Instytut Informatyki
ZMiTAC

LABORATORIUM SMIW

Ćwiczenia 7, 9

Temat: Układ przerwań.

Oleg Antemijczuk

Ćwiczenie 7. Interfejs szeregowy. Praca w pętli programowej

Cel ćwiczenia:
Nauka programowania elementów interfejsu szeregowego komputera typu IBM PC. Przygotowanie do obsługi układu przerwań.

Wymagania sprzętowe:
Dwa komputery typu IBM PC (w sieci NOVELL) połączone kablem interfejsu szeregowego i równoległego.

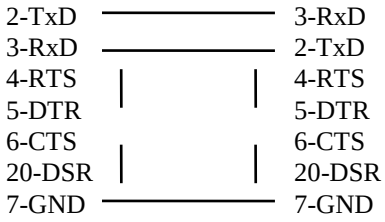
Wymagania programowe:
assembler TASM, linker TLINK oraz Borland Turbo C++ kompilator, edytor tekstu, program HELP (opis przerwań DOS).

Wprowadzenie:
Komputery typu IBM PC posiadają wbudowane interfejsy transmisji szeregowej i równoległej. W wyposażeniu standardowym zaopatrzone są w dwa łącza szeregowe COM1 i COM2 (w standardzie RS232) oraz w zależności od konfiguracji jeden bądź dwa łącza transmisji równoległej (Centronics) LPT1 i LPT2. BIOS komputerów zawiera odpowiednie procedury ich obsługi. Konstrukcja płyty głównej komputera zapewnia rozbudowę łącza szeregowego do czterech (COM1..COM4) oraz łącza równoległego (LPT1..LPT3).
Typowa karta zawiera dwa gniazda 9-pin (lub 25 pin dla starszych modeli) do łączności szeregowej, jedno gniazdo 25-pin dla łączności równoległej (Centronics printer port)

Konfigurowanie interfejsu I/O polega na odpowiednim ustaleniu adresu bazowego portu LPT, odblokowaniu/zablokowaniu sygnałów przerwań generowanych przez porty oraz odblokowaniu/zablokowaniu portów.
Obsługa interfejsu szeregowego i równoległego może być zrealizowana na dwa sposoby:

- programowe odpytywanie buforów danych transmitowanych w pętli programowej (tzw. pooling),
- poprzez wykorzystanie układu przerwań do nadawania lub odbioru danych.

Połączenie komputerów poprzez interfejs szeregowy realizowane jest przy pomocy kabla łączności szeregowej asynchronicznej 3-żyłowego. Jego konstrukcja jest następująca:
Pin:



gdzie:

TxD - Transmit Data,
RxD - Receive Data,
CTS - Clear To Send,
RTS - Request To Send,
DTR - Data Terminal Ready,
DSR - Data Set Ready,
GND - Ground,

Adresy portów szeregowych:

COM1: 3F8h..3FFh przerwanie IRQ4 obsługiwane przez wektor INT 0Ch
COM2: 2F8h..2FFh przerwanie IRQ3 obsługiwane przez wektor INT 0Bh

Obsługa portów szeregowych zaimplementowana jest w BIOSie poprzez INT 14h, możliwe jest także bezpośrednie sterowanie interfejsem szeregowym poprzez oprogramowanie odpowiednich rejestrów karty Multi I/O.

Interfejs szeregowy realizujący standard transmisji szeregowej RS232 oparty jest na elemencie Intel 8251. Element ten jest programowalny i zawiera 7 rejestrów widzialnych w przestrzeni adresowej I/O procesora płyty głównej w zależności od numeru portu szeregowego COM1.. COM4.

