programu broj je konstanta koju ćemo zameniti stringom – on je upisan u program i ne može biti izmenjen nakon kompajliranja).

Prvi način ispisivanja broja neće dozvoliti da uskladištena vrednost koju je uneo korisnik programa bude prikazana na ekranu, zato što je deo stringa teksta koji se ne može menjati (poznata je kao konstanta stringa). Uskladištenu vrednost promenljive ćemo zameniti stringom, baš kao što se broj 19 zamenjuje drugom tekstualnom porukom.

Program za obradu numeričkih vrednosti možda sada neće imati mnogo smisla, jer dve naredbe `printf()` proizvode isti rezultat, ali to rade na drugačiji način.

Poglavlje 3 • Operator poređenja i odluke

U ovom poglavlju ćete raditi sledeće:

- Naučiti kako da upoređujete celobrojne vrednosti
- Pisati programe koji donose odluke

3.1 • Upoređivanje vrednosti

Operatori upoređivanja se koriste da bi se uporedile vrednosti, na primer – da li je vrednost promenljiva X veća od vrednosti promenljive Y? Ove vrednosti se mogu uskladištiti kao konstante ili promenljive. Operatori upoređivanja su poznati i kao relacioni operatori.

3.1.1 • Vrednost True i False

Kada se izračunava izraz koji sadrži operator poređenja, on će imati vrednost true ili false. U logičkim sistemima i računarima vrednost true je predstavljena brojem 1, a vrednost false brojem 0. U sledećem programu pod nazivom **compare** će biti prikazano značenje ovih vrednosti.

Naziv projekta: compare
Naziv lokacije: CH3\compare
main.c

```
#include <stdio.h>
#include "stdio_setup.h"

int main(void)
{
    int temper = 20;

    UartInit();

    printf("Is the temperature less than 23? %d\r\n", temper < 23);
    printf("Is the temperature more than 23? %d\r\n", temper > 23);

    printf("\nTemperature is now increasing...\r\n\r\n");
    temper = 50;

    printf("Is the temperature less than 23? %d\r\n", temper < 23);
    printf("Is the temperature more than 23? %d\r\n", temper > 23);

    while(1) {
    }
}
```

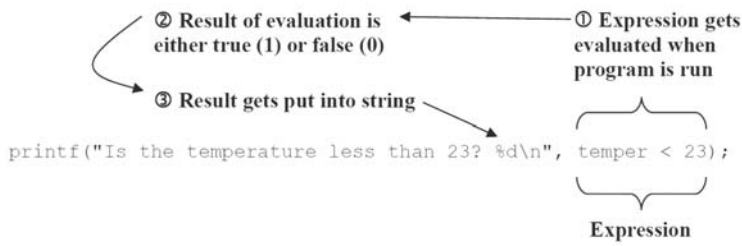
Rezultat programa

```
Is the temperature less than 23? 1
Is the temperature more than 23? 0

Temperature is now increasing...
```

U ovom primeru promenljiva pod nazivom temper se definiše kako bi se uskladištila temperaturu. Početna vrednost dodeljena promenljivoj je 20, što predstavlja 20 stepeni Celzijusovih. Kao što možete videti u primeru, promenljiva može da se definiše i inicijalizuje pomoću vrednosti u jednoj liniji koda. U primerima u prethodnim poglavljama koristili smo dve linije koda da bismo to uradili. U prvoj naredbi `printf()`, vrednost sačuvana u promenljivoj temper se upoređuje sa vrednošću konstante 23 pomoću matematičkog znaka "manje od": `temper < 23`. Drugim rečima kažemo: "Da li je vrednost sačuvana u promenljivoj `temper` manja od 23?"

Kada se ovaj izraz izračunava u jeziku C, imaće vrednost true (1) ili false (0). Rezultat će potom zameniti specifikator formata. Kao što možete videti prilikom rada programa, rezultat pitanja: "Da li je broj 20 (sačuvan u promenljivoj `temper`) manji od broja 23?" je 1 (true). Logično, to se dešava zato što je 20 manje od 23. Pogledajte sliku 3-1.



Slika 3-1 Rad operatora poređenja

U sledećoj naredbi `printf()` pitaćemo da li je 20 veće od (>) 23 i dobićemo rezultat 0 (false).

Kada vrednost u promenljivoj `temper` promenimo u 50, a mi postavimo isto pitanje, dobićemo rezultat false za naše pitanje: "Da li je 50 manje od 23?", a vrednost true za pitanje: "Da li je 50 veće od 23?", kao što smo i očekivali.

3.1.2 • Operatori poređenja u jeziku C

U tabeli 3-1 prikazani su operatori poređenja u jeziku C. Već ste videli kako funkcionišu operatori "manje od" i "veće od".

Tabela 3-1 Operatori poređenja u jeziku C

Operator	Značenje	Primer	Rezultat (1 = true i 0 = false)
<	Manje od	A < B	ako je A manje od B, rezultat je 1, a u suprotnom je 0
>	Veće od	A > B	ako je A veće od B, rezultat je 1, a u suprotnom je 0
<=	Manje od ili jednako	A <= B	ako je A manje ili jednako sa B, rezultat je 1, a u suprotnom je 0
>=	Veće od ili jednako	A >= B	ako je A veće ili jednako sa B, rezultat je 1, a u suprotnom je 0
==	Jednako	A == B	ako je A jednako sa B, rezultat je 1, a u suprotnom je 0
!=	Nije jednako	A != B	ako A nije jednako sa B, rezultat je 1, a u suprotnom je 0

Poglavlje 4 • Petlja while i komentarisanje koda

U ovom poglavlju ćete raditi sledeće:

- Naučiti kako da koristite petlju `while`
- Poboljšati primer programa za kontrolisanje temperature
- Naučiti komentarisanje programa i stil programiranja

U prethodnom poglavlju videli smo da program može da se napravi tako da se grana kada donosi odluku. Grananje će izazvati da se regularni tok programa izmeni od vrha ka dnu kroz listu naredbi. Petlje, takođe, menjaju tok programa, kao što ćemo objasniti u ovom poglavlju.

4.1 • Petlja while

Petlja `while` će izazvati da se blok programskog koda više puta pokrene od vrha ka dnu dok određeni uslov ima vrednost true. Pokretanje koda više puta se zove pravljenje petlje.

Petlja `while` je prikazana u programu **water**.

Naziv projekta: **water**
Lokacija projekta: CH4\water
main.c

```
#include <stdio.h>
#include "stdio_setup.h"

int main(void)
{
    int water_level = 0;

    UartInit();

    printf("Pumping 10L of water...\r\n\r\n");

    while (water_level < 10) {
        water_level = water_level + 1;
        printf("Progress: %d\r\n", water_level);
    }

    printf("\r\nFinished.");

    while(1) {
    }
}
```

Rezultat programa

```
Pumping 10L of water...

Progress: 1
Progress: 2
Progress: 3
Progress: 4
```

```
Progress: 5
Progress: 6
Progress: 7
Progress: 8
Progress: 9
Progress: 10

Finished.
```

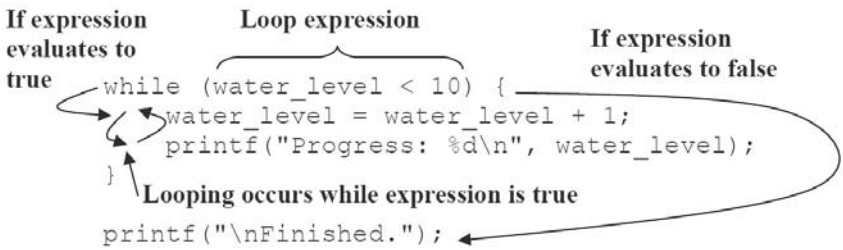
Petlja `while` izračunava izraz poređenja da bi se proverilo da li ima vrednost true ili false na isti način kao što je to radila naredba `if` u prethodnom poglavlju. Blok koda između vitičastih zagrada koji se nalazi ispod petlje `while` će nastaviti da radi sve dok uslov koji se izračunava ne postane false.

Kada se petlja prvi put pokrene, dobija se izraz water_level < 10. Pošto je promenljiva `water_level` inicijalizovana pomoću vrednosti 0, rezultat izračunavanja je true (1), pa će se kod ispod petlje zbog toga pokrenuti. Ako je rezultat false, programski tok će ignorisati ili "preskočiti" ceo blok koda unutar petlje i nastaviće da izvršava svaku naredbu koja se pojavi nakon toga. Prva linija koda koja radi u petlji dodaje promenljivoj `water_level` vrednost 1 i potom skladišti ovaj rezultat nazad u promenljivu `water_level (0 + 1 = 1)`. Ovaj rezultat se ispisuje u terminalskom prozoru.

Programski tok je sada dostigao dno petlje, pa se izraz `water_level < 10` ponovo izračunava. Vrednost promenljive `water_level` je 1, a 1 je manje od 10, pa je rezultat izračunavanja ponovo true i prva naredba unutar petlje je opet pokrenuta.

Ponovno pokretanje prve naredbe dovodi do toga da promenljiva `water_level` sada sadrži vrednost 2 (1 + 1 = 2). Kao i ranije, rezultat se ispisuje u terminalskom prozoru, a izraz petlje se ponovo izračunava.

Ovaj proces će nastaviti da pravi petlju i svaki put će povećavati promenljivu `water_level` za vrednost jedan kroz petlju i ispisivati rezultat u terminalskom prozoru.



