


# Politechnika Wrocławska

## Technika cyfrowa 2

### wykład 7

Katedra Metrologii Elektronicznej i Fotonicznej  
Andrzej Stępień




# Reakcja procesora

**Programowa reakcja procesora** - wykonywanie stałych pętli testujących, cyklicznych procedur testujących (**Polling**):

- znaczne obciążenie procesora, spadek efektywnego wykorzystania czasu pracy procesora
- brak natychmiastowej reakcji procesora wskutek zdeterminowanej przez program kolejności wykonywanych procedur testujących

**Sprzętowa reakcja procesora** - wykorzystanie systemu przerwań (**Interrupt**):

- brak obciążenia procesora jeśli brak żądania obsługi przerwania (Interrupt Request)
- natychmiastowa reakcja procesora po spełnieniu pewnych warunków, dodatkowych warunków; rozpoczęcie wykonywania procedury obsługi przerwania (podobieństwo do wywołania podprogramu)




# Przerwanie (interrupt)

→zawieszenie (przerwanie) wykonywanego programu, najczęściej po spełnieniu dodatkowych warunków, zapamiętanie wszystkich danych niezbędnych do kontynuowania zawieszonego programu

→wykonanie podprogramu (**ISR** - Interrupt Service Routine) obsługi zdarzenia (sygnału) żądającego (wymagającego) obsługi


← kontynuacja zawieszonego programu





# Typy przerwań


- **sprzętowe zerowanie procesora (Reset)** – rozpoczęcie wykonywania procedury bez warunków wstępnych (zawsze), **zmiana wartości rejestrów**
- **przerwanie sprzętowe niemaskowalne (NMI – Non - Maskable Interrupt)** rozpoczęcie wykonywania obsługi przerwania bez warunków wstępnych (zawsze)
- **przerwanie sprzętowe maskowalne (Maskable Interrupt)** rozpoczęcie wykonywania procedury obsługi przerwania zależnie od spełnienia określonych warunków wstępnych; programowe blokowanie / odblokowanie obsługi przerwań (Enable / Disable Interrupt)
- **przerwanie programowe (SWI – SoftWare Interrupt)** programowe wywołanie obsługi przerwania
- **operacja pułapki (Trap)** inicjowana pojawieniem się błędu, np. próba dzielenia przez zero



# Identyfikacja urządzenia

**Programowa / sprzętowa identyfikacja urządzenia / sygnału zgłaszającego przerwanie:**

- badanie obiegami realizowane programowo w procedurach testujących
- metoda łańcuchowa:
  - urządzenia dołączone do procesora w kolejności ważności (pierwszeństwa) obsługi przerwania
  - sygnał potwierdzenia przyjęcia przerwania jest bramkowany w urządzeniu zgłaszającym przerwanie, które podaje swój numer identyfikacyjny szyną danych
- metoda wektorowa wygenerowanie adresu podprogramu obsługi przerwania od urządzenia



# Adresy procedur obsługi przerwań

**Adres początkowy procedury obsługi przerwania:**

- dostarczany przez urządzenie zewnętrzne, to samo, które wywołało przerwanie; po akceptacji przerwania przesyłanie szyną danych adresu początku obsługi przerwania
- w tablicy wektorów przerwań:
  - w tablicy wyłącznie  $2 * N$  bajtów adresów przerwań (16 bitowa szyna adresowa); pozycja adresu zależna od rodzaju przerwania

stałe adresy ustalone przez producenta procesora, np. co 4 lub co 8 bajtów (3, 0Bh, 13h itd.)



Politechnika Wrocławska

## Koniec procedury obsługi przerwania

Każdy **podprogram** obsługi przerwania **musi kończyć się właściwą instrukcją**

- RETI (MCS 51)
- RETFIE (PIC17Cxx)
- RTI (68HC7xx)
- IRET (ST7)

aby **odtworzyć stan rejestrów kontrolera priorytetów przerw**

Politechnika Wrocławska

## Jednopoziomowa obsługa przerw

- blokowanie przyjęcia następnego przerwania w przypadku, gdy nie jest zakończona obsługa poprzedniego przerwania
- wykonanie jednej instrukcji z programu głównego przed rozpoczęciem obsługi zgłoszonego (czekającego w kolejce) przerwania

