Politechnika Opolska

LABORATORIUM

Technika Mikroprocesorowa

KIERUNEK STUDIÓW:	AiR Ns	ROK STUDIÓW:	III
SEMESTR:	VI	ROK AKADEMICKI:	2019/2020

Temat ćwiczenia:
Program wykorzystujący funkcje przerwania

Projekt wykonali:			
Nazwisko i imię:		*Nazwisko i imię:*	
1.	Marco Tiszbierek	**2.**	Klaudiusz Tacica
3.	Marek Szczekała	**4.**	

Ocena:	*Data:*	*Uwagi:*

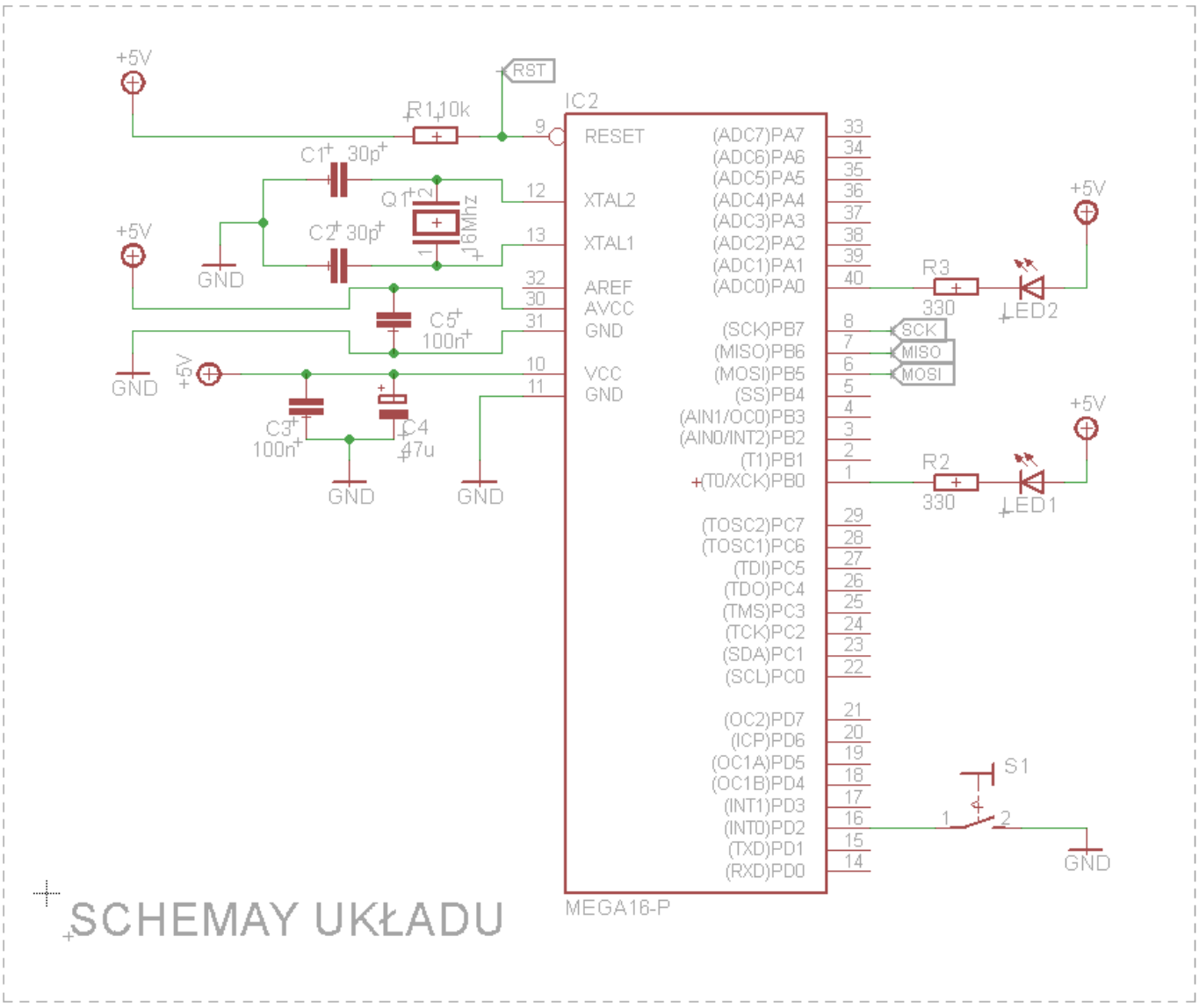
1. Wstęp:

Z powodu braku posiadania Płytki firmy Texas Instruments MSP-EXP430FR4133 lub innej z jakich korzystamy na laboratoriach, na której powinno zostać wykonane ćwiczenie. W celu wykonania zadania układ został zbudowany na płytce stykowej oraz zaprogramowany przy użyciu programatora ASP-USBASP.