Slika 4-1 Petlja while

Kada vrednost promenljive dostigne 9, izraz petlje će i dalje imati vrednost true. Nakon što je u sledećem prolazu kroz petlju došlo do povećanja ove promenljive, ona će imati vrednost 10 koja će biti ispisana u terminalskom prozoru. Izraz petlje se ponovo izračunava i dobija se vrednost false, zato što 10 nije manje od 10. Kod u petlji se sada neće ponovo pokrenuti. Programski tok nastavlja izvršavanje u naredbi `printf()` koja se nalazi ispod petlje `while`. Na slici 4-1 je prikazano kako funkcioniše petlja `while`. Nakon što se pokrenu dve naredbe u petlji, izraz će se ponovo izračunavati.

Poglavlje 5 • Funkcije

U ovom poglavlju ćete raditi sledeće:

- Naučiti kako da napišete funkcije
- Koristiti pretprocesorske naredbe
- Naučiti datoteke zaglavlja
- Videti kako su funkcije povezane sa datotekama povezivanja i biblioteke

Vi ste već koristili funkcije `printf()` i `scanf()`. Takođe, pisali ste funkciju svaki put kada ste pisali i C program – funkciju `main()`. Funkcija `printf()` se nalazi u GCC alatnom lancu koji se instalira sa Atmel Studiom. Stvarni C kod koji čini ovu funkciju je prilično dugačak i složen, zato što funkcija sadrži mnogo mogućnosti formatiranja teksta. Ako biste morali da ovaj kod napišete svaki put kada želite da tekst ispišete na ekranu, programi bi bili veoma veliki, pa je zato bolje napisati funkciju jer se kod za funkciju piše samo jednom, a funkcija se koristi više puta.

Funkcije se mogu iskoristiti i da biste program razdvojili u više datoteka. Ako se veliki program uskladišti u jednu datoteku, biće ga veoma teško čitati i održavati. Kada program podelite na funkcije, on se može razdvojiti u dve ili više datoteka. Kada program razdvajate u datoteke, slične funkcije će biti zajedno grupisane. Možete odlučiti da sve funkcije koje su povezane sa serijskom komunikacijom sačuvate u datoteku pod nazivom `serial.c`, a da funkcije koje su povezane sa tajmerima sačuvate u datoteku timer.c.

Kada je kod podeljen na više datoteka, može isto tako da se koristi više puta. Funkcije za određeni deo hardvera ili algoritma pišete samo jednom, a zatim u svakom programu koji želite da napišete dodajete datoteku koja sadrži funkcije. Na primer, mi smo u svim našim programima koristili funkciju `printf()`, a da pri tom nismo morali da pišemo funkciju `printf()`. Ako je `printf()` funkcija koju ste vi napisali, treba samo da je kreirate jednom i zatim možete da je koristite kada god to želite.

Tehnički termini
Algoritmi – Algoritam je metoda ili skup zadataka za rešavanje problema. Primer algoritma je kod koji se koristi za prepoznavanje otisaka prsta iz skenera za otisak prsta. Metod koji se u ovom kodu koristi za obradu, upoređivanje i prepoznavanje otiska prsta nazvaćemo algoritam za prepoznavanje otiska prsta.

5.1 • Vaša druga funkcija

Pošto ste već napisali funkciju main() svaki put kada ste pisali i C program, ovo će biti vaša druga funkcija koju ćete napisati. U programu contfunc prikazano je kako da napišete funkciju.

Naziv projekta: countfunc
Lokacija projekta: CH5\countfunc
main.c

```
#include <stdio.h>
#include "stdio_setup.h"

void Count(void);                          /* function prototype */

int main(void)
{
    UartInit();

    printf("Counting to 5\r\n");
    Count();                                /* call the function */
    printf("Counting to 5 again\r\n");
    Count();                                /* call the function */

    while(1) {
    }
}

/* the function */
void Count(void)
{
    int count = 1;                          /* count from 1 */

    while (count <= 5) {                    /* count to 5 */
        printf("%d\r\n", count);            /* display the count */
        count = count + 1;                  /* increment the count */
    }
}
```

Rezultat programa

```
Counting to 5
1
2
3
4
5
Counting to 5 again
1
2
3
4
5
```

U ovom primeru je kreirana funkcija pod nazivom `Count()`, pa je zatim dva puta pozvana u glavnom programu, kao što je prikazano na slici 5-1. Funkcija jednostavno ispisuje broj od 1 do 5. Blizu vrha izvorne datoteke se nalazi "prototip funkcije" koji se završava znakom tačka-zarez, tj. `void Count(void);`. Prototip funkcije ukazuje kompajleru da će se ta funkcija koristiti u programu, što je slično definisanju promenljive koja će se koristiti u programu. Funkcija `main()` je posebna funkcija koja ne zahteva prototip funkcije.

Stvarna funkcija je napisana ispo d završne vitičaste zagrade funkcije `main()`. Telo ove funkcije se nalazi između početnih i završnih zagrada. Ako u ovom primeru pro-

Poglavlje 6 • Brojevni sistemi

U ovom poglavlju ćete raditi sledeće:

- Naučiti zašto koristimo binarne brojeve
- Videti kako funkcioniše binarni brojevni sistem
- Naučiti da čitate i koristite heksadecimalne brojeve
- Koristite heksadecimalne brojeve u C programima
- ASCII alfanumerički kod

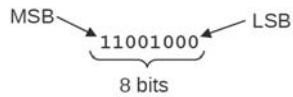
Sve u računaru ili mikrokontroleru, bilo to tekst, program, fotografija ili video, je uskladišteno kao binarni broj. Računar koristi samo binarne brojeve. Zbog toga je važno da da znate kako funkcionišu brojevni sistemi i u ovom poglavlju ću objasniti šta treba da znate o brojevima kao C programer ugrađenih sistema.

6.1 • Osnove binarnog sistema

Ovo je primer binarnog broja:

```
11001000
```

Za ovaj broj se kaže da je 8-bitni broj, jer se sastoji od 8 cifara ili bitova. Svaka binarna cifra je nula (0) ili jedan (1). Cifra koja se nalazi na samoj desnoj strani broja je poznata kao najmanje značajan bit (LSB) ili najmanje značajna cifra (LSD). Cifra koja se nalazi na samoj levoj strani broja je poznata kao najznačajniji bit (MSB) ili najznačajnija cifra (MSD). To je uvek slučaj, bez obzira od koliko bitova se sastoji broj.



Reč "bit" potiče od reči **b**inary dig**it** (binarna cifra).

Kada je reč o memoriji u računarima, ona se obično određuje u bajtovima, npr. 16 kilo bajta. **Bajt** se sastoji od 8 bitova. **Nibl** je pola bajta ili 4 bita. Dužina reči zavisi od arhitekture sistema na kojoj se primenjuje i, na primer, može da bude 8, 16 ili 32 bita. Reč dužine 32 bita se naziva 32-bitna reč.

1 bit = jedna binarna cifra i može da ima vrednot 0 ili 1
1 nibl = niz od 4 bitova
1 bajt = niz od 8 bitova
1 reč = zavisi od širine magistrale za podatke procesora

Kada se binarni broj piše u tekstu, obično se završava slovom 'B' da bi se razlikovao od decimalnog broja ili broja iz različitog brojevnog sistema.

i primeri:

1 bit	1B
1 nibl	1010B
1 bajt	00101111B
1 reč od 16 bitova	1000100100001101B

U nekim tekstovima se na kraju binarnog broja može koristiti malo slovo b, npr. 1001b. U drugim se može koristiti malo 2, npr. 11100101_2., koje predstavlja vrednost dve cifre (0 i 1) binarnog sistema.

6.2 • Potreba za binarnim brojevima

Brojevi koji su predstavljeni na računaru se sastoje od naponskih nivoa. Koriste se samo dva naponska nivoa: nulti naponi (0 V) i pozitivni napon, npr. pet volti (5 V). Binarni brojevi se sastoje od samo dve cifre, nula (0) i jedan (1). Binarna cifra 0 se skladišti u naponskom nivou od 0 volti, a binarna cifra 1 u pozitivnom naponu, na primer, u 5 V.

Ako biste otvorili memorijski čip koji je priključen na računarski sistem i koristili voltmetar (ili multimetar postavljen na napon) da biste izmerili svaku binarnu cifru u čipu, mogli biste da zapišete ono što je uskladišteno u njemu. Svaki put kada izmerite 5 V zapisali biste 1, a svaki put kada izmerite 0 V zapisali biste 0.

Dakle, niz merenja: 5V, 0V, 0V, 5V, 0V, 0V, 0V, 5V bi predstavljao binarni broj: 10010001

Stvarni napon koji se koristi za predstavljanje binarne cifre 1 zavisi od sistema koji se koristi. Ranije se u digitalnim sistemima za predstavljanje binarne cifre 1 uvek koristio napon od 5 V. Mnogi savremeni mikroprocesori rade na nižim naponima, kao što je 1,8 V. U ovim sistemima binarna cifra 1 je predstavljena naponom od 1,8 V. Memorija koja je povezana sa jednim od ovih sistema bi onda morala da se sastoji od 1,8 V memorijskih čipova. AVR mikrokontroleri na Arduino Uno i MEGA pločicama su u stanju da rade pri naponima ispod 5 V, ali napajanje na ovim Arduino pločicama je postavljeno na 5 V. Na ovim Arduino pločicama binarna cifra 1 bi bila predstavljena naponom od 5 V.

