O OPERATIVNIM SISTEMIMA ZA MIKROKONTROLERE

Svetomir Simonović [1]

Rezime: Rad daje osnovnie karakteristike operativnih sistema koji se primenjuju pri programiranju mikrokontrolera sa posebnim osvrtom na primenu operativnih sistema Salvo i Free RTOS. Osnovni principi su demonstrirani kroz primer jednostavnog softverskog rešenja za robotizovano vozilo koje može primenom prekidačkih senzora da reaguje na dodir i uz primenu senzora ultrazvuka na postojanje objekata u prostoru. Dati su programski listinzi zasnovani na programskom jeziku C posebno prilagođenom za Microchip PIC mikrokontrolere i pokazano je kako se programski jezik proširuje posebnim programskim bibliotekama koje osnovnom programskom jeziku mikrokontrolera dodaju funkcije koje ga stavljaju u okruženje multitasking operativnog sistema koji je pogodan za realizaciju projekata zasnovanih na višem nivou veštačke inteligencije, odnosno za projektovanje sistema koji su u stanji da iskažu povišen nivo interakcije sa okolinom.

Ključne reči: Mikrokontroler, Multitasking, Inteakcija, Okolina, Senzor

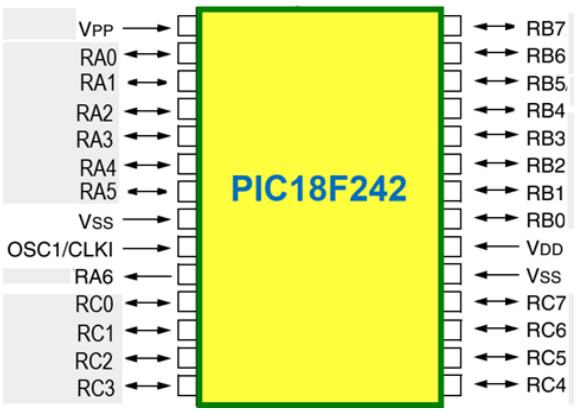
ON MICROCONTROLLERS OPERATING SYSTEMS

Abstract: The work is about basic characteristics of operating systems that are applied with microcontroler programming. Special respect is given to Salvo and Free RTOS operative systems. The basic principles are demonstrated through application of simple software solution dealing with robotised vehicle capable of tactile reaction by applying swching sensors and to react to distant objects in its enviroment by applying ultrasound sensor. In the work programming listings are exposed written in C programming language that is especially adapted to Microchip PIC microcontrolers, and also there is shown the method of widening of the basic microcontroller programming C with separate programming libraries that put it in the environment of a multitascing operating system that is suitable for projects realisation which are based on higher level of artificial inteligence, that is for designing systems capable of exposing higher level of reacting to environment.

Key words: Microcontroller, Multitasking, Interaction, Environment, Senzor

1. UVOD

Rad je izložen na primeru Microchip- ovog PIC18F242 mikrokontrolera, te su terminologija i programski jezik C koji se koristi u radu primereni specifičnosti arhitekture tog mikrokontrolera i u potpunosti je objašnjena u referencama [1] i [2]. Na slici 1. prikazana je uprošćena šema mkrokontrolera PIC18F242 na kojoj su, radi preglednosti, prikazane samo one funkcije koje se koriste za realizaciju softverskog projekta koji je prikazan listingom u ovome radu.



Slika 1 –Uprošćena skica mikrokontrolera PIC18F242

[1]Doktor, profesor strukovnih studija, Akademija strukovnih studija Politehnika, Beograd, ul. Katarine Ambrozić br. 3, e-mail: ssimonovic@politehnika.edu.rs

RA, RB, RC su oznake pinova na portovima A, B , C mikrokontolera, respektivno. Upotreba portova je detaljno opisana komentarima u priloženom listingu, [3], [4].

