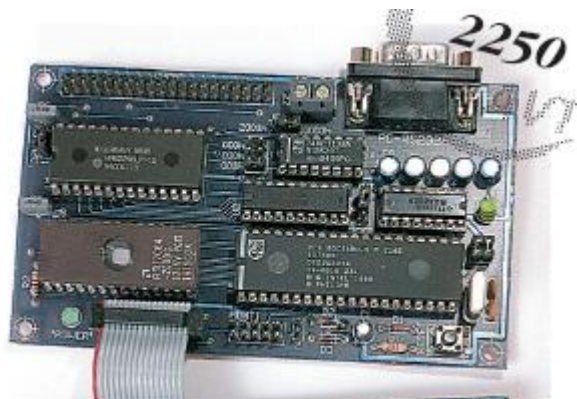


Sprawozdanie z praktyk:

„Mikrokontrolery '51”



Konrad Dobrzyniecki
Rok II, Elektronika i Telekomunikacja
Nr albumu: 116553

Data praktyk: 1-30 września 2004r.

Miejsce: Akademia Górniczo-Hutnicza w Krakowie,
Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki
Katedra Elektroniki

Prowadzący: mgr Mirosław Żołądź, katedra Elektroniki AGH

Zestawienie zagadnień:

- zaprojektowanie oraz fizyczne wykonanie układu na mikrokontrolerze Atmel z rodziny 8051 (wykonanie układu stabilizatora, programatora, dołączenie do wyjść układu diod, wyświetlaczy i połączenie układu do urządzenia zewnętrznego za pomocą portu szeregowego);

- pisanie programów sterujących pracą układu, w kompilatorze Keil opartego na składni języka C++;

- połączenie i zaprogramowanie układu do współpracy z urządzeniami zewnętrznymi (np. komunikacja z HyperTerminal);

1. Opis przebiegu zajęć w trakcie odbytych praktyk

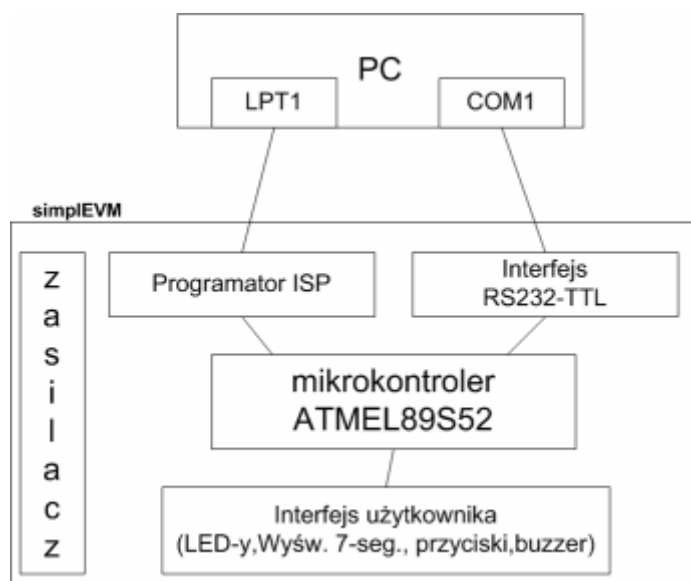
Pierwszy tydzień praktyki obejmował złożenie zestawu uruchomieniowego opartego o układ scalony firmy ATMEL 89S52. Każdy z uczestników otrzymał własną płytkę drukowaną i w trakcie kolejnych dni montował części układu: zasilacz, programator, układ mikrokontrolera, wyświetlacz siedmiosegmentowy, oraz dwa wejścia/wyjścia służące do komunikacji z komputerem: za pomocą portu LPT, oraz do transmisji szeregowej przez port COM. Środowiskiem służącym do komunikacji mikrokontrolera z komputerem był dla nas program dostarczony przez firmę ATMEL: Atmel Microcontroller ISP Software, natomiast programy tworzyliśmy przy pomocy pakietu KEIL μ Vision 2.

W drugim tygodniu praktyk zaczęliśmy prace nad częścią programową układu. Były to programy polegające głównie nad obsługą wyświetlaczy. Poznaliśmy zasadę obsługi przez mikrokontroler timer'a i przerwań. Po wmontowaniu również układu MAX232, przez który 51 łączył się z portem COM zajęliśmy się pisaniem programów obsługujących Hyper Terminal (wysyłanie do terminala , pobieranie danych z klawiatury).

