

## Datoteka MOTOR.H

```
#include <stdio.h>
#include "arb8051.h"
const int ADR_BRZINA=0x40;
const int ADR_PREKIDAC=0x90;
class motor {
pointer<direct> BrzinaMotora;
pointer<flag> Prekidac;
public:
static int BrojMotora;
motor(void);
motor(int a);
č motor(){--BrojMotora;}
void CitajBrzinu(void){}
direct & Brzina(void){return
*BrzinaMotora;}
void Start(void){*Prekidac=visok;}
void Stop(void){*Prekidac=nizak;}
friend void Obrada(void);
};
int motor::BrojMotora=0;
motor::motor(void)
{
BrzinaMotora=ADR_BRZINA+BrojMotora;
Prekidac=ADR_PREKIDAC+BrojMotora;
++BrojMotora;
}
motor::motor(int a)
{
BrzinaMotora=ADR_BRZINA+a;
Prekidac=ADR_PREKIDAC+a;
++BrojMotora;
}
void Obrada(void)
{
for(int i=0;i<motor::BrojMotora;i++)
{
motor * pMotor=new motor(i);
pMotor->CitajBrzinu();
delete pMotor;
}
return ;
}
```

instrukcija mikrokontrolera. Posebno ćemo rasvetliti poslednja dva slučaja. Varijablama 'pDirect' i 'pFlag' nije pridružen nijedan registar koji obavlja pokazivačku funkciju za razliku od predhodnih slučajeva. Promenljivim ovog tipa možemo dodeljivati celobrojnu vrednost (pDirect=10), uvećavati ih ili umanjivati (++pflag,—pDirect), dodeljivati ulogu indeksne promenljive (pDirect[5] ili (pDirect+10)) i naravno, ovim varijablama dodeljivati ulogu pokazivača (\*pFlag=visok). U pojedinim situacijama, kao u našem sledećem primeru, upotreba pokazivača je poželjna, štaviše neophodna. Pretpostavimo sledeći slučaj: potrebno je upravljati radom jednog ili više motora. Mikrokontroler vrši uključivanje i isključivanje motora, obavlja očitavanje njegove (njihove) brzine i eventualno, na displeju prikazuje

trenutnu brzinu motora. Za naš krajnje pojednostavljen primer, prihvat ćemo da se podatak o brzini sprema u lokaciju direktno adresiranu privatnom promenljivom 'BrzinaMotora', a da se startovanje ( i stopiranje rada ) motora obavlja preko bita direktno adresiranog varijablom 'Prekidac'. Dakle, imamo jednu strukturu podataka koja se sastoji od jednog objekta 'direct' i jednog tipa 'flag'. Budući da je moguće postojanje više objekata tipa 'motor', treba obezbediti da se podaci novih objekata ne preklapaju sa podacima starih, pa se stoga uvodi celobrojna javna varijabla static int BrojMotora koja čuva informaciju o tome koliko je objekata tipa 'motor' do tog momenta formirano. Očitavanje brzine se vrši u pridruženoj funkciji 'CitajBrzinu()' (koja je u našem slučaju prazna) i koja u principu zavisi od hardverske konfiguracije uređaja. Stvaranje objekta je prepušteno konstruktorima motor() i motor(int a). U prvom slučaju struktura podataka se skladišti u skladu sa vrednošću varijable 'BrojMotora', a u drugom se pozicija strukture određuje eksplicitno preko celobrojnog parametra 'a'. "Priateljska" funkcija 'Obrada()', u for petlji, objedinjuje izvršenja svih funkcija 'CitajBrzinu()' nezavisno koliko je objekata tipa 'motor' do tog momenta formirano.

Stvaranjem klase 'motor', sebi donosimo silna olakšanja u postupku programiranja. Više ne moramo da vodimo računa o tome gde je bit-prekidač za pojedini motor, niti u kojoj smo lokaciji smestili podatak o trenutnoj brzini nekog motora.

Zaključak: biblioteka ARB nudi **novi pogled na programiranje mikrokontrolera**. Sistematski rad na stvaranju novih biblioteka funkcija i klasa doprinosi spajanju dva sveta: assemblera i viših programskih jezika. Pomisao da u sistemski orijentisanom okruženju posedujemo objekte kao što su na primer celi i decimalni brojevi, real-time objekti i procedure, PLC varijable čine inspirativnim sve buduće napore usmerene ka stvaranju komfornijeg i produktivnijeg programiranja za mikrokontrolere.

### Kontakt:

Za sva pitanja u vezi teksta obratite se autoru na tel. 026 / 223-120 svakog radnog dana između 18 i 19 časova.



Charles Calvert

## DELPHI punom snagom

Pre svega ova knjiga predstavlja relativno potpun i kvalitetan vodič za programera bez velikog iskustva u pravljenju kompletnih projekata. Naravno, svako razvojno okruženje ima isti cilj, ostaje utisak da je ovaj paket idealan za početnike, male timove i individualce. Knjiga u potpunosti prati i objašnjava razvoj pojedinih delova koji učestvuju u jednom uobičajenom projektu. Treba naglasiti da ovaj paket predstavlja dobar izbor za one korisnike u čijem radu postoji potreba za izradom kvalitetnog korisničkog interfejsa, da se na jednostavan način dodaje "motor" odnosno program koji će nešto konkretno da radi. Dalje, u poslednjih nekoliko poglavlja dobro je objašnjeno OOP ili objektno orijentisano programiranje. Ova tema je, naravno, veoma opširna i ne može se opisati na malo prostora, ali ono što ovde pročitate je više nego dovoljno za kvalitetan start i dalji samostalni napredak. Navedimo i nekoliko zamerki. Knjiga je napisana za prvu verziju Delphi-ja, a sada je već izdata i četvrta verzija, zbog toga pojedina objašnjenja koja su data u knjizi će biti nepotpuna ili čak i netačna. Naravno, ovo će te primetiti jedino ako jako pažljivo čitate knjigu ili ukoliko vam trebaju neki specifični delovi alat koji se u novijim verzijama programa prebaćeni u neki drugi deo menija. Ipak ostaje utisak da je ova knjiga pravi izbor za učenje kako samog jezika tako i alata. Naravno, ni druge knjige, koje su izdate na našem jeziku, nije loše imati na polici, ali stiće se utisak da "Delphi programiranje punom snagom" predstavlja najpotpunije rešenje. **B.I.**

Izdavač:

CET Computer Equipment and Trade Skadarska 45  
11000 Beograd  
tel: 011 3243-043, 3237-246  
fax: 011 3235-139