Wydział Fizyki, Matematyki i Informatyki
Politechnika Krakowska

Instytut Teleinformatyki

Systemy Wbudowane

Sprawozdanie z Laboratorium 2

Prowadzący:
dr hab. Zbisław Tabor
prof. dr hab. inż. Piotr Malecki

Autor:
Kamil Wójcik

1 Wprowadzenie

Celem laboratorium było zapoznanie się z systemem przerwań w układach mikrokontrolerów, na przykładzie zestawu ZL15AVR wyposażonego w AVR ATmega32.

2 Zadanie

Celem wybranego zadania(2.2) było napisanie programu odliczającego sekundy na wyświetlaczu 7-segmentowym wykorzystując przerwania od Timera0.

2.1 Obsługa przerwań

Zgodnie z wymaganiem do odliczania czasu użyjemy przerwań od Timera0. Zastosowany zestaw uruchomieniowy posiada wbudowany oscylator o $f = 1MHz$. Zakładając najwyższą wartość preskalera $T = 1024$ oraz fakt że użyjemy przerwań od przepełnienia rejestru możemy określić że potrzebna liczna przerwań do odliczenia $1s$ będzie wynosić ≈ 3.81. Aby rozwiązać ten problem wprowadzimy dodatkową zmienną która będzie określała liczbę przerwań.

$$C = \frac{\frac{f_{clk}}{f_{czas}}}{RT}$$

gdzie:
C - licznik przerwań
f_{clk} - częstotliwość pracy oscylatora (1MHz)
f_{czas} - częstotliwość pożądana (1Hz)
R - ilość cykli timera do przerwania (255). W naszym przypadku przerwania ustawione są od przepełnienia a Timer0 jest timerem 8-bitowym
T - wartość preskalera(64)

Podstawiając do wzoru otrzymujemy: $C = 61,2745$. Łatwo można obliczyć że czas odliczany tym sposobem będzie wynosił $\approx 0.9955s$. Na takiej precyzji poprzestaniemy. Wymienię tylko sposoby jakimi można osiągnąć większą precyzję.

- Wprowadzenie kolejnej zmiennej pozwalającej dokładniej zliczać przerwania przy mniejszej wartości preskalera
- Zastosowanie zegara czasu rzeczywistego

W bloku obsługi przerwania pomniejszamy licznik przerwań. W przypadku gdy limit został osiągnięty wyświetlana cyfra zostaje zwiększona a w razie przepełnienia wyzerowana. Po zwiększeniu cyfry licznik przerwań jest resetowany.

2.2 Obsługa wyświetlacza

Do sterowania wyświetlaczem posłużymy się portem A. W tym celu ustawiamy wartość rejestru DDRA w stan wysoki(0xFF). W celu konwersji wartości cyfr do wartości sterującej zapalonymi segmentami wyświetlacza definiujemy tablice *digitValues* jej wartości ustawiamy w funkcji *init*. Przy takim założeniu piny segmentów powinny być podłączone w kolejności *gfedcba* począwszy od pierwszego pinu Portu A. Do ostatniego pinu portu A podłączamy pin wyboru cyfry wyświetlacza oraz ustawiamy wyjście w stan wysoki (0x80).

```
#include <avr/io.h>
#include <util/delay.h>
#define BAUD(x) (((F_CPU/16)/x))
void USART_Init(int baud);
void USART_PutChar(unsigned char data);
unsigned char USART_GetChar(void);
void USART_GetString(char *s);
void USART_PutString(char *s);
int main(void)
{
    char Message[18];
    DDRD = 0xFF;
    USART_Init(BAUD(1200));
    do
    {
        USART_GetString(Message);
        USART_PutString(Message);
    }
    while (1);
    return 0;
}
void USART_Init(int baud)
```

```
{
    UBRRH = (unsigned char)(baud >> 8);
    UBRRL = (unsigned char)baud;
    UCSRB = (1 << RXEN) | (1 << TXEN);
    UCSRC = (1 << URSEL) | (3 << UCSZ0);
}
void USART_PutChar(unsigned char data)
{
    while (!(UCSRA & (1 << UDRE)));
    UDR = data;
}
unsigned char USART_GetChar(void)
{
    while (!(UCSRA & (1 << RXC)));
    return UDR;
}
void USART_PutString(char *s)
{
    while (*s)
        USART_PutChar(*s++);
}
void USART_GetString(char *s)
{
    char c;
    do
    {
        c = USART_GetChar();
        USART_PutChar(c);
        if (c == '\b') // if backspace
            s--;
        else
            *s++ = c;
        if (c == '\r') // if CR
            *s = 0;
    }
    while (c != '\r');
}
```

Listing 1: Program zadanie 2.2

3 Literatura

Literatura

[1] Instrukcja do ćwi czeń laboratoryjnych

[2] Nota katalogowa mikrokontrolera