W ostatnim tygodniu pracowaliśmy nad projektem obsługi zegara. Mikrokontroler miał równocześnie podawać sygnał na wyświetlacz, Hyper Terminal, czyli konieczna była obsługa dwóch rodzajów timera (jednego sterującego wyświetlaczami, oraz drugiego służącego do odbioru oraz transmisji danych). Efektem końcowym był program odmierzający czas i wysyłający go na wyświetlacz i terminal, z równoczesną możliwością jego ustawiania przez użytkownika.

2. Możliwości aplikacyjne oraz zastosowanie mikrokontrolerów '51

Każdy z mikrokontrolerów jest samodzielną jednostką, która umożliwia wykonywanie operacji arytmetycznych i sterowanie elementami zewnętrznymi. Jego budowę można w skrócie określić jako składającą się z jednostki procesora CPU, pamięci, oraz układu wejścia/wyjścia. Jego pamięć jest typu FLASH, czyli można go programować wielokrotnie, a każdy proces kasowania danych trwa ok 10ms. Dzięki zastosowaniu systemu programowania mikrokontrolera w układzie (ISP- In System Programming) nie ma potrzeby przekładania kości do programatora i spowrotem. Umożliwia to między innymi transmisja szeregową z komputerem. Prosty schemat blokowy jest przedstawiony na rysunku poniżej:



Rys.1 Schemat blokowy układu mikrokontrolera firmy ATMEL

Dużym ułatwieniem jest możliwość programowania mikrokontrolerów '51 w języku C. Jest on łatwiejszy w analizie i bardziej przejrzysty, niestety tracimy tutaj na szybkości w stosunku do asemblera, oraz kod jest nieco dłuższy. Przykładowy kod programu służącego do obsługi zegara z możliwością wprowadzania zmian czasu przez użytkownika przedstawiamy w załączniku.

Mikrokontrolery '51 mogą obsługiwać różnego rodzaju procedury począwszy od arytmetycznych, takich jak dzielenie, mnożenie, dodawanie i odejmowanie, różnego rodzaju konwersje np. z jednego rodzaju kodu na drugi, a także wiele innych – między innymi obsługa zegara, portu szeregowego, różnego rodzaju liczniki, ograniczniki jakiś wartości np. w systemach sterowania.

Budowa zdecydowanej większości mikrokontrolerów '51 i ich lista rozkazów pozwalają na korzystanie z:

- pamięci programu o pojemności do 64 kB,
- zewnętrznej pamięci danych o pojemności do 64 kB,
- wewnętrznej pamięci danych o pojemności do 256 bajtów,
- 128-bajtowego obszaru rejestrów specjalnego przeznaczenia (SFR - Special Function Registers).

