ASSEMBLER

porty

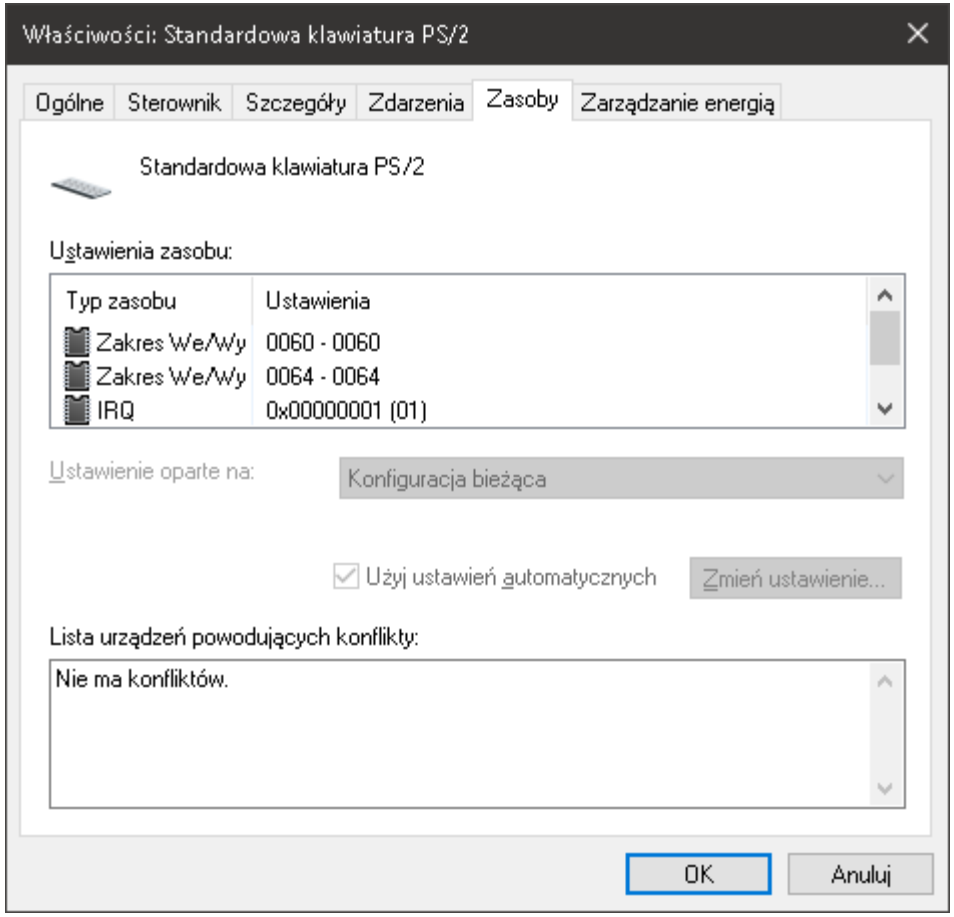
- Dla każdego urządzenia zewnętrzngo w komputerze zarezerwowano conajmniej jeden port, który musi być adresowany podczas wymiany informacji.
- Procesor steruje pracą tych urządzeń przez przesyłanie odpowiednich wartości na dany port.

PORTY - łączność między procesorem a urządzeniami zewnętrznymi

Procesor może porozumiewać z urządzeniami przez wydzielone obszary RAM-u.

Te informacje można zobaczyć w Windows we właściwościach urządzenia, na karcie Zasoby, pod hasłem Zakres pamięci.