6.3 • Brojevni sistemi

6.3.1 • Kratak pregled decimalnih brojeva

Da biste razumeli kako funkcionišu binarni brojevi, pogledaćemo kratak pregled decimalnog brojevnog sistema.

Trebalo bi da ste upoznati sa decimalnim brojevnim sistemom. Decimalni brojevi se sastoje od deset cifara 0 do 9 (1 do 9 = 9 cifara, 0 do 9 = 10 cifara), pa se zato za decimalni sistem kaže da ima bazu deset. Ako upotrebimo samo deset cifara, to nas ne sprečava da brojimo više od devet. Kada brojimo do vrednosti koja je veća od 9, koristimo dve ili više cifri. Položaj svake cifre u broju ukazuje na veličinu cifre. To je poznato kao ponderisani brojni sistem – na primer, broj 44 sadrži iste dve cifre (znakove), ali njihov položaj u broju određuje vrednost svake cifre.

Težina svake cifre je "stepen broja 10" koji se povećava sleva nadesno, sa prvom cifrom koja sa desne strane ima vrednost 1. U primeru za broj 44, prva cifra sa desne strane ima težinu 1, a druga cifra sa desne strane ima težinu 10:
4 × 10 +
4 × 1
= 44 (4 desetke i 4 jedinice)
"Stepen broja 10" je matematički izraz i piše se kao:

Poglavlje 7 • Memorija i mikrokontroleri

U ovom poglavlju ćete raditi sledeće:

- Saznati kako funkcioniše memorija
- Naučiti kako mikroprocesor pristupa memoriji i periferijama
- Videti kako da pristupite memoriji pomoću adrese memorije
- Naučiti druge tipove podataka u C jeziku

7.1 Osnove memorije

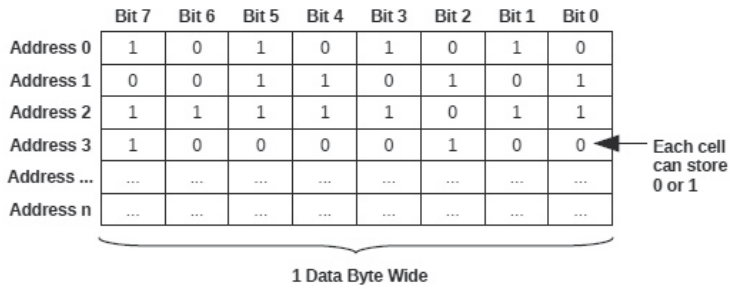
U mikrokontrolerima i ugrađenim sistema prisutna su dva glavna tipa memorije, koji se zovu ROM i RAM. Ovi tipovi memorije se koriste za skladištenje instrukcija programa iz kompajliranog C programa i promenljive podataka koje su definisane u C programu.

7.1.1 ROM (Memorija samo za čitanje)

Ovaj tip memorije se koristi za skladištenje programa koji je napisan i kompajliran. To je trajna memorija, što znači da zadržava svoj sadržaj, čak, i kada se isključi napajanje. ROM memorija je na savremenim mikrokontrolerima, poput Arduino AVR mikrokontrolera, prisutna u obliku Flash memorije. Sadržaj Flash memorije se obično ne menja kada se pokrene aplikacijski program korisnika, iako se Flash memorija nekih mikrokontrolera može programirati unutar aplikacije. Program koji je sačuvan u ROM memoriji će biti program koji je pokrenut kada se napajanje isporuči mikrokontroleru.

7.1.2 RAM (Radna memorija)

RAM se koristi za skladištenje promenljive i podataka iz našeg programa koji se mogu menjati tokom rada programa. Ovaj tip memorije nije postojan, što znači da će se sadržaj čipa izgubiti kada se ukloni napajanje. Mikrokontroleri, poput AVR mikrokontrolera, obično, sadrže nešto SRAM memorije (statički RAM).



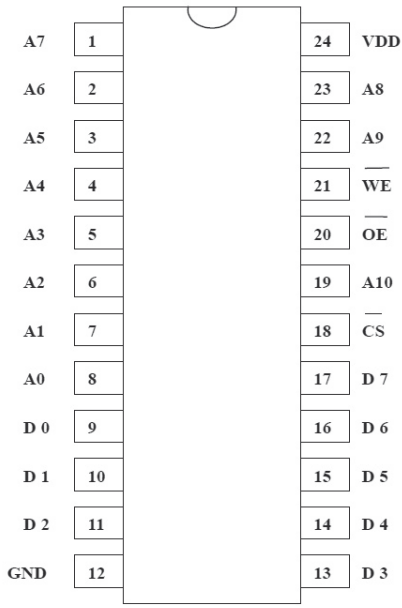
Slika 7-1 Prikaz skladištenja podataka u memoriju

7.1.3 Podaci uskladišteni u memoriju

Podaci se skladište u memoriju kao niz bajtova. Svaki bajt ima jedinstvenu adresu i može pojedinačno da se adresira. Podaci se skladište u memoriju pomoću operacije upisivanja i preuzimaju iz memorije pomoću operacije čitanja. Svaki bajt memorijskog čipa se naziva ćelija i može da sadrži logički nivo 1 ili 0, kao što je prikazano na slici 7-1. Kapacitet memorije zavisi od toga koliko adresabilnih bajtova sadrži.

7.2 Pregled memorijskog čipa

Hajde da pogledamo kako radi SRAM memorijski čip da bismo bolje razumeli memoriju. Na slici 7-2 je prikazan gornji deo memorijskog čipa od 2 kB (dva kilo bajta). Na slici 7.3 prikazan je blok dijagram istog čipa. Veličina memorije zavisi od toga koliko bajtova se može adresirati. To se određuje brojem reda za adresu na čipu. Čip ima 11 redova za adresu, pa može da adresira 2_{11} bajtova, što je jednako 2048 bajtova (ili 2 kilo bajta, zato što je 1 kilo bajt = 1024 bajtova ili 2_{10}).



VDD and GND:	Power supply to chip
A0 to A10:	Address bus – data byte address placed here
D0 to D7:	Data bus – 8 bit data byte read and written here
$\overline{CS}$:	Chip select
$\overline{OE}$:	Output enable } Control bus
WE:	Write enable

Slika 7-2 8-bitni SRAM memorijski čip

Magistrala je samo grupa žica na računarskom sistemu. Magistrala za adresu je grupa žica koje se koriste za adresiranje memorije i periferijskih uređaja i na prikazanom memorijskom čipu je široka 11 bitova. Magistrala za podatke je grupa žica koja se koristi za slanje i primanje podataka u memoriju i iz memorije i periferijskih uređaja. Magistrala za podatke na memorijskom čipu koja je prikazna na slici je široka 8 bitova. Kontrolna magistrala je grupa žica koje se koriste za kontrolu na memorijskom čipu ili periferijskom uređaju. Na memorijskom čipu postoje 3 kontrolne linije. Periferijski uređaj je neki deo hardvera koji je povezan sa mikrokontrolerom, bilo da je to interni i eksterni uređaj za mikrokontroler.

Poglavlje 8 – Pristup AVR portovima u C programu

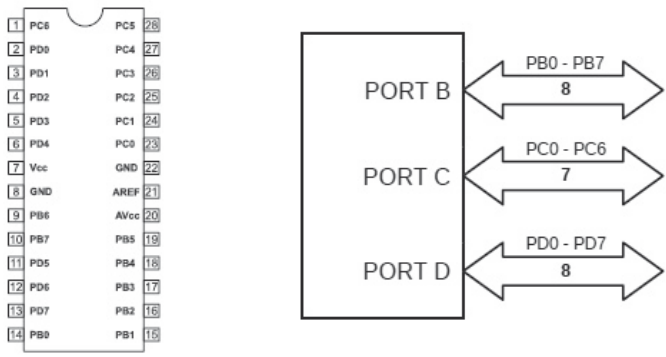
U ovom poglavlju ćete raditi sledeće:

- Upisivati u AVR hardverske registre pomoću pokazivača
- Postaviti port AVR mikokontrolera kao izlazni port
- Uključivati i isključivati LED-ove pomoću AVR registra
- Naučiti operator za povećanje i umanjenje
- Započeti novi Atmel Studio projekat od početka bez upotrebe šablona

8.1 • AVR pinovi i portovi

AVR mikrokontroleri poseduju veliki broj pinova koji su poznati kao ulazno/izlazni (I/O) pinovi opšte namene. Ove pinove možete pojedinačno konfigurisati kao izlazne pinove za učitavanje priključka – npr. za pokretanje ili uključivanje i isključivanje LED-a; ili kao ulazne pinove za čitanje logičkog nivoa smeštenog na pinu – npr. da biste testirali da li je priključeni prekidač otvoren ili zatvoren. Osim toga, ovi pinovi imaju alternativne funkcije u kojima mogu biti konfigurisani za povezivanje sa unutrašnjim hardverskim uređajem, kao što je UART, koji se koristi za serijsku komunikaciju ili ADC za čitanje analognih vrednosti. U ovom poglavlju ćemo pogledati konfiguracione pinove kao izlaze za pokretanje LED-a.