Poniżej przedstawiamy przygotowany przez nas kod programu polegającego na komunikacji pomiędzy mikrokontrolerem a otoczeniem – zegarek oparty na przerwaniach

```

#include <reg51.h>

#define BAUDRATE 9600 // 9600 bps szybkość transmisji
#define LICZWYSW 4
#define CZESTWYSW 50
#define CZESTKWARCU 11059200
#define CZESTTIMERA (CZESTWYSW*LICZWYSW)
#define KOD_TH0 ((2^16-(CZESTKWARCU/12/CZESTTIMERA))/256)
#define KOD_TL0 ((2^16-(CZESTKWARCU/12/CZESTTIMERA))%256)
#define TXBUFFERLENGTH 8
#define TOKENINDEXMAX 4

/* tablica kodu siedmiosegmentowego */
unsigned char Kod7seg[] = {0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90,0x88,0x83,0xC6,0xA1,0x86,0x8E};
/* tablica do sterowania wyświetlaczami */
unsigned char KodP0[LICZWYSW]={0x07,0x0B,0x0D,0x0E};
/* tablica wyświetlanych wartości na wyświetlaczu */
unsigned char Wyświetl[LICZWYSW]={0,0,0,0};
/* zmienne pomocnicze */
int BiezWysw=0,StanWysw=0,Cykle=0,TxBufCounter=0,RxBufCounter=0,RxBufEnd,
    TokenIndexCounter=0,TokenValueCounter=0;
int TokenValue[TOKENINDEXMAX];
bit rqSendSek,rqSendMin,RxBufReady;
unsigned char TxBuffer[TXBUFFERLENGTH]='S','e','k',' ','0','0','0','0';
    RxBuf[TXBUFFERLENGTH],TokenIndex[TOKENINDEXMAX];

unsigned char LiczSekundy=0,LiczMinuty=0;
char Data;

/***** wysyłanie czasu na hyper terminal *****/
void SendSek(void)
{
    TxBuffer[0]='S';
    TxBuffer[1]='e';
    TxBuffer[2]='k';
    TxBuffer[3]=' ';
    TxBuffer[4]=LiczSekundy/10+48;
    TxBuffer[5]=LiczSekundy%10+48;
    SBUF=TxBuffer[6];
    SBUF=TxBuffer[7];
    TxBufferCounter=0;
}

void SendMin(void)
{
    TxBuffer[0]='M';
    TxBuffer[1]='i';
    TxBuffer[2]='n';
    TxBuffer[3]=' ';
    TxBuffer[4]=LiczMinuty/10+48;
    TxBuffer[5]=LiczMinuty%10+48;
    SBUF=TxBuffer[6];
    SBUF=TxBuffer[7];
    TxBufferCounter=0;
}

/***** analiza danych pobieranych z klawiatury *****/
void AnalizeRxBuffer (void)
{
    for(RxBufCounter=0;RxBufCounter<RxBufEnd;RxBufCounter++)
    {
        if((RxBuf[RxBufCounter]>=97 && RxBuf[RxBufCounter]<=122) ||
            (RxBuf[RxBufCounter]>=65 && RxBuf[RxBufCounter]<=90) ||
            (RxBuf[RxBufCounter]>=48 && RxBuf[RxBufCounter]<=57))
        {
            TokenIndex[TokenIndexCounter]=RxBufCounter;
            RxBufCounter++;
            while((RxBuf[RxBufCounter]>=97 && RxBuf[RxBufCounter]<=122) ||
                (RxBuf[RxBufCounter]>=65 && RxBuf[RxBufCounter]<=90) ||
                (RxBuf[RxBufCounter]>=48 && RxBuf[RxBufCounter]<=57))
            {
                RxBufCounter++;
            }
            TokenIndexCounter++;
        }
        else
        {
            RxBuf[RxBufCounter]=0x00;
        }
    }
}

void GiveValue (void)
{
    for(TokenIndexCounter=0;TokenIndexCounter<TOKENINDEXMAX;TokenIndexCounter++)
    {
        RxBufCounter=TokenIndex[TokenIndexCounter];
        if(RxBuf[RxBufCounter]>=48 && RxBuf[RxBufCounter]<=57)
        {
            while(RxBuf[RxBufCounter]>=48 && RxBuf[RxBufCounter]<=57)
            {
                TokenValue[TokenValueCounter]=(TokenValue[TokenValueCounter])*10+
                    ((int)RxBuf[RxBufCounter]-48);
                RxBufCounter++;
            }
            TokenValueCounter++;
            TokenIndexCounter++;
        }
        else
        {
            TokenValue[TokenValueCounter]=-1;
            TokenValueCounter++;
        }
    }
}

