Instytut Teleinformatyki

Wydział Fizyki, Matematyki i Informatyki
Politechnika Krakowska

Mikroprocesory i Mikrokontrolery

„System przerwań”

laboratorium: 11

autorzy: dr hab. Zbisław Tabor, prof. PK
mgr inż. Paweł Pławiak

Kraków, 2015

Spis treści

1. Wiadomości wstępne

Pierwsza część niniejszej instrukcji zawiera podstawowe wiadomości teoretyczne dotyczące omawianego tematu. Poznanie tych wiadomości umożliwi prawidłowe zrealizowanie praktycznej części laboratorium.

1.1. Transmisja szeregowa

Rejestrami modułu USART są UDR, UCSRA, UCSRB, UCSRC, UBRRH i UBRRL. Zgodnie z notą katalogową mikrokontrolera ATmega32, szybkość transmisji BOUD w zwykłym asynchronicznym trybie jest dana wzorem:

$$BAUD = \frac{FOSC}{(16 \cdot (UBRR + 1))}$$

gdzie UBRR jest 16-bitową liczbą, której bardziej znaczący bajt jest zapisany w rejestrze **UBRRH**, a mniej znaczący - w rejestrze **UBRRL**. Konfigurację transmisji szeregowej rozpoczyna wyznaczenie dla zadanej prędkości transmisji żądanej zawartości rejestrów UBRRH i UBRRL. Prędkość transmisji może być podwojona poprzez ustawienie bitu U2X w rejestrze UCSRA (wówczas w mianowniku powyższego wzoru należy zastąpić liczbę 16 liczbą 8).
UDR jest rejestrem, z którego pobiera się przychodzące dane i do którego wpisuje się dane przeznaczone do transmisji.
UCSRA jest rejestrem, pozwalającym kontrolować stan transmisji lub błędy transmisji: ustawiony bit **UDRE** tego rejestru oznacza gotowość modułu USART do transmisji danych, a ustawiony bit RXC oznacza bufor odbiorczy zawierający dane do odczytu.
UCSRB jest rejestrem, którego trzy najstarsze bity (RXCIE, TXCIE, UDRIE) pozwalają włączyć/ wyłączyć przerwania od transmisji szeregowej, bity RXEN i TXEN właczają moduły odbiorczy i nadawczy, a pozostałe bity - łącznie z bitami rejestru **UCSRC** konfigurują tryb transmisji (ilość bitów danych, parzystość, ilość bitów stopu).

Listing 1 Transmisja szeregowa w trybie odpytywania (F_CPU należy ustawić na wartość 1MHz, fuse bit CKSEL/SUT musi być również ustawiony na 1MHz)

```
//*****************************************************************************
//
//      Połączenia:
//      RxD (con7) - PD0 (con18)
//      TxD (con7) - PD1 (con18)
//
//*****************************************************************************

#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');
}
```

1.2. Przerwania zewnętrzne

Przerwania zewnętrzne są kontrolowane przez dwa rejestry: **MCUCR**, ustawiający zdarzenie wyzwalające przerwanie (opadające zbocze, narastające zbocze, zmiana, niski poziom) i **GICR**, którego ustawione bity odblokowują przerwania zewnętrzne. Odblokowane przerwanie INT0 będzie wyzwalane przy zmianach na wyjściu PD2 (zgodnie z notą katalogową, która określa taką alternatywną funkcję wyjścia PD2) - przerwanie można będzie zatem wyzwalać na przykład po podłączeniu klawisza do PD2. W pewnych sytuacjach wartość wpisaną do rejestru **GICR** warto powtórzyć w rejestrze **GIFR**.