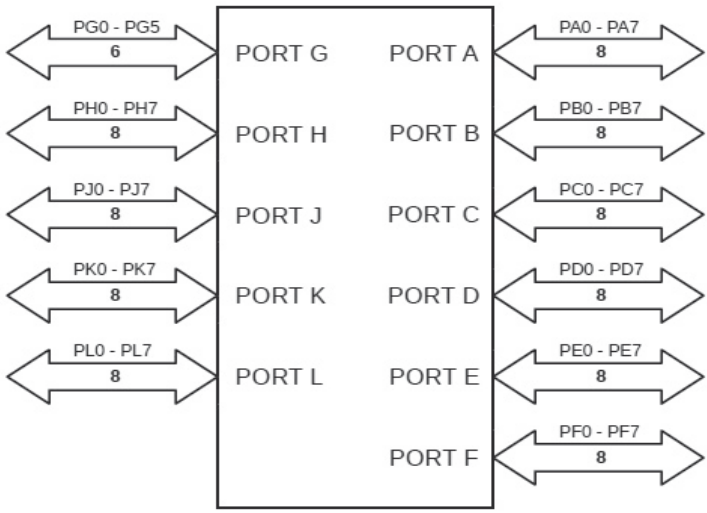
AVR 8-bitni mikrokontroleri poseduju 8-bitne portove koji predstavljaju grupe I/O pinova imenovane slovima – npr. port A, port B, port C itd. Svaki port kontroliše skup registara pomoću kojih možete da konfigurišete pinove kao ulaze ili izlaze. Registri su kao memorijske lokacije i zato mogu da budu napisani radi korišćenja pokazivača u C programu. Stvarne nazive i funkcije registara možete pronaći u tabeli sa podacima mikrokontrolera.



Slika 8-1 Numerisanje pinova i portovi mikrokontrolera Atmega328P

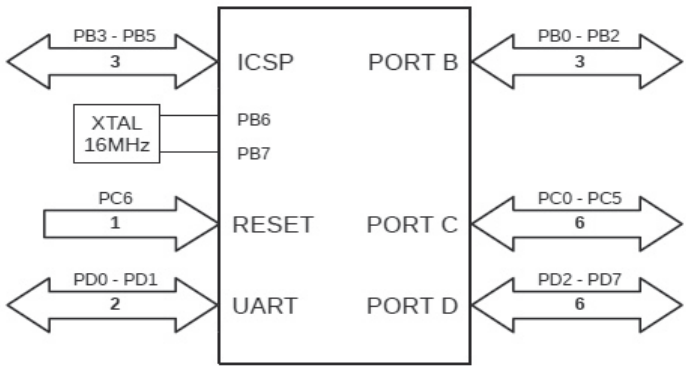
Na slici 8-1 je prikazan gornji deo fizičkog pakovanja mikrokontrolera Atmega328P koji se nalazi na pločici Arduino Uno koja prikazuje numerisanje pinova (od pina 1 do pina 28) i pojedinačno numerisanje pinova koji čine portove, odnosno PB0 od PB7, PC0 do PC6 i PD0 do PD7. Na desnoj strani slike je prikazan blok dijagram sa portovima B i D koji su 8 bitova široki i portom C koji je širok 7 bitova.

Na slici 8-2 prikazan je blok dijagram pinova i portova mikrokontrolera Atmega2560 koji se nalazi na pločici Arduino MEGA 2560.



Slika 8-2 Portovi mikrokontrolera Atmega2560

Na gornjoj slici su prikazani nekorišćeni pinovi mikrokontrolera, ali, kada su mikrokontroleri postavljeni na pločicu Arduino, neki pinovi porta se koriste za određene funkcije, kao šo je serijski port. Na slici koja sledi prikazani su pinovi porta koji se koriste za namenske funkcije na Arduino pločicama i pinovi koji su slobodni za opštu namenu.



Slika 8-3 ATmega328P na Arduino Uno-u koji sadrži korišćene pinove (L) i slobodne pinove (R).

Na slici 8-3 sa desne strane prikazani su pinovi porta mikrokontrolera Atmega328P koji su korišćeni za Arduino Uno pločicu, a na levoj strani prikazani su pinovi koji su dostupni za opštu namenu. PB3, PB4 i PB5 se koriste za ICSP zaglavlje zajedno sa pinom za resetovanje PC6. PC6, odnosno pin za resetovanje, je povezan sa ICSP zaglavljem i dugmetom za resetovanje na pločici. ICSP zaglavlje je zaglavlje na koje je povezan uređaj za programiranje, kao što je AVRIPS ili AVRD Dragon. Pin PB5 za ICSP zaglavlja je, takođe, povezan sa LED-om označenim slovom L na pločici koji vodi do Arduino pina 13.

Poglavlje 9 • I/O i mape memorije

U ovom poglavlju ćete raditi sledeće:

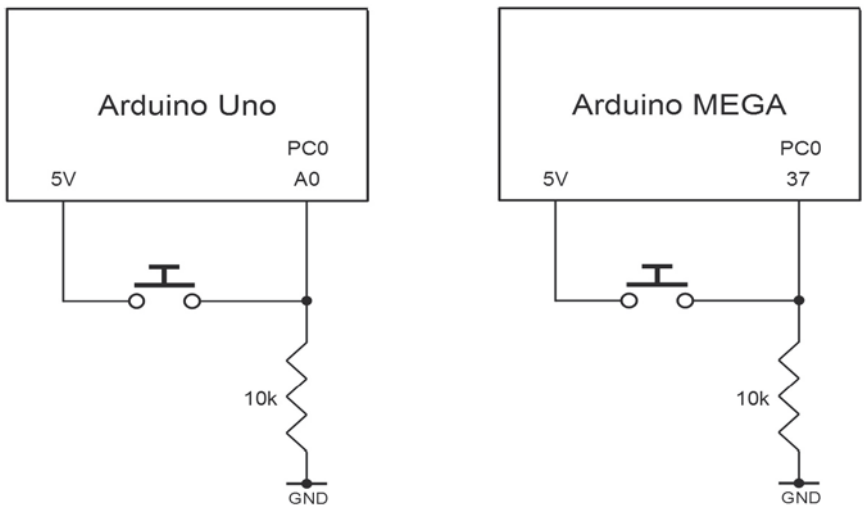
- Očitavati stanje prekidača koji je povezan sa ulaznim pinom mikrokontrolera
- Koristiti ulazne i izlazne pinove
- Videti kako se hardver i memorija mapiraju u AVR mikrokontrolerima

9.1 • Ulazni pinovi

U prethodnom poglavlju smo koristili izlaz samo tako što smo vrednost poslali izvan mikrokontrolera da bismo uključili i isključili LED-ove koji su povezani sa pinovima na AVR portu. Pinovi porta su podešeni kao izlazni pinovi i korišćeni su za pokretanje LED-ova. Pinovi porta mogu, takođe, da budu podešeni kao ulazi, a logička stanja ovih pinova može da očita mikrokontroler. Prekidač koji je povezan sa pinom mikrokontrolera podešenim kao ulaz može da se očita kako biste videli da li je prekidač zatvoren ili otvoren.

9.1.1 • Povezivanje prekidača sa pinom porta

Da biste očitali stanje prekidača u C programu, prekidač prvo mora da povežete sa željenim pinom porta AVR mikrokontrolera na Arduinu. Prekidač koji se koristi za ovaj primer može da bude prekidač koji treba samo da zatvori kolo, a zatim da ga otvori. Za zatvaranje i otvaranje kola može da se koristi tasterski prekidač, sklopka, ili, čak, i žica.



Slika 9-1 Povezivanje prekidača sa pinom PC0 na pločicama Uno (levo) i MEGA (desno)

Na slici 9-1 prikazan je prekidač koji je povezan sa pinom porta PC0 na pločicama Arduino Uno i MEGA, čija se dva kola razlikuju samo po Arduino broju pina sa kojim je povezan prekidač. Zajednički GND na jednom kraju otpornika povezan je sa GND priključkom na Arduinu.

Kada je prekidač otvoren, pull-down otpornik od 10 k povezuje pin PC0 sa GND-om ili napajanjem od 0 V, tako što na pinu postavlja logički nivo 0. Kada je prekidač zatvoren, napajanje u pinu se smanjuje za 5 V, pa se na pinu postavlja logički nivo 1. Otpornik od 10 k sprečava da se napajanje u GND-u smanji za 5 V.

9.1.2 ● Očitavanje stanja prekidača u C programu

Kreirajte novi Atmel Studio projekat pod nazivom **switch_val** tako što ćete koristiti datoteku šablona i povezati prekidač sa Arduinom pomoću prototipske ploče, kao što je prikazano u kolu na slici 9.1. Dodajte sledeći kod u projekat.