```

```

    }
}

/***** funkcja porownujaca dwa stringi *****/
bit CmpStr(char *String1,char *String2)
{
    int Index=0;
    while((String1[Index])!=0x00 && (String2[Index])!=0x00 && (String1[Index])!=0x0D && (String2[Index])!=0x0D)
    {
        if((String1[Index])==(String2[Index]))
        {
            Index++;
        }
        else
        {
            return 0;
        }
    }
    return 1;
}

/***** czyszczenie bufora *****/
void EraseBuffer (void)
{
    for(RxBufferCounter=0;RxBufferCounter<TXBUFFERLENGTH;RxBufferCounter++)
    {
        RxBuffer[RxBufferCounter]=0x00;
    }
    for(TokenIndexCounter=0;TokenIndexCounter<TOKENINDEXMAX;TokenIndexCounter++)
    {
        TokenIndex[TokenIndexCounter]=0x00;
        TokenValue[TokenIndexCounter]=0x00;
    }
    RxBufferCounter=0;
    TokenIndexCounter=0;
    TokenValueCounter=0;
}

/***** przepisanie danych z klawiatury do tablicy *****/
void GetData (void)
{
    if(Data!=0x0D)
    {
        RxBuffer[RxBufferCounter]=Data;
        RxBufferCounter++;
    }
    else
    {
        RxBufferEnd=RxBufferCounter;
        AnalizeRxBuffer();
        GiveValue();
        if(TokenValue[0]==(-1) && TokenValue[1]!=(-1))
        {
            if(CmpStr(RxBuffer+TokenIndex[0],"sek")==1)
            {
                LiczSekundy=TokenValue[1];
                Cykle=0;
                rqSendSek=1;
            }
            if(CmpStr(RxBuffer+TokenIndex[0],"min")==1)
            {
                LiczMinuty=TokenValue[1];
                Cykle=0;
                rqSendMin=1;
            }
        }
        EraseBuffer();
    }
}

/***** obsluga timera0 do odliczania czasu *****/
void timer0 (void) interrupt 1
{
    TL0=KOD_TL0;
    TH0=KOD_TH0;

/***** funkcja odswiezajaca wyswietlacze *****/
    P0=0xFF;
    StanWysw=Wyswietl[BiezWysw];
    P2=Kod7seg[StanWysw];
    P0=KodP0[BiezWysw];
    BiezWysw++;
    if (BiezWysw==LICZWYSW)
    {
        BiezWysw=0;
    }

/***** funkcja odswiezajaca zegar *****/
    Cykle++;
    if (Cykle==CZESTIMERA)
    {
        Cykle=0;
        LiczSekundy++;
        rqSendSek=1;
        if (LiczSekundy>5)
        {
            LiczSekundy=0;
            LiczMinuty++;
            rqSendMin=1;
            if (LiczMinuty>59)

```

```

        {
            LiczMinuty=0;
        }
    }
    Wswietl[0]=LiczMinuty/10;
    Wswietl[1]=LiczMinuty%10;
    Wswietl[2]=LiczSekundy/10;
    Wswietl[3]=LiczSekundy%10;
}

/***** obsluga pobierania danych i transmisji *****/
void com_isr (void) interrupt 4 using 1
{
    /***** obsluga otrzymanych danych *****/
    if (RI)
    {
        Data = SBUF;
        RI=0;
        RxBufferReady=1;
    }
    /***** obsluga transmisji danych *****/
    if (TI)
    {
        TI=0;
        if(TxBufferCounter<TXBUFFERLENGTH)
        {
            SBUF=TxBuffer[TxBufferCounter];
            TxBufferCounter++;
        }
    }
}

/***** funkcja main *****/
void main (void)
{
    EA = 0;          // globalne blokowanie przerwan

    /*****ustawienia dla timera 0*****/
    TMOD &=0xF0;    // zostawia pierwsze cztery bity i zeruje cztery ostatnie
    TMOD |=0x01;    //dodaje do ostatnich bitow 1 na koncu (wlaczenie M0)
    TR0 = 1;
    ET0 = 1;        //wlaczenie obslugi przerwan

    /*****ustawienia dla timera 1*****/
    TMOD |= 0x20;   // Timer T1 w trybie 2 (GATE=0, C/T=0, M1=1, M2=0)
    TH1 = (unsigned char) (256 - ( CZESTKWARCU / (BAUDRATE*16L*12L) )); // ustawienie starszego bajtu timera
    TR1 = 1;        // start Timera T1

    /***** ustawienia portu szeregowego *****/
    PCON |= 0x80; // 0x80=SMOD:
    SCON = 0x50; // serial port MODE 1, umozliwienie szeregowego odbierania danych

    ES = 1; // zezwolenie na przerwania szeregowo
    EA=1; //globalne zezwolenie na przerwania
    TI=1;
    while (1)
    {
        if(rqSendSek==1)
        {
            SendSek();
            rqSendSek=0;
        }
        else if(rqSendMin==1)
        {
            SendMin();
            rqSendMin=0;
        }
        else if(RxBufferReady==1)
        {
            GetData();
            RxBufferReady=0;
        }
    }
}

```