Politechnika Wrocławska

## Wielopoziomowa obsługa przerw

przyporządkowanie różnych poziomów priorytetów procedurom obsługi przerw, co umożliwia przerwanie bieżącej procedury przez inną procedurę o wyższym priorytecie:

- priorytety programowe są ustalane przez programistę, zwykle od 2 do 16 poziomów priorytetów
- priorytety sprzętowe są ustalane przez producenta procesora

Politechnika Wrocławska

## Czas reakcji

**Czas reakcji** (Response Time) - czas liczony od momentu przyjęcia zgłoszenia przerwania do zakończenia ostatniej wykonywanej instrukcji w przerywanym programie; czas zależny od:

- sposobu testowania kolejności zgłoszonych przerw
- liczby cykli maszynowych potrzebnych do zakończenia bieżącej, wykonywanej instrukcji
- typu wykonywanych instrukcji, np. programowania kontrolera priorytetów przerw

Politechnika Wrocławska

## Czas opóźnienia

**Czas opóźnienia** (Latency Time) - czas liczony od momentu zgłoszenia przerwania do momentu rozpoczęcia wykonywania podprogramu obsługi przerwania (ISR); czas zależny dodatkowo od:

- sposobu wywołania podprogramu obsługi przerwania, szybkości przesłania na stos licznika rozkazów PC
- liczby bajtów innych rejestrów przesyłanych na stos przed rozpoczęciem podprogramu obsługi przerwania

Politechnika Wrocławska

## Czas opóźnienia i reakcji

**PICmicro MID-RANGE MCU FAMILY. Microchip Technology, December 1997, 33023a.pdf, 8.3 Interrupt Latency (p.132):**

**Interrupt latency** is defined as the time from the **interrupt event** (the interrupt flag bit gets set) to the time that the instruction at address 0004h **starts execution** (when that interrupt is enabled).

**8-bit Microcontrollers. Databook - 1997. Temic Semiconductors, 1997**

**Response time** ... if a request is active and conditions are right for it to be acknowledged, a hardware subroutine call to the requested service routine will be the next instruction to be executed (**polling cycle + long CALL** to interrupt vector address).



Politechnika Wrocławska

## Obsługa przerwań

**Ochrona**, przechowywanie **stanu rejestrów** procesora po przyjęciu i rozpoczęciu realizacji procedury obsługi przerwania:

- **przełączanie danych** (*data swapping*)  
przechowywanie na stosie zawartości rejestrów wykorzystywanych w obsłudze przerwania
  - **sprzętowe**  
bez konieczności wprowadzania dodatkowych instrukcji na początku i końcu programu obsługi przerwania, np. MSP430
  - **programowe**  
dodatkowe instrukcje na początku (PUSH) i końcu (POP) programu obsługi przerwania, np. x86
- **przełączanie kontekstowe** (*context switching*)  
wykorzystanie dodatkowych zestawów, banków rejestrów, np. C51

Politechnika Wrocławska

## Obsługa przerwań w C51 (1/2)

- brak sprzętowego przełączania danych
- programowe przełączanie danych tylko w odniesieniu do:
  - standardowo: A, PSW i DPTR (jeśli występuje tylko jeden)
  - nietypowo: IE, IP itp.

**IrqInit:**

PUSH	ACC	; ochrona akumulatora
PUSH	DPL	; ochrona DPTR
PUSH	DPH	
..... ; zasadnicza część ISR		
POP	DPH	; odtworzenie DPTR
POP	DPL	
POP	ACC	; odtworzenie akumulatora
RETI		

Politechnika Wrocławska

## Obsługa przerwań w C51 (2/2)

- przełączanie kontekstowe, standard w obsłudze przerwania:

**IrqInit:**

PUSH	ACC	; ochrona akumulatora
PUSH	PSW	; ochrona rejestru statusowego
ANL	PSW, #1110\$0111b	; wybór banku RB1
ORL	PSW, #0000\$1000b	
..... ; zasadnicza część ISR		
POP	PSW	; odtworzenie rejestru statusowego
POP	ACC	; odtworzenie akumulatora
RETI		