Naziv projekta: switch_val
Lokacija projekta: CH9\switch_val
main.c

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include "stdio_setup.h"

int main(void)
{
    DDRC = 0x00;                              // set port C pins as outputs
    UartInit();

    while(1) {
        printf("PORT C: 0x%02x\r\n", PINC);   // print the port state
        _delay_ms(200);
    }
}
```

Rezultat programa

```
PORT C: 0x00
PORT C: 0x01
PORT C: 0x01
PORT C: 0x01
PORT C: 0x00
PORT C: 0x00
...
```

Kada je prekidač otvoren, vrednost porta C koja je prikazana u terminalskom prozoru je 0x00. Kada je prekidač zatvoren, prikazana vrednost je 0x01. Na početku programa DDRC-u se dodeljuje vrednost 0x00 da bi se svi pinovi porta na portu C konfigurisali kao ulazi. To, u stvari, nije neophodno, zato što su pinovi porta podrazumevano konfigurisani kao ulazi, ali to pokazuje nameru programera da će se ceo port koristiti kao ulazni port, ali mi ćemo u primeru programa koristiti samo jedan pin ulaznog porta.

U petlji `while(1)` vrednost PINC registra se ispisuje u terminalskom programu koji radi na računaru. Vrednost iz ovog registra se koristi direktno u naredbi `printf()`, pa ne morate prvo da je očitavate u privremenu promenljivu. U naredbi `printf()`se nalazi zakašnjenje od 200 ms koje sprečava terminalski program da bude zatrpan velikim brojem uzastopnih poruka.

Poglavlje 10 • Prethodne revidirane teme o C programiranju

U ovom poglavlju ćete raditi sledeće:

- Pogledati detaljnije prethodne teme o C programiranju
- Saznati više o specifikatorima formata, specifikatorima širine polja i izlaznim sekvencama
- Dodatne petlje – predstavljene su petlje `do-while` i `for`
- Programiranje AVR USART / serijskog porta
- Ugnežđene petlje i odluke
- Donošenje odluka pomoću konstrukcije `switch` i uslovnog operatora
- Funkcije i pokazivači
- Opseg primene promenljivih i statičke promenljive
- Saznati više o promenljivama sa pokretnom tačkom
- Konverzije u C programu

10.1 • Specifikatori formata

U tabeli 10-1 prikazani su specifikatori formata u C programu, a vi ste već koristili neke od njih. U programu **formats** se koriste neki od novih formata specifikatora.

Naziv projekta: formats
Lokacija projekta: CH10\formats

main.c

```
#include <stdio.h>
#include "stdio_setup.h"

int main(void)
{
    UartInit();

    /* characters and strings */
    printf("%s %c\r\n", "This is the letter", 't');
    /* integers */
    printf("Ints:%d, %i, %u\r\n", -1000, -123, 958);
    printf("Oct:%o, hex:%x, HEX:%X\r\n", 191, 191, 191);
    /* floating point numbers */
    printf("%f, %e, %E\r\n", 45.345, 45.345, 45.345);

    while(1) {
    }
}
```

Rezultat programa

```
This is the letter t
Ints:-1000, -123, 958
Oct:277, hex:bf, HEX:BF
45.345001, 4.534500e+01, 4.53400E+01
```

Već ste videli specifikator formata %d za ispisivanje celih decimalnih brojeva, specifikator %d radi to isto, a specifikator %u koristi neoznačene cele brojeve. Specifikatori formata %x i %X se koriste za ispisivanje broja u oktalnom formatu.

Oktalni sistem je još jedan brojevni sistem poput binarnog ili heksadecimalnog, ali koristi bazu 8. Specifikator formata %f treba da vam je poznat, a koristi se za ispisivanje broja sa pokretnom tačkom. Specifikatori %e i %E ispisuju broj u eksponencijalnoj notaciji, notaciji koju koriste naučnici i inženjeri.

Tabela 10-1 Specifikatori formata u C programiranju

Specifikator formata	Tip podataka	Format ispisa
%d ili %i	`int`	Decimalni broj
%u	`unsigned int`	Neoznačeni decimalni broj
%x	`unsigned int`	Heksadecimalno malo slovo
%X	`unsigned int`	Heksadecimalno veliko slovo
%o	`unsigned int`	Oktalni neoznačeni broj
%c	`char or int`	Znak
%s	`string`	String znakova
%f	`float`	Broj sa pokretnom tačkom
%e	`float`	Veliko slovo eksponencijalne notacije
%E	`float`	Malo slovo eksponencijalne notacije

10.2 ● Revidirani specifikatori širine polja

Specifikatore širine polja smo već koristili za brojeve sa pokretnom tačkom da bi broj cifara ispisali posle decimalne tačke. Broj cifara može, takođe, da se postavi pre decimalne tačke, ili kod decimalnih i heksadecimalnih brojeva da se broj čuvara mesta postavi sa leve strane broja. Kao programer ugrađenih sistema, možda želite da koristite ovaj specifikator da biste prikazali širinu broja kada koristite heksadecimalne brojeve ili da biste poravnali brojeve. Na primer, broj 0x5 je 3-bitni broj (101_2), ali u mikrokontroleru ili računaru memorija se uvek prikazuje u više bajtova, pa možda želimo da ovaj broj prikažemo tako što ćemo ga dopuniti nulom da bismo predstavili svih 8 bitova - 0x05 ($0000\ 0101_2$). Sledeći program, pod nazivom **padding**, prikazuje kako da različite brojeve dopunite nulama ili razmacima. Kreirajte program tako što ćete koristiti datoteku šablona.
Naziv projekta: padding
Lokacija projekta: CH10\padding
main.c

```
#include <stdio.h>
#include "stdio_setup.h"

int main(void)
{
    UartInit();

    // a 3 bit hex number padded to 8 bits
    printf("0x%X\n", 0x5);          // unpadded
    printf("0x%2X\n", 0x5);         // pad with spaces
    printf("0x%02X\n\n", 0x5);      // pad with 0

    // a 3 bit hex number padded to 16 bits
    printf("0x%4X\n", 0x5);         // pad with spaces
    printf("0x%04X\n\n", 0x5);      // pad with 0
```

Poglavlje 11 • Nizovi i stringovi

U ovom poglavlju ćete raditi sledeće:

- Videti kako se nizovi koriste za pristup sekvencijalnim memorijskim lokacijama
- Videti kako se nizovi koriste i skladište u memoriju
- Koristiti funkcije stringa C standardne biblioteke
- Napisati funkcije koje uzimaju stringove i nizove kao argumente

Već ste radili sa stringovima u C jeziku kada ste koristili funkciju `printf()`. C jezik ne sadrži tekstualne tipove podataka, ali umesto toga koristi nizove znakova. Niz znakova je jednostavno veliki broj uzastopnih znakova u memoriji. U C jeziku string je niz znakova, ali, isto tako, moguće je koristiti niz različitih tipova podataka. Prvo ću uopšteno objasniti nizove i videćemo kako se definišu i koriste nizovi u C programiranju. Zatim ćemo detaljnije pogledati stringove i neke funkcije standardne biblioteke koje se koriste za upravljanje tim stringovima.

11.1 • Nizovi

Niz je skup uzastopnih memorijskih lokacija koje skladište podatke određenog tipa. Ono što je ovde važno je princip *uzastopno*, jer to je ono što čini nizove mogućim. Na primer, u C jeziku niz od 5 celih brojeva je definisan na sledeći način:

```
int arr_num[5];
```

To ukazuje kompajleru da treba da oslobodi prostor za niz koji sadrži pet celih brojeva. Svaki element (ceo broj) je niz kome se može pristupiti pomoću notacije niza u C jeziku. Ako svakom elementu u nizu želimo da dodelimo vrednost, to možemo da uradimo na sledeći način:

```
arr_num[0] = 20;
arr_num[1] = 34;
arr_num[2] = 9;
arr_num[3] = 1023;
arr_num[4] = 700;
```

U nizu postoji pet elemenata, ali oni su indeksirani počev od 0, pa im se pristupa pomoću indeksa od 0 do 4, a ne od 1 do 5.

Definicija niza:

```
int arr_num[5];
```

slična je definisanju 5 odvojenih promenljivih celog broja:

```
int arr_num0, arr_num1, arr_num2, arr_num3, arr_num4;
```

Definisanje odvojenih promenljivih, ipak, ne garantuje da su one poređane redom u memoriji. Odvojene promenljive možda nisu referencirane pomoću notacije niza koja, kao što ćemo videti, je veoma korisna.

Program **index_array** inicijalizuje niz celih brojeva, a zatim koristi petlju for da bi se kretao kroz listu celih brojeva i prikazao ih u serijskom portu.

Naziv projekta: index_array
Lokacije projekta: CH11\index_array

main.c

```
#include <stdio.h>
#include "stdio_setup.h"

int main(void)
{
    int num_list[5] = {34, 789, 327, 1000, 45};
    int index;

    UartInit();

    for (index = 0; index < 5; index++) {
        printf("Element %d = %d\r\n", index, num_list[index]);
    }
    while (1);
}
```

Rezultat programa

```
Element 0 = 34
Element 1 = 789
Element 2 = 327
Element 3 = 1000
Element 4 = 45
```

Prvi programski red unutar funkcije `main()` definiše niz od 5 celih brojeva i inicijalizuje svaki od ovih celih brojeva pomoću vrednosti koja se pojavljuje između zagrada posle znaka =. Inicijalizacija ima isti efekat kao i kada biste svakom elementu niza dodelili vrednosti na sledeći način:

```
int num_list[5];
num_list[0] = 34;
num_list[1] = 789;
num_list[2] = 327;
num_list[3] = 1000;
num_list[4] = 45;
```

Petlja `for` se koristi za brojanje od 0 do 4. Unutar ove petlje se ispisuje trenutna vrednost brojanja (promenljiva `index`) i svaki element niza. Kada svaki element u nizu referenciramo pomoću trenutnog brojanja od 0 do 4, pristupamo svakom elementu u nizu od 0 do 4.

Kada indeks sadrži vrednost 0, onda `num_list[index]` preuzima vrednost prvog elementa u nizu, tj. 34 u ovom primeru. Nakon što je promenljiva `index` uvećana za 1, `num_list[index]`, koji je jednak `num_list[1]`, sada ima vrednost drugog elementa u nizu, tj. 789. Zapamtite da brojimo od 0, pa je prvi element u nizu indeksiran na poziciji 0, drugi element na poziciji 1, treći element na poziciji 2, itd. Na slici 11-1 prikazan je dijagram niza koji je korišćen u ovom primeru, kao što se pojavljuje u memoriji.

Poglavlje 12 • Upravljanje bitom i logički operatori

U ovom poglavlju ćete raditi sledeće:

- Videti kako možete da promenite pojedinačne bitove unutar promenljive pomoću operatora nad bitovima
- Pomeriti sadržaj promenljive ulevo ili udesno pomoću operatora pomeranja
- Koristiti logičke operatore da biste kombinovali izraze

12.1 • Upravljanje bitom pomoću operatora nad bitovima

Operatori nad bitovima omogućavaju programeru da modifikuje pojedinačne bitove u memorijskoj lokaciji i mogu da se koriste za rad sa bajtovima ili većim grupama bitova. Programski jezik C sadrži četiri Bulova operatora nad bitovima i dva operatora pomeranja, kao što je prikazano u tabeli 12-1. Operator nad bitovima AND (&) bi trebalo da vam bude poznat.

Tabela 12-1 Bulovi operatori nad bitovima i operatori pomeranja

Bulovi operatori nad bitovima			
Operator	Operacija	Primer	Rezultat
&	AND	1101 & 0111	0101
|	OR	1101 | 1001	1101
^	XOR (Exclusive OR)	1101 ^ 1001	0100
~	NOT (Compliment)	~1101	0010
Operatori pomeranja			
Operator	Operacija	Primer	Rezultat
<<	Left Shift	1010 << 1	10100
>>	Right Shift	1010 >> 1	0101

Windowsov kalkulator će ponovo biti od koristi dok budete učili operatore nad bitovima, jer sadrži sve dostupne operatore nad bitovima. Na slici 12-1 prikazan je Windowsov kalkulator, a ispod je prikazan ekvivalent C operatora nad bitovima.

Operatore nad bitovima možete da testirate pomoću kalkulatora u binarnom ili heksadecimalnom režimu, a možete i da menjate ove režime tokom obavljanja operacije. Tasteri koji se koriste za izračunavanje primera iz tabele 12-1 na kalkulatoru su prikazani u nastavku od vrha ka dnu – prvo postavite kalkulator u režim **Programmer**, a zatim kliknite radio-dugme **Button**. Objašnjenje svakog operatora nad bitovima sledi u nastavku.

1101 And 111 =

Napominjemo da se čuvar mesta za 0 sa leve strane drugog broja izostavlja prilikom unosa drugog broja i ne prikazuje se u rezultatima na kalkulatoru, tj. prikazani rezultat je 101, a ne 0101.

1101 Or 1001 =
1101 Xor 1001 =
1101 Not

Rezultat ove operacije na kalkulatoru je da sve nevidljive čuvare mesta za 0 sa leve strane broja pretvara u jedinice i zato je broj sa svoje leve strane ispunjen jedini-

cama. Kliknite radio-dugme **Button** da biste skratili broj za 8 bitova, jer se tako lakše može pročitati.

1010 Lsh 1 =

(Pomeranje ulevo za 1 bit)

1010 Rsh 1 =

(Pomeranje udesno za 1 bit)



Slika 12-1 Operacije nad bitovima na Windowsovom kalkulatoru

U Windowsovom kalkulatoru dugmadi imaju sledeća značenja uz ekvivalente operatora u sredini, koji se koriste u C jeziku:
Or | operator nad bitovima – simbol koji se koristi u C jeziku je vertikalna crta koja se nalazi na tasteru \
Xor ^ operator nad bitovima XOR (ekskluzivni OR)
Lsh << pomeranje ulevo
Rsh >> pomeranje udesno
Not ~ komplementiranje
And & operator nad bitovima AND

Iako operatori **RoL** (rotiranje ulevo) i **RoR** (rotiranje udesno) rade sa bitovima u Windowsovom kalkulatoru, oni nemaju ekvivalent u C jeziku.

Da biste razumeli kako funkcionišu Bulovi operatori nad bitovima, morate da pogledate tabele istinitosti ovih operatora. Tabele istinitosti su prikazane na slici 12-2. U tim tabelama je prikazan rezultat svakog operatora nad bitovima kada se koristi sa ulaznim bitovima.

& (AND)		
Inputs		Output
0	0	0
0	1	0
1	0	0
1	1	1

| (OR)		
Inputs		Output
0	0	0
0	1	1
1	0	1
1	1	1

^ (XOR)		
Inputs		Output
0	0	0
0	1	1
1	0	1
1	1	0

~ (NOT)	
Input	Output
0	1
1	0

Slika 12-2 Tabele istinitosti za Bulove operatore nad bitovima

Poglavlje 13 • Programiranje hardvera

U ovom poglavlju ćete raditi sledeće:

- Koristiti hardver / tajmerski brojač u AVR-u da biste odredili precizno vreme kašnjenja
- Očitavati eksterne analogne vrednosti pomoću analogno-digitalnog pretvarača
- Videti kako funkcioniše nadzorni tajmer
- Detaljnije pogledati I/O portove

Do sada smo pokrili veći deo C jezika. U ovom poglavlju ćemo iskoristiti ovo znanje za programiranje više hardverskih uređaja na mikrokontroleru, po imenu tajmeri, kao što su nadzorni tajmer i analogno-digitalni pretvarač (ADC). Takođe ćemo detaljnije pogledati I/O portove. Kada savladate C jezik, samo treba da naučite kako funkcioniše novi hardver i da ga programirate da biste prešli sa jednog ugrađenog sistema na drugi ili sa jedne arhitekture mikrokontrolera na drugu.

Pošto je tema ove knjige učenje C jezika, neću detaljno objašnjavati hardver mikrokontrolera, već ću pokazati kako se podešavaju i koriste različiti hardverski uređaji.

13.1 Hardverski tajmer/brojač

Tajmerski brojači (TC) u AVR mikrokontrolerima mogu da se koriste za više namena, kao što je brojanje događaja, generisanje impulsa, podešavanje vremena kašnjenja i još mnogo štošta. Više informacija o tajmerima/brojačima možete pronaći u AVR tabeli sa podacima za određeni AVR mikrokontroler koji se koristi. Mikrokontroleri na pločicama Arduino Uno i MEGA 2560 imaju sledeći ugrađeni čip tajmera/brojača.

Arduino Uno (ATmega328P)

- 8-bitni Timer/Counter0
- 16-bitni Timer/Counter1
- 8-bitni Timer/Counter2

Arduino MEGA 2560 (ATmega2560)

- 8-bitni Timer/Counter0
- 16-bitni Timer/Counter1
- 8-bitni Timer/Counter2
- 16-bitni Timer/Counter3
- 16-bitni Timer/Counter4
- 16-bitni Timer/Counter5

Hardverski tajmer funkcioniše tako što sadrži taktni impuls koji izaziva brojanje svakog taktnog ciklusa. Vrednost brojanja se skladišti u jednom od tajmerskih registara koji ažurira hardver u svakom taktnom ciklusu. C program može da očitava 16-bitnu vrednost u 16-bitnom tajmeru ili 8-bitnu vrednost ako je to 8-bitni tajmer. Ako u tajmeru imamo radni takt od 1 Hz, onda bi do brojanja došlo svake sekunde, zato što je 1 Hz jedan taktni ciklus po sekundi. Vremenski period od 5 s

možete da generišete tako što ćete pokrenuti tajmer i pratiti kada tajmer dostigne brojanje do 5. Mikrokontroleri obično imaju izvore radnog takta koji su mnogo brži od 1 Hz i najčešće se izvode iz sistemskog radnog takta koji radi na MHz frekventnom opsegu.

Ako se sistemski radni takt od 16 MHz uvede u 16-bitni tajmer, taj tajmer će prekoračiti ograničenje posle približno 4 ms. Prekoračenje nastaje kada tajmer broji od nule do maksimalne vrednosti 65535 i zatim ponovo prelazi na nulu. Da biste u 16-bitnom tajmeru odredili duži vremenski period, koristite preskaler da biste sistemski radni takt podelili na manju vrednost pre nego što ga uvedete u tajmer. Preskaler može da se omogući iz jednog od registara tajmera / brojača i sadrži veliki broj različitih vrednosti koje mogu da podele sistemski radni takt.

Trajanje ili vremenski period taktnog ciklusa može da se izračuna pomoću sledeće formule:

$$t = \frac{1}{f}$$

U kojoj je:
t = vremenski period
f = frekvencija radnog takta

Za radni takt od 1 Hz izračunavanje je veoma jednostavno:

$$t = \frac{1}{1\,Hz} = 1s$$

Period radnog takta od 16 MHz se izračunava na sledeći način:

$$\begin{aligned} t &= \frac{1}{f} \\ &= \frac{1}{16\,MHz} \\ &= \frac{1}{16,000,000} \\ &= 0.0000000625 \\ &= 62.5\,ns \end{aligned}$$

Radni takt od 16 MHz ima period od 62,5 nanosekundi.

U tajmeru možemo lako da generišemo određeni vremenski period kada znamo koliko dugo traje jedan taktni ciklus. Tajmer će povećati svaki taktni ciklus ili taktni period, pa je zato: $t_{timer}=t_{period}$ *x count*

U kojoj je:
t_{timer} = *generisani period*
count = *brojanje tajmera*
t_{period} = *taktni period ulaska takta u tajmer*

Tajmer može zato da uđe u više taktnih perioda iz taktnog izvora koji koristi.

Poglavlje 14 • Debagiranje

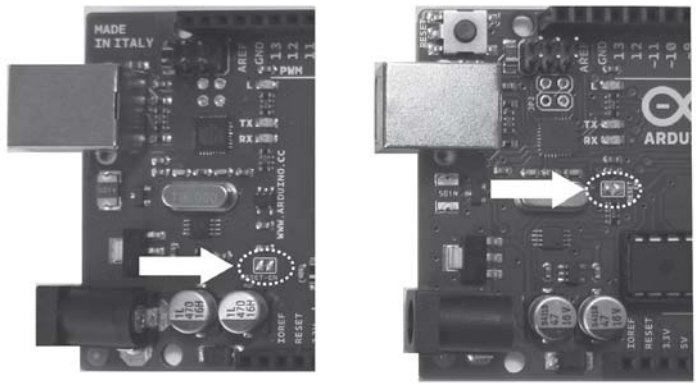
U ovom poglavlju ćete raditi sledeće:

- Koristiti AVR Dragon ili simulator u kombinaciji sa Atmel Studiom da biste debagirali program
- Kretati se korak po korak kroz izvorni kod u Atmel Studiu
- Pogledati unutar promenljivih, registara i memorije pomoću alatki za debagiranje

AVR Dragon koji se koristi u kombinaciji sa Atmel Studiom može da se kreće korak po korak kroz C kod u projektu da bi pronašao greške kao deo procesa debagiranja. Sadržaj promenljivih, memorije i registara može da se ispita pomoću AVR Dragon-a. Ako nemate AVR Dragon, možete da koristite AVR simulator, ali promene u spoljnom hardveru, kao što su LED-ovi koji se uključuju, ne možete videti pomoću simulatora.

14.1 • Konfigurisanje Arduino pločica radi debagiranja

AVR mikrokontroleri na Arduino Uno i MEGA pločicama imaju sistem za debagiranje na čipu pod nazivom debugWIRE, koji povezuje softver za debagiranje sa AVR-om preko uređaja, kao što je AVR Dragon povezan sa ICSP zaglavljem. Da bi debugWIRE radio na Arduino pločicama AVR mikrokontrolera, kolo za resetovanje mora da bude isključeno iz pina za resetovanje. Savremene Arduino pločice sadrže vezu za lemljenje sa oznakom RESET-EN, koja se sastoji od dva jastučića za lemljenje koji su povezani pomoću dva štampana strujna voda. Kada se koristi debugWire, vod mora biti prekinut da bi se deo kola za resetovanje isključio iz pina za resetovanje.



Slika 14-1 Lokacija RESET-EN jastučića za lemljenje na Arduino MEGA 2560 (levo) i Arduino Uno-u (desno)

Na slici 14-2 prikazano je gde su smešteni RESET-EN jastučiči za lemljenje. Koristite oštar nož ili skalpel da biste prekinuli vod koji povezuje dva jastučića za lemljenje. Vodite računa da ne biste promašili i prekinuli druge vodove na pločici. Obično je potrebno nekoliko puta prekinuti vod da biste ga uklonili, tako da se jastučići pravilno isključe.

14.2 ● Debagiranje primera programa

Kreirajte program **debug** ispod da biste započeli novi Atmel Studio GCC C projekat koji se može napraviti od nule bez datoteke šablona. Ovaj projekat je korišćen da bi smo prikazali kako se koristi debager. Za ovaj projekat koristite kolo iz Dodatka C.

Naziv projekta: debug
Lokacija projekta: CH14\debug

main.c