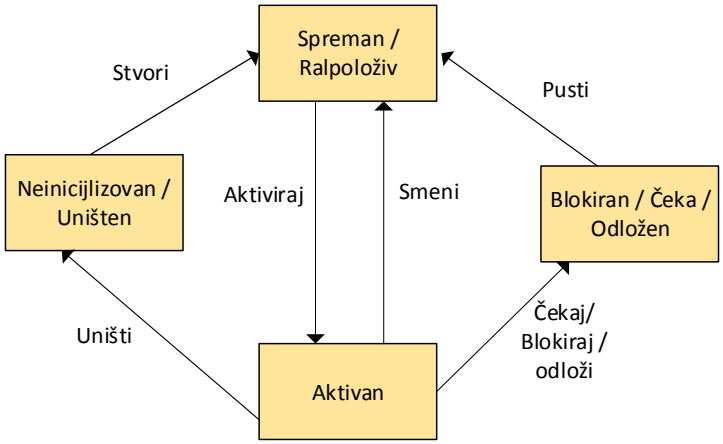
U datom primeru primene operativnih sistema glavni elementi vozila kojim se upravlja preko gore pomenutih pinova su levi i desni motor (koji nezavisno okreću levi i desni točak), levi i desni prednji prekidači za detekciju sudara (dodira) i ultravzučni senzor usmeren naviše za detekciju postojanja objekata iznad vozila, [3], [4].

Operativni sistemi Salvo and Free RTOS softverski se inkorporiraju preko programskih biblioteka koje postojećim softverskim sistemima dodaju sposobnost multitakinga. Kod Free RTOS-a reč je o bibliotekama datoteka pisanim u programskom jeziku C, a kod Salvo-a o bibliotekama datoteka pisanim u programskom jeziku C i objektnim datotekama. Dakle, dodavanjem osobine multitaskinga ne menja se semantika programskog jezika C, malo se menja sintaksa utoliko što se dodaje nekoliko novih komandi i drastično se menja način koncipiranja programa, [5]- [10].

Po pravilu, pri projektovaju multitasking softvera, program se posmatra i piše kao zbir međusobno razdvojenih, relativno samostalnih, programskih celina, tzv. zadataka (tasks) tako da svaki zadatak predstavlja funkciju u zatvorenoj beskonačnoj petlji. Izlazak iz petlje prenosi ingerencije izvršenja na deo programa koji se zove raspoređivač (sheduler). Kod preemptivnih operativnih sistema izlazak inicira sam rapoređivač a kod kooperativnih izlazak se inicira putem određene programske komande unutar same petlje zadatka. Posle izlaska iz petlje predmeni zadatak postaje neraspoređen, a raspoređivač određuje koji će se naredni zadatak izvršiti. Cilj je da se zadovolje stroga vremenska ograničenja za izvršenje pojedinih zadataka, što je i osnovni razlog uvodjenja ovih operativnih sistema, [5]- [10].

2. OSNOVNI ALGORITAM MULTITASKINGA

Na slici 2. prikazan je osnovni algoritam multitaskinga koji sprovodi šeduler sam i /ili uz pomoć određenih komandi dotičnog operativnog sistema unutar petlji zadataka, [3], [4].



Slika 2 –Osnovni algoritam multitaskinga

Na slici 2. se unutar pravougaonika nalaze moguća stanja zadataka, a nazivi strelica označavaju aktivnosti operativnog sistema koje dovode do promene tih stanja u smerovima označenim strelicama.

Na primer, u sintaksi Salvo-a, kada se zadatak stvara komandom **OSCreateTask**, on dolazi u stanje **spremnosti**/ raspoloživosti, i na raspolaganju je raspoređivaču koji ga **aktivira** u skladu sa njegovim prioritetom (u ovom slučaju takođe određenim komandom **OSCreateTask)**. Iz aktivnog stanja proces može preći u stanje **odloženosti** npr. primenom komande **OS_Delay** ili **čekanja**

primenom komande **OS_Wait_Msg** koje se nalaze u petlji tog zadatka. Iz stanja čekanja zadatak može preći u stanje spremnosti komandom **OS_Signal_Msg** koja se nalazi u petlji nekog drugog zadatka**,** [6], [7].