2. Cel ćwiczenia:

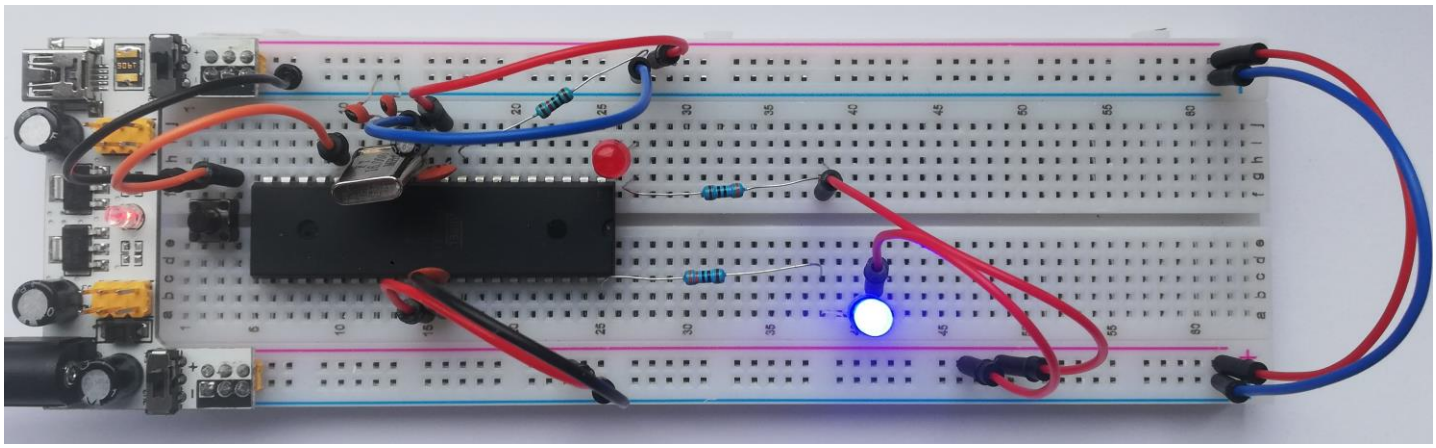
Głównym celem ćwiczenia było zaznajomienie się z funkcją przerwania. Do tego celu należało stworzyć program wykorzystujący właśnie te funkcje. Program zostało napisany w języku C w programie eclipse.

3. Schemat zbudowanego układu:

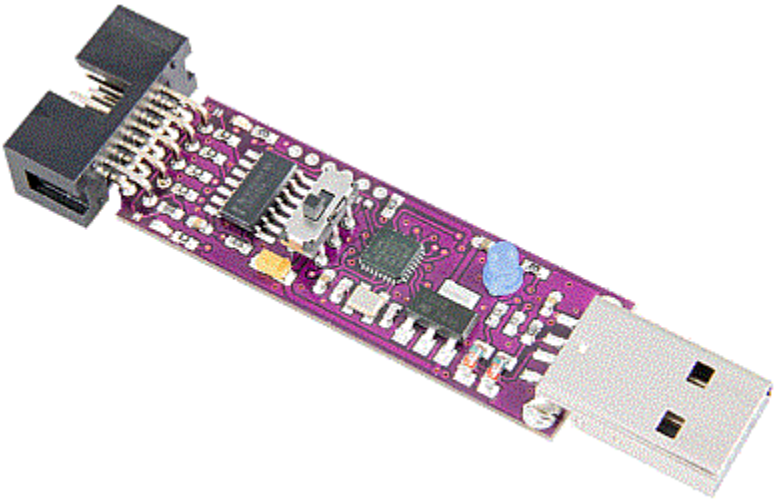


Rys.1. Schemat zbudowanego na płytce stykowej układu w celu realizacji ćwiczenia.

4. Zdjęcie zbudowanego układu oraz programatora wykorzystanych w zadaniu:



Zdj.1. Zdjęcie zbudowanego układu na mikrokontrolerze firmy Atmel model ATmega16.



Zdj.2. Zdjęcie użytego programatora ATB-USBASP.

5. Opis działania programu:

W celu wizualizacji działania zawartej w programie funkcji przerwania, program został napisany w taki sposób, aby po podłączeniu układu do zasilania oraz po rozpoczęciu działania pętli głównej programu na pinie PA0, do którego została podłączona dioda LED niebieska był zmieniany stan wyjścia na przeciwny co określony odstęp czasowy wynoszący 700ms w skutek czego dioda migała z częstotliwością 0,714 Hz. W takim stanie układ pozostawał do momentu wykrycia stanu niskiego na wejściu pina PD2(INT0) w skutek czego następowało natychmiastowe przerwanie wykonywania się pętli głównej i uruchamiana została procedura obsługi przerwań, w której została umieszczona pętla wykonująca się 12 razy zmieniając stan wyjścia pinu PB0, do którego została podłączona dioda LED czerwona, która mrugała z częstotliwością 2 Hz, w skutek czego zapalała się i gasła 6 razy po wykonaniu tej pętli znajdującej się wewnątrz obsługi przerwań program wracał do miejsca, w momencie którego zostało wywołane przerwanie kontynuując wykonywanie się pętli głównej. W momencie wciśnięcia przycisku stan na diodzie niebieskiej pozostawał taki jaki był w momencie wywołania obsługi przerwań stan ten pozostawał niezmienny przez cały czas wykonywania się obsługi przerwania do momentu powrotu programu do pętli głównej.