```
void Delay(void);

int main(void)
{
    int var = 1;
    int result, val1, val2;
    float flt_res;
    unsigned char LED_val = 0;

    DDRC  |= 0x0F;                      // four LEDs
    PORTC |= 0x0A;                      // switch two LEDs on

    val1 = 25;
    val2 = 39;
    result = val1 * val2;

    result /= 100;
    flt_res = result / 100.0;

    while (!(PINC & (1 << PINC4)));     // wait for switch press

    while (1)  {
        if (PINC & 1 << PINC5) {        // reset count
            LED_val = 0;
        }
        PORTC &= 0xF0;
        PORTC |= LED_val;               // display count on LEDs
        LED_val++;
        if (LED_val > 0x0F) {           // keep count within range
            LED_val = 0;
        }
        Delay();
        var++;
    }
}

void Delay(void)
{
    volatile unsigned long del = 100000;

    while (del) {
        del = del - 1;
    }
}
```

Rezultat programa

Na LED-ovima se prikazuje broj 1010 koji prati neka izračunavanja koja se mogu videti samo pomoću debagera.

Kada se pritisne dugme na pinu PC4, na LED-ovima će biti prikazano brojanje.

Kada se pritisne dugme na pinu PC5, brojanje se resetuje tako što isključuje sve LED-ove.

Poglavlje 15 • Prekidi

U ovom poglavlju ćete raditi sledeće:

- Saznati kako funkcionišu prekidi
- Koristiti prekide za servisiranje hardvera, umesto tehnike polling
- Koristiti prekide tajmera

Do sada samo "ispitivali" (polling) hardver – kontinuirano proveravajući da li se u programu desio neki uslov. Ako pogledate program timer u odeljku 13.1.2, videćete jedan od nedostataka polling hardvera. Kada je pozvana funkcija `TC1Delay()`, ona koristi vreme obrade mikrokontrolera da bi proverila ili ispitala tajmer. Ako hoćete da ovu funkciju koristite u korisnijem programu, ostatak ugrađenog sistema neće biti u stanju da odgovori na pritiske dugmadi korisnika ili na drugi hardver koji možda treba da bude servisiran dok ova funkcija radi. Prekidi za rešavanje ovog problema su dostupni na mikrokontroleru.

15.1 • Obrada prekida u C programu

Ne postoji standardni način za obradu prekida u C programu. Konfiguracija prekida zavisi od toga kako su prekidi implementirani u arhitekturu određenog mikrokontrolera i u kompajleru. Različiti C kompajleri za isti mikrokontroler mogu da koriste različiti kod za konfiguraciju prekida, tako da uvek proverite dokumentaciju kompajlera da biste videli kako da koristite prekide.

Prekidi mogu da budu konfigurisani na mikrokontroleru tako da ukoliko, na primer, koristite tajmer, on može da se pokrene i da radi sve dok glavni program nastavlja rad. Kada tajmer dostigne vrednost prekida, npr. nakon 1 ms, prekid će biti aktiviran. To prekida trenutni tok glavnog programa i pokreće se funkcija Service Routine (ISR) da bi tajmer bio servisiran. Kada se desi prekid, trenutno stanje mikrokontrolera je sačuvano i dolazi do skoka na ISR funkciju u kojoj radi kod koji će reagovati na tajmer. Kada ISR funkcija završi rad, dolazi do skoka na mesto gde je stao glavni program i sačuvano stanje mikrokontrolera se resetuje. Nakon toga, program nastavlja da radi normalno dok ne dođe do drugog prekida.

15.2 • Korišćenje prekida tajmera

Program **timer_interrupt** koristi 16-bitni timer/counter 1 da bi generisao prekid svake sekunde. Kada se nakon 1 sekunde dostigne prekid tajmera, rutina za obradu prekida se pokreće, čime se povećava vrednost brojanja koja predstavalja proteklo vreme u sekundama.

U petlji `while(1)` programa naredba `if` proverava da li je došlo do promene vrednosti brojanja. Ako je došlo do promene, nova vrednost brojanja će biti prikazana u terminalskom prozoru da se terminal ne bi prepunio vrednostima brojanja, što može da se desi ako se ta vrednost kontinuirano šalje u glavnu petlju.

Kreirajte program **timer_interrupt** koristeći datoteku šablona.

Naziv projekta: timer_interrupt
Lokacija projekta: CH15\timer_interrupt

main.c

```
#include <stdio.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include "stdio_setup.h"

volatile unsigned long g_count_s = 0;

int main(void)
{
    unsigned long saved_count = 0;

    OCR1A  = 15624;
    TCCR1B = (1 << WGM12) | (1 << CS12) | (1 << CS10);
    TIMSK1 = (1 << OCIE1A);
    sei();

    UartInit();

    while(1) {
        if (saved_count != g_count_s) {
            printf("%lu\r\n", g_count_s);
            saved_count = g_count_s;
        }
    }
}

ISR(TIMER1_COMPA_vect)
{
    g_count_s++;
}
```

Rezultat primera programa

```
1
2
3
4
...
```

U gornjem kodu tajmer je podešen i pokrenut kao ranije, ali ovoga puta je podešen da aktivira prekid svake sekunde. Kada tajmer aktivira prekid, kod koji radi u petlji `while(1)` se "prekida" i izvršava se rutina za obradu prekida `ISR(TIMER1_COMPA_vect)`. ISR funkcija jednostavno povećava globalnu promenljivu `g_count` za jedan. Kada se ISR funkcija vrati, kod u petlji `while(1)` nastavlja da se izvršava tamo gde je stao kada je došlo do prekida. Hajde da pogledamo kako funkcioniše kod.

U sledeće dve linije koda timer/counter1 je podešen pomoću perioda od 1 sekunde, deljenja preskalera sa 1024 i CTC režima, kao što je urađeno u prethodnom primeru zadatog tajmera.