U konkretnom primeru stvorena su dva zadatka i jedna poruka, respektivno: Motor_zadatak, Uzv_zadatak i Por_za_Motor. Posle stvaranja Motor_zadatak se stavlja u stanje čekanja na poruku komandom **OS_Wait_Msg**. U zavisnosti od toga da li je poruku primio od desnog prekidača, levog prekidača ili ultrazvučnog senzora Motor_zadatak će preći u aktivno stanje i, posle raspoređivanja od strane raspoređivača u skladu sa svojim prioritetom, izvršiti primerene aktivacije levog, odnosno desnog motora (koji pokreću levi, odnosno desni točak) da bi vozilo izbeglo percipirane prepreke. Motor_zadatak potrebno manevrisanje obavlja putem funkcija levimot_npr(), desnimot_npr(), levimot_naz() i desnnimot_naz(). Komanda OS_Yield() smenjuje aktivni zadatak; OSSched() aktivira raspoređivač.

Poruke od senzora dodira idu preko interapta, a poruka od ultrazvučnog senzora, zbog tehnološke složenosti, ide preko posebno formiranog zadatka: Uzv_zadatak.

Algoritam kretanja vozila i detalji programa mogu se videti iz programskog listinga koji sledi.

3. LISTING PROGRAMA ZA UPRAVLJANJE VOZILOM

```
/*********************************************************************
Ovaj program implementira  navigaciju vozila Dervot na slepo, sa nagore okrenutim ultrazvučnim
senzorom da bi detektovao da li je iznad robota svod.
Zadaci su: Uzv_zadatak (viši prioritet), Motor_zadatak (niži prioritet)
Interupti su: Mikroprekidač (Niži prioritet) i Tajmer 0 (Visoki prioritet) za otkucaje sata)
Takođe se koristi jedna poruka (Message) i više odlaganja (Delays)
Primenjuje verziju operativnog sistema Salvo LITE sa bibliotekom sfc18sfa.

*********************************************************************/
//Frekvencija sata je 4MHz
#include <salvo.h>
#undef OSC //neophodno za ovu verziju  Salvo-a, pošto ona isto koristi ovo ime
#include <p18f242.h>
#include <timers.h> // datoteka headera za tajmere
#include <delays.h> //datoteka headera za  odlaganja
#include <pwm.h> //datoteka headera za  PWM (upravljanje putem širine pulsa)
#pragma config OSC = HS, OSCS = OFF //oscilator tipa HS, isključi oscilator
#pragma config PWRT = ON, BOR = OFF /uključi  tajmer uključenja, isključi detetovanje brown-out -a
#pragma config WDT = OFF //isključi watchdog tajmer
#pragma config STVR = ON, LVP = OFF //uključi resetovanje pri pojavi prelivanja Stack-a,
                                    //isključi programiranje pri niskom naponu
//Prototipovi funkcija koje je definisao korisnik:
void Chip_Init(void);
void levimot_npr (void);
void desnimot_npr (void);
void levimot_naz (void);
void desnitmot_naz (void);



//Donje funkcije su zadaci
void Motor_zadatak( void ); //Pokreće motor u skladu sa primljenom porukom
void Uzv_zadatak( void ); //Periodično aktivira ultrazvučni senzor

//Prenosi poruke od mikroprekidača sudara i ultrazvučnog senzora:
#define Por_za_Motor OSECBP(1)
char Otvor = 0x18; //Formalno prisutno, ne proverava se u programu

/*********************************************************************
// Funkcije definisane od strane korisnika, uključujući zadatke:
*********************************************************************/

//sledeći  zadatak kontroliše ponašanje motora koje je određeno pristiglom porukom
void Motor_zadatak( void )
{ static char pord; //Drži pročitanu poruku
OStypeMsgP porP; //Deklariše porP kao specijalni pointer  Salvo-a
for (;;) //Postavlja beskonačnu petlju zadatka
{
```

```
desnimot_npr (); //Stavlja motore u pogn napred. Ovo je tekuće stanje dok ne stigne poruka
levimot_npr  ();
//Čeka na poruku
OS_WaitMsg(Por_za_Motor,&porP,OSNO_TIMEOUT);
// Produžava kada stigne poruka
pord = *(char*)porP;
PORTAbits.RA5 = 0; //Zaustavlja motore na 500ms
PORTAbits.RA2 = 0;
OS_Delay (50);
if( (pord == 0x80)| | (pord == 0x01) ) //Da li je u pitanju mikroprekidač?
{
    desnitmot_naz (); //Da, oba motora se kreću unazad
    levimot_naz  ();
OS_Delay (100);
if(pord == 0x80) //Da li je u pitanju levi mikroprekidač?
{levimot_npr (); //Da, skretanje desno
OS_Delay (80);
}
else // Desni prekidač je uključen, pa