devmgmt.msc >



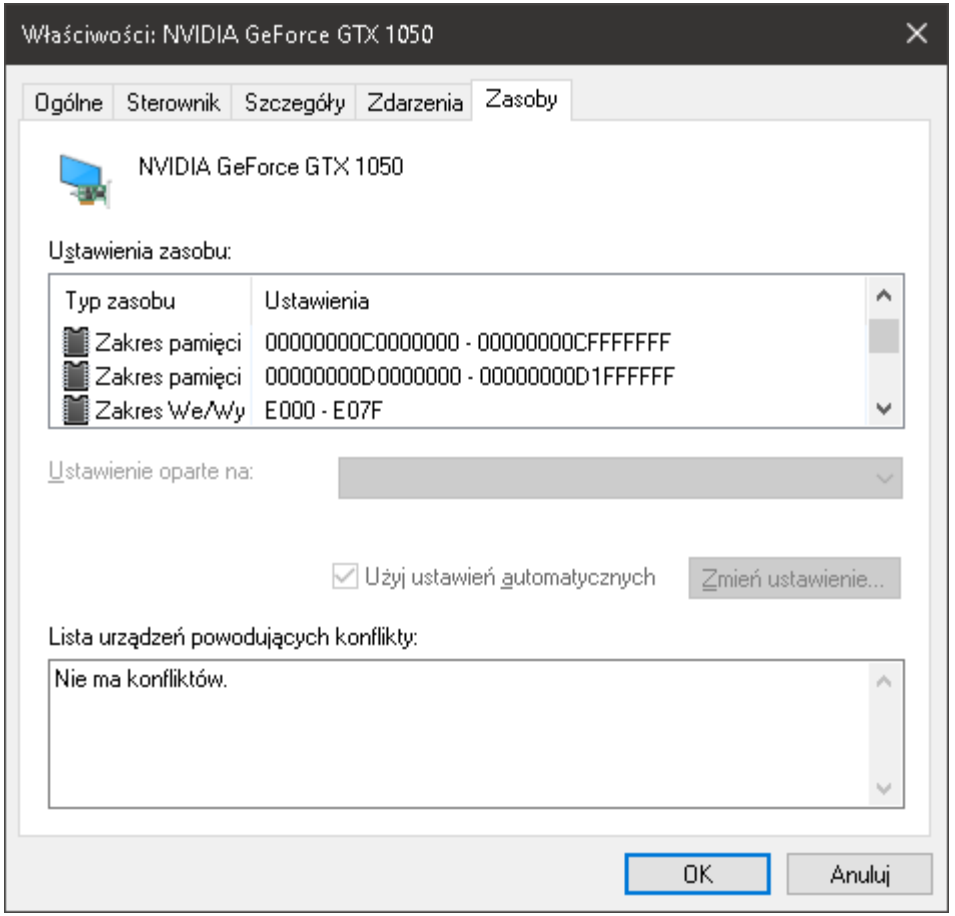
PORTY - łączność między procesorem a urządzeniami zewnętrznymi

Głównym sposobem komunikacji pozostają porty (Zasoby - Zakres we/wy)

Porty - specjalne adresy,

pod które procesor może wysyłać dane.

Stanowią oddzielną strefę adresową (16-bit, więc max teoretyczny numer portu wynosi 65535); czasami do niektórych portów można dostać się przez RAM. Są to porty mapowane do pamięci (**memory-mapped**).



Porty, czyli łączność między procesorem a urządzeniami

Procesor posiada dwie instrukcje przeznaczone specjalnie do tego celu. Są to **IN i OUT.**
Ich podstawowa składnia wygląda tak:

```
in    al/ax/eax, #_portu
out   #_portu,  al/ax/eax
```

Przykład:

```
in      al, 0          ; pobierz bajt z portu 0
out     60h, eax       ; wyślij 4 bajty na port 60h
mov     dx, 300        ; 300 > 255, więc musimy użyć DX
in      al, dx         ; wczytaj 1 bajt z portu 300
out     dx, ax         ; wyślij 2 bajty na port 300
```

Zabawa diodami na klawiaturze

0060 RW KB controller data port or **keyboard input buffer**
(ISA, EISA) should only be read from after status port bit0 = 1 should only be written to if status port bit1 = 0

Bitfields for keyboard controller read status (ISA, EISA):

Bit(s)	Description
7	parity error on transmission from keyboard
6	receive timeout
5	transmit timeout
4	keyboard interface inhibited by keyboard lock or by password server mode
3	**=1 data written to input register is command (PORT 0064h)** **=0 data written to input register is data (PORT 0060h)**
2	system flag status: 0=power up or reset 1=selftest OK
1	input buffer full (input 60/64 has data for 8042) no write access allowed until bit clears
0	output buffer full (output 60 has data for system) bit is cleared after read access

Procedura sprawdzającą zajętość portu

```
czy_mozna_pisac     proc    near
                    push eax
testuj:  in         al, 64h ; PORT KLAWIATURY
         and        al, 2   ; sprawdzamy bit nr 1
         jnz        testuj  ; jeśli #0, to sprawdzaj do skutku
         pop eax
         ret

czy_mozna_pisac
endp
```

(przeskocz port 60h)

```
0060 RW KB controller data port or keyboard input buffer (ISA, EISA)
        should only be read from after status port bit0 = 1
        should only be written to if status port bit1 = 0
```

Jak widać, trzeba też znaleźć jakiś port statusu. Jest to port 64h:

(przeskocz port 64h)

```
Bitfields for keyboard controller read status (ISA, EISA):
Bit(s)  Description     (Table P0398)
 7      parity error on transmission from keyboard
 6      receive timeout
 5      transmit timeout
 4      keyboard interface inhibited by keyboard lock
        or by password server mode
 3      =1 data written to input register is command (PORT 0064h)
        =0 data written to input register is data (PORT 0060h)
 2      system flag status: 0=power up or reset  1=selftest OK
 1      input buffer full (input 60/64 has data for 8042)
        no write access allowed until bit clears
 0      output buffer full (output 60 has data for system)
        bit is cleared after read access
```

Porty są adresowane 16-bitowo, tzn. każdy port ma swój unikalny dwubajtowy adres w systemie - np. com1 - 03F8H, lpt1 - 0378H, **klawiatura - 0060H do 0064H** itd.