1.3. Timery

Mikrokontrolery ATmega32 są wyposażone w 2 timery 8-bitowe i jeden 16-bitowy. Dokładny opis rejestrów sterujących pracą timetów można jak zwykle znaleźć w nocie katalogowej - poniżej opisane są (po części) rejestry sterujące pracą Timera0. Za konfigurację Timera0 jest odpowiedzialny rejestr **TCCR0**. Bity WGM00 i WGM01 kontrolują metodę odliczania licznika. W zwykłym trybie (oba bity wyczyszczone) licznik jest zerowany po przepełnieniu. Ustawiony bit WGM1 powoduje zerowanie licznika jeśli odliczona w liczniku wartość jest równa wartości wpisanej w rejestrze **OCR0**. Aktualna wartość licznika jest zapisana w rejestrze **TCNT0**.
Trzy najmłodsze bity rejestru TCCR0 (CS00, CS01, CS02) kontrolują szybkość odliczania timera, a jeśli są wyczyszczone - **timer jest zatrzymany i nie odlicza**. Inne ustawienia (nota katalogowa, str. 82) powodują, że timer odlicza z częstotliwością, równą częstotliwości oscylatora, podzielonej przez preskaler (liczba 1, 8, 64, 256 lub 1024, w zależności od ustawienia bitów). Przerwania od licznika odblokowuje się, ustawiając bity rejestru **TIMSK**: bit TOIE0 odblokowuje przerwanie od przepełnienia, a bit OCIE0 - od porównania z rejestrem OCR0.

1.4. Obsługa przerwań

Obsługa przerwań mikroprocesora ATmega32 jest dość prosta. Przede wszystkim należy dołączyć do projektu plik nagłówkowy interrupt.h:

```
#include <avr/interrupt.h>
```

W części kodu, inicjującej mikrokontroler należy ustawić globalne zezwolenie na przerwania, ustawiając najstarszy bit rejestru SREG lub wykonując instrukcję **sei()**. Globalną blokadę przerwań uzyskuje się czyszcząc w.w. bit lub wykonując instrukcję **cli()**.
Dla każdego aktywowanego przerwania należy zaimplementować procedurę jego obsługi. Implementacja ma postać:

```
ISR(nazwa_przerwania)
{
      //instukcje wykonywane w trybie przerwania
}
```

Nazwa przerwania to np.:
USART_RXC_vect dla przerwania RXC modułu UART,
TIMER0_COMP_vect dla przerwania dla trybu CTC Timera0
INT0_vect dla przerwania INT0

Nazwy przerwań, podane w pliku iom32.h w *\AVR Toolchain\avr\include\avr\ są skojarzone z nazwami źródeł przerwań, podanymi w dokumentacji mikrokontrolera (str. 44).

1.5. Zagadnienia do przygotowania

Przed przystąpieniem do realizacji laboratorium należy zapoznać się z zagadnieniami dotyczącymi:

- mikrokontrolera AVR ATmega32
- zestawu ZL15AVR
- transmisji szeregowej
- systemu przerwań

Literatura:

[1] Rafał Baranowski, *„Mikrokontrolery AVR ATmega w praktyce”*
[2] Tomasz Francuz, *„Język C dla mikrokontrolerów AVD od podstaw do zaawansowanych aplikacji”*
[3] Nota katalogowa mikrokontrolera – http://www.atmel.com/Images/doc2503.pdf
[4] Dokumentacja zestawu ewaluacyjnego ZL15AVR – http://dl.btc.pl/kamami_wa/zl15avr.pdf
[5] Instrukcja do poprzednich ćwiczeń laboratoryjnych

2. Przebieg laboratorium

Druga część instrukcji zawiera zadania do praktycznej realizacji, które demonstrują zastosowanie technik z omawianego zagadnienia.

2.1. Zadanie 1. Na ocenę 3.0 (dst)

Proszę napisać program obsługujący przerwanie INT0: w głównej pętli zapalane są po kolei diody, po wywołaniu przerwania wszystkie diody mrugają kilkukrotnie po czym program przechodzi do przerwanego odliczania.

2.2. Zadanie 2. Na ocenę 4.0 (db)

Proszę napisać program odliczający sekundy na wyświetlaczu 7-segmentowym. Program ma korzystać z przerwań od Timera0.

2.3. Zadanie 3. Na ocenę 5.0 (bdb)

Proszę napisać program, startujący i zatrzymujący odliczanie sekund na wyświetlaczu 7-segmentowym z wykorzystaniem klawiatury PC - program ma korzystać z przerwań od modułu transmisji szeregowej.