8251 Pinout

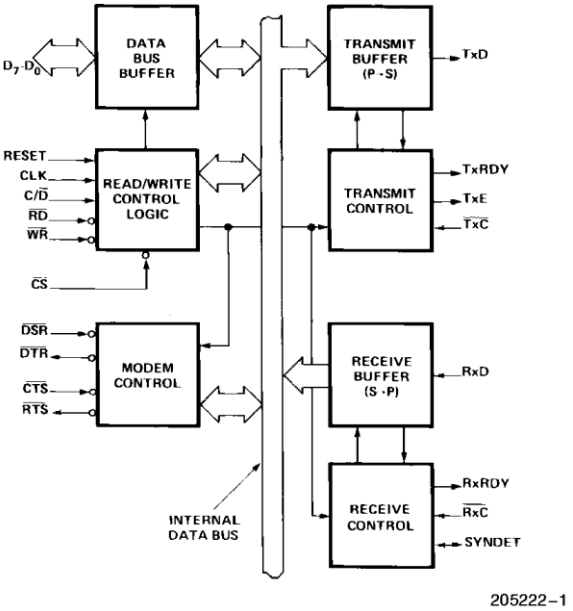


Figure 1. Block Diagram

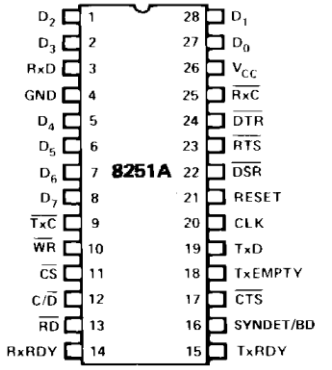


Figure 2. Pin Configuration

Funkcje rejestrów elementu 8251 COM1:

Port Znaczenie
3F8h zapis: bufor nadawczy (8 bitowy),
odczyt: bufor odbiorczy (8 bitowy),
dla DLAB=1 dzielnik częstotliwości (młodszy bajt dzielnika)

częstotliwość	dzielnik
110	1040
150	768
300	384
600	192
1200	96
2400	48
4800	24
9600	12

3F9h zapis: starszy bajt dzielnika dla DLAB=1

zapis: rejestr blokady przerwań

bit:
0: 1 - odblokuj intr. dla danych odbieranych,
1: 1 - odblokuj intr. gdy bufor nadawczy jest pusty,
2: 1 - odblokuj intr. na odbiorczej linii statusu
3: 1 - odblokuj intr. dla statusu modemu (CTS,DSR,RI,RLSD),
4..7 = 0.

3FAh odczyt: rejestr identyfikacji przerwania,
bit:
0: 1 - nie ma przerwań (np. pooling)
1..2 00 - status intr. odbiornika
przepełnienie, błąd odczytu, parzystość ,
01 - dane odebrane ważne, wyzerowanie po odczycie 3F8h
10 - bufor nadawczy pusty, zerowanie poprzez zapis do 3F8h
11 - modem status, wykrywanie stanów CTS,DSR,RI,RLSD

3FBh odczyt/zapis: rejestr kontroli linii,

bit:
0..1: długość słowa 00 - 5, 01 - 6, 10 - 7, 11 - 8,
2: bity stopu 0 - 1, 1 - 2,
3..4: parzystość: x0 - brak, 01 - odd, 11 - even,
5: nie używane przez BIOS
6: odblokowanie sterowania przerwań 1 - start wysyłania spacji,
7: DLAB 1 prędkość transmisji, 0 - normalny

3FCh zapis: rejestr sterujący modemem,

bit:
0: 1 - uaktywnienie DTR, 0 - blokada,
1: 1 - uaktywnienie RTS, 0 - blokada,
2: 1 - uaktywnienie OUT1 (wyjscie użytkownika)
3: 1 - uaktywnienie OUT2
4: 1 - uaktywnienie pętli diagnostycznej

3FDh odczyt: rejestr statusu linii,

bit:
0: 1 - dane gotowe /DR/ reset poprzez odczyt,
1: 1 - błąd /OE/ poprzedni znak zostaje utracony,
2: 1 - błąd parzystości /PE/ reset poprzez odczyt linii statusu,
3: 1 - błąd ramki /FE/ zły bit stopu,
4: 1 - wykrycie przerwy /BI/ odebranie podtrzymania,
5: 1 - bufor nadawczy pusty, OK dla następnego znaku do nadania,
6: 1 - nadajnik pusty, brak danych do wysłania,
7: 0,

3FEh odczyt: rejestr statusu modemu,

bit:
0: 1 - DCTS zmienił stan,
1: 1 - DDSR zmienił stan,
2: 1 - TERI aktywny,

3: 1 - DDCD zmienił stan,
4: 1 - CTS aktywny,
5: 1 - DSR aktywny,
6: 1 - RI aktywny,
7: 1 - DCD aktywny,

Treść ćwiczenia:

Napisać program transmisji znaków poprzez interfejs szeregowy bez wykorzystania przerwań.

Ćwiczenie 9. Układ przerwań.

Cel ćwiczenia:
Nauka programowania układu przerwań komputera typu IBM PC.