{desnimot_npr (); //  okreni levo
OS_Delay (80);
}
}
else //Nalazi se ispod svoda, pa okreće nadesno za 180 stepeni
{desnitmot_naz ();
    levimot_npr();
OS_Delay (200);
}
} //kraj petlje"for"
}

/* Sledeći zadatak periodično pulsira ultrazvukom i šalje poruku ako je detektovano nadsvođenje.
U tom slučaju suspenduje pulsiranje da bi vozilo imalo vremena da izađe iz "otvora" */
void Uzv_zadatak(void)
{ int eho_vreme = 0; //Merenje distance putem brojanja odziva
for (;;) //Postavlja beskonačnu petlju zadatka
{
OS_Delay (20,USnd_Task1); //Menja zadatak i pravi pauzu od 20x10ms, (200ms)
INTCONbits.GIE = 0//Isključuje interapte pošto je ovo merenje vremenski osetljivo

eho_vreme = 0;
PORTCbits.RC5 = 1; //Pokreće pulsiranje
Delay10TCYx(2); //Odlaganje od 20us da bi se pulsu dala odgovarajuća širina
PORTCbits.RC5 = 0;
Delay10TCYx(30); //Pauzira da sačeka pojavu 1 na pinu
//Vrednosti u ovoj petlji su određene eksperimentalno da bi dale prag detekcije od približno 30cm
while (eho_vreme < 50) //Ograničava merenje na objekte u blizini
{Delay10TCYx(1); //odlaganje od 10us
eho_vreme ++; //inkrementira brojač
if(PORTCbits.RC6 == 0) //Ako je cilj otkriven šalje poruku
{OSSignalMsg(Msg_to_Motor,(OStypeMsgP)&otvor);
//Uklkjučuje interapte pre odlaganja:
INTCONbits.GIE = 1;
INTCONbits.PIE = 1;
//Susppenduje zadatak ultrazvučnnog senzora da bi omogućio vozilu da napusti "otvor":
OS_Delay (250);
OS_Delay (250);
break;
}
}
//Ukkjučuje interapte:
INTCONbits.GIE = 1
INTCONbits.PIE = 1;

OS_Yield();
} //Kraj petlje  "for"
}

// Ova funkcija inicijalizuje periferije mikrokontrolera:
void Chip_Init(void)
{
//Inicijalizuje portove:
```

```
TRISA = 0b00000000; // Svi bitovi postavljeni kao izlazni,  od kojih su bitovi 2 i 5 za uključivanje
motora.
TRISB = 0b00110000; // Samo su bitovi  5 i 4 (mikroprekidači) postavljeni kao ulazni
TRISC = 0b11000000; // Svi bitovi osim bitova 7 (promena režima) i 6 (ultrazvučni senzor) su izlazni
                    // Bitovi  1 i 2 se koriste za upravljanje širinom pulsa
ADCON1 = 0b00000110; //Postavlja port A kao digitalni
//Isključuje sve portove:
PORTA = PORTB = PORTC = 0;

/* Inicijalizuje Tajmer 0: uključen interapt, koristi interni sat, 16-bitni rad,
preskaler deli sa 16, otuda  (sa frekvencijom sata od 4MHz ) period ulaznog ciklusa je 16us: */
OpenTimer0 (TIMER_INT_ON & T0_SOURCE_INT & T0_16BIT & T0_PS_1_16);
WriteTimer0 (64918); //and initialise
//Inicijalizuje upravljanje širinom pulsa:
OpenTimer2 (TIMER_INT_OFF & T2_PS_1_1 & T2_POST_1_1);
OpenPWM1 (0xFF); //Uklkjučuje upravljanje putem širine pulsa 1 i postavlja period pulsa
OpenPWM2 (0xFF); //Uklkjučuje upravljanje putem širine pulsa 2 i postavlja period pulsa
//Daje interaptu promene na portu B nizak prioritet:
RCONbits.IPEN = 1; //Uključuje interapte niskog prioriteta
INTCON2bits.RBIP = 0; //Daje promeni na portu B nizak prioritet interupta
INTCONbits.RBIE = 1;//Uključuje interapt promene na portu B
}

/*********************************************************************
Main
*********************************************************************/
void main( void )
{
//Inicijalizuje mikrokontoler:
Chip_Init();
Delay10KTCYx (250); //pauzira 2.5 sekunde

//Kreira zadatke i poruku
OSCreateTask(Motor_zadatak, OSTCBP(1), 4);
OSCreateTask(Uzv_zadatak, OSTCBP(2), 2);
OSCreateMsg(Por_za_Motor, (OStypeMsgP)0);
//Ukljkučuje interapte:
INTCONbits.GIE = 1;
INTCONbits.PIE = 1;


// Petlja raspoređivača:
for (;;)
OSSched();
}
/*********************************************************************
Funkcije pogona motora
*********************************************************************/
void levimot_npr (void) //Pokreće levi motor naped
{ CCPR2L = 196;
PORTAbits.RA5 = 1; //Uključuje levi motor
}
void desnimot_npr (void) //Pokreće desni motor napred
{ CCPR1L = 196;
PORTAbits.RA2 = 1; //Uklkjučuje desni motor
}
void levimot_naz  (void) //Pokreće levi motor unazad
{ CCPR1L = 60;
PORTAbits.RA5 = 1; //Uključuje levi motor
}
void desnitmot_naz (void) //Pokreće desni motor unazad
{ CCPR1L = 60;
PORTAbits.RA2 = 1; //Uključuje desni motor
}
```