- adresowanie bezpośrednie rejestrów

USING	2	; deklaracja banku RB2
PUSH	AR4	; (SP) ← (R4 <sub>RB2</sub> ) = (2*8 + 4)

Politechnika Wrocławska

## PICmicro MID-RANGE MCU FAMILY (1/3)

PICmicro MCUs can have many sources of interrupt (one interrupt source for each peripheral module):

- INT Pin Interrupt (external interrupt), Parallel Slave Port Interrupt
- Timer0 / Timer1 / Timer2 module Overflow Interrupt
- PORTB Change Interrupt (pins RB7:RB4)
- Comparator Change Interrupt
- USART Interrupts
- Receive / Transmit Interrupt
- A/D Conversion Complete Interrupt
- LCD Interrupt
- Data EEPROM Write Complete Interrupt
- Capture, Compare and PWM (CCP) Interrupt
- Synchronous Serial Port (SSP) Interrupt

Politechnika Wrocławska

## PICmicro MID-RANGE MCU FAMILY (2/3)

When an interrupt is responded to, the **Global Interrupt Enable** bit (GIE) bit is cleared to disable any further interrupt, the return address is pushed into the stack and the PC is loaded with 0004h.

**Interrupt to CPU**  
clear GIE bit  
addr = 0004h

**GIE**

Once in the interrupt service routine the source(s) of the interrupt can be determined by **polling the interrupt flag bit(s)**. Generally the interrupt flag bit(s) must be cleared in software before re-enabling the global interrupt to avoid recursive interrupts.

Politechnika Wrocławska

## PICmicro MID-RANGE MCU FAMILY (3/3)

### Context Saving During Interrupts:

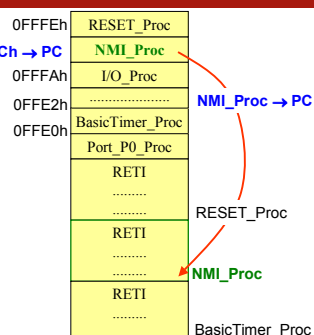
- during an interrupt, only the return PC value is saved on the stack
- typically, users may wish to save key registers during an interrupt e.g. Working register (W) and STATUS register; this has to be implemented in software.

The action of saving information is commonly referred to as "PUSHing," while the action of restoring the information before the return is commonly referred to as "POPing." These (PUSH, POP) are not instruction mnemonics, but are conceptual actions. This action can be implemented by a sequence of instructions.

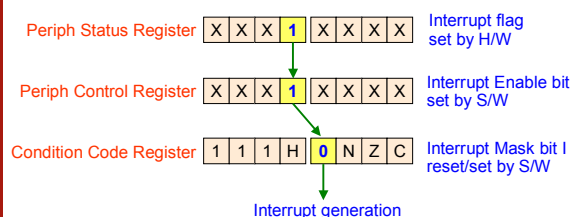


- 16-bitowa (word) struktura pamięci programu i danych
- indywidualne, pośrednie adresy wektorów przerwań
- kolejne adresy wektorów przerwań co 2

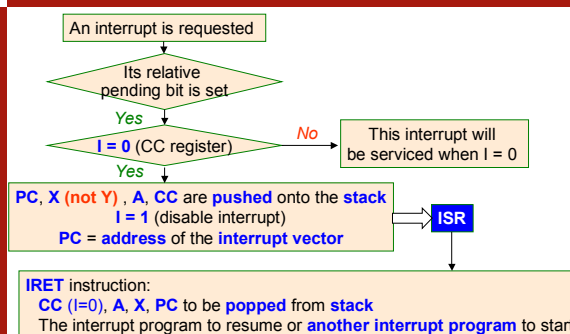
- przyjęcie przerwania kasuje znacznik **GIE** i zachowuje na stosie **Status Register (SR)**
- RETI odtwarza znacznik **GIE** oraz **SR**