Wymagania sprzętowe:
Dwa komputery typu IBM PC (w sieci NOVELL) połączone kablem interfejsu szeregowego i równoległego.

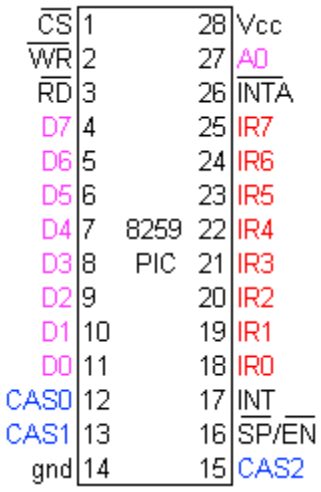
Wymagania programowe:
System operacyjny DOS, Borland C++ kompilator, edytor tekstu, program HELP (opis przerwań DOS).

Wprowadzenie:
Komputery klasy IBM PC zawierają w swojej konstrukcji układ przerwań zbudowany w oparciu o dwa elementy Intel 8259 połączone kaskadowo obsługujące do 15 urządzeń zewnętrznych. Element 8259 będący ekspanderem przerwań zawiera rejestry dostępne programowo oraz szereg modułów realizujących funkcje maskowania wejść, ustalania ich priorytetów oraz komunikacji z procesorem 80x86.

Lista przerwań obsługiwanych przez układ przerwań komputera typu IBM PC.
- Master 8259
    - IRQ0 – Intel 8253 or Intel 8254 Programmable Interval Timer, aka the system timer
    - IRQ1 – Intel 8042 keyboard controller
    - IRQ2 – not assigned in PC/XT; cascaded to slave 8259 INT line in PC/AT
    - IRQ3 – 8250 UART serial port COM2 and COM4
    - IRQ4 – 8250 UART serial port COM1 and COM3
    - IRQ5 – hard disk controller in PC/XT; Intel 8255 parallel port LPT2 in PC/AT
    - IRQ6 – Intel 82072A floppy disk controller
    - IRQ7 – Intel 8255 parallel port LPT1 / spurious interrupt
- Slave 8259 (PC/AT and later only)
    - IRQ8 – real-time clock (RTC)
    - IRQ9 – no common assignment
    - IRQ10 – no common assignment
    - IRQ11 – no common assignment
    - IRQ12 – Intel 8042 PS/2 mouse controller
    - IRQ13 – math coprocessor
    - IRQ14 – hard disk controller 1
    - IRQ15 – hard disk controller 2

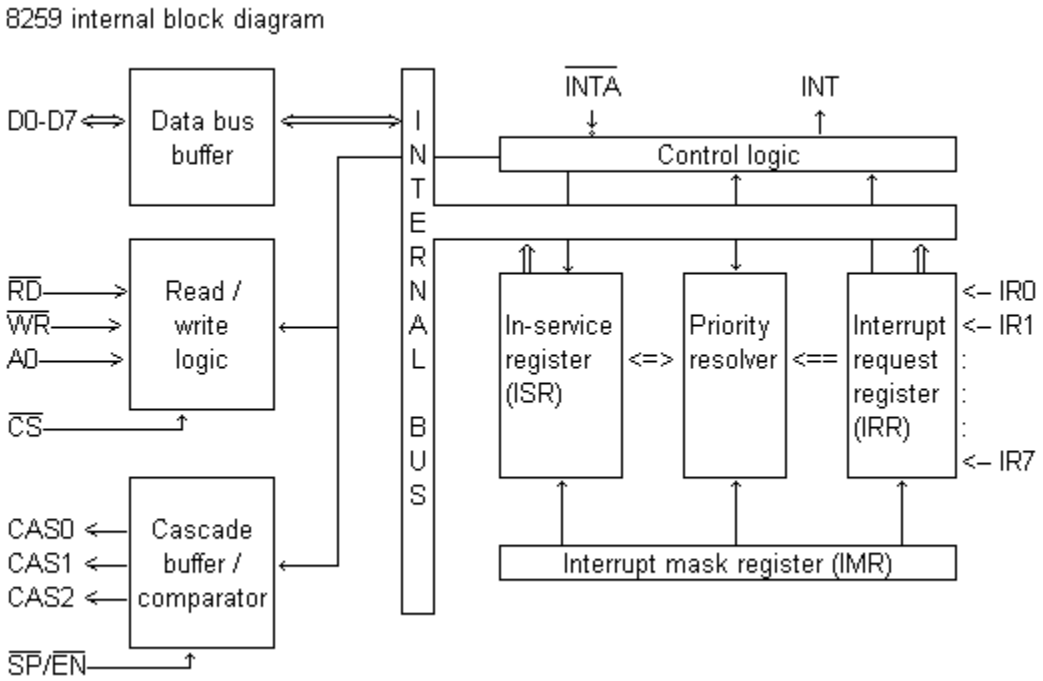
Funkcje:
- 8 levels of interrupts.
- Can be cascaded in master-slave configuration to handle 64 levels of interrupts.
- Internal priority resolver.
- Fixed priority mode and rotating priority mode.
- Individually maskable interrupts.
- Modes and masks can be changed dynamically.
- Accepts IRQ, determines priority, checks whether incoming priority > current level being serviced, issues interrupt signal.
- In 8085 mode, provides 3 byte CALL instruction. In 8086 mode, provides 8 bit vector number.
- Polled and vectored mode.
- Starting address of ISR or vector number is programmable.
- No clock required.