```
OCR1A = 15624;
TCCR1B = (1 << WGM12) | (1 << CS12) | (1 << CS10);
```

Poglavlje 16 • Kratak pregled C jezika

U ovom poglavlju ćete raditi sledeće:

- Koristiti strukture za grupisanje različitih tipova podataka
- Koristiti pokazivače za pristup strukturama
- Naučiti unije
- Videti kako funkcionišu nabrojani tipovi podataka
- Pogledati različite teme u C programu da biste potpuno pokrili C jezik

16.1 • Strukture

Nizovi omogućavaju da uskladištimo i pristupimo više promenljivih *istog tipa*. Strukture možete da koristite da biste grupisali podatke različitih tipova – na primer, strukture mogu da sadrže tipove float, int i char, ali niz može da se sastoji samo od jednog tipa podataka.

16.1.1 • Upotreba struktura u C programima

U projektu **sample_struct** koristimo strukturu za grupisanje vremenske vrednosti i primera datuma vremenske oznake da bismo demonstrirali upotrebu strukture u C programu. Kreirajte ovaj projekat pomoću datoteke šablona.
Naziv projekta: sample_struct
Lokacija projekta: CH16\sample_struct
main.c

```
#include <stdio.h>
#include "stdio_setup.h"

int main(void)
{
    struct sample {                /* declare a data type structure */
        int time;                  /* integer member of structure   */
        float data;                /* float member of structure     */
    };
    struct sample samp1;           /* define a structure of type sample */

    UartInit();

    samp1.time = 20;               /* assign values to struct members   */
    samp1.data = 56.23;
    printf("\r\nTime in seconds is: %d\r\n", samp1.time);
    printf("Sampled data is: %.2f", samp1.data);
    while (1);
}
```

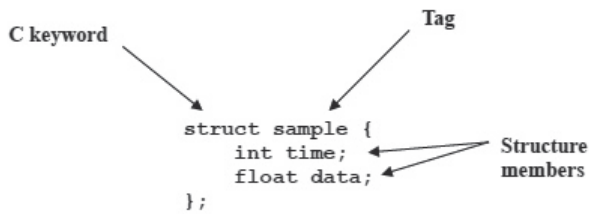
Rezultat programa

```
Time in seconds is: 20
Sampled data is: 56.23
```

Struktura može da sadrži bilo koji broj različitih tipova podataka i kompajler zato ne prepoznaje ono što ćete smestiti u strukturu, pa prvo mora da deklarišete strukturu. U programu sample_struct to se radi pomoću sledećeg iskaza:

```
struct sample {
    int time;
    float data;
};
```

Ovo deklariše novi tip podataka, pod nazivom `struct sample`, koji sadrži dve promenljive člana. Naziv `sample` je poznat kao tag, a to je naziv tipa – tj. deklarisali smo novi tip podataka koji sada može da se koristi u našem programu. Na slici 16-1 prikazani su detalji C deklaracije strukture.



Slika 16-1 C deklaracija strukture

Na isti način na koji možemo da koristimo ključnu reč `int` da bismo definisali promenljivu celog broja u programu, možemo da koristimo i novi tip podataka `struct sample` da bismo definisali novu strukturu ovog tipa. To je ono što radi sledeći iskaz u programu:

```
struct sample samp1;
```

Deklaracija novog tipa strukture nije dodelila memoriju za strukturu, već samo obaveštava kompajler o novom tipu podataka koji ćemo koristiti u našem programu. Gornji iskaz sada kreira novu promenljivu pod nazivom `samp1` tipa `struct sample`. Time se dodeljuje memorija za strukturu i naziv promenljive za tu strukturu je sada `samp1`. U programu možemo da deklarišemo drugu promenljivu tipa `struct sample` na sledeći način:

```
struct sample samp2;
```

To će u programu definisati drugu strukturu tipa struct sample pomoću njoj pripadajućeg naziva. Drugim rečima, druga struktura istog tipa će postojati u memoriji kojoj se može pristupiti korišćenjem naziva – samp2, baš kao što u programu možemo da definišemo dva cela broja ili druge tipove promenljivih:

```
int val1;
int val2;
struct sample samp1;
struct sample samp2;
```

Svakom članu promenljive strukture možemo da pristupimo pomoću operatora tačka (.), kao što možemo da vidimo u sledećim linijama iz programa:

```
samp1.time = 20;
samp1.data = 56.23;
```

Poglavlje 17 • C projekti

U ovom poglavlju ćete raditi sledeće:

- Napisati funkcije za drajver serijskog porta da biste zamenili funkciju `printf()`
- Napisati funkcije za pretvaranje brojeva u decimalne i heksadecimalne stringove
- Napraviti voltmetar projekat
- Kreirati štopericu

Ovo poglavlje sadrži niz projekata za C kod u kome je prikazana upotreba C jezika koji ste već naučili u prethodnim projektima. Nekoliko tema o C jeziku je dodato i prošireno u pratećim objašnjenjima za ovaj projekat.

17.1 • Funkcije za drajver serijskog porta

Kao što je već objašnjeno, funkcije `printf()` koriste dosta memorije koja predstavlja ograničeni resurs u 8-bitnom AVR mikrokontroleru, posebno na manjim delovima koji poseduju manje memorije. U Poglavlju 10 je kao rešenje predloženo da napišete funkcije za serijski port koje mogu da šalju i primaju tekst i znakove preko UART serijskog porta. U projektu iz ovog odeljka ćemo modifikovati kod iz projekta serial_port koji možete pronaći u odeljku 10.5.2, Poglavlja 10. U projektu serial_port načinjene su sledeće izmene.

- Funkcije za drajver serijskog porta / UART-a su smeštene u njihovoj C datoteci
- Podešavanja brzine prenosa za UART i frekvencija radnog takta za mikrokontroler su postavljeni u datoteci global.h
- Dodata je nova funkcija koja prenosi string sa završetkom null
- U programu je prikazana upotreba komentarisanja

Funkcije za drajver serijskog porta omogućavaju slanje stringova preko Arduino serijskog / USB porta na isti način kao što to radi funkcija `printf()` kada je projekat kreiran pomoću datoteke šablona, ali funkcije za drajver serijskog porta ne poseduju mogućnosti formatiranja funkcije `printf()`. Upotrebom funkcije serijskog porta za prenos stringa umesto funkcije `printf()` smanjuje se potrošnja programske memorije.

17.1.1 • Napomene o dizajnu drajvera serijskog porta

Iako se hardverski uređaj u AVR mikrokontroleru koji obrađuje komunikacije serijskog porta naziva USART, koristi se kao UART u kodu drajvera serijskog porta, pa se nazivi datoteke i funkcije koje se koriste za USART nazivaju UART, umesto USART.

Projekat **uart_driver** koji sadrži funkcije za drajver serijskog porta je podeljen na četiri odvojene datoteke izvornog koda, odnosno na datoteke main.c, uart c. i global.h. Datoteka main.c sadrži kod koji prikazuje upotrebu funkcija serijskog porta iz datoteke uart.h. Datoteka uart.h je datoteka zaglavlja za uart.c i mora da bude uvrštena u C izvornu datoteku koja koristi funkcije za drajver serijskog porta. Datoteka global.h sadrži frekvenciju radnog takta ugrađenog sistema za kod koji će biti kompajliran, definisanu kao F_CPU, kao i podešavanje brzine prenosa za UART. Odlučeno je da podešavanje brzine prenosa bude smešteno u globalnu datoteku zaglavlja, tako da brzina prenosa može da se menja bez promena UART datoteka

zaglavlja. Na taj način, datoteke drajvera možete da kopirate iz jednog projekta u drugi, a da ne pritom ne morate da ih modifikujete. Svaka promena brzine prenosa ili frekvencije mikrokontrolera može da se izvrši u globalnoj datoteci zaglavlja global.h.

17.1.2 ● Kod za drajver serijskog porta

Listinzi koda za sve četiri izvorne datoteke projekta **uart_driver** su prikazani u nastavku.

Naziv projekta: uart_driver
Lokacija projekta: CH17\uart_driver
main.c

```
/*-------------------------------------------------------------------------
    FILE NAME:       main.c from uart_driver project

    DESCRIPTION:     Tests the UART driver functions from uart.c

    AUTHOR:          W.A. Smith

    DATE:            30 October 2015
  -------------------------------------------------------------------------*/
#include <avr/io.h>
#include "uart.h"

int main(void)
{
    int rx_data;   // stores data received from the UART

    UartInit();    // initialize the UART to the baud rate set in global.h

    // test the UartTxString() and UartTxByte() functions
    UartTxString("Start of UART driver test program.\r\n");
    UartTxString("Transmitting a character:");
    UartTxByte('C');
    UartTxString("\r\nEchoing typed characters:\r\n");

    // echo back any character typed in the terminal window and terminate with
    // carriage return and newline characters
    while(1) {
        rx_data = UartRxByte();       // get a byte from the UART if available
        if (rx_data > -1) {           // was a byte received
            UartTxByte(rx_data);      // echo the received byte back over the UART
            UartTxString("\r\n");     // terminal window cursor to new line
        }
    }
}
```

global.h

```
#ifndef GLOBAL_H_
#define GLOBAL_H_

// microcontroller clock frequency
#define F_CPU 16000000UL

// USART baud rate settings
#define BAUD 38400
#define BAUD_TOL 1

#endif /* GLOBAL_H_ */
```