6. Informacje potrzebne do konfiguracji rejestrów przerwań:

W celu włącznie funkcji przerwań w pierwszej kolejności należy odpowiednio ustawić poszczególne rejestry. Wszelkie informacje można bez problemu znaleźć w notach katalogowych poszczególnych układów. W różnych układach rejestry mogą przybierać inne nazwy wiec zawsze należy posługiwać się notami katalogowymi.

Tab.1. Rejestr kontrolny MCU zawiera bity kontrolne do kontroli wyczuwania przerwań i ogólnie Funkcje MCU.

Bit	7	6	5	4	3	2	1	0	
	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Tab.2. Tabela przedstawia różne stany jakimi można wywoływać przerwania na pinie INT0 przy odpowiedniej konfiguracji poszczególnych rejestrów.

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

Tab.3. Przedstawia w jaki sposób należy ustawić odpowiednio rejestr, aby na poszczególnym pinie w naszym wypadku INT0 zezwolić na przerwania.

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

7. Kod programu:

```
// Program z funkcja obsługi przerwań
// Wykonane przez : Tacica, Tiszbierek, Szczekała AiR ns semestr VI

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

//*****************************************************************************Definicje dla procesora

#define LED_1 (1<<PB0)  // definicja pinu, do którego jest podłaczona LED(czerwona)
#define LED_1_TOG PORTB ^= LED_1  // makrodefinicja zmiany stanu diody LED1

#define LED_2 (1<<PA0)    // definicja pinu, do którego jest podłaczona led2(niebieska)
#define LED_2_TOG PORTA ^= LED_2 // makrodefinicja zmiany stanu diody LED2

#define SW (1<<PD2)    //definicja, do którego jest podłaczony przycisk SW

//*****************************************************************************Petla główna main()

int main(void)
{
    int x=0;
//***********************************************************************************Inicjalizacja
    DDRB |= LED_1;       // Rejestr kierunku PB0 - wyjscie
    DDRA |= LED_2;       // Rejestr kierunku PA0 - wyjscie
    DDRD &=~ SW;         // Rejestr kierunku PD2 - wejście

    PORTA |= LED_2;     // Wyłacznie diody LED_2 (podłaczona anoda do pinu)
    PORTB |= LED_1;     // Wyłacznie diody LED_1 (podłaczona anoda do pinu)
    PORTD |= SW ;       // Podciągniecie pinu pod VCC (wewnetrzny rezystor)

    GICR = 1<<INT0;     // Zezwolenie na przerwania INT0 na wejsciu PD2
    MCUCR &=~ 1<<ISC01 | 1<<ISC00; // INT0 wyzwalane niskim stanem na wejsciu PD2
    poczatek:;          // etykieta
    x=0 ;          // Tworzenie zmiennej x typu int wartośc początkowa 0

    sei();          // Włacznie globalnych przerwań
//*****************************************************************************Petla główna programu
    while(x<10){   // Petla While zliczająca od x do 10

        LED_2_TOG;  // Zmiana stanu diody na przeciwny
     _delay_ms(700);   // Przerwa 700 ms
     x+=1;           // Inkrementacja wartosci x
     if(x==10) goto poczatek;  // sprawdzanie wyrażenia x==10 jesli tak skok do początek
}

}
ISR(INT0_vect)  // Procedura obsługi przerwań
{
    for(int i=0 ; i<12 ; i = i++)  // petla for i=0 ; wykonana 12 razy ; i++ inkrementacja i
    {
    LED_1_TOG;      // Zmiana stanu LED_1 na przeciwny
    _delay_ms(250);     // Odczekanie 250 ms
    }


}
//*****************************************************************************Koniec main()
```

8. Wnioski:

Podsumowując wykonane ćwiczenie można stwierdzić ze funkcja ta należy do podstawowych zagadnień jakie powinien znać każdy programista, ponieważ funkcja ta jest wykorzystywana bardzo często w bardziej rozbudowanych układach, kiedy dochodzi do komunikacje z urządzeniami zewnętrznymi np. klawiaturą. Przerwania sygnału powodują zmianę przepływ sterowania, niezależnie od aktualnie wykonywanego programu i wykonanie przez procesor kodu procedury obsługi przerwania. Procedura ta wykonuje czynności związane z obsługą przerwania i na końcu wydaje instrukcję powrotu z przerwania, która powoduje powrót do programu realizowanego przed przerwaniem. Ćwiczenie okazuje się bardzo przydanego, gdyż nie tylko uczymy przy tym się poprawnego pisania kodu, ustawiania rejestrów, ale również sprawnego poruszania się po notach katalogowych bez których ustawienie poszczególnych rejestrów odpowiadających za poszczególne funkcje było by nie możliwe.