N°	Source Block	Description	Register Label	Priority Order	Exit from HALT	Address Vector	
	RESET	Reset		Highest Priority	yes	FFFEh-FFFFh	
	TRAP	Software Interrupt			no	FFFFCh-FFFFDh	
0		Not used				FFFFAh-FFFFBh	
1	eI0	External Interrupt 0	N/A	Higher address = higher priority		FFFF8h-FFFF9h	
2	eI1	External Interrupt 1				FFFF6h-FFFF7h	
3	eI2	External Interrupt 2			yes	FFFF4h-FFFF5h	
4	eI3	External Interrupt 3				FFFF2h-FFFF3h	
5		Not used				FFFF0h-FFFF1h	
6		Not used				FFFECh-FFFEFh	
7	SI	A/D interrupt	SICSR			no	FFFECh-FFFEFh
8	AT TIMER	AT TIMER Output Compare Interrupt	PWM0CSR			no	FFEAh-FFEFh
9		AT TIMER Overflow Interrupt	ATCSR			yes	FFFE8h-FFFE9h
10	LITE TIMER	LITE TIMER Input Capture Interrupt	LTCSCR			no	FFFE6h-FFFE7h
11		LITE TIMER RTC Interrupt	LTCSCR		yes	FFFE4h-FFFE5h	
12	SPI	SPI Peripheral Interrupts	SPICSR	Lowest Priority	yes	FFFE2h-FFFE3h	
13		Not used					FFFE0h-FFFE1h



**Context switch takes 10 CPU clock cycles**



**Concurrent Interrupt:**

- an interrupt can not be interrupted by another one
- except by the NMI (Non Maskable Interrupt) or TLI (Top level interrupt)
- interrupt Vectors: up 16 Vectors
- S/W Priority (Nested mode only): 4 levels user configurable

