

# OK. Vama je potrebna ... CAN mreža



Easy8051B razvojni sistem i CAN-SPI moduli

Neretko se javlja potreba da se više mikrokontrolera koji vrše različite operacije integrišu u jedan sistem kako bi funkcionisali kao celina. U ovom tekstu ćemo pokazati kako umrežiti tri mikrokontrolera u CAN mrežu. Takođe, objasnimo kako se koriste filtri u CAN čvorovima u cilju selekcije poruka.

Zoran Ristic  
MikroElektronika - Sektor za razvoj softvera

Obzirom da više perifernih jedinica koristi istu magistralu za razmenu podataka, neophodno je uvesti red u način korišćenja te magistrale. CAN standard precizno opisuje sve detalje umrežavanja više uređaja i kao takav je široko prihvaćen u industriji. CAN standard precizno definiše prvenstvo korišćenja magistrale i hardverski rešava problem „sudara“ u slučaju da više perifernih jedinica počne da komunicira u isto vreme.

## Hardver

U našem primeru CAN mreža će biti konfigurisana tako da prvi uređaj emituje poruke sa ID-jem 0x10 i 0x11, a drugi i treći će emitovati poruke sa ID-jem 0x12 i 0x13 respektivno. Takođe, CAN čvorovi će biti konfigurisani tako da drugi čvor odgovara samo na dolazne pakete sa ID-jem 0x10, a treći uređaj odgovara samo na one pakete čiji je ID jednak 0x11. Konačno, prvi uređaj je podešen tako da prima poruke sa ID-jem 0x12 i 0x13 (slika 2.). Filtriranje poruka ćemo lako podesiti tako što ćemo pozvati rutinu CANSPISetFilter koja će izvršiti sva neophodna podešavanja registra mikrokontrolera i CAN SPI pločice.

Generalno, CAN standard ne zahteva prisustvo master uređaja na magistrali,

ali ćemo mi u cilju lakšeg razumevanja primera i bez gubitka opštosti podesiti da samo prvi uređaj inicira komunikaciju na mreži, a da preostala dva uređaja odgovaraju na pojedinačne prozivke.

## Softver

Prilikom emisije poruke master čvor ostavlja dovoljno vremena da se prozvani čvor odazove. U slučaju da nakon isteka tog vremena udaljeni čvor ne pošalje odgovor, tada master proglašava grešku za tekuću poruku i nastavlja dalje sa prozivkom ostalih čvorova (slika 3.). U slučaju da nakon isteka očekivanog vremena periferni CAN čvor ipak krene u odgovor u isto vreme kada i neki drugi čvor, tada će doći do kolizije na CAN magistrali. Međutim, prioritet adresa uređaja i koncepcija CAN mreže je takva da će se uređaj nižeg prioriteta u ovom slučaju povući sa magistrale čime infor-

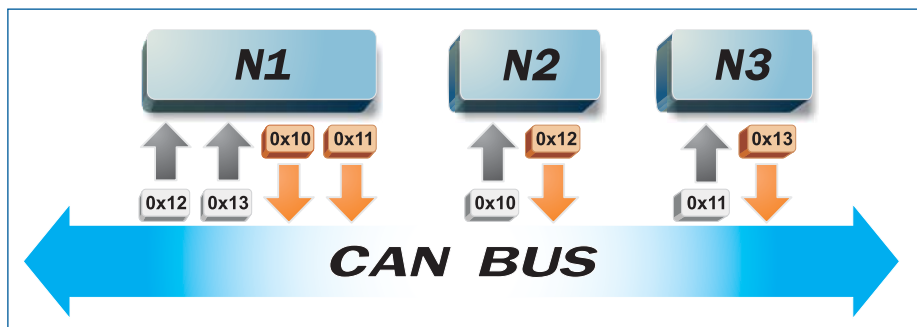
macija višeg prioriteta neometano nastavlja da se emituje preko mreže.

Kao što smo napomenuli, iskoristićemo interni SPI modul mikrokontrolera za slanje podataka na CAN magistralu. Neke od prednosti korišćenja internog SPI modula mikrokontrolera su:

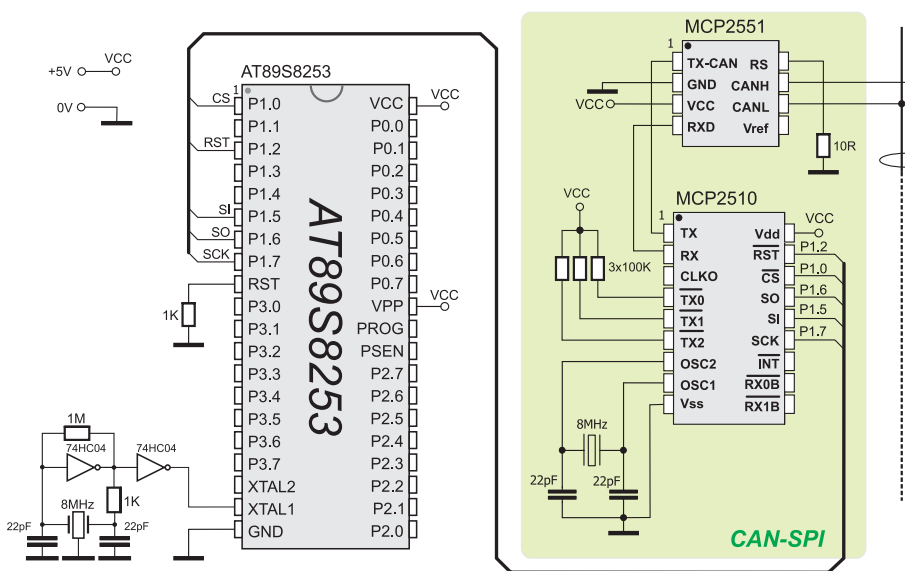
1. Mogućnost generisanja prekida pri likom predaje i prijema podataka;
2. Nezavisnost rada SPI modula od ostalih periferija mikrokontrolera; i
3. Jednostavna konfiguracija.

CAN SPI biblioteka omogućava podešavanje režima rada CAN mreže, postavljanje filtera za predajne i prijemne čvorove, čitanje podataka iz bafera CAN SPI pločice itd.

Priloženi primer uključuje LED na pinovima mikrokontrolera i time daje in-



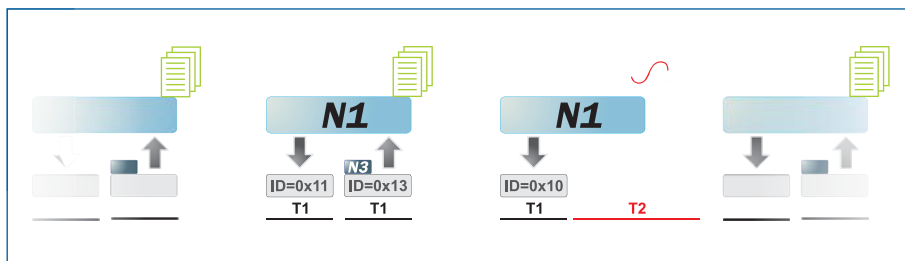
Slika 1. Filtriranje poruka



Šema 1. Povezivanje CAN-SPI modula sa AT89S8253 mikrokontrolerom

dikaciju da je mreža funkcionalna. Kada se čvor broj 2 odazove na poziv čvora broj 1, uključuju se LED diode na portu B. U slučaju da čvor broj 3 odgovori na poziv, uključuju se LED diode na portu D.

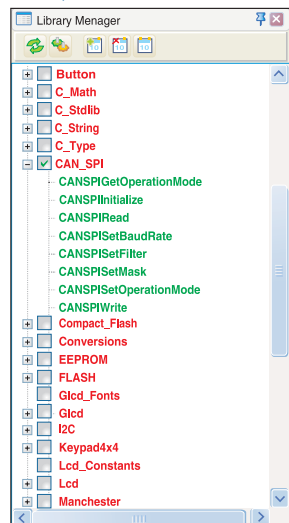
U primeru je dat izvorni kod za sva tri čvora u mreži. Da bi se napravio HEX za svaki od čvorova pojedinačno, potrebno je ostaviti uvek jednu i samo jednu DEFINE direktivu u zaglavlju primera.



Sluka 2. Primer komunikacije

U ovom primeru smo predstavili jedan način umrežavanja mikrokontrolera u CAN mrežu. Na primeru komunikacionog protokola smo objasnili kako se vrši detekcija grešaka u slučaju da udaljeni čvor ne šalje očekivane informacije. Takođe, pokazali smo kako se filtriraju poruke pomoću CAN filtera i kako se generalno vrši komunikacija na CAN magistrali.

Editor biblioteka kompajlera mikroBASIC for 8051® sa gotovim bibliotekama kao što su: CAN\_SPI, GLCD, Ethernet itd.



### Funkcije koje se koriste u programu

CANSPIGetOperationMode()	Aktivni mod rada
CANSPINitialize()*	Inicijalizacija CANSPI modula
CANIRead()*	Čitanje poruke
CANSPISetBaudRate()	Podešavanje CANSPI baud rejta
CANSPISetFilter()*	Konfiguracija filtera za poruke
CANSPISetMask()*	Napredna konfiguracija filtera
CANSPISetOperationMode()*	Aktivni mod rada
CANSPISetMask()	Upis poruke
* Funkcije CANSPI biblioteke korišćene u programu	
Ostale funkcije kompajlera mikroBASIC for 8051 korišćene u programu:	
Delay_us()	
SPI1_init()	
SPI1_read()	

### Program za demonstriranje rada CAN magistrale

```

program CanSPI
  ` Description: This program demonstrates how to
  ` make a CAN network using mikroElektronika
  ` CANSPI boards and mikroBasic
  ` compiler.
  ` Target device: AT89S8253
  ` Oscillator: 8MHz crystal

dim Can_Init_Flags, Can_Send_Flags, Can_Rcv_Flags as
byte
  ` can flags
  Rx_Data_Len as byte
  ` received data length in bytes
  RxTx_Data as byte[8]
  ` can rx/tx data buffer
  Msg_Rcvd as byte
  ` reception flag
  Tx_ID, Rx_ID as longint
  ` can rx and tx ID

  ErrorCount as byte
  ` CANSPI module connections
dim CanSpi_CS as sbit at P1.B0
  ` Chip select (CS) pin for CANSPI board
  CanSpi_Rst as sbit at P1.B2
  ` Reset pin for CANSPI board
  ` End CANSPI module connections
main:
  ErrorCount = 0
  ` Error flag
  Can_Init_Flags = 0 Can_Send_Flags = 0 Can_Rcv_Flags = 0
  ` clear flags

  Can_Send_Flags = CAN_TX_PRIORITY_0 and
  ` Form value to be used
  CAN_TX_XTD_FRAME and
  ` with CANSPIwrite
  CAN_TX_NO_RTR_FRAME

  Can_Init_Flags = CAN_CONFIG_SAMPLE_THRICE and
  ` Form value to be used
  CAN_CONFIG_PHSEG2_PRG_ON and
  ` with CANSPIInit
  CAN_CONFIG_XTD_MSG and
  CAN_CONFIG_DBL_BUFFER_ON and
  CAN_CONFIG_VALID_XTD_MSG

  SPI_Init()
  CANSPIInitialize(1, 3, 3, 3, 1, Can_Init_Flags)
  ` Initialize external CANSPI module
  CANSPISetOperationMode(CAN_MODE_CONFIG, TRUE)
  ` set CONFIGURATION mode
  CANSPISetMask(CAN_MASK_B1, -1, CAN_CONFIG_XTD_MSG)
  ` set all mask1 bits to ones
  CANSPISetMask(CAN_MASK_B2, -1, CAN_CONFIG_XTD_MSG)
  ` set all mask2 bits to ones

  CANSPISetFilter(CAN_FILTER_B2_F4, 0x12, CAN_CONFIG_XTD_MSG)
  ` Node1 accepts messages with
  ID 0x12
  CANSPISetFilter(CAN_FILTER_B1_F1, 0x13, CAN_CONFIG_XTD_MSG)
  ` Node1 accepts messages with
  ID 0x13

  CANSPISetOperationMode(CAN_MODE_NORMAL, 0xFF)
  ` set NORMAL mode
  RxTx_Data[0] = 0x40
  ` set initial data to be sent

  Tx_ID = 0x10
  ` set transmit ID for CAN message

  CANSPISetMask(CAN_MASK_B1_F1, 0x13, CAN_CONFIG_XTD_MSG)
  ` Node1 sends initial message

  while (TRUE)
  ` endless loop
    Msg_Rcvd = CANSPIRead(Rx_ID, RxTx_Data, Rx_Data_
  Len, Can_Rcv_Flags)
    ` attempt receive message
    if (Msg_Rcvd) then
    ` if message is received then check id

    if Rx_ID = 0x12 then
    ` check ID
      P0 = RxTx_Data[0]
      ` output data at PORT0
    else
      P2 = RxTx_Data[0]
      ` output data at PORT2
    end if
    delay_ms(50)
    ` wait for a while between messages
  CANSPISetMask(CAN_MASK_B1_F1, 0x13, CAN_CONFIG_XTD_
  Flags)
  inc(Tx_ID)
  ` switch to next message
  if Tx_ID > 0x11 then Tx_ID = 0x10 end if
  ` check overflow

  else
  ` an error occurred, wait for a while

  inc(ErrorCount)
  ` increment error indicator
  Delay_ms(10)
  ` wait for 10ms
  if (ErrorCount > 10) then
  ` timeout expired - process errors
    ErrorCount = 0
    ` reset error counter
    inc(Tx_ID)
    ` switch to another message
    if Tx_ID > 0x11 then Tx_ID = 0x10 end if
    ` check overflow
    CANSPISetMask(CAN_MASK_B1_F1, 0x13, CAN_
  Send_Flags)
    ` send new message
  end if

  end if
wend
end.
  
```



GO TO Ovaj program, pisan za 8051® mikrokontrolere u programima C, Basic i Pascal kao i programe napisane za mikrokontrolere PIC®, dsPIC i AVR® možete naći na našem web sajtu: [www.mikroe.com/en/article/](http://www.mikroe.com/en/article/)