4. ZAKLJUČAK

Operativni sistemi za mikrokontrolere daju elegantno programsko rešenje uvođenja multitaskinga u programsku šemu i daju mogućnost da se ostvare najstrožiji vremenski zahtevi za izvršenje kritičnih zadataka na mikrokontrolerima. Međutim, ova mogućnost ima svoju cenu po pitanju povećane programske memorije koja je potrebna za akomodaciju operativnih sistema i generalnog usporavanja

izvršenja programa zbog posredničke prirode operativnih sistema. Osim toga kod mikroprocesora koj imaju jedno jezgro i nije moguće ostvariti paralelno izvršavanje programskih instrukcija, operativni sistemai pisani za mikrokontrolere u suštini su uvek sekvencijalni programi i paralelno izvršavanje instrukcija na mikroprocesoru je samo prividno. U tom smislu treba uvek razmisliti i o klasičnim implementacijama softverskih rešenja koje bi mogle imale manju cenu u odnosu na rešenja sa operativnim sistemima.

5. LITERATURA

[1] MPLAB® C18 C COMPILER USER'S GUIDE, Microchip Technology Incorporated, 2004.
[2] PIC18FXX2 Data Sheet, Microchip Technology Incorporated, 2006.
[3] Wilmshurst, T.: *Designing Embedded Systems with PIC Microcontrollers*, Elsevier Ltd, Oxford, UK, 2007.
[4] Wilmshurst, T.: *Designing Embedded Systems with PIC Microcontrollers*, Elsevier Ltd, Oxford, UK, 2010.
[5] Ibrahim, D.: *Advanced PIC microcontroller projects in C: from USB to RTOS with the PIC18F series*, Elsevier Ltd, Oxford, UK, 2008.
[6] Salvo Compiler Reference Manual – Microchip MPLAB-C18, Pumpkin Inc., 2007.
[7] Salvo User Manual, Pumpkin, Inc, 750 Naples Street, San Francisco, 2010.
[8] Barry, R.: Mastering the FreeRTOS™ Real Time Kernel, Amazon.com, Inc. or its affiliates, https://www.freertos.org/Documentation/02-Kernel/07-Books-and-manual/01-RTOS_book, 2023.
[9] FreeRTOS: User Guide, Amazon Web Services, Inc. and/or its affiliates, 2024.
[10] The FreeRTOS™ Reference Manual, Amazon.com, Inc. or its affiliates, 2017.