**Nested Interrupt can be interrupted by:**

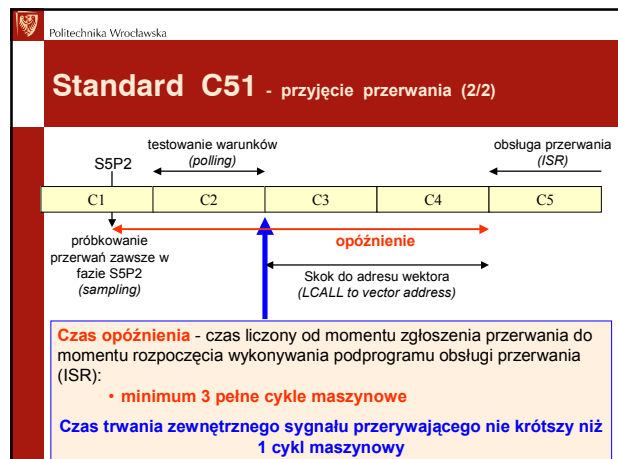
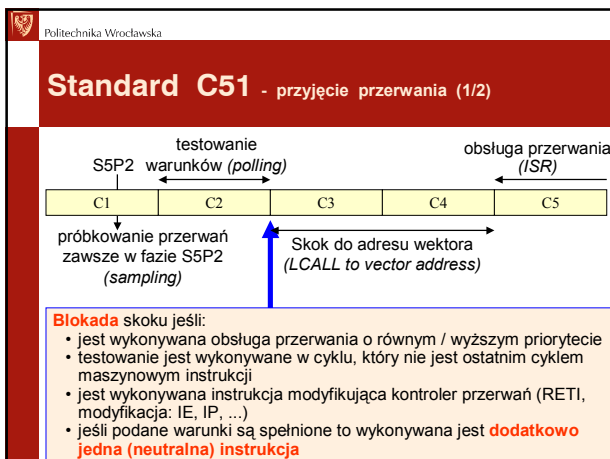
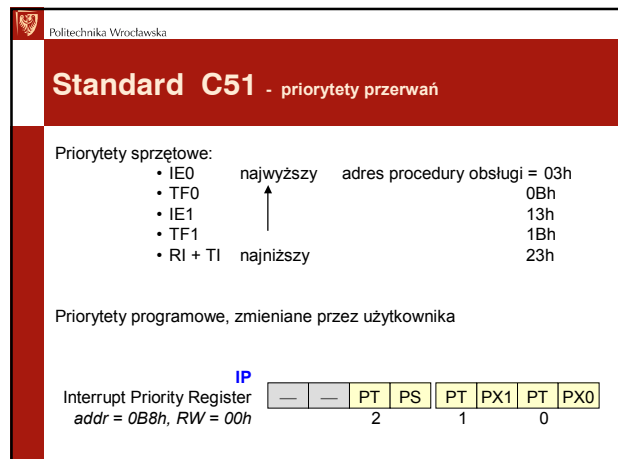
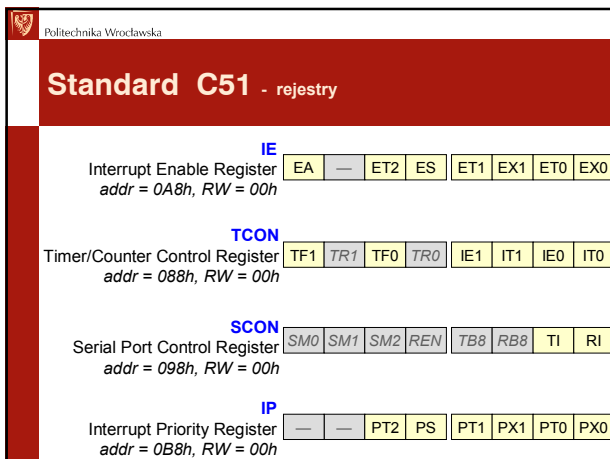
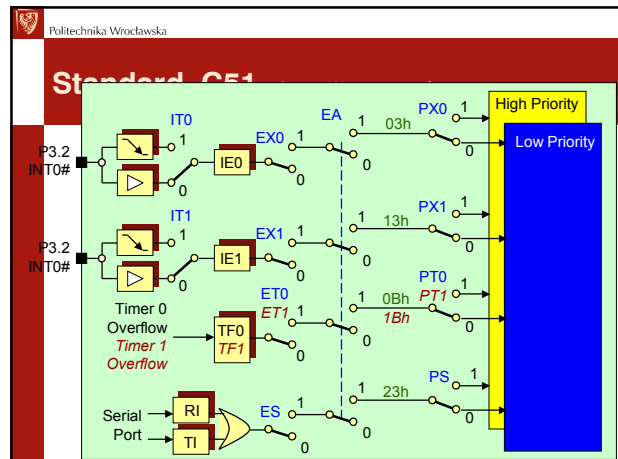
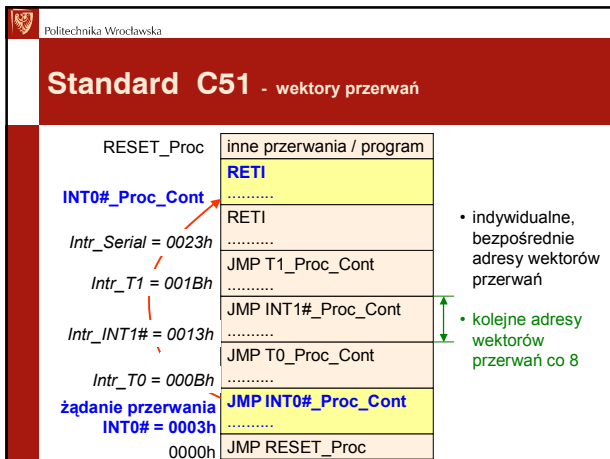
- the NMI (Non Maskable Interrupt) or TLI (Top Level Interrupt)
- an interrupt request having an higher software Priority

**RIM:** Reset Interrupt Mask (I bit =0, allows the interrupts)

**SIM:** Set interrupt Mask (I bit =1, disable the interrupts)

- a special instruction TRAP produces the same effect as an externally hardware generated interrupt request, but under program control
- the TRAP instruction uses of the interrupt mechanism within the regular execution of the main program
- the trap instruction triggers interrupt processing regardless of the state of the I bit in the Condition Code register
- an example of the use of the TRAP instruction is the real-time debugger. When the user sets a breakpoint somewhere in the program, the debugger replaces the instruction at which the execution must stop with a TRAP instruction. The interrupt thus generated is processed by displaying on the screen the state of the microcontroller at that precise time. However, other uses of this instruction may be found as well.







