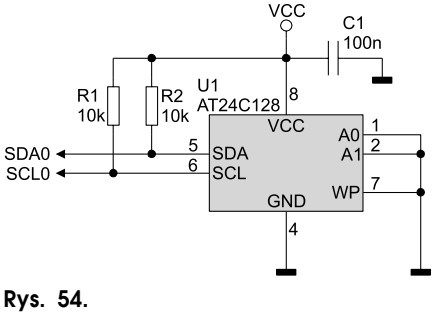
Mikrokontrolery z rdzeniem ARM, część 19

Interfejsy szeregowe: I²C – przykład

Omówiliśmy rejestry niezbędne do posługiwania się sprzętowym interfejsem I²C, zatem możemy przystąpić do części praktycznej. W artykule przedstawiamy przykład obsługi pamięci EEPROM dołączanej do mikrokontrolera LPC za pomocą magistrali I²C.

Niestety w zestawie ZL6ARM, który jest naszą platformą sprzętową, nie ma żadnego urządzenia podłączonego do magistrali I²C, dlatego przed uruchomieniem przykładu będziemy musieli przeprosić się z lutownicą i zmontować na „pająku" układ z pamięcią AT24C128 przedstawiony na **rys. 54**. Linie SDA i SCL zgodnie ze specyfikacją I²C są „podciągnięte" za pomocą rezystorów 10 kΩ do plusa napięcia zasilającego. Tak podłączona pamięć będzie widoczna na magistrali pod adresem 0xA0. Na **list. 10** pokazano fragment programu (*ep8c.zip*, dostępny na CD–EP6/2007B) realizujący procedury zapisu i odczytu pamięci AT24C128.

Funkcja *EepromInit()* dokonuje inicjalizacji kontrolera I²C. Najpierw ustawiane są funkcje alternatywne portu, tak aby pełniły one rolę linii SDA oraz SCL, ustawiana jest prędkość transmisji magistrali I²C, a na koniec włączany jest kontroler magistrali. Funkcja *EepromWrite()* zapisuje pod adresem *addr* w pamięci AT24C128 liczbę *val*. Operacja ta rozpoczyna się od nadania bitu startu poprzez ustawienie bitu STA w rejestrze I2C0CONSET, następnie program wchodzi do pętli, w której w zależności od odpowiedniego stanu rejestru I2C0STAT podejmuje określone czynności. Po nadaniu bitu startu, rejestr I2C0STAT przyjmuje wartość 0x08, w wyniku czego do rejestru I2C0DAT wpisywany jest sprzętowy adres pamięci EEPROM, ustawiany jest bit potwierdzenia i kasowany bit startu. Następnie kasowany jest bit SI, co powoduje wykonanie następnego



Rys. 54.

List. 10.

```
/Ustawienia kontrolera VIC
#define SCL0_P02_SEL (1<<4)
#define SDA0_P03_SEL (1<<6)
//Adres urzadzenia na magistrali I2C
#define I2C_MEMADDR 0xA0

/* Inicjalizuje interfejs I2C pamiec AT24C128*/
void EepromInit(void)
{
  PINSEL0 |= SCL0_P02_SEL | SDA0_P03_SEL;
  //FI2C = PCLK/(SCLL+SCLH) - 100KHz
  I2C0SCLL = 300;
  I2C0SCLH = 300;
  //Wyzeruj wszystkie flagi
  I2C0CONCLR = 0x6C;
  //Wlacz interfejs I2C - MASTER
  I2C0CONSET = I2C0CONSET_I2EN;
}

/* Zapisuje komorke pamieci AT24C128
 * addr - adres komorki pamieci do zapisania
 * val - vartosc liczbowa do zapisania
 * Zwraca 0 dla sukces lub wartosc ujemna dla bledu
 */
int EepromWrite(unsigned short addr,unsigned char val)
{
  int tmp;
  //Licznik nadanych bajtow
  int cnt = 0;
  //Rozpocznij nadawanie bitu start
  I2C0CONSET = I2C0CONSET_STA;
  //Petla oczekiwania
  while(1)
  {
     //Czekaj na zdarzenia
     while(I2C0STAT==0xF8);
     //Status magistrali
     switch(I2C0STAT)
     {
     //Bit start zostal nadany
     case 0x08:
        //Wyslij adres pamieci I2C
        I2C0DAT = I2C_MEMADDR;
        I2C0CONSET = I2C0CONSET_AA;
        I2C0CONCLR = I2C0CONCLR_SIC|I2C0CONCLR_STAC;
        break;
     //Adres I2C zostal nadany
     case 0x18:
        //Wyslij starszy bajt adresu
        I2C0DAT = addr>>8;
        I2C0CONSET = I2C0CONSET_AA;
        I2C0CONCLR = I2C0CONCLR_SIC;
        cnt=0;
        break;
     //Dane zostaly nadane
     case 0x28:
        //Pierwszy raz
        if(cnt==0)
        {
           //Mlodsza czesc adresu
           I2C0DAT = addr;
           I2C0CONSET = I2C0CONSET_AA;
           I2C0CONCLR = I2C0CONCLR_SIC;
        }
        //Drugi raz
        else if(cnt==1)
        {
           //Dana do umieszczenia w komorce pamieci
           I2C0DAT = val;
           I2C0CONSET = I2C0CONSET_AA;
           I2C0CONCLR = I2C0CONCLR_SIC;
        }
```