8259 Pinout



D0-D7	Bi-directional, tristated, buffered data lines. Connected to data bus directly or through buffers
RD-bar	Active low read control
WR-bar	Active low write control
A0	Address input line, used to select control register
CS-bar	Active low chip select
CAS0-2	Bi-directional, 3 bit cascade lines. In master mode, PIC places slave ID no. on these lines. In slave mode, the PIC reads slave ID no. from master on these lines. It may be regarded as slave-select.
SP-bar / EN-bar	Slave program / enable. In non-buffered mode, it is SP-bar input, used to distinguish master/slave PIC. In buffered mode, it is output line used to enable buffers
INT	Interrupt line, connected to INTR of microprocessor
INTA-bar	Interrupt ack, received active low from microprocessor
IR0-7	Asynchronous IRQ input lines, generated by peripherals.

8259 Block diagram



Rejestry programowe kontrolera 8259:

ICW1 (Initialization Command Word One)

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	A7	A6	A5	1	LTIM	ADI	SNGL	IC4

D0: IC4: 0=no ICW4, 1=ICW4 required
D1: SNGL: 1=Single PIC, 0=Cascaded PIC
D2: ADI: Address interval. Used only in 8085, not 8086. 1=ISR's are 4 bytes apart (0200, 0204, etc) 0=ISR's are 8 byte apart (0200, 0208, etc)
D3: LTIM: level triggered interrupt mode: 1=All IR lines level triggered. 0=edge triggered
D4-D7: A5-A7: 8085 only. ISR address lower byte segment. The lower byte is

A7	A6	A5	A4	A3	A2	A1	A0

of which A7, A6, A5 are provided by D7-D5 of ICW1 (if ADI=1), or A7, A6 are provided if ADI=0. A4-A0 (or A5-A0) are set by 8259 itself:

ADI=1 (spacing 4 bytes)

IRQ	A7	A6	A5	A4	A3	A2	A1	A0
IR0	A7	A6	A5	0	0	0	0	0
IR1	A7	A6	A5	0	0	1	0	0
IR2	A7	A6	A5	0	1	0	0	0
IR3	A7	A6	A5	0	1	1	0	0
IR4	A7	A6	A5	1	0	0	0	0
IR5	A7	A6	A5	1	0	1	0	0
IR6	A7	A6	A5	1	1	1	0	0
IR7	A7	A6	A5	1	1	1	0	0

ADI=0 (spacing 8 bytes)

IRQ	A7	A6	A5	A4	A3	A2	A1	A0
IR0	A7	A6	0	0	0	0	0	0
IR1	A7	A6	0	0	1	0	0	0
IR2	A7	A6	0	1	0	0	0	0
IR3	A7	A6	0	1	1	0	0	0
IR4	A7	A6	1	0	0	0	0	0
IR5	A7	A6	1	0	1	0	0	0
IR6	A7	A6	1	1	0	0	0	0
IR7	A7	A6	1	1	1	0	0	0

ICW2 (Initialization Command Word Two)

Higher byte of ISR address (8085), or 8 bit vector address (8086).