Politechnika Wrocławska

## Standard C51 - znaczniki przerwania

Przyjęcie przerwania, **rozpoczęcie procedury** obsługi przerwania:

- kasuje automatycznie znacznik** przerwania jeśli:
  - **IE0 / IE1** - zewnętrzne przerwania INTO# lub INT1# reagują na zbocze opadające
  - **TF0 / TF1** - przepełnienie od licznika T0 lub T1
- nie kasuje automatycznie znacznika** przerwania jeśli:
  - zewnętrzne przerwania INTO# lub INT1# reagują na poziom niski
  - wystąpiło przerwania od nadajnika TI lub odbiornika RI portu szeregowego

Politechnika Wrocławska

High ←		→ Low / High ↑
External Interrupt 0 IE0	Serial Channel 1 Interrupt RI1 + TI1	A/D Converter Interrupt IADC
Timer 0 Interrupt TF0		External Interrupt 2 IEX2
External Interrupt 1 IE1		External Interrupt 3 IEX3
Timer 1 Interrupt TF1	Compare Timer Interrupt CTF	External Interrupt 4 IEX4
Serial Channel 0 Interrupt RIO + TI0		External Interrupt 5 IEX5
Timer 2 Interrupt TF2 + EXF2		External Interrupt 6 IEX6

IP Bit	Irq Flag	
IP1.0 / IP0.0	IE0	RI1 + TI1
IP1.1 / IP0.1	TF0	IADC
IP1.2 / IP0.2	IE1	IEX2
IP1.3 / IP0.3	TF1	IEX3
IP1.4 / IP0.4	RIO + TI0	IEX4
IP1.5 / IP0.5	TF2 + EXF2	IEX5

Low ↓

Politechnika Wrocławska

## Standard C51 - sprzętowa praca krokowa

P3.2  
INT0#

```

CSEG AT 0
PoczProg:
  JMP  ProgDalej

  ORG 3
IrqINT0:
  JNB P3.2, $ ; oczekiwanie na P3.2/INT0# = High
  JB P3.2, $ ; oczekiwanie na P3.2/INT0# = Low
  RETI

ProgDalej:
  CLR IT0 ; INT0# reaguje na poziom niski
  SETB EX0 ; odblokowanie INT0#
  SETB EA ; odblokowanie wszystkich przerw
  PracaKrokowa:
    MOV P1, #0FEh ; instrukcje wykonywane krokowa
    MOV P1, #0FDh
    MOV P1, #0FBh
    ..... ; cd. programu użytkownika
          
```

• P3.2/INT0# reaguje na poziom niski, LOW

• w **trakcie** obsługi przerwania, przyjęcie **kolejnego przerwania** jest możliwe **po** wykonaniu instrukcji **RETI** i **jednej instrukcji** z programu głównego

Politechnika Wrocławska

## Standard C51 - praca krokowa z przerwaniem TF0/1

```

  ORG 0Bh ; początek procedury obsługi
  IrqT0: CLR TR0 ; przepełnienia licznika T0
           ; stop licznika T0

  IrqT0_BR: ; miejsce na pułapkę
            ; dalsza część programu
            ; analizowana krokowo

  SETB TR0 ; start licznika T0
  RETI ; koniec procedury obsługi
         ; przepełnienia licznika T0
          
```

Politechnika Wrocławska

## Standard C51 - trzeci poziom przerw (1/2)

Priorytety sprzętowe, konstrukcyjne:

- zewnętrzne przerwania INTO# - priorytet 3 (najwyższy)
- przepełnienie licznika T0 - priorytet 2 (średni)
- przepełnienie licznika T1 - priorytet 1 (najniższy)

Priorytety zmienione w programie użytkownika:

- przepełnienie licznika T1 - priorytet 3 (najwyższy); tylko w czasie obsługi przerwania od przepełnienia licznika T0
- przepełnienie licznika T1 - priorytet 2 (średni);
- zewnętrzne przerwania INTO# - priorytet 1 (najniższy)