polecenia przez kontroler I²C. Po nadaniu adresu sprzętowego w ten sam sposób jest nadawana starsza część adresu pamięci, następnie młodsza część adresu pamięci oraz dane do zapisania. Na zakończenie nadawany jest bit stopu. Jeśli rejestr I2C0STAT przyjmie wartość inną od wyszczególnionych w sekcjach *case*, oznacza to wystąpienie błędu. Funkcja wychodzi wówczas z pętli głównej i zwraca wartość mniejszą od zera. Funkcja *ReadEeprom()* odczytuje bajt spod adresu pamięci EEPROM wskazanego jako argument. Jest ona nieco bardziej skomplikowana od poprzedniej z uwagi na to, że najpierw musimy zapisać do pamięci adres, spod którego chcemy odczytać dane. Następnie musimy wysłać ponownie bit startu, przesłać adres sprzętowy tym razem z najmłodszym bitem ustawionym do odczytu i dopiero po tej czynności dane mogą być odczytane z pamięci. Ponieważ odczytujemy tylko jeden bajt danych, w stanie po wysłaniu adresu w trybie do odczytu (0x58) zerujemy bit AA w rejestrze I2C0CONCLR, w wyniku czego nie zostanie wysłane potwierdzenie. Będzie to sygnałem dla układu podrzędnego, że jest to ostatnia dana. W przypadku, gdy chcielibyśmy przesłać większą ilość danych, należy bit AA ustawić, a wyzerować tuż przed odbieraniem ostatniego bajtu danych. Funkcje te zostały napisane bez użycia systemu przerwań, tak aby pokazać samą ideę użycia kontrolera I²C. Bez wykorzystania systemu przerwań pożytek ze sprzętowego interfejsu I²C w trybie nadrzędnym i tak jest niewielki, ponieważ mikrokontroler w aktywnej pętli cały czas zajmuje się badaniem rejestru statusu. Dopiero wykorzystanie systemu przerwań pozwoli wykorzystać procesor do realizacji innych zadań.

Program *Ep8c.zip* wykorzystując poprzednio omówione procedury interfejsu UART, wysyła napis powitalny, a następnie oczekuje od użytkownika wpisania komendy *write=tekst* Wydanie tej komendy powoduje zapisanie w zewnętrznej pamięci EEPROM łańcucha tekstowego, którego odczyt jest możliwy za pomocą komendy *read*.

Lucjan Bryndza, EP
lucjan.bryndza@ep.com.pl

List. 10. c.d.

```
      //Trzeci raz
      else if(cnt==2)
      {
         //Wyslij STOP
         I2C0CONSET = I2C0CONSET_AA|I2C0CONSET_STO;
         I2C0CONCLR = I2C0CONCLR_SIC;
         return 0;
      }
      cnt++;
      break;
    //Jezeli blad zatrzymaj i wyjdz
    default:
      tmp = I2C0STAT;
      I2C0CONSET = I2C0CONSET_AA|I2C0CONSET_STO;
      I2C0CONCLR = I2C0CONCLR_SIC;
      return -tmp;
    }
  }
}

/* Odczytuje komorke pamieci AT24C128
 * addr - adres komorki pamieci
 * Zwraca zawartosc komorki gdy OK w przypadku bledu
 * zwraca wartosc mniejsza od zera
 */
int EepromRead(unsigned short addr)
{
  int tmp;
  int cnt = 0;
  //Transmit Start BIT
  I2C0CONSET = I2C0CONSET_STA;
  //Petla oczekiwania
  while(1)
  {
    //Czekaj na zdarzenie
    while(I2C0STAT==0xF8);
    //Okresl rodzaj
    switch(I2C0STAT)
    {
    //Bit startu nadany
    case 0x08:
      //Wyslij Adres pamieci I2C
      I2C0DAT = I2C_MEMADDR;
      I2C0CONSET = I2C0CONSET_AA;
      I2C0CONCLR = I2C0CONCLR_SIC|I2C0CONCLR_STAC;
      break;
    //Adres zostal nadany
    case 0x18:
      //Wyslij starsza czesc adresu
      I2C0DAT = addr>>8;
      I2C0CONSET = I2C0CONSET_AA;
      I2C0CONCLR = I2C0CONCLR_SIC;
      cnt=0;
      break;
    //Dane zostaly nadane
    case 0x28:
      //Pierwszy raz
      if(cnt==0)
      {
         //Wyslij mlodsza czesc adresu
         I2C0DAT = addr;
         I2C0CONSET = I2C0CONSET_AA;
         I2C0CONCLR = I2C0CONCLR_SIC;
      }
      else
      {
         //Wyslij ponowny bit startu
         I2C0CONSET = I2C0CONSET_STA;
         I2C0CONCLR = I2C0CONCLR_SIC;
      }
      cnt++;
      break;
    //Kolejny bit startu zostal nadany
    case 0x10:
      //Nadaj adres I2C w trybie read
      I2C0DAT = I2C_MEMADDR|1;
      I2C0CONSET = I2C0CONSET_AA;
      I2C0CONCLR = I2C0CONCLR_SIC|I2C0CONCLR_STAC;
      break;
    //Wyslano adres pomyslnie
    case 0x40:
      //Rozpocznik odbieranie danej
      I2C0CONCLR = I2C0CONCLR_SIC|I2C0CONCLR_AAC;
      break;
    //Odebrano dana - pierwsza i ostatnia
    case 0x58:
      //Wyslij stop
      tmp = I2C0DAT;
      I2C0CONSET = I2C0CONSET_AA|I2C0CONSET_STO;
      I2C0CONCLR = I2C0CONCLR_SIC;
      return tmp;
    //Jezeli blad
    default:
      //Zatrzymaj i wyjdz
      tmp = I2C0STAT;
      I2C0CONSET = I2C0CONSET_AA|I2C0CONSET_STO;
      I2C0CONCLR = I2C0CONCLR_SIC;
      return -tmp;
    }
  }
}
```