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	A15	A14	A13	A12	A11	A10	A9	A8

ICW3 (Initialization Command Word Three)

A0		D7	D6	D5	D4	D3	D2	D1	D0
1	Master	S7	S6	S5	S4	S3	S2	S1	S0
	Slave	0	0	0	0	0	ID3	ID2	ID1

- Master mode: 1 indicates slave is present on that interrupt, 0 indicates direct interrupt
- Slave mode: ID3-ID2-ID1 is the slave ID number. Slave 4 on IR4 has ICW3=04h (0000 0100)

ICW4 (Initialization Command Word Four)

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	SFNM	BUF	M/S	AEOI	Mode

- SFNM: 1=Special Fully Nested Mode, 0=FNM
- M/S: 1=Master, 0=Slave
- AEOI: 1=Auto End of Interrupt, 0=Normal
- Mode: 0=8085, 1=8086

OCW1 (Operational Command Word One)

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	M7	M6	M5	M4	M3	M2	M1	M0

IRn is masked by setting Mn to 1; mask cleared by setting Mn to 0 (n=0..7)

OCW2 (Operational Command Word Two)

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	R	SL	EOI	0	0	L3	L2	L1

	R	SL	EOI	Action
EOI	0	0	1	Non specific EOI (L3L2L1=000)
	0	1	1	Specific EOI command (Interrupt to clear given by L3L2L1)
Auto rotation of priorities (L3L2L1=000)	1	0	1	Rotate priorities on non-specific EOI
	1	0	0	Rotate priorities in auto EOI mode set
	0	0	0	Rotate priorities in auto EOI mode clear
Specific rotation of priorities (Lowest priority ISR=L3L2L1)	1	1	1	Rotate priority on specific EOI command (resets current ISR bit)
	1	1	0	Set priority (does not reset current ISR bit)
	0	1	0	No operation

OCW3 (Operational Command Word Three)

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	D7	ESMM	SMM	0	1	MODE	RIR	RIS

ESMM	SMM	Effect
0	X	No effect
1	0	Reset special mask
1	1	Set special mask

Interrupt sequence (single PIC)

1. One or more of the IR lines goes high.
2. Corresponding IRR bit is set.
3. 8259 evaluates the request and sends INT to CPU.
4. CPU sends INTA-bar.
5. Highest priority ISR is set. IRR is reset.
6. 8259 releases CALL instruction on data bus.
7. CALL causes CPU to initiate two more INTA-bar's.
8. 8259 releases the subroutine address, first lowbyte then highbyte.
9. ISR bit is reset depending on mode.

Przykład programu transmisji znaków z wykorzystaniem układu przerwań

```
uses Dos,Crt;

const
    COM          =  0x01;          {COM1 =  00,    COM2 =  01}
    COM_INT      =  0x0B;          {COM1 =  0x0C, COM2 =  0x0B}
    COM_BAS      = 0x200;          {COM1 = 0x300, COM2 = 0x200}
var
  IntSave : Pointer;

procedure Init_COM;
begin
    { inicjacja portu }
    ustawienie parametrów transmisji:
    prędkość, parzystość, liczba bitów danych, liczba bitów stopu,
    odblokowanie przerwań dla danych odbieranych (port COM_BAS+0xF9)
    aktywacja OUT2, DTR, RTS (port COM_BAS+0xFC),
    Odblokowanie kontrolera przerwań 8259:
    INT0..INT7 - bit 0..7 rej. maski przerwań (adres rej. maski 0x21),
end;

procedure Send_Message (Snd_Buffer : string);
begin
....    { wysłanie znaków }
end;

procedure Recv_Message;interrrupt;
begin
    { odbiór znaków do bufora odczytu }
    odblokowanie układu przerwań port 0x20,
    Port[0x20] := 20h;
    sti
    odczyt danych z bufora (port COM_BAS+0xF8)
    sprawdzenie statusu,
    ....
    aktywacja OUT2, DTR, RTS,
end;

begin
    GetVect (COM_INT,IntSave);
    SetVect (COM_INT,Addr (Recv_Message));
    Init_COM;
    .....
    pętla główna programu
    ....
    zablokowanie układu przerwań dla INT3, INT4,
    SetVect (COM_INT,IntSave));
end.
```

Treść ćwiczenia:

1. Napisać program transmisji znaków poprzez interfejs szeregowy z wykorzystania przerwań.
2. Napisać program obsługi zegara czasu rzeczywistego wykorzystując przerwanie systemowe (wektor 0x1C). W procedurze obsługi zegara odczytać czas i wyświetlać go na ekranie.

UWAGA:

Sprawozdanie z ćwiczenia powinno zawierać listingi programów z ćw. 7 oraz 9 (trzy części) oraz być wysłane po zrealizowaniu ćw. 9!