program

INTO# ↑ TF0 ↑ TF1 ↑

Politechnika Wrocławska

```

CSEG AT 0
PoczProg:
  JMP  ProgDalej

IrqINT0:
  JMP  IrqINT0dalej ; początek obsługi INTO#

IrqT0:
  ORG 0Bh
  JMP  IrqT0dalej ; początek obsługi przepełnienia T0

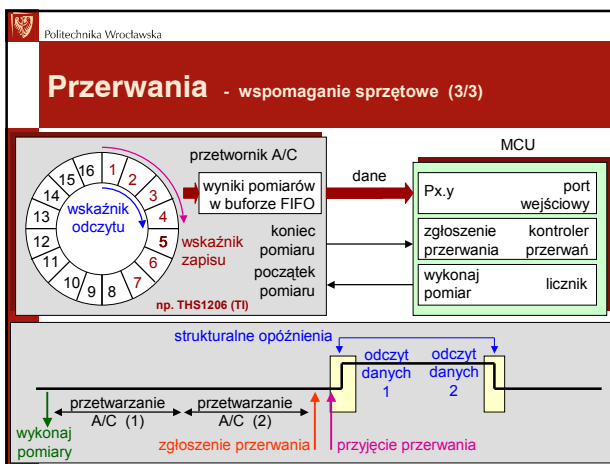
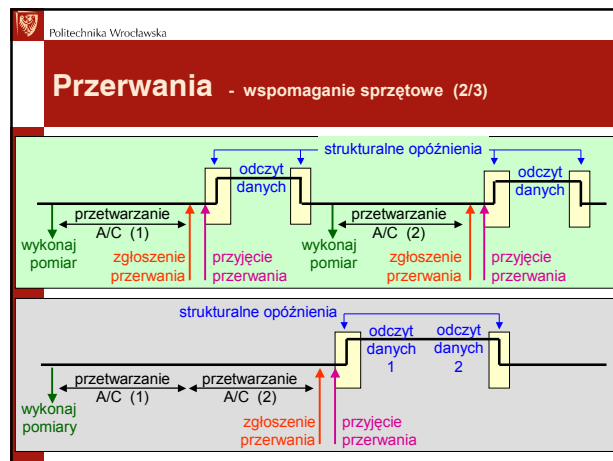
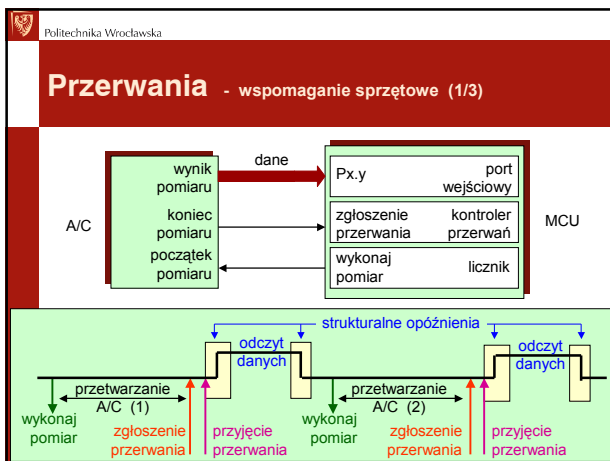
IrqT1:
  ORG 1Bh
  RETI ; początek obsługi przepełnienia licznika T1

IrqT0dalej:
  PUSH IE
  MOV IE, #1000$1000b ; ustalenie nowego priorytetu przerw
  CALL IrqT0end ; zablokowanie przerw z wyjątkiem TF1
  POP IE ; odtworzenie starego priorytetu przerw
  RET
IrqT0end:
  RETI ; redefinicja kontrolera przerw

IrqINT0dalej:
  ..... ; cd. programu obsługi INTO#
  RETI

ProgDalej:
  MOV IE, #1000$1011b ; EA = ET1 = ET0 = EX0 = 1
  MOV IP, #0000$1010b ; wyższy priorytet dla T1 i T0, niższy dla INTO#
  ..... ; program użytkownika
          
```





Politechnika Wrocławska

## Dobre rady (1/2)

- **każde przerwanie** musi być **świadomie uaktywnione**, globalnie i indywidualnie
- **żądanie obsługi przerwania nie jest realizowane natychmiast**; czas opóźnienia zależy od wykonywanych w danym momencie instrukcji
- **sygnał wywołujący przerwanie** reagujące na **poziom** musi się **zakończyć przed zakończeniem procedury obsługi przerwania** - w przeciwnym razie przerwanie jest ponownie wywoływane
- **priorytet** przyjmowanych przerwań zależy od:
  - konstruktorów mikrokontrolera (struktura ustalona w fazie produkcji)
  - programisty (struktura dostosowana do potrzeb aplikacji)
- obsługa **przerwań wielopoziomowych** jest możliwa tylko przez **programową zmianę priorytetów**

Politechnika Wrocławska

## Dobre rady (2/2)

- **każdy** program obsługi **przerwania** musi być zakończony **specjalną instrukcją**
- **kolejne żądanie** obsługi **przerwania** mogą się pojawić po zakończeniu obsługi bieżącego przerwania + czas wykonania jednej, dodatkowej instrukcji
- **wielopoziomowe, hierarchiczne struktury procedur obsługi przerw** stosują tylko **najlepsi lub szaleńcy**

Politechnika Wrocławska

## Obsługa przerwania - licznik T0 przykład (1/4)

```
#include <reg515.h>
#define Stan_T0 -250


sbit Buzzer= P1^1;

idata char wart_main;
idata char wart_intr;

void TimerT0(void) interrupt 1 {
    wart_intr--;
    if (!wart_intr) {
        wart_intr=10;
        Buzzer=~Buzzer;
    }
}

void main(void) {
    TMOD=0x02;
    TL0=Stan_T0;
    TH0=Stan_T0;
    EAL=1;
    ET0=1;
    TR0=1;
    wart_main=0;
    wart_intr=10;
    while (1) {
        wart_main++;
        if (wart_main==16) {
            wart_main=0;
        }
    }
}
```






Politechnika Wroclawska

Obługa przerwy

```

void main(void) {
    TMOD=0x02;           //      MOV    TMOD, #02h
    TL0=Stan_T0;         //      MOV    TL0, #06h
    TH0=Stan_T0;         //      MOV    TH0, #06h
    EAL=1;               //      SETB   EAL
    ET0=1;               //      SETB   ET0
    TR0=1;               //      SETB   TR0
    wart_main=0;         //      CLR    A
                        //      MOV    R0, #Wart_main(0x08)
                        //      MOV    @R0, A
    wart_intr=10;        //      INC    R0
                        //      MOV    @R0, #0Ah
    while (1) {
        wart_main++;     // C:0024 MOV    R0, #Wart_main(0x08)
                        //      INC    @R0
        if (wart_main==16) {
            wart_main=0; //      MOV    A, @R0
                        //      CJNE   A, #0x10, C:0024
                        //      CLR    A
                        //      MOV    @R0, A
                        //      SJMP   C:0024
        }
    }
}

```




Politechnika Wroclawska

Obługa przerwy

```

void TimerT0(void) interrupt 1 { // C:000Bh LJMP TimerT0
    // TimerT0: PUSH ACC (0xE0)
    //          PUSH PSW(0xD0)
    //          MOV  PSW(0xD0), #0x00
    //          PUSH 0x00
    wart_intr--;           //      MOV    R0, #wart_intr(0x09)
                        //      DEC    @R0
    if (!wart_intr) {      //      MOV    A, @R0
                        //      JNZ    C:0042
        wart_intr=10;     //      MOV    @R0, #0x0A
        Buzzer=~Buzzer;   //      CPL    Buzzer(0x90.1)
    }
                        // C:0042 POP    0x00
                        //          POP    PSW(0xD0)
                        //          POP    ACC(0xE0)
                        //          RETI
}

```



Politechnika Wroclawska

Obługa przerwy

```

void TimerT0(void) interrupt 1 using 1 { // C:000Bh LJMP TimerT0
    // TimerT0: PUSH ACC (0xE0)
    //          PUSH PSW(0xD0)
    //          MOV  PSW(0xD0), #0x08
    //          PUSH 0x00
    brak instrukcji:
    wart_intr--;           //      MOV    R0, #wart_intr(0x09)
                        //      DEC    @R0
    if (!wart_intr) {      //      MOV    A, @R0
                        //      JNZ    C:0040
        wart_intr=10;     //      MOV    @R0, #0x0A
        Buzzer=~Buzzer;   //      CPL    Buzzer(0x90.1)
    }
    brak instrukcji:      //      POP    0x00
                        // C:0040 POP    PSW(0xD0)
                        //          POP    ACC(0xE0)
                        